Referenzhandbuch

StarBasic

Für StarOffice 4.0

Star Division GmbH

Achtung

IBM, PC-DOS, VGA, SAA, OS/2 sind eingetragene Warenzeichen der International Business Machines Corp. MS-DOS, MS-Windows, Windows-NT, Windows 95, MS-Word, Word für Windows, Visual Basic, Visual Basic for Applications, ODBC und OLE sind eingetragene Warenzeichen der Microsoft Corp. WordStar ist ein eingetragenes Warenzeichen der WordStar International Corp. dBase, Borland und Quattro sind eingetragene Warenzeichen der Borland Corp. Lotus 1-2-3 ist ein eingetragenes Warenzeichen der Lotus Development Corp. PC Paintbrush ist ein eingetragenes Warenzeichen der Z-Soft Corp. AutoCad ist ein eingetragenes Warenzeichen der Autodesk Corp. Novell ist ein eingetragenes Warenzeichen der Novell Netware Inc. WordPerfect ist ein Produkt der WordPerfect Inc. PostScript ist ein eingetragenes Warenzeichen von Adobe Systems Inc. Laserjet ist ein eingetragenes Warenzeichen der Hewlett Packard Corp. Ventura Publisher ist ein eingetragenes Warenzeichen von Ventura Software Inc. Das GIF-Format (Graphics Interchange Format) sowie der Name Compuserve sind eingetragene Warenzeichen der CompuServe Incorporated.Btrieve, Netware und Novell sind eingetragene Warenzeichen der Novell Inc. Oracle ist eingetragenes Warenzeichen der Oracle Corporation. Q+E ist eingetragenes Warenzeichen der Pioneer Software Systems Corporation. X-Printer ist eingetragenes Warenzeichen der Bristol Technology Inc. Solaris sind eingetragenes Warenzeichen der Sun Microsystems Inc. SPARC ist eingetragenes Warenzeichen der SPARC International Inc. AIX ist Warenzeichen der IBM Corp. HP-UX ist eingetragenes Warenzeichen der der Hewlett Packard Corp. Irix und SGI sind eingetragene Warenzeichen der Silicon Graphics Inc.SCO ist eingetragenes Warenzeichen der The Santa Cruz Operation Inc. UNIX ist ein eingetragenes Warenzeichen, das in den USA und anderen Staaten ausschließlich durch X/Open Company Ltd. lizensiert wird. Motiv ist eingetragenes Warenzeichen der Open Software Foundation Inc. Sinix ist eingetragenes Warenzeichen der Siemens Nixdorf Informationssysteme GmbH. X Window System ist ein Warenzeichen der X Consortium Inc. PCD ist eingetragenes Warenzeichen der Eastman Kodak Corporation. Alle in diesem Handbuch zusätzlich verwendeten Programmnamen, Schriftnamen etc. sind ebenfalls eingetragene Warenzeichen der Herstellerfirmen und dürfen nicht gewerblich oder in sonstiger Weise verwendet werden. Postscript ist eingetragenes Warenzeichen der ADOBE Systems Inc. Das Copyright der in diesem Programm verwendeten Import- und Exportfilter liegt bei der Independent JPEG-Group und Star Division. Das Copyright der mitgelieferten Cliparts liegt bei Management Graphics. Irrtümer vorbehalten.

Copyright © 1993-1997 by: STAR DIVISION Software
Entwicklung und Vertriebs GmbH
Sachsenfeld 4
20097 Hamburg

Alle Rechte vorbehalten, kein Teil des Handbuchs sowie des dazugehörigen Programms darf in irgendeiner Form (Druck, Fotokopie oder sonstige Verfahren) ohne schriftliche Genehmigung von STAR DIVISION reproduziert oder vervielfältigt werden. Wir werden jede uns bekanntwerdende nicht autorisierte Weitergabe des Programms mit allen uns zur Verfügung stehenden Mitteln verfolgen. Der Kunde erhält das Programm inklusive Handbuch und Datenträgern. Für die Fehlerlosigkeit des Programms und für Schäden, die durch die Benutzung des Programms entstehen, kann leider keine Haftung übernommen werden. Sollte ein Fehler entdeckt werden, ist STAR DIVISION bestrebt, diesen so schnell wie möglich zu korrigieren. Die Benutzung des Programms erfolgt ausschließlich aufgrund der Lizenzbestimmungen, die dem Produkt beiliegen.

Inhaltsverzeichnis

Programmieren.....1

Grundlagen.....1

Wozu benötigt man StarBasic?.....1
Voraussetzungen.....3
StarBasic ist objektorientiert.....3
StarBasic ist modular.....8
Das StarBasic-Konzept.....10
Das erste Makroprogramm.....16

"Technische Daten".....43

Variablen.....43
Verwenden von Objekten.....51
Prozeduren und Funktionen.....57
DLLs von Basic aus aufrufen.....62
Bibliotheken, Module und Dialoge.....64

Integrierte Entwicklungsumgebung (IDE).....65

Überblick IDE.....65
Anlegen und Verwalten von Bibliotheken und Modulen.....67
Der Basic-Editor.....71
Fehlersuche.....73

Weitere Möglichkeiten.....76

Zuweisung von Makros an bestimmte Programmereignisse.....76
Verknüpfung von Makros mit Symbolleisten, Tastatur oder Pulldown-Menüs 78
Import von Makros aus StarBasic 3.1.....78

Runtimefunktionen.....81

Bildschirmein-/ausgabefunktionen.....81

Bildschirmausgabefunktionen.....81
Bildschirmeingabefunktionen.....85
Farbberechnungsfunktionen.....86

Dateiein-/ausgabefunktionen.....90

Öffnen und Schließen von Dateien.....90
Ein-/Ausgabe.....94

Dateipositionierung.....99
Dateiorganisation.....103

Datums- und Zeitfunktionen.....114

Umwandeln von Datumswerten.....114
Umwandeln von Uhrzeiten.....120
Systemdatum und -uhrzeit.....124

DDE-Anweisungen und Funktionen.....126

DDEExecute-Anweisung.....126
DDEInitiate-Funktion.....128
DDEPoke-Anweisung.....129
DDERequest-Funktion.....130
DDETerminate-Anweisung.....131
DDETerminateAll-Anweisung.....131

Fehlerbehandlungsfunktionen.....132

Erl-Funktion.....132
Err-Funktion.....133
Error-Funktion.....134
Error-Anweisung.....134
On Error GoTo...Resume-Anweisung.....135

Logische Operatoren.....136

And-Operator.....136
Eqv-Operator.....137
Imp-Operator.....138
Not-Operator.....139
Or-Operator.....139
Xor-Operator.....140

Mathematische Operatoren.....141

"-" Operator.....141
"*" Operator.....141
"+" Operator.....142
"/" Operator.....143
"^" Operator.....143

Numerische Funktionen.....144

Trigonometrische Funktionen.....144
Exponential- und Logarithmusfunktion.....148
Zufallszahl-Generierung.....149

Quadratwurzelberechnung....151
Ganzzahlermittlung....151
Absolutwertermittlung....152
Vorzeichen eines Ausdrucks....153
Zahlenumwandlung....154

Steuern von Programmabläufen....155

Entscheidungen....155
Schleifen....158
Sprünge....164
Weitere Anweisungen....168

Variablen....177

CBool-Funktion....177
CDate-Funktion....178
CDbl-Funktion....179
CInt-Funktion....179
CLng-Funktion....180
Const-Anweisung....181
CreateObject-Funktion....182
CSng-Funktion....182
CStr-Funktion....183
DefBool-Anweisung....183
DefDate-Anweisung....184
DefDbl-Anweisung....185
DefInt-Anweisung....186
DefLng-Anweisung....186
DefObj-Anweisung....187
DefSng-Anweisung....188
DefStr-Anweisung....189
DefVar-Anweisung....189
Dim-Anweisung....190
ReDim-Anweisung....192
IsArray-Funktion....194
IsDate-Funktion....194
IsEmpty-Funktion....195
IsError-Funktion....195
IsNull-Funktion....196
IsNumeric-Funktion....196
IsObject-Funktion....197
LBound-Funktion....197
UBound-Funktion....198

Let-Anweisung....199
Option Base-Anweisung....199
Option Explicit-Anweisung....200
Public-Anweisung....200
Static-Anweisung....200
TypeName-Funktion; VarType-Funktion....201

Vergleichsoperatoren....202

=, <, >, <=, >=, <> - Operatoren....202

Zeichenfolgen....203

ASCII/ANSI in Zeichenfolgen....203
Inhalte vervielfachen....206
Inhalte bearbeiten....207
Zeichenfolgelängen bearbeiten....218

Sonstige Befehle....221

Beep-Anweisung....221
GetGUIType-Funktion....221
GetGUIVersion-Funktion....222
GetSystemType-Funktion....222
Shell-Funktion....223
Wait-Anweisung....224

Eigenschaften für Kontrollfelder....225

AddItem-Methode....226
Alignment....227
Auto3D....227
AutoAdjustSize....228
BackColor....229
BackStyle....230
Border....231
Cancel....232
Caption-Eigenschaft....232
Change....234
Clear-Methode....236
Cls-Methode....236
CurrentStep....237
CurrentX-Eigenschaft....238
CurrentY-Eigenschaft....238
Default....238
DefaultChecked....239

DefaultSelected....241
DefaultText....242
DrawBox-Methode....244
DrawEllipse-Methode....249
DrawLine-Methode....251
DrawLineTo-Methode....252
DrawMode-Eigenschaft....252
DrawPicture-Methode....253
DrawPoint-Methode....254
DrawPolygon-Methode....254
DrawStyle-Eigenschaft....254
DrawText-Methode....255
DrawWidth-Eigenschaft....256
Enabled....256
FileUpload....257
FillStyle-Eigenschaft....258
Font....258
FontTransparent-Eigenschaft....260
ForeColor....260
Form....262
Forms bearbeiten....262
Forms....262
Name....263
URL....263
Target....263
Encoding Type....263
Methode....263
Beenden....263
Hinzufügen....264
Ändern....264
Makros....266
GetTextHeight-Methode....266
GetTextWidth-Methode....267
Height....267
HelpContextID....268
HelpText....269
Hide-Methode....269
Image-URL....270
ItemData....270
LargeChange....272
Left....273
Link....274

List....275
ListCount-Eigenschaft....277
ListIndex-Eigenschaft....278
Load-Methode....278
Macros....279
Max....279
MaxLength....280
Min....281
MousePointer....283
Move-Methode....284
MultiLine....284
MultiSelect....285
Name-Eigenschaft....286
NewIndex-Eigenschaft....287
OutputHeight-Eigenschaft....288
OutputWidth-Eigenschaft....289
Parent-Eigenschaft....289
PassWordChar....289
Printable....290
ReadOnly....292
Refresh-Methode....292
RemoveItem-Methode....293
ScrollBars....294
Selected-Eigenschaft....294
SelLength-Eigenschaft....295
SelStart-Eigenschaft....296
SelText-Eigenschaft....297
SetFocus-Methode....298
Show-Methode....299
SmallChange....299
Sorted....301
Step....302
Style....302
TabIndex....303
TabStop....304
Tag....305
Target....306
Text-Eigenschaft....307
Top....308
TopIndex-Eigenschaft....309
UnLoad-Methode....310
URL....310

UseMnemonic....311
Value-Eigenschaft....312
Visible....313
Width....316
WordWrap-Eigenschaft....317

Klassenbaum.....320

Klasse SearchAttributes.....320

Klasse ReplaceAttributes.....320

Klasse ChartFont.....320

Klasse Title.....320

Klasse Auditing.....320

Klasse Pivot.....322

Klasse TableFont.....322

Klasse TextPage.....322

Klasse TextFont.....322

Klasse HyphenZone.....322

Klasse DropCap.....322

Klasse Paragraph.....323

Klasse Shadow.....323

Klasse PrintSettings.....323

Klasse ViewSettings.....323

Klasse Address.....323

Klasse Interior.....324

Klasse SearchSettings.....324

Klasse Border.....324

Klasse BorderLine.....324

Klasse TabStops.....325

Klasse Module.....325

Klasse StarChart.....325

Klasse StarCalc.....325

Klasse StarImpress.....325

Klasse StarImage.....326

Klasse StarMath.....326

Klasse StarWriter.....326
Klasse Application.....326
Klasse Desktop.....326
Klasse View.....327
Klasse ChartView.....327
Klasse TablePrintPreview.....327
Klasse TableEditView.....327
Klasse OutlineView.....327
Klasse SlideView.....327
Klasse DrawView.....328
Klasse ImpressEditView.....328
Klasse FormulaView.....328
Klasse WebSourceView.....328
Klasse TextPrintPreview.....328
Klasse BaseTextEditView.....328
Klasse WebEditView.....329
Klasse TextEditView.....329
Klasse MessageView.....329
Klasse FrameSetView.....329
Klasse Window.....330
Klasse IPWindow.....330
Klasse BrowseWindow.....330
Klasse FrameWindow.....331
Klasse TopWindow.....331
Klasse TaskWindow.....331
Klasse PlugInWindow.....331
Klasse Document.....332
Klasse ExplorerDocument.....332

Klasse OfficeDocument....332
Klasse ChartDocument....332
Klasse TableDocument....332
Klasse DrawDocument....333
Klasse ImpressDocument....333
Klasse ImageDocument....333
Klasse FormulaDocument....334
Klasse BaseTextDocument....334
Klasse WebDocument....334
Klasse TextDocument....335
Klasse MessageDocument....335
Klasse FrameSetDocument....335
Klasse DocumentInfo....336
Klasse SaveSettings....336
Klasse PathSettings....336
Klasse Progress....337
Klasse Selection....337
Klasse TableText....337
Klasse TableDraw....337
Klasse RangeSelection....337
Klasse TableOLEObject....338
Klasse TableSelection....338
Klasse WebListInText....338
Klasse WebListInTable....338
Klasse WebTextInTable....338
Klasse ListInText....339
Klasse ListInTable....339
Klasse TextInTable....339

Klasse BaseTextOLEObject....339
Klasse WebOLEObject....339
Klasse TextOLEObject....340
Klasse BaseTextGraphic....340
Klasse WebGraphic....340
Klasse TextGraphic....340
Klasse BaseTextTable....340
Klasse WebTable....341
Klasse TextTable....341
Klasse BaseTextFrame....341
Klasse WebFrame....341
Klasse TextFrame....341
Klasse BaseTextList....342
Klasse WebList....342
Klasse TextList....342
Klasse TextDrawText....342
Klasse BaseTextDrawControl....342
Klasse WebDrawControl....343
Klasse TextDrawControl....343
Klasse BaseTextDrawBase....343
Klasse WebDrawBase....343
Klasse TextDrawBase....343
Klasse TextDraw....344
Klasse BaseText....344
Klasse WebText....344
Klasse Text....344
Klasse BaseTextSelection....344
Klasse WebSelection....345

Klasse TextSelection.....345
Klasse Collection.....345
Klasse Data.....345
Klasse Tables.....345
Klasse GlobalContents.....345
Klasse Controls.....346
Klasse Graphics.....346
Klasse Frames.....346
Klasse TextTables.....346
Klasse Strings.....347
Klasse OLEObjects.....347
Klasse Windows.....347
Klasse Documents.....347

Ableitungsdiagramm.....349

Referenzhilfe für StarOffice.....353

Eigenschaften und Methoden mit "A".....353

AboutDialog-Methode.....353
Activate-Methode.....353
Activate-Methode.....354
ActiveDocument-Eigenschaft.....354
ActiveFrameWindow-Eigenschaft.....355
ActiveHelp-Eigenschaft.....356
ActiveModule-Eigenschaft.....356
ActiveWindow-Eigenschaft.....357
Add-Methode.....357
Add-Methode.....358
AddBookmark-Methode.....359
AddDatabaseConnection-Methode.....360
AddLibrary-Methode.....365
Address-Eigenschaft.....367
Application-Eigenschaft.....368
ArrangeHorizontal-Methode.....368
ArrangeIcons-Methode.....369

ArrangeVertical-Methode....370
AutoBackup-Eigenschaft....371
AutoBrowse-Methode....372
AutoCorrectDlg-Methode....373
AutoPilotAgenda-Methode....373
AutoPilotFax-Methode....373
AutoPilotLetter-Methode....374
AutoPilotMemo-Methode....374
AutoSave-Eigenschaft....374
AutoSaveInterval-Eigenschaft....375

Eigenschaften und Methoden mit "B"....377

BackColor-Eigenschaft....377
Backup-Eigenschaft....378
Backward-Eigenschaft....379
Basic-Eigenschaft....379
BasicIDEAppear-Methode....381
BasicStop-Methode....381
Beamer-Eigenschaft....381
BeamerTaskSensitive-Eigenschaft....382
BottomLine-Eigenschaft....382
BrowseBackward-Methode....383
BrowseForward-Methode....384
BrowseHome-Methode....384
BrowserDialog-Methode....384

Eigenschaften und Methoden mit "C"....385

CancelDownloads-Methode....385
Caption-Eigenschaft....385
Cascade-Methode....386
CaseSensitive-Eigenschaft....387
City-Eigenschaft....387
ClassName-Eigenschaft....389
Clear-Methode....389
ClearDiskCache-Methode....390
ClearHistory-Methode....390
ClearMemCache-Methode....391
Close-Methode....391
Close-Methode....392
Close-Methode....393
CloseDatabaseConnection-Methode....394
CloseQueryConnection-Methode....396

CloseTableConnection-Methode....398
Closing-Eigenschaft....400
Color-Eigenschaft....400
CommonTaskBarVisible-Eigenschaft....401
Company-Eigenschaft....402
Config-Eigenschaft....403
ConfigureDialog-Methode....404
ConfigureToolboxVisible-Eigenschaft....405
Count-Eigenschaft....405
Count-Eigenschaft....406
Count-Eigenschaft....406
Country-Eigenschaft....408
CountryId-Eigenschaft....409
CreateForm-Methode....410
CreateIndex-Methode....412
CreateLink_InSelected-Methode....413
CreateQuery-Methode....414
CreateReport-Methode....414
CreateTable-Methode....416
CreationDate-Eigenschaft....419
CreationTime-Eigenschaft....420
Creator-Eigenschaft....421
CurrentDate-Eigenschaft....422
CurrentTime-Eigenschaft....422

Eigenschaften und Methoden mit "D"....423

DataAppendRecord-Methode....423
DataAppendWholeRecord-Methode....425
Database-Eigenschaft....427
DataColumn-Methode....429
DataColumnName-Methode....430
DataColumnNumber-Methode....432
DataColumnType-Methode....434
DataDefinedDatabase-Methode....438
DataDefinedDatabaseNumber-Methode....440
DataDeleteRecord-Methode....441
DataFirstRecord-Methode....443
DataGetTableName-Methode....445
DataLastRecord-Methode....447
DataNextRecord-Methode....449
DataPhysicalRecordID-Methode....451
DataPreviousRecord-Methode....453

DataRecord-Methode....455
DataRecordID-Methode....457
DataSwitch-Methode....458
DataTableNumber-Methode....460
DefaultFilePath-Eigenschaft....462
DeleteDatabaseConnection-Methode....463
DeleteForm-Methode....465
DeleteFrame-Methode....466
DeleteIndex-Methode....468
DeleteQuery-Methode....469
DeleteReport-Methode....471
DeleteTable-Methode....472
Description-Eigenschaft....474
DesignerDialog-Eigenschaft....476
Dictionary-Eigenschaft....476
Distance-Eigenschaft....478
Document-Eigenschaft....479
DocumentInfo-Eigenschaft....479
DocumentManager-Methode....481
Documents-Eigenschaft....481

Eigenschaften und Methoden mit "E"....481

EditTime-Eigenschaft....482
EMail-Eigenschaft....483
EnterWait-Methode....484
Explorer-Eigenschaft....485
ExtendedHelp-Eigenschaft....485

Eigenschaften und Methoden mit "F"....486

Fax-Eigenschaft....486
Field1-Eigenschaft....487
Field1Title-Eigenschaft....489
Field2-Eigenschaft....490
Field2Title-Eigenschaft....491
Field3-Eigenschaft....492
Field3Title-Eigenschaft....494
Field4-Eigenschaft....495
Field4Title-Eigenschaft....496
FileDialog-Methode....498
FileDocument-Methode....499
FileName-Eigenschaft....499
FillColor-Eigenschaft....500

FirstName-Eigenschaft....501
FocusBeamer-Eigenschaft....502
FocusExplorer-Eigenschaft....503
FocusUrlBox-Methode....504
FormatColumnValue-Methode....504
FrameContent-Eigenschaft....506
FrameCount-Eigenschaft....508
FrameName-Eigenschaft....510
FramesetSource-Eigenschaft....511
FrameSpacing-Eigenschaft....512
FullName-Eigenschaft....514
FullName-Eigenschaft....515
FullScreen-Eigenschaft....515
FunctionBarVisible-Eigenschaft....516

Eigenschaften und Methoden mit "G"....516

GeneralOptions-Methode....516
GetActiveDatabaseConnection-Methode....517
GetActiveTableConnection-Methode....518
GetAdjust-Methode....520
GetDataRecordCount-Methode....522
GetDecimal-Methode....524
GetFillChar-Methode....525
GetFormNames-Methode....526
GetFrameWindow-Methode....527
GetPosition-Methode....529
GetQueryColumns-Methode....530
GetQueryNames-Methode....533
GetReportNames-Methode....535
GetSQL_Statement-Methode....536
GetTableColumns-Methode....538
Glossary-Eigenschaft....540
Graphics-Eigenschaft....541
GraphicsIndep-Eigenschaft....543
GrfFilename-Eigenschaft....544
GrfFiltername-Eigenschaft....544
GrfPosition-Eigenschaft....545

Eigenschaften und Methoden mit "H"....546

Height-Eigenschaft....546
Help-Methode....547
HelperDialog-Eigenschaft....548

HelpIndex-Methode....548
HelpOnHelp-Methode....549
HelpTip-Eigenschaft....549
HorizontalLine-Eigenschaft....549

Eigenschaften und Methoden mit "I"....550

ID-Eigenschaft....550
InSelection-Eigenschaft....552
Insert-Methode....553
InsertHyperlink-Methode....554
Interactive-Eigenschaft....555
InternetDialog-Methode....555
InternetOnline-Eigenschaft....556
InternetUpdate-Methode....556
IsFirstRecord-Methode....557
IsLastRecord-Methode....559
IsLoading-Eigenschaft....561
IsLoadingImages-Eigenschaft....563
IsPrinting-Eigenschaft....564
IsQueryNative-Methode....564
IsQueryReadOnly-Methode....567
IsTableReadOnly-Methode....569
IsTopWindow-Eigenschaft....571
IsWait-Methode....571
Item-Methode....572

Eigenschaften und Methoden mit "K"....573

KeyWords-Eigenschaft....573

Eigenschaften und Methoden mit "L"....574

LanguageId-Eigenschaft....574
LastName-Eigenschaft....575
LateInitDone-Eigenschaft....577
LeaveWait-Methode....577
Left-Eigenschaft....578
LeftLine-Eigenschaft....578
Load-Methode....579
LoadAccel-Methode....581
LoadConfiguration-Methode....581
LoadLibrary-Methode....582
LoadMenu-Methode....583
LoadStatusBar-Methode....584

LoadToolBox-Methode....584
Longer-Eigenschaft....585

Eigenschaften und Methoden mit "M"....586

MacroBarVisible-Eigenschaft....586
MacroDialog-Methode....587
MailFollowup-Methode....587
MailForward-Methode....587
MailGroupReply-Methode....588
MailMarkAsRead-Methode....588
MailNext-Methode....588
MailNextDelete-Methode....589
MailNextUnread-Methode....589
MailNotifyReset-Methode....589
MailPrev-Methode....590
MailPrevDelete-Methode....590
MailPrevUnread-Methode....590
MailReceipt-Eigenschaft....591
MailReply-Methode....591
MailSend-Methode....592
MarkMail-Methode....592
MarkThreadAsRead-Methode....592
MarkThreadAsUnread-Methode....593
Maximized-Eigenschaft....593
Minimize-Methode....594
Minimized-Eigenschaft....595
ModificationAuthor-Eigenschaft....595
ModificationDate-Eigenschaft....597
ModificationTime-Eigenschaft....598
Modified-Eigenschaft....599
Module-Eigenschaft....600
Modules-Eigenschaft....601
MultiQuickSearch-Eigenschaft....602

Eigenschaften und Methoden mit "N"....602

Name-Eigenschaft....602
Navigator-Eigenschaft....604
NewMail-Methode....604
NewWindow-Methode....605

Eigenschaften und Methoden mit "O"....606

ObjectBarVisible-Eigenschaft....606

OnActivate-Eigenschaft....606
OnClose-Eigenschaft....607
OnCreate-Eigenschaft....608
OnDeactivate-Eigenschaft....608
OnDocActivate-Eigenschaft....609
OnDocClose-Eigenschaft....610
OnDocCreate-Eigenschaft....611
OnDocDeactivate-Eigenschaft....611
OnDocOpen-Eigenschaft....612
OnDocPrepareClose-Eigenschaft....613
OnDocPrint-Eigenschaft....614
OnDocSave-Eigenschaft....615
OnDocSaveAs-Eigenschaft....616
OnlineOptions-Methode....616
OnlineRegistrationDlg-Methode....617
OnOpen-Eigenschaft....617
OnPrepareClose-Eigenschaft....618
OnPrint-Eigenschaft....619
OnQuit-Eigenschaft....620
OnSave-Eigenschaft....620
OnSaveAs-Eigenschaft....621
OnStart-Eigenschaft....622
Open-Methode....623
OpenDatabaseConnection-Methode....624
OpenForm-Methode....626
OpenQuery-Methode....628
OpenQueryConnection-Methode....629
OpenReport-Methode....630
OpenTable-Methode....631
OpenTableConnection-Methode....633
OpenTemplate-Methode....635
Optimize-Methode....636
OptionBarVisible-Eigenschaft....636
Organizer-Methode....637
Other-Eigenschaft....637

Eigenschaften und Methoden mit "P"....638

Palette-Eigenschaft....638
Parent-Eigenschaft....639
ParentWindow-Eigenschaft....640
Path-Eigenschaft....640
Path-Eigenschaft....641

PathSettings-Eigenschaft....641
Pattern-Eigenschaft....643
Phone-Eigenschaft....643
PhonePrivate-Eigenschaft....645
PlayMacro-Methode....646
PlugInsActive-Eigenschaft....647
PopupDocInfo-Eigenschaft....647
Position-Eigenschaft....648
Print-Methode....650
PrintDefault-Methode....650
Printer-Eigenschaft....651
PrintOut-Methode....652

Eigenschaften und Methoden mit "Q"....652

QueryAutoSave-Eigenschaft....652
Quit-Methode....653

Eigenschaften und Methoden mit "R"....654

ReadOnly-Eigenschaft....654
Receiver-Eigenschaft....655
Recording-Eigenschaft....655
Redo-Methode....656
RedoAction-Eigenschaft....657
RegExp-Eigenschaft....658
Relaxed-Eigenschaft....658
Reload-Methode....659
ReloadForEdit-Methode....660
Remove-Methode....661
RemoveLibrary-Methode....662
Repeat-Methode....662
RepeatAction-Eigenschaft....663
Restore-Methode....664
Resume-Methode....664
Revision-Eigenschaft....665
RightLine-Eigenschaft....666

Eigenschaften und Methoden mit "S"....667

Save-Methode....667
Save-Methode....668
SaveAll-Methode....669
SaveAs-Methode....670
SaveAsTemplate-Methode....671

SaveConfiguration-Methode....672
Saved-Eigenschaft....672
SaveSettings-Eigenschaft....673
SaveWorkingset-Eigenschaft....674
ScEditOptions-Methode....675
SdEditOptions-Methode....675
SearchDialog-Eigenschaft....676
SearchOriginal-Methode....676
Selection-Eigenschaft....677
Selection-Eigenschaft....677
SelectionText-Eigenschaft....678
SelectionTextExt-Eigenschaft....678
SendMail-Methode....679
SetFrameContent-Methode....680
SetFrameName-Methode....682
Shorter-Eigenschaft....684
ShowAddressBook-Methode....685
ShowBrowser-Eigenschaft....685
ShowQuery-Methode....686
ShowStatement-Methode....686
SimEditOptions-Methode....687
Similarity-Eigenschaft....687
SmEditOptions-Methode....688
SplitHorizontal-Methode....688
SplitParentHorizontal-Methode....690
SplitParentVertical-Methode....692
SplitVertical-Methode....694
StarCalc-Eigenschaft....696
StarDraw-Eigenschaft....696
Start-Methode....697
StarWriter-Eigenschaft....698
StatusBar-Eigenschaft....698
Stop-Methode....699
Street-Eigenschaft....699
Style-Eigenschaft....701
StyleCatalog-Methode....703
SubIni-Eigenschaft....703
Subject-Eigenschaft....704
Subject-Eigenschaft....706
Suspend-Methode....707
SwEditOptions-Methode....708
SyncExplorer-Methode....708

Eigenschaften und Methoden mit "T"....708

TaskBarVisible-Eigenschaft....708
Template-Eigenschaft....709
Tile-Methode....710
Title-Eigenschaft....711
Title-Eigenschaft....712
Title-Eigenschaft....714
ToolBarVisible-Eigenschaft....715
Top-Eigenschaft....716
TopLine-Eigenschaft....716
TopWindow-Eigenschaft....717
Transparent-Eigenschaft....718

Eigenschaften und Methoden mit "U"....718

Undo-Methode....719
UndoAction-Eigenschaft....720
UndoCount-Eigenschaft....721
UnloadLibrary-Methode....722

Eigenschaften und Methoden mit "V"....722

Value-Eigenschaft....723
VerticalLine-Eigenschaft....723
View-Eigenschaft....724
Visible-Eigenschaft....725

Eigenschaften und Methoden mit "W"....725

Width-Eigenschaft....725
Windows-Eigenschaft....726
WordOnly-Eigenschaft....727
Work-Eigenschaft....727

Eigenschaften und Methoden mit "Z"....728

ZIPCode-Eigenschaft....729

Referenzhilfe für Textdokumente....731

Eigenschaften und Methoden mit "A"....731

ActionEnd-Methode....731
ActionStart-Methode....732
Activate-Methode....732
AddAllUnknownWords-Methode....733
AlignBottom-Methode....734

AlignCharBottom-Methode....735
AlignCharTop-Methode....736
AlignHorizontalCenter-Methode....737
AlignLeft-Methode....739
Alignment-Eigenschaft....740
AlignRight-Methode....742
AlignRowBottom-Methode....743
AlignRowTop-Methode....744
AlignTop-Methode....744
AlignVerticalCenter-Methode....746
AlignVerticalCharCenter-Methode....747
AlignVerticalRowCenter-Methode....748
AnchorOnly-Eigenschaft....748
AnchorType-Eigenschaft....749
Auto-Eigenschaft....750
AutoFormat-Methode....751
AutoPilotAgenda-Methode....752
AutoPilotFax-Methode....752
AutoPilotLetter-Methode....753
AutoPilotMemo-Methode....753

Eigenschaften und Methoden mit "B"....754

BackColor-Eigenschaft....754
BackgroundColor-Eigenschaft....755
BackgroundDialog-Methode....755
Backspace-Methode....756
Bold-Eigenschaft....756
BookmarkCount-Eigenschaft....759
BookmarkName-Methode....760
Border-Eigenschaft....761
BorderDialog-Methode....762
Bottom-Eigenschaft....763
BottomMargin-Eigenschaft....764
Bounds-Eigenschaft....765
Break-Eigenschaft....767
BringToFront-Methode....768
BrowseView-Eigenschaft....769
BulletCode-Eigenschaft....769
BulletFont-Eigenschaft....770
BulletImageURL-Eigenschaft....772
BulletsAndNumberingDialog-Methode....773

Eigenschaften und Methoden mit "C"....773

Calc-Methode....773
CalculateSel-Methode....774
Caption-Eigenschaft....774
CaseMap-Eigenschaft....776
CellName-Eigenschaft....778
ChangeDatabaseField-Methode....779
ChapterNumberingDialog-Methode....780
CharSet-Eigenschaft....781
CharStyle-Eigenschaft....782
CharStyleApply-Methode....782
CharStyleCount-Eigenschaft....783
ClearHistory-Methode....784
CloseDatabaseConnection-Methode....785
CloseQueryConnection-Methode....787
CloseTableConnection-Methode....787
Color-Eigenschaft....789
Columns-Eigenschaft....792
Contour-Eigenschaft....792
ContourDialog-Eigenschaft....793
ContourWrap-Eigenschaft....794
ControlObject-Eigenschaft....794
Controls-Eigenschaft....796
ConvertTableText-Methode....797
Copy-Methode....798
Count-Eigenschaft....798
CreateAbstract-Methode....799
CreateForm-Methode....800
CreateGlossaryGroup-Methode....801
CreateIndex-Methode....802
CreateQuery-Methode....804
CreateReport-Methode....805
CreateTable-Methode....806
Crosshair-Eigenschaft....809
CurFieldSubType-Eigenschaft....811
CurFieldType-Eigenschaft....812
CursorXPos-Eigenschaft....812
CursorYPos-Eigenschaft....813
Cut-Methode....814

Eigenschaften und Methoden mit "D"....814

DataAppendRecord-Methode....815

DataAppendWholeRecord-Methode....817
DataBaseBrowser-Methode....820
DataColumn-Methode....821
DataColumnName-Methode....823
DataColumnNumber-Methode....824
DataColumnSubType-Methode....827
DataColumnType-Methode....829
DataDefinedDatabase-Methode....833
DataDefinedDatabaseNumber-Methode....835
DataDeleteRecord-Methode....836
DataFirstRecord-Methode....838
DataGetTableName-Methode....840
DataLastRecord-Methode....842
DataNextRecord-Methode....844
DataPhysicalRecordID-Methode....846
DataPreviousRecord-Methode....848
DataRecord-Methode....849
DataRecordID-Methode....851
DataSelect-Methode....853
DataSQLStatement-Methode....855
DataSwitch-Methode....856
DataTableNumber-Methode....858
DataUpdate-Methode....860
DataUpdateFields-Methode....862
DataUpdatePrint-Methode....863
DataUpdateRecord-Methode....865
DataUpdateWholeRecord-Methode....866
DataUsedDatabase-Methode....868
DataUsedDatabaseNumber-Methode....869
Deactivate-Methode....870
DecrementIndent-Methode....870
DecrementLevel-Methode....871
DecrementSubLevels-Methode....872
DefaultBullet-Methode....873
Delete-Methode....874
DeleteBookmark-Methode....874
DeleteColumns-Methode....876
DeleteForm-Methode....876
DeleteGlossary-Methode....877
DeleteGlossaryGroup-Methode....879
DeleteIndex-Methode....880
DeleteQuery-Methode....881

DeleteRegion-Methode....882
DeleteReport-Methode....883
DeleteRows-Methode....884
DeleteStyle-Methode....884
DeleteTable-Methode....885
DeleteUserField-Methode....888
DeleteVariable-Methode....888
DelLine-Methode....889
DelToEndOfLine-Methode....889
DelToEndOfPara-Methode....890
DelToEndOfSentence-Methode....891
DelToEndOfWord-Methode....891
DelToStartOfLine-Methode....892
DelToStartOfPara-Methode....893
DelToStartOfSentence-Methode....894
DelToStartOfWord-Methode....894
Distance-Eigenschaft....895
DistributeColumns-Methode....896
DistributeRows-Methode....896
DontExpand-Methode....897
DoUndo-Methode....898
DrawFont-Eigenschaft....899
Drawing-Eigenschaft....899
DropCap-Eigenschaft....901

Eigenschaften und Methoden mit "E"....902

EditGlossary-Methode....902
EditRegion-Methode....902
EditStyle-Methode....903
EntireColumn-Methode....904
EntireRow-Methode....905
Escape-Methode....906
Escapement-Eigenschaft....906
Exchange-Methode....909
ExecHyperlinks-Eigenschaft....909
ExpandGlossary-Methode....910

Eigenschaften und Methoden mit "F"....911

Family-Eigenschaft....911
FamilyName-Eigenschaft....912
FaxName-Eigenschaft....915
Field-Eigenschaft....916

FieldCount-Methode....919
FieldDialog-Methode....921
FieldName-Eigenschaft....922
FieldPrompt-Eigenschaft....924
FillColor-Eigenschaft....924
Flash-Eigenschaft....925
Font-Eigenschaft....927
FontColor-Eigenschaft....930
FontDialog-Methode....930
FontWork-Eigenschaft....931
FootnoteBackground-Eigenschaft....931
FootnoteDialog-Methode....933
ForeColor-Eigenschaft....934
FormatArea-Methode....934
FormatColumnValue-Methode....934
FormatGroup-Methode....936
FormatLine-Methode....936
FormatUngroup-Methode....937
Formula-Eigenschaft....937
FrameColumns-Methode....938
FrameDialog-Methode....939
FrameDistance-Methode....939
FrameName-Eigenschaft....941
Frames-Eigenschaft....942
FramesetSource-Eigenschaft....942
FrameStyle-Eigenschaft....943
FtnAdjust-Eigenschaft....944
FtnBottomDistance-Eigenschaft....945
FtnHeight-Eigenschaft....946
FtnPenStyle-Eigenschaft....948
FtnPenWidth-Eigenschaft....949
FtnTopDistance-Eigenschaft....950
FtnWidth-Eigenschaft....952

Eigenschaften und Methoden mit "G"....953

GetActiveDatabaseConnection-Methode....953
GetActiveTableConnection-Methode....955
GetCharCount-Methode....957
GetCharStyleName-Methode....958
GetClipboardFormat-Methode....959
GetClipboardFormatCount-Methode....960
GetClipboardFormatName-Methode....961

GetDatabaseFieldValue-Methode....962
GetDatabaseLimits-Methode....964
GetDataRecordCount-Methode....966
GetFieldFmt-Methode....968
GetFieldName-Methode....968
GetFieldPar1-Methode....971
GetFieldPar2-Methode....972
GetFieldTypeCount-Methode....973
GetFieldTypeId-Methode....974
GetFieldTypeName-Methode....976
GetFormNames-Methode....978
GetFramestyleMacroLib-Methode....979
GetFramestyleMacroSub-Methode....980
GetFramestyleScript-Methode....981
GetGlossaryBlockCount-Methode....983
GetGlossaryBlockName-Methode....984
GetGlossaryGroupCount-Methode....986
GetGlossaryGroupName-Methode....987
GetGlossaryShortName-Methode....988
GetHandler-Methode....990
GetHyperlinkMacroLib-Methode....991
GetHyperlinkMacroSub-Methode....992
GetHyperlinkScript-Methode....993
GetLineCount-Methode....993
GetPageStyleCount-Eigenschaft....994
GetPageStyleNameAt-Methode....995
GetParaStyleName-Methode....995
GetQueryColumns-Methode....996
GetQueryNames-Methode....998
GetRegionCondition-Methode....1001
GetRegionCount-Methode....1001
GetRegionLink-Methode....1002
GetRegionLinkBookmark-Methode....1003
GetRegionLinkFilter-Methode....1004
GetRegionName-Methode....1005
GetReportNames-Methode....1006
GetSQL_Statement-Methode....1007
GetTableColumns-Methode....1009
GetUserFieldValue-Methode....1011
GlobalContents-Eigenschaft....1012
GoDown-Methode....1014
GoIntoFrame-Methode....1015

GoldCut-Eigenschaft....1016
GoLeft-Methode....1016
GoRight-Methode....1017
GoToAnchor-Methode....1018
GotoBookmark-Methode....1019
GotoCell-Methode....1020
GoToEnd-Methode....1021
GoToEndOfColumn-Methode....1021
GoToEndOfDoc-Methode....1022
GoToEndOfLine-Methode....1023
GoToEndOfNextColumn-Methode....1025
GoToEndOfNextPage-Methode....1025
GoToEndOfPage-Methode....1026
GoToEndOfPara-Methode....1027
GoToEndOfPrevColumn-Methode....1028
GoToEndOfPrevPage-Methode....1029
GotoFrame-Methode....1030
GotoNextInputField-Methode....1031
GotoNextObject-Methode....1032
GoToNextPara-Methode....1033
GotoNextPlacemarker-Methode....1033
GoToNextSentence-Methode....1034
GoToNextWord-Methode....1035
GotoPage-Methode....1036
GotoPrevInputField-Methode....1037
GotoPrevObject-Methode....1037
GoToPrevPara-Methode....1038
GotoPrevPlacemarker-Methode....1039
GoToPrevSentence-Methode....1040
GoToPrevWord-Methode....1040
GoToStart-Methode....1041
GoToStartOfColumn-Methode....1042
GoToStartOfDoc-Methode....1043
GoToStartOfLine-Methode....1044
GoToStartOfNextColumn-Methode....1045
GoToStartOfNextPage-Methode....1046
GoToStartOfPage-Methode....1047
GoToStartOfPara-Methode....1048
GoToStartOfPrevColumn-Methode....1048
GoToStartOfPrevPage-Methode....1049
GotoTable-Methode....1050
GoUp-Methode....1051

Graphic-Eigenschaft....1052
GraphicDialog-Methode....1054
GraphicHeight-Eigenschaft....1054
GraphicMirror-Eigenschaft....1055
GraphicName-Eigenschaft....1056
Graphics-Eigenschaft....1057
GraphicURL-Eigenschaft....1058
GraphicWidth-Eigenschaft....1058
GrfFilename-Eigenschaft....1059
GrfFiltername-Eigenschaft....1060
GrfLinkFilterName-Eigenschaft....1061
GrfLinkName-Eigenschaft....1062
GrfPosition-Eigenschaft....1063
GroupCount-Methode....1064
GroupObject-Methode....1065
GroupUndoOnOff-Methode....1066
Grow-Methode....1066

Eigenschaften und Methoden mit "H"....1068

HeadlineRepeat-Eigenschaft....1068
Height-Eigenschaft....1069
HeightPercent-Eigenschaft....1070
HiddenText-Eigenschaft....1071
HideSpellMark-Eigenschaft....1073
HorzOrientation-Eigenschaft....1074
HorzRelation-Eigenschaft....1075
HorzRuler-Eigenschaft....1076
HorzScrollbar-Eigenschaft....1078
Hyphen-Eigenschaft....1080
Hyphenate-Methode....1081
HyphenZone-Eigenschaft....1081

Eigenschaften und Methoden mit "I"....1082

ImageMapDialog-Eigenschaft....1082
ImageMapURLAt-Methode....1084
ImageMapURLCount-Eigenschaft....1085
IncrementIndent-Methode....1087
IncrementLevel-Methode....1087
IncrementSubLevels-Methode....1088
IndentFirstLine-Eigenschaft....1089
Index-Eigenschaft....1090
IndexEntry-Eigenschaft....1091

IndexEntryDialog-Methode....1093
IndexMarkToIndex-Methode....1094
InsDelRowCol-Methode....1094
Insert-Methode....1096
InsertAnnotation-Methode....1097
InsertApplet-Methode....1097
InsertAuthorField-Methode....1098
InsertBookmark-Methode....1098
InsertBreak-Methode....1100
InsertCaptionDialog-Methode....1100
InsertColumnBreak-Methode....1101
InsertColumns-Methode....1101
InsertDateField-Methode....1102
InsertDBField-Methode....1103
InsertDocument-Methode....1105
InsertEnvelop-Methode....1107
InsertField-Methode....1107
InsertFile-Methode....1113
InsertFootnote-Methode....1114
InsertFootnoteDirect-Methode....1116
InsertFrame-Methode....1117
InsertGlossary-Methode....1118
InsertGraphic-Methode....1120
InsertGraphicRuler-Methode....1121
InsertHardHyphen-Methode....1122
InsertHyperlinkDlg-Methode....1122
InsertIndexEntry-Methode....1123
InsertIndexesEntry-Methode....1124
InsertIndexesIndex-Methode....1126
InsertIndexesIndex-Methode....1128
InsertIndexesTOC-Methode....1130
InsertIndexesTOC-Methode....1132
InsertIndexesUserIndex-Methode....1134
InsertIndexesUserIndex-Methode....1136
InsertLabels-Methode....1139
InsertLinebreak-Methode....1139
InsertMode-Eigenschaft....1140
InsertNamedTable-Methode....1140
InsertNeutralParagraph-Methode....1141
InsertNonBreakingSpace-Methode....1142
InsertNumbering-Methode....1142
InsertObject-Methode....1143

InsertObjectFloatingFrame-Methode....1144
InsertObjectStarChart-Methode....1144
InsertObjectStarImage-Methode....1145
InsertObjectStarMath-Methode....1145
InsertPagebreak-Methode....1146
InsertPageCountField-Methode....1147
InsertPageNumberField-Methode....1147
InsertPara-Methode....1148
InsertPlugin-Methode....1149
InsertRegion-Methode....1149
InsertRows-Methode....1151
InsertScript-Methode....1152
InsertSoftHyphen-Methode....1153
InsertSymbol-Methode....1153
InsertTable-Methode....1154
InsertTimeField-Methode....1155
InsertTitleField-Methode....1155
InsertTOCEntry-Methode....1156
InsertTopicField-Methode....1157
InsertUserEntry-Methode....1158
InsertVCControl-Methode....1159
InsertVCControlAnchored-Methode....1161
Interior-Eigenschaft....1162
InterLineSpace-Eigenschaft....1163
InterSpace-Eigenschaft....1164
IsAlignment-Eigenschaft....1165
IsBold-Eigenschaft....1166
IsCaseMap-Eigenschaft....1168
IsClientImageMap-Eigenschaft....1169
IsContour-Eigenschaft....1170
IsControl-Eigenschaft....1171
IsEndOfDoc-Eigenschaft....1172
IsEndOfLine-Eigenschaft....1173
IsEndOfPara-Eigenschaft....1174
IsEndOfSentence-Eigenschaft....1175
IsEndOfWord-Eigenschaft....1176
IsEscapement-Eigenschaft....1177
IsFirstRecord-Methode....1178
IsFont-Eigenschaft....1180
IsFontColor-Eigenschaft....1182
IsFontHeight-Eigenschaft....1183
IsHyphenZone-Eigenschaft....1184

IsItalic-Eigenschaft....1185
IsLanguage-Eigenschaft....1186
IsLastRecord-Methode....1187
IsLeftRightParaMargin-Eigenschaft....1190
IsLineSpacing-Eigenschaft....1191
IsOrphan-Eigenschaft....1192
IsPaintLocked-Eigenschaft....1193
IsParaSplit-Eigenschaft....1193
IsQueryNative-Methode....1195
IsQueryReadOnly-Methode....1197
IsRegionHidden-Methode....1199
IsRegionPassword-Eigenschaft....1200
IsRegionProtected-Methode....1201
IsServerImageMap-Eigenschaft....1201
IsShadowed-Eigenschaft....1203
IsSpacing-Eigenschaft....1204
IsStartOfDoc-Eigenschaft....1205
IsStartOfPara-Eigenschaft....1206
IsStartOfWord-Eigenschaft....1207
IsStrikeOut-Eigenschaft....1208
IsTableReadOnly-Methode....1209
IsTopBottomMargin-Eigenschaft....1211
IsUnderline-Eigenschaft....1211
IsWidow-Eigenschaft....1213
IsWordMode-Eigenschaft....1214
Italic-Eigenschaft....1215

Eigenschaften und Methoden mit "J"....1218

JumpDownThisLevel-Methode....1218
JumpToBookmark-Methode....1219
JumpToEndOfDoc-Methode....1220
JumpToFooter-Methode....1221
JumpToFootnoteOrAnchor-Methode....1221
JumpToHeader-Methode....1222
JumpToMark-Methode....1223
JumpToNextBookmark-Methode....1224
JumpToNextCell-Methode....1226
JumpToNextFootnote-Methode....1228
JumpToNextFrame-Methode....1229
JumpToNextRegion-Methode....1230
JumpToNextSelection-Methode....1231
JumpToNextTable-Methode....1231

JumpToPrevBookmark-Methode....1232
JumpToPrevCell-Methode....1233
JumpToPrevFootnote-Methode....1234
JumpToPrevRegion-Methode....1235
JumpToPrevSelection-Methode....1236
JumpToPrevTable-Methode....1236
JumpToReference-Methode....1237
JumpToRegion-Methode....1238
JumpToStartOfDoc-Methode....1239
JumpToURL-Methode....1239
JumpUpThisLevel-Methode....1240

Eigenschaften und Methoden mit "K"....1241

Eigenschaften und Methoden mit "L"....1241

Language-Eigenschaft....1241
LaunchStarImage-Methode....1245
Left-Eigenschaft....1245
LeftMargin-Eigenschaft....1246
LineColor-Eigenschaft....1247
LineDash-Eigenschaft....1248
LineHeight-Eigenschaft....1249
Lines-Eigenschaft....1250
LineSpace-Eigenschaft....1251
LineText-Eigenschaft....1252
LinkDialog-Methode....1253
LinkName-Eigenschaft....1253
LinkTarget-Eigenschaft....1254
LinkType-Eigenschaft....1255
LinkURL-Eigenschaft....1256
LoadStyles-Methode....1256
Location-Eigenschaft....1258
LockPaint-Methode....1258
LogicalPageNumber-Eigenschaft....1259

Eigenschaften und Methoden mit "M"....1260

ManualBreak-Eigenschaft....1260
MaxHyphens-Eigenschaft....1263
MergeCells-Methode....1264
MergeDialog-Methode....1264
MetricUnit-Eigenschaft....1264
MinLead-Eigenschaft....1265
MinTrail-Eigenschaft....1266

MoveDown-Methode....1267
MoveDownSubItems-Methode....1268
MoveRowCol-Methode....1269
MoveUp-Methode....1270
MoveUpSubItems-Methode....1271
MultiLine-Eigenschaft....1272

Eigenschaften und Methoden mit "N"....1272

NewField-Methode....1273
NewGlobalDoc-Methode....1273
NewGlossary-Methode....1275
NewStyle-Methode....1277
NewWindow-Methode....1278
NextFieldByType-Methode....1278
NonbreakingSpace-Eigenschaft....1279
Notes-Eigenschaft....1282
NumberFormat-Eigenschaft....1284
NumberFormatCurrency-Methode....1284
NumberFormatDate-Methode....1285
NumberFormatDecimal-Methode....1285
NumberFormatPercent-Methode....1286
NumberFormatScientific-Methode....1286
NumberFormatStandard-Methode....1287
NumberFormatTime-Methode....1287
NumberLevel-Eigenschaft....1288
NumberOfPages-Eigenschaft....1289
NumberOrNoNumber-Eigenschaft....1290
NumberType-Eigenschaft....1290

Eigenschaften und Methoden mit "O"....1292

ObjectBackOne-Methode....1292
ObjectForwardOne-Methode....1293
ObjectMirrorHorizontal-Methode....1294
ObjectMirrorVertical-Methode....1294
ObjectName-Eigenschaft....1294
OleObject-Eigenschaft....1295
OLEObjects-Eigenschaft....1296
OnlineAutoFormat-Eigenschaft....1297
OnlineOptions-Methode....1297
Opaque-Eigenschaft....1298
OpenDatabaseConnection-Methode....1298
OpenForm-Methode....1301

OpenQuery-Methode....1302
OpenQueryConnection-Methode....1304
OpenReport-Methode....1304
OpenTable-Methode....1306
OpenTableConnection-Methode....1308
Orphan-Eigenschaft....1310
OutlineFont-Eigenschaft....1311
Outside-Eigenschaft....1313

Eigenschaften und Methoden mit "P"....1314

PageColumnDialog-Methode....1314
PageDialog-Methode....1314
PageDown-Methode....1315
PageEnd-Eigenschaft....1316
PageHeight-Eigenschaft....1317
PageNumber-Eigenschaft....1318
PageOffset-Eigenschaft....1319
PageOffsetDialog-Methode....1320
PagePaperBin-Eigenschaft....1321
PageStyle-Eigenschaft....1321
PageStyleApply-Methode....1322
PageStyleName-Eigenschaft....1323
PageUp-Methode....1324
PageWidth-Eigenschaft....1324
PairKerning-Eigenschaft....1325
PaperFromSetup-Eigenschaft....1328
Paragraph-Eigenschaft....1329
ParagraphDialog-Methode....1330
ParagraphEnd-Eigenschaft....1330
ParaSplit-Eigenschaft....1332
ParaStyle-Eigenschaft....1334
ParaStyleApply-Methode....1334
ParaStyleCount-Eigenschaft....1335
Paste-Methode....1336
PasteSpecial-Methode....1336
Pitch-Eigenschaft....1338
PrevFieldByType-Methode....1339
Print-Methode....1340
Printable-Eigenschaft....1341
PrintAnnotations-Eigenschaft....1342
PrintBackground-Eigenschaft....1343
PrintBlackFont-Eigenschaft....1345

PrintControl-Eigenschaft....1346
PrintDefault-Methode....1348
PrintDrawing-Eigenschaft....1348
PrintGraphic-Eigenschaft....1350
PrintLayout-Eigenschaft....1351
PrintLeftPage-Eigenschaft....1352
PrintPagePreView-Methode....1353
PrintPreview-Eigenschaft....1354
PrintProspect-Eigenschaft....1355
PrintReverse-Eigenschaft....1356
PrintRightPage-Eigenschaft....1358
PrintSettings-Eigenschaft....1359
PrintTable-Eigenschaft....1360
Prop-Eigenschaft....1362
PropLineHeight-Eigenschaft....1363
Protect-Methode....1364
Protected-Eigenschaft....1365

Eigenschaften und Methoden mit "Q"....1366

QualifierSeparator-Methode....1366
QuoteIdentifier-Methode....1367

Eigenschaften und Methoden mit "R"....1368

Redo-Methode....1368
RedoAction-Eigenschaft....1369
Refresh-Methode....1370
RelativeSize-Eigenschaft....1370
Remove-Methode....1373
RenameGlossary-Methode....1374
RenameRegion-Methode....1375
RenameUserFields-Methode....1376
Repaginate-Methode....1376
Repeat-Methode....1377
RepeatAction-Eigenschaft....1377
RepeatSearch-Methode....1378
Replace-Methode....1379
ReplaceAll-Methode....1381
ReplaceAttributes-Eigenschaft....1382
Reset-Methode....1383
ResetAllActions-Methode....1384
ResetAllLocks-Methode....1385
ResetAttributes-Methode....1385

ResetTableProtection-Methode....1387
Right-Eigenschaft....1387
RightMargin-Eigenschaft....1388
Rows-Eigenschaft....1389

Eigenschaften und Methoden mit "S"....1390

SaveAs-Methode....1390
Search-Methode....1391
SearchAll-Methode....1393
SearchAttributes-Eigenschaft....1394
SearchCount-Methode....1395
SearchSettings-Eigenschaft....1396
Select-Methode....1397
SelectAll-Methode....1397
SelectionCount-Eigenschaft....1398
SelectionMode-Eigenschaft....1398
SelectText-Methode....1399
SelectWord-Methode....1400
SendAbstractToStarImpress-Methode....1400
SendFax-Methode....1401
SendOutlineToClipboard-Methode....1402
SendOutlineToStarImpress-Methode....1403
SendToBack-Methode....1404
SetActGlossaryGroup-Methode....1405
SetActions-Methode....1406
SetAlignment-Methode....1407
SetAnchorToChar-Methode....1408
SetAnchorToPage-Methode....1409
SetAnchorToPara-Methode....1409
SetBold-Methode....1410
SetCaseMap-Methode....1411
SetColumnWidth-Methode....1412
SetContour-Methode....1413
SetCurField-Methode....1414
SetDefaultBorder-Methode....1415
SetEscapement-Methode....1416
SetFixDateTimeFields-Methode....1417
SetFont-Methode....1417
SetFontColor-Methode....1419
SetFontHeight-Methode....1420
SetFramestyleMacro-Methode....1421
SetFrameStyleScript-Methode....1422

SetHandler-Methode....1424
SetHyperlink-Methode....1425
SetHyperlink-Methode....1426
SetHyperlinkMacro-Methode....1427
SetHyperlinkScript-Methode....1428
SetHyphenZone-Methode....1429
SetItalic-Methode....1430
SetLanguage-Methode....1431
SetLeftRightParaMargin-Methode....1433
SetLineSpacing-Methode....1433
SetLocks-Methode....1435
SetObjectToBackground-Methode....1435
SetObjectToForeground-Methode....1436
SetOptimalColumnWidth-Methode....1437
SetOptimalRowHeight-Methode....1437
SetOrphan-Methode....1438
SetPageStyleCols-Methode....1439
SetPageStyleFooter-Methode....1439
SetPageStyleHeader-Methode....1441
SetParaSplit-Methode....1442
SetRegionCondition-Methode....1443
SetRegionHidden-Methode....1444
SetRegionLink-Methode....1445
SetRegionLinkBookmark-Methode....1446
SetRegionPassword-Methode....1447
SetRegionProtected-Methode....1447
SetRowHeight-Methode....1448
SetShadowed-Methode....1449
SetSpacing-Methode....1450
SetStrikeOut-Methode....1451
SetTopBottomMargin-Methode....1452
SetUnderline-Methode....1452
SetUserFieldSubType-Methode....1453
SetUserFieldValue-Methode....1454
SetWidow-Methode....1455
SetWordMode-Methode....1456
Shadow-Eigenschaft....1457
Shadowed-Eigenschaft....1457
ShowCursor-Eigenschaft....1460
ShowFourPages-Methode....1461
ShowHiddenPara-Eigenschaft....1462
ShowQuery-Methode....1464

ShowStatement-Methode....1464
ShowTwoPages-Methode....1465
Shrink-Methode....1466
Size-Eigenschaft....1468
SmoothScroll-Eigenschaft....1471
SoftHyphen-Eigenschaft....1473
SolidHandles-Eigenschaft....1475
Sort-Methode....1477
SortDialog-Methode....1479
Space-Eigenschaft....1479
Spacing-Eigenschaft....1481
Spelling-Methode....1484
SpellOnline-Eigenschaft....1484
SplitCell-Methode....1485
StartAutoCorrect-Methode....1485
StatusLine-Eigenschaft....1486
StrikeOut-Eigenschaft....1488
Style-Eigenschaft....1491
StyleApply-Methode....1492
StyleApplyState-Eigenschaft....1494
StyleCatalog-Methode....1495
StyleName-Eigenschaft....1495
StyleNewByExample-Methode....1496
StyleUpdateByExample-Methode....1498
StyleWatercanMode-Eigenschaft....1499
SubScript-Eigenschaft....1500
SuperScript-Eigenschaft....1502
Surround-Eigenschaft....1505
SwEditOptions-Methode....1506

Eigenschaften und Methoden mit "T"....1506

Tab-Eigenschaft....1506
Table-Eigenschaft....1508
TableBounds-Eigenschaft....1510
TableDialog-Methode....1512
TableModeFix-Methode....1513
TableModeFixProp-Methode....1513
TableModeVariable-Methode....1514
TableName-Eigenschaft....1515
TableNumberFormatDialog-Methode....1515
TableOptimizeMinMax-Methode....1516
Tables-Eigenschaft....1516

Tabstops-Eigenschaft....1517
Tag-Eigenschaft....1518
TargetFrame-Eigenschaft....1519
TextAttributes-Methode....1519
TextWrap-Methode....1520
ThesaurusDialog-Methode....1520
Title-Methode....1520
Top-Eigenschaft....1522
TopMargin-Eigenschaft....1523
TransformDialog-Methode....1525
Transparent-Eigenschaft....1525
Type-Eigenschaft....1526

Eigenschaften und Methoden mit "U"....1528

Underline-Eigenschaft....1528
UnderlineDouble-Eigenschaft....1530
Undo-Methode....1531
UndoAction-Eigenschaft....1532
UndoCount-Eigenschaft....1533
UndoMode-Eigenschaft....1534
UnsetCellsReadOnly-Methode....1535
UnvisitedFmt-Eigenschaft....1535
UpdateAllIndexes-Methode....1536
UpdateAllLinks-Methode....1537
UpdateDocstat-Methode....1539
UpdateFields-Methode....1539
UpdateInputFields-Methode....1540
URL-Eigenschaft....1541
URLFieldCount-Methode....1541

Eigenschaften und Methoden mit "V"....1542

Value-Eigenschaft....1542
ValueAndBreaks-Eigenschaft....1544
ValueAtCursor-Eigenschaft....1545
VerticalCellAlignment-Eigenschaft....1546
VertOrientation-Eigenschaft....1546
VertRelation-Eigenschaft....1548
VertRuler-Eigenschaft....1549
VertScrollbar-Eigenschaft....1551
ViewSettings-Eigenschaft....1553
VisitedFmt-Eigenschaft....1556

Eigenschaften und Methoden mit "W"....1557

WatercanStyleModeOnOff-Methode....1557
Widow-Eigenschaft....1558
Width-Eigenschaft....1559
WidthPercent-Eigenschaft....1560
WordMode-Eigenschaft....1561
Wrap-Methode....1563

Eigenschaften und Methoden mit "X"....1565

XPos-Eigenschaft....1565

Eigenschaften und Methoden mit "Y"....1566

YPos-Eigenschaft....1566

Eigenschaften und Methoden mit "Z"....1567

Zoom-Eigenschaft....1567

Referenzhilfe für Tabellendokumente....1570

Eigenschaften und Methoden mit "A"....1570

Activate-Methode....1570
ActiveCell-Eigenschaft....1572
ActiveTable-Eigenschaft....1572
Add-Methode....1573
AddChartData-Methode....1574
AddPrintArea-Methode....1575
Address-Eigenschaft....1575
AlignBlock-Methode....1576
AlignBottom-Methode....1577
AlignHCenter-Methode....1578
AlignLeft-Methode....1579
AlignRight-Methode....1580
AlignTop-Methode....1581
AlignVCenter-Methode....1582
AuditingFillMode-Methode....1583
AutoComplete-Eigenschaft....1583
AutoFill-Methode....1584
AutoFormat-Methode....1584
AutomaticCalculation-Eigenschaft....1585
AutoOutline-Methode....1586

Eigenschaften und Methoden mit "B"....1587

BackgroundColor-Eigenschaft....1587
Backward-Eigenschaft....1588
Bold-Eigenschaft....1589
Border-Eigenschaft....1589

Eigenschaften und Methoden mit "C"....1590

Calculate-Methode....1591
CalculateHard-Methode....1591
CaseSensitive-Eigenschaft....1592
Cells-Methode....1593
Cells-Methode....1594
CellText-Methode....1595
ChangeChartData-Methode....1596
ChangePrintArea-Methode....1597
CharSet-Eigenschaft....1598
Clear-Methode....1599
ClearArrowDependents-Methode....1600
ClearArrowPrecedents-Methode....1600
ClearArrows-Methode....1601
ClearContents-Methode....1602
ClearOutline-Methode....1602
Color-Eigenschaft....1603
Column-Eigenschaft....1604
ColumnWidth-Eigenschaft....1606
ConditionalFormatDialog-Methode....1607
Copy-Methode....1607
CreateNames-Methode....1608
CursorEndOfScreen-Methode....1608
CursorTopOfScreen-Methode....1609
Cut-Methode....1609

Eigenschaften und Methoden mit "D"....1610

DataAreaRefresh-Methode....1610
DataConsolidate-Methode....1610
DataDataPilotRun-Methode....1611
DataFilterAutoFilter-Methode....1611
DataFilterHideAutoFilter-Methode....1612
DataFilterRemoveFilter-Methode....1612
DataFilterSpecialFilter-Methode....1612
DataFilterStandardFilter-Methode....1613
DataImport-Methode....1613

DataReImport-Methode....1614
DataSelect-Methode....1615
DataSort-Methode....1615
DataSubTotals-Methode....1616
Deactivate-Methode....1616
DecrementIndent-Methode....1617
DefineDBName-Methode....1618
DefineLabelRange-Methode....1618
DefineName-Methode....1619
DefinePrintArea-Methode....1620
Delete-Methode....1620
DeleteColumnbreak-Methode....1621
DeleteColumns-Methode....1621
DeletePrintArea-Methode....1622
DeleteRowbreak-Methode....1622
DeleteRows-Methode....1623
DeleteTemplate-Methode....1623
Deselect-Methode....1624

Eigenschaften und Methoden mit "E"....1624

EditHeaderAndFooter-Methode....1624
EditLinks-Methode....1625
EditPrintArea-Methode....1625
EditTemplate-Methode....1626

Eigenschaften und Methoden mit "F"....1626

Family-Eigenschaft....1626
FamilyName-Eigenschaft....1628
FillDown-Methode....1628
FillLeft-Methode....1629
FillRight-Methode....1630
FillSeries-Methode....1630
FillUp-Methode....1632
Font-Eigenschaft....1632
FormatCellDialog-Methode....1633
Formula-Eigenschaft....1634
FreezePanes-Methode....1634

Eigenschaften und Methoden mit "G"....1635

GoalSeekDialog-Methode....1635
GoDown-Methode....1635
GoDownBlock-Methode....1636

GoDownToEndOfData-Methode....1637
GoLeft-Methode....1638
GoLeftBlock-Methode....1638
GoLeftToStartOfData-Methode....1639
GoRight-Methode....1640
GoRightBlock-Methode....1641
GoRightToEndOfData-Methode....1641
GoToCell-Methode....1642
GoToCurrentCell-Methode....1643
GotoDocument-Methode....1643
GoToEndOfData-Methode....1644
GoToEndOfRow-Methode....1644
GoToObject-Methode....1645
GoToStart-Methode....1645
GoToStartOfRow-Methode....1646
GoUp-Methode....1647
GoUpBlock-Methode....1647
GoUpToStartOfData-Methode....1648
Group-Methode....1649

Eigenschaften und Methoden mit "H"....1650

Hide-Methode....1650
HideColumn-Methode....1651
HideDetail-Methode....1651
HideRow-Methode....1652
HorizontalAlignment-Eigenschaft....1653

Eigenschaften und Methoden mit "I"....1654

ImageMapDialog-Eigenschaft....1654
IncrementIndent-Methode....1655
Index-Eigenschaft....1656
InSelection-Eigenschaft....1656
Insert-Methode....1657
Insert-Methode....1658
InsertApplet-Methode....1659
InsertCell-Methode....1659
InsertCellsDown-Methode....1660
InsertCellsRight-Methode....1661
InsertColumnBreak-Methode....1661
InsertColumns-Methode....1662
InsertContents-Methode....1663
InsertGraphic-Methode....1664

InsertMatrix-Methode....1665
InsertName-Methode....1665
InsertObject-Methode....1666
InsertObjectFloatingFrame-Methode....1666
InsertObjectStarChart-Methode....1666
InsertObjectStarImage-Methode....1668
InsertObjectStarMath-Methode....1668
InsertPlugin-Methode....1669
InsertRowBreak-Methode....1669
InsertRows-Methode....1670
Interior-Eigenschaft....1670
Italic-Eigenschaft....1671

Eigenschaften und Methoden mit "J"....1672

JumpToNextCell-Methode....1672
JumpToNextTable-Methode....1673
JumpToNextUnprotected-Methode....1673
JumpToPreviousCell-Methode....1674
JumpToPreviousUnprotected-Methode....1674
JumpToPrevTable-Methode....1675
JumpToTable-Methode....1676

Eigenschaften und Methoden mit "L"....1676

Longer-Eigenschaft....1676

Eigenschaften und Methoden mit "M"....1677

MergeCells-Methode....1677
MetricUnit-Eigenschaft....1678
Mode-Eigenschaft....1679
Move-Methode....1680

Eigenschaften und Methoden mit "N"....1681

Name-Eigenschaft....1681
NewTemplate-Methode....1682
NoteText-Eigenschaft....1682
NoteVisible-Eigenschaft....1683
NumberFormat-Eigenschaft....1684
NumberFormatCurrency-Methode....1685
NumberFormatDate-Methode....1685
NumberFormatDecDecimals-Methode....1686
NumberFormatDecimal-Methode....1687
NumberFormatIncDecimals-Methode....1688

NumberFormatPercent-Methode....1689
NumberFormatScientific-Methode....1690
NumberFormatStandard-Methode....1690
NumberFormatTime-Methode....1691

Eigenschaften und Methoden mit "O"....1692

Offset-Methode....1692
OLEObject-Eigenschaft....1693
OLEObjects-Eigenschaft....1694
Order-Eigenschaft....1695
Other-Eigenschaft....1696

Eigenschaften und Methoden mit "P"....1697

PageFormatDialog-Methode....1697
ParaStyle-Eigenschaft....1697
Paste-Methode....1698
PasteSpecial-Methode....1698
Pitch-Eigenschaft....1698
PrintPreview-Eigenschaft....1699
Prop-Eigenschaft....1700
Protect-Methode....1700
PutCell-Methode....1701

Eigenschaften und Methoden mit "R"....1702

Range-Methode....1702
RefreshArrows-Methode....1703
RegExp-Eigenschaft....1703
Relaxed-Eigenschaft....1704
Remove-Methode....1705
Repaint-Methode....1706
RepeatSearch-Methode....1706
Replace-Methode....1707
ReplaceAll-Methode....1707
ResetAttributes-Methode....1708
Row-Eigenschaft....1709
RowHeight-Eigenschaft....1710

Eigenschaften und Methoden mit "S"....1712

Scale-Methode....1712
ScEditOptions-Methode....1712
Search-Methode....1712
SearchAll-Methode....1713

SearchSettings-Eigenschaft....1714
Select-Methode....1715
SelectAll-Methode....1715
SelectColumn-Methode....1716
SelectedObject-Eigenschaft....1716
Selection-Eigenschaft....1717
SelectRow-Methode....1718
SetHyperlink-Methode....1718
SetInputMode-Methode....1719
SetOptimalColumnWidth-Methode....1720
SetOptimalColumnWidthDirect-Methode....1721
SetOptimalRowHeight-Methode....1721
Shorter-Eigenschaft....1722
Show-Methode....1723
ShowColumn-Methode....1724
ShowDependents-Methode....1724
ShowDetail-Methode....1725
ShowErrors-Methode....1726
ShowInvalid-Methode....1727
ShowPrecedents-Methode....1727
ShowRow-Methode....1728
Similarity-Eigenschaft....1728
Size-Eigenschaft....1729
SortAscending-Methode....1730
SortDescending-Methode....1730
Spelling-Methode....1731
SplitCell-Methode....1731
SplitWindow-Methode....1732
StarChartDialog-Methode....1733
StatusDocPos-Eigenschaft....1733
StatusSelectionMode-Eigenschaft....1734
StyleApply-Methode....1734
StyleName-Eigenschaft....1736
StyleNewByExample-Methode....1737
StyleUpdateByExample-Methode....1738
StyleWatercanMode-Eigenschaft....1739

Eigenschaften und Methoden mit "T"....1740

Table-Eigenschaft....1740
TableCount-Eigenschaft....1741
TableOperationDialog-Methode....1742
Tables-Eigenschaft....1742

TableSelectAll-Methode....1743
TextValue-Eigenschaft....1743
Thesaurus-Methode....1744
ToggleRelative-Methode....1744
ToolProtectionDocument-Methode....1746

Eigenschaften und Methoden mit "U"....1746

Underline-Eigenschaft....1746
Ungroup-Methode....1747
UpdateChart-Methode....1748

Eigenschaften und Methoden mit "V"....1749

Validation-Methode....1749
Value-Eigenschaft....1750
VerticalAlignment-Eigenschaft....1751
ViewInputLineOnOff-Methode....1752
ViewRowColumnHeadersOnOff-Methode....1752
ViewValueHighlightingOnOff-Methode....1753
Visible-Eigenschaft....1754

Eigenschaften und Methoden mit "W"....1754

WordOnly-Eigenschaft....1754
WrapText-Eigenschaft....1755

Eigenschaften und Methoden mit "Z"....1756

Zoom-Eigenschaft....1756

Programmieren

Hier erhalten Sie grundlegende Informationen über den Umgang mit Makros und StarBasic.

Grundlagen

Hier vermittlen wir Ihnen das Grundlagenwissen, um mit StarBasic arbeiten zu können.

Wozu benötigt man StarBasic?

An dieser Stelle wird beschrieben, für welche Arbeiten es sich lohnt, StarBasic zu benutzen.

Einleitung

Je intensiver Sie mit einem StarOffice-Programm arbeiten, desto mehr werden Sie feststellen, daß sich die eigentliche Arbeit mit dem Programm dank der Objektorientiertheit von StarOffice auf immer wiederkehrende Handgriffe reduziert. Daher verlangt man irgendwann nach noch stärkerer Automatisierung. Um das Arbeiten mit StarOffice gänzlich zu perfektionieren haben sich die StarOffice-Entwickler eine Besonderheit einfallen lassen: Die Makro-Programmiersprache von StarOffice - StarBasic.

Die einfachste Möglichkeit, von diesen Makros Gebrauch zu machen, ist, sie von der Applikation, in der Sie sie benötigen, aufzeichnen zu lassen. Sie starten den Makrorekorder, und das StarOffice-Programm, in dem Sie sich befinden, merkt sich von diesem Zeitpunkt an solange alle Aktionen, die Sie durchführen, bis Sie die Makroaufzeichnung beenden. Ein auf diese Art und Weise aufgezeichnetes Makro können Sie dann jederzeit wieder ablaufen lassen und damit Arbeitsvorgänge automatisieren.

Doch damit nicht genug: Wie der Name schon sagt, handelt es sich bei der Makro-Programmiersprache StarBasic um eine richtige Programmiersprache - nämlich BASIC. StarBasic ist so mächtig, daß man sie schon als Programmentwicklungssystem bezeichnen kann. In dieser Dokumentation möchten wir Ihnen dieses Programmentwicklungssystem StarBasic nahebringen.

Aufbau der Hilfe

Diese Hilfe beschreibt Ihnen zunächst das Konzept von StarBasic. Sie erfahren, was genau hinter der Makroaufzeichnung steckt und wie eine

StarOffice-Applikation Makros als Basic-Programm ablegt. Anhand eines ersten kleinen StarBasic-Projekts stellt Ihnen dieser Abschnitt das grundsätzliche StarBasic-Konzept und seine Möglichkeiten vor.

Wir gehen auf alle notwendigen Fakten, die Sie für die Entwicklung eines StarBasic-Projektes benötigen, im Detail ein. Sie erfahren, wie man Projekte mit eigenen Dialogen erstellt und wie Sie mit dem StarBasic-Editor arbeiten. Sie erfahren, wie Sie Basic-Projekte in Modulen und Bibliotheken strukturieren und auf welche Art und Weise Sie spezielle Funktionen des jeweiligen StarOffice-Bausteins (Text-, Tabellen-, Präsentations- oder Diagrammdokument) aus Basic heraus nutzen können. Auch das Fehlerbereinigen von Programmen wird beschrieben.

StarBasic effektiv einsetzen

Es gibt zwei Möglichkeiten, die Makro-Programmiersprache StarBasic zu nutzen. Zum einen können Sie, wie schon erwähnt, jede StarOffice-Applikation anweisen, ein Makro aufzuzeichnen und diese aufgezeichneten Arbeitsanweisungen jederzeit wiederholen.

Sieht man sich dabei die Aufzeichnung eines Makros näher an, stellt man fest, daß die jeweilige Applikation bei jeder Makroaufzeichnung ein StarBasic-Programm schreibt, das den zuvor durchgeführten und protokollierten Anwenderaktionen entspricht. Die Ausführung dieser Aktionen bewirkt, daß nacheinander die den Anwenderaktionen entsprechenden StarBasic-Funktionen aufgerufen werden.

StarBasic ist jedoch für weit mehr geeignet. Denn zum anderen können Sie mit Hilfe von StarBasic und dessen Benutzerumgebung, der sogenannten "Integrated Development Environment", kurz IDE genannt, wie in einer professionellen Basic-Entwicklerumgebung arbeiten. Sie entwickeln hier Ihre eigenen Basic-Projekte mit der professionellen Unterstützung einer Entwicklungsumgebung, die anderen Basic-Entwicklungssystemen in keiner Weise nachsteht - nur mit dem Unterschied, daß Sie die komplette Funktionsvielfalt einer jeden StarOffice-Applikation von Basic aus mitnutzen können.

Die Möglichkeiten, die sich hieraus ergeben, sind enorm: Sie können im Prinzip eine Office-Oberfläche aufbauen, die auf Ihre ganz bestimmten Bedürfnisse zugeschnitten ist. Oder gehen Sie noch einen Schritt weiter und entwickeln Sie kommerzielle Anwendungen für den Vertrieb.

Voraussetzungen

Hier wird erläutert, welche Vorkenntnisse Sie haben müssen, um StarBasic zu verstehen.

Wenn Sie selber Hand anlegen und StarBasic für mehr als nur für die reine Makroaufzeichnung nutzen möchten, dann sollten Sie die Beherrschung eines Basic-Dialektes als Voraussetzung mitbringen. Diese Hilfe soll Ihnen nicht als Einführung in diese Programmiersprache dienen, denn das würde den Rahmen eindeutig sprengen. Möchten Sie Basic lernen, erkundigen Sie sich im Fachhandel nach einschlägiger Literatur zu Basic; es gibt mehr als genug.

Beherrschen Sie jedoch einen Basic-Dialekt, und möchten Sie in die Entwicklung von StarBasic-Anwendungen einsteigen, dann ist das eine ausreichende Voraussetzung. Sicherlich gibt es hier und da einige Abweichungen von Ihnen bekannten Basic-Konzepten. Aber die finden Sie an den jeweiligen Stellen dieser Dokumentation beschrieben. Sie werden nach kurzer Einarbeitungszeit feststellen, daß es ein Leichtes ist, eigene StarBasic-Applikationen zu entwerfen.

StarBasic ist objektorientiert

In einer objektorientierten Programmiersprache stehen dem Programmierer Objekte zur Verfügung, die vorgegebene Eigenschaften haben und bestimmte Methoden ermöglichen. Der Programmierer versucht, durch das Anwenden der möglichen Methoden auf die vorhandenen Objekte das Ziel des Programms zu erreichen.

Was sind Klassen?

Eine Klasse ist ein abstrakter Begriff für eine offene Menge von Objekten, mit ähnlichen Eigenschaften. In der realen Welt könnte man von der KLASSE DER MENSCHEN sprechen. Mit Unter- bzw. Oberklassen (auch Sub- bzw. Super-Klassen genannt) sind somit Unter- bzw. Obermengen gemeint, z. B. ist die Klasse der Kinder eine Unterklasse der Oberklasse Menschen oder die Klasse der Lebewesen eine Oberklasse zur Unterklasse der Menschen.

Gibt es von einer Klasse keine Instanzen (Objekte), die nicht tatsächlich Instanz einer Unterklasse sind, so spricht man von ABSTRAKTEN KLASSEN (z. B. gibt es kein Lebewesen, das nur Lebewesen ist, sondern es gibt Menschen, Pferde, Hunde, etc.). Die nicht abstrakten Klassen werden KONKRETE KLASSEN genannt. Im StarOffice können wir analog die abstrakte Dokument-Klasse betrachten. Tatsächlich gibt es nur Instanzen ihrer Unterklassen TextDocument, TableDocument etc., dieses sind daher konkrete Klassen.

Was sind Objekte?

Als Objekt bezeichnen wir die konkrete Instanz einer Klasse. So wäre in dem o. g. Vergleich Klaus Mustermann ein Objekt (als konkrete Instanz der Klasse Mensch). Oder analog wäre das Dokument "brief" ein Objekt (z. B. als konkrete Instanz der Klasse TextDocument).

Wie auch Menschen, so haben Objekte Namen. Die Namen müssen ebenfalls nicht eindeutig sein, analog: wie viele Helmut Schmidt es wohl gibt? Die meisten Objekte haben auch einen Parent. Der Parent ist das Objekt, in dem ein Objekt enthalten ist. Z. B. wäre nach dieser Definition der Parent eines Beines der dazugehörige Mensch, bzw. der Parent einer DocumentInfo das dazugehörige Dokument.

Was sind Eigenschaften und Methoden?

So wie jeder Mensch bestimmte Eigenschaften und Fähigkeiten hat, so auch jedes Objekt. Wir nennen diese hier Eigenschaften (Properties) und Methoden.

Eigenschaften

Eigenschaften können abgefragt werden (wie groß ist dieser Mensch, welche Textfarbe hat dieser Buchstabe), und die meisten können auch gesetzt (also geändert) werden (setze die Rahmenbreite auf 2 mm). Eigenschaften können direkt Werte enthalten (z. B. 2mm oder "Schmidt") oder Verweise auf Objekte. Objekte sind daher niemals direkt in den Eigenschaften enthalten.

Methoden

Methoden werden aufgerufen oder angewendet, ohne daß Sie im einzelnen wissen müssen, wie die Methoden funktionieren. Die Print-Methode zum Beispiel druckt das genannte Objekt, egal ob es ein Textdokument oder ein benannter Bereich einer Tabelle ist. Das Objekt kennt die Methode - Sie als Programmierer brauchen die Methode nur für das Objekt aufzurufen. Die Methoden können Werte zurückgeben, die Sie im Programm abfragen können (vergl. FUNCTION). So kann die Print-Methode zum Beispiel melden, ob der Druck erfolgreich war oder ob es zu einem Fehler gekommen ist.

Klassenhierarchie

Wie bereits erwähnt, werden alle Objekte mit gleichen Eigenschaften und Methoden zu einer Klasse zusammengefaßt. Diese Zusammenfassung ist nicht ausschließlich: jedes Objekt kann zu mehreren Klassen gehören.

Insbesondere können die Klassen hierarchisch geschachtelt sein, so daß zum Beispiel die Klasse Mensch eine Unterklasse der Klasse Säugetiere ist, diese eine Unterklasse der Wirbeltiere, die zu den Lebewesen gehören, usw. Dabei gilt, daß grundsätzlich die Merkmale der übergeordneten Klasse auf alle Unterklassen vererbt werden. Wenn ein Mensch zu den Säugetieren gehört, und diese zu den Wirbeltieren, dann hat der Mensch mithin auch Wirbel. Aber er hat auch spezielle Eigenschaften, die ihn von den anderen Säugetieren unterscheiden. Entsprechend kann zum Beispiel eine Texttabelle im StarWriter einige Eigenschaften und Methoden aufweisen, über die nur eine Texttabelle verfügt, während sie andere Eigenschaften, zum Beispiel die Eigenschaft, einen Namen zu haben, von den übergeordneten Klassen geerbt hat.

Wie werden Klassen, Objekte und Eigenschaften verbunden?

Wenn ein Mensch einen bestimmten Beruf ausübt, so kann man den Beruf als eine seiner Eigenschaften bezeichnen. Auch andere Menschen (Objekte) können dieselben Eigenschaften besitzen. Zu den Eigenschaften eines Windows gehört zum Beispiel DOCUMENT, über die das in dem Window dargestellte Dokument-Objekt erreichbar ist. Es gibt auch globale Eigenschaften, z. B. ACTIVEWINDOW, die tatsächlich Eigenschaften des Applikations-Standard-Basic-Objekts sind.

Wollen Sie zum Beispiel das aktive Fenster schließen, lautet der StarBasic-Befehl einfach ACTIVEWINDOW.CLOSE. Die Methode CLOSE, die für alle Objekte der Klasse WINDOW definiert ist, wird auf das Objekt, auf das im Moment aus ACTIVEWINDOW verwiesen wird, angewendet.

StarBasic arbeitet nicht allein in Ihrem Computer. Während Ihr StarBasic-Programm läuft, arbeiten viele andere Prozesse oder Threads parallel. Insbesondere kann der Anwender zum Beispiel in seinem StarOffice ein beliebiges Window anklicken und es so zum aktiven Window machen, während im Hintergrund Ihr Programm darauf vertraut, daß ein ganz anderes Window das ActiveWindow ist. Sprechen Sie also in gutem Programmierstil besser das gewünschte Window über eine Variable an, der Sie den Namen des gewünschten Fensters geben.

Die Eigenschaften ActiveWindow, ActiveDocument etc. sollten mit Vorsicht angewendet werden, da i. d. R. der Benutzer nicht daran gehindert ist, während des Programmablaufs das aktive Fenster (und damit auch das aktive Dokument) umzuschalten. Besser ist die Verwendung von eigenen Objektvariablen.

Objekthierarchie im StarOffice

Kernobjekte im StarBasic sind das Applikations-Standard-Basic-Objekt, die Dokument-Standard-Basic-Objekte und das StarOffice-Objekt (die Applikation an sich). Vom StarBasic aus wird man i. d. R. häufig Eigenschaften des Applikations-Standard-Basic-Objekts benutzen, um auf die benötigten Objekte zuzugreifen, da diese unqualifiziert verwendet werden können (unqualifiziert bedeutet: ohne ein Objekt zu benennen); von OLE-Automation beginnt jeder Zugriff mit dem Applikations-Objekt StarOffice.

Die wichtigsten globalen und Applikations-Eigenschaften

Jedes offene Dokument, Window und jedes Modul sind im Applikations-Standard-Basic-Objekt enthalten und können daher direkt mit ihren Namen angesprochen werden (z. B. StarWriter, [brief1] oder [unbenannt1:1]).

Außerdem stehen im Applikations-Standard-Basic-Objekt und im Applikationsobjekt StarOffice die folgenden wichtigen Eigenschaften zur Verfügung:

Application

Liefert das Applikationsobjekt StarOffice. Diese Eigenschaft ist insbesondere bei Verwendung von OLE-Automation interessant.

ActiveWindow

Das aktive Window (im StarBasic aggregiert (zusammengefaßt) mit der zum Dokument gehörenden View).

ActiveDocument

Das Dokument im aktiven Window (z. B. ein TextDocument).

ActiveModule

Das zum aktiven Dokument gehörige Modul (z. B. StarWriter).

Selection

Die Selektion im aktiven Window (Objekte verschiedenster Klassen).

Documents / Windows

Die Collections der geladenen Dokumente (Documents-Collection) bzw. der geöffneten Windows (Windows-Collection).

Die wichtigsten Window-Eigenschaften

Zu den verschiedenen anderen Objekten in StarOffice gibt es weitere Eigenschaften. Die Eigenschaften eines Objekts sind je nach Art des Objekts unterschiedlich. Ein OBJEKT der Klasse WINDOW verfügt zum Beispiel über folgende Eigenschaften:

Document

Das Dokument, das momentan in diesem Window dargestellt wird.

TopWindow

Falls das Window ein TopWindow ist, das Window selbst, sonst das umgebende TopWindow (FrameSets).

ActiveFrameWindow

Das Window welches innerhalb des angesprochenen Window aktiv ist (FrameSets).

View

Die zum dargestellten Dokument gehörende View ist über diese Eigenschaft erreichbar. Aus StarBasic liegt eine Aggregation vor, so daß Sie auf Methoden und Eigenschaften der View so zugreifen können, als wären es Methoden bzw. Eigenschaften des Window, extern (z. B. OLE-Automation) muß diese Eigenschaft zwischengeschaltet werden. View und Window sind nicht dasselbe; wäre das so, dann müßte es für jede View/Window-Kombination (SW, SC, ... / Preview, EditView / InPlace, PlugIn, MDI etc.) eine eigene Klasse geben.

Manche Dokumenttypen verfügen über verschiedene Views, z. B. eine Preview und eine Editview. Außerdem können selbstverständlich in mehreren Windows Views auf dasselbe Dokument existieren. Wichtig ist der Umstand, daß jede View ihre eigene Selektion hat.

Wenn Sie über den Menübefehl FENSTER - NEUES FENSTER ein weiteres Fenster öffnen, können Sie die Ansicht dort auf einen anderen Teil des Dokuments setzen. Das neue Window mit Namen [Doc1:2] hat also eine eigene View als Eigenschaft, die sich von der View des Windows [Doc1:1] unterscheiden kann.

Exkurs: Die Klasse aller Views hat den Namen VIEW. Sie ist so allgemein, daß Sie sie nicht zu nennen brauchen. Sie hat aber unter anderem eine Unterklasse BASEVIEW, die alle StarWriter-Views enthält. Dazu gehört das Objekt

mit Namen VIEW. Dieses Objekt hat unter anderem die Eigenschaft PRINT. Wollen Sie das Dokument drucken, müßten Sie in vollständiger Form folgendes schreiben: [Doc1:1].View.Print. Da aber die Print-Methode in allen aggregierten Objekten gesucht wird ("Aggregation"), genügt es, folgenden Befehl einzugeben: [Doc1:1].Print oder ActiveWindow.Print.

Selection

Die verschiedenen Arten von Selektions-Objekten haben wenig gemeinsam. Um zum Beispiel einem Text das Attribut FETT zu geben, muß der Text zuerst markiert werden. Dabei definiert der Textcursor, wenn er ohne Markierung in einem Wort steht, bei manchen Methoden das gesamte Wort als Selektion. Wollen Sie Text einfügen, so verwenden Sie die Methode INSERT, die für das Objekt TEXT der Klasse BASETEXT definiert ist. Der Befehl wäre Selection.Text.Insert. Auch hier können Sie aufgrund der Aggregation, wie oben bei den Views beschrieben, den Befehl verkürzen: Selection.Insert bewirkt dasselbe.

Welche Objekte gibt es?

Es gibt fast unendlich viele Objekte. Alles was Sie in StarOffice mit einem Namen versehen können oder was automatisch einen Namen erhält, kann als Objekt dienen, zum Beispiel ein Dokument, ein Window, ein selbst entworfener Dialog, ein Textrahmen, ein Zeichenrechteck, ein Bereich von Tabellenzellen, und so weiter.

Die Namen der vordefinierten Objekte sind meist nicht so wichtig, denn Sie sprechen jedes Objekt i. d. R. über Eigenschaften wie ActiveWindow an oder über selbst definierte Variable.

StarBasic ist modular

Sie programmieren in StarBasic in einzelnen Subs und Functions, die jeweils durch End Sub oder End Function abgeschlossen werden. In jeder Sub oder Function können andere Subs und Functions aufgerufen werden. Ein Makro ist die Schnittstelle zwischen Mensch und Ihrem Programm: der Anwender ruft das Makro auf, dieses ruft die erste zugehörige Sub auf. Der Aufruf des Makros erfolgt ereignisgesteuert. Wenn Sie mit dem Makrorekorder ein Makro aufzeichnen, wird damit eine Sub Prozedur erstellt, die Sie anschließend von Hand nachbearbeiten können.

Was ist eine Sub?

Sub ist die Kurzform von Subroutine, ein Unterprogramm also, das innerhalb eines Programms eine bestimmte Aufgabe übernimmt. Zerlegen Sie die Aufgabe, die Sie programmieren wollen, in einzelne Prozeduren, dann zerlegen Sie jede Prozedur weiter in die Schritte, die für diese Sub erforderlich sind, bis Sie das ganze Programm in übersichtliche Teilaufgaben aufgegliedert haben, die in Prozeduren und Unterprozeduren hierarchisch aufgeteilt sind.

Was ist eine Function?

Eine Function ist grundsätzlich eine Prozedur, die direkt einen Rückgabewert liefert. Sie können somit eine Function auch auf der rechten Seite einer Variablenzuweisung einsetzen, oder an anderen Stellen, an denen sonst Werte stehen.

Je allgemeiner eine Sub oder Function formuliert wurde, um so höher die Wahrscheinlichkeit, daß Sie sie in Zukunft auch für andere Programme nutzen können. Müssen Sie zum Beispiel die Eingabe in einem Textfeld in Ihrem Programm darauf überprüfen, ob sie mehr als 6 Zeichen enthält, so sollten Sie eine Function schreiben, der Sie den Grenzwert 6 als Parameter übergeben. Sie können die Function dann auch verwenden, wenn Sie einen Text darauf prüfen müssen, ob er zum Beispiel mehr als 20 Zeichen enthält.

Globale und lokale Variablen

Sie können in StarBasic Variablen deklarieren, die global für alle Subs gelten. Variablen, die Sie nicht ausdrücklich in einer Dim-Anweisung deklarieren, sind ebenfalls möglich; diese werden von StarBasic intern deklariert. Siehe auch Prozeduren und Funktionen.

Variablen, die Sie innerhalb einer Sub oder Function deklarieren, gelten nur innerhalb dieser Sub oder Function. Diese Variablen haben dabei Vorrang vor eventuell vorhandenen gleichnamigen Variablen in übergeordneten Subs oder vor globalen Variablen, die erst dann wieder gelten, wenn die betreffende Sub beendet wird. Übergeordnete oder globale Variablen gelten ansonsten im ganzen Programm.

Strukturieren der Programme

Nachdem Sie Ihr Programm in Prozeduren und Funktionen (Subs und Functions) aufgeteilt haben, wollen Sie vielleicht zusammengehörende Prozeduren und Funktionen in eigenen Dateien ablegen, um diese auch in anderen Projekten zur Verfügung zu haben. StarBasic unterstützt Sie hier durch die

Möglichkeit, Module und Bibliotheken anzulegen. Alle Subs und Functions stehen immer in Modulen, und Sie können fast beliebig viele Module verwenden. Die Module können Sie global anlegen oder einem bestimmten Dokument zuordnen. Mehrere Module können zu einer Bibliothek zusammengefaßt werden, und von den Bibliotheken kann es beliebig viele geben.

Sie können sowohl einzelne Subs und Functions als auch Module und ganze Bibliotheken von einer Datei in eine andere kopieren oder verschieben. Eine Beschreibung finden Sie bei der Erklärung des Dialogs Makro.

Das StarBasic-Konzept

Hier wird beschrieben, welches Konzept sich hinter StarBasic verbirgt. Sie erfahren insbesondere, wie das Zusammenspiel von Dokumenten, Dokumentvorlagen und Basic-Projekten funktioniert.

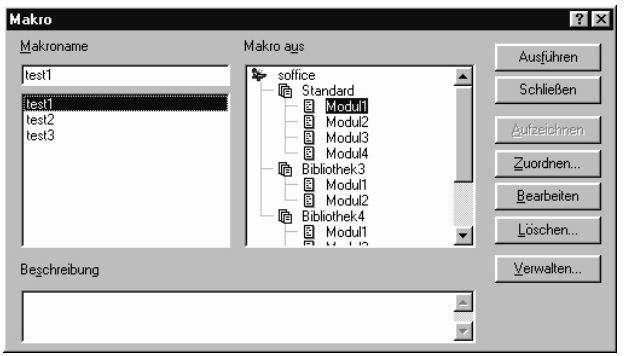
Aufzeichnen und Ausführen eines Makros

In dem Moment, in dem Sie ein Makro aus einer StarOffice-Applikation aufzeichnen, passiert folgendes: Der Makrorekorder beobachtet eigentlich gar nicht, welche Elemente der Oberfläche Sie bedienen, sondern lediglich, welchen Funktionsaufruf die Bedienung eines Elementes (Aufruf eines Pulldown-Menüs, Verlassen eines Dialogs mit bestimmten Einstellungen, usw.) zur Folge hat. Diese Funktionsaufrufe, die sich hinter der Bedienung der Elemente verbergen, "merkt" sich der Makrorekorder und macht daraus ein StarBasic-Programm. Beim Ablauf eines StarBasic-Programms werden diese gemerkten Funktionen nacheinander aufgerufen. Das Ergebnis ist identisch zu Ihrer manuellen Vorgehensweise.

Ein Beispiel anhand eines Textdokuments in StarOffice soll das verdeutlichen: Oft genug kommen Buchstabendreher bei der Erfassung von Texten vor, daß also zwei Buchstaben innerhalb eines Wortes "vertaushct" werden. In der Regel merkt man einen solchen Dreher sofort; das Korrigieren kostet aber Zeit. Der Cursor wird rückwärts bewegt, es werden beide Buchstaben gelöscht und neu geschrieben. Ein Makro könnte die Buchstaben tauschen. Dazu bräuchte es lediglich den einen Buchstaben auszuschneiden, den Cursor einen Schritt nach links zu bewegen und den ausgeschnittenen Buchstaben wieder an der neuen Position einzufügen.

Zeichnen wir ein solches Makro einmal auf:

- Öffnen Sie ein Textdokument und schreiben Sie direkt im Dokument ein Wort, das einen Buchstabendreher enthält (z. B. "Bildschrim"). Positionieren Sie den Cursor hinter dem letzten der beiden vertauschten Zeichen.
- Wählen Sie im Menü EXTRAS den Menüpunkt MAKRO. Sie sehen jetzt den folgenden Dialog:



- Geben Sie an dieser Stelle im Textfeld MAKRONAME den Namen "Test1" ein. In dem Moment, in dem Sie diesen Namen eingegeben haben, sehen Sie, daß im Listenfeld MAKRO AUS der Eintrag "Module1" selektiert wurde.
- Klicken Sie nun auf die Schaltfläche AUFZEICHNEN. Der obige Dialog verschwindet vom Bildschirm, und Sie sehen einen weiteren sehr kleinen Dialog - den Makrorekorder -, der Ihnen anzeigt, daß die Makroaufzeichnung aktiviert ist. Das bedeutet: StarOffice merkt sich von jetzt an alle Funktionen, die Sie über die verschiedenen Kontrollelemente (Pulldown-Menüs, Dialogelemente etc.) aufrufen - "schreibt" sie also quasi mit.
- Bewegen Sie jetzt den Cursor per Tastatur zwei Zeichen nach links.
- Halten Sie die Umschalttaste gedrückt und bewegen Sie den Cursor ein Zeichen nach rechts. Das erste "falsche" Zeichen ist damit markiert. Betätigen Sie die (Strg)(X)-Taste, um das Zeichen auszuschneiden.
- Betätigen Sie die Cursortaste Rechts und fügen Sie das Zeichen an der richtigen Position mit (Strg)(V)-Taste ein. Der Buchstabendreher ist korrigiert.
- Klicken Sie nun auf die STOP-Schaltfläche des Makrorekorders. Die Makroaufzeichnung ist damit abgeschlossen.

Das Ergebnis dieser Aufzeichnung wollen wir nun im folgenden betrachten:

- Wählen Sie wieder den Menüpunkt EXTRAS - MAKRO an.
- In dem Moment, in dem Sie im Listenfeld MAKRO AUS auf den Eintrag "Module1" unterhalb des Eintrags "soffice" klicken, zeigt StarOffice den Makronamen des Makros, das Sie gerade aufgezeichnet haben, im linken Listenfeld an. Da dieser Eintrag zunächst der einzige ist, ist er bereits selektiert. Sollten Sie bereits weitere Makronamen an dieser Stelle vorfinden, selektieren Sie bitte den Eintrag "Test1" durch Mausklick.
- Klicken Sie jetzt auf die Schaltfläche BEARBEITEN. StarOffice ruft jetzt die IDE auf, die StarBasic-Entwicklungsumgebung. Dort finden Sie unser aufgezeichnetes Makro als Programm wieder:

```
Sub Test1
   'Aufgezeichnetes Makro: Dienstag, 26. März 1996, 14:36
   Selection.GoLeft( 2, FALSE )
   Selection.GoRight( 1, TRUE )
   Selection.Cut()
   Selection.GoRight( 1, FALSE )
   Selection.Paste()
End Sub
```

Sie können die Vorgehensweise zum Aufbau eines Makros sehr schön erkennen: Aus dem aufgezeichneten Makro ist eine Basic-Unterroutine geworden, die genau die Aufrufe der Befehle enthält, welche sich ebenfalls hinter den Aktionen verbergen, die wir benutzt haben, während das Makro aufgezeichnet wurde. Die meisten der im Programm verwendeten Anweisungen und Funktionen entpuppen sich bei genauerem Hinsehen als Methoden eines bestimmten Objektes, welches in diesem Fall der Selektion (SELECTION) entspricht, die hier durch die Position des Cursors im Textdokument gegeben ist. Sobald eine Methode eines Objektes verwendet wird, beeinflußt das gleichzeitig eine Eigenschaft des Objektes: Wird in unserem Beispiel die GOLEFT-Methode auf die SELECTION angewendet, hat das eine Veränderung der Position der Schreibmarke zur Folge.

Mit diesem Aufbau von Makros sind gleich zwei Ansprüche erfüllt. Hinter nahezu allen Möglichkeiten, die eine StarOffice-Applikation bietet, stecken Basic-Äquivalente, die sich auf die hier vorgestellte Art und Weise aufzeichnen lassen. Dadurch, daß die meisten Befehle eines StarOffice-Programms auch im Basic enthalten sind, ergibt sich eine weitere Konsequenz: Fast alle Funktionen einer StarOffice-Applikation lassen sich durch Basic-Programme bedienen. Dieses Konzept erschließt dem Makro-Programmierer fast grenzenlose Welten: Die Programmierung maßgeschneideter Lösungen, natürlich mit Schwerpunkt auf der jeweiligen Stärke eines StarOffice-Bausteins Textverarbeitung, Tabellenkalkulation oder Präsentation und Gestaltung, wird

zum Kinderspiel. Programmieren Sie sich Ihre eigene Benutzeroberfläche für die Berechnung und Darstellung komplizierter mathematischer Vorgänge, entwickeln Sie eigene Redaktions- oder Autorensysteme mit den Textdokumenten des StarOffice oder automatisieren Sie eine Multimediapräsentation für Ausstellungen mit interaktiver Benutzersteuerung - dank StarBasic ist all dies machbar.

Elemente eines Makro-Projekts

Bisher haben wir lediglich einen kleinen Bestandteil dessen kennengelernt, was zu einem Makro-Projekt, oder nennen wir es lieber StarBasic-Projekt, gehört - eine einzelne Unterroutine.

Wie jeder Basic-Dialekt kennt auch StarBasic Unterroutinen und Unterfunktionen, die SUBs und FUNCTIONs als elementare Hilfe zum Aufbau strukturierter Programme, die alle unabhängig voneinander aufrufbar sind, sich aber auch gegenseitig aufrufen können.

SUBs und FUNCTIONs

Im Idealfall wird ein Basic-Projekt in verschiedene Teilprobleme zerlegt. Diese Teilprobleme werden abermals in Teilprobleme aufgegliedert etc. Dieser Vorgang geschieht solange, bis innerhalb von SUBs und FUNCTIONs die "Lösung" des kleinsten Problembausteins erfolgt. Erst hier finden sich spezielle Programmieralgorithmen und angewandtes Programmieren wieder. Ansonsten besteht ein Basic-Projekt im Idealfall nur aus dem Aufruf von SUBs und FUNCTIONs, durch die eine große Problemlösung in kleine Teile aufgeteilt wird.

Zeichnen Sie ein Makro in einer StarOffice-Applikation auf, wird das Ergebnis einer SUB zugeordnet, die als Funktionsnamen den Namen trägt, den Sie bei der Makroaufzeichnung bestimmt haben.

Bibliotheken und Module

SUBs und FUNCTIONs strukturieren ein Programm und machen es leichter lesbar - soviel haben wir bislang schon erfahren. Jedoch: In dem Moment, in dem Ihr Programmprojekt wirklich überdurchschnittlich umfangreich zu werden droht, kann selbst mit dieser Strukturierungstechnik die Übersicht noch verloren gehen, daher bietet StarBasic zwei weitere Strukturierungshilfen an: Die Anordnung von StarBasic-Unterroutinen in Modulen und in Bibliotheken.

Module können Bestandteile von Programmen - also SUBs und FUNCTIONs - aufnehmen und kathegorisieren. Mehrere Module selbst können wiederum in Bibliotheken zusammengefaßt werden. Diese Struktur können Sie sich anhand der obigen Abbildung verdeutlichen:

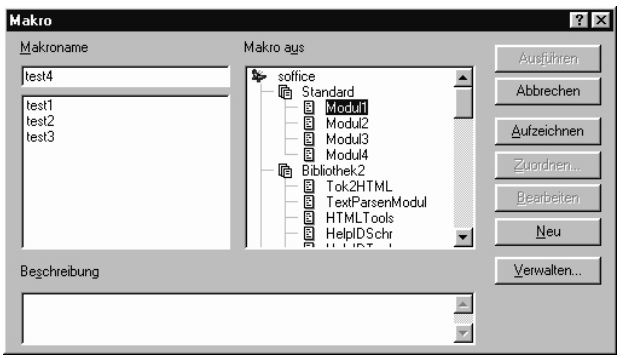
In der ersten Zeile des Listenfeldes MAKRO AUS erkennen Sie das Element, dem das gesamte StarBasic-Projekt zugeordnet ist, also StarOffice. Jedem Element, dem man ein StarBasic-Projekt zuordnen kann - wie im Beispiel der Applikation selbst, einem Dokument oder einer Dokumentvorlage - ist wenigstens eine Bibliothek zugeordnet. Standardmäßig gibt StarBasic den Bibliotheksnamen "Standard" dafür selbst vor und legt eine solche Bibliothek auch für jede Applikation oder für jedes neue Dokument oder jede neue Dokumentvorlage auch selbst an. Innerhalb dieser Bibliothek haben dann die verschiedenen Module Platz. Die Module selber schließlich nehmen die verschiedenen Unterprogramme, also die SUBs und FUNCTIONs, auf.

Makros organisieren und ausführen

Nun stellen sich Ihnen sicherlich folgende Fragen: Wie kann ich ein Makro aufzeichnen, das als SUB oder FUNCTION an einer "bestimmten Stelle" auftaucht? Wie kann ich ein bestimmtes Makro ausführen bzw. wo befindet sich überhaupt der für den Programmstart relevante Beginn eines StarBasic-Projektes?

So bestimmen Sie, welchem Element, welcher Bibliothek und welchem Modul ein Makro bei der Aufzeichnung zugeordnet wird

❍ Rufen Sie über das Menü EXTRAS - MAKRO den Dialog MAKRO auf. In diesem Dialog findet sich die Struktur der Makros wieder.



- Im Dialog erkennen Sie das Listenfeld MAKRO AUS. Vor der Aufzeichnung bestimmen Sie per Mausklick auf den entsprechenden Eintrag, welchem Modul innerhalb welcher Bibliothek und welchem übergeordneten Element (Applikation, Dokument oder Dokumentvorlage) das Makro zugeordnet werden soll.
- Möchten Sie, daß das Makro einem Modul eines Dokumentes oder einer Dokumentvorlage zugeordnet wird, müssen Sie die entsprechende Datei natürlich vorher öffnen.
- In der Liste erscheinen dann alle Dokumentennamen bzw. Dokumentvorlagennamen mit der vorgegebenen Bibliothek "Standard", die direkt die Unterprogramme, sprich: die aufgezeichneten Makros, aufnehmen kann.
- Nachdem Sie das gewünschte Element ausgewählt haben, bestimmen Sie im Textfeld den Namen des Makros und klicken anschließend auf die Schaltfläche AUFZEICHNEN.

So bestimmen Sie, welches Makro aus welchem übergeordneten Element abgespielt werden soll.

Genau das gleiche gilt für das Abspielen eines Makros: Hier bestimmen Sie im selben Dialog, welches Makro Sie abspielen lassen möchten. Sie wählen ebenfalls wieder per Mausklick aus der Liste MAKRO AUS das gewünschte Makro, klicken dann jedoch nicht auf die Schaltfläche AUFZEICHNEN sondern auf die Schaltfläche AUSFÜHREN.

Ein wenig anders verhält es sich, wenn Sie StarBasic-Projekte entwickeln, bei denen sich gezwungenermaßen Makroaufzeichnungen und "Eigenentwicklungen" vermischen. Sobald Sie ein eigenes Projekt entwickeln, legen Sie als erstes ein neues Modul an, das Ihr Projekt aufnimmt. In dem Moment, in dem Sie ein neues Modul anlegen, definiert StarBasic automatisch eine zunächst noch leere SUB namens MAIN, mit der das Programm startet, da sie die erste SUB in der IDE ist.

Besonderheiten bei Dokumentvorlagen

Dokumentvorlagen dienen in der Regel dazu, bestimmte Inhalte zu sichern, die beim Erstellen eines neuen Dokuments automatisch mit in das neue Dokument übernommen werden. Das gilt für Text-, Tabellen- und Präsentationsdokumente für den jeweils gültigen Dokumententyp. Haben Sie beispielsweise in einem Textdokument eine Dokumentvorlage angelegt, die den Text für einen Fax-Vordruck beinhaltet, und öffnen Sie ein neues Dokument auf deren Basis, wird diese Fax-Vorlage automatisch in das neue Dokument übernommen.

Bei allen drei Applikationen gilt das gleiche auch für Makros oder StarBasic-Projekte, die Sie innerhalb einer Dokumentvorlage zugeordnet haben. Erstellen Sie beispielsweise ein neues Dokument auf Basis einer Dokumentvorlage, die auch Makros enthält, werden Kopien dieser Makros oder StarBasic-Projekte als Kopie in Ihrem neuen Dokument angelegt.

Das erste Makroprogramm

In diesem Abschnitt stellen wir Ihnen die Makro-Programmierung an einem Beispiel vor. Wir entwickeln ein Programm, mit dessen Hilfe sich in beliebigen Dateien nach einer bestimmten Zeichenkette suchen läßt. Dabei erfahren Sie, wie Sie vorgehen müssen, um ein neues Makro-Projekt zu erstellen, den Programm-Code auszuformulieren, das Projekt um einen Dialog zu ergänzen, es an eine Dokumentvorlage zu koppeln und schließlich auszuführen.

Ein neues Makro anlegen

Hier zeigen wir Ihnen, wie Sie ein neues Makro anlegen. In unserem Beispiel soll das neue Makro-Programm an eine Dokumentvorlage gekoppelt werden. Das erste, was also zu tun ist, ist eine neue Dokumentvorlage zu erstellen, die dann das Makro aufnimmt. Der Vorteil dabei: Wann immer ein neues Dokument auf Basis dieser Vorlage erstellt wird, steht das entwickelte Makro ebenfalls zur Verfügung.

Um eine neue Dokumentvorlage zu erstellen, öffnen Sie ein neues Textdokument und speichern es als Dokumentvorlage, für unser Beispiel unter dem Namen "Maktest". Gehen Sie dazu folgendermaßen vor:

- Wählen Sie im Menü DATEI den Menüpunkt SPEICHERN UNTER.
- Bestimmen Sie im Kombinationsfeld DATEITYP "StarWriter 4.0 Vorlage/Template" durch Auswahl aus der Liste, und geben Sie im Textfeld DATEINAME den Namen "Maktest" ein.
- Bestätigen Sie den Dialog mit SPEICHERN, um das Dokument als Vorlage zu speichern.

Als nächstes gilt es, die Hauptroutine des Programms zu entwerfen. Da wir später das Programm unter einem bestimmten Namen als Makro aufrufen möchten, benötigen wir als allererstes eine Unterroutine, mit der das Programm gestartet wird. Diese Unterroutine, eine SUB, kann dann auch später als Makro angesprochen und z. B. in Symbolleisten oder als Eintrag eines Pulldown-Menüs konfiguriert werden. Das Makro soll den Namen "ErweiterteSuche" tragen, und genau dieser Name wird daher auch der Name der SUB sein. Legen wir also ein solches Makro an:

- Gehen Sie dazu im Menü EXTRAS auf den Eintrag MAKRO.
- Bestimmen Sie unter MAKRO AUS durch Mausklick auf den Eintrag "Standard" unterhalb des Eintrags "Maktest", daß das Makroprojekt in der Dokumentvorlage "Maktest" abgespeichert werden soll.
- Geben Sie nun den Makro-(SUB-)Namen "ErweiterteSuche" im Textfeld MAKRONAME ein, und rufen Sie die BASIC-IDE - die StarBasic-Entwicklerumgebung - durch die Schaltfläche NEU auf.

In dem Moment, in dem Sie eine Dokumentvorlage erstellt haben, die ein Makro oder Makro-Projekt aufnehmen soll, legt das jeweilige StarOffice-Programm automatisch eine Bibliothek mit dem Namen "Standard" an, die alle Dialoge oder weitere Module aufnimmt. SUBs oder FUNCTIONs werden grundsätzlich in Modulen gesichert. Wenn Sie also explizit kein Modul anlegen, legt das StarOffice-Programm automatisch ein Modul namens "Modul1" an, das die neue SUB aufnimmt.

Nun haben Sie die Vorbereitungen zu Ihrem neuen Makro getroffen und können als nächstes den eigentlichen Programm-Code eingeben.

So formulieren Sie den Code

Hier lernen Sie, den Programm-Code zu formulieren.

Nachdem Sie im ersten Arbeitsschritt ein neues Makro angelegt haben, sehen Sie, daß sich der Cursor im Editor befindet. Den Prozedurnamen "ErweiterteSuche" hat die IDE an dieser Stelle schon vorgegeben. Den eigentlichen Code können Sie nun an dieser Stelle formulieren.

Sie brauchen das Programm nicht abzutippen, sondern können es direkt aus der Online-Hilfe kopieren und in Ihren Basic-Editor einfügen.

Farbgebung während der Code-Eingabe

Sie werden bei der Eingabe der Code-Zeilen feststellen, daß bestimmte Komponenten einer Befehlszeile zur optischen Hilfestellung direkt eingefärbt werden. Der Editor verfährt dabei nach folgenden Richtlinien:

- Der Editor hat ein Basic-Schlüsselwort (wie Print oder Input) erkannt und färbt es blau ein.
- Der Editor hat einen Kommentar (eingeleitet durch REM oder " ' ") erkannt und färbt den gesamten Kommentar grau ein.
- Der Editor hat einen Ausdruck erkannt und färbt diesen grün ein.

❍ Der Editor hat eine Zeichenkette oder einen Wert erkannt und färbt diese rot ein.

Lange Zeilen in der IDE

Sie können mehrere Befehle in einer Zeile hintereinander schreiben, wenn Sie sie durch einen Doppelpunkt (:) voneinander trennen. Beachten Sie aber, daß ein Doppelpunkt, der direkt und ohne Leerzeichen auf einen Namen folgt, diesen Namen als Label definiert.

Wenn eine Codezeile besonders lang wird, können Sie sie in mehrere Textzeilen aufteilen. Geben Sie am Ende jeder Textzeile den Unterstrich "_" ein, dann wird die IDE beim Ausführen des Codes diese Zeile und die nächstfolgende Zeile wie eine einzige Zeile behandeln.

Listing des Beispielprogramms

```
Sub ErweiterteSuche
   dim DateiNamen$(1 to 100)    Speicher für die Dateinamen
   dim counter%       'Zähler, wieviele Dateien durchsucht
werden
   dim text$       'String, der den Suchtext aufnimmt
   dim z%          'Zähler für FOR/NEXT
   counter%=1
   do
      DateiNamen$(counter%)=InputBox("Bitte geben Sie den Namen
der Datei ein (* für Ende)","Erweiterte Suche","")
      if DateiNamen$(counter%)="*" then
         counter%=counter%-1
         exit do
      endif
      counter%=counter%+1
   loop until counter%=101
   text$=InputBox("Bitte Suchtext eingeben","Erweiterte
Suche","")
   for z%=1 to counter%
      Documents.Open(DateiNamen$(z%))
      anzahl%=-1
      do
         ActiveWindow.Search(Text$)
         anzahl%=anzahl%+1
      loop until Selection.Value=""
      print "Der Suchbegriff trat im Dokument
";DateiNamen$(z%);" ";anzahl%;" mal auf."
      ActiveDocument.Close False
   next z%
End Sub
```

Die Programmbeschreibung

Kommen wir zur Erklärung dieses kleinen Programms. Das Programm startet mit der SUB-Deklaration. In der Zeile "Sub..." wird der Name des Makros bzw. der Basic-Routine definiert. Anschließend werden im DIM-Block nach gewohnter Basic-Manier die erforderlichen Variablen deklariert, die im weiteren Programm erforderlich sind. Die entsprechenden Bedeutungen können Sie aus den beistehenden Kommentaren ersehen.

In der ersten DO...LOOP-Schleife liest das Programm mit Hilfe des INPUTBOX -Befehls nacheinander die Namen der Dokumente ein, die es auf ein bestimmtes Schlüsselwort zu untersuchen gilt. Beachten Sie, daß der bekannte INPUT-Befehl selber bei Programmentwicklungssystemen, die unter einer grafischen Benutzeroberfläche wie Windows, Xwindows oder OS/2 laufen, nur für Dateioperationen verwendet werden sollte.

Die eingelesenen Namen werden zunächst in einem String-Array gespeichert. Die Schleife, die dieses Einlesen übernimmt, wird erst dann beendet, wenn der Anwender ein Sternchen ("*") als Abbruchkriterium angibt.

Anschließend wird der Anwender aufgefordert, den Text einzugeben, nach dem die Dokumente durchsucht werden sollen. Dies geschieht mit dem entsprechenden INPUTBOX-Befehl, der den Text in der Variablen *Text$* aufnimmt.

Das eigentliche Problem löst das Makro im letzten Schleifenblock mit Hilfe einer FOR...NEXT-Schleife. Mit der entsprechenden Methode öffnet das Programm das Dokument (OPEN). Damit wird das geöffnete Dokument zum aktiven Fenster und kann über das ACTIVEWINDOW-Objekt angesprochen werden. Genau das geschieht bei der Suche des Suchtextes im aktiven Fenster mit Hilfe der SEARCH-Methode. Diese Methode macht nichts anderes, als wenn Sie in einem StarOffice-Programm im Menü BEARBEITEN die Funktion SUCHEN UND ERSETZEN aufrufen und anschließend durch OK nach dem zuvor eingegebenen Suchtext suchen lassen. Er sucht das durch den Zeichenfolgeparameter angegebene Wort im aktiven Dokument und selektiert dort die entsprechende Zeichenkette, sofern der Ausdruck gefunden wurde.

In unserem Falle sollen alle Dokumente durchsucht werden. Da uns die Namen der Dokumente bekannt sind, brauchen wir Sie lediglich in einer Schleife nacheinander zu öffnen (OPEN), zu untersuchen (DO...LOOP-Block) und dann wieder zu schließen (ACTIVEDOCUMENT.CLOSE).

Die Auswertung eines einzigen Dokumentes erfolgt über die DO...LOOP -Schleife. Der Suchvorgang wird durch SEARCH gestartet. War er erfolgreich, ist der gefundene Text im Dokument markiert. Jeder im Dokument markierte Text kann mit Hilfe des Selection-Objektes ermittelt werden. Auf diese Art

und Weise können wir feststellen, ob ein Suchergebnis zutraf, denn in diesem Fall ergibt eine Abfrage des markierten Textes mit dem Selection-Objekt keinen Leer-String. Folglich wird die DO...LOOP-Schleife solange durchlaufen, bis der Suchaufruf keine Markierung und damit auch kein Ergebnis (mehr) zur Folge hatte. Die Schleife wird dann durch die UNTIL-Anweisung verlassen. Solange jedoch Suchergebnisse aufgetreten sind, wird die Integervariable *Anzahl%* heraufgezählt und enthält damit später die Trefferquote der Suche. Diese Trefferquoten werden nacheinander für jede Datei durch den PRINT-Befehl auf dem Bildschirm angezeigt.

Wie Sie Ihr Makro starten, erfahren Sie im nächsten Abschnitt.

So starten Sie das Makro

Hier erfahren Sie, wie Sie ein Makro starten.

Sie haben jetzt die erste Version Ihres Programms als Source-(Quell-)Code vor Augen. Sie möchten als nächstes in den Genuß dieses Programms kommen. Um es zu starten, gibt es mehrere Möglichkeiten:

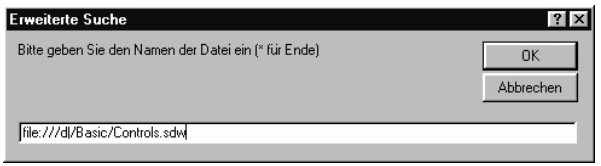
- Starten eines StarBasic-Programms aus einem StarOffice-Programm heraus:
  Sollten Sie sich nicht innerhalb der IDE befinden, starten Sie ein StarBasic-Programm, indem Sie im Menü EXTRAS - MAKRO das entsprechende Makro (und damit die entsprechende SUB) bestimmen und anschließend auf die Schaltfläche AUSFÜHREN klicken.
- Starten eines StarBasic-Programms aus der IDE:
  Wählen Sie das AUSFÜHREN-Symbol aus der Objektleiste aus. StarBasic kompiliert das Programm und startet es anschließend.

Kompilieren bedeutet, daß das Programm so für den Prozessor Ihres Computers aufbereitet wird, daß dieser die Befehle in "seiner Sprache" versteht, um Sie damit zum gewünschten Ergebnis zu bringen.

Das Programm wird dabei mit der Routine gestartet, die sich als "oberste" SUB im Source-Code befindet. In unserem Beispiel gibt es nur die SUB "ErweiterteSuche", die umgehend gestartet wird, sobald Sie das AUSFÜHREN -Symbol anklicken.

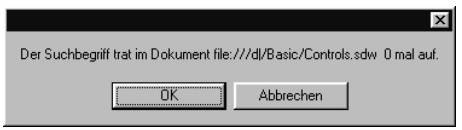
Sollten Sie den Programm-Code richtig übernommen haben und sich in der Basic-IDE befinden, passiert nach dem Programmstart folgendes:

Das Basic-Programm fragt Sie durch den INPUTBOX-Befehl nach den Dateinamen:



- Geben Sie hier nacheinander alle Dateinamen ein, und bestätigen Sie jede Eingabe mit OK. Haben Sie alle Dateinamen bestimmt, die untersucht werden sollen, geben Sie ein Sternchen ("*") ein.
- Geben Sie im nächsten Schritt den Suchbegriff ein, und bestätigen Sie auch diesen InputBox-Dialog mit OK.

Sie können jetzt am Bildschirm mitverfolgen, wie das Basic-Programm alle angegebenen Dateien nacheinander öffnet, und die Dokumente auf das Schlüsselwort überprüft. Anschließend bekommen Sie die Auswertung der Prozedur durch den Print-Befehl mitgeteilt:



Unterbrechen des Programms

StarBasic erlaubt Ihnen das Abbrechen eines Programms auf unterschiedliche Art und Weise:

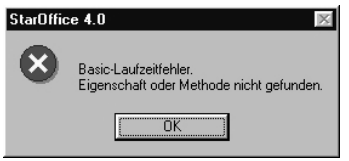
Grundsätzlich unterbrechen Sie den Programmablauf mit der Tastenkombination (Strg) (Alt) (S) oder (Strg) (Pause). Beim Abbrechen eines Programms durch (Strg) (Alt) (S) oder (Strg) (Pause) erhalten Sie grundsätzlich folgende "Fehlermeldung":

Haben Sie das Programm aus der IDE heraus gestartet, können Sie das Programm mit Hilfe des Anhalten-Symbols in der Objektleiste unterbrechen.

In einem Dialog, der beispielsweise durch Print oder InputBox auf den Bildschirm gebracht wurde, können Sie das Programm mit einem Klick auf Abbrechen oder mit der Tastenkombination (Strg) (Alt) (S) oder (Strg) (Pause) anhalten. Auch hierbei erhalten Sie die gezeigte Abbruch-Fehlermeldung.

Fehlermeldungen

Einen ähnlichen Dialog erhalten Sie, wenn StarBasic während des Kompilierens auf einen Syntaxfehler im Programmcode oder während der Programmausführung auf ein unbekanntes "Element" stößt. Das kann beispielsweise der Aufruf eines Befehls sein, der überhaupt nicht existiert, den Sie falsch geschrieben haben, oder Sie haben einen Fehler in der Anzahl der Parameter gemacht, die einem Befehl übergeben worden sind. In der Regel sagt Ihnen der StarBasic-Compiler im Klartext, was Sie falsch gemacht haben. Die entsprechende Zeile, in der der Fehler aufgetreten ist, wird zusätzlich im Programmcode kenntlich gemacht; die Behebung eines Fehlers wird dadurch erleichtert.



Ein Programm um einen Dialog ergänzen

Hier wird beschrieben, wie Sie Ihr Programm um einen Dialog ergänzen.

Unser Beispielprogramm hat bislang nur durch Dialoge mit dem Benutzer kommuniziert, die direkt durch Basic-Befehle wie PRINT oder INPUTBOX erzeugt worden sind. StarBasic kann aber noch eine ganze Menge mehr. Mit Hilfe von StarBasic können Sie, vergleichbar mit einem Zeichenprogramm, Ihre Dialoge entwerfen und so Ihren eigenen Programmen eine ergonomische und professionelle Benutzeroberfläche zuordnen.

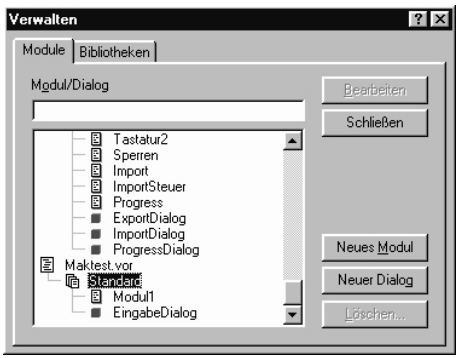
Wir wollen unser Beispielprogramm um einen Dialog erweitern. Ziel ist es, daß der Anwender das Verzeichnis bestimmt, in dem die Dokumentdateien vorhanden sind, und er anschließend eine Auflistung aller dort vorhandenen "*.sdw"-Dateien in einem Listenfeld angezeigt bekommt. Des weiteren kann er diese Dateien auswählen und in einer weiteren Listbox zusammenstellen.

So ergänzen Sie ein Projekt um einen Dialog

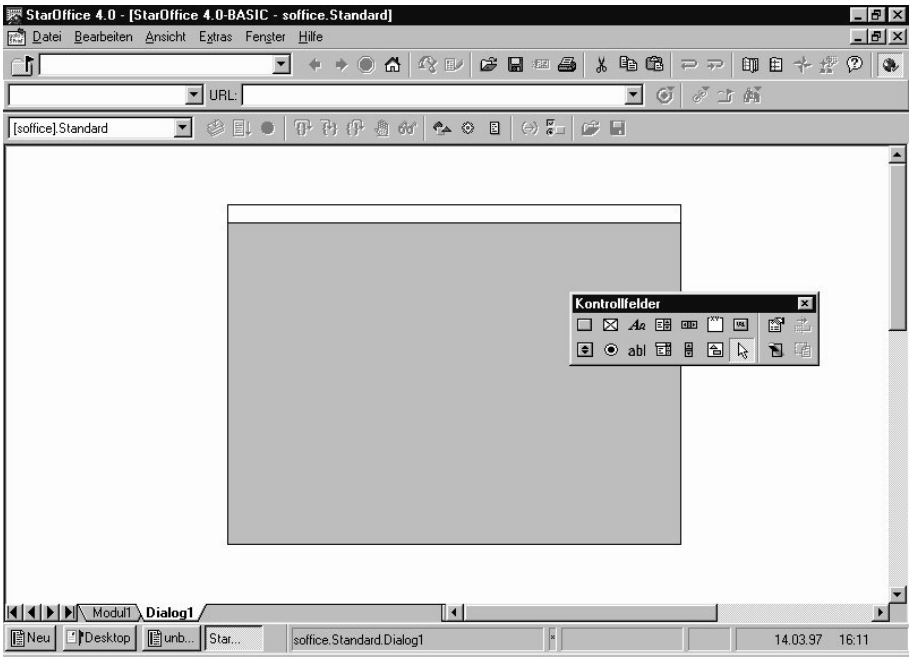
Als erstes muß dem Projekt ein zunächst noch "leerer" Dialog hinzugefügt werden. Dazu verfahren Sie wie folgt:

❍ Falls Sie sich nicht mehr in der IDE befinden, wechseln Sie dort hinein, indem Sie das Menü EXTRAS - MAKRO wählen.

- Klicken Sie im Dialog unterhalb des Eintrags "Maktest" auf den Eintrag "Modul1", anschließend auf den Makronamen "Erweiterte Suche" und schließlich auf die Schaltfläche BEARBEITEN. Sie befinden sich anschließend in der IDE und können den bisherigen Programm-Code des Beispielprogramms erkennen.
- Wählen Sie in der IDE das MODULE-Symbol aus. Daraufhin erscheint ein Dialog, mit dessen Hilfe Sie einen neuen Dialog anlegen können:



- Klicken Sie nun auf die Bibliothek, in der Sie einen Dialog hinzufügen möchten. Das ist für unser Beispiel die Bibliothek "Standard", die zum Dokument "Maktest" gehört.
- Klicken Sie in diesem Dialog auf die Schaltfläche NEUER DIALOG. Die IDE hat der Bibliothek nun einen Dialog namens "Dialog1" hinzugefügt.
- Um den Namen dieses Dialogs zu ändern, halten Sie die (Alt)-Taste gedrückt und klicken Sie auf den Namen "Dialog1". Sie können den Namen des Dialogs jetzt editieren.
- Geben Sie den neuen Namen des Dialogs ein; für unser Beispiel wählen Sie den Dialognamen "EingabeDialog". Bestätigen Sie dieses anschließend.
- Klicken Sie anschließend auf BEARBEITEN, so wechseln Sie dadurch in den Dialogeditor der IDE.



Was Sie an dieser Stelle erkennen können, wirkt im Moment noch vergleichsweise "karg". Sie sehen in der Mitte des Bildschirms lediglich den momentan noch leeren Dialog. Es fehlen die Hilfsmittel, mit denen Sie auf der einen Seite die Elemente des Dialogs zusammenstellen und auf der anderen Seite die speziellen Eigenschaften der in dem Dialog plazierten Elemente definieren können. Diese Elemente befinden sich in der Abreißleiste KONTROLLFELDER, die Sie in der Objektleiste des Dialogeditors finden und nach Belieben auf dem Bildschirm plazieren können.

So plazieren Sie ein Bedienungselement in einem Dialog

- Klicken Sie für unser Beispiel auf das TEXTBOX-Symbol in der Abreißleiste KONTROLLFELDER. Der Mauszeiger nimmt die Gestalt eines kleinen Fadenkreuzes an.
- Im Dialog ziehen Sie nun das Textfeld mit der Gummibandfunktion auf. Plazieren Sie es so, daß es in der linken, oberen Ecke steht.

Damit haben Sie das erste Bedienungselement im Dialog plaziert.

Komplettieren des Dialogs

Auf die eben beschriebene Art und Weise plazieren Sie alle weiteren Dialogelemente im Dialog. Wir wollen zunächst alle benötigten Elemente in den Dialog bringen, und sie erst im nächsten Schritt richtig sortieren.

- Wählen Sie aus der Abreißleiste das CommandButton-Symbol an. Ziehen Sie damit fünf Schaltflächen im Dialog auf.
- Wählen Sie aus der Abreißleiste das ListBox-Symbol aus, und ziehen Sie damit zwei Listenfelder im Dialog auf.
- Wählen Sie aus der Abreißleiste das Label-Symbol aus, und ziehen Sie damit ein Beschriftungsfeld im Dialog oberhalb des Eingabefeldes auf.

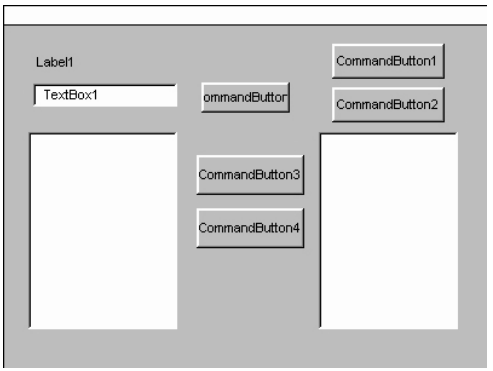
So verschieben, vergrößern und verkleinern Sie ein Dialogelement

Im nächsten Schritt werden wir die Elemente so anordnen, daß der Dialog "aufgeräumt" ist. Wählen Sie dazu in der Abreißleiste das Selektieren-Symbol aus.

- Um ein Dialogelement zu verschieben, klicken Sie mit dem Mauszeiger auf den Rand, halten Sie die Maustaste gedrückt und verschieben Sie so das Element an die gewünschte Position.
- Um ein Dialogelement zu vergrößern oder zu verkleinern, klicken Sie mit dem Mauszeiger an einen beliebigen Punkt im Element. Sie können erkennen, daß das entsprechende Element mit acht Kontrollpunkten ausgestattet ist, mit deren Hilfe Sie es in alle Richtungen vergrößern und verkleinern, dehnen oder stauchen können.

> Falls Ihnen der Platz innerhalb des Dialogs nicht ausreicht, können Sie den Dialog selbst vergrößern, verkleinern oder umpositionieren. Klikken Sie dazu auf den Rand des Dialogs und verfahren Sie, wie in den vorherigen beiden Anleitungspunkten beschrieben.

- Ordnen Sie die Elemente für unser Beispiel so an, daß Sie folgendes Bild sehen:



Sie erkennen, daß der Text in der Schaltfläche neben dem Eingabefeld nicht richtig in das Feld paßt. Diese Tatsache können Sie immer dann ignorieren, wenn Sie im Vorfeld wissen, daß der finale Text wesentlich kürzer als der Vorgabetext ist.

Aus der Anordnung der Dialogelemente, wie sie in der Grafik dargestellt ist, können Sie schon ungefähr erkennen, wozu der Dialog dienen soll. Im oberen Eingabefeld können Sie den Pfad bestimmen, in dem das Programm nach "*.sdw"-Dateien suchen soll. Klicken Sie anschließend auf das darunterstehende Listenfeld, so werden dort alle Dateien aufgelistet, die in dem angegebenen Verzeichnis vorhanden sind.

Klicken Sie auf die obere Schaltfläche, die sich zwischen den Listenfeldern befindet, soll eine zuvor ausgewählte Datei dem zweiten Listenfeld hinzugefügt werden; wenn Sie auf die untere Schaltfläche klicken, wird eine angegebene Datei wieder entfernt. Wählen Sie die Schaltfläche an, die sich rechts oben befindet, startet das eigentliche Programm und durchsucht alle Dokumente, die im rechten Listenfeld aufgelistet sind.

Was aber noch fehlt, ist die korrekte Beschriftung der einzelnen Elemente. Dazu müssen Sie den Dialogelementen Eigenschaften zuweisen.

Eigenschaften definieren

Hier erfahren Sie, wie Sie Aussehen und Eigenschaften eines Dialogs und der dazugehörigen Dialogelemente festlegen.

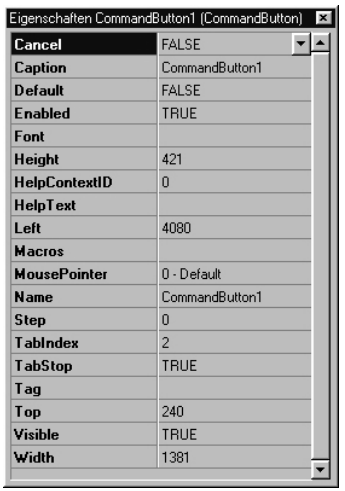
Farben, Bezeichnungen, Größe und Verhalten von Dialogelementen unter bestimmten Situationen werden als Eigenschaften bezeichnet. Diese Eigenschaften lassen sich sowohl während der Konstruktion eines Dialoges als

auch zur Laufzeit des Programms verändern. Einige Eigenschaften lassen sich dabei nur während der Laufzeit, andere nur während der Dialogerstellung definieren.

Definition von Eigenschaften während der Dialogerstellung

Um die Eigenschaften von Elementen während der Dialogerstellung festzulegen, bietet uns der Dialogeditor ein besonderes Hilfsmittel an, das Eigenschaften-Fenster. Probieren wir das an unserem Beispiel aus:

❍ Klicken Sie auf die Schaltfläche, dessen Eigenschaft Sie festlegen möchten, und anschließend auf das Symbol EIGENSCHAFTEN in der Abreißleiste KONTROLLFELDER. Das Eigenschaftenfenster öffnet sich.



Dieses Eigenschaftenfenster enthält sämtliche Eigenschaften, die Sie für das zuvor selektierte Element bestimmen können. Sie finden dort auch die für uns zunächst wichtigen Eigenschaften NAME und CAPTION. Mit der NAME-Eigenschaft bestimmen Sie den Namen des Elementes. Das ist der Name, mit dessen Hilfe Sie das Element später, zur Laufzeit des Programms, aufrufen können.

Die CAPTION-Eigenschaft dient zur Beschriftung des Elements. Beide Eigenschaften haben nichts miteinander zu tun, auch wenn der Dialogeditor beiden Eigenschaften die gleiche Voreinstellung zuweist. Die CAPTION -Eigenschaft können Sie daher auch nur für die Elemente bestimmen, die irgendeine dokumentierende Aufgabe haben, z. B. die Beschriftung von Schaltflächen.

Wir wollen im folgenden die Namen und die Beschriftungen der Dialogelemente ändern:

❍ Beginnen wir mit der zukünftigen OK-Schaltfläche, die sich in unserem Beispiel rechts oben im Dialog befindet. Klicken Sie dieses Element zunächst einmal an, um ihm eine Eigenschaft zuzuweisen.

- Klicken Sie jetzt in das Feld im Eigenschaftenfenster, das sich neben der Eigenschaft CAPTION befindet. Die Eigenschaft CAPTION wird farbig unterlegt. Klicken Sie abermals in dieses Feld, erscheint ein Cursor, und Sie können den vorhandenen Text löschen und dann die Zeichenfolge "OK" eingeben. Anschließend bestätigen Sie Ihre Eingabe.
- Selektieren Sie auf die gleiche Art und Weise die Eigenschaft NAME und ersetzen Sie den vorhandenen Text ebenfalls durch "OK".
- Klicken Sie im Dialog auf die darunterstehende Schaltfläche, und bestimmen Sie sowohl als CAPTION- als auch als NAME-Eigenschaft "Abbruch".
- Klicken Sie im Dialog auf die obere Schaltfläche, die sich zwischen den Listenfeldern befindet, und bestimmen Sie sie als CAPTION-Eigenschaft ">>dazu" und als NAME-Eigenschaft "dazu".
- Klicken Sie im Dialog auf die darunterstehende Schaltfläche, und bestimmen Sie "<<löschen" als CAPTION und "loeschen" als NAME.
- Benennen Sie als NAME- und CAPTION-Eigenschaft die letzte Schaltfläche neben der Eingabezeile als "Aktualisieren".

> Für die Namensgebung von Dialogelementen gelten die gleichen Konventionen wie für Variablennamen. Sie dürfen keine Umlaute, keine Leerzeichen oder Satz- und Sonderzeichen enthalten!

- Klicken Sie jetzt auf das Eingabefeld. Sie werden im Eigenschaftenfenster jetzt vergeblich nach einer CAPTION-Eigenschaft suchen, denn: Das Eingabefeld hat schlicht und ergreifend keine CAPTION-Eigenschaft, denn es gibt bei einem Eingabefeld nichts zu beschriften. Das gleiche gilt im übrigen für Listenfelder. Aus diesem Grund bestimmen wir für die übrigen Dialogelemente nur noch die NAME-Eigenschaft. Bestimmen Sie als NAME-Eigenschaft für das Eingabefeld den Text "Pfadname".
- Bestimmen Sie für die linke Listbox "Dateiliste" als NAME-Eigenschaft.
- Bestimmen Sie für die rechte Listbox "Dokumentenliste" als NAME-Eigenschaft.

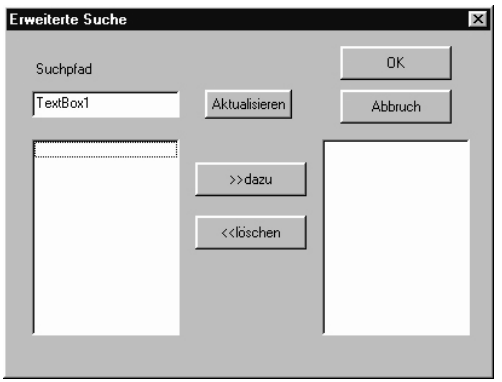
Die Definition der Eigenschaften ist damit abgeschlossen.

Natürlich gibt es für jedes Element eine Vielzahl von Eigenschaften, die an dieser Stelle nicht alle erklärt werden sollen. Bedienen Sie sich der Dialogreferenz, in der Sie alle Eigenschaften der existierenden Dialogelemente ausführlich beschrieben finden.

So bestimmen Sie die Titelzeile des Dialogs

Wie fast für jedes Dialogelement, können Sie auch hier die Beschriftung des Dialogs selbst bestimmen. In diesem Fall wird die Beschriftung in der Titelzeile positioniert. Um den Titel des Dialogs zu definieren, brauchen Sie lediglich den Dialog zu selektieren (einfach auf den Rand des Dialogs klicken) und die CAPTION-Eigenschaft zu bestimmen.

❍ Bestimmen Sie den Text "Erweiterte Suche" als Dialog-CAPTION-Eigenschaft.



So testen Sie den Dialog

Der Dialogeditor bietet Ihnen die Möglichkeit, Ihren Dialog praxisnah zu testen. In der Abreißleiste finden Sie dazu das Symbol TESTMODUS AKTIVIEREN. Klicken Sie auf dieses Symbol, wechselt der Dialogeditor in den Testmodus, und Sie können zur Probe Eingaben vornehmen oder Schaltflächen anklikken, um festzustellen, ob die Anordnung der Elemente tatsächlich der entspricht, die Sie geplant hatten. Sie haben ebenfalls die Möglichkeit, mit der (Tab)-Taste den Fokus auf die verschiedenen Dialogelemente zu setzen, so wie Sie es auch von Dialogen unter Ihrem Betriebssystem gewohnt sind.

Sollten Sie Dialog-Elementen bereits Ereignisse zugewiesen haben, so werden diese auch im Test-Modus aufgerufen, wenn Sie die entsprechenden Elemente des Dialogs bedienen.

So bestimmen Sie den TAB-Index

Dabei werden Sie möglicherweise feststellen, daß, wenn Sie die (Tab)-Taste gedrückt haben, die Reihenfolge der Elemente, die jeweils als nächstes angesprungen werden, nicht korrekt ist. Der Fokus scheint zufällig auf das näch-

ste Element gesetzt zu werden. In Wirklichkeit verhält es sich jedoch so, daß sich die Fokussier-Reihenfolge aus der Reihenfolge des Plazierens der Elemente ergibt. Mit anderen Worten: In der Reihenfolge, in der Sie die Elemente im Dialog plaziert haben, werden Sie auch mit der (Tab)-Taste nacheinander angesteuert. Für die meisten Anwendungen ist das jedoch nicht wünschenswert.

Aus diesem Grund gibt es für jedes Kontrollelement eine TabIndex-Eigenschaft. Diese Eigenschaft beinhaltet einen Wert, der die Reihenfolge des Ansteuerns durch die (Tab)-Taste bestimmt. Um diese Eigenschaft zu definieren, verfahren Sie wie folgt:

- Verlassen Sie den Testmodus zunächst durch (Esc).
- Klicken Sie zunächst das "erste" Element der Tab-Index-Reihenfolge an. Dieses Element ist auch gleichzeitig das Element, das standardmäßig fokussiert ist, wenn der Dialog aufgerufen wird. In unserem Beispiel sollte das das Eingabefeld sein.
- Klicken Sie im Eigenschaftenfenster auf die Eigenschaft TabIndex und tragen Sie dort den Wert "0" ein.
- Vergeben Sie für die linke Listbox den Wert "1", für >>dazu den Wert "2", für <<löschen den Wert "3", für die rechte Listbox den Wert "4" und für OK den Wert "5".

Schalten Sie nun zurück in den Testmodus, können Sie die Änderungen feststellen, indem Sie mit (Tab) zum jeweils nächsten Element springen - dieses Mal in der richtigen Reihenfolge.

Aufruf eines Dialogs vom Programmcode aus

Hier erfahren Sie, wie Sie einen Dialog von einem Programm heraus aufrufen.

Nachdem Sie Ihr Programm um einen Dialog ergänzt haben, möchten Sie diesen Dialog nun in Ihren Programmcode einbinden. Sie befinden sich noch im Dialogeditor und müssen zunächst wieder in die StarBasic-IDE gelangen.

So wechseln Sie zwischen Modulen und Dialogen

Sie können an der unteren Seite des Dialog-Editierfensters zwei Registerzungen erkennen, mit deren Hilfe Sie im Projekt zwischen dem Dialog und dem Modul hin- und herspringen können. Sollten Sie in umfangreicheren Projekten mehrere Module oder Dialoge verwenden, werden dort natürlich entsprechend mehr Zungen angezeigt, mit denen Sie gezielt zu einem Modul

oder einem Dialog springen können. Klicken Sie für unser Beispiel auf Modul1; Sie befinden sich jetzt wieder in der StarBasic-IDE.

Einbindung des Dialogaufrufes in das Beispiel-Programm

Ergänzen Sie in unserem Beispiel-Programm die SUB "ErweiterteSuche" unterhalb des Dim-Blocks um die Befehlszeile:

```
EingabeDialog.Show
```

Starten Sie jetzt das Programm, erscheint als allererstes unser Dialog. Schließen Sie diesen Dialog mit dem Schließfeld in der Titelleiste, fährt das Programm, wie bekannt, mit der ganz normalen Abarbeitung fort.

Methoden

Hier erfahren Sie, wie Sie Methoden anwenden. Auf jedes in StarBasic bekannte Objekt sind verschiedene Methoden anwendbar, die irgendeine Auswirkung auf das Objekt haben.

Die Befehlszeile:

```
EingabeDialog.Show
```

wendet die Methode Show auf das Objekt "EingabeDialog" an. Das ist der Dialog, den Sie im Rahmen dieser StarBasic-Einführung zu unserem Beispiel-Programm hinzugefügt haben. Mit Hilfe der Show-Methode wird das Objekt auf dem Bildschirm dargestellt.

Die Befehlszeile:

```
ActiveWindow.Search("Suchtext")
```

bewirkt, daß in dem Objekt Window (z. B. eine zuvor geladene Ansicht eines Dokumentes) nach einer bestimmten Zeichenkette gesucht wird.

An dieser Stelle werden wir nicht im Detail auf alle möglichen Objektmethoden eingehen. Die Referenz beschreibt Ihnen hierzu die genauen Details.

Ereignisse

Hier lernen Sie, mit Ereignissen zu arbeiten. Ihr Betriebssystem ist so ausgelegt, daß immer, wenn ein Benutzer ein Element der Benutzeroberfläche angewählt oder bestimmte Vorgänge mit diesem Element durchgeführt hat, eine bestimmte Aktion ausgelöst wird. Mit anderen Worten: Alle Programme, die ein Anwender verwendet, werden durch Ereignisse gesteuert, die durch den Anwender ausgelöst werden.

In StarBasic ist das nicht anders. Jedes Element, das Sie in einem Dialog plazieren, kann unterschiedliche Ereignisse ausführen. Sie selber als Programmierer haben lediglich dafür zu sorgen, daß eine bestimmte SUB existiert, die dann durchlaufen wird, wenn der Anwender während des Programmablaufes das entsprechende Element auf irgendeine Art und Weise "manipuliert" hat.

Das wichtigste Element ist das OnClick-Ereignis. Klickt der Anwender auf eine Schaltfläche, löst das Betriebssystem dieses Ereignis automatisch aus. StarBasic überprüft jetzt, ob Sie eine SUB mit diesem Ereignis verknüpft haben. Findet StarBasic eine derartige Verknüpfung, sorgt es dafür, daß diese SUB automatisch aufgerufen wird - Sie selber brauchen sich um diesen Ablauf überhaupt nicht zu kümmern.

So verknüpfen Sie ein Makro mit einem Ereignis

Die Vorgehensweise zur Verknüpfung eines Makros oder einer SUB mit einem Ereignis ist denkbar einfach. Dies möchten wir Ihnen hier anhand unseres Beispiels zeigen.

Zunächst bestimmen wir das Makro, das wir mit einem Ereignis verknüpfen möchten:

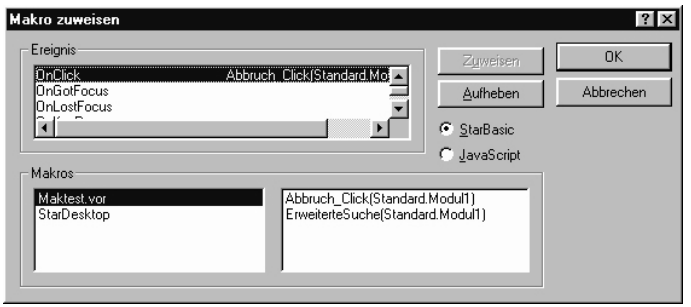
- Schalten Sie zurück in die IDE, indem Sie die Registerzunge Modul1 anklicken. Ergänzen Sie das Programm um folgende SUB:

```
Sub Abbruch_Click
   EingabeDialog.Hide
End Sub
```

Die SUB "Abbruch_Click" bewirkt, daß der "EingabeDialog" geschlossen wird, und soll nun dem Ereignis "Klick auf die Schaltfläche Abbruch" zugeordnet werden:

- Wechseln Sie mit der entsprechenden Registerzunge an der Unterseite des IDE-Fensters zurück in den Dialogeditor.
- Falls das Eigenschaftenfenster nicht mehr aktiviert ist, rufen Sie es mit dem Eigenschaften-Symbol der Abreißleiste Kontrollfelder auf.
- Klicken Sie jetzt im Dialog das Element an, dem Sie ein Ereignis hinzufügen möchten. Wählen Sie für unser Beispiel die Abbruch-Schaltfläche an.
- Klicken Sie im Eigenschaftenfenster in das Feld Macros und anschließend auf die Schaltfläche (...) im zugehörigen rechten Feld. Sie öffnen den Dialog Makro zuweisen.

- Wir möchten definieren, daß die SUB "Abbruch_Click" durchlaufen wird, sobald der Anwender auf die gewählte Schaltfläche klickt. Das dafür zuständige Ereignis ist OnClick. Wählen Sie zu diesem Ereignis das entsprechende Makro aus und klicken Sie auf Zuweisen. Beenden Sie den Dialog mit OK.



Starten Sie jetzt unser Beispiel-Programm "Erweiterte Suche", erscheint der zugehörige Dialog. Sobald Sie auf Abbruch klicken, wird der Dialog beendet. In diesem Moment ruft StarBasic automatisch die SUB "Abbruch_Click" auf. Die Hide-Methode sorgt dafür, daß der Dialog geschlossen wird. Da der Dialog in unserem Beispiel mit der SUB "ErweiterteSuche" auf den Bildschirm gebracht wurde, fährt das Programm, nachdem es "Abbruch_Click" durchlaufen hat, mit dem nächsten Befehl der SUB "ErweiterteSuche" fort. In unserem Beispiel gibt es keinen weiteren Befehl außer End Sub, so daß das Programm mit dieser Anweisung zu Ende ist.

Wichtige Informationen zu Ereignissen

- Die SUBs, die mit bestimmten Ereignissen verknüpft werden, sollten in derselben Bibliothek wie der Dialog stehen.
- Wird für ein Ereignis eine SUB definiert, die in keinem Modul existiert, gibt StarBasic eine entsprechende Fehlermeldung aus.
- Eine FUNCTION darf kein Ereignis verarbeiten.

Grundsätzlich gibt es keine Konventionen zur Namensvergabe von Ereignissen, außer denen, die auch für "normale" SUBs gelten. Sollten Sie jedoch umfangreiche Projekte mit vielen Dialogen entwickeln, verwenden Sie vorsorglich die Konvention
Elementname_Ereignisname
zur Namensvergebung, damit Sie schon im Programmtext erkennen können, ob es sich bei einer bestimmten SUB um eine Ereignis-SUB oder eine "ganz normale" SUB handelt. In unserem Beispiel haben wir diese Konvention mit dem SUB-Namen "Abbruch_Click" ebenfalls beachtet.

Dialog-Ereignisse

Daß ein Dialog selbst bestimmte Eigenschaften haben kann, haben wir mit der CAPTION-Eigenschaft schon kennengelernt. Dialoge selbst können ebenfalls bestimmte Ereignisse auslösen. Möchten Sie ein Dialog-Ereignis definieren, klicken Sie im Dialogeditor auf den Rand des Dialoges. Im Ereignisfenster werden in dem Moment alle für den Dialog möglichen Ereignistypen aufgelistet, die auf bekannte Art und Weise bestimmt werden können.

Die wichtigsten Ereignistypen

Hier werden die wichtigsten Ereignistypen vorgestellt.

In der Referenz finden Sie die möglichen Ereignisse aller Dialogelemente und auch die Ereignisse eines Dialogs selbst ausführlich beschrieben. Wir möchten an dieser Stelle gesondert auf die wichtigsten Ereignistypen aufmerksam machen.

OnClick, OnSelect- und OnDblClick-Ereignis eines Elements

Das ONCLICK- oder das ONSELECT-Ereignis wird ausgelöst, wenn der Anwender auf ein bestimmtes Element eines Dialoges, wie z. B. eine Schaltfläche oder ein Listenfeld, klickt.

Das ONDBLCLICK–Ereignis wird ausgelöst, wenn der Anwender auf ein Dialogelement, wie z. B. ein Textfeld, einen Doppelklick ausführt.

Load- und Activate-Ereignis eines Dialogs

Das LOAD-Ereignis tritt immer dann ein, wenn der Dialog das erste Mal angezeigt wird. Innerhalb des LOAD-Ereignisses haben Sie die Möglichkeit, den Dialog initialisierende Schritte zu unternehmen. Sollen beispielsweise Ein-

gabefelder mit bestimmten Texten vom Programm-Code aus vorbelegt werden, können Sie die SUB, die das vornimmt, als LOAD-Ereignis des Dialogs definieren.

Das ACTIVATE-Ereignis wird dann ausgelöst, wenn der Dialog durch den Anwender in den Vordergrund geholt wird. Das ist beispielsweise dann der Fall, wenn der Anwender von einem anderen Fenster durch Mausklick in den Dialog wechselt oder den Dialog zuvor auf Symbolgröße verkleinert hat und ihn dann wieder auf Normalgröße bringt.

Nachdem Sie einen Dialog aufgerufen haben, bleibt das Programm zunächst an der Stelle stehen, an der der Dialog aufgerufen wird, also bei der Show-Methode. Sämtliche weiteren Manipulationen am Dialog können jetzt nur noch über die verschiedenen Ereignisse umgesetzt werden. Möchten Sie bestimmte Dialogeigenschaften bestimmen oder verändern, bevor der Dialog dargestellt wird (denn danach geht es außer durch die Ereignisprozeduren nicht mehr), müssen Sie den Dialog zuvor mit der Load-Methode laden, seine Eigenschaften verändern und ihn erst anschließend mit der Show-Methode darstellen. Hierzu ein Beispiel:

```
Sub Beispiel
  EingabeDialog.Load
  erst jetzt können Eigenschaften am Dialog verändert werden
  EingabeDialog.PfadName.Text="C:\*.TXT"
  EingabeDialog.Show
End Sub
```

Laufzeiteigenschaften definieren und abfragen

Hier erfahren Sie, wie Sie Laufzeiteigenschaften definieren und abfragen.

Bestimmte Eigenschaften, die man Dialogelementen während der Dialogerstellung zuweisen kann, haben wir schon kennengelernt. StarBasic ermöglicht Ihnen aber ebenfalls, bestimmte Eigenschaften auch innerhalb eines StarBasic-Programms mit Hilfe entsprechender Befehlszeilen zu ändern oder zu definieren. Die Syntax dafür ist die folgende:

```
Objektname.Eigenschaft [=Ausdruck]
```

Möchten Sie beispielsweise den vordefinierten Text des Eingabefeldes namens Pfadname mit dem Text "C:*.TXT" belegen, verwenden Sie dazu folgende Befehlszeile:

```
PfadName.Text="C:\*.TXT"
```

Umgekehrt können Sie die Eigenschaft eines Objektes auch wie einen Ausdruck behandeln und das Ergebnis dieses Ausdrucks weiterverwenden, z. B. in eine Datei ausgeben oder einer Variablen zuweisen:

```
VarName=Objektname.Eigenschaft
```

Möchten Sie beispielsweise den in dem Eingabefeld Pfadname durch den Anwender eingegebenen Text auslesen und der Variablen *text$* zuweisen, verwenden Sie dazu folgende Befehlszeile:

```
text$=PfadName.Text
```

Sie finden die Eigenschaften aller Dialogelemente in der Referenz ausführlich erläutert.

Umsetzung von Eigenschaften, Methoden und Ereignissen

Hier wird anhand des Beispielprojekts beschrieben, wie Eigenschaften, Methoden und Ereignisse umgesetzt werden.

In der folgenden Tabelle finden Sie für unser Beispiel eine Auflistung, welchem Ereignis und welchem Dialogelement welche Ereignis-SUB zugeordnet worden ist. Im Anschluß daran finden Sie das dazugehörige, kommentierte Listing.

Element	Ereignis-name	Ereignisprozedur	Aufgabe
Aktualisieren	OnClick	Aktualisieren_Click	Wird ausgeführt, wenn AKTUALISIEREN angeklickt wird.

Das dokumentierte Beispiellisting

```
option explicit
'Variable, zum Sichern des Pfades
Dim Pfad$          'Variable wird auch im Dialog benötigt,
             'wird daher VOR den SUBs und FUNCTIONs
             'deklariert, und ist damit modulglobal
Sub ErweiterteSuche
   pfad$="C:\"        'Definiert den "Ausgangspfad",
             'in dem alle Dateien
             'ausgelesen werden
   EingabeDialog.Show           'Ruft den Dialog auf
End Sub
```

Diese SUB bildet den Programmstart. Sie setzt den Vorgabepfad für das Dateilistenfeld unseres Beispiels durch "Pfad$" und macht sonst nichts weiter, als den Dialog mit der SHOW-Methode aufzurufen.

```
Sub EingabeDialog_Load
   dim DokName$
   exit sub
   PfadName.Text=Pfad$
   DokName$=Dir$(Pfad$)
   if DokName$<>"" then
      DateiListe.AddItem DokName$,DateiListe.ListCount
      do
         DokName$=Dir$
         if DokName$="" then
            exit do
         endif
         DateiListe.AddItem DokName$, DateiListe.ListCount
      loop
   endif
End Sub
```

Bei dieser SUB handelt es sich um ein Ereignis. Das Ereignis wird ohne weiteres Zutun des Programmierers dann aufgerufen, wenn der Dialog das erste Mal dargestellt, also geladen, wird. Diese SUB setzt zunächst den zuvor in "Pfad$" gespeicherten Vorgabepfad mit der TEXT-Eigenschaft des Eingabefeld-Elementes "PfadName" in das Eingabefeld ein. In der anschließenden DO...LOOP-Schleife liest die Routine mit Hilfe der DIR$-Funktion alle im Pfad vorhandenen Dateien ein und stellt sie mit der ADDITEM-Methode in die Dateiliste.

```
'Ereignisprozedur:
'Wird aufgerufen, wenn [Dazu] angeklickt wird
Sub Dazu_Click
   If Dateiliste.ListIndex=-1 then
      MsgBox "Kein Eintrag markiert"
   else
      DokumentListe.AddItem
DateiListe.List(Dateiliste.ListIndex),Dokumentliste.ListIndex+1
   endif
End Sub
```

DAZU_CLICK ist ebenfalls ein Ereignis, das aufgerufen wird, wenn der Anwender die DAZU-Schaltfläche betätigt. Diese Routine überprüft zunächst, mit der LISTINDEX-Eigenschaft angewendet auf die Dateiliste, ob überhaupt ein Eintrag der Liste angeklickt ist. Ist dies der Fall, liest sie mit der LIST-Eigenschaft den Text des angeklickten Eintrags aus und stellt diesen, wiederum mit der ADDITEM-Methode, in die andere Listbox.

Listboxen erlauben es durch Methoden und Eigenschaften, die Liste mit Werten zu füllen (.ADDITEM Eintrag$), mit der LISTINDEX-Eigenschaft festzustellen, ob und welche Eintragsnummer (von 0 bis (n-1) durchnummeriert) der Anwender angeklickt hat (-1 als Funktionsergebnis bedeutet dabei, daß kein Eintrag selektiert ist), und einen bestimmten Eintrag mit Hilfe der LIST-Eigenschaft (.LIST Eintragsnummer%) auszulesen.

```
'Ereignisprozedur:
'Wird aufgerufen, wenn [löschen] angeklickt wird
Sub Loeschen_Click
   If DokumentListe.ListIndex=-1 then
      MsgBox "Kein Eintrag markiert"
   else
      DokumentListe.RemoveItem(DokumentListe.ListIndex)
   endif
End Sub
```

Mit Hilfe dieser Ereignisprozedur wird ein Eintrag aus der Dokumentliste gelöscht, falls der Anwender die Schaltfläche <<LÖSCHEN angeklickt und zuvor einen Eintrag selektiert hat. Mit Hilfe der REMOVEITEM-Methode (.REMOVEITEM Eintragsnummer%) wird der Eintrag aus der Dokumentliste entfernt.

```
'Ereignisprozedur:
'Wird aufgerufen, wenn auf Aktualisieren geklickt wird
Sub Aktualisieren_Click()
   Dim DokName$
   Pfad$=PfadName.Text
   DateiListe.Clear
   DokName$=Dir$(Pfad$)
   if DokName$<>"" then
```

```
      DateiListe.AddItem DokName$,DateiListe.ListCount
      do
         DokName$=Dir$
         if DokName$="" then
            exit do
         endif
         DateiListe.AddItem DokName$, DateiListe.ListCount
      loop
   endif
End Sub
```

Diese Ereignisprozedur sorgt dafür, daß die Liste neu geladen wird, wenn der Benutzer im Eingabefeld einen neuen Pfad bestimmt und anschließend AKTUALISIEREN angeklickt hat. Das Verfahren ist dabei dasselbe wie beim LOAD-Ereignis des Dialogs, mit einem Unterschied: Damit die vorhandenen Einträge nicht um die neuen ergänzt werden, wird die Datei zuvor mit Hilfe der CLEAR-Eigenschaft gelöscht.

```
'Ereignisprozedur:
'Wird aufgerufen, wenn [OK] angeklickt wird
Sub OK_Click
   dim DateiNamen$(1 to 100)    'Speicher für die Dateinamen
   dim counter%       'Zähler, wieviele Dateien durchsucht
werden
   dim text$       'String, der den Suchtext aufnimmt
   dim z%          'Zähler für FOR/NEXT
   counter%=1
   do
      DateiNamen$(counter%)=InputBox("Bitte geben Sie den Namen
der Datei ein (* für Ende)","Erweiterte Suche","")
      if DateiNamen$(counter%)="*" then
         counter%=counter%-1
         exit do
      endif
      counter%=counter%+1
   loop
   text$=inputBox("Bitte Suchtext eingeben","Erweiterte
Suche","")
   for z%=1 to counter%
      Documents.Open(DateiNamen$(z%))
      anzahl%=0
      do
         ActiveWindow.Search(Text$)
         anzahl%=anzahl%+1
      loop until Selection.Value=""
      print "Der Suchbegriff trat im Dokument
";DateiNamen$(z%);" ";anzahl%;" mal auf."
      ActiveDocument.Close
   next z%
End Sub
```

Diese Ereignisroutine bildet das ursprüngliche Hauptprogramm. Klickt der Anwender OK, liest die Routine alle in der Dokumentenliste stehenden Da-

teien in ein String-Array ein. Anschließend wird der Programmteil ausgeführt, der den Suchbegriff einliest, und die angegebenen Dokumente darauf untersucht. Diese Schritte haben wir bereits im allerersten Beispielprogramm kennengelernt.

```
'Ereignisprozedur:
'Wird aufgerufen, wenn [Abbruch] angeklickt wird
Sub Abbruch_Click
   EingabeDialog.Hide 'Dialog schließen,
            'da ErweiterteSuche Basis-Routine war,
            'ist das Programm damit beendet
End Sub
```

In dieser letzten Ereignisprozedur wird das Programm abgebrochen, sobald der Anwender ABBRUCH angeklickt hat. Mit der HIDE-Methode wird der Dialog geschlossen.

Speichern des Projektes

Hier erfahren Sie, wie Sie ein StarBasic-Projekt speichern können.

Sie haben bereits erfahren, daß Makros bzw. StarBasic-Programme fest mit einem Dokument bzw. einer Dokumentvorlage verknüpft sind. Das bedeutet, daß Sie beim Speichern eines Dokumentes auch gleichzeitig das mit ihm verbundene StarBasic-Projekt speichern.

So speichern Sie ein Basic-Projekt, das mit einem Dokument oder einer Dokumentvorlage verbunden ist

Um ein Basic-Projekt zu speichern, das mit einem Dokument oder einer Dokumentvorlage verbunden ist, speichern Sie lediglich das Dokument bzw. die Dokumentvorlage. Alle Bibliotheken, Module und damit auch SUBs und FUNCTIONs werden dabei automatisch mit gespeichert. Um ein an eine Vorlage oder ein Dokument gebundenes Basic-Projekt zu speichern, verlassen Sie die IDE über die Schließ-Schaltfläche des Fensters oder das Menü DATEI - SCHLIEßEN und speichern das Dokument auf die bekannte Art und Weise mit DATEI - SPEICHERN bzw. DATEI - SPEICHERN UNTER.

Möchten Sie die IDE nicht verlassen, sondern anschließend weiterprogrammieren, können Sie mit der Tastenkombination (Strg)(Tab) in das Dokumentenfenster wechseln, das Dokument und damit Ihr Programm mit DATEI - SPEICHERN speichern und durch die gleiche Tastenkombination wieder in die IDE zurückkehren.

So speichern Sie ein Basic-Projekt, das mit einer StarOffice-Anwendung verbunden ist

Hierbei brauchen Sie sich um überhaupt nichts zu kümmern. Sämtliche Module, Bibliotheken und damit auch SUBs und FUNCTIONs, die Sie der Applikation hinzugefügt haben, werden beim Verlassen der StarOffice-Anwendung automatisch mitgespeichert.

"Technische Daten"

Hier erfahren Sie alles über die Leistungsfähigkeit von StarBasic. Sie erfahren beispielsweise, welche Variablentypen StarBasic kennt, in welchen Bereichen sich Werte bestimmter Variablen bewegen dürfen, aber auch, welche Stukturierungsmöglichkeiten Ihnen durch Werkzeuge wie Module und Bibliotheken gegeben werden und welchen Konventionen die Anwendung dieser Werkzeuge unterliegt.

Variablen

Hier wird beschrieben, was es mit den Variablen auf sich hat.

Variablen sind "das A und O" bei der Programmierung von Basic. Variablen speichern zur Laufzeit bestimmte numerische Werte, Zeichenketten oder sogenannte Objekte. Man kann Variablen auslesen oder deren Inhalte verändern. In StarBasic müssen Variablen nicht gesondert deklariert werden. Sie sind gültig, sobald Sie das erste Mal verwendet werden. Mit Hilfe der Anweisung OPTION EXPLICIT kann man jedoch die Deklaration von Variablen mit Hilfe des DIM-Statements zwingend machen.

Konventionen bei der Vergabe von Variablennamen

Variablennamen dürfen bis zu 255 Zeichen lang sein und alle Zeichen des Alphabetes sowie Ziffern enthalten, solange der Variablenname nicht mit einer Ziffer beginnt. Die Verwendung von deutschen Umlauten sowie von Satz- und Sonderzeichen außer dem Unterstrich ("_") ist nicht erlaubt. StarBasic unterscheidet nicht zwischen der Groß- und Kleinschreibung von Variablen. Variablen dürfen Leerzeichen enthalten, müssen dann aber in eckigen Klammern eingeschlossen sein.

Im folgenden ein paar Beispiele für das korrekte und nichtkorrekte Vergeben von Variablennamen:

```
Vorname=5          'Korrekt, keine unerlaubten Zeichen
Vorname5=15        'Korrekt, Variable beginnt nicht mit einer
Ziffer
```

```
Vorname_5=20      'Korrekt, Unterstrich ist erlaubt
Vor Name=20       'Nicht korrekt, Variable enthält Leerzeichen
und
                     'ist nicht in eckigen Klammern einge-
schlossen
Müller=25         'Nicht korrekt, Variable enthält Umlaut
5Vornamen=12      'Nicht korrekt, Variable beginnt mit einer
Ziffer
[Vor Name]=12     'Korrekt trotz Leerzeichen, denn Variable
steht
                     'in eckigen Klammern
Vor,Name=12       'Nicht Korrekt, Variable enthält Satzzeichen
```

Variablendeklarationen

Variablen brauchen in StarBasic nicht explizit deklariert zu werden, wie Sie es vielleicht von anderen Programmiersprachen her kennen. Lediglich Arrays müssen vor Verwendung deklariert werden. Eine Variablendeklaration erfolgt mit dem DIM-Befehl; mehrere Deklarationen können mit Komma getrennt erfolgen. Um den Typ einer Variablen bei einer Deklaration festzulegen, verwendet man entweder ein spezielles Typendeklarationszeichen, das hinter dem Namen angegeben wird, oder ein entsprechendes Schlüsselwort. Beispiele:

```
DIM a$                      'Deklaration der Variable "a" als
String
DIM a as STRING             'Deklaration der Variablen "a" als
String
DIM a$, b as INTEGER        'Deklaration eines Strings und
eines Integers
                               'durch Komma getrennt
```

Wichtig bei der Deklaration von Variablen ist, daß das Typendeklarationszeichen immer wieder verwendet werden muß, wenn es auch bei der Deklaration anstelle des Schlüsselwortes verwendet wurde. Die folgenden Zeilen sind also nicht statthaft:

```
DIM a$                      'Deklaration als String
a="Testzeichenkette"        'Falsch, es müßte "a$=" heißen
```

> Variablen mit einem bestimmten Namen, die einmal als bestimmter Typ deklariert wurden, dürfen unter dem gleichen Namen nicht nochmals als anderer Typ deklariert werden!

Erzwingung von Variablendeklarationen

StarBasic bietet Ihnen die Möglichkeit, jede Variable zwingend zu deklarieren, und zwar mit der Anweisung

```
OPTION EXPLICIT
```

die als erste Zeile im Programmcode eines Moduls, also noch vor der ersten SUB, stehen muß. Normalerweise braucht der Programmierer in StarBasic lediglich Arrays, also Felder, zu deklarieren. Andere Variablen werden in Abhängigkeit des Typendeklarationszeichens oder, falls keines angegeben ist, als Single-Variablentyp deklariert, sobald Sie das erste Mal verwendet werden. Der gewollte Deklarationszwang einer jeden im Programm verwendeten Variable ist zwar ein wenig aufwendiger, hat aber folgenden Vorteil: Programmfehler, die durch Tippfehler in Variablennamen auftreten, werden damit ausgeschlossen. Ein Beispiel:

Sie verwenden die Variable *Datenspeicher$* in einer SUB. Irgendwo im Programmtext arbeiten Sie mit der Variablen weiter und weisen ihr ein bestimmtes Ergebnis zu. Angenommen, Sie haben sich vertippt und weisen dieses Ergebnis mit der folgenden Zeile zu:

```
Datnespeicher$=Ausdruck$
```

StarBasic würde diesen Ausdruck anstandslos akzeptieren, da es davon ausgeht, daß Sie im folgenden eine neue Variable verwenden möchten. StarBasic muniert es jedoch, wenn OPTION EXPLICIT definiert wurde, da der Variablenname nicht definiert wurde. Sie erkennen in diesem Fall den Programmfehler als Tippfehler beim Programmablauf, sobald die "falsche" Variable das erste Mal verwendet wird.

Variablentypen

StarBasic kennt eine ganze Anzahl verschiedener Variablentypen. Grundsätzlich werden Variablentypen in vier Typenkategorien unterteilt:

NUMERISCHE VARIABLEN: Numerische Variablen nehmen Zahlen mit unterschiedlichen Wertebereichen auf. Es gibt Typen, die größere und andere, die kleinere Wertebereiche speichern. Man unterscheidet des weiteren zwischen Typen, die ganzzahlige Werte, also solche ohne Nachkommastellen, speichern, und Fließkommatypen, die auch gebrochene Zahlen speichern können. Alle Typen haben Vor- und Nachteile: Typen, die kleinere Wertebereiche und nur ganze Zahlen verarbeiten, benötigen vergleichsweise wenig Speicherplatz und beschleunigen die Bearbeitungsgeschwindigkeit. Fließkommatypen sind zwar rechenintensiver und benötigen mehr Speicherplatz, jedoch lassen sich mit ihnen sehr präzise Berechnungen durchführen.

STRING-VARIABLEN: String-Variablen nehmen beliebige Zeichenketten auf.

BOOLEAN-VARIABLEN: Boolean-Variablen speichern nur die Zustände TRUE und FALSE (wahr und falsch).

OBJEKT-VARIABLEN: Objekt-Variablen nehmen Objekte ganz unterschiedlicher Art auf. Objekte können beispielsweise Dokumente oder Tabellen innerhalb

eines Dokuments sein. Mit Hilfe entsprechender Eigenschaften oder Methoden können diese Objekte dann vom Programm "bearbeitet" werden.

Integervariablen

Integervariablen speichern ganzzahlige Werte zwischen -32768 und 32767. Bei der Zuweisung von Fließkommawerten an eine Integervariable werden die Nachkommastellen abgeschnitten. "%" ist das Typendeklarationszeichen. Integervariablen lassen sich extrem schnell verarbeiten und werden vorwiegend als Zählvariablen in Schleifen eingesetzt. Integervariablen haben einen internen Speicherbedarf von 2 Bytes.

Deklaration:

```
DIM Variable%
DIM Variable as INTEGER
```

Long-Integervariablen

Long-Integervariablen speichern ganzzahlige Werte zwischen -2.147.483.648 und 2.147.483.647. Bei der Zuweisung von Fließkommawerten an eine Integervariable werden die Nachkommastellen abgeschnitten. "&" ist das Typendeklarationszeichen. Long-Integervariablen lassen sich sehr schnell verarbeiten und werden vorwiegend als Zählvariablen in Schleifen eingesetzt, die einen erweiterten Wertebereich benötigen. Long-Integervariablen haben einen internen Speicherbedarf von 4 Bytes.

Deklaration:

```
DIM Variable&
DIM Variable as LONG
```

Single-Variablen

Single-Variablen speichern Werte zwischen 3,402823 x 10E38 und 1,401298 x 10E-45, wobei der Wertebereich sowohl positiv als auch negativ sein kann. Single-Variablen fallen unter die Kategorie der Fließkomma-Variablen. Diese Typen haben die Eigenschaft, daß die Genauigkeit der Darstellung hinter dem Komma abnimmt, je größer der Teil vor dem Komma des darzustellenden Wertes wird. "!" ist das Typendeklarationszeichen. Single-Variablen werden bei mathematischen Berechnungen eingesetzt, bei denen eine durchschnittliche Berechnungsgenauigkeit ausreicht. Sie werden langsamer als alle Integervariablen und Currency-Variablen, aber schneller als Double-Variablen verarbeitet. Intern werden sie durch 4 Bytes gespeichert.

```
DIM Variable!
DIM Variable as SINGLE
```

Double-Variablen

Double-Variablen speichern Werte zwischen 1,79769313486232 x 10E308 und 4,94065645841247 x 10E-324, wobei der Wertebereich sowohl positiv als auch negativ sein kann. Double-Variablen fallen wie Single-Variablen unter die Kategorie der Fließkomma-Variablen, bei denen die Genauigkeit der Darstellung hinter dem Komma abnimmt, je größer der Teil vor dem Komma des darzustellenden Wertes wird. "#" ist das Typendeklarationszeichen. Double-Variablen werden bei mathematischen Berechnungen eingesetzt, bei denen eine extrem hohe Berechnungsgenauigkeit verlangt wird. Intern werden sie durch 8 Bytes gespeichert.

```
DIM Variable#
DIM Variable as DOUBLE
```

Currency-Variablen

Currency-Variablen werden intern als 64-bit-Zahlen (8 Bytes) in einem ganzzahligen Format dargestellt und durch 10.000 dividiert. Dadurch ergibt sich eine Festkommazahl mit 15 Vorkomma- und 4 Nachkommastellen. Diese Darstellung ergibt einen Wertebereich von –922.337.203.685.477,5808 bis +922.337.203.685.477,5807. Das Typendeklarationszeichen für Currency ist "@". Currency-Variablen eignen sich besonders für Berechnungen mit Geldbeträgen und Festkommaberechnungen, die eine hohe Genauigkeit erfordern.

```
DIM Variable@
DIM Variable as CURRENCY
```

String-Variablen

String-Variablen speichern Zeichenketten von bis zu 64.000 Zeichen Länge. Ein Zeichen wird dabei durch ein Byte als ASCII-Code gespeichert. String-Variablen eignen sich hervorragend zur Textverarbeitung innerhalb von Programmen, können aber auch zur Zwischenspeicherung von beliebigen, nicht druckbaren Zeichen bis zur Länge von 64 KByte verwendet werden. "$" ist das Typendeklarationszeichen. Der verwendete Speicherplatz hängt von der Anzahl der Buchstaben bzw. Bytes ab, die in einer Variablen gespeichert werden.

```
DIM Variable$
DIM Variable as STRING
```

Strings-Variablen

Eine Strings-Variable ist eine Art String-Array, in dem mehrere String-Variablen zu einer einzigen Variablen zusammengefaßt werden können. Diese wird intern vom Programm als Variable vom Typ Object behandelt. Strings versteht folgende Eigenschaften und Methoden:

Add	Fügt der Strings-Variablen einen beliebigen String hinzu.
Remove(i)	Enfernt einen mit dem Index i bezeichneten String aus der Strings-Variablen.
Item(i)	Ermöglicht den Zugriff auf das i-te Element der Strings-Variablen.
Count	Ermittelt die Anzahl der Elemente in einer Strings-Variablen.
Clear	Löscht den Inhalt einer Strings-Variablen.

Die Strings-Variable ist ein komplexer Datentyp, der dynamisch arbeitet. Das bedeutet, daß mit einer Strings-Variablen ein kompliziertes Gebilde konstruiert werden kann, wobei sich die Bestandteile dieser Variablen erst zur Laufzeit definieren.

Dies läßt sich anhand der CreateTable-Methode verdeutlichen, mit der eine neue Datenbank-Tabelle erzeugt wird. Die Syntax lautet am Writer-Modul:

ActiveWindow.CreateTable(TableName$, Columns As Object)

Der Parameter TableName$ legt den Namen der neuen Tabelle fest, die Definition der Spalten erfolgt mit einer Strings-Variablen:

```
DIM Columns as new STRINGS
```

Die Strings-Variable wird mit den einzelnen Spalten der Tabelle gefüllt. Dazu muß zunächst jede einzelne Tabellenspalte definiert werden. Eine Spalte ist definiert durch die Angaben Feldname, Feldtyp, Feldlänge und der Information, ob eine Eingabe erforderlich ist oder nicht, also z. B.:

```
S1$ = "Column1, Text, 10, False"          'definiere Spalte 1
im Textformat mit einer Länge von 10 Zeichen
S2$ = "Column2, Numeric, Double, True"    'definiere Spalte 2
im Zahlenformat Double, wobei kein Datenfeld leer sein darf
S3$ = "Column3, Currency, False"          'definiere Spalte 3
im Währungsformat
```

Die Spalten S1 bis S3 sind nun jeweils in einem String definiert, und die einzelnen Spalten müssen jetzt für den Columns-Parameter der CreateTable-Methode zusammengefügt werden. Dies geschieht mit Hilfe der Add-Methode:

```
Columns.Add S1$
Columns.Add S2$
Columns.Add S3$
```

Die Strings-Variable Columns wird so durch das Hinzufügen der drei String-Variablen S1$ bis S3$ für die einzelnen Spalten definiert und kann von der CreateTable-Methode ausgewertet werden.

Boolean-Variablen

Boolean-Variablen speichern lediglich die Zustände Wahr und Falsch. Wird der Inhalt einer Boolean-Variable z. B. durch den PRINT-Befehl ausgegeben, werden diese Zustände in -1 (TRUE, Wahr) und 0 (FALSE, Falsch) ausgegeben. Boolean-Variablen eignen sich dazu, binäre Zustände zu speichern, z. B. das Ergebnis eines Vergleichs. Intern werden Boolean-Variablen durch einen 2-Byte-Integerwert dargestellt. Wird einer Boolean-Variable ein Wert zugewiesen, so wird dieser automatisch zu "falsch" (0) konvertiert, solange er nicht genau -1 war. Boolean-Variablen haben kein spezielles Typendeklarationszeichen und können nur durch das entsprechende Schlüsselwort deklariert werden.

```
DIM Variable as BOOLEAN
```

Date-Variablen

Date-Variablen speichern ausschließlich Datumswerte und Zeitwerte in einem bestimmten, internen Format, das den Programmierer jedoch nicht weiter interessieren muß. Date-Variablen können Werte zugewiesen werden, die beispielsweise durch DATESERIAL, DATEVALUE, TIMESERIAL oder TIMEVALUE von der "normal lesbaren" Darstellung in das interne Format umgewandelt werden. Die Zurückwandlung erfolgt mit den Funktionen DAY, MONTH und YEAR bzw HOUR, MINUTE, SECOND. Date-Variablen haben den Sinn, Daten, die in Variablen gespeichert sind, vergleichen zu können; dies geschieht durch die Umwandlung in eine ganze Zahl und Berechnung von Differenzen. Es gibt kein spezielles Typendeklarationszeichen - Date-Variablen können nur durch das Schlüsselwort DATE deklariert werden.

```
DIM Variable as DATE
```

Definition von Variablen

Sobald eine Variable deklariert wird, ist Sie automatisch auf den "Null"-Wert zurückgesetzt, d. h., daß der Programmierer nicht darauf achten muß, daß eine Variable, die deklariert wurde, manuell auf einen Basis-Wert zurückgesetzt werden muß. Es gilt folgende Regel:

NUMERISCHEN VARIABLEN wird automatisch der Wert 0 zugewiesen, sobald die Variable deklariert wurde.

DATE-VARIABLEN wird intern der Wert 0 zugewiesen, das Ergebnis entspricht also einer entsprechenden Umwandlung des Wertes "0" mit einer der Funktionen DAY, MONTH oder YEAR bzw. HOUR, MINUTE, SECOND.

STRING-VARIABLEN wird direkt bei der Deklaration ein Leerstring ("") zugewiesen.

Vordefinierte Systemvariablen

StarBasic kennt eine Reihe vordefinierter Systemvariablen, die bei einer Abfrage immer ein bestimmtes Ergebnis liefern. Diese finden Sie im folgenden beschrieben:

TRUE und FALSE

Diese beiden Variablen sind vom Typ Boolean und liefern den Wert 0 (FALSE, Falsch) oder den Wert -1 (TRUE, Wahr) zurück. Man verwendet diese Variablen zum Beispiel in IF...THEN...ELSE-Abfragen, um den Wahrheitsgehalt eines Vergleichs zu testen. Wie in Basic üblich, werden Vergleiche durch die Vergleichsoperatoren (=, <, >) abgefragt. Die Anweisung

```
PRINT 5=3
```

ergibt beispielsweise den Wert 0 (FALSE, Falsch), da der Ausdruck das Ergebnis "Aussage (Vergleich) unwahr" hat. Die Anweisung

```
PRINT 5>3
```

ergibt hingegen den Wert -1 (TRUE, Wahr), da der Ausdruck das Ergebnis "Aussage (Vergleich) wahr" hat. Die Verwendung der Systemvariablen bildet rein technisch keinen Unterschied zur Verwendung der direkten Zahlenwerte 0 und -1; Programmcode, der mit Hilfe dieser Variablen arbeitet, wird jedoch für Außenstehende leichter lesbar.

PI

PI ist eine mathematische Konstante und ergibt als numerisches Ergebnis stets die Kreiszahl Pi = 3,141592657...

Arrays

StarBasic kennt ein- oder mehrdimensionale Arrays (Felder), denen ein bestimmter angebbarer Variablentyp zugrunde liegt. Arrays werden immer dann verwendet, wenn im Programm Listen oder Tabellen verarbeitet werden sollen. Der Vorteil von Arrays ist, daß man die einzelnen Elemente über

Indizes ansprechen kann, die selbst ebenfalls in Form von numerischen Ausdrücken oder Variablen formuliert werden können.

Arrays werden mit dem DIM-Befehl deklariert. Es gibt dabei zwei Möglichkeiten festzulegen, in welchem Bereich die Indizes angesprochen werden können:

```
DIM text$(20)            '21 Elemente, von 0 bis 20 durchnume-
riert
DIM text$(10,2)          '33 Elemente, Dimension 0, von 0 bis
10,  Dimension 1, von 0 bis 10, Dimension 2, von 0 bis 10.
DIM text$(5 to 25)       '21 Elemente, von 5 bis 25 durchnume-
riert
DIM text$(-15 to 5)      '21 Elemente (das 0. ebenfalls!),
                            ' von -15 bis 5 durchnumeriert
```

Der Bereich der Indizes kann sowohl positive als auch negative Werte annehmen. Die maximal mögliche Anzahl der Elemente, die durch Indizes angesprochen werden können, beträgt 16368.

Konstanten

Konstanten sind eine "Abart" von Variablen und dienen nur der besseren Lesbarkeit eines Programms. Konstanten werden nicht als bestimmter Typ definiert, sondern werden als reiner Programmcode-Platzhalter verwendet. Eine Konstante kann nur ein einziges Mal im Programm definiert und dann nie wieder umdefiniert werden. Sie definieren eine Konstante mit folgender Befehlszeile:

```
CONST ConstName=Ausdruck
```

Dabei ist es völlig egal, um was für einen Ausdruck es sich handelt. Wird ein Programm gestartet, so wird StarBasic-intern der Programm-Code umgearbeitet; dabei wird für jede Konstante der zuvor definierte Ausdruck eingesetzt. Im Prinzip wird intern ein "Suchen/Ersetzen" durchgeführt. Der Konstantenname, wann immer er im Basic-Programm-Code verwendet wurde, wird durch den festgelegten Ausdruck ersetzt. Aus diesem Grund spielt der Typ, der einer Konstanten zugewiesen wird, auch überhaupt keine Rolle, da intern sowieso nur der Ausdruck als solcher verwendet wird.

Verwenden von Objekten

An dieser Stelle erfahren Sie, wie Sie in StarBasic mit Objekten arbeiten.

Die Frage, die sich stellt, möchte man aus einer Programmiersprache ein Anwendungsprogramm bedienen, lautet: Was ist der flexibelste, effizienteste und dennoch unkomplizierteste Weg, bestimmte Elemente zu manipulieren?

StarBasic bedient sich dazu sogenannter Objekte und Objekt-Collections, um bestimmte, in der Regel StarOffice-programmabhängige Problemlösungen realisieren zu können. Objekte haben einen großen Vorteil: sie werden durch Einstellen von Eigenschaften oder Anwenden von Methoden auf immer dieselbe Art und Weise bedient, unabhängig davon ob es sich bei einem Objekt z. B. um ein Dokument, eine Tabelle oder das Programm selbst handelt. Ferner erleichtern sie durch ihre quasi selbstdokumentierende Struktur sowohl die Lesbarkeit als auch das Entwickeln eines Programms.

Die folgenden Abschnitte beschreiben Ihnen, wie Sie Text-, Tabellen- oder Präsentationsdokumente mit Hilfe von Objekten verwenden können.

StarBasic kennt Objekte, die "einfach vorhanden" sind, und andere, die Sie mithilfe bestimmter Anweisungen erst konstruieren müssen. Das mag zunächst ein wenig abstrakt erscheinen, relativiert sich jedoch, sobald Sie die Dinge im Detail betrachten.

Anwenden von Objekten im Programm-Code

Betrachten wir zunächst die Syntax beim Ansprechen von Objekten, also wie sie in StarBasic verwendet werden. Dabei lernen wir gleichzeitig das erste Objekt kennen, das DOCUMENT-Objekt, das ein geöffnetes Textdokument repräsentiert. Für ein erstes Beispiel soll es das Ziel sein, den Inhalt dieses Dokumentes zu sichern und das Dokument anschließend zu schließen. Die Syntax, um das DOCUMENT-Objekt anzusprechen, kennen Sie schon von der Anwendung von Methoden und der Definition von Eigenschaften von Dialogelementen (denn sie sind auch nichts anderes als Objekte).

```
[unbenannt1].Save "Dateiname"
[unbenannt1].Close False
```

"[unbenannt1]" bezeichnet hierbei das Objekt. SAVE und CLOSE sind Methoden, die auf dieses Objekt angewendet werden.

Anwenden von Methoden

Methoden werden auf Objekte angewendet und haben irgendeine Aktion zur Folge. Die Syntax lautet dabei grundsätzlich:

```
Objekt.Methode (Parameter)
```

Das bedeutet: "Wende die Methode mit den folgenden Parametern auf das Objekt an." Sichtbar wird das im oben stehenden Beispiel. Die SAVEAS-Methode, eine für das DOCUMENT-Objekt gültige Methode, speichert beispielsweise den Inhalt des Dokumentes ab. Das Dokument selber spielt die Rolle des Objektes.

Zuweisen von Eigenschaften

Eigenschaften bestimmen das Aussehen oder die zukünftige Verhaltensweise eines Objektes. Um beim DOCUMENT-Objekt zu bleiben, betrachten wir beispielsweise die WINDOWSTYLE-Eigenschaft, die verschiedene Werte annimmt.

```
[unbenannt1:1].Maximized=True
a%=[unbenannt1:1].Maximized
```

Die erste Zeile bestimmt die MAXIMIZED-Eigenschaft des Fensters. Sie bestimmt, ob das Fenster voll vergrößert dargestellt werden soll oder nicht. Durch Zuweisen des Wertes an die Eigenschaft eines Objektes wird also das Objektverhalten oder -aussehen beeinflußt. Für die Definition einer Objekteigenschaft gilt grundsätzlich:

```
Objekt.Eigenschaft=Wert
```

Andersherum (2. Zeile des Beispiels) kann eine Eigenschaft auch als Ausdruck behandelt werden, so daß der Programmierer die Möglichkeit hat, eine Objekteigenschaft zu erfragen. Es gilt dabei grundsätzlich

```
ausdruck=Objekt.Eigenschaft
```

Objekte, die Objekte zurückliefern

Bestimmte Objekte, die wie Eigenschaften abgefragt werden, liefern ihrerseits Objekte als Ergebnis zurück. Betrachten wir beispielsweise die ACTIVEWINDOW-Eigenschaft des APPLICATION-Objektes, die das aktive Fenster des angesprochenen Dokumentes als WINDOW-Objekt zurückliefert.

```
a$ = Application.ActiveWindow.Name
```

Das WINDOW-Objekt enthält u. a. den Namen des Fensters eines Dokumentes. Die NAME-Eigenschaft filtert aus allen Informationen, die das WINDOW-Objekt speichert, genau den Namen des Fensters heraus. Die Reihenfolge der Eigenschaftenangabe ist damit klar. Im Klartext bedeutet die obige Anweisung: "Liefere den Namen des Fensters durch die NAME-Eigenschaft des Objektes, das durch die ACTIVEWINDOW-Eigenschaft des APPLICATION-Objektes ermittelt wurde, und speichere ihn in A$ ab".

Objekte in StarBasic

Document-Objekt

Dieses Objekt erlaubt den Zugriff auf ein geöffnetes Dokument eines StarOffice-Programms. Es gibt grundsätzlich mehrere Möglichkeiten, Zugriff auf ein Dokument zu nehmen. Die erste Methode spricht das Dokument direkt per Namen an, beispielsweise:

```
[unbenannt1].SaveAs "Dateiname"
```

Eine weitere Möglichkeit ist das direkte Ansprechen des Objektes als aktives Dokument. Das ACTIVEDOCUMENT-Objekt liefert beispielsweise das aktive Dokument als DOCUMENT-Objekt zurück. Ein Beispiel für diese Art der Verwendung lautet:

```
ActiveDocument.SaveAs "Dateiname"
```

Es gibt eine dritte Möglichkeit ein DOCUMENT-Objekt anzusprechen - über eine Objekt-Collection.

Window-Objekt

Dieses Objekt erlaubt den Zugriff auf ein Fenster eines geöffneten Dokumentes eines StarOffice-Programms. Es gibt mehrere Möglichkeiten, ein WINDOW-Objekt anzusprechen. Die erste Möglichkeit ist, das Objekt direkt über den Fensternamen anzusprechen. Da pro Dokument mehrere Fenster geöffnet sein können, muß mit dem Dokumentennamen die Nummer, mit Doppelpunkt getrennt, angegeben werden, z. B.:

```
[unbenannt1:1].Maximize=True
```

Eine weitere Möglichkeit ist das Ansprechen des Objektes als aktives Fenster, z. B.:

```
ActiveWindow.Maximized=True
```

Es gibt eine dritte Möglichkeit ein WINDOW-Objekt anzusprechen - über eine Objekt-Collection.

Module-Objekt

Dieses Objekt beinhaltet die eigentliche Applikation als Objekt und erlaubt damit die Steuerung der Applikation durch Eigenschaften und Methoden. Beispiel:

```
ActiveModule.AboutDialog        REM Zeigt den About-Dialog an
```

Selection-Objekt

Das "Selection-Objekt" bezieht sich auf alles, was im StarOffice in irgendeiner Form mit einem selektierten Objekt zu tun hat. In einem Textdokument kann das selektierte Objekt beispielsweise durch die Position der Schreibmarke oder einen selektiertem Textbereich gegeben sein. Aber auch Rahmen, Tabellen und Grafiken können Selection-Objekte darstellen.

Ist ein Text markiert und wird das Selection-Objekt durch Methoden oder Eigenschaften für Textdokumente verwendet, beziehen sich Veränderungen

auf den markierten Textbereich. Ist hingegen kein Objekt selektiert, so beziehen sich die Veränderungen im Textdokument auf die Position der Schreibmarke. Einzige Besonderheit ist die INSERT-Methode: Sie wird ebenfalls auf das Selection-Objekt angewendet und bewirkt, daß ein Text an die aktuelle Position der Schreibmarke eingefügt wird (und damit in das aktuelle Fenster gelangt, was auch der Grund für die Ausnahme ist).

Die Objekt-Collection

Objekt-Collections sind, wie der Name schon andeutet, Objekt-Sammlungen. Strenggenommen ist eine Objekt-Collection nichts anderes als ein Objekt, das verschiedene Objekte gleichen Typs aufnimmt. Am besten läßt sich eine Objekt-Collection mit einem Variablen-Array vergleichen, mit dem Unterschied, daß Objekt-Collections als eigener Typ vordefiniert sind, Variablen-Arrays nicht.

Ein Beispiel für eine Objekt-Collection ist beispielsweise die Documents-Collection. Man achte dabei auf das kleine, aber entscheidene "s", das den Unterschied zum DOCUMENT-Objekt ausmacht. Die Hauptaufgabe einer Collection ist es, die Verwendung von Objekten allgemeingültig zu machen. Dazu ein Beispiel:

Sie entwickeln ein Programm, das beispielsweise einen Begriff durch einen anderen ersetzt und das immer nur mit einem, nicht mit mehreren geöffneten Dokumenten arbeiten kann, denn Sie verwenden "lediglich" ein DOCUMENT -Objekt. Stellen Sie sich vor, der Anwender Ihres Programm sagt: "Ich möchte diese Funktion auf alle geöffneten Dokumente anwenden und nicht immer nur auf ein bestimmtes". In diesem Falle strecken Sie die Waffen, denn Sie wissen nur, ob Ihr eines Objekt (Dokument) geöffnet ist oder nicht, die anderen jedoch kennen Sie nicht.

Bedienen Sie sich in diesem Fall der Documents-Collection, wird die ganze Sache einfacher. Sie stellen mit der COUNT-Eigenschaft fest, wieviele DOCUMENT-Objekte es innerhalb der Documents-Collection gibt, und wenden Ihr Programm nacheinander (am besten innerhalb einer Schleife) auf jedes Dokument an:

```
Sub Test
    :
   anz%=Documents.Count                    'Wieviele gibt es?
   For z%=1 to anz%                'Alle Dokumente bearbeiten
      Documents(z%).Search...      'Methoden und Eigenschaften,
              'also Ihr Programm anwenden
   Next z%          'Alle Dokumente wurden berücksichtigt
    :
End Sub
```

Hier erkennen wir schon, auf welche Art und Weise wir einer Objekt-Collection ein einziges Objekt entlocken können - wir indizieren es wie Variablen in Arrays. Andere Möglichkeiten sind beispielsweise die Angabe eines Namens, um das Objekt zurückzuliefern. In diesem Fall durchsucht StarBasic die gesamte Objekt-Collection nach dem Namen und liefert, so gefunden, das passende Objekt zurück:

```
Documents("unbenannt1").Close
```

Das über Eigenschaften und Methoden Gesagte gilt natürlich auch für Objekte, die über eine Objekt-Collection angesprochen wurden.

Objekt-Collections in StarBasic

Windows-Collection

Die Windows-Collection beinhaltet alle in einem StarOffice-Programm geöffneten Fenster als WINDOW-Objekt.

Durch Indizierung der Collection mit dem Namen des Fensters (inclusive View-Nummer, siehe WINDOW-Objekt) oder mit der Fenster-Nummer (die Reihenfolge entspricht der, wie sie im Menü FENSTER aufgeführt ist) wird ein WINDOW-Objekt zurückgeliefert. Beispiele:

```
Windows(3)
Windows("unbenannt1:1")
```

Documents-Collection

Die Documents-Collection beinhaltet alle in einem StarOffice-Programm geöffneten Dokumente als DOCUMENT-Objekt.

Durch Indizierung der Collection mit dem Namen des Dokumentes oder mit der Dokument-Nummer (die Reihenfolge entspricht der, wie sie im Menü FENSTER aufgeführt ist) wird ein DOCUMENT-Objekt zurückgeliefert. Beispiele:

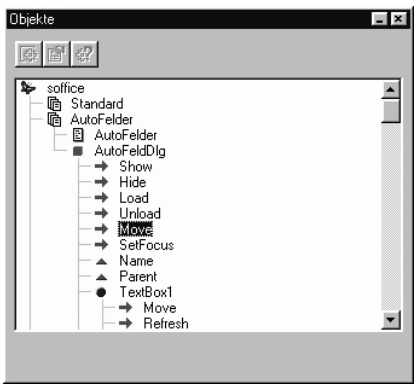
```
Documents(3)
Documents("unbenannt1")
```

Der Objektkatalog

Der Objektkatalog bietet für sämtliche Module und Dialoge, die Sie in Ihrem StarOffice eingebaut haben, eine Auflistung der Eigenschaften und Methoden, die auf das jeweils aufgeführte Objekt angewendet werden können. Zu jedem Modul werden sämtliche SUBs aufgeführt und zu jedem Dialog erhalten Sie eine Liste der anwendbaren Eigenschaften und Methoden. Auch sämtliche Kontrollfeldobjekte, die Sie in Ihren Dialog eingebaut haben, werden hier mitsamt ihrer Eigenschaften und Methoden aufgelistet.

Der Objektkatalog bietet die Möglichkeit, Eigenschaften und Methoden, die auf das jeweils aufgeführte Objekt angewendet werden sollen, direkt per Drag&Drop in den Basic-Editor zu übernehmen. So können Sie mit seiner Hilfe eine Befehlszeile in den Programm-Code an der aktuellen Cursorposition einfügen. Dazu verfahren Sie wie folgt:

- Klicken Sie das OBJEKTKATALOG-Symbol aus der Objektleiste an. StarBasic ruft den Objektkatalog auf.



- Der Dialog zeigt eine Liste aller verfügbaren Objekte. Viele Objekte (Dokumente, Vorlagen, die Applikation selbst und dort enthaltene Bibliotheken und Module) enthalten Unterobjekte, die Sie per Doppelklick ein- und ausblenden können. Ein Doppelklick auf einen Modul-Eintrag zeigt Ihnen beispielsweise alle dort enthaltenen SUBs oder FUNCTIONs an.
- Möchten Sie z. B. die Befehlszeile "AutoFeldDlg.TextBox1.Move" in den Basic-Editor einfügen, dann klicken Sie auf den entsprechenden Eintrag im Objektkatalog und ziehen Sie ihn mit der Maus an die Stelle im Programm-Code, an der der Befehlstext eingefügt werden soll.
- Schließen Sie den Objektkatalog durch Mausklick auf die Schließ-Schaltfläche, wenn Sie ihn nicht mehr benötigen.

Prozeduren und Funktionen

An dieser Stelle wird beschrieben, was es mit Prozeduren (SUBs) und Funktionen (FUNCTIONs) auf sich hat.

Prozeduren und Funktionen helfen Ihnen, Ihr Programm in immer kleinere Teilproblemlösungen aufzuteilen und damit übersichtlich zu strukturieren. Sie erstellen sich bei der Programmentwicklung nach und nach einen immer größeren Befehlsvorrat; Ihr Hauptprogramm muß dann nur noch auf diese "Teilproblemlöser" zurückgreifen, um das gesamte Problem, das durch Ihr StarBasic-Projekt bearbeitet werden soll, mit einem Schlag zu lösen.

Ein weiterer Vorteil von SUBs und FUNCTIONs ist die Tatsache, daß Sie das Rad nicht ständig neu erfinden müssen. Haben Sie einmal den Programmcode entwickelt, der eine kleine Teilaufgabe eines großen Problems löst, so können Sie diesen Code in Ihrem Projekt, aber auch in allen zukünftigen Projekten immer wieder verwenden. Ein Beispielprogramm soll das verdeutlichen:

Sie können dieses Programm nachvollziehen, indem Sie es aus der Online-Hilfe kopieren und in die IDE einfügen. Möchten Sie das Programm abtippen, wählen Sie EXTRAS - MAKRO, geben unter MAKRONAME "Start" ein und wählen die Schaltfläche NEU, um zur IDE zu wechseln.

```
option explicit
Sub Start
   dim text$
   text$=inputbox("Bitte geben Sie einen Text ein","Textein-
gabe","")
   print "Dieser Text enthält";WordCount%(text$);"Worte"
End Sub
'*** Funktion zählt Leerzeichen in einem String, und
'*** damit die Worte.
function WordCount%(WortText$)
   dim z%, count%
   count%=1
   for z%=1 to len(WortText$)
      if mid$(WortText$,z%,1)=" " then
         count%=count%+1
      endif
   next z%
   WordCount%=count%
end function
```

Starten Sie dieses Programm direkt aus der IDE, startet das Programm mit der SUB "Start" (einfach deswegen, weil Sie im Source-Code an erster Stelle steht). Innerhalb von "Start" wird der Anwender aufgefordert, einen Text einzugeben. Dieser Text wird der Funktion WORDCOUNT% übergeben, die die

Anzahl der Leerzeichen und damit die Anzahl der Worte ermittelt, die im Text vorhanden sind.

Der Vorteil vom Einsatz einer solchen FUNCTION liegt auf der Hand. Innerhalb des Hauptprogramms "Start" wird der Lesefluß nicht durch den Programm-Code von WordCount belastet. Aus dem Namen geht hervor, welche Aufgabe die Funktion übernimmt. Ein weiterer, viel entscheidenderer Vorteil: Braucht man diese Funktion erneut, müßte man Sie lediglich ein weiteres Mal aufrufen. Der Code selber bleibt dabei aber nur ein einziges Mal im Programm vorhanden - ergo: Der Programmcode wird kompakter und übersichtlicher.

Übergabe von Variablen an SUBs oder FUNCTIONs

In diesem Beispiel können Sie eine weitere Eigenschaft von SUBs bzw. FUNCTIONs erkennen. Beiden Strukturen kann man Variablen übergeben und sie damit so verwenden, als wären sie ein fester Bestandteil der StarBasic-Programmiersprache. Voraussetzung dafür ist, daß die SUB oder FUNCTION, die aufgerufen wird, diese Parameter erwartet. Die Deklaration dafür ist denkbar einfach:

```
SUB SubName(Parameter1 as Typ, Parameter2 as Typ,...)
   Programmcode
END SUB
```

Der Aufruf einer solchen SUB würde durch die folgende Befehlszeile vonstatten gehen:

```
SubName(Wert1, Wert2,...)
```

Sie selber brauchen dabei nur darauf zu achten, daß die Parameter, die Sie einer SUB übergeben, dieselben sind wie in der SUB-Deklaration angegeben.

Das gleiche gilt für die Verwendung von FUNCTIONs, nur mit dem Unterschied, daß sie ein Funktionsergebnis zurückliefern. Dieses Funktionsergebnis bestimmen Sie am besten kurz vor Erreichen des Funktionsendes. Sie bestimmen das Funktionsergebnis, indem Sie den Wert, den die Funktion zurückliefern soll, dem Funktionsnamen wie einem Parameter zuweisen (vergleiche Beispiellisting).

```
FUNCTION FunctionName(Parameter1 as Typ, Parameter2 as Typ,...)
as Typ
   Programmcode
   FunctionName=Funktionsergebnis
End Function
```

Der Aufruf einer solchen FUNCTION würde durch die folgende Befehlszeile vonstatten gehen:

```
Variable=FunctionName(Parameter1, Parameter2,...)
```

So bestimmen Sie den Rückgabetyp einer FUNCTION

Wie Sie in der oben stehenden Beispiel-FUNCTION-Definitionszeile erkennen können, muß bei FUNCTIONs definiert werden, welcher Art der Rückgabetyp ist, den die Funktion zurückliefern soll. Wie bei Variablen auch, gibt man dazu einfach hinter dem Funktionsnamen ein entsprechendes Typendeklarationszeichen oder hinter der Parameterliste den Typ mit dem Schlüsselwort "As" und dem entsprechenden Typ-Schlüsselwort an. Die Definitionszeile unseres Beispielprogramms kann also in zwei Variationen lauten:

```
FUNCTION WordCount(WortText$) as Integer
FUNCTION WordCount%(WortText$)
```

Übergabe von Variablen als Wert und als Referenz

Sehr wichtig bei der Nutzung der Möglichkeit, FUNCTIONs oder SUBs Variablen übergeben zu können, ist das Wissen um den Unterschied, Variablen als sogenannte Referenz oder als Wert zu übergeben.

Solange nichts anderes gesagt wird, übergeben Sie einer FUNCTION oder SUB eine Variable immer als Referenz. Das bedeutet, daß, wenn Sie innerhalb einer SUB oder FUNCTION eine Variable verändern, die dieser SUB oder FUNCTION übergeben wurde, sich die Variable ebenfalls ändert, die im aufrufenden Code der SUB oder FUNCTION übergeben wurde. Dazu ein Beispiel: Ändern Sie das obige Beispielprogramm wie folgt ab.

Fügen Sie in der SUB "Start" hinter dem PRINT-Befehl eine weitere Befehlszeile ein:

```
PRINT text$
```

Fügen Sie vor der Zeile "End Function" der FUNCTION WORDCOUNT% eine weitere Befehlszeile ein:

```
WortText$="Funktion beendet"
```

Starten Sie im folgenden das Programm, und geben Sie einen Text ein. StarBasic gibt Ihnen die Anzahl der Leerzeichen aus und direkt im Anschluß den Inhalt der Variablen *text$*: "Funktion beendet". Der Inhalt dieser Variablen wurde augenscheinlich verändert, aber nicht durch das Hauptprogramm, sondern durch die FUNCTION WORDCOUNT%. Dadurch, daß der FUNCTION die Variable direkt übergeben wurde, konnte die FUNCTION den Inhalt der Variablen ändern, obwohl sie eigentlich nur im Hauptprogramm gültig war.

Wünschen Sie, daß diese Eigenschaft unterdrückt wird, stellen Sie vor die Übergabevariable beim Aufruf einer SUB oder FUNCTION einfach das Schlüsselwort "ByVal" (By Value, Übergabe durch Wert). Ändern Sie die erste PRINT-Zeile in der SUB "Start" wie folgt ab:

```
print "Dieser Text enthält";WordCount%(ByVal Text$);"Worte"
```

Starten Sie das Programm jetzt, wird genau der Text angezeigt, den Sie zum Überprüfen eingegeben haben; die Unterroutine konnte den Inhalt der Variablen nicht mehr verändern.

Wenn Sie ein neues Modul anlegen, richtet StarBasic automatisch eine SUB namens Main ein. Dieser Name ist lediglich ein Vorgabename und hat nichts mit dem Ablauf oder dem Startpunkt eines StarBasic-Projektes zu tun. Ändern Sie den Namen dieser Vorgabe-SUB nach Ihren Bedürfnissen.

Gültigkeit von Variablen innerhalb von Prozeduren und Modulen und außerhalb von Modulen

Wird eine Variable innerhalb einer SUB oder einer FUNCTION definiert, dann bleibt sie dort nur solange gültig, bis die Prozedur verlassen wird. Danach gibt es die Variable quasi nicht mehr, und sie wird neu "angelegt", wenn die SUB oder FUNCTION abermals benötigt wird. Man sagt, die Variable wurde "lokal" definiert. In vielen Fällen ist es aber notwendig, Variablen zu definieren, so daß sie in allen Prozeduren gültig sind. Teilweise ist es sogar nötig, daß Variablen in allen Modulen aller Bibliotheken gültig sind. Oder aber es wird gewünscht, daß Variablen erhalten bleiben, selbst wenn eine SUB oder FUNCTION verlassen wurde. Diese Eigenschaften können Sie gezielt durch bestimmte Schlüsselworte bei der Deklaration einer Variable durch DIM steuern. Es gelten folgende Richtlinien:

Deklaration von Variablen außerhalb einer SUB oder FUNCTION

```
DIM PUBLIC VarName as TYPENNAME
```

Die Variable hat in allen Modulen Gültigkeit.

```
DIM PRIVATE VarName as TYPENNAME
```

Die Variable hat nur im Modul Gültigkeit.

```
DIM VarName as TYPENNAME
```

Wie vorhergehende Anweisung.

Sichern von Variableninhalten nach Verlassen einer SUB oder FUNCTION

```
DIM STATIC VarName as TYPNAME
```

Die Variable behält ihren Wert bis zum nächsten Betreten der FUNCTION oder SUB. Beachten Sie bitte hierbei, daß die Deklaration selbstverständlich innerhalb einer SUB oder FUNCTION stattfinden muß.

Lesen Sie bitte auch die Ausführungen über die Fehlersuche (Debuggen) in Programmen. Dort finden Sie weitere Demonstrationen zur Gültigkeit von Variablen, insbesondere auch, was mögliche Fehlerquellen durch falsch verwendete Variablen anbelangt.

DLLs von Basic aus aufrufen

Hier wird beschrieben, wie Sie aus StarBasic heraus Funktionen einer DLL aufrufen.

StarBasic erlaubt das Aufrufen von Unterroutinen, die in DLLs abgelegt sind. DLL steht für "Dynamic Link Library" - Programmbibliotheken die nur dann geladen werden, wenn die dort enthaltenen Funktionen gebraucht werden und die sich mehrere Applikationen teilen können. Die Aufrufe von Basic aus gestalten sich dabei genau wie das Aufrufen von "Basic-internen" SUBs oder FUNCTIONs, nur mit dem Unterschied, daß der auszuführende Programmcode sich eben nicht im Basic, sondern in der jeweiligen DLL befindet.

Gerade beim Aufruf von Betriebssystem-Funktionen, sollten Sie sehr vorsichtig und sorgfältig agieren, da beispielsweise eine falsche Anzahl übergebener Parameter an eine Betriebssystem-DLL schon den Absturz des gesamten Systems zur Folge haben kann. Wenn Sie also mit DLL-Aufrufen arbeiten, empfiehlt es sich dringend, Ihr Projekt abzuspeichern, bevor Sie das Programm austesten!

Um die Funktion einer DLL-Datei aus StarBasic heraus zu verwenden, wird die zu verwendende Funktion mit Hilfe des Schlüsselwortes DECLARE am Anfang des Programmcodes eines Moduls wie folgt deklariert:

```
DECLARE {SUB|FUNCTION} SubFuncName LIB "Libname" [ALIAS
"Ersatzname"][BYVAL argumente][AS Type]
```

Parameter	Beschreibung
SubFuncName	Name der Unterroutine innerhalb der DLL. Der Name muß derselbe sein, wie der in der DLL, es sei denn, die "Alias"-Option wird verwendet (s.u.).
Libname	Datei- oder Systemname der Dynamic Link Library. Handelt es sich um keine Systemlibrary (die sowieso immer geladen ist), wird diese Library automatisch geladen, sobald die erste deklarierte Funktion dieser DLL das erste Mal benötigt wird.
Ersatzname	Bietet die Möglichkeit, einen anderen SubFuncNamen als den in der DLL definierten als Funktionsnamen zum Aufruf aus Basic heraus anzugeben. In diesem Falle definieren Sie SubFuncName beliebig und geben mit dem Schlüsselwort "Alias" den in der DLL definierten Funktionsnamen an. Rufen Sie im folgenden die DLL-Funktion mit dem frei definierten SubFuncName-Funktionsnamen auf, so nimmt StarBasic intern eine entsprechende Namenskonvertierung vor.
Argumentliste	Eine Liste mit Parametern, die als Wert (Value) und nicht als Referenz übergeben werden müssen. Dazu dient das Schlüsselwort "ByVal", das festlegt, daß der DLL-Funktion nur der reine Wert und nicht ein Zeiger der Variablen übergeben wird.
Typ	Gilt nur für FUNCTIONs, nicht für SUBs. Wie bei Variablendeklarationen wird definiert, welchen Typs der Rückgabewert sein soll, den die Funktion zurückliefert.

Funktionen, die sich in EXE-Dateien befinden, können nicht aus Basic heraus aufgerufen werden.

Das folgende kleine Beispielprogramm soll den Umgang mit DLLs aus StarBasic unter Windows 95 bzw. Windows NT heraus demonstrieren. Es ermittelt das Windows-System-Verzeichnis und stellt anschließend mit Hilfe der entsprechenden System-Funktionen fest, ob eine Maus installiert ist.

```
Declare Function GetWindowsDirectory Lib "Kernel32" Alias
"GetWindowsDirectoryA" ( ByVal lpBuffer as string, ByVal nSize
as long) as long
Declare Function GetSystemMetrics Lib "user32" alias "GetSy-
stemMetrics" (ByVal nIndex as long) as long
Sub Main
```

```
 dim StrPath as string * 255
 StrPath = String(255,0)
 const SM_MOUSEPRESENT = 19
  StrWorked = GetWindowsDirectory(StrPath, Len(StrPath))
     Print StrWorked
     Print StrPath
  if GetSystemMetrics(SM_MOUSEPRESENT) then Msgbox "Maus
installiert"
End Sub
```

Der Aufruf von DLL-Funktionen gestaltet sich, was die Syntax der DECLARE-Anweisungen anbelangt, unter den Betriebssystemen Windows 16 Bit, Windows 32 Bit und OS/2 gleich. Arbeiten Sie unter Windows 95, Windows NT oder OS/2, können Sie mit Hilfe der Declare-Anweisung ausschließlich 32-Bit-Routinen aufrufen.

Bibliotheken, Module und Dialoge

Hier wird beschrieben, wie Sie Ihre StarBasic-Projekte mit Hilfe von Modulen und Bibliotheken strukturieren.

Damit Sie sich gerade beim Entwickeln von größeren StarBasic-Projekten besser organisieren können, stellt Ihnen StarBasic spezielle Strukturierungswerkzeuge zur Verfügung, mit deren Hilfe Sie Projekte organisieren können.

StarBasic kennt mehrere "Einheiten", in denen sich die einzelnen SUBs und FUNCTIONs, aus denen ein Basic-Projekt besteht, zusammenfassen lassen. Diese finden Sie im folgenden beschrieben.

Bibliotheken

Bibliotheken enthalten Module. Sie dienen quasi als zusätzliches, "übergeordnetes" Organisationswerkzeug und fassen die in ihnen enthaltenen Module lediglich zusammen. Bibliotheken können sowohl mit einem Dokument als auch mit einer Dokumentvorlage verbunden werden. Ist eine Bibliothek an eine Vorlage oder ein Dokument gebunden und wird das Dokument bzw. die Dokumentvorlage gespeichert, werden alle sich in der Bibliothek befindlichen Module und damit auch alle SUBs und FUNCTIONs, die in diesen Modulen vorhanden sind, automatisch mitgespeichert.

Rein theoretisch können sich bis zu 16 000 Module in einer Bibliothek befinden; in der Regel wird aber die mögliche Anzahl von Modulen durch den vorhandenen Speicherplatz begrenzt.

Module

Ein Modul enthält SUBs und FUNCTIONs sowie Variablendeklarationen für die Variablen, die entweder im gesamten Modul oder in allen Modulen gültig sein sollen. Die Länge des Programmtextes, der in einem Modul gespeichert werden kann, ist auf 64 KByte beschränkt. Sollte dieser Platz nicht ausreichen, verteilt man ein StarBasic-Projekt einfach auf mehrere Module, die dann zusammen in einer Bibliothek gespeichert werden sollten.

Sie sollten für Einzel-Makros, die Sie in einem Dokument oder für die Applikation benötigen, ein einzelnes Modul reservieren. StarBasic-Projekte, die aus mehreren Makros (da aus mehreren SUBs) bestehen, sollten in einem eigenen Modul, sehr umfangreiche Projekte sogar in einer eigenen Bibliothek gespeichert werden, damit sich richtige Makros, die Sie zum Arbeiten in Ihrem StarOffice-Programm benötigen, nicht mit Ihrem StarBasic-Projekt "vermischen". Wie Sie neue Module bzw. neue Bibliotheken anlegen und sie verwalten, erfahren Sie im Kapitel über die IDE.

Dialogmodule

Dialogmodule fassen den Aufbau einer Dialogbox, die Eigenschaften eines jeden Elementes innerhalb des Dialoges und die Zuordnung bestimmter Ereignisse zu SUBs in einem Modul zusammen. Da in einem Dialogmodul nur ein einziger Dialog gespeichert werden kann, spricht man überlicherweise auch nur vom "Dialog".

Dialoge werden vom Programm-Code aus mit der SHOW-Methode aufgerufen und mit der HIDE-Methode wieder geschlossen. Die Überwachung der kompletten Bearbeitung eines Dialoges durch den Anwender wird ausschließlich von StarBasic übernommen, der Programmierer braucht lediglich bestimmte Ereignisse auszuwerten.

Die Vorgehensweise zur Programmierung von Modulen können Sie unserem Beispielprojekt entnehmen. Eine ausführliche Referenz aller vorhandenen Dialogboxelemente, eine Liste der Eigenschaften, die diese Elemente annehmen und welche Ereignisse diese Elemente auslösen können, finden Sie in der Referenz beschrieben.

Integrierte Entwicklungsumgebung (IDE)

Hier werden die Funktionen der Integrierten Entwicklungsumgebung (englisch: Integrated Development Environment, kurz IDE) beschrieben. Sie erfahren, wie Sie Module oder Bibliotheken verwalten, welche Funktionen

Ihnen der Editor der IDE zur Verfügung stellt, und wie Sie die IDE bei der Fehlersuche von Programmen - dem sogenannten Debugging - unterstützen kann.

Überblick IDE

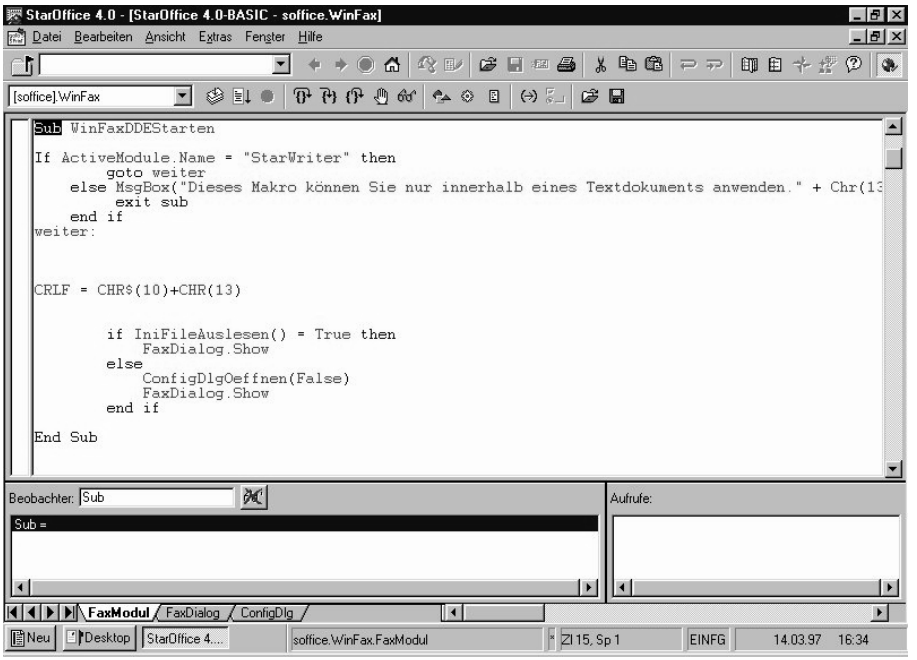
Hier erhalten Sie einen Überblick über die StarBasic-IDE. Sie erreichen die IDE, sobald Sie ein Makro zur Bearbeitung aufgerufen haben.

Die IDE verfügt über eine besondere Objektleiste, die für das Bearbeiten und Testen von Programmen verschiedene Funktionssymbole bereit hält. Die Erklärung dieser Symbole finden Sie bei der Beschreibung der jeweiligen Themen in den folgenden Abschnitten.

Direkt unter der Objektleiste befindet sich das große Editor-Fenster. In diesem Fenster können Sie den Basic-Programm-Code bearbeiten. Links vom Editorfenster erkennen Sie eine kleine Spalte, die Haltepunkte aufnimmt und Ihnen während der Einzelschrittbearbeitung eines Programms anzeigt, welche Programmzeile aktuell abgearbeitet wird.

Unterhalb des Editorfensters auf der linken Seite befindet sich der Beobachter. Mit seiner Hilfe können Sie sich während einer Einzelschrittbearbeitung die Inhalte von Variablen oder Arrays anzeigen lassen. Rechts davon befindet sich das Aufrufe-Fenster. Es gibt Ihnen Aufschluß über die Aufrufhierarchie von SUBs und FUNCTIONs.

Die Möglichkeiten, die Ihnen diese Benutzeroberfläche bei der Entwicklung von Programmen bietet, finden Sie ausführlich in den folgenden Abschnitten beschrieben.



Anlegen und Verwalten von Bibliotheken und Modulen

Hier erfahren Sie, wie Sie Bibliotheken und Module verwalten. Sie lernen, wie Sie sie anlegen und löschen und wie Sie StarBasic-Projekte zwischen Dokumenten, Dokumentvorlagen oder der Applikation verschieben.

Sie sollten gerade für größere StarBasic-Projekte grundsätzlich ein eigenes Modul einrichten und nicht mehrere Projekte in einem Modul speichern. Diese Vorgehensweise wird um so nötiger, als daß Variablen, die für das gesamte Modul gültig sein sollen, wirklich nur projektspezifisch verwendet werden sollten und sich nicht mit denen von anderen Projekten oder aufgezeichneten Makros "vermischen" dürfen. Daher: Legen Sie für jedes neue Projekt wenigstens ein eigenes Modul, am besten sogar eine ganze Bibliothek an. Legen Sie für SUBs, die Sie als Makro aufzeichnen möchten, ebenfalls eine eigene Bibliothek an, damit sich die verschiedenen Komponenten nicht "vermischen".

Verwalten von Bibliotheken

So legen Sie eine neue Bibliothek an

- Sollten Sie sich nicht in der IDE befinden, wählen Sie aus dem Menü EXTRAS den Menüpunkt MAKRO und klicken Sie im Dialog, der jetzt erscheint, auf die Schaltfläche VERWALTEN. Das Programm zeigt Ihnen jetzt einen Dialog, mit dessen Hilfe Sie eine neue Bibliothek anlegen können.
- Sollten Sie sich bereits in der IDE befinden, erreichen Sie den entsprechenden Dialog ebenfalls über das MODULE-Symbol.
- Klicken Sie auf das Register BIBLIOTHEK.

- Wählen Sie aus dem Kombinationsfeld BIBLIOTHEK das Objekt aus, mit dem die Bibliothek verknüpft werden soll. Das kann entweder das StarOffice-Programm selbst sein oder ein beliebiges Dokument oder eine Dokumentvorlage, das oder die jedoch geöffnet sein muß.
- Klicken Sie anschließend auf die Schaltfläche NEU. StarBasic hat jetzt eine neue Bibliothek angelegt.

So benennen Sie eine Bibliothek

- Möchten Sie den Vorgabenamen ändern, verfahren Sie wie folgt:
- Klicken Sie den Vorgabenamen der Bibliothek bei gedrückter (Alt)-Taste an.
- Ein Cursor erscheint hinter dem Vorgabenamen, den Sie jetzt editieren können. Beenden Sie die Neubenennung durch Drücken der Eingabetaste.

So bestimmen Sie, ob eine Bibliothek geladen sein soll oder nicht

Gerade wenn Sie viele Bibliotheken mit einem Dokument oder einer Vorlage oder, was noch häufiger der Fall sein kann, mit der Applikation selbst verbunden haben, kann die Menge an vorhandenen Modulen leicht dazu führen, daß die Übersicht verloren geht. StarBasic erlaubt es daher, jede Bibliothek ein- bzw. auszublenden. Sie finden im oben gezeigten Dialog vor jeder Bibliothek ein Markierfeld. Wenn Sie eine Bibliothek abwählen, wird Sie aus dem Arbeitsspeicher entfernt. Ihnen stehen grundsätzlich nur die Bibliotheken zur Verfügung, vor deren Namen ein angewähltes Markierfeld steht.

So löschen Sie eine Bibliothek

❍ Wählen Sie aus dem Kombinationsfeld BIBLIOTHEK des oben gezeigten Dialogs das Objekt aus, mit dem die Bibliothek verknüpft ist, die gelöscht werden soll. Das kann entweder das StarOffice-Programm selbst sein oder ein beliebiges Dokument oder eine Dokumentvorlage das oder die jedoch geöffnet sein muß.

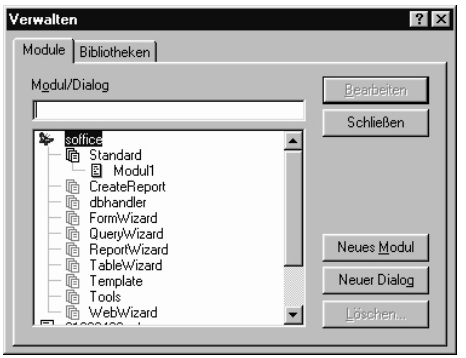
> Bevor Sie den Vorgang vollenden, bedenken Sie bitte: Sobald Sie eine Bibliothek löschen, gehen damit alle in ihr vorhandenen Module und sämtliche SUBs und FUNCTIONs unwiderruflich verloren.

❍ Klicken Sie anschließend auf die Schaltfläche LÖSCHEN und bestätigen Sie die Sicherheitsabfrage mit JA.

Verwalten von Modulen und Dialogen

So legen Sie ein neues Modul oder einen Dialog an

❍ Wählen Sie aus dem Menü EXTRAS den Menüpunkt MAKRO und klicken im Dialog, der jetzt erscheint, auf die Schaltfläche VERWALTEN. Das Programm zeigt Ihnen jetzt einen Dialog, mit dessen Hilfe Sie neue Module oder Dialoge anlegen können. Sollten Sie sich in der IDE befinden, erreichen Sie den entsprechenden Dialog ebenfalls über das MODULE-Symbol.



❍ Klicken Sie auf das Register MODULE und anschließend auf die Schaltfläche NEUES MODUL. StarBasic legt darauf ein neues Modul an. Klicken Sie auf NEUER DIALOG, wird ein neuer, leerer Dialog angelegt.

So benennen Sie ein Modul oder einen Dialog

Möchten Sie den Vorgabenamen ändern, verfahren Sie wie folgt:

❍ Klicken Sie den Vorgabenamen des Moduls oder des Dialogs bei gedrückter (Alt)-Taste an. Ein Cursor erscheint hinter dem Vorgabenamen, den Sie jetzt editieren können.

❍ Beenden Sie die Benennung durch Drücken der Eingabetaste.

So löschen Sie ein Modul oder einen Dialog

❍ Wählen Sie per Mausklick aus dem Listenfeld das Modul oder den Dialog aus, welches gelöscht werden soll.

> Bedenken Sie vor dem Löschen eines Moduls bitte: Sobald Sie ein Modul löschen, gehen damit alle in ihm vorhandenen SUBs und FUNCTIONs unwiderruflich verloren!

❍ Klicken Sie anschließend auf die Schaltfläche LÖSCHEN und bestätigen Sie die Sicherheitsabfrage mit Ja.

Organisieren von Projekten zwischen Dokumenten bzw. Dokumentvorlagen

Wenn Sie ein Programm bzw. Programmprojekt mit StarBasic entwickelt haben, so möchten Sie Teile des Programms bzw. das ganze Makro-Projekt möglicherweise zu anderen Dokumenten oder Dokumentvorlagen verschieben. StarBasic bietet Ihnen diese Möglichkeit.

Bitte beachten Sie, daß ein StarBasic-Programm in der Regel aus mehr als nur einem Makro, sprich: aus nur einer SUB besteht. Sollten Sie also ein Projekt entwickelt haben, das mehrere SUBs und FUNCTIONs verwendet, die sich gegenseitig aufrufen, so sollten Sie darauf achten, daß Sie alle benötigten Elemente mitkopieren, da ansonsten Ihr Programm an ein anderes Dokument oder eine andere Vorlage gebunden nicht mehr lauffähig ist.

So verschieben oder kopieren Sie Module, Dialoge oder Bibliotheken zwischen Dokumenten, Vorlagen und der Applikation

- Öffnen Sie zunächst alle Dokumente bzw. Vorlagen, zwischen denen Sie Objekte (Module, Bibliotheken, Dialoge) verschieben wollen.
- Wählen Sie aus dem Menü EXTRAS den Menüpunkt MAKRO, und klicken Sie im Dialog auf die Schaltfläche VERWALTEN.
- Um ein Objekt an eine andere Stelle (anderes Dokument, Applikation) zu verschieben, halten Sie es durch Drücken der Maustaste fest und ziehen Sie es zur gewünschten neuen Position. Durch eine Linie zeigt Ihnen StarBasic dabei an, zwischen welchen Objekten das verschobene Objekt eingefügt werden soll.

Um ein Objekt zu kopieren, verfahren Sie auf die gleiche Art und Weise; Sie halten dabei lediglich die (Strg)-Taste gedrückt.

So fügen Sie Bibliotheken aus anderen StarOffice-Programmen hinzu

Bibliotheken werden in jeder StarOffice-Applikation erstellt. Sobald Sie gespeichert werden, befinden Sie sich als SBL-Dateien auf dem Datenträger. Sie können eine solche Bibliothek-Datei in Ihr Projekt einfügen.

- Wählen Sie aus dem Menü EXTRAS den Menüpunkt MAKRO und klicken Sie im Dialog auf die Schaltfläche VERWALTEN.
- Klicken Sie auf die Schaltfläche HINZUFÜGEN, und wählen Sie jetzt die SBL-Datei aus, die Sie hinzufügen möchten.

Der Basic-Editor

An dieser Stelle informieren wir Sie über den Basic-Editor.

Der Basic-Editor hält die Standard-Editierungsfunktionen bereit, die Sie auch von einem Textdokument her kennen. Er kennt die Funktionen des BEARBEITEN-Menüs (Ausschneiden, Löschen, Einfügen), die Markierungsmöglichkeiten mit der Umschalttaste sowie alle Cursor-Positionierungsfunktionen (z. B. wortweise springen mit (Strg) und den Cursortasten etc.), wie sie auch vom Arbeiten mit einem Textdokument her bekannt sind.

Navigieren im Projekt

StarBasic hält einige Funktionen bereit, die Sie beim Navigieren gerade in größeren StarBasic-Projekten unterstützen. Die folgenden Abschnitte beschreiben diese Hilfsmittel:

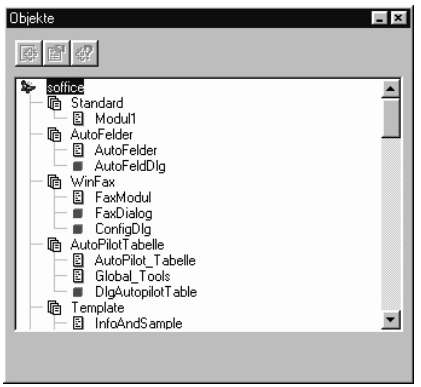
So verwenden Sie das Listenfeld Bibliothek der StarBasic-Objektleiste

Standardmäßig finden Sie auf der linken Seite oberhalb des eigentlichen Editor-Fensters das BIBLIOTHEK-Listenfeld, mit dessen Hilfe Sie durch Anklicken auf den entsprechenden Eintrag in das erste Modul einer Bibliothek wechseln können.

So verwenden Sie den Objektkatalog zum Bewegen im StarBasic-Projekt

Um sich in größeren Projekten zu bewegen, die aus mehreren Bibliotheken und damit in der Regel aus noch mehr Modulen bestehen, verwenden Sie den Navigator. Verfahren Sie dazu wie folgt:

❍ Klicken Sie das OBJEKTKATALOG-Symbol aus der Objektleiste an. StarBasic ruft nun den Objektkatalog auf.



Dieser Objektkatalog hat die Eigenschaft, daß er wie eine Werkzeugleiste auch dann bestehen bleibt, wenn Sie mit der Maus das Programm-Code-Fenster wieder aktivieren.

- Der Dialog zeigt eine Liste aller verfügbaren Objekte. Viele Objekte (Dokumente, Vorlagen, die Applikation selbst und dort enthaltene Bibliotheken und Module) enthalten Unterobjekte, die Sie per Doppelklick ein- und ausblenden können.
- Um ein bestimmtes Modul im Editor anzeigen zu lassen oder den Cursor direkt auf einer bestimmten SUB oder FUNCTION eines Moduls zu positionieren, klicken Sie den entsprechenden Eintrag an und anschließend auf das Symbol ZEIGEN.
- Schließen Sie den Objektkatalog durch Mausklick auf die Schließ-Schaltfläche, wenn Sie ihn nicht mehr benötigen.

Speichern und Laden von Basic-Quelltexten

StarBasic erlaubt es, Basic-Programm-Code für andere Programmiersysteme zu exportieren oder Basic-Programm-Code, der im ASCII-Format vorliegt, aus anderen Programmiersystemen zu übernehmen.

So speichern Sie den Programm-Code eines Modules als ASCII-Text ab

- Rufen Sie mit dem Objektkatalog den Programm-Code des Moduls auf, den Sie als ASCII-Text exportieren möchten.

- Klicken Sie in der IDE-Objektleiste auf das Symbol QUELLTEXT SPEICHERN UNTER. StarBasic öffnet eine Dateiauswahl, in der Sie einen Dateinamen bestimmen können.
- Bestimmen Sie in diesem Dialog Pfad und Dateinamen auf bekannte Art und Weise, und starten Sie den Speichervorgang mit OK.

So fügen Sie ASCII-Programm-Code an der aktuellen Cursorposition ein

- Rufen Sie mit dem Objektkatalog den Programm-Code des Moduls auf, in den Sie Programm-Code als ASCII-Text importieren möchten.
- Positionieren Sie den Cursor an der Stelle, an der Sie den Programm-Code einfügen möchten.
- Klicken Sie in der IDE-Objektleiste auf das Symbol QUELLTEXT EINFÜGEN, und wählen Sie anschließend im Dialog Pfad und Namen der einzufügenden Programm-Code-Datei aus. Starten Sie den Vorgang mit OK.

Fehlersuche

Hier beschreiben wir die erfolgreiche Fehlersuche in StarBasic-Programmen.

Gerade größere Projekte, die Sie mit Hilfe von StarBasic entwickeln, werden nicht von Beginn an fehlerfrei funktionieren. StarBasic bietet zum Finden von Fehlern einige Hilfsmittel, mit denen Sie in Ihren Programmen Fehler finden können. Diese Hilfsmittel finden Sie in den folgenden Abschnitten beschrieben.

Haltepunkte und Einzelschrittbearbeitung

Mit Hilfe der Einzelschrittbearbeitung können Sie ein Programm Befehl für Befehl abarbeiten lassen. Fehler lassen sich auf diese Art und Weise vergleichsweise schnell lokalisieren, da Sie direkt am Bildschirm nachvollziehen können, welchen Weg ein Programmablauf nimmt. Die entsprechende Zeile, die dabei abgearbeitet wird, zeigt die IDE im Editor durch einen Pfeil in der Haltepunktespalte besonders gekennzeichnet an. Möchten Sie, daß ein Programm gezielt an bestimmen Stellen unterbrochen wird, erlaubt Ihnen die IDE dort Haltepunkte zu setzen.

So setzen Sie Haltepunkte vor Programmzeilen

Neben dem Editorfenster sehen Sie eine Spalte, in der Sie per Doppelklick neben eine betroffene Zeile Haltepunkte per Doppelklick hinzufügen oder, falls schon ein Haltepunkt gesetzt wurde, per Doppelklick ebenfalls wieder

löschen können. Trifft ein Programm auf einen Haltepunkt, wird die Programmausführung unterbrochen, und Sie haben mit dem EINZELSCHRITT-Symbol die Möglichkeit, das Programm schrittweise "weiterlaufen" zu lassen.

Trifft das Programm während der Einzelschrittbearbeitung auf einen SUB-Aufruf, so verzweigt es bei der Verwendung des EINZELSCHRITT-Symbols in die SUB. Möchten Sie hingegen, daß ein SUB-Aufruf lediglich durchgeführt, also so, wie ein ganz normaler Basic-Befehl behandelt wird, dann vollziehen Sie die Einzelschritt-Bearbeitung mit dem PROZEDURSCHRITT-Symbol.

So definieren Sie die Eigenschaften eines Haltepunktes

Öffnen Sie das Kontextmenü eines Haltepunktes durch Mausklick mit der rechten Maustaste auf einen Haltepunkt, erlaubt Ihnen die IDE durch einen weiteren Mausklick auf den Menüeintrag AKTIV den Haltepunkt zu aktivieren bzw. zu deaktivieren. Beim Deaktivieren bleibt der Haltepunkt zwar nach wie vor vorhanden, jedoch unterbricht er den Programmablauf nicht mehr. Klicken Sie auf den Eintrag Eigenschaften, öffnet sich ein Dialog, und Sie können dort weitere Optionen bestimmen:

Sie sehen im Listenfeld HALTEPUNKTE eine Liste aller Haltepunkte. Die entsprechenden Nummern geben die Zeilennummern im Programmtext an. Die Zeilennummer, an deren Stelle der Haltepunkt definiert war, bevor Sie den Dialog geöffnet haben, ist in der Liste selektiert. Mit dem Markierfeld AKTIV bestimmen Sie, ob der Haltepunkt weiterhin aktiv sein soll oder nicht.

Im Textfeld DURCHLAUF legen Sie fest, wie oft ein Haltepunkt beim Programmablauf passiert werden muß, bevor der Programmablauf unterbrochen wird. Tragen Sie hier den Wert 0 (das ist die Voreinstellung) ein, wird der Programmablauf grundsätzlich unterbrochen, sobald das Programm auf die Zeile mit dem gesetzten Haltepunkt trifft. Jeder Wert größer als 0 gibt an, wie oft das Programm die entsprechende Stelle passiert haben muß, bis der Programmablauf beim nächsten Erreichen der Zeile mit dem gesetzten Haltepunkt unterbrochen wird.

Klicken Sie auf LÖSCHEN, können Sie in diesem Dialog ebenfalls den Haltepunkt ganz entfernen.

Variablen-Beobachtung

Während einer Einzelschrittbearbeitung haben Sie die Möglichkeit, sich bestimmte Variablen im Beobachter-Fenster anzeigen zu lassen. Um eine Variable in das Beobachter-Fenster aufzunehmen, führen Sie entweder im Programmtext selbst einen Doppelklick auf den Variablennamen aus und klicken anschließend auf das Symbol BEOBACHTER ANZEIGEN, oder Sie geben den Variablennamen direkt in das Beobachter-Eingabefeld ein und schließen die Eingabe durch Drücken der Eingabetaste ab.

Die Werte der Variablen, die in der aktuellen SUB oder FUNCTION gültig sind - also die Variablen, die dort lokal deklariert sind und alle global deklarierten Variablen - werden im Beobachter-Fenster angezeigt. Werden Variablen im Beobachter-Fenster angezeigt, die zur Zeit keine Gültigkeit haben, da Sie für die SUB bzw. FUNCTION, die derzeit abgearbeitet wird, nicht lokal definiert sind, wird anstelle des Wertes nur der Hinweis "Out of Scope" (sinngemäß: "momentan nicht erreichbar") ausgegeben.

Sie können auch Arrays zur Beobachtung aufnehmen. Geben Sie direkt in die Beobachter-Eingabezeile den Namen der Array-Variablen ohne Index an, so wird der Inhalt des gesamten Array angezeigt.

Klicken Sie bei gedrückter (Alt)-Taste auf eine Variable, können Sie deren Wert zur Laufzeit zu Testzwecken verändern.

Fahren Sie mit dem Mauszeiger im Beobachter-Fenster zur Laufzeit über eine gültige (und schon definierte Variable), zeigt Ihnen die IDE den Inhalt der Variablen wie eine Tip-Hilfe an, sobald Sie einen Moment über der Variablen verharren.

Arbeiten mit dem Aufrufe-Fenster

Das Aufrufe-Fenster ist bei der Einzelschrittbearbeitung grundsätzlich aktiv. Es informiert Sie über die "Aufruf"-Hierarchie von SUBs bzw. FUNCTIONs. Sie können durch die Reihenfolge der SUB- und FUNCTION-Anordnung feststellen, welche SUB welche nächste SUB aufgerufen hat, und damit gleichzeitig erkennen, zu welcher SUB das Programm zurückspringt, wenn die aktuelle SUB beendet wurde.

Weitere Möglichkeiten

Hier erfahren Sie, welche weiteren Möglichkeiten Ihnen StarOffice bietet.

Zuweisung von Makros an bestimmte Programmereignisse

Jede StarOffice-Applikation erlaubt es, ein Makro und damit eine SUB automatisch ausführen zu lassen, wenn ein bestimmtes Ereignis eintritt. Dazu müssen Sie dem Ereignis lediglich das gewünschte Makro zuweisen. Die möglichen Ereignisse sind dabei in allen StarOffice-Programmen dieselben. Die folgende Tabelle gibt Ihnen Aufschluß darüber, welche Programmereignisse es gibt und wann die von Ihnen zugeordneten Makros aufgerufen werden.

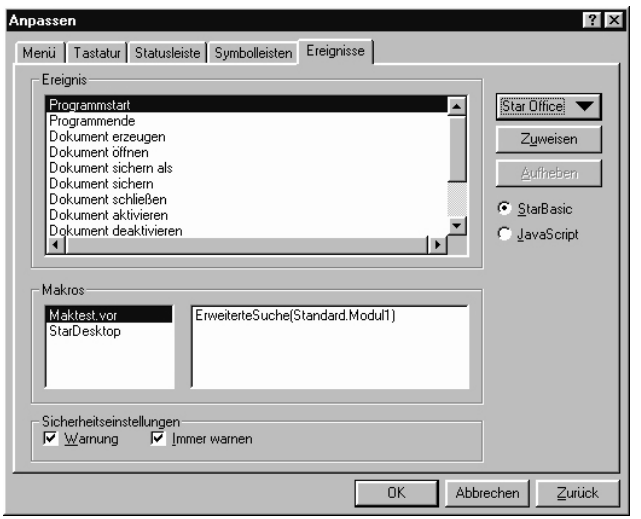
Ereignisname	Makro (SUB) wird automatisch aufgerufen,...
Programmstart	...direkt nachdem das StarOffice-Programm startet.
Programmende	...kurz bevor das StarOffice-Programm beendet wird.
Dokument erzeugen	...direkt nachdem ein neues Dokument mit DATEI - NEU oder dem Symbol NEU erstellt wurde.
Dokument öffnen	...direkt nachdem ein Dokument mit DATEI - ÖFFNEN oder dem Symbol ÖFFNEN geöffnet wurde.
Dokument sichern als	...direkt bevor ein Dokument unter einem bestimmten Namen gesichert wurde (entweder mit DATEI - SPEICHERN UNTER oder mit DATEI - SPEICHERN oder dem SPEICHERN-Symbol, wenn zuvor noch kein Dokumentenname bestimmt wurde).
Dokument sichern	...direkt bevor ein Dokument mit DATEI - SPEICHERN oder mit dem SPEICHERN-Symbol gesichert wurde, sofern zuvor schon ein Dokumentenname für das Dokument bestimmt wurde.
Dokument schließen	...direkt nachdem ein Dokument geschlossen wurde. ACHTUNG: Wurde das Dokument seit der letzten Änderung nicht gespeichert, und der Anwender bestätigt die Speicher-Abfrage mit JA, wird automatisch das Ereignis "Dokument sichern" zusätzlich ausgelöst.
Dokument aktivieren	...direkt nachdem ein Dokument entweder mit dem entsprechenden Eintrag im Fenster-Menü oder mit der Tastenkombination (Strg) (Tab) in den Vordergrund geholt wurde.

Ereignisname	Makro (SUB) wird automatisch aufgerufen,...
Dokument deaktivieren	...direkt nachdem ein anderes Dokument entweder mit dem entsprechenden Eintrag im Fenster-Menü oder mit der Tastenkombination (Strg) (Tab) in den Vordergrund geholt wurde.
Dokument drucken	...direkt nachdem der DRUCKEN-Dialog beendet wurde aber noch vor dem eigentlichen Druckprozeß.
Serienbrief drucken	...direkt nachdem der DRUCKEN-Dialog beendet wurde aber noch vor dem eigentlichen Druckprozeß. Dieses Ereignis tritt bei jedem Exemplardruck auf.

Ein Makro wird, sofern es mit einem Ereignis verknüpft wurde, ebenfalls automatisch ausgeführt, wenn das entsprechende Ereignis durch ein gestartetes Makro ausgelöst wurde!

So verknüpfen Sie ein Ereignis mit einem Makro

❍ Wählen Sie im Menü EXTRAS den Menüpunkt ANPASSEN und dort das Register EREIGNISSE.



❍ Wählen Sie im Listenfeld EREIGNIS den Ereignistyp aus.

❍ Bestimmen Sie im linken Listenfeld des Bereichs MAKROS, in welchem Objekt (geladene Dokumente oder Dokumentvorlagen oder die Applikation selbst) das zuzuweisende Makro zu finden ist.

❍ Wählen Sie aus dem rechten Listenfeld des Bereichs MAKRO den Namen des Makros (bzw. der SUB) aus.

❍ Bestätigen Sie Ihre Auswahl mit OK.

So heben Sie eine Ereignisverknüpfung wieder auf

❍ Wählen Sie im oben gezeigten Dialog das Ereignis aus, dessen Verknüpfung Sie aufheben möchten, und klicken Sie anschließend auf AUFHEBEN.

Verknüpfung von Makros mit Symbolleisten, Tastatur oder Pulldown-Menüs

Jede StarOffice-Applikation erlaubt Ihnen, ein Makro und damit jedes StarBasic-Projekt auf bestimmte Symbolleisten zu legen, Tastaturkürzel zum Aufruf zu definieren oder auch ein Makro (Projekt) mit einem Pulldown-Menüeintrag zu starten. Wählen Sie zur Zuordnung das Menü EXTRAS - ANPASSEN und dort das entsprechende Register, mit dem Sie alle gewünschten Zuordnungen vornehmen können.

Import von Makros aus StarBasic 3.1

Die Makrofunktionalität wurde erweitert und überarbeitet, damit die neuen Funktionalitäten der neuen StarOffice-Version auch genutzt werden können. Die alten Makros aus der Vorgänger-Version können im neuen StarOffice importiert werden.

Hierzu gehen Sie bitte im neuen StarOffice in den Menüpunkt EXTRAS - MAKRO und klicken Sie im Dialog auf VERWALTEN, danach wählen Sie BIBLIOTHEKEN. Nun klicken Sie auf die Schaltfläche HINZUFÜGEN und wählen den Dateityp "Basic". Wechseln Sie jetzt innerhalb des Dialogs in das Basic-Verzeichnis des StarOffice 3.1. Dort werden Ihnen nun die Bibliotheken angezeigt. Klicken Sie eine der Bibliotheken an und in einem weiteren Dialog werden Ihnen jetzt alle Module angezeigt, die Sie entweder anwählen (=übernehmen) oder abwählen können. Dann wird diese Selektion mit einem Klick auf "OK" durchgeführt, und die Module werden direkt in die Struktur des neuen StarOffice übernommen und stehen dort zur Verfügung.

Runtimefunktionen

Hier erhalten Sie Informationen über sämtliche Runtimefunktionen, die Ihnen für StarOffice-Anwendungen zur Verfügung stehen.

Bildschirmein-/ausgabefunktionen

Hier sind alle Funktionen aufgeführt, die Dialoge zum Einlesen und Auswerten von Benutzereingaben aufrufen.

Bildschirmausgabefunktionen

Hier finden Sie alle Bildschirmausgabefunktionen.

MsgBox-Anweisung

Stellt einen Dialog auf dem Bildschirm dar, der einen Hinweistext enthält.

SYNTAX:
MsgBox text As String [,typ As Integer [,dialogtitle As String]] (as Statement) or MsgBox (text As String [,typ As Integer [,dialogtitle As String]]) (as Function)

PARAMETER:
text : Zeichenkettenausdruck, der als Meldung im Dialog gezeigt wird. Zeilenumbrüche werden durch Einfügen von Chr$(13) in den Zeichenkettenausdruck realisiert. Nur die ersten 255 Zeichen einer Zeichenkette werden dargestellt, alle weiteren Zeichen bleiben unberücksichtigt.

dialogtitle : Zeichenkettenausdruck, der die Titelzeile des Dialogs bestimmt. Wird DIALOGTITLE ausgelassen, steht in der Titelzeile der Name der jeweiligen Applikation.

typ : Beliebiger Integer-Ausdruck, der den Typ des Dialogs bestimmt und ferner definiert, mit welchem Symbol und mit welcher Art Schaltfläche(n) der Dialog ausgestattet sein soll. TYP stellt dabei eine Kombination von Bitmustern dar, d. h. Sie bestimmen die Elemente des Dialogs durch die Addition der verschiedenen Werte der folgenden Werte der folgenden Übersicht:

0 : Nur OK-Schaltfläche wird dargestellt.

1 : OK- und Abbrechen-Schaltfläche werden dargestellt.

2 : Abbrechen- und Wiederholen-Schaltfläche werden dargestellt.

3 : Ja-, Nein- und Abbrechen-Schaltfläche werden dargestellt.

4 : Ja- und Nein-Schaltfläche werden dargestellt.

5 : Wiederholen- und Abbrechen-Schaltfläche werden dargestellt.

16 : Das Stop-Symbol wird mit in den Dialog aufgenommen.

32 : Das Fragezeichen-Symbol wird mit in den Dialog aufgenommen.

48 : Das Erklärungs-Symbol wird mit in den Dialog aufgenommen.

64 : Das Informations-Symbol wird mit in den Dialog aufgenommen.

0 : Erste Schaltfläche des Dialogs ist voreingestellt.

256 : Zweite Schaltfläche des Dialogs ist voreingestellt.

512 : Dritte Schaltfläche des Dialogs ist voreingestellt.

BEISPIEL:

```
Sub ExampleMsgBox
   MsgBox("Es trat ein unerwarteter Fehler auf." + Chr(13) + _
   "Es wird jedoch mit dem Programmablauf
fortgefahren.",16,"Fehler"
End sub
```

MsgBox-Funktion

Stellt einen Dialog auf dem Bildschirm dar, der einen Hinweistext enthält. Als Funktion verwendet, kann der Anwender diesen Dialog zusätzlich noch mit einer vom Programmierer ermittelbaren Schaltfläche bestätigen.

SYNTAX:
MsgBox (text As String [,typ As Integer [,dialogtitle As String]])

RÜCKGABE:
Integer

PARAMETER:
text : Zeichenkettenausdruck, der als Meldung im Dialog gezeigt wird. Zeilenumbrüche werden durch Einfügen von Chr$(13) in den Zeichenkettenausdruck realisiert. Nur die ersten 255 Zeichen einer Zeichenkette werden dargestellt, alle weiteren Zeichen bleiben unberücksichtigt.

dialogtitle : Zeichenkettenausdruck, der die Titelzeile des Dialogs bestimmt. Wird DIALOGTITLE ausgelassen, steht in der Titelzeile der Name der jeweiligen Applikation.

typ : Beliebiger Integer-Ausdruck, der den Typ des Dialogs bestimmt und ferner definiert, mit welchem Symbol und mit welcher Art Schaltfläche(n) der Dialog ausgestattet sein soll. TYP stellt dabei eine Kombination von Bit-

mustern dar, d. h. Sie bestimmen die Elemente des Dialogs durch die Addition der verschiedenen Werte der folgenden Übersicht:

WERTE

0 : Nur OK-Schaltfläche wird dargestellt.

1 : OK- und Abbrechen-Schaltfläche werden dargestellt.

2 : Abbrechen- und Wiederholen-Schaltfläche werden dargestellt.

3 : Ja-, Nein- und Abbrechen-Schaltfläche werden dargestellt.

4 : Ja- und Nein-Schaltfläche werden dargestellt.

5 : Wiederholen- und Abbrechen-Schaltfläche werden dargestellt.

16 : Das Stop-Symbol wird mit in den Dialog aufgenommen.

32 : Das Fragezeichen-Symbol wird mit in den Dialog aufgenommen.

48 : Das Erklärungs-Symbol wird mit in den Dialog aufgenommen.

64 : Das Informations-Symbol wird mit in den Dialog aufgenommen.

0 : Erste Schaltfläche des Dialogs ist voreingestellt.

256 : Zweite Schaltfläche des Dialogs ist voreingestellt.

512 : Dritte Schaltfläche des Dialogs ist voreingestellt.

RÜCKGABEWERT:
1 : OK

2 : Abbrechen

4 : Wiederholen

5 : Ignorieren

6 : Ja

7 : Nein

Wird MsgBox als Funktion aufgerufen, wird die Nummer des Buttons zurückgeliefert, der für das Abbrechen des Dialoges sorgte. Dabei gelten folgende Konventionen:

RÜCKGABEWERT
1 : OK

2 : Abbrechen

4 : Wiederholen

5 : Ignorieren

6 : Ja

7 : Nein

BEISPIEL:

```
Sub ExampleMsgBox
Dim sVar as Integer
sVar = MsgBox("Star Division")
sVar = MsgBox("Star Division",1)
sVar = MsgBox( "Star Division",256 + 16 + 2,"Titel im Dialog")
end sub
```

Print-Anweisung

Gibt die angegebenen Zeichenfolgen oder numerischen Ausdrücke in einem Dialog auf dem Bildschirm aus.

SYNTAX:
Print Expression1[{;|,} [Spc(Number As Integer);] [Tab(pos As Integer);] [Expression2[...]]

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenfolgeausdruck, der ausgegeben werden soll. Werden mehrere Ausdrücke hintereinander angegeben, so werden sie bei der Ausgabe nebeneinander gesetzt, sofern sie durch ein Semikolon getrennt wurden. Wurden sie durch ein Komma getrennt, werden die einzelnen Ausdrücke jeweils zur nächsten Tabulatorposition gerückt. Die Tabulatorpositionen sind nicht einstellbar.

Number : Anzahl von Leerzeichen, die durch die Spc-Funktion eingefügt werden sollen.

pos : Es werden soviele Leerzeichen eingefügt, bis die angegebene Position erreicht ist.

Steht hinter dem letzten auszugebenden Ausdruck entweder ein Semikolon oder ein Komma, dann merkt sich StarBasic zunächst den auszugebenden Text in einem internen Speicher und fährt mit der Ausführung der folgenden Anweisung fort, ohne daß etwas gedruckt wird. Erst wenn ein Print-Befehl erreicht wird, der mit keinem der beiden Zeichen endet, werden alle bis dorthin "gemerkten" Ausdrücke dargestellt.

Bei der Ausgabe von positiven numerischen Ausdrücken wird ein Leerzeichen quasi als imaginäres Plus-Zeichen vorangestellt. Negative Ausdrücke werden mit vorgestelltem Minuszeichen ausgegeben. Wird bei Fließkomma-

werten ein bestimmter Bereich überschritten, wird der jeweilige numerische Ausdruck in einer Exponential-Notation ausgegeben.

Sollte der auszugebende Ausdruck eine bestimmte Länge überschreiten, wird die Darstellung automatisch in die nächste Zeile umbrochen.

Sie können die Tab-Funktion mit Semikolon getrennt zwischen Argumente stellen, um die Ausgabeposition bis zu einer bestimmten Stelle zu rükken, oder die SPC-Funktion verwenden, um eine entsprechende Anzahl von Leerzeichen einzufügen.

BEISPIEL:

```
Sub ExamplePrint
Print "ABC"
Print "ABC","123"
end Sub
```

Bildschirmeingabefunktionen

Hier finden Sie alle Bildschirmeingabefunktionen.

InputBox-Funktion

Zeigt einen Dialog mit einem Eingabefeld und erlaubt dem Benutzer die Eingabe einer Zeichenfolge bis zur Bestätigung von OK. Die Eingabe wird in einer Variablen aufgenommen.

Die INPUTBOX-Anweisung bietet eine komfortable Möglichkeit der Eingabe über einen Dialog. Die Eingabeebene kann durch die Eingabetaste oder die Schaltfläche OK verlassen werden; die vom Anwender bis dahin eingegebene Zeichenfolge wird als Funktionsergebnis zurückgegeben. Wird der Dialog hingegen mit Abbrechen verlassen, liefert INPUTBOX eine Nullzeichenfolge ("") zurück.

SYNTAX:
InputBox (Msg As String[, Title As String[, Default As String[, x_pos As Integer, y_pos As Integer]]])

RÜCKGABE:
String

PARAMETER:
Msg : Zeichenfolgeausdruck, der den im Dialog anzuzeigenden Leittext enthält.

Title : Zeichenfolgeausdruck, der in der Titelleiste des Dialogs erscheinen soll.

Default : Zeichenfolgeausdurck, der im Eingabefeld des Dialogs als Voreinstellung angezeigt wird.

x_pos : Integer-Ausdruck, der die horizontale Position des Dialogs angibt. Die Position ist eine absolute Koordinate und bezieht sich nicht auf das Fenster der Office-Applikation.

y_pos : Integer-Ausdruck, der die vertikale Position des Dialogs angibt. Die Position ist eine absolute Koordinate und bezieht sich nicht auf das Fenster der Office-Applikation.

Ohne Angabe von X_POS und Y_POS erfolgt eine Zentrierung des Dialogs. Als Maßeinheit wird Twips verwendet.

BEISPIEL:

```
Sub ExampleInputBox
Dim sText As String
sText  = InputBox ("Bitte geben einen Gedanken ein: ","Lieber
Benutzer")
MsgBox ( sText , 64, "Bestätigung des Gedankens")
End Sub
```

Farbberechnungsfunktionen

Hier finden Sie alle Farbberechnungsfunktionen.

Blue-Funktion

Ermittelt den Blau-Anteil des angegebenen Farbcodes.

SYNTAX:
Blue (Color As Long)

RÜCKGABE:
Integer

PARAMETER:
Color: Long-Integer-Ausdruck, der einen beliebigen Farbcode angibt, dessen Blau-Anteil ermittelt werden soll.

BEISPIEL:

```
Sub ExampleColor
Dim lVar As Long
lVar = rgb(128,0,200)
MsgBox "Die Farbe " & lVar & " hat die Anteile:" & Chr(13) &_
"rot = " &  Red(lVar) & Chr(13)&_
"grün= " &  Green(lVar) & Chr(13)&_
"blau= " &  Blue(lVar) & Chr(13) , 64,"Farben"
End Sub
```

Green-Funktion

Ermittelt den Grün-Anteil des angegebenen Farbcodes.

SYNTAX:
Green (Color As Long)

RÜCKGABE:
Integer

PARAMETER:
Color: Long-Integer-Ausdruck, der einen beliebigen Farbcode angibt, dessen Grün-Anteil ermittelt werden soll.

BEISPIEL:

```
Sub ExampleColor
Dim lVar As Long
lVar = rgb(128,0,200)
msgbox "Die Farbe " & lVar & " hat die Anteile:" & Chr(13) &_
"rot = " &  red(lVar) & Chr(13)&_
"grün= " &  green(lVar) & Chr(13)&_
"blau= " &  blue(lVar) & Chr(13) , 64,"Farben"
end sub
```

Red-Funktion

Ermittelt den Rot-Anteil des angegebenen Farbcodes.

SYNTAX:
Red (ColorNumber As Long)

RÜCKGABE:
Integer

PARAMETER:
ColorNumber: Long-Integer-Ausdruck, der einen beliebigen Farbcode angibt, dessen Rot-Anteil ermittelt werden soll.

BEISPIEL:

```
Sub ExampleColor
Dim lVar As Long
lVar = rgb(128,0,200)
msgbox "Die Farbe " & lVar & " hat die Anteile:" & Chr(13) &_
"rot= " &  red(lVar) & Chr(13)&_
"grün= " &  green(lVar) & Chr(13)&_
"blau= " &  blue(lVar) & Chr(13) , 64,"Farben"
end sub
```

QBColor-Funktion

Gibt den RGB-Farbcode der Farbe zurück, die der Funktion als Farbnummer eines älteren MS-DOS-Programmiersystems übergeben wurde.

SYNTAX:
QBColor (ColorNumber As Integer)

RÜCKGABE:
Long

PARAMETER:
ColorNumber: Beliebiger Integerausdruck, der die Farbnummer der Farbe angibt, die in älteren MS-DOS-Programmiersystemen zur Farbauszeichnung verwendet wurde.

COLORNUMBER können die folgenden Werte zugewiesen werden:

0 : Schwarz

1 : Blau

2 : Grün

3 : Cyan

4 : Rot

5 : Magenta

6 : Gelb

7 : Weiß

8 : Grau

9 : Hellblau

10 : Hellgrün

11 : Hellcyan

12 : Hellrot

13 : Hellmagenta

14 : Hellgelb

15 : Leuchtend weiß

Diese Funktion wird ausschließlich zur Konvertierung früherer MS-DOS BASIC-Applikationen verwendet, die mit den zuvor aufgeführten Farbwerten arbeiten. Die Funktion gibt einen Long-Integer-Wert zurück, der den

Farbcode der angegebenen Farbe angibt, der in der StarOffice-Entwicklungsumgebung zu verwenden ist.

BEISPIEL:

```
Sub ExampleQBColor
Dim iColor As Integer
Dim sText As String
iColor = 7
   sText = "RGB= " & Red(QBColor( iColor) ) & ":" &
Blue(QBColor( iColor) ) & ":"  & Green(QBColor( iColor) )
MsgBox stext,0,"Farbe " & iColor
End Sub
```

RGB-Funktion

Ermittelt einen Long-Integer-Farbwert aus Rot-, Grün- und Blauanteilen.

SYNTAX:
RGB (Red, Green, Blue)

RÜCKGABE:
Long

PARAMETER:
Red: Beliebiger Integer-Ausdruck, der den Rot-Anteil (0-255) der Gesamtfarbe repräsentiert.

Green: Beliebiger Integer-Ausdruck, der den Grün-Anteil (0-255) der Gesamtfarbe repräsentiert.

Blue: Beliebiger Integer-Ausdruck, der den Blau-Anteil (0-255) der Gesamtfarbe repräsentiert.

BEISPIEL:

```
Sub ExampleColor
Dim lVar As Long
lVar = rgb(128,0,200)
msgbox "Die Farbe " & lVar & " hat die Anteile:" & Chr(13) &_
"rot= " &  red(lVar) & Chr(13)&_
"grün= " &  green(lVar) & Chr(13)&_
"blau= " &  blue(lVar) & Chr(13) , 64,"Farben"
end sub
```

Dateiein-/ausgabefunktionen

Mit diesen Funktionen können Sie eigene Datendateien erstellen und verwalten.

Spezielle Funktionen erlauben den Aufbau sogenannter relativer Dateien, in denen Sie Datensätze gezielt speichern und durch direkte Eingabe einer Satznummer wieder laden können. Einige Basisdateifunktionen helfen bei der Organisation von Dateien. Sie können mit diesen Funktionen beispielsweise die Größe einer Datei, den aktuell eingestellten Pfad oder alle Dateien und Unterverzeichnisse eines bestimmten Verzeichnisses ermitteln. Auch läßt sich herausfinden, wann eine Datei oder ein Verzeichnis erstellt oder das letzte Mal geändert wurde.

Öffnen und Schließen von Dateien

Hier finden Sie alle Funktionen und Anweisungen für das Öffnen und Schließen von Dateien.

Close-Anweisung

Schließt eine Datei, die mit der Open-Anweisung geöffnet wurde.

SYNTAX:
Close FileNumber As Integer[, FileNumber2 As Integer[,...]]

PARAMETER:
FileNumber : Beliebiger Integer-Ausdruck, der die Nummer des Datenkanals beschreibt, der zuvor mit der OPEN-Anweisung geöffnet wurde.

BEISPIEL:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("switer4")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
```

```
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

FreeFile-Funktion

Ermittelt die nächste freie Dateinummer zum Öffnen einer Datei. Das ist sinnvoll, wenn Sie bereits mehrere Dateien geöffnet haben, eine weitere Datei öffnen möchten und nicht sicher sind, welche Dateinummer noch nicht belegt ist.

SYNTAX:
FreeFile

RÜCKGABE:
Integer

PARAMETER:
Diese Funktion macht natürlich nur dann Sinn, wenn sie direkt vor einer Open-Anweisung aufgerufen wird. FreeFile ermittelt nur die nächste Dateikanalnummer, reserviert sie aber nicht.

BEISPIEL:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("switer4")
wait 1
iNumber = Freefile
Open aFile For Input As #iNumber
While not eof(#iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

Open-Anweisung

Öffnet einen Datenkanal.

SYNTAX:
Open FileName As String [For Modus] [Access IOMode] [protected] As [#]FileName As Integer [Len = Datasetlength]

PARAMETER:
FileName: Name und Pfad der Datei, die geöffnet werden soll. Sollte die Datei nicht existieren, erscheint eine Fehlermeldung, wenn sie zum Lesen (Access = Read) geöffnet wird. Wird sie dagegen zum Schreiben (Access = Write) geöffnet, so wird die Datei neu erstellt.

Modus : Schlüsselwort, das den Dateimodus beschreibt. Es existieren: Append (sequentielle Datei wird erweitert), Binary (Daten können nur byteweise über Get und Put gelesen werden), Input (Datenkanal wird zum Lesen geöffnet), Output (Datenkanal wird zum Schreiben geöffnet) und Random (relative Dateien werden bearbeitet).

Access : Schlüsselwort, das definiert, auf welche Art der Zugriff erfolgen kann. Erlaubt sind: Read (nur lesen), Write (nur schreiben), Read Write (beides).

Protected : Schlüsselwort, das den Sicherheitsstatus der Datei nach dem Öffnen definiert. Es gelten: Shared (Datei kann auch von anderen Programmen geöffnet werden), Lock Read (Datei wird vor dem Lesen geschützt), Lock Write (Datei wird vor dem Beschreiben geschützt), Lock Read Write(auf die Datei kann gar nicht zugegriffen werden).

FileName : Beliebiger Integer-Ausdruck im Wertebereich von 0-511, der die Nummer eines freien Datenkanals angibt, über den später durch diverse Befehle auf die Datei zugegriffen werden kann. Die Dateinummer kann nicht irgendeine sein, sondern sollte direkt vor der Open-Anweisung mit der FreeFile-Funktion ermittelt werden.

Datasetlength : Bestimmt bei relativen Dateien die Länge eines Datensatzes. Diese Angabe ist nur notwendig, wenn die Datei als "Random-Access" geöffnet wurde.

Eine Datei muß durch die Open-Anweisung geöffnet werden, bevor irgendwelche Dateioperationen, die den Inhalt der Datei beeinflussen, stattfinden können. Beim Öffnen einer Datei kommt es zu einer Fehlermeldung, wenn die Datei bereits geöffnet wurde.

Beispiel:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("switer4")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

Reset-Anweisung

Schließt alle geöffneten Dateien und schreibt den Inhalt aller Dateipuffer auf den Datenträger.

Syntax:
Reset

Beispiel:

```
Sub ExampleReset
On Error Goto ErrorHandler
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Close #iNumber
Documents.Add("sCalc4")
Selection.Insert("Gleich kommt ein Fehler")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
```

```
If sZeile <>"" then
Selection.Text.InsertPara()               REM hier Stimmt der
aufruf nicht
Selection.Text.Insert(Trim$(sZeile)) REM und es kommt zu einem
Laufzeitfehler
end if
wend
Close #iNumber
Exit Sub
ErrorHandler:
Reset
MsgBox "Alle Dateien werden geschlossen",0,"Fehler"
End Sub
```

Ein-/Ausgabe

Hier finden Sie alle Anweisungen für die Eingabe und Ausgabe von Dateien.

Get-Anweisung

Liest einen Datensatz aus einer relativen Datei oder eine Folge von Bytes aus einer binären Datei in eine Variable ein.

Siehe auch: PUT-Anweisung

SYNTAX:
Get [#] FileNumber As Integer, [Position], Variable

PARAMETER:
FileNumber : Beliebiger Integer-Ausdruck, der die Dateinummer bestimmt.

Position : Wenn Dateimodus Random ist: Datensatznummer, an der der Lesevorgang beginnt.

Wenn Dateimodus Binary ist: Byte-Position innerhalb der Datei, an der der Lesevorgang beginnt.

Wird position nicht angegeben, so gilt die aktuelle Position bzw. der aktuelle Datensatz der Datei.

Variable : Name der Variablen, die gelesen werden soll. Erlaubt sind alle Elementartypen inkl. Varianten und Arrays. Object-Variablen können nicht gelesen werden.

BEISPIEL:

```
Sub ExampleRandomAccess
Dim iNumber As Integer
Dim sText As Variant REM muß Variant sein
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
```

```
Open aFile For Random As #iNumber Len=32
Seek #iNumber,1   REM auf Anfang positionieren
Put #iNumber,, "Das ist die 1.Zeile Text"  REM der Reihe nach
mit Text füllen
Put #iNumber,, "Das ist die 2.Zeile Text"
Put #iNumber,, "Das ist die 3.Zeile Text"
Seek #iNumber,2
Get #iNumber,,sText
Print sText
Close #iNumber
iNumber = Freefile
Open aFile For Random As #iNumber Len=32
Get #iNumber,2,sText
Put #iNumber,,"Das ist ein neuer Text"
Get #iNumber,1,sText
Get #iNumber,2,sText
Put #iNumber,20,"Das ist Text in Datensatz 20"
Print Lof(#iNumber)
Close #iNumber
end sub
```

Input#-Anweisung

Erlaubt das Einlesen von Daten aus einer geöffneten sequentiellen Datei.

SYNTAX:
Input #FileNumber As Integer; var1[, var2[, var3[,...]]]

PARAMETER:
FileNumber : Nummer der Datei, aus der die Daten eingelesen werden sollen. Die Datei muß zuvor mit der Open-Anweisung und mit dem Zusatz INPUT geöffnet worden sein.

var : Eine Variable beliebigen Typs, die das Ergebnis des Einlesens aufnimmt. Erlaubt sind hier sowohl numerische Variablen als auch Zeichenfolgevariablen.

Mit Hilfe der INPUT#-Anweisung können Sie numerische Werte oder Zeichenfolgen aus einer geöffneten Datei in eine oder mehrere Variablen einlesen. Eine numerische Variable wird grundsätzlich bis zum ersten Carriage-Return (Asc=13), Line Feed (Asc=10), Leerzeichen oder Komma gelesen. Zeichenfolgevariablen werden bis zum ersten Carriage-Return (Asc=13), Line Feed (Asc=10) oder Komma gelesen.

Die Daten und Datentypen in einer Datei müssen in der Anzahl und Reihenfolge angeordnet sein, die der INPUT#-Anweisung mit der Variablenliste übergeben wurde. Werden den numerischen Variablen nichtnumerische Daten zugewiesen, so erhält die Variable den Wert Null.

Aus der Tatsache, daß Datensätze in der Datei auch durch Kommata abgegrenzt sein können, folgt die logische Konsequenz, daß diese Kommata nicht in eine Zeichenfolgevariable eingelesen werden können. In der Datei enthaltene Anführungszeichen (") werden ebenfalls nicht beachtet. Sollte das Einlesen dieser Zeichen dennoch erforderlich sein, verwenden Sie die LINE INPUT#-Anweisung, um reine Textdateien (solche, die keine Steuerzeichen sondern nur druckbare Zeichen enthalten) zeilenweise nacheinander zu lesen.

Wird während des Lesens eines Datenelements das Ende der Datei erreicht, so wird der Lesevorgang abgebrochen, und es tritt ein Fehler auf.

BEISPIEL:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("switer4")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

Line Input #-Anweisung

Erlaubt das sequentielle Einlesen von Zeichenfolgen aus einer Datei.

SYNTAX:
Line Input #FileNumber As Integer, Var As String

PARAMETER:
FileNumber : Nummer der Datei, aus der die Daten gelesen werden sollen. Die Datei muß zuvor mit der Open-Anweisung und mit dem Zusatz READ geöffnet worden sein.

var : Eine Zeichenfolgevariable, die das Ergebnis des Einlesens aufnimmt.

Mit Hilfe der LINE INPUT#-Anweisung können Sie Zeichenfolgen aus einer geöffneten Datei in eine Variable einlesen. Zeichenfolgevariablen werden bis zum ersten Carriage-Return (Asc=13) oder Line Feed (Asc=10), also zeilenweise gelesen, wobei die Zeilenendenmarken selbst nicht mit in die Zeichenfolge übernommen werden.

BEISPIEL:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("swriter4")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

Put-Anweisung

Schreibt einen Datensatz in eine relative Datei oder eine Folge von Bytes in eine binäre Datei.

Siehe auch: Get-Anweisung

SYNTAX:
Put [#] FileNumber As Integer, [position], Variable

PARAMETER:
FileNumber: Beliebiger Integer-Ausdruck, der bestimmt, in welche Datei die Ausgabe erfolgen soll.

position: Bei relativen Dateien (Random Access-Dateien) die Nummer des Datensatzes, der geschrieben werden soll.

Bei binären Dateien (Binary Access) die Byte-Position innerhalb der Datei, in die geschrieben werden soll.

Variable: Beliebige Variable, deren Inhalt geschrieben werden soll.

Es gilt für relative Dateien: Entspricht der Inhalt dieser Variablen nicht der Länge eines Datensatzes wie er beim Öffnen der Datei durch den LEN-ZUSATZ bei OPEN definiert wurde, so bleiben die in der Datei vorhandenen Daten ab dem Ende der neu geschriebenen Daten bis zum nächsten Datensatz vorhanden; die Startposition beim Schreiben ist garantiert immer ein Datensatzanfang.

Es gilt für binäre Dateien: Der Inhalt der Variablen wird an die Position geschrieben, der Dateizeiger entsprechend auf die hinter dem letzten Byte geschriebene Position gerückt.

BEISPIEL:

```
Sub ExampleRandomAccess
Dim iNumber As Integer
Dim sText As Variant REM muß Variant sein
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Random As #iNumber Len=32
Seek #iNumber,1   REM auf Anfang positionieren
Put #iNumber,, "Das ist die 1.Zeile Text"  REM der Reihe nach
mit Text füllen
Put #iNumber,, "Das ist die 2.Zeile Text"
Put #iNumber,, "Das ist die 3.Zeile Text"
Seek #iNumber,2
Get #iNumber,,sText
Print sText
Close #iNumber
iNumber = Freefile
Open aFile For Random As #iNumber Len=32
Get #iNumber,2,sText
Put #iNumber,,"Das ist ein neuer Text"
Get #iNumber,1,sText
Get #iNumber,2,sText
Put #iNumber,20,"Das ist Text in Datensatz 20"
Print Lof(#iNumber)
Close #iNumber
end sub
```

Write-Anweisung

Schreibt Daten in eine sequentielle Datei.

SYNTAX:
Write [#]FileName, [Expressionlist]

PARAMETER:
FileName : Beliebiger numerischer Ausdruck, der die durch Open festgelegte Dateinummer für die entsprechende Datei enthält.

Expressionlist : Durch Kommata getrennte Variablen oder Ausdrücke, die in eine Datei geschrieben werden sollen.

Wird die Expressionlist nicht mit angegeben, fügt die WRITE#-Anweisung der Datei eine Leerzeile hinzu.

Eine neue Datei, die eine Expressionlist aufnehmen soll oder eine bereits bestehende Datei, der eine Expressionlist hinzugefügt werden soll, muß im Modus OUTPUT oder APPEND geöffnet sein.

Die WRITE#-Anweisung schreibt Daten von Anführungszeichen umschlossen und durch Kommata getrennt in eine Datei. Es ist daher im Umterschied zur PRINT#-Anweisung nicht notwendig, gesonderte Trennzeichen in die Liste einzugeben. Das Ende einer mit der WRITE#-Anweisung erstellten Datei ist durch ein Zeilenendezeichen gekennzeichnet.

BEISPIEL:

```
Sub ExampleWrite
Dim iCount As Integer
Dim sValue As String
iCount = Freefile
open "C:\daten.txt" for OutPut as iCount
sValue = "Hamburg"
Write #iCount,sValue,200
sValue = "New York"
Write #iCount,sValue,300
sValue = "Miami"
Write #iCount,sValue,450
close #iCount
end sub
```

Dateipositionierung

Hier finden Sie alle Funktionen und Anweisungen für die Dateipositionierung.

Eof-Funktion

Stellt fest, ob der Dateizeiger das letzte Byte einer Datei erreicht hat.

SYNTAX:
Eof (intexpression As Integer)

RÜCKGABE:
Bool

PARAMETER:
intexpression: Beliebiger Intergerausdruck, der die Nummer der geöffneten Datei bestimmt.

Sie verwenden Eof, um Fehler zu vermeiden, die bei Leseversuchen hinter dem Dateiende auftreten. Immer wenn Sie mit Hilfe der Input-oder der Get-Anweisung aus einer Datei lesen, wird der Dateizeiger um die Anzahl an Zeichen weitergesetzt, die eingelesen wurden. Erreicht der Dateizeiger das Ende der Datei, liefert Eof als Funktionsergebnis den Wert True (-1) zurück.

BEISPIEL:

```
Sub ExampleWorkWithAFile
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Print #iNumber, "Das ist eine weitere Zeile Text"
Close #iNumber
Documents.Add("switer4")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()
Selection.Text.Insert(Trim$(sZeile))
end if
wend
Close #iNumber
End Sub
```

Loc-Funktion

Gibt Aufschluß über die gegenwärtige Position innerhalb einer geöffneten Datei.

SYNTAX:
Loc (FileNumber)

PARAMETER:
FileNumber: Beliebiger numerischer Ausdruck, der die durch Open festgelegte Dateinummer für die entsprechende Datei enthält.

Wird die Loc-Funktion für eine geöffnete Direktzugriffsdatei verwendet, liefert sie die Nummer des zuletzt gelesenen oder geschriebenen Datensatzes zurück.

In Verbindung mit einer sequentiellen Datei gibt die Loc-Funktion die Position in der Datei an, dividiert durch 128. Für Binärdateien gilt die Position des zuletzt gelesenen oder geschriebenen Bytes.

BEISPIEL:

```
Sub f_Loc
Dim iNumber As Integer
Dim sZ1 As String
Dim Pos1 As Integer
Dim temp As String * 1
iNumber = FreeFile
Open "c:\test\daten.txt" For Input As #iNumber
Do While not EOF(iNumber)
Input #iNumber, Temp
   sZ1 = sZ1 & Temp
   Pos1 = Loc(iNumber)
   Print sZ1 & " " & Pos1
Loop
Close #iNumber
end sub
```

Lof-Funktion

Liefert die Größe einer geöffneten Datei zurück.

SYNTAX:
Lof (FileNumber)

RÜCKGABE:
Long

PARAMETER:
FileNumber : Beliebiger numerischer Ausdruck, der die durch Open festgelegte Dateinummer für die entsprechende Datei enthält.

Möchten Sie die Dateilänge einer nicht geöffneten Datei ermitteln, verwenden Sie die FILELEN-Funktion.

BEISPIEL:

```
Sub ExampleRandomAccess
Dim iNumber As Integer
Dim sText As Variant REM muß Variant sein
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Random As #iNumber Len=32
Seek #iNumber,1   REM auf Anfang positionieren
Put #iNumber,, "Das ist die 1.Zeile Text"  REM der Reihe nach
mit Text füllen
```

```
Put #iNumber,, "Das ist die 2.Zeile Text"
Put #iNumber,, "Das ist die 3.Zeile Text"
Seek #iNumber,2
Get #iNumber,,sText
Print sText
Close #iNumber
iNumber = Freefile
Open aFile For Random As #iNumber Len=32
Get #iNumber,2,sText
Put #iNumber,,"Das ist ein neuer Text"
Get #iNumber,1,sText
Get #iNumber,2,sText
Put #iNumber,20,"Das ist Text in Datensatz 20"
Print Lof(#iNumber)
Close #iNumber
end sub
```

Seek-Funktion

Die Seek-Anweisung setzt die Dateiposition für den nächsten Schreib- oder Lesezugriff, die Seek-Funktion ermittelt diese.

SYNTAX:
Seek (FileNumber)

RÜCKGABE:
Long

PARAMETER:
FileNumber : Beliebiger numerischer Ausdruck, der die durch Open festgelegte Dateinummer für die entsprechende Datei enthält.
position : Position innerhalb der geöffneten Datei, an der der nächste Schreib- oder Lesezugriff durchgeführt werden soll.
Für Direktzugriffsdateien liefert die Seek-Funktion die Nummer des nächsten Datensatzes zurück, der gelesen werden soll oder bestimmt bei Verwendung der Seek-Anweisung die Datensatznummer des nächsten zu schreibenden Datensatzes.

Für alle anderen Dateien gilt die Position des Bytes innerhalb der geöffneten Datei, das als nächstes geschrieben oder gelesen werden soll.

BEISPIEL:

```
Sub ExampleSeek
Dim i As Integer
Dim iNumber As Integer
iNumber = Freefile
Open "c:\Daten.txt" For Random As #iNumber
For i = 1 To 10
PUT #iNumber, , i
Next i
```

```
Seek #iNumber, 2
Get #1, , i
Print "Datum: "; i;"  Nächster Datensatz: "; Seek(iNumber)
end sub
```

Seek-Anweisung

Die Seek-Anweisung setzt die Dateiposition für den nächsten Schreib- oder Lesezugriff, die Seek-Funktion ermittelt diese.

SYNTAX:
Seek [#]FileNumber, position (as Statement)

PARAMETER:
FileNumber : Beliebiger numerischer Ausdruck, der die durch Open festgelegte Dateinummer für die entsprechende Datei enthält.
position : Position innerhalb der geöffneten Datei, an der der nächste Schreib- oder Lesezugriff durchgeführt werden soll.
Für Direktzugriffsdateien liefert die Seek-Funktion die Nummer des nächsten Datensatzes zurück, der gelesen werden soll oder bestimmt bei Verwendung der Seek-Anweisung die Datensatznummer des nächsten zu schreibenden Datensatzes.

Für alle anderen Dateien gilt die Position des Bytes innerhalb der geöffneten Datei, das als nächstes geschrieben oder gelesen werden soll.

Dateiorganisation

Hier finden Sie alle Funktionen und Anweisungen für die Dateiorganisation.

ChDir-Anweisung

Wechselt das aktuelle Verzeichnis oder/und das aktuelle Laufwerk.

SYNTAX:
ChDir Text As String

PARAMETER:
Text : Beliebiger Zeichenkettenausdruck, der den Verzeichnispfad oder/und die Laufwerksangabe angibt.

Möchten Sie lediglich das aktuelle Laufwerk wechseln, müssen Sie den Laufwerksbuchstaben gefolgt von einem Doppelpunkt angeben.

BEISPIEL:

```
Sub ExampleChDir
Dim sDir1 as String , sDir2 as String
sDir1 = "c:\Test"
sDir2 = "d:\privat"
ChDir( sDir1 )
msgbox CurDir
ChDir( sDir2 )
msgbox CurDir
end sub
```

ChDrive-Anweisung

Wechselt das aktuelle Laufwerk.

SYNTAX:
ChDrive Text As String

PARAMETER:
Text : Beliebiger Zeichenkettenausdruck, der den großen Laufwerkbuchstaben des neuen aktuellen Laufwerks enthält.

Die Laufwerksbezeichnung muß mit einem großen Buchstaben übergeben werden und wird durch die mit LASTDRV angegebene Einstellung begrenzt. Wird der Anweisung eine Zeichenfolge mit mehreren Zeichen übergeben, wird nur das erste Zeichen ausgewertet. Versucht der Anwender auf ein Laufwerk zuzugreifen, das nicht vorhanden ist, tritt ein Fehler auf, der mit der ONERROR-ANWEISUNG abgefangen und bearbeitet werden kann.

BEISPIEL:

```
sub ExampleCHDrive
ChDrive "D"  Rem Nur möglich wenn ein Laufwerk REM DREM
vorhanden ist.
end sub
```

CurDir-Funktion

Liefert den auf dem angegebenen oder aktuellen Laufwerk derzeit eingestellten Pfad zurück.

SYNTAX:
CurDir [(Text As String)]

RÜCKGABE:
String

PARAMETER:
Text : Ein beliebiger Zeichenfolgeausdruck, der den Laufwerksbuchstaben

des Laufwerks angibt, auf dem der aktuelle Pfad ermittelt werden soll (beispielsweise "C" für die erste Partition der ersten Festplatte).

Wird das Funktionsargument weggelassen, liefert CurDir das aktuell eingestellte Laufwerk zurück. Werden Laufwerksbeschreibungen mit ungültiger Syntax übergeben oder Laufwerke bestimmt, die entweder nicht vorhanden oder hinter denen liegen, die in der CONFIG.SYS mit der Lastdrive-Anweisung angegeben sind, meldet StarBasic einen Fehler.

Groß- und Kleinschreibung hat keinen Einfluß auf die Funktion.

BEISPIEL:

```
Sub ExampleCurDir
Dim sDir1 as String , sDir2 as String
sDir1 = "c:\Test"
sDir2 = "d:\privat"
ChDir( sDir1 )
msgbox CurDir
ChDir( sDir2 )
msgbox CurDir
end sub
```

Dir-Funktion

Ermittelt den Namen einer Datei oder eines Verzeichnisses oder alle Dateien und Verzeichnisse eines Laufwerkes oder eines Verzeichnisses, auf die der angegebene Suchpfad paßt.

SYNTAX:
Dir [(Text As String) [, Attrib As Integer]]

RÜCKGABE:
String

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck, der nur beim ersten Aufruf der Anweisung angegeben wird. Er muß entweder den Suchpfad oder den Namen des Verzeichnisses oder der Datei enthalten, das oder die es zu überprüfen gilt.

attrib : Beliebiger Integer-Ausdruck, der ein Attributmuster, das bitweise codiert ist, enthält. Geben Sie diesen Attributparameter an, findet die Dir-Funktion nur solche Dateien oder Verzeichnisse, auf die die ausgewählten Attribute zutrafen. Verschiedene Attribute sind durch Addition der Attributwerte miteinander kombinierbar:

ATTRIB:

0 : Normale Dateien.

2 : Versteckte Dateien.

4 : Systemdateien.

8 : Der Name des Volumes wird zurückgeliefert.

16 : Nur Verzeichnisnamen werden zurückgeliefert.

Mit Hilfe dieser Funktion können Sie entweder die Existenz einer Datei oder eines Verzeichnisses überprüfen oder alle Dateien und Verzeichnisse in einem bestimmten Verzeichnis ermitteln.

Möchten Sie die Existenz einer Datei überprüfen, geben Sie den kompletten Pfad zusammen mit dem Dateinamen an. Liefert die Dir-Funktion den Namen zurück, ist das Verzeichnis bzw. die Datei vorhanden. Liefert sie hingegen einen Leerstring ("") zurück, ist das Verzeichnis oder die Datei nicht vorhanden.

Wünschen Sie, daß die Dir-Funktion eine komplette Liste der Dateien ausgibt, die sich innerhalb eines bestimmten Verzeichnisses befinden, verfahren Sie wie folgt: Übergeben Sie beim ersten Mal, wenn Sie die Dir-Funktion aufrufen, den kompletten Suchpfad der Dateien, also beispielsweise "D:\OFFICE4*.SDW". Sie erhalten, sofern Pfadangaben gültig sind und die Suche wenigstens eine Datei zum Ergebnis hatte, den Namen der ersten Datei zurück, die dem Suchpfad entspricht. Rufen Sie den Dir-Befehl im folgenden ohne Parameter auf, liefert er nacheinander alle weiteren Dateien zurück, die dem Suchpfad entsprechen. Wenn die Dir-Funktion einen Leerstring zurückliefert, bedeutet das, daß keine weiteren Dateien mehr vorhanden sind, die dem Suchkriterium entsprechen.

Wenn Sie nur Verzeichnisse ermitteln möchten, nehmen Sie den Attributparameter zu Hilfe. Das gleiche gilt, wenn Sie den Namen eines Volumes (z. B. eine Festplattenpartition) ermitteln möchten.

BEISPIEL:

```
Sub ExampleDirDirectories       REM zeigt die Verzeichnisse
Dim sDatei As String
Dim sPath As String
sPath = "c:\"
sDatei = Dir$(sPath,16)
If sDatei <> "" Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
Sub ExampleDirVolumeName   REM zeigt den Namen des Datenträgers
Dim sDatei as String
```

```
Dim sPath As String
sPath = "c:\"
sDatei = Dir$(sPath,8)
If sDatei <> "" Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
Sub ExampleDirFiles   REM zeigt due NAmen aller Dateien
Dim sDatei as String
Dim sPath As String
sPath = "c:\"
sDatei = Dir$(sPath & "*.*")
If sDatei <> "" Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
Sub ExampleDirAll  REM zeigt alle Dateien und die Verzeichnisse
Dim sDatei as String
Dim sPath As String
sPath = "c:\"
sDatei = Dir$(sPath,22)   REM 16 + 2 + 4
If sDatei <> "." and sDatei <> ".." Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
```

FileAttr-Funktion

Liefert die MS-DOS-Dateikennung oder den Zugriffsmodus einer geöffneten Datei zurück.

SYNTAX:
FileAttr (FileNumber As Integer, Attribut As Integer)

RÜCKGABE:
Integer

PARAMETER:
FileNumber : Die Nummer der geöffneten Datei.

Attribute : Integer-Ausdruck. Erlaubt sind nur die Werte 1 und 2.

WERT

1: Die FileAttr-Funktion liefert die MS-DOS-Dateikennung (OSH, Operating System Handel) zurück.

2 : Damit stellen Sie fest, in welchem Zugriffsmodus sich die Datei befindet. Dabei gelten folgende Werte:

WERT

1 : OUTPUT (Datei zur Ausgabe geöffnet)

2 : INPUT (Datei zur Eingabe geöffnet)

4 : RANDOM (Datei mit wahlfreiem Zugriff)

8 : APPEND (Datei zum Erweitern geöffnet)

32 : BINARY (Datei im Binärzugriff geöffnet).

BEISPIEL:

```
Sub ExampleFileAttr
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
MsgBox FileAttr(#iNumber, 1 ),0,"Dateiattribute"
MsgBox FileAttr(#iNumber, 2 ),0,"Zugriffsmodus"
Close #iNumber
End Sub
```

FileCopy-Anweisung

Kopiert eine Datei.

SYNTAX:
FileCopy TextFrom As String, TextTo As String

PARAMETER:
TextFrom : Beliebiger Zeichenkettenausdruck, der den Dateinamen der zu kopierenden Datei angibt und das Laufwerk sowie den Pfad enthalten kann.

TextTo : Beliebiger Zeichenkettenausdruck, der das Ziellaufwerk und das Zielverzeichnis der zu kopierenden Datei sowie den Namen der Zieldatei angeben kann.

Mit der FileCopy-Anweisung können nur Dateien kopiert werden, die zum Zeitpunkt des Kopierens nicht geöffnet sind.

Beispiel:

```
Sub ExampleFilecopy
Filecopy "c:\autoexec.bat", "c:\Temp\Autoexec.sav"
end sub
```

FileDateTime-Funktion

Liefert eine Zeichenfolge zurück, die Datum und Uhrzeit der Erstellung oder letzten Modifizierung einer Datei enthält.

Syntax:
FileDateTime (Text As String)

Parameter:
Text : Beliebiger Zeichenfolgeausdruck, der den eindeutigen Namen der Datei samt Pfadangabe enthält.

Mit Hilfe dieser Funktion ermitteln Sie den genauen Zeitpunkt der Erstellung oder Modifikation einer Datei. Übergeben Sie den eindeutigen Dateinamen, und Sie erhalten Datum und Uhrzeit in einer Zeichenfolge im Format "TT.MM.JJJJ HH.MM.SS" zurück.

Beispiel:

```
Sub ExampleFileDateTime
msgbox FileDateTime("C:\autoexec.bat")
end sub
```

FileLen-Funktion

Gibt die Länge einer Datei in Bytes zurück.

Syntax:
FileLen (Text As String)

Rückgabe:
Long

Parameter:
Text : Beliebiger Zeichenfolgeausdruck, der den eindeutigen Namen der Datei samt Pfadangabe enthält.

Mit Hilfe dieser Funktion ermitteln Sie die Länge einer Datei. Übergeben Sie den eindeutigen Dateinamen, und Sie erhalten die Dateilänge als Funktionsergebnis zurück. Wird die FileLen-Funktion auf eine geöffnete Datei angewendet, gibt sie die Größe der Datei vor dem Öffnen zurück. Um die Größe einer geöffneten Datei zu ermitteln, sollten Sie die Lof-Funktion verwenden.

BEISPIEL:

```
Sub ExampleFileLen
msgbox FileLen("C:\autoexec.bat")
end sub
```

GetAttr-Funktion

Liefert ein Bitmuster zurück, daß den Dateityp kennzeichnet oder anzeigt, ob es sich um eine Volumebezeichnung oder ein Verzeichnis handelt.

SYNTAX:
GetAttr (Text As String)

RÜCKGABE:
Integer

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck, der den eindeutigen Namen der Datei samt Pfadangabe enthält.

Mit Hilfe dieser Funktion ermitteln Sie die MS-DOS-Dateiattribute einer Datei. Diese Funktion liefert als Funktionsergebnis ein Bitmuster zurück, mit dessen Hilfe Sie die einzelnen Dateiattribute gemäß der folgenden Übersicht feststellen können:

WERT

0 : Normale Dateien.

1 : Nur-Lese-Dateien (Read-Only).

2 : Versteckte Dateien.

4 : Systemdateien.

8 : Der Name des Volumes wird zurückgeliefert.

16 : Nur Verzeichnisnamen werden zurückgeliefert.

32 : Datei hat sich nach dem letzten Back-Up geändert (Archive-Bit).

Möchten Sie wissen, ob ein Bit des Attributbytes gesetzt ist, bedienen Sie sich folgender Abfrage-Methodik:

BEISPIEL:

```
Sub ExampleSetGetAttr
On Error Goto ErrorHandler REM Sprungziel für Fehlerbehandlung
definieren
If Dir("C:\test",16)="" Then MkDir "C:\test"
If Dir("C:\test\autoexec.sav")="" THEN Filecopy
"c:\autoexec.bat", "c:\test\autoexec.sav"
```

```
SetAttr "c:\test\autoexec.sav" ,0
Filecopy "c:\autoexec.bat", "c:\test\autoexec.sav"
SetAttr "c:\test\autoexec.sav" ,1
print GetAttr( "c:\test\autoexec.sav" )
end
ErrorHandler:
Print Error
end
end sub
```

Kill-Anweisung

Löscht eine Datei von einem Datenträger.

SYNTAX:
Kill File As String

PARAMETER:
Datei : Beliebiger Zeichenfolgeausdruck, der den eindeutigen Namen der Datei samt Pfadangabe enthält.

BEISPIEL:

```
sub ExampleKill
Kill "C:\datafile.dat"  REM Datei sollte vorher angelegt werden
end sub
```

MkDir-Anweisung

Erstellt ein Verzeichnis auf einem Datenträger.

SYNTAX:
MkDir Text As String

PARAMETER:
Text : Beliebiger Zeichenkettenausdruck, der den Verzeichnisnamen einschließlich dem Pfad enthält.

Wird der Pfad nicht bestimmt, wird das angegebene Verzeichnis im aktuellen Verzeichnis erzeugt.

BEISPIEL:

```
Sub ExampleMkDir
If Dir("C:\test",16)="" Then MkDir "C:\test"
end sub
```

Name-Anweisung

Gibt einer bereits vorhandenen Datei oder einem bereits vorhandenen Verzeichnis einen neuen Namen.

Syntax:
Name OldName As String As NewName As String

Parameter:
OldName, NewName: Beliebiger Zeichenkettenausdruck, der den Verzeichnisnamen einschließlich dem Pfad enthält.
Wird der Pfad nicht bestimmt, wird das angegebene Verzeichnis im aktuellen Verzeichnis erzeugt.

Beispiel:

```
Sub ExampleName
On Error Goto Fehler
Filecopy "c:\autoexec.bat", "c:\temp\autoexec.sav"
Name "c:\test\autoexec.sav" as "c:\temp\autoexec.bat"
end
Fehler:
if err = 76 then
msgbox "Datei existiert bereits"
end if
end
end sub
```

RmDir-Anweisung

Löscht ein Verzeichnis auf einem Datenträger.

Syntax:
RmDir Text As String

Parameter:
Text : Beliebiger Zeichenkettenausdruck, der den Verzeichnisnamen einschließlich Pfad enthält.

Wird lediglich die Bezeichnung des Verzeichnisses ohne Pfad übergeben, sucht die RmDir-Anweisung das angegebene Verzeichnis im aktuellen Pfad. Ist es dort nicht enthalten, wird eine Fehlermeldung ausgegeben.

Beispiel:

```
Sub ExampleRmDir
MkDir "C:\Test2"
ChDir "C:\test2"
msgbox Curdir
ChDir "\"
RmDir "C:\test2"
end sub
```

SetAttr-Anweisung

Setzt die Dateiattribute der angegebenen Datei.

SYNTAX:
SetAttr FileName As String, Attribute As Integer

PARAMETER:
FileName : Name der Datei einschließlich Pfadangabe, dessen Attribute bestimmt werden sollen. Wird der Pfad nicht angegeben, erwartet die SETATTR -Anweisung die Datei im aktuellen Verzeichnis.

Attribute : Bitmuster, das über die zu setzenden bzw. zu löschenden Attribute Auskunft gibt. Es gilt:

WERT

0 : Normale Dateien.

1 : Nur-Lese-Dateien (Read-Only).

2 : Versteckte Dateien.

4 : Systemdateien.

32 : Datei hat sich nach dem letzten Back-Up geändert (Archive-Bit).

Sie können mehrere Attribute gleichzeitig setzen, indem Sie die entsprechenden Bits der Attribute mit einer logischen ODER-Verknüpfung zusammenfassen.

BEISPIEL:

```
Sub ExampleSetGetAttr
On Error Goto ErrorHandler REM Sprungziel für Fehlerbehandlung
definieren
If Dir("C:\test",16)="" Then MkDir "C:\test"
If Dir("C:\test\autoexec.sav")="" THEN Filecopy
"c:\autoexec.bat", "c:\test\autoexec.sav"
SetAttr "c:\test\autoexec.sav" ,0
Filecopy "c:\autoexec.bat", "c:\test\autoexec.sav"
SetAttr "c:\test\autoexec.sav" ,1
print GetAttr( "c:\test\autoexec.sav" )
end
ErrorHandler:
Print Error
end
end sub
```

Datums- und Zeitfunktionen

Mit diesen Anweisungen und Funktionen können Sie Differenzen zwischen Uhrzeiten und Datumsangaben berechnen. Innerhalb dieser Funktionen gibt es zwei Kategorien, die sich entweder mit Datumsfunktionen oder mit Zeitfunktionen beschäftigen.

Normalerweise können Sie weder mit Zeitfunktionen noch mit Datumsfunktionen Differenzen berechnen. Eine Zeichenfolge wie beispielsweise "12:30:45" von "12:45:21" zu subtrahieren macht einfach keinen Sinn. Sowohl für Zeiten als auch für Datumswerte geht StarBasic hier einen einfachen aber unkonventionellen Weg: Sowohl Zeit- als auch Datumswerte werden in sogenannte fortlaufende numerische Werte umgerechnet, so daß sich ein Zeitpunkt (ein Tag, eine Uhrzeit) als einzelner Wert darstellen läßt. Mit diesen Werten können Sie wie mit normalen Zahlen rechnen. Haben Sie auf diese Art und Weise ein neues Datum oder eine neue Zeit berechnet, können Sie mit Hilfe von speziellen Funktionen die fortlaufenden Zeitzahlen wieder in "verständliche" Zeitangaben zurückwandeln.

Beide Arten von fortlaufenden Zahlenwerten (Datum, Zeit) sind miteinander kombinierbar. Umrechnungen von Datumswerten in eine einzige Zahl liefern grundsätzlich nachkommalose Werte, während Umrechnungen von Zeitwerten vorkommalose Werte ergeben. Dadurch können Datum und Uhrzeit zusammen durch simple Addition in einer einzigen Fließkommazahl gespeichert werden. StarBasic kennt den Variablen-Typ Date, der eine Zeitangabe, die Datum und Uhrzeit enthält, aufnehmen kann.

Umwandeln von Datumswerten

Hier finden Sie alle Funktionen zur Umwandlung von Datumswerten in berechenbare Werte und zurück.

DateSerial-Funktion

Liefert einen Wert vom Typ DATE zurück, der die angegebene Jahres-, Monats- und Tageszahl enthält.

SYNTAX:
DateSerial (year, month, day)

RÜCKGABE:
Date

PARAMETER:
year : Integer-Ausdruck, der das Jahr des Datums bestimmt. Geben Sie die Jahreszahlen zweistellig ein, wenn sie sich im Bereich von 1900 bis 1999 bewegen sollen. Jahreszahlen, die außerhalb dieses Bereichs liegen (möglich sind die Jahre 100-9999 inklusive) müssen grundsätzlich vollständig angegeben werden (45 entspricht also 1945; 1812 müßte natürlich vierstellig angegeben werden).

month : Integer-Ausdruck, der den Monat innerhalb des angegebenen Jahres bestimmt. Der Wertebereich bewegt sich dabei in der Regel zwischen 1 und 12. Werte außerhalb dieses Bereichs sind nicht möglich.

day : Integer-Ausdruck, der den Tag innerhalb des angegebenen Monats definiert. Der Wertebereich bewegt sich dabei in der Regel zwischen 1 und 28, 29, 30 oder 31 (je nach Anzahl der Tage im angegebenen Monat). Werte außerhalb dieses Bereichs sind nicht möglich.

Die DATESERIAL-FUNKTION ermittelt die Anzahl der Tage zwischen dem 30.12.1899 und dem angegebenen Datum. Mit Hilfe der von der Funktion zurückgegebenen Werte können Sie Differenzen zwischen verschiedenen Daten berechnen.

Die Funktion liefert den Datentypen Variant mit VarType 7 (Date) zurück; intern wird dieser Wert jedoch als Wert doppelter Genauigkeit gespeichert. Der 1.1.1900 entspricht dabei dem Wert 2. Negative Werte entsprechen einem Datum, das vor dem 30.12.1899 (nicht inklusive) entspricht. Zurückgelieferte Werte entsprechen also dem Zeitpunkt relativ zum 31.12.1899 in Tagen.

Wird der Funktion ein Datum übergeben, das sich außerhalb des angegebenen Bereichs befindet, meldet StarBasic eine Fehlermeldung.

Der Unterschied zur DATEVALUE-FUNKTION ist, daß Sie dort eine Zeichenfolge als Funktionsargument übergeben, die das Datum beinhaltet. Hingegen übergeben Sie der DATESERIAL-FUNKTION die einzelnen Parameter (Jahr, Monat, Tag) getrennt als numerische Ausdrücke.

BEISPIEL:

```
Sub ExampleDateSerial
Dim lDate as Long
Dim sDate as String
lDate = DateSerial(1964, 4, 9)
sDate = DateSerial(1964, 4, 9)
msgbox lDate REM liefert 23476
msgbox sDate REM liefert 09.04.1964 00:00:00
end sub
```

DateValue-Funktion

Erzeugt aus einer Datumszeichenfolge eine fortlaufende Nummer, die in einem einzigen numerischen Wert ein komplettes Datum darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Daten berechnen.

SYNTAX:
DateValue [(date)]

RÜCKGABE:
Long

PARAMETER:
date : Zeichenfolgeausdruck, der das zu berechnende Datum in einem (fast) beliebigen Format enthält. Im Beispieldatum "12.6.1995" sind die folgenden Darstellungsvarianten erlaubt:

12.06.1995

12.6.95

12.6.1995

12.6, 1995

12.06, 1995

12,06.95

12,6.1995

12. Juni 1995

12. Jun 95

12, Juni 1995

12, Jun 95

12/06/1995

12/Jun/95

Mit Hilfe dieser Funktion können Sie ein Datum zwischen dem 1. Dezember 1582 und dem 31. Dezember 9999 in einen einzigen Integerwert umrechnen, mit dem Sie wiederum Datendifferenzen berechnen können. Wird der Funktion ein Datum übergeben, das sich außerhalb des angegebenen Bereichs befindet, meldet StarBasic eine Fehlermeldung.

Der Unterschied zur DateSerial-Funktion ist, daß Sie dort Jahr, Monat und Tag getrennt als numerische Ausdrücke übergeben. Hingegen übergeben Sie

der DateValue-Funktion einen Zeichenfolgenausdruck im Format "Tag.[,]Monat.[,]Jahr".

BEISPIEL:

```
Sub ExampleDateValue
msgbox DateValue("12/02/1997") REM 12.12.1997 00:00:00
end sub
```

Day-Funktion

Liefert aus einer fortlaufenden Datumszahl, die mit DATESERIAL oder mit DATEVALUE generiert wurde, den Tag des Datums im entsprechenden Monat zurück.

SYNTAX:
Day (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Ganzzahliger numerischer Ausdruck, der die fortlaufende Datumszahl enthält, aus der der Tag im Monat eines Datums ermittelt werden soll.

Diese Funktion stellt quasi teilweise die Umkehrfunktion zur DateSerial-Funktion dar. Sie ermittelt aus der fortlaufenden Datumszahl, die mit DATESERIAL oder mit DATEVALUE generiert wurde, wieder den Tag innerhalb des Monats eines bestimmten Datums. Der Ausdruck

Print Day(DateSerial(1994, 12, 20))
ergibt also den Wert 20.

BEISPIEL:

```
sub ExampleDay
Print "Tag " & Day(DateSerial(1994, 12, 20)) & " des Monats"
end sub
```

Month-Funktion

Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, den Monat des Datums im entsprechenden Jahr zurück.

SYNTAX:
Month (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Ganzzahliger numerischer Ausdruck, der die fortlaufende Datumszahl enthält, aus der der Monat im Jahr eines Datums ermittelt werden soll.

Diese Funktion stellt quasi teilweise die Umkehrfunktion zur DATESERIAL -Funktion dar. Sie ermittelt aus der fortlaufenden Datumszahl, die mit der DATESERIAL-Funktion oder mit der DATEVALUE-Funktion generiert wurde, wieder den Monat innerhalb des Jahres eines bestimmten Datums. Der Ausdruck

Print Month(DateSerial(1994, 12, 20))
ergibt also den Wert 12.

BEISPIEL:

```
Sub ExampleMonth
MsgBox "" & Month(Now) ,64,"Der aktuelle Monat"
End sub
```

WeekDay-Funktion

Liefert aus einer fortlaufenden Datumszahl, die mit mit DateSerial oder mit DateValue generiert wurde, die Nummer des Wochentags zurück.

SYNTAX:
WeekDay (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Ganzzahliger numerischer Ausdruck, der die fortlaufende Datumszahl enthält, aus der der Wochentag (1-7) ermittelt werden soll.
Das folgende Beispielprogramm erlaubt die Eingabe eines Datums und ermittelt anschließend mit Hilfe der WeekDay-Funktion den Wochentag. Der Wochentagname wird anschließend ausgegeben.

BEISPIEL:

```
Sub ExampleWeekDay
Dim sTag As String
REM Wochentag ermitteln und ausgeben
Select Case WeekDay( Now )
case 1
sTag="Sonntag"
case 2
```

```
sTag="Montag"
case 3
sTag="Dienstag"
case 4
sTag="Mittwoch"
case 5
sTag="Donnerstag"
case 6
sTag="Freitag"
case 7
sTag="Samstag"
End Select
msgbox "" + sTag,64,"Heute ist"
End Sub
```

Year-Funktion

Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, die Jahreszahl zurück.

SYNTAX:
Year (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Ganzzahliger numerischer Ausdruck, der die fortlaufende Datumszahl enthält, aus der die Jahreszahl ermittelt werden soll.
Diese Funktion stellt gewissermaßen die Umkehrfunktion zur DATESERIAL -Funktion dar. Sie ermittelt aus der fortlaufenden Datumszahl, die mit der DATESERIAL-Funktion oder mit der DATEVALUE-Funktion generiert wurde, wieder die Jahreszahl. Der Ausdruck

Print Year(DateSerial(1994, 12, 20))
ergibt den Wert 1994.

BEISPIEL:

```
Sub ExampleYear
MsgBox "" & Year(Now) ,64,"Das aktuelle Jahr"
End sub
```

Umwandeln von Uhrzeiten

Hier finden Sie alle Funktionen zur Umwandlung von Uhrzeiten in berechenbare Werte.

Hour-Funktion

Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Stunde der Uhrzeit zurück.

SYNTAX:
Hour (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Numerischer Ausdruck, der die fortlaufende Uhrzeitzahl enthält, aus dem die Stunde berechnet werden soll.

Diese Funktion stellt zum Teil die Umkehrfunktion zur TIMESERIAL-Funktion dar. Sie ermittelt aus der fortlaufenden Uhrzeitzahl, die mit der TimeSerial-Funktion oder mit der TIMEVALUE-Funktion generiert wurde, wieder die Stunde einer bestimmten Uhrzeit zurück. Der Ausdruck

Print Hour(TimeSerial(12:30:41))
ergibt also den Wert 12.

BEISPIEL:

```
Sub ExampleHour
Print "Es ist die Stunde " &  Hour( Now )
End Sub
```

Minute-Funktion

Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Minute der gesamten Uhrzeit zurück.

SYNTAX:
Minute (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Numerischer Ausdruck, der die fortlaufende Uhrzeitzahl enthält, aus dem die Stunde berechnet werden soll.

Diese Funktion stellt quasi teilweise die Umkehrfunktion zur TimeSerial -Funktion dar. Sie ermittelt aus der fortlaufenden Uhrzeitzahl, die mit der TimeSerial-Funktion oder mit der TimeValue-Funktion generiert wurde, wieder die Minute einer bestimmten Uhrzeit zurück. Der Ausdruck

Print Minute(TimeSerial(12:30:41))
ergibt also den Wert 30.

Beispiel:

```
Sub ExampleMinute
MsgBox "Die laufende Minute ist die "& Minute(Now)& "."
end sub
```

Now-Funktion

Gibt das aktuelle Systemdatum und die aktuelle Systemzeit als einen Wert vom Datentyp Date zurück.

Syntax:
Now

Rückgabe:
Date

Beispiel:

```
Sub ExampleNow
msgbox "Es ist " & Now
End sub
```

Second-Funktion

Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Sekunde der gesamten Uhrzeit zurück.

Syntax:
Second (Number)

Rückgabe:
Integer

Parameter:
Number : Numerischer Ausdruck, der die fortlaufende Uhrzeitzahl enthält, aus dem die Sekunde berechnet werden soll.

Diese Funktion stellt quasi teilweise die Umkehrfunktion zur TimeSerial -Funktion dar. Sie ermittelt aus der fortlaufenden Uhrzeitzahl, die mit der TimeSerial-Funktion oder mit der TimeValue-Funktion generiert wurde, wieder die Sekunde einer bestimmten Uhrzeit zurück. Der Ausdruck

Print Second(TimeSerial(12,30,41))
ergibt also den Wert 41.

BEISPIEL:

```
Sub ExampleSecond
MsgBox "Die laufende Sekunde ist "& Second(  Now )
End sub
```

TimeSerial-Funktion

Erzeugt aus der Angabe von Stunde, Minute und Sekunde - Parameter, die als numerische Werte übergeben werden - eine fortlaufende Nummer, die in einem einzigen numerischen Wert eine komplette Uhrzeit darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Uhrzeiten berechnen.

Siehe auch: DateSerial-Funktion, TimeValue-Funktion, DateValue-Funktion.

SYNTAX:
TimeSerial (hour, minute, second)

RÜCKGABE:
Date

PARAMETER:
hour: Beliebiger Integer-Ausdruck, der die Stunde der gesamten Uhrzeit angibt, aus der der fortlaufende Zeitwert ermittelt werden soll. Erlaubt sind hier Werte im Bereich von 0-23.

minute: Beliebiger Integer-Ausdruck, der die Minute der gesamten Uhrzeit angibt, aus der der fortlaufende Zeitwert ermittelt werden soll. Normalerweise geben Sie hier Werte zwischen 0 und 59 an. Werte außerhalb dieses Bereichs sind aber dennoch möglich. In diesem Fall beeinflußt die Minute auch die Stunde.

second: Beliebiger Integer-Ausdruck, der die Sekunde der gesamten Uhrzeit angibt, aus der der fortlaufende Zeitwert ermittelt werden soll. Normalerweise geben Sie hier Werte zwischen 0 und 59 an. Werte außerhalb dieses Bereichs sind aber dennoch möglich. In diesem Fall beeinflußt die Sekunde auch die Minute.

Beispiele:

12, -5, 45 entspricht 11, 55, 45

12, 61, 45 entspricht 13, 2, 45

12, 20, -2 entspricht 12, 19, 58

12, 20, 63 entspricht 12, 21, 4

Mit Hilfe dieser Funktion können Sie eine beliebige Uhrzeit in einen einzigen Wert umrechnen, mit dem Sie wiederum Zeitdifferenzen berechnen können.

Die TimeSerial-Funktion liefert den Typen Variant mit VarType 7 (Date) zurück; intern wird dieser Wert jedoch als Wert doppelter Genauigkeit gespeichert, der Werte zwischen 0 und 0,9999999999 zuläßt. Im Gegensatz zur DateSerial- bzw. DateValue-Funktion, die fortlaufende Zahlen aus Datums-Angaben so errechnen, daß sie Tage relativ zu einem festen Datum ergeben, können Sie zwar mit den Werten rechnen, die die TimeSerial-Funktion zurückliefert, diese aber nicht direkt auswerten.

Der Unterschied zur TimeValue-Funktion ist, daß Sie dort eine Zeichenfolge als Funktionsargument übergeben, die die Uhrzeit beinhaltet. Hingegen übergeben Sie der TimeSerial-Funktion die einzelnen Parameter (Stunde, Minute, Sekunde) getrennt als numerische Ausdrücke.

BEISPIEL:

```
Sub ExampleTimeSerial
Dim dDate As Double, sDate As String
dDate = TimeSerial(8,30,15)
sDate = TimeSerial(8,30,15)
MsgBox dDate,64,"Zeit als Zahl"
MsgBox sDate,64,"Formatierte Zeit"
End Sub
```

TimeValue-Funktion

Erzeugt aus der Angabe von Stunde, Minute und Sekunde - Parameter, die als Zeichenfolge übergeben werden - eine fortlaufende Nummer, die in einem einzigen numerischen Wert eine komplette Uhrzeit darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Uhrzeiten berechnen.

SYNTAX:
TimeValue (Text As String)

RÜCKGABE:
Date

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck, der die zu berechnende Uhrzeit im Format "HH:MM:SS" beschreibt.

Mit Hilfe dieser Funktion können Sie eine beliebige Uhrzeit in einen einzigen Wert umrechnen, mit dem Sie wiederum Zeitdifferenzen berechnen können.

Die TimeValue-Funktion liefert den Typen Variant mit VarType 7 (Date) zurück; intern wird dieser Wert jedoch als Wert doppelter Genauigkeit gespeichert, der Werte zwischen 0 und 0,9999999999 zuläßt.

Im Gegensatz zur DateSerial- bzw. DateValue-Funktion, die fortlaufende Zahlen aus Datums-Angaben so errechnet, daß sich Tage relativ zu einem festen Datum ergeben, können Sie zwar mit den Werten rechnen, die die TimeValue-Funktion zurückliefert, diese aber nicht direkt auswerten.

Der Unterschied zur TimeSerial-Funktion ist, daß Sie dort die einzelnen Parameter (Stunde, Minute, Sekunde) getrennt als numerische Ausdrücke übergeben. Hingegen übergeben Sie der TimeValue-Funktion eine Zeichenfolge als Funktionsargument, die die Uhrzeit beinhaltet.

Beispiel:

```
Sub ExampleTimerValue
Dim daDT as Date
Documents.Add("scalc4")  REM Daten vorbereiten
wait 100
Range("a1").Insert("Arbeitsbeginn")
Range("b1").Insert("Arbeitsende")
Range("c1").Insert("Arbeitszeit")
Range("a2").Insert("8:34")
Range("b2").Insert("18:12")
daDT = TimeValue(  Range("b2").TextValue ) - TimeValue(
Range("a2").TextValue )
Range("c2").Insert(
""+Hour(daDT)+":"+Minute(daDT)+":"+Second(daDT)   )
end sub
```

Systemdatum und -uhrzeit

Hier finden Sie Anweisungen und Funktionen. mit denen Sie das Systemdatum und die Systemuhrzeit ermitteln oder neu bestimmen.

Date-Anweisung

Liefert das aktuelle Systemdatum als Zeichenfolge zurück oder setzt das Datum neu. Das Datumsformat ist abhängig von den länderspezifischen Einstellungen in der Systemsteuerung. Wurde hier "Deutschland" gewählt, wird die Zeit im Format "TT.MM.JJJJ"

Syntax:
Date ; Date = Text As String

PARAMETER:
Text : Wird nur dann benötigt, wenn das Systemdatum neu gesetzt werden soll. In diesem Fall muß der Zeichenfolgeausdruck das Datumsformat erhalten, was aus der länderspezifischen Einstellung der Systemsteuerung hervorgeht. Ist hier "Deutschland" eingestellt, lautet das Format "TT.MM.JJJJ", für den 24. Juli 1969 also beispielsweise "24.07.1969".

BEISPIEL:

```
Sub ExampleDate
msgbox "Heute ist der " & Date
end sub
```

Time-Anweisung

Diese Funktion liefert die aktuelle Systemzeit als Zeichenfolge im Format "HH:MM:SS" zurück.

SYNTAX:
Time

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck, der die neue Uhrzeit im Format "HH:MM:SS" beschreibt.

BEISPIEL:

```
Sub ExampleTime
MsgBox Time,0,"Die Uhrzeit"
end sub
```

Timer-Funktion

Liefert einen Wert zurück, der die vergangene Zeit seit Mitternacht in Sekunden angibt.

Beachten Sie bitte, daß die Variable, mit der die Timer-Funktion aufgerufen wird, zuvor mit dem Datentyp Long deklariert werden muß, da sonst ein Wert vom Datentyp Date zurückgegeben wird.

SYNTAX:
Timer

RÜCKGABE:
Date

BEISPIEL:

```
Sub ExampleTimer
Dim lSec as long,lMin as long,lHour as long
lSec = Timer
MsgBox lSec,0,"Sekunden seit Mitternacht"
lMin = lSec / 60
lSec = lSec Mod 60
lHour = lMin / 60
lMin = lMin Mod 60
MsgBox Right("00" & lHour , 2) & ":"&  Right("00" & lMin , 2) &
":" &  Right("00" & lSec , 2) ,0,"Die berechnete Uhrzeit"
end sub
```

DDE-Anweisungen und Funktionen

Mit diesen Anweisungen und Funktionen können Sie einen Dynamic Data Exchange (DDE) steuern.

DDE befähigt Programme, Daten mit anderen Programmen auszutauschen, die zur selben Zeit aktiv sind. Es lassen sich ebenfalls Programme durch andere Programme "fernsteuern", so sie beide mit diesem Feature ausgestattet sind.

DDEExecute-Anweisung

Sendet einen Befehl oder mehrere Befehle per DDE an eine andere Applikation.

SYNTAX:
DDEExecute channel As Integer, Command As String

PARAMETER:
channel : Integerausdruck, der die Nummer des DDE-Kanals zu dem Topic der Applikation angibt, auf dem die Befehle ausgeführt werden sollen. Die Nummer des Kanals erhalten Sie durch die DDEInitiate-Anweisung.

Command : Beliebiger Zeichenkettenausdruck, der den zur übergebenden Befehl an die empfangende Applikation enthält. Command$ muß ein DDE-Kommando enthalten, das der empfangenden Applikation bekannt ist. Mehrere Befehle können zugleich übergeben werden; jeder Befehl muß dabei in eckigen Klammern eingeschlossen sein.

Die DDEExecute-Anweisung erlaubt es Ihnen, mehrere Befehle zur gleichen Zeit zu senden. Durch die folgende Anweisung wird StarCalc z. B. angewiesen, eine Tabelle zu öffnen und sofort wieder zu schließen:

DDEExecute Kanal, "[Documents.Add][ActiveDocument.Close]"

Beachten Sie, daß sich zwischen den in eckigen Klammern stehenden Befehlen kein Leerzeichen befindet, da das einen Laufzeitfehler hervorrufen würde.

Die obige Anweisung ist gleichbedeutend mit den beiden folgenden Anweisungen:

DDEExecute Kanal, "[Documents.Add]"

DDEExecute Kanal, "[ActiveDocument.Close]"

Die direkte Angabe von Zeichenfolgen als Parameter erfordert, daß die Zeichenkette in Anführungszeichen eingeschlossen ist. Problematisch wird es in dem Moment, in dem ein Anführungszeichen, das ja Anfang und Ende der Zeichenfolge kennzeichnet, Bestandteil der Zeichenfolge sein soll. Daher müssen Sie Anführungszeichen in Zeichenketten mit Chr$(34) einfügen oder die Anführungsstriche direkt doppelt hintereinander angeben. Durch folgende Anweisung wird StarWriter angewiesen, die Datei BEISPIEL.SDW zu öffnen:

DDEExecute Kanal,
"[Documents.Add("+Chr$(34)+"BEISPIEL.SDW"+Chr$(34)+")]"

Die andere Möglichkeit (Angabe eines Anführungszeichens durch zwei aufeinanderfolgende Anführungszeichen) ist:

DDEExecute Kanal, "[Documents.Add(""BEISPIEL.SDW"")]"

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
```

```
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

DDEInitiate-Funktion

Öffnet einen DDE-Kanal zur empfangenden Applikation, die von der sendenden Applikation Befehle entgegennehmen und ausführen kann und liefert die Kanalnummer zurück.

SYNTAX:
DDEInitiate (Application As String, Object As String)

RÜCKGABE:
Integer

PARAMETER:
Application : Beliebiger Zeichenkettenausdruck, der den Dateinamen der Applikation ohne die Endung ".EXE" und ohne Pfad enthält. Wichtig: Die Applikation muß gestartet sein, damit die DDEInitiate-Funktion einen Kanal öffnen kann.

Object : Die Bezeichnung eines der Applikation bekannten Objects. Das kann bei StarOffice-Programmen z. B. der Name eines geöffneten Dokumentes sein. Ist das Object beispielsweise der Name eines Dokumentes, so muß das Dokument zuvor geöffnet werden.

Diese Funktion öffnet einen Kanal zur Applikation und liefert einen numerischen Ausdruck zurück, der der Nummer des Kanals entspricht. Mit dieser Kanalnummer können Sie über DDE-Anweisungen mit dem Objekt der Applikation kommunizieren und es dadurch beeinflussen. Wird das angegebene Topic nicht von der Applikation zur Verfügung gestellt, gibt die Funktion den Wert Null zurück.

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
```

```
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

DDEPoke-Anweisung

Sendet Daten zu einem Element eines Objektes einer Applikation über einen geöffneten DDE-Kanal.

SYNTAX:
DDEPoke channel As Integer, element As String, dates As String

PARAMETER:
channel : Beliebiger Integerausdruck, der die Nummer des DDE-Kanals zur entsprechenden Applikation enthält. Die Nummer des Kanals erhalten Sie durch die DDEInitiate-Anweisung.

element : Name des Elements, daß sich auf das zuvor mit DDEInitiate angesprochene Objekt der empfangenden Applikation bezieht.

dates : Daten, die dem Element des Objektes übermittelt werden sollen.

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
```

```
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

DDERequest-Funktion

Ermittelt ein Datum eines Elementes des durch DDEInitiate angesprochenen Objektes einer Applikation.

SYNTAX:
DDERequest (channel As Integer, element As String)

RÜCKGABE:
String

PARAMETER:
channel : Integerausdruck, der die Nummer des DDE-Kanals zu dem gewünschten Topic der Applikation angibt. Die Nummer des Kanals erhalten Sie durch die DDEInitiate-Anweisung.

element : Name des Elements, daß sich auf das zuvor mit DDEInitiate angesprochene Objekt der empfangenden Applikation bezieht.

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

DDETerminate-Anweisung

Schließt einen bestimmten DDE-Kanal, der zuvor mit DDEInitiate geöffnet wurde.

SYNTAX:
DDETerminate channel As Integer

PARAMETER:
channel : Beliebiger Integerausdruck, der die Nummer des DDE-Kanals zur entsprechenden Applikation enthält. Die Nummer des Kanals erhalten Sie durch die DDEInitiate-Anweisung.

Ein geöffneter DDE-Kanal sollte immer geschlossen werden, wenn er nicht mehr benötigt wird, da er Systemressourcen beansprucht, die erst nach dem Schließen des Kanals wieder freigegeben werden.

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

DDETerminateAll-Anweisung

Schließt alle geöffneten DDE-Kanäle.

SYNTAX:
DDETerminateAll

BEISPIEL:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

Fehlerbehandlungsfunktionen

Mit diesen Anweisungen und Funktionen können Sie festlegen, wie StarBasic auf Fehler während einer Programmausführung reagieren soll.

StarBasic bietet Ihnen verschiedene Möglichkeiten, auf Fehler zu reagieren, die während der Ausführung Ihres Programmes auftreten können. In Fehlerbehandlungsroutinen werten Sie diese Fehler aus und lassen Ihr Programm darauf entsprechend reagieren. Einem Programmabbruch nach einem Laufzeitfehler wird damit entgegengewirkt.

Folgende Funktionen und Anweisungen zur Fehlerbehandlung werden unterstützt:

Erl-Funktion

Gibt Aufschluß über die Zeilennummer, in der während der Programmausführung ein Fehler auftrat.

SYNTAX:
Erl

Rückgabe:
Integer

Parameter:

Die Erl-Funktion liefert nur Zeilennummern und keine Zeilenmarken zurück.

Beispiel:

```
sub ExampleError
on error goto ErrorHandler  REM Fehlerbehandlung anmelden
Dim iVar as Integer
Dim sVar As String
REM hier wird ein Fehler verursacht, da die Datei nicht
existiert
iVar = Freefile
Open "\file9879.txt" for Input as #iVar
Line Input #iVar, sVar
Close #iVar
exit sub
ErrorHandler:
MsgBox "Fehler " & err & ": " & error$ + chr(13) + "In line : "
+ Erl + chr(13) + Now , 16 ,"Fehler aufgetreten"
end sub
```

Err-Funktion

Liefert nach dem Auftreten eines Fehlers während der Programmausführung einen Laufzeitfehlercode zurück, der den Fehler kennzeichnet.

Syntax:
Err

Rückgabe:
Integer

Parameter:
Die Err-Funktion wird in Fehlerbehandlungsroutinen zur Bestimmung des aufgetretenen Fehlers verwendet. In der Fehlerbehandlungsroutine wird dann entsprechend den Anweisungen des Programmierers auf diesen Fehler reagiert und die Programmausführung anschließend fortgesetzt.

Beispiel:

```
sub ExampleError
on error goto ErrorHandler  REM Fehlerbehandlung anmelden
Dim iVar as Integer
Dim sVar As String
REM hier wird ein Fehler verursacht, da die Datei nicht
existiert
```

```
iVar = Freefile
Open "\file9879.txt" for Input as #iVar
Line Input #iVar, sVar
Close #iVar
exit sub
ErrorHandler:
MsgBox "Fehler " & Err & ": " & Error$ + chr(13) + "In line : "
+ Erl + chr(13) + Now , 16 ,"Fehler aufgetreten"
end sub
```

Error-Funktion

Liefert zu einem Fehlercode die Fehlermeldung.

SYNTAX:
Error (Expression)

RÜCKGABE:
String

PARAMETER:
expression: Beliebiger numerischer Ausdruck, der die Fehlernummer der Fehlermeldung enthält, die zurückgegeben werden soll.

Die Error-Funktion ohne Parameter liefert die Meldung des Fehlers zurück, der zuletzt während der Programmausführung auftrat.

BEISPIEL:

```
sub ExampleError
on error goto anError  REM Fehlerbehandlung anmelden
REM Hier trat ein Error auf ...
exit sub
anError:
MsgBox error + chr(13) + "In line : " + Erl + chr(13) + Now ,
16 ,"Error"
end sub
```

Error-Anweisung

Simuliert das Auftreten eines Fehlers während der Programmausführung.

SYNTAX:
Error errornumber As Integer

PARAMETER:
errornumber : Integer-Ausdruck, der die Nummer des Fehlers enthält, dessen Auftreten simuliert werden soll.

Die Error-Anweisung simuliert das Auftreten eines Fehlers während der Programmausführung. Ein Fehler, der von einer Fehlerbehandlungsroutine ab-

gefangen wurde, wird nicht mehr gemeldet, gleichgültig, ob der Programmierer ihn berücksichtigt hat oder nicht. Die ERROR-ANWEISUNG wird dazu verwendet, die vom Programmierer unberücksichtigten Fehler, die häufig zu einer inkorrekten Programmausführung oder zu Programmabstürzen führen, zu melden und die Programmausführung anschließend abzubrechen.

BEISPIEL:

```
sub ExampleError
on error goto ErrorHandler  REM Fehlerbehandlung anmelden
Dim iVar as Integer
Dim sVar As String
REM hier wird ein Fehler verursacht, da die Datei nicht
existiert
iVar = Freefile
Open "\file9879.txt" for Input as #iVar
Line Input #iVar, sVar
Close #iVar
exit sub
ErrorHandler:
MsgBox "Fehler " & err & ": " & error$ + chr(13) + "In line : "
+ Erl + chr(13) + Now , 16 ,"Fehler aufgetreten"
end sub
```

On Error GoTo...Resume-Anweisung

Verzweigt nach dem Auftreten eines Fehlers während der Programmausführung in eine Fehlerbehandlungsroutine oder setzt die Programmausführung fort.

SYNTAX:
On [Local] {Error GoTo Labelname | GoTo 0 | Resume Next}

PARAMETER:
Local: Kennzeichnet eine Fehlerbehandlungsroutine, speziell für eine Prozedur. Wird Local nicht angegeben, bezieht sich die Fehlerbehandlung auf das gesamte Modul.

GoTo Labelname : Verzweigt bei einem Fehler zu der Fehlerbehandlungsroutine, deren erste Zeile mit der Zeilenmarke Labelname versehen ist.

Resume Next : Tritt ein Fehler auf, wird die Programmausführung ab der Zeile fortgesetzt, die der fehlerverursachenden folgt.

GoTo 0 : Schaltet die Fehlerbehandlung aus.

Die On Error GoTo-Anweisung ermöglicht eine Reaktion des Programmierers auf Fehler, die möglicherweise in seinem Programm auftreten können. Dabei ist zu beachten, daß die Anweisung, handelt es sich um eine lokale

Fehlerbehandlung, am Anfang einer Prozedur oder wenn es sich um eine globale Fehlerbehandlung, am Anfang des Moduls stehen muß.

Beispiel:

```
Sub ExampleReset
On Error Goto ErrorHandler
Dim iNumber As Integer
Dim sZeile As String
Dim aFile As String
aFile = "c:\daten.txt"
iNumber = Freefile
Open aFile For Output As #iNumber
Print #iNumber, "Das ist eine Zeile Text"
Close #iNumber
Documents.Add("sCalc4")
Selection.Insert("Gleich kommt ein Fehler")
wait 1
iNumber = Freefile
Open aFile For Input As iNumber
While not eof(iNumber)
Line Input #iNumber, sZeile
If sZeile <>"" then
Selection.Text.InsertPara()  REM hier stimmt der Aufruf nicht
Selection.Text.Insert(Trim$(sZeile)) REM und es kommt zu einem
Laufzeitfehler
end if
wend
Close #iNumber
Exit Sub
ErrorHandler:
Reset
MsgBox "Alle Dateien werden geschlossen",0,"Fehler"
End Sub
```

Logische Operatoren

Hier erfahren Sie, welche Logischen Operatoren StarBasic unterstützt.

Logische Operatoren verknüpfen bitweise den Inhalt zweier Ausdrücke oder Variablen. Sie werden beispielsweise verwendet, um zu prüfen, ob bestimmte Bits gesetzt sind oder nicht.

Diese Logischen Operatoren unterstützt StarBasic:

And-Operator

Führt eine logische UND-Verknüpfung zwischen zwei Ausdrücken durch.

Syntax:
Result = Expression1 And Expression2

PARAMETER:
Result : Eine beliebige numerische Variable, die das Ergebnis der Verknüpfung aufnimmt.

Expression1, Expression2 : Beliebige Ausdrücke, die miteinander verknüpft werden sollen.

Bei einer UND-Verknüpfung von boolschen Ausdrücken wird nur dann der Wert TRUE zurückgegeben, wenn die beiden miteinander verknüpften Ausdrücke TRUE sind:

TRUE AND TRUE gibt TRUE, alle anderen Kombinationen geben FALSE.

Bei einer bitweisen UND-Verknüpfung werden nur die Bits gesetzt, die auch in den beiden angegebenen Ausdrücken gesetzt sind.

BEISPIEL:

```
Sub ExampleAnd
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vVarOut as Variant
A = 10: B = 8: C = 6: D = Null
vVarOut = A > B And B > C   REM Liefert -1
vVarOut = B > A And B > C   REM Liefert 0
vVarOut = A > B And B > D   REM Liefert -1
vVarOut = (B > D And B > A) REM Liefert 0
vVarOut = B And A           REM Liefert 8 wegen bitweiser UND-
Verknüpfung der beiden Argumente
End Sub
```

Eqv-Operator

Führt eine logische Äquivalenz-Verknüpfung zwischen zwei Ausdrücken durch.

SYNTAX:
Result = Expression1 Eqv Expression2

PARAMETER:
result: Eine beliebige numerische Variable, die das Ergebnis der Verknüpfung aufnimmt.

Expression1, expression2 : Beliebige Ausdrücke, die miteinander verknüpft werden sollen.

Bei einer Äquivalenz-Verknüpfung von boolschen Ausdrücken wird der Wert TRUE zurückgegeben, wenn entweder beide Ausdrücke TRUE oder beide Ausdrücke FALSE sind.

Bei einer bitweisen ÄQUIVALENZ-VERKNÜPFUNG ist ein Bit im Ergebnis gesetzt, wenn entweder beide Bits in den Ausdrücken gesetzt oder gelöscht sind. Ansonsten ist das Bit im Ergebnis gelöscht.

BEISPIEL:

```
Sub ExampleEqv
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
vA = 10: vB = 8: vC = 6: vD = Null
vOut = A > B Eqv B > C   REM Liefert -1
vOut = B > A Eqv B > C   REM Liefert -1
vOut = A > B Eqv B > D   REM Liefert -1
vOut = (B > D Eqv B > A) REM Liefert 0
vOut = B Eqv A           REM Liefert -1
End Sub
```

Imp-Operator

Führt eine logische Implikationsverknüpfung zwischen zwei Ausdrücken durch.

SYNTAX:
Result = Expression1 Imp Expression2

PARAMETER:
Result : Eine beliebige numerische Variable, die das Ergebnis der Verknüpfung aufnimmt.

Expression1, Expression2 : Beliebige Ausdrücke, die miteinander verknüpft werden sollen.

Bei einer Implikationsverknüpfung von boolschen Ausdrücken wird der Wert False nur dann zurückgeben, wenn der erste Ausdruck True und der zweite Ausdruck False ist.

Bei einer bitweisen Implikationsverknüpfung ist ein Bit im Ergebnis nur dann gelöscht, wenn das entsprechende Bit im ersten Ausdruck gesetzt und das entsprechende Bit im zweiten Ausdruck gelöscht worden ist.

BEISPIEL:

```
Sub ExampleImp
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
A = 10: B = 8: C = 6: D = Null
vOut = A > B Imp B > C   REM Liefert -1
vOut = B > A Imp B > C   REM Liefert 0
vOut = A > B Imp B > D   REM Liefert -1
```

```
vOut = (B > D Imp B > A) REM Liefert 0
vOut = B Imp A           REM Liefert -3
End Sub
```

Not-Operator

Negiert einen Ausdruck durch Vertauschen der Bitzustände.

SYNTAX:
Result = Not Expression

PARAMETER:
Result : Eine beliebige numerische Variable, die das Ergebnis der Negation aufnimmt.

Not Expression : Beliebiger Ausdruck, der negiert werden soll.

Bei einer Negation mit boolschen Ausdrücken wird der Wert False in den Wert True und der Wert True in den Wert False umgekehrt.
Bei einer bitweisen Negation werden nicht, wie bei allen anderen logischen Operationsbefehlen, zwei Bits logisch miteinander verknüpft, sondern ein einzelnes Bit umgekehrt.

BEISPIEL:

```
Sub ExampleNot
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
vA = 10: vB = 8: vC = 6: vD = Null
vOut = Not vA REM Liefert -11
vOut = Not(vC > vD) REM Liefert 0
vOut = Not(vB > vA) REM Liefert -1
vOut = Not(vA > vB) REM Liefert 0
end Sub
```

Or-Operator

Führt eine logische ODER-Verknüpfung zwischen zwei Ausdrücken durch.

SYNTAX:
Result = Expression1 Or Expression2

PARAMETER:
Result: Eine beliebige numerische Variable, die das Ergebnis der Verknüpfung aufnimmt.

Expression1, Expression2: Beliebige Ausdrücke, die miteinander verknüpft werden sollen.

Bei einer ODER-Verknüpfung von booleschen Ausdrücken wird dann der Wert True zurückgegeben, wenn mindestens einer der beiden miteinander verknüpften Ausdrücke True ist.

Bei einer bitweisen ODER-Verknüpfung werden nur die Bits gesetzt, die in mindestens einem der beiden angegebenen Ausdrücken gesetzt sind.

Beispiel:

```
Sub ExampleOr
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
vA = 10: vB = 8: vC = 6: vD = Null
vOut = vA > vB Or vB > vC   REM -1
vOut = vB > vA Or vB > vC   REM -1
vOut = vA > vB Or vB > vD   REM -1
vOut = (vB > vD Or vB > vA) REM -1
vOut = vB Or vA           REM 10
End Sub
```

Xor-Operator

Führt eine logische Exklusiv-Oder-Verknüpfung zwischen zwei Ausdrücken durch.

Syntax:
Result = Expression1 Xor Expression2

Parameter:
Result : Eine beliebige numerische Variable, die das Ergebnis der Verknüpfung aufnimmt.

Expression1, Expression2 : Beliebige numerische Ausdrücke, die miteinander verknüpft werden sollen.

Bei einer logischen Exklusiv-ODER-Verknüpfung zweier boolscher Ausdrücke wird nur dann True zurückgegeben, wenn die beiden miteinander verknüpften Ausdrücke unterschiedlich sind.

Bei einer bitweisen Exklusiv-ODER-Verknüpfung ist ein Bit im Ergebnis gesetzt, wenn das entsprechende Bit bei nur einem der beiden Ausdrücke gesetzt ist.

Beispiel:

```
Sub ExampleXor
Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
Dim vOut as Variant
vA = 10: vB = 8: vC = 6: vD = Null
vOut = vA > vB Xor vB > vC   REM Liefert 0
vOut = vB > vA Xor vB > vC   REM Liefert -1
```

```
vOut = vA > vB Xor vB > vD   REM Liefert 0
vOut = (vB > vD Xor vB > vA) REM Liefert -1
vOut = vB Xor vA           REM Liefert 2
End Sub
```

Mathematische Operatoren

Hier erfahren Sie, welche mathematischen Operatoren StarBasic unterstützt.

Dieses Kapitel zeigt Ihnen in einer kurzen Übersicht alle arithmetischen Operatoren, die Sie für Berechnungen innerhalb Ihrer Programme verwenden können.

Diese mathematischen Operatoren finden Sie im StarBasic:

"-" Operator

Subtrahiert zwei Werte.

SYNTAX:
Result = Expression1 - Expression2

PARAMETER:
Result : Beliebiger numerischer Ausdruck, der das Ergebnis der Subtraktion aufnimmt.

Expression1, Expression2 : Beliebige numerische Ausdrücke, die voneinander subtrahiert werden sollen.

BEISPIEL:

```
Sub ExampleSubtraction1
Print 5 - 5
End sub
Sub ExampleSubtraction2
Dim iValue1 as Integer
Dim iValue2 as Integer
iValue1 = 5
iValue2 = 10
Print iValue1 - iValue2
End Sub
```

"*" Operator

Multipliziert zwei Werte.

SYNTAX:
Result = Expression1 * Expression2

PARAMETER:
Result: Beliebiger numerischer Ausdruck, der das Ergebnis der Multiplikation aufnimmt.

expression1, expression2: Beliebige numerische Ausdrücke, die miteinander multipliziert werden sollen.

BEISPIEL:

```
Sub ExampleMultiplication1
Print 5 * 5
End sub
Sub ExampleMultiplication2
Dim iValue1 as Integer
Dim iValue2 as Integer
iValue1 = 5
iValue2 = 10
Print iValue1 * iValue2
End Sub
```

"+" Operator

Addiert zwei Ausdrücke oder fügt diese zusammen.

SYNTAX:
Result = Expression1 + Expression2

PARAMETER:
Result : Beliebiger numerischer Ausdruck oder Zeichenfolgenausdruck, der das Ergebnis der Addition aufnimmt.

Expression1, Expression2 : Beliebige numerische Ausdrücke oder Zeichenfolgenausdrücke, die miteinander addiert bzw. zusammengefügt werden

BEISPIEL:

```
Sub ExampleAddition1
Print 5 + 5
End sub
Sub ExampleAddition2
Dim iValue1 as Integer
Dim iValue2 as Integer
iValue1 = 5
iValue2 = 10
Print iValue1 + iValue2
End Sub
```

"/" Operator

Dividiert zwei Werte.

SYNTAX:
Result = Expression1 / Expression2

PARAMETER:
Result : Beliebiger numerischer Ausdruck, der das Ergebnis der Division aufnimmt.

Expression1, Expression2 : Beliebige numerische Ausdrücke, die dividiert werden sollen.

BEISPIEL:

```
Sub ExampleDivision1
Print 5 / 5
End sub
Sub ExampleDivision2
Dim iValue1 as Integer
Dim iValue2 as Integer
iValue1 = 5
iValue2 = 10
Print iValue1 / iValue2
End Sub
```

"^" Operator

Potenziert einen Wert.

SYNTAX:
Result = Expression ^ Exponent

PARAMETER:
Result : Beliebiger numerischer Ausdruck, der das Ergebnis der Potenzierung aufnimmt.

Expression : Wert, der potenziert werden soll.

Exponent : Die Hochzahl, mit der Expression potenziert werden soll.

BEISPIEL:

```
Sub ExamplePot
Print ( 12.345 ^ 23 )
Print Exp ( 23 * Log( 12.345 ) ) REM Potenzieren durch
Logarithmenbildung
End Sub
```

Numerische Funktionen

Hier finden Sie alle Funktionen, die numerische Berechnungen durchführen können. Nicht einzeln aufgeführt sind dabei die sogenannten mathematischen und Booleschen Operatoren, denen ein eigener Abschnitt gewidmet ist. Funktionen unterscheiden sich von Operatoren insofern, als daß Ihnen ein Funktionsargument übergeben wird und sie ein Ergebnis zurückliefern. Operatoren hingegen liefern ein Ergebnis zurück, indem Sie zwei numerische Ausdrücke miteinander verknüpfen.

Sie finden in diesem Kapitel folgende mathematische Funktionen beschrieben:

Trigonometrische Funktionen

Hier finden Sie alle trigonometrischen Funktionen, die StarBasic unterstützt.

Atn-Funktion

Ermittelt den Arkustangens eines numerischen Ausdrucks. Das Ergebnis wird im Bogenmaß (Radiant) zurückgegeben und liegt im Bereich -(Pi/2) und +(Pi/2).

Der Arkustangens ist die Umkehrfunktion des Tangens. Das bedeutet, die Atn-Funktion bestimmt aus dem Tangens eines Winkels Alpha, bzw. dem Längenverhältnis von Gegenkathete zu Ankathete in einem rechtwinkligen Dreieck, den zugehörigen Winkel Alpha im Bogenmaß. Es gilt:
Atn(Gegenkathete/Ankathete) = Alpha

SYNTAX:
Atn (Number)

RÜCKGABE:
Double

PARAMETER:
Number: Beliebiger numerischer Ausdruck, der den Tangens eines Winkels bzw. das Verhältnis von Gegenkathete zu Ankathete angibt, zu dem der Arkustangens ermittelt werden soll.

Zur Umrechnung von Radiant in Grad muß das Bogenmaß mit 180/Pi multipliziert werden.

Grad = (Radiant*180)/Pi
Radiant = (Grad*Pi)/180
Pi ist hierbei die feste Kreiskonstante mit dem Wert von 3,141592657... .

Das folgende Beispielprogramm berechnet in einem rechtwinkligen Dreieck aus dem Tangens des Winkels Alpha den Winkel Alpha:

BEISPIEL:

```
Sub ExampleATN
REM Pi = 3.1415926 ist eine vordefiníerte Konstante
Dim d1 as Double
Dim d2 as Double
d1 = InputBox$ ("Bitte geben Sie die Länge der Ankathete ein:
","Ankathete")
d2 = InputBox$ ("Bitte geben Sie die Länge der Gegenkathete
ein: ","Gegenkathete")
Print "Der Winkel Alpha beträgt"; (atn (d2/d1) * 180 / Pi); "
Grad"
End Sub
```

Cos-Funktion

Ermittelt den Kosinus eines Winkels. Der Winkel wird im Bogenmaß (Radiant) angegeben. Das Ergebnis liegt im Bereich von –1 bis 1.

Die Cos-Funktion bestimmt aus einem Winkel Alpha das Längenverhältnis von Ankathete zu Hypotenuse in einem rechtwinkligen Dreieck. E sgilt: Cos(Alpha) = Ankathete/Hypotenuse

SYNTAX:
Cos (Number)

RÜCKGABE:
Double

PARAMETER:
Number: Numerischer Ausdruck, der einen Winkel im Bogenmaß angibt, dessen Kosinus berechnet werden soll.

Zur Umrechnung von Grad in Radiant muß die Gradangabe mit Pi/180 multipliziert werden, und zur Umrechnung von Radiant in Grad muß das Bogenmaß mit 180/Pi multipliziert werden.

Grad = (Radiant*180)/Pi
Radiant = (Grad*Pi)/180
Pi ist hierbei die feste Kreiskonstante mit dem Wert von 3,141592657... .

Das folgende Beispielprogramm erlaubt für ein rechtwinkeliges Dreieck die Eingabe von Ankathete und Winkel (in Grad) und berechnet daraus die Länge der Hypothenuse:

Beispiel:

```
Sub ExampleCosine
REM Pi = 3.1415926
Dim d1 as Double, dAngle as Double
d1 = InputBox$ ("Bitte geben Sie die Länge der Ankathete ein:
","Ankathete")
dAngle = InputBox$ ("Bitte geben Sie den Winkel Alpha ein (in
Grad): ","Alpha")
Print "Die Länge der Hypothenuse beträgt"; (d1 / cos (dAngle *
Pi / 180))
End Sub
```

Sin-Funktion

Ermittelt den Sinus eines Winkels. Der Winkel wird im Bogenmaß (Radiant) angegeben. Das Ergebnis liegt im Bereich von –1 bis 1.

Die Sin-Funktion bestimmt aus einem Winkel Alpha das Längenverhältnis von Gegenkathete zu Hypotenuse in einem rechtwinkligen Dreieck. E sgilt: Sin(Alpha) = Gegenkathete/Hypotenuse

Syntax:
Sin (Number)

Rückgabe:
Double

Parameter:
Number: Numerischer Ausdruck, der einen Winkel im Bogenmaß angibt, dessen Sinus berechnet werden soll.

Zur Umrechnung von Grad in Radiant muß die Gradangabe mit Pi/180 multipliziert werden, und zur Umrechnung von Radiant in Grad muß das Bogenmaß mit 180/Pi multipliziert werden.

Grad = (Radiant*180)/Pi
Radiant = (Grad*Pi)/180
Pi ist hierbei die feste Kreiskonstante mit dem Wert von 3,141592657... .

Das folgende Beispielprogramm erlaubt für ein rechtwinkeliges Dreieck die Eingabe von Gegenkathete und Winkel (in Grad) und berechnet daraus die Länge der Hypothenuse:

Beispiel:

```
Sub ExampleSinus
REM Pi = 3.1415926 ist vordefinierte variable
Dim d1 as Double
Dim dAlpha as Double
d1  = InputBox$ ("Bitte geben Sie die Länge der Gegenkathete
```

```
ein: ","Gegenkathete")
dAlpha = InputBox$ ("Bitte geben Sie den Winkel Alpha ein (in
Grad): ","Alpha")
Print "Die Länge der Hypothenuse beträgt"; (d1 / sin (dAlpha *
Pi / 180))
End Sub
```

Tan-Funktion

Ermittelt den Tangens eines Winkels. Der Winkel wird im Bogenmaß (Radiant) angegeben.

Die Tan-Funktion bestimmt aus einem Winkel Alpha das Längenverhältnis von Gegenkathete zu Ankathete in einem rechtwinkligen Dreieck. E sgilt: Tan(Alpha) = Gegenkathete/Ankathete

SYNTAX:
Tan (Number)

RÜCKGABE:
Double

PARAMETER:
Number : Beliebiger numerischer Ausdruck, der einen Winkel im Bogenmaß angibt, dessen Tangens berechnet werden soll.

Zur Umrechnung von Grad in Radiant muß die Gradangabe mit Pi/180 multipliziert werden, und zur Umrechnung von Radiant in Grad muß das Bogenmaß mit 180/Pi multipliziert werden.

Grad = (Radiant*180)/Pi
Radiant = (Grad*Pi)/180
Pi ist hierbei die feste Kreiskonstante mit dem Wert von 3,141592657... .

Das folgende Beispielprogramm erlaubt die Eingabe der beiden Katheten und berechnet daraus den entsprechenden Winkel Alpha in einem rechtwinkligen Dreieck:

BEISPIEL:

```
Sub ExampleTan
REM pi# = 3.141592654 vordefiniert
Dim dg As Double, da As Double
dg = Int(InputBox$ ("Bitte geben Sie die Länge der Gegenkathete
ein: ","Gegenkathete"))
da = Int(InputBox$ ("Bitte geben Sie die Länge der Ankathete
ein: ","Ankathete"))
Print "Der Winkel Alpha beträgt"; Tan(dg/da); " im Bogenmaß"
Print "Der Winkel Alpha beträgt"; (180 * Tan(dg/da)) / pi; " in
Grad"
End Sub
```

Exponential- und Logarithmusfunktion

Hier sind Exponential- und Logarithmusfunktion aufgeführt.

Exp-Funktion

Potenziert die Basis des natürlichen Logarithmus (die Eulersche Zahl, e = 2,718282) um die angegebene Potenz.

SYNTAX:
Exp (Number)

RÜCKGABE:
Double

PARAMETER:
Number: Beliebiger numerischer Ausdruck, der die Potenz angibt, mit der die Eulersche Zahl e (die Basis des natürlichen Logarithmus) berechnet werden soll. Die Potenz muß für:

Zahlen einfacher Genauigkeit <= 88,02969 und für Zahlen doppelter Genauigkeit <= 709,782712893 sein, da StarBasic sonst einen Überlauffehler meldet (der maximal darstellbare Zahlenbereich wurde dabei überschritten).

BEISPIEL:

```
Sub ExampleLogExp
Dim dValue as Double
const b1=12.345e12
const b2=1.345e34
dValue=Exp( Log(b1)+Log(b2) )
MsgBox "" & dValue & chr(13) & (b1*b2) ,0,"Multiplikation durch
Logarithmenbildung"
end sub
```

Log-Funktion

Liefert den natürlichen Logarithmus eines numerischen Ausdrucks.

SYNTAX:
Log (Number)

RÜCKGABE:
Double

PARAMETER:
Number : Beliebiger numerischer Ausdruck, dessen natürlicher Logarithmus berechnet werden soll.

Der natürliche Logarithmus ist der Logarithmus zur Basis e. Die Basis e ist eine Konstante mit dem Wert 2,718282...

Sie können Logarithmen zu jeder beliebigen Basis (n) für jede Zahl (x) berechnen, indem Sie den natürlichen Logarithmus von x durch den natürlichen Logarithmus von n dividieren, wie nachstehend aufgeführt:

Log n(x) = Log(x) / Log(n)

Beispiel:

```
Sub ExampleLogExp
Dim a as Double
Dim const b1=12.345e12
Dim const b2=1.345e34
a=Exp( Log(b1)+Log(b2) )
MsgBox "" & a & chr(13) & (b1*b2) ,0,"Multiplikation durch
Logarithmenbildung"
end sub
```

Zufallszahl-Generierung

Hier finden Sie Anweisungen und Funktionen zum Erzeugen von Zufallszahlen.

Randomize-Anweisung

Initialisiert den Zufallsgenerator.

Syntax:
Randomize [Number]

Parameter:
Number : Beliebiger Integerwert, der den Zufallsgenerator initialisiert. Wird Number nicht mit angegeben, wird der Zufallsgenerator mit dem Wert des System-Timers initialisiert.

Beispiel:

```
Sub ExampleRandomize
Dim iVar As Integer, sText As String
Dim iSpectral(10) As Integer
Randomize 2^14-1
For iCount = 1 To 1000
iVar = Int((10 * Rnd) ) REM Bereich von 0 bis 9
iSpectral(iVar) = iSpectral(iVar) +1
Next iCount
sText = " | "
For iCount = 0 To 9
sText = sText & iSpectral(iCount) & " | "
```

```
Next iCount
MsgBox sText,0,"Spektrale Verteilung"
end sub
```

Rnd-Funktion

Ermittelt einen Zufallswert zwischen 0 und 1.

SYNTAX:
Rnd [(Expression)]

RÜCKGABE:
Double

PARAMETER:
Expression : numerischer Ausdruck, der bestimmt, auf welche Art und Weise Zufallszahlen generiert werden sollen.

Kleiner als Null : Generiert immer dieselbe Zufallszahl.

Größer als Null : Die nächste Zufallszahl der Folge.

Gleich Null : Die zuletzt generierte Zufallszahl.

Nicht angegeben : Die nächste Zufallszahl der Folge.

Wird bei jedem Aufruf der Rnd-Funktion mit dem Parameter Expression dieselbe Zahl übergeben, wird immer dieselbe Zufallszahlenfolge generiert, da der Parameter Expression als Startwert für die nächste Zahl in der Folge dient.

Die RND-Funktion liefert grundsätzlich nur Werte zwischen 0 und 1. Um Zufallszahlen ohne Nachkommastellen in bestimmten Zahlenbereichen zu generieren, können Sie sich folgender Formel bedienen:

BEISPIEL:

```
Sub ExampleRandomSelect
Dim iVar As Integer
iVar = Int((15 * Rnd) -2)
Select Case iVar
Case 1 To 5
Print "Zahl von 1 bis 5"
Case 6, 7, 8
Print "Zahl von 6,7, oder  8"
Case Is > 8 And iVar < 11
Print "Größer als 8"
Case Else
Print "Außerhalb von 1 bis 10"
End Select
end sub
```

Quadratwurzelberechnung

Mit dieser Funktion können Sie Quadratwurzeln berechnen.

Sqr-Funktion

Ermittelt die Quadratwurzel des angegebenen numerischen Ausdrucks.

SYNTAX:
Sqr (Number)

RÜCKGABE:
Double

PARAMETER:
Number : Beliebiger numerischer Ausdruck, dessen Quadratwurzel berechnet werden soll.

Die Quadratwurzel ist die Zahl, die man in der Gleichung a^2=c mit 2 potenzieren muß, um die Zahl c zu erhalten (6^2 ist beispielsweise 36; die Quadratwurzel aus 36 ist folglich 6).

BEISPIEL:

```
Sub ExampleSqr
Dim iVar As Single
iVar = 36
Msgbox Sqr(iVar)
end sub
```

Ganzzahlermittlung

Hier finden Sie die Funktionen, mit denen Sie Werte zu ganzen Zahlen runden.

Fix-Funktion

Liefert die Vorkommazahl eines numerischen Ausdrucks durch Abschneiden des Nachkommateils zurück.

SYNTAX:
Fix (Expression)

RÜCKGABE:
Double

PARAMETER:
Expression : Numerischer Ausdruck, dessen Vorkommateil zurückgeliefert werden soll.

Beispiel:

```
sub ExampleFix
Print Fix(3.14159) REM ergibt den Wert 3.
Print Fix(0)       REM ergibt den Wert 0.
Print Fix(-3.14159)REM ergibt den Wert -3.
end sub
```

Int-Funktion

Liefert die Vorkommazahl eines numerischen Ausdrucks durch Abrunden zurück.

Syntax:
Int (Number)

Rückgabe:
Double

Parameter:
Number : numerischer Ausdruck, dessen Vorkommateil durch Abrunden ermittelt werden soll.

Beispiel:

```
sub ExampleINT
Print Int(3.14159) REM ergibt den Wert 3.
Print Int(0) REM ergibt den Wert 0.
Print Int(-3.14159) REM ergibt den Wert -4.
end sub
```

Absolutwertermittlung

Mit dieser Funktion können Sie Absolutwerte ermitteln.

Abs-Funktion

Ermittelt den Absolutwert eines numerischen Ausdrucks.

Syntax:
Abs (Number)

Rückgabe:
Double

Parameter:
Number: Beliebiger numerischer Ausdruck, dessen Absolutwert (vorzeichenloser Wert) ermittelt werden soll. Das bedeutet: Jede positive Zahl und 0 werden original zurückgeliefert, jede negative Zahl wird in eine positive Zahl verwandelt.

Das folgende Beispiel berechnet die Differenz zweier eingegebener Beträge mit Hilfe der ABS-Funktion. Dabei ist es ganz egal, ob Sie zuerst den größeren oder den kleineren Betrag eingeben, die Abs-Funktion korrigiert ein etwaiges negatives Ergebnis.

BEISPIEL:

```
Sub ExampleDifference
Dim siW1 As Single
Dim siW2 As Single
siW1 = Int(InputBox$ ("Bitte geben Sie den ersten Betrag
ein","Werteingabe"))
siW2 = Int(InputBox$ ("Bitte geben Sie den zweiten Betrag
ein","Werteingabe"))
Print "Die Differenz beträgt "; Abs(siW1 - siW2)
End Sub
```

Vorzeichen eines Ausdrucks

Mit dieser Funktion ermitteln Sie das Vorzeichen eines numerischen Ausdrucks.

Sgn-Funktion

Liefert einen ganzzahligen Wert zwischen -1 und 1, der anzeigt, ob der der Funktion übergebende Ausdruck positiv oder negativ oder Null ist.

SYNTAX:
Sgn (Number)

RÜCKGABE:
Integer

PARAMETER:
Number : Numerischer Ausdruck, dessen Vorzeichen ermittelt werden soll.

NumAusdruck	Rückgabewert
negativ	Sgn liefert -1 zurück.
0	Sgn liefert 0 zurück.
positiv	Sgn liefert 1 zurück.

BEISPIEL:

```
Sub ExampleSgn
Print sgn(-10) REM liefert -1
Print sgn(0)   REM liefert 0
Print sgn(10)  REM liefert 1
end sub
```

Zahlenumwandlung

Hier finden Sie die Funktionen zur Umwandlung von Zahlen von einem Zahlensystem in ein anderes.

Hex-Funktion

Liefert das Ergebnis der Umrechnung einer Zahl in das Hexadezimalformat.

SYNTAX:
Hex (Number)

RÜCKGABE:
String

PARAMETER:
Number : Beliebiger numerischer Ausdruck, der in eine Hexadezimalzahl umgewandelt werden soll.

BEISPIEL:

```
Sub ExampleHex
REM Verwendet BasicFormeln im StarCalc
Documents.Add("Scalc4")
Wait 1000
Range("a2").Insert("&H3E8")
Range("b2").Insert("=Hex2Int(a2)")
Range("c2").Insert("=Int2Hex(b2)")
End Sub
Function Hex2Int( sHex As String ) As Long
   REM Berechnet aus einem hexadezimalen Wert einen Long-
Integer.
Hex2Int = clng( sHex )
End Function
Function Int2Hex( iLong As Long) As String
   REM Berechnet einen hexadezimalen Wert in Integer.
Int2Hex = "&H" & Hex( iLong )
End Function
```

Oct-Funktion

Liefert das Ergebnis der Umrechnung einer Zahl in das Oktalformat.

SYNTAX:
Oct (Number)

RÜCKGABE:
String

PARAMETER:
Number : Beliebiger numerischer Ausdruck, der in eine Oktalzahl umgewandelt werden soll.

BEISPIEL:

```
Sub ExampleOkt
Msgbox Oct(255)
end sub
```

Steuern von Programmabläufen

Hier finden Sie alle Anweisungen, mit denen Sie Programmabläufe steuern können.

Normalerweise wird ein Programm von oben nach unten, Befehl für Befehl nacheinander abgearbeitet. In vielen Fällen ist es jedoch notwendig, bestimmte Programmteile nur dann auszuführen, wenn bestimmte Bedingungen erfüllt sind. Oder aber Sie müssen innerhalb einer Funktion (Function) oder Unterroutine (Sub) einen bestimmten Programmteil mehrfach ausführen. Eine weitere Möglichkeit ist, daß Sie Teile des Programms in sogenannten Schleifen so oft wiederholen lassen, bis oder solange bestimmte Bedingungen erfüllt sind. Diese Anweisungen zur Programmsteuerung gliedern sich in die Unterkategorien Entscheidungsanweisungen, Schleifen und Sprunganweisungen.

Entscheidungen

Hier finden Sie die Entscheidungsanweisungen.

If...Then...Else-Anweisung

Definiert einen oder mehrere Befehlsblöcke, die jeweils nur dann ausgeführt werden, wenn eine bestimmte Bedingung wahr ist.

SYNTAX:
If condition=true Then

Befehlsblock

[ElseIf bedingung=wahr Then]

Befehlsblock

[Else]

Befehlsblock

End If

PARAMETER:
Mit Hilfe dieser Struktur-Befehle können Sie die Ausführung von Teilen Ihres Programms von bestimmten Bedingungen abhängig machen. Trifft StarBasic auf eine If-Anweisung, prüft es zunächst, ob die nachstehende Bedingung wahr ist. Ist sie es, werden alle direkt nachstehenden Befehle bis zur nächsten ELSE- oder ELSEIF-Anweisung ausgeführt. Ist die Bedingung nicht wahr, und es folgt im Programm eine ELSEIF-Anweisung, prüft StarBasic wiederum die Bedingung, die dahinter steht und führt die nachstehenden Befehle aus, falls die Bedingung wahr gewesen ist. Falls nicht, geht die Programmausführung entweder beim nächsten ELSEIF oder - falls vorhanden - beim nächsten ELSE weiter. Die unter ELSE stehenden Befehle werden nur in dem Fall ausgeführt, wenn keine der vorher abgefragten Bedingungen (also hinter IF oder hinter allen vorhandenen ELSEIF-Anweisungen stehenden) zutraf. Sind alle Bedingungen entsprechend ihres Wahrheitsgehaltes ausgewertet und ist der entsprechende Programmteil abgearbeitet worden, fährt StarBasic mit dem Programm hinter der ENDIF-Anweisung fort.

IF...THEN-Abfragen können verschachtelt werden. D. h., innerhalb eines IF...THEN-Blocks dürfen weitere IF...THEN-Blöcke eingeschachtelt werden.

ELSE und ELSEIF-Statements brauchen nicht mit angegeben zu werden, sind also optional.

Ihr Programm darf zwar mit Hilfe von GOTO und GOSUB aus einer IF...THEN-Konstruktion heraus springen, jedoch nicht in eine IF...THEN -Struktur hineinspringen.

Das folgende Beispielprogramm läßt Sie das Verfalldatum eines Produktes eingeben und gibt aus, ob das Verfalldatum abgelaufen ist oder nicht oder ob es am heutigen Tage abläuft.

BEISPIEL:

```
Sub ExampleIfThenDate
Dim sDate as String
Dim sToday as String
sDate = InputBox("Bitte geben Sie das Verfalldatum ein
(TT.MM.JJJJ)")
sDate = Right$(sDate, 4) +  Mid$(sDate, 4, 2)  + Left$(sDate,
2)
sToday = Date$
sToday = Right$(sToday, 4)+   Mid$(sToday, 4, 2)  +
Left$(sToday, 2)
If sDate < sToday Then
```

```
MsgBox "Das Verfalldatum ist abgelaufen"
ElseIf sDate > sToday Then
MsgBox "Das Verfalldatum ist noch nicht abgelaufen"
Else
MsgBox "Das Verfalldatum läuft heute ab"
End If
End Sub
```

Select...Case-Anweisung

Definiert einen oder mehrere Befehlsblöcke, die in Abhängigkeit von dem Ergebnis eines bestimmten Ausdrucks stehen.

SYNTAX:
Select Case condition Case expression StatementBlock[Case expression2 StatementBlock][Case Else]SatementBlockEnd Select

PARAMETER:
Condition : Beliebiger Ausdruck, von dessen Ergebnis die Abarbeitung der jeweiligen hinter Case stehenden Befehlsblöcke abhängig ist.

Ausdruck : Beliebiger Ausdruck, der typkompatibel zu PruefAusdruck sein muß. Ein hinter Case stehender Befehlsblock wird dann abgearbeitet, wenn PRUEFAUSDRUCK und AUSDRUCK gleich sind.

BEISPIEL:

```
Sub ExampleRandomSelect
Dim iVar As Integer
iVar = Int((15 * Rnd) -2)
Select Case iVar
Case 1 To 5
Print "Zahl von 1 bis 5"
Case 6, 7, 8
Print "Zahl von 6,7, oder  8"
Case Is > 8 And iVar < 11
Print "Größer als 8"
Case Else
Print "Außerhalb von 1 bis 10"
End Select
end sub
```

IIf-Anweisung

Liefert eines zweier möglicher Funktionsergebnisse zurück, abhängig vom Wahrheitsgehalt eines zu überprüfenden Ausdrucks.

SYNTAX:
IIf (Expression, ExpressionTrue, ExpressionFalse)

PARAMETER:
Expression : Beliebiger Ausdruck, der überprüft werden soll. Ist dieser TRUE, liefert die Funktion das Ergebnis des Ausdrucks ExpressionTrue, ansonsten das Ergebnis des Ausdrucks ExpressionFalse.

ExpressionTrue, ExpressionFalse : Beliebige Ausdrücke, von denen einer in Abhängigkeit der Wahrheitsprüfung als Funktionsergebnis zurückgeliefert wird.

BEISPIEL:

```
sub ExampleIIF
MsgBox "Dokumentendruck: Seiten " & IIF( StarWriter.PrintSet-
tings.PrintReverse,"hintere zuerst","normal"),64,"StarWriter"
end sub
```

Schleifen

Hier finden Sie die Anweisungen für das Ausführen von Schleifen.

Do...Loop-Anweisung

Wiederholt die zwischen der Do- und der Loop-Anweisung aufgeführten Anweisungen solange (While) die angegebene Bedingung wahr ist oder bis (Until) die angegebene Bedingung wahr wird.

SYNTAX:
see Parameter

PARAMETER:
SYNTAX

Do [{While | Until} Bedingung = True]

Befehlsblock

[Exit Do]

Befehlsblock

Loop

oder

Do

Befehlsblock

[Exit Do]

Befehlsblock

Loop [{While | Until} Bedingung = True]

PARAM./ELEMENT

Bedingung : Ein Vergleichsausdruck, numerischer Ausdruck oder Zeichenfolgenausdruck, der wahr (True) oder falsch (False) ist.

Befehlsblock : Die Anweisungen, die wiederholt werden, solange die angegebene Bedingung wahr ist oder sobald die angegebene Bedingung wahr wird.

Die DO...LOOP-ANWEISUNG eignet sich, um eine Schleife zu konstruieren, die solange durchlaufen wird, bis oder solange eine bestimmte Bedingung erfüllt ist. Die Abbruchkriterien müssen entweder hinter der Do-Anweisung oder hinter der Loop-Anweisung angegeben werden; beides kombiniert ist nicht statthaft. Folgende vier Kombinationen sind also möglich.

SYNTAX

Do While Bedingung = True

...Befehlsblock

Loop

Das Programm überprüft, ob die Bedingung wahr ist. Ist sie falsch, geht es mit der Programmausführung direkt hinter Loop weiter. Ist die Bedingung wahr, wird die Schleife durchlaufen, das Programm trifft auf Loop und springt zurück zur Do-Anweisung. Ist die Bedingung jetzt immer noch wahr, wird die Schleife abermals durchlaufen, ansonsten fährt das Programm mit der Ausführung des hinter der Loop-Anweisung stehenden Befehls fort.

Kurz zusammengefaßt: Die Schleife wird nur dann und nur solange, wie die Bedingung wahr ist, ausgeführt.

Do Until Bedingung = True

...Befehlsblock

Loop

Das Programm überprüft, ob die Bedingung wahr ist. Ist sie es, geht es mit der Programmausführung direkt hinter Loop weiter. Ist die Bedingung nicht wahr, wird die Schleife durchlaufen, das Programm trifft auf Loop, springt zurück zur Do-Anweisung. Ist die Bedingung jetzt immer noch nicht wahr, wird die Schleife abermals durchlaufen, ansonsten fährt das Programm mit der Ausführung der hinter der Loop-Anweisung stehenden Anweisung fort.

Kurz zusammengefaßt: Die Schleife wird nur dann ausgeführt, wenn die Bedingung zunächst falsch ist, und dann solange bis die Bedingung wahr ist.

Do

...Befehlsblock

Loop While Bedingung = True

Das Programm tritt sofort in die Schleife ein. Trifft es auf Loop, überprüft es, ob die Bedingung wahr ist. Ist sie es, geht es mit der Programmausführung bei Do weiter - die Schleife wird wiederholt. Erst wenn die Bedingung hinter Loop falsch wird, tritt das Programm aus der Schleife aus.

Kurz zusammengefaßt: Die Befehle innerhalb der Schleife werden in jedem Fall einmal und solange wie die Bedingung wahr ist, ausgeführt.

Do

...Befehlsblock

Loop Until Bedingung = True

Das Programm tritt sofort in die Schleife ein. Trifft es auf Loop, überprüft es, ob die Bedingung wahr ist. Ist sie es nicht, geht es mit der Programmausführung bei Do weiter - die Schleife wird wiederholt. Erst, wenn die Bedingung hinter Loop wahr wird, tritt das Programm aus der Schleife aus.

Kurz zusammengefaßt: Die Befehle innerhalb der Schleife werden in jedem Fall einmal und solange bis die Bedingung wahr wird, ausgeführt.

Wichtigster Unterschied zwischen den ersten beiden und den letzten beiden Konstruktionen: Bei den ersten beiden Konstruktionen wird die Schleife nicht unbedingt, also nur in Abhängigkeit von der Bedingung ausgeführt. Bei den letzten beiden Konstruktionen werden die zwischen Do und Loop stehenden Befehle wenigstens ein einziges Mal ausgeführt.

Mit Hilfe der Anweisung Exit Do kann die Schleife unbedingt verlassen werden. Dieser Befehl befindet sich innerhalb eines Befehlsblocks, der durch Do und Loop umschlossen ist. In der Regel konstruieren Sie hier mit Hilfe der If...Then-Struktur eine Abbruchbedingung im Sinne von:

Do...

Befehle

If Bedingung = True Then Exit Do

Befehle

Loop...

BEISPIEL:

```
Sub ExampleDoLoop
Dim sDatei As String
Dim sPath As String
sPath = "c:\"
sDatei = Dir$( sPath ,22)
If sDatei <> "" Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
```

For...Next-Anweisung

Wiederholt die zwischen der For- und der Next-Anweisung aufgeführten Anweisungen so oft, bis der Schleifenzähler den angegebenen Wert erreicht hat.

SYNTAX:
see Parameter

PARAMETER:
SYNTAX:

For counter=start To end [Step step]

Befehlsblock

[Exit For]

Befehlsblock

Next [counter]

Variablen : Aufgabe

counter : Schleifenzähler, dem zunächst der Wert rechts des Gleichheitszeichens (start) zugewiesen wird. Erlaubt sind nur numerische Variablen. Der Schleifenzähler wird abhängig von der Variablen step solange erhöht oder vermindert, bis end überschritten (bei Erhöhung des Schleifenzählers) bzw. unterschritten wird (bei Verminderung des Schleifenzählers).

Start : Numerische Variable, die bestimmt, mit welchem Startwert die Schleife beginnen soll.

End : Numerische Variable, die angibt, bis zu welchem Grenzwert counter erhöht bzw. vermindert wird, bevor die Schleife beendet ist.

Step : Bestimmt, in welchen Schritten die Zählvariable counter hoch- bzw. heruntergezählt wird. Wird step nicht mit angegeben, erhöht sich counter jeweils um den Wert 1. end muß in diesem Fall natürlich größer als start sein. Möchten Sie counter herunterzählen, muß end größer als start angegeben werden. In diesem Fall muß step ein negativer Wert zugewiesen werden, und die Step-Anweisung kann dann auch nicht weggelassen werden.

Mit Hilfe der FOR...NEXT-Schleife lassen sich alle Anweisungen, die innerhalb eines Programmabschnittes durch diese beiden Schlüsselwörter eingekesselt sind, sooft wiederholen, wie es aus den Übergabeparametern hervorgeht.

Beim Herunterzählen der Zählervariable überprüft StarBasic zur Laufzeit eine Bereichsunter- bzw. Überschreitung. Sie brauchen selber nicht dafür Sorge zu tragen, daß counter genau den Wert erreicht, der durch end bestimmt wurde. Die Abbruchbedingung ist in dem Moment gegeben, in dem end durch counter über- (beim Heraufzählen durch positives step) oder unterschritten (beim Herunterzählen durch negatives step) wird.

FOR...NEXT-Schleifen lassen sich ineinander verschachteln. Geben Sie keine Variable hinter einer Next-Anweisung an, bezieht sich das Next immer automatisch auf das zuletzt aufgetretene For.

Bestimmen Sie eine Schrittweite von 0, dann werden die Anweisungen zwischen FOR und NEXT endlos ausgeführt.

Nachdem die Schleife das letzte Mal durchlaufen wurde, erhöht bzw. verringert die NEXT-Anweisung die Variable count ein letztes Mal. Bei einem Schleifendurchlauf von 1 bis 10 (For counter=1 To 10 [Step 1]) mit einer Schrittweite von 1 geht counter mit dem Wert 11 aus der Schleife heraus.

Mit Hilfe der EXIT FOR-Anweisung kann die Schleife unbedingt verlassen werden. Dieser Befehl befindet sich innerhalb eines Befehlsblocks, der durch For und Next umschlossen wird. In der Regel konstruieren Sie hier mit Hilfe der IF...THEN-Struktur eine Abbruchbedingung im Sinne von:

For...

Befehle

If Bedingung = True Then Exit For

Befehle

Next

Hinweis für die Verschachtelung von FOR...NEXT-Schleifen: Beim unbedingten Verlassen einer Schleife durch EXIT FOR wird grundsätzlich nur eine Schleife beendet.

Das folgende Beispiel nutzt zwei ineinander verschachtelte Schleifen, um ein Zeichenfolgefeld mit 10 Elementen (sEntry()), die zunächst mit verschiedenen Namen vorbelegt werden, zu sortieren:

BEISPIEL:

```
Sub ExampleSort
Dim sEntry(10) As String
Dim iCount As Integer
Dim iCount2 As Integer
Dim sTemp As String
sEntry(1) = "Klaus"
sEntry(2) = "Uwe"
sEntry(3) = "Thorsten"
sEntry(4) = "Manfred"
sEntry(5) = "Erwin"
sEntry(6) = "Timo"
sEntry(7) = "Karsten"
sEntry(8) = "Regina"
sEntry(9) = "Christiane"
sEntry(10) = "Uta"
For iCount = 1 To 10
For iCount2 = iCount + 1 To 10
If sEntry(iCount) > sEntry(iCount2) Then
sTemp = sEntry(iCount)
sEntry(iCount) = sEntry(iCount2)
sEntry(iCount2) = sTemp
End If
Next iCount2
Next iCount
For iCount = 1 To 10
Print sEntry(iCount)
Next iCount
End Sub
```

While...Wend -Anweisung

Trifft das Programm auf eine While-Anweisung, wird überprüft, ob die Bedingung wahr ist. Ist sie falsch, geht es mit der Programmausführung direkt hinter Wend weiter. Ist die Bedingung wahr, wird die Schleife durchlaufen, das Programm trifft auf Wend und springt zurück zur WHILE-Anweisung. Ist die Bedingung jetzt immer noch wahr, wird die Schleife abermals durchlaufen, ansonsten fährt das Programm mit der Ausführung der hinter der WEND -Anweisung stehenden Anweisung fort. Kurz zusammengefaßt: Die Schleife wird nur dann und nur solange wie die Bedingung wahr ist, ausgeführt.

Anders als bei Do...Loop kann eine WHILE...WEND-Schleife nicht mit Exit verlassen werden. Sie sollten ebenfalls davon Abstand nehmen, eine WHILE...WEND-Schleife mit GoTo zu verlassen, da dies zu Laufzeitfehlern führen kann.

In jedem Fall ist die Verwendung von DO...LOOP flexibler und damit empfehlenswert.

SYNTAX:
While Bedingung [Statement] Wend

BEISPIEL:

```
Sub ExampleWhileWend
Dim stext As String
Dim iRun As Integer
sText ="Das ist ein kleiner Text"
iRun = 1
while iRun < Len(sText)
   if Mid(sText,iRun,1 )<> " " then Mid( sText ,iRun, 1, Chr( 1
+ Asc( Mid(sText,iRun,1 )) )
   iRun = iRun + 1
Wend
MsgBox sText,0,"Text verschlüsselt"
end sub
```

Sprünge

Hier finden Sie die Anweisungen für das Ausführen von Sprüngen.

GoSub...Return-Anweisung

Ruft innerhalb einer Unterroutine (Sub) oder einer Funktion (Function) einen Programmteil auf, der durch eine Marke gekennzeichnet ist. Die dort nachfolgenden Befehle werden bis zur Return-Anweisung abgearbeitet; die Programmausführung wird danach hinter der GOSUB-Anweisung.

SYNTAX:
see Parameter

PARAMETER:
Sub/Function

Befehlsblock

GoSub Marke

Befehlsblock

GoSub Marke

Exit Sub/Function

Marke:

Befehlsblock

Return

End Sub/Function

Mit Hilfe der GoSub-Anweisung rufen Sie innerhalb einer Unterroutine (Sub) oder einer Funktion (Function) eine sogenannte "lokale Unterroutine" auf. Der Beginn einer derartigen Unterroutine wird mit einer Marke markiert. Um eine Marke zu setzen, benennen Sie sie beliebig und schließen sie mit einem Doppelpunkt (":") ab. Das Programm fährt mit der Programmausführung, nachdem es auf eine GoSub-Anweisung gestoßen ist, direkt mit dem Programmtext fort, der nach der Marke folgt. Die dort folgenden Anweisungen und Funktionen werden solange abgearbeitet, bis das Programm auf die Return-Anweisung trifft. In diesem Moment kehrt das Programm zur GoSub-Anweisung zurück und fährt mit der Abarbeitung der Befehle hinter der GoSub-Anweisung fort.

Trifft das Programm auf eine Return-Anweisung, ohne daß zuvor ein GoSub erfolgte, meldet StarBasic eine Fehlermeldung. Sie müssen selber dafür Sorge tragen, daß Ihr Programm ein Unterprogramm oder eine Funktion mit der Exit Sub- bzw. Exit Function-Anweisung (siehe dort) verläßt, bevor es auf einen Programmteil stößt, der mit Return abgeschlossen ist.

Eigentlich sind GoSub und Return Relikte älterer Basicdialekte, die nur aus Kompatibilitätsgründen erhalten geblieben sind. Verwenden Sie GoSub und Return sehr häufig, wird Ihr Programm für Sie, aber auch für andere Programmierer, schwer lesbar. Sie erreichen eine viel größere Ordnung und auch einen sehr viel saubereren Programmierstil, wenn Sie Unterroutinen in Subs oder Functions (siehe dort) anordnen.

Das folgende Beispielprogramm demonstriert die Anwendung von GoSub und Return. Es fordert den Benutzer auf, zweimal hintereinander eine Zahl einzugeben, und berechnet mit Hilfe eines Programmteils, der zweimal verwendet wird, jeweils die Quadratwurzel aus der angegebenen Zahl.

BEISPIEL:

```
Sub ExampleGoSub
dim iInputa as Single
dim iInputb as Single
dim iInputc as Single
iInputa = Int(InputBox$ ("Bitte geben Sie Zahl ein: ","Zahlen-
eingabe"))
iInputb = Int(InputBox$ ("Bitte geben Sie eine weitere Zahl
ein: ","Zahleneingabe"))
iInputc=iInputa
GoSub Quadratwurzel
Print "Die Quadratwurzel aus";iInputa;" ist";iInputc
iInputc=iInputb
GoSub Quadratwurzel
Print "Die Quadratwurzel aus";iInputb;" ist";iInputc
Exit Sub
Quadratwurzel:
iInputc=sqr(iInputc)
Return
End Sub
```

GoTo-Anweisung

Sorgt dafür, daß das Programm innerhalb einer Unterroutine (Sub) oder einer Funktion (Function) mit der Abarbeitung des Programmteils fortfährt, der durch eine Marke gekennzeichnet ist.

SYNTAX:
see Parameter

PARAMETER:
Sub/Function

Befehlsblock

GoTo Marke1

MARKE2:

Befehlsblock

Exit Sub

MARKE1:

Befehlsblock

GoTo Marke2

End Sub/Function

Mit Hilfe der GoTo-Anweisung weisen Sie StarBasic an, mit der Programmausführung an einem ganz anderen Teil des StarBasic-Programms innerhalb einer Unterroutine (Sub) oder Funktion (Function) fortzufahren. Die Stelle, an der das Programm mit der Ausführung fortfahren soll, ist mit einer Marke gekennzeichnet. Um eine Marke zu setzen, benennen Sie sie beliebig und schließen Sie mit einem Doppelpunkt (":") ab.

Bitte beachten Sie, daß Sie mit der GoTo-Anweisung niemals aus einem Unterprogramm oder einer Funktion hinausspringen dürfen.

Die GoTo-Anweisung ist ein Relikt älterer Basicdialekte, die nur aus Kompatibilitätsgründen erhalten geblieben sind. Verwenden Sie GoTo sehr häufig, wird Ihr Programm für Sie, aber auch für andere Programmierer, schwer lesbar. Versuchen Sie sich daher einen Programmstil anzueignen, der möglichst wenig, am besten gar keinen Gebrauch der GoTo -Anweisung macht. Einzige Ausnahmen sind On...GoTo und On Error GoTo (siehe dort).

Beispiel:

```
Sub ExampleOnGosub
Dim iVar As Integer
Dim sVar As String
iVar = 2
sVar =""
On iVar GoSub Sub1, Sub2
On iVar GoTo Line1, Line2
Exit Sub
Sub1:
sVar =sVar &  " Von Sub 1 nach" : Return
Sub2:
sVar =sVar &  " Von Sub 2 nach" : Return
Line1:
sVar =sVar &  " Marke 1"
Line2:
sVar =sVar &  " Marke 2"
Print sVar
End Sub
```

On...GoSub-Anweisung; On...GoTo-Anweisung

Verzweigt an eine von mehreren angegebenen anderen Stellen des Programms in Abhängigkeit eines numerischen Ausdrucks.

Syntax:
On N GoSub Label1[, Label2[, Label3[,...]]]

On NumAusdruck GoTo Label1[, Label2[, Label3[,...]]]

PARAMETER:
NumAusdruck : Beliebiger numerischer Ausdruck zwischen 0 und 255, der bestimmt, zu welcher der angegebenen Sprungmarken gesprungen werden soll. Ist NumAusdruck 0, wird der Befehl ausgelassen. Ist er größer 0, wird zum entsprechenden Label gesprungen, dessen Positionsnummer in der Liste dem Ausdruck entspricht (1 = 1. Label wird verwendet; 2 = 2. Label wird verwendet etc.)

Label : Sprung-Label gemäß GOTO- bzw. GOSUB-Konvention.

Es gelten die Konventionen für GOTO bzw. GOSUB.

BEISPIEL:

```
Sub ExampleOnGosub
Dim iVar As Integer
Dim sVar As String
iVar = 2
sVar =""
On iVar GoSub Sub1, Sub2
On iVar GoTo Line1, Line2
Exit Sub
Sub1:
sVar =sVar &  " Von Sub 1 nach" : Return
Sub2:
sVar =sVar &  " Von Sub 2 nach" : Return
Line1:
sVar =sVar &  " Marke 1" : GoTo Ende
Line2:
sVar =sVar &  " Marke 2"
Ende:
MsgBox sVar,0,"On...Gosub"
End Sub
```

Weitere Anweisungen

Hier finden Sie weitere Anweisungen, die keiner der anderen Runtime-Kategorien zuzuordnen sind.

Call-Anweisung

Übergibt die Programmsteuerung einer Sub, einer Function oder einer DLL.

SYNTAX:
[Call] Name [Parameter]

PARAMETER:
Name : Name der Sub, der Function oder der DLL

Parameter : Parameter, die die aufgerufene Routine erwartet. Die Typen der Parameter sowie deren Anzahl sind dabei von der aufgerufenen Routine abhängig.

Das Schlüsselwort kann beim Aufruf einer Routine weggelassen werden. Wird eine Function als Ausdruck gerufen, so müssen die Parameter an der Function eingeklammert sein. Wird eine DLL gerufen, so muß diese zuvor durch die DECLARE-ANWEISUNG deklariert worden sein.

BEISPIEL:

```
Sub ExampleCall
Dim sVar As String
sVar = "StarOffice"
Call f_callFun sVar
end Sub
Sub f_callFun (sText as String)
Msgbox sText
end sub
```

Choose-Funktion

Gibt aus einer Argumentliste den gewählten Wert zurück.

SYNTAX:
Choose (Index, Selection1[, Selection2, ... [,Selection_n]])

PARAMETER:
Index : Ein numerischer Ausdruck, der auf den Wert verweist, der zurückgegeben werden soll.

Selection1 : Ein beliebiger Ausdruck, der einen der möglichen Auswahlwerte enthält.

Die CHOOSE-Funktion gibt den Wert aus der Ausdruckliste zurück, auf den der übergebene Index verweist. Ist Index der Wert 1 zugewiesen, gibt die Funktion den ersten Auswahlwert der Liste zurück. Ist Index der Wert 2 zugewiesen, wird der zweite Auswahlwert zurückgegeben usw.

Die CHOOSE-Funktion gibt Null zurück, wenn Index kleiner als 1 oder größer als die Anzahl der aufgeführten Auswahlmöglichkeiten ist.

Das nachstehende Beispiel verwendet die CHOOSE-Funktion, um eine Zeichenfolge aus mehreren zu wählen, die zusammen ein Menü ergeben:

BEISPIEL:

```
Sub ExampleChoose
Dim sReturn As String
sReturn = ChooseMenu(2)
```

```
Print sReturn
end sub
Function ChooseMenu(Index As Integer)
ChooseMenu = Choose(Index, "Quick-Format", "Save-Format",
"System-Format")
End Function
```

Declare-Anweisung

Deklariert und definiert eine Unterroutine aus einer DLL (Dynamic Link Library) für den Aufruf aus StarBasic heraus.

Siehe auch: FreeLibrary

SYNTAX:
Declare {Sub | Function} Name Lib "Libname" [Alias "Aliasname"] [Parameter] [As Typ]

PARAMETER:
PARAM./ELEMENT

Name : Name, der nicht dem Namen der Unterroutine in der DLL entsprechen muß, unter dem sie aber von Basic aus aufgerufen werden soll.

Aliasname : Name der Unterroutine, so wie sie in der DLL mit Namen definiert ist.

Libname : Datei- oder Systemname der Dynamic Link Library. Diese Library wird automatisch geladen, sobald die Funktion das erste Mal benötigt wird.

Argumentliste : Parameter, die die aufgerufene Routine erwartet. Die Typen der Parameter sowie deren Anzahl sind dabei von der aufgerufenden Routine abhängig.

Typ : Bestimmt bei Funktionsdeklarationen, welchen Rückgabetyp die Funktion liefert. Kann ausgelassen werden, wenn hinter Name - wie bei Variablen - ein Typendeklarationszeichen angegeben wird.

> Parameter müssen durch das Schlüsselwort BYVAL gekennzeichnet werden, wenn sie nicht als Referenz sondern als Wert an die Unterroutine übergeben werden sollen.

BEISPIEL:

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep"
( long )
Sub ExampleDeclare
   Dim lValue As Long
   lValue = 5000
```

```
   MyMessageBeep( lValue )
   FreeLibrary("user32.dll" )
End Sub
```

End-Anweisung

Beendet eine Prozedur oder einen Prozedurabschnitt (Block).

SYNTAX:
End, End Function, End If, End Select, End Sub

PARAMETER:
Sie können die End-Anweisung wie folgt verwenden:

ANWEISUNG

End : Ist niemals erforderlich, kann aber an einer beliebigen Stelle innerhalb einer Prozedur angegeben werden, um den Programmablauf zu stoppen.

End Function : Beendet eine FUNCTION-Anweisung.

End If : Kennzeichnet das Ende eines IF...THEN...ELSE-Blockes.

End Select : Kennzeichnet das Ende eines SELECT CASE-Blockes.

End Sub : Beendet eine SUB-Anweisung.

BEISPIEL:

```
Sub ExampleRandomSelect
Dim iVar As Integer
iVar = Int((15 * Rnd) -2)
Select Case iVar
Case 1 To 5
Print "Zahl von 1 bis 5"
Case 6, 7, 8
Print "Zahl von 6,7, oder  8"
Case Is > 8 And iVar < 11
Print "Größer als 8"
Case Else
Print "Außerhalb von 1 bis 10"
End Select
end sub
```

Exit-Anweisung

Beendet einen Block, der mit DO...LOOP, FOR...NEXT, Function oder Sub definiert wurde.

SYNTAX:
see Parameter

PARAMETER:
EXIT DO

Ist nur innerhalb einer DO...LOOP-Anweisung zulässig und verläßt diese. Danach wird mit der ersten Anweisung fortgesetzt, die der Loop-Anweisung folgt. Sind DO...LOOP-Anweisungen verschachtelt, so übernimmt die nächsthöhere Verschachtelungsebene die Steuerung.

EXIT FOR

Ist nur innerhalb einer FOR...NEXT-Schleife zulässig und verläßt diese. Danach wird mit der ersten Anweisung fortgesetzt, die der Next-Anweisung folgt. Sind FOR...NEXT-Anweisungen verschachtelt, so übernimmt die nächsthöhere Verschachtelungsebene die Steuerung.

EXIT FUNCTION

Beendet mit sofortiger Wirkung die Function-Prozedur. Die Ausführung wird mit der Anweisung fortgesetzt, die dem Aufruf der FUNCTION-Prozedur folgt.

EXIT SUB

Beendet mit sofortiger Wirkung die SUB-Prozedur. Die Ausführung wird mit der Anweisung fortgesetzt, die dem Aufruf der SUB-Prozedur folgt.

Die Exit-Anweisung definiert nicht das Ende einer Struktur und darf nicht mit der End-Anweisung verwechselt werden.

BEISPIEL:

```
Sub ExampleExit
Dim sReturn As String
Dim sListArray(10) as String
Dim siStep as Single
For siStep = 0 to 10  REM Array mit Testdaten füllen
sListArray(siStep) = chr(siStep + 65)
msgbox sListArray(siStep)
next siStep
sReturn = LinSearch(sListArray(), "B")
Print sReturn
end sub
Function LinSearch( sList(), sItem As String ) as integer
dim iCount as Integer
REM LinSearch durchsucht ein TextArray:sList() nach einem
Texteintrag:
REM Rückgabe ist der Index des Eintrags oder 0 ( Null)
for iCount=1 to Ubound( sList() )
if sList( iCount ) = sItem then
exit for REM sItem gefunden
```

```
end if
next iCount
if iCount = Ubound( sList() ) then iCount = 0
LinSearch = iCount
end function
```

FreeLibrary-Funktion

Gibt per Declare-Anweisung geladene DLLs frei. Eine freigegebene DLL wird beim Aufruf einer ihrer Funktionen automatisch neu geladen. Siehe auch: Declare

SYNTAX:
FreeLibrary (LibName As String)

PARAMETER:
LibName : Zeichenfolgenausdruck, der den Namen der DLL angibt.

Mit FreeLibrary können nur zur Basic-Laufzeit angezogene DLLs freigegeben werden.

BEISPIEL:

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep"
( long )
Sub ExampleDeclare
   Dim lValue As Long
   lValue = 5000
   MyMessageBeep( lValue )
   FreeLibrary("user32.dll" )
End Sub
```

Function-Anweisung

Definiert eine Unterroutine, die als Ausdruck verwendet werden kann, also einen Rückgabetyp liefert.

SYNTAX:
see Parameter

PARAMETER:
Syntax

Function Name[(VarName1 [As Typ][, VarName2 [As Typ][,...]]]) [As Typ]

Befehlsblock

[Exit Function]

Befehlsblock

End Function

Parameter

Name : Name der Unterroutine, der den Wert aufnimmt, der von der Funktion zurückgegeben wird.

VarName : Parameter, den die Unterroutine erwartet.
Typ : Typendeklarationsschlüsselwort.

Beispiel:

```
Sub ExampleExit
Dim sReturn As String
Dim sListArray(10) as String
Dim siStep as Single
For siStep = 0 to 10  REM Array mit Testdaten füllen
sListArray(siStep) = chr$(siStep + 65)
msgbox sListArray(siStep)
next siStep
sReturn = LinSearch(sListArray(), "B")
Print sReturn
end sub
Function LinSearch( sList(), sItem As String ) as integer
dim iCount as Integer
REM Linsearch durchsucht ein TextArray:sList() nach einem
TextEintrag:
REM Rückgabe ist der Index des Eintrags oder 0 ( Null)
for iCount=1 to Ubound( sList() )
if sList( iCount ) = sItem then
exit for REM sItem gefunden
end if
next iCount
if iCount = Ubound( sList() ) then iCount = 0
LinSearch = iCount
end function
```

Rem-Anweisung

Erlaubt die Definition einer Programmzeile oder des auf REM folgenden Teils einer Programmzeile als Kommentar. Der Rest der Programmzeile nach dem REM wird ignoriert.

Syntax:
Rem Text

Parameter:
Text: Beliebiger Text, der rein dokumentierende Wirkung hat.

Anstelle von Rem können Sie auch das einfache Anführungszeichen zum Auskommentieren verwenden. Dieses Zeichen kann auch direkt rechts auf Programmcode folgend in einer Programmzeile stehen und einen Kommentar bis zum Ende der Zeile einleiten.

BEISPIEL:

```
Sub ExampleMid
DIM sVar As String
sVar = "Star Division"
Print Mid(sVar,3,5) REM liefert "ar Di"
REM Hier wird nichts getan
end sub
```

Stop-Anweisung

Beendet das Programm.

SYNTAX:
Stop

BEISPIEL:

```
Sub ExampleStop
Dim iVar As Single
iVar = 36
Stop
Msgbox Sqr(iVar)
end sub
```

Sub-Anweisung

Definiert eine Unterroutine.

SYNTAX:
see Parameter

PARAMETER:
Syntax

Sub Name[(VarName1 [As Typ][, VarName2 [As Typ][,...]])]

Befehlsblock

[Exit Sub]

Befehlsblock

End Sub

PARAMETER

Name : Name der Unterroutine.

VarName : Parameter, die die Unterroutine erwartet.

Typ : Typendeklarationsschlüsselwort.

BEISPIEL:

```
Sub ExampleShell
Shell( Application.PathSettings.Modules +"\" + "Shelp.exe")
end sub
```

Switch-Funktion

Wertet eine Argumentliste mit Ausdrücken und Werten aus, in der jeweils ein Wert einem Ausdruck folgt. Ein Ausdruck und ein Wert bilden eine Zuordnungseinheit. Die Switch-Funktion gibt den Wert zurück, der dem Ausdruck zugeordnet ist, der dieser Funktion übergeben wurde.

SYNTAX:
Switch (Expression1, Value1[, Expression2, Value2[..., Expression_n, Value_n]])

PARAMETER:
Die Ausdrücke der Argumentliste der SWITCH-Funktion werden in der Reihenfolge ihrer Anordnung von links nach rechts ausgewertet. Es wird der Wert zurückgegeben, der dem der Funktion übergebenen Ausdruck zugeordnet ist. Werden die Zuordnungseinheiten nicht paarweise angegeben, tritt ein Laufzeitfehler auf.

Expression : Der Ausdruck, dem ein Wert zugewiesen ist, der zurückgegeben werden kann.

Value : Der Wert, der zurückgegeben wird, sofern er dem Ausdruck zugeordnet ist, der dem der Funktion übergebenen Ausdruck entspricht.

Das nachstehende Beispiel verwendet die SWITCH-Funktion, um dem der Funktion übergebenen Namen den entsprechenden Genus zuzuordnen:

BEISPIEL:

```
Sub ExampleSwitch
Dim sGenus As String
sGenus = GetGenusIndex( "Willi" )
MsgBox sGenus
End Sub
Function GetGenusIndex (sName As String) As String
GetGenusIndex = Switch(sName = "Elfriede", "weiblich", sName =
"Willi", "männlich")
End Function
```

With-Anweisung

Bestimmt ein Objekt zum Vorgabeobjekt, auf das sich alle Eigenschaften oder Methoden bis zur End With-Anweisung beziehen, solange keine weiteren Objektnamen angegeben werden.

SYNTAX:
With Object Statementblock End With

PARAMETER:
Verwenden Sie With und End With, wenn Sie viele Eigenschaften oder Methoden für ein bestimmtes Objekt verwenden. Sie sparen durch diese Möglichkeit viel Schreibaufwand.

BEISPIEL:

```
Sub ExampleWith
Documents.Add("swriter4")
wait 1
With ActiveWindow.Selection.Text
.Insert( "Das ist ein kleiner Text" )
.InsertPara()
end with
end sub
```

Variablen

Hier finden Sie alle Anweisungen und Funktionen, mit denen Sie Variablen verwalten können. So können Sie beispielsweise Variablen deklarieren oder definieren, Variablen von einem Typ in den anderen umwandeln, den Variablentyp feststellen etc.

Diese Funktionen und Anweisungen stehen zur Manipulation und Prüfung von Variablen zur Verfügung:

CBool-Funktion

Konvertiert einen Zeichenfolgenvergleich, respektive einen Vergleich numerischer Werte in einen boolschen Ausdruck oder wandelt einen einzelnen numerischen Ausdruck in einen boolschen Ausdruck um.

SYNTAX:
CBool (Expression1 {= | <> | < | > | <= | >=} Expression2) or CBool (Number)

RÜCKGABE:
Bool

PARAMETER:
Expression1, Expression2 : Beliebige numerische Ausdrücke oder Zeichenfolgenausdrücke, die miteinander verglichen werden sollen. Stimmen beide Ausdrücke miteinander überein, gibt die CBOOL-Funktion den Wert TRUE zurück. Andernfalls wird FALSE zurückgegeben.

Number : Beliebiger numerischer Ausdruck, der konvertiert werden soll. Ist der Ausdruck gleich 0 wird der Wert FALSE zurückgegeben. Wird der Funktion ein Ausdruck ungleich 0 übergeben, gibt die Funktion den Wert TRUE zurück.

Das folgende Beispiel verwendet die CBOOL-Funktion zur Auswertung des von der INSTR-Funktion zurückgegebenen Wertes. Geprüft wird, ob in einem vom Anwender eingegebenen Satz das Wort UND vorkommt.

BEISPIEL:

```
Sub ExampleCBool
Dim sText As String
sText = InputBox("Bitte geben Sie einen kurzen Satz ein:")
Rem Prüfen, ob in dem vom Anwender eingegebenen Satz das Wort
»und« vorkommt.
Rem Anstatt der Befehlszeile
Rem             If Instr(Eingabe, "und")<>0 Then...
Rem wird die CBool-Funktion wie folgt verwendet:
If CBool(Instr(sText, "und")) Then
MsgBox "In dem von Ihnen eingegebenen Satz kommt das Wort »und«
vor!"
EndIf
End Sub
```

CDate-Funktion

Konvertiert einen beliebigen numerischen Ausdruck oder einen Zeichenkettenausdruck in einen Wert vom Datentyp Date.

SYNTAX:
CDate (Expression)

RÜCKGABE:
Date

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenkettenausdruck, der konvertiert werden soll.

Werden Zeichenkettenausdrücke umgewandelt, muß die zu konvertierende Zeiteingabe im Format DD.MM.YYYY HH.MM.SS vorliegen, so wie es die Konventionen zur DATEVALUE- bzw. TIMEVALUE-Funktion vorsehen. Werden

numerische Ausdrücke umgewandelt, bildet die Vorkommazahl des Wertes die Anzahl der Tage, gerechnet vom 31.12.1899. Die Nachkommazahl bestimmt die Uhrzeit.

BEISPIEL:

```
sub ExampleCDate
msgbox cDate(1000.25) REM 26.09.1902 06:00:00
msgbox cDate(1001.26) REM 27.09.1902 06:14:24
end sub
```

CDbl-Funktion

Konvertiert einen beliebigen numerischen Ausdruck oder einen String-Ausdruck in einen Double-Typ.

SYNTAX:
CDbl (Expression)

RÜCKGABE:
Double

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenkettenausdruck, der konvertiert werden soll. Wird ein Zeichenkettenausdruck umgewandelt, so muß die Zahl im Klartext vorliegen (also z. B. "123,5"). Bitte beachten Sie dabei, daß die jeweilige Zahlennotation gemäß der Landeseinstellung Ihres Computers (im Gegensatz zur VAL-Funktion; dort müßte es "123.5" heißen) berücksichtigt wird.

BEISPIEL:

```
Sub ExampleCountryConvert
Msgbox CDbl(1234.5678)
Msgbox CInt(1234.5678)
Msgbox CLng(1234.5678)
end sub
```

CInt-Funktion

Konvertiert einen beliebigen numerischen Ausdruck oder einen Zeichenkettenausdruck in einen Integer-Typ.

SYNTAX:
CInt (Expression)

RÜCKGABE:
Integer

PARAMETER:
Expression : Beliebiger numerischer Ausdruck, der konvertiert werden soll. Überschreitet EXPRESSION den Wertebereich zwischen -32768 und 32767, meldet StarBasic einen Überlauffehler. Wird ein Zeichenkettenausdruck umgewandelt, so muß die Zahl im Klartext vorliegen (also z. B. "123,5"). Bitte beachten Sie dabei, daß die jeweilige Zahlennotation gemäß der Landeseinstellung Ihres Computers (im Gegensatz zur VAL-Funktion; dort müßte es "123.5" heißen) berücksichtigt wird.

Eventuell vorhandene Nachkommastellen bleiben bei der Konvertierung unberücksichtigt und fallen weg.

BEISPIEL:

```
Sub ExampleCountryConvert
Msgbox CDbl(1234.5678)
Msgbox CInt(1234.5678)
Msgbox CLng(1234.5678)
end sub
```

CLng-Funktion

Konvertiert einen beliebigen numerischen Ausdruck oder einen Zeichenkettenausdruck in einen Long-Integer-Typ.

SYNTAX:
CLng (Expression)

RÜCKGABE:
Long

PARAMETER:
Expression : Beliebiger numerischer Ausdruck, der konvertiert werden soll. Überschreitet EXPRESSION den gültigen Long-Int-Wertebereich zwischen -2.147.483.648 und 2.147.483.647, meldet StarBasic einen Überlauffehler. Wird ein Zeichenkettenausdruck umgewandelt, so muß die Zahl im Klartext vorliegen (also z. B. "123,5"). Bitte beachten Sie dabei, daß die jeweilige Zahlennotation gemäß der Landeseinstellung Ihres Computers (im Gegensatz zur VAL-Funktion; dort müßte es "123.5" heißen) berücksichtigt wird.

Eventuell vorhandene Nachkommastellen bleiben bei der Konvertierung unberücksichtigt und fallen weg.

BEISPIEL:

```
Sub ExampleCountryConvert
Msgbox CDbl(1234.5678)
Msgbox CInt(1234.5678)
Msgbox CLng(1234.5678)
end sub
```

Const-Anweisung

Definiert eine Zeichenkette als Konstante.

SYNTAX:
Const Text = Expression

PARAMETER:
Text : Beliebiger Konstantenname; es gelten für die Benennung eines Konstantennamens die gleichen Konventionen wie für Variablen.

Konstanten sind eine "Abart" von Variablen und dienen nur der besseren Lesbarkeit eines Programms. Konstanten werden nicht als bestimmter Typ definiert, sondern werden als reiner Programmcodeplatzhalter verwendet. Eine Konstante kann nur ein einziges Mal im Programm definiert und dann nie wieder umdefiniert werden. Sie definieren eine Konstante mit folgender Befehlszeile:

CONST ConstName=Ausdruck

Dabei ist es völlig egal, um was für einen Ausdruck es sich handelt. Wird ein Programm gestartet, so wird StarBasic intern den Programm-Code so umarbeiten, daß jedesmal, wenn eine Konstante verwendet wurde, der zuvor festgelegte Ausdruck dafür eingesetzt wird. Im Prinzip wird intern ein "Suchen/Ersetzen" durchgeführt. Der Konstantenname, wann immer er im Basic-Programm-Code verwendet wurde, wird durch den Ausdruck ersetzt. Aus diesem Grund spielt der Typ, der einer Konstanten zugewiesen wird, auch überhaupt keine Rolle, da intern sowieso nur der Ausdruck als solcher verwendet wird.

BEISPIEL:

```
Sub ExampleConst
Const iVar = 1964
Msgbox iVar
Const sVar = "StarOffice", dVar As Double = 4.00
Msgbox sVar & " " & dVar
end sub
```

CreateObject-Funktion

Erstellt ein OLE-Objekt.

SYNTAX:
CreateObject (Class)

PARAMETER:
Class : Die der Funktion zu übergebende Klasse setzt sich aus den Argumenten:

APPLIKATIONSNAME.OBJEKTTYP

zusammen. Die Funktion liefert das Objekt zurück, die mit dem SET-Befehl einer Objektvariablen zugewiesen werden muß.

CSng-Funktion

Konvertiert einen beliebigen numerischen Ausdruck oder einen String-Ausdruck in einen Single-Typ.

SYNTAX:
CSng (Expression)

RÜCKGABE:
Single

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenkettenausdruck, der konvertiert werden soll. Wird ein Zeichenkettenausdruck umgewandelt, so muß die Zahl im Klartext vorliegen (also z.B. "123,5"). Bitte beachten Sie dabei, daß die jeweilige Zahlennotation gemäß der Landeseinstellung Ihres Computers (im Gegensatz zur VAL-Funktion; dort müßte es "123.5" heißen) berücksichtigt wird.

BEISPIEL:

```
Sub ExampleCSNG
Msgbox CDbl(1234.5678)
Msgbox CInt(1234.5678)
Msgbox CLng(1234.5678)
Msgbox CSng(1234.5678)
end sub
```

CStr-Funktion

Konvertiert einen beliebigen numerischen Ausdruck in einen Zeichenkettenausdruck.

SYNTAX:
CStr (Expression)

RÜCKGABE:
String

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenkettenausdruck, der konvertiert werden soll. Es gilt:

Typ :	Konvertierung ergibt...
Boolean :	...eine Zeichenkette, die entweder True oder False lautet.
Date :	...eine Zeichenkette, die das Datum und die Zeit in der normalen Schreibweise enthält.
Null :	...einen Laufzeitfehler.
Empty :	...eine leere Zeichenkette (Null-String).
beliebig :	...die entsprechende Zahl als Zeichenkette in der jeweils üblichen Landesnotation.

Nullen am Ende der Nachkommazahl einer Gleitkommazahl werden nicht in die Zeichenfolge übernommen, die an den Aufrufer zurückgegeben wird.

BEISPIEL:

```
Sub ExampleCSTR
Dim sVar As String
Msgbox CDbl(1234.5678)
Msgbox CInt(1234.5678)
Msgbox CLng(1234.5678)
Msgbox CSng(1234.5678)
sVar = CStr(1234.5678)
MsgBox sVar
end sub
```

DefBool-Anweisung

Legt den Vorgabevariablentyp fest.

Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

Syntax:
Defxxx Characterrange1[, Characterrange2[,...]]

Parameter:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

Schlüsselwort : voreingestellter Variablentyp

DefBool : Boolean

Beispiel:

```
REM Prefix-Definitionen für die verschiedenen Typen von Variablen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefBool
bOK=TRUE    REM bOK ist implizit eine Boolesche Variable
end sub
```

DefDate-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

Syntax:
Defxxx Characterrange1[, Characterrange2[,...]]

Parameter:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

Schlüsselwort : voreingestellter Variablentyp

DefDate : Date

Beispiel:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefDate
tDate=Date    REM tDate ist implizit eine Datumsvariable
end sub
```

DefDbl-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

Syntax:
Defxxx Characterrange1[, Characterrange2[,...]]

Parameter:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

Schlüsselwort : voreingestellter Variablentyp

DefDbl : Double

Beispiel:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefDBL
dValue=1.23e43    REM dValue ist implizit eine Variable vom Typ
Double
end sub
```

DefInt-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

SYNTAX:
Defxxx Characterrange1[, Characterrange2[,...]]

PARAMETER:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

SCHLÜSSELWORT : VOREINGESTELLTER VARIABLENTYP

DefInt : Integer

BEISPIEL:

```
REM Prefix-Definitionen für die verschiedenen Typen von Variablen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefInt
iCount=200    REM iCount ist implizit eine ganzahlige Variable
end sub
```

DefLng-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

SYNTAX:
Defxxx Characterrange1[, Characterrange2[,...]]

PARAMETER:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Va-

riablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

SCHLÜSSELWORT : VOREINGESTELLTER VARIABLENTYP

DefLng : Long

BEISPIEL:

```
REM Prefix-Definitionen für die verschiedenen Typen von Variablen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefLng
lCount=123456789    REM lCount ist implizit eine ganzahlige Variable mit großem Zahlbereich
end sub
```

DefObj-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

SYNTAX:
Defxxx Characterrange1[, Characterrange2[,...]]

PARAMETER:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

SCHLÜSSELWORT : VOREINGESTELLTER VARIABLENTYP

DefObj : Object

Beispiel:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefObj
oWin=ActiveWindow    REM oWin ist implizit eine Objekt-Variable
end sub
```

DefSng-Anweisung

Legt den Vorgabevariablentyp fest.

Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

Syntax:
Defxxx Characterrange1[, Characterrange2[,...]]

Parameter:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

Schlüsselwort : voreingestellter Variablentyp

DefBool : Boolean

Beispiel:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefBool
bOK=TRUE    REM bOK ist implizit eine Boolesche Variable
end sub
```

DefSng : Single

DefStr-Anweisung

Legt den Vorgabevariablentyp fest.

Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

SYNTAX:
Defxxx Characterrange1[, Characterrange2[,...]]

PARAMETER:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

SCHLÜSSELWORT : VOREINGESTELLTER VARIABLENTYP

DefBool : Boolean

BEISPIEL:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefBool
bOK=TRUE    REM bOK ist implizit eine Boolesche Variable
end sub
```

DefStr : String

DefVar-Anweisung

Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.

SYNTAX:
Defxxx Characterrange1[, Characterrange2[,...]]

PARAMETER:
Characterrange : Zeichen, die den Buchstabenbereich festlegen, für die Variablennamen, die voreingestellt für den angegebenen Variablentypen deklariert werden sollen.

xxx : Schlüsselwort, das den voreinzustellenden Variablentypen definiert. Es gilt:

SCHLÜSSELWORT : VOREINGESTELLTER VARIABLENTYP

DefVar : Variant

BEISPIEL:

```
REM Prefix-Definitionen für die verschiedenen Typen von Varia-
blen
DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v
Sub ExampleDefVar
vDiv=99    REM vDiv ist implizit ein Variant
vDiv="Hallo Welt"
end sub
```

Dim-Anweisung

Deklariert eine Variable oder ein Array.

Mit DIM deklarieren Sie lokale Variablen innerhalb von Subs. Globale Variablen werden außerhalb von Subs mit den Anweisungen PUBLIC oder PRIVATE deklariert.

SYNTAX:
[ReDim]Dim VarName [(start To end)] [As VarTyp][, VarName2 [(start To end)] [As VarTyp][,...]]

PARAMETER:
VarName : Beliebiger Variablen- oder Arrayname.

start, end : Numerische Werte oder Konstanten, die numerische Werte beinhalten zwischen -32768 und 32767, die die Anzahl der Elemente (AnzahlElemente=(end-start)+1) und den Indexbereich definieren.

start und end können numerische Ausdrücke sein, wenn ReDim auf Prozeduren-Ebene verwendet wurde.

VarTyp : Variablendeklarationsschlüsselwort. Es gelten:

Schlüsselwort : Variablentyp

Bool : Boolean-Variable (True, False)

Date : Date-Variable

Double : Double-Fließkommavariable (1,79769313486232 x 10E308 - 4,94065645841247 x 10E-324)

Integer : Integer-Variable (-32768 - 32767)

Long : Long-Integervariable (-2.147.483.648 - 2.147.483.647)

Object : Object-Variable (Achtung: kann später nur mit Set definiert werden!)

[Single] : Single-Fließkommavariable (3,402823 x 10E38 - 1,401298 x 10E-45). Wird kein Schlüsselwort angegeben, wird eine Variable automatisch als Single-Typ definiert, es sei denn, eine der Anweisungen DefBool bis DefVar ist zuvor verwendet worden.

String : Zeichenkettenvariable. Nimmt beliebige Zeichen gemäß des ASCII auf. Zeichenketten können maximal 64.000 Zeichen umfassen.
Variant : Variant-Variablentyp (kann alle Typen beinhalten und wird bei der und durch die Definition typisiert).

Variablen brauchen in StarBasic nicht explizit deklariert zu werden, wie Sie es vielleicht von anderen Programmiersprachen her kennen. Lediglich Arrays müssen vor Verwendung deklariert werden. Eine Variablendeklaration erfolgt mit dem Dim-Befehl; mehrere Deklarationen können mit Komma getrennt erfolgen. Um den Typen für eine Variable bei einer Deklaration festzulegen, verwendet man entweder ein spezielles Typendeklarationszeichen, das hinter dem Namen angegeben wird oder ein entsprechendes Schlüsselwort.

StarBasic kennt ein- oder mehrdimensionale Arrays (Felder), denen ein bestimmter angebbarer Variablentyp zugrunde liegt. Arrays werden immer dann verwendet, wenn im Programm Listen oder Tabellen verarbeitet werden sollen. Der Vorteil von Arrays ist, daß man die einzelnen Elemente über sogenannte Indizes ansprechen kann, die selbst ebenfalls in Form von numerischen Ausdrücken oder Variablen formuliert werden können.

Arrays werden mit dem Dim-Befehl deklariert. Es gibt dabei zwei Möglichkeiten, festzulegen, in welchem Bereich die Indizes angesprochen werden können:

DIM text(20) as String REM 21 Elemente, von 0 bis 20 durchnumeriert

DIM text(5 to 25) as String REM 21 Elemente, von 5 bis 25 durchnumeriert

DIM text(-15 to 5) as String REM 21 Elemente (das 0. ebenfalls!),

REM von -15 bis 5 durchnumeriert
Variablenfelder gleich welchen Typs können dynamisch gehalten werden, wenn sie auf Prozeduren-Ebene (also in Subs oder Functions) mit der ReDim-Anweisung dimensioniert werden. Normalerweise können die Grenzen eines Arrays nur ein einziges Mal definiert werden und bleiben dann unveränderbar. Innerhalb einer Prozedur kann ein Array mit REDIM deklariert werden; bei ReDim können die Grenzen hier durch numerische Ausdrücke bestimmt werden, so daß die Felder nur so groß wie tatsächlich benötigt gehalten werden.

BEISPIEL:

```
Sub ExampleDim
Dim sVar As String
Dim iVar As Integer
sVar = "Star Division"
End Sub
```

ReDim-Anweisung

Deklariert eine Variable oder ein Array.

SYNTAX:
[ReDim]Dim VarName [(start To end)] [As VarTyp][, VarName2 [(start To end)] [As VarTyp][,...]]

PARAMETER:
VarName : Beliebiger Variablen- oder Arrayname.

start, end : Numerische Werte oder Konstanten, die numerische Werte beinhalten zwischen -32768 und 32767, die die Anzahl der Elemente (AnzahlElemente=(ende-start)+1) und den Indexbereich definieren.

start und end können numerische Ausdrücke sein, wenn ReDim auf Prozeduren-Ebene verwendet wurde.

VarTyp : Variablendeklarationsschlüsselwort. Es gelten:

Schlüsselwort : Variablentyp

Bool : Boolean-Variable (True, False)

Date : Date-Variable

Double : Double-Fließkommavariable (1,79769313486232 x 10E308 - 4,94065645841247 x 10E-324)

Integer : Integer-Variable (-32768 - 32767)

Long : Long-Integervariable (-2.147.483.648 - 2.147.483.647)

Object : Object-Variable (Achtung: kann später nur mit Set definiert werden!)

[Single] : Single-Fließkommavariable (3,402823 x 10E38 - 1,401298 x 10E-45). Wird kein Schlüsselwort angegeben, wird eine Variable automatisch als Single-Typ definiert, es sei denn, eine der Anweisungen DefBool bis DefVar ist zuvor verwendet worden.

String : Zeichenkettenvariable. Nimmt beliebige Zeichen gemäß des ASCII auf. Zeichenketten können maximal 64.000 Zeichen umfassen.

Variant : Variant-Variablentyp (kann alle Typen beinhalten und wird bei der und durch die Definition typisiert).

Variablen brauchen in StarBasic nicht explizit deklariert zu werden, wie Sie es vielleicht von anderen Programmiersprachen her kennen. Lediglich Arrays müssen vor Verwendung deklariert werden. Eine Variablendeklaration erfolgt mit dem Dim-Befehl; mehrere Deklarationen können mit Komma getrennt erfolgen. Um den Typen für eine Variable bei einer Deklaration festzulegen, verwendet man entweder ein spezielles Typendeklarationszeichen, das hinter dem Namen angegeben wird oder ein entsprechendes Schlüsselwort.

StarBasic kennt ein- oder mehrdimensionale Arrays (Felder), denen ein bestimmter angebbarer Variablentyp zugrunde liegt. Arrays werden immer dann verwendet, wenn im Programm Listen oder Tabellen verarbeitet werden sollen. Der Vorteil von Arrays ist, daß man die einzelnen Elemente über sogenannte Indizes ansprechen kann, die selbst ebenfalls in Form von numerischen Ausdrücken oder Variablen formuliert werden können.

Arrays werden mit dem Dim-Befehl deklariert. Es gibt dabei zwei Möglichkeiten, festzulegen, in welchem Bereich die Indizes angesprochen werden können:

DIM text(20) as string REM 21 Elemente, von 0 bis 20 durchnumeriert

DIM text(5 to 25) as string REM 21 Elemente, von 5 bis 25 durchnumeriert

DIM text$(-15 to 5) as string REM 21 Elemente (das 0. ebenfalls!),

rem von -15 bis 5 durchnumeriert

Variablenfelder gleich welchen Typs können dynamisch gehalten werden, wenn sie auf Prozeduren-Ebene (also in Subs oder Functions) mit der Re-Dim-Anweisung dimensioniert werden. Normalerweise können die Grenzen

eines Arrays nur ein einziges Mal definiert werden und bleiben dann unveränderbar. Innerhalb einer Prozedur kann ein Array mit ReDim deklariert werden; bei ReDim können die Grenzen hier durch numerische Ausdrücke bestimmt werden, so daß die Felder nur so groß wie tatsächlich benötigt gehalten werden.

Beispiel:

```
Sub ExampleRedim
Dim iVar() As Integer, iCount As Integer
ReDim iVar(5) As integer
For iCount = 1 To 5
iVar(iCount) = iCount
Next iCount
ReDim iVar(10) As integer
For iCount = 1 To 10
iVar(iCount) = iCount
Next iCount
end sub
```

IsArray-Funktion

Ermittelt, ob es sich bei der angegebenen Variablen um ein Array, also um ein Datenfeld handelt.

Syntax:
IsArray (Var)

Rückgabe:
Bool

Parameter:
Var : Beliebige Variable, die daraufhin überprüft werden soll, ob sie als Array-Variable deklariert wurde. Wurde die Variable als Array deklariert, gibt die Funktion den Wert True zurück, andernfalls den Wert False.

Beispiel:

```
Sub ExampleIsArray
Dim sDatf(10) as String
print isarray(sdatf())
end Sub
```

IsDate-Funktion

Überprüft, ob ein angegebener numerischer Ausdruck oder ein Zeichenkettenausdruck in den Variablentyp Date umwandelbar ist oder nicht.

Syntax:
IsDate (Expression)

RÜCKGABE:
Bool

PARAMETER:
Expression : Beliebiger numerischer Ausdruck oder Zeichenkettenausdruck der überprüft werden soll. Ist der Ausdruck in einen Date-Typ umwandelbar, liefert die Funktion den Wert TRUE anderenfalls den Wert FALSE zurück.

BEISPIEL:

```
Sub ExampleIsDate
Dim sDateVar as String
sDateVar = "12.12.1997"
print IsDate(sDateVar) REM Liefert True
sDateVar = "12121997"
print IsDate(sDateVar) REM Liefert False
end sub
```

IsEmpty-Funktion

Überprüft, ob eine Variable vom Typ Variant den Wert Empty (Leer) enthält.

SYNTAX:
IsEmpty (Var)

RÜCKGABE:
Bool

PARAMETER:
Var : Beliebige Variable, die überprüft werden soll. Hat die Variable den Typ Variant und den Inhalt Empty, so wird True zurückgegeben, anderenfalls False.

BEISPIEL:

```
Sub ExampleIsEmpty
Dim sVar as Variant
sVar = Empty
print IsEmpty(sVar) REM Liefert True
end sub
```

IsError-Funktion

Liefert TRUE oder FALSE zurück und zeigt an, ob der übergebene Ausdruck ein gültiger Ausdruck des Typs Error war.

SYNTAX:
IsError (Var)

PARAMETER:
Var : Variable, die überprüft werden soll. Wird TRUE zurückgeliefert, handelt es sich um einen Error-Typ.

BEISPIEL:

```
Sub f_Error
On error Goto ErrorHandler
Dim iVar as Integer
Dim sVar As String
iVar = Freefile
Open "\file.txt" for Input as #iVar
Line Input #iVar, sVar
Close #iVar
Exit Sub
Errorhandler:
Msgbox IsError(Err)
end sub
```

IsNull-Funktion

Überprüft, ob eine Variable vom Typ Variant den besonderen Wert Null ("Nichts") enthält. Der Wert Null in einer Variantvariable ist dann vorhanden, wenn diese nicht initialisiert ist.

SYNTAX:
IsNull (Var)

RÜCKGABE:
Bool

PARAMETER:
Var : Beliebige Variable, die überprüft werden soll. Liefert die Funktion True zurück, handelt es sich bei der Variablen um eine Variant Variable mit dem Inhalt Null.

BEISPIEL:

```
Sub ExampleIsNull
Dim vVar As Variant
msgbox IsNull(vVar)
end sub
```

IsNumeric-Funktion

Ermittelt, ob der angegebene Ausdruck eine Zahl ist. Wird der Ausdruck als Zahl erkannt, wird TRUE zurückgegeben, andernfalls FALSE.

SYNTAX:
IsNumeric (Var)

RÜCKGABE:
Bool

PARAMETER:
Var: Beliebiger Ausdruck, der überprüft werden soll.

BEISPIEL:

```
Sub ExampleIsNumeric
Dim vVar as variant
vVar = "ABC"
print IsNumeric(sVar) REM Liefert False
vVar = "123"
print IsNumeric(sVar) REM Liefert True
end sub
```

IsObject-Funktion

Stellt fest, ob es sich bei der angegebenen Objektvariablen um ein OLE-Objekt handelt. Ist der angegebene Ausdruck eine Variable vom Typ OBJECT, wird TRUE zurückgegeben, andernfalls FALSE.

SYNTAX:
IsObject (ObjectVar)

RÜCKGABE:
Bool

PARAMETER:
ObjectVar : Beliebige Variable, die überprüft werden soll. Beinhaltet die Objektvariable ein OLE-Objekt, lautet der Rückgabewert True.

BEISPIEL:

```
Sub ExampleIsObject
Dim vWindow as Variant
Msgbox IsObject(aWindow) REM False, da ist ja nichts
aWindow = ActiveWindow
Msgbox IsObject(aWindow) REM True
end sub
```

LBound-Funktion

Ermittelt die Untergrenze eines Arrays.

SYNTAX:
LBound (ArrayName [, Dimension])

RÜCKGABE:
Integer

PARAMETER:
ArrayName : Name des Arrays, dessen Ober- ('Ubound') bzw. Untergrenze ('LBound') ermittelt werden soll.

[Dimension] : Integer Zahl die angibt, für welche Dimension die Ober- ('Ubound') bzw. Untergrenze ('LBound') ermittelt werden soll. Wird kein Wert angegeben, wird der Wert 1 angenommen.

BEISPIEL:

```
Sub ExampleUboundLbound
Dim sVar(10 to 20) As String
print LBound(sVar())
print UBound(sVar())
end Sub
Sub ExampleUboundLbound2
Dim sVar(10 to 20,5 To 70) As String
print LBound(sVar())   REM  liefert 10 zurück
print UBound(sVar())   REM  liefert 20 zurück
print LBound(sVar(),2) REM  liefert 5 zurück
print UBound(sVar(),2) REM  liefert 70 zurück
end Sub
```

UBound-Funktion

Ermittelt die Obergrenze eines Arrays.

SYNTAX:
UBound (ArrayName [, Dimension])

RÜCKGABE:
Integer

PARAMETER:
ArrayName : Name des Arrays, dessen Ober- ('Ubound') bzw. Untergrenze ('LBound') ermittelt werden soll.

[Dimension] : Integer Zahl die angibt, für welche Dimension die Ober- ('Ubound') bzw. Untergrenze ('LBound') ermittelt werden soll. Wird kein Wert angegeben, wird der Wert 1 angenommen.

BEISPIEL:

```
Sub ExampleUboundLbound
Dim sVar(10 to 20) As String
print LBound(sVar())
print UBound(sVar())
end Sub
Sub ExampleUboundLbound2
Dim sVar(10 to 20,5 To 70) As String
print LBound(sVar())   REM  liefert 10 zurück
print UBound(sVar())   REM  liefert 20 zurück
```

```
print LBound(sVar(),2) REM  liefert 5 zurück
print UBound(sVar(),2) REM  liefert 70 zurück
end Sub
```

Let-Anweisung

Weist einer Variablen einen Wert zu.

SYNTAX:
[Let] VarName=Expression

PARAMETER:
VarName : Variable beliebigen Typs, der ein Ausdruck zugewiesen wird. Bitte beachten Sie, daß Ausdruck und Variablentyp miteinander kompatibel sein müssen.

Wie in nahezu allen Basic-Dialekten ist die Angabe des Schlüsselwortes Let optional.

BEISPIEL:

```
Sub ExampleLen
Dim stext as String
Let sText = "StarDivision"
msgbox Len(sText) REM liefert 12
End Sub
```

Option Base-Anweisung

Definiert die voreingestellte Untergrenze für Arrays als 0 bzw. 1.

SYNTAX:
Option Base { 0 | 1}

PARAMETER:

Diese Anweisung muß vor dem ersten ausführbaren Programm-Code innerhalb eines Moduls stehen.

BEISPIEL:

```
option Base 1
Sub ExampleOptionBase
Dim sVar(20) As String
msgbox LBound(sVar())
end sub
```

Option Explicit-Anweisung

Bestimmt, daß jede im Programm-Code verwendete Variable explizit mit der Dim-Anweisung deklariert werden muß.

SYNTAX:
Option Explicit

PARAMETER:

Diese Anweisung muß vor dem ersten ausführbaren Programm-Code innerhalb eines Moduls stehen.

BEISPIEL:

```
Option Explicit
Sub ExampleExplicit
Dim sVar As String
sVar = "Star Division"
For i% = 1 to 10  REM Hier gibts einen Laufzeitfehler
REM
Next i%
End Sub
```

Public-Anweisung

Dimensioniert eine Variable oder ein Array auf Modulebene (also außerhalb einer Sub oder einer Function), damit die Variable und das Array in allen Bibliotheken und Modulen Gültigkeit hat.

SYNTAX:
Public VarName[(start To end)] [As VarTyp][, VarName2[(start To end)] [As VarTyp][,...]]

BEISPIEL:

```
Public iVar As Integer
Sub ExamplePublic
iVar = iVar + 1
MsgBox iVar
End sub
```

Static-Anweisung

Dimensioniert eine Variable oder ein Array auf Prozedurebene (also innerhalb einer Sub oder einer Function), damit die Werte der Variablen oder des Arrays beim Verlassen der Sub oder der Function erhalten bleiben. Es gelten ansonsten die Konventionen der Dim-Anweisung.

Bei der STATIC-ANWEISUNG können keine variablen Arrays definiert werden. Arrays müssen mit einer festen Größe angegeben werden.

SYNTAX:
Static VarName[(start To end)] [As VarTyp], VarName2[(start To end)] [As VarTyp], ...

BEISPIEL:

```
Sub ExampleStatic
Static iInit As Integer
if iInit = 0 then iInit = InitVar()                    REM
kontrolliere ob Variable schon initialisiert wurde
iInit = iInit + 1
MsgBox iInit,0,"Die Antwort lautet"
End Sub
REM Funktione zur initialisierung der statischen Variablen
Function InitVar() As Integer
   InitVar = 40  REM ein Wert zur Initialisierung
End Function
```

TypeName-Funktion; VarType-Funktion

Liefert eine Zeichenkette (TypeName) oder einen numerischen Wert (VarType) zurück, der oder die Informationen über eine Variable gibt.

SYNTAX:
TypeName (Variable)VarType (Variable)

RÜCKGABE:
String; Integer

PARAMETER:
Variable: Variable, deren Typ ermittelt werden soll. Es gelten folgende Rückgabeergebnisse:

Schlüsselwort	VarType	Variablentyp
Boolean	11	Boolean-Variable
Date	7	Date-Variable
Double	5	Double-Fließkommavariable
Integer	2	Integer-Variable
Long	3	Long-Integer-Variable
Object	9	Objekt-Variable
Single	4	Single-Fließkommavariable

Schlüsselwort	VarType	Variablentyp
String	8	Zeichenketten-Variable.
Variant	12	Variant-Variable (kann alle Typen beinhalten und wird bei der und durch die Definition typisiert)
Empty	0	Variable ist nicht initialisiert
Null	1	Keine gültigen Daten

BEISPIEL:

```
Sub ExampleType
Dim iVar As Integer
Dim sVar As String
Dim siVar As Single
Dim dVar As Double
Dim bVar As Boolean
Dim lVar As Long
Msgbox TypeName(iVar) & " " & VarType(iVar) & Chr(13) &_
TypeName(sVar) & " " & VarType(dVar) & Chr(13) &_
TypeName(siVar) & " " & VarType(siVar) & Chr(13) &_
TypeName(dVar) & " " & VarType(dVar) & Chr(13) &_
TypeName(bVar) & " " & VarType(bVar) & Chr(13) &_
TypeName(lVar) & " " & VarType(lVar),0,"Einige Typen in Star-
Basic"
end Sub
```

Vergleichsoperatoren

Hier werden alle Vergleichsoperatoren erklärt.

=, <, >, <=, >=, <> - Operatoren

Mit den Vergleichsoperatoren können Sie in Ihrem Programm oder Macro je zwei Ausdrücke miteinander vergleichen. Sie erhalten als Ergebnis des Vergleichs einen Wahrheitswert (Booleschen Ausdruck) zurück, der Ihnen angibt, ob der Vergleich wahr (True, entsprechend -1) oder falsch (False, entsprechend 0) ergibt.

SYNTAX:
Result = Expression1 { = | < | > | <= | >= } Expression2

PARAMETER:
Result : Boolescher Ausdruck, der das Ergebnis des Vergleichs aufnimmt. Das Ergebnis kann nur den Wert True (-1) für einen Vergleich, der zutrifft, oder den Wert False (0), für einen Vergleich der nicht zutrifft, annehmen.

Expression1, Expression2 : Beliebige numerische Werte oder Zeichenketten, die miteinander verglichen werden sollen.

VERGLEICHSOPERATOREN

= : gleich

< : kleiner als

> : größer als

<= : kleiner gleich

>= : größer gleich

<> : ungleich

BEISPIEL:

```
Sub ExampleUnequal
Dim sDatei as String
Dim sRoot as String REM REM  Verzeichnis ab dem alle Dateien
und Verzeichnisse ausgegeben werden
sRoot = "c:\"
sDatei = Dir$( sRoot ,22)
If sDatei <> "" Then
Do
Msgbox sDatei
sDatei = Dir$
Loop Until sDatei = ""
End If
End sub
```

Zeichenfolgen

Hier finden Sie alle Funktionen und Anweisungen, die Zeichenfolgen untersuchen und ermitteln.

Mit Hilfe von Zeichenfolgen können Texte innerhalb eines StarBasic-Programms verarbeitet werden. Die hier beschriebenen Funktionen helfen, Zeichenfolgen zu untersuchen oder auf unterschiedlichste Art und Weise zu manipulieren.

ASCII/ANSI in Zeichenfolgen

Hier finden Sie alle Funktionen, die Zeichenfolgen in ASCII/ANSI umwandeln und umgekehrt.

Asc-Funktion

Gibt den ASCII-Wert (American Standard Code for Information Interchange) eines Zeichenfolgeausdrucks zurück.

SYNTAX:
Asc (Text As String)

RÜCKGABE:
Integer

PARAMETER:
Text: Beliebiger Zeichenfolgeausdruck. Relevant dafür ist jedoch nur das erste Zeichen, der Rest der Zeichenfolge wird unberücksichtigt gelassen.
Die Asc-Funktion wird häufig verwendet, um Eingaben von der Tastatur abzufragen. Sie bildet quasi die Umkehrfunktion von der CHR$-Funktion. Wird der Asc-Funktion ein Leerstring (Zeichenfolge ohne Inhalt) übergeben, meldet StarBasic einen Fehler. Neben den reinen druckbaren ASCII-Zeichen (Codes 32-255), kann die ASCII-Funktion ebenfalls dazu herhalten, bestimmte Tastencodes, die ebenfalls im ASCII kodiert sind, abzufangen.

BEISPIEL:

```
Sub ExampleASC
Print ASC("A") REM liefert 65
Print ASC("Z") REM liefert 90
Print ASC("StarDivision") REM liefert 83, weil nur das erste
Zeichen berücksichtigt wird
End Sub
```

Chr-Funktion

Liefert das Zeichen, das dem angegebenen ASCII-Wert entspricht.

SYNTAX:
Chr(Expression As Integer)

RÜCKGABE:
String

PARAMETER:
Expression : Numerische Variable, die gültigen ASCII-Wert aufnimmt (0-255).

Die CHR$-Funktion wird oft verwendet, um spezielle Steuersequenzen an den Drucker oder ein anderes Ausgabegerät zu senden, aber auch, um beispielsweise Anführungszeichen in einen Zeichenfolgeausdruck einzufügen, was in der Programmierumgebung direkt nur sehr schwer zu realisieren ist,

da das Anführungszeichen selbst den Start einer Zeichenfolge im StarBasic-Quelltext kennzeichnet.

BEISPIEL:

```
sub ExampleChr
REM Dieses Beispiel fügt ein Anführungszeichen (ASCII-Wert 34)
in eine Zeichenfolge ein.
MsgBox "Der "+ Chr(34)+"kleine" + Chr(34)+" Riese."
REM Bei der Ausgabe erscheint im Dialogfeld: Der "kleine"
Riese.
end sub
```

Str-Funktion

Wandelt einen numerischen Ausdruck in eine Zeichenfolge um.

SYNTAX:
Str (Expression)

RÜCKGABE:
String

PARAMETER:
Expression : Ein beliebiger numerischer Ausdruck.

Mit der STR-Funktion ist es möglich, Rechenergebnisse oder den Inhalt numerischer Variablen in Zeichenfolgen umzuwandeln und bei Bedarf einzubinden. Negativen numerischen Ausdrücken wird dabei ein Minuszeichen vorangestellt. Beachten Sie aber bitte, daß die STR-Funktion bei einem positiven numerischen Ausdruck ein Leerzeichen (anstelle des Plus-Zeichens) vor die zurückgegebene Zeichenfolge setzt.

BEISPIEL:

```
Sub ExampleStr
Dim iVar As Single
Dim sVar As String
iVar = 123.123
sVar = LTrim(Str(iVar))
Msgbox sVar & chr(13)  & Str(iVar)
end sub
```

Val-Funktion

Wandelt eine Zeichenfolge in einen numerischen Ausdruck um.

SYNTAX:
Val (Text As String)

RÜCKGABE:
Double

PARAMETER:
Text : Eine Zeichenfolge, die eine Zahl darstellt.

Mit der Val-Funktion ist es möglich, Zeichenfolgen, die Zahlen darstellen, in numerische Ausdrücke umzuwandeln. Sie bildet damit das Gegenstück zur STR-Funktion. Besteht die Zeichenfolge nur teilweise aus in numerischen Ausdrücken umwandelbaren Zeichen, werden nur die ersten möglichen Zeichen der Zeichenfolge umgewandelt. Handelt es sich bei der Zeichenfolge um einen gar nicht übersetzbaren Ausdruck, liefert die VAL-Funktion das Ergebnis 0 zurück.

BEISPIEL:

```
Sub ExampleVal
msgbox Val("123.123")
msgbox Val("A123.123")
end Sub
```

Inhalte vervielfachen

Hier finden Sie alle Funktionen, die Zeichenfolgeninhalte vervielfachen.

Space-Funktion

Ergibt eine Zeichenfolge, bestehend aus Leerzeichen.

SYNTAX:
Space (n As Integer)

RÜCKGABE:
String

PARAMETER:
n : Numerischer Ausdruck, der die Anzahl der Leerzeichen in der Zeichenfolge festlegt.

BEISPIEL:

```
Sub ExampleSpace
Dim sText As String,sOut As String
DIm iLen As Integer
iLen = 10
sText = "StarOffice"
sOut  = sText & Space(iLen) &  sText & Chr(13) &_
sText & Space(iLen*2) &  sText & Chr(13) &_
sText & Space(iLen*4) &  sText & Chr(13)
msgBox sOut,0,"StarOffice"
End Sub
```

String-Funktion

Erzeugt eine Zeichenfolge des angegebenen Zeichens oder des ersten Zeichens einer Zeichenkette, die der Funktion übergeben wird.

SYNTAX:
String (n As Integer, {expression As Integer | character As String})

RÜCKGABE:
String

PARAMETER:
n : Integer-Ausdruck, der die Anzahl der Zeichen angibt, aus der die zu erzeugende Zeichenkette bestehen soll.

expression : Numerischer Ausdruck, der den ASCII-Code des Zeichens bezeichnet, aus dem die Zeichenfolge bestehen soll.

character : Ein Zeichen, aus dem die Zeichenfolge bestehen soll, oder eine Zeichenkette. Wird eine Zeichenkette übergeben, wird lediglich das erste Zeichen zur Bildung der Zeichenfolge verwendet.

BEISPIEL:

```
Sub ExampleSpace
Dim sText As String,sOut As String
DIm iLen As Integer
iLen = 10
sText = "StarOffice"
sOut  = sText & Space(iLen) &  sText & Chr(13) &_
sText & Space(iLen*2) &  sText & Chr(13) &_
sText & Space(iLen*4) &  sText & Chr(13)
msgBox sOut,0,"StarOffice"
End Sub
```

Inhalte bearbeiten

Hier finden Sie alle Funktionen, die Zeichenfolgeninhalte manipulieren, formatieren und ausrichten.

Format-Funktion

Wandelt eine Zahl in eine Zeichenkette um, die einer Anweisung entsprechend formatiert wird.

Wird die Zahl als String übergeben, wird keine Typenkonvertierung durchgeführt, und das Ergebnis ist ein Leerstring.

SYNTAX:
Format (Number [, Format As String])

RÜCKGABE:
String

PARAMETER:
Number: Ein numerischer Ausdruck, der in eine Zeichenkette formatiert abgelegt werden soll.

Format: Eine Zeichenfolge, die die Anweisungen zur Formatierung der Zahl enthält. Wird "Format" nicht angegeben, verhält sich die Format-Funktion wie die Str-Funktion.

Die folgende Übersicht beschreibt die Zeichen der Zeichenfolge "Format", die Sie zur Zuweisung der Formatierung einer Zahl verwenden können:

0: Wenn sich im Ausdruck "Number" eine Ziffer an der Position befindet, an der in der Formatzeichenfolge die 0 erscheint, wird die Ziffer angezeigt; ansonsten wird eine Null ausgegeben.

Hat der Ausdruck "Number" weniger Ziffern als der Formatausdruck "Format" Nullen (auf beiden Seiten des Dezimalzeichens), so werden führende oder nachgestellte Nullen angezeigt. Hat die Zahl mehr Ziffern vor dem Dezimalzeichen als der Formatausdruck Nullen, so werden die zusätzlichen Stellen ohne Änderung angezeigt.

Nachkommastellen der zu formatierenden Zahl werden auf die Anzahl der Nullen gerundet, die im Formatausdruck "Format" hinter dem Dezimalzeichen angegeben sind.

#: Wenn "Number" eine Ziffer an der Position des #-Platzhalters in der Formatzeichenfolge "Format" enthält, wird die Ziffer angezeigt; andernfalls wird an dieser Position nichts angezeigt.

Dieses Zeichen verhält sich wie der Platzhalter 0, mit dem Unterschied, daß führende oder nachgestellte Nullen nicht angezeigt werden, wenn sich im Formatausdruck mehr #-Platzhalter vor oder nach dem Dezimalzeichen befinden, als Ziffern in der Zahl enthalten sind. Damit werden nur die für die Zahl relevanten Ziffern angezeigt.

.: Der Punkt dient als Platzhalter für das Dezimalzeichen. Er bestimmt, wieviele Stellen links und rechts vom Dezimalzeichen angezeigt werden.

Wenn der Formatausdruck nur #-Platzhalter links von diesem Symbol enthält, beginnen Zahlen, die kleiner als 1 sind, mit einem Dezimalzeichen. Sollen Dezimalzahlen immer mit einer führenden Null angezeigt werden, so

müssen Sie 0 als Platzhalter für die erste Ziffer links vom Dezimalzeichen angeben.

%: Die Zahl wird mit 100 multipliziert. Das Prozentzeichen (%) wird an der Stelle eingefügt, an der es in der Formatzeichenfolge erscheint.

E- E+ e- e+: Wenn der Formatausdruck mindestens einen Ziffern-Platzhalter (0 oder #) rechts von dem Symbol E-, E+, e- oder e+ enthält, erscheint die Zahl im wissenschaftlichen Format oder Exponentialformat. Dabei wird der Buchstabe E oder e zwischen der Zahl und dem Exponenten eingefügt, und die Anzahl der Ziffern-Platzhalter rechts neben dem Symbol bestimmt die Anzahl der Stellen im Exponenten.

Ist der Exponent negativ, so wird bei den Symbolen E-, E+, e-, e+ ein Minuszeichen unmittelbar vor dem Exponenten ausgegeben. Ist der Exponent positiv, so wird nur bei den Symbolen E+ oder e+ ein Pluszeichen ausgegeben.

,: Das Komma dient als Platzhalter für das Tausendertrennzeichen. Es trennt Tausender- von Hunderterstellen in einer mindestens vierstelligen Zahl.

Das Tausendertrennzeichen wird ausgegeben, wenn der Formatausdruck den entsprechenden Platzhalter enthält und dieser von Ziffern-Platzhaltern (0 oder #) umgeben ist.

- + $ ( ) Leerzeichen: Die Zeichen Plus (+), Minus (-), Dollar ($), Leerzeichen und Klammern können Sie zur Formatierung der Zahl in der Zeichenfolge "Format" direkt eingeben.

Wenn Sie andere als die hier angegebenen Zeichen ausgeben möchten, müssen Sie einen umgekehrten Schrägstrich (\) vor das entsprechende Zeichen setzen oder das Zeichen in Anführungsstriche (" ") setzen.

\: Der umgekehrte Schrägstrich bewirkt, daß ein darauf folgendes Zeichen in der Formatzeichenfolge angezeigt wird. Er entspricht dem Einschließen des nächsten Zeichens in Anführungsstriche.

Viele Zeichen im Formatausdruck haben eine besondere Bedeutung und können nur dann als Literalzeichen angezeigt werden, wenn ihnen ein umgekehrter Schrägstrich vorangestellt wird. Der umgekehrte Schrägstrich selbst wird nicht angezeigt. Sie können ihn jedoch durch zwei umgekehrte Schrägstriche (\\) anzeigen lassen.

Beispiele von Zeichen, die nicht als Literalzeichen angezeigt werden können, sind die Buchstaben zur Formatierung von Datums- und Zeitangaben (a, c, d, h, m, n, p, q, s, t, w, y, /, :) sowie die Zeichen zur Formatierung von Zahlen (#, 0, %, E, e, Komma, Punkt) und Zeichenfolgen (@, &, <, >, !).

Die Verwendung von Punkt und Komma als Dezimal- oder Tausendertrennzeichen ist von dem jeweiligen Gebietsschema abhängig. Das tatsächlich als Dezimalzeichen ausgegebene Zeichen hängt von Ihren Systemeinstellungen für das Zahlenformat ab. Bei allen hier genannten Beispielen gehen wir davon aus, daß Sie als Gebietsschema für Ihr System "Deutschland" gewählt haben.

Zur Formatierung einer Zahl stehen Ihnen auch vordefinierte Formate zur Verfügung, die in der folgenden Tabelle aufgelistet werden. Mit Ausnahme des Formats "General Number" liefern alle Formatierungen die eingegebene Zahl als Dezimalzahl mit zwei Nachkommastellen zurück.

Bei Verwendung vordefinierter Formate muß der jeweilige Name im Formatausdruck in Anführungszeichen stehen.

VORDEFINIERTE FORMATE

General Number: Zahlen werden so angezeigt, wie sie eingegeben wurden.

Currency: Es wird ein Dollarzeichen vor die Zahl gesetzt, und negative Zahlen werden in Klammern gesetzt.

Fixed: Es wird mindestens eine Ziffer vor dem Dezimalzeichen angezeigt.

Standard: Es wird das Tausendertrennzeichen verwendet.

Percent: Der jeweilige Wert wird mit 100 multipliziert; an das Ergebnis wird ein Prozentzeichen angehängt.

Scientific: Es wird die übliche wissenschaftliche Schreibweise verwendet (also z. B. 1,00E+03 für 1000).

Eine Formatierungsanweisung kann in drei, durch Semikola getrennte, Abschnitte unterteilt werden. Der erste Abschnitt nimmt die Anweisung zur Formatierung positiver, der zweite die zur Formatierung negativer Zahlen auf und der dritte formatiert Nullzeichenfolgen. Wird nur eine Zeichenfolge mit Formatierungsanweisungen angegeben, gilt diese für alle Zahlen.

BEISPIEL:

```
Sub ExampleFormat
MsgBox Format(6328.2, "##,##0.00") REM = 6,328.20
End Sub
```

LCase-Funktion

Wandelt alle Zeichen einer Zeichenfolge in Kleinbuchstaben um. Verändert werden dabei nur die großgeschriebenen Buchstaben des Alphabets, die sich in der Zeichenfolge befinden. Kleinbuchstaben und alle sonstigen Zeichen bleiben unverändert.

Siehe auch: UCase-Funktion

SYNTAX:
LCase (Text As String)

RÜCKGABE:
String

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck, der bearbeitet werden soll.

BEISPIEL:

```
Sub ExampleLUCase
Dim sVar As String
sVar = "StarOffice"
print LCase(sVar) REM liefert "staroffice"
print UCase(sVar) REM liefert "STAROFFICE"
end Sub
```

Left-Funktion

Kürzt einen Zeichenfolgeausdruck auf eine angegebene Anzahl von n Zeichen. Die linken n Zeichen werden zurückgegeben.

SYNTAX:
Left (Text As String, n As Integer)

RÜCKGABE:
String

PARAMETER:
Text: Beliebiger Zeichenfolgeausdruck, der bearbeitet werden soll.

n: Integer-Ausdruck, der die Anzahl der Zeichen bestimmt, auf die der Zeichenfolgeausdruck verkürzt werden soll. Wird N = 0, wird eine leere Zeichenfolge (Nullstring) zurückgegeben.

Das nachfolgende Beispiel wandelt das vom Anwender eingegebene US-amerikanische Datum im Format (MM-TT-JJJJ) in einen Zeichenfolgeausdruck um, der das Datum im deutschen Format angibt (TT.MM.JJJJ).

BEISPIEL:

```
Sub ExampleGermanDate
Dim sInput As String
Dim sGe_date As String
sInput = InputBox("Geben Sie bitte ein Datum im englischen
Format MM-DD-YYYY ein")
sGe_date = Mid(sInput, 4, 2)
sGe_date = sGe_date  & "."
sGe_date = sGe_date  & Left(sInput, 2)
```

```
sGe_date = sGe_date  & "."
sGe_date = sGe_date  & Right(sInput, 4)
MsgBox sGe_date
End Sub
```

LSet-Anweisung

Ordnet eine Zeichenkette linksbündig innerhalb einer Zeichenfolgenvariablen an oder kopiert eine Variable eines benutzerdefinierten Datentyps in eine andere Variable eines benutzerdefinierten Datentyps.

SYNTAX:
LSet Var As String = Text or LSet Var1 = Var2

PARAMETER:
Var: Beliebige Zeichenfolgevariable, in der die angegebene Zeichenkette linksbündig ausgerichtet werden soll.

Text : Zeichenkette, die linksbündig in der Zeichenfolgevariable angeordnet werden soll.

Var1 : Bezeichnung der Variable des benutzerdefinierten Datentyps, in die eine andere Variable des benutzerdefinierten Datentyps kopiert werden soll.

Var2 : Bezeichnung der Variable des benutzerdefinierten Datentyps, die in die zuvor angegebene Variable des benutzerdefinierten Datentyps kopiert werden soll.

Ist die Zeichenkette kürzer als die Zeichenfolgevariable, ordnet die LSET -Anweisung die Zeichenkette linksbündig in der Zeichenfolgevariable an. Die freien Stellen der Zeichenfolgevariable werden mit Leerzeichen aufgefüllt. Ist die Zeichenkette länger, werden nur soviele Zeichen der Zeichenkette linksbündig in der Zeichenfolgevariable angeordnet, wie diese aufnehmen kann. Mit der LSET-Anweisung ist es außerdem möglich, die Variable eines benutzerdefinierten Datentyps in eine andere des selben Datentyps zu kopieren.

BEISPIEL:

```
Sub ExampleRLSet
   Dim sVar as string
   Dim sExpr as string

   sVar = String(40,"*")
   sExpr = "SBX"
   REM  "SBX" im 40 Zeichen langen Referenz-String rechts
ausrichten
REM  die Sternchen durch Leerzeichen ersetzen
   RSet sVar = sExpr
   Print ">"; sVar; "<"
```

```
   sVar = String(5,"*")
   sExpr = "123457896"
   RSet sVar = sExpr
   Print ">"; sVar; "<"

   sVar = String(40,"*")
   sExpr = "SBX"
   REM "SBX" im 40 Zeichen langen Referenz-String links
ausrichten
   LSet sVar = sExpr
   Print ">"; sVar; "<"
   sVar = String(5,"*")
   sExpr = "123456789"
   LSet sVar = sExpr
   Print ">"; sVar; "<"
End Sub
```

LTrim-Funktion

Schneidet alle führenden Leerzeichen eines Zeichenfolgeausdrucks ab.

SYNTAX:
LTrim (Text As String)

RÜCKGABE:
String

PARAMETER:
Text : Ein beliebiger Zeichenfolgeausdruck.

Verwenden Sie diese Funktion, um Leerzeichen vor einem Zeichenfolgeausdruck zu entfernen.

BEISPIEL:

```
Sub ExampleSpaces
Dim sText2 as String,sText as String,sOut as String
sText2 = "   <*StarOffice*>   "
sOut = "REM "+sText2  +"REM "+ Chr(13)
sText = Ltrim(sText2)         REM sText = "<*StarOffice*>   "
sOut = sOut + "REM "+sText +"REM " + Chr(13)
sText = Rtrim(sText2)         REM sText = "   <*StarOffice*>"
sOut = sOut +"REM "+ sText +"REM " + Chr(13)
sText = Trim(sText2)  REM sText = "<*StarOffice*>"
sOut = sOut +"REM "+ sText  +"REM "
MsgBox sOut
end sub
```

Mid-Funktion, Mid-Anweisung

Gibt den angegebenen Abschnitt eines Zeichenfolgeausdrucks zurück (Mid-Funktion) oder ersetzt diesen durch eine andere Zeichenfolge ('Mid-Anweisung').

SYNTAX:
Mid (Text As String, Start As Integer [, Length As Integer]) or Mid (Text As String, Start As Integer , Length As Integer, Text As String)

RÜCKGABE:
String (only by Function)

PARAMETER:
Text: Beliebiger Zeichenfolgeausdruck, der bearbeitet werden soll.

Start: Integer-Ausdruck, der die Position des Zeichens innerhalb der Zeichenfolge angibt, das den Beginn des Zeichenfolgenabschnitts kennzeichnet, der zurückgegeben oder ersetzt werden soll.

Length: Integer-Ausdruck, der die Anzahl der Zeichen angibt, die zurückgegeben oder ersetzt werden sollen.

Entfällt bei der MID-FUNKTION die Angabe zu length, werden alle Zeichen ab der angegebenen Position bis zum letzten Zeichen der Zeichenfolge ermittelt.

Ist length bei der MID-ANWEISUNG kleiner als die Länge des zu ersetzenden Textes, wird dieser auf die angegebene Länge verkürzt.

Text: Die Zeichenfolge, die den Zeichenfolgeausdruck ersetzen soll (nur MID -Anweisung).

BEISPIEL:

```
Sub ExampleGermanDate
Dim sInput As String
Dim sGe_date As String
sInput = InputBox("Geben Sie bitte ein Datum im englischen
Format MM-DD-YYYY ein")
sGe_date = Mid(sInput, 4, 2)
sGe_date = sGe_date & "."
sGe_date = sGe_date & Left(sInput, 2)
sGe_date = sGe_date & "."
sGe_date = sGe_date &  Right(sInput, 4)
MsgBox sGe_date
End Sub
```

Right-Funktion

Kürzt einen Zeichenfolgeausdruck auf eine angegebene Anzahl von Zeichen. Den Ausgangspunkt hierfür bildet das erste rechte Zeichen der Folge.

Siehe auch: Left-Funktion.

SYNTAX:
Right (Text As String, n As Integer)

RÜCKGABE:
String

PARAMETER:
Text: Beliebiger Zeichenfolgeausdruck, der bearbeitet werden soll.

n: Integer-Ausdruck, der die Anzahl der Zeichen bestimmt, auf die der Zeichenfolgeausdruck verkürzt werden soll. Wird N = 0, wird eine leere Zeichenfolge (Nullstring) zurückgegeben.

Das nachfolgende Beispiel wandelt das vom Anwender eingegebene US-amerikanische Datum im Format (MM-TT-JJJJ) in einen Zeichenfolgeausdruck um, der das Datum im deutschen Format angibt (TT.MM.JJJJ).

BEISPIEL:

```
Sub ExampleGermanDate
Dim sInput As String
Dim sGe_date As String
sInput = InputBox("Geben Sie bitte ein Datum im englischen
Format MM-DD-YYYY ein")
sGe_date = Mid(sInput, 4, 2)
sGe_date = sGe_date & "."
sGe_date = sGe_date & Left(sInput, 2)
sGe_date = sGe_date & "."
sGe_date = sGe_date & Right(sInput, 4)
MsgBox sGe_date
End Sub
```

RSet-Anweisung

Ordnet eine Zeichenkette rechtsbündig innerhalb einer Zeichenfolgevariablen an oder kopiert einen benutzerdefinierten Variablentyp in einen anderen.

SYNTAX:
RSet Text As String = Text or RSet Variable1 = Variable2

PARAMETER:
Text : Beliebige Zeichenfolgevariable.

Text : Zeichenkette, die rechtsbündig in der Zeichenfolgevariable angeordnet werden soll.

Variable1 : Benutzerdefinierte Variable, in die eine andere kopiert werden soll.

Variable2 : Benutzerdefinierte Variable, die in eine andere kopiert werden soll.

Ist die Zeichenkette kürzer als die Zeichenfolgevariable, ordnet die RSET-ANWEISUNG die Zeichenkette rechtsbündig in der Zeichenfolgevariable an. Die freien Stellen der Zeichenfolgevariable werden mit Leerzeichen aufgefüllt. Ist die Zeichenkette länger, werden nur so viele Zeichen der Zeichenkette rechtsbündig in der Zeichenfolgevariable angeordnet wie diese aufnehmen kann.

Außerdem ist es mit der RSET-ANWEISUNG möglich, eine benutzerdefinierte Variable einer anderen zuzuweisen.

Das folgende Programm erläutert die RSET- und LSET-Anweisung anhand der rechts- und linksbündigen Anordnung einer Zeichenkette in eine andere Anordnung.

BEISPIEL:

```
Sub ExampleRLSet
   Dim sVar as string
   Dim sExpr as string

   sVar = String(40,"*")
   sExpr = "SBX"
   REM  "SBX" im 40 Zeichen langen Referenz-String rechts
ausrichten
REM  die Sternchen durch Leerzeichen ersetzen
   RSet sVar = sExpr
   Print ">"; sVar; "<"
   sVar = String(5,"*")
   sExpr = "123457896"
   RSet sVar = sExpr
   Print ">"; sVar; "<"

   sVar = String(40,"*")
   sExpr = "SBX"
   REM "SBX" im 40 Zeichen langen Referenz-String links
ausrichten
   LSet sVar = sExpr
   Print ">"; sVar; "<"
   sVar = String(5,"*")
   sExpr = "123456789"
   LSet sVar = sExpr
   Print ">"; sVar; "<"
End Sub
```

RTrim-Funktion

Entfernt alle Leerzeichen am Ende eines Zeichenfolgeausdrucks.

Siehe auch: LTrim-Funktion

SYNTAX:
RTrim (Text As String)

RÜCKGABE:
String

PARAMETER:
Text : Beliebiger Zeichenfolgeausdruck.

BEISPIEL:

```
Sub ExampleSpaces
Dim sText2 as String,sText as String,sOut as String
sText2 = "   <*StarOffice*>   "
sOut = "REM "+sText2  +"REM "+ Chr(13)
sText = Ltrim(sText2)         REM sText = "<*StarOffice*>   "
sOut = sOut + "REM "+sText +"REM " + Chr(13)
sText = Rtrim(sText2)         REM sText = "   <*StarOffice*>"
sOut = sOut +"REM "+ sText +"REM " + Chr(13)
sText = Trim(sText2)  REM sText = "<*StarOffice*>"
sOut = sOut +"REM "+ sText  +"REM "
MsgBox sOut
end sub
```

Trim-Funktion

Schneidet alle führenden und nachfolgenden Leerzeichen eines Zeichenfolgeausdrucks ab.

SYNTAX:
Trim(Text As String)

RÜCKGABE:
String

PARAMETER:
Text : Ein beliebiger Zeichenfolgeausdruck.

Verwenden Sie diese Funktion, um führenden und nachfolgenden Leerzeichen aus einem Zeichenfolgeausdruck zu entfernen.

BEISPIEL:

```
Sub ExampleSpaces
Dim sText2 as String,sText as String,sOut as String
sText2 = "   <*StarOffice*>   "
sOut = "REM "+sText2  +"REM "+ Chr(13)
```

```
sText = Ltrim(sText2)         REM sText = "<*StarOffice*>   "
sOut = sOut + "REM "+sText +"REM " + Chr(13)
sText = Rtrim(sText2)         REM sText = "   <*StarOffice*>"
sOut = sOut +"REM "+ sText +"REM " + Chr(13)
sText = Trim(sText2)  REM sText = "<*StarOffice*>"
sOut = sOut +"REM "+ sText  +"REM "
MsgBox sOut
end sub
```

UCase-Funktion

Wandelt alle Zeichen einer Zeichenfolge in Großbuchstaben um. Umgewandelt werden dabei nur die kleingeschriebenen Buchstaben des Alphabets, die sich in der Zeichenfolge befinden. Großbuchstaben und alle sonstigen Zeichen bleiben unverändert.

Siehe auch: LCase-Funktion

SYNTAX:
UCase (Text As String)

RÜCKGABE:
String

PARAMETER:
Text : Beliebiger Zeichenfolgenausdruck, der bearbeitet werden soll.

BEISPIEL:

```
Sub ExampleLUCase
Dim sVar As String
sVar = "StarOffice"
print LCase(sVar) REM liefert "staroffice"
print UCase(sVar) REM liefert "STAROFFICE"
end Sub
```

Zeichenfolgelängen bearbeiten

Hier finden Sie alle Funktionen, die Zeichenfolgenlängen ermitteln und Zeichenfolgen vergleichen.

InStr-Funktion

Gibt die Position einer in einem Zeichenfolgeausdruck vorhandenen Zeichenfolge zurück.

SYNTAX:
InStr ([Start As Integer,] Text1 As String, Text2 As String[, Compare])

RÜCKGABE:
Integer

PARAMETER:
Start : Beliebiger numerischer Ausdruck, der festlegt, ab welcher Position innerhalb eines Zeichenfolgeausdrucks die Suche nach einer bestimmten Zeichenfolge einsetzen soll. Wird diese Angabe ausgelassen, beginnt die Suche beim ersten Zeichen.

Text1 : Zu durchsuchender Zeichenfolgeausdruck.

Text2 : Zu suchender Zeichenfolgeausdruck.

Compare : Optionaler, numerischer Ausdruck, der die Art des Zeichenfolgevergleichs bestimmt. Dem Parameter kann der Wert 0 oder 1 zugewiesen werden. Die Voreinstellung 0 führt zu einem binären Vergleich, der Wert 1 zu einem Textvergleich ohne Berücksichtigung der Groß-/Kleinschreibung.

Die INSTR-Funktion gibt die Position des Zeichens zurück, ab dem die gesuchte Zeichenfolge in der zu durchsuchenden Zeichenfolge enthalten ist. Ist die gesuchte Zeichenfolge dort nicht enthalten, wird der Wert 0 zurückgegeben.

BEISPIEL:

```
Sub ExamplePosition
Dim sInput As String
Dim iPos as Integer
sInput = "Star Division"
iPos = Instr(sInput,"v")
print iPos
end sub
```

Len-Funktion

Liefert die Anzahl der Zeichen in einer Zeichenfolge oder, wenn Variablen angegeben werden, die nicht vom Typ String (Zeichenfolge) sind, die Anzahl der Bytes, die benötigt werden, um den Inhalt der Variablen anzuzeigen.

SYNTAX:
Len (Text As String)

RÜCKGABE:
Long

PARAMETER:
Text : Ein beliebiger Zeichenfolgeausdruck oder eine Variable eines anderen Typs.

BEISPIEL:

```
Sub ExampleLen
Dim sText as String
sText = "StarDivision"
msgbox Len(sText) REM liefert 12
End Sub
```

StrComp-Funktion

Vergleicht zwei Zeichenfolgeausdrücke miteinander und liefert als Ergebnis zurück, ob sie gleich waren, ob ein Leerstring dabei war oder ob der eine Ausdruck länger bzw. kürzer war als der andere.

SYNTAX:
StrComp (Text1 As String, Text2 As String[, Compare])

RÜCKGABE:
Integer

PARAMETER:
Text1 : Beliebiger Zeichenfolgeausdruck, der mit dem im folgendenen übergebenen verglichen werden soll.

Text2 : Beliebiger Zeichenfolgeausdruck, der mit dem zuvor übergebenen verglichen werden soll.

Compare : Bestimmt die Vergleichsmethode. Dieses Argument ist optional. Wenn Compare 0 ist, wird die Groß-/Kleinschreibung beim Vergleich berücksichtigt. Ist Compare 1, wird sie hingegen nicht berücksichtigt (d. h., daß beispielsweise ein "K" genauso wie ein "k" behandelt wird).

Mit Hilfe dieser Funktion vergleichen Sie zwei Zeichenfolgen miteinander. Verglichen werden dabei die Wertigkeiten der Buchstaben nach den Telefonbuch-Sortierregeln bzw. nach den durch das Argument Compare bestimmten Konventionen.

Die Funktion liefert Ganzzahlenwerte zwischen -1 und 1 sowie "Null" zurück, um das Ergebnis des Vergleichs anzuzeigen:

RÜCKGABEWERT

-1 : Die erste Zeichenfolge ist kleiner als die zweite (z. B. "Maier" und "Müller").

0 : Die erste Zeichenfolge entspricht der zweiten (z. B. "Maier" und "Maier").

1 : Die erste Zeichenfolge ist größer als die zweite (z. B. "Müller" und "Maier").

Null : Einer der beiden Zeichenfolge war ein Leerstring ("", Null-String), z. B. "Peters" und "".

BEISPIEL:

```
Sub ExampleStrComp
Dim iVar As Single
Dim sVar As String
iVar = 123.123
sVar = Str$(iVar)
Msgbox strcomp(sVar , Str$(iVar),1)
end sub
```

Sonstige Befehle

Hier finden Sie alle Anweisungen und Funktionen, die sich nicht in die anderen aufgeführten Kategorien einordnen lassen.

Beep-Anweisung

Gibt einen Ton über die Computer-Lautsprecher aus. Der ausgegebene Ton ist systemabhängig und in der Regel in Lautstärke und Tonhöhe nicht beeinflußbar.

SYNTAX:
Beep

BEISPIEL:

```
Sub ExampleBeep
beep
beep
beep
end sub
```

GetGUIType-Funktion

Gibt einen numerischen Wert zurück, der angibt, welches grafische Benutzer-Interface gegenwärtig verwendet wird.

SYNTAX:
GetGUIType()

RÜCKGABE:
Integer

PARAMETER:
Rückgabewerte:

1 : Windows

2 : Presentation Manager

3 : Macintosh

4 : Motif

5 : Open Window

Beispiel:

```
Sub ExampleEnvironment
MsgBox GetGUIType
MsgBox GetGUIVersion
MsgBox GetSystemType
end sub
```

GetGUIVersion-Funktion

Gibt einen numerischen Wert zurück, der die Versionsnummer des gegenwärtig verwendeten grafischen Benutzer-Interface angibt.

Syntax:
GetGUIVersion()

Rückgabe:
Long

Parameter:
Rückgabewerte

311 : Windows 3.11

400 : Windows 95

Beispiel:

```
Sub ExampleEnvironment
MsgBox GetGUIType
MsgBox GetGUIVersion
MsgBox GetSystemType
end sub
```

GetSystemType-Funktion

Gibt einen numerischen Wert zurück, der das System beschreibt, auf dem die Applikation ausgeführt wird.

Syntax:
GetSystemType()

Rückgabe:
Integer

PARAMETER:
Rückgabewerte

1 : Personal Computer

2 : Macintosh

3 : Sun

4 : RS6000

5 : Alpha

6 : Alpha Personal Computer

7 : HP9000

8 : AVION

9 : VAX

BEISPIEL:

```
Sub ExampleEnvironment
MsgBox GetGUIType
MsgBox GetGUIVersion
MsgBox GetSystemType
end sub
```

Shell-Funktion

Startet ein anderes Anwendungsprogramm und bestimmt bei Bedarf dessen Darstellungsform.

SYNTAX:
Shell (Pathname As String[, Windowstyle As Integer][, Param As String][, bSync])

PARAMETER:
Pathname : Kompletter Pfad und Programmname des zu startenden Programms.

Windowstyle : Optionaler beliebiger Integer-Ausdruck, der die Darstellungsart des Programms bestimmt. Es gilt:

Beschreibung

0 - Das Programmfenster ist versteckt, und der Fokus liegt auf dem versteckten Fenster.

1 - Das Programmfenster wird in der ursprünglichen Größe angezeigt und ist fokussiert.

2 - Das Programmfenster wird als Symbol angezeigt und ist fokussiert.

3 - Das Programmfenster wird in Maximalgröße angezeigt und ist fokussiert.

4 - Das Programmfenster wird in der ursprünglichen Größe angezeigt, ist aber nicht fokussiert.

6 - Das Programmfenster wird als Symbol angezeigt; das aktuelle Fenster behält den Fokus.

10 - Gesamtbildschirm öffnen.

Param : Beliebiger Zeichenkettenausdruck, der die Übergabe der Kommandozeile erlaubt.

bSync : Boolscher Ausdruck, der bestimmt, ob zu der Applikation direkt nach dem Start eine DDE-Verbindung aufgebaut werden kann (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleShell
Shell( Application.PathSettings.Modules +"\" + "Shelp.exe")
end sub
```

Wait-Anweisung

Unterbricht die Programmausführung für die Dauer der angegebenen Anzahl an Millisekunden.

SYNTAX:
Wait millisek

PARAMETER:
millisek : Numerischer Ausdruck, der die Zeitdauer in Millisekunden aufnimmt, für die die Programmausführung unterbrochen werden soll.

BEISPIEL:

```
Sub ExampleWait
Dim lTick As Long
lTick = GetSystemTicks()
wait 2000
lTick =  (GetSystemTicks() - lTick)
MsgBox "" &  lTick & " Ticks" ,0,"Die Pause dauerte"
End Sub
```

Eigenschaften für Kontrollfelder

Mit dieser Funktion können Sie die Kontrollfeld-Eigenschaften sehen und bearbeiten. Das Eigenschaftenfenster zeigt jeweils die Eigenschaften des aktuell selektierten Kontrollfeldes. Wenn Sie in StarBasic einen Dialog erstellen, in dem verschiedene Kontrollfelder angeordnet werden, so können Sie auch dem Dialog selbst spezifische Eigenschaften zuweisen oder sie einsehen, indem Sie einfach den Dialog markieren.

So erreichen Sie diese Funktion

Menü FORMAT - KONTROLLFELD-EIGENSCHAFTEN...

Symbol in der Abreißleiste KONTROLLFELDER der Werkzeugleiste, Kontrollfeldleiste:

Eigenschaften

Je nach gewähltem Kontrollfeld stehen unterschiedliche Eigenschaften zur Wahl, außerdem unterscheiden sich die auswählbaren Eigenschaften abhängig davon, ob Sie das Kontrollfeld in ein StarOffice-Dokument oder in einen StarBasic-Dialog einfügen. So gibt es z. B. für Befehlsschaltflächen (CommandButtons) die Eigenschaften FORM-NAME und IMAGE-URL nur dann, wenn Sie den CommandButton direkt in ein Dokument einfügen. Diese Eigenschaften dienen zur Gestaltung von Web-Seiten und stehen daher nicht für die Erstellung von StarBasic-Dialogen zur Verfügung. Dafür können Sie dort z. B. die Eigenschaften TABINDEX oder MOUSEPOINTER vergeben, die bei Dokumenten nicht zur Verfügung stehen.

Je nach der zu bearbeitenden Eigenschaft können Sie im Feld rechts vom Namen der Eigenschaft entweder Werte direkt eingeben und bearbeiten, aus einem Listenfeld auswählen oder, durch Anklicken einer Schaltfläche mit drei Punkten, einen Dialog zur Steuerung der Eigenschaften aufrufen.

Die Kontrollfeld-Eigenschaften können Sie zur Entwurfszeit festlegen. Das ist die Zeit, in der Sie die Anwendung in der Entwicklungsumgebung entwerfen, indem Sie Kontrollfelder hinzufügen, bearbeiten und gestalten. Dagegen wird der Zeitraum, in dem der Programmcode ausgeführt wird, als Laufzeit bezeichnet. Es gibt Kontrollfeld-Eigenschaften, die zur Laufzeit nicht mehr verändert werden können, sie lassen sich im StarBasic-Code dann meist abfragen.

AddItem-Methode

Fügt einen Eintrag einem Listenfeld oder einem Kombinationslistenfeld hinzu.

Ist die Liste sortiert, wird der Positionsparameter ignoriert und der Eintrag entsprechend der Sortierreihenfolge eingefügt. Ist keine Position angegeben, wird der Eintrag an das Ende gehaengt.

Die Numerierung der Einträge findet von 0 bis n statt, wobei n die Anzahl aller vorhandenen Einträge -1 beschreibt. Die Anzahl der vorhandenen Einträge können Sie mit der ListCount-Eigenschaft ermitteln.

Syntax:
Objekt.AddItem Text AS String [,Position AS Integer]

Parameter:
Text: Zeichenfolge die in die Listbox oder in das Kombinationsfeld aufgenommen werden soll.

Position: Eine Ganzzahl, die die Position angibt, an der der neue Eintrag eingefügt werden soll.

Für Element:

ListBox, ComboBox

Beispiel:

```
Sub ExampleAddItem
Const sItem = "Eintrag "
Dim sCtrlName As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
End Sub
```

Alignment

Bestimmt die Ausrichtung des Textes in einem Kontrollfeld. Die ALIGNMENT-Eigenschaft steht sowohl für Bezeichnungsfelder (Label) als auch für Textfelder (TextBox) zur Verfügung.

> Hat die MULTILINE-Eigenschaft eines Textfeldes den Wert False, dann kann die standardmäßige Voreinstellung durch die ALIGNMENT-Eigenschaft nicht geändert werden. In diesem Fall wird der Text stets linksbündig ausgerichtet.

Die ALIGNMENT-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Alignment [= Alignment As Integer]

RÜCKGABE:
Integer

PARAMETER:
Alignment: Integer-Ausdruck, der die Ausrichtung des Textes bestimmt. Die verschiedenen Einstellungen entsprechen folgenden Werten:

0 - Der Text wird linksbündig ausgerichtet (Voreinstellung).

1 - Der Text wird rechtsbündig ausgerichtet.

2 - Der Text wird zentriert ausgerichtet.

FÜR ELEMENT:

Label, TextBox

Beispiel:

```
Zur Laufzeit können die Einstellungen nicht verändert werden.
```

Auto3D

Bestimmt, ob ein Dialog im 3D-Design dargestellt wird. Die AUTO3D-Eigenschaft ist voreingestellt auf True gesetzt. Bitte beachten Sie, daß die BACKCOLOR- und FORECOLOR- sowie die BACKSTYLE- und BORDER-Eigenschaft keine sichtbaren Auswirkungen haben, wenn die Auto3D-Eigenschaft eingeschaltet (=True) ist.

Die AUTO3D-Eigenschaft läßt sich nur zur Entwurfszeit festlegen, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Switch As Boolean = Object.Auto3D

PARAMETER:
True: Der Dialog wird im 3D-Design dargestellt.

False: Der Dialog wird nicht im 3D-Design dargestellt.

FÜR ELEMENT:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

AutoAdjustSize

Sorgt für eine automatische Anpassung der Größe eines Dialogs an die Schriftgröße des Betriebssystems. Diese Eigenschaft ist bei der Erstellung von plattformunabhängigen Dialogen notwendig, da die unterschiedlichen Betriebssysteme verschiedene System-Zeichensätze mit unterschiedlichen Schriftgrößen verwenden. AUTOADJUSTSIZE ändert die Dialoggröße, falls dies bei einem Wechsel der Plattform nötig ist.

Die AUTOADJUSTSIZE-Eigenschaft funktioniert nur dann, wenn für die Kontrollfelder des Dialogs der System-Zeichensatz genommen wird. Mit der Festlegung des Schriftstils durch die FONT-Eigenschaft für Kontrollfelder verliert AUTOADJUSTSIZE seine Wirkung.

Die AUTOADJUSTSIZE-Eigenschaft kann nur zur Entwurfszeit festgelegt werden.

PARAMETER:
True: Der Dialog wird bei Änderungen des Betriebssytems automatisch in seiner Größe angepaßt.

False: Es erfolgt keine automatische Anpassung.

BackColor

Bestimmt die Hintergrundfarbe eines Objekts. Diese Eigenschaft steht sowohl für Kontrollfelder als auch für Dialoge zur Verfügung.

Bei Kontrollfeldern in Dialogen ist folgendes zu beachten:

Die Änderung der Hintergrundfarbe ist nur dann sichtbar, wenn der Auto3D-Eigenschaft des Dialoges der Wert False zugewiesen wurde.

Bei Bezeichnungsfeldern (Label) wird die BackColor-Eigenschaft ignoriert, wenn die BackStyle-Eigenschaft den Wert 0 (Transparent) hat.

Die BACKCOLOR-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.BackColor [= Color As Long]

PARAMETER:
Color: Long-Integer-Ausdruck, der den Farbcode für die Hintergrundfarbe angibt.

FÜR ELEMENT:

CheckBox, ComboBox, Dialog, TextBox, Label, FrameControl, ListBox, Preview, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTextboxProperties
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es werden bei der Textbox verschiedene
Eigenschaften verändert."
Const sMessage_2 = "Hintergrundfarbe : "
Const sMessage_3 = "Vordergrundfarbe : "
Const sMessage_4 = "Umrandung : "
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.BackColor = RGB(
255, 0, 0 )
Activewindow.Controls(sCtrlName).ControlObject.ForeColor = RGB(
0, 255, 0 )
Activewindow.Controls(sCtrlName).ControlObject.Border = FALSE
```

```
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & RGB( 255, 0, 0 )
Selection.InsertPara
Selection.Insert sMessage_3 & RGB( 0, 255, 0 )
Selection.InsertPara
Selection.Insert sMessage_4 & "FALSE"
Selection.InsertPara
End Sub
```

BackStyle

Legt fest, ob ein Bezeichnungsfeld (Label) eines Dialoges durchsichtig oder undurchsichtig ist.

Diese Eigenschaft hat nur dann eine Wirkung, wenn die Auto3D-Eigenschaft des Dialoges den Wert False hat.

Die BackStyle-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.BackStyle [= Number As Integer]

Parameter:
Number: Integer-Ausdruck, der den Zustand des Label-Objekts definiert. Folgende Werte sind möglich:

0: Das Label-Objekt ist durchsichtig (Transparent); die BackColor-Eigenschaft wird ignoriert. Hinter dem Label ist die Farbe des Dialogs sichtbar.

1: Das Label-Objekt ist undurchsichtig (Opaque); es wird mit der durch die BackColor-Eigenschaft festgelegten Farbe dargestellt. Der durch den Dialog gebildete Hintergrund ist nicht sichtbar.

Für Element:

Label

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Border

Bestimmt, ob ein Kontrollfeld mit einer Umrandung versehen wird oder nicht.

Die BORDER-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Border [= True | False]

PARAMETER:
True: Kontrollfeld wird umrandet.

False: Kontrollfeld wird nicht umrandet.

FÜR ELEMENT:

TextBox, Label, Preview

Beispiel:

```
Sub ExampleTextboxProperties
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es werden bei der Textbox verschiedene
Eigenschaften verändert."
Const sMessage_2 = "Hintergrundfarbe : "
Const sMessage_3 = "Vordergrundfarbe : "
Const sMessage_4 = "Umrandung : "
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.BackColor = RGB(
255, 0, 0 )
Activewindow.Controls(sCtrlName).ControlObject.ForeColor = RGB(
0, 255, 0 )
Activewindow.Controls(sCtrlName).ControlObject.Border = FALSE
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & RGB( 255, 0, 0 )
Selection.InsertPara
Selection.Insert sMessage_3 & RGB( 0, 255, 0 )
Selection.InsertPara
Selection.Insert sMessage_4 & "FALSE"
Selection.InsertPara
End Sub
```

Cancel

Bestimmt, ob eine Befehlsschaltfläche die Funktion hat, einen Dialog abzubrechen. Der Benutzer hat damit die Möglichkeit, ungewünschte Änderungen innerhalb eines Dialoges rückgängig zu machen.

Wenn Sie eine neue Befehlsschaltfläche (CommandButton) erstellen, handelt es sich standardmäßig nicht um eine Abbrechen-Schaltfläche, die CANCEL-Eigenschaft hat standardmäßig die Voreinstellung False. Weisen Sie der CANCEL-Eigenschaft den Wert True zu, hat sie die gleiche Funktion wie die (Esc)-Taste oder das Schließen-Symbol in der Titelleiste des Dialoges.

Die CANCEL-Eigenschaft läßt sich nur zur Entwurfszeit festlegen, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Switch As Boolean = Object.Cancel

PARAMETER:
True: Die Schaltfläche ist eine Abbrechen-Schaltfläche.

False: Die Schaltfläche ist keine Abbrechen-Schaltfläche.

FÜR ELEMENT:

CommandButton

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Caption-Eigenschaft

Bestimmt die Beschriftung eines Objektes. Diese Eigenschaft läßt sich sowohl Dialogen als auch sämtlichen Kontrollfeldern zuweisen, die in oder neben dem jeweiligen Feld einen Text zur Bezeichnung aufweisen.

Wenn Sie ein neues Kontrollfeld erstellen, wird standardmäßig die als NAME -Eigenschaft vordefinierte Bezeichnung zur Beschriftung des Kontrollfeldes genommen. Dabei setzt sich die Bezeichnung zusammen aus dem Kontrollfeldnamen und einer Ganzzahl zur Numerierung des Kontrollfeldes, also z. B. CommandButton1. Mit der CAPTION-Eigenschaft können Sie dem Kontrollfeld eine aussagekräftigere Bezeichnung zuweisen, so daß die Beschriftung die Funktion des Kontrollfeldes wiedergibt.

Die CAPTION-Eigenschaft dient lediglich zur Beschriftung eines Objekts an der für den Benutzer sichtbaren Oberfläche. Beachten Sie, daß zur Laufzeit ein Objekt stets über die NAME-Eigenschaft angesprochen wird.

Die Farbe der Beschriftung können Sie über die FORECOLOR-Eigenschaft ändern.

Die CAPTION-Eigenschaft leistet bei Kontrollfeldern in Dialogen neben der Beschriftung des jeweiligen Kontrollfeldes noch etwas anderes: Innerhalb des Beschriftungstextes können Sie einen sogenannten Accelerator definieren, mit dessen Hilfe der Anwender später, beim Ablauf und der Bedienung des Programms direkt per Tastenkürzel ein Element des Dialoges anwählen kann. Um ein Element mit der Tastatur anwählbar zu machen, muß sich innerhalb des Beschriftungstextes wenigstens einmal der Buchstabe befinden, mit dessen Hilfe der Anwender die Bedienung vornehmen kann. Vor dem Buchstaben muß ferner das "&"-Zeichen gesetzt werden. Anstelle dieses Zeichens erscheint im Dialog der entsprechende Buchstabe als Hinweis auf die Tastaturbedienbarkeit unterstrichen.

Einige Kontrollfelder, wie z. B. Listenfelder, besitzen keine CAPTION-Eigenschaft. In diesem Falle kann die Beschriftung durch ein zusätzliches Label-Element erfolgen. Wird dieses Label-Element in der durch die TABINDEX-Eigenschaft festgelegten Reihenfolge direkt vor dem entsprechenden Kontrollfeld positioniert, wird durch einen Accelerator automatisch das in der TabIndex-Reihenfolge am nächsten liegende und fokussierbare Dialogelement angesprungen, da das Label-Element zwar eine CAPTION -Eigenschaft hat, aber selber nicht fokussiert werden kann.

Die CAPTION-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Caption [= Text As String]

PARAMETER:
Text: Zeichenfolgenausdruck, der die Bezeichnung des Kontrollfeldes aufnimmt.

FÜR ELEMENT:

CheckBox, Dialog, TextBox, Label, FrameControl, CommandButton, OptionButton

Beispiel:

```
Sub ExamplePrintable
Const sCombobox1 = "Diese Combobox wird gedruckt"
Const sCombobox2 = "Diese Combobox wird nicht gedruckt"
Const sMessage_1 = "Es wird nur eine von beiden Commandbutton
gedruckt."
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 2, "6cm", "4cm", "6cm", "2cm" )
REM Commandbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.caption = sCom-
bobox1
Activewindow.Controls(sCtrlName).ControlObject.Printable = TRUE
Selection.Escape
Activewindow.InsertVCControl( 2, "6cm", "7cm", "6cm", "2cm" )
REM Commandbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCom-
bobox2
Activewindow.Controls(sCtrlName).ControlObject.Printable =
FALSE
Selection.Escape
ActiveWindow.PrintDefault
End Sub
```

Change

Wird ausgelöst, sobald sich der Text eines Eingabefeldes (TextBox, ComboBox, Label) oder die Position eines Schiebebalkens (HScrollBar, VSCrollBar) ändert.

Bei den Eigenschaften HScrollBar und VSCrollBarLegt wird festgelegt, in welchen Schritten sich der Wert in Drehfeldern ändert. Bei jedem Mausklick auf den Auf- oder Abwärtspfeil des Kontrollfeldes nimmt der durch die VALUE-Eigenschaft definierte Wert um den als CHANGE-Eigenschaft definierten Betrag zu oder ab.

Die CHANGE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Change [= Step As Integer]

PARAMETER:
Step: Integer-Ausdruck, der angibt, um welchen Betrag der Wert im Drehfeld bei jedem Mausklick zu- oder abnimmt.

FÜR ELEMENT:

ComboBox, TextBox, Label, HScrollBar, VScrollBar

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleSpinbutton
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sSpinbutton = "NeuerSpinbutton"
Const sMessage = "Trägt den Wert des Spinbuttons in fünfer-
schritten in die Textbox ein"
Dim sTmpControlName As String, sDocName As String, sProcedure
As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 3, "6cm", "5cm", "2cm", "2cm" )
REM Spinbutton
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sSpinbutton
ActiveWindow.Controls(sSpinbutton).ControlObject.Min = 1
ActiveWindow.Controls(sSpinbutton).ControlObject.Max = 100
ActiveWindow.Controls(sSpinbutton).ControlObject.Change = 5
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
sProcedure = "Standard.Modul1.SpinbuttonValue"
Selection.SetHandler(14, sProcedure )  REM OnSpinUp
Selection.SetHandler(15, sProcedure )  REM OnSpinDown
Selection.Escape
End Sub
Sub SpinbuttonValue
Const sTextbox = "NeueTextbox"
Const sSpinbutton = "NeuerSpinbutton"
Dim iSpinValue As Integer
iSpinValue =
Activewindow.Controls(sSpinbutton).ControlObject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iSpinValue
Selection.Escape
End Sub
```

Clear-Methode

Löscht den Inhalt der Zwischenablage oder den Inhalt eines Listenfeldes bzw.Kombinationslistenfeldes.

SYNTAX:
Object.Clear

PARAMETER:
FÜR ELEMENT:

Clipboard, ComboBox, ListBox

Beispiel:

```
Sub ExampleClear
Const sItem = "Eintrag "
Dim sCtrlName As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.Clear
End Sub
```

Cls-Methode

Löscht den Inhalt der Preview-Dialogelements.

SYNTAX:
Object.Cls

PARAMETER:
FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

CurrentStep

Legt für einen Dialog fest, welcher Index der Step-Eigenschaft gerade aktuell ist. Mit CurrentStep wird bestimmt, welche Dialogelemente momentan sichtbar sind, da alle Elemente ausgeblendet werden, bei denen der Step-Index (außer 0) nicht dem Wert von CurrentStep entspricht.

Diese Eigenschaft dient zur Erstellung von Dialog-Assistenten, die schrittweise durchlaufen werden. Ein Beispiel für solch einen Dialog-Assistenten ist der AutoPilot von StarOffice. Sämtliche Dialogelemente werden in einem "Dialog-Container" bereitgestellt, alle Elemente, die zum gleichen Zeitpunkt, also auf derselben Seite des Assistenten, eingeblendet werden sollen, erhalten durch die Step-Eigenschaft den gleichen Step-Index.

Zur Laufzeit sorgt der Programmierer dafür, daß der jeweils nächste Schritt bzw. die jeweils nächste Seite des Assistenten durch eine entsprechende Maßnahme des Anwenders (in der Regel ein Klick auf die Weiter-Schaltfläche) angezeigt wird. Dabei wird die CurrentStep-Eigenschaft des Dialoges auf den jeweils nächsten Wert erhöht. Dialogelemente, die dem aktuellen Wert von CurrentStep entsprechen, werden automatisch eingeblendet, während solche, die diesen Wert nicht besitzen, ausgeblendet werden.

Wurde die Bestimmung eines Wertes für die Step-Eigenschaft ausgelassen, wird das entsprechende Dialogelement unabhängig von der CurrentStep-Eigenschaft immer angezeigt.

Ist CurrentStep = 0, werden alle Elemente unabhängig von ihrer Step-Eigenschaft angezeigt.

Die CurrentStep-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.CurrentStep [= Number As Integer]

Parameter:
Number: Integer-Ausdruck, der den aktuellen Index der Step-Eigenschaft definiert.

Für Element:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

CurrentX-Eigenschaft

Ermittelt oder bestimmt die aktuelle X-Position der nächsten Ausgabeposition in der Preview. Die Maßeinheit ist Twips.

SYNTAX:
Object.CurrentX[=PosX As Integer]

PARAMETER:
PosX: Beliebiger Integer-Ausdruck, der die neue Position des Cursors definiert.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

CurrentY-Eigenschaft

Ermittelt oder bestimmt die aktuelle Y-Position der nächsten Ausgabeposition in der Preview. Die Maßeinheit ist Twips.

SYNTAX:
Object.CurrentY[=PosX As Integer]

PARAMETER:
PosY: Beliebiger Integer-Ausdruck, der die neue Position des Cursors definiert.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Default

Bestimmt, ob eine Befehlsschaltfläche (CommandButton) die Funktion hat, die Standardeinstellungen zu aktivieren. Der Benutzer hat damit die Möglichkeit, sämtliche vorgenommenen Änderungen innerhalb eines Dialoges, die nicht der Standardeinstellung entsprechen, rückgängig zu machen.

Wenn Sie eine neue Befehlsschaltfläche (CommandButton) erstellen, handelt es sich standardmäßig nicht um eine Standard-Schaltfläche, die DEFAULT -Eigenschaft hat standardmäßig die Voreinstellung False.

Die DEFAULT-Eigenschaft kann nur zur Entwurfszeit gesetzt werden, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Switch As Boolean = Object.Default

PARAMETER:
True: Die Schaltfläche ist eine Standard-Schaltfläche.

False: Die Schaltfläche ist keine Standard-Schaltfläche.

FÜR ELEMENT:

CommandButton

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

DefaultChecked

Bestimmt, ob eine Option oder eine Markierung standardmäßig gesetzt ist. Diese Eigenschaft gilt für Optionsfelder (OptionBox) oder Markierfelder (CheckBox), die zur Gestaltung einer Web-Seite in ein Dokument eingefügt wurden. In Verbindung mit der FORM-Eigenschaft für Befehlschaltflächen (CommandButton) definiert DEFAULTCHECKED den Zustand des Kontrollfeldes, wenn der CommandButton die Funktion "Zurücksetzen" hat.

Die DEFAULTCHECKED-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.DefaultChecked [= True | False]

PARAMETER:
True: Das Kontrollfeld ist standardmaßig aktiv.

False: Das Kontrollfeld ist standardmäßig nicht aktiv (Voreinstellung).

Beispiel:

```
Sub ExampleDefault
Const sFileName = "\HTMLDoc.htm"
Const sItem = "Eintrag "
Const sCaption_1 = "Diese Checkbox ist nach dem Laden gesetzt."
Const sCaption_2 = "Dieser Optionbutton ist gesetzt."
```

```
Const sMessage_1 = "In diesem HTML-Dokument sind folgende
Eigenschaften voreingestellt."
Const sMessage_2 = "Bei dieser Combobox ist der dritte vorein-
gestellt."
Const sMessage_3 = "Bei dieser Listbox ist der zweite Eintrag
bei dem Laden gesetzt worden."
Dim sCtrlName As String, sPath As String
Dim iCount As Integer
sPath = Application.PathSettings.Work & sFileName
Documents.Add( "sWriter4/web" )
Selection.InsertTable( 5, 1 )
Selection.Insert sMessage_1
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 0 , 1, "7cm", "0.5cm" )
REM Checkbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_1
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 1 , 1, "7cm", "0.5cm" )
REM Optionbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_2
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 7 , 1, "3cm", "0.5cm" )
REM Combobox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 3
Activewindow.Controls(sCtrlName).ControlObject.DefaultText =
sItem & "3"
Selection.Escape
Selection.JumpToNextCell
Selection.SetRowHeight( "3.2cm" )
Activewindow.InsertVCControlAnchored( 6 , 1, "3cm", "3cm" ) REM
Listbox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
```

```
Activewindow.Controls(sCtrlName).ControlObject.DefaultSelected
= 2
Selection.Escape
Selection.JumpToStartOfDoc
ActiveDocument.Close TRUE, sPath
Documents.Open( sPath, "swriter4: HTML (StarWriter)" )
End Sub
```

DefaultSelected

Bestimmt, welcher Listeneintrag standardmäßig gesetzt ist. Diese Eigenschaft gilt für Listenfelder (ListBox), die zur Gestaltung einer Web-Seite in ein Dokument eingefügt wurden. In Verbindung mit der Form-Eigenschaft für Befehlschaltflächen (CommandButton) definiert DefaultSelected den Zustand des Kontrollfeldes, wenn der CommandButton die Funktion "Zurücksetzen" hat.

Die DefaultSelected-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.DefaultSelected [= Number As Integer]

Parameter:
Number: Integer-Ausdruck, der den Eintrag in dem Listenfeld angibt, welcher standardmäßig gesetzt ist.

Beispiel:

```
Sub ExampleDefault
Const sFileName = "\HTMLDoc.htm"
Const sItem = "Eintrag "
Const sCaption_1 = "Diese Checkbox ist nach dem Laden gesetzt."
Const sCaption_2 = "Dieser Optionbutton ist gesetzt."
Const sMessage_1 = "In diesem HTML-Dokument sind folgende
Eigenschaften voreingestellt."
Const sMessage_2 = "Bei dieser Combobox ist der dritte vorein-
gestellt."
Const sMessage_3 = "Bei dieser Listbox ist der zweite Eintrag
bei dem Laden gesetzt worden."
Dim sCtrlName As String, sPath As String
Dim iCount As Integer
sPath = Application.PathSettings.Work & sFileName
Documents.Add( "sWriter4/web" )
Selection.InsertTable( 5, 1 )
Selection.Insert sMessage_1
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 0 , 1, "7cm", "0.5cm" )
REM Checkbox
sCtrlName = ActiveWindow.ObjectName
```

```
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_1
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 1 , 1, "7cm", "0.5cm" )
REM Optionbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_2
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 7 , 1, "3cm", "0.5cm" )
REM Combobox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 3
Activewindow.Controls(sCtrlName).ControlObject.DefaultText =
sItem & "3"
Selection.Escape
Selection.JumpToNextCell
Selection.SetRowHeight( "3.2cm" )
Activewindow.InsertVCControlAnchored( 6 , 1, "3cm", "3cm" ) REM
Listbox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
Activewindow.Controls(sCtrlName).ControlObject.DefaultSelected
= 2
Selection.Escape
Selection.JumpToStartOfDoc
ActiveDocument.Close TRUE, sPath
Documents.Open( sPath, "swriter4: HTML (StarWriter)" )
End Sub
```

DefaultText

Bestimmt, welcher Text in einem Kontrollfeld standardmäßig gesetzt ist. Diese Eigenschaft gilt für Textfelder (TextBox) oder Kombinationsfelder (ComboBox), die zur Gestaltung einer Web-Seite in ein Dokument eingefügt wurden. In Verbindung mit der Form-Eigenschaft für Befehlschaltflächen

(CommandButton) definiert DefaultText den Zustand des Kontrollfeldes, wenn der CommandButton die Funktion "Zurücksetzen" hat.

Die DefaultText-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.DefaultText [= Text As String]

Parameter:
Text: Zeichenfolge, die den Text angibt, der standardmäßig gesetzt wird.

Beispiel:

```
Sub ExampleDefault
Const sFileName = "\HTMLDoc.htm"
Const sItem = "Eintrag "
Const sCaption_1 = "Diese Checkbox ist nach dem Laden gesetzt."
Const sCaption_2 = "Dieser Optionbutton ist gesetzt."
Const sMessage_1 = "In diesem HTML-Dokument sind folgende
Eigenschaften voreingestellt."
Const sMessage_2 = "Bei dieser Combobox ist der dritte vorein-
gestellt."
Const sMessage_3 = "Bei dieser Listbox ist der zweite Eintrag
bei dem Laden gesetzt worden."
Dim sCtrlName As String, sPath As String
Dim iCount As Integer
sPath = Application.PathSettings.Work & sFileName
Documents.Add( "sWriter4/web" )
Selection.InsertTable( 5, 1 )
Selection.Insert sMessage_1
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 0 , 1, "7cm", "0.5cm" )
REM Checkbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_1
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
Activewindow.InsertVCControlAnchored( 1 , 1, "7cm", "0.5cm" )
REM Optionbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion_2
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Activewindow.Controls(sCtrlName).ControlObject.DefaultChecked =
TRUE
Selection.Escape
Selection.JumpToNextCell
```

```
Activewindow.InsertVCControlAnchored( 7 , 1, "3cm", "0.5cm" )
REM Combobox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 3
Activewindow.Controls(sCtrlName).ControlObject.DefaultText =
sItem & "3"
Selection.Escape
Selection.JumpToNextCell
Selection.SetRowHeight( "3.2cm" )
Activewindow.InsertVCControlAnchored( 6 , 1, "3cm", "3cm" ) REM
Listbox
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
Activewindow.Controls(sCtrlName).ControlObject.DefaultSelected
= 2
Selection.Escape
Selection.JumpToStartOfDoc
ActiveDocument.Close TRUE, sPath
Documents.Open( sPath, "swriter4: HTML (StarWriter)" )
End Sub
```

DrawBox-Methode

Zeichnet ein Rechteck oder ein Quadrat im Preview-Dialogelement. Die Koordinaten werden in der Maßeinheit Twips angegeben.

Beim Positionieren eines Rechtecks im PREVIEW-Element werden die Attribute berücksichtigt, die durch die DRAWMODE-, die DRAWSTYLE-, die DRAWWIDTH- und FILLSTYLE-Eigenschaften bestimmt werden können.

SYNTAX:
Object.DrawBox Left As Integer, Top As Integer, Right As Integer, Bottom As Integer

PARAMETER:
Left: Beliebiger Integer-Ausdruck, der die Position der linken Seite des Rechtecks definiert.

Top: Beliebiger Integer-Ausdruck, der die Position der oberen Seite des Rechtecks definiert.

Right: Beliebiger Integer-Ausdruck, der die Position der rechten Seite des Rechtecks definiert.

Bottom: Beliebiger Integer-Ausdruck, der die Position der unteren Seite des Rechtecks definiert.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Für dieses Beispiel muß ein Dialog angelegt werden.
REM Der muß ein Preview-Element und einem CommandButton
besitzen.
REM Die Namen dieser Elemente sind, falls keine Standard-Namen
verwendet werden in den
REM Const-Bereich einzutragen.
REM Außerdem muß an dem CommandButton das Makro "ActionPreview"
für das Ereignis REM OnClickREM  konfiguriert werden.
REM Und für das Preview-Element muß das Ergnis REM OnPain-
tREM  mit "RepaintPreview" verbunden werden.
Const sDialog="Dialog1" REM Standard-Namen nach der Erstellung
der Elemente
Const sPreview="Preview1"
Const sButton="CommandButton1"
Const iPrevHeight=5400 REM Ausmaße des Preview Controls
Const iPrevWidth =7200
Private iAction As Integer REM dient zur Ablaufsteuerung
Dim oDialog As Object, oPreview As Object, oButton As Object
Dim iWidth As Integer, iHeight As Integer
Sub ExamplePreview
On Error Goto ErrorHandler
oDialog = FindObject( sDialog )
oPreview = FindPropertyObject(oDialog, sPreview )
oButton = FindPropertyObject(oDialog, sButton )
oDialog.Load
oDialog.Width = iPrevWidth +1600
oDialog.Height = 6000
oDialog.Caption = "Das ist eine Preview-Demonstration "
With oPreview REM Preview-Controlelement positionieren
.Top = 200
.Left = 200
.Height = iPrevHeight
.Width = iPrevWidth
.Border = TRUE
.BackColor = 0
.Refresh
iWidth  = .OutputWidth
iHeight = .OutputHeight
End With
With oButton REM Button formatieren
.Top = 200
.Left = iWidth + 400
```

```
.Height = 400
.Width = 1000
.Caption = "Weiter" REM Die Beschriftung für den CommandButton
.MousePointer = 10
End With
iAction = 0 REM Ablaufsteuerung zurücksetzen
oDialog.Show REM Dialog anzeigen
Exit Sub
ErrorHandler:
iInfo = Err
s = Error
MsgBox "Die Namen und Typen ihrer Kontroll-Elemente sind wahr-
scheinlich nicht korrekt." & Chr(13) &_
"Bitte kontrollieren Sie Ihren erstellten Dialog." ,16,"Es ist
ein Fehler aufgetreten:"
End Sub
REM Dieses Makro wird für das Repaint-Event verwendet.
Sub RepaintPreview
Dim oPic as Object
Set oPic=LoadPicture( Application.PathSettings.Config &
"\END.BMP" )
oPreview.DrawPicture oPic , 0 ,0 , iWidth, iHeight
End Sub
REM Dieses Makro wird für die Ablauf-Steuerung verwendet.
Sub ActionPreview
DIM iX As Integer, iY As Integer
DIM iPosX As Integer, iPosY As Integer
oButton.Enabled = FALSE
iAction = iAction + 1
oPreview.Cls
Select case iAction
Case 1 : REM Rechtecke malen
oDialog.Caption = "Das sind Rechtecke."
oPreview.DrawMode = 13 REM Überschreiben
oPreview.Fillstyle = 0
For iCount = 1 to 20
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawBox( iWidth*rnd() , iHeight*rnd(), iWidth*rnd()
,iHeight*rnd() )
Next iCount
Case 2 : REM Ellipsen malen
oDialog.Caption = "Das sind Ellipsen."
oPreview.DrawMode = 7 REM Invertieren
oPreview.Fillstyle = 0
For iCount = 1 to 50
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawEllipse( iWidth*rnd() , iHeight*rnd(),
iWidth*rnd() , iHeight*rnd() )
Next iCount
Case 3 : REM Linie
oDialog.Caption = "Das sind farbige Linien."
oPreview.BackColor = 0
oPreview.Cls
oPreview.DrawMode = 13 REM Überschreiben
```

```
oPreview.Fillstyle = 0
oPreview.DrawWidth = 50
oPreview.DrawStyle = 6 * rnd()
iCount = 0
For iCount1 = 0 to 256 Step 15
For iCount2 = 0 to 256 Step 15
For iCount3 = 0 to 256 Step 15
oPreview.CurrentX = iWidth/2
oPreview.CurrentY = iHeight/2
oPreview.ForeColor=RGB(iCount1,iCount2,iCount3)
oPreview.DrawLineTo( iWidth/2+Cos((iCount/360)*Pi)*iWidth/2
,iHeight/2+ Sin((iCount/360)*Pi)*iHeight/2)
iCount = iCount +1
Next iCount3
Next iCount2
Next iCount1
Case 4 : REM Linie
oDialog.Caption = "Das sind Linien."
oPreview.DrawMode = 13 REM Überschreiben
oPreview.Fillstyle = 0
For iCount = 0 to iWidth/2 Step 4
oPreview.DrawWidth = 100 * rnd()
oPreview.DrawStyle = 6 * rnd()
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawLine( iCount , iHeight*rnd(), iWidth - iCount ,
iHeight*rnd() )
Next iCount
Case 5 : REM Polygone
oDialog.Caption = "Das sind Polygone."
oPreview.DrawMode = 13 REM Überschreiben
oPreview.Fillstyle = 1
For iCount = 1 to 20
oPreview.DrawWidth = 100 * rnd()
oPreview.DrawStyle = 6 * rnd()
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawPolygon( iWidth*rnd() , iHeight*rnd(),
iWidth*rnd() , iHeight*rnd(), _
iWidth*rnd() , iHeight*rnd(), iWidth*rnd() , iHeight*rnd(),
iWidth*rnd() , iHeight*rnd() )
Next iCount
Case  7: REM Textausgabe
oPreview.BackColor = -1
oPreview.Cls
oDialog.Caption = "Das sind Texte."
For iCount = 1 to 200
oPreview.FontBold = (rnd()>0.5)
oPreview.FontItalic = (rnd()>0.5)
oPreview.FontName = "Times New Roman"
oPreview.FontSize = 48* rnd()
oPreview.FontStrikeout = (rnd()>0.5)
oPreview.FontUnderline = (rnd()>0.5)
iPosX=2*iWidth*rnd()-iWidth/2
iPosY=2*iHeight*rnd()-iHeight/2
iX = rnd() *4
```

```
oPreview.DrawMode = Switch(iX=0, 6, iX=1, 7, iX=2, 9, iX=3, 10,
iX=4, 13 )
oPreview.DrawStyle = 5 * rnd()
oPreview.FillStyle = 7 * rnd()
oPreview.DrawWidth = 100 * rnd()
iX = oPreview.GetTextWidth (" StarOffice 4.0")
iY = oPreview.GetTextHeight(" StarOffice 4.0")
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawBox( iPosX ,iPosY, iPosX+ iX, iPosY+iY )
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.FontTransparent = (rnd()>0.5)
oPreview.DrawText iPosX, iPosY, " StarOffice 4.0"
Next iCount
Case 6 : REM Einige bunte Punkte malen
oDialog.Caption = "Das sind Punkte."
oPreview.DrawMode = 13
oPreview.DrawStyle = 0
oPreview.FillStyle = 0
For iCount = 1 to 250
oPreview.DrawWidth = 1000 * rnd()
oPreview.ForeColor=RGB(255 * rnd(),255 * rnd(),255 * rnd())
oPreview.DrawPoint( (iWidth-1)*rnd() , (iHeight-1)*rnd() )
Next iCount
Case 8: REM Bild einfügen
oPreview.DrawMode = 13
oDialog.Caption = "Das ist ein Bild."
Call RePaintPreview
Case Else
iAction = 0 REM Wieder von vorne
End Select
oButton.Enabled = TRUE
If iAction = 0 Then
oDialog.Hide
oDialog.Unload
End If
End Sub
```

DrawEllipse-Methode

Zeichnet eine Ellipse oder einen Kreis im Preview-Dialogelement. Der Funktion werden zwei Koordinatenpaare übergeben, welche die Ausmaße eines Rechtecks bestimmen, die die zu zeichnende Ellipse exakt definieren. Die Koordinaten werden in der Maßeinheit Twips angegeben.

Beim Positionieren einer Ellipse im PREVIEW-Element werden die Attribute berücksichtigt, die durch die DRAWMODE-, die DRAWSTYLE-, die DRAWWIDTH- und FILLSTYLE-Eigenschaften bestimmt werden können.

SYNTAX:
Object.DrawEllipse Left As Integer, Top As Integer, Right As Integer, Bottom As Integer

PARAMETER:
Left: Beliebiger Integer-Ausdruck, der die Position der linken Seite des Rechtecks definiert, das die zu zeichnende Ellipse begrenzt.

Top: Beliebiger Integer-Ausdruck, der die Position der oberen Seite des Rechtecks definiert.

Right: Beliebiger Integer-Ausdruck, der die Position der rechten Seite des Rechtecks definiert.

Bottom: Beliebiger Integer-Ausdruck, der die Position der unteren Seite des Rechtecks definiert.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawLine-Methode

Zeichnet eine Linie oder Linienstrahlen im Preview-Dialogelement. Die Koordinaten werden in der Maßeinheit Twips angegeben.

Beim Zeichnen einer Linie im PREVIEW-Element werden die Attribute berücksichtigt, die durch die DRAWMODE-, die DRAWSTYLE-, die DRAWWIDTH- und FILLSTYLE-Eigenschaften bestimmt werden können.

SYNTAX:
Object.DrawLine X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer

PARAMETER:
X1, Y1: Beliebige Integerausdrücke, die die Koordinaten des Startpunktes der Linie beschreiben.

X2, Y2: Beliebige Integerausdrücke, die die Koordianten des Endpunktes der Linie beschreiben.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawLineTo-Methode

Zeichnet eine Linie oder Linienstrahlen im Preview-Dialogelement. Die Koordinaten werden in der Maßeinheit Twips angegeben.

> Beim Zeichnen einer Linie im PREVIEW-Element werden die Attribute berücksichtigt, die durch die DRAWMODE-, die DRAWSTYLE-, die DRAWWIDTH- und FILLSTYLE-Eigenschaften bestimmt werden können.

SYNTAX:
Object.DrawLineTo X2 As Integer, Y2 As Integer

PARAMETER:
X2, Y2: Beliebige Integerausdrücke, die die Koordianten des Endpunktes der Linie beschreiben.

Die Startkoordinaten ergeben sich aus der CURRENTX- und CURRENTY-Eigenschaft.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawMode-Eigenschaft

Bestimmt den Zeichenmodus für das Zeichnen mit den Preview-Zeichenmethoden.

SYNTAX:
Object.DrawMode = Modus As Integer

RÜCKGABE:
Integer

PARAMETER:
[= Modus]: Beliebiger Integer-Ausdruck, dessen Wert den Zeichenmodus festlegt. Es gilt:

6 - Vorhandene Farben werden durch deren Komplementärfarben ersetzt (INVERT).

7 - Mit dem Bitmuster eines Pixels an jeder Position wird eine logische Exclusive-Oder-Verknüpfung (XOR) durchgeführt.

9 - Mit dem Bitmuster eines Pixels an jeder Position wird eine logische Und-Verknüpfung (AND) durchgeführt.

10 - Mit dem Bitmuster eines Pixels an jeder Position wird eine logische Not-Exclusive-Oder-Verknüpfung (NXOR) durchgeführt.

13 - Vorhandene Pixel werden gemäß den Attributeinstellungen durch die DrawMode-, die DrawStyle-, die DrawWidth- und FillStyle-Eigenschaften überschrieben (OVERPAINT).

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Methode für das
Preview-Kontrollfeld.
```

DrawPicture-Methode

Lädt eine Bitmap-Datei in ein Preview-Steuerelement.

SYNTAX:
Object.DrawPicture pict, x1, y1, x2, y2

PARAMETER:
pict: Object, daß durch LoadPicture und Set initialisiert wurde.

y1, y1: Position an die die im pict-Objekt gespeicherte die Bilddatei im Preview-Element positioniert werden soll. Bezugspunkt ist die linke obere Ecke der Grafik.

Y2, y2: Diese Koordinaten bestimmen die Ausmaße der Grafik.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawPoint-Methode

Zeichnet einen einzelnen Pixel. Als Maßeinheit wird Twips verwendet.

SYNTAX:
Object.DrawPoint X As Integer,Y As Integer

PARAMETER:
X,Y: Beliebige Integerausdrücke, die die Koordinaten beschreiben, an denen der Punkt gezeichnet werden soll.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawPolygon-Methode

Zeichnet einen geschlossenen Linienzug.

SYNTAX:
Object.DrawPolygon(x1,y1, x2,y2, ..., xn,yn)

PARAMETER:
X1, Y1, Xn: Beliebige Integerausdrücke, die die Koordinaten des Startpunktes der Linie beschreiben.

X2, Y2, Yn: Beliebige Integerausdrücke, die die Koordianten des Endpunktes der Linie beschreiben.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawStyle-Eigenschaft

Bestimmt die "Linienstrichelung" die beim Zeichnen von Linien, Linienstrahlen, Rechtecken oder Ellipsen berücksichtigt werden soll.

SYNTAX:
Object.DrawStyle=Modus As Integer

PARAMETER:
Modus: Beliebiger Integer-Ausdruck, der durch seinen Wert den Linienzeichenmodus definiert. Es gilt:

0 - SOLID - Es werden durchgezogene Linien gezeichnet.

1 - DASH - Es werden gestrichelte Linien gezeichnet.

2 - DOT - Es werden gepunktete Linien gezeichnet.

3 - DASHDOT - Es werden wechselnd gestrichelt-gepunktete Linien gezeichnet.

4 - DASHDOT - wie 2

5 - NULL - Es werden keine Linien gezeichnet. Bei Flächenoperationen wie DRAWBOX, DRAWELLIPSE oder DRAWPOLYGONE wird das entsprechende Element nicht mit einer Linie umrandet.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawText-Methode

Positioniert eine Zeichenkette als Grafiktext im Preview-Fenster.

SYNTAX:
Object.DrawText X As Integer,Y As Integer, text As String

PARAMETER:
X, Y: Beliebiger Integerausdrücke, die die Koordinaten repräsentieren, an denen der erste Buchstabe der Zeichenkette ausgegeben werden soll. Es gilt als Richtlinie die Unterseite des ersten Buchstabens.

text: Beliebiger Zeichenkettenausdruck, der im Preview-Fenster ausgegeben werden soll.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

DrawWidth-Eigenschaft

Bestimmt die Linienstärke für alle Preview-Zeichenfunktionen.

SYNTAX:
Object.DrawWidth=Width As Integer

PARAMETER:
Width: Beliebiger Integer-Ausdruck, der die Linienstärke von Linien, Linienstrahlen, Rechtecken oder Ellipsen bestimmt. Die Linienstärke wird in Twips angegeben.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Enabled

Bestimmt, ob ein Kontrollfeld aktivierbar ist oder nicht. Ein aktivierbares Kontrollfeld ist dadurch gekennzeichnet, daß es auf vom Benutzer ausgelöste Ereignisse reagieren kann. Deaktivierte Kontrollfelder sind dagegen grau hinterlegt und stehen nicht zur Verwendung zur Verfügung. Diese Eigenschaft läßt sich sämtlichen Kontrollfeldern zuweisen, die in einen Dialog oder in ein Dokument eingefügt wurden.

Mit der ENABLED-Eigenschaft können Kontrollfelder zur Laufzeit aktiviert oder gesperrt werden, damit sie nur bei einem bestimmten Zustand der Anwendung verwendet werden.

Die ENABLED-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Enabled [= True | False]

PARAMETER:
True: Das Kontrollfeld ist aktivierbar (Voreinstellung).

False: Das Kontrollfeld ist nicht aktivierbar.

FÜR ELEMENT:

CheckBox, ComboBox, TextBox, Label, FrameControl, HScrollBar, ListBox, Preview, CommandButton, OptionButton, SpinButton, VScrollBar

Beispiel:

```
Sub ExampleEnabled
Const sTextboxText = "Nicht aktiv"
Const sMessage_1 = "Diese Textbox ist nicht aktiv"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4cm", "2cm", "0.5cm" )
REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.Enabled = FALSE
Selection.Escape
End Sub
```

FileUpload

Bestimmt, ob ein Textfeld (TextBox) in einem Dokument eine Schaltfläche zum Öffnen einer Datei enthält. Mit der FileUpload-Eigenschaft wird dem Textfeld eine Schaltfläche (...) zugewiesen, durch die der Dialog Öffnen zum Laden einer Datei aufgerufen wird.

Die FileUpload-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.FileUpload [= True | False]

Parameter:
True: Dem Textfeld wird eine Schaltfläche (...) zum Öffnen einer Datei hinzugefügt.

False: Das Textfeld wird nicht mit einer zusätzlichen Schaltfläche versehen (Voreinstellung).

Beispiel:

```
Sub ExampleFileUpload
Const sTextboxText = "Dateiauswahl"
Const sMessage_1 = "Mit dieser Textbox kann man jetzt Dateien
auswählen."
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4cm", "3.5cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
```

```
Text
Activewindow.Controls(sCtrlName).ControlObject.FileUpload =
TRUE
Selection.Escape
End Sub
```

FillStyle-Eigenschaft

Bestimmt die Schraffur, die beim Zeichnen von Rechtecken, Ellipsen oder Polygonen berücksichtigt werden soll.

SYNTAX:
Object.FillStyle=Modus As Integer

PARAMETER:
Modus: Beliebiger Integer-Ausdruck, der durch seinen Wert die Art der Schraffur definiert. Es gilt:

0 - SOLID - Flächen werden komplett ausgefüllt.

1 - NULL - Flächen werden gelöscht.

2 - HORZ - Es wird horizontal schraffiert.

3 - VERT - Es wird vertikal schraffiert.

4 - UPDIAG - Es wird mit diagonal aufsteigenden Linien schraffiert.

5 - DOWNDIAG - Es wird mit diagonal abfallenden Linien schraffiert.

6 - CROSS - Es wird mit horizontal und vertikal kreuzenden Linien schraffiert.

7 - DIACROSS - Es wird mit sich diagonal kreuzenden Linien schraffiert.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Font

Bestimmt die Art der Darstellung von Schrift bei Kontrollfeldern. Die FONT-Eigenschaft steht für alle Kontrollfelder zur Verfügung, die eine Beschriftung aufweisen, die durch die CAPTION-Eigenschaft festgelegt wird, oder bei Text-, Listen- und Kombinationsfeldern. Weiterhin kann sie dem Dialogelement Preview zugewiesen werden, um die Art der Darstellung von Schrift in einem Vorschaufeld zu bestimmen.

Ein Klick auf die Schaltfläche (...) neben dem Feld FONT im Eigenschaftenfenster öffnet den Dialog SCHRIFTART, in dem die Darstellung der Schrift festgelegt werden kann. Die Funktionen dieses Dialogs, der von Ihrem Betriebssystem bereitgestellt wird, finden Sie im SCHRIFT-Register des Menübefehls FORMAT - ZEICHEN... wieder.

Ein Doppelklick auf das Feld im Eigenschaftenfenster blendet sämtliche Font-Eigenschaften ein, die zur Verfügung stehen und die zur Laufzeit gesetzt oder gelesen werden können:

Object.FontBold [= True | False]

Object.FontItalic [= True | False]

Object.FontName [= Name As String]

Object.FontSize [= Size As Integer]

Object.FontStrikeout [= True | False]

Object.Underline [= True | False]

SYNTAX:
see Description

PARAMETER:
True: Die Beschriftung wird fett (Bold), kursiv (Italic), durchgestrichen (Strikeout) bzw. unterstrichen (Underline) dargestellt.

False: Die Beschriftung wird nicht mit dem jeweiligen Attribut versehen.

Name: Zeichenfolge, die den Schrifttyp bzw. den Zeichensatz festgelegt, also z. B. "New Times Roman". Sämtliche möglichen Schriften sind im Dialog SCHRIFTART im gleichnamigen Listenfeld aufgeführt.

Size: Integer-Ausdruck, der die Schriftgröße in Punkt angibt.

Beispiel:

```
Sub ExampleFont
Const sFontName = "Times New Roman"
Const sTextboxText = "Alle Zeichenattribute angewendet"
Const sMessage_1 = "Bei diese Textbox sind folgende Zeichenat-
tribute angewendet :"
Const sMessage_2 = "Zeichensatzname : "
Const sMessage_3 = "Zeichensatzgröße : 14"
Const sMessage_4 = "Der Zeichensatz ist : Unterstrichen, Fett,
Kursiv"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
```

```
Activewindow.InsertVCControl( 8, "8cm", "5.5cm", "7.5cm",
"0.8cm" ) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.FontName =
sFontName
Activewindow.Controls(sCtrlName).ControlObject.FontSize = 14
Activewindow.Controls(sCtrlName).ControlObject.FontUnderline =
True
Activewindow.Controls(sCtrlName).ControlObject.FontBold = True
Activewindow.Controls(sCtrlName).ControlObject.FontItalic =
True
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & sFontName
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
End Sub
```

FontTransparent-Eigenschaft

Bestimmt oder ermittelt, ob die Textausgeaben transparent oder nicht transparent erfolgen.

SYNTAX:
Object.FontTransparent [= Enable As BOOL]

PARAMETER:
Enable: True = Textausgaben werden transparent erfolgen. False = Textausgaben werden nicht transparent erfolgen.

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

ForeColor

Bestimmt die Vordergrundfarbe, die zum Anzeigen von Text bei einem Objekt verwendet wird. Diese Eigenschaft steht sowohl für Kontrollfelder als auch für Dialoge zur Verfügung.

Die FORECOLOR-Eigenschaft zeigt bei Kontrollfeldern im Dialog nur dann eine Wirkung, wenn der AUTO3D-Eigenschaft des Dialoges der Wert False zugewiesen wurde.

Die FORECOLOR-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.ForeColor [= Color As Long]

PARAMETER:
Color: Long-Integer-Ausdruck, der den Farbcode für die Vordergrundfarbe angibt.

FÜR ELEMENT:

CheckBox, ComboBox, Dialog, TextBox, Label, FrameControl, ListBox, Preview, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTextboxProperties
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es werden bei der Textbox verschiedene
Eigenschaften verändert."
Const sMessage_2 = "Hintergrundfarbe : "
Const sMessage_3 = "Vordergrundfarbe : "
Const sMessage_4 = "Umrandung : "
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.BackColor = RGB(
255, 0, 0 )
Activewindow.Controls(sCtrlName).ControlObject.ForeColor = RGB(
0, 255, 0 )
Activewindow.Controls(sCtrlName).ControlObject.Border = FALSE
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & RGB( 255, 0, 0 )
Selection.InsertPara
Selection.Insert sMessage_3 & RGB( 0, 255, 0 )
Selection.InsertPara
Selection.Insert sMessage_4 & "FALSE"
Selection.InsertPara
End Sub
```

Form

Bestimmt das Verhalten einer Befehlsschaltfläche (CommandButton) in einem Dokument. Dabei kann entschieden werden, ob ein Betätigen der Schaltfläche zum Versenden (beispielsweise von Daten) oder zum Zurücksetzen auf die in anderen Kontrollfeldern festgelegten Standardwerte führen soll. Die Standardwerte werden, abhängig vom jeweiligen Kontrollfeld, über die Eigenschaften DefaultChecked, DefaultSelected oder DefaultText festgelegt.

Die Form-Eigenschaft dient zur Erstellung von Web-Seiten. Sie kann nur zur Entwurfszeit festgelegt werden. Folgende Optionen sind dabei möglich:

Keine Aktion	Die Schaltfläche hat keine Funktion.
Senden	Betätigen der Schaltfläche führt zum Versenden von Daten, die in anderen Kontrollfeldern eingegeben werden.
Zurücksetzen	Betätigen der Schaltfläche führt zu einem Zurücksetzen der in anderen Kontrollfeldern angegebenen Daten auf die vordefinierten Standardwerte.

Forms bearbeiten

Mit dieser Funktion weisen Sie einem Kontrollfeld Eigenschaften einer Form zu. Unter einer Form wird hierbei ein elektronisches Formularblatt mit verschiedenen Kontrollfeld-Komponenten verstanden. Der Benutzer gibt hier seine Daten ein, um sie über das Internet zu versenden. Die Daten der Kontrollfelder einer Form werden an einen URL übertragen und können hier weiterverarbeitet werden.

Wenn Sie mit der Maus ein Kontrollfeld in Ihrem Dokument markieren und im Eigenschaftenfenster das Feld Form-Name anwählen, können Sie mit einem Klick auf die Schaltfläche (...) im Dialog Forms bearbeiten dem aktuellen Kontrollfeld spezielle Form-Eigenschaften zuweisen oder bereits bestehende einsehen und bearbeiten. Diese Eigenschaft steht nur für Kontrollfelder zur Verfügung, welche zur Gestaltung einer Web-Seite in ein Dokument eingefügt wurden.

Forms

In diesem Listenfeld sehen Sie alle namentlich definierten Forms. Hier können Sie aus der Liste per Mausklick die Form auswählen, der das aktuelle Kontrollfeld zugewiesen werden soll, oder Sie selektieren eine Form, um sie zu bearbeiten.

Name

In diesem Textfeld können Sie einen Namen für eine neue Form eingeben oder einen bestehenden Namen (außer "Standard") ändern.

URL

Hier wird der URL angegeben, zu dem die Inhalte der Kontrollfelder eines Formularblatts zur weiteren Bearbeitung gesendet werden.

Target

In diesem Feld geben Sie den Zielrahmen ein, in dem der angesprungene URL angezeigt werden soll. Die Bedeutung der vorgegebenen Einträge sehen Sie in der Tabelle der Bedeutungen.

Encoding Type

Wählen Sie hier den Encoding-Typ.

Methode

Wählen Sie hier die Methode, mit der die Daten des ausgefüllten Formularblatts übertragen werden sollen.

Get

Hier wählen Sie die Get-Methode. Die Daten jedes Kontrollfeldes werden als Environment-Variable übertragen. Sie werden an den URL in der Form "?Kontrollfeld1=Inhalt1&Kontrollfeld2=Inhalt2&..." angehängt, die Zeichenfolge wird von einem Programm auf dem Server des Empfängers ausgewertet.

Post

Hier wählen Sie die Post-Methode. Aus dem Inhalt der Form wird ein Dokument erzeugt, welches an den angegebenen URL gesendet wird.

Beenden

Mit dieser Schaltfläche beenden Sie den aktuellen Vorgang. Der Dialog wird geschlossen.

Hinzufügen

Wenn Sie einen neuen Namen eingegeben haben, können Sie ihn mit dieser Schaltfläche in die Liste der Forms aufnehmen.

Ändern

Wenn Sie in diesem Dialog Eigenschaften geändert haben, wird diese Schaltfläche aktiv, und Sie können die Änderungen permanent machen.

Löschen

Löscht das markierte Element oder die markierten Elemente ohne vorherige Sicherheitsabfrage.

Fenster-schließen-Symbol

Schließt den Dialog ohne Änderungen zu übernehmen. Die Schaltfläche in einem Dokumentfenster und die Schaltfläche in der Titelleiste der Anwendung schließen das Dokument oder die Anwendung. Wurde das Dokument noch nicht gespeichert, so werden Sie in einem Dialog gefragt, ob Sie dies nachholen möchten.

Fenster-maximieren-Symbol

Vergrößert das Fenster auf den maximal zur Verfügung stehenden Platz.

Fenster-minimieren-Symbol

Verkleinert das Fenster auf Symbolgröße.

Fenster-wiederherstellen-Symbol

Vergrößert das auf Symbolgröße verkleinerte Fenster wieder auf seine Ursprungsgröße und positioniert es an der vorherigen Position. Verkleinert das auf Vollbild vergrößerte Fenster wieder auf seine vorherige Größe (nicht Vollbild, nicht Symbol).

Vorschaufeld

In diesem Feld sehen Sie an einem stilisierten Beispiel, wie sich Änderungen, die Sie jetzt vornehmen können, auf Ihr Dokument auswirken werden.

Drehfeld

Geben Sie hier in das Textfeld den numerischen Wert ein, oder wählen Sie ihn mit der Pfeil-aufwärts- oder Pfeil-abwärts-Schaltfläche. Auf der Tastatur können Sie mit den Tasten (Pfeil-nach-oben) und (Pfeil-nach-unten) den Wert im Drehfeld erhöhen und verringern, während Sie mit (Bild-auf) und (Bild-ab) den maximalen und minimalen Wert für dieses Drehfeld einstellen.

Handelt es sich um ein Drehfeld für numerischen Werte, dann können Sie auch eine Maßeinheit mit angeben, also z. B. 1 cm oder 5 mm oder 12 pt oder 2" (für 2 Zoll).

Abreißleiste

Einige Symbole öffnen durch einfachen Klick Abreißleisten. Das sind Ausklappmenüs, in denen Sie unter verschiedenen Elementen per Mausklick eines auswählen können. Wenn Sie aber auf das Symbol zeigen, die Maustaste drücken und bei gedrückter Maustaste die Maus von dem Symbol wegziehen, so folgt der Umriß des gesamten Ausklappmenüs dem Mauszeiger.Sobald Sie die Maustaste loslassen, bleibt die Abreißleiste an der neuen Position stehen. Sie können sie dann auf Wunsch an der Titelleiste greifen und an einen anderen Ort verschieben, oder Sie schließen sie durch Klick auf das Fenster-schließen-Symbol.

Kontextmenü

Rufen Sie das Kontextmenü zu einem Objekt auf, indem Sie zuerst das Objekt mit der linken Maustaste anklicken, um es zu markieren, und dann mit der rechten Maustaste klicken. Kontextmenüs existieren außer zu markierten Objekten (Textstellen, Zeichenobjekten, eingefügten Grafiken, Tabellen usw.) auch zu dem im Navigator und im Stylisten enthaltenen Inhalt, ferner zu den Einträgen im Explorer und Beamer und an vielen anderen Stellen. Klicken Sie während Ihrer Arbeit mit StarOffice einfach versuchsweise überall auch einmal mit der rechten Maustaste.

Ein Kontextmenü bietet Ihnen die Möglichkeit, auch ohne den Umweg durch die Menüstruktur der Menüleiste schnell einige der in diesem Zusammenhang (Kontext) wichtigsten Befehle aufzurufen. Sie ersparen sich dabei nicht nur, das richtige Menü für die von Ihnen beabsichtigte Funktion oder den Befehl aufzuklappen und dann anzuwählen, sondern Sie brauchen auch nicht einmal weit mit der Maus zu wandern, denn das Kontextmenü zu einem markierten oder selektierten Objekt wird in der Regel direkt neben dem Objekt eingeblendet. Damit ist der Mausweg vom Markieren des Objekts bis zum Aufruf des Befehls besonders kurz und schnell. Das Kontextmenü zeigt

Ihnen zum selektierten Objekt einige Menübefehle, die in diesem Zusammenhang von Nutzen sind.

Verkleinern/Vergrößern

Klicken Sie auf dieses Symbol, um den Dialog auf die Größe des Eingabefelds zu verkleinern oder um den Dialog in Orginalgröße wiederherzustellen. Sie können dann besser den gewünschten Bezug in der Tabelle markieren. Das Symbol wird automatisch zum Vergrößern-Symbol. Klicken Sie auf das Vergrößern-Symbol, und der Dialog wird wiederhergestellt.

Das Verkleinern des Dialogs erfolgt automatisch, wenn Sie in der Tabelle die Maustaste drücken und einen Bereich aufziehen. Sobald Sie die Maustaste loslassen, wird der Dialog wieder in ganzer Größe gezeigt.

Verkleinern

Vergrößern

Zusätze

Mit einem Klick auf diese Schaltfläche erweitern Sie den Dialog um weitere Optionen. Mit einem erneuten Klick stellen Sie die ursprüngliche Ansicht des Dialoges wieder her.

Makros...

Über diese Schaltfläche gelangen Sie in den Dialog Makro zuweisen. Hier können Sie ein Makro zuweisen, das ausgeführt werden soll.

GetTextHeight-Methode

Ermittelt die in einem Preview-Steuerelement auszugebende Höhe eines Textes. Als Maßeinheit wird Twips verwendet.

SYNTAX:
Object.GetTextHeight(text As String)

PARAMETER:
text: Beliebiger Zeichenkettenausdruck, dessen Ausmaße ermittelt werden sollen.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

GetTextWidth-Methode

Ermittelt die in einem Preview-Steuerelement auszugebende Länge eines Textes. Als Maßeinheit wird Twips verwendet.

SYNTAX:
PreviewName.GetTextWidth(text As Integer)

PARAMETER:
text: Beliebiger Zeichenkettenausdruck, dessen Ausmaße ermittelt werden sollen.

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Height

Bestimmt die Höhe eines Kontrollfeld-Objektes.

Die HEIGHT-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Height [= Height As Integer]

PARAMETER:
Height: Integer-Ausdruck, der die Höhe des Objekts in Twips angibt.

FÜR ELEMENT:

CheckBox, ComboBox, Dialog, TextBox, Label, FrameControl, HScrollBar, ListBox, Preview, CommandButton, OptionButton, SpinButton, VScrollBar

Beispiel:

```
Sub ExampleTextboxPosition
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Diese Textbox hat folgende Koordinaten"
Const sMessage_2 = "Von links : "
Const sMessage_3 = "Von oben : "
Const sMessage_4 = "Breite : "
```

```
Const sMessage_5 = "Höhe : "
Dim sCtrlName As String
Dim lLeft As Long, lTop As Long, lWidth As Long, lHeight As
Long
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "10cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
lLeft = Activewindow.Controls(sCtrlName).ControlObject.Left
lTop = Activewindow.Controls(sCtrlName).ControlObject.Top
lWidth = Activewindow.Controls(sCtrlName).ControlObject.Width
lHeight = Activewindow.Controls(sCtrlName).ControlObject.Height
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & lLeft & " Twips"
Selection.InsertPara
Selection.Insert sMessage_3 & lTop & " Twips"
Selection.InsertPara
Selection.Insert sMessage_4 & lWidth & " Twips"
Selection.InsertPara
Selection.Insert sMessage_5 & lHeight & " Twips"
Selection.InsertPara
End Sub
```

HelpContextID

Bestimmt eine eindeutige HelpID-Nummer. Die HelpID-Nummer dient zur Zuordnung einer kontextsensitiven Hilfe für das jeweilige Kontrollfeld-Objekt und kann vom Anwender frei gewählt werden.

Die HEIGHT-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.HelpContextID [= Number As Integer]

PARAMETER:
Number: Integer-Ausdruck, der die HelpID-Nummer für eine kontextbezogene Hilfe angibt. Hat Number den Wert 0 (Voreinstellung), so wird keine spezielle HelpID-Nummer zugeordnet.

FÜR ELEMENT:

CheckBox, ComboBox, Dialog, TextBox, Label, FrameControl, HScrollBar, ListBox, Preview, CommandButton, OptionButton, SpinButton, VScrollBar

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

HelpText

Bestimmt den Hilfetext, der in der aktiven Hilfe erscheinen soll. Im StarOffice wird die aktive Hilfe gestartet durch den entsprechenden Menübefehl im HILFE-Menü oder durch einen Mausklick auf das Fragezeichen-Symbol eines Dialoges. Sie zeigt die eingetragenen Hilfetexte als Popup an.

Die HELPTEXT-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.HelpText [= HelpString As String]

RÜCKGABE:
String

PARAMETER:
HelpString: Zeichenfolge, die die kontextbezogenen Hilfe angibt.

FÜR ELEMENT:

CheckBox, ComboBox, Dialog, TextBox, Label, FrameControl, HScrollBar, ListBox, Preview, CommandButton, OptionButton, SpinButton, VScrollBar

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Hide-Methode

Entfernt einen Dialog vom Bildschirm, jedoch nicht aus dem Arbeitsspeicher. Auf den Dialog kann weiterhin zugegriffen werden, da dieser lediglich vom Bildschirm, nicht aber aus dem Arbeitsspeicher entfernt wurde. Möchten Sie den Dialog ganz aus dem Arbeitsspeicher entfernen, verwenden Sie bitte die UNLOAD-Methode.

SYNTAX:
Object.Hide

PARAMETER:
FÜR ELEMENT:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Image-URL

Mit dieser Eigenschaft wird eine Befehlsschaltfläche (CommandButton) in einem Dokument mit einer Grafik dargestellt.

Wenn Sie im Eigenschaftenfenster das Feld IMAGE-URL markiert haben, können Sie mit einem Klick auf die Schaltfläche (...) einen Dialog aufrufen, in dem Sie eine Grafik wählen können. Die Grafik wird als Bild auf der Schaltfläche oder im Bereich gezeigt und überdeckt einen eventuellen Text an dieser Stelle.

Der Dialog IMAGE-URL SETZEN entspricht in seinem Aufbau dem Dialog GRAFIK EINFÜGEN.

Diese Eigenschaft kann nur zur Entwurfszeit festgelegt werden.

BEISPIEL:

```
Kein Beispiel.
```

ItemData

Bestimmt die Nummer für einen Eintrag in einem Kontrollfeld. Die ITEMDATA-Eigenschaft gilt für Listenfelder (ListBox) und für Kombinationsfelder (ComboBox).

Klicken Sie im Eigenschaftenfenster auf die Pfeilschaltfläche, so können Sie im aufklappenden Listenfeld eine Nummer für einen Objekteintrag eingeben. Drücken Sie die Eingabetaste bei gedrückt gehaltener Umschalttaste, um weitere Nummern zu vergeben. Die Objekteinträge legen Sie mit der LIST-Eigenschaft fest.

Die ITEMDATA-Eigenschaft kann zur Entwurfszeit oder zur Laufzeit gesetzt und zur Laufzeit abgefragt oder geändert werden.

SYNTAX:
Object.ItemData(Index) [= Number As Integer]

PARAMETER:
Index: Der Index eines bestimmten Eintrags im Objekt. Die Einträge sind fortlaufend durchnummeriert, wobei der erste Eintrag den Index 0 aufweist.

Number: Die Nummer, die dem bestimmten Eintrag zugeordnet werden soll.

FÜR ELEMENT:

ComboBox, ListBox

Beispiel:

```
Sub ExampleList
Const sItem = "Eintrag "
Const sHeader_1 = "Folgende Einträge sind der der Listbox."
Const sHeader_2 = "Die Einträge in der Listbox während der
Eingabe sortiert."
Const sHeader_3 = "Eintrag : ItemData "
Dim sCtrlName As String, sValue As String
Dim sItemData as String
Dim iCount As Integer, iNewIndex As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "12cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Sorted = TRUE
For iCount = 1 To 9
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
10 - iCount
iNewIndex =
Activewindow.Controls(sCtrlName).ControlObject.NewIndex
Activewindow.Controls(sCtrlName).ControlObject.ItemData(
iNewIndex ) = iCount
Next iCount
Selection.Escape
Selection.Insert sHeader_1
Selection.InsertPara
Selection.Insert sHeader_2
Selection.InsertPara
Selection.Insert sHeader_3
Selection.InsertPara
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
sValue =
Activewindow.Controls(sCtrlName).ControlObject.List(iCount)
sItemData =
Activewindow.Controls(sCtrlName).ControlObject.ItemData(iCount)
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sValue  & " : "
Selection.Insert sItemData
Selection.InsertPara
Next iCount
End Sub
```

LargeChange

Die LargeChange-Eigenschaft bestimmt den Wert, um den die Value-Eigenschaft vermindert oder erhöht wird, wenn auf die Bildlaufleiste zwischen einem der Bildlaufpfeile und dem Bildlauffeld geklickt wird. Wird die Bildlaufleiste über, respektive links von dem Bildlauffeld angeklickt, vermindert sich der Wert der Value-Eigenschaft, während ein Klick unter oder rechts von dem Bildlauffeld den Wert erhöht. Die SmallChange-Eigenschaft verändert den Wert der Value-Eigenschaft, wenn auf einen der Bildlaufpfeile der entsprechenden Bildlaufleisten geklickt wird. Der Wertebereich kann im Bereich von 1 bis 32.767 liegen.

Die LARGECHANGE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.LargeChange [= Number As Integer]

RÜCKGABE:
Integer

PARAMETER:
Number: Integer-Ausdruck, der die Größe der Änderung der VALUE-Eigenschaft angibt. Standardmäßig hat Number den Wert 1.

FÜR ELEMENT:

HScrollBar, VScrollBar

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
```

```
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

Left

Bestimmt den Abstand zwischen dem linken Rand des Kontrollfelds und dem linken Rand des Dokumentes oder des Dialoges.

Die LEFT-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Left [= Distance As Integer]

PARAMETER:
Distance: Integer-Ausdruck, der den Wert des Abstands in Twips festlegt.

FÜR ELEMENT:

Dialog, CheckBox, ListBox, ComboBox, FrameControl, TextBox, Label, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTextboxPosition
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Diese Textbox hat folgende Koordinaten"
Const sMessage_2 = "Von links : "
```

```
Const sMessage_3 = "Von oben : "
Const sMessage_4 = "Breite : "
Const sMessage_5 = "Höhe : "
Dim sCtrlName As String
Dim lLeft As Long, lTop As Long, lWidth As Long, lHeight As
Long
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "10cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
lLeft = Activewindow.Controls(sCtrlName).ControlObject.Left
lTop = Activewindow.Controls(sCtrlName).ControlObject.Top
lWidth = Activewindow.Controls(sCtrlName).ControlObject.Width
lHeight = Activewindow.Controls(sCtrlName).ControlObject.Height
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & lLeft & " Twips"
Selection.InsertPara
Selection.Insert sMessage_3 & lTop & " Twips"
Selection.InsertPara
Selection.Insert sMessage_4 & lWidth & " Twips"
Selection.InsertPara
Selection.Insert sMessage_5 & lHeight & " Twips"
Selection.InsertPara
End Sub
```

Link

Die Eigenschaft verknüpft ein Kontrollfeld mit einem Feldbefehl, wobei der Zustand des Kontrollfeldes den Wert des Feldbefehls bestimmt. Wenn Sie als LINK-Eigenschaft den Namen eines benutzerdefinierten Feldbefehles eintragen, wird der Wert des Feldbefehles automatisch an den aktuellen Zustand des Kontrollfeldes angepaßt.

Bei den Kontrollfeldern CheckBox und OptionBox nimmt der Fehlbefehl den Wert True oder False an, je nachdem ob das Kontrollfeld markiert wurde oder nicht. Bei den Kontrollfeldern ComboBox, ListBox und TextBox wird der aktuelle Eintrag zum Wert des Feldbefehls. Bei Bildlaufleisten und Drehfeldern (HScrollBar, VscrollBar, SpinButton) gibt der Wert des Feldbefehls die durch die Value-Eigenschaft definierte Position des Objektes an.

Die LINK-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Link [=Statement As String]

PARAMETER:
Statement: Zeichenfolge, die den Namen des Feldbefehls angibt, mit dem das Kontrollfeld verknüpft werden soll.

Beispiel:

```
Sub ExampleLink
Const sFieldName = "Benutzerfeld"
Const sFieldCaption = "Verknüpft mit einem Benutzerfeld"
Const sMessage_1 = "Diese Checkbox ist mit dem Benutzerfeld
verbunden."
Const sMessage_2 = "Ändert man die Checkbox, dann wird automa-
tisch das Benutzerfeld geändert."
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertField( 20, 0, "Benutzerfeld", "TRUE" )
Selection.Insert " <->   "
Activewindow.InsertVCControl( 0, "5cm", "5cm", "8cm", "0.5cm" )
REM Checkbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sMes-
sage_1
Activewindow.Controls(sCtrlName).ControlObject.Link = sField-
Name
Activewindow.Controls(sCtrlName).ControlObject.Value = TRUE
Selection.Escape
Selection.JumpToStartOfDoc
End Sub
```

List

Ermittelt Objekteinträge oder legt sie fest. Die LIST-Eigenschaft gilt für Listenfelder (ListBox) und für Kombinationsfelder (ComboBox).

Klicken Sie zur Entwurfszeit im Eigenschaftenfenster auf die Pfeilschaltfläche, so können Sie im aufklappenden Listenfeld einen Eintrag für das Objekt eingeben. Drücken Sie die Eingabetaste bei gedrückt gehaltener Umschalttaste, um weitere Einträge einzugeben.

Diese Einträge können Sie mit der LIST-Eigenschaft auch zur Laufzeit setzen oder abfragen.

Um mit StarBasic einen Eintrag in ein Listen- oder Kombinationsfeld hinzuzufügen, können Sie auch die ADDITEM-Methode verwenden.

SYNTAX:
Object.List(Index) [= Entry As String]

RÜCKGABE:
String

PARAMETER:
Index: Der Index eines bestimmten Eintrags im Objekt. Die Einträge sind fortlaufend durchnummeriert, wobei der erste Eintrag den Index 0 aufweist.
Entry: Zeichenfolge, die den Objekteintrag festlegt.

FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleList
Const sItem = "Eintrag "
Const sHeader_1 = "Folgende Einträge sind der der Listbox."
Const sHeader_2 = "Die Einträge in der Listbox während der
Eingabe sortiert."
Const sHeader_3 = "Eintrag : ItemData "
Dim sCtrlName As String, sValue As String
Dim sItemData as String
Dim iCount As Integer, iNewIndex As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "12cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Sorted = TRUE
For iCount = 1 To 9
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
10 - iCount
iNewIndex =
Activewindow.Controls(sCtrlName).ControlObject.NewIndex
Activewindow.Controls(sCtrlName).ControlObject.ItemData(
iNewIndex ) = iCount
Next iCount
Selection.Escape
Selection.Insert sHeader_1
Selection.InsertPara
Selection.Insert sHeader_2
Selection.InsertPara
Selection.Insert sHeader_3
Selection.InsertPara
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
sValue =
Activewindow.Controls(sCtrlName).ControlObject.List(iCount)
sItemData =
Activewindow.Controls(sCtrlName).ControlObject.ItemData(iCount)
Selection.Escape
Selection.JumpToEndOfDoc
```

```
Selection.Insert sValue  & " : "
Selection.Insert sItemData
Selection.InsertPara
Next iCount
End Sub
```

ListCount-Eigenschaft

Mit dieser Funktion ermitteln Sie die Anzahl aller vorhandenen Einträge eines Listen- oder Kombinationslistenfeldes. Es sind keine Einträge vorhanden, wenn der Wert 0 zurückgeliefert wird. Das erste Element in der Liste hat den Wert 0.

SYNTAX:
Object.ListCount

PARAMETER:
FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleList
Const sItem = "Eintrag "
Const sHeader_1 = "Folgende Einträge sind der der Listbox."
Const sHeader_2 = "Die Einträge in der Listbox während der
Eingabe sortiert."
Const sHeader_3 = "Eintrag : ItemData "
Dim sCtrlName As String, sValue As String
Dim sItemData as String
Dim iCount As Integer, iNewIndex As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "12cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Sorted = TRUE
For iCount = 1 To 9
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
10 - iCount
iNewIndex =
Activewindow.Controls(sCtrlName).ControlObject.NewIndex
Activewindow.Controls(sCtrlName).ControlObject.ItemData(
iNewIndex ) = iCount
Next iCount
Selection.Escape
Selection.Insert sHeader_1
Selection.InsertPara
Selection.Insert sHeader_2
Selection.InsertPara
Selection.Insert sHeader_3
Selection.InsertPara
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
```

```
sValue =
Activewindow.Controls(sCtrlName).ControlObject.List(iCount)
sItemData =
Activewindow.Controls(sCtrlName).ControlObject.ItemData(iCount)
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sValue  & " : "
Selection.Insert sItemData
Selection.InsertPara
Next iCount
End Sub
```

ListIndex-Eigenschaft

Ermittelt oder bestimmt den Index des gegenwärtig ausgewählten Eintrags eines Listenfeldes oder Kombinationslistenfeldes. Ist in einer Liste keine Feld selektiert, dann wird er Wert -1 zurückgeliefert.

SYNTAX:
Object.ListIndex [=Index]

PARAMETER:
Index: Beliebiger numerischer Ausdruck, der den gegenwärtig ausgewählten Eintrag eines Listenfeldes oder Kombinationslistenfeldes kennzeichnet.

FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleAddItem
Const sItem = "Eintrag "
Dim sCtrlName As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
End Sub
```

Load-Methode

Lädt einen Dialog in den Arbeitsspeicher, zeigt diesen aber noch nicht auf dem Bildschirm an.

SYNTAX:
Object.Load

PARAMETER:
FÜR ELEMENT:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Macros

Das Feld MACROS im Eigenschaftenfenster dient zur Zuordnung eines Makros. Wenn Sie das Feld markiert haben, können Sie mit einem Klick auf die Schaltfläche (...) den Dialog MAKRO ZUWEISEN aufrufen, in dem Sie dem aktuellen Objekt ein Makro zuweisen, das bei einem bestimmten Ereignis eintritt.

Der Dialog MAKRO ZUWEISEN entspricht dem Register MAKRO im StarOffice.

BEISPIEL:

```
Kein Beispiel.
```

Max

Bestimmt den maximalen Wert der VALUE-Eigenschaft eines Kontrollfelds. Dieser Wert wird erreicht, wenn sich die Bildlaufleiste an der unteren Position (VScrollBar) oder der rechten Position (HScrollBar) befindet oder wenn ein Drehfeld (SpinButton) seinen maximalen Betrag annimmt.

Normalerweise werden die MAX- oder MIN Eigenschaften zur Entwurfszeit festgelegt. In der Voreinstellung hat der Maximalwert den größtmöglichen Wert 32767. Dieser Wert kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Max [= Max As Integer]

PARAMETER:
Max: Integer-Ausdruck, durch den der maximale Wert der VALUE-Eigenschaft festgelegt wird.

FÜR ELEMENT:

HScrollBar, VscrollBar, SpinButton

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
```

```
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

MaxLength

Bestimmt die maximale Anzahl von Zeichen in einem Textfeld (TextBox). Über die MaxLength-Eigenschaft kann die Anzahl der Zeichen in einem Textfeld begrenzt werden. Ist der Wert 0 eingetragen, können maximal 32 kB Zeichen eingetragen werden. Eine Begrenzung ist dann sinnvoll, wenn die Inhalte aus der TextBox in eine vorhandene Datenbankstruktur übertragen werden sollen.

Die MaxLength-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.MaxLength [= Number As Integer]

Rückgabe:
Integer

Parameter:
Number: Integer-Ausdruck, der die maximale Anzahl von Zeichen angibt, die der Benutzer in das Textfeld eingeben kann.

Für Element:

TextBox

Beispiel:

```
Sub ExampleMaxLength
Const sTextboxText = "NochSichtbarNichtSichtbar"
Const sMessage_1 = "Bei dieser Textbox können nur 12 Zeichen
eingegeben werden."
Const sMessage_2 = "Folgender Text wurde eingeben : "
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4.5cm", "4cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.MaxLength = 12
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & sTextboxText
Selection.InsertPara
End Sub
```

Min

Bestimmt den minimalen Wert der Value-Eigenschaft eines Kontrollfelds. Dieser Wert wird erreicht, wenn sich die Bildlaufleiste an der oberen Position (VScrollBar) oder der linken Position (HScrollBar) befindet oder wenn ein Drehfeld (SpinButton) seinen minimalen Betrag annimmt.

Normalerweise werden die Max- oder Min Eigenschaften zur Entwurfszeit festgelegt. In der Voreinstellung hat der Minimalwert den kleinstmöglichen Wert 0. Dieser Wert kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Min [= Min As Integer]

PARAMETER:
Min: Integer-Ausdruck, durch den der minimale Wert der VALUE-Eigenschaft festgelegt wird.

FÜR ELEMENT:

HScrollBar, VScrollBar, SpinButton

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
```

```
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

MousePointer

Bestimmt die Gestalt des Mauscursors, wenn dieser sich über dem Objekt befindet.

Die MousePointer-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.MousePointer [= Number As Integer]

Parameter:
Number: Integer-Ausdruck, der den Typ des Mauscursors bestimmt. Folgende Werte sind möglich:

0 - Standardeinstellung

1 - Pfeil (Arrow)

2 - Kreuz (Cross)

3 - I-Cursor (I-Beam) bei TextBoxen

4 - Icon (wird zur Zeit nicht verwendet)

5 - Kreuzzeiger in alle vier Richtungen mit Pfeilspitzen (Size)

6 - Zeiger nach Nord-Ost und Süd-West (Size NW SW)

7 - Zeiger nach West und Ost (Size N S)

8 - Zeiger nach Nord-West und Süd-Ost (Size NW SE)

9 - Zeiger nach Nord und Süd (Size W E)

10 - Handzeiger (UpArrow)

11 - Wartezustand (Hourglass)

12 - Kein Ablegen (NoDrop)

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Move-Methode

Positioniert die unter Dialog-Elemente aufgeführten Elementen oder weist ihnen eine bestimmte Größe zu.

SYNTAX:
Element.Move X [, Y[, Breite[, Höhe]]]

PARAMETER:
X: Ausdruck vom Typ Single. Dieser Parameter gibt die x-Koordinate der linken, oberen Begrenzung des Elementes an.

Y: Ausdruck vom Typ Single. Dieser Parameter gibt die y-Koordinate der linken, oberen Begrenzung des Elementes an.

Breite: Ausdruck vom Typ Single, der die Breite des Objektes kennzeichnet.

Höhe: Ausdruck vom Typ Single, der die Höhe des Objektes kennzeichnet.

Als Maßeinheit wird Twips verwendet.

FÜR ELEMENT:

CheckBox, Dialog, TextBox, Label, FrameControl, ListBox, ComboBox, Preview, CommandButton, OptionButton, HScrollBar, VScrollBar, SpinButton.

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

MultiLine

Mit dieser Eigenschaft wird festgelegt, ob ein Textfeld (TextBox) einen Zeilenumbruch darstellen kann. Standardmäßig werden in Textfeldern keine Zeilenumbrüche verarbeitet. Das heißt, wenn in einer TextBox die Zeichen CR (ASCII 13) und LF (ASCII 10) vorhanden sind, wird die Zeile nicht umgebrochen. Dieses Verhalten ändert sich, wenn die Eigenschaft MULTILINE = True ist.

Die MULTILINE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.MultiLine [= True | False]

PARAMETER:
True: Mehrere Textzeilen sind möglich.

False: Ein Zeilenumbruch findet nicht statt.

Beispiel:

```
Sub ExampleMultiline
Const sTextboxText = "Dies ist ein mehrzeiliger Textboxinhalt"
Const sMessage_1 = "Diese Textbox ist mehrzeilig und hat
Scrollbars"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4cm", "6cm", "2cm" )
REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.MultiLine = TRUE
Selection.Escape
End Sub
```

MultiSelect

Bestimmt, ob und wie in einem Listenfeld (ListBox) mehrere Einträge markiert werden können.

Die MULTISELECT-Eigenschaft wird zur Entwurfszeit festgelegt. Die Einstellung kann zur Laufzeit abgefragt, aber nicht geändert werden.

SYNTAX:
Value As Integer = Object.MultiSelect

PARAMETER:
Value: Integer-Ausdruck, der die Möglichkeit der Mehrfachselektion festlegt. Value kann folgende Werte annehmen:

0 - Keine Mehrfachauswahl möglich.

1 - Mehrfachauswahl möglich (Strg+Mausklick).

2 - Erweiterte Mehrfachauswahl möglich (wie 1 aber zusätzlich kann beim

Ziehen mit gedrückter Maustaste die gesamte Liste markiert werden).

Beispiel:

```
Kein Beispiel.
```

Name-Eigenschaft

Mit dieser Eigenschaft wird der Name des Objektes festgelegt, mit dem das Objekt aus StarBasic heraus angesprochen wird. Standardmäßig wird der Name automatisch beim Erstellen eines Objektes festgelegt. Dabei setzt sich die Bezeichnung zusammen aus dem Objektnamen und einer Ganzzahl zur Numerierung des Objekts, also z. B. ListBox1. Dieser Name kann nur zur Entwurfszeit geändert werden.

Die Bezeichnung eines Objektes muß eindeutig sein und mit einem Buchstaben beginnen. Die Länge der Bezeichnung darf 40 Zeichen nicht überschreiten. Satz- und Leerzeichen sind ebensowenig zulässig wie deutsche Umlaute. Es gelten desweiteren dieselben Konventionen wie bei der Vergabe von Variablennamen.

PARAMETER:
FÜR ELEMENT:

CheckBox, Dialog, TextBox, Label, Font, FrameControl, ListBox, ComboBox, Preview, CommandButton, OptionButton, HScrollBar, VScrollBar, SpinButton

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
```

```
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

NewIndex-Eigenschaft

Bestimmt den Index des Eintrages, der zuletzt einem Listenfeld oder einem Kombinationslistenfeld hinzugefügt wurde.

SYNTAX:
Index As Integer = Object.NewIndex

RÜCKGABE:
Integer

PARAMETER:
Index: Zurückgegebener Integer-Ausdruck, der den Index des Eintrages bezeichnet, der zuletzt einem Listenfeld oder Kombinationslistenfeld hinzugefügt wurde.

FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleList
Const sItem = "Eintrag "
Const sHeader_1 = "Folgende Einträge sind der der Listbox."
Const sHeader_2 = "Die Einträge in der Listbox während der
Eingabe sortiert."
Const sHeader_3 = "Eintrag : ItemData "
Dim sCtrlName As String, sValue As String
Dim sItemData as String
Dim iCount As Integer, iNewIndex As Integer
Documents.Add( "sWriter4" )
```

```
Selection.InsertVCControl( 6, "12cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Sorted = TRUE
For iCount = 1 To 9
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
10 - iCount
iNewIndex =
Activewindow.Controls(sCtrlName).ControlObject.NewIndex
Activewindow.Controls(sCtrlName).ControlObject.ItemData(
iNewIndex ) = iCount
Next iCount
Selection.Escape
Selection.Insert sHeader_1
Selection.InsertPara
Selection.Insert sHeader_2
Selection.InsertPara
Selection.Insert sHeader_3
Selection.InsertPara
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
sValue =
Activewindow.Controls(sCtrlName).ControlObject.List(iCount)
sItemData =
Activewindow.Controls(sCtrlName).ControlObject.ItemData(iCount)
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sValue  & " : "
Selection.Insert sItemData
Selection.InsertPara
Next iCount
End Sub
```

OutputHeight-Eigenschaft

Bestimmt die beschreibbare Höhe eines Preview-Kontrollfeldes (Höhe des Controls abzueglich Rahmen).

SYNTAX:
Object.OutputHeight

PARAMETER:
(nur Lesen)

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

OutputWidth-Eigenschaft

Bestimmt die beschreibbare Breite eines Preview-Kontrollfeldes (Breite des Controls abzüglich Rahmen).

SYNTAX:
Object.OutputWidth

PARAMETER:
(nur Lesen)

FÜR ELEMENT:

Preview

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

Parent-Eigenschaft

Gibt das übergeordnete Objekt (den Dialog) eines Kontrollfeldes zurueck.

SYNTAX:
Object.Parent

RÜCKGABE:
Variant

PARAMETER:
(nur Lesen)

FÜR ELEMENT:

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

PassWordChar

Legt fest, ob in einem Textfeld Platzhalter für die vom Benutzer eingegebenen Zeichen angezeigt werden. Mit dieser Eigenschaft lassen sich Textfelder erstellen, die zur Eingabe von Kennwörtern dienen.

Wird im Eigenschaftenfenster eines markierten Textfeldes (TextBox) in dem Feld PASSWORDCHAR ein beliebiges Zeichen eingetragen, so wird im Textfeld für jedes vom Benutzer eingegebene Zeichen ein Sternchen (*) als Platzhalter angezeigt, so daß der vom Benutzer eingetragene Text nicht sichtbar ist. Die TEXT-Eigenschaft selbst, die die vom Benutzer eingegebene Zeichenfol-

ge als Wert enthält, bleibt dabei unverändert. Um diesen Text wieder sichtbar zu machen, setzen Sie die PASSWORDCHAR-Eigenschaft wieder auf eine Null-Zeichenfolge ("") zurück (Voreinstellung).

Diese Eigenschaft gilt nur für einzeilige Textfelder, d. h. die MULTILINE-Eigenschaft muß den Wert False aufweisen, ansonsten hat PASSWORDCHAR keine Wirkung.

Die PASSWORDCHAR-Eigenschaft läßt sich nur zur Entwurfszeit definieren, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Text As String = Object.PasswordChar

PARAMETER:
Text: Beliebige Zeichenfolge, mit der bestimmt wird, ob in einem Textfeld eingegebener Text sichtbar gemacht oder mittels Platzhalter dargestellt werden soll. Bei einem Leerstring werden keine Platzhalter verwendet.

FÜR ELEMENT:

TextBox

Beispiel:

```
Sub ExamplePasswordChar
Const sTextboxText = "Paßwort"
Const sMessage_1 = "Diese Textbox kann zur Paßworteingabe
verwendet werden"
Const sMessage_2 = "Folgender Text wurde eingeben : "
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4.5cm", "4cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Pass-
wordChar = "*"
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & sTextboxText
Selection.InsertPara
End Sub
```

Printable

Bestimmt, ob beim Drucken eines Dokuments das eingefügte Kontrollfeld mit ausgedruckt wird.

Die PRINTABLE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Printable [= True | False]

PARAMETER:
True: Das Kontrollfeld wird mit ausgedruckt.

False: Das Kontrollfeld wird nicht mit ausgedruckt.

Beispiel:

```
Sub ExamplePrintable
Const sCombobox1 = "Diese Combobox wird gedruckt"
Const sCombobox2 = "Diese Combobox wird nicht gedruckt"
Const sMessage_1 = "Es wird nur eine von beiden Commandbutton
gedruckt."
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 2, "6cm", "4cm", "6cm", "2cm" )
REM Commandbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.caption = sCom-
bobox1
Activewindow.Controls(sCtrlName).ControlObject.Printable = TRUE
Selection.Escape
Activewindow.InsertVCControl( 2, "6cm", "7cm", "6cm", "2cm" )
REM Commandbutton
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCom-
bobox2
Activewindow.Controls(sCtrlName).ControlObject.Printable =
FALSE
Selection.Escape
ActiveWindow.PrintDefault
End Sub
```

ReadOnly

Bestimmt, ob der Inhalt eines Textfeldes geändert werden kann. Wenn die ReadOnly-Eigenschaft den Wert True aufweist, kann zwar der Cursor in das Textfeld gesetzt werden, Änderungen am Inhalt können aber nicht vorgenommen werden.

Die ReadOnly-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

Syntax:
Object.ReadOnly [= True | False]

Rückgabe:
Boolean

Parameter:
True: Das Textfeld ist schreibgeschützt.

False: Der Inhalt des Textfeldes ist editierbar (Voreinstellung).

Für Element:

TextBox

Beispiel:

```
Sub ExampleReadonly
Const sTextboxText = "Schreibgeschützt"
Const sMessage_1 = "Diese Textbox ist schreibgeschützt"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4cm", "6cm", "2cm" )
REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.Readonly = TRUE
Selection.Escape
End Sub
```

Refresh-Methode

Stellt das angegebene Element neu dar und löst ein Paint-Ereignis aus.

Syntax:
Element.Refresh

PARAMETER:
FÜR ELEMENT:

Dialog, TextBox, Label, CheckBox, ComboBox, FrameControl, HScrollBar, ListBox, VScrollBar, Preview, CommandButton, OptionButton, SpinButton.

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

RemoveItem-Methode

Entfernt einen Eintrag aus einem Listenfeld oder einem Kombinationslistenfeld.

SYNTAX:
Object.RemoveItem Index

PARAMETER:
Index: Numerischer Ausdruck, der den Index des zu löschenden Eintrages angibt. Dem ersten Listeneintrag ist der Wert 0 zugewiesen.

FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleRemoveItem
Const sItem = "Eintrag "
Const sMessage_1 = "Es wird eine Listbox mit Daten gefüllt und
dann wieder entfernt."
Const sMessage_2 = "Es wurden zehn Eintrage in die Listbox
eingetragen..."
Const sMessage_3 = "...und wieder entfernt."
Dim sCtrlName As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Wait( 100 )
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Selection.Escape
Selection.Insert sMessage_2
Selection.InsertPara
Wait( 500 )
For iCount = 1 To 10
```

```
Activewindow.Controls(sCtrlName).ControlObject.RemoveItem(0)
Next iCount
Selection.Escape
Selection.Insert sMessage_3
Selection.InsertPara
End Sub
```

ScrollBars

Legt fest, ob an einem Textfeld (TextBox) die horizontale und die vertikale Bildlaufleiste angezeigt wird. Diese Eigenschaft steht nur für Textfelder zur Verfügung, die als Dialogelemente erstellt werden.

Horizontale und vertikale Bildlaufleiste werden stets dann angezeigt, wenn dem Textfeld die MULTILINE-Eigenschaft = True zugeordnet wurde. Mit der SCROLLBARS-Eigenschaft läßt sich die Ansicht der Bildlaufleisten ein- und ausschalten. Trägt die MULTILINE-Eigenschaft den Wert False, hat die SCROLLBARS-Eigenschaft keine Wirkung.

Die SCROLLBARS-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.ScrollBars [= True | False]

PARAMETER:
True: Bildlaufleisten werden angezeigt.

False: Bildlaufleisten werden nicht angezeigt.

FÜR ELEMENT:

TextBox

Beispiel:

```
Kein Beispiel.
```

Selected-Eigenschaft

Bestimmt den Status der Markierung eines Eintrages in einem Listenfeld oder ermittelt diesen Status.

SYNTAX:
Object.Selected (Index) [=Bool]

PARAMETER:
Index: Numerischer Ausdruck, der den Eintrag kennzeichnet, dessen Markierungszustand bestimmt oder ermittelt werden soll.

[Bool]: Booleanscher Ausdruck, der den Status der Markierung angibt:

True: Der entsprechende Eintrag im Listenfeld ist markiert.

False (Default): Der entsprechende Eintrag im Listenfeld ist nicht markiert.

FÜR ELEMENT:

ListBox

Beispiel:

```
Sub ExampleSelected
Const sItem = "Eintrag "
Const sMessage_1 = "Es wird der markierte Eintrag ermittelt."
Const sMessage_2 = "Folgender Eintrag ist markiert : "
Dim sCtrlName As String, sSelect As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
Activewindow.Controls(sCtrlName).ControlObject.ListIndex = 2
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
If
Activewindow.Controls(sCtrlName).ControlObject.Selected(iCount)
= TRUE Then
sSelect = Activewindow.Controls(sCtrlName).ControlObject.List(
iCount )
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & sSelect
End If
Next iCount
Selection.Escape
End Sub
```

SelLength-Eigenschaft

Bestimmt oder ermittelt die Länge einer markierten Zeichenkette in einem Textfenster oder dem Eingabefeld eines Kombinationslistenfeldes.

SYNTAX:
Object.SelLength[=Length As Integer]

RÜCKGABE:
Integer

PARAMETER:
Length: Integer-Ausdruck, dem die Länge der markierten Zeichenkette zugewiesen ist.

FÜR ELEMENT:

TextBox, ComboBox

Beispiel:

```
Sub ExampleSelTextbox
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es wird bei der Textbox auf dem Dokument
etwas markiert"
Const sMessage_2 = "Beginn der Selektion : "
Const sMessage_3 = "Länge der Selektion : "
Const sMessage_4 = "Der selektierte Text : "
Dim sCtrlName As String, sSelText As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.SelStart = 4
Activewindow.Controls(sCtrlName).ControlObject.SelLength = 5
sSelText =
Activewindow.Controls(sCtrlName).ControlObject.SelText
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & "5"
Selection.InsertPara
Selection.Insert sMessage_3 & "5"
Selection.InsertPara
Selection.Insert sMessage_4 & sSelText
Selection.InsertPara
End Sub
```

SelStart-Eigenschaft

Ermittelt oder bestimmt die Position des ersten Zeichens einer markierten Zeichenkette in einem Textfenster oder dem Eingabefeld eines Kombinationslistenfeldes. Ist keine Zeichenkette markiert, ermittelt die SELSTART-Eigenschaft die Position der Schreibmarke.

SYNTAX:
Object.SelStart[=Position As Integer]

RÜCKGABE:
Integer

PARAMETER:
Position: Integer-Ausdruck, der die Position des ersten Zeichens einer markierten Zeichenkette oder der Schreibmarke aufnimmt.

FÜR ELEMENT:

TextBox, ComboBox

Beispiel:

```
Sub ExampleSelTextbox
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es wird bei der Textbox auf dem Dokument
etwas markiert"
Const sMessage_2 = "Beginn der Selektion : "
Const sMessage_3 = "Länge der Selektion : "
Const sMessage_4 = "Der selektierte Text : "
Dim sCtrlName As String, sSelText As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.SelStart = 4
Activewindow.Controls(sCtrlName).ControlObject.SelLength = 5
sSelText =
Activewindow.Controls(sCtrlName).ControlObject.SelText
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & "5"
Selection.InsertPara
Selection.Insert sMessage_3 & "5"
Selection.InsertPara
Selection.Insert sMessage_4 & sSelText
Selection.InsertPara
End Sub
```

SelText-Eigenschaft

Liefert die Zeichenfolge, die in in einem Textfenster oder dem Eingabefeld eines Kombinationslistenfeldes markiert wurde, oder ersetzt den markierten Text durch einen Zeichenkettenausdruck.

SYNTAX:
Object.SelText[= Text As String]

RÜCKGABE:
String

PARAMETER:
Text: Zeichenkettenausdruck, dem die markierte Zeichenfolge zugewiesen ist.

FÜR ELEMENT:

TextBox, ComboBox

Beispiel:

```
Sub ExampleSelTextbox
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Es wird bei der Textbox auf dem Dokument
etwas markiert"
Const sMessage_2 = "Beginn der Selektion : "
Const sMessage_3 = "Länge der Selektion : "
Const sMessage_4 = "Der selektierte Text : "
Dim sCtrlName As String, sSelText As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "12cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.SelStart = 4
Activewindow.Controls(sCtrlName).ControlObject.SelLength = 5
sSelText =
Activewindow.Controls(sCtrlName).ControlObject.SelText
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & "5"
Selection.InsertPara
Selection.Insert sMessage_3 & "5"
Selection.InsertPara
Selection.Insert sMessage_4 & sSelText
Selection.InsertPara
End Sub
```

SetFocus-Methode

Setzt den Fokus für eines der unter ELEMENTE aufgeführten Elemente. Nachdem ein Element fokussiert wurde, beziehen sich alle Eingaben vom Anwender auf dieses Element.

Der Fokus kann nur für solche Elemente gesetzt werden, die sichtbar sind ('Visible'-Eigenschaft ist auf True gesetzt) und auf die zugegriffen werden kann ('Enabled'-Eigenschaft ist auf True gesetzt). Die SETFOCUS-Methode kann daher nicht in der Unterroutine eines LOAD-Ereignisses für einen Dialog

verwendet werden, da der Dialog erst dann angezeigt wird, nachdem das Ereignis ausgeführt wurde.

SYNTAX:
Object.SetFocus

PARAMETER:
FÜR ELEMENT:

ComboBox, ListBox, CheckBox, Dialog, TextBox, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Show-Methode

Startet einen Dialog und zeigt ihn auf dem Bildschirm an.

SYNTAX:
Object.Show

PARAMETER:
FÜR ELEMENT:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

SmallChange

Bestimmt die Größe der Änderung in der VALUE-Eigenschaft einer Bildlaufleiste, wenn der Benutzer auf einen Bildlaufpfeil klickt. Der Wertebereich kann im Bereich von 1 bis 32.767 liegen.

Die SMALLCHANGE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.SmallChange [= Number As Integer]

PARAMETER:
Number: Integer-Ausdruck, der die Größe der Änderung der VALUE-Eigenschaft angibt. Standardmäßig hat Number den Wert 1.

FÜR ELEMENT:

HScrollBar, VScrollBar

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

Sorted

Legt fest, ob automatisch eine alphabetische Sortierung der Einträge in Listen- oder Kombinationsfeldern erfolgt. Die SORTED-Eigenschaft gilt für die Objekte ListBox und ComboBox. Sie kann nur zur Entwurfszeit festgelegt werden, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Switch = Object.Sorted

PARAMETER:
True: Die Einträge werden alphabetisch sortiert.

False: Es erfolgt keine automatische Sortierung.

FÜR ELEMENT:

ListBox, ComboBox

Beispiel:

```
Sub ExampleList
Const sItem = "Eintrag "
Const sHeader_1 = "Folgende Einträge sind der der Listbox."
Const sHeader_2 = "Die Einträge in der Listbox während der
Eingabe sortiert."
Const sHeader_3 = "Eintrag : ItemData "
Dim sCtrlName As String, sValue As String
Dim sItemData as String
Dim iCount As Integer, iNewIndex As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "12cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Sorted = TRUE
For iCount = 1 To 9
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
10 - iCount
iNewIndex =
Activewindow.Controls(sCtrlName).ControlObject.NewIndex
Activewindow.Controls(sCtrlName).ControlObject.ItemData(
iNewIndex ) = iCount
Next iCount
Selection.Escape
Selection.Insert sHeader_1
Selection.InsertPara
Selection.Insert sHeader_2
Selection.InsertPara
Selection.Insert sHeader_3
Selection.InsertPara
For iCount = 0 To Activewindow.Controls(sCtrlName).ControlOb-
ject.ListCount - 1
sValue =
Activewindow.Controls(sCtrlName).ControlObject.List(iCount)
```

```
sItemData =
Activewindow.Controls(sCtrlName).ControlObject.ItemData(iCount)
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sValue  & " : "
Selection.Insert sItemData
Selection.InsertPara
Next iCount
End Sub
```

Step

Definiert einen Index, der bei einem Seitenwechsel die Ansicht der Dialogelemente bestimmt. Mit dem durch STEP definierten Index kann man Dialogelemente derart gruppieren, daß nur solche mit gleichem Index und Index Null (0) sichtbar sind.

Diese Eigenschaft wird im StarOffice z. B. beim Autopiloten verwendet, um die Dialogelemente von einer Seite zur nächsten auszutauschen, indem Step um eins erhöht wird. Welcher Index gerade aktuell ist, wird durch die CURRENTSTEP-Eigenschaft des Dialoges festgelegt.

Die STEP-Eigenschaft läßt sich nur zur Entwurfszeit festlegen, zur Laufzeit läßt sie sich abfragen.

SYNTAX:
Number As Integer = Object.Step

PARAMETER:
Value: Integer-Ausdruck, der den Index des Kontrollfeldes angibt.

FÜR ELEMENT:

CheckBox, ComboBox, TextBox, Label, FrameControl, HScrollBar, ListBox, Preview, CommandButton, OptionButton, SpinButton, VScrollBar

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Style

Legt den Typ eines Kombinationsfeldes (ComboBox) und das Verhalten der zugehörigen Listenfeld-Komponente fest. Die STYLE-Eigenschaft kann zur Laufzeit nicht geändert werden, sie läßt sich nur abfragen.

SYNTAX:
Value As Integer = Object.Style

PARAMETER:
Value: Integer-Ausdruck, der die Art des Objekts ComboBox definiert. Folgende Werte können angenommen werden:

0 - Es können nur Einträge ausgewählt werden, die in der Liste vorhanden sind. Beim Aufruf ist die Liste nicht aufgeklappt. (DropDown List)

1 - Es können Einträge ausgewählt werden, die in der Liste vorhanden sind. Zusätzlich kann vom Anwender ein weiterer Eintrag vorgenommen werden. Dieser Eintrag wird allerdings nicht mit in die Liste aufgenommen. Beim Aufruf ist die Liste aufgeklappt. (Simple Combo)

2 - Es können Einträge ausgewählt werden, die in der Liste vorhanden sind. Zusätzlich kann vom Anwender ein weiterer Eintrag vorgenommen werden. Dieser Eintrag wird allerdings nicht mit in die Liste aufgenommen. Beim Aufruf ist die Liste geschlossen. (DropDown Combo)

FÜR ELEMENT:

ComboBox

Beispiel:

```
Kein Beispiel
```

TabIndex

Legt die Reihenfolge fest, in der die Kontrollfelder in einem Dialog fokussiert werden, wenn die Tabulatortaste betätigt wird. Sind in einem Dialog mehrere Elemente angeordnet, bewegt ein Druck auf die Tabulatortaste den Fokus auf das jeweils nächste Element. Die Reihenfolge, in der dies geschieht, wird mit der TABINDEX-Eigenschaft bestimmt. Der Index des ersten Elementes, das den Fokus erhalten soll, ist 0.

Bei der Erstellung eines Dialoges wird den Elementen, die diesem Dialog hinzugefügt werden, automatisch ein Index in der Reihenfolge ihrer Anordnung zugewiesen. Wird die TABINDEX-Eigenschaft eines Elementes geändert, werden die Indizes der restlichen Elemente automatisch aktualisiert. Den Elementen, die nicht sichtbar sind, auf die nicht zugegriffen werden kann oder die nicht fokussiert werden können, wird ebenfalls ein TabIndex zugewiesen. Diese Elemente werden jedoch bei einer Betätigung der Tabulatortaste übersprungen.

Der Wert der TABINDEX-Eigenschaft kann nur zur Entwurfszeit im Eigenschaftenfenster festgelegt werden, zur Laufzeit läßt sich der Wert in StarBasic abfragen.

SYNTAX:
Value As Integer = Object.TabIndex

PARAMETER:
Value: Integer-Ausdruck, der den Tabulator-Index angibt, mit dem das Kontrollfeld in dem Dialog belegt ist.

FÜR ELEMENT:

ComboBox, ListBox, CheckBox, FrameControl, TextBox, Label, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton.

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

TabStop

Legt fest, ob ein Objekt in einem Dialog mit der Tabulatortaste angesprungen werden kann. Diese Eigenschaft läßt sich nur zur Entwurfszeit festlegen, zur Laufzeit kann sie abgefragt werden.

SYNTAX:
Switch As Boolean = Object.TabStop

RÜCKGABE:
Boolean

PARAMETER:
True: Das Kontrollfeld kann mit der Tabulatortaste fokussiert werden.

False: Mit Betätigung der Tabulatortaste wird das Kontrollfeld bei der Fokussierung übersprungen. Der TabIndex wird beibehalten.

FÜR ELEMENT:

CheckBox, ComboBox, ListBox, TextBox, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton.

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Tag

Die Eigenschaft wird vom StarBasic zur Zeit nicht ausgewertet. Die TAG-Eigenschaft kann vom Anwender frei belegt und ausgewertet werden.

Mit Hilfe dieser Eigenschaft können zusätzliche Daten zu einem Kontrollfeld oder einem Dialog in Form einer Zeichenkette abgelegt werden. Verwenden Sie diese Eigenschaft, um einem Element bestimmte Informationen zuzuweisen, die es beispielsweise identifizieren, oder die Ihnen als Anmerkung zu diesem Element dienen.

Die TAG-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Tag [= AddData As String]

PARAMETER:
AddData: Zeichenkette, die zusätzliche Informationen zum Element enthält.

FÜR ELEMENT:

Dialog, CheckBox, ListBox, ComboBox, FrameControl, TextBox, Label, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTag
Const sTagText = "Text im Tag"
Const sTextboxText = "Tag-Beispiel"
Const sMessage_1 = "Diese Textbox hat einen Eintrag in der Tag-
Eigenschaft"
Const sMessage_2 = "In der Tag-Eigenschaften ist folgendes
eingetragen : "
Dim sCtrlName As String, sTag As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "6cm", "4.5cm", "3cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
Activewindow.Controls(sCtrlName).ControlObject.Tag = sTagText
sTag = Activewindow.Controls(sCtrlName).ControlObject.Tag
Selection.Escape
Selection.Insert sMessage_2 & sTag
Selection.InsertPara
End Sub
```

Target

Die Eigenschaft gibt für das Kontrollfeld URLButton den Zielrahmen an, in dem der angesprungene URL angezeigt werden soll. Die URL-Adresse legen Sie mit der URL-Eigenschaft fest.

Die TARGET-Eigenschaft läßt sich nur zur Entwurfszeit festlegen, zur Laufzeit kann sie abgefragt werden.

SYNTAX:
Frame As String = Object.Target

PARAMETER:
Frame: Zeichenfolge, die angibt, in welchem Frame das Folgedokument geladen werden soll. Folgende Möglichkeiten gibt es:

_self: Das Folgedokument wird im selben Frame aufgebaut.

_blank: Das Folgedokument wird in einer eigenen View, also in einem leeren neuen Frame, aufgebaut.

_parent: Das Folgedokument wird in einem Parent, also einem übergeordneten Frame, aufgebaut; gibt es keinen Parent, wird das Dokument im selben Frame aufgebaut.

_top: Das Folgedokument wird in einem Top-Window, also im höchsten Frame der Hierarchie, aufgebaut; ist der Ausgangs-Frame bereits ein Top-Window, wird das Dokument im selben Frame aufgebaut.

Beispiel:

```
Sub ExampleTarget
Const sCaption = "Lädt die Star Division Homepage"
Const sMessage_1 = "Diese URL-Button lädt die Stardivision
Homepage in einem neuem Dokument"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 25, "6cm", "4.5cm", "5cm", "1cm"
)
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion
Activewindow.Controls(sCtrlName).ControlObject.Target =
"_blank"
Activewindow.Controls(sCtrlName).ControlObject.URL =
"http://www.stardivision.de"
Selection.Escape
End Sub
```

Text-Eigenschaft

Über diese Eigenschaft kann ein Textfeld (TextBox) mit einem Inhalt vorbelegt werden. Bei Kombinationsfeldern (ComboBox) wird der Bearbeitungsbereich mit einem Text belegt, wenn die STYLE-Eigenschaft hier nicht den Wert 0 hat.

Die TEXT-Eigenschaft kann zur Entwurfzeit festgelegt oder während der Laufzeit gesetzt und abgefragt werden.

SYNTAX:
Object.Text [= Text As String]

PARAMETER:
Text: Zeichenfolge, die den Eintrag der TextBox oder der ComboBox angibt.

FÜR ELEMENT:

ComboBox, ListBox, TextBox

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
```

```
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

Top

Bestimmt den Abstand zwischen dem oberen Rand des Kontrollfelds und dem oberen Rand des Dokumentes oder des Dialoges.

Die TOP-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Top [= Distance As Integer]

PARAMETER:
Distance: Integer-Ausdruck, der den Wert des Abstands in Twips festlegt.

FÜR ELEMENT:

Dialog, CheckBox, ListBox, ComboBox, FrameControl, TextBox, Label, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTextboxPosition
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Diese Textbox hat folgende Koordinaten"
Const sMessage_2 = "Von links : "
Const sMessage_3 = "Von oben : "
Const sMessage_4 = "Breite : "
Const sMessage_5 = "Höhe : "
Dim sCtrlName As String
Dim lLeft As Long, lTop As Long, lWidth As Long, lHeight As
Long
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
```

```
Activewindow.InsertVCControl( 8, "10cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
lLeft = Activewindow.Controls(sCtrlName).ControlObject.Left
lTop = Activewindow.Controls(sCtrlName).ControlObject.Top
lWidth = Activewindow.Controls(sCtrlName).ControlObject.Width
lHeight = Activewindow.Controls(sCtrlName).ControlObject.Height
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & lLeft & " Twips"
Selection.InsertPara
Selection.Insert sMessage_3 & lTop & " Twips"
Selection.InsertPara
Selection.Insert sMessage_4 & lWidth & " Twips"
Selection.InsertPara
Selection.Insert sMessage_5 & lHeight & " Twips"
Selection.InsertPara
End Sub
```

TopIndex-Eigenschaft

Bestimmt oder ermittelt den an erster Stelle in einem Listenfeld angeordneten Eintrag. Die Voreinstellung dieser Eigenschaft ist 0, für den ersten Eintrag.

SYNTAX:
Object.TopIndex[=Index As Integer]

PARAMETER:
Index: Index des an erster Stelle des Listenfeldes angeordneten Eintrages.

FÜR ELEMENT:

ListBox

Beispiel:

```
Sub ExampleTopIndex
Const sItem = "Eintrag "
Const sMessage = "TopIndex liefert folgendes zurück : "
Dim sCtrlName As String
Dim iCount As Integer, iTopIndex As Integer
Documents.Add( "sWriter4" )
Selection.InsertVCControl( 6, "5cm", "5cm", "3cm", "3cm" )
sCtrlName = ActiveWindow.ObjectName
For iCount = 1 To 10
Activewindow.Controls(sCtrlName).ControlObject.AddItem sItem &
iCount
Next iCount
iTopIndex =
Activewindow.Controls(sCtrlName).ControlObject.TopIndex
```

```
Selection.Escape
Selection.insert sMessage & iTopIndex
End Sub
```

UnLoad-Methode

Entfernt den Dialog aus dem Speicher.

SYNTAX:
Object.UnLoad

PARAMETER:
FÜR ELEMENT:

Dialog

Beispiel:

```
REM Ein Beispiel befindet sich bei der DrawBox-Funktion für das
Preview-Kontrollfeld
```

URL

Bestimmt die URL-Adresse, die bei Betätigung des Kontrollfeldes URLButton angesprungen wird.

Die URL-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.URL [= Address As String]

PARAMETER:
Address: Zeichenfolge, die den URL angibt, der mit der Schaltfläche aktiviert wird.

Beispiel:

```
Sub ExampleTarget
Const sCaption = "Lädt die Star Division Homepage"
Const sMessage_1 = "Diese URL-Button lädt die Stardivision
Homepage in einem neuem Dokument"
Dim sCtrlName As String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 25, "6cm", "4.5cm", "5cm", "1cm"
)
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Caption = sCap-
tion
Activewindow.Controls(sCtrlName).ControlObject.Target =
```

```
"_blank"
Activewindow.Controls(sCtrlName).ControlObject.URL =
"http://www.stardivision.de"
Selection.Escape
End Sub
```

UseMnemonic

Legt fest, ob das Zeichen "&" in der als CAPTION-Eigenschaft festgelegten Bezeichnung als Accelerator Key verwendet wird. Diese Eigenschaft steht für ein Label-Objekt im Dialog zur Verfügung.

StarBasic erlaubt es, sogenannte Acceleratoren zu definieren, mit denen man dialogunterstützte Programme auch über die Tastatur bedienen kann. Es gilt dabei die Konvention, daß sich die Dialogelemente durch (Alt) und den in der Beschriftung des Dialogelementes unterstrichenen Buchstaben ansteuern lassen. Einige Dialogelemente besitzen keine CAPTION-Eigenschaft zur Beschriftung. Man behilft sich in diesem Fall mit einem Label-Dialogelement, das das Element beschriftet und ebenfalls einen Accelerator definieren kann. In diesem Fall muß das Element, das über die Tastatur angesprochen werden soll, in der TabIndex-Reihenfolge direkt dem Label-Element folgen. Der Buchstabe, der die Bedienung auslösen soll, muß im Beschriftungstext mit vorangestelltem "&"-Zeichen vorhanden sein. In diesem Fall ersetzt StarBasic das "&"-Zeichen automatisch durch einen Unterstrich, der sich unterhalb des jeweiligen Buchstabens befindet und ihn dadurch als Tastaturkürzel kennzeichnet.

Die USEMNEMONIC-Eigenschaft bestimmt nun, ob die Tastenkürzel-Definition durch das "&"-Zeichen in einem Label-Dialogelement möglich ist (True) oder nicht (False). Wird sie auf False gesetzt, werden "&"-Zeichen wie ganz normale Zeichen behandelt.

Die USEMNEMONIC-Eigenschaft wird zur Entwurfszeit festgelegt, sie kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.UseMnemonic [= True | False]

PARAMETER:
True: Das Zeichen "&" in der CAPTION-Eigenschaft wird als Accelerator Key verwendet.

False: Das Zeichen "&" in der CAPTION-Eigenschaft wird zur Bezeichnung verwendet.

FÜR ELEMENT:

Label

Beispiel:

```
REM Ein Beispiel befindet sich bei der Visible-Eigenschaft für
Kontrollfelder.
```

Value-Eigenschaft

Bestimmt den momentanen Zustand des Kontrollfeld-Objektes.

Bei Options- und Markierfeldern bestimmt die VALUE-Eigenschaft, ob die Kontrollfelder aktiviert wurden.

Bei Bildlaufleisten und Drehfeldern gibt die VALUE-Eigenschaft die aktuelle Position des Bildlauffeldes auf der Bildlaufleiste bzw. die aktuelle Position des Drehfeldes an, wobei die möglichen Werte in dem durch die MIN- und die MAX-Eigenschaft definierten Intervall liegen.

Die VALUE-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Value [= True | False] for OptionButton, CheckBoxObject.Value [= Position As Integer] for HScrollBar, VScrollBar, SpinButton

PARAMETER:
True: Das Objekt wurde aktiviert.

False: Das Objekt wurde nicht aktiviert.

Position: Integer-Ausdruck, der die aktuelle Position innerhalb des Objekts angibt.

FÜR ELEMENT:

HScrollBar, VScrollBar, SpinButton

Beispiel:

```
REM Damit dieses Makro funktioniert, muß ein neues Textdokument
angelegt werden.
REM Kopieren Sie dieses Makro in das REM Modul1REM  des Doku-
mentes.
Sub ExampleScrollbar
Const sFileName = "\Control.sdw"
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Const sMessage = "Trägt die Position der Scrollbar in die
Textbox ein"
Dim sTmpControlName As String, sDocName As String
sDocName = Application.PathSettings.Work & sFileName
ActiveDocument.SaveAs sDocName
Selection.Insert sMessage
```

```
Activewindow.InsertVCControl( 8, "4.5cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sTextbox
Selection.Escape
Activewindow.InsertVCControl( 10, "6cm", "5cm", "0.5cm", "5cm"
) REM Scrollbar
sTmpControlName = ActiveWindow.ObjectName
ActiveWindow.Controls(sTmpControlName).ControlObject.Name =
sScrollbar
ActiveWindow.Controls(sScrollbar).ControlObject.Min = 1
ActiveWindow.Controls(sScrollbar).ControlObject.Max = 100
ActiveWindow.Controls(sScrollbar).ControlObject.SmallChange = 1
ActiveWindow.Controls(sScrollbar).ControlObject.LargeChange =
10
REM Befindet sich die Procedure nicht in der Bibliothek REM
StandardREM  oder im
REM Modul1, dann ändern Sie bitte SetHandler.
Selection.SetHandler(2, "Standard.Modul1.ScrollbarValue")
Selection.Escape
End Sub
Sub ScrollbarValue
Const sTextbox = "NeueTextbox"
Const sScrollbar = "NeueScrollbar"
Dim iScrollValue As Integer
iScrollValue = Activewindow.Controls(sScrollbar).ControlOb-
ject.Value
ActiveWindow.Controls(sTextbox).ControlObject.Text = iScroll-
Value
Selection.Escape
End Sub
```

Visible

Bestimmt, ob das Objekt zur Laufzeit sichtbar ist oder nicht. Die VISIBLE-Eigenschaft läßt sich nur zur Entwurfszeit im Eigenschaftenfenster des Kontrollfeldes festlegen, zur Laufzeit läßt sie sich in StarBasic abfragen.

SYNTAX:
Switch = Object.Visible

PARAMETER:
True: Das Objekt ist zur Laufzeit sichtbar.

False: Das Objekt ist zur Laufzeit nicht sichtbar.

FÜR ELEMENT:

Dialog, CheckBox, ListBox, ComboBox, FrameControl, TextBox, Label, Preview, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
REM Für dieses Beispiel muß ein Dialog angelegt werden, mit
einem Label-Element und drei CommandButtons.
REM Die Namen dieser Elemente sind, falls keine Standard-Namen
verwendet, werden in den
REM Const-Bereich einzutragen.
REM Außerdem muß an den CommandButton die MakroREM s HANDLER-
BUTTON+INDEX für das Ereignis REM OnClickREM  konfiguriert
werden.
Const sDialog="Dialog1"  REM Standard-Namen nach der Erstellung
der Elemente
Const sButton="CommandButton" REM Namen der Knoepfe auf dem
Dialog ohne Nummer
Const sLabel ="Label1" REM Ein Labelelement
Const iHeight=2400
Const iWidth =8000
Dim oDlgObj(10) As Object
Sub ExampleDialog
on Error Goto ErrorHandler
Dim iInfo As Integer, iCount As Integer, bInfo As Boolean
oDlgObj(0) = FindObject( sDialog )
oDlgObj(1) = FindPropertyObject(oDlgObj(0), sButton & "1" ) REM
Durchnumerierte Buttons
oDlgObj(2) = FindPropertyObject(oDlgObj(0), sButton & "2" )
oDlgObj(3) = FindPropertyObject(oDlgObj(0), sButton & "3" )
oDlgObj(4) = FindPropertyObject(oDlgObj(0), sLabel )
oDlgObj(0).Caption = oDlgObj(4).Parent.Name
oDlgObj(0).Load
oDlgObj(0).Width = iWidth
oDlgObj(0).Height = iHeight
oDlgObj(0).Caption = "Das ist Ihr Dialog "
oDlgObj(0).CurrentStep = 0 REM Gemeinsame Dialogfelder
anzeigen.
bInfo = oDlgObj(0).Auto3D
For iCount = 1 To 3
REM ist eine Cancel-Button definiert.
If oDlgObj( iCount ).Cancel = TRUE Then
oDlgObj( iCount ).Caption = "&Abbrechen" REM den Cancel-Button-
Text definieren
oDlgObj( iCount ).Step
ElseIf oDlgObj( iCount ).Default = TRUE Then
oDlgObj( iCount ).Caption = "&OK"   REM den Default-Button-Text
definieren
Else
oDlgObj( iCount ).Caption = "Knopf &" & iCount REM Die Stan-
dardknöpfe definieren
oDlgObj( iCount ).SetFocus
End If
iInfo = oDlgObj( iCount ).TabStop
iInfo = oDlgObj( iCount ).TabIndex
With  oDlgObj( iCount )
iInfo = .Step  REM Sollte immer 0 sein.
.Move( 300 * iCount  + (iCount-1) * 2300 , 1400,  2100, 600 )
```

```
.HelpContextID = 0
REM Hilfetexte definieren ...Kann man im Tip über dem Element
lesen
.Helptext="Das ist ein aktiver Hilfetext " & iCount
End With
Next
With oDlgObj(4)
.Move( 200  , 200,  iWidth - 400 , 1000 )
.UseMnemonic = FALSE
.Caption = "Adler & Söhne" REM der Text mit REM &REM   für das
Label-Feld
.FontSize = 36
.FontName = "Times New Roman"
.FontBold = TRUE
.FontItalic = TRUE
.Border = TRUE
.BackStyle = 1
End With
oDlgObj(0).Show
Exit Sub
ErrorHandler:
iInfo = Err
s = Error
MsgBox "Die Namen und Typen ihrer Kontroll-Elemente sind wahr-
scheinlich nicht korrekt." & Chr(13) &_
"Bitte kontrollieren Sie Ihren erstellten Dialog." ,16,"Es ist
ein Fehler aufgetreten"
End Sub
REM der Klick-Handler für CommandButton1
Sub HandlerButton1
oDlgObj(4).Visible = NOT oDlgObj(4).Visible
End Sub
REM der Klick-Handler für CommandButton2
Sub HandlerButton2
static iClick As Integer
iClick = iClick + 1
oDlgObj(4).Caption = "Klick " & iClick & " auf Knopf 2"
End Sub
REM der Klick-Handler für CommandButton3
Sub HandlerButton3
static iClick As Integer
iClick = iClick + 1
oDlgObj(4).Caption = "Klick " & iClick & " auf Knopf 3"
End Sub
```

Width

Bestimmt die Breite eines Kontrollfeld-Objektes.

Die WIDTH-Eigenschaft wird zur Entwurfszeit festgelegt und kann zur Laufzeit abgefragt und geändert werden.

SYNTAX:
Object.Width [= Width As Integer]

PARAMETER:
Width: Integer-Ausdruck, der die Breite des Objekts in Twips angibt.

FÜR ELEMENT:

Dialog, CheckBox, ListBox, ComboBox, FrameControl, TextBox, Label, Preview, Picture, HScrollBar, VScrollBar, CommandButton, OptionButton, SpinButton

Beispiel:

```
Sub ExampleTextboxPosition
Const sTextboxText = "Textboxinhalt"
Const sMessage_1 = "Diese Textbox hat folgende Koordinaten"
Const sMessage_2 = "Von links : "
Const sMessage_3 = "Von oben : "
Const sMessage_4 = "Breite : "
Const sMessage_5 = "Höhe : "
Dim sCtrlName As String
Dim lLeft As Long, lTop As Long, lWidth As Long, lHeight As
Long
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Activewindow.InsertVCControl( 8, "10cm", "4cm", "2cm", "0.5cm"
) REM Textbox
sCtrlName = ActiveWindow.ObjectName
Activewindow.Controls(sCtrlName).ControlObject.Text = sTextbox-
Text
lLeft = Activewindow.Controls(sCtrlName).ControlObject.Left
lTop = Activewindow.Controls(sCtrlName).ControlObject.Top
lWidth = Activewindow.Controls(sCtrlName).ControlObject.Width
lHeight = Activewindow.Controls(sCtrlName).ControlObject.Height
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2 & lLeft & " Twips"
Selection.InsertPara
Selection.Insert sMessage_3 & lTop & " Twips"
Selection.InsertPara
Selection.Insert sMessage_4 & lWidth & " Twips"
Selection.InsertPara
```

```
Selection.Insert sMessage_5 & lHeight & " Twips"
Selection.InsertPara
End Sub
```

WordWrap-Eigenschaft

Bestimmt, ob Text umbrochen wird oder nicht. Diese Eigenschaft steht für die Kontrollfelder Label und TextBox zur Verfügung. Bei einem Label-Objekt bestimmt WordWrap, ob der durch die Caption-Eigenschaft festgelegte Text zur Bezeichnung umbrochen wird, bei einem TextBox-Objekt bestimmt WordWrap, ob nach Betätigung der Eingabetaste ein Umbruch stattfindet.

Die WordWrap-Eigenschaft hat für Textfelder nur dann eine Wirkung, wenn MultiLine = True ist.

Je nachdem, ob die Kontrollfelder in einen Dialog oder in ein Dokument eingefügt wurden, gibt es unterschiedliche Einstellmöglichkeiten. Bei Kontrollfeldern im Dialog regelt ein Boolescher Parameter, ob ein Umbruch stattfindet. Hier kann die WordWrap-Eigenschaft zur Entwurfszeit definiert und zur Laufzeit gesetzt oder abgefragt werden. Bei Kontrollfeldern im Dokument kann zur Entwurfszeit zwischen drei Optionen gewählt werden, die festlegen, ob ein automatischer Umbruch stattfinden soll und ob es sich hierbei um einen harten oder einen weichen Umbruch handelt. Diese Optionen lassen sich nur zur Entwurfszeit festlegen und sind vom StarBasic aus nicht steuerbar.

Folgende Modi stehen zur Verfügung, wenn es sich um Kontrollfelder im Dokument handelt:

Off: Kein automatischer Umbruch. Harte Umbrüche werden über die Text -Eigenschaft geregelt.

Hard: Automatischer Umbruch. Harte Umbrüche werden über die Text-Eigenschaft geregelt.

Soft: Automatischer Umbruch. Harte Umbrüche werden durch Leerzeichen ersetzt.

Auch der HTML-Filter unterstützt diese Optionen, und zwar als WRAP-Option des TEXTAREA-Tags. Beim Import werden neben OFF, HARD und SOFT auch noch PHYSICAL (wie HARD) und VIRTUAL (wie SOFT) unterstützt, da diese Tags von anderen Web Browsern unterstützt werden. Exportiert werden nur HARD und SOFT, da OFF der Default ist.

Syntax:
Object.WordWrap [= True | False]

PARAMETER:
Für Kontrollfelder im Dialog:

True: Der Text wird umbrochen, wenn dieser länger ist als das Feld selber.

False: Der Text wird abgeschnitten, wenn dieser länger ist als das Feld selbst (Voreinstellung).

FÜR ELEMENT:

Label

Weitere Informationen zu diesem Thema erhalten Sie unter Eigenschaften definieren.

Klassenbaum

Hier erhalten Sie eine Übersicht des Klassenbaumes für StarBasic.

Klasse SearchAttributes

... ist die Klasse in dem die Formate enthalten sind, die für das Suchen angewandt werden können.

Zugriff:
ActiveWindow.SearchAttributes
Windows(iCount AS Integer).SearchAttributes

Klasse ReplaceAttributes

... ist die Klasse, in dem die Formate enthalten sind, die beim Ersetzen angewandt werden.

Zugriff:
ActiveWindow.ReplaceAttributes
Windows(iCount AS Integer).ReplaceAttributes

Klasse ChartFont

... ist die Klasse für die Fonts innerhalb von Charts.

Zugriff:

Klasse Title

... ist die Klasse für die Titelzeile in einem Diagramm.

Zugriff:

Klasse Auditing

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse Pivot

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse TableFont

... ist die Klasse für die Fontattributierung innerhalb einer Tabellenzelle.

Zugriff:

Klasse TextPage

... enthält die Hintergrundfarbe einer Seite.

Zugriff:
ActiveWindow.Page

Klasse TextFont

... enthält Methoden und Eigenschaften für Zeichenattribute.

Zugriff:
Selection.Font

Klasse HyphenZone

... enthält die Trennregeln des Absatzes. HyphenZone ist in der Klasse Paragraph enthalten.

Zugriff:
Selection.Paragraph.HyphenZone

Klasse DropCap

... enthält die Einstellungen für Initialen. Sie ist in der Klasse Paragraph enthalten.

Zugriff:
ActiveWindow.Paragraph.DropCap
Selection.Paragraph.DropCap

Klasse Paragraph

... ist die Klasse für Absatzattribute. Sie ist in der Klasse TEXT enthalten. Diese Klasse kann mit ActiveWindow.Paragraph.<Befehl> angesprochen werden.

Zugriff:
Selection.Paragraph

Klasse Shadow

... enthält die Eigenschaften und Methoden für den Schatten von Objekten.

Zugriff:
Selection.Shadow

Klasse PrintSettings

... enthält die Eigenschaften und Methoden für die Druckeinstellungen.

Zugriff:
ActiveModule.PrintSettings
StarWriter.PrintSettings

Klasse ViewSettings

... enthält die Eigenschaften und Methoden für die Ansichtsoptionen.

Zugriff:
ActiveModule.ViewSettings
StarWriter.ViewSettings

Klasse Address

... enthält die Adresse und einige Zusatzdaten des Benutzers. Diese Daten wurden bei der Installation erfragt und können über EXTRAS - OPTIONEN - ALLGEMEIN - BENUTZERDATEN geändert werden. Die einzige Instanz dieser Klasse über die gleichnamige Eigenschaft der Application-Instanz ansprechbar:

Zugriff:
Application.Address

Klasse Interior

... enthält Eigenschaften und Methoden für den Hintergrund. Sie ist für alle Objekte relevant, denen Hintergrundfarben zugeordnet werden können.

Zugriff:
Selection.Interior

Klasse SearchSettings

... enthält die Einstellungen, die für die Suche angewandt werden.

Zugriff:
ActiveWindow.SearchSettings
Windows(iCount AS Integer).SearchSettings

Klasse Border

... beschreibt die Umrandung von diversen Objekten in Dokumenten, wie z. B. Tabellen, Rahmen oder auch Absätze. Nicht alle Eigenschaften sind in jedem Anwendungsfall verfügbar. Instanzen dieser Klasse sind beispielsweise wie folgt ansprechbar:

Zugriff:
Selection.Border
ActiveWindow.Selection.Border

Klasse BorderLine

... ist eine Hilfsklasse für BORDER und beschreibt eine einzelne Umrandungslinie (Dicke, Farbe, Stil). Instanzen dieser Klasse sind beispielsweise wie folgt ansprechbar:

Zugriff:
Selection.Border.TopLine
ActiveWindow.Selection.Border.LeftLine

Klasse TabStops

... beschreibt die Tabulatoren für einen Absatz, ein Textobjekt oder eine Zelle. Instanzen können auch mit "dim aTabs as TabStops" angelegt werden. Alternativ sind sie wie folgt zu erreichen:

Zugriff:
Selection.TabStops
Windows(iCount AS Integer).Selection.TabStops

Klasse Module

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse StarChart

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse StarCalc

... ist die Klasse für das StarCalc-Modul.

Zugriff:
ActiveModule
StarCalc

Klasse StarImpress

... ist die Klasse für das StarImpress-Modul.

Zugriff:
ActiveModule
StarDraw

Klasse StarImage

... ist die Klasse für das StarImage-Modul.

Zugriff:
ActiveModule
StarImage

Klasse StarMath

... ist die Klasse für das StarMath-Modul.

Zugriff:

Klasse StarWriter

... ist die Klasse für das StarWriter-Modul.

Zugriff:
ActiveModule
StarWriter

Klasse Application

... ist die Klasse für alle globalen Eigenschaften und Methoden im StarOffice.

Zugriff:
Application

Klasse Desktop

... enthält globale Funktionen und Eigenschaften, die i. d. R. Auswirkung auf alle Module haben, oder die so allgemein sind, daß sie an kein bestimmtes Objekt gehören. Die einzigste Instanz dieser Klasse ist wie folgt ansprechbar:

Zugriff:
Application
ActiveWindow.Application
ActiveDocument.Application

Klasse View

... ist die Basisklasse für alle Dokumentansichten.

Zugriff:
ActiveWindow

Klasse ChartView

... ist die Klasse für die Ansicht der StarChart Dokumente.

Zugriff:

Klasse TablePrintPreview

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse TableEditView

... ist die Klasse für die Ansicht eines Tabellendokumentes.

Zugriff:
ActiveWindow

Klasse OutlineView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse SlideView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse DrawView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse ImpressEditView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse FormulaView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse WebSourceView

... steht für die StarWriter-Quelltextansicht von HTML-Dokumenten.

Zugriff:
ActiveWindow
Windows(iCount AS Integer)

Klasse TextPrintPreview

... enthält die Eigenschaften und Methoden für die Seitenansicht.

Zugriff:
ActiveWindow

Klasse BaseTextEditView

... ist die Basisklasse für ein StarWriter-Window.

Zugriff:
ActiveWindow
Windows(iCount AS Integer)

Klasse WebEditView

... steht für das StarWriter-Window bei HTML-Dokumenten. Diese Klasse ist von der Klasse BASETEXTEDITVIEW abgeleitet.

Zugriff:
ActiveWindow
Windows(iCount AS Integer)

Klasse TextEditView

... steht für das StarWriter-Window bei Textdokumenten. Diese Klasse ist von der Klasse BASETEXTEDITVIEW abgeleitet.

Zugriff:
ActiveWindow
Windows(iCount AS Integer)

Klasse MessageView

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse FrameSetView

... ist eine spezielle View-Subklasse, die FrameSetDocument-Instanzen darstellen kann. Instanzen dieser Klasse können wie jede View beispielsweise wie folgt angesprochen werden:

Zugriff:
ActiveWindow.View
Application.ActiveWindow
Windows(iCount AS Integer)
Windows("docname:1")
ActiveWindow.TopWindow

Klasse Window

... beschreibt ein Fenster, in dem sich eine View auf Dokumente befinden kann. Window ist eine Basisklasse von der es keine direkten Instanzen gibt; diese gibt es erst für Subklassen wie TopWindow, FrameWindow oder PlugInWindow. Instanzen dieser Subklassen können beispielsweise wie folgt angesprochen werden:

Zugriff:
ActiveWindow
Application.ActiveWindow
ActiveWindow.TopWindow
ActiveWIndow.ParentFrame

Klasse IPWindow

... ist ein spezielles Fenster für InPlace-Editing von Embedded-Objekten in fremden (oder eigenen) Views. In ihm können theoretisch alle Dokument-Views dargestellt werden, wie in einem normalen Window auch, i. d. R. sind sie aber in ihrer Funktionalität eingeschränkt. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow
Application.ActiveWindow
Windows("docname:1")

Klasse BrowseWindow

... ist die Klasse für Windows, deren Inhalt austauschbar ist.

Zugriff:
ActiveWindow

Klasse FrameWindow

... beschreibt ein Unterfenster für einzelne Frames in einem FrameSet. In ihm können alle Dokument-Views dargestellt werden, wie in einem normalen Window auch. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow
Application.ActiveWindow
Windows("framename")

Klasse TopWindow

... ist die Klasse für ein Window, das keinen Parent mehr hat.

Zugriff:
ActiveWindow

Klasse TaskWindow

... ist die Klasse für ein Window, das in der Taskleiste angezeigt wird.

Zugriff:
ActiveWindow

Klasse PlugInWindow

... ist ein spezielles Fenster für PlugIn-Darstellung von Objekten in fremden (oder eigenen) Views. In ihm können theoretisch alle Dokument-Views dargestellt werden, wie in einem normalen Window auch, i. d. R. sind sie aber in ihrer Funktionalität eingeschränkt. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow
Application.ActiveWindow
Windows("docname:1")

Klasse Document

... ist die Basisklasse für alle Dokumente.

Zugriff:
ActiveDocument

Klasse ExplorerDocument

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse OfficeDocument

... ist die Basisklasse für Officedokumente.

Zugriff:
ActiveDocument

Klasse ChartDocument

... ist die Dokument-Klasse für StarChart-Dokumente (Diagramme). Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:

Klasse TableDocument

... ist die Dokument-Klasse für StarCalc-Dokumente (Tabellendokumente). Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow.TopWindow.Document
Documents(iCount AS Integer)
Documents("docname")

Der DOCNAME muß Casesensitive angegeben werden.

Klasse DrawDocument

... ist die Dokument-Klasse für StarImpress-Dokumente. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow.TopWindow.Document
Documents(iCount AS Integer)
Documents("docname")

Der DOCNAME muß Casesensitive angegeben werden.

Klasse ImpressDocument

... ist die Dokument-Klasse für StarImpress-Dokumente. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow.TopWindow.Document
Documents(iCount AS Integer)
Documents("docname")

Der DOCNAME muß Casesensitive angegeben werden.

Klasse ImageDocument

... ist die Dokument-Klasse für StarImage-Dokumente. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow.TopWindow.Document
Documents(iCount AS Integer)
Documents("docname")

Der DOCNAME muß Casesensitive angegeben werden.

Klasse FormulaDocument

... ist die Dokument-Klasse für StarMath-Dokumente (Formeln). Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

```
Zugriff:
ActiveWindow.TopWindow.Document
Documents( iCount AS Integer )
Documents( "docname" )
```

Der DOCNAME muß Casesensitive angegeben werden.

Klasse BaseTextDocument

... ist die Basisklasse für Textdokumente. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

```
Zugriff:
ActiveWindow.TopWindow.Document
Documents( iCount AS Integer )
Documents( "docname" )
```

Der DOCNAME muß Casesensitive angegeben werden.

Klasse WebDocument

... ist von der Klasse BASETEXTDOCUMENT abgeleitet und stellt die Dokument-Klasse für HTML-Dokumente dar. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

```
Zugriff:
ActiveWindow.TopWindow.Document
Documents( iCount AS Integer )
Documents( "docname" )
```

Der DOCNAME muß Casesensitive angegeben werden.

Klasse TextDocument

... ist von der Klasse BASETEXTDOCUMENT abgeleitet und stellt die Dokument-Klasse für Textdokumente dar. Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveDocument
Documents(iCount AS Integer)

Klasse MessageDocument

... ist eine spezielle Dokument-Klasse für Nachrichten-Dokumente (Mail und News). Instanzen sind beispielsweise auf folgenden Wegen ansprechbar:

Zugriff:
ActiveWindow.TopWindow.Document
Documents(iCount AS Integer)
Documents("docname")

Der DOCNAME muß Casesensitive angegeben werden.

Klasse FrameSetDocument

... ist eine spezielle Document-Subklasse, die verschachtelte FrameSets und Frames enthalten kann. Instanzen dieser Klasse können wie folgt angesprochen werden:

Zugriff:
ActiveDocument
Application.ActiveDocument
Documents(iCount AS Integer)
Documents("docname")
ActiveWindow.Document

Der DOCNAME muß Casesensitive angegeben werden.

Klasse DocumentInfo

... enthält Zusatzinformationen zu Dokumenten, wie z. B. den Autor, einen Beschreibungstext etc. Über die gleichnamige Eigenschaft an jedem Office-Dokument kann auf eine solche Instanz zugegriffen werden. Außerdem können solche Instanzen mit "dim aDocInfo as DocumentInfo" angelegt und mit aDocInfo.Load dateiname.ext geladen sowie mit aDocInfo.Save gespeichert werden. Dies bietet einen schnelleren Zugriff auf die DocumentInfo, als das gesamte Dokument zu laden um dann nur dessen DocumentInfo zu verwenden.

Zugriff:
ActiveDocument.DocumentInfo
Documents(3).DocumentInfo

Klasse SaveSettings

... enthält Einstellungen zum Verhalten beim Laden und Speichern von Dokumenten; diese können über Extras - Optionen - Allgemein - Speichern geändert werden. Von dieser Klasse gibt es genau eine Instanz, welche über die Eigenschaft SaveSettings von der Application-Instanz zugänglich ist:

Zugriff:
Application.SaveSettings
Desktop.SaveSettings

Klasse PathSettings

... enthält die Pfadeinstellungen, die bei der Installation vordefiniert und über Extras - Optionen - Allgemein - Pfade geändert werden können. Von dieser Klasse gibt es genau eine Instanz, welche über die Eigenschaft PathSettings von der Application-Instanz zugänglich ist:

Zugriff:
Application.PathSettings
Desktop.PathSettings

Klasse Progress

... über die Klasse Progress kann Zugriff auf den Fortschrittsbalken in der Statuszeile genommen werden.

Zugriff:
dim aProgress as new Progress

Klasse Selection

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse TableText

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse TableDraw

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse RangeSelection

... stellt einen Zellbereich dar. (Selection gibt z. B. ein Range-Objekt zurück). An der RangeSelection hängen die meisten StarCalc-Basic-Funktionen.

Zugriff:
Range("A1")
Range("A1:B5")
Range("Tabelle1.A1")
ActiveDocument.Range("A1")

Klasse TableOLEObject

... ist die Klasse für ein OLE-Objekt im Tabellendokument. Mit der Eigenschaft TableOLEObject erhält man daraus ein Objekt mit den Methoden des eigentlichen OLE-Objekts (z. B. Chart-Funktionen).

Zugriff:
SelectedObject

Klasse TableSelection

... ist die Klasse für eine Tabelle in einem Tabellendokument.

Zugriff:
ActiveTable

Klasse WebListInText

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse WebListInTable

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse WebTextInTable

... an dieser Klasse stehen keine Eigenschaften und Methoden zur Verfügung, die über StarBasic angewendet werden können.

Zugriff:

Klasse ListInText

... ist von der Klasse TextList und von der Klasse Text abgeleitet. Sie ist gültig, wenn sich die Selektion innerhalb einer Aufzählung oder Numerierung befindet.

Zugriff:
Selection

Klasse ListInTable

... ist von der Klasse TextList und von der Klasse Table abgeleitet. Sie ist gültig, wenn sich die Selektion innerhalb einer Aufzählung oder Numerierung innerhalb einer Tabelle befindet.

Zugriff:
Selection

Klasse TextInTable

... ist von der Klasse Text und der Klasse TextTable abgeleitet. Sie ist gültig, wenn die Selektion innerhalb einer Tabelle steht.

Zugriff:
Selection

Klasse BaseTextOLEObject

... ist die Basisklasse für OLEObjecte in Textdokumenten.

Zugriff:

Klasse WebOLEObject

... ist von der Klasse BaseTextOleObject abgeleitet und enthält zusätzlich einige Methoden und Eigenschaften, die für OLE-Objekte in HTML-Dokumenten verfügbar sind.

Zugriff:
Selection

Klasse TextOLEObject

... ist von der Klasse BASETEXTOLEOBJECT abgeleitet und enthält zusätzlich einige Methoden und Eigenschaften, die für OLE-Objekte verfügbar sind.

Zugriff:
Selection

Klasse BaseTextGraphic

... ist die Basisklasse für Grafiken in Textdokumenten.

Zugriff:
Selection

Klasse WebGraphic

... ist von der Klasse BASETEXTGRAPHIC abgeleitet und enthält zusätzlich einige Methoden und Eigenschaften, die für Grafiken in HTML-Dokumenten verfügbar sind.

Zugriff:
Selection

Klasse TextGraphic

... ist von der Klasse BASETEXTGRAPHIC abgeleitet und enthält zusätzlich einige Methoden und Eigenschaften, die für Grafiken in Textdokumenten verfügbar sind.

Zugriff:
Selection

Klasse BaseTextTable

... ist die Basisklasse für Tabellen in Textdokumenten. Sie enthält die Eigenschaften und Methoden für die Tabellen in Textdokumenten.

Zugriff:
Selection

Klasse WebTable

... ist abgeleitet von der Klasse BASETEXTTABLE. Sie enthält zusätzliche Eigenschaften und Methoden für die Tabellen in HTML-Dokumenten.

Zugriff:
Selection

Klasse TextTable

... ist abgeleitet von der Klasse BASETEXTTABLE. Sie enthält zusätzliche Eigenschaften und Methoden für die Tabellen in Textdokumenten.

Zugriff:
Selection

Klasse BaseTextFrame

... ist die Basisklasse für Textrahmen. Sie enthält alle Eigenschaften und Methoden, die für Textrahmen verfügbar sind.

Zugriff:
Selection

Klasse WebFrame

... ist abgeleitet von der Klasse BASETEXTFRAME und enthält zusätzliche Eigenschaften und Methoden, die für Textrahmen in HTML-Dokumenten verfügbar sind.

Zugriff:
Selection

Klasse TextFrame

... ist abgeleitet von der Klasse BASETEXTFRAME und enthält zusätzliche Eigenschaften und Methoden, die für Textrahmen in Textdokumenten verfügbar sind.

Zugriff:
Selection

Klasse BaseTextList

... ist die Basisklasse für Aufzählungen oder Numerierungen. Sie ist gültig, wenn sich die Selektion innerhalb einer Aufzählung oder Numerierung befindet.

Zugriff:
Selection

Klasse WebList

... ist abgeleitet von der Klasse BaseTextList und enthält zusätzliche Eigenschaften und Methoden für Numerierungen und Aufzählungen in HTML-Dokumenten.

Zugriff:
Selection

Klasse TextList

... ist abgeleitet von der Klasse BaseTextList und enthält zusätzliche Eigenschaften und Methoden für Numerierungen und Aufzählungen in Textdokumenten.

Zugriff:
Selection

Klasse TextDrawText

... enthält Methoden und Eigenschaften für Texte in Zeichenobjekten.

Zugriff:
Selection

Klasse BaseTextDrawControl

... ist die Basisklasse für Kontrollfelder im Dokument.

Zugriff:
Selection

Klasse WebDrawControl

... ist die Klasse für Kontrollfelder in einem HTML-Dokument.

Zugriff:
Selection

Klasse TextDrawControl

... ist die Klasse für Kontrollfelder in einem Textdokument.

Zugriff:
Selection

Klasse BaseTextDrawBase

... ist die Basisklasse für alle Zeichenobjekte innerhalb von Textdokumenten.

Zugriff:
Selection

Klasse WebDrawBase

... ist abgeleitet von der Klasse BaseTextDrawBase und enthält zusätzliche Eigenschaften und Methoden für Zeichenobjekte innerhalb von HTML-Dokumenten.

Zugriff:
Selection

Klasse TextDrawBase

... ist abgeleitet von der Klasse BaseTextDrawBase und enthält zusätzliche Eigenschaften und Methoden für Zeichenobjekte innerhalb von Textdokumenten.

Zugriff:
Selection

Klasse TextDraw

... enthält die Methoden und Eigenschaften für Zeichenobjekte.

Zugriff:
Selection

Klasse BaseText

... ist die Basisklasse für Text.

Zugriff:
Selection

Klasse WebText

... ist abgeleitet von der Klasse BaseText und enthält zusätzliche Eigenschaften und Methoden für den Text in HTML-Dokumenten.

Zugriff:
Selection

Klasse Text

... ist abgeleitet von der Klasse BaseText und enthält zusätzliche Eigenschaften und Methoden für den Text in Textdokumenten.

Zugriff:
Selection

Klasse BaseTextSelection

... ist die Basisklasse für alle Selektionen in Textdokumenten. Eigenschaften und Methoden dieser Klasse sind an allen Selectionen verfügbar.

Zugriff:
Selection

Klasse WebSelection

... ist die Klasse für alle Selektionen in HTML-Dokumenten.

Zugriff:
Selection

Klasse TextSelection

... ist die Klasse für alle Selektionen in Textdokumenten.

Zugriff:
Selection

Klasse Collection

... ist die Klasse für die im StarOffice vorhandenen Collections.

Zugriff:

Klasse Data

... ist die Klasse für die Collections der StarChart Dokumente.

Zugriff:

Klasse Tables

... ist die Klasse für die Collections der StarCalc Tabellen.

Zugriff:

Klasse GlobalContents

... ist die Klasse für die Collection der Globaldokumente.

Zugriff:

Klasse Controls

... ist die Collection für Kontrollfelder. Erlaubt sind Remove(Index AS Integer), Remove(Name AS String), Count() und Controls(Index AS Integer).Befehl und Controls(Name AS String).Befehl. "Befehl" ist dabei ein gültiger Ausdruck im Objekt DrawCtrl. Der Index beginnt bei eins.

Zugriff:
ActiveWindow.Controls
Windows(iCount AS Integer).Controls

Klasse Graphics

... ist die Collection für Grafiken. Erlaubt sind Remove(Index AS Integer), Remove(Name AS String), Count() und Graphics(Index AS Integer).Befehl und Graphics.(Name AS String).Befehl. "Befehl" ist dabei ein gültiger Ausdruck am Objekt Graphic. Der Index beginnt bei eins.

Zugriff:
ActiveWindow.Graphics
Windows(iCount AS Integer).Graphics

Klasse Frames

... ist die Collection für Textrahmen. Erlaubt sind Remove(Index AS Integer), Remove(Name AS String), Count() und Frames(Index AS Integer).Befehl und Frames.(Name AS String).Befehl. "Befehl" ist dabei ein gültiger Ausdruck an der Klasse Frame. Der Index beginnt bei eins.

Zugriff:
ActiveWindow.Frames
Windows(ICount AS Integer).Frames

Klasse TextTables

... ist die Collection für Tabellen in Textdokumenten. Erlaubt sind Remove(Index AS Integer), Remove(Name AS String), Count() und Tables(Index AS Integer).Befehl und Tables(Name AS String).Befehl. Befehl ist dabei ein gültiger Ausdruck im Objekt Table.

Zugriff:
ActiveWindow.Tables
Windows(iCount As Integer).Tables

Klasse Strings

... ist die frei verwendbare COLLECTION von Zeichenketten. Kann alternativ auch mit "dim aStrings as Strings" angelegt werden.

Klasse OLEObjects

... ist die Collection für OLE-Objekte. Erlaubt sind Remove(Index AS Integer), Remove(Name As String), Count() und OLEObjects(Index AS Integer).Befehl und OLEObjects.(Name AS String).Befehl. "Befehl" ist dabei ein gültiger Ausdruck am OLEObject. Der Index beginnt bei eins.

Zugriff:
ActiveWindow.OLEObjects
Windows(iCount As Integer).OLEObjects

Klasse Windows

... ist die COLLECTION aller zur Zeit geöffneten Windows. Von dieser Klasse gibt es genau eine Instanz, die über die gleichnamige globale Variable und die ebenfalls gleichnamige Eigenschaft der Application zugänglich ist.

Zugriff:
Windows
Application.Windows
Desktop.Windows

Klasse Documents

... ist die COLLECTION aller zur Zeit geöffneten Dokumente. Von dieser Klasse gibt es genau eine Instanz, die über die gleichnamige globale Variable und die ebenfalls gleichnamige Eigenschaft der Application zugänglich ist.

Zugriff:
Documents
Application.Documents
Desktop.Documents

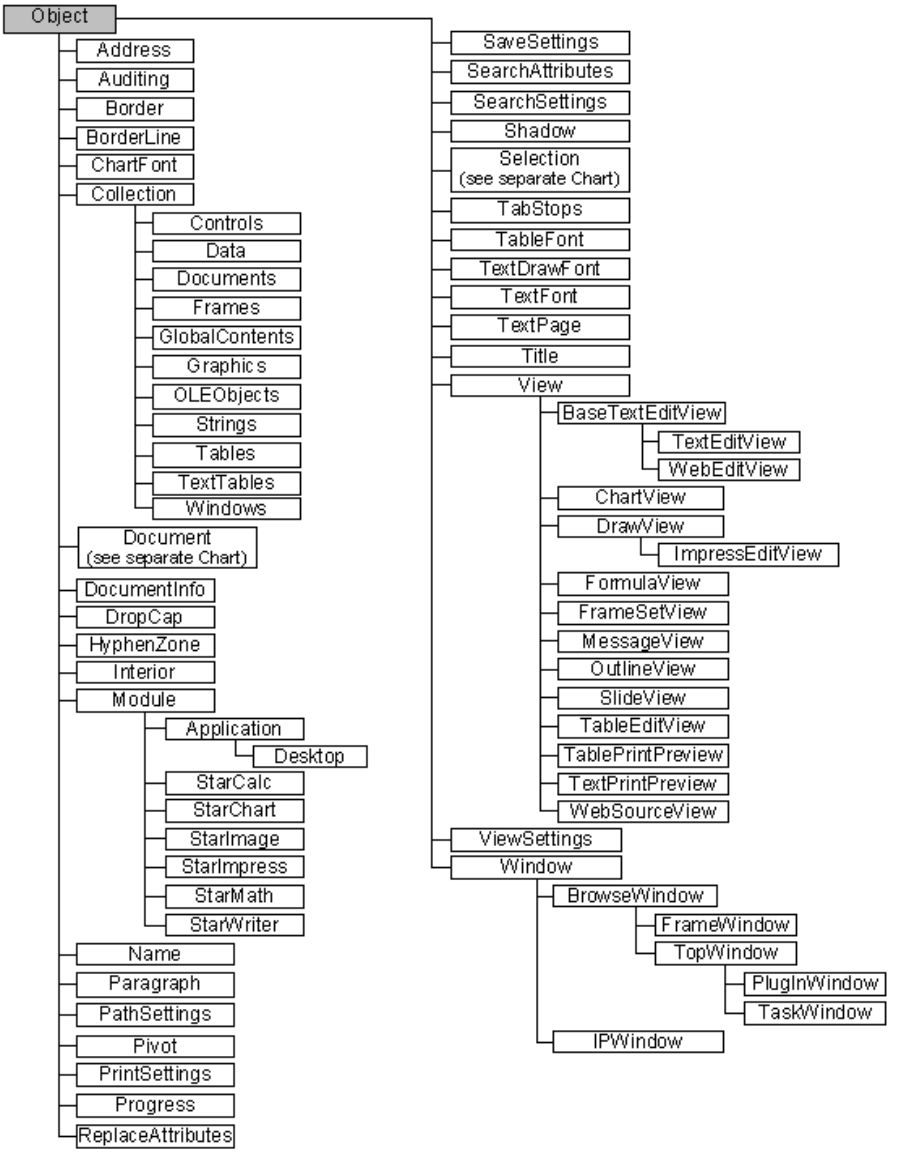
Ableitungsdiagramm

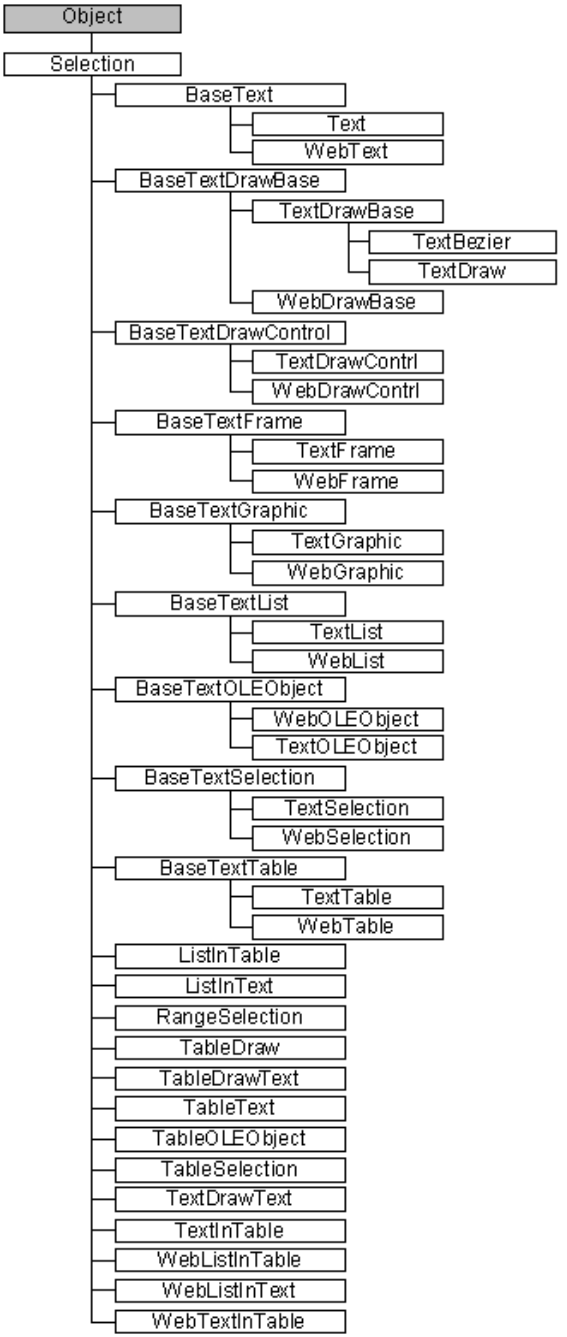
Hier finden Sie einen Überblick über die Ableitungshierarchie der Klassen im StarBasic.

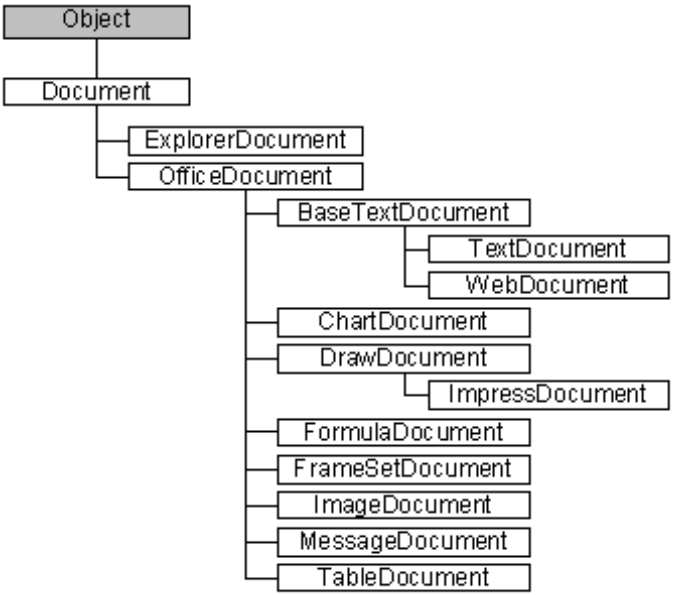
Um eine Eigenschaft oder Methode einer Klasse aufzurufen, muß zuerst ein Zugriffsweg auf ein Objekt dieser Klasse gefunden werden, z.B. eine Eigenschaft einer anderen Klasse, die ein Objekt der gesuchten Klasse enthält.

Für alle übergeordneten Klassen existieren Eigenschaften, die auf Objekte dieser Klassen verweisen. So verweist zum Beispiel die ActiveDocument-Eigenschaft auf ein Objekt der Klasse Document. Es können somit alle Eigenschaften und Methoden, die an der Klasse Document vorhanden sind an dem Objekt aufgerufen werden, das die ActiveDocument-Eigenschaft zurückliefert. Je nachdem, um welches Objekt es sich handelt, können die Methoden und Eigenschaften der betreffenden Subklasse, z. B. TextDocument verwendet werden.

Ist eine Klasse untergeordnet, so muß, um eine Eigenschaft oder Methode eines Objektes dieser Klasse anzusprechen, eine Eigenschaft in einer übergeordneten Klasse gesucht werden, die einen Verweis auf ein Objekt der Vaterklasse beinhaltet. So kann zum Beispiel eine Methode die an der Klasse TEXTLIST vorhanden ist, über eine Eigenschaft angesprochen werden, die auf ein Objekt der Klasse SELECTION verweist.


Object
Address
Auditing
Border
BorderLine
ChartFont
Collection
Controls
Data
Documents
Frames
GlobalContents
Graphics
OLEObjects
Strings
Tables
TextTables
Windows
Document
(see separate Chart)
DocumentInfo
DropCap
HyphenZone
Interior
Module
Application
Desktop
StarCalc
StarChart
StarImage
StarImpress
StarMath
StarWriter
Name
Paragraph
PathSettings
Pivot
PrintSettings
Progress
ReplaceAttributes
SaveSettings
SearchAttributes
SearchSettings
Shadow
Selection
(see separate Chart)
TabStops
TableFont
TextDrawFont
TextFont
TextPage
Title
View
BaseTextEditView
TextEditView
WebEditView
ChartView
DrawView
ImpressEditView
FormulaView
FrameSetView
MessageView
OutlineView
SlideView
TableEditView
TablePrintPreview
TextPrintPreview
WebSourceView
ViewSettings
Window
BrowseWindow
FrameWindow
TopWindow
PlugInWindow
TaskWindow
IPWindow


Object
Selection
BaseText
Text
WebText
BaseTextDrawBase
TextDrawBase
TextBezier
TextDraw
WebDrawBase
BaseTextDrawControl
TextDrawContrl
WebDrawContrl
BaseTextFrame
TextFrame
WebFrame
BaseTextGraphic
TextGraphic
WebGraphic
BaseTextList
TextList
WebList
BaseTextOLEObject
WebOLEObject
TextOLEObject
BaseTextSelection
TextSelection
WebSelection
BaseTextTable
TextTable
WebTable
ListInTable
ListInText
RangeSelection
TableDraw
TableDrawText
TableText
TableOLEObject
TableSelection
TextDrawText
TextInTable
WebListInTable
WebListInText
WebTextInTable


Object
Document
ExplorerDocument
OfficeDocument
BaseTextDocument
TextDocument
WebDocument
ChartDocument
DrawDocument
ImpressDocument
FormulaDocument
FrameSetDocument
ImageDocument
MessageDocument
TableDocument

Referenzhilfe für StarOffice

Hier erhalten Sie in alphabetischer Ordnung Informationen über sämtliche Eigenschaften und Methoden, die Ihnen für StarOffice-Anwendungen zur Verfügung stehen. Die einer speziellen Anwendung entnehmen Sie bitte der jeweiligen Hilfe zu Text-, und Tabellendokumenten.

Eigenschaften und Methoden mit "A"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "A" beginnen.

AboutDialog-Methode

Ruft das Informationsfenster ÜBER STAROFFICE auf.

KLASSE:
Application

SYNTAX:
AboutDialog()

BEISPIEL:

```
Sub ExampleAboutDialog
Application.AboutDialog()
End Sub
```

Activate-Methode

Erklärt ein geöffnetes Dokument zum aktiven Dokument. Aktiv bedeutet hierbei, daß ein Fenster des Dokuments im Vordergrund steht und Werkzeuge, Tasten etc. darauf wirken.

Der Rückgabewert ist das aktivierte Fenster.

KLASSE:
Document

SYNTAX:
Activate()

RÜCKGABE:
Object

Beispiel:

```
Sub ExampleActivate
Const sMessage = "Es wurde folgendes Dokument aktiviert: "
Dim oWindowObject As Object
Dim sDocName As String
Documents.Add( "sWriter4" )
sDocName = ActiveDocument.Name
Documents.Add( "sWriter4" )
oWindowObject = Documents( sDocName ).Activate
Print sMessage & oWindowObject.Name
End Sub
```

Activate-Methode

Erklärt ein geöffnetes Fenster zum aktiven Fenster. Aktiv bedeutet hierbei, daß das Fenster im Vordergrund steht und Werkzeuge, Tasten etc. darauf wirken.

Klasse:
BrowseWindow

Syntax:
Activate()

Rückgabe:
Object

Beispiel:

```
Sub ExampleWindowActivate
REM Durch die Fensterliste blaettern
Dim iCount As Integer
For iCount =1 to Application.Windows.Count
Application.Windows.Item(iCount).Activate
Wait 100
Next iCount
end sub
```

ActiveDocument-Eigenschaft

Enthält das Dokument, welches zur Zeit im aktiven Window dargestellt wird.

Klasse:
Application

Syntax:
ActiveDocument

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleApplication
Const sDoc="TestDoc.sdw"
Documents.Add("sWriter4")
ActiveDocument.SaveAs( Application.PathSettings.Work & "\" &
sDoc )
Selection.Insert("Name= "& ActiveDocument.Name)
Selection.InsertPara
Selection.Insert("Pfad= "& ActiveDocument.Path)
Selection.InsertPara
Selection.Insert("Pfad mit Name= " &ActiveDocument.FullName)
Selection.InsertPara
End Sub
```

ActiveFrameWindow-Eigenschaft

Bei Windows, die Frames enthalten, erhält man das Window-Objekt des aktiven Frames.

Enthält das Window keine Frames, wird das Window selbst zurückgegeben.

KLASSE:
Window

SYNTAX:
ActiveFrameWindow

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleTopWindow
Documents.Open("http://www.stardivision.de")
ActiveWindow.ActiveFrameWindow.Activate REM Den focusierten
Frame aktivieren.
MsgBox Iif(ActiveWindow.IsTopWindow,"Ja","Nein"),0,"Ist das
aktive Fenster ein Hauptfenster ?" REM ein einzelner Frame ist
nicht ein TopWindow
MsgBox ActiveWindow.ActiveFrameWindow.Name,0,"Name des aktiven
Frame" REM Der Aktive Frame
MsgBox ActiveWindow.TopWindow.Name,0,"Name des Hauptfensters"
REM Das Hauptfenster
End Sub
```

ActiveHelp-Eigenschaft

Bestimmt oder ermittelt, ob die Aktive Hilfe eingeblendet wird.

KLASSE:
Application

SYNTAX:
ActiveHelp

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Aktive Hilfe ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleActiveHelp
Application.ActiveHelp = TRUE  REM Schaltet die aktive Hilfe
ein.
End Sub
```

ActiveModule-Eigenschaft

Enthält das Modul des Dokuments, welches zur Zeit im aktiven Window dargestellt wird, zum Beispiel "StarWriter". Diese Eigenschaft kann alternativ dazu verwendet werden, auf einzelne Objekte der verschiedenen Module-Klassen zuzugreifen. So entspricht beispielsweise "Application.ActiveModule.Name" bei einem Textdokument "StarWriter.Name", bei einem Tabellendokument "StarCalc.Name" usw.

KLASSE:
Application

SYNTAX:
ActiveModule

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleActive
Dim oObj As Object, sText As String
Documents.Add("sWriter4")
   oObj = Application
   sText = sText & "Applikation: " & oObj.Name & Chr(13)

   oObj = Application.ActiveModule
```

```
   sText = sText & "Modul: " & oObj.Name & Chr(13)

   oObj = Application.ActiveDocument
   sText =sText & "Dokument: " & oObj.Name & Chr(13)

   oObj = Application.ActiveWindow
   sText = sText & "Fenster: " & oObj.Name
   MsgBox sText,0,"Alles über das Dokument"
End Sub
```

ActiveWindow-Eigenschaft

Enthält das zur Zeit aktive Window-Objekt.

KLASSE:
Application

SYNTAX:
ActiveWindow

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleActive
Dim oObj As Object, sText As String
Documents.Add("sWriter4")
   oObj = Application
   sText = sText & "Applikation: " & oObj.Name & Chr(13)

   oObj = Application.ActiveModule
   sText = sText & "Modul: " & oObj.Name & Chr(13)

   oObj = Application.ActiveDocument
   sText =sText & "Dokument: " & oObj.Name & Chr(13)

   oObj = Application.ActiveWindow
   sText = sText & "Fenster: " & oObj.Name
   MsgBox sText,0,"Alles über das Dokument"
End Sub
```

Add-Methode

Fügt ein neues Objekt in die Strings-Collection ein.

KLASSE:
Strings

SYNTAX:
Add(Entry As String)

RÜCKGABE:
String

PARAMETER:
Entry: Zeichenfolge, die eingefügt werden soll.

BEISPIEL:

```
Sub ExampleStrings
Const sFirst = "Erster"
Const sSecond = "Zweiter"
Const sMessage = "Folgender Eintrag ist in der Collection
vorhanden: "
Dim Example As New Strings
Dim iCount As Integer
Example.Add sFirst
Example.Add sSecond
For iCount = 1 To Example.Count
Print sMessage & Example.Item( iCount )
Next iCount
Example.Remove(2)
End Sub
```

Add-Methode

Fügt ein neues Objekt in die Documents-Collection ein.

Wenn Sie die Documents-Collection um ein Objekt erweitern, so erhalten Sie in der jeweiligen StarOffice-Anwendung ein neues Dokument. Wenn Sie die Add-Methode auf die Documents-Collection anwenden, können Sie daher zwei Parameter übergeben, die, wie beim Erstellen eines neuen Dokuments im Programm, die Angabe von Dokumentvorlagenbereich und -name erlauben.

Wollen Sie neue unbenannte Dokumente anlegen, sind für die Angabe der zugehörigen Vorlage jeweils spezielle Bezeichner erforderlich (siehe Beispiel). Geben Sie nur den ersten Parameter an, wird der übergebene String als Dokumenttyp interpretiert und ein entsprechend neues Dokument angelegt.

Wenn Sie keine Parameter übergeben, so bewirkt die Add-Methode standardmäßig das Erstellen eines neuen Textdokuments.

KLASSE:
Documents

SYNTAX:
Add(Region As String, Name As String)

RÜCKGABE:
Object

PARAMETER:
[Region] : Zeichenfolgenausdruck, der bestimmt, welchem Vorlagenbereich die Dokumentvorlage zugeordnet werden soll. Dieser Parameter ist optional; wenn Sie ihn nicht angeben, wird standardmäßig ein Textdokument erzeugt.

[Name]: Zeichenfolgenausdruck, der den Namen der Dokumentvorlage angibt. Wenn Sie keinen Namen übergeben, wird ein neues unbenanntes Dokument angelegt.

BEISPIEL:

```
Sub ExampleDocumentsAdd
REM Öffnet diverse neue Dokumente
Documents.Add("swriter4") REM Textdokument
Documents.Add("scalc4") REM Tabellendokument
Documents.Add("Message") REM Nachricht
Documents.Add("simpress4") REM Präsentation
Documents.Add("smath4") REM Formel
Documents.Add("simage4") REM Bild
Documents.Add("swriter4/web") REM Hypertext-Dokument
Documents.Add("schart4") REM Diagramm
Documents.Add("swriter4/GlobalDocument") REM Globaldokument
End Sub
```

AddBookmark-Methode

Speichert einen URL im aktuellen Bookmarkordner.

KLASSE:
TaskWindow

SYNTAX:
AddBookmark(URL As String, Title As String)

PARAMETER:
URL: Gibt die URL-Adresse an, die im Bookmarkordner abgelegt werden soll.

[Title]: Gibt den Titel des Bookmarks an, unter dem der URL abgespeichert werden soll. Wird dieser Parameter nicht übergeben, wird der URL selbst als Titel genommen.

BEISPIEL:

```
REM Der aktive Bookmarkordner muß bereits gewählt worden sein.
ActiveWindow.AddBookmark("http://www.stardiv.de","home")
Application.Beamer = FALSE REM Den Beamer wieder schließen
```

AddDatabaseConnection-Methode

Meldet eine neue Datenbank im StarOffice an.

Siehe auch: DeleteDatabaseConnection

KLASSE:
Desktop

SYNTAX:
AddDatabaseConnection(DBName As String, DBParams As String, Title As String)

PARAMETER:
DBName: In dieser Stringvariablen muß die Zeichenkette übergeben werden, die im Explorer erscheinen soll. Zu beachten ist allerdings, daß unbedingt das Explorer-Verzeichnis mit angegeben werden muß, da andernfalls der Eintrag nicht im Explorer angezeigt wird. Beispiel:

DBName = "C:\office40\Explorer\Firmenkunden"

DBParams: Zeichenfolge, die verschiedene Merkmale der zu importierenden Datenbank aufnimmt (siehe unten). Die einzelnen Eigenschaften werden durch Semikola getrennt übergeben. Sie sind jeweils optional, werden keine speziellen Parameter angeben, werden die Standardwerte genommen.

Titel: Über diesen Parameter wird der Eintrag im Explorer definiert. Dieser Parameter ist optional. Wird er nicht angegeben, wird der Name aus "DBName" verwendet.

DATENBANKTYP:

Es gibt drei Typen zu importierender Datenbanken:

Textdateien=TXT

Dbase-Dateien=DBF

FileMaker=FMK (Nur MAC)

und Datenbanken, die über ODBC-Treiber angesprochen werden,

ODBC-Dateien=ODBC

Das Schlüsselwort ist TYP, z.B. typ=TXT.

DATENBANKDATEI:

Als nächstes muß festgelegt werden, welches Verzeichnis die zu importierenden Daten enthält und wo sie sich innerhalb des Filesystems befindet. Angegeben wird "Laufwerk:kompletter Pfad".

Das Schlüsselwort ist DSN, z.B. dsn=D:\SAVE'Dateierweiterung'

Über das Schlüsselwort EXT kann eine Dateierweiterung angegeben werden, damit nur Dateien importiert werden, die die angegebene Dateiendung enthalten. Beispiel: ext=TXT

VERWENDETER ZEICHENSATZ BEI DER IMPORTIERUNG:

Mit dem Schlüsselwort CHARSET wird der beim Importieren verwendete Zeichensatz festgelegt. Zur Auswahl stehen folgende Zeichensätze:

ANSI; MAC; DOS; IBMPC; IBMPC_437; IBMPC_850; IBMPC_860; IBMPC_861; IBMPC_863; IBMPC_865; System

Systemeinstellungen sind:

für TXT, DBF (IBMPC)

für ODBC (SYSTEM)

für FMK (MAC)

Bei Datenbanken, die im gleichen System erstellt wurden, sollte mit dem Zeichensatz "System" gearbeitet werden.

Das Schlüsselwort ist CHARSET, z.B. charset=System.

DIE FOLGENDEN PARAMETER GELTEN NUR FÜR TEXT-DATENBANKEN!

Wird eine Text-Datenbank eingebunden, gibt der Parameter "fix" an, daß die erste Zeile der Datenbank die Dimension für die folgenden Datensätze festlegt.

Das Schlüsselwort ist FIX, z.B. fix=false.

KOPFZEILE:

Beim Import von Textdateien in Datenbankbearbeitungsmodule oder Datenbanken wird von den Applikationen angeboten, in der ersten Zeile einen Datensatz zu übergeben. Die Felder dieses Datensatzes enthalten dabei die Spaltennamen, die in die Datenbankanwendungen übernommen werden sollen. Im Parameterstring kann ein Vorhandensein dieser Kopfzeile über das Schlüsselwort "hdr" festgelegt werden. Als mögliche Parameter stehen hier "true" oder "false" zur Verfügung. Wird der Wert "false" gesetzt, so werden die Daten eingelesen und in Felder gelegt. Die einzelnen Spalten erhalten eine fortlaufende Numerierung in der Form C1, C2,...

Das Schlüsselwort ist HDR, z.B. hdr=false Voreinstellung ist FALSE.

FELDTRENNER:

Mit dem Schlüsselwort "del" wird festgelegt, durch welches ASCII-Zeichen die einzelnen Felder innerhalb eines Datensatzes getrennt sind. StarOffice bietet folgende Zeichen:

Doppelpunkt = 58

Komma = 44

Leerzeichen = 32

Tabulator = 9

Semikolon = 59

Das Schlüsselwort ist DEL, z.B. del=44.

TEXTTRENNER:

Die genaue Wirkungsweise dieser Zeichen kann noch nicht angegeben werden. Sie sind für die Unterscheidung von Text- und numerischen Werten gedacht. Möglich sind folgende Zeichen:

zwei Anführungszeichen oben = 34

ein Anführungszeichen oben = 39

Das Schlüsselwort ist STR, z.B. str=34.

DEZIMALTRENNZEICHEN:

Der Parameter "decdel" gibt an, welches Zeichen innerhalb numerischer Werte die Nachkommastellen von den Vorkommastellen trennt. Mögliche Zeichen sind:

Doppelpunkt = 58

Komma = 44

Semikolon = 59

Punkt = 46

Das Schlüsselwort ist DECDEL, z.B. decdel=46.

TAUSENDERTRENNER:

Der Parameter "thodel" gibt an, welches Zeichen innerhalb numerischer Werte die Tausenderstellen angibt. Mögliche Zeichen sind:

Komma = 44

Punkt = 46

Das Schlüsselwort ist THODEL, z.B. thodel=44.

Einige Beispiele für den Übergabeparameterstring:

"typ=TXT;dsn=d:\office31\CONFIG\lager.sdf;charset=ANSI;hdr=False;del=59;str34"

"typ=TXT;dsn=d:\SAVE\Firmen1.txt;charset=IBMPC;fix=false;hdr;del=59;str=34;datedir=DMY;

datesep=46;datezero=false;datefour=false;timedel=58;decdel=46;isdir=false;thodel=44;"

DBASE-DATENBANKEN

GELÖSCHTE DATENSÄTZE NICHT MIT ANZEIGEN:

Bei dBase-Datenbanken werden Datensätze nicht physikalisch gelöscht, sondern nur als gelöscht markiert. Mit dem Parameter "deleted" wird angegeben, ob markierte Datensätze angezeigt werden sollen.

Das Schlüsselwort ist DELETED, z.B. deleted=false Voreinstellung ist FALSE.

ALLE DBASE-DATEIEN IM VERZEICHNIS WERDEN EINGEBUNDEN:

BEISPIEL:

"DBF-Lager=typ=DBF;dsn=d:\office40\CONFIG\lager.dbf;charset=IBMPC;deleted=false;isdir=false;"

ODBC-DATENBANK

USERID FÜR DIE ODBC-DATENBANK:

Mit dem Parameter "uid" kann man seine UserID (Zugangsnamen) für die ODBC-Datenbank zur Datenbank übertragen. Dies ist nur bei Datenbanken sinnvoll, für die Sie einen Zugangsnamen benötigen. Der Zugangsname wird ohne Anführungszeichen übergeben (z. B. uid=Zugang).

Das Schlüsselwort ist UID, z. B. uid=Zugangsname.

PASSWORT FÜR DIE ODBC-DATENBANK:

Mit dem Parameter "pwd" wird angegeben, ob beim Öffnen der ODBC-Datenbank ein Passwort übergeben wird. Der Anwender wird dann nach einem Passwort gefragt.

Das Schlüsselwort ist PWD, z. B. pwd=*.

Beispiel:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
```

```
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

AddLibrary-Methode

Fügt zur Verwaltung der StarBasic-Bibliotheken eine neue Bibliothek hinzu.

Siehe auch: RemoveLibrary, LoadLibrary, UnloadLibrary.

KLASSE:
OfficeDocument, Application

SYNTAX:
AddLibrary(LibraryFileName As String, Mode As Long, Replace As Boolean, LibraryName As String)

RÜCKGABE:
Boolean

PARAMETER:
LibraryFileName: Name und ggf. Pfad der Datei, aus der die Bibliothek geladen werden soll. Wird nur der Dateiname angegeben, wird der Pfad genommen, der unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE für Basic eingestellt ist. Bei relativen Pfadnamen wird die Datei im Basic-Pfad gesucht (sbl-Datei oder ein Office-Dokument).

Mode: Optionaler Parameter, der bestimmt, auf welche Art und Weise die Bibliothek eingebunden werden soll. Folgende Werte sind möglich:

0 - Als Kopie (default)

1 - Als Referenz

2 - Als externe Datei

Replace: Optionaler Parameter, der angibt, ob die neue Bibliothek eine bereits vorhandene gleichen Namens ersetzen soll (True) oder nicht (False). Ohne Angabe des Parameters wird er standardmäßig auf True gesetzt.

LibraryName: Optionaler Parameter, der den Namen der Bibliothek in der Datei angibt, aus der sie geladen wird. Wird dieser Parameter nicht übergeben, wird der Name der Datei genommen (LibraryName = LibraryFileName ohne Extension)

BEISPIEL:

```
Sub ExampleAddLibrary
Const sLibName = "NeueBibliotek"
REM Const BiblioDoc = REM Pfad und Dateiname von dem Dokument
das die Bibliothek enthält
Dim vStatus as Variant
vStatus = Application.AddLibrary(BiblioDoc , 1, TRUE, sLibName
)
vStatus = Application.RemoveLibrary( sLibName )
End Sub
```

Address-Eigenschaft

Liefert das Address-Objekt, welches die Benutzerdaten enthält. Diese werden entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Desktop

SYNTAX:
Address

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
```

```
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Application-Eigenschaft

Liefert das zugehörige Application-Objekt "StarOffice". OLE-Automation definiert, daß diese Eigenschaft von jedem Objekt unterstützt werden muß.

KLASSE:
Object

SYNTAX:
Application

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleApplication
Const sDoc="TestDoc.sdw"
Documents.Add("sWriter4")
ActiveDocument.SaveAs( Application.PathSettings.Work & "\" &
sDoc )
Selection.Insert("Name= "& ActiveDocument.Name)
Selection.InsertPara
Selection.Insert("Pfad= "& ActiveDocument.Path)
Selection.InsertPara
Selection.Insert("Pfad mit Name= " & ActiveDocument.FullName)
Selection.InsertPara
End Sub
```

ArrangeHorizontal-Methode

Ordnet die nicht als Symbol dargestellten Fenster horizontal an.

Siehe auch: ArrangeVertical

KLASSE:
Windows

SYNTAX:
ArrangeHorizontal()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT"  REM Textsegment, um neue Dokumente
identifizieren zu können
Dim iCount As Integer
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
REM Alle neuen Dokumente schließen
iCount=1 REM Der erste Index der WindowsCollection ist 0.
Do While iCount <= Windows.Count
If Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 Then
Windows.Remove( iCount ) REM Entfernen ohne Nachfrage!
Else
iCount = iCount + 1
End If
Loop
Windows.Close REM Alles Schließen mit Nachfrage
End Sub
```

ArrangeIcons-Methode

Ordnet die als Sinnbilder dargestellten Dokumente neu an.

Falls keine Icons existieren, kommt es bei Aufruf dieser Methode zu einem Laufzeitfehler.

KLASSE:
Windows

SYNTAX:
ArrangeIcons()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT" REM Textsegment, um neue Dokumente
identifizieren zu können
Dim iCount As Integer
```

```
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
REM Alle neuen Dokumente schließen
iCount=1 REM Der erste Index der WindowsCollection ist 0.
Do While iCount <= Windows.Count
If Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 Then
Windows.Remove( iCount ) REM Entfernen ohne Nachfrage!
Else
iCount = iCount + 1
End If
Loop
Windows.Close REM Alles Schließen mit Nachfrage
End Sub
```

ArrangeVertical-Methode

Ordnet die nicht als Symbol dargestellten Fenster vertikal an.

Siehe auch: ArrangeHorizontal

KLASSE:
Windows

SYNTAX:
ArrangeVertical()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT" REM Textsegment, um neue Dokumente
identifizieren zu können
Dim iCount As Integer
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
```

```
REM Alle neuen Dokumente schließen
iCount=1 REM Der erste Index der WindowsCollection ist 0.
Do While iCount <= Windows.Count
If Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 Then
Windows.Remove( iCount ) REM Entfernen ohne Nachfrage!
Else
iCount = iCount + 1
End If
Loop
Windows.Close REM Alles Schließen mit Nachfrage
End Sub
```

AutoBackup-Eigenschaft

Bestimmt oder ermittelt, ob beim Speichern von Dokumenten automatisch ein Backup der alten Version angelegt werden soll.

KLASSE:
SaveSettings

SYNTAX:
AutoBackup

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob ein Backup angelegt werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
```

```
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

AutoBrowse-Methode

Aktiviert die AutoBrowse-Funktion vom StarOffice. Dabei wird die Selektion in der Dokument-Ansicht als URL interpretiert und der URL geladen, oder die aktuelle Selektion wird als Name im Adreßbuch gesucht und der dortige URL als Sprungziel genommen.

Die Arbeitsweise dieser Methode hängt davon ob, ob ein Parameter übergeben wird oder nicht: Wird kein Übergabeparameter angegeben, wird die aktuelle Selektion als URL interpretiert und der URL geladen. Wird nur der erste Parameter für die URL-Adresse übergeben, wird der angegebene URL geladen. Wird nur der zweite Parameter angegeben, wird der angegebene Name als Eintrag im Adreßbuch gesucht und der URL geladen, der im Adreßbuch eingetragen ist.

KLASSE:
Desktop

SYNTAX:
AutoBrowse(BrowseAddress As String, Name As String)

RÜCKGABE:
String

PARAMETER:
BrowseAddress: Der URL, der geladen werden soll. Dieser Parameter kann weggelassen werden, wenn der zweite Parameter "Name" angegeben wird und die URL-Adresse im Adreßbuch vorliegt.

Name: Der zum URL zugehörige Name, der als Adreßbuch-Eintrag vorliegt. Dieser Parameter kann weggelassen werden, wenn "BrowseAddress" bereits angegeben wurde.

Beispiel:

```
Sub ExampleAutoBrowse
Documents.Add( "swriter4" )
ActiveWindow.Selection.Insert("Star Division GmbH anbrowsen")
Application.AutoBrowse("http://www.stardivision.de")
End Sub
```

AutoCorrectDlg-Methode

Ruft den Dialog AutoKorrektur auf, in dem die Optionen für die automatische Korrektur gesetzt werden können.

Klasse:
Desktop

Syntax:
AutoCorrectDlg()

Beispiel:

```
Sub ExampleAutoCorrectDlg
Application.AutoCorrectDlg
End Sub
```

AutoPilotAgenda-Methode

Startet den AutoPiloten zum Erstellen einer Agenda.

Klasse:
Desktop

Syntax:
AutoPilotAgenda()

Beispiel:

```
Sub ExampleAutopilotAgenda
Application.AutoPilotAgenda
End Sub
```

AutoPilotFax-Methode

Startet den AutoPiloten zum Erstellen eines Faxes.

Klasse:
Desktop

Syntax:
AutoPilotFax()

Beispiel:

```
Sub ExampleAutoPilotFax
Application.AutoPilotFax
End Sub
```

AutoPilotLetter-Methode

Startet den AutoPiloten zum Erstellen eines Briefes.

Klasse:
Desktop

Syntax:
AutoPilotLetter()

Beispiel:

```
Sub ExampleAutoPilotLetter
Application.AutoPilotLetter
End Sub
```

AutoPilotMemo-Methode

Startet den AutoPiloten zum Erstellen eines Memos.

Klasse:
Desktop

Syntax:
AutoPilotMemo()

Beispiel:

```
Sub ExampleAutoPilotMemo
Application.AutoPilotMemo
End Sub
```

AutoSave-Eigenschaft

Bestimmt oder ermittelt, ob Dokumente automatisch periodisch gespeichert werden sollen.

Siehe auch: AutoSaveInterval

Klasse:
SaveSettings

Syntax:
AutoSave

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob ein automatisches Speichern erfolgen soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

AutoSaveInterval-Eigenschaft

Bestimmt oder ermittelt, in welchen Zeitabständen Dokumente automatisch periodisch gespeichert werden sollen. Diese Eigenschaft hat nur dann eine Wirkung, wenn mit AutoSave = True ein automatisches Speichern gesetzt ist.

Siehe auch: AutoSave

KLASSE:
SaveSettings

SYNTAX:
AutoSaveInterval

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Zeitabstände für das automatische Speichern in Minuten angibt.

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

Eigenschaften und Methoden mit "B"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "B" beginnen.

BackColor-Eigenschaft

Bestimmt oder ermittelt die Farbe des Hintergrundmusters des selektierten Objekts.

Die BackColor-Eigenschaft hat für Interior-Objekte nur dann eine Wirkung, wenn zuvor mit der Style-Eigenschaft ein Füllmuster gesetzt worden ist.

Siehe auch: FillColor, Transparent, Style.

KLASSE:
Interior

SYNTAX:
BackColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe des Hintergrundmusters angibt.

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.BackColor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

Backup-Eigenschaft

Bestimmt oder ermittelt den Pfad für Sicherungskopien. Dieser wird im Office unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
Backup

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad für Sicherungskopien angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makro-Dateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörterbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOffice-Module: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
```

```
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Backward-Eigenschaft

Bestimmt oder ermittelt, ob die Suchoption RÜCKWÄRTS SUCHEN aktiv ist.

KLASSE:
SearchSettings

SYNTAX:
Backward

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob ab der Cursorposition rückwärts (True) oder vorwärts (False) gesucht werden soll.

BEISPIEL:

```
Sub ExampleBackward
Const sText="Text" REM Text, der vorhanden ist
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GotoEndOfDoc TRUE
Wait 2000
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.InSelection = TRUE
ActiveWindow.SearchSettings.CaseSensitive = TRUE
ActiveWindow.SearchSettings.WordOnly = TRUE
ActiveWindow.Search sText
MsgBox Selection.Value,64,"Suchen und Finden"
End Sub
```

Basic-Eigenschaft

Bestimmt oder ermittelt den Pfad für StarBasic-Dateien. Dieser wird im Office unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
Basic

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad für StarBasic angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makro-Dateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörterbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOffice-Module: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

BasicIDEAppear-Methode

Ruft die Entwicklungsumgebung für Makros auf. Dabei wird die Basic-IDE geöffnet und die Standard-Bibliothek "soffice.sbl" geladen.

KLASSE:
Desktop

SYNTAX:
BasicIDEAppear()

BEISPIEL:

```
Sub ExampleBasicIDEAppear
Documents.Add "sWriter4/Web"
Wait 3000
Application.BasicIDEAppear
End Sub
```

BasicStop-Methode

Stoppt das laufende StarBasic-Programm.

KLASSE:
Application

SYNTAX:
BasicStop()

BEISPIEL:

```
Sub ExampleBasicStop
Application.BasicStop
Print "Test" REM Diese Meldung erscheint nicht mehr.
End Sub
```

Beamer-Eigenschaft

Schaltet die Ansicht des Beamers ein oder aus oder ermittelt, ob der Beamer eingeblendet ist.

KLASSE:
Application

SYNTAX:
Beamer

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Beamer ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleBeamer
Application.Beamer = TRUE REM schaltet den Beamer an
End Sub
```

BeamerTaskSensitive-Eigenschaft

Bestimmt oder ermittelt, ob die Sichtbarkeit und der Inhalt des Beamers von der aktiven Task, d. h. dem aktuellen TaskWindow, festgelegt wird oder ob beides unabhängig davon ist, welche Task gerade aktiv ist.

KLASSE:
Application

SYNTAX:
BeamerTaskSensitive

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Zustand des Beamers von dem aktiven TaskWindow abhängt (True) oder nicht (False)

BEISPIEL:

```
Sub ExampleBeamerTaskSensitive
Application.BeamerTaskSensitive = TRUE
End Sub
```

BottomLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die untere Umrandungslinie beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

KLASSE:
Border

SYNTAX:
BottomLine

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

BrowseBackward-Methode

Geht in der Task auf das vorherige Dokument zurück. Diese Methode entspricht in seiner Funktion dem Symbol RÜCKWÄRTS in der Funktionsleiste.

KLASSE:
TaskWindow

SYNTAX:
BrowseBackward(nSteps As Long)

PARAMETER:
[nSteps]: Gibt die Anzahl der Schritte an, die rückwärts gesprungen werden soll. Wird kein Wert übergeben, wird standardmäßig nSteps = 1 genommen.

BEISPIEL:

```
REM Es muß bereits gebrowst worden sein.
ActiveWindow.TopWindow.BrowseBackward
```

BrowseForward-Methode

Geht in der Task wieder auf das nächste Dokument, falls man zuvor mit BrowseBackward zurückgegangen ist. Diese Methode entspricht in seiner Funktion dem Symbol VORWÄRTS in der Funktionsleiste.

KLASSE:
TaskWindow

SYNTAX:
BrowseForward(nSteps As Long)

PARAMETER:
[nSteps]: Gibt die Anzahl der Schritte an, die vorwärts gesprungen werden soll. Wird kein Wert übergeben, wird standardmäßig nSteps = 1 genommen.

BEISPIEL:

```
REM Es muß bereits gebrowst worden sein.
ActiveWindow.TopWindow.BrowseForward
```

BrowseHome-Methode

Lädt die eingestellte Homepage in den aktuellen Task. Die Homepage kann im Office unter EXTRAS - OPTIONEN - BROWSER... - SONSTIGES bestimmt werden.

KLASSE:
TaskWindow, Application

SYNTAX:
BrowseHome()

BEISPIEL:

```
Sub ExampleBrowseHome
Application.BrowseHome
End Sub
```

BrowserDialog-Methode

Ruft den Dialog OPTIONEN BROWSER auf.

KLASSE:
Desktop

SYNTAX:
BrowserDialog()

BEISPIEL:

```
Sub ExampleBrowserDialog
Application.BrowserDialog
End Sub
```

Eigenschaften und Methoden mit "C"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "C" beginnen.

CancelDownloads-Methode

Bricht alle Internet-Datenübertragungen ab.

Findet bei Aufruf dieser Methode momentan kein Download statt, dann kommt es zu einer Fehlermeldung.

KLASSE:
Application

SYNTAX:
CancelDownloads()

BEISPIEL:

```
Sub ExampleCancelDownloads
Documents.Open( "http://www.Stardivision.de" )
Application.CancelDownloads
End Sub
```

Caption-Eigenschaft

Ermittelt die Titelzeile des Programmfensters.

KLASSE:
Application

SYNTAX:
Caption

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

Beispiel:

```
Sub ExampleApplicationCaption
Print Application.Caption REM liefert "StarOffice 4.0"
End Sub
```

Cascade-Methode

Ordnet die nicht als Symbol dargestellten Fenster hintereinander an.

Klasse:
Windows

Syntax:
Cascade()

Beispiel:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT"  REM Textsegment, um neue Dokumente
identifizieren zu können
Dim iCount As Integer
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
REM Alle neuen Dokument schließen
iCount=1 REM Der erste Index der Windows-Collection ist 0.
Do While iCount <= Windows.Count
If Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 Then
Windows.Remove( iCount ) REM Entfernen ohne Nachfrage!
Else
iCount = iCount + 1
End If
Loop
Windows.Close REM Alles Schließen mit Nachfrage
End Sub
```

CaseSensitive-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche nach Groß- und Kleinschreibung unterschieden werden soll. Diese Funktion entspricht der Suchoption EXAKTE SUCHE im Dialog SUCHEN & ERSETZEN.

KLASSE:
SearchSettings

SYNTAX:
CaseSensitive

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche Groß- und Kleinschreibung berücksichtigt (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleBackward
Const sText="Text" REM Text, der vorhanden ist
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GotoEndOfDoc TRUE
Wait 2000
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.InSelection = TRUE
ActiveWindow.SearchSettings.CaseSensitive = TRUE
ActiveWindow.SearchSettings.WordOnly = TRUE
ActiveWindow.Search sText
MsgBox Selection.Value,64,"Suchen und Finden"
End Sub
```

City-Eigenschaft

Bestimmt oder ermittelt den Wohnort aus der Adresse, die bei den Benutzerdaten aufgeführt ist. Die Adresse wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
City

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Wohnort aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
```

```
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

ClassName-Eigenschaft

Liefert den Namen der Klasse, zu der das jeweilige Objekt gehört. Hat man z. B. ein Document-Objekt, kann man über ClassName herausfinden, ob es sich etwa um ein Textdokument oder ein Tabellendokument handelt, um dann an dem Objekt für den jeweiligen Dokumenttyp spezifische Methoden und Eigenschaften aufzurufen. Genauso kann über Selection.ClassName herausgefunden werden, was für ein Objekt gerade selektiert ist.

KLASSE:
Object

SYNTAX:
ClassName

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleParent
Dim oObj As Object
Documents.Add("sWriter4")
Selection.Insert( "Eine kleine Vererbungslehre:")
Selection.InsertPara
oObj = Selection.Paragraph
Selection.Insert( oObj.ClassName )
Selection.InsertPara
oObj = oObj.Parent
Selection.Insert( oObj.ClassName )
Selection.InsertPara
oObj = oObj.Parent
Selection.Insert( oObj.ClassName )
Selection.InsertPara
End Sub
```

Clear-Methode

Löscht alle Elemente aus einer Strings-Collection.

KLASSE:
Strings

SYNTAX:
Clear()

BEISPIEL:

```
Sub ExampleClear
Const sEntry = "Eintrag "
Const sMessage_1 = "Anzahl der Einträge der Strings-Collection:
"
Const sMessage_2 = "Anzahl der Einträge, nachdem Clear ausge-
führt wurde: "
Dim stList As New Strings
Dim iCount As Integer
For iCount = 1 To 3
stList.Add( sEntry & iCount )
Next iCount
Print sMessage_1 & stList.Count
stList.Clear
Print sMessage_2 & stList.Count
End Sub
```

ClearDiskCache-Methode

Löscht den Internet-Cache im File-System.

KLASSE:
Application

SYNTAX:
ClearDiskCache()

BEISPIEL:

```
Sub ExampleClearDiskCache
Application.ClearDiskCache
End Sub
```

ClearHistory-Methode

Löscht die Liste der Aktionen, die über Menü BEARBEITEN - RÜCKGÄNGIG rückgängig gemacht werden können und gibt somit Speicherplatz frei.

Ist das aktuelle Window-Objekt ein schreibgeschütztes Fenster oder existieren keine Aktionen, die rückgängig gemacht werden können, dann kommt es bei Aufruf dieser Methode zu einem Laufzeitfehler.

KLASSE:
Window

SYNTAX:
ClearHistory()

BEISPIEL:

```
Sub ExampleClearHistory
REM Es wird der Laufzeitfehler abgefangen.
On Error Goto UNDOERROR
Const sMessage_1 = "Der Undo-Puffer wird gelöscht."
Const sMessage_2 = "Dieser Text kann nicht rückgängig gemacht
werden."
Const sMessage_3 = "Mit gelöschtem Undo-Puffer kann nichts
rückgängig gemacht werden."
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
ActiveWindow.ClearHistory
REM Es kommt hier zu einem Laufzeitfehler, da das Undo ausge-
schaltet ist.
Selection.Undo
Exit Sub
UNDOERROR:
Selection.Insert sMessage_3
End Sub
```

ClearMemCache-Methode

Löscht den Internet-Cache im Hauptspeicher.

KLASSE:
Application

SYNTAX:
ClearMemCache()

BEISPIEL:

```
Sub ExampleClearMemCache
Application.ClearMemCache
End Sub
```

Close-Methode

Schließt das Fenster und damit eine Sicht eines Dokumentes. Gibt es noch weitere Sichten auf das gleiche Dokument, bleiben diese geöffnet.

Erst wenn die letzte Sicht auf ein Dokument geschlossen wird, wird auch das Dokument geschlossen. Wurde das Dokument geändert, erfolgt eine Rückfrage, ob die Änderungen gespeichert werden sollen.

KLASSE:
TaskWindow

SYNTAX:
Close()

BEISPIEL:

```
Sub ExampleWindowDocument
Dim oDoc As Object
Documents.Add("sWriter4")
ActiveWindow.NewWindow  REM Ein andere Sicht
ActiveDocument.PrintPreview=TRUE
oDoc=ActiveWindow.Document
MsgBox oDoc.Name,64,"Der Name des Dokumentes ist unabhängig von
der Sicht."
ActiveDocument.Close "",False
Do While ActiveDocument.Closing REM Warten bis Dokument fertig
geschlossen wurde.
   Wait 10
Loop
End Sub
```

Close-Methode

Schließt das Dokument und damit auch alle seine Sichten. Mit Hilfe der Übergabeparameter kann man dabei ein Dokument unter einem anderen Namen speichern.

Konnte die Methode erfolgreich durchgeführt werden, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
Document

SYNTAX:
Close(saveChanges As Boolean, fileName As String)

RÜCKGABE:
Boolean

PARAMETER:
SaveChanges: Boolescher Ausdruck, der bestimmt, ob das Dokument vor dem Schließen gespeichert werden soll (True) oder nicht (False).

FileName: Gültiger Dateiname oder URL, unter dem das Dokument gespeichert werden soll.

BEISPIEL:

```
Sub ExampleDocumentsOpen
Const sDoc1 = "Test1"
Const sDoc2 = "Test2"
Const sOfficeFilter="soffice4: StarOffice Dokumente"
Dim sPath1 As String, sPath2 As String
sPath1 = Application.PathSettings.Work & "\" & sDoc1 & ".sdw"
sPath2 = Application.PathSettings.Work & "\" & sDoc2 & ".sdw"
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 1"
ActiveDocument.SaveAs sPath1
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 2"
ActiveDocument.SaveAs sPath2
Windows(sDoc1 & ":1").Selection.Insert ": geändert" REM Jeweils
die erste Sicht bearbeiten
Windows(sDoc2 & ":1").Selection.Insert ": geändert"
Documents.SaveAll
Documents(sDoc1).Close False, ""
Documents(sDoc2).Close False, ""
Documents.Open sPath1,sOfficeFilter  REM Öffnen mit Filterliste
End Sub
```

Close-Methode

Schließt alle Fenster beziehungsweise alle Dokumente.

Wird dabei ein modifiziertes Dokument geschlossen, dann wird es auf dem Bildschirm nach vorne geholt, und es erfolgt eine Abfrage, ob das Dokument gespeichert werden soll.

Diese Methode hat keinen Einfluß auf das Desktopfenster. Das Desktopfenster wird niemals geschlossen, es bleibt also auch bei Aufruf von "Documents.Close" oder "Windows.Close" offen.

KLASSE:
Documents, Windows

SYNTAX:
Close()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT"  REM Textsegment, anhand dessen neue
Dokumente identifiziert werden können
Dim iCount As Integer
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
```

```
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
REM alle neuen Dokument schliessen
iCount=1 REM erster Index der Windows-Collection ist 0
Do While iCount <= Windows.Count
if Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 then
Windows.Remove( iCount ) REM entfernen ohne Nachfrage !
else
iCount = iCount + 1
end if
loop
Windows.Close REM Alles Schliessen mit Nachfrage
End Sub
```

CloseDatabaseConnection-Methode

Schließt eine geöffnete Datenbank inklusive aller Verbindungen zu darin enthaltenen Tabellen.

Siehe auch: OpenDatabaseConnection

KLASSE:
Desktop

SYNTAX:
CloseDatabaseConnection(DBName As String)

PARAMETER:
DBName: Ein Zeichenfolgenausdruck, der die Bezeichnung der Datenbank aufnimmt, die geschlossen werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadOnly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
```

```
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldeten Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet.
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden!
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes?
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden.
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
```

```
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

CloseQueryConnection-Methode

Beendet die Verbindung zur einer Datenbank-Abfrage.

Siehe auch: OpenQueryConnection

KLASSE:
Desktop

SYNTAX:
CloseQueryConnection(QueryName As String)

PARAMETER:
[QueryName]: Name der Datenbank-Abfrage, zu der die Verbindung aufgehoben werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, dann wird die Verbindung zur zuletzt verwendeten Abfrage beendet.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New Strings
Dim stAppendRec As New Strings
```

```
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nach dem Öffnen muß man zuerst zu einem Datensatz springen.
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Strings-Collection enthält die gleichen
REM Informationen wie die stDataRow-Strings-Collection.
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End If
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End If
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
End If
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
If iQueCount > 0 Then
```

```
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End If
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht.
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt.
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

CloseTableConnection-Methode

Beendet die Verbindung zu einer Datenbank-Tabelle.

Siehe auch: OpenTableConnection

KLASSE:
Desktop

SYNTAX:
CloseTableConnection(TableName As String)

PARAMETER:
[TableName]: Name der Datenbank-Tabelle, zu der die Verbindung aufgehoben werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, dann wird die Verbindung zur zuletzt verwendeten Tabelle beendet.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadOnly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
```

```
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet.
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes auf einen Datensatz positio-
niert werden!
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes?
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden.
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
```

```
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

Closing-Eigenschaft

Gibt an, ob das Dokument gerade geschlossen wird. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
Document

SYNTAX:
Closing

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleWindowDocument
Dim oDoc As Object
Documents.Add("sWriter4")
ActiveWindow.NewWindow  REM Ein andere Sicht
ActiveDocument.PrintPreview=TRUE
oDoc=ActiveWindow.Document
MsgBox oDoc.Name,64,"Der Name des Dokumentes ist unabhängig von
der Sicht."
ActiveDocument.Close "",False
Do While ActiveDocument.Closing REM Warten bis Dokument fertig
geschlossen wurde.
   Wait 10
Loop
End Sub
```

Color-Eigenschaft

Bestimmt oder ermittelt die Farbe einer Umrandungslinie.

Siehe auch: Style

KLASSE:
BorderLine

SYNTAX:
Color

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe angibt.

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

CommonTaskBarVisible-Eigenschaft

Bestimmt oder fragt ab, ob die Präsentationsleiste aus StarImpress eingeschaltet ist.

KLASSE:
Application

SYNTAX:
CommonTaskBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Prasentationsleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleCommonTaskBarVisible
Application.CommonTaskBarVisible = FALSE
Wait 1000
Application.CommonTaskBarVisible = TRUE
End Sub
```

Company-Eigenschaft

Bestimmt oder ermittelt die Firma aus den Benutzerdaten. Diese werden entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Company

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Firma aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
```

```
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Config-Eigenschaft

Bestimmt oder ermittelt den Pfad für Konfigurationsdateien. Dieser wird unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
Config

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad des Config-Verzeichnisses angibt.

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makro-Dateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörterbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOffice-Module: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

ConfigureDialog-Methode

Ruft den Dialog Anpassen auf, mit dem die StarOffice-Oberfläche frei konfiguriert werden kann.

Klasse:
Application

Syntax:
ConfigureDialog()

Beispiel:

```
Sub ExampleConfigureDialog
Application.ConfigureDialog
End Sub
```

ConfigureToolboxVisible-Eigenschaft

Bestimmt oder ermittelt, ob der Dialog Symbolleisten bearbeiten eingeblendet wird.

Klasse:
Application

Syntax:
ConfigureToolboxVisible

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Dialog Symbolleisten bearbeiten ein- (True) oder ausgeblendet (False) ist.

Beispiel:

```
Sub ExampleConfigureToolbox
Application.ConfigureToolboxVisible = TRUE
Wait 2000
Application.ConfigureToolboxVisible = FALSE
End Sub
```

Count-Eigenschaft

Liefert die Anzahl der Elemente einer Strings-Collection.

Klasse:
Strings

Syntax:
Count

Rückgabe:
Long

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleClear
Const sEntry = "Eintrag "
Const sMessage_1 = "Anzahl der Einträge der Strings-Collection:
"
Const sMessage_2 = "Anzahl der Einträge, nachdem Clear ausge-
führt wurde: "
Dim stList As New Strings
Dim iCount As Integer
For iCount = 1 To 3
stList.Add( sEntry & iCount )
Next iCount
Print sMessage_1 & stList.Count
stList.Clear
Print sMessage_2 & stList.Count
End Sub
```

Count-Eigenschaft

Liefert die Anzahl der Elemente in der jeweiligen Collection.

Klasse:
Collection

Syntax:
Count

Rückgabe:
Long

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleCollectionCount
MsgBox("Momentan sind " &_
   Application.Documents.Count &_
   " Dokumente und " &_
   Application.Windows.Count &_
   " Ansichten geöffnet.",0,"StarOffice"
End Sub
```

Count-Eigenschaft

Zählt die Anzahl der Tabulatoren in der jeweiligen Selektion.

Klasse:
TabStops

Syntax:
Count

Rückgabe:
Long

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM" ) REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM" )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount = 0 To iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText = sText & "Position: " & s1 & " Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &" Füllzeichen: " & s4 & Chr(13)
Next
End With
```

```
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

Country-Eigenschaft

Bestimmt oder ermittelt den Ländereintrag aus der Adresse, die bei den Benutzerdaten aufgeführt ist. Die Adresse wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Country

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Ländereintrag aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
```

```
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

CountryId-Eigenschaft

Ermittelt die ID-Nummer für landesspezifische Eigenschaften, wie zum Beispiel Standard-Maßeinheit, Dezimaltrenner usw.

Folgende Rückgabewerte gibt es:

255 - Keine

1030 - Dänisch

1031 - Deutsch

2055 - Deutsch (CH)

2057 - Englisch (UK)

1033 - Englisch (US)

1035 - Finnisch

1036 - Französisch

1040 - Italienisch

1043 - Niederländisch

1044 - Norwegisch Bokmal

2068 - Norwegisch Nynorsk

2070 - Portugiesisch

1053 - Schwedisch

1034 - Spanisch

Siehe auch: Language, LanguageID.

KLASSE:
Desktop

SYNTAX:
CountryId

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleCountryID
Print "Die CountryID dieses Systems lautet : " &
Application.CountryID
End Sub
```

CreateForm-Methode

Legt ein neues Datenbank-Formular aus einem bestehenden StarOffice-Textdokument in der geöffneten Datenbank ab.

KLASSE:
Desktop

SYNTAX:
CreateForm(FormName As String, URL As String)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Name des neuen Formulares.

URL: Pfadangabe für die SDW-Datei, aus der das Formular erzeugt werden soll.

Beispiel:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig.
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei, die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt.
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt.
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).TopWindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

CreateIndex-Methode

Legt einen Index auf eine Tabellenspalte einer Datenbank-Tabelle an.

KLASSE:
Desktop

SYNTAX:
CreateIndex(TableName As String, IndexName As String, Columns As Object, Unique As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Datenbank-Tabelle.

IndexName: Name des Index.

Columns: Parameter vom Typ Strings, der die Indizierung einer Tabellenspalte bestimmt. Columns besteht aus einer String-Variablen der Form "ColumnName, Order".

ColumnName: Name der Tabellenspalte, die mit einem Index belegt werden soll.

Order: Boolescher Ausdruck, der angibt, ob der Index aufsteigend (True) oder absteigend (False) ist.

Eine gültige Indexbeschreibung sieht z. B. so aus: "VORNAME,TRUE".

Unique: Boolescher Ausdruck, der angibt, ob gleiche Feldinhalte zusammengefaßt (True) oder separat betrachtet (False) werden sollen. Taucht ein Feldinhalt innerhalb einer Spalte mehrfach auf, dann wird mit Unique = True immer nur der erste Datensatz angezeigt.

BEISPIEL:

```
Sub ExampleDatabaseIndex
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Dim stDataRow As New Strings
Dim stIndexStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
```

```
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
stIndexStrings.Add sTextName & "1,TRUE"
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
vStatus = Application.CreateIndex(sTabName, sIndexName, stIn-
dexStrings, FALSE)
vStatus = Application.DeleteIndex(sIndexName)
Application.CloseTableConnection(sTabName)
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

CreateLink_InSelected-Methode

Erzeugt eine Verknüpfung auf den angegebenen URL mit dem angegebenen Titel in dem im Explorer selektierten Ordner.

KLASSE:
TaskWindow

SYNTAX:
CreateLink_InSelected(URL As String, Title As String)

PARAMETER:
URL: Zeichenfolge, die den Ziel-URL angibt, auf den die Verknüpfung verweisen soll.

[Title]: Zeichenfolge, die den Titel beziehungsweise den Namen der Verknüpfung angibt. Wird dieser Parameter nicht übergeben, wird der URL selbst als Titel genommen.

BEISPIEL:

```
REM Der Bookmarkordner muß bereits selektiert worden sein.
ActiveWindow.CreateLink_InSelected("http://www.stardivision.de"
,"home")
```

CreateQuery-Methode

Legt eine Datenbank-Abfrage in der aktuell geöffneten Datenbank an.

KLASSE:
Desktop

SYNTAX:
CreateQuery(QueryName As String, SQL_Statement As String, Native As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Abfrage, die neu angelegt werden soll.

SQL_Statement: Gültiger SQL-Ausdruck, mit dem die Abfrage definiert wird.

Native: Gibt an, ob es sich um eine Native SQL handelt (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleShowQuery
Const sQueryName = "Abfrage"
Dim sDataName As String, sTableName As String
Dim vStatus As Variant
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.ShowQuery( sQueryName )
Application.DeleteQuery( sQueryName )
End Sub
```

CreateReport-Methode

Legt einen neuen Datenbank-Bericht in der aktuell geöffneten Datenbank an.

Siehe auch: DeleteReport

KLASSE:
Desktop

SYNTAX:
CreateReport(ReportName As String, URL As String)

RÜCKGABE:
String

PARAMETER:
ReportName: Gibt den Namen des neuen Berichts an.

URL: Gibt den Pfad des StarWriter-Dokuments an, aus dem der Bericht erzeugt werden soll.

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt.
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt.
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).TopWindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
```

```
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

CreateTable-Methode

Legt eine neue Datenbank-Tabelle in der aktuell geöffneten Datenbank an.

KLASSE:
Desktop

SYNTAX:
CreateTable(TableName As String, Columns As Object)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Gibt den Namen der Tabelle an, die neu angelegt werden soll.

Columns: Strings-Variable, die die Datensatzfelder der Tabelle bestimmt. Columns setzt sich aus mehreren String-Variablen zusammen, die je eine Spalte der Tabelle definieren. Eine Spaltenbeschreibung hat folgende Bestandteile:

Spalte=Feldname, Feldtyp, Feldlänge, Eingabe.

Feldname: Gibt den Namen der Spalte an.

Eingabe: Gibt an, ob eine Eingabe später zwingend erforderlich ist (True) oder ob ein Datenfeld in dieser Spalte auch leer sein darf (False).

Feldtyp und Feldlänge legen Typ und ggf. Subtyp einer Spalte fest. Je nach Datenbankstruktur können Feldtyp und Feldlänge variieren. Bei einer Tabelle im Format dBaseIII gelten folgende Werte:

Wert	Feldtyp	Subtyp
TEXT	Text	Zeichenlänge
COUNTER	Zähler	
IMAGE	Bild	
MEMO	Memo	
NUMERIC	Zahl	Zahlenformat
DATE/TIME	Datum/Zeit	
DATE	Datum	
TIME	Zeit	

Wert	Feldtyp	Subtyp
CURRENCY	Währung	
BOOL	Ja/Nein	

Der Subtyp Zahlenformat kann folgende Werte annehmen:

Basic-Wert	dBase-Zahlenformat	Länge
INTEGER	Smallint	2 Byte = 16 Bit
LONG	Integer	4 Byte = 32 Bit
SINGLE	Single	4 Byte Fließkomma
DOUBLE	Double	8 Byte Fließkomma

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
```

```
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
```

```
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

CreationDate-Eigenschaft

Ermittelt das Datum, an dem das Dokument erstmalig gespeichert wurde. Dieses Datum wird aus den Dokumenteigenschaften gelesen.

KLASSE:
DocumentInfo

SYNTAX:
CreationDate

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
```

```
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

CreationTime-Eigenschaft

Ermittelt die Uhrzeit, zu der das Dokument erstmalig gespeichert wurde. Diese Uhrzeit wird aus den Dokumenteigenschaften gelesen.

KLASSE:
DocumentInfo

SYNTAX:
CreationTime

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
```

```
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Creator-Eigenschaft

Diese Eigenschaft ist nur aus Kompatiblitätsgründen zu OLE-Automation vorhanden und im StarOffice nicht belegt. Sie kann nur über externe Programme angesprochen werden.

KLASSE:
Object

SYNTAX:
Creator

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

Beispiel:

```
REM Nur wegen OLE-Automation vorhanden.
```

CurrentDate-Eigenschaft

Ermittelt das aktuelle Tagesdatum in dem durch die Systemeinstellung vorgegebenen Format.

Klasse:
Application

Syntax:
CurrentDate

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleDateAndTime
MsgBox "Datum: "& Application.CurrentDate & Chr(13) &_
"Zeit: " & Application.CurrentTime ,0,"Datum und Zeit"
End Sub
```

CurrentTime-Eigenschaft

Ermittelt die aktuelle Uhrzeit in dem durch die Systemeinstellung vorgegebenen Format.

Klasse:
Application

Syntax:
CurrentTime

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleDateAndTime
MsgBox "Datum: "& Application.CurrentDate & Chr(13) &_
"Zeit: " & Application.CurrentTime ,0,"Datum und Zeit"
End Sub
```

Eigenschaften und Methoden mit "D"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "D" beginnen.

DataAppendRecord-Methode

Fügt der aktuellen Datenbank einen neuen Datensatz hinzu. Die Datenfeldinhalte müssen als Zeichenfolge durch Kommata getrennt in der Reihenfolge der Datenspalten des Datensatzes übergeben werden.

Konnte der Datensatz erfolgreich hinzugefügt werden, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
Desktop

SYNTAX:
DataAppendRecord(NewRecord As String)

RÜCKGABE:
Boolean

PARAMETER:
NewRecord: Zeichenfolgenausdruck, der die Inhalte der Datenfelder des neuen Datensatzes durch Kommata getrennt angibt.

In NewRecord müssen alle Datenfelder des neuen Datensatzes berücksichtigt werden. Soll einem Feld kein Wert zugewiesen werden, wird dies mit zwei aufeinanderfolgenden Kommata angegeben.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
```

```
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nach dem Öffnen muß man zuerst zu einem Datensatz springen.
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Strings-Collection enthält die gleichen
REM Informationen wie die stDataRow-Strings-Collection.
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End If
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End If
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
End If
```

```
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
If iQueCount > 0 Then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End If
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht.
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt.
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataAppendWholeRecord-Methode

Fügt der aktuellen Datenbank einen neuen Datensatz hinzu. Konnte das Hinzufügen des neuen Datensatzes erfolgreich durchgeführt werden, liefert die Methode den Wert True zurück, andernfalls False.

KLASSE:
Desktop

SYNTAX:
DataAppendWholeRecord(ColumnNames As Object, Content As Object)

RÜCKGABE:
Boolean

PARAMETER:
ColumnNames: Strings-Variable, die die Spaltennamen enthält.

Content: Strings-Variable, die den Inhalt der Spalten enthält.

Wenn nur ein Parameter übergeben wird, wird der gesamte Datensatzinhalt im ersten Parameter erwartet. Wenn beide Parameter übergeben werden, werden nur die Spalten gefüllt, die in ColumnNames angegeben wurden. Alle anderen Spalten bleiben leer.

Beispiel:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
```

```
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

Database-Eigenschaft

Bestimmt oder ermittelt den Pfad für Datenbanken. Dieser wird unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

Syntax:
Database

Rückgabe:
String

Parameter:
[= String]: Zeichenfolge, die den Pfad für Datenbanken angibt.

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makro-Dateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörterbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOffice-Module: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

DataColumn-Methode

Liefert den Inhalt der angegebenen Spalte des aktuellen Datensatzes als String zurück.

KLASSE:
Desktop

SYNTAX:
DataColumn(ColumnNumber As String)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Zeichenfolge, die die Spalte angibt, deren Inhalt ermittelt werden soll. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadOnly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldeten Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet.
```

```
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes auf einen Datensatz positio-
niert werden!
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes?
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden.
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataColumnName-Methode

Liefert den Spaltennamen der angegebenen Spalte aus einer Datenbank-Tabelle.

KLASSE:
Desktop

SYNTAX:
DataColumnName(ColumnNumber As Long)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Nummer des Datenbankfeldes, dessen Bezeichnung ermittelt werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadOnly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet.
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes auf einen Datensatz positio-
niert werden!
```

```
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden.
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataColumnNumber-Methode

Liefert die Anzahl der Spalten einer Datenbank-Tabelle.

KLASSE:
Desktop

SYNTAX:
DataColumnNumber()

RÜCKGABE:
Integer

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
```

```
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
```

```
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataColumnType-Methode

Gibt den Feldtyp des angegebenen Datenbankfeldes zurück.

Die folgenden Feldtypen werden von der Methode zurückgegeben:

Feldtyp	**Beschreibung**
Bookmark	Das Bookmarkfeld der Datenbank
Serial	Selbständig inkrementierendes Datenbankfeld (Zählerfeld)
Char	Zeichenfeld mit fester Länge

Feldtyp	Beschreibung
VarChar	Zeichenfeld mit variabler Länge
LongVarChar	Zeichenfeld mir variabler Länge und sehr großer Kapazität (Memofeld)
Decimal	Festkommazahl
Numeric	Festkommazahl (größere Kapazität als Decimal)
Bit	Ja/Nein-Feld (1 Bit)
TinyInt	Feld für Ganzzahlen bis 8 Bit
SmallInt	Feld für Ganzzahlen bis 16 Bit
Integer	Feld für Ganzzahlen bis 32 Bit
BigInt	Feld für Ganzzahlen bis 64 Bit
Real	Fließkommazahl
Float	Fließkommazahl (größere Kapazität als Real)
Double	Fließkommazahl (größere Kapazität als Float)
Binary	Feld für Binärdaten fester Länge
VarBinary	Feld für Binärdaten variabler Länge
LongVarBinary	Feld für Binärdaten variabler Länge (größere Kapazität als VarBinary)
Date	Datumsfeld
Time	Uhrzeitfeld
TimeStamp	Feld, das zur Verwaltung mehrerer Benutzer benötigt wird
Currency	Währungsformatfeld
Boolean	Ja/Nein-Feld (1 Bit)

Größe und Eigenschaften einzelner Datenbankfelder können sich abhängig von der verwendeten Plattform und Datenbank voneinander unterscheiden. Einige Datenbanken weisen nicht alle aufgeführten Datenbankfelder auf.

KLASSE:
Desktop

SYNTAX:
DataColumnType(ColumnNumber As String)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Zeichenfolge, die die Spalte angibt, deren Feldtyp ermittelt werden soll. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
```

```
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
```

```
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataDefinedDatabase-Methode

Liefert den Namen einer angemeldeten Datenbank.

Siehe auch: DataDefinedDatabaseNumber

KLASSE:
Desktop

SYNTAX:
DataDefinedDatabase(DefinedDatabaseNumber As Long)

RÜCKGABE:
String

PARAMETER:
DefinedDatabaseNumber: Nummer der Datenbank, deren Name ermittelt werden soll. Der Wertebereich liegt zwischen 1 und der Anzahl der angemeldeten Datenbanken.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
```

```
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataDefinedDatabaseNumber-Methode

Liefert die Anzahl der angemeldeten Datenbanken.

KLASSE:
Desktop

SYNTAX:
DataDefinedDatabaseNumber()

RÜCKGABE:
Integer

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
```

```
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataDeleteRecord-Methode

Löscht den selektierten Datensatz in der aktuellen Datenbank.

KLASSE:
Desktop

SYNTAX:
DataDeleteRecord()

RÜCKGABE:
Boolean

Beispiel:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
```

```
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataFirstRecord-Methode

Positioniert auf den ersten Datensatz einer Datenbank-Tabelle oder eines aus der aktuellen Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum ersten Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum ersten Datensatz der Datenbank-Tabelle gesprungen.

Bei SQL-Abfragen sollte diese Methode immer direkt nach einer Abfrage ausgeführt werden. Nur dann ist sichergestellt, daß auch alle Datensätze einer Abfrage zur Verfügung stehen.

Siehe auch: DataLastRecord, DataNextRecord, DataPreviousRecord, DataRecord.

KLASSE:
Desktop

SYNTAX:
DataFirstRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
```

```
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataGetTableName-Methode

Liefert den Namen einer Datenbank-Tabelle. Die Datenbank muß bereits im StarOffice angemeldet sein.

KLASSE:
Desktop

SYNTAX:
DataGetTableName(DatabaseName As String, TableNumber As Long)

RÜCKGABE:
String

PARAMETER:
DatabaseName: Name der Datenbank, die die zu erfragende Tabelle enthält.

TableNumber: Nummer der Tabelle, deren Name ermittelt werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
```

```
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataLastRecord-Methode

Positioniert auf den letzten Datensatz einer Datenbank-Tabelle oder eines aus der aktuellen Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum letzten Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum letzten Datensatz der Datenbank-Tabelle gesprungen.

Siehe auch: DataFirstRecord, DataNextRecord, DataPreviousRecord, Data-Record.

KLASSE:
Desktop

SYNTAX:
DataLastRecord()

RÜCKGABE:
Boolean

Beispiel:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
```

```
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataNextRecord-Methode

Springt zum nächsten Datensatz einer Datenbank-Tabelle oder eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum nächsten Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum nächsten Datensatz der Datenbank-Tabelle gesprungen.

Siehe auch: DataFirstRecord, DataLastRecord, DataPreviousRecord, DataRecord.

KLASSE:
Desktop

SYNTAX:
DataNextRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
```

```
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
```

```
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataPhysicalRecordID-Methode

Liefert die aktuelle Datensatznummer eines Datensatzes einer Datenbanktabelle. Die zurückgelieferte Datensatznummer ist abhängig von der verwendeten Abfrage. Bei dem ersten Datensatz einer Abfrage wird immer eine 0 zurückgeliefert, auch wenn es nicht der erste Datensatz der Tabelle ist.

KLASSE:
Desktop

SYNTAX:
DataPhysicalRecordID(GetUserRecordId As Boolean)

RÜCKGABE:
Long

PARAMETER:
[GetUserRecordID]: Dieser Parameter wird nur intern vom Programm ausgewertet und sollte nicht übergeben werden.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
```

```
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
```

```
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataPreviousRecord-Methode

Springt zum vorangehenden Datensatz einer Datenbank-Tabelle oder eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum vorigen Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum vorigen Datensatz der Datenbank-Tabelle gesprungen.

Siehe auch: DataNextRecord, DataFirstRecord, DataLastRecord, DataRecord.

KLASSE:
Desktop

SYNTAX:
DataPreviousRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
```

```
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataRecord-Methode

Springt zu dem angegebenen Datensatz einer Datenbank-Tabelle oder eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum angegebenen Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum angegebenen Datensatz der Datenbank-Tabelle gesprungen.

Siehe auch: DataFirstRecord, DataLastRecord, DataNextRecord, DataPreviousRecord.

KLASSE:
Desktop

SYNTAX:
DataRecord(RecordNumber As Long)

RÜCKGABE:
Boolean

PARAMETER:
RecordNumber: Ein beliebiger Long-Integer-Ausdruck, der die Nummer des Datensatzes angibt, zu dem gesprungen werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
```

```
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataRecordID-Methode

Gibt die Nummer des aktuellen Datensatzes in dem aus der aktuellen Datenbank generierten SQL-Abfrageergebnis zurück.

KLASSE:
Desktop

SYNTAX:
DataRecordID(GetUserRecordId As Boolean)

RÜCKGABE:
Long

PARAMETER:
[GetUserRecordID]: Dieser Parameter wird nur intern vom Programm ausgewertet und sollte nicht übergeben werden.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
```

```
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataSwitch-Methode

Schaltet auf eine andere, bereits geöffnete Datenverbindung um. Wurde der Befehl erfolgreich ausgeführt, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
DataSwitch(DBName As String, TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
DBName: Name der Datenbank, die aktiviert werden soll.

TableName: Name der Tabelle, zu der umgeschaltet werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
```

```
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataTableNumber-Methode

Liefert die Anzahl der Tabellen einer im StarOffice angemeldeten Datenbank.

KLASSE:
Desktop

SYNTAX:
DataTableNumber(DatabaseName As String)

RÜCKGABE:
Integer

PARAMETER:
DatabaseName: Name der Datenbank, deren Tabellenanzahl ermittelt werden soll.

Beispiel:

```
Sub ExampleOpenDatabaseConnection
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Application.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Application.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print "IsTableReadonly liefert : " &  Application.IsTableRea-
dOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Application.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Application.GetDataba-
seLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Application.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Application.DataCo-
lumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Application.DataCo-
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord  REM Spingt zum nächsten
Datensatz
```

```
vStatus = Application.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Application.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Application.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Application.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Application.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
     vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End if
End if
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DefaultFilePath-Eigenschaft

Bestimmt oder ermittelt den Vorgabepfad für alle Dokument-Dateidialoge.

KLASSE:
Application

SYNTAX:
DefaultFilePath

RÜCKGABE:
String

PARAMETER:
[= Path]: Beliebiger Zeichenfolgenausdruck, der den Pfadnamen als Vorgabepfad für alle Dateidialoge beinhaltet.

BEISPIEL:

```
Sub ExampleDefaultFilePath
Const sMessage = "Der Vorgabepfad lautet : "
Print sMessage & Application.DefaultFilePath
End Sub
```

DeleteDatabaseConnection-Methode

Meldet eine im StarOffice angemeldete Datenbank wieder ab. Dabei wird der Datenbank-Eintrag inklusive aller erstellten Abfragen, Berichte und Formulare aus dem Explorer entfernt.

Siehe auch: AddDatabaseConnection

KLASSE:
Desktop

SYNTAX:
DeleteDatabaseConnection(DBName As String)

PARAMETER:
DBName: Ein Zeichenfolgenausdruck, der die Bezeichnung der Datenbank aufnimmt, die abgemeldet werden soll.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
```

```
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
```

```
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DeleteForm-Methode

Löscht ein Formular einer im StarOffice angemeldeten Datenbank.

KLASSE:
Desktop

SYNTAX:
DeleteForm(FormName As String)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Name des Formulares, das gelöscht werden soll.

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig.
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt.
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
```

```
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).TopWindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = ActiveWindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

DeleteFrame-Methode

Entfernt einen Frame aus einem FrameSet.

KLASSE:
FrameSetView

SYNTAX:
DeleteFrame(Frame As String)

PARAMETER:
Frame: Name des Frames, der gelöscht werden soll.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
```

```
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
```

```
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

DeleteIndex-Methode

Löscht einen Index der gerade geöffneten Datenbank-Tabelle.

KLASSE:
Desktop

SYNTAX:
DeleteIndex(IndexName As String)

RÜCKGABE:
Boolean

PARAMETER:
IndexName: Name des Index, der gelöscht werden soll.

BEISPIEL:

```
Sub ExampleDatabaseIndex
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Dim stDataRow As New Strings
Dim stIndexStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
stIndexStrings.Add sTextName & "1,TRUE"
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
vStatus = Application.CreateIndex(sTabName, sIndexName, stIn-
dexStrings, FALSE)
vStatus = Application.DeleteIndex(sIndexName)
Application.CloseTableConnection(sTabName)
```

```
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DeleteQuery-Methode

Löscht eine Datenbank-Abfrage einer im StarOffice angemeldeten Datenbank.

KLASSE:
Desktop

SYNTAX:
DeleteQuery(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage, die gelöscht werden soll.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
```

```
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
```

```
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DeleteReport-Methode

Löscht einen Bericht einer im StarOffice angemeldeten Datenbank.

Siehe auch: CreateReport

KLASSE:
Desktop

SYNTAX:
DeleteReport(ReportName As String)

RÜCKGABE:
Boolean

PARAMETER:
ReportName: Name des Berichts, der gelöscht werden soll.

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt
```

```
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

DeleteTable-Methode

Löscht eine Datenbank-Tabelle einer im StarOffice angemeldeten Datenbank. Die Datenbank muß dabei geöffnet sein.

> Diese Methode funktioniert nur dann, wenn keine Verbindung zu der zu löschenden Tabelle besteht. Ggf. ist es notwendig, vorher eine bestehende Verbindung mit der CloseTableConnection-Methode zu schließen.

KLASSE:
Desktop

SYNTAX:
DeleteTable(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Tabelle, die aus der Datenbank gelöscht werden soll.

Beispiel:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
```

```
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

Description-Eigenschaft

Bestimmt oder ermittelt die Beschreibung des Dokumentinhalts aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Description

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Beschreibung des Dokuments angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
```

```
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

DesignerDialog-Eigenschaft

Bestimmt oder ermittelt, ob der Stylist eingeblendet wird.

KLASSE:
Application

SYNTAX:
DesignerDialog

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Stylist ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleDesignerDialog
Documents.Add("sWriter4")
Application.DesignerDialog = TRUE
End Sub
```

Dictionary-Eigenschaft

Bestimmt oder ermittelt den Pfad für Wörterbücher. Dieser wird unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
Dictionary

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad für Wörterbücher angibt.

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Distance-Eigenschaft

Bestimmt oder ermittelt den Abstand des Inhalts zur Umrandung.

Diese Funktion steht nur für Textdokumente zur Verfügung.

KLASSE:
Border

SYNTAX:
Distance

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der den Abstand in der Standard-Maßeinheit angibt. Der Wert kann auch als Zeichenfolge mit Angabe der gewünschten Einheit gesetzt werden.

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

Document-Eigenschaft

Ermöglicht den Zugriff auf das Dokument, welches zur Zeit in dem Window dargestellt wird. Diese Eigenschaft kann alternativ dazu verwendet werden, auf Objekte der Klasse Document zuzugreifen. So entspricht beispielsweise "ActiveWindow.Document.Name" in seiner Funktion "ActiveDocument.Name".

KLASSE:
Window

SYNTAX:
Document

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleWindowDocument
Dim oDoc As Object
Documents.Add("sWriter4")
ActiveWindow.NewWindow  REM Ein andere Sicht
ActiveDocument.PrintPreview=TRUE
oDoc=ActiveWindow.Document
MsgBox oDoc.Name,64,"Der Name des Dokumentes ist unabhängig von
der Sicht."
ActiveDocument.Close "",False
Do While ActiveDocument.Closing REM Warten bis Dokument fertig
geschlossen wurde.
   Wait 10
Loop
End Sub
```

DocumentInfo-Eigenschaft

Liefert die aktuellen Dokumenteigenschaften eines Dokumentes als DokumentInfo-Objekt zurück.

KLASSE:
OfficeDocument

SYNTAX:
DocumentInfo

RÜCKGABE:
Object

Beispiel:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schliessen mit Spei-
chern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

DocumentManager-Methode

Öffnet den Dokument-Manager.

KLASSE:
Application

SYNTAX:
DocumentManager()

BEISPIEL:

```
Sub ExampleDocumentManager
Application.DocumentManager
End Sub
```

Documents-Eigenschaft

Verweist auf die globale Collection aller offenen Dokumente.

KLASSE:
Application

SYNTAX:
Documents

RÜCKGABE:
Object

BEISPIEL:

```
sub ExampleDocuments
Dim sText As String,iCount As Integer
For iCount =1 to Application.Documents.Count
sText = sText & Application.Documents.Item(iCount).Name &
Chr(13)
Next iCount
MsgBox sText,0,"Dokumentenliste" REM Ausgabe der Namen aller
geöffneten Dokumente
end sub
```

Eigenschaften und Methoden mit "E"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "E" beginnen.

EditTime-Eigenschaft

Ermittelt die Uhrzeit, zu der das Dokument zuletzt gespeichert wurde. Diese wird aus den Dokumenteigenschaften gelesen.

KLASSE:
DocumentInfo

SYNTAX:
EditTime

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
```

```
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

EMail-Eigenschaft

Bestimmt oder ermittelt die E-Mail-Adresse aus den Benutzerdaten. Sie wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
EMail

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die E-Mail-Adresse aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
```

```
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

EnterWait-Methode

Schaltet den Warte-Mauszeiger (Sanduhr) ein. Dabei wird ein interner Zähler um 1 erhöht. Ist der Zähler größer 0, dann ist der Warte-Mauszeiger eingeschaltet.

Siehe auch: LeaveWait

KLASSE:
Desktop

SYNTAX:
EnterWait()

Beispiel:

```
Sub ExampleWaitCursor
Const sMessage = "Es ist jetzt der Wartecursor eingeschaltet"
Documents.Add( "sWriter4" )
Application.EnterWait
If Application.IsWait = TRUE Then
Selection.Insert sMessage
End If
Wait 3000
Application.LeaveWait
Selection.GotoStartOfLine TRUE
Selection.Delete
End Sub
```

Explorer-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption des Explorers.

Klasse:
Application

Syntax:
Explorer

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Explorer ein- (True) oder ausgeblendet (False) ist.

Beispiel:

```
Sub ExampleExplorer
Application.Explorer = TRUE
Wait 1000
Application.Explorer = FALSE
End Sub
```

ExtendedHelp-Eigenschaft

Bestimmt oder ermittelt, ob die kontextsensitive Hilfe aktiv ist.

Klasse:
Application

Syntax:
ExtendedHelp

Rückgabe:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die kontextsensitive Hilfe ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleExtendedHelp
Application.ExtendedHelp = TRUE
End Sub
```

Eigenschaften und Methoden mit "F"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "F" beginnen.

Fax-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten die Fax-Nummer. Diese wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Fax

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Fax-Nummer aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
```

```
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Field1-Eigenschaft

Bestimmt oder ermittelt den Inhalt des ersten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field1

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Inhalt des Infofeldes angibt.

Beispiel:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field1Title-Eigenschaft

Bestimmt oder ermittelt den Namen des ersten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field1Title

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Namen des Feldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
```

```
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field2-Eigenschaft

Bestimmt oder ermittelt den Inhalt des zweiten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field2

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Inhalt des Infofeldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
```

```
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field2Title-Eigenschaft

Bestimmt oder ermittelt den Namen des zweiten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field2Title

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Namen des Feldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
```

```
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field3-Eigenschaft

Bestimmt oder ermittelt den Inhalt des dritten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field3

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Inhalt des Infofeldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
```

```
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field3Title-Eigenschaft

Bestimmt oder ermittelt den Namen des dritten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field3Title

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Namen des Feldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
```

```
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field4-Eigenschaft

Bestimmt oder ermittelt den Inhalt des vierten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field4

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Inhalt des Infofeldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
```

```
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Field4Title-Eigenschaft

Bestimmt oder ermittelt den Namen des vierten Infofeldes aus den Dokumenteigenschaften.

KLASSE:
DocumentInfo

SYNTAX:
Field4Title

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Namen des Feldes angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

FileDialog-Methode

Öffnet einen Datei-Dialog, um eigene Dateien zu öffnen oder zu speichern. Diese Methode blendet den Dialog lediglich ein. Nach dem Schließen des Dialoges liefert FileDialog einen String mit Dateinamen und Pfad zurück. Bricht der Anwender den Dialog ab, wird ein Leerstring zurückgeliefert.

KLASSE:
Desktop

SYNTAX:
FileDialog(Flags As String, Title As String, Path As String)

RÜCKGABE:
String

PARAMETER:
Flags: Zeichenfolge, die bestimmt, ob der Dialog zum Öffnen oder zum Speichern von Dateien oder Grafiken dienen soll. In Abhängigkeit von dem übergebenen Parameter werden folgende System-Dialoge angezeigt:

O - Der Betriebssystem-Dialog ÖFFNEN

S - Der Betriebssystem-Dialog SPEICHERN UNTER

OG - Der Dialog GRAFIK EINFÜGEN

OS - Der Dialog GRAFIK SPEICHERN

Title: Zeichenfolge, die den Titel des Datei-Dialoges angibt, der geöffnet werden soll.

Path: Zeichenfolge, die den Startpfad angibt, in dem die Datei geöffnet oder gespeichert werden soll.

BEISPIEL:

```
Sub ExampleFileDialog
Const sTitle_1 = "Wählen Sie eine Datei aus"
Const sTitle_2 = "Wählen Sie eine Grafik aus"
Const sTitle_3 = "In welches Verzeichnis soll gespeichert
werden "
Const sMessage = "Folgende Datei wurde ausgewählt : "
Dim sFileName As String
sFileName = Application.FileDialog( "O", sTitle_1, "c:\" )
Print sMessage & sFileName
sFileName = Application.FileDialog( "OG", sTitle_2, "c:\" )
Print sMessage & sFileName
sFileName = Application.FileDialog( "S", sTitle_3, "c:\" )
Print sMessage & sFileName
End Sub
```

FileDocument-Methode

Speichert das Dokument in dem im Explorer ausgewählten Ordner ab.

KLASSE:
TaskWindow

SYNTAX:
FileDocument()

BEISPIEL:

```
REM Der Ordner muß im Explorer bereits selektiert worden sein.
Activewindow.FileDocument
```

FileName-Eigenschaft

Liefert den Dateinamen des aktuellen Dokuments. Dieser wird aus den Dokumenteigenschaften ermittelt.

KLASSE:
DocumentInfo

SYNTAX:
FileName

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
```

```
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

FillColor-Eigenschaft

Bestimmt oder ermittelt die Hintergrundfarbe des selektierten Objekts.

Die FillColor-Eigenschaft hat für Interior-Objekte nur dann eine Wirkung, wenn zuvor mit der Style-Eigenschaft ein Füllmuster gesetzt worden ist.

Siehe auch: BackColor, Transparent, Style.

KLASSE:
Interior

SYNTAX:
FillColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe für den Hintergrund angibt.

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

FirstName-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten den Vornamen. Dieser wird entweder bei der Programminstallation angegeben oder unter Extras - Optionen - Allgemein... - Benutzerdaten eingetragen.

Klasse:
Address

Syntax:
FirstName

Rückgabe:
String

Parameter:
[= String]: Zeichenfolge, die den Vornamen aus den Benutzerdaten angibt.

Beispiel:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
```

```
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

FocusBeamer-Eigenschaft

Bestimmt oder ermittelt, ob der Beamer fokussiert ist oder nicht. Befindet sich der Fokus beispielsweise auf dem Desktop-Dokument, so wird er mit FocusBeamer auf den Beamer gezogen, ein erneutes Anwenden der Eigenschaft setzt den Eingabe-Fokus wieder zurück auf das Desktop-Dokument. Dabei spielt es keine Rolle, ob FocusBeamer = True oder FocusBeamer = False gesetzt wird. Der ursprüngliche Zustand entscheidet, ob der Fokus auf den Beamer oder das Desktop-Dokument gezogen wird.

Wird diese Eigenschaft abgefragt, dann wird stets der jeweils aktuelle Zustand zurückgeliefert, also True, wenn der Beamer fokussiert ist und False, wenn sich der Fokus nicht auf dem Beamer befindet.

Befindet sich der Beamer im AutoHide-Modus, so wird er durch diese Eigenschaft aktiviert, und der Eingabe-Fokus wird auf den Beamer gesetzt, unabhängig davon, ob FocusBeamer = True oder FocusBeamer = False.

Siehe auch: FocusExplorer

KLASSE:
Application

SYNTAX:
FocusBeamer

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob sich der Eingabe-Fokus auf dem Beamer befindet (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleFocusBeamer
Application.FocusBeamer = TRUE
End Sub
```

FocusExplorer-Eigenschaft

Bestimmt oder ermittelt, ob der Explorer fokussiert ist oder nicht. Befindet sich der Fokus beispielsweise auf dem Desktop-Dokument, so wird er mit FocusExplorer auf den Explorer gezogen, ein erneutes Anwenden der Eigenschaft setzt den Eingabe-Fokus wieder zurück auf das Desktop-Dokument. Dabei spielt es keine Rolle, ob FocusExplorer = True oder FocusExplorer = False gesetzt wird. Der ursprüngliche Zustand entscheidet, ob der Fokus auf den Explorer oder das Desktop-Dokument gezogen wird.

Wird diese Eigenschaft abgefragt, dann wird stets der jeweils aktuelle Zustand zurückgeliefert, also True, wenn der Explorer fokussiert ist und False, wenn sich der Fokus nicht auf dem Explorer befindet.

Befindet sich der Explorer im AutoHide-Modus, so wird er durch diese Eigenschaft aktiviert, und der Eingabe-Fokus wird auf den Explorer gesetzt, unabhängig davon, ob FocusExplorer = True oder FocusExplorer = False.

Siehe auch: FocusBeamer

KLASSE:
Application

SYNTAX:
FocusExplorer

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob sich der Eingabe-Fokus auf dem Explorer befindet (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleFocusExplorer
Application.FocusExplorer = TRUE
End Sub
```

FocusUrlBox-Methode

Zieht den Eingabe-Fokus auf das URL-Eingabefeld in der Funktionsleiste.

KLASSE:
Application

SYNTAX:
FocusUrlBox()

BEISPIEL:

```
Sub ExampleFocusURLBox
Application.FocusURLBox
End Sub
```

FormatColumnValue-Methode

Formatiert einen beliebigen String anhand der Formatierungsregeln der angegebenen Spalte einer Datenbank-Tabelle. Diese muß zuvor mit der OpenTableConnection-Methode geöffnet worden sein.

Siehe auch: OpenTableConnection

KLASSE:
Desktop

SYNTAX:
FormatColumnValue(Value As String, Column As String, Format As Boolean)

RÜCKGABE:
String

PARAMETER:
Value: Beliebige Zeichenfolge, die den Inhalt des Datenbankfeldes angibt, der formatiert werden soll.

Column: Zeichenfolge, die die Spalte angibt, deren Formatierungsregeln angewendet werden sollen. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

[Format]: Optionaler Parameter, der bestimmt, ob eine Formatierung durchgeführt werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, so ist er automatisch auf True gesetzt. Wenn Format = False ist, wird der String ohne Formatierung zurückgegeben.

BEISPIEL:

```
Sub ExampleFormatColumnValue
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sMessage = "Die Zahl  1  sieht jetzt so aus : "
Dim stDataRow As New Strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,SINGLE,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
Application.DataFirstRecord
Application.DataAppendRecord("Test_1,1,Test_2,2")
Application.DataUpdate
Print sMessage & Application.FormatColumnValue(1 , sNumericName
& "1", TRUE )
Application.CloseTableConnection( sTabName )
vStatus = Application.DeleteTable( sTabName )
```

```
Application.CloseDatabaseConnection( sDBName )
Application.DeleteDatabaseConnection( sDBName )
End Sub
```

FrameContent-Eigenschaft

Bestimmt oder ermittelt den Inhalt (URL) eines Frames in einem FrameSet.

Siehe auch: FrameName, SetFrameContent.

KLASSE:
FrameWindow

SYNTAX:
FrameContent

RÜCKGABE:
String

PARAMETER:
[= URL]: Zeichenfolge, die den Inhalt des Frames in Form der URL-Adresse angibt.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
```

```
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next

msgBox sText,0,sHead
end sub
```

FrameCount-Eigenschaft

Liefert bei FrameSet-Dokumenten die Anzahl der Frames in dem Fenster. Dabei zählen nur "direkte Kinder" keine "Enkel", d.h. wenn ein Frame ein Dokument enthält, das weitere Frames nachlädt, zählt zum FrameCount nur der Frame selbst, nicht aber dessen "Kinder".

Bei FrameSet-Dokumenten liefert FrameCount immer mindestens den Wert 1. Bei anderen Dokumenten, z.B. Text- oder Tabellendokumenten, liefert diese Eigenschaft immer den Wert 0.

KLASSE:
Window

SYNTAX:
FrameCount

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
```

```
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

FrameName-Eigenschaft

Bestimmt oder ermittelt den Namen eines Frames in einem FrameSet.

Siehe auch: SetFrameName, FrameContent.

KLASSE:
FrameWindow

SYNTAX:
FrameName

RÜCKGABE:
String

PARAMETER:
[= Name]: Zeichenfolge, die den Namen des Frames angibt.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
```

```
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

FramesetSource-Eigenschaft

Bestimmt oder ermittelt, ob es zu dem FrameSet auch eine Ansicht des HTML-Quelltextes gibt.

KLASSE:
FrameSetView

SYNTAX:
FramesetSource

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der bestimmt, ob eine Ansicht des HTML-Quelltextes existiert (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleCutCopyPaste
Dim sWin As String REM für den Fensternamen
REM TextDokument öffnen
Documents.Add("sWriter4")
sWin = ActiveWindow.Name
REM jetzt wird ein wenig HTML-Code eingegeben
ActiveWindow.Selection.Insert(Chr(60)&"HTML"&Chr(62)&Chr(60)&"B
ODY"&Chr(62)&"This is HTML !"&Chr(60)&"/BODY"&Chr(62))
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut REM HTML-Code ausschneiden
Documents.Add("sWriter4/Web") REM HTML-Dokument öffnen und ...
Wait 100
ActiveDocument.FrameSetSource = True REM ... in die Quelltex-
tansicht ...
ActiveWindow.SelectAll
ActiveWindow.Paste REM .. den HTML-Code einfügen.
ActiveDocument.FrameSetSource = False
REM und zurück ...
ActiveDocument.FrameSetSource = TRUE
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveDocument.FrameSetSource = False
Windows( sWin ).Activate
ActiveWindow.Paste
end sub
```

FrameSpacing-Eigenschaft

Bestimmt oder ermittelt den Abstand zwischen Frames in einem FrameSet.

KLASSE:
FrameSetView

SYNTAX:
FrameSpacing

RÜCKGABE:
Long

PARAMETER:
[= Spacing]: Long-Integer-Ausdruck, der den Abstand in Pixel angibt.

Beispiel:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
```

```
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

FullName-Eigenschaft

Liefert den kompletten Dateinamen des Dokuments inklusive Pfad. Bei Web-Dokumenten wird ein URL zurückgegeben, bei Dokumenten aus dem File-System die normale Dateipfad-Darstellung und bei neuen Dokumenten der Name (z.B. "unbenannt1").

KLASSE:
Document

SYNTAX:
FullName

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleApplication
Const sDoc="TestDoc.sdw"
Documents.Add("sWriter4")
ActiveDocument.SaveAs( Application.PathSettings.Work & "\" &
sDoc )
Selection.Insert("Name= "& ActiveDocument.Name)
Selection.InsertPara
Selection.Insert("Pfad= "& ActiveDocument.Path)
Selection.InsertPara
```

```
Selection.Insert("Pfad mit Name= " &ActiveDocument.FullName)
Selection.InsertPara
End Sub
```

FullName-Eigenschaft

Liefert den kompletten Programmnamen der Applikation inklusive Pfad.

KLASSE:
Application

SYNTAX:
FullName

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleApplicationFullName
Print Application.FullName REM liefert Pfad und Namen der
Applikation
end sub
```

FullScreen-Eigenschaft

Bestimmt oder ermittelt, ob die Arbeitsfläche des dem Objekt entsprechenden StarOffice-Programmfensters maximiert oder im Normalmodus dargestellt wird. Maximieren in diesem Zusammenhang bedeutet die Darstellung der Arbeitsfläche als Vollbild - also ohne Menü-, Funktions-, Objekt- und Statusleiste.

KLASSE:
Application

SYNTAX:
FullScreen

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Fenster im Vollbild- (True) oder im Normalmodus (False) dargestellt wird.

BEISPIEL:

```
Sub ExampleFullSreen
Application.FullScreen = TRUE
Wait 5000
Application.FullScreen = FALSE
End Sub
```

FunctionBarVisible-Eigenschaft

Bestimmt oder ermittelt, ob die Funktionsleiste sichtbar ist.

KLASSE:
Application

SYNTAX:
FunctionBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Funktionsleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleFunctionBarVisible
Application.FunctionBarVisible = FALSE
Wait 1000
Application.FunctionBarVisible = TRUE
End Sub
```

Eigenschaften und Methoden mit "G"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "G" beginnen.

GeneralOptions-Methode

Ruft den Dialog OPTIONEN ALLGEMEIN auf.

KLASSE:
Desktop

SYNTAX:
GeneralOptions()

Beispiel:

```
Sub ExampleGeneralOptions
Application.GeneralOptions
End Sub
```

GetActiveDatabaseConnection-Methode

Liefert den Namen der aktuellen Datenbankverbindung.

Siehe auch: OpenDatabaseConnection

Klasse:
Desktop

Syntax:
GetActiveDatabaseConnection()

Rückgabe:
String

Beispiel:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
```

```
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetActiveTableConnection-Methode

Liefert den Namen der aktuellen Verbindung zu einer Datenbank-Tabelle.

Siehe auch: OpenTableConnection

KLASSE:
Desktop

SYNTAX:
GetActiveTableConnection()

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
```

```
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetAdjust-Methode

Liefert den Typ eines Tabulators als String zurück.

Mögliche Rückgabewerte sind:

L - Links

R - Rechts

C - Zentriert

D - Dezimal

KLASSE:
TabStops

SYNTAX:
GetAdjust(No As Long)

RÜCKGABE:
String

PARAMETER:
No: Nummer des Tabulators, dessen Typ ermittelt werden soll.

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM" ) REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM" )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount = 0 To iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText = sText & "Position: " & s1 & " Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &" Füllzeichen: " & s4 & Chr(13)
Next
End With
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

GetDataRecordCount-Methode

Liefert die Anzahl aller Datensätze einer Datenbank-Tabelle oder eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird die Anzahl der in der Abfrage befindlichen Datensätze zurückgeliefert. Wurde keine Abfrage ausgeführt, dann werden alle Datensätze der aktuellen Datenbank-Tabelle gezählt.

Diese Methode kann bei großen Datenbanken sehr zeitaufwendig werden, da u.U. alle vorhandenen Datensätze einmal gelesen werden.

KLASSE:
Desktop

SYNTAX:
GetDataRecordCount()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
```

```
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetDecimal-Methode

Liefert das verwendete Dezimaltrennzeichen für einen Tabulator als String zurück. Dies ist in der Regel ein Komma (",") oder ein Punkt (".").

KLASSE:
TabStops

SYNTAX:
GetDecimal(No As Long)

RÜCKGABE:
String

PARAMETER:
No: Index des Tabulators, dessen Dezimaltrennzeichen ermittelt werden soll.

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM" ) REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM" )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount = 0 To iTabCount-1
```

```
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText = sText & "Position: " & s1 & " Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &" Füllzeichen: " & s4 & Chr(13)
Next
End With
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

GetFillChar-Methode

Ermittelt das Füllzeichen für einen Tabulator und liefert es als String zurück.

KLASSE:
TabStops

SYNTAX:
GetFillChar(No As Long)

RÜCKGABE:
String

PARAMETER:
No: Index des Tabulators, dessen Füllzeichen ermittelt werden soll.

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM"   )  REM
Punkt und Komma sind relevant
```

```
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM"   )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount =  0 To iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText  = sText &  "Position: " & s1 & "  Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &"  Füllzeichen: " & s4 & Chr(13)
Next
End With
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

GetFormNames-Methode

Liefert eine Liste aller vorhandenen Formularnamen, die an der geöffneten Datenbank definiert sind. Der Rückgabeparameter ist vom Typ Strings.

KLASSE:
Desktop

SYNTAX:
GetFormNames()

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
```

```
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPath = Application.Pathsettings.Database & sFileName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
sStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = Application.DataGetTableName(sDataName, 1)
Documents.Add "sWriter4"
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close True, sPath
Application.CreateForm sFormName, sPath
Application.CreateReport sReportName, sPath
Application.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
stStringsColl = Application.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = Application.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = Application.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
Application.DeleteForm sFormName
Application.DeleteReport sReportName
Application.DeleteQuery sQueryName
End Sub
```

GetFrameWindow-Methode

Liefert das erste FrameWindow mit dem angegebenen Namen rekursiv innerhalb des aktuellen Fensters.

In Verbindung mit FrameCount kann mit GetFrameWindow eine Iteration programmiert werden (siehe Beispiel).

Siehe auch: FrameCount

KLASSE:
Window

SYNTAX:
GetFrameWindow(WindowName As String)

RÜCKGABE:
Object

PARAMETER:
WindowName: Name oder Index des FrameWindows.

Beispiel:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
```

```
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

GetPosition-Methode

Liefert die horizontale Position eines Tabulators im Dokument.

KLASSE:
TabStops

SYNTAX:
GetPosition(No As Long)

RÜCKGABE:
Long

PARAMETER:
No: Nummer bzw. Index des Tabulators, dessen Position ermittelt werden soll.

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
```

```
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM" ) REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM" )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount = 0 To iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText = sText & "Position: " & s1 & " Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &" Füllzeichen: " & s4 & Chr(13)
Next
End With
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

GetQueryColumns-Methode

Liefert die Spaltenbeschreibungen der Datenbank-Abfrage. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: GetTableColumns

KLASSE:
Desktop

SYNTAX:
GetQueryColumns(QueryName As String)

RÜCKGABE:
Object

PARAMETER:
QueryName: Name der Datenbank-Abfrage, deren Spaltenbeschreibung ermittelt werden soll.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
```

```
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetQueryNames-Methode

Liefert die Namen aller Abfragen einer geöffneten Datenbank. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: OpenDatabaseConnection

KLASSE:
Desktop

SYNTAX:
GetQueryNames()

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
```

```
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
```

```
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetReportNames-Methode

Liefert eine Liste aller Berichte der geöffneten Datenbank. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: OpenDatabaseConnection, CreateReport, DeleteReport.

KLASSE:
Desktop

SYNTAX:
GetReportNames()

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPath = Application.Pathsettings.Database & sFileName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
sStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = Application.DataGetTableName(sDataName, 1)
Documents.Add "sWriter4"
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close True, sPath
Application.CreateForm sFormName, sPath
Application.CreateReport sReportName, sPath
```

```
Application.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
stStringsColl = Application.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = Application.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = Application.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
Application.DeleteForm sFormName
Application.DeleteReport sReportName
Application.DeleteQuery sQueryName
End Sub
```

GetSQL_Statement-Methode

Liefert das SQL-Statement einer Datenbank-Abfrage.

KLASSE:
Desktop

SYNTAX:
GetSQL_Statement(QueryName As String)

RÜCKGABE:
String

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
```

```
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
```

```
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetTableColumns-Methode

Liefert die Beschreibung aller Spalten einer Datenbank-Tabelle als Variable vom Datentyp Strings zurück. Die einzelnen Bestandteile der ermittelten Strings-Variablen beschreiben je eine Spalte der Tabelle und sind vom gleichen Format wie bei der CreateTable-Methode ("Feldname, Feldtyp, Feldlänge, Eingabe").

KLASSE:
Desktop

SYNTAX:
GetTableColumns(TableName As String)

RÜCKGABE:
Object

PARAMETER:
TableName: Name der Tabelle, aus der die Werte erfragt werden sollen.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
```

```
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
```

```
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

Glossary-Eigenschaft

Liefert oder setzt den Pfad, in dem nach AutoText-Bausteinen gesucht wird.

KLASSE:
PathSettings

SYNTAX:
Glossary

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad angibt. Die Angabe mehrerer Verzeichnisse, durch Semikolon getrennt, ist zulässig.

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Graphics-Eigenschaft

Liefert oder setzt den Pfad für Grafiken. Dieser wird unter Extras - Optionen - Allgemein... - Pfade eingestellt.

Klasse:
PathSettings

SYNTAX:
Graphics

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad für Grafiken angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

GraphicsIndep-Eigenschaft

Bestimmt oder ermittelt, ob Grafiken in Dokumenten plattformunabhängig gespeichert werden.

Beachten Sie, daß bei plattformunabhängigem Speichern mehr Speicherplatz benötigt wird.

KLASSE:
SaveSettings

SYNTAX:
GraphicsIndep

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Grafiken plattformunabhängig gespeichert werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
```

```
"mit gespeichert."
.InsertPara
End With
End Sub
```

GrfFilename-Eigenschaft

Setzt oder ermittelt Pfad und Dateinamen einer Hintergrundgrafik, welche für den Hintergrund als verknüpfte Grafik eingefügt worden ist.

Siehe auch: GrfFilername, GrfPosition.

KLASSE:
Interior

SYNTAX:
GrfFilename

RÜCKGABE:
String

PARAMETER:
[= FileName]: Zeichenfolgenausdruck, der Pfad und Dateinamen der verknüpften Hintergrundgrafik angibt.

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.BackColor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

GrfFiltername-Eigenschaft

Setzt oder ermittelt den Namen des Dateifilters einer Hintergrundgrafik.

Siehe auch: GrfFilename, GrfPosition.

KLASSE:
Interior

Syntax:
GrfFiltername

Rückgabe:
String

Parameter:
[= FilterName]: Zeichenfolgenausdruck, der den Namen des Dateifilters angibt.

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.BackColor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

GrfPosition-Eigenschaft

Setzt oder ermittelt die Position einer Hintergrundgrafik.

Siehe auch: GrfFilename, GrfFiltername.

Klasse:
Interior

Syntax:
GrfPosition

Rückgabe:
Long

Parameter:
[= Position]: Long-Integer-Ausdruck, der die Position der Hintergrundgrafik angibt. Folgende Werte sind möglich:

0 - Keine Grafik

1 - Oben links

2 - Oben horizontal zentriert

3 - Oben rechts

4 - Links vertikal zentriert

5 - Vertikal und horizontal zentriert

6 - Rechts vertikal zentriert

7 - Unten links

8 - Unten horizontal zentriert

9 - Unten rechts

10 - Über die gesamte Fläche

11 - Gekachelt

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.BackColor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

Eigenschaften und Methoden mit "H"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "H" beginnen.

Height-Eigenschaft

Bestimmt oder ermittelt die Höhe des Fensters, das durch das entsprechende Objekt bestimmt wird.

Siehe auch:Width, Left, Top.

KLASSE:
TaskWindow, Application

SYNTAX:
Height

RÜCKGABE:
Long

PARAMETER:
[= Height]: Beliebiger Long-Integer-Ausdruck, der die neue Höhe des Fensters in Pixel festlegt.

BEISPIEL:

```
Sub ExampleDimension
with Application
   .Height = 300
   .Width = 400
   .Left = 200
   .Top = 100
   wait 5000
   .Maximized=TRUE
end with
```

Help-Methode

Startet die kontextsensitive Hilfe. Wird kein Parameter übergeben oder ist die Hilfedatei eine SVH-Datei, dann wird das von Star Division verwendete HTML-Hilfeformat verwendet. Ansonsten wird die für das Betriebssystem spezifische Hilfe geladen.

KLASSE:
Application

SYNTAX:
Help(helpfile As String, helpcontextID As Long, helpstring As String)

PARAMETER:
[HelpFile]: Name der Hilfedatei, die den Text der kontextsensitiven Hilfe enthält.

[HelpContextID]: Integer-Ausdruck, der die kontextsensitive Hilfe als Hilfe-ID identifiziert. Dieser Parameter ist optional.

[HelpString]: Zeichenfolge, die der kontextsensitiven Hilfe zugewiesen ist und zusätzlich zur Hilfe-ID angegeben werden kann.

Die Angaben von HelpContextID und HelpString sind optional. Läßt man diese beiden Parameter weg, erhält man den Index der Hilfedatei. Ferner erhält man die Standard-Hilfedatei, wenn man keinen aktuellen Parameter angibt oder als HelpFile einen Leerstring übergibt.

Beispiel:

```
Sub ExampleHelp
REM Hilfe für Deutsch=49
Application.Help(Application.Pathsettings.Modules
&"/help/49/shelp.svh")
End Sub
```

HelperDialog-Eigenschaft

Bestimmt oder ermittelt, ob der Help Agent eingeblendet wird.

Klasse:
Application

Syntax:
HelperDialog

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Help Agent ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
Sub ExampleHelperDialog
Application.HelperDialog = True
End Sub
```

HelpIndex-Methode

Zeigt den Hilfe-Index in der Online-Hilfe an. Das heißt, es wird die Online-Hilfe der jeweils aktuellen Applikation gestartet.

Klasse:
Application

Syntax:
HelpIndex()

Beispiel:

```
Sub ExampleHelpIndex
Application.HelpIndex
End Sub
```

HelpOnHelp-Methode

Zeigt Hilfe über das Hilfesystem in der Online-Hilfe an.

KLASSE:
Application

SYNTAX:
HelpOnHelp()

BEISPIEL:

```
Sub ExampleHelpOnHelp
Application.HelpOnHelp
End Sub
```

HelpTip-Eigenschaft

Bestimmt oder ermittelt, ob die Tip-Hilfe aktiv ist.

KLASSE:
Application

SYNTAX:
HelpTip

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Tip-Hilfe ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleHelpTip
Application.HelpTip = TRUE
End Sub
```

HorizontalLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die horizontale Linie in Tabellen beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

Siehe auch: VerticalLine

KLASSE:
Border

SYNTAX:
HorizontalLine

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

Eigenschaften und Methoden mit "I"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "I" beginnen.

ID-Eigenschaft

Bestimmt oder ermittelt das Kürzel aus den Benutzerdaten. Es wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

RÜCKGABE:
String

PARAMETER:
[= ID]: Zeichenfolge, die das Kürzel aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
```

```
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

InSelection-Eigenschaft

Bestimmt oder ermittelt, ob die Suche nur innerhalb eines markierten Bereichs erfolgen soll. Diese Funktion entspricht der Suchoption NUR IN SELEKTION im Dialog SUCHEN & ERSETZEN.

Bei erfolgreicher Suche wird die vorher aufgespannte Selektion aufgelöst. Nach der Suche befindet sich nur die gefundene Zeichenkette in der Selektion.

KLASSE:
SearchSettings

SYNTAX:
InSelection

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob nur innerhalb der Selektion gesucht werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleBackward
Const sText="Text" REM Text, der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GotoEndOfDoc TRUE
Wait 2000
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.InSelection = TRUE
ActiveWindow.SearchSettings.CaseSensitive = TRUE
ActiveWindow.SearchSettings.WordOnly = TRUE
ActiveWindow.Search sText
MsgBox Selection.Value,64,"Suchen und Finden"
End Sub
```

Insert-Methode

Mit dieser Methode wird ein neuer Tabulator eingefügt.

KLASSE:
TabStops

SYNTAX:
Insert(Pos As Long, Adjust As String, Decimal As String, Fill As String)

PARAMETER:
Pos: Gibt die Position des Tabulators in Dokumentmaßeinheiten an.

[Adjust]: Gibt die Anordnung des Tabulators an. Folgende Werte sind möglich:

L - Links

R - Rechts

C - Zentriert

D - Dezimal

Wird dieser Parameter nicht gesetzt, wird der Tabulator linksbündig angeordnet.

[Decimal]: Bestimmt das Trennzeichen für Dezimalzeichen (Punkt, Komma). Wird dieser Parameter nicht gesetzt, wird das Komma als Trennzeichen genommen.
[Fill]: Bestimmt das Füllzeichen für die Spalte (z.B. ...). Wird dieser Parameter nicht gesetzt, wird ein Leerstring übergeben.

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9) REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops REM erst einmal
drei Tabulatoren setzen
REM rechtsbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabulator, mit Komma als dezimalem Trennzeichen
und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
End With
With ActiveWindow.Selection
```

```
REM Einfacher als jeden Absatz zu formatieren, ist es,
REM eine Vorlage zu definieren und anzuwenden.
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM" ) REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM" )
.InsertPara
.StyleApply(sBrandNewStyle,2)
End With
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
For iCount = 0 To iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText = sText & "Position: " & s1 & " Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &" Füllzeichen: " & s4 & Chr(13)
Next
End With
MsgBox(sText,64,"" & iTabCount & " Tabulatoren") REM Hier die
Tabulatoren ausgeben
End Sub
```

InsertHyperlink-Methode

Schaltet die Hyperlinkleiste in einem Toggle-Verhalten ein oder aus. Toggle-Verhalten bedeutet, daß die Funktion vom momentanen Zustand der Hyperlinkleiste abhängt: Ist sie ausgeschaltet, wird sie von der Methode eingeschaltet und umgekehrt.

KLASSE:
Desktop

SYNTAX:
InsertHyperlink()

BEISPIEL:

```
Sub ExampleInsertHyperlink
Application.InsertHyperlink
Wait 3000
Application.InsertHyperlink
End Sub
```

Interactive-Eigenschaft

Bestimmt oder ermittelt, ob die Möglichkeit zur Bedienung der Applikation aktiv ist.

Achten Sie immer darauf, die Interactive-Eigenschaft wieder einzuschalten, nachdem Sie sie deaktiviert haben, da Sie andernfalls nicht in die IDE-Umgebung zurückkehren können.

KLASSE:
Application

SYNTAX:
Interactive

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Möglichkeit zur Bedienung der Applikation ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleInteractive
Const sText = "Während das hier geschrieben wird, kann der
Benutzer nichts machen."
Dim iCount As Integer
Application.Interactive = FALSE
Documents.Add("sWriter4")
For iCount =1 To Len(sText ) Step 3
Selection.Insert( mid(sText,iCount,3))
Wait 1
Next iCount
Application.Interactive = TRUE
End Sub
```

InternetDialog-Methode

Ruft den Dialog OPTIONEN INTERNET auf. Dieser Dialog ist modal, d.h. das Makro wartet solange bis der Dialog vom Anwender geschlossen wurde.

KLASSE:
Desktop

SYNTAX:
InternetDialog()

BEISPIEL:

```
Sub ExampleInternetDialog
Application.InternetDialog
End Sub
```

InternetOnline-Eigenschaft

Bestimmt oder ermittelt, ob der Online-Modus aktiv ist.

Befindet man sich im Online-Modus, dann ist das Browsen im Internet und das Lesen von Mails und News online möglich. Auch bei indirektem Bedarf, z.B. beim Versenden einer Mail, dürfen Verbindungen ins Internet aufgebaut werden. Im Offline-Modus dagegen werden solche Verbindungen unterdrückt, es können aber ggf. Daten aus dem Cache gelesen werden. Befindet man sich im Offline-Modus, dann kann bei direktem Bedarf allerdings trotzdem eine Internet-Verbindung aufgebaut werden, z.B. wenn im Kontextmenü des Ausgangskorbs der Befehl AKTUALISIEREN gewählt wird.

KLASSE:
Application

SYNTAX:
InternetOnline

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob man im Internet browst (True) oder im eigenen Cache (False).

BEISPIEL:

```
Sub ExampleInternetOnline
MsgBox "Du bist" & IIF(Application.InternetOnline," nicht","")
& " online",64,"Online Status"
end sub
```

InternetUpdate-Methode

Holt neue Daten vom Internet und versendet neu erstellte Inhalte. Diese Methode wirkt sich zur Zeit nur auf Mails und News aus.

KLASSE:
Application

SYNTAX:
InternetUpdate()

BEISPIEL:

```
Sub ExampleInternetUpDate
Application.InternetUpDate
End Sub
```

IsFirstRecord-Methode

Ermittelt, ob der aktuelle Datensatz auch der erste Datensatz einer Datenbank-Tabelle ist. Ist dies der Fall, liefert diese Methode den Wert True zurück, ansonsten False.

Um diese Methode anwenden zu können, muß die Datenbank-Tabelle geöffnet sein.

Siehe auch: OpenTableConnection, IsLastRecord.

KLASSE:
Desktop

SYNTAX:
IsFirstRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
```

```
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
```

```
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsLastRecord-Methode

Ermittelt, ob der aktuelle Datensatz auch der letzte Datensatz einer Datenbank-Tabelle ist. Ist das der Fall, liefert diese Methode den Wert True zurück, ansonsten False.

Um diese Methode anwenden zu können, muß die Datenbank-Tabelle geöffnet sein.

Siehe auch: OpenTableConnection, IsFirstRecord.

KLASSE:
Desktop

SYNTAX:
IsLastRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
```

```
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
```

```
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsLoading-Eigenschaft

Ermittelt, ob ein Dokument noch dabei ist, geladen zu werden. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

Enthält das Dokument Grafiken, so kann es vorkommen, daß IsLoading bereits False zurückliefert, obwohl noch nicht alle Grafiken vollständig geladen worden sind. Benutzen Sie zur Abfrage, ob ein Dokument vollständig inklusive Grafiken geladen worden ist, zusätzlich die IsLoadingImages-Eigenschaft.

Ein nicht vollständig geladenenes Dokument darf nicht verändert werden.

Siehe auch: IsLoadingImages

KLASSE:
OfficeDocument

SYNTAX:
IsLoading

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleIsLoading
   Dim iState As Integer
   Documents.Open("http://www.stardivision.com")
   Print "Hello"
   iState = WaitUntilComplete(100)
   If iState = 0 then
      MsgBox("Die StarDivision HomePage wurde noch nicht voll-
ständig geladen.",32,"StarOffice")
   Else
      MsgBox("Das Laden der Homepage ist
beendet.",32,"StarOffice")
   End If
End Sub
Function WaitUntilComplete(iTime As Integer) As Integer
REM Wartet bis das Dokument komplett geladen wurde.
REM iTime ist dabei die Maximale Wartezeit in 0.5 Sek
REM Ist iTime = 0 dann wird bis zuende gewartet.
REM Die Funktion gibt die Wartezeit zurück und bei Ablauf der
Zeit eine 0
Dim bOk As Boolean ,iCount As Integer
iCount = 1
If iTime = 0 Then iTime = 32000 REM Defaultwert
For iCount = 1 to iTime
   bOk = ActiveDocument.IsLoading OR ActiveDocument.IsLoadingI-
mages
   If  bOk Then
      Wait 500
      iCount = iCount + 1
   Else
      Exit for
   End If
Next iCount
If  bOK Then
   WaitUntilComplete = 0
Else
```

```
   WaitUntilComplete = iCount
End If
End Function
```

IsLoadingImages-Eigenschaft

Ermittelt, ob die Grafiken eines Dokuments bereits vollständig geladen worden sind. Solange die Grafiken noch nicht vollständig geladen worden sind, wird der Wert True zurückgegeben, andernfalls False.

Zusammen mit der IsLoading-Eigenschaft stellt die IsLoadingImages-Eigenschaft sicher, daß ein Dokument dann vollständig geladen worden ist, wenn beide Eigenschaften den Wert False zurückliefern.

Siehe auch: IsLoading

KLASSE:
OfficeDocument

SYNTAX:
IsLoadingImages

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleIsLoading
   Dim iState As Integer
   Documents.Open("http://www.stardivision.com")
   Print "Hello"
   iState = WaitUntilComplete(100)
   If iState = 0 then
      MsgBox("Die StarDivision HomePage wurde noch nicht voll-
ständig geladen.",32,"StarOffice")
   Else
      MsgBox("Das Laden der Homepage ist
beendet.",32,"StarOffice")
   End If
End Sub
Function WaitUntilComplete(iTime As Integer) As Integer
REM Wartet bis das Dokument komplett geladen wurde.
REM iTime ist dabei die Maximale Wartezeit in 0.5 Sek
REM Ist iTime = 0 dann wird bis zuende gewartet.
REM Die Funktion gibt die Wartezeit zurück und bei Ablauf der
Zeit eine 0
Dim bOk As Boolean ,iCount As Integer
iCount = 1
If iTime = 0 Then iTime = 32000 REM Defaultwert
```

```
For iCount = 1 to iTime
   bOk = ActiveDocument.IsLoading OR ActiveDocument.IsLoadingI-
mages
   If  bOk Then
      Wait 500
      iCount = iCount + 1
   Else
      Exit for
   End If
Next iCount
If  bOK Then
   WaitUntilComplete = 0
Else
   WaitUntilComplete = iCount
End If
End Function
```

IsPrinting-Eigenschaft

Ermittelt, ob das dem Objekt entsprechende Dokument zur Zeit ausgedruckt wird. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
OfficeDocument

SYNTAX:
IsPrinting

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleIsPrinting
Documents.Add( "sWriter4" )
Selection.Insert "Dieses Dokument wird ausgedruckt"
ActiveDocument.PrintOut( 1, 1 )
Do Until ActiveDocument.IsPrinting
Wait 300
Loop
End Sub
```

IsQueryNative-Methode

Ermittelt, ob eine Datenbank-Abfrage eine Native SQL ist. Ist das der Fall, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
IsQueryNative(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
```

```
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsQueryReadOnly-Methode

Gibt an, ob eine Datenbank-Abfrage verändert werden kann. Ist sie schreibgeschützt, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
IsQueryReadOnly(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
```

```
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = Application.CreateTable(sTabName, stDataRow)
vStatus = Application.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = Application.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = Application.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> Application.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = Application.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = Application.DataFirstRecord
If Application.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = Application.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & Application.DataColumnType(3)
Print sMessage_4 & Application.DataColumnSubType(3)
Application.DataLastRecord
If Application.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = Application.DataDeleteRecord
Print sMessage_6 & Application.DataColumnNumber & sMessage_6a
vStatus = Application.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = Application.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & Application.IsQueryNative(sQueryName)
Application.CloseQueryConnection(sQueryName)
vStatus = Application.OpenQueryConnection(sQueryName)
stQueryRows = Application.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & Application.IsQueryReadOnly(sQueryName)
sSQLStatement = Application.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
Application.DeleteQuery sQueryName
```

```
End If
Application.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = Application.DeleteTable(sTabName)
Application.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsTableReadOnly-Methode

Gibt an, ob eine Datenbank-Tabelle verändert werden kann. Ist sie schreibgeschützt, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
IsTableReadOnly(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Tabelle.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
```

```
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

IsTopWindow-Eigenschaft

Ermittelt, ob das aktuelle Fenster ein Top-Level-Fenster ist. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

Top-Level-Fenster sind übergeordnete Fenster oder "Hauptfenster" der Applikation und stellen im StarBasic die Objekte der Klasse TopWindow dar. Ein Beispiel für Top-Level-Fenster oder TopWindow-Objekte sind MDI-Fenster (MDI = Multiple Document Interface). Solche Fenster spielen bei FrameSets eine Rolle, daher ist die Eigenschaft IsTopWindow gerade bei FrameSets von Bedeutung.

KLASSE:
Window

SYNTAX:
IsTopWindow

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleTopWindow
Documents.Open("http://www.stardivision.de")
ActiveWindow.ActiveFrameWindow.Activate REM Den fokussierten
Frame aktivieren.
MsgBox IIf(ActiveWindow.IsTopWindow,"Ja","Nein"),0,"Ist das
aktive Fenster ein Hauptfenster?" REM Ein einzelner Frame ist
kein TopWindow.
MsgBox ActiveWindow.ActiveFrameWindow.Name,0,"Name des aktiven
Frame" REM Der aktive Frame
MsgBox ActiveWindow.TopWindow.Name,0,"Name des Hauptfensters"
REM Das Hauptfenster
End Sub
```

IsWait-Methode

Ermittelt, ob ein Warte-Mauszeiger angezeigt wird. Ist das der Fall, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
IsWait()

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleWaitCursor
Const sMessage = "Es ist jetzt der Wartecursor eingeschaltet"
Documents.Add( "sWriter4" )
Application.EnterWait
If Application.IsWait = TRUE Then
Selection.Insert sMessage
End If
Wait 3000
Application.LeaveWait
Selection.GotoStartOfLine TRUE
Selection.Delete
End Sub
```

Item-Methode

Ermöglicht den Zugriff auf ein Objekt in einer Collection. Das Objekt erhält man als Rückgabewert. Der Typ des Objekts hängt davon ab, um welche Collection es sich handelt.

KLASSE:
Collection

SYNTAX:
Item(Index As String)

RÜCKGABE:
Object

PARAMETER:
Index: Gibt den Index oder den Namen des gewünschten Objekts in der Collection an. Der Index kann als Zeichenfolge oder als Integer-Ausdruck übergeben werden.

BEISPIEL:

```
sub ExampleAppWindows
Dim sText As String,iCount As Integer
For iCount =1 to Application.Windows.Count
sText = sText & Application.Windows.Item(iCount).Name & Chr(13)
Next iCount
MsgBox sText,0,"Fensterliste" REM Ausgabe der Namen aller
geöffneten Fenster
end sub
```

Eigenschaften und Methoden mit "K"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "K" beginnen.

KeyWords-Eigenschaft

Bestimmt oder ermittelt Schlüsselwörter des Dokuments. Diese werden aus den Dokumenteigenschaften gelesen oder bei diesen eingetragen.

Diese Eigenschaft sollte nur an einem DocumentInfo-Objekt aufgerufen werden. Sie ist lediglich für OLE-Automation auch für die Klasse OfficeDocument verfügbar.

KLASSE:
DocumentInfo, OfficeDocument

SYNTAX:
Keywords

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Schlüsselworte angibt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
```

```
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Eigenschaften und Methoden mit "L"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "L" beginnen.

LanguageId-Eigenschaft

Ermittelt die Sprache, in der das Produkt seinen Text ausgibt. Ein Umstellen wirkt erst nach erneutem Programmstart.

Folgende Rückgabewerte gibt es:

255 - Keine

1030 - Dänisch

1031 - Deutsch

2055 - Deutsch (CH)

2057 - Englisch (UK)

1033 - Englisch (US)

1035 - Finnisch

1036 - Französisch

1040 - Italienisch

1043 - Niederländisch

1044 - Norwegisch Bokmal

2068 - Norwegisch Nynorsk

2070 - Portugiesisch

1053 - Schwedisch

1034 - Spanisch

Siehe auch: Language, CountryID.

KLASSE:
Desktop

SYNTAX:
LanguageId

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleLanguageID
Print "Die LanguageID dieses Systems lautet : " &
Application.LanguageID
End Sub
```

LastName-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten den Nachnamen. Dieser wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
LastName

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Nachnamen aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
```

```
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

LateInitDone-Eigenschaft

Diese Eigenschaft liefert den Wert True sobald alle Programminitialisierungen durchgeführt sind.

KLASSE:
Application

SYNTAX:
LateInitDone

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleLateInitDone
If Application.LateInitDone Then MsgBox("Ist
beendet.",48,"Initialisierung des StarOffice")
End Sub
```

LeaveWait-Methode

Schaltet den Zähler für die Anzeige des Warte-Mauszeigers um eins herunter. Wird der Wert 0 erreicht, wird kein Warte-Mauszeiger mehr angezeigt.

Siehe auch: EnterWait

KLASSE:
Desktop

SYNTAX:
LeaveWait()

BEISPIEL:

```
Sub ExampleWaitCursor
Const sMessage = "Jetzt ist der Wartecursor eingeschaltet."
Documents.Add( "sWriter4" )
Application.EnterWait
If Application.IsWait = TRUE Then
Selection.Insert sMessage
End If
Wait 3000
Application.LeaveWait
```

```
Selection.GotoStartOfLine TRUE
Selection.Delete
End Sub
```

Left-Eigenschaft

Bestimmt oder ermittelt die horizontale Position des Fensters, das durch das entsprechende Objekt bestimmt wird.

Siehe auch: Top, Height, Width.

KLASSE:
TaskWindow, Application

SYNTAX:
Left

RÜCKGABE:
Long

PARAMETER:
[= Left]: Beliebiger Long-Integer-Ausdruck, der die neue horizontale Position des Fensters in Pixel festlegt.

BEISPIEL:

```
Sub ExampleDimension
with Application
   .Height = 300
   .Width = 400
   .Left = 200
   .Top = 100
   wait 5000
   .Maximized=TRUE
end with
```

LeftLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die linke Umrandungslinie beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

Siehe auch: BottomLine, TopLine, RightLine.

KLASSE:
Border

SYNTAX:
LeftLine

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

Load-Methode

Lädt eine DokumentInfo-Instanz aus einer StarOffice-Dokument-Datei.

KLASSE:
DocumentInfo

SYNTAX:
Load(FileName As String)

RÜCKGABE:
Boolean

PARAMETER:
FileName: Dokument, aus dem die Dokumentinformation geladen werden soll.

Beispiel:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

LoadAccel-Methode

Lädt aus einer Konfigurationsdatei (*.cfg) die Tastaturbelegung. Ohne Übergabeparameter wird das Register TASTATUR des Dialogs ANPASSEN aufgerufen, und man kann dort über die Schaltfläche LADEN... eine Datei auswählen.

Siehe auch: LoadConfiguration, LoadMenu, LoadStatusBar, LoadToolBox.

KLASSE:
Application

SYNTAX:
LoadAccel(FileName As String)

PARAMETER:
[FileName]: Zeichenfolge, die den Dateinamen oder Namen und Pfad einer StarOffice-Konfigurationsdatei (*.cfg) angibt. Wird nur der Dateiname angegeben, wird die Datei aus dem Verzeichnis geladen, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

LoadConfiguration-Methode

Lädt und setzt sämtliche Einstellungen aus einer bestehenden Konfigurationsdatei (*.cfg). Die Konfigurationsdatei kann die Menü-, Tastatur- und Symbolleistenkonfiguration und Ereignisse enthalten.

Siehe auch: LoadAccel, LoadMenu, LoadStatusBar, LoadToolBox, SaveConfiguration.

KLASSE:
Application

SYNTAX:
LoadConfiguration(FileName As String)

PARAMETER:
FileName: Zeichenfolge, die den Dateinamen oder Namen und Pfad einer StarOffice-Konfigurationsdatei (*.cfg) angibt. Wird nur der Dateiname angegeben, wird die Datei aus dem Verzeichnis geladen, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

LoadLibrary-Methode

Lädt eine per Referenz oder extern eingebundene Bibliothek, welche im StarOffice bereits angemeldet ist. Die Methode entspricht der Auswahl einer sbl-Datei unter EXTRAS - MAKRO... - VERWALTEN... - BIBLIOTHEKEN.

Um eine Bibliothek mit LoadLibrary laden zu können, muß sie zuvor mit AddLibrary angemeldet worden sein.

Siehe auch: AddLibrary, RemoveLibrary, UnloadLibrary.

KLASSE:
OfficeDocument, Application

SYNTAX:
LoadLibrary(LibraryName As String)

RÜCKGABE:
Boolean

PARAMETER:
LibraryName: Name der Bibliothek, die geladen werden soll.

BEISPIEL:

```
Sub ExampleLibrary
On Error goto LibraryNotFound
Dim iState as integer
Documents.Add("sWriter4")
iState = Application.LoadLibrary("GIMMICKS")
```

```
Call Gimmicks.Userfields.StartChangesUserfields REM ruft
Routine aus Biliothek gimmicks auf
iState = Application.UnloadLibrary("GIMMICKS")
REM jetzt findet StarOffice die Bibliothek nicht mehr und
verursacht einen Fehler
Call Gimmicks.Userfields.StartChangesUserfields
Exit Sub
LibraryNotFound:
MsgBox("Biliothek nicht gefunden",16,"Fehler absichtlich verur-
sacht")
End Sub
```

LoadMenu-Methode

Lädt aus einer Konfigurationsdatei (*.cfg) die Pull-Down-Menü-Belegung. Ohne Übergabeparameter wird das MENÜ-Register des Dialogs ANPASSEN aufgerufen, und man kann dort über die Schaltfläche LADEN... eine Datei auswählen.

Siehe auch: LoadAccel, LoadConfiguration, LoadStatusBar, LoadToolBox.

KLASSE:
Application

SYNTAX:
LoadMenu(FileName As String)

PARAMETER:
[FileName]: Zeichenfolge, die den Dateinamen oder Namen und Pfad einer StarOffice-Konfigurationsdatei (*.cfg) angibt. Wird nur der Dateiname angegeben, wird die Datei aus dem Verzeichnis geladen, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

LoadStatusBar-Methode

Lädt aus einer Konfigurationsdatei (*.cfg) die Statusleisten-Konfiguration. Ohne Übergabeparameter wird das Register STATUSLEISTE des Dialogs ANPASSEN aufgerufen, und man kann dort über die Schaltfläche LADEN... eine Datei auswählen.

Siehe auch: LoadAccel, LoadConfiguration, LoadMenu, LoadToolBox.

KLASSE:
Application

SYNTAX:
LoadStatusBar(FileName As String)

PARAMETER:
[FileName]: Zeichenfolge, die den Dateinamen oder Namen und Pfad einer StarOffice-Konfigurationsdatei (*.cfg) angibt. Wird nur der Dateiname angegeben, wird die Datei aus dem Verzeichnis geladen, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

LoadToolBox-Methode

Lädt aus einer Konfigurationsdatei (*.cfg) die Symbolleisten-Definition. Ohne Übergabeparameter wird das SYMBOLLEISTEN-Register des Dialogs ANPASSEN aufgerufen, und man kann dort über die Schaltfläche LADEN... eine Datei auswählen.

Siehe auch: LoadAccel, LoadConfiguration, LoadMenu, LoadStatusBar, LoadEvents.

KLASSE:
Application

SYNTAX:
LoadToolBox(FileName As String)

PARAMETER:
[FileName]: Zeichenfolge, die den Dateinamen oder Namen und Pfad einer StarOffice-Konfigurationsdatei (*.cfg) angibt. Wird nur der Dateiname angegeben, wird die Datei aus dem Verzeichnis geladen, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

Longer-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche zu einem Wort hinzugefügt werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

KLASSE:
SearchSettings

SYNTAX:
Longer

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, um wieviele Zeichen das Wort im Dokument länger sein darf als der Suchbegriff, um noch als gleich zu gelten.

BEISPIEL:

```
Sub ExampleSimilarity
Const sSearch="Test" REM Text der gesucht wird.
Const sText="eTxst" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Mode = 1
```

```
ActiveWindow.SearchSettings.Similarity = TRUE
ActiveWindow.SearchSettings.Relaxed = TRUE
ActiveWindow.SearchSettings.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.SearchSettings.Shorter = 2 REM Zeichen entfernen
(y)
ActiveWindow.SearchSettings.Longer = 2 REM Zeichen hinzu-
fügen (z)
ActiveWindow.Search sSearch
MsgBox "gesucht: " & sSearch  & Chr(13) & "gefunden: " & Selec-
tion.Value ,64,"Ähnlichkeitssuche:"
End Sub
```

Eigenschaften und Methoden mit "M"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "M" beginnen.

MacroBarVisible-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption der Makro-Symbolleiste.

KLASSE:
Application

SYNTAX:
MacroBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Makro-Symbolleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleMacroBarVisible
Dim bOK As Boolean
bOK = Application.MacroBarVisible
Application.MacroBarVisible = NOT bOK
Wait 2000
Application.MacroBarVisible = bOK
End Sub
```

MacroDialog-Methode

Ruft den Dialog MAKRO zum Starten oder Bearbeiten von Makros auf.

KLASSE:
Desktop

SYNTAX:
MacroDialog()

BEISPIEL:

```
Sub ExampleMacroDialog
Application.MacroDialog
End Sub
```

MailFollowup-Methode

Öffnet ein neue Nachricht, um die aktuelle Nachricht öffentlich zu erwidern.

KLASSE:
MessageDocument

SYNTAX:
MailFollowup()

BEISPIEL:

```
Sub ExampleMailFollowup
ActiveWindow.TopWindow.Document.MailFollowup
End Sub
```

MailForward-Methode

Öffnet eine neue Nachricht, um die aktuelle Nachricht weiterzuleiten.

KLASSE:
MessageDocument

SYNTAX:
MailForward()

BEISPIEL:

```
Sub ExampleMailForward
ActiveWindow.TopWindow.Document.MailForward
End Sub
```

MailGroupReply-Methode

Öffnet eine neue Nachricht, um die aktuelle Nachricht an deren Absender und alle ihre Adressaten zu beantworten.

KLASSE:
MessageDocument

SYNTAX:
MailGroupReply()

BEISPIEL:

```
Sub ExampleMailGroupReply
ActiveWindow.TopWindow.Document.MailGroupReply
End Sub
```

MailMarkAsRead-Methode

Setzt die aktuell geöffnete Nachricht auf "ungelesen". Bei wiederholtem Anwenden dieser Methode wird die Nachricht wieder auf "gelesen" gesetzt.

KLASSE:
MessageDocument

SYNTAX:
MailMarkAsRead()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleMailMarkAsRead
ActiveWindow.TopWindow.Document.MailMarkAsRead
End Sub
```

MailNext-Methode

Ersetzt die geöffnete Nachricht durch die nächste Nachricht des geöffneten Nachrichtenkorbs bzw. der geöffneten Newsgroup.

KLASSE:
MessageDocument

SYNTAX:
MailNext()

Beispiel:

```
Sub ExampleMailNext
ActiveWindow.TopWindow.Document.MailNext
End Sub
```

MailNextDelete-Methode

Löscht die geöffnete Nachricht und ersetzt sie in der Anzeige durch die nächste Nachricht des geöffneten Nachrichtenkorbs bzw. der geöffneten Newsgroup.

Klasse:
MessageDocument

Syntax:
MailNextDelete()

Beispiel:

```
Sub ExampleMailNextDelete
ActiveWindow.TopWindow.Document.MailNextDelete
End Sub
```

MailNextUnread-Methode

Ersetzt die aktuell geöffnete Nachricht durch die nächste ungelesene Nachricht des geöffneten Nachrichtenkorbs bzw. der geöffneten Newsgroup.

Klasse:
MessageDocument

Syntax:
MailNextUnread()

Beispiel:

```
Sub ExampleMailNextUnread
ActiveWindow.TopWindow.Document.MailNextUnread
End Sub
```

MailNotifyReset-Methode

Setzt in der Statuszeile die Anzeige für das Vorliegen neuer Nachrichten zurück. Daß neue Nachrichten vorliegen wird durch Blinken des Symbols im Feld Neue Nachricht erhalten angezeigt. Mit MailNotifyReset wird dieses Blinken gestoppt.

Klasse:
Application

SYNTAX:
MailNotifyReset()

BEISPIEL:

```
Sub ExampleMailNotifyReset
   Application.MailNotifyReset
End Sub
```

MailPrev-Methode

Ersetzt die geöffnete Nachricht durch die vorherige Nachricht des geöffneten Nachrichtenkorbs bzw. der geöffneten Newsgroup.

KLASSE:
MessageDocument

SYNTAX:
MailPrev()

BEISPIEL:

```
Sub ExampleMailPrev
ActiveWindow.TopWindow.Document.MailPrev
End Sub
```

MailPrevDelete-Methode

Löscht die geöffnete Nachricht und ersetzt sie in der Anzeige durch die vorherige Nachricht des geöffneten Nachrichtenkorbs bzw. der geöffneten Newsgroup.

KLASSE:
MessageDocument

SYNTAX:
MailPrevDelete()

BEISPIEL:

```
Sub ExampleMailPrevDelete
ActiveWindow.TopWindow.Document.MailPrevDelete
End Sub
```

MailPrevUnread-Methode

Ersetzt die aktuell geöffnete Nachricht durch die vorherige ungelesene Nachricht im geöffneten Nachrichtenkorb bzw. der geöffneten Newsgroup.

KLASSE:
MessageDocument

SYNTAX:
MailPrevUnread()

BEISPIEL:

```
Sub ExampleMailPrevUnread
ActiveWindow.TopWindow.Document.MailPrevUnread
End Sub
```

MailReceipt-Eigenschaft

Ermittelt, ob im Posteingang neue Nachrichten vorhanden sind und liefert die Anzahl der eingegangenen Mails zurück.

Diese Eigenschaft funktioniert nur, wenn beim Mail-Account, das AKTUALISIEREN DURCH INTERNET-UPDATE eingeschaltet ist.

KLASSE:
Application

SYNTAX:
MailReceipt

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleMailReceipt
MsgBox "Sie haben " & Application.MailReceipt & " neue Nach-
richten.",0,"PostOffice"
End Sub
```

MailReply-Methode

Öffnet eine neue Nachricht, um die aktuell geöffnete Nachricht an deren Absender zu beantworten.

KLASSE:
MessageDocument

SYNTAX:
MailReply()

Beispiel:

```
Sub ExampleMailReply
ActiveWindow.TopWindow.Document.MailReply
End Sub
```

MailSend-Methode

Verschickt das aktuelle Dokument als Nachricht über das Mailsystem.

Klasse:
MessageDocument

Syntax:
MailSend()

MarkMail-Methode

Markiert die aktuell geöffnete Nachricht.

Klasse:
MessageDocument

Syntax:
MarkMail()

Rückgabe:
Boolean

Beispiel:

```
Sub ExampleMarkMail
ActiveWindow.TopWindow.Document.MarkMail
End Sub
```

MarkThreadAsRead-Methode

Setzt alle Nachrichten des aktuellen Threads auf "gelesen".

Klasse:
MessageDocument

Syntax:
MarkThreadAsRead()

Beispiel:

```
Sub ExampleMarkThreadAsRead
ActiveWindow.TopWindow.Document.MarkThreadAsRead
End Sub
```

MarkThreadAsUnread-Methode

Setzt alle Nachrichten des aktuellen Threads auf "nicht gelesen".

KLASSE:
MessageDocument

SYNTAX:
MarkThreadAsUnread()

BEISPIEL:

```
Sub ExampleMarkThreadAsUnread
ActiveWindow.TopWindow.Document.MarkThreadAsUnread
End Sub
```

Maximized-Eigenschaft

Bestimmt oder ermittelt, ob das Fenster, an dem die Eigenschaft aufgerufen wird, maximiert oder im Normalmodus dargestellt wird.

Siehe auch: Minimized

KLASSE:
TaskWindow, Application

SYNTAX:
Maximized

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Fenster maximiert (True) oder im Normalmodus (False) dargestellt wird.

BEISPIEL:

```
sub ExampleWindow
Dim sWin As String
Const sMessage1 = "Das Fenster wird minimiert, "
Const sMessage2 = "maximiert und in den Zustand bei seiner
Erzeugung gebracht."
Const sMessage3 = "Danach wird es entfernt."
Documents.Add "sWriter4"
ActiveWindow.Selection.Insert(sMessage1 & sMessage2 & sMes-
sage3)
Wait 2000
sWin =  ActiveWindow.Name
Windows( sWin ).Minimized = True
Wait 2000
Windows( sWin ).Maximized = True
```

```
Wait 2000
Windows( sWin ).Restore
Wait 2000
Windows.Remove( sWin )
end sub
```

Minimize-Methode

Minimiert alle TaskWindows der Windows-Collection. Dies ist in der Regel (außer beim Macintosh) die Icon-Darstellung.

KLASSE:
Windows

SYNTAX:
Minimize()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT"  REM Textsegment, anhand dessen neue
Dokumente identifiziert werden können
Dim iCount As Integer
For iCount=1 to 8
Documents.Add "sWriter4"
Wait 100
ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize REM Alle Fenster minimieren
Windows.ArrangeIcons
REM alle neuen Dokument schliessen
iCount=1 REM erster Index der WindowsCollection ist 0
Do While iCount <= Windows.Count
if Instr( 1, UCASE( Windows( iCount ).Name ) , UCase(sNewDoc)
)>0 then
Windows.Remove( iCount ) REM entfernen ohne Nachfrage !
else
iCount = iCount + 1
end if
loop
Windows.Close REM Alles Schliessen mit Nachfrage
End Sub
```

Minimized-Eigenschaft

Bestimmt oder ermittelt, ob das dem Objekt entsprechende Fenster minimiert oder im Normalmodus dargestellt wird.

Siehe auch: Maximized

KLASSE:
TaskWindow, Application

SYNTAX:
Minimized

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Fenster minimiert (True) oder im Normalmodus (False) dargestellt wird.

BEISPIEL:

```
Sub ExampleApplicationMinMax
Application.Minimized = TRUE
Wait 2000
Application.Maximized = TRUE
End Sub
```

ModificationAuthor-Eigenschaft

Ermittelt den Autor, der das Dokument zuletzt gespeichert hat. Dieser wird aus den zugehörigen Dokumenteigenschaften gelesen (Label "Geändert am:").

KLASSE:
DocumentInfo

SYNTAX:
ModificationAuthor

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
```

```
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

ModificationDate-Eigenschaft

Ermittelt das Datum, an dem das Dokument zuletzt gespeichert wurde. Dieses wird aus den zugehörigen Dokumenteigenschaften gelesen (Label "Geändert am:").

KLASSE:
DocumentInfo

SYNTAX:
ModificationDate

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
```

```
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

ModificationTime-Eigenschaft

Ermittelt die Uhrzeit, zu der das Dokument zuletzt gespeichert wurde. Diese wird aus den zugehörigen Dokumenteigenschaften gelesen (Label "Geändert am:").

KLASSE:
DocumentInfo

SYNTAX:
ModificationTime

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo As New DocumentInfo
Dim sDocPath As String
Dim vInfo As Variant, iState As Integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden, ohne das Dokument komplett laden zu müssen."
REM Gültigen Dokumentnamen ermitteln
sDocPath = Application.PathSettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
.Description = sDescription
vInfo = .Author
```

```
vInfo = .CreationDate
vInfo = .CreationTime
vInfo = .Description
vInfo = .EditTime
vInfo = .Field1
vInfo = .Field1Title
vInfo = .Field2
vInfo = .Field2Title
vInfo = .Field3
vInfo = .Field3Title
vInfo = .Field4
vInfo = .Field4Title
vInfo = .FileName
vInfo = .KeyWords
vInfo = .ModificationAuthor
vInfo = .ModificationDate
vInfo = .ModificationTime
vInfo = .Revision
vInfo = .Subject
vInfo = .Template
.Title = sTitle
End With
ActiveDocument.Close True, sDocPath REM Schließen mit Speichern
Wait 100
REM Jetzt die Dokumentinformation kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokumentinformationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Modified-Eigenschaft

Bestimmt oder ermittelt, ob das Dokument seit dem letzten Speichern verändert wurde.

KLASSE
OfficeDocument

SYNTAX:
Modified

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Dokument seit dem letzten Speichern verändert wurde (True) oder nicht (False).

Beispiel:

```
Sub ExampleModified
Const sNewName = "NewName" REM An diesen Namen kommt noch eine
Ziffer als Versionnummer dran
Dim sDoc As String,iCount As Integer
Documents.Add("sWriter4")
Selection.Insert("Dieses Dokument wurde erzeugt um gespeichert
zu werden.")
Selection.InsertPara
For iCount = 1 to 12
   sDoc = Application.PathSettings.Work & "\" &_
          sNewName & ((Val( Right( ActiveDocument.Name ,1 )) +
1) Mod 10) & ".sdw"
   If ActiveDocument.Modified Then ActiveDocument.SaveAs sDoc
   Selection.Insert("Versionsnummer hochzählen " & ((Val(
Right( ActiveDocument.Name ,1 )) + 1) Mod 10) )
   Selection.InsertPara
Next iCount
If NOT ActiveDocument.Saved Then ActiveDocument.Save
End Sub
```

Module-Eigenschaft

Über diese Eigenschaft ist das zum Dokument gehörige Modul-Objekt erreichbar. Im Modul-Objekt können global Einstellungen für den Dokumenttyp vorgenommen werden. Module sind z. B. StarWriter, StarCalc etc.

Klasse:
Document

Syntax:
Module

Rückgabe:
Object

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleDocumentModul
Dim oObj As Object, sText As String
Documents.Add("sWriter4")
   oObj = Application.ActiveWindow.Document.Module
   sText =sText & "Applikations-Modul: " & oObj.Name
   MsgBox sText,0,"Name des Moduls für das aktive Fenster"
End Sub
```

Modules-Eigenschaft

Liefert oder setzt den Pfad für Module. Dieser wird unter Extras - Optionen - Allgemein... - Pfade eingestellt.

Klasse:
PathSettings

Syntax:
Modules

Rückgabe:
String

Parameter:
[= String]: Zeichenfolge, die den Pfad angibt.

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
```

```
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

MultiQuickSearch-Eigenschaft

Bestimmt oder fragt ab, ob die Mehrfachselektion der Schnellsuche in der Detailansicht des Desktop-Dokuments angeschaltet ist.

KLASSE:
Application

SYNTAX:
MultiQuickSearch

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Mehrfach-Suche in der Detailansicht an- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleMultiQuickSearch
Application.MultiQuickSearch = TRUE
End Sub
```

Eigenschaften und Methoden mit "N"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "N" beginnen.

Name-Eigenschaft

Ermittelt den Namen eines Module-, Document- oder Window-Objekts. Das entsprechende Objekt kann dann im Code durch diesen Namen direkt angesprochen werden, statt "ActiveDocument", "ActiveWindow" oder "ActiveModule" zu verwenden. Bei Window-Objekten muß der Name dabei in eckigen Klammern angegeben werden, bei Dokumenten oder Modulen ist das zwar auch möglich, aber nicht zwingend erforderlich.

Der Name eines Document-Objekts ist mit dem Dateinamen oder, wenn das Dokument irgendeine Ordneransicht darstellt, mit dem Ordnernamen des

entsprechenden Dokuments identisch, bei unbenannten Dokumenten ist er "unbenanntX" (X=1,2,...).

Der Name eines Window-Objekts ist der Name des dazugehörigen Document-Objekts, an den ein Doppelpunkt (":") und eine Nummer angehängt werden. Die Nummer folgt aus einer internen Zählung der Fenster pro Dokument.

KLASSE:
Document, Window, Module

SYNTAX:
Name

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
REM Dieses Beispiel demonstriert die Textausgabe in einem nicht
aktiven Fenster.
Global sInfoWin as String REM Wie das Informationsausgabefen-
ster heißt
Sub ExampleName
   TextAppend( "Dieser Text erscheint in einem nicht aktiven
Fenster")
End Sub
Sub TextAppend( sAppendText As String )
if NOT WindowExists( sInfoWin ) then REM ob das Fenster bereits
exitiert, wenn nicht dann
   AktWin$ = ActiveWindow.Name REM das gerade aktive Fenster
merken
   Documents.Add( "sWriter4" ) REM neues anlegen
   ActiveDocument.DocumentInfo.Title ="Information"
   REM bei unbenannte Dokumenten erscheint der title inder
Titelzeile, sonst der Name
   sInfoWin = ActiveWindow.Name
   REM Nach dem Öffnene Namen in globale Variable sichern
   Windows( AktWin ).Activate REM  Bei Bedarf altes Fenster
wieder aktivieren
end if
With Windows( sInfoWin ).Selection
   .Escape
   .JumpToEndOFDoc
   .Insert( sAppendText )
   .InsertPara
End With
End Sub
Function WindowExists( WinName as String ) as Boolean
```

```
Dim sDummy As String
On Error goto WindowDoesntExist
sDummy = Windows( WinName ).Name
REM falls das Fenster nicht existiert wird hier eine Exception
eingeleitet
WindowExists = TRUE
exit Function
WindowDoesntExist:
WindowExists = FALSE
end Function
```

Navigator-Eigenschaft

Öffnet oder schließt den Navigator oder ermittelt, ob der Navigator ein- oder ausgeblendet ist.

KLASSE:
Desktop

SYNTAX:
Navigator

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Navigator ein- (True) oder ausgeblendet (False) wird.

BEISPIEL:

```
Sub ExampleNavigator
Documents.Add "sWriter4/Web"
Application.Navigator = TRUE
Wait 3000
Application.Navigator = FALSE
End Sub
```

NewMail-Methode

Erstellt ein neues leeres Mail-Dokument.

KLASSE:
Desktop, MessageDocument

SYNTAX:
NewMail()

Beispiel:

```
Sub ExampleNewMail
Const sMailText = "Sehr geehrte Damen und Herren,"
Dim oTopwinObject As Object
Dim oMailbodyObject As Object
Dim oMailAttachObject As Object
Dim oMailHeaderObject As Object
Application.NewMail
Wait 10  REM Das Erstellen des Maildokuments dauert länger
oTopwinObject = ActiveWindow.Topwindow
oMailHeaderObject = oTopwinObject.GetFrameWindow(1)
oMailAttachObject = oTopwinObject.GetFrameWindow(2)
oMailBodyObject = oTopwinObject.GetFrameWindow(3)
oTopwinObject.Document.Subject = "Subject"
oMailBodyObject.insert sMailText
End Sub
```

NewWindow-Methode

Erzeugt ein neues Top-Level-Fenster mit gleichem strukturellen Aufbau und Inhalt wie das Window, an dem diese Methode aufgerufen wird. Dies erfolgt nach Möglichkeit durch Öffnen einer zweiten Sicht oder durch erneutes Laden (dann allerdings schreibgeschützt).

Klasse:
BrowseWindow

Syntax:
NewWindow()

Beispiel:

```
Sub ExampleNewWindow
Const sMessage=" Fenster "
Dim sDoc As String
Documents.Add "sWriter4"
sDoc = ActiveDocument.Name
For iCount = 1 to 10
Selection.Insert(sMessage & iCount)
ActiveWindow.NewWindow
Next iCount
Windows.Tile
Wait 2000
Documents(sDoc ).Close "",FALSE
End Sub
```

Eigenschaften und Methoden mit "O"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "O" beginnen.

ObjectBarVisible-Eigenschaft

Bestimmt oder ermittelt, ob die Ansicht der Objektleiste eingeschaltet ist.

KLASSE:
Application

SYNTAX:
ObjectBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Objektleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleObjectBarVisible
Application.ObjectBarVisible = FALSE
Wait 2000
Application.ObjectBarVisible = TRUE
End Sub
```

OnActivate-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument aktiviert, d. h. sein Fenster in den Vordergrund geholt wird. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument erneut aktiviert wird, sonst bei keinem anderen.

Siehe auch: OnDeactivate, OnDocActivate.

KLASSE:
Document

SYNTAX:
OnActivate

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname (Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnActivate
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Aktivieren dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnActivate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnClose-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument geschlossen wird. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument geschlossen wird, sonst bei keinem anderen. Das zugehörige Ereignis unter EXTRAS - ANPASSEN... - EREIGNISSE hat den Namen "Dokument wird geschlossen".

Siehe auch: OnOpen, OnDocClose, OnPrepareClose.

KLASSE:
Document

SYNTAX:
OnClose

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname(Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnClose
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Schließen dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnClose = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnCreate-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn aus der aktuellen Dokumentvorlage ein neues Dokument erzeugt wird. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau aus diesem Vorlagendokument ein neues Dokument erstellt wird, sonst bei keinem anderen.

Siehe auch: OnDocCreate.

KLASSE:
Document

SYNTAX:
OnCreate

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname (Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnCreate
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Erstellen dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnCreate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDeactivate-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument deaktiviert, d. h. ein anderes Dokument in den Vordergrund geholt wird und das aktuelle Dokument den Fokus verliert. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument deaktiviert wird, sonst bei keinem anderen.

Siehe auch: OnActivate, OnDocDeactivate.

KLASSE:
Document

SYNTAX:
OnDeactivate

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname (Bibliotheksname.Modulname)

BEISPIEL:

```
Sub ExampleOnDeActivate
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Deaktivieren dieses Dokumentes wird
ein Makro gestartet"
Activedocument.OnDeActivate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDocActivate-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, wenn ein beliebiges Dokument aktiviert wird.

Siehe auch: OnDocDeactivate

KLASSE:
Application

SYNTAX:
OnDocActivate

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein Dokument aktiviert wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDeActivate
Documents.Add( "sWriter4" )
Selection.Insert "Beim Deaktivieren eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocActivate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDocClose-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, wenn ein beliebiges Dokument geschlossen wird. Das zugehörige Ereignis unter EXTRAS - ANPASSEN... - EREIGNISSE hat den Namen "Dokument wird geschlossen".

Siehe auch: OnDocOpen, OnDocPrepareClose, OnClose.

KLASSE:
Application

SYNTAX:
OnDocClose

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald das Ereignis "Dokument wird geschlossen" eintritt. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocClose
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Schließen eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocClose = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnDocCreate-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald ein neues Dokument erzeugt wird. Dabei spielt es keine Rolle, um welchen Dokumenttyp es sich handelt.

KLASSE:
Application

SYNTAX:
OnDocCreate

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein neues Dokument erzeugt wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocCreate
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Erzeugen eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocCreate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDocDeactivate-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, wenn ein beliebiges Dokument deaktiviert wird.

Siehe auch: OnDocActivate

KLASSE:
Application

SYNTAX:
OnDocDeactivate

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein Dokument deaktiviert wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocDeactivate
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Deaktivieren eines beliebigen Doku-
mentes wird ein Makro gestartet"
Application.OnDocDeactivate =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDocOpen-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald ein beliebiges Dokument geöffnet wird.

Siehe auch: OnDocClose

KLASSE:
Application

SYNTAX:
OnDocOpen

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein Dokument geladen wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocOpen
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Öffnen eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocOpen = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnDocPrepareClose-Eigenschaft

Bestimmt oder ermittelt das Makro, das beim Schließen eines beliebigen Dokuments vor der Speichernrückfrage ausgeführt werden soll. Das zugehörige Ereignis unter EXTRAS - ANPASSEN... - EREIGNISSE hat den Namen "Dokument schließen".

Im Gegensatz zur OnDocClose-Methode bietet OnDocPrepareClose die Möglichkeit, statt des Standard-Rückfragedialogs einen eigenen Dialog einzublenden. Der Standard-Dialog erscheint nicht, wenn das Dokument vor dem Schließen gespeichert wird (z. B. mittels Save) oder wenn ein zuvor nicht gespeichertes Dokument mittels der Modified-Eigenschaft auf "nicht modifiziert" gesetzt wird. Hat das Dokument den Status "modifiziert", erscheint automatisch ein Rückfrage-Dialog zum Speichern des Dokuments.

Beachten Sie bitte, daß es nicht möglich ist, ein an ein Ereignis geknüpftes Makro abzubrechen, da bei solchen Makros keine Auswertung von Rückgabewerten stattfindet.

Siehe auch: OnDocClose, OnPrepareClose, Modified.

KLASSE:
Application

SYNTAX:
OnDocPrepareClose

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald das Ereignis "Dokument schließen" eintritt. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocPrepareClose
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Schließen eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocPrepareClose =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnDocPrint-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, wenn ein beliebiges Dokument gedruckt wird.

KLASSE:
Application

SYNTAX:
OnDocPrint

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein Dokument gedruckt wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocPrint
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Drucken eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocPrint = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnDocSave-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald ein beliebiges Dokument gespeichert wird. Dies entspricht dem Ereignis "Dokument sichern" unter EXTRAS - ANPASSEN... - EREIGNISSE.

KLASSE:
Application

SYNTAX:
OnDocSave

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald ein Dokument gespeichert wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocSave
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Speichern eines beliebigen Dokumentes
wird ein Makro gestartet"
Application.OnDocSave = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnDocSaveAs-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald ein beliebiges Dokument unter einem Namen gespeichert wird. Dies entspricht dem Ereignis "Dokument sichern als" unter EXTRAS - ANPASSEN... - EREIGNISSE.

Siehe auch: OnDocSave, OnSaveAs.

KLASSE:
Application

SYNTAX:
OnDocSaveAs

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald das Ereignis "Dokument sichern als" eintritt. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnDocSaveAs
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Speichern unter eines beliebigen
Dokumentes wird ein Makro gestartet"
Application.OnDocSaveAs =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnlineOptions-Methode

Ruft den Dialog OPTIONEN HTML-DOKUMENT auf. Dieser Dialog ist modal, d. h. das Makro wird solange unterbrochen, bis der Dialog vom Anwender geschlossen wird.

KLASSE:
Desktop

SYNTAX:
OnlineOptions()

BEISPIEL:

```
Sub ExampleOnlineOptions
Application.OnlineOptions
End Sub
```

OnlineRegistrationDlg-Methode

Öffnet die Support-Seite zur Online-Registrierung. Voraussetzung für diese Methode ist ein Zugang zum Internet.

KLASSE:
Desktop

SYNTAX:
OnlineRegistrationDlg()

BEISPIEL:

```
Sub ExampleOnlineRegistrationDlg
Application.OnlineRegistrationDlg
End Sub
```

OnOpen-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument geöffnet wird. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument geöffnet wird, sonst bei keinem anderen.

KLASSE:
Document

SYNTAX:
OnOpen

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname (Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnOpen
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Öffnen dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnOpen = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnPrepareClose-Eigenschaft

Bestimmt oder ermittelt das Makro, das beim Schließen eines Dokuments vor der Speichernrückfrage ausgeführt werden soll. Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument geschlossen wird, sonst bei keinem anderen. Das zugehörige Ereignis unter EXTRAS - ANPASSEN... - EREIGNISSE hat den Namen "Dokument schließen".

Im Gegensatz zur OnClose-Methode bietet OnPrepareClose die Möglichkeit, einen eigenen Dialog einzublenden. Der Standard-Dialog erscheint nicht, wenn das Dokument vor dem Schließen gespeichert wird (z. B. mittels Save) oder wenn ein zuvor nicht gespeichertes Dokument mittels der Modified-Eigenschaft auf "nicht modifiziert" gesetzt wird. Hat das Dokument den Status "modifiziert", erscheint automatisch ein Rückfrage-Dialog zum Speichern des Dokuments.

Beachten Sie bitte, daß es nicht möglich ist, ein an ein Ereignis geknüpftes Makro abzubrechen, da bei solchen Makros keine Auswertung von Rückgabewerten stattfindet.

Siehe auch: OnClose, OnDocPrepareClose, Modified.

KLASSE:
Document

SYNTAX:
OnPrepareClose

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald das Ereignis "Dokument schließen" eintritt. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnPrepareClose
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Schliessen dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnPrepareClose =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnPrint-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument ausgedruckt wird.

Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument gedruckt wird, sonst bei keinem anderen.

Siehe auch: OnDocPrint

KLASSE:
Document

SYNTAX:
OnPrint

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält: Makroname(Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnPrint
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Drucken dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnPrint = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnQuit-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald StarOffice beendet wird.

Siehe auch: OnStart

KLASSE:
Application

SYNTAX:
OnQuit

RÜCKGABE:
String

PARAMETER:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald die Applikation beendet wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

BEISPIEL:

```
Sub ExampleOnQuit
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Schließen des Office wird ein Makro
gestartet"
Application.OnQuit = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnSave-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument gespeichert wird. Dies entspricht dem Ereignis "Dokument sichern" unter EXTRAS - ANPASSEN... - EREIGNISSE.

Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument gespeichert wird, sonst bei keinem anderen. Siehe auch: OnDocSave, OnSaveAs.

KLASSE:
Document

SYNTAX:
OnSave

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname(Bibliotheksname.Modulname).

BEISPIEL:

```
Sub ExampleOnSave
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Speichern dieses Dokumentes wird ein
Makro gestartet"
Activedocument.OnSave = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

OnSaveAs-Eigenschaft

Bestimmt oder ermittelt, welches Makro automatisch aufgerufen wird, wenn ein Dokument unter einem Namen gespeichert wird. Dies entspricht dem Ereignis "Dokument sichern als" unter EXTRAS - ANPASSEN... - EREIGNISSE.

Das Ereignis wird für das aktive Dokument gesetzt, und es wird ausgelöst, wenn genau dieses Dokument gespeichert wird, sonst bei keinem anderen. Siehe auch: OnDocSaveAs, OnSave.

KLASSE:
Document

SYNTAX:
OnSaveAs

RÜCKGABE:
String

PARAMETER:
[= Makro]: Zeichenfolgenausdruck, der die Angabe des Makros in folgendem Format enthält:

Makroname(Bibliotheksname.Modulname).

Beispiel:

```
Sub ExampleOnSaveAs
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Speichern unter dieses Dokumentes
wird ein Makro gestartet"
Activedocument.OnSaveAs =
"StartChangesUserfields(Gimmicks.Userfields)"
End Sub
```

OnStart-Eigenschaft

Bestimmt oder ermittelt das Makro, das ausgeführt werden soll, sobald StarOffice gestartet wird.

Siehe auch: OnQuit

Klasse:
Application

Syntax:
OnStart

Rückgabe:
String

Parameter:
[= Macro]: Zeichenfolgenausdruck, der das Makro angibt, das ausgeführt werden soll, sobald die Applikation gestartet wird. Dieser Parameter wird in dem Format "MacroName(Library.Modul)" angegeben.

MacroName: Gibt die Bezeichnung des Makros an, das ausgeführt werden soll.

Library: Gibt die Bezeichnung der Bibliothek an, in der das Makro enthalten ist.

Modul: Gibt die Bezeichnung des Moduls an, in dem das Makro enthalten ist.

Beispiel:

```
Sub ExampleOnStart
Documents.Add( "sWriter4" )
Selection.Insert "Bei dem Starten des Office wird ein Makro
gestartet"
Application.OnStart = "StartChangesUserfields(Gimmicks.User-
fields)"
End Sub
```

Open-Methode

Öffnet ein auf einem Datenträger bereits vorhandenes Dokument.

KLASSE:
Documents

SYNTAX:
Open(FileName As String, FilterName As String, OpenFlags As String, FilterFlags As String)

RÜCKGABE:
Object

PARAMETER:
FileName: Beliebiger Zeichenfolgenausdruck, der den Dateinamen der zu öffnenden Datei bestimmt.

[FilterName]: Zeichenfolgenausdruck, der die Bezeichnung des Textfilters angibt, mit dem das angegebene Dokument geöffnet werden soll. Wird dieser Parameter nicht übergeben, so wird der richtige Filter von der Applikation ausgewählt.

[OpenFlags]: Zeichenfolgenausdruck, der den Zugriffsmodus bestimmt (H=Hidden, R=ReadOnly).

[FilterFlags]: Applikationsspezifische Filterangabe, die nur dann ausgewertet wird, wenn eine ASCII-Text-Datei geöffnet wird. FilterFlags legt die Textimport-Optionen für Feldseparator, Textdelimiter und Zeichensatz fest und wird als String mit drei Werten angegeben, die durch Kommata voneinander getrennt sind:

FilterFlags = " Feldseparator,Textdelimiter,Zeichensatz".

Feldseparator und Textdelimiter werden durch ihren ASCII-Code festgelegt, für den Zeichensatz sind folgende Werte möglich:

ANSI

MAC

IBMPC

IBMPC_437

IBMPC_850

IBMPC_860

IBMPC_861

IBMPC_863

IBMPC_865

SYSTEM

Wird nur ein Feldtrenner angegeben, wird als Textdelimiter das Anführungszeichen (") genommen und der Zeichensatz auf SYSTEM gesetzt.

Der Parameter FilterFlags ist optional. Wird er nicht übergeben, erscheint ein Abfragedialog für die Import-Optionen.

FilterFlags legt nur die Feld- und Trennoptionen für den Textimport fest. Andere Einstellungen, die im StarOffice im Dialog TEXTIMPORT vorgenommen werden können, sind im StarBasic nicht steuerbar. Datenfeldtypen werden standardmäßig vom Programm selbst bestimmt.

BEISPIEL:

```
Sub ExampleDocumentsOpen
Const sDoc1 = "Test1"
Const sDoc2 = "Test2"
Const sOfficeFilter="soffice4: StarOffice Dokumente"
Dim sPath1 as string, sPath2 as string
sPath1 = Application.PathSettings.Work & "\" & sDoc1 & ".sdw"
sPath2 = Application.PathSettings.Work & "\" & sDoc2 & ".sdw"
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 1"
Activedocument.SaveAs sPath1
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 2"
Activedocument.SaveAs sPath2
Windows(sDoc1 & ":1").Selection.Insert ": geändert" REM jeweils
die erste Sicht bearbeiten
Windows(sDoc2 & ":1").Selection.Insert ": geändert"
Documents.SaveAll
Documents(sDoc1).Close False, ""
Documents(sDoc2).Close False, ""
Documents.Open sPath1,sOfficeFilter  REM Öffnen mit Filterliste
End Sub
```

OpenDatabaseConnection-Methode

Öffnet eine Datenbank, die im StarOffice bereits registriert ist. Dabei wird lediglich eine Verbindung zur Datenbank hergestellt, sie wird nicht sichtbar geöffnet.

Siehe auch: AddDatabaseConnection GetActiveDatabaseConnection, OpenTableConnection, OpenQueryConnection.

KLASSE:
Desktop

SYNTAX:
OpenDatabaseConnection(DBName As String)

RÜCKGABE:
Boolean

PARAMETER:
DBName: Zeichenfolge, die den Namen angibt, mit dem die zu öffnende Datenbank im StarOffice angelegt wurde.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
```

```
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

OpenForm-Methode

Öffnet ein an einer Datenbank angemeldetes Formular in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenReport, OpenQuery.

KLASSE:
Desktop

SYNTAX:
OpenForm(FormName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Zeichenfolge, die den Namen des Formulars angibt.

ReadOnly: Boolescher Ausdruck, der angibt, ob das Formular schreibgeschützt geladen werden soll (True) oder ob die Einträge geändert werden können (False).

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
```

```
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

OpenQuery-Methode

Öffnet eine Datenbank-Abfrage in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenReport, OpenForm.

KLASSE:
Desktop

SYNTAX:
OpenQuery(QueryName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Zeichenfolge, die den Namen der Datenbank-Abfrage angibt.

ReadOnly: Boolescher Ausdruck, der angibt, ob die Datenbank-Abfrage im Bearbeiten-Modus geladen werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
```

```
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

OpenQueryConnection-Methode

Stellt eine Verbindung zu einer Datenbank-Abfrage her, wenn die zugehörige Datenbank zuvor mit der OpenDatabaseConnection-Methode geöffnet wurde. OpenQueryConnection stellt lediglich eine Verbindung zur Abfrage her, sie wird nicht sichtbar geöffnet.

Siehe auch: OpenDatabaseConnection, OpenQuery, OpenTableConnection.

KLASSE:
Desktop

SYNTAX:
OpenQueryConnection(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Zeichenfolge, die den Namen der Abfrage angibt, die geöffnet werden soll.

Beispiel:

```
Sub ExampleQueries
Const sQueryName = "TestAbfrage"
Const sMessage = "Der Inhalt des ersten Feldes : "
Dim sDataName As String, sTabName As String, sTPath As String
Dim vStatus As Variant
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = Application.DataGetTableName(sDataName, 1)
Application.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
vStatus = Application.OpenQueryConnection(sQueryName)
Application.DataFirstRecord
Print sMessage & Application.DataColumn(1)
Application.CloseQueryConnection(sQueryName)
Application.DeleteQuery sQueryName
End Sub
```

OpenReport-Methode

Öffnet einen Datenbank-Bericht in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenForm, OpenQuery.

Klasse:
Desktop

Syntax:
OpenReport(ReportName As String, ReadOnly As Boolean)

Rückgabe:
Boolean

Parameter:
ReportName: Zeichenfolge, die den Namen des Berichts angibt.

ReadOnly: Gibt an, ob die Daten des Berichts schreibgeschützt angezeigt werden (True) oder ob der Bericht im Entwurfsmodus geöffnet werden soll (False).

Beispiel:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
```

```
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

OpenTable-Methode

Öffnet eine Datenbank-Tabelle in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenReport, OpenForm, OpenQuery.

KLASSE:
Desktop

SYNTAX:
OpenTable(TableName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Zeichenfolge, die den Namen der Tabelle angibt, die eingeblendet werden soll.

ReadOnly: Boolescher Ausdruck, der angibt, ob die Tabelle schreibgeschützt (True) oder im Bearbeiten-Modus (False) geladen werden soll.

BEISPIEL:

```
Sub ExampleOpenForm
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateForm( sFormName, sPathName )
Application.CreateReport( sReportName, sPathName )
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
```

```
Application.OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Application.OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Application.DeleteForm( sFormName )
Application.DeleteReport( sReportName )
Application.DeleteQuery( sQueryName )
End Sub
```

OpenTableConnection-Methode

Stellt eine Verbindung zu einer Datenbank-Tabelle her, wenn die zugehörige Datenbank zuvor mit der OpenDatabaseConnection-Methode geöffnet wurde. OpenTableConnection stellt lediglich eine Verbindung zur Tabelle her, sie wird nicht sichtbar geöffnet.

Um den Zugriff auf die Daten der Tabelle zu ermöglichen, muß zunächst auf einen Datensatz positioniert werden. Es ist also erforderlich, nach dem Öffnen der Tabelle vor einem ersten Verwenden der Daten zu einem gewünschten Datensatz zu springen, z.B. mittels DataRecord.

Siehe auch: GetActiveTableConnection, OpenDatabaseConnection, OpenQueryConnection, OpenTable.

KLASSE:
Desktop

SYNTAX:
OpenTableConnection(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Zeichenfolge, die den Namen der Tabelle angibt, die geöffnet werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
```

```
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Application.DataDefinedDatabaseNumber()
sDataName = Application.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = Application.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Application.GetActiveDatabaseConnection
Print sMessage_2 & Application.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Application.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Application.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Application.IsTableReadOnly(sTableName)
vStatus = Application.OpenTableConnection(sTableName)
Print sMessage_4 & Application.GetActiveTableConnection
Print sMessage_5 &  Application.GetDatabaseLimits(1)
Application.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Application.DataFirstRecord
lRecordCount = Application.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Application.DataRecordID
Print sMessage_8 & Application.DataColumn(1)
sFieldName = Application.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Application.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Application.DataNextRecord
vStatus = Application.DataRecord(2)
vStatus = Application.DataPreviousRecord
vStatus = Application.DataLastRecord
End If
Print sMessage_9 & Application.DataPhysicalRecordID
Print sMessage_10 & Application.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
```

```
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Application.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Application.CloseTableConnection(sTableName)
Next iTableNumber
Application.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

OpenTemplate-Methode

Öffnet eine Dokumentvorlage zum Bearbeiten.

KLASSE:
Documents

SYNTAX:
OpenTemplate(Region As String, Name As String)

RÜCKGABE:
Object

PARAMETER:
Region: Zeichenfolge, die den Bereichsnamen der Vorlage angibt. Wird der Parameter "Name" nicht angegeben, dann bestimmt Region nicht den Namen des Vorlagenbereichs, sondern den Namen der Vorlage selbst.

[Name]: Zeichenfolge, die den Namen der Vorlage angibt. Wird dieser Parameter nicht angegeben, dann wird die im ersten Parameter übergebene Zeichenfolge als Vorlagenname interpretiert.

BEISPIEL:

```
sub ExampleOpenTemplate
Const sTempGroup="Standard"
Const sTempName="NewDocTemplate"
Documents.Add("sWriter4")
Wait 500
ActiveWindow.Selection.Insert("Das wird eine Dokumentvorlage")
ActiveWindow.Selection.InsertPara
ActiveDocument.SaveAsTemplate( sTempGroup , sTempName )
ActiveDocument.Close "",FALSE
Wait 2000
Documents.OpenTemplate( sTempGroup ,  sTempName )
ActiveWindow.Selection.Insert("Jetzte wurde die Dokumentvorlage
geändert.")
ActiveWindow.Selection.InsertPara
```

```
ActiveDocument.Save
ActiveDocument.Close "",FALSE
end sub
```

Optimize-Methode

Weist dem aktuellen Fenster die optimale Größe zu.

KLASSE:
TaskWindow

SYNTAX:
Optimize()

BEISPIEL:

```
Sub ExampleOptimize
Documents.Add "sWriter4"
Activewindow.Optimize
End Sub
```

OptionBarVisible-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption der Optionen-Symbolleiste aus StarImpress.

KLASSE:
Application

SYNTAX:
OptionBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Symbolleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleOptionBarVisible
Application.OptionBarVisible = FALSE
Wait 2000
Application.OptionBarVisible = TRUE
End Sub
```

Organizer-Methode

Ruft den Dialog DOKUMENTVORLAGEN VERWALTEN auf. Dieser Dialog ist modal, d.h. das Programm wartet solange, bis der Dialog beendet wird.

KLASSE:
Application

SYNTAX:
Organizer()

BEISPIEL:

```
Sub ExampleOrganizer
Application.Organizer
End Sub
```

Other-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche vertauscht werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

KLASSE:
SearchSettings

SYNTAX:
Other

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, wieviele Zeichen des Suchbegriffs vertauscht sein dürfen, damit die Suche einen Treffer meldet.

BEISPIEL:

```
Sub ExampleSimilarity
Const sSearch="Test" REM Text der gesucht wird.
Const sText="eTxst" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Mode = 1
ActiveWindow.SearchSettings.Similarity = TRUE
ActiveWindow.SearchSettings.Relaxed = TRUE
ActiveWindow.SearchSettings.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.SearchSettings.Shorter = 2 REM Zeichen entfernen
(y)
```

```
ActiveWindow.SearchSettings.Longer = 2 REM Zeichen hinzu-
fügen (z)
ActiveWindow.Search sSearch
MsgBox "gesucht: " & sSearch  & Chr(13) & "gefunden: " & Selec-
tion.Value ,64,"Ähnlichkeitssuche:"
End Sub
```

Eigenschaften und Methoden mit "P"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "P" beginnen.

Palette-Eigenschaft

Liefert oder setzt den Pfad für Farbpaletten. Dieser wird unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
Palette

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad der Paletten angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
```

```
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Parent-Eigenschaft

Ermöglicht bei jedem Objekt den Zugriff auf das übergeordnete Objekt. Das zurückgelieferte Parent-Objekt enthält das Objekt, an dem diese Eigenschaft erfragt wird. So ist z. B. eine DocumentInfo in ihrem zugehörigen Dokument enthalten, falls sie nicht mit "New DocumentInfo" neu angelegt wurde.

OLE-Automation definiert, daß diese Eigenschaft von jedem Objekt unterstützt werden muß.

KLASSE:
Object

SYNTAX:
Parent

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleParent
Dim oObj As Object
Documents.Add("sWriter4")
Selection.Insert( "Eine kleine Vererbungslehre:")
Selection.InsertPara
oObj = Selection.Paragraph
```

```
Selection.Insert( oObj.ClassName )
Selection.InsertPara
oObj = oObj.Parent
Selection.Insert( oObj.ClassName )
Selection.InsertPara
oObj = oObj.Parent
Selection.Insert( oObj.ClassName )
Selection.InsertPara
End Sub
```

ParentWindow-Eigenschaft

Ermöglicht bei einem Window den Zugriff auf das übergeordnete Window. Wenn das Fenster ein TopWindow ist, ergibt der Versuch, das ParentWindow zu erfragen, einen Laufzeitfehler.

KLASSE:
Window

SYNTAX:
ParentWindow

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleParentWindow
Documents.Add( "sWriter4" )
If NOT ActiveWindow.IsTopWindow Then
Print "Das ParentWindow des aktiven Fensters heißt " + Active-
Window.ParentWindow.Name
Else
Print "Das aktive Fenster hat kein ParentWindow!"
End If
End Sub
```

Path-Eigenschaft

Liefert den Pfad eines Dokuments.

KLASSE:
Document

SYNTAX:
Path

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleApplication
Const sDoc="TestDoc.sdw"
Documents.Add("sWriter4")
ActiveDocument.SaveAs( Application.PathSettings.Work & "\" &
sDoc )
Selection.Insert("Name= "& ActiveDocument.Name)
Selection.InsertPara
Selection.Insert("Pfad= "& ActiveDocument.Path)
Selection.InsertPara
Selection.Insert("Pfad mit Name= " &ActiveDocument.FullName)
Selection.InsertPara
End Sub
```

Path-Eigenschaft

Liefert den Pfad der Applikation.

KLASSE:
Application

SYNTAX:
Path

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExamplApplicationPath
Print Application.Path
Print Application.PathSettings.Modules
End Sub
```

PathSettings-Eigenschaft

Liefert die aktuellen Pfadeinstellungen für die verschiedenen Module einer StarOffice-Anwendung als PathSettings-Objekt zurück.

KLASSE:
Application

Syntax:
PathSettings

Rückgabe:
Object

Beispiel:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das ArbeitsVerzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Pattern-Eigenschaft

Bestimmt oder fragt ab, ob bei der Suche nach Vorlagen gesucht werden soll.

KLASSE:
SearchSettings

SYNTAX:
Pattern

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche nach Vorlagen ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExamplePattern
Const sTemp="SuchVorlage"
Documents.Add "sWriter4"
Selection.Insert("Dieser Absatz wird nicht gefunden.")
Selection.InsertPara
Selection.StyleNewByExample( sTemp , 2)
Selection.Insert("Dieser Text hat die Absatzvorlage " & sTemp)
Selection.InsertPara
ActiveWindow.SearchSettings.Similarity = FALSE
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Pattern = TRUE
ActiveWindow.Search sTemp
MsgBox Selection.Value ,64,"Bei Suchen nach Vorlagen wurde
gefunden:"
End Sub
```

Phone-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten die geschäftliche Telefonnummer. Diese wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Phone

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Telefonnummer aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
```

```
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

PhonePrivate-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten die private Telefonnummer. Sie wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
PhonePrivate

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Telefonnummer aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
```

```
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

PlayMacro-Methode

Startet ein StarBasic-Makro. Mit dieser Methode können StarBasic-Makros dynamisch abgespielt werden. Wird das Makro synchron abgespielt, wird so lange gewartet, bis das Makro beendet ist.

KLASSE:
OfficeDocument, Application

SYNTAX:
PlayMacro(Statement As String, Asynchron As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
Statement: Zeichenfolgenausdruck, der die StarBasic-Funktion angibt. Dieser Parameter wird in dem Format "Bibliotheksname.Modulname.Prozedurname" angegeben.

[Asynchron]: Gibt an, ob das Makro asynchron (True) oder synchron (False) abgespielt werden soll. Wird dieser Parameter nicht gesetzt, wird das Makro in der Standardeinstellung synchron abgespielt (Asynchron = False).

Beispiel:

```
Sub ExamplePlayMacro
Dim iState as integer
Documents.Add("sWriter4")
iState = Application.LoadLibrary("GIMMICKS")
REM genau wie ein CALL-Aufruf jedoch auch bei DDE etc.
Application.PlayMacro("Gimmicks.UserFields.StartChangesUser-
fields", False)
iState = Application.UnloadLibrary("GIMMICKS")
end Sub
```

PlugInsActive-Eigenschaft

Bestimmt oder ermittelt, ob PlugIns in Dokumenten aktiviert werden sollen. Ggf. wird eine Ersatzdarstellung angezeigt.

Klasse:
View

Syntax:
PlugInsActive

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob PlugIns aktiviert werden sollen (True) oder nicht (False).

Beispiel:

```
REM Damit dieses Makro funktioniert, muß bereits ein PlugIn
REM vorhanden sein.
Sub ExamplePlugInsActive
ActiveWindow.PlugInsActive = TRUE
End Sub
```

PopupDocInfo-Eigenschaft

Bestimmt oder ermittelt, ob beim Speichern neuer Dokumente automatisch der Dokumentinfo-Dialog gestartet werden soll.

Klasse:
SaveSettings

Syntax:
PopupDocInfo

Rückgabe:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Dokumentinfo-Dialog beim Speichern eingeblendet werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

Position-Eigenschaft

Bestimmt oder ermittelt aus den Benutzerdaten die Position innerhalb einer Firma. Diese wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Position

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Position aus den Benutzerdaten angibt, z. B. "Projektleiter".

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
```

```
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Print-Methode

Druckt das aktuelle Dokument auf dem Drucker aus. Werden dieser Funktion keine Parameter übergeben, so wird der Dialog DRUCKEN aufgerufen.

KLASSE:
View, OfficeDocument

SYNTAX:
Print(from As Integer, to As Integer, copies As Integer)

RÜCKGABE:
Boolean

PARAMETER:
[From]: Beliebiger Integer-Ausdruck, der die Seitenzahl der ersten zu drukkenden Seite definiert. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung From = 1 genommen.

[To]: Beliebiger Integer-Ausdruck, der die Seitenzahl der letzten zu druckenden Seite definiert. Wird dieser Wert nicht übergeben, wird der Standardwert To = 9999 genommen.

[Copies]: Beliebiger Integer-Ausdruck, der die Anzahl der zu druckenden Kopien des Dokuments angibt. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung Copies = 1 genommen.

BEISPIEL:

```
Sub ExamplePrint
ActiveDocument.Print REM ... lädt den Drucken-Dialog
End Sub
```

PrintDefault-Methode

Druckt das aktuelle Dokument, ohne zuvor den DRUCKEN-Dialog aufzurufen.

KLASSE:
View

SYNTAX:
PrintDefault()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExamplePrintDefault
Documents.Add( "sWriter4" )
Selection.Insert "Diese Dokument wird ausgedruckt"
ActiveWindow.PrintDefault
Do Until ActiveDocument.IsPrinting
Wait 300
Loop
End Sub
```

Printer-Eigenschaft

Bestimmt oder ermittelt den aktuellen Drucker. Dieser wird über die im System verwendete Bezeichnung angesprochen.

KLASSE:
View

SYNTAX:
Printer

RÜCKGABE:
String

PARAMETER:
[= PrinterName]: Zeichenfolgenausdruck, der exakt die System-Bezeichnung des Druckers wiedergibt.

BEISPIEL:

```
Sub ExamplePrint
Const sMessage_1 = "Dieses Dokument wird zweimal auf dem REM "
Const sMessage_2 = "REM  ausgedruckt."
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.Insert ActiveWindow.Printer
Selection.Insert sMessage_2
REM Die erste Seiten mit 2 Kopien ausdrucken
ActiveWindow.Print( 1, 1, 2 )
End Sub
```

PrintOut-Methode

Druckt das aktuelle Dokument auf dem Drucker aus. Werden dieser Funktion keine Parameter übergeben, so wird der Dialog Drucken aufgerufen.

Klasse:
OfficeDocument

Syntax:
PrintOut(from As Integer, to As Integer, copies As Integer)

Parameter:
[From]: Beliebiger Integer-Ausdruck, der die Seitenzahl der ersten zu druckenden Seite definiert. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung From = 1 genommen.

[To]: Beliebiger Integer-Ausdruck, der die Seitenzahl der letzten zu druckenden Seite definiert. Wird dieser Wert nicht übergeben, wird als Standardwert To = 9999 genommen.

[Copies]: Beliebiger Integer-Ausdruck, der die Anzahl der zu druckenden Exemplare angibt. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung Copies = 1 genommen.

Beispiel:

```
Sub ExampleIsPrinting
Documents.Add( "sWriter4" )
Selection.Insert "Dieses Dokument wird ausgedruckt"
ActiveDocument.PrintOut( 1, 1 )
Do Until ActiveDocument.IsPrinting
Wait 300
Loop
End Sub
```

Eigenschaften und Methoden mit "Q"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Q" beginnen.

QueryAutoSave-Eigenschaft

Bestimmt oder ermittelt, ob beim automatischen, periodischen Speichern der Dokumente eine Rückfrage an den Benutzer erfolgen soll.

Klasse:
SaveSettings

SYNTAX:
QueryAutoSave

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Speichern mit (True) oder ohne (False) Rückfrage erfolgen soll.

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

Quit-Methode

Beendet das StarOffice-Programm.

KLASSE:
Application

SYNTAX:
Quit()

BEISPIEL:

```
Sub  ExampleQuit
Application.Quit
End Sub
```

Eigenschaften und Methoden mit "R"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "R" beginnen.

ReadOnly-Eigenschaft

Ermittelt, ob das aktuelle Dokument schreibgeschützt ist. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
OfficeDocument

SYNTAX:
ReadOnly

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleReadOnly
Const sDoc="NurLesen.sdw"
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert("Wird gespeichert.")
ActiveWindow.Selection.InsertPara
ActiveDocument.SaveAs Application.PathSettings.Work & "\" &
sDoc
Documents.Open( ActiveDocument.FullName )
If ActiveDocument.ReadOnly then
   MsgBox "Ist schreibgeschützt",64,"Dokument: " & ActiveDocu-
ment.Name
else
   ActiveWindow.Selection.Insert("Jetzt wird da Dokument geän-
dert.")
   ActiveWindow.Selection.InsertPara
   ActiveDocument.Save
end if
end sub
```

Receiver-Eigenschaft

Bestimmt oder ermittelt den Empfänger der Nachricht.

KLASSE:
MessageDocument

SYNTAX:
Receiver

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Empfänger angibt.

BEISPIEL:

```
sub Main
Documents.Add "Message"
Print ActiveWindow.TopWindow.Document.Receiver
end sub
```

Recording-Eigenschaft

Ermittelt, ob die Makroaufzeichnung derzeitig aktiv ist. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
Application

SYNTAX:
Recording

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
sub ExampleRecording
Print Application.Recording
end sub
```

Redo-Methode

Wiederholt die vorher rückgängig gemachte Aktion. Ist eine Wiederholung nicht möglich, kommt es zu einem Laufzeitfehler.

Siehe auch: Repeat, Undo, RedoAction.

KLASSE:
Window

SYNTAX:
Redo()

BEISPIEL:

```
Sub ExampleDoUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite "
Const sMessage_2 = "Neue Seite "
Const sMessage_3 = "Kein weiteres Undo möglich "
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sMessage_5 = "RedoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.DoUndo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
Print sMessage_5 & ActiveWindow.Selection.RedoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.DoUndo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

RedoAction-Eigenschaft

Ermittelt die im Menü BEARBEITEN unter dem Menüpunkt WIEDERHOLEN aufgeführte Beschreibung der zuvor rückgängig gemachten Aktion. Ist eine Wiederholung nicht möglich, kommt es zu einem Laufzeitfehler.

Siehe auch: UndoAction, RepeatAction, Redo.

KLASSE:
Window

SYNTAX:
RedoAction

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDoUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite "
Const sMessage_2 = "Neue Seite "
Const sMessage_3 = "Kein weiteres Undo möglich "
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sMessage_5 = "RedoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.DoUndo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
Print sMessage_5 & ActiveWindow.Selection.RedoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.DoUndo
StarWriter.UndoCount = 50
Exit Sub
```

```
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

RegExp-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche nach regulären Ausdrücken gesucht wird.

KLASSE:
SearchSettings

SYNTAX:
RegExp

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche mit Hilfe von Platzhaltern erfolgen soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleRegExp
Const sText = "Text"
Documents.Add "sWriter4"
ActiveWindow.SearchSettings.Similarity = FALSE
ActiveWindow.SearchSettings.Pattern = FALSE
Selection.Insert sText
Selection.InsertPara
Selection.Insert "Noch ein " & sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.RegExp = TRUE
ActiveWindow.Search "^" & sText REM  Findet nur, wenn es am
Zeilenanfang steht
MsgBox "Am Absatzbeginn gefunden: " & Selection.Value ,64,"Mit
regulären Audrücken suchen"
End Sub
```

Relaxed-Eigenschaft

Bestimmt oder ermittelt, ob die drei Randbedingungen der Shorter-, Longer- und Other-Eigenschaft für die Ähnlichkeitssuche kombiniert werden sollen.

Siehe auch: Shorter, Longer, Other.

KLASSE:
SearchSettings

SYNTAX:
Relaxed

RÜCKGABE:
Boolean

PARAMETER:
[= Switch:] Boolescher Ausdruck, der angibt, ob die Ähnlichkeitssuche mit einer Kombination der drei Randbedingungen erfolgen soll:

True: Der Suchbegriff gilt dann als gefunden, wenn das Wort im Dokument durch eine beliebige Kombination der drei Randbedingungen erzeugt werden kann.

False: Die Ähnlichkeitssuche wird ohne eine Kombination der drei Randbedingungen durchgeführt.

BEISPIEL:

```
Sub ExampleSimilarity
Const sSearch="Test" REM Text der gesucht wird.
Const sText="eTxst" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Mode = 1
ActiveWindow.SearchSettings.Similarity = TRUE
ActiveWindow.SearchSettings.Relaxed = TRUE
ActiveWindow.SearchSettings.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.SearchSettings.Shorter = 2 REM Zeichen entfernen
(y)
ActiveWindow.SearchSettings.Longer = 2 REM Zeichen hinzu-
fügen (z)
ActiveWindow.Search sSearch
MsgBox "gesucht: " & sSearch  & Chr(13) & "gefunden: " & Selec-
tion.Value ,64,"Ähnlichkeitssuche:"
End Sub
```

Reload-Methode

Lädt das in dem Window dargestellte Dokument neu. Wurde es verändert, wird die Änderung damit rückgängig gemacht, ggf. kann dadurch auch eine neuere Version vom Medium (z.B. Internet) geladen werden.

KLASSE:
BrowseWindow

SYNTAX:
Reload()

Rückgabe:
Boolean

Beispiel:

```
sub ExampleReload
Documents.Open "http://www.stardivision.de"
wait 1000
Activewindow.Reload
Activewindow.ReloadForEdit
end sub
```

ReloadForEdit-Methode

Schaltet das in dem Window dargestellte Dokument vom schreibgeschützten Modus in den Bearbeiten-Modus und umgekehrt. Dazu wird es ggf. automatisch neu geladen, z.B. wenn im File-System eine neuere Version vorgefunden wird.

Klasse:
BrowseWindow

Syntax:
ReloadForEdit(Editable As Boolean)

Rückgabe:
Boolean

Parameter:
[Editable]: Gibt an, ob das Dokument im Bearbeiten-Modus geladen werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird kein Parameter angegeben, wird der aktuelle Modus einfach umgeschaltet, also ein nicht schreibgeschütztes Dokument wird schreibgeschützt geladen und umgekehrt.

Beispiel:

```
sub ExampleReload
Documents.Open "http://www.stardivision.de"
wait 1000
Activewindow.Reload
Activewindow.ReloadForEdit
end sub
```

Remove-Methode

Löscht ein Element aus der Collection.

Bei Anwendung der Remove-Methode auf die Documents-Collection ist zu beachten, daß Remove nicht auf ein Objekt der Klasse ExplorerDocument wirken kann. Beinhaltet die Documents-Collection ein Objekt dieser Klasse, dann wird es per Remove nicht gelöscht. Welcher Klassenname einem Objekt zugeordnet ist, läßt sich durch die ClassName-Eigenschaft abfragen.

KLASSE:
Collection

SYNTAX:
Remove(Index As String)

PARAMETER:
Index: Enthält den Namen oder den Index des zu löschenden Elements aus der Collection.

BEISPIEL:

```
sub ExampleWindow
Dim sWin As String
Const sMessage1 = "Das Fenster wird minimiert, "
Const sMessage2 = "maximiert und in den Zustand bei seiner
Erzeugung gebracht."
Const sMessage3 = "Danach wird es entfernt."
Documents.Add "sWriter4"
ActiveWindow.Selection.Insert(sMessage1 & sMessage2 & sMes-
sage3)
Wait 2000
sWin =  ActiveWindow.Name
Windows( sWin ).Minimized = True
Wait 2000
Windows( sWin ).Maximized = True
Wait 2000
Windows( sWin ).Restore
Wait 2000
Windows.Remove( sWin )
end sub
```

RemoveLibrary-Methode

Entfernt eine Bibliothek aus der Verwaltung der StarBasic-Bibliotheken. Dabei wird die Bibliothek auch "physikalisch" aus dem Basic entfernt.

Siehe auch: AddLibrary, LoadLibrary, UnloadLibrary.

KLASSE:
OfficeDocument, Application

SYNTAX:
RemoveLibrary(LibraryName As String)

RÜCKGABE:
Boolean

PARAMETER:
LibraryName: Name der Bibliothek, die entfernt werden soll.

BEISPIEL:

```
Sub ExampleAddLibrary
Const sLibName = "NeueBibliotek"
REM Const BiblioDoc = REM Pfad und Dateiname von dem Dokument
das die Bibliothek enthält
Dim vStatus as Variant
vStatus = Application.AddLibrary(BiblioDoc , 1, TRUE, sLibName
)
vStatus = Application.RemoveLibrary( sLibName )
End Sub
```

Repeat-Methode

Mit dieser Methode wird die letzte Benutzeraktion in dem aktuellen Window wiederholt, z.B. eine Zeichenattributierung. Ist es nicht möglich, die letzte Aktion zu wiederholen, dann kommt es bei Aufruf dieser Methode zu einem Laufzeitfehler.

Siehe auch: Undo, Redo, RepeatAction.

KLASSE:
Window

SYNTAX:
Repeat()

BEISPIEL:

```
Sub ExampleRepeatAction
Const sMessage_1 = "Dieser Text wird durch RepeatAction mehr-
fach eingefügt"
Const sMessage_2 = "RepeatAction liefert den Text: "
```

```
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveWindow.Paste REM Diese Action wird mehrfach wiederholt
Print sMessage_2 & ActiveWindow.Selection.RepeatAction
ActiveWindow.Selection.Repeat
ActiveWindow.Selection.Repeat
End Sub
```

RepeatAction-Eigenschaft

Ermittelt die im Menü BEARBEITEN unter dem Menüpunkt LETZTER BEFEHL aufgeführte Beschreibung der zuletzt ausgeführten Aktion, die wiederholt werden kann. Ist es nicht möglich, den letzten Befehl zu wiederholen, dann kommt es bei Aufruf dieser Eigenschaft zu einem Laufzeitfehler.

Siehe auch: UndoAction, RedoAction, Repeat.

KLASSE:
Window

SYNTAX:
RepeatAction

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleRepeatAction
Const sMessage_1 = "Dieser Text wird durch RepeatAction mehr-
fach eingefügt"
Const sMessage_2 = "RepeatAction liefert den Text: "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveWindow.Paste REM Diese Action wird mehrfach wiederholt
Print sMessage_2 & ActiveWindow.Selection.RepeatAction
ActiveWindow.Selection.Repeat
ActiveWindow.Selection.Repeat
End Sub
```

Restore-Methode

Stellt die ursprüngliche Größe eines maximierten Fensters wieder her.

KLASSE:
TaskWindow

SYNTAX:
Restore()

BEISPIEL:

```
sub ExampleWindow
Dim sWin As String
Const sMessage1 = "Das Fenster wird minimiert, "
Const sMessage2 = "maximiert und in den Zustand bei seiner
Erzeugung gebracht."
Const sMessage3 = "Danach wird es entfernt."
Documents.Add "sWriter4"
ActiveWindow.Selection.Insert(sMessage1 & sMessage2 & sMes-
sage3)
Wait 2000
sWin =  ActiveWindow.Name
Windows( sWin ).Minimized = True
Wait 2000
Windows( sWin ).Maximized = True
Wait 2000
Windows( sWin ).Restore
Wait 2000
Windows.Remove( sWin )
end sub
```

Resume-Methode

Läßt eine unterbrochene Fortschrittsanzeige in der Statuszeile für einen auf dem Rechner ablaufenden Vorgang wieder weiterlaufen. Die beim Start gesetzten Optionen (Wartecursor, Locking etc.) bleiben gültig.

Siehe auch: Start, Stop, Suspend, Value.

KLASSE:
Progress

SYNTAX:
Resume()

BEISPIEL:

```
sub ExampleProgress
Const sMessage="Hier wird gearbeitet"
Dim aProgress As New Progress
aProgress.Start(sMessage , 10000)
For aProgress.Value = 0 To 10000
```

```
If aProgress.Value = 5000 Then
aProgress.Suspend REM ... dringende unterbrechung
Wait 3000
aProgress.Resume
End If
Next aProgress.Value
aProgress.Stop
End Sub
```

Revision-Eigenschaft

Enthält die Information der Version eines Dokuments. Diese wird aus den Dokumenteigenschaften gelesen (Label "Version").

KLASSE:
DocumentInfo

SYNTAX:
Revision

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
```

```
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

RightLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die rechte Umrandungslinie beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

Siehe auch: TopLine, BottomLine, LeftLine.

KLASSE:
Border

SYNTAX:
RightLine

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
```

```
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

Eigenschaften und Methoden mit "S"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "S" beginnen.

Save-Methode

Speichert die mit Load geladene Dokumentinfo wieder in die dort benannte Datei zurück.

KLASSE:
DocumentInfo

SYNTAX:
Save()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
```

```
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Save-Methode

Speichert das aktuelle Dokument ab. Hat das Dokument noch keinen Namen, kommt es zu einem Laufzeitfehler.

KLASSE:
OfficeDocument

SYNTAX:
Save()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleModified
Const sNewName = "NewName" REM An diesen Namen kommt noch eine
Ziffer als Versionnummer dran
Dim sDoc As String,iCount As Integer
Documents.Add("sWriter4")
Selection.Insert("Dieses Dokument wurde erzeugt um gespeichert
zu werden.")
Selection.InsertPara
For iCount = 1 to 12
   sDoc = Application.PathSettings.Work & "\" &_
          sNewName & ((Val( Right( ActiveDocument.Name ,1 )) +
1) Mod 10) & ".sdw"
   If ActiveDocument.Modified Then ActiveDocument.SaveAs sDoc
   Selection.Insert("Versionsnummer hochzählen " & ((Val(
Right( ActiveDocument.Name ,1 )) + 1) Mod 10) )
   Selection.InsertPara
Next iCount
If NOT ActiveDocument.Saved Then ActiveDocument.Save
End Sub
```

SaveAll-Methode

Speichert alle modifizierten Dokumente.

KLASSE:
Documents

SYNTAX:
SaveAll()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDocumentsOpen
Const sDoc1 = "Test1"
Const sDoc2 = "Test2"
Const sOfficeFilter="soffice4: StarOffice Dokumente"
Dim sPath1 as string, sPath2 as string
sPath1 = Application.PathSettings.Work & "\" & sDoc1 & ".sdw"
sPath2 = Application.PathSettings.Work & "\" & sDoc2 & ".sdw"
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 1"
Activedocument.SaveAs sPath1
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 2"
Activedocument.SaveAs sPath2
```

```
Windows(sDoc1 & ":1").Selection.Insert ": geändert" REM jeweils
die erste Sicht bearbeiten
Windows(sDoc2 & ":1").Selection.Insert ": geändert"
Documents.SaveAll
Documents(sDoc1).Close False, ""
Documents(sDoc2).Close False, ""
Documents.Open sPath1,sOfficeFilter  REM Öffnen mit Filterliste
End Sub
```

SaveAs-Methode

Speichert das aktuelle Dokument unter einem bestimmten Namen ab.

KLASSE:
OfficeDocument

SYNTAX:
SaveAs(FileName As String, FilterName As String, PassWord As String, FilterOptions As String)

RÜCKGABE:
Boolean

PARAMETER:
FileName: Dateiname des Dokumentes inkl. Pfad.

[FilterName]: Filtername des Dokumentes, der das Format bestimmt, in dem das Dokument abgespeichert werden soll. Es gelten die Einstellungen der Dateiauswahl im StarOffice. Wird kein Filtername angegeben, wird der bevorzugte Filter des jeweiligen Formats verwendet (der erste in der Filterliste).

[PassWord]: Paßwort für den geschützten Zugriff auf das Dokument. Dieser Parameter ist optional.

[FilterOptions]: Applikationsspezifische Filterangabe, die im Office nicht ausgewertet wird. Dieser optionale Parameter ist hier ohne Bedeutung, da es im StarOffice-Modul keinen Filter gibt, der diesen Parameter verwendet.

BEISPIEL:

```
Sub ExampleDocumentsOpen
Const sDoc1 = "Test1"
Const sDoc2 = "Test2"
Const sOfficeFilter="soffice4: StarOffice Dokumente"
Dim sPath1 as string, sPath2 as string
sPath1 = Application.PathSettings.Work & "\" & sDoc1 & ".sdw"
sPath2 = Application.PathSettings.Work & "\" & sDoc2 & ".sdw"
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 1"
Activedocument.SaveAs sPath1
```

```
Documents.Add "sWriter4"
Selection.Insert "Neues Dokument 2"
Activedocument.SaveAs sPath2
Windows(sDoc1 & ":1").Selection.Insert ": geändert" REM jeweils
die erste Sicht bearbeiten
Windows(sDoc2 & ":1").Selection.Insert ": geändert"
Documents.SaveAll
Documents(sDoc1).Close False, ""
Documents(sDoc2).Close False, ""
Documents.Open sPath1,sOfficeFilter  REM Öffnen mit Filterliste
End Sub
```

SaveAsTemplate-Methode

Speichert das aktuelle Dokument als Dokumentvorlage.

KLASSE:
OfficeDocument

SYNTAX:
SaveAsTemplate(TemplateRegion As String, TemplateName As String)

RÜCKGABE:
Boolean

PARAMETER:
TemplateRegion: Zeichenfolgenausdruck, der den Bereich angibt, dem die Dokumentvorlage zugewiesen werden soll.

TemplateName: Zeichenfolgenausdruck, der den Dateinamen angibt, unter dem die Dokumentvorlage gespeichert werden soll.

BEISPIEL:

```
sub ExampleOpenTemplate
Const sTempGroup="Standard"
Const sTempName="NewDocTemplate"
Documents.Add("sWriter4")
Wait 500
ActiveWindow.Selection.Insert("Das wird eine Dokumentvorlage")
ActiveWindow.Selection.InsertPara
ActiveDocument.SaveAsTemplate( sTempGroup , sTempName )
ActiveDocument.Close "",FALSE
Wait 2000
Documents.OpenTemplate( sTempGroup ,  sTempName )
ActiveWindow.Selection.Insert("Jetzt wurde die Dokumentvorlage
geändert.")
ActiveWindow.Selection.InsertPara
ActiveDocument.Save
ActiveDocument.Close "",FALSE
end sub
```

SaveConfiguration-Methode

Speichert die aktuelle Konfiguration der Applikation unter dem angegebenen Dateinamen.

Siehe auch: LoadConfiguration

KLASSE:
Application

SYNTAX:
SaveConfiguration(FileName As String)

PARAMETER:
FileName: Zeichenfolgenausdruck, der den Dateinamen oder Namen und Pfad der Datei (*.cfg) angibt, unter dem die Konfiguration gespeichert werden soll. Wird nur der Dateiname angegeben, wird die Datei in dem Verzeichnis gespeichert, dessen Pfad unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE als Konfigurations-Pfad eingestellt ist.

BEISPIEL:

```
Sub ExampleConfigurations
Const sConfig="MyConfig.cfg"  REM ... wird im Config-
Verzeichnis abgespeichert
Application.SaveConfiguration( sConfig )
Application.LoadAccel( sConfig )
Application.LoadMenu( sConfig )
Application.LoadStatusBar( sConfig )
Application.LoadToolBox( sConfig )
REM Die gesamte Konfiguration laden
Application.LoadConfiguration( sConfig )
End Sub
```

Saved-Eigenschaft

Ermittelt, ob das aktuelle Dokument gespeichert wurde, nachdem die letzten Änderungen daran vorgenommen wurden. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
OfficeDocument

SYNTAX:
Saved

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleModified
Const sNewName = "NewName" REM An diesen Namen kommt noch eine
Ziffer als Versionnummer dran
Dim sDoc As String,iCount As Integer
Documents.Add("sWriter4")
Selection.Insert("Dieses Dokument wurde erzeugt um gespeichert
zu werden.")
Selection.InsertPara
For iCount = 1 to 12
   sDoc = Application.PathSettings.Work & "\" &_
          sNewName & ((Val( Right( ActiveDocument.Name ,1 )) +
1) Mod 10) & ".sdw"
   If ActiveDocument.Modified Then ActiveDocument.SaveAs sDoc
   Selection.Insert("Versionsnummer hochzählen " & ((Val(
Right( ActiveDocument.Name ,1 )) + 1) Mod 10) )
   Selection.InsertPara
Next iCount
If NOT ActiveDocument.Saved Then ActiveDocument.Save
End Sub
```

SaveSettings-Eigenschaft

Liefert die aktuellen Einstellungen zum Speichern im StarOffice als SaveSettings-Objekt zurück.

KLASSE:
Application

SYNTAX:
SaveSettings

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
```

```
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

SaveWorkingset-Eigenschaft

Bestimmt oder ermittelt, ob beim Start des StarOffice die Dokumente wieder geladen werden sollen, die beim letzten Beenden der Anwendung geladen waren.

KLASSE:
SaveSettings

SYNTAX:
SaveWorkingset

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die beim letzten Beenden der Anwendung geöffneten Dokumente beim erneuten Start geladen werden (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSaveSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Ausdruck der Sicherungsoptionen")
.StyleApply("Überschrift 1",2) REM Als Überschrift formatieren
.InsertPara
.Insert("Die Option Automatisch Speichern wurde " &_
IIF(Application.SaveSettings.AutoBackup,"","nicht ") &_
"gewählt."
```

```
.InsertPara
.Insert("Das Intervall für automatisches Speichern ist " &_
Application.SaveSettings.AutoSaveInterval &_
" Sekunden."
.InsertPara
.Insert("Die Grafiken werden " &_
IIF(Application.SaveSettings.GraphicsIndep ,"","nicht " ) &_
"systemunabhängig gespeichert."
.InsertPara
.Insert("Die Dokumentinformationen werden beim Speichern " &_
IIF( Application.SaveSettings.PopupDocInfo ,"","nicht ") &_
"nachgefragt"
.InsertPara
.Insert("Beim automatischen Speichern wird " &_
IIF(Application.SaveSettings.QueryAutoSave,"","nicht " ) &_
"nachgefragt."
.InsertPara
.Insert("Die Arbeitsansicht wird " &_
IIF(Application.SaveSettings.SaveWorkingset ,"","nicht " ) &_
"mit gespeichert."
.InsertPara
End With
End Sub
```

ScEditOptions-Methode

Ruft den Dialog OPTIONEN TABELLENDOKUMENT auf.

KLASSE:
Desktop

SYNTAX:
ScEditOptions()

BEISPIEL:

```
Sub ExampleSimEditOptions
Application.ScEditOptions
End Sub
```

SdEditOptions-Methode

Ruft den Dialog OPTIONEN PRÄSENTATION auf.

KLASSE:
Desktop

SYNTAX:
SdEditOptions()

Beispiel:

```
Sub ExampleSdEditOptions
Application.SdEditOptions
End Sub
```

SearchDialog-Eigenschaft

Bestimmt oder ermittelt, ob der Dialog Suchen & Ersetzen eingeblendet wird.

Klasse:
Application

Syntax:
SearchDialog

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Suchen-Dialog ein- (True) oder ausgeblendet (False) ist.

Beispiel:

```
Sub ExampleSearchDialog
Application.SearchDialog = TRUE
wait 2000
Application.SearchDialog = FALSE
end sub
```

SearchOriginal-Methode

Springt im Office zur Originaldatei einer Verknüpfung, indem das Verzeichnis der Originaldatei in einem neuen Fenster geöffnet und die Originaldatei markiert wird.

Klasse:
Application

Syntax:
SearchOriginal(FileName As String)

Parameter:
FileName: Name und Pfad der Datei-Verknüpfung.

Beispiel:

```
REM Die Verknüpfung muß bereits angelegt worden sein.
Application.SearchOriginal(
"D:\Office40\Explorer\Bookmarks\Verknüpfung mit test.url" )
```

Selection-Eigenschaft

Liefert die aktuelle Selektion als Objekt zurück und ermöglicht somit einen Zugriff auf das Selection-Objekt. Ist beispielsweise Text markiert oder befindet sich der Cursor in einem Text, liefert diese Eigenschaft ein Text-Objekt, befindet sich der Cursor in einer Tabelle, liefert sie das TextInTable-Objekt.

Klasse:
Window

Syntax:
Selection

Rückgabe:
Object

Beispiel:

```
Sub ExampleWindowSelection
Documents.Add "sWriter4"
Selection.InsertTable(3,3)
MsgBox ActiveWindow.Selection.Name,0,"Name des Selektionstyps:"
End Sub
```

Selection-Eigenschaft

Liefert die aktuelle Selektion als Objekt zurück und ermöglicht somit einen Zugriff auf das Selection-Objekt. Ist beispielsweise Text markiert oder befindet sich der Cursor in einem Text, liefert diese Eigenschaft ein Text-Objekt, befindet sich der Cursor in einer Tabelle, liefert sie das TextInTable-Objekt.

Klasse:
Application

Syntax:
Selection

Rückgabe:
Object

Beispiel:

```
Sub ExampleApplicationSelection
Documents.Add "sWriter4"
Selection.InsertTable(3,3)
MsgBox Application.Selection.Name,0,"Name des Selektionstyps:"
End Sub
```

SelectionText-Eigenschaft

Liefert die aktuelle Selektion als String zurück.

Klasse:
View

Syntax:
SelectionText

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleSelectionText
Documents.Add "sWriter4"
Selection.Insert "Dieser Text wird selektiert und ausgelesen."
MsgBox Activewindow.SelectionTextExt,0,"SelectionTextExt"
Selection.SelectAll REM ... muß vorher selektiert werden
MsgBox Activewindow.SelectionText ,0,"SelectionText"
End Sub
```

SelectionTextExt-Eigenschaft

Liefert das komplette Wort, in dem sich der Cursor befindet als String zurück. Es darf keine Selektion (Markierung) vorhanden sein.

Klasse:
View

Syntax:
SelectionTextExt

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

SendMail-Methode

Versendet eine E-Mail.

Klasse:
Application

SYNTAX:
SendMail(Receiver As String, Subject As String, MailText As String, Priority As Long)

RÜCKGABE:
Boolean

PARAMETER:
Receiver: Empfängerliste mit Kommata getrennt. Zulässig sind news:URLs und mailto:URLs.

Subject: Der Titel der Nachricht

MailText: Der Inhalt der Nachricht

[Priority]: Priorität 1-5 (1 = Hoch bis 5 = Niedrig). Dieser Parameter ist optional, wird er nicht übergeben, wird die Standardeinstellung Priority = 3 (Normal) genommen.

BEISPIEL:

```
Sub ExampleSendMail
Dim sId As String
Const sSubject="Testmail an mich selbst."
Const sContent="Damit du einmal eine Mail bekommst."
sId= Application.Address.ID
Application.SendMail( "mailto:" & sID, "Subject: "&  sSubject ,
sContent ,3 )
End Sub
```

SetFrameContent-Methode

Bestimmt den Inhalt eines Frames in einem FrameSet.

Siehe auch: FrameContent, SetFrameName.

KLASSE:
FrameSetView

SYNTAX:
SetFrameContent(FrameName As String, FrameContent As String)

PARAMETER:
FrameName: Zeichenfolge, die die Nummer oder den Namen des Frames definiert.

FrameContent: Zeichenfolge, die den initialen Inhalt (URL) definiert.

Beispiel:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
```

```
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

SetFrameName-Methode

Bestimmt den Namen eines Frames in einem FrameSet.

Siehe auch: FrameName, SetFrameContent.

KLASSE:
FrameSetView

SYNTAX:
SetFrameName(OldName As String, NewName As String)

PARAMETER:
OldName: Nummer oder alter Name des Frames.

NewName: Neue Bezeichnung des Frames.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
```

```
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
```

```
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

Shorter-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche aus einem Wort entfernt werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

KLASSE:
SearchSettings

SYNTAX:
Shorter

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, um wieviele Zeichen das Wort im Dokument kürzer sein darf als der Suchbegriff, um noch als gleich zu gelten.

BEISPIEL:

```
Sub ExampleSimilarity
Const sSearch="Test" REM Text der gesucht wird.
Const sText="eTxst" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Mode = 1
ActiveWindow.SearchSettings.Similarity = TRUE
ActiveWindow.SearchSettings.Relaxed = TRUE
ActiveWindow.SearchSettings.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.SearchSettings.Shorter = 2 REM Zeichen entfernen
(y)
ActiveWindow.SearchSettings.Longer = 2 REM Zeichen hinzu-
fügen (z)
ActiveWindow.Search sSearch
MsgBox "gesucht: " & sSearch  & Chr(13) & "gefunden: " & Selec-
tion.Value ,64,"Ähnlichkeitssuche:"
End Sub
```

ShowAddressBook-Methode

Blendet das Adreßbuch ein.

KLASSE:
MessageDocument

SYNTAX:
ShowAddressBook()

RÜCKGABE:
Boolean

BEISPIEL:

```
REM Erstellt ein neues Nachrichten-Dokument und öffnet das
Adreßbuch
REM in dem Beamer.
Sub ExampleShowAddressBook
Documents.Add( "Message" )
ActiveWindow.TopWindow.Document.ShowAddressBook
End Sub
```

ShowBrowser-Eigenschaft

Bestimmt oder ermittelt, ob der Eigenschaften-Dialog für Kontrollfelder eingeblendet ist.

KLASSE:
Desktop

SYNTAX:
ShowBrowser

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Eigenschaften-Dialog ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleShowBrowser
Application.ShowBrowser = TRUE
Print "Es wurde der Browser eingeschaltet"
Application.ShowBrowser = FALSE
Print "...und wieder ausgeschaltet"
End Sub
```

ShowQuery-Methode

Zeigt eine Datenbank-Abfrage im Beamer an. Die Abfrage muß vorher für diese Datenbank definiert worden sein, und die Datenbank muß geöffnet sein.

Konnte die Methode erfolgreich ausgeführt werden, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
Desktop

SYNTAX:
ShowQuery(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Zeichenfolge, die den Namen der Abfrage angibt.

BEISPIEL:

```
Sub ExampleShowQuery
Const sQueryName = "Abfrage"
Dim sDataName As String, sTableName As String
Dim vStatus As Variant
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = Application.DataGetTableName(sDataName, 1)
Application.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Application.ShowQuery( sQueryName )
Application.DeleteQuery( sQueryName )
End Sub
```

ShowStatement-Methode

Zeigt eine SQL-Selektion im Beamer an. Voraussetzung ist, daß vor Aufruf dieser Methode die zugehörige Datenbank, an der die SQL-Abfrage durchgeführt wird, mittels OpenDatabaseConnection geöffnet wurde.

Siehe auch: OpenDatabaseConnection

KLASSE:
Desktop

SYNTAX:
ShowStatement(StatementName As String)

RÜCKGABE:
Boolean

PARAMETER:
StatementName: Zeichenfolge, die einen gültigen SQL-Ausdruck aufnimmt, z. B. "Select * from Adressen".

BEISPIEL:

```
Sub ExampleShowStatement
Dim sDataName As String, sTabName As String
Dim vStatus As Variant
Documents.Add( "sWriter4" )
sDataName = Application.DataDefinedDatabase(1)
vStatus = Application.OpenDatabaseConnection(sDataName)
sTabName = Application.DataGetTableName(sDataName, 1)
Application.ShowStatement("SELECT * FROM " & sTabName)
End Sub
```

SimEditOptions-Methode

Ruft den Dialog OPTIONEN BILD auf.

KLASSE:
Desktop

SYNTAX:
SimEditOptions()

BEISPIEL:

```
Sub ExampleSimEditOptions
Application.SimEditOptions
End Sub
```

Similarity-Eigenschaft

Bestimmt oder ermittelt, ob die Ähnlichkeitssuche aktiv ist.

KLASSE:
SearchSettings

SYNTAX:
Similarity

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Ähnlichkeitssuche ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
Sub ExampleSimilarity
Const sSearch="Test" REM Text der gesucht wird.
Const sText="eTxst" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.Mode = 1
ActiveWindow.SearchSettings.Similarity = TRUE
ActiveWindow.SearchSettings.Relaxed = TRUE
ActiveWindow.SearchSettings.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.SearchSettings.Shorter = 2 REM Zeichen entfernen
(y)
ActiveWindow.SearchSettings.Longer = 2 REM Zeichen hinzu-
fügen (z)
ActiveWindow.Search sSearch
MsgBox "gesucht: " & sSearch  & Chr(13) & "gefunden: " & Selec-
tion.Value ,64,"Ähnlichkeitssuche:"
End Sub
```

SmEditOptions-Methode

Ruft den Dialog Optionen Formel zur Festlegung der Druckoptionen von Formeln auf.

Klasse:
Desktop

Syntax:
SmEditOptions()

Beispiel:

```
Sub ExampleSmEditOptions
Application.SmEditOptions
End Sub
```

SplitHorizontal-Methode

Teilt einen Frame in einem Frameset horizontal. Diese Funktion entspricht dem Befehl Horizontal teilen im Kontextmenü eines Frames in einem Frameset.

Siehe auch: SplitVertical, SplitParentHorizontal.

Klasse:
FrameSetView

SYNTAX:
SplitHorizontal(Frame As String)

RÜCKGABE:
Object

PARAMETER:
Frame: Zeichenfolge, die die Nummer oder den Namen des Frames angibt.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
```

```
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

SplitParentHorizontal-Methode

Teilt ein SubFrameset in einem Frameset horizontal. Als Rückgabewert erhält man den neu erzeugten Frame in Form eines Window-Objekts. Diese Funktion entspricht dem Befehl HORIZONTAL EINFÜGEN im Kontextmenü eines Frames in einem SubFrameset.

Siehe auch: SplitParentVertical, SplitHorizontal.

KLASSE:
FrameSetView

SYNTAX:
SplitParentHorizontal(Frame As String)

RÜCKGABE:
Object

PARAMETER:
Frame: Name eines Frames in dem SubFrameset, das geteilt werden soll. Der Frame muß ein direkter Bestandteil dieses SubFramesets sein.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
```

```
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

SplitParentVertical-Methode

Teilt ein SubFrameset in einem Frameset horizontal. Als Rückgabewert erhält man den neu erzeugten Frame in Form eines Window-Objekts. Diese Funktion entspricht dem Befehl VERTIKAL EINFÜGEN im Kontextmenü eines Frames in einem SubFrameset.

Siehe auch: SplitVertical, SplitParentHorizontal.

KLASSE:
FrameSetView

SYNTAX:
SplitParentVertical(Frame As String)

RÜCKGABE:
Object

PARAMETER:
Frame: Name eines Frames in dem SubFrameset, das geteilt werden soll. Der Frame muß ein direkter Bestandteil dieses SubFramesets sein.

Beispiel:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
```

```
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

SplitVertical-Methode

Teilt einen Frame in einem Frameset vertikal. Diese Funktion entspricht dem Befehl VERTIKAL TEILEN im Kontextmenü eines Frames in einem Frameset.

Siehe auch: SplitParentVertical, SplitHorizontal.

KLASSE:
FrameSetView

SYNTAX:
SplitVertical(Frame As String)

RÜCKGABE:
Object

PARAMETER:
Frame: Zeichenfolge, die die Nummer oder den Namen des Frames angibt.

BEISPIEL:

```
sub ExampleWindowFrames
REM Dieses Makro ermittelt alle in einem Fenster geladenen
Frames inklusive aller
REM Unterframes und gibt eine Liste mit Namen und Links aus
dim stList as new strings
Call FramesExample() REM Hier wird erstmal ein wenig
gebrowst...
Call FrameBrowse( ActiveWindow.TopWindow , stList )
MsgBoxList(stList , "URL-Liste aller Frames")
end sub
```

```
sub FramesExample
Const sFrameRight="RechterFrame"
Const sFrameLeft ="LinkerFrame"
Documents.Add("FrameSet")
ActiveWindow.Maximized=True
ActiveWindow.SplitVertical("1")
ActiveWindow.SetFrameContent("1","http://www.mopo.de/")
ActiveWindow.SetFrameName("1", sFrameLeft)
ActiveWindow.SetFrameContent("2","http://www.adobe.com/")
ActiveWindow.SetFrameName("2", sFrameRight )
ActiveWindow.FrameSpacing=5
dim oFrame as Object
dim sName as String
ActiveWindow.SplitParentHorizontal( sFrameRight ).FrameName=
"MyBestFrame"
ActiveWindow.SetFrameContent("MyBestFrame", "http://stardivi-
sion.de" )
oFrame = ActiveWindow.SplitParentVertical( sFrameRight )
oFrame.FrameName ="NewFrame"
sName = oFrame.FrameName
ActiveWindow.SetFrameName( sName , "NewFrameName" )
oFrame.FrameContent="file://c|/" REM auch directories kann man
in Frames laden
ActiveWindow.DeleteFrame( sFrameLeft )
end sub
sub FrameBrowse(oWin as Object, stList as Strings ) REM Star-
tObject , Ausgabe-Liste
REM Es wird eine Liste mit Objecten abgearbeitet, währendessen
werden
REM die in jeder Ebene ermittelten Informationen wieder an
diese Liste angehängt,
Dim oNewWin(100) as Object REM Speicher für alle Fenster-
Objekte
Dim iFirst as integer, iNext as integer, iTemp as integer
iFirst = 0
oNewWin(0) = oWin REM Startobject speichern
iNext = 0
REM Informationen an Liste anhängen
stList.Add("Name=" & oNewWin( iNext ).Name )
iNext = iNext+1
do
   iTemp = oNewWin(iFirst).FrameCount
   for iCount = 1 to iTemp
      oNewWin( iNext ) = oNewWin( iFirst ).GetFrameWindow(
iCount )
      REM Informationen an Liste anhängen
      stList.Add("Name=" & oNewWin( iNext ).Name &"  -->
Link="& oNewWin( iNext ).FrameContent )
      iNext = iNext+1
   next
   iFirst = iFirst + 1
loop until iFirst >= iNext
end sub
REM Diese Procedure gibt eine Liste in einer MessageBox aus
```

```
Sub MsgBoxList( stList as Strings , sHead As String)
Dim sText as String
Dim iCount as Integer
sText = ""
for iCount = 1 to stList.Count
sText = sText + stList.Item(iCount) + chr(13)
next
msgBox sText,0,sHead
end sub
```

StarCalc-Eigenschaft

Über diese Eigenschaft ist das StarCalc-Modul zugänglich.

KLASSE:
Application

SYNTAX:
StarCalc

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleStarCalc
Application.StarCalc.ScEditOptions
End Sub
```

StarDraw-Eigenschaft

Über diese Eigenschaft ist das StarDraw-Modul zugänglich.

KLASSE:
Application

SYNTAX:
StarDraw

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleStarDraw
Application.StarDraw.SdEditOptions
End Sub
```

Start-Methode

Blendet in der Statuszeile den Status eines aktuell auf dem Rechner ablaufenden Vorgangs ein.

Um lediglich Text in der Statusleiste einzublenden, wird die StatusBar-Eigenschaft verwendet.

Siehe auch: Resume, Stop, Suspend, Value, StatusBar.

KLASSE:
Progress

SYNTAX:
Start(Text As String, MaxValue As Long, WaitPointer As Boolean, LockAll As Boolean, Frame As Object)

PARAMETER:
[Text]: Zeichenfolge, die den Text angibt, der während der Rechnervorgangs in der Statuszeile eingeblendet werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird ein Leerstring übergeben.

[MaxValue]: Gibt den maximalen Wert der Value-Eigenschaft an. Dieser Parameter ist optional, wird er nicht angegeben, wird standardmäßig der Wert 100 übergeben.

[WaitPointer]: Optionaler Parameter, der angibt, ob der Warte-Cursor angezeigt werden soll (True) oder nicht (False). Wird der Parameter nicht übergeben, wird er standardmäßig auf False gesetzt.

[LockAll]: Optionaler Parameter, der angibt, ob während des in der Statuszeile eingeblendeten Rechnervorgangs Aktionen auf dem Rechner möglich sind (False) oder nicht (True).

[Frame]: Dieser Parameter ist bedeutungslos und kann ignoriert werden.

BEISPIEL:

```
sub ExampleProgress
Const sMessage="Hier wird gearbeitet"
Dim aProgress As New Progress
aProgress.Start(sMessage , 10000)
For aProgress.Value = 0 To 10000
If aProgress.Value = 5000 Then
aProgress.Suspend REM ... dringende unterbrechung
Wait 3000
aProgress.Resume
End If
```

```
Next aProgress.Value
aProgress.Stop
End Sub
```

StarWriter-Eigenschaft

Über diese Eigenschaft ist das StarWriter-Modul zugänglich.

KLASSE:
Application

SYNTAX:
StarWriter

RÜCKGABE:
Object

BEISPIEL:

```
ExampleStarWriter
Application.StarWriter.SwEditOptions
End Sub
```

StatusBar-Eigenschaft

Bestimmt oder ermittelt den Text in der Statusleiste.

KLASSE:
Application

SYNTAX:
StatusBar

RÜCKGABE:
String

PARAMETER:
[= Text]: Beliebiger Zeichenfolgenausdruck, der in der Statuszeile des Applikationsfensters eingeblendet wird.

BEISPIEL:

```
Sub ExampleStatusBar
Application.StatusBar = "Das kann man in der Statusleiste
lesen."
End Sub
```

Stop-Methode

Blendet in der Statuszeile die Fortschrittsanzeige für einen auf dem Rechner ablaufenden Vorgang aus. Damit wird der mittels der Start-Methode aktivierte Zugriff auf die Statusleiste beendet.

Siehe auch: Start, Resume, Suspend, Value.

KLASSE:
Progress

SYNTAX:
Stop()

BEISPIEL:

```
sub ExampleProgress
Const sMessage="Hier wird gearbeitet"
Dim aProgress As New Progress
aProgress.Start(sMessage , 10000)
For aProgress.Value = 0 To 10000
If aProgress.Value = 5000 Then
aProgress.Suspend REM ... dringende unterbrechung
Wait 3000
aProgress.Resume
End If
Next aProgress.Value
aProgress.Stop
End Sub
```

Street-Eigenschaft

Bestimmt oder ermittelt die Straße aus der Adresse, die bei den Benutzerdaten aufgeführt ist. Die Adresse wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Street

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die die Straße aus den Benutzerdaten angibt.

Beispiel:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Style-Eigenschaft

Bestimmt oder ermittelt den Linienstil einer Umrandungslinie oder das Füllmuster eines Hintergrunds.

Siehe auch: Color

KLASSE:
BorderLine, Interior

SYNTAX:
Style

RÜCKGABE:
Integer

PARAMETER:
Für Objekte der Klasse BorderLine:

[= Value]: Long-Integer-Ausdruck, der den Linienstil angibt. Folgende Werte sind möglich:

-1 - Keine (nur Lesen möglich)

0 - 0,05 pt einfach

1 - 1,00 pt einfach

2 - 2,50 pt einfach

3 - 4,00 pt einfach

4 - 5,00 pt einfach

5 - 1,10 pt doppelt

6 - 3,00 pt doppelt

7 - 7,50 pt doppelt

8 - 6,00 pt außen, doppelt

9 - 4,50 pt innen, doppelt

10 - 9,00 pt außen, doppelt

11 - 9,00 pt innen, doppelt

Für Objekte der Klasse Interior:

[= Pattern]: Long-Integer-Ausdruck, der angibt, welches Hintergrund- bzw. Füllmuster erscheinen soll. Pattern können die folgenden Muster zugewiesen werden:

Wert	Hintergrund / Schatten
0	Der Hintergrund oder der Schatten wird transparent dargestellt.
1	Ein Hintergrund wird weiß, ein Schatten schwarz dargestellt.
2	
3	
4	
5	
6	
7	
8	
9	
10	

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0)  REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
```

```
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

StyleCatalog-Methode

Ruft den Vorlagenkatalog auf.

KLASSE:
View

SYNTAX:
StyleCatalog()

BEISPIEL:

```
Sub ExampleStyleCatalog
Documents.Add("sWriter4")
ActiveWindow.StyleCatalog
End Sub
```

SubIni-Eigenschaft

Liefert oder setzt den Pfad für untergeordnete INI-Dateien. Dieser wird unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt.

KLASSE:
PathSettings

SYNTAX:
SubIni

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
```

```
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörtbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Subject-Eigenschaft

Bestimmt oder ermittelt das Thema des Dokuments. Dabei wird der Inhalt des Feldes THEMA aus den Dokumenteigenschaften gesetzt oder gelesen.

KLASSE:
DocumentInfo

SYNTAX:
Subject

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die das Thema des Dokuments bestimmt.

Beispiel:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Subject-Eigenschaft

Bestimmt oder ermittelt das Thema des Dokuments. Dabei wird der Inhalt des Feldes THEMA aus den Dokumenteigenschaften gesetzt oder gelesen.

KLASSE:
DocumentInfo

SYNTAX:
Subject

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die das Thema des Dokuments bestimmt.

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
```

```
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Suspend-Methode

Hält eine laufende Fortschrittsanzeige in der Statuszeile für einen auf dem Rechner ablaufenden Vorgang an. Dabei werden Statusanzeige und Wartecursor entfernt. Benutzereingaben sind wieder möglich, auch wenn die Fortschrittsanzeige mit LockAll=True gestartet wurde.

Siehe auch: Start, Stop, Resume, Value.

KLASSE:
Progress

SYNTAX:
Suspend()

BEISPIEL:

```
sub ExampleProgress
Const sMessage="Hier wird gearbeitet"
Dim aProgress As New Progress
aProgress.Start(sMessage , 10000)
For aProgress.Value = 0 To 10000
If aProgress.Value = 5000 Then
aProgress.Suspend REM ... dringende unterbrechung
Wait 3000
aProgress.Resume
End If
Next aProgress.Value
aProgress.Stop
End Sub
```

SwEditOptions-Methode

Ruft den Dialog OPTIONEN TEXTDOKUMENT auf.

KLASSE:
Desktop

SYNTAX:
SwEditOptions()

BEISPIEL:

```
Sub ExampleSwEditOptions
Documents.Add( "sWriter4" )
Application.SwEditOptions
End Sub
```

SyncExplorer-Methode

Positioniert im Explorer auf den Eintrag, der im Beamer angezeigt wird.

KLASSE:
Application

SYNTAX:
SyncExplorer()

BEISPIEL:

```
Sub ExampleSyncExplorer
Application.SyncExplorer
End Sub
```

Eigenschaften und Methoden mit "T"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "T" beginnen.

TaskBarVisible-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption der Statusleiste.

KLASSE:
Application

SYNTAX:
TaskBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Statusleiste ein- (True) oder ausgeblendet (False) wird.

BEISPIEL:

```
ExampleTaskBarVisible
Application.TaskBarVisible = FALSE
Wait 2000
Application.TaskBarVisible = TRUE
End Sub
```

Template-Eigenschaft

Bestimmt oder ermittelt Ort oder Typ der Dokumentvorlage.

Für ein PathSettings-Objekt bestimmt oder ermittelt diese Eigenschaft den Pfad, der unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE für Vorlagen eingestellt wird.

Für ein DocumentInfo-Objekt liefert diese Eigenschaft den Namen der Vorlage, der als Eintrag unter VORLAGE in den Dokumenteigenschaften aufgeführt ist.

KLASSE:
DocumentInfo, PathSettings

SYNTAX:
Template

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
```

```
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

Tile-Methode

Ordnet die nicht als Symbol dargestellten Fenster so an, daß alle Fenster sichtbar sind und den kompletten Arbeitsbereich ausfüllen.

KLASSE:
Windows

SYNTAX:
Tile()

BEISPIEL:

```
Sub ExampleWindows
Const sNewDoc="UNBENANNT"  REM Textsegment anhand dessen neue
Dokumente identifiziert werden können
Dim iCount As Integer
For iCount=1 to 8
```

```
   Documents.Add "sWriter4"
   Wait 100
   ActiveWindow.Selection.Insert("Text" & iCount)
Next iCount
Windows.ArrangeVertical
Windows.ArrangeHorizontal
Windows.Tile
Windows.Cascade
Windows.Minimize  REM Alle Fenster minimieren
Windows.ArrangeIcons
REM alle neuen Dokument schliessen
iCount=1 REM erster Index der WindowsCollection ist 0
Do While iCount <= Windows.Count
     if Instr( 1, UCASE( Windows( iCount ).Name ) ,
UCase(sNewDoc) )>0 then
        Windows.Remove( iCount )          REM entfernen ohne
Nachfrage !
     else
        iCount = iCount + 1
     end if
   loop
Windows.Close REM Alles Schliessen mit Nachfrage
End Sub
```

Title-Eigenschaft

Bestimmt oder ermittelt den Titel aus den Benutzerdaten, also zum Beispiel Prof. Dr. Dieser wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
Title

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Titel aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
```

```
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Title-Eigenschaft

Bestimmt oder ermittelt den Titel aus den Benutzerdaten, also zum Beispiel Prof. Dr. Dieser wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

Syntax:
Title

Rückgabe:
String

Parameter:
[= String]: Zeichenfolge, die den Titel aus den Benutzerdaten angibt.

Beispiel:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
```

```
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Title-Eigenschaft

Liefert den Titel aus den Dokumenteigenschaften. Dieser ist im allgemeinen nicht mit dem Titel des Dokuments identisch, wie er in der Titelleiste des Fensters angezeigt wird.

KLASSE:
DocumentInfo

SYNTAX:
Title

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleDocumentInfo
Dim oDocInfo as new Documentinfo
Dim sDocPath as String
Dim vInfo as Variant, iState as integer
Const sTitle = "Das ist der Dokumenttitel"
Const sDescription = "Diese Informationen können bearbeitet
werden ohne das Dokument komplett laden zu müssen."
REM gültigen Dokumentnamen ermitteln
sDocPath = Application.Pathsettings.Work & "\Test.sdw"
Documents.Add( "sWriter4" )
Wait 100
ActiveWindow.Selection.Insert( sDescription )
ActiveWindow.Selection.InsertPara
Wait 500
With ActiveDocument.DocumentInfo
   .Description = sDescription
   vInfo = .Author
   vInfo = .CreationDate
   vInfo = .CreationTime
   vInfo = .Description
   vInfo = .EditTime
   vInfo = .Field1
   vInfo = .Field1Title
```

```
   vInfo = .Field2
   vInfo = .Field2Title
   vInfo = .Field3
   vInfo = .Field3Title
   vInfo = .Field4
   vInfo = .Field4Title
   vInfo = .FileName
   vInfo = .KeyWords
   vInfo = .ModificationAuthor
   vInfo = .ModificationDate
   vInfo = .ModificationTime
   vInfo = .Revision
   vInfo = .Subject
   vInfo = .Template
   .Title = sTitle
End With
ActiveDocument.Close True, sDocPath       REM Schliessen mit
Speichern
Wait 100
REM jetzt die Dokument Information kontrollieren
iState = oDocInfo.Load(sDocPath)
MsgBox oDocInfo.Description ,64,"Dokument Informationen" REM
Ausgabe der Beschreibung
Erase oDocInfo
End Sub
```

ToolBarVisible-Eigenschaft

Bestimmt oder ermittelt, ob die Ansicht der Werkzeugleiste eingeschaltet ist.

KLASSE:
Application

SYNTAX:
ToolBarVisible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Werkzeugleiste ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleToolBarVisible
Application.ToolBarVisible = FALSE
Wait 2000
Application.ToolBarVisible = TRUE
End Sub
```

Top-Eigenschaft

Bestimmt oder ermittelt die vertikale Position des Fensters, das durch das entsprechende Objekt festgelegt wird.

Siehe auch: Left, Height, Width.

KLASSE:
TaskWindow, Application

SYNTAX:
Top

RÜCKGABE:
Long

PARAMETER:
[= Value]: Beliebiger Long-Integer-Ausdruck, der die neue vertikale Position des Fensters in Pixel festlegt.

BEISPIEL:

```
Sub ExampleDimension
with Application
   .Height = 300
   .Width = 400
   .Left = 200
   .Top = 100
   wait 5000
   .Maximized=TRUE
end with
```

TopLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die obere Umrandungslinie beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

Siehe auch: BottomLine, LeftLine, RightLine.

KLASSE:
Border

SYNTAX:
TopLine

RÜCKGABE:
Object

Beispiel:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

TopWindow-Eigenschaft

Über diese Eigenschaft ist das zugehörige Top-Window zugänglich.

Klasse:
Window

Syntax:
TopWindow

Rückgabe:
Object

Beispiel:

```
Sub ExampleTopWindow
Documents.Open("http://www.stardivision.de")
ActiveWindow.ActiveFrameWindow.Activate REM Den fokussierten
Frame aktivieren.
MsgBox Iif(ActiveWindow.IsTopWindow,"Ja","Nein"),0,"Ist das
aktive Fenster ein Hauptfenster?" REM Ein einzelner Frame ist
kein TopWindow.
```

```
MsgBox ActiveWindow.ActiveFrameWindow.Name,0,"Name des aktiven
Frame" REM Der Aktive Frame
MsgBox ActiveWindow.TopWindow.Name,0,"Name des Hauptfensters"
REM Das Hauptfenster
End Sub
```

Transparent-Eigenschaft

Bestimmt oder ermittelt die Einstellung für einen transparenten Hintergrund. Diese Eigenschaft entspricht der Option "Keine Füllung" im HINTERGRUND -Register des StarOffice.

KLASSE:
Interior

SYNTAX:
Transparent

RÜCKGABE:
Boolean

PARAMETER:
[= Bool]: Boolescher Ausdruck, der angibt, ob der Hintergrund transparent dargestellt werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

Eigenschaften und Methoden mit "U"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "U" beginnen.

Undo-Methode

Mit dieser Methode wird die letzte Benutzeraktion in dem aktuellen Window rückgängig gemacht, z.B. das Löschen eines Wortes. Ist es nicht möglich, die letzte Aktion rückgängig zu machen, dann kommt es bei Aufruf dieser Methode zu einem Laufzeitfehler.

Siehe auch: Repeat, Redo, UndoAction.

KLASSE:
Window

SYNTAX:
Undo()

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UndoAction-Eigenschaft

Ermittelt die im Menü BEARBEITEN unter dem Menüpunkt RÜCKGÄNGIG aufgeführte Beschreibung der zuletzt ausgeführten Aktion. Ist es nicht möglich, eine Aktion rückgängig zu machen, dann kommt es bei Aufruf dieser Eigenschaft zu einem Laufzeitfehler.

Siehe auch: Undo, RepeatAction, RedoAction.

KLASSE:
Window

SYNTAX:
UndoAction

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
StarWriter.UndoCount = 50
Exit Sub
```

```
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UndoCount-Eigenschaft

Bestimmt oder ermittelt die Zahl der Aktionen, die im Speicher erhalten bleiben sollen, um sie mit dem Befehl RÜCKGÄNGIG widerrufen zu können.

KLASSE:
Application

SYNTAX:
UndoCount

RÜCKGABE:
Long

PARAMETER:
[= Count]: Beliebiger Long-Integer-Ausdruck, der die maximale Anzahl der Schritte definiert, die rückgängig gemacht werden können.

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
```

```
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UnloadLibrary-Methode

Gibt eine per Referenz oder extern eingebundene Bibliothek, die zuvor mittels LoadLibrary geladen worden ist, wieder frei.

Siehe auch: LoadLibrary, AddLibrary, RemoveLibrary.

KLASSE:
OfficeDocument, Application

SYNTAX:
UnloadLibrary(LibraryName As String)

RÜCKGABE:
Boolean

PARAMETER:
LibraryName: Name der eingebundenen Bibliothek, die freigegeben werden soll.

BEISPIEL:

```
Sub ExampleLibrary
On Error goto LibraryNotFound
Dim iState As Integer
Documents.Add("sWriter4")
iState = Application.LoadLibrary("GIMMICKS")
Call Gimmicks.Userfields.StartChangesUserfields REM ruft
Routine aus Biliothek gimmicks auf
iState = Application.UnloadLibrary("GIMMICKS")
REM jetzt findet StarOffice die Bibliothek nicht mehr und
verursacht einen Fehler
Call Gimmicks.Userfields.StartChangesUserfields
Exit Sub
LibraryNotFound:
MsgBox("Bibliothek nicht gefunden",16,"Fehler absichtlich
verursacht")
End Sub
```

Eigenschaften und Methoden mit "V"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "V" beginnen.

Value-Eigenschaft

Bestimmt oder ermittelt den momentanen Zustand der Fortschrittsanzeige in der Statuszeile für einen auf dem Rechner ablaufenden Vorgang. Der mögliche Wertebereich hängt von einem festgelegten Intervall ab.

Siehe auch: Start, Stop, Resume, Suspend.

KLASSE:
Progress

SYNTAX:
Value

RÜCKGABE:
Long

BEISPIEL:

```
sub ExampleProgress
Const sMessage="Hier wird gearbeitet"
Dim aProgress As New Progress
aProgress.Start(sMessage , 10000)
For aProgress.Value = 0 To 10000
If aProgress.Value = 5000 Then
aProgress.Suspend REM ... dringende unterbrechung
Wait 3000
aProgress.Resume
End If
Next aProgress.Value
aProgress.Stop
End Sub
```

VerticalLine-Eigenschaft

Liefert ein Objekt der Klasse BorderLine zurück, welches die vertikale Linie in Tabellen beschreibt. Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color.

Siehe auch: HorizontalLine

KLASSE:
Border

SYNTAX:
VerticalLine

RÜCKGABE:
Object

Beispiel:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

View-Eigenschaft

Über diese Eigenschaft ist die View des Dokuments, welches in dem Window dargestellt wird, zugänglich. Die View beschreibt die Sicht auf den Inhalt eines Dokuments in einem Fenster. Von StarBasic aus können alle View-Methoden auch direkt erreicht werden, für OLE-Automation und C-Schnittstelle muß über diese Eigenschaft zugegriffen werden.

Klasse:
Window

Syntax:
View

Rückgabe:
Object

BEISPIEL:

```
Sub ExampleView
Dim oView as Object
oView = ActiveWindow.View
MsgBox "" & oView.Name ,64,"Der Name der aktuellen View"
End Sub
```

Visible-Eigenschaft

Setzt oder ermittelt die Sichtbarkeit der gesamten Applikation auf dem Bildschirm.

KLASSE:
Application

SYNTAX:
Visible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Applikation sichtbar ist (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleAppVisible
Application.Visible = FALSE
MsgBox "StarOffice ist verschwunden.",0,"StarOffice"
Application.Visible = TRUE
End Sub
```

Eigenschaften und Methoden mit "W"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "W" beginnen.

Width-Eigenschaft

Bestimmt oder ermittelt die Breite des Fensters, das durch das entsprechende Objekt festgelegt wird.

Siehe auch: Height, Left, Top.

KLASSE:
TaskWindow, Application

Syntax:
Width

Rückgabe:
Long

Parameter:
[= Value]: Beliebiger Long-Integer-Ausdruck, der die neue Breite des Fensters in Pixel festlegt.

Beispiel:

```
Sub ExampleDimension
with Application
   .Height = 300
   .Width = 400
   .Left = 200
   .Top = 100
   wait 5000
   .Maximized=TRUE
end with
```

Windows-Eigenschaft

Diese Eigenschaft enthält ein Collection-Objekt mit allen Windows.

Klasse:
Application

Syntax:
Windows

Rückgabe:
Object

Beispiel:

```
Sub ExampleAppWindows
Dim sText As String,iCount As Integer
For iCount =1 To Application.Windows.Count
sText = sText & Application.Windows.Item(iCount).Name & Chr(13)
Next iCount
MsgBox sText,0,"Fensterliste" REM Ausgabe der Namen aller
geöffneten Fenster
End Sub
```

WordOnly-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche nur nach ganzen Wörtern gesucht wird.

KLASSE:
SearchSettings

SYNTAX:
WordOnly

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Suchbegriff nur als ganzes Wort (True) oder auch als Bestandteil eines Wortes (False) gesucht werden soll.

BEISPIEL:

```
Sub ExampleBackward
Const sText="Text" REM Text der vorhanden ist.
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert sText
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GotoEndOfDoc TRUE
Wait 2000
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.InSelection = TRUE
ActiveWindow.SearchSettings.CaseSensitive = TRUE
ActiveWindow.SearchSettings.WordOnly = TRUE
ActiveWindow.Search sText
MsgBox Selection.Value,64,"Suchen und Finden"
End Sub
```

Work-Eigenschaft

Bestimmt oder ermittelt den Pfad für das Arbeitsverzeichnis. Dieser kann auch unter EXTRAS - OPTIONEN - ALLGEMEIN... - PFADE eingestellt werden.

KLASSE:
PathSettings

SYNTAX:
Work

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Pfad für das Arbeitsverzeichnis angibt.

BEISPIEL:

```
Sub ExamplePathSettings
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Das Backup-Verzeichnis: " &
Application.PathSettings.Backup )
.InsertPara
.Insert("Das Verzeichnis der Makrodateien: " &
Application.PathSettings.Basic )
.InsertPara
.Insert("Das Verzeichnis der Konfigurationsverzeichnisse: " &
Application.PathSettings.Config )
.InsertPara
.Insert("Das Datenbankverzeichnis: "  & Application.PathSet-
tings.Database)
.InsertPara
.Insert("Das Wörterbuchverzeichnis: " & Application.PathSet-
tings.Dictionary)
.InsertPara
.Insert("Das Verzeichnis der Inhaltsverzeichnisse: " & Applica-
tion.PathSettings.Glossary )
.InsertPara
.Insert("Das Verzeichnis der Grafiken: " & Application.PathSet-
tings.Graphics )
.InsertPara
.Insert("Das Verzeichnis der StarOfficemodule: " & Applica-
tion.PathSettings.Modules )
.InsertPara
.Insert("Das Palettenverzeichnis: " & Application.PathSet-
tings.Palette )
.InsertPara
.Insert("Das Verzeichnis der SubIni-Datei: " &
Application.PathSettings.SubIni )
.InsertPara
.Insert("Das Vorlagenverzeichnis: " & Application.PathSet-
tings.Template )
.InsertPara
.Insert("Das Arbeitsverzeichnis: " &
Application.PathSettings.Work )
.InsertPara
End With
End Sub
```

Eigenschaften und Methoden mit "Z"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Z" beginnen.

ZIPCode-Eigenschaft

Bestimmt oder ermittelt die Postleitzahl aus der Adresse, die bei den Benutzerdaten aufgeführt ist. Die Adresse wird entweder bei der Programminstallation angegeben oder unter EXTRAS - OPTIONEN - ALLGEMEIN... - BENUTZERDATEN eingetragen.

KLASSE:
Address

SYNTAX:
ZIPCode

RÜCKGABE:
String

PARAMETER:
[= Code]: Zeichenfolge, die die Postleitzahl aus den Benutzerdaten angibt.

BEISPIEL:

```
Sub ExampleAddress
Const sHeader = "In Ihrem Adreßbuch ist folgendes eingetragen:
"
Const sMessage = "Eintrag "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage & "City : "
Selection.Insert Application.Address.City
Selection.InsertPara
Selection.Insert sMessage & "Company : "
Selection.Insert Application.Address.Company
Selection.InsertPara
Selection.Insert sMessage & "Country : "
Selection.Insert Application.Address.Country
Selection.InsertPara
Selection.Insert sMessage & "EMail : "
Selection.Insert Application.Address.EMail
Selection.InsertPara
Selection.Insert sMessage & "Fax : "
Selection.Insert Application.Address.Fax
Selection.InsertPara
Selection.Insert sMessage & "FirstName : "
Selection.Insert Application.Address.FirstName
Selection.InsertPara
Selection.Insert sMessage & "ID : "
Selection.Insert Application.Address.ID
Selection.InsertPara
Selection.Insert sMessage & "LastName : "
Selection.Insert Application.Address.LastName
Selection.InsertPara
```

```
Selection.Insert sMessage & "Phone : "
Selection.Insert Application.Address.Phone
Selection.InsertPara
Selection.Insert sMessage & "PhonePrivate : "
Selection.Insert Application.Address.PhonePrivate
Selection.InsertPara
Selection.Insert sMessage & "Position : "
Selection.Insert Application.Address.Position
Selection.InsertPara
Selection.Insert sMessage & "Street : "
Selection.Insert Application.Address.Street
Selection.InsertPara
Selection.Insert sMessage & "Title : "
Selection.Insert Application.Address.Title
Selection.InsertPara
Selection.Insert sMessage & "ZIPCode : "
Selection.Insert Application.Address.ZIPCode
Selection.InsertPara
End Sub
```

Referenzhilfe für Textdokumente

Hier erhalten Sie in alphabetischer Ordnung Informationen über sämtliche Eigenschaften und Methoden, die Ihnen für Textdokumente zur Verfügung stehen.

Eigenschaften und Methoden mit "A"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "A" beginnen.

ActionEnd-Methode

Schaltet die sichtbare Formatierung des Dokumentes wieder ein, wenn sie zuvor mit der ActionStart-Methode ausgeschaltet wurde. Da beide Methoden verschachtelt werden können, ist ein paarweises Setzen notwendig. Um die Formatierung des Dokuments sichtbar zu machen, muß ActionEnd genau so-oft aufgerufen werden wie zuvor ActionStart.

Siehe auch: ActionStart

KLASSE:
BaseTextSelection

SYNTAX:
ActionEnd()

BEISPIEL:

```
Sub ExampleActions
Const Message = "Gleich kommt noch mehr Text"
Const Message_1 = "Dieser Text erscheint erst nach einer Verzö-
gerung"
Dim iAction As Integer
Documents.Add( "sWriter4" )
Selection.Insert Message
Selection.InsertPara
Selection.ActionStart
Selection.ActionStart
Selection.ActionStart
Selection.Insert Message_1
Selection.InsertPara
Wait 100
Selection.ActionEnd
Selection.ActionEnd
Selection.ActionEnd
End Sub
```

ActionStart-Methode

Schaltet die sichtbare Formatierung des Dokuments aus. Nach einem Aufruf dieser Methode werden alle nachfolgenden Formatierungen nicht mehr im Dokument angezeigt. Das Ergebnis dieser Formatierungen wird erst dann wieder sichtbar, wenn die ActionEnd-Methode aufgerufen wird. Zusammen mit der ActionEnd-Methode ist eine Verschachtelung möglich, so daß ein paarweises Setzen von ActionStart und ActionEnd notwendig ist.

Makros mit sehr zeitintensiven Formatierungen sollten diese Methoden zur Beschleunigung des Programmablaufs verwenden, da eine Formatierung des Dokuments nur im Arbeitsspeicher sehr viel schneller als mit gleichzeitiger Ansicht auf dem Bildschirm durchgeführt wird.

Siehe auch: ActionEnd

KLASSE:
BaseTextSelection

SYNTAX:
ActionStart()

BEISPIEL:

```
Sub ExampleActions
Const Message = "Gleich kommt noch mehr Text"
Const Message_1 = "Dieser Text erscheint erst nach einer Verzö-
gerung"
Dim iAction As Integer
Documents.Add( "sWriter4" )
Selection.Insert Message
Selection.InsertPara
Selection.ActionStart
Selection.ActionStart
Selection.ActionStart
Selection.Insert Message_1
Selection.InsertPara
Wait 100
Selection.ActionEnd
Selection.ActionEnd
Selection.ActionEnd
End Sub
```

Activate-Methode

Aktiviert ein selektiertes OLE-Objekt.

Der Zugriff erfolgt über die Selection oder über die entsprechende Collection "OLEObjects".

KLASSE:
BaseTextOLEObject

SYNTAX:
Activate()

BEISPIEL:

```
Sub ExampleOLEObjects
Const sMessage = "Der Name des Charts lautet : "
Dim oActwin As Object
Documents.Add( "sWriter4" )
Set oActwin = ActiveWindow REM Merkt sich die View für Deacti-
vate
If ActiveWindow.OLEObjects.Count = 0 Then
ActiveWindow.InsertObjectStarChart()
Else
ActiveWindow.OLEObjects(1).Activate
End If
oActwin.DeActivate REM Hier muß die View des Dokumentes
verwendet werden
Print sMessage & ActiveWindow.OLEObjects(1).OLEObject.Name
End Sub
```

AddAllUnknownWords-Methode

Startet die Rechtschreibprüfung und nimmt automatisch alle unbekannten Wörter in das angegebene Wörterbuch auf.

Der Name des Wörterbuches muß mit der Dateiendung angegeben werden, andernfalls wird ein neues Wörterbuch ohne Dateiendung angelegt.

KLASSE:
BaseTextEditView

SYNTAX:
AddAllUnknownWords(Dictionary As String)

PARAMETER:
Dictionary: Name des Wörterbuches, in das die Wörter aufgenommen werden sollen.

BEISPIEL:

```
Sub ExampleAddAllUnknownWords
Documents.Add("sWriter4")
Selection.Insert "Dies ist ein Testsatz"
Selection.JumpToStartOfDoc
ActiveWindow.AddAllUnknownWords("Standard.dic")
End Sub
```

AlignBottom-Methode

Ordnet das selektierte Objekt am unteren Rand an. Die Ausrichtung ist abhängig von der Art der Verankerung. Ist das Objekt an der Seite verankert, wird es am unteren Seitenrand angeordnet, bei Verankerung am Absatz wird seine untere Kante am Absatzende ausgerichtet.

Siehe auch: AlignCharBottom, AlignTop

KLASSE:
BaseTextFrame

SYNTAX:
AlignBottom()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
```

```
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

AlignCharBottom-Methode

Ordnet ein selektiertes, am Zeichen verankertes Objekt am Zeichen unten an. Dabei wird die obere Kante des Objekts in Zeichenhöhe gesetzt.

Siehe auch: AlignBottom, AlignCharTop.

KLASSE:
BaseTextFrame

SYNTAX:
AlignCharBottom()

BEISPIEL:

```
Sub ExampleFrameAnchorCharacter
Const sText1="Das ist ein zeichengebundener Rahmen."
Const sText2="Am letzten Zeichen dieses Textes befindet sich
ein Textrahmen. "
Dim sFirstFrame As String
Dim iCount As Integer
```

```
Documents.Add("sWriter4")
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
Selection.Insert("Z")
Selection.InsertFrame(1, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
Selection.GotoAnchor
For iCount = Len( sText2) To 1 Step -1
   ActiveWindow.JumpToStartOfDoc
   Selection.Insert( Mid(sText2,iCount,1) )
Next iCount
With ActiveWindow.Frames( sFirstFrame )
.AlignCharBottom
   Wait 500
.AlignCharTop
   Wait 500
.AlignVerticalCharCenter
   Wait 500
End With
Selection.GotoAnchor
End Sub
```

AlignCharTop-Methode

Ordnet ein selektiertes, am Zeichen verankertes Objekt am Zeichen oben an. Dabei wird die untere Kante des Objekts in Zeichenhöhe gesetzt.

Siehe auch: AlignTop, AlignCharBottom.

KLASSE:
BaseTextFrame

SYNTAX:
AlignCharTop()

BEISPIEL:

```
Sub ExampleFrameAnchorCharacter
Const sText1="Das ist ein zeichengebundener Rahmen."
Const sText2="Am letzten Zeichen dieses Textes befindet sich
ein Textrahmen. "
Dim sFirstFrame As String
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
```

```
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
Selection.Insert("Z")
Selection.InsertFrame(1, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
Selection.GotoAnchor
For iCount = Len( sText2) To 1 Step -1
   ActiveWindow.JumpToStartOfDoc
   Selection.Insert( Mid(sText2,iCount,1) )
Next iCount
With ActiveWindow.Frames( sFirstFrame )
.AlignCharBottom
   Wait 500
.AlignCharTop
   Wait 500
.AlignVerticalCharCenter
   Wait 500
End With
Selection.GotoAnchor
End Sub
```

AlignHorizontalCenter-Methode

Ordnet das selektierte Objekt horizontal zentriert an. Je nach Art der Verankerung wird es entweder horizontal mittig auf der Seite angeordnet (Verankerung an der Seite) oder horizontal mittig bezogen auf die Textbegrenzung (Verankerung am Absatz).

Ist das Objekt am Zeichen verankert, steht diese Methode nicht zur Verfügung.

Siehe auch: AlignVerticalCenter

KLASSE:
BaseTextFrame

SYNTAX:
AlignHorizontalCenter()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
```

```
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
```

```
wait 500
.GotoAnchor
End With
End Sub
```

AlignLeft-Methode

Ordnet das selektierte Objekt am linken Rand an. Je nach Art der Verankerung wird es entweder am linken Seitenrand angeordnet (Verankerung an der Seite) oder an der linken Textbegrenzung (Verankerung am Absatz).

Ist das Objekt am Zeichen verankert, steht diese Methode nicht zur Verfügung.

Siehe auch: AlignRight

KLASSE:
BaseTextFrame

SYNTAX:
AlignLeft()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
```

```
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

Alignment-Eigenschaft

Bestimmt oder ermittelt die Ausrichtung des selektierten Absatzes. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetAlignment gesetzte Ausrichtung abgefragt.

Eine Abfrage der Alignment-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsAlignment bestimmt wurde, daß nach einer bestimmten Ausrichtungsoption gesucht werden soll, d. h. IsAlignment = 2.

Siehe auch: SetAlignment, IsAlignment.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
Alignment

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur Lesen)

Bei Absätzen:

[= AlignType]: Integer-Ausdruck, der die Art der Ausrichtung des selektierten Absatzes angibt. AlignType kann folgende Werte annehmen:

0 - Linksbündig

1 - Rechtsbündig

2 - Blocksatz

3 - Zentriert

BEISPIEL:

```
Sub ExampleJustifyAll
REM Formatiert das gesamte Dokument auf Blocksatz
Const sMotherTemp="Standard" REM Die Mutter aller Absatzvor-
lagen
Const sBlindText = "BT" REM den Blindtext verwenden
Documents.Add("sWriter4")
With ActiveWindow.Selection
   .ExpandGlossary( sBlindText )
   sStyle = .ParaStyle      REM rette aktuelle Vorlage
   .Paragraph.ParaStyleApply(sMotherTemp)
   .Paragraph.Alignment = 2
   .StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
   Wait 1000
   .Paragraph.Alignment = 0     REM aus Standardwert zurück-
setzen
   .StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
   .Paragraph.ParaStyleApply( sStyle )   REM wieder auf
ursprüngliche Vorlage setzen
end with
End Sub
```

AlignRight-Methode

Ordnet das selektierte Objekt am rechten Rand an. Je nach Art der Verankerung wird es entweder am rechten Seitenrand angeordnet (Verankerung an der Seite) oder an der rechten Textbegrenzung (Verankerung am Absatz).

Ist das Objekt am Zeichen verankert, steht diese Methode nicht zur Verfügung.

Siehe auch: AlignLeft

KLASSE:
BaseTextFrame

SYNTAX:
AlignRight()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
```

```
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

AlignRowBottom-Methode

Ordnet ein Objekt an der Zeile unten an.

KLASSE:
BaseTextFrame

SYNTAX:
AlignRowBottom()

BEISPIEL:

```
Selection.AlignRowBottom
```

AlignRowTop-Methode

Ordnet ein Objekt an der Zeile oben an.

KLASSE:
BaseTextFrame

SYNTAX:
AlignRowTop()

BEISPIEL:

```
Selection.AlignRowTop
```

AlignTop-Methode

Ordnet das selektierte Objekt am oberen Rand an. Die Ausrichtung ist abhängig von der Art der Verankerung: Ist das Objekt an der Seite verankert, wird es am oberen Seitenrand angeordnet, bei Verankerung am Absatz wird seine obere Kante am Absatzanfang ausgerichtet.

Siehe auch: AlignBottom, AlignCharTop.

KLASSE:
BaseTextFrame

SYNTAX:
AlignTop()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
```

```
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

AlignVerticalCenter-Methode

Ordnet das selektierte Objekt vertikal zentriert an. Die Ausrichtung ist abhängig von der Art der Verankerung: Ist das Objekt an der Seite verankert, wird es vertikal mittig auf der Seite angeordnet, bei Verankerung am Absatz wird es vertikal mittig bezogen auf den Absatz ausgerichtet.

Siehe auch: AlignHorizontalCenter, AlignVerticalCharCenter

KLASSE:
BaseTextFrame

SYNTAX:
AlignVerticalCenter()

BEISPIEL:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
```

```
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

AlignVerticalCharCenter-Methode

Ordnet ein selektiertes, am Zeichen verankertes Objekt relativ zum Zeichen vertikal zentriert an.

Siehe auch: AlignVerticalCenter

KLASSE:
BaseTextFrame

SYNTAX:
AlignVerticalCharCenter()

BEISPIEL:

```
Sub ExampleFrameAnchorCharacter
Const sText1="Das ist ein zeichengebundener Rahmen."
Const sText2="Am letzten Zeichen dieses Textes befindet sich
ein Textrahmen. "
Dim sFirstFrame As String
Dim iCount As Integer
```

```
Documents.Add("sWriter4")
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
Selection.Insert("Z")
Selection.InsertFrame(1, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
Selection.GotoAnchor
For iCount = Len( sText2) To 1 Step -1
   ActiveWindow.JumpToStartOfDoc
   Selection.Insert( Mid(sText2,iCount,1) )
Next iCount
With ActiveWindow.Frames( sFirstFrame )
.AlignCharBottom
   Wait 500
.AlignCharTop
   Wait 500
.AlignVerticalCharCenter
   Wait 500
End With
Selection.GotoAnchor
End Sub
```

AlignVerticalRowCenter-Methode

Ordnet ein Objekt an der Zeile vertikal zentriert an.

KLASSE:
BaseTextFrame

SYNTAX:
AlignVerticalRowCenter()

BEISPIEL:

```
Selection.AlignVerticalRowCenter
```

AnchorOnly-Eigenschaft

Schaltet die Option NUR ERSTER ABSATZ beim Umlauf von Textrahmen ein.

KLASSE:
BaseTextFrame

SYNTAX:
AnchorOnly

Rückgabe:
Boolean

Parameter:
True: Option nur erster Absatz aktivieren

False: Option nur erster Absatz ausschalten

Beispiel:

```
Sub ExampleTextContour
Const sText1 ="Das ist ein Text der um die Kontur der Grafik
fließt. "
Dim sFrmName as String
Documents.Add("sWriter4")
Selection.InsertFrame(0,,,"4cm","5cm")
sFrmName = Selection.FrameName
Selection.AlignHorizontalCenter
Selection.ContourWrap= TRUE
Selection.Wrap(2)
Selection.GotoAnchor
For iCount=1 To 10
   Selection.Insert( sText1)
Next iCount
Selection.InsertPara
For iCount=1 To 5
   Selection.Insert( sText1)
Next iCount
Wait 2000
ActiveWindow.Frames(sFrmName).AnchorOnly = TRUE
ActiveWindow.Frames(sFrmName).GotoAnchor
End Sub
```

AnchorType-Eigenschaft

Bestimmt die Verankerung des Objekts.

Klasse:
BaseTextFrame

Syntax:
AnchorType

Rückgabe:
Long

Parameter:
0 - absatzgebunden

1 - zeichengebunden

2 - seitengebunden

Beispiel:

```
Sub ExampleFrameAnchorType
Const sText1="Das ist ein Rahmen "
Const sText1a="am Absatz "
Const sText1b="am Zeichen "
Const sText1c="an der Seite "
Const sText2="Dieser Text füllt einen Absatz. "
Dim sFirstFrame As String
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
For iCount=1 to 5
   Selection.Insert( sText1 )
Next iCount
Selection.InsertFrame(1, "5cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
With ActiveWindow.Frames( sFirstFrame )
.Insert(sText1)
   Wait 500
.AnchorType = 0
.Insert(sText1 + sText1a)
   Wait 500
.AnchorType = 1
.Insert(sText1 + sText1b)
   Wait 500
.AnchorType = 2
.Insert(sText1 + sText1c)
   Wait 500
.GotoAnchor

End With
End Sub
```

Auto-Eigenschaft

Auto beschreibt zusammen mit Break das Attribut "Umbruch am Absatz".

Ist ein Umbruch eingeschaltet, entscheidet Auto, ob das Umbruch-Attribut nach Einfügen eines Absatzumbruchs in den folgenden Absatz übernommen wird oder nicht.

Klasse:
Paragraph

Syntax:
Auto

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Umbruch nur für den aktuellen Absatz gilt (True) oder ob das Attribut auf den neuen Absatz ausgeweitet wird (False).

Beispiel:

```
sub ExampleParagraphBreak
Const sM1 = "Vor diesem Absatz wird ein Seitenumbruch mit
Vorlage: "
Const sTemp="Rechte Seite"  REM Vorlage Linke Seite wird beim
Seitenumbruch verwendet.
Const sM2 = " eingefügt"
Const sTempWithBreak="Erster Absatz einer Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
With Activewindow.Selection
   .Insert("Erster Absatz hat Standardformat.")
   .InsertPara
   .Insert( sM1 & sTemp & sM2 )
   .Paragraph.Break = 4 REM Seitenumbruch vor dem Absatz
   .Paragraph.Auto = TRUE
   .Paragraph.PageStyle=sTemp REM ... mit dieser Seitenvorlage
   .StyleNewByExample( sTempWithBreak, 2 ) REM die Absatzattri-
bute in eine Vorlage räumen
   .InsertPara
   .Paragraph.PageStyle=""  REM Mit "" wird die gesamte
Umbruch-Attributierung hart gelöscht !
   .Insert("Der letzte Absatz in diesem Dokument hat wieder
Standardformat.")
   .InsertPara
End With
```

AutoFormat-Methode

Aktiviert die AutoFormat-Funktion. Bei einer selektierten Tabelle im Dokument ruft diese Methode den Dialog Autoformat zur Formatierung der aktuellen Tabelle auf, wenn kein Übergabeparameter angegeben wird. Wird ein Parameter übergeben, so wird die aktuelle Tabelle mit dem angegebenen Format formatiert.

Klasse:
BaseTextTable

SYNTAX:
AutoFormat(aFormatName As String)

PARAMETER:
[aFormatName]: Zeichenfolge, die die Bezeichnung des Formates angibt, das der aktuellen Tabelle zugewiesen werden soll. aFormatName kann die Bezeichnungen eines im Dialog AUTOFORMAT aufgeführten Formates aufnehmen; die Tabelle wird dann mit diesem Format formatiert. Wird aFormatName nicht übergeben, wird der Dialog AUTOFORMAT zur Formatierung der Tabelle aufgerufen.

BEISPIEL:

```
Sub ExampleAutoFormat
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
REM Tabelle mit dem Format REM StandardREM  formatieren
Selection.AutoFormat("Standard")
End Sub
```

AutoPilotAgenda-Methode

Ruft den AutoPiloten zur Erstellung einer neuen Agenda auf.

KLASSE:
StarWriter

SYNTAX:
AutoPilotAgenda()

BEISPIEL:

```
Sub ExampleAutopilotAgenda
Documents.Add "sWriter4"
StarWriter.AutopilotAgenda
End Sub
```

AutoPilotFax-Methode

Ruft den AutoPiloten zur Erstellung eines neuen Faxes auf.

KLASSE:
StarWriter

SYNTAX:
AutoPilotFax()

Beispiel:

```
Sub ExampleAutopilotfax
Documents.Add "sWriter4"
StarWriter.Autopilotfax
End Sub
```

AutoPilotLetter-Methode

Ruft den AutoPiloten zur Erstellung eines neuen Briefes auf.

Klasse:
StarWriter

Syntax:
AutoPilotLetter()

Beispiel:

```
Sub ExampleAutopilotLetter
Documents.Add "sWriter4"
StarWriter.AutopilotLetter
End Sub
```

AutoPilotMemo-Methode

Ruft den AutoPiloten zur Erstellung eines neuen Memos auf.

Klasse:
StarWriter

Syntax:
AutoPilotMemo()

Beispiel:

```
Sub ExampleAutopilotMemo
Documents.Add "sWriter4"
StarWriter.AutopilotMemo
End Sub
```

Eigenschaften und Methoden mit "B"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "B" beginnen.

BackColor-Eigenschaft

Bestimmt oder ermittelt die Farbe für das Hintergrundmuster des selektierten Bereichs oder für den Schatten, der dem selektierten Bereich zugewiesen ist.

Die BackColor-Eigenschaft hat für TextPage- und BaseTextFrame-Objekte nur dann eine Wirkung, wenn zuvor mit der Style-Eigenschaft ein Füllmuster gesetzt worden ist.

Siehe auch: Style

KLASSE:
TextPage, Shadow, TextFont, BaseTextDrawControl, BaseTextFrame, BaseTextEditView

SYNTAX:
BackColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe für das Hintergrundmuster oder den Schatten des selektierten Bereichs angibt.

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

BackgroundColor-Eigenschaft

Bestimmt oder ermittelt die Hintergrundfarbe des selektierten Objekts.

KLASSE:
BaseTextSelection

SYNTAX:
BackgroundColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe des selektierten Objekts angibt.

BEISPIEL:

```
Sub ExampleBackcolor
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.BackgroundColor = RGB( 0, 255, 0 )
End Sub
```

BackgroundDialog-Methode

Ruft das Register HINTERGRUND auf, in dem die Einstellungen für die Hintergrundfarbe, das Muster und die Musterfarbe vorgenommen werden können.

KLASSE:
BaseTextSelection

SYNTAX:
BackgroundDialog()

BEISPIEL:

```
Sub ExampleBackgroundDialog
Documents.Add "sWriter4"
Selection.Insert("Gleich kommt der Hintergrund-Dialog")
Selection.BackgroundDialog()
End Sub
```

Backspace-Methode

Löscht die aktuelle Selektion. Die Backspace-Methode entspricht einem Betätigen der Löschtaste: Ist ein Objekt oder ein Textbereich selektiert, dann wird das selektierte Objekt bzw. der markierte Text gelöscht. Befindet sich der Cursor in Text, wird das Zeichen gelöscht, das links neben der Schreibmarke steht.

KLASSE:
TextBezier, BaseTextSelection, BaseTextDrawBase

SYNTAX:
Backspace()

BEISPIEL:

```
Sub ExampleBackspace
Const sMessage = "Das erste Wort wird gleich gelöscht"
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
ActiveWindow.SelectionTextExt
Selection.Backspace
End Sub
```

Bold-Eigenschaft

Bestimmt oder ermittelt die Intensität der Fett-Darstellung des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetBold gesetzte Fontstärke abgefragt.

Eine Abfrage der Bold-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsBold bestimmt wurde, daß nach einer bestimmten Fontstärke gesucht werden soll, d. h. IsBold = 2.

Siehe auch: SetBold, IsBold.

KLASSE:
TextDrawText, ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Bold

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Text-Objekten:

[= Value]: Integer-Ausdruck, der die Intensität der Fett-Darstellung des Textes angibt. Folgende Werte sind möglich:

0: Der Text wird normal dargestellt.

1 - 10: Der Text wird fett dargestellt. Die Intensität der Fett-Darstellung erhöht sich mit zunehmendem Wert. Der Wert 8 gibt die Standardeinstellung des Attributs Fett wieder.

Nicht alle Fonts unterstützen alle Stufen des Bold-Attributs. Der Standardwert 8 ist jedoch fast immer verfügbar.

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
```

```
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
```

```
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

BookmarkCount-Eigenschaft

Ermittelt die Anzahl der Textmarken im gesamten Dokument.

KLASSE:
BaseTextSelection

SYNTAX:
BookmarkCount

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
```

```
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

BookmarkName-Methode

Die Methode erlaubt den indizierten Zugriff auf die Namen der im Dokument vorhandenen Textmarken.

Referenz: BookmarkCount, GotoBookmark, JumpToBookmark, DeleteBookmark, JumpToNextBookmark, JumpToPrevBookmark, InsertBookmark.

KLASSE:
BaseTextSelection

SYNTAX:
BookmarkName(Position As Long)

RÜCKGABE:
String

PARAMETER:
Der Wertebereich für die Position reicht von Null bis BookmarkCount - 1

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
```

```
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

Border-Eigenschaft

Das Border-Interface enthält die komplette Information für die Umrandungslinien. Die Linien sind ihrerseits Objekte, folgende Linien-Objekte gibt es:

TopLine

BottomLine

LeftLine

RightLine

HorizontalLine (nur in Tabellen)

VerticalLine (nur in Tabellen)

Die Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style

Color

Den Abstand der Umrandung regelt die Eigenschaft:

Distance

KLASSE:
BaseTextSelection

SYNTAX:
Border

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleBorder
Documents.Add( "sWriter4" )
REM Fügt einen seitengebunden Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.Border.TopLine.Style = 2
Selection.Border.TopLine.Color = RGB( 255, 0, 0 )
Selection.Border.BottomLine.Style = 3
Selection.Border.BottomLine.Color = RGB( 0, 255, 0 )
Selection.Border.LeftLine.Style = 4
Selection.Border.LeftLine.Color = RGB( 0, 0, 255 )
Selection.Border.RightLine.Style = 5
Selection.Border.RightLine.Color = RGB( 255, 255, 0 )
Selection.Border.Distance = "0.5cm"
End Sub
```

BorderDialog-Methode

Ruft das Register UMRANDUNG zum Umranden des selektierten Objekts auf. Dies kann z. B. ein Absatz, eine Tabelle oder ein Rahmen sein. Einstellungen in diesem Dialog wirken sich immer auf den gesamten Absatz aus.

KLASSE:
BaseTextSelection

SYNTAX:
BorderDialog()

Beispiel:

```
Sub ExampleBorderDialog
Documents.Add "sWriter4"
Selection.Insert("Gleich kommt der Umrandungs-Dialog")
Selection.BorderDialog()
End Sub
```

Bottom-Eigenschaft

Bestimmt oder ermittelt den Wert für das Skalieren der Ansicht der selektierten Grafik an der unteren Seite. Negative Werte beschneiden die Ansicht der Grafik, positive verbreitern den Rand zwischen Rahmen und Grafik und verkleinern die Grafik dabei.

Siehe auch: Left, Right und Top.

Klasse:
BaseTextGraphic

Syntax:
Bottom

Rückgabe:
Long

Parameter:
[= Cut]: Gibt an, um welchen Wert die Grafik am unteren Rand geändert werden soll. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleGraphicCrop
Dim sGrfPath as String
Dim iCount As Integer
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Wait 3000
With Selection  REM in zwei Stufen wird das Bild verkleinert
For iCount = 2 to -1 Step -1
   .Bottom = Trim(Str(iCount) & "cm")
   .Top= Trim(Str(iCount) & "cm")
   .Left= Trim(Str(iCount) & "cm")
   .Right= Trim(Str(iCount) & "cm")
   Wait 3000
Next iCount
End With
Selection.Escape
End Sub
```

BottomMargin-Eigenschaft

Bestimmt oder ermittelt den Abstand vom unteren Rand des selektierten oder aktiven Objektes zu seiner Umgebung. Dieser Abstand ist für verschiedene Objekte unterschiedlich definiert: Bei einem TextEditView-Objekt ist es der Abstand bis zur unteren Papierkante. Bei Objekten der Klassen TextFrame, TextTable, TextGraphic, TextOLEObject ist es der Abstand bis zur ersten, darunterliegenden Zeile des umlaufenden Textes oder des nachfolgenden Absatzes. Ist ein Absatz selektiert (Paragraph, TextInTable, ListInTable), so liefert BottomMargin den Abstand zwischen der letzten Zeile des selektierten und der ersten Textzeile des nachfolgenden Absatzes.

Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetTopBottomMargin gesetzte Abstand abgefragt.

Eine Abfrage der BottomMargin-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsTopBottomMargin bestimmt wurde, daß nach einem bestimmten Absatzabstand gesucht werden soll., d. h. IsTopBottomMargin = 2.

Siehe auch: LeftMargin, RightMargin, TopMargin.

Nur bei SearchAttributes, ReplaceAttributes:

SetTopBottomMargin, IsTopBottomMargin.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
BottomMargin

RÜCKGABE:
Long

PARAMETER:
[= Distance]: Gibt den jeweiligen Abstand an. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleTableMargin
Documents.Add("sWriter4")
Selection.InsertTable( 4 , 4 )
```

```
Selection.Select
REM Tabellenausrichtung auf REM ManuellREM  stellen
Selection.Alignment = 0
Selection.BottomMargin = "3cm"
Selection.TopMargin = "2cm"
Selection.RightMargin = "1cm"
Selection.LeftMargin = "1cm"
End Sub
```

Bounds-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für Textbegrenzungen. Es handelt sich hierbei um die Einstellungen, die auch unter EXTRAS - OPTIONEN - TEXTDOKUMENT - REGISTER "LAYOUT" vorgenommen werden können.

KLASSE:
ViewSettings

SYNTAX:
Bounds

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Textbegrenzungen ein- (True) oder ausgeblendet (False) werden sollen.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption  "
Const sMessage_2 = "  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Break-Eigenschaft

Führt einen Seiten- oder Spaltenumbruch vor oder hinter dem aktuellen Absatz aus, oder ermittelt, ob ein Seiten- oder Spaltenumbruch vor oder hinter dem aktuellen Absatz erfolgt.

KLASSE:
Paragraph

SYNTAX:
Break

RÜCKGABE:
Long

PARAMETER:
[= Setting]: Long-Integer-Ausdruck, der angibt, wie der Umbruch erfolgen soll:

0: Kein Umbruch (Voreinstellung).

1: Spaltenumbruch vor dem Absatz.

2: Spaltenumbruch nach dem Absatz.

3: Spaltenumbruch vor und nach dem Absatz.

4: Seitenumbruch vor dem Absatz.

5: Seitenumbruch nach dem Absatz.

6: Seitenumbruch vor und nach dem Absatz.

BEISPIEL:

```
sub ExampleParagraphBreak
Const sM1 = "Vor diesem Absatz wird ein Seitenumbruch mit
Vorlage: "
```

```
Const sTemp="Rechte Seite"  REM Vorlage Linke Seite wird beim
Seitenumbruch verwendet.
Const sM2 = " eingefügt"
Const sTempWithBreak="Erster Absatz einer Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
With Activewindow.Selection
   .Insert("Erster Absatz hat Standardformat.")
   .InsertPara
   .Insert( sM1 & sTemp & sM2 )
   .Paragraph.Break = 4 REM Seitenumbruch vor dem Absatz
   .Paragraph.Auto = TRUE
   .Paragraph.PageStyle=sTemp REM ... mit dieser Seitenvorlage
   .StyleNewByExample( sTempWithBreak, 2 ) REM die Absatzattri-
bute in eine Vorlage räumen
   .InsertPara
   .Paragraph.PageStyle=""  REM Mit "" wird die gesamte
Umbruch-Attributierung hart gelöscht !
   .Insert("Der letzte Absatz in diesem Dokument hat wieder
Standardformat.")
   .InsertPara
End With
```

BringToFront-Methode

Ordnet das selektierte Objekt vor allen anderen Objekten im Vordergrund an.

KLASSE:
BaseTextDrawBase, BaseTextFrame

SYNTAX:
BringToFront()

BEISPIEL:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
   Selection.Insert(sText)
   if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALS
Selection.Printable = TRUE
```

```
Selection.SendToBack
Selection.BringToFront
end sub
```

BrowseView-Eigenschaft

Schaltet die Ansicht ONLINE-LAYOUT ein. Diese Eigenschaft kann nur an StarWriter Dokumenten aufgerufen werden.

KLASSE:
TextDocument

SYNTAX:
BrowseView

RÜCKGABE:
Boolean

PARAMETER:
Liefert den Wert TRUE, wenn das ONLINE-LAYOUT eingeschaltet ist, anderfalls den Wert FALSE.

BEISPIEL:

```
Sub ExampleBrowseView
Documents.Add "sWriter4"
ActiveDocument.BrowseView = TRUE
MsgBox "Sie haben die Online-Layout" &_
   IIF(ActiveDocument.BrowseView,""," nicht") &_
   " aktiviert",64,"Online-Layout"
End Sub
```

BulletCode-Eigenschaft

Bestimmt oder ermittelt den Zeichencode des Aufzählungszeichens in der aktuellen Numerierungsebene.

KLASSE:
BaseTextList

SYNTAX:
BulletCode

RÜCKGABE:
Long

PARAMETER:
[= Code]: Long-Integer-Ausdruck, der den ASCII-Code des Aufzählungszeichen angibt.

Beispiel:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

BulletFont-Eigenschaft

Bestimmt oder ermittelt den Fontnamen des Aufzählungszeichens in der aktuellen Ebene. Die möglichen Optionen finden Sie unter Format - Numerierung/Aufzählung - Optionen - Zeichen - Schrift.

Klasse:
BaseTextList

SYNTAX:
BulletFont

RÜCKGABE:
String

PARAMETER:
[= FontName]: Zeichenfolge, die den Zeichensatz des Aufzählungszeichens angibt.

BEISPIEL:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

BulletImageURL-Eigenschaft

Bestimmt oder ermittelt den URL der Grafik, die in der aktuellen Ebene als Aufzählungszeichen benutzt wird. Beim Setzen des URL wird die Numerierungsebene auf den Typ Bitmap eingestellt.

KLASSE:
BaseTextList

SYNTAX:
BulletImageURL

RÜCKGABE:
String

PARAMETER:
[= Image]: Zeichenfolge, die den URL der Grafik angibt.

BEISPIEL:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
```

```
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

BulletsAndNumberingDialog-Methode

Ruft den Dialog NUMERIERUNG/AUFZÄHLUNG auf, in dem die Einstellungen für Aufzählungen und Numerierungen vorgenommen werden können.

KLASSE:
BaseText

SYNTAX:
BulletsAndNumberingDialog()

BEISPIEL:

```
Sub ExampleBulletsAndNumberingDialog
Documents.Add "sWriter4"
Selection.Insert("Gleich kommt der Numerierungs-Dialog")
Selection.BulletsAndNumberingDialog()
End Sub
```

Eigenschaften und Methoden mit "C"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "C" beginnen.

Calc-Methode

Berechnet die aktuelle Tabelle neu.

KLASSE:
BaseTextTable

SYNTAX:
Calc()

BEISPIEL:

```
Sub ExampleTableCalc
Const sMessage_1 = "Die Formel aus der Zelle B2 berechnet : "
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoCell("A2", FALSE)
Selection.Insert "2"
```

```
Selection.GotoCell("A3", FALSE)
Selection.Insert "3"
Selection.GotoCell("B2", FALSE)
Selection.Formula = "<A2>+<A3>"
Selection.GotoCell("A2", FALSE)
Selection.Insert "4"
Selection.Calc
Selection.GotoCell("B2", FALSE)
Print sMessage_1 & Selection.LineText
End Sub
```

CalculateSel-Methode

Berechnet die im Dokument selektierte Formel. Das Ergebnis wird in der Zwischenablage abgelegt.

KLASSE:
BaseText

SYNTAX:
CalculateSel()

BEISPIEL:

```
Sub ExampleCalculateSel
Const sMessage_1 = "Die Formel aus der Zelle B2 berechnet : "
Documents.Add("sWriter4")
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.InsertTable(4, 4)
Selection.GotoCell("A2", FALSE)
Selection.Insert "2"
Selection.GotoCell("A3", FALSE)
Selection.Insert "3"
Selection.GotoCell("B2", FALSE)
Selection.Formula = "<A2>+<A3>"
Selection.SelectionTextExt
Selection.CalculateSel
Selection.Escape
Selection.JumpToEndOfDoc
Selection.Insert sMessage_1
ActiveWindow.Paste
End Sub
```

Caption-Eigenschaft

Bestimmt oder ermittelt die Beschriftung des selektierten Kontrollfeldes.

KLASSE:
BaseTextDrawControl

Syntax:
Caption

Rückgabe:
String

Parameter:
[= String]: Zeichenfolge, die die Beschriftung des Kontrollfeldes angibt.

Beispiel:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

CaseMap-Eigenschaft

Angewendet auf ein Font-Objekt bestimmt oder ermittelt diese Eigenschaft den Schrifteffekt des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetCaseMap gesetzte Schrifteffekt abgefragt.

Eine Abfrage der CaseMap-Eigenschaft ist nur dann sinnvoll, wenn mit IsCaseMap bestimmt wurde, daß nach einem bestimmten Schrifteffekt gesucht werden soll, d. h. IsCaseMap = 2.

Siehe auch: IsCaseMap, SetCaseMap.

KLASSE:
ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
CaseMap

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[Value]: Integer-Ausdruck, der den Schrifteffekt angibt. Folgende Werte sind möglich:

0 - Keine Auszeichnung

1 - Versalien

2 - Gemeine

3 - Titelschrift

4 - Kapitälchen

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
```

```
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
```

```
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

CellName-Eigenschaft

Ermittelt den Namen der aktuellen Tabellenzelle oder der Tabellenselektion in der Form "A1" oder "A1:B5".

KLASSE:
BaseTextTable

SYNTAX:
CellName

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

ChangeDatabaseField-Methode

Wechselt die Datenbank an allen Feldern und Bereichen aus (auch in Bedingungen).

KLASSE:
BaseTextSelection

SYNTAX:
ChangeDatabaseField(OldName As String, NewName As String)

PARAMETER:
OldName: Alter Datenbank- und Tabellenname

NewName: Neuer Datenbank- und Tabellenname

Syntax von Old/NewName: Datenbankname.Tabellenname

BEISPIEL:

```
Sub ExampleChangeDatabaseField
Const sDBName = "TestDB"
Const sTableName_1 = "Tabelle1"
Const sTableName_2 = "Tabelle2"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sMessage_1 = "Alle Felder der Tabelle : "
Dim stDataRow As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
Documents.Add( "sWriter4" )
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
```

```
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = Application.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,SINGLE,FALSE"
Next iCount
vStatus = Application.CreateTable(sTableName_1, stDataRow)
vStatus = Application.OpenTableConnection(sTableName_1)
Selection.Insert sMessage_1 & sTableName_1
Selection.InsertPara
For iCount = 1 To Application.DataColumnNumber
Selection.Insert Application.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDBName, sTableName_1,
Application.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
vStatus = Application.CreateTable(sTableName_2, stDataRow)
vStatus = Application.OpenTableConnection(sTableName_1)
Selection.ChangeDatabaseField( sDBName & "." &sTableName_1,
sDBName & "." &sTableName_2 )
Application.CloseTableConnection(sTableName_1)
vStatus = Application.DeleteTable( sTableName_1 )
Application.CloseTableConnection(sTableName_2)
vStatus = Application.DeleteTable( sTableName_2 )
Application.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection( sDBName )
End Sub
```

ChapterNumberingDialog-Methode

Ruft den Dialog KAPITELNUMERIERUNG auf.

KLASSE:
BaseText

SYNTAX:
ChapterNumberingDialog()

BEISPIEL:

```
Sub ExampleChapterNumberingDialog
Documents.Add "sWriter4"
Selection.Insert("Gleich kommt der Numerierungs-Dialog")
Selection.ChapterNumberingDialog()
End Sub
```

CharSet-Eigenschaft

Bestimmt oder ermittelt den Zeichensatz des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen; beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetFont gesetzte Zeichensatz abgefragt.

Siehe auch: SetFont, IsFont, StyleName, Pitch, Family, FamilyName.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
CharSet

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Set]: Integer-Ausdruck, der den Zeichensatz bestimmt, der der Schrift zugewiesen werden kann:

0 - Voreinstellung, unbekannter Zeichensatz

1 - ANSI-Zeichensatz

2 - Macintosh-Zeichensatz

3 - IBM PC Zeichensatz 437

4 - IBM PC Zeichensatz 850

5 - IBM PC Zeichensatz 860

6 - IBM PC Zeichensatz 861

7 - IBM PC Zeichensatz 863

8 - IBM PC Zeichensatz 865

9 - System-Zeichensatz

10 - Symbol-Zeichensatz

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Beispiel:

```
Sub ExampleCharSet
Documents.Add "sWriter4"
Print "CharSet liefert : " & Selection.Font.CharSet
End Sub
```

CharStyle-Eigenschaft

Die Eigenschaft enthält den Namen der bei der Selektion gültigen Zeichenvorlage.

Klasse:
BaseTextSelection

Syntax:
CharStyle

Rückgabe:
String

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleCharStyle
Const sStyleName = "Neuer Style"
Const sMessage_1 = "Hier wird die Zeichenvorlage neu gesetzt"
Const sMessage_2 = "Die neue Zeichenvorlage heißt : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine True
REM Neue Zeichenvorlage anlegen
Selection.StyleNewByExample( sStyleName, 1 )
Selection.CharStyleApply( sStyleName )
Print sMessage_2 & Selection.CharStyle
End Sub
```

CharStyleApply-Methode

Weist dem selektierten Text eine Zeichenvorlage zu. Ohne Übergabe eines Parameters wird der Dialog Zeichen aufgerufen.

Klasse:
BaseText

Syntax:
CharStyleApply(Style As String)

PARAMETER:
Style: Ein Zeichenfolgeausdruck, der die Bezeichnung der Zeichenvorlage angibt, die dem selektierten Text zugewiesen werden soll.

BEISPIEL:

```
Sub ExampleCharStyle
Const sStyleName = "Neuer Style"
Const sMessage_1 = "Hier wird die Zeichenvorlage neu gesetzt"
Const sMessage_2 = "Die neue Zeichenvorlage heißt : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine True
REM Neue Zeichenvorlage anlegen
Selection.StyleNewByExample( sStyleName, 1 )
Selection.CharStyleApply( sStyleName )
Print sMessage_2 & Selection.CharStyle
End Sub
```

CharStyleCount-Eigenschaft

Diese Eigenschaft liefert die Anzahl der im Dokument vorhandenen Zeichenvorlagen.

KLASSE:
BaseText

SYNTAX:
CharStyleCount

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
sub ExampleCharStyleName
Const sText="Das ist ein Text in dem Zeichenvorlagen verwendet
werden."
Const sCharStyle1="Fett"
Const sCharStyle2="Unterstrichen"
Documents.Add "sWriter4"
Selection.Insert sText
Selection.Escape
Selection.JumpToStartOfDoc
REM zuerst zwei Zeichenvorlagen definieren
With Selection
.GoToNextWord( TRUE )
.Font.Bold=8 REM auf Fett setzen
.StyleNewByExample( sCharStyle1, 1)
```

```
.GoToNextWord( FALSE )
.GoToNextWord( FALSE )
.GoToNextWord( TRUE )
.Font.Underline=2 REM unterstrichen und Rot
.Font.Color= RGB(255,0,0)
.StyleNewByExample( sCharStyle2, 1)
End With
Wait 1000    REM Vorlagen anwenden
With Selection
   .CharStyleApply( sCharStyle1 )
   .GoToNextWord( FALSE )
   .GoToNextWord( FALSE )
   .GoToNextWord( TRUE )
   .GoToNextWord( TRUE )
   .CharStyleApply( sCharStyle2 )
End With
Rem Vorlagen ausgeben
For iCount=1 To Selection.CharStyleCount
MsgBox " " & iCount &": "& Selection.GetCharStyleName(a%),64,"
" & Selection.CharStyleCount  & " definierte Zeichenvorlage"
next iCount
end sub
```

ClearHistory-Methode

Bewirkt ein Löschen des Undo-Puffers.

KLASSE:
BaseTextEditView

SYNTAX:
ClearHistory()

BEISPIEL:

```
Sub ExampleClearHistory
REM Es wird der Laufzeitfehler abgefangen
On Error Goto UNDOERROR
Const sMessage_1 = "Der Undo-Puffer wird gelöscht"
Const sMessage_2 = "Dieser Text kann nicht rückgängig gemacht
werden"
Const sMessage_3 = "Mit gelöschtem Undo-Puffer kann nichts
rückgängig gemacht werden"
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Activewindow.ClearHistory
REM Es kommt hier zu einem Laufzeitfehler, da das Undo ausge-
schaltet ist
Selection.Undo
Exit Sub
```

```
UNDOERROR:
Selection.Insert sMessage_3
End Sub
```

CloseDatabaseConnection-Methode

Schließt eine geöffnete Datenbank inklusive aller Verbindungen zu darin enthaltenen Tabellen.

Siehe auch: OpenDatabaseConnection

KLASSE:
BaseTextEditView

SYNTAX:
CloseDatabaseConnection(DBName As String)

PARAMETER:
DBName: Ein Zeichenfolgeausdruck, der die Bezeichnung der Datenbank aufnimmt, die geschlossen werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
```

```
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

CloseQueryConnection-Methode

Beendet die Verbindung zur einer Datenbank-Abfrage.

Siehe auch: OpenQueryConnection

KLASSE:
BaseTextEditView

SYNTAX:
CloseQueryConnection(QueryName As String)

PARAMETER:
[QueryName]: Name der Datenbank-Abfrage, zu der die Verbindung aufgehoben werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, dann wird die Verbindung zur zuletzt verwendeten Abfrage beendet.

BEISPIEL:

```
Sub ExampleQueries
Const sQueryName = "TestAbfrage"
Const sMessage = "Der Inhalt des ersten Feldes : "
Dim sDataName As String, sTabName As String, sTPath As String
Dim vStatus As Variant
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
ActiveWindow.DataFirstRecord
Print sMessage & ActiveWindow.DataColumn(1)
ActiveWindow.CloseQueryConnection(sQueryName)
ActiveWindow.DeleteQuery sQueryName
End Sub
```

CloseTableConnection-Methode

Beendet die Verbindung zu einer Datenbank-Tabelle.

Siehe auch: OpenTableConnection

KLASSE:
BaseTextEditView

SYNTAX:
CloseTableConnection(TableName As String)

PARAMETER:
[TableName]: Name der Datenbank-Tabelle, zu der die Verbindung aufgehoben werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, dann wird die Verbindung zur zuletzt verwendeten Tabelle beendet.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
```

```
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

Color-Eigenschaft

Bestimmt oder ermittelt die Farbe des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen.

KLASSE:
TextDrawFont, TextFont

SYNTAX:
Color

RÜCKGABE:
Long

PARAMETER:
[= Value]: Beliebiger numerischer Ausdruck, der die Farbe des Textes angibt.

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
```

```
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Columns-Eigenschaft

Ermittelt die Anzahl der Spalten einer Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
Columns

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

Contour-Eigenschaft

Ermittelt die mit SetContour gesetzte Suchoption für das Zeichenattribut Kontur, wenn die Eigenschaft IsContour den Wert 2 liefert.

Siehe auch: SetContour, IsContour.

KLASSE:
ReplaceAttributes, SearchAttributes, BaseTextFrame

SYNTAX:
Contour

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExampleSetContour
Const sMessage_1 = "Es wird ein, mit Konturschrift formatierter
Text gesucht"
Const sMessage_2 = "Bei diesem Absatz ist der Text als Kontur-
schrift formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.OutlineFont = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsContour = 2
ActiveWindow.SearchAttributes.SetContour( TRUE )
If ActiveWindow.SearchAttributes.Contour = TRUE Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

ContourDialog-Eigenschaft

Bestimmt oder ermittelt den Zustand des Fensters Kontur-Editor.

Klasse:
TextSelection

Syntax:
ContourDialog

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Kontur-Editor ein- (True) oder ausgeblendet (False) ist.

Beispiel:

```
Sub ExampleContourDialog
Dim sGrfPath as String
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Selection.ContourDialog = TRUE
End Sub
```

ContourWrap-Eigenschaft

Mit dieser Eigenschaft kann die Umlaufoption KONTUR an Rahmen erfragt oder gesetzt werden.

KLASSE:
BaseTextFrame

SYNTAX:
ContourWrap

RÜCKGABE:
Boolean

PARAMETER:
True: Umlaufoption KONTUR aktivieren

False: Umlaufoption KONTUR deaktivieren

BEISPIEL:

```
Sub ExampleGraphicContour
Const sText1 ="Das ist ein Text, der um die Kontur der Grafik
fließt."
Dim sGrfName as String
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows" )
sGrfName = Selection.GraphicName
Selection.ContourWrap= TRUE
Selection.Wrap(2)
Selection.Escape
For iCount=1 To 10
Selection.Insert( sText1)
Next iCount
Selection.InsertPara
For iCount=1 To 5
Selection.Insert( sText1)
Next iCount
Wait 2000
ActiveWindow.Graphics(sGrfName).AnchorOnlyWrap = TRUE
ActiveWindow.Graphics(sGrfName).GotoAnchor
End Sub
```

ControlObject-Eigenschaft

Liefert das selektierte Kontrollfeld als Objekt zurück und ermöglicht den Zugriff auf die Kontrollfeld-Eigenschaften.

Diese Eigenschaft gilt nur für Kontrollfelder, die sich in einem Dokument befinden.

KLASSE:
BaseTextDrawControl

SYNTAX:
ControlObject

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

Controls-Eigenschaft

Ermöglicht einen Zugriff auf die Collection der Kontrollfelder.

Anwendbar sind folgende Eigenschaften und Methoden:

Count, Remove(Index), Remove(Name)

und der indizierte Zugriff mittels

Controls(Index).Befehl

Controls(Name).Befehl,

wobei "Befehl" ein für das Interface DrawControl gültiger Bezeichner ist. Index beginnt bei eins.

Diese Eigenschaft gilt nur für Kontrollfelder innerhalb eines Dokumentes.

Wird ein Control-Objekt über dessen Namen angesprochen, so muß zwischen Groß- und Kleinschreibung unterschieden werden.

Siehe auch: GroupCount, GroupObject.

KLASSE:
BaseTextEditView

SYNTAX:
Controls

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
```

```
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

ConvertTableText-Methode

Konvertiert einen Text in eine Tabelle und umgekehrt. Ohne Übergabeparameter wird bei selektiertem Text der Dialog TEXT IN TABELLE UMWANDELN und bei einer selektierten Tabelle TABELLE IN TEXT UMWANDELN aufgerufen, in dem das Trennzeichen für die Konvertierung festgelegt werden kann.

Um Text zu einer Tabelle konvertieren zu können, muß bei Aufruf dieser Methode ein Textbereich selektiert sein, es genügt nicht, nur den Cursor in den Text zu setzen. Soll eine Tabelle konvertiert werden, dann funktioniert diese Methode auch, wenn sich der Cursor nur innerhalb der Tabelle befindet.

KLASSE:
BaseTextSelection

SYNTAX:
ConvertTableText(Delimiter As String)

PARAMETER:
[Delimiter]: Zeichenfolgenausdruck, der das Trennzeichen angibt, das den Beginn einer neuen Spalte kennzeichnet. Dieser Parameter ist optional.

BEISPIEL:

```
Sub ExampleConvertTableText
Documents.Add( "sWriter4" )
Selection.InsertTable(1, 2)
Selection.Insert( "Eintrag 1" )
Selection.JumpToNextCell
Selection.Insert( "Eintrag 2" )
```

```
REM Als Texttrenner einen Tabulator verwenden
Selection.ConvertTableText(chr(9))
End Sub
```

Copy-Methode

Kopiert die aktuelle Selektion in das Clipboard.

KLASSE:
TextDrawText, WebSourceView, BaseTextSelection

SYNTAX:
Copy()

BEISPIEL:

```
Sub ExampleCutCopyPaste
Dim sWin As String REM für den  Fensternamen
REM TextDokument öffnen
Documents.Add("sWriter4")
sWin = ActiveWindow.Name
REM jetzt wird ein wenig HTML-Code eingegeben
ActiveWindow.Selection.Insert(Chr(60)&"HTML"&Chr(62)&Chr(60)&"B
ODY"&Chr(62)&"This is HTML !"&Chr(60)&"/BODY"&Chr(62))
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut REM HTML-Code ausschneiden
Documents.Add("sWriter4/Web") REM HTML-Dokument öffnen und ...
Wait 100
ActiveDocument.FrameSetSource = True REM ... in die Quelltex-
tansicht ...
ActiveWindow.SelectAll
ActiveWindow.Paste REM .. den HTML-Code einfügen.
ActiveDocument.FrameSetSource = False
REM und zurück ...
ActiveDocument.FrameSetSource = TRUE
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveDocument.FrameSetSource = False
Windows( sWin ).Activate
ActiveWindow.Paste
end sub
```

Count-Eigenschaft

Bei einem DropCap-Objekt bestimmt oder ermittelt diese Eigenschaft die Anzahl der Zeichen eines Absatzes oder eines beliebigen Textes, die als Initiale dargestellt werden sollen. Bei einer Objekt-Collection wird die Anzahl der jeweils vorhandenen Elemente innerhalb der Collection ermittelt.

Siehe auch: Distance, Lines.

KLASSE:
DropCap

SYNTAX:
Count

RÜCKGABE:
Integer

PARAMETER:
Beim DropCap-Objekt:

[= CharCount]: Numerischer Ausdruck vom Typ Integer, der die Anzahl der ersten Zeichen eines Absatzes oder eines beliebigen Textes angibt, die als Initiale dargestellt werden sollen. Bis zu neun Zeichen können als Initiale dargestellt werden.

Bei einer Collection:

(nur lesen)

BEISPIEL:

```
Sub ExampleDropCap
Documents.Add("sWriter4")
Selection.Insert "Das erste Zeichen"
Selection.InsertLineBreak
Selection.Insert "wird über drei"
Selection.InsertLineBreak
Selection.Insert "Zeilen dargestellt"
Selection.InsertLineBreak
Selection.GotoStartOfDoc FALSE
Selection.Paragraph.DropCap.Count = 1
Selection.Paragraph.DropCap.Lines = 3
Selection.Paragraph.DropCap.Distance = "0.5cm"
End Sub
```

CreateAbstract-Methode

Ruft den Dialog AUTOABSTRACT ERZEUGEN auf. In diesem Dialog können die Einstellungen zur Erzeugung eines AutoAbstract-Textdokuments vorgenommen werden, einer Gliederung also, die in ein neues StarWriter-Dokument übernommen wird.

Diese Methode ist nur dann aufrufbar, wenn in Ihrem Dokument Kapitelebenen vorhanden sind, die mit einer bestimmten Absatzvorlage verknüpft sind. Diese werden im Office unter EXTRAS - KAPITELNUMERIERUNG definiert.

KLASSE:
TextDocument

SYNTAX:
CreateAbstract()

BEISPIEL:

```
REM Damit die Prozedure funktioniert müssen bereits
REM auf dem Dokument Kapitelebenen vorhanden sein,
Sub ExampeCreateAbstract
Activedocument.CreateAbstract
End Sub
```

CreateForm-Methode

Legt ein neues Datenbank-Formular, aus einem bestehenden StarOffice-Textdokument in der geöffneten Datenbank ab.

KLASSE:
BaseTextEditView

SYNTAX:
CreateForm(FormName As String, URL As String)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Name des neuen Formulars.

URL: Pfadangabe für die SDW-Datei, aus der das Formular erzeugt werden soll.

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Database & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close( True, sPath )
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
```

```
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateForm( sFormName, sPath )
ActiveWindow.CreateReport( sReportName, sPath )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " & sTab-
Name, FALSE )
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
ActiveWindow.DeleteReport sReportName

ActiveWindow.DeleteQuery sQueryName
End Sub
```

CreateGlossaryGroup-Methode

Legt eine neue AutoText-Gruppe an.

KLASSE:
BaseText

SYNTAX:
CreateGlossaryGroup(Group As String)

RÜCKGABE:
Boolean

PARAMETER:
Group: Zeichenfolge, die den Namen der neuen Gruppe angibt. Der Name darf noch nicht vergeben sein.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
```

```
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

CreateIndex-Methode

Legt einen Index auf eine Tabellenspalte einer Datenbank-Tabelle an.

KLASSE:
BaseTextEditView

SYNTAX:
CreateIndex(TableName As String, IndexName As String, Columns As Object, Unique As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Tabelle.

IndexName: Name des Indexes.

Columns: Parameter vom Typ Strings, der die Indizierung einer Tabellenspalte bestimmt. Columns besteht aus einer String-Variablen der Form "ColumnName, Order".

ColumnName: Name der Tabellenspalte, die mit einem Index belegt werden soll.

Order: Boolescher Ausdruck, der angibt, ob der Index aufsteigend (True) oder absteigend (False) ist.

Eine gültige Indexbeschreibung sieht z. B. so aus: "VORNAME,TRUE".

Unique: Boolescher Ausdruck, der angibt, ob gleiche Feldinhalte zusammengefaßt (True) oder separat betrachtet (False) werden sollen. Taucht ein Feldinhalt innerhalb einer Spalte mehrfach auf, dann wird mit Unique = True immer nur der erste Datensatz angezeigt.

BEISPIEL:

```
Sub ExampleDatabaseIndex
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Dim stDataRow As New Strings
Dim stIndexStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim sStatus As Integer, iCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
sStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
```

```
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
stIndexStrings.Add sTextName & "1,TRUE"
sStatus = Activewindow.CreateTable(sTabName, stDataRow)
sStatus = Activewindow.OpenTableConnection(sTabName)
sStatus = Activewindow.CreateIndex(sTabName, sIndexName, stIn-
dexStrings, FALSE)
sStatus = Activewindow.DeleteIndex(sIndexName)
Activewindow.CloseTableConnection(sTabName)
sStatus = Activewindow.DeleteTable(sTabName)
Activewindow.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

CreateQuery-Methode

Legt eine Datenbank-Abfrage an.

KLASSE:
BaseTextEditView

SYNTAX:
CreateQuery(QueryName As String, SQL_Statement As String, Native As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Abfrage

SQL_Statement: Gültiger SQL-Ausdruck

Native: Gibt an, ob es sich um eine Native SQL handelt (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleQueries
Const sQueryName = "TestAbfrage"
Const sMessage = "Der Inhalt des ersten Feldes : "
Dim sDataName As String, sTabName As String, sTPath As String
Dim vStatus As Variant
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
ActiveWindow.DataFirstRecord
```

```
Print sMessage & ActiveWindow.DataColumn(1)
ActiveWindow.CloseQueryConnection(sQueryName)
ActiveWindow.DeleteQuery sQueryName
End Sub
```

CreateReport-Methode

Legt einen neuen Datenbank-Bericht in der aktuell geöffneten Datenbank an. Als Basis für diesen Bericht wird ein StarWriter-Dokument verwendet.

Siehe auch: DeleteReport

KLASSE:
BaseTextEditView

SYNTAX:
CreateReport(ReportName As String, URL As String)

RÜCKGABE:
String

PARAMETER:
ReportName: Gibt den Namen des neuen Berichts an.

URL: Gibt den Pfad an, in dem das StarWriter-Dokument steht, aus dem der Bericht erzeugt werden soll.

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Database & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close( True, sPath )
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
```

```
ActiveWindow.CreateForm( sFormName, sPath )
ActiveWindow.CreateReport( sReportName, sPath )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " & sTab-
Name, FALSE )
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
ActiveWindow.DeleteReport sReportName
ActiveWindow.DeleteQuery sQueryName
End Sub
```

CreateTable-Methode

Legt eine neue Datenbank-Tabelle an.

KLASSE:
BaseTextEditView

SYNTAX:
CreateTable(TableName As String, Columns As Object)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Gibt den Namen der Tabelle an, die neu angelegt werden soll.

Columns: Strings-Variable, die die Datensatzfelder der Tabelle bestimmt. Columns setzt sich aus mehreren String-Variablen zusammen, die je eine Spalte der Tabelle definieren. Eine Spaltenbeschreibung hat folgende Bestandteile:

Spalte=Feldname, Feldtyp, Feldlänge, Eingabe.

Feldname: Gibt den Namen der Spalte an.

Eingabe: Gibt an, ob eine Eingabe später zwingend erforderlich ist (True) oder ob ein Datenfeld in dieser Spalte auch leer sein darf (False).

Feldtyp und Feldlänge legen Typ und ggf. Subtyp einer Spalte fest. Je nach Datenbankstruktur können Feldtyp und Feldlänge variieren. Bei einer Tabelle im Format dBaseIII gelten folgende Werte:

Wert	Feldtyp	Subtyp
TEXT	Text	Zeichenlänge
COUNTER	Zähler	
IMAGE	Bild	
MEMO	Memo	
NUMERIC	Zahl	Zahlenformat
DATE/TIME	Datum/Zeit	
DATE	Datum	
TIME	Zeit	
CURRENCY	Währung	
BOOL	Ja/Nein	

Der Subtyp Zahlenformat kann folgende Werte annehmen:

Basic-Wert	dBase-Zahlenformat	Länge
INTEGER	Smallint	2 Byte = 16 Bit
LONG	Integer	4 Byte = 32 Bit
SINGLE	Single	4 Byte Fließkomma
DOUBLE	Double	8 Byte Fließkomma

Beispiel:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
```

```
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
```

```
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

Crosshair-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für Hilfslinien.

KLASSE:
ViewSettings

SYNTAX:
Crosshair

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Hilfslinien angezeigt werden (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

CurFieldSubType-Eigenschaft

Liefert die TypeID des Feldbefehles an der Cursorposition. Wird der Wert -1 geliefert, zeigt das an, daß kein Feldbefehl an der aktuellen Cursorposition existiert.

KLASSE:
BaseText

SYNTAX:

CurFieldSubType

RÜCKGABE:
Integer

BEISPIEL:

```
Sub ExampleCurField
Const sCreateDate = "Erzeugt"
Const sMessage = "Dieser Feldbefehl enthält das Erzeugungsdatum
: "
Const sFieldType = "Der Feldtyp des aktuellen Feldes : "
Const sFieldSubType = "Der SubFeldtyp des aktuellen Feldes : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage
```

```
Selection.InsertField( 18, 8, sCreateDate, "", 2 )
Selection.GoLeft 1, TRUE
Print sFieldType & Selection.CurFieldType
Print sFieldSubType & Selection.CurFieldSubType
End Sub
```

CurFieldType-Eigenschaft

Liefert die TypeID des Feldbefehles an der Cursorposition. Wird der Wert -1 geliefert, zeigt das an, daß kein Feldbefehl an der aktuellen Cursorposition existiert.

KLASSE:
BaseText

SYNTAX:
CurFieldType

RÜCKGABE:
Integer

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleCurField
Const sCreateDate = "Erzeugt"
Const sMessage = "Dieser Feldbefehl enthält das Erzeugungsdatum
: "
Const sFieldType = "Der Feldtyp des aktuellen Feldes : "
Const sFieldSubType = "Der SubFeldtyp des aktuellen Feldes : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.InsertField( 18, 8, sCreateDate, "", 2 )
Selection.GoLeft 1, TRUE
Print sFieldType & Selection.CurFieldType
Print sFieldSubType & Selection.CurFieldSubType
End Sub
```

CursorXPos-Eigenschaft

Ermittelt die aktuelle horizontale Position der Schreibmarke. Die Position ist immer relativ zum Anfang der ersten Seite des Dokumentes.

KLASSE:
BaseText

SYNTAX:
CursorXPos

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleCursorPos
Dim vX as Variant, vY As Variant
Documents.Add("sWriter4")
Selection.InsertPara
Selection.InsertPara
Selection.InsertPara
Selection.Insert("Das ist ein Text der die Einfügemarke ein
wenig verschiebt.>")
vX = Selection.CursorXPos
vY = Selection.CursorYPos
Selection.Insert("< Die Position lautete: X=" & vX & "; Y=" &
vY)
End Sub
```

CursorYPos-Eigenschaft

Ermittelt die aktuelle vertikale Position der Schreibmarke. Die Position ist immer relativ zum Anfang der ersten Seite des Dokumentes.

Klasse:
BaseText

Syntax:
CursorYPos

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleCursorPos
Dim vX as Variant, vY As Variant
Documents.Add("sWriter4")
Selection.InsertPara
Selection.InsertPara
Selection.InsertPara
Selection.Insert("Das ist ein Text der die Einfügemarke ein
wenig verschiebt.>")
vX = Selection.CursorXPos
vY = Selection.CursorYPos
```

```
Selection.Insert("< Die Position lautete: X=" & vX & "; Y=" &
vY)
End Sub
```

Cut-Methode

Kopiert die aktuelle Selektion in das Clipboard und löscht sie anschließend.

KLASSE:
TextDrawText, WebSourceView, BaseTextSelection

SYNTAX:
Cut()

BEISPIEL:

```
Sub ExampleCutCopyPaste
Dim sWin As String REM für den  Fensternamen
REM TextDokument öffnen
Documents.Add("sWriter4")
sWin = ActiveWindow.Name
REM jetzt wird ein wenig HTML-Code eingegeben
ActiveWindow.Selection.Insert(Chr(60)&"HTML"&Chr(62)&Chr(60)&"B
ODY"&Chr(62)&"This is HTML !"&Chr(60)&"/BODY"&Chr(62))
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut REM HTML-Code ausschneiden
Documents.Add("sWriter4/Web") REM HTML-Dokument öffnen und ...
Wait 100
ActiveDocument.FrameSetSource = True REM ... in die Quelltex-
tansicht ...
ActiveWindow.SelectAll
ActiveWindow.Paste REM .. den HTML-Code einfügen.
ActiveDocument.FrameSetSource = False
REM und zurück ...
ActiveDocument.FrameSetSource = TRUE
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveDocument.FrameSetSource = False
Windows( sWin ).Activate
ActiveWindow.Paste
end sub
```

Eigenschaften und Methoden mit "D"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "D" beginnen.

DataAppendRecord-Methode

Fügt der aktuellen Datenbank einen neuen Datensatz hinzu. Die Datenfeldinhalte müssen als Zeichenfolge durch Kommata getrennt in der Reihenfolge der Datenspalten des Datensatzes übergeben werden.

Konnte der Datensatz erfolgreich hinzugefügt werden, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
DataAppendRecord(NewRecord As String)

RÜCKGABE:
Boolean

PARAMETER:
NewRecord: Zeichenfolgeausdruck, der die Inhalte der Datenfelder des neuen Datensatzes durch Kommata getrennt angibt.

In NewRecord müssen alle Datenfelder des neuen Datensatzes berücksichtigt werden. Soll einem Feld kein Wert zugewiesen werden, wird dies mit zwei aufeinanderfolgenden Kommata angegeben.

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
```

```
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
```

```
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataAppendWholeRecord-Methode

Fügt der aktuellen Datenbank einen neuen Datensatz hinzu. Konnte das Hinzufügen des neuen Datensatzes erfolgreich durchgeführt werden, liefert die Methode den Wert True zurück, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
DataAppendWholeRecord(ColumnNames As Object, Content As Object)

RÜCKGABE:
Boolean

PARAMETER:
ColumnNames: Strings-Variable, die die Spaltennamen enthält.

Content: Strings-Variable, die den Inhalt der Spalten enthält.

Wenn nur ein Parameter übergeben wird, wird der gesamte Datensatzinhalt im ersten Parameter erwartet. Wenn beide Parameter übergeben werden, werden nur die Spalten gefüllt, die in ColumnNames angegeben wurden. Alle anderen Spalten bleiben leer.

Beispiel:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
```

```
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataBaseBrowser-Methode

Öffnet den Datenbankbeamer und zeigt darin die am Dokument gesetzte Datenbanktabelle an. Das ist die Tabelle, die zuletzt im Dokument verwendet wurde.

Diese Methode entspricht einem Betätigen der Funktionstaste (F4) bei geöffnetem Textdokument. Ist keine Datenbanktabelle am Dokument gesetzt, wird im Beamer das Adreßbuch eingeblendet.

KLASSE:
StarWriter

SYNTAX:
DataBaseBrowser()

BEISPIEL:

```
Sub ExampleDataBaseBrowser
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Dim sDataName As String, sTableName As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 to ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
ActiveWindow.DataUpdateFields( sDataName, sTableName, "SELECT *
FROM " & sTableName, 1 )
Activemodule.DataBaseBrowser
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

DataColumn-Methode

Liefert den Inhalt der angegebenen Spalte des aktuellen Datensatzes als String zurück.

KLASSE:
BaseTextEditView

SYNTAX:
DataColumn(ColumnNumber As String)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Zeichenfolge, die die Spalte angibt, deren Inhalt ermittelt werden soll. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

BEISPIEL:

```
Sub ExampleDataRecord
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Activewindow.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Activewindow.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
```

```
Print "IsTableReadonly liefert : " &  Activewindow.IsTableRea-
dOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Activewindow.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Activewindow.GetDataba-
seLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Activewindow.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord  REM Spingt zum nächsten
Datensatz
vStatus = Activewindow.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Activewindow.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Activewindow.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Activewindow.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Activewindow.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End if
End if
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataColumnName-Methode

Liefert den Spaltennamen der angegebenen Spalte aus einer Datenbank-Tabelle.

KLASSE:
BaseTextEditView

SYNTAX:
DataColumnName(ColumnNumber As Long)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Nummer des Datenbankfeldes, dessen Bezeichnung ermittelt werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
```

```
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataColumnNumber-Methode

Liefert die Anzahl der Spalten einer Datenbank-Tabelle.

KLASSE:
BaseTextEditView

SYNTAX:
DataColumnNumber()

RÜCKGABE:
Integer

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
```

```
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataColumnSubType-Methode

Liefert den Subtyp der Spalte einer Datenbank-Tabelle.

Mit DataColumnType ermittelt man den Typ eines Datenbankfeldes mit DataColumnSubType wird dieser Typ weiter aufgeschlüsselt. Der Subtyp eines Feldes ändert sich mit dem Feldtyp.

Feldtyp	Subtype
Text	Feldlänge
Zahl	Zahlendatentyp
Datum/Zeit	Leer
Ja/Nein	Leer
Memo	Leer

Siehe auch: CreateTable

KLASSE:
BaseTextEditView

SYNTAX:
DataColumnSubType(ColumnNumber As String)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Zeichenfolge, die die Spalte angibt, deren Subtyp ermittelt werden soll. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
```

```
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
```

```
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataColumnType-Methode

Gibt den Feldtyp des angegebenen Datenbankfeldes zurück.

KLASSE:
BaseTextEditView

SYNTAX:
DataColumnType(ColumnNumber As String)

RÜCKGABE:
String

PARAMETER:
ColumnNumber: Zeichenfolge, die die Spalte angibt, deren Feldtyp ermittelt werden soll. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

Die folgenden Feldtypen werden von der Methode zurückgegeben:

Wert : Feldtyp : Beschreibung

0 : Bookmark : Das Bookmarkfeld der Datenbank

1 : Serial : Selbständig inkrementierendes Datenbankfeld (Zählerfeld)

2 : Char : Zeichenfeld mit fester Länge

3 : VarChar : Zeichenfeld mit variabler Länge

4 : LongVarChar : Zeichenfeld mit variabler Länge und sehr großer Kapazität (Memofeld)

5 : Decimal : Festkommazahl

6 : Numeric : Festkommazahl (größere Kapazität als Decimal)

7 : Bit : Ja/Nein-Feld (1 Bit)

8 : TinyInt : Feld für Ganzzahlen bis 8 Bit

9 : SmallInt : Feld für Ganzzahlen bis 16 Bit

10 : Integer : Feld für Ganzzahlen bis 32 Bit

11 : BigInt : Feld für Ganzzahlen bis 64 Bit

12 : Real : Fließkommazahl

13 : Float : Fließkommazahl (größere Kapazität als Real)

14 : Double : Fließkommazahl (größere Kapazität als Float)

15 : Binary : Feld für Binärdaten fester Länge

16 : VarBinary : Feld für Binärdaten variabler Länge

17 : LongVarBinary : Feld für Binärdaten variabler Länge (größere Kapazität als VarBinary)

18 : Date : Datumsfeld

19 : Time : Uhrzeitfeld

20 : TimeStamp : Feld, das zur Verwaltung mehrerer Benutzer benötigt wird

21 : Currency : Währungsformatfeld

22 : Boolean : Ja/Nein-Feld (1 Bit)

Größe und Eigenschaften einzelner Datenbankfelder können sich abhängig von der verwendeten Plattform und Datenbank voneinander unterscheiden. Einige Datenbanken weisen nicht alle aufgeführten Datenbankfelder auf.

Beispiel:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
```

```
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataDefinedDatabase-Methode

Liefert den Namen einer angemeldeten Datenbank.

Siehe auch: DataDefinedDatabaseNumber

KLASSE:
BaseTextEditView

SYNTAX:
DataDefinedDatabase(DefinedDatabaseNumber As Long)

RÜCKGABE:
String

PARAMETER:
DefinedDatabaseNumber: Nummer der Datenbank, deren Name ermittelt werden soll. Der Wertebereich liegt zwischen 1 und der Anzahl der angemeldeten Datenbanken.

BEISPIEL:

```
Sub ExampleDataRecord
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Activewindow.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Activewindow.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print "IsTableReadonly liefert : " &  Activewindow.IsTableRea-
```

```
dOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Activewindow.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Activewindow.GetDataba-
seLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Activewindow.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord  REM Spingt zum nächsten
Datensatz
vStatus = Activewindow.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Activewindow.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Activewindow.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Activewindow.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Activewindow.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End if
End if
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataDefinedDatabaseNumber-Methode

Liefert die Anzahl der angemeldeten Datenbanken.

KLASSE:
BaseTextEditView

SYNTAX:
DataDefinedDatabaseNumber()

RÜCKGABE:
Integer

BEISPIEL:

```
Sub ExampleDataRecord
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Activewindow.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Activewindow.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print "IsTableReadonly liefert : " &  Activewindow.IsTableRea-
dOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Activewindow.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Activewindow.GetDataba-
seLimits(1)
```

```
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Activewindow.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord  REM Spingt zum nächsten
Datensatz
vStatus = Activewindow.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Activewindow.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Activewindow.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Activewindow.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Activewindow.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
     vStatus = Activewindow.DataSwitch(sDataName, sOld-
TableName)
Else
sOldTableName = sTableName
End if
End if
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataDeleteRecord-Methode

Löscht den selektierten Datensatz in der aktuellen Datenbank.

KLASSE:
BaseTextEditView

SYNTAX:
DataDeleteRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Dim stDataRow as new Strings
Dim stNewDataRow as new Strings
Dim stAllQueries as new strings
Dim stQueryRows as new strings
Dim stAppendRec as new Strings
Dim stColumnsStrings as new Strings
Dim sDBPfad as string, sConfigStr as String, sSQLStatement as
string
Dim vStatus as integer, iCount as integer,  iQueCount as
integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
application.AddDatabaseConnection(sDBPfad & "TestDB", sCon-
figStr, "TestDB")
vStatus = ActiveWindow.OpenDatabaseConnection("TestDB")
for iCount=1 to 3
stDataRow.Add "TEXT_" & iCount & ",TEXT,10,FALSE"
stDataRow.Add "NUMERIC_" & iCount & ",NUMERIC,INTEGER,FALSE"
next iCount
vStatus = ActiveWindow.CreateTable("TestTab", stDataRow)
vStatus = ActiveWindow.OpenTableConnection("TestTab")
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns("TestTab")
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print "Fehler in der Tabellenstruktur" &
Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print "Es wurde nicht zum ersten Datensatz gesprungen"
End if
for iCount=1 to 3
```

```
stAppendRec.Add "TEXT_" & iCount
stAppendRec.Add iCount
stColumnsStrings.Add "TEXT_" & iCount
stColumnsStrings.Add "NUMERIC_" & iCount
next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print "Der Feldtyp des dritten Feldes lautet : " & Active-
Window.DataColumnType(3)
Print "Das dritten Feld hat eine Größe von : " &
ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print "Es wurde nicht zum letzten Datensatz gesprungen"
end if
vStatus = ActiveWindow.DataDeleteRecord
Print "Die Tabelle enthält " & ActiveWindow.DataColumnNumber &
" Spalten"
vStatus = ActiveWindow.CreateQuery("TestQuery",  "SELECT * FROM
TestTab;" , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print "IsQueryNative liefert : " & ActiveWindow.IsQueryNa-
tive("TestQuery")
ActiveWindow.CloseQueryConnection("TestQuery")
vStatus = ActiveWindow.OpenQueryConnection("TestQuery")
stQueryRows = ActiveWindow.GetQueryColumns("TestQuery")
If stQueryRows.Count = 6 then
Print "IsQueryReadOnly liefert : " & ActiveWindow.IsQueryRea-
dOnly("TestQuery")
sSQLStatement = ActiveWindow.GetSQL_Statement("TestQuery")
Print "Die Abfrage verwendet folgendes SQL-Statement : " &
sSQLStatement
End if
ActiveWindow.DeleteQuery "TestQuery"
end if
ActiveWindow.CloseTableConnection("TestTab")
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable("TestTab")
ActiveWindow.CloseDatabaseConnection("TestDB")
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection("TestDB")
end sub
```

DataFirstRecord-Methode

Springt zum ersten Datensatz eines aus der aktuellen Datenbank generierten SQL-Abfrageergebnisses.

KLASSE:
BaseTextEditView

SYNTAX:
DataFirstRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
```

```
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataGetTableName-Methode

Liefert den Namen einer Datenbank-Tabelle.

KLASSE:
BaseTextEditView

SYNTAX:
DataGetTableName(DatabaseName As String, TableNumber As Long)

RÜCKGABE:
String

PARAMETER:
DatabaseName: Name der Datenbank, die im StarOffice angemeldet ist und die die zu erfragende Tabelle enthält.

TableNumber: Nummer der Tabelle, deren Name ermittelt werden soll.

Beispiel:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
```

```
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataLastRecord-Methode

Positionierung auf den letzten Datensatz.

KLASSE:
BaseTextEditView

SYNTAX:
DataLastRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
```

```
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
```

```
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataNextRecord-Methode

Springt zum nächsten Datensatz eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses.

KLASSE:
BaseTextEditView

SYNTAX:
DataNextRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
```

```
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
```

```
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataPhysicalRecordID-Methode

Liefert die aktuelle Datensatznummer des Datensatzes einer Datenbanktabelle. Die zurückgelieferte Datensatznummer ist abhängig von der verwendeten Abfrage. Bei dem ersten Datensatz einer Abfrage wird immer eine 0 zurückgeliefert, auch wenn es nicht der erste Datensatz der Tabelle ist.

KLASSE:
BaseTextEditView

SYNTAX:
DataPhysicalRecordID(GetUserRecordId As Boolean)

RÜCKGABE:
Long

PARAMETER:
[GetUserRecordID]: Dieser Parameter wird nur intern vom Programm ausgewertet und sollte nicht übergeben werden.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
```

```
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataPreviousRecord-Methode

Springt zum vorangehenden Datensatz eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses.

KLASSE:
BaseTextEditView

SYNTAX:
DataPreviousRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
```

```
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataRecord-Methode

Springt zu dem angegebenen Datensatz einer Datenbank-Tabelle oder eines aus der aktiven Datenbank generierten SQL-Abfrageergebnisses. Wurde vor Aufruf dieser Methode eine SQL-Abfrage durchgeführt, dann wird zum angegebenen Datensatz der Abfrage gesprungen. Wurde keine Abfrage ausgeführt, dann wird zum angegebenen Datensatz der Datenbank-Tabelle gesprungen.

KLASSE:
BaseTextEditView

SYNTAX:
DataRecord(RecordNumber As Long)

RÜCKGABE:
Boolean

PARAMETER:
RecordNumber: Ein beliebiger Long-Integer-Ausdruck, der die Nummer des Datensatzes angibt, zu dem gesprungen werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
```

```
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataRecordID-Methode

Gibt die Nummer des aktuellen Datensatzes in dem aus der aktuellen Datenbank generierten SQL-Abfrageergebnis zurück. Wenn im Beamer mehrere Datensätze markiert wurden, hat der erste markierte Eintrag die Nummer 1.

KLASSE:
BaseTextEditView

SYNTAX:
DataRecordID(GetUserRecordId As Boolean)

Rückgabe:
Long

Parameter:
[GetUserRecordID]: Dieser Parameter wird nur intern vom Programm ausgewertet und sollte nicht übergeben werden.

Beispiel:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
```

```
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataSelect-Methode

Führt eine SQL-Abfrage für die geöffnete Datenbank aus. War die Abfrage erfolgreich, wird der Wert True zurückgeliefert, ansonsten False.

KLASSE:
BaseTextEditView

SYNTAX:
DataSelect(Select As String)

RÜCKGABE:
Boolean

PARAMETER:
Select: Ein Zeichenfolgeausdruck, der die SQL-Anweisung aufnimmt.

Beispiel:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
```

```
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataSQLStatement-Methode

Liefert das SQL-Statement einer im Dokument verwendeten Datenbank aus der mit DataUsedDatabaseNumber aufgebauten Liste.

Wenn die Datenbank nur mit einem Tabellennamen geöffnet wurde, kann das SQL-Statement leer sein.

KLASSE:
BaseTextEditView

SYNTAX:
DataSQLStatement(DatabaseNumber As Long)

RÜCKGABE:
String

PARAMETER:
DatabaseNumber: Index innerhalb der mit DataUsedDatabaseNumber aufgebauten Liste.

Rückgabe: SQL-Statement

Beispiel:

```
Sub ExampleDataUsedDatabase
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument ange-
meldet : "
Const sMessage_3 = "Folgende SQL-Abfrage wird am Dokument
verwendet : "
Dim sDataName As String, sTableName As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 to ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
ActiveWindow.DataUpdateFields( sDataName, sTableName, "SELECT *
FROM " & sTableName, 1 )
For iCount=0 To ActiveWindow.DataUsedDatabaseNumber - 1
Print sMessage_2 & ActiveWindow.DataUsedDatabase( iCount )
Print sMessage_3 & ActiveWindow.DataSQLStatement( iCount )
Next iCount
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

DataSwitch-Methode

Schaltet auf eine andere, bereits geöffnete Datenverbindung um. Wurde der Befehl erfolgreich ausgeführt, wird der Wert True zurückgegeben, andernfalls False.

Klasse:
BaseTextEditView

Syntax:
DataSwitch(DBName As String, TableName As String)

Rückgabe:
Boolean

PARAMETER:
DBName: Name der Datenbank, die aktiviert werden soll.

TableName: Name der Tabelle, zu der umgeschaltet werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
```

```
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataTableNumber-Methode

Liefert die Anzahl der Tabellen einer Datenbank.

KLASSE:
BaseTextEditView

SYNTAX:
DataTableNumber(DatabaseName As String)

RÜCKGABE:
Integer

PARAMETER:
DatabaseName: Name der angemeldeten Datenbank, deren Tabellenanzahl ermittelt werden soll.

Beispiel:

```
Sub ExampleDataRecord
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Activewindow.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Activewindow.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print "IsTableReadonly liefert : " &  Activewindow.IsTableRea-
dOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Activewindow.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Activewindow.GetDataba-
seLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Activewindow.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
```

```
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord  REM Spingt zum nächsten
Datensatz
vStatus = Activewindow.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Activewindow.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Activewindow.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Activewindow.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Activewindow.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
     vStatus = Activewindow.DataSwitch(sDataName, sOld-
TableName)
Else
sOldTableName = sTableName
End if
End if
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

DataUpdate-Methode

Aktualisiert die Datensätze der aktiven Datenbank-Tabelle. Dabei wird die aktuelle Verbindung zur Tabelle kurz unterbrochen und anschließend sofort wieder aufgenommen. Dadurch werden ggf. inzwischen vom Benutzer vorgenommene Änderungen an Datensätzen lesbar. Ohne DataUpdate würde man stets nur den Stand bekommen, der beim Öffnen der Tabelle vorlag.

KLASSE:
BaseTextEditView

SYNTAX:
DataUpdate()

BEISPIEL:

```
Sub ExampleDataUpdateRecord
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
```

```
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sError = "Das Ändern des Datensatzes hat nicht geklappt"
Dim stDataRow As New Strings
Dim stChangeRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Documents.Add( "sWriter4" )
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
vStatus = ActiveWindow.DataAppendRecord( sTextName & "1,1," &
sTextName & "2,2" )
vStatus = ActiveWindow.DataAppendRecord( sTextName & "2,2," &
sTextName & "3,3" )
For iCount=1 To 3
stChangeRec.Add sTextName & iCount * 11
stChangeRec.Add iCount * 11
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataLastRecord
vStatus = ActiveWindow.DataUpdateWholeRecord(stColumnsStrings,
stChangeRec)
If ActiveWindow.DataColumn(sTextName & "1") <> sTextName & "11"
Then
Print sError
End if
vStatus = ActiveWindow.DataFirstRecord
vStatus = ActiveWindow.DataUpdateRecord(sTextName & "2",
sTextName & "22")
ActiveWindow.DataUpdate
If ActiveWindow.DataColumn(sTextName & "2") <> sTextName & "22"
Then
Print sError
End if
ActiveWindow.CloseTableConnection(sTabName)
vStatus = ActiveWindow.DeleteTable(sTabName)
```

```
ActiveWindow.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataUpdateFields-Methode

Übernimmt aus einer Datenbank einen oder mehrere angegebene Datensätze in die Datenbankfelder des Dokuments.

Um die Inhalte eines Datensatzes in ein Dokument zu übernehmen, müssen die entsprechenden Datenbankfelder des Datensatzes über den Dialog FELDBEFEHL EINFÜGEN in das Dokument eingefügt worden sein.

KLASSE:
BaseText

SYNTAX:
DataUpdateFields(DBName As String, TableName As String, Statement As String, StartRec As Long)

PARAMETER:
DBName: Zeichenfolgeausdruck, der die Bezeichnung der Datenbank angibt, die die zu übernehmenden Datensätze enthält.

TableName: Zeichenfolgeausdruck, der den Namen der Datenbank-Tabelle angibt, deren Inhalte in die Datenbankfelder des Dokuments eingefügt werden sollen.

Statement: Zeichenfolgeausdruck, der die SQL-Anweisung angibt, die die Datensätze filtert, deren Inhalt den Datenbankfeldern des Dokuments zugewiesen werden sollen. Wird ein Leerstring übergeben, wird automatisch eine SQL-Anweisung generiert, die alle Felder aus der gewählten Tabelle selektiert (SELECT *).

StartRec: Optionaler Long-Integer-Ausdruck, der die Nummer des Datensatzes bestimmt, dessen Inhalt in die Datenbankfelder des Dokuments eingefügt werden soll. Wird dieser Parameter nicht angegeben, wird der erste Datensatz der SQL-Abfrage verwendet.

BEISPIEL:

```
Sub ExampleDataUpdateFields
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument angemeldet : "
Dim sDataName As String, sTableName As String, sSQL As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter")
```

```
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 To ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
sSQL = "SELECT * FROM " & sTableName
REM Es wird der zweite Dateensatz in das Dokument eingemischt
Selection.DataUpdateFields(sDataName, sTableName, sSQL, 2)
REM Alle Datensätze der Tabelle auf dem Standarddrucker
ausdrucken
Selection.DataUpdatePrint(sDataName, sTableName, sSQL, "",
FALSE)
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

DataUpdatePrint-Methode

Druckt Serien des aktuellen Dokuments, das die Datenbankfelder der angegebenen Datenbank enthält. Den Datenbankfeldern wird der Inhalt der Datensätze zugewiesen, die sich aus dem Ergebnis der SQL-Anweisung ergeben, die der Methode übergeben werden muß.

KLASSE:
BaseText

SYNTAX:
DataUpdatePrint(DBName As String, TableName As String, Statement As String, Printer As String, SingleJob As Boolean)

PARAMETER:
DBName: Zeichenfolgeausdruck, der die Bezeichnung der Datenbank aufnimmt, die die zu druckenden Daten enthält.

TableName: Zeichenfolgeausdruck, der den Namen der Datenbank-Tabelle angibt, deren Inhalte im Serienbrief gedruckt werden sollen.

Statement: Zeichenfolgeausdruck, der die SQL-Anweisung aufnimmt, die die Datensätze filtert, deren Inhalt den Datenbankfeldern des Dokuments zugewiesen werden sollen. Wird ein Leerstring übergeben, werden alle Datensätze ausgedruckt.

Printer: Zeichenfolgeausdruck, der den Namen des Druckers angibt. Die Bezeichnung des Druckers entnehmen Sie bitte dem DRUCKEN-Dialog, den Sie über das Menü DATEI aufrufen. Dort sind alle Drucker im Listenfeld DRUCKER aufgeführt. Wird ein Leerstring übergeben, wird der aktuelle Standarddrukker verwendet.

SingleJob: True, es werden alle Serienbriefe in einem einzigen Druckjob zum Drucker geschickt. False, es wird für jeden einzelnen Serienbrief ein eigener Druckjob gebildet.

BEISPIEL:

```
Sub ExampleDataUpdateFields
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument ange-
meldet : "
Dim sDataName As String, sTableName As String, sSQL As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 To ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
sSQL = "SELECT * FROM " & sTableName
REM Es wird der zweite Dateensatz in das Dokument eingemischt
Selection.DataUpdateFields(sDataName, sTableName, sSQL, 2)
REM Alle Datensätze der Tabelle auf dem Standarddrucker
ausdrucken
Selection.DataUpdatePrint(sDataName, sTableName, sSQL, "",
FALSE)
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

DataUpdateRecord-Methode

Ändert Spalten des aktuellen Datensatzes.

Die Änderung wird erst dann in den aktuellen Datensatz übernommen, wenn eine neue SQL-Abfrage ausgeführt wird.

KLASSE:
BaseTextEditView

SYNTAX:
DataUpdateRecord(Columnname As String, Value As String)

RÜCKGABE:
Boolean

PARAMETER:
ColumnName: Zeichenfolgeausdruck, der die Bezeichnung des Datenfeldes im Datensatz angibt, dessen Inhalt geändert werden soll.

Value: Zeichenfolgeausdruck, der den neuen Inhalt des Datenfeldes angibt.

BEISPIEL:

```
Sub ExampleDataUpdateRecord
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sError = "Das Ändern des Datensatzes hat nicht geklappt"
Dim stDataRow As New Strings
Dim stChangeRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Documents.Add( "sWriter4" )
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
```

```
vStatus = ActiveWindow.DataFirstRecord
vStatus = ActiveWindow.DataAppendRecord( sTextName & "1,1," &
sTextName & "2,2" )
vStatus = ActiveWindow.DataAppendRecord( sTextName & "2,2," &
sTextName & "3,3" )
For iCount=1 To 3
stChangeRec.Add sTextName & iCount * 11
stChangeRec.Add iCount * 11
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataLastRecord
vStatus = ActiveWindow.DataUpdateWholeRecord(stColumnsStrings,
stChangeRec)
If ActiveWindow.DataColumn(sTextName & "1") <> sTextName & "11"
Then
Print sError
End if
vStatus = ActiveWindow.DataFirstRecord
vStatus = ActiveWindow.DataUpdateRecord(sTextName & "2",
sTextName & "22")
ActiveWindow.DataUpdate
If ActiveWindow.DataColumn(sTextName & "2") <> sTextName & "22"
Then
Print sError
End if
ActiveWindow.CloseTableConnection(sTabName)
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataUpdateWholeRecord-Methode

Ändert Spalten des aktuellen Datensatzes. Wurde die Änderung erfolgreich durchgeführt, liefert die Methode den Wert True zurück, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
DataUpdateWholeRecord(ColumnNames As Object, Content As Object)

RÜCKGABE:
Boolean

PARAMETER:
ColumnNames: Strings-Variable, die die Spaltennamen enthält.

Content: Strings-Variable, die den Inhalt der Spalten enthält.

Wenn nur ein Parameter übergeben wird, wird der gesamte Datensatz geändert. Wenn beide Parameter übergeben werden, werden nur die Spalten geändert, die in ColumnNames angegeben wurden.

Beispiel:

```
Sub ExampleDataUpdateRecord
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sError = "Das Ändern des Datensatzes hat nicht geklappt"
Dim stDataRow As New Strings
Dim stChangeRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim vStatus As Variant, iCount As Integer
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Documents.Add( "sWriter4" )
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
vStatus = ActiveWindow.DataAppendRecord( sTextName & "1,1," &
sTextName & "2,2" )
vStatus = ActiveWindow.DataAppendRecord( sTextName & "2,2," &
sTextName & "3,3" )
For iCount=1 To 3
stChangeRec.Add sTextName & iCount * 11
stChangeRec.Add iCount * 11
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataLastRecord
vStatus = ActiveWindow.DataUpdateWholeRecord(stColumnsStrings,
stChangeRec)
If ActiveWindow.DataColumn(sTextName & "1") <> sTextName & "11"
Then
Print sError
End if
vStatus = ActiveWindow.DataFirstRecord
```

```
vStatus = ActiveWindow.DataUpdateRecord(sTextName & "2",
sTextName & "22")
ActiveWindow.DataUpdate
If ActiveWindow.DataColumn(sTextName & "2") <> sTextName & "22"
Then
Print sError
End if
ActiveWindow.CloseTableConnection(sTabName)
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DataUsedDatabase-Methode

Liefert den Namen einer im Dokument verwendeten Datenbank aus der mit DataUsedDatabaseNumber aufgebauten Liste als String zurück.

Vor Anwendung dieser Methode muß mit DataUsedDatabaseNumber die Anzahl der am Dokument angemeldeten Datenbank-Tabellen ermittelt werden, sonst liefert DataUsedDatabase einen Leerstring zurück.

Siehe auch: DataUsedDatabaseNumber

KLASSE:
BaseTextEditView

SYNTAX:
DataUsedDatabase(DatabaseNumber As Long)

RÜCKGABE:
String

PARAMETER:
DatabaseNumber: Index innerhalb der mit Hilfe der DataUsedDatabaseNumber aufgebauten Liste.

BEISPIEL:

```
Sub ExampleDataUsedDatabase
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument ange-
meldet : "
Dim sDataName As String, sTableName As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
```

```
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 to ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
For iCount=0 To ActiveWindow.DataUsedDatabaseNumber - 1
Print sMessage_2 & ActiveWindow.DataUsedDatabase(iCount)
Next iCount
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

DataUsedDatabaseNumber-Methode

Liefert die Anzahl der am Dokument verwendeten Datenbanken, zum Beispiel in Feldbefehlen und Bedingungen.

Siehe auch: DataUsedDatabase

KLASSE:
BaseTextEditView

SYNTAX:
DataUsedDatabaseNumber()

RÜCKGABE:
Integer

BEISPIEL:

```
Sub ExampleDataUsedDatabase
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument ange-
meldet : "
Dim sDataName As String, sTableName As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 to ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
```

```
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
For iCount=0 To ActiveWindow.DataUsedDatabaseNumber - 1
Print sMessage_2 & ActiveWindow.DataUsedDatabase(iCount)
Next iCount
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

Deactivate-Methode

Deaktiviert ein OLE-Objekt, wenn es gerade InPlace-aktiv ist. InPlace-aktiv bedeutet hierbei, daß das OLE-Objekt bearbeitbar ist. Ein lediglich selektiertes OLE-Objekt ist nicht InPlace-aktiv.

KLASSE:
BaseTextOLEObject

SYNTAX:
Deactivate()

BEISPIEL:

```
Sub ExampleOLEObjects
Const sMessage = "Der Name des Charts lautet : "
Dim oActwin As Object
Documents.Add( "sWriter4" )
Set oActwin = ActiveWindow REM Merkt sich die View für Deacti-
vate
If ActiveWindow.OLEObjects.Count = 0 Then
ActiveWindow.InsertObjectStarChart()
Else
ActiveWindow.OLEObjects(1).Activate
End If
oActwin.DeActivate REM Hier muß die View des Dokumentes
verwendet werden
Print sMessage & ActiveWindow.OLEObjects(1).OLEObject.Name
End Sub
```

DecrementIndent-Methode

Vermindert den Absatzeinzug um einen festgelegten Wert. Dabei wird der unter EXTRAS - OPTIONEN - TEXTDOKUMENT - LAYOUT als Tabulatorenabstand definierte Standardwert genommen.

KLASSE:
Paragraph

Syntax:
DecrementIndent()

Beispiel:

```
Sub ExampleParagraphIndent
Documents.Add("sWriter4")
Wait 1
With Selection.Paragraph
   Selection.Insert("Einrückung eines Textes erhöhen")
   Wait 500
   .IncrementIndent
   Wait 500
   .IncrementIndent
   Wait 500
   .IncrementIndent
   Wait 500
   .SelectText
   Selection.Insert("Einrückung eines Textes verringern")
   Wait 500
   .DecrementIndent
   Wait 500
   .DecrementIndent
   Wait 500
   .DecrementIndent
   Wait 500
End With
```

DecrementLevel-Methode

Stuft eine Aufzählung oder Numerierung um eine Ebene ab.

Klasse:
BaseTextList

Syntax:
DecrementLevel()

Beispiel:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
```

```
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

DecrementSubLevels-Methode

Stuft eine Aufzählung oder Numerierung einschließlich ihrer Unterpunkte um eine Ebene ab.

KLASSE:
BaseTextList

SYNTAX:
DecrementSubLevels()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
```

```
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

DefaultBullet-Methode

Setzt die Standardeinstellungen für Aufzählungen und schaltet die Aufzählung für den selektierten Text ein.

KLASSE:
BaseText

SYNTAX:
DefaultBullet()

BEISPIEL:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
```

```
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

Delete-Methode

Löscht das Zeichen rechts neben der Schreibmarke oder die Selektion.

KLASSE:
TextBezier, BaseTextSelection, BaseTextDrawBase

SYNTAX:
Delete()

BEISPIEL:

```
Sub ExampleDelete
Documents.Add( "sWriter4" )
Selection.Insert "Dieser Satz wird gelöscht."
Selection.GotoStartOfLine TRUE
Selection.Delete
End Sub
```

DeleteBookmark-Methode

Löscht eine bestimmte Textmarke.

KLASSE:
BaseTextSelection

SYNTAX:
DeleteBookmark(Marke As String)

PARAMETER:
Marke: Zeichenfolgeausdruck, der die Bezeichnung der zu löschenden Textmarke angibt.

Die Bezeichnung der zu löschenden Textmarke muß unter Berücksichtigung der Groß- und Kleinschreibung übergeben werden, wie sie bei der Definition der Textmarke angegeben wurde.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

DeleteColumns-Methode

Löscht die aktuelle Tabellenspalte, in der sich der Cursor befindet.

Die Schreibmarke können Sie mit der GotoCell-Methode in eine bestimmte Tabellenzelle setzen.

Siehe auch: GotoCell

KLASSE:
BaseTextTable

SYNTAX:
DeleteColumns()

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

DeleteForm-Methode

Löscht ein Formular einer im StarOffice angemeldeten Datenbank.

KLASSE:
BaseTextEditView

SYNTAX:
DeleteForm(FormName As String)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Name des Formulars, das gelöscht werden soll.

Beispiel:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Database & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close( True, sPath )
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateForm( sFormName, sPath )
ActiveWindow.CreateReport( sReportName, sPath )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " & sTab-
Name, FALSE )
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
ActiveWindow.DeleteReport sReportName
ActiveWindow.DeleteQuery sQueryName
End Sub
```

DeleteGlossary-Methode

Löscht einen AutoText-Baustein.

Klasse:
BaseText

Syntax:
DeleteGlossary(Group As String, ShortName As String)

PARAMETER:
Group: Name des AutoText-Bereichs, aus dem ein Textbaustein gelöscht werden soll.

ShortName: Kürzel des Textbausteins, der gelöscht werden soll.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
```

```
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

DeleteGlossaryGroup-Methode

Löscht die angegebene AutoText-Gruppe.

KLASSE:
BaseText

SYNTAX:
DeleteGlossaryGroup(Group As String)

PARAMETER:
Group: Name der AutoText-Gruppe, die gelöscht werden soll. Beachten Sie, daß hierbei zwischen Groß-/Kleinschreibung unterschieden wird.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
```

```
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

DeleteIndex-Methode

Löscht einen Index der gerade geöffneten Datenbank-Tabelle.

KLASSE:
BaseTextEditView

SYNTAX:
DeleteIndex(IndexName As String)

RÜCKGABE:
Boolean

PARAMETER:
IndexName: Name des Index, der gelöscht werden soll.

BEISPIEL:

```
Sub ExampleDatabaseIndex
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Dim stDataRow As New Strings
Dim stIndexStrings As New Strings
```

```
Dim sDBPfad As String, sConfigStr As String
Dim sStatus As Integer, iCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
sStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
stIndexStrings.Add sTextName & "1,TRUE"
sStatus = Activewindow.CreateTable(sTabName, stDataRow)
sStatus = Activewindow.OpenTableConnection(sTabName)
sStatus = Activewindow.CreateIndex(sTabName, sIndexName, stIn-
dexStrings, FALSE)
sStatus = Activewindow.DeleteIndex(sIndexName)
Activewindow.CloseTableConnection(sTabName)
sStatus = Activewindow.DeleteTable(sTabName)
Activewindow.CloseDatabaseConnection(sDBName)
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DeleteQuery-Methode

Löscht eine Datenbank-Abfrage einer im StarOffice angemeldeten Datenbank.

KLASSE:
BaseTextEditView

SYNTAX:
DeleteQuery(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage, die gelöscht werden soll.

BEISPIEL:

```
Sub ExampleQueries
Const sQueryName = "TestAbfrage"
Const sMessage = "Der Inhalt des ersten Feldes : "
```

```
Dim sDataName As String, sTabName As String, sTPath As String
Dim vStatus As Variant
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
ActiveWindow.DataFirstRecord
Print sMessage & ActiveWindow.DataColumn(1)
ActiveWindow.CloseQueryConnection(sQueryName)
ActiveWindow.DeleteQuery sQueryName
End Sub
```

DeleteRegion-Methode

Löscht einen Bereich. Dabei wird nur die Zuordung zu einem Bereich aufgehoben und eine evtl. Verknüpfung gelöst. Der Inhalt bleibt erhalten.

KLASSE:
BaseTextSelection

SYNTAX:
DeleteRegion(RegionName As String)

PARAMETER:
RegionName: Zeichenfolgeausdruck, der die Bezeichnung des zu löschenden Bereichs angibt.

BEISPIEL:

```
Sub ExampleGetRegionName
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich "
Const sMessage_3 = " hat folgenden Namen : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
For iCount = 1 to Selection.GetRegionCount
Selection.Insert sMessage_2 & iCount & sMessage_3 &
Selection.GetRegionName(iCount-1)
Next iCount
Selection.DeleteRegion sRegionName
End Sub
```

DeleteReport-Methode

Löscht einen Bericht einer im StarOffice angemeldeten Datenbank.

Siehe auch: CreateReport

KLASSE:
BaseTextEditView

SYNTAX:
DeleteReport(ReportName As String)

RÜCKGABE:
Boolean

PARAMETER:
ReportName: Name des Berichts, der gelöscht werden soll.

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Database & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close( True, sPath )
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateForm( sFormName, sPath )
ActiveWindow.CreateReport( sReportName, sPath )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " & sTab-
Name, FALSE )
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
```

```
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
ActiveWindow.DeleteReport sReportName
ActiveWindow.DeleteQuery sQueryName
End Sub
```

DeleteRows-Methode

Löscht die aktuelle Tabellenzeile, in der sich der Cursor befindet.

KLASSE:
BaseTextTable

SYNTAX:
DeleteRows()

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

DeleteStyle-Methode

Löscht eine angegebene Benutzervorlage. Konnte die Vorlage erfolgreich gelöscht werden, so wird der Wert 1 zurückgegeben, andernfalls der Wert 0.

KLASSE:
BaseTextDocument

SYNTAX:
DeleteStyle(Param As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Param: Zeichenfolge, die den Namen der Vorlage angibt, die gelöscht werden soll.

Family: Long-Integer-Ausdruck, der die Familie der zu löschenden Vorlage bestimmt. Folgende Werte sind möglich:

1 - Zeichenvorlage

2 - Absatzvorlage

4 - Rahmenvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

DeleteTable-Methode

Löscht eine Datenbank-Tabelle einer im StarOffice angemeldeten Datenbank. Die Datenbank muß dabei geöffnet sein.

Diese Methode funktioniert nur dann, wenn keine Verbindung zu der zu löschenden Tabelle besteht. Ggf. ist es notwendig, vorher eine bestehende Verbindung mit der CloseTableConnection-Methode zu schließen.

Siehe auch: OpenDatabaseConnection, CloseTableConnection.

KLASSE:
BaseTextEditView

SYNTAX:
DeleteTable(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Tabelle, die aus der Datenbank gelöscht werden soll.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
```

```
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
```

```
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

DeleteUserField-Methode

Löscht ein Benutzerfeld aus dem Dokument. Wenn das Benutzerfeld nicht vorhanden ist, wird FALSE zurückgegeben. Groß- und Kleinschreibung wird hierbei nicht berücksichtigt.

KLASSE:
BaseTextSelection

SYNTAX:
DeleteUserField(Name As String)

RÜCKGABE:
Boolean

PARAMETER:
Name: Name des Benutzerfeldes

BEISPIEL:

```
Sub ExampleDeleteUserfield
Const UserFieldName = "Benutzerfeld"
Const UserFieldContent = "Inhalt"
Const sMessage = "Das Benutzerfeld wird gelöscht : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.InsertField( 20, 0, UserFieldName, UserFieldContent )
Selection.DeleteUserField( UserFieldName )
End Sub
```

DeleteVariable-Methode

Löscht eine Variable aus dem Dokument. Diese Funktion löscht zuerst alle Feldbefehle dieser Variablen und entfernt dann die Variable selbst aus dem Dokument. Wenn der Variablenname nicht vorhanden ist, wird FALSE zurückgegeben.

KLASSE:
BaseTextSelection

Syntax:
DeleteVariable(Name As String)

Rückgabe:
Boolean

Parameter:
Name: Name der im Dokument definierten Variablen.

Beispiel:

```
Sub ExampleDeleteVariable
Const FieldName = "VariablenFeld"
Const FieldContent = "Inhalt"
Const sMessage = "Der Variablen-Feldbefehl wird gelöscht : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.InsertField( 8, 1, FieldName, FieldContent, 0 )
Selection.DeleteVariable( FieldName )
End Sub
```

DelLine-Methode

Löscht den Text der aktuellen Zeile. Nach Anwendung der Methode bleibt die Zeile als Leerzeile vorhanden.

Klasse:
BaseText

Syntax:
DelLine()

Beispiel:

```
Sub ExampleDelLine
Documents.Add( "sWriter4" )
Selection.Insert "Diese Zeile wird gleich wieder gelöscht."
Selection.DelLine
End Sub
```

DelToEndOfLine-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Ende der Zeile.

Klasse:
BaseText

Syntax:
DelToEndOfLine()

Beispiel:

```
Sub ExampleDelToEndOfLine
Const sBookmark = "Marke"
Const sPara_1 = "Erster Teil der Absatzes. "
Const sPara_2 = "Zweiter Teil der Absatzes."
Const sMessage_1 = "Es wird jetzt der zweite Teil des Absatzes
gelöscht"
Const sMessage_2 = "Es wird jetzt der erste Teil des Absatzes
gelöscht"
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.GoLeft 1, FALSE
Selection.InsertBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.Insert sPara_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToEndOfLine
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToStartOfLine
End Sub
```

DelToEndOfPara-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Ende des Absatzes.

Klasse:
BaseText

Syntax:
DelToEndOfPara()

Beispiel:

```
Sub ExampleDelToEndOfPara
Const sBookmark = "Marke"
Const sPara_1 = "Erster Teil der Absatzes. "
Const sPara_2 = "Zweiter Teil der Absatzes."
Const sMessage_1 = "Es wird jetzt der zweite Teil des Absatzes
gelöscht"
Const sMessage_2 = "Es wird jetzt der erste Teil des Absatzes
gelöscht"
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.GoLeft 1, FALSE
```

```
Selection.InsertBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.Insert sPara_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToEndOfPara
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToStartOfPara
End Sub
```

DelToEndOfSentence-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Ende des Satzes.

KLASSE:
BaseText

SYNTAX:
DelToEndOfSentence()

BEISPIEL:

```
Sub ExampleDeleteSentence
REM Löscht den Satz in dem sich die Einfügemarke befindet
Const sText1="Das ist Satz 1. "
Const sText2="Dieser Satz wird gelöscht. "
Const sText3="Das ist Satz 3. "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert( sText1 & sText2  & sText3)
ActiveWindow.JumpToStartOfDoc
ActiveWindow.GoRight Len(sText1)+5
Wait 2000
With ActiveWindow.Selection
.DelToEndOfSentence
.DelToStartOfSentence
end with
end sub
```

DelToEndOfWord-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Ende des Wortes.

KLASSE:
BaseText

Syntax:
DelToEndOfWord()

Beispiel:

```
Sub ExampleDeleteWord
REM Die Prozedur löscht das Wort in der sich die Einfügemarke
befindet
Const sText1="Dieses "
Const sText2="Wort "
Const sText3="wird gelöscht "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert( sText1 & sText2  & sText3)
ActiveWindow.JumpToStartOfDoc
ActiveWindow.GoRight Len(sText1)+2
Wait 2000
With ActiveWindow.Selection
.DelToEndOfWord
.DelToStartOfWord
end with
end sub
```

DelToStartOfLine-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Anfang der Zeile.

Klasse:
BaseText

Syntax:
DelToStartOfLine()

Beispiel:

```
Sub ExampleDelToEndOfLine
Const sBookmark = "Marke"
Const sPara_1 = "Erster Teil der Absatzes. "
Const sPara_2 = "Zweiter Teil der Absatzes."
Const sMessage_1 = "Es wird jetzt der zweite Teil des Absatzes
gelöscht"
Const sMessage_2 = "Es wird jetzt der erste Teil des Absatzes
gelöscht"
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.GoLeft 1, FALSE
Selection.InsertBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.Insert sPara_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GotoBookmark( sBookmark )
```

```
Selection.GoRight 1, FALSE
Selection.DelToEndOfLine
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToStartOfLine
End Sub
```

DelToStartOfPara-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Anfang des Absatzes.

KLASSE:
BaseText

SYNTAX:
DelToStartOfPara()

BEISPIEL:

```
Sub ExampleDelToEndOfPara
Const sBookmark = "Marke"
Const sPara_1 = "Erster Teil der Absatzes. "
Const sPara_2 = "Zweiter Teil der Absatzes."
Const sMessage_1 = "Es wird jetzt der zweite Teil des Absatzes
gelöscht"
Const sMessage_2 = "Es wird jetzt der erste Teil des Absatzes
gelöscht"
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.GoLeft 1, FALSE
Selection.InsertBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.Insert sPara_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToEndOfPara
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
Selection.GotoBookmark( sBookmark )
Selection.GoRight 1, FALSE
Selection.DelToStartOfPara
End Sub
```

DelToStartOfSentence-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Anfang des Satzes.

KLASSE:
BaseText

SYNTAX:
DelToStartOfSentence()

BEISPIEL:

```
Sub ExampleDeleteSentence
REM Löscht den Satz in dem sich die Einfügemarke befindet
Const sText1="Das ist Satz 1. "
Const sText2="Dieser Satz wird gelöscht. "
Const sText3="Das ist Satz 3. "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert( sText1 & sText2  & sText3)
ActiveWindow.JumpToStartOfDoc
ActiveWindow.GoRight Len(sText1)+5
Wait 2000
With ActiveWindow.Selection
.DelToEndOfSentence
.DelToStartOfSentence
end with
end sub
```

DelToStartOfWord-Methode

Löscht den Text von der aktuellen Position der Schreibmarke bis zum Anfang des Wortes.

KLASSE:
BaseText

SYNTAX:
DelToStartOfWord()

BEISPIEL:

```
Sub ExampleDeleteWord
REM Die Prozedur löscht das Wort in der sich die Einfügemarke
befindet
Const sText1="Dieses "
Const sText2="Wort "
Const sText3="wird gelöscht "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert( sText1 & sText2  & sText3)
ActiveWindow.JumpToStartOfDoc
ActiveWindow.GoRight Len(sText1)+2
```

```
Wait 2000
With ActiveWindow.Selection
.DelToEndOfWord
.DelToStartOfWord
end with
end sub
```

Distance-Eigenschaft

Bestimmt oder ermittelt den Abstand zwischen einer Initiale und dem nachfolgenden Text.

Siehe auch: Count, Lines.

KLASSE:
DropCap

SYNTAX:
Distance

RÜCKGABE:
Long

PARAMETER:
[= TextDistance]: Bestimmt den Abstand der Initiale zum nachfolgenden Text. Der Wert kann auch unter Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExampleDropCap
Documents.Add("sWriter4")
Selection.Insert "Das erstes Zeichen"
Selection.InsertLineBreak
Selection.Insert "wird über drei"
Selection.InsertLineBreak
Selection.Insert "Zeilen dargestellt"
Selection.InsertLineBreak
Selection.GotoStartOfDoc FALSE
Selection.Paragraph.DropCap.Count = 1
Selection.Paragraph.DropCap.Lines = 3
Selection.Paragraph.DropCap.Distance = "0.5cm"
End Sub
```

DistributeColumns-Methode

Verteilt die Spalten einer selektierten Tabelle oder eines selektierten Tabellenbereichs gleichmäßig auf die Breite, die sie zum Zeitpunkt des Aufrufes gemeinsam einnehmen.

KLASSE:
BaseTextTable

SYNTAX:
DistributeColumns()

BEISPIEL:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

DistributeRows-Methode

Paßt die Höhe aller selektierten Zeilen einer Tabelle an die der aktuell höchsten Zeile an. Die eingestellte Höhe ist eine Minimalhöhe.

KLASSE:
BaseTextTable

SYNTAX:
DistributeRows()

BEISPIEL:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
```

```
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

DontExpand-Methode

Schließt eine aufgespannte Zeichenattributierung ab. Die Methode liefert einen Rückgabewert zurück, der angibt, ob die entsprechende Attributierung an der Cursorposition erfolgreich abgeschlossen werden konnte (True) oder nicht (False).

Befindet sich der Cursor am Ende einer aufgespannten Zeichenattributierung, so daß bei Eingabe des nächsten Zeichens auch dieses in die aktuelle Attributierung einbezogen werden würde, so kann mit DontExpand diese Attributierung abgeschlossen werden. Haben Sie beispielsweise mit Insert einen Text eingefügt und dann das Zeichenattribut Fett gesetzt, so erhält ein neu eingefügter Text das Attribut Fett, und zwar solange, bis mittels DontExpand alle Zeichenattribute zurückgesetzt werden. Der Cursor steht dann optisch zwar noch an der gleichen Position, logisch steht er aber hinter der Attributierung. Die Methode liefert den Wert True zurück, wenn der Cursor hinter das Attribut gesetzt werden konnte.

KLASSE:
BaseText

SYNTAX:
DontExpand()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleDontExpand
Const sMessage_1 = "Dieser Satz wird fett geschrieben. "
Const sMessage_2 = "Dieser wieder normal."
Documents.Add( "sWriter4" )
Selection.Insert " "
Selection.Font.Bold = soWeightBold
Selection.Insert sMessage_1
Selection.DontExpand
Selection.Insert sMessage_2
End Sub
```

DoUndo-Methode

Macht die letzte Aktion rückgängig.

KLASSE:
BaseTextSelection

SYNTAX:
DoUndo()

BEISPIEL:

```
Sub ExampleDoUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sMessage_5 = "RedoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte keine Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.DoUndo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
Print sMessage_5 & ActiveWindow.Selection.RedoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.DoUndo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

DrawFont-Eigenschaft

DrawFont ist das Interface für die Zeichenattribute für Zeichnungsobjekte.

KLASSE:
TextDrawText

SYNTAX:
DrawFont

RÜCKGABE:
Object

BEISPIEL:

```
REM Muß ein bereits definiertes Zeichen-Objekt sein.
print Selection.DrawFont.Size
```

Drawing-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für Zeichnungselemente.

KLASSE:
ViewSettings

SYNTAX:
Drawing

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Zeichnungselemente angezeigt werden (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption  "
Const sMessage_2 = "  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

DropCap-Eigenschaft

DropCap ist das Objekt für die Initialen.

(Integer) Lines

(Integer) Count

(LONG) Distance

KLASSE:
Paragraph

SYNTAX:
DropCap

RÜCKGABE:
Object

PARAMETER:
Lines - Anzahl der Zeilen für die Initiale

Count - Anzahl der Initialenzeichen

Distance - Abstand zum Text

BEISPIEL:

```
Sub ExampleDropCap
Documents.Add("sWriter4")
Selection.Insert "Das erste Zeichen"
Selection.InsertLineBreak
```

```
Selection.Insert "wird über drei"
Selection.InsertLineBreak
Selection.Insert "Zeilen dargestellt"
Selection.InsertLineBreak
Selection.GotoStartOfDoc FALSE
Selection.Paragraph.DropCap.Count = 1
Selection.Paragraph.DropCap.Lines = 3
Selection.Paragraph.DropCap.Distance = "0.5cm"
End Sub
```

Eigenschaften und Methoden mit "E"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "E" beginnen.

EditGlossary-Methode

Ruft den Dialog AutoText auf.

Klasse:
BaseText

Syntax:
EditGlossary()

Beispiel:

```
Sub ExampleEditGlossary
Documents.Add "sWriter4"
Selection.Insert("Hier können Sie Textbausteine definieren: ")
Selection.EditGlossary()
End Sub
```

EditRegion-Methode

Setzt die Einstellungen für einen bestimmten Bereich.

Klasse:
BaseTextSelection

Syntax:
EditRegion(RegionName As String, RegionCondition As String, RegionHidden As Boolean, RegionProtect As Boolean)

Parameter:
RegionName: Zeichenfolgeausdruck, der die Bezeichnung des Bereiches aufnimmt, dessen Einstellungen bearbeitet werden sollen.

RegionCondition: Zeichenfolgeausdruck, der die Bedingung für die Anzeige des gewählten Bereiches aufnimmt.

RegionHidden: Boolescher Ausdruck, der angibt, ob der gewählte Bereich aus- (True) oder eingeblendet (False) werden soll.

RegionProtect: Boolescher Ausdruck, der angibt, ob der Text im gewählten Bereich geändert werden darf (False) oder nicht (True).

BEISPIEL:

```
Sub ExampleEditRegion
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  ist geschützt"
Const sMessage_4 = "REM  ist nicht geschützt"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sRegionName, "False", FALSE, FALSE)
Selection.Escape
Selection.EditRegion( sRegionName, "True", FALSE, TRUE)
If Selection.IsRegionProtected(sRegionName) = TRUE Then
Selection.Insert sMessage_2 & sRegionName & sMessage_3
Else
Selection.Insert sMessage_2 & sRegionName & sMessage_4
End If
End Sub
```

EditStyle-Methode

Ruft den Dialog zur Bearbeitung einer Vorlage auf. Die zu bearbeitende Vorlage wird durch die beiden Übergabeparameter bestimmt. Werden keine Parameter übergeben, wird der Dialog für die aktuelle Absatzvorlage aufgerufen.

KLASSE:
BaseTextDocument

SYNTAX:
EditStyle(Param As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Param: Zeichenfolge, die den Namen der Vorlage angibt, die bearbeitet werden soll.

Family: Long-Integer-Ausdruck, der die Familie der zu bearbeitenden Vorlage angibt. Folgende Werte sind möglich:

1 - Zeichenvorlage

2 - Absatzvorlage

4 - Rahmenvorlage

8 - Seitenvorlage

Beispiel:

```
Sub ExampleEditStyle
Const sTemp="Standard"
Documents.Add "sWriter4"
Selection.Insert("Hier können Sie die Vorlage " & sTemp & "
ändern.")
ActiveDocument.EditStyle(sTemp,2)
End Sub
```

EntireColumn-Methode

Selektiert die Tabellenspalte, in der sich die Schreibmarke befindet.

Klasse:
BaseTextTable

Syntax:
EntireColumn()

Beispiel:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
```

```
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

EntireRow-Methode

Selektiert die Tabellenzeile, in der sich die Schreibmarke befindet.

KLASSE:
BaseTextTable

SYNTAX:
EntireRow()

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

Escape-Methode

Löst ein Escape-Ereignis aus, wodurch der aktuelle Vorgang abgebrochen oder die aktuelle Selektion aufgehoben wird.

KLASSE:
TextBezier, TextDrawText, BaseTextSelection, BaseTextEditView

SYNTAX:
Escape()

BEISPIEL:

```
Sub ExampleEscape
Documents.Add "sWriter4"
REM Fügt einen seitengebundenen Rahmen ein
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
REM Hebt die Selektion am Rahmen auf
Selection.Escape
End Sub
```

Escapement-Eigenschaft

Bestimmt oder ermittelt die Hoch- oder Tiefstellung eines selektierten Textes prozentual zur Zeilenhöhe. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetEscapement gesetzte Zeichenposition abgefragt.

Selektierter Text wird mit der SuperScript- bzw. mit der SubScript-Eigenschaft höher- oder tiefergestellt.

Eine Abfrage der Escapement-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsEscapement bestimmt wurde, daß nach einer bestimmten Zeichenposition gesucht werden soll, d. h. IsEscapement = 2.

Siehe auch: SuperScript, SubScript, SetEscapement, IsEscapement, RelativeSize.

KLASSE:
ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Escapement

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Value]: Integer-Ausdruck, der die Prozentzahl aufnimmt, um die der selektierte Text prozentual zur Zeilenhöhe hoch- oder tiefgestellt werden soll.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
```

```
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
```

```
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Exchange-Methode

Wendet in der aktuellen Aufzählung eine benannte Numerierung an.

KLASSE:
BaseTextList

SYNTAX:
Exchange(Name As String)

RÜCKGABE:
Boolean

PARAMETER:
Name: Zeichenfolge, die den Namen der benannten Numerierung angibt.

BEISPIEL:

```
REM Benannte Numerierung REM TestREM  muß bereits existieren
Selection.Exchange("Test")
```

ExecHyperlinks-Eigenschaft

Bestimmt oder emittelt, ob der Modus "Hyperlinks ausführen" oder der Modus "Hyperlinks bearbeiten" aktiv ist.

KLASSE:
BaseTextEditView

SYNTAX:
ExecHyperlinks

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der den aktuellen Modus angibt: "Hyperlinks ausführen" (True) oder "Hyperlinks bearbeiten" (False).

BEISPIEL:

```
Sub ExampleExecHyperlinks
Documents.Add "sWriter4"
If ActiveWindow.ExecHyperlinks = TRUE Then
Print "Der Modus Hyperlinks ausführen ist aktiv"
Else
```

```
Print "Der Modus Hyperlinks bearbeiten ist aktiv"
End If
End Sub
```

ExpandGlossary-Methode

Fügt an der aktuellen Cursorposition einen AutoText-Eintrag in das Dokument ein.

Siehe auch: InsertGlossary, NewGlossary, SetActGlossaryGroup.

KLASSE:
BaseText

SYNTAX:
ExpandGlossary(ShortName As String)

RÜCKGABE:
Boolean

PARAMETER:
[ShortName]: Zeichenfolgeausdruck, der die Kurzbezeichnung des AutoText-Eintrags angibt. Dieser Parameter ist optional. Wird kein Parameter übergeben, wird die aktuelle Selektion oder das Wort, in dem der Cursor steht, als Kurzbezeichnung ausgewählt und durch den entsprechenden AutoText-Eintrag ersetzt. Existiert zu einem Wort, in dem der Cursor bei Aufruf dieser Methode steht, kein zugeordneter AutoText, dann wird es lediglich selektiert.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
```

```
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

Eigenschaften und Methoden mit "F"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "F" beginnen.

Family-Eigenschaft

Bestimmt oder ermittelt die Schriftfamilie des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetFont gesetzte Schriftfamilie abgefragt.

Siehe auch: SetFont, IsFont, StyleName, Pitch, CharSet, FamilyName.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
Family

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Text-Objekten:

[= Group]: Integer-Ausdruck, der die Schriftfamilie kennzeichnet, die der Schriftart zugewiesen werden kann:

0 - Voreinstellung, unbekannte Schriftartgruppe 1 - Decorative 2 - Modern 3 - Roman

4 - Script

5 - Swiss

6 - System

Wird eine Schriftart gewählt, die nicht vorhanden ist, dann wird eine adäquate Schriftart aus der angegebenen Schriftfamilie verwendet.

BEISPIEL:

```
Sub ExampleFamily
Documents.Add "sWriter4"
Print "Family liefert : " & Selection.Font.Family
End Sub
```

FamilyName-Eigenschaft

Bestimmt oder ermittelt den Namen der Schrift des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetFont gesetzte Name der Font-Familie erfragt, wenn die Eigenschaft IsFont den Wert 2 liefert.

Siehe auch: SetFont, IsFont, StyleName, Pitch, CharSet, Family.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, Text-Font

SYNTAX:
FamilyName

RÜCKGABE:
String

PARAMETER:
Bei Text-Objekten:

[= String]: Zeichenfolge, die den Namen der Schriftfamilie angibt, z. B. "Arial".

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
```

```
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
```

```
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

FaxName-Eigenschaft

Bestimmt oder ermittelt den Treiber für das Versenden von Fax-Nachrichten. Der Treibername entspricht dem vom Fax-Programm angemeldeten Fax-Druckertreiber.

KLASSE:
PrintSettings

SYNTAX:
FaxName

RÜCKGABE:
String

PARAMETER:
[= Name]: Zeichenfolgeausdruck, der die Bezeichnung des Treibers für das Versenden von Fax-Nachrichten angibt.

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
```

```
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

Field-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für farbig hinterlegte Feldfunktionen.

KLASSE:
ViewSettings

SYNTAX:
Field

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Feldfunktionen im Dokument farbig hinterlegt angezeigt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption  "
Const sMessage_2 = "  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

FieldCount-Methode

Ermittelt die am Dokument definierte Anzahl des angegebenen Feldtyps.

Diese Methode ist nur bei solchen Feldtypen sinnvoll, die unterschiedlich definiert sein können. Das sind die Feldtypen Variable setzen, Referenz setzen, DDE-Feld, Benutzerdefiniertes Feld und Nummernkreis. Alle anderen Feldtypen sind nur einmal am Dokument definiert, so daß FieldCount hier lediglich den Wert 1 zurückliefert.

Die Feldtypen Variable setzen und Nummernkreis liefern mindestens den Wert 3 zurück, da es für diese Feldtypen standardmäßig drei Definitionen am Dokument gibt: Abbildung, Tabelle und Text. Wurden die Feldtypen DDE-Feld und Benutzerdefiniertes Feld vom Benutzer nicht speziell definiert, so ermittelt FieldCount hier den Wert 0.

Siehe auch: GetFieldName, GetFieldTypeCount, GetFieldTypeID, GetFieldTypeName.

KLASSE:
BaseTextEditView

SYNTAX:
FieldCount(FieldType As Long)

RÜCKGABE:
Long

PARAMETER:
FieldType: Long-Integer-Ausdruck, der den Feldtyp angibt, von dem die Anzahl der Definitionen ermittelt werden soll. Folgende Feldtypen liefern unterschiedliche Werte zurück:

8: Variable setzen

12: Referenz setzen

14: DDE-Feld

20: Benutzerdefiniertes Feld

23: Nummernkreis

Folgende Feldtypen sind nur einmal am Dokument definiert:

0: Datum

1: Uhrzeit

2: Dateiname

3: Datenbankname

4: Kapitel

5: Seitennummer

6: Statistik

7: Autor

9: Variable anzeigen

10: Formel einfügen

11: Versteckter Text

13: Referenz einfügen

15: Makro ausführen

16: Eingabefeld

17: Versteckter Absatz

18: Dokumentinfo-Bereich

19: Serienbrieffeld

21: Notizfeld

22: Dokumentvorlage

24: Nächster Datensatz

25: Beliebiger Datensatz

26: Datensatznummer

27: Bedingter Text

28: Nächste Seite

29: Vorherige Seite

30: Absender

35: Seitenvariable setzen

36: Seitenvariable anzeigen

38: Platzhalter

39: Scriptfeld

Beispiel:

```
Sub ExampleUserFieldCount
Const sMessage_1 = "Erstes Benutzerfeld : "
Const sMessage_2 = "Zweites Benutzerfeld : "
Const sMessage_3 = "Anzahl der Benutzerfelder auf dem Dokument
: "
Const sMessage_4 = "Der Inhalt vom ersten Benutzerfeld : "
Const sUserName_1 = "Benutzerfeld_1"
Const sUserName_2 = "Benutzerfeld_2"
Const sUserContent_1 = "Benutzerfeld 1"
Const sUserContent_2 = "Benutzerfeld 2"
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_1, sUserCon-
tent_1, 0 )
Selection.InsertPara
Selection.Insert( sMessage_2 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_2, sUserCon-
tent_2, 0 )
Selection.InsertPara
Selection.Insert sMessage_4
Selection.Insert Selection.GetUserFieldValue( sUserName_1 )
Selection.InsertPara
iCount = ActiveWindow.FieldCount( TYP_USERFLD )
MsgBox sMessage_3 & iCount, 64, ""
End Sub
```

FieldDialog-Methode

Ruft den Dialog Feldbefehl bearbeiten auf.

Bevor Sie diese Methode aufrufen, müssen Sie zunächst ein Feld im Dokument selektieren.

Klasse:
BaseText

Syntax:
FieldDialog()

Beispiel:

```
Sub ExampleFileDialog
Documents.Add("swriter4")
with ActiveWindow.Selection
.Insert("Die Uhrzeit ist ")
.InsertTimeField()
.InsertPara
.Insert("Das Datum ist ")
.InsertDateField()
```

```
.GoLeft 1
.FieldDialog
end with
end sub
```

FieldName-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für Feldbefehle im Dokument.

KLASSE:
ViewSettings

SYNTAX:
FieldName

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Namen (True) oder die Inhalte (False) der im Dokument eingefügten Feldbefehle angezeigt werden sollen.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

FieldPrompt-Eigenschaft

Mit dieser Eigenschaft kann der Hinweistext von Eingabefeldern, die ihren Inhalt aus Variablen beziehen, ausgelesen und gesetzt werden. Voraussetzung ist allerdings, daß der Cursor vor dem Feldbefehl steht.

KLASSE:
BaseText

SYNTAX:
FieldPrompt

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleFieldPrompt
Const sFieldContent = "Inhalt"
Const sFieldName = "Eingabefeld"
Const sNewPrompt = "Neuer Inhalt"
Documents.Add( "sWriter4" )
Selection.InsertField( 8, 1, sFieldName, sFieldName, 0 )
Selection.InsertPara
Selection.InsertField( 16, 3, sFieldName, sFieldContent, 0 )
Selection.GoLeft( 1, FALSE )
Selection.FieldPrompt = sNewPrompt
End Sub
```

FillColor-Eigenschaft

Bestimmt oder ermittelt die Farbe des Hintergrundes des selektierten Bereiches.

KLASSE:
TextPage, Shadow, BaseTextFrame, BaseTextEditView

SYNTAX:
FillColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Farbe des Hintergrundes angibt.

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

Flash-Eigenschaft

Bestimmt, ob der selektierte Text mit dem Attribut Blinkend versehen werden soll, oder ermittelt, ob das Attribut für den selektierten Text gesetzt ist.

KLASSE:
TextFont

SYNTAX:
Flash

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Attribut Blinkend gesetzt ist (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
```

```
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
```

```
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Font-Eigenschaft

Font ist das Interface für die Einstellungen von Zeichenattributten.

KLASSE:
BaseText

SYNTAX:
Font

RÜCKGABE:
Object

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
```

```
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

FontColor-Eigenschaft

Ermittelt die Farbe, die für das Suchen und Ersetzen von Attributen mittels SetFontColor als Fontfarbe gesetzt ist, wenn IsFontColor den Wert 2 liefert.

Siehe auch: SetFontColor, IsFontColor.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
FontColor

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Const sMessage_1 = "Es wird ein Text, der mit der Farbe Rot
formatiert ist, gesucht"
Const sMessage_2 = "Bei diesem Absatz hat die Schrift die Farbe
Rot"
Sub ExampleFontColor
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.SetFontColor( RGB( 255, 0, 0 ) )
If ActiveWindow.SearchAttributes.FontColor = RGB( 255, 0, 0 )
Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

FontDialog-Methode

Ruft den Dialog ZEICHEN zum Setzen der Zeichenattribute des selektierten Bereiches auf.

KLASSE:
TextDrawText, TextFont

SYNTAX:
FontDialog()

Beispiel:

```
Sub ExampleFontDialog
Documents.Add "sWriter4"
Selection.Font.FontDialog
End Sub
```

FontWork-Eigenschaft

Bestimmt oder ermittelt, ob FontWork aktiv ist.

Klasse:
TextDraw, TextDrawText

Syntax:
FontWork

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob FontWork ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
REM Muß ein bereits definiertes Zeichen-Objekt sein.
Selection.Fontwork = True
```

FootnoteBackground-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für farbig hinterlegte Fußnoten.

Klasse:
ViewSettings

Syntax:
FootnoteBackground

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Fußnoten im Dokument farbig hinterlegt angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption  "
```

```
Const sMessage_2 = "  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

FootnoteDialog-Methode

Ruft das Register Fußzeile auf.

Klasse:
Text, TextEditView

Syntax:
FootnoteDialog()

Beispiel:

```
Sub ExampleFootnoteDialog
Const sTemp="Standard"
Documents.Add "sWriter4"
Selection.Insert("So können Sie Fußnoten definieren.")
ActiveWindow.FootnoteDialog
End Sub
```

ForeColor-Eigenschaft

Bestimmt oder ermittelt die Vordergrundfarbe eines Kontrollfeldes.

KLASSE:
BaseTextDrawControl

SYNTAX:
ForeColor

RÜCKGABE:
Long

PARAMETER:
[= Value]: Numerischer Ausdruck, der die Vordergrundfarbe bestimmt.

BEISPIEL:

```
Sub ExampleForeColor
Const sCaption = "Dieses Textfeld ist schön farbig"
Documents.Add( "sWriter4" )
Activewindow.InsertVCControl( 8, "5cm", "4cm", "7cm", "3cm")
ActiveWindow.ControlObject.ForeColor = RGB(255,0,0)
ActiveWindow.ControlObject.BackColor = RGB(0,0,255)
ActiveWindow.ControlObject.Text = sCaption
End Sub
```

FormatArea-Methode

Öffnet den Dialog FLÄCHE für das selektierte Zeichenobjekt.

KLASSE:
TextDraw

SYNTAX:
FormatArea()

BEISPIEL:

```
REM Ein Zeichenobjekt muß bereits vorhanden sein.
Selection.FormatArea
```

FormatColumnValue-Methode

Formatiert einen beliebigen String anhand der Formatierungsregeln der angegebenen Spalte einer Datenbank-Tabelle. Diese muß zuvor mit der OpenTableConnection-Methode geöffnet worden sein.

Siehe auch: OpenTableConnection

KLASSE:
BaseTextEditView

SYNTAX:
FormatColumnValue(Value As String, Column As String, Format As Boolean)

RÜCKGABE:
String

PARAMETER:
Value: Beliebige Zeichenfolge, die den Inhalt des Datenbankfeldes angibt, der formatiert werden soll.

Column: Zeichenfolge, die die Spalte angibt, deren Formatierungsregeln angewendet werden sollen. Die Spalte kann entweder durch ihre Bezeichnung oder durch ihre Nummer angegeben werden. Die Spaltennummer kann sowohl als Zeichenfolge als auch als Integer-Ausdruck übergeben werden.

Format: Optionaler Parameter, der bestimmt, ob eine Formatierung durchgeführt werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, so ist er automatisch auf True gesetzt. Wenn Format = False ist, wird der String ohne Formatierung zurückgegeben.

BEISPIEL:

```
Sub ExampleFormatColumnValue
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sIndexName = "Index"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sMessage = "Die Zahl REM 1REM  sieht jetzt so aus : "
Dim stDataRow As New Strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As String, sConfigStr As String
Dim sStatus As Integer, iCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
sStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
```

```
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,SINGLE,FALSE"
Next iCount
sStatus = Activewindow.CreateTable(sTabName, stDataRow)
sStatus = Activewindow.OpenTableConnection(sTabName)
ActiveWindow.DataFirstRecord
Activewindow.DataAppendRecord("Test_1,1,Test_2,2")
Activewindow.DataUpdate
Print sMessage & Activewindow.FormatColumnValue(1 , sNume-
ricName & "1", TRUE )
Activewindow.CloseTableConnection( sTabName )
sStatus = Activewindow.DeleteTable( sTabName )
Activewindow.CloseDatabaseConnection( sDBName )
Application.DeleteDatabaseConnection( sDBName )
End Sub
```

FormatGroup-Methode

Gruppiert die selektierten Zeichnungselemente oder Kontrollfelder.

KLASSE:
BaseTextDrawBase

SYNTAX:
FormatGroup()

BEISPIEL:

```
REM Dieses Makro geht davon aus, daß bereits eine Gruppe
REM von Zeichenobjekten selektiert ist.
Sub ExampleFormatGroup
Selection.FormatGroup
Wait 100
Selection.FormatUnGroup
End Sub
```

FormatLine-Methode

Öffnet den Dialog LINIE für das selektierte Zeichenobjekt.

KLASSE:
TextDraw

SYNTAX:
FormatLine()

BEISPIEL:

```
REM Ein Zeichenobjekt muß bereits vorhanden sein.
Selection.Formatline
```

FormatUngroup-Methode

Hebt eine Gruppierung von Zeichnungselementen oder Kontrollfeldern wieder auf.

KLASSE:
BaseTextDrawBase

SYNTAX:
FormatUngroup()

BEISPIEL:

```
REM Dieses Makro geht davon aus, daß bereits eine Gruppe
REM von Zeichenobjekten selektiert ist.
Sub ExampleFormatGroup
Selection.FormatGroup
Wait 100
Selection.FormatUnGroup
End Sub
```

Formula-Eigenschaft

Setzt oder ermittelt eine Rechenformel in einer Tabellenzelle.

KLASSE:
BaseTextTable

SYNTAX:
Formula

RÜCKGABE:
String

PARAMETER:
[= Formel]: Zeichenfolge, die die Rechenformel angibt. Das Setzen der Formel erfolgt ohne vorangestelltes Gleichheitszeichen ("=").

BEISPIEL:

```
Sub ExampleTableCalc
Const sMessage_1 = "Die Formel aus der Zelle B2 berechnet : "
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoCell("A2", FALSE)
Selection.Insert "2"
Selection.GotoCell("A3", FALSE)
Selection.Insert "3"
Selection.GotoCell("B2", FALSE)
Selection.Formula = "<A2>+<A3>"
Selection.GotoCell("A2", FALSE)
Selection.Insert "4"
```

```
Selection.Calc
Selection.GotoCell("B2", FALSE)
Print sMessage_1 & Selection.LineText
End Sub
```

FrameColumns-Methode

Bestimmt die Einstellung der Spalten einschließlich der Spaltentrennlinien eines Textrahmens.

KLASSE:
BaseTextFrame

SYNTAX:
FrameColumns(Count As Long, GutterWidth As Long, LineAdj As Long, LineHeight As Long, PenStyle As Long)

PARAMETER:
Count: Gibt die Anzahl der Spalten an.

GutterWidth: Gibt den Abstand zwischen den Spalten an. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

LineAdj: Bestimmt die Ausrichtung der Trennlinie. Folgende Werte sind möglich:

0 - Keine

1 - Oben

2 - Mitte

3 - Unten

LineHeight: Gibt die Höhe der Trennlinie in Prozent an.

PenStyle: Gibt die Linienbreite in Pixel an.

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
```

```
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

FrameDialog-Methode

Ruft bei einem selektierten Objekt den entsprechenden Dialog des jeweiligen Objekts auf, in dem spezielle Einstellungen vorgenommen werden können. Je nach selektiertem Objekt wird entweder der Dialog OBJEKT (TextOLEObject), der Dialog GRAFIK (TextGraphic) oder der Dialog RAHMEN (TextFrame) aufgerufen.

KLASSE:
BaseTextFrame

SYNTAX:
FrameDialog()

BEISPIEL:

```
Sub ExampleFrameDialog
Dim sFrmName as String
Documents.Add("sWriter4")
Selection.InsertFrame(0,,,"4cm","5cm")
sFrmName = Selection.FrameName
Selection.AlignHorizontalCenter
Selection.FrameDialog
Selection.GotoAnchor
End Sub
```

FrameDistance-Methode

Bestimmt die Abstände des selektierten Objekts zu seiner Umgebung und legt somit den Umlauf des Objekts fest. Diese Methode gilt für OLE-Objekte, Rahmen oder Grafiken. Bei der FrameDistance-Methode wird das Objekt in ein fiktives Rechteck eingebettet, welches die Begrenzung für den Umlauf definiert und dessen Größe und Position durch die anzugebenden Parameter festgelegt werden.

KLASSE:
BaseTextFrame

SYNTAX:
FrameDistance(Left As Long, Top As Long, Width As Long, Height As Long)

PARAMETER:
Left: Abstand zwischen dem linken Seitenrand und dem rechten Rand des Textes, der das Objekt an der linken Seite umläuft. (Ist der im Register UMLAUF eingestellte Abstand LINKS 0, so gibt Left den Abstand zwischen dem linken Seitenrand und dem linken Rand des Objekts an.)

Top: Abstand zwischen dem oberen Seitenrand und der letzten Zeile des Textes, der das Objekt an der oberen Seite umläuft. (Ist der im Register UMLAUF eingestellte Abstand OBEN 0, so gibt Top den Abstand zwischen dem oberen Seitenrand und dem oberen Rand des Objekts an.)

Width: Gibt den Abstand zwischen dem rechten Rand des linken Umlauftextes und dem linken Rand des rechten Umlauftextes an. Dies entspricht der Breite des Begrenzungsrechtecks, wodurch der Umlauf an der linken und rechten Seite des Objekts bestimmt wird.

Height: Gibt den Abstand zwischen der letzten Zeile des oberen Umlauftextes und der ersten Zeile des unteren Umlauftextes an. Dies entspricht der Höhe des Begrenzungsrechtecks, wodurch der Umlauf an der oberen und unteren Seite des Objekts bestimmt wird.

Als Maßeinheit wird stets die unter EXTRAS - OPTIONEN - TEXTDOKUMENT - LAYOUT eingestellte Maßeinheit genommen. Die Parameter können auch als Zeichenfolge mit Angabe der gewünschten Maßeinheit übergeben werden.

BEISPIEL:

```
Sub ExampleDistance
Const sMessage_1 = "Dies ist ein Absatz um zu zeigen wie Frame-
Distance "
Const sMessage_2 = "sich im Dokument auswirkt."
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.Insert sMessage_2
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.FrameDistance(1136, 1136, 1702, 1702)
End Sub
```

FrameName-Eigenschaft

Bestimmt oder ermittelt den Namen des selektierten Objekts.

KLASSE:
BaseTextFrame

SYNTAX:
FrameName

RÜCKGABE:
String

PARAMETER:
[Name]: Zeichenfolge, die den Namen des Objekts angibt. Der zu setzende Name darf nicht bereits an einen anderen Rahmen, eine Grafik oder ein OLE-Objekt vergeben worden sein.

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

Frames-Eigenschaft

Ermöglicht den Zugriff auf die Collection für Rahmen.

Siehe auch: Tables und Collections

KLASSE:
BaseTextEditView

SYNTAX:
Frames

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

FramesetSource-Eigenschaft

Schaltet die Ansicht "HTML-Quelltext" ein.

KLASSE:
WebDocument

SYNTAX:
FramesetSource

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleCutCopyPaste
Dim sWin As String REM für den  Fensternamen
   REM TextDokument öffnen
Documents.Add("sWriter4")
sWin = ActiveWindow.Name
REM jetzt wird ein wenig HTML-Code eingegeben
ActiveWindow.Selection.Insert(Chr(60)&"HTML"&Chr(62)&Chr(60)&"B
ODY"&Chr(62)&"This is HTML !"&Chr(60)&"/BODY"&Chr(62))
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut REM HTML-Code ausschneiden
Documents.Add("sWriter4/Web") REM HTML-Dokument öffnen und ...
Wait 100
ActiveDocument.FrameSetSource = True REM ... in die Quelltex-
tansicht ...
ActiveWindow.SelectAll
ActiveWindow.Paste REM .. den HTML-Code einfügen.
ActiveDocument.FrameSetSource = False
REM und zurück ...
ActiveDocument.FrameSetSource = TRUE
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveDocument.FrameSetSource = False
Windows( sWin ).Activate
ActiveWindow.Paste
end sub
```

FrameStyle-Eigenschaft

Ermittelt die an der Selektion gültige Rahmenvorlage.

Diese Eigenschaft gilt nur für Objekte, denen eine Rahmenvorlage zugewiesen werden kann. Das sind Rahmen, Grafiken und OLE-Objekte.

KLASSE:
BaseTextSelection

SYNTAX:
FrameStyle

RÜCKGABE:
String

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExamplesFrameStyle
Const sFrameStyle = "OLE"
Const sMessage = "Es wurde die Rahmenvorlage nicht zugewiesen"
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
REM Rahmenvorlage setzen
Selection.StyleApply(sFrameStyle, 4)
If Selection.FrameStyle <> sFrameStyle Then
Print sMessage
End If
End Sub
```

FtnAdjust-Eigenschaft

Setzt oder ermittelt die Ausrichtung der Trennlinie für den Fußnotenbereich an der aktuellen Seitenvorlage.

Klasse:
BaseTextEditView

Syntax:
FtnAdjust

Rückgabe:
Integer

Parameter:
[= Ausrichtung]: Integer-Ausdruck, der die Ausrichtung der Trennlinie angibt. Folgende Werte sind möglich:

0 - links

1 - Mitte

2 - rechts

Beispiel:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
```

```
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnBottomDistance-Eigenschaft

Setzt oder ermittelt den Abstand zwischen dem Textbereich und der ersten Fußnote im Fußnotenbereich an der aktuellen Seitenvorlage.

KLASSE:
BaseTextEditView

SYNTAX:
FtnBottomDistance

RÜCKGABE:
Long

PARAMETER:
[= Distance]: Long-Integer-Ausdruck, der den Abstand zwischen Text und Fußnote angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnHeight-Eigenschaft

Setzt oder ermittelt die maximale Höhe des Fußnotenbereichs an der aktuellen Seitenvorlage.

Klasse:
BaseTextEditView

Syntax:
FtnHeight

Rückgabe:
Long

Parameter:
[= Height]: Long-Integer-Ausdruck, der die maximale Höhe des Fußnotenbereichs angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
```

```
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnPenStyle-Eigenschaft

Setzt oder ermittelt die Art der Trennlinie für den Fußnotenbereich an der aktuellen Seitenvorlage.

KLASSE:
BaseTextEditView

SYNTAX:
FtnPenStyle

RÜCKGABE:
Integer

PARAMETER:
[= Style]: Integer-Ausdruck, der die Art der Trennlinie bestimmt. Folgende Werte sind möglich:

0 - keine Linie

1 - durchgezogene Linie

2 - Punktline

3 - Strichlinie

4 - Strichpunktlinie

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
```

```
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnPenWidth-Eigenschaft

Setzt oder ermittelt die Breite der Trennlinie für den Fußnotenbereich an der aktuellen Seitenvorlage.

KLASSE:
BaseTextEditView

SYNTAX:
FtnPenWidth

RÜCKGABE:
Long

PARAMETER:
[= Width]: Long-Integer-Ausdruck, der die Breite der Trennlinie angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
```

```
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnTopDistance-Eigenschaft

Setzt oder ermittelt den Abstand zwischen dem Textbereich der Seite und der Trennlinie für den Fußnotenbereich an der aktuellen Seitenvorlage.

KLASSE:
BaseTextEditView

SYNTAX:
FtnTopDistance

RÜCKGABE:
Long

PARAMETER:
[= Distance]: Long-Integer-Ausdruck, der den Abstand angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

FtnWidth-Eigenschaft

Setzt oder ermittelt die prozentuale Länge der Trennlinie für den Fußnotenbereich an der aktuellen Seitenvorlage.

KLASSE:
BaseTextEditView

SYNTAX:
FtnWidth

RÜCKGABE:
Integer

PARAMETER:
(= Width]: Integer-Ausdruck, der die Länge der Trennlinie in Prozent angibt.

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
```

```
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

Eigenschaften und Methoden mit "G"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "G" beginnen.

GetActiveDatabaseConnection-Methode

Liefert den Namen der aktuellen Datenbankverbindung.

Siehe auch: OpenDatabaseConnection

KLASSE:
BaseTextEditView

SYNTAX:
GetActiveDatabaseConnection()

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
```

```
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
```

```
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetActiveTableConnection-Methode

Liefert den Namen der aktuellen Verbindung zu einer Datenbank-Tabelle.

Siehe auch: OpenTableConnection

KLASSE:
BaseTextEditView

SYNTAX:
GetActiveTableConnection()

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
```

```
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetCharCount-Methode

Zählt alle Zeichen im aktuellen Absatz, im gesamten Dokument oder innerhalb einer Selektion und liefert die Anzahl der Zeichen zurück. Wenn bereits eine Selektion besteht, werden immer nur die Zeichen der Selektion gezählt, unabhängig davon, welcher Wert im ersten Übergabeparameter gesetzt wird.

KLASSE:
BaseText

SYNTAX:
GetCharCount(SelectType As Long, WithInternalChars As Boolean)

RÜCKGABE:
Long

PARAMETER:
[SelectType]: Gibt an, ob Zeichen eines Absatzes oder des Dokuments gezählt werden sollen. Folgende Werte sind möglich:

0: Es werden keine Zeichen gezählt. In diesem Fall liefert die Methode den Wert 0 zurück.

1: Zeichen im aktuellen Absatz werden gezählt.

2: Zeichen im gesamten Dokument werden gezählt.

Dieser Parameter ist optional. Wird er nicht gesetzt, wird die Standardeinstellung SelectType = 0 genommen.

Ist ein Textbereich markiert, wird der Parameter SelectType ignoriert.

[WithInternalChars]: Gibt an, ob interne Steuerzeichen mitgezählt werden sollen (True) oder nicht (False). Wird dieser Parameter nicht gesetzt, werden diese Steuerzeichen nicht mitgezählt (WithInternalChars = False).

BEISPIEL:

```
Sub ExampleGetCharCount
Documents.Add( "sWriter4" )
Selection.Insert "Anzahl der Zeichen am Absatz : "
Selection.Insert Selection.GetCharCount(1, FALSE)
End Sub
```

GetCharStyleName-Methode

Liefert den Namen einer Zeichenvorlage über den angegebenen Index. Um den Index der einzelnen Vorlagen angeben zu können, muß zuvor mit der CharStyleCount-Eigenschaft die Anzahl der verwendeten Zeichenvorlagen ermittelt werden.

In einem neuen Dokument ist keine Zeichenvorlage gesetzt.

Siehe auch: CharStyleCount

KLASSE:
BaseText

SYNTAX:
GetCharStyleName(Idx As Long)

RÜCKGABE:
String

PARAMETER:
Idx: Index der gewünschten Vorlage, der zuvor mittels CharStyleCount definiert wurde.

BEISPIEL:

```
sub ExampleCharStyleName
Const sText="Das ist ein Text in dem Zeichenvorlagen verwendet
werden."
Const sCharStyle1="Fett"
Const sCharStyle2="Unterstrichen"
Documents.Add "sWriter4"
Selection.Insert sText
Selection.Escape
Selection.JumpToStartOfDoc
REM zuerst zwei Zeichenvorlagen definieren
With Selection
.GoToNextWord( TRUE )
.Font.Bold=8 REM auf Fett setzen
.StyleNewByExample( sCharStyle1, 1)
.GoToNextWord( FALSE )
.GoToNextWord( FALSE )
.GoToNextWord( TRUE )
.Font.Underline=2 REM unterstrichen und Rot
.Font.Color= RGB(255,0,0)
.StyleNewByExample( sCharStyle2, 1)
End With
Wait 1000    REM Vorlagen anwenden
With Selection
   .CharStyleApply( sCharStyle1 )
   .GoToNextWord( FALSE )
```

```
   .GoToNextWord( FALSE )
   .GoToNextWord( TRUE )
   .GoToNextWord( TRUE )
   .CharStyleApply( sCharStyle2 )
End With
Rem Vorlagen ausgeben
For iCount=1 To Selection.CharStyleCount
MsgBox " " & iCount &": "& Selection.GetCharStyleName(a%),64,"
" & Selection.CharStyleCount  & " definierte Zeichenvorlage"
next iCount
end sub
```

GetClipboardFormat-Methode

Mit dieser Methode wird die in der Zwischenablage vorhandene Format-ID zurückgeliefert. Mit dieser Format-ID setzen Sie bei PasteSpecial, in welchem Format der Zwischenablageinhalt eingefügt werden soll. Die Format-IDs sind nicht konstant, da die Formate zum Teil erst zur Laufzeit registriert werden.

Klasse:
BaseTextSelection

Syntax:
GetClipboardFormat(Index As Long)

Rückgabe:
Long

Parameter:
Index: Ein Wert, der in dem Bereich liegt, der mit der Funktion GetClipboardFormatCount ermittelt wurde.

Beispiel:

```
Sub ExamplePasteSpecial
Const sWarn1="Das HTML Format wurde nicht gefunden."
Const sWarn2="Es konnte die StarCalc-Tabelle nicht korrekt
eingefügt werden."
Dim sFormat As String
Dim iFormat As long
Dim iState as Integer
Documents.Add("sCalc4")
wait 100
with ActiveDocument
.PutCell(1,1,1,"Text")   REM Ein wenig Text zum Testen
.PutCell(1,2,1,"=a1")    REM und eine Formel
.Range("a1:b2").Copy     REM ins Clipboard stellen
end with
Documents.Add("sWriter4")
wait 100
REM jetzt muessen alle ClipBoardFormate untersucht werden
```

```
for iState = 0 to ActiveWindow.Selection.GetClipboardFormat-
Count() -1
sFormat = Selection.GetClipboardFormatName( ActiveWindow.Selec-
tion.GetClipboardFormat(iState) )
REM Um dem Benutzer klar zu erläutern welche Formate im Clip-
board zur Verfügung stehen,
REM werden die Formate mit den Identifikatoren in den Text
eingetragen.
ActiveWindow.Selection.Insert( sFormat & " : " & iState)
ActiveWindow.Selection.InsertPara
REM Wir verwenden das HTML Format um die Tabelle aus StarCalc
in StarWriter zu kopieren
if InStr( 1, Ucase(sFormat) , "HTML" ) > 0 then iFormat = Acti-
veWindow.Selection.GetClipboardFormat(iState)
next iState
ActiveWindow.Selection.InsertPara
iState = ActiveWindow.Selection.PasteSpecial( iFormat )
if iState<>1 then MsgBox sWarn1 & Chr(13) & sWarn2  ,64,"Clip-
board"
end sub
```

GetClipboardFormatCount-Methode

Mit dieser Methode wird die Anzahl der in der Zwischenablage vorhandenen Formate ermittelt.

KLASSE:
BaseTextSelection

SYNTAX:
GetClipboardFormatCount()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExamplePasteSpecial
Const sWarn1="Das HTML Format wurde nicht gefunden."
Const sWarn2="Es konnte die StarCalc-Tabelle nicht korrekt
eingefügt werden."
Dim sFormat As String
Dim iFormat As long
Dim iState as Integer
Documents.Add("sCalc4")
wait 100
with ActiveDocument
.PutCell(1,1,1,"Text")   REM Ein wenig Text zum Testen
.PutCell(1,2,1,"=a1")    REM und eine Formel
.Range("a1:b2").Copy     REM ins Clipboard stellen
end with
Documents.Add("sWriter4")
wait 100
```

```
REM jetzt muessen alle ClipBoardFormate untersucht werden
for iState = 0 to ActiveWindow.Selection.GetClipboardFormat-
Count() -1
sFormat = Selection.GetClipboardFormatName( ActiveWindow.Selec-
tion.GetClipboardFormat(iState) )
REM Um dem Benutzer klar zu erläutern welche Formate im Clip-
board zur Verfügung stehen,
REM werden die Formate mit den Identifikatoren in den Text
eingetragen.
ActiveWindow.Selection.Insert( sFormat & " : " & iState)
ActiveWindow.Selection.InsertPara
REM Wir verwenden das HTML Format um die Tabelle aus StarCalc
in StarWriter zu kopieren
if InStr( 1, Ucase(sFormat) , "HTML" ) > 0 then iFormat = Acti-
veWindow.Selection.GetClipboardFormat(iState)
next iState
ActiveWindow.Selection.InsertPara
iState = ActiveWindow.Selection.PasteSpecial( iFormat )
if iState<>1 then MsgBox sWarn1 & Chr(13) & sWarn2  ,64,"Clip-
board"
end sub
```

GetClipboardFormatName-Methode

Liefert den Namen eines in der Zwischenablage vorhandenen Formats.

KLASSE:
BaseTextSelection

SYNTAX:
GetClipboardFormatName(Format As Long)

RÜCKGABE:
String

PARAMETER:
Format: Ein Wert, der im Bereich von 1 bis GetClipboardFormatCount liegt.

BEISPIEL:

```
Sub ExamplePasteSpecial
Const sWarn1="Das HTML Format wurde nicht gefunden."
Const sWarn2="Es konnte die StarCalc-Tabelle nicht korrekt
eingefügt werden."
Dim sFormat As String
Dim iFormat As long
Dim iState as Integer
Documents.Add("sCalc4")
wait 100
with ActiveDocument
.PutCell(1,1,1,"Text")   REM Ein wenig Text zum Testen
.PutCell(1,2,1,"=a1")    REM und eine Formel
.Range("a1:b2").Copy     REM ins Clipboard stellen
```

```
end with
Documents.Add("sWriter4")
wait 100
REM jetzt muessen alle ClipBoardFormate untersucht werden
for iState = 0 to ActiveWindow.Selection.GetClipboardFormat-
Count() -1
sFormat = Selection.GetClipboardFormatName( ActiveWindow.Selec-
tion.GetClipboardFormat(iState) )
REM Um dem Benutzer klar zu erläutern welche Formate im Clip-
board zur Verfügung stehen,
REM werden die Formate mit den Identifikatoren in den Text
eingetragen.
ActiveWindow.Selection.Insert( sFormat & " : " & iState)
ActiveWindow.Selection.InsertPara
REM Wir verwenden das HTML Format um die Tabelle aus StarCalc
in StarWriter zu kopieren
if InStr( 1, Ucase(sFormat) , "HTML" ) > 0 then iFormat = Acti-
veWindow.Selection.GetClipboardFormat(iState)
next iState
ActiveWindow.Selection.InsertPara
iState = ActiveWindow.Selection.PasteSpecial( iFormat )
if iState<>1 then MsgBox sWarn1 & Chr(13) & sWarn2  ,64,"Clip-
board"
end sub
```

GetDatabaseFieldValue-Methode

Gibt den Inhalt des angegebenen Datenbankfeldes eines im Dokument eingemischten Datensatzes zurück. Hierbei wird auf den aktuell angezeigten Datensatz zurückgegriffen.

Der Inhalt des angegebenen Datenbankfeldes wird nur dann zurückgegeben, wenn zuvor ein Datensatz in die Datenbankfelder des Dokuments eingemischt wurde.

Siehe auch: DataUpdateFields

KLASSE:
BaseText

SYNTAX:
GetDatabaseFieldValue(DBName As String, TableName As String, ColumnName As String)

RÜCKGABE:
String

PARAMETER:
DBName: Zeichenfolgenausdruck, der den Namen der Datenbank angibt, in der sich der in das Dokument eingemischte Datensatz befindet.

TableName: Zeichenfolgeausdruck, der den Namen der Datenbank-Tabelle angibt, dessen Datensatz in das Dokument eingemischt wurde.

ColumnName: Zeichenfolgenausdruck, der die Bezeichnung des Datenbankfeldes angibt, dessen Inhalt zurückgegeben werden soll. Der mit ColumnName übergebene Feldname muß so geschrieben werden, wie er im Dokument erscheint, da die GetDatabaseFieldValue-Methode andernfalls einen Leerstring zurückgibt.

Beispiel:

```
Sub ExampleGetDatabaseFieldValue
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Inhalt des Feldes REM "
Const sMessage_3 = "REM  ist : "
Dim sDataName As String, sTableName As String, sFieldValue As
String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 to ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
ActiveWindow.DataUpdateFields( sDataName, sTableName, "SELECT *
FROM " & sTableName, 1 )
sFieldValue = Selection.GetDatabaseFieldValue( sDataName,
sTableName, Activewindow.DataColumnName(1) )
Selection.Insert sMessage_2 & Activewindow.DataColumnName(1) &
sMessage_3
Selection.Insert sFieldValue
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

GetDatabaseLimits-Methode

Liefert die Maximalwerte einer geöffneten Datenbank.

Diese Methode läßt sich nur dann aufrufen, wenn keine Verbindung zu einer Datenbank-Tabelle besteht.

Siehe auch: OpenDatabaseConnection, CloseTableConnection.

KLASSE:
BaseTextEditView

SYNTAX:
GetDatabaseLimits(Type As Long)

RÜCKGABE:
Long

PARAMETER:
Type: Gibt an, welcher Maximalwert ermittelt werden soll. Folgende Werte sind möglich:

1: Maximale Länge der Tabellennamen.

2: Maximale Länge der Spaltennamen.

3: Maximale Anzahl der Spalten einer Tabelle.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
```

```
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
```

```
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetDataRecordCount-Methode

Liefert die Anzahl aller Datensätze.

Diese Methode kann bei großen Datenbanken sehr zeitaufwendig werden, da jeder vorhandene Datensatz einmal gelesen wird.

KLASSE:
BaseTextEditView

SYNTAX:
GetDataRecordCount()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
```

```
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

GetFieldFmt-Methode

Gibt einen Long-Integer-Ausdruck zurück, der die Darstellungsform des selektierten Feldbefehls angibt. Der zurückgegebene Ausdruck ist eine Nummer, die die Position der Darstellungsform in dem Listenfeld FORMAT des Dialoges FELDBEFEHL EINFÜGEN bzw. FELDBEFEHL BEARBEITEN bezeichnet, wobei dem ersten Eintrag der Wert 0 zugewiesen ist.

KLASSE:
BaseText

SYNTAX:
GetFieldFmt()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleCurFieldFormat
Dim iFmt As Integer
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert("Nächster Start um: ")
ActiveWindow.Selection.InsertField( 1, 1, "", "" , 0 )
ActiveWindow.Selection.GoLeft 1
ActiveWindow.Selection.InsertPara
Selection.Escape
Activewindow.Selection.JumpToStartOfDoc
ActiveWindow.NextFieldByType( 1 , "" )
ActiveWindow.Selection.SetCurField(1,"","") REM setze auf 24
Stundenformat
iFmt = ActiveWindow.Selection.GetFieldFmt()
Activewindow.Selection.JumpToEndOfDoc
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert("Feld-Format-Identifikation ist:
" & iFmt)
End Sub
```

GetFieldName-Methode

Gibt die Bezeichnung des angegebenen Feldbefehls zurück. Diese entspricht der im Textfeld NAME des Dialogs FELDBEFEHL EINFÜGEN bzw. FELDBEFEHL BEARBEITEN aufgeführten Bezeichnung.

Die einzelnen Definitionen eines Feldtyps sind indiziert und werden über den jeweiligen Index angesprochen. Die Indizierung beginnt bei 0.

Diese Methode ist an allen Feldbefehlen aufrufbar. Bei solchen Feldbefehlen, denen kein Name zugewiesen wird, d. h. die nicht unterschiedlich definiert sind, liefert diese Methode einen Leerstring zurück. Felder, die unterschiedlich definiert sein können, sind vom Typ VARIABLE SETZEN, REFERENZ SETZEN, DDE-FELD, BENUTZERDEFINIERTES FELD und NUMMERNKREIS.

Siehe auch: FieldCount, GetFieldTypeCount, GetFieldTypeID, GetFieldTypeName.

KLASSE:
BaseTextSelection

SYNTAX:
GetFieldName(Pos As Long, FieldType As Long)

RÜCKGABE:
String

PARAMETER:
Pos: Long-Integer-Ausdruck, der die Position des angegebenen Feldes innerhalb aller anderen Felder vom selben Feldtyp bestimmt. Die Position richtet sich nach der Reihenfolge der Feldbefehle im Listenfeld AUSWAHL des Dialogs FELDBEFEHL EINFÜGEN, wobei der erste Eintrag mit dem Index 0 belegt ist. Lediglich bei den Feldtypen VARIABLE SETZEN und NUMMERNKREIS beginnt Pos mit dem Index 3, da die ersten drei Indizes vom System bereits vergeben sind.

FieldType: Long-Integer-Ausdruck, der den Feldbefehl angibt, dessen Bezeichnung ermittelt werden soll. Die folgenden Felder können angegeben werden:

0: Datum

1: Uhrzeit

2: Dateiname

3: Datenbankname

4: Kapitel

5: Seitennummer

6: Statistik

7: Autor

8: Variable setzen

9: Variable anzeigen

10: Formel einfügen

11: Versteckter Text

12: Referenz setzen

13: Referenz einfügen

14: DDE-Feld

15: Makro ausführen

16: Eingabefeld

17: Versteckter Absatz

18: Dokumentinfo-Bereich

19: Serienbrieffeld

20: Benutzerdefiniertes Feld

21: Notizfeld

22: Dokumentvorlage

23: Nummernkreis

24: Nächster Datensatz

25: Beliebiger Datensatz

26: Datensatznummer

27: Bedingter Text

28: Nächste Seite

29: Vorherige Seite

30: Absender

35: Seitenvariable setzen

36: Seitenvariable anzeigen

38: Platzhalter

39: Scriptfeld

Beispiel:

```
Sub ExampleUserFieldName
Const sFieldName = "Feldname"
Const sFieldContent = "Feldinhalt"
Const sMessage = "Der Feldname lautet : "
Documents.Add("sWriter4")
```

```
Selection.InsertField( 20, 1, sFieldName, sFieldContent , 0 )
Selection.GoLeft 1, FALSE
Print sMessage & Selection.GetFieldName(0, 20 )
End Sub
```

GetFieldPar1-Methode

Gibt einen Zeichenfolgenausdruck zurück, der die Bezeichnung eines selektierten benutzerdefinierten Feldes oder die Bedingung eines selektierten Funktionsfeldes aufnimmt. Das sind die Einträge, die im Dialog FELDBEFEHL EINFÜGEN bzw. FELDBEFEHL BEARBEITEN in den Textfeldern NAME oder BEDINGUNG aufgeführt sind.

Diese Methode ist an allen Feldbefehlen aufrufbar. Bei solchen Feldbefehlen, denen weder Name noch Bedingung zugewiesen werden, liefert diese Methode einen Leerstring zurück.

Siehe auch: GetFieldPar2

KLASSE:
BaseText

SYNTAX:
GetFieldPar1()

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleUserVariables
REM Es werden einige Felder definiert die mit vordefinierten
Konstanten gefüllt werden.
REM
Const sWeather0="Ohne Vorhersage"
Const sWeather1="Schlecht"
Const sWeather2="Bedeckter Himmel"
Const sWeather3="Sonnig"
Dim sValue As String , sVar As String
REM
Documents.Add("sWriter4")
with ActiveWindow.Selection
.Insert("Das Wetter heute: ")
.InsertField( 8, 1, "aVariable", sWeather1 , 0 )
.InsertPara
.Insert("Das Wetter morgen: ")
.InsertField( 8, 1, "aVariable", sWeather3 , 0 )
.InsertPara
End With
Selection.Escape
Activewindow.JumpToStartOfDoc
```

```
do while ActiveWindow.NextFieldByType( 8 , "aVariable" )
sVar = ActiveWindow.Selection.GetFieldPar1()  REM ... hat das
Feld den korrekten Typ
sValue = ActiveWindow.Selection.GetFieldPar2()  REM .. hole mir
den Inhalt
REM ... setzt neues Wetter
if sValue = sWeather1 then ActiveWindow.Selection.SetCurField(
0 ,"aVariable", sWeather2 )
loop
End Sub
```

GetFieldPar2-Methode

Gibt einen Zeichenfolgenausdruck zurück, der die vom Feldtyp des selektierten Feldes abhängigen Feldinhalte oder Feldwerte aufnimmt. Das sind die Einträge, die im Feldbefehl-Dialog in den Textfeldern Wert, DDE-Anweisung, Hinweis, Text einfügen usw. aufgeführt sind. Beim Feldtyp Bedingter Text wird der Inhalt der Felder Dann, Sonst in einem String in der Form "Dann|Sonst" zurückgeliefert.

Siehe auch: GetFieldPar1

Klasse:
BaseText

Syntax:
GetFieldPar2()

Rückgabe:
String

Beispiel:

```
Sub ExampleUserVariables
REM Es werden einige Felder definiert die mit vordefinierten
Konstanten gefüllt werden.
REM
Const sWeather0="Ohne Vorhersage"
Const sWeather1="Schlecht"
Const sWeather2="Bedeckter Himmel"
Const sWeather3="Sonnig"
Dim sValue As String , sVar As String
REM
Documents.Add("sWriter4")
with ActiveWindow.Selection
.Insert("Das Wetter heute: ")
.InsertField( 8, 1, "aVariable", sWeather1 , 0 )
.InsertPara
.Insert("Das Wetter morgen: ")
.InsertField( 8, 1, "aVariable", sWeather3 , 0 )
.InsertPara
End With
```

```
Selection.Escape
Activewindow.JumpToStartOfDoc
do while ActiveWindow.NextFieldByType( 8 , "aVariable" )
sVar = ActiveWindow.Selection.GetFieldPar1()  REM ... hat das
Feld den korrekten Typ
sValue = ActiveWindow.Selection.GetFieldPar2()  REM .. hole mir
den Inhalt
REM ... setzt neues Wetter
if sValue = sWeather1 then ActiveWindow.Selection.SetCurField(
0 ,"aVariable", sWeather2 )
loop
End Sub
```

GetFieldTypeCount-Methode

Ermittelt die Anzahl der im Dokument benutzten Felder oder der zur Verfügung stehenden Feldbefehle.

Siehe auch: FieldCount, GetFieldName, GetFieldTypeID, GetFieldTypeName.

KLASSE:
BaseTextSelection

SYNTAX:
GetFieldTypeCount(IsUsed As Boolean)

RÜCKGABE:
Long

PARAMETER:
[IsUsed]: Boolescher Ausdruck, der bestimmt, welche Feldtypenanzahl ermittelt werden soll:

True - Es wird ermittelt, wieviele Felder im Dokument vorhanden sind. Notiz- und Scriptfelder werden ebenfalls mitgezählt.
False - Die Anzahl aller zur Verfügung stehenden Feldbefehle wird ermittelt. Dabei werden alle Feldbefehl-Definitionen gezählt, die es gibt; die Anzahl ist abhängig davon, ob eigene Felder definiert worden sind (z. B. Benutzerfelder oder Felder vom Typ "Variable setzen", "Referenz setzen", "DDE-Feld", oder "Nummernkreis"). Notiz- und Scriptfelder werden nicht mitgezählt.

Der Übergabeparameter ist optional. Wird er nicht angegeben, wird IsUsed = False als Standard-Einstellung genommen.

Beispiel:

```
Sub ExampleGetFieldTypeCount
Const sFieldName = "Feldname"
Const sFieldContent = "Feldinhalt"
Const sMessage_1 = "Anzahl auf dem Dokument vorhandenen Felder
: "
Const sMessage_2 = "Anzahl aller verfügbaren Felder : "
Documents.Add("sWriter4")
Selection.InsertField( 20, 1, sFieldName, sFieldContent , 0 )
Print sMessage_1 & Selection.GetFieldTypeCount(TRUE)
Print sMessage_2 & Selection.GetFieldTypeCount(FALSE)
End Sub
```

GetFieldTypeId-Methode

Ermittelt die ID-Nummer eines Feldbefehls, der mit dem jeweiligen Index ausgewählt wird. Mögliche Rückgabewerte sind:

0: Datum

1: Uhrzeit

2: Dateiname

3: Datenbankname

4: Kapitel

5: Seitennummer

6: Statistik

7: Autor

8: Variable setzen

9: Variable anzeigen

10: Formel einfügen

11: Versteckter Text

12: Referenz setzen

13: Referenz einfügen

14: DDE-Feld

15: Makro ausführen

16: Eingabefeld

17: Versteckter Absatz

18: Dokumentinfo-Bereich

19: Serienbrieffeld

20: Benutzerdefiniertes Feld

21: Notizfeld

22: Dokumentvorlage

23: Nummernkreis

24: Nächster Datensatz

25: Beliebiger Datensatz

26: Datensatznummer

27: Bedingter Text

28: Nächste Seite

29: Vorherige Seite

30: Absender

35: Seitenvariable setzen

36: Seitenvariable anzeigen

38: Platzhalter

39: Scriptfeld

Siehe auch: FieldCount, GetFieldName, GetFieldTypeCount, GetFieldTypeName.

KLASSE:
BaseTextSelection

SYNTAX:
GetFieldTypeId(Index As Long, IsUsed As Boolean)

RÜCKGABE:
Long

PARAMETER:
[Index]: Index des Feldbefehls, dessen ID ermittelt werden soll. Der Wertebereich ist abhängig vom zweiten Übergabeparameter.

Der Parameter Index ist optional. Wird er nicht angegeben, wird Index = 0 genommen. Die Methode liefert dann die ID-Nummer des ersten Feldes im Dokument (IsUsed = True) oder des ersten zur Verfügung stehenden Feldtyps (IsUsed = False).

[IsUsed]: Boolescher Ausdruck, der angibt, über welche Feldtypenanzahl gezählt wird. Diese Anzahl bestimmt den Wert des Parameters Index. Folgende Möglichkeiten gibt es:

True - Die Anzahl der im Dokument vorhandenen Felder wird ermittelt.

False - Die Anzahl aller zur Verfügung stehenden Feldtypen wird ermittelt.

Der Parameter IsUsed ist optional. Wird er nicht angegeben, wird IsUsed = False genommen.

BEISPIEL:

```
Sub ExampleGetFieldTypeID
Const sTableHeader_1 = "Name"
Const sTableHeader_2 = "Typ"
Const sMessage = "Alle verfügbaren Felder mit ihren Namen und
Typen"
Selection.Insert sMessage
Selection.InertPara
Selection.InsertTable( 2, 2)
Selection.Insert sTableHeader_1
Selection.JumpToNextCell
Selection.Insert sTableHeader_2
Selection.JumpToNextCell
For iCount = 0 To Selection.GetFieldTypeCount(FALSE) - 1
Selection.Insert Selection.GetFieldTypeName( iCount, FALSE )
Selection.JumpToNextCell
Selection.Insert Selection.GetFieldTypeId( iCount, FALSE )
Selection.JumpToNextCell
Next iCount
End Sub
```

GetFieldTypeName-Methode

Ermittelt den Namen eines Feldbefehls, der mit dem jeweiligen Index ausgewählt wird.

Feldbefehle, die nicht unterschiedlich definiert sein können, haben keine unterschiedliche Namen. Hier liefert GetFieldTypeName einen Leerstring zurück. Sinnvolle Rückgabewerte liefern die Felder VARIABLE SETZEN, REFERENZ SETZEN, DDE-FELD, BENUTZERDEFINIERTES FELD und NUMMERNKREIS.

Siehe auch: FieldCount, GetFieldName, GetFieldTypeCount, GetFieldTypeID.

KLASSE:
BaseTextSelection

SYNTAX:
GetFieldTypeName(Index As Long, IsUsed As Boolean)

RÜCKGABE:
String

PARAMETER:
[Index]: Index des Feldbefehls, dessen Name ermittelt werden soll. Der Wertebereich ist abhängig vom zweiten Übergabeparameter.

Der Parameter Index ist optional. Wird er nicht angegeben, wird Index = 0 genommen. Die Methode liefert dann den Namen des ersten Feldes im Dokument (IsUsed = True) oder des ersten zur Verfügung stehenden Feldtyps (IsUsed = False).

[IsUsed]: Boolescher Ausdruck, der angibt, über welche Feldtypenanzahl gezählt wird. Diese Anzahl bestimmt den Wert des Parameters Index. Folgende Möglichkeiten gibt es:

True - Die Anzahl der im Dokument vorhandenen Felder wird ermittelt.

False - Die Anzahl aller zur Verfügung stehenden Feldtypen wird ermittelt.

Der Parameter IsUsed ist optional. Wird er nicht angegeben, wird IsUsed = False genommen.

BEISPIEL:

```
Sub ExampleGetFieldTypeID
Const sTableHeader_1 = "Name"
Const sTableHeader_2 = "Typ"
Const sMessage = "Alle verfügbaren Felder mit ihren Namen und
Typen"
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.Insert sMessage
Selection.InsertPara
Selection.InsertTable( 2, 2)
Selection.Insert sTableHeader_1
Selection.JumpToNextCell
Selection.Insert sTableHeader_2
Selection.JumpToNextCell
For iCount = 0 To Selection.GetFieldTypeCount(FALSE) - 1
Selection.Insert Selection.GetFieldTypeName( iCount, FALSE )
Selection.JumpToNextCell
Selection.Insert Selection.GetFieldTypeId( iCount, FALSE )
Selection.JumpToNextCell
Next iCount
End Sub
```

GetFormNames-Methode

Liefert eine Liste aller vorhandenen Formularnamen, die an der geöffneten Datenbank definiert sind. Der Rückgabeparameter ist vom Typ Strings.

KLASSE:
BaseTextEditView

SYNTAX:
GetFormNames()

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wird ein Formular bzw.
Bericht werden."
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Database & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close( True, sPath )
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateForm( sFormName, sPath )
ActiveWindow.CreateReport( sReportName, sPath )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " & sTab-
Name, FALSE )
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
```

```
ActiveWindow.DeleteReport sReportName
ActiveWindow.DeleteQuery sQueryName
End Sub
```

GetFramestyleMacroLib-Methode

Liefert die Bibliothek des Makros das einem bestimmten Ereignis zugeordnet wurde.

KLASSE:
BaseTextDocument

SYNTAX:
GetFramestyleMacroLib(StyleName As String, Event As Long)

RÜCKGABE:
String

PARAMETER:
StyleName: Name der Rahmenvorlage

Event: Nummer des Ereignisses dem ein Makro zugeordnet wurde. Folgende Ereignisse werden unterstützt:

Eventnummer - Ereignis

1 - Maus über Objekt

2 - Hyperlink auslösen

3 - Maus verläßt Objekt

4 - Objekt anklicken

5 - Eingabe von Alphazeichen

6 - Eingabe von nicht Alphazeichen

7 - Rahmengröße verändern

8 - Rahmen verschieben

9 - Grafik laden erfolgreich

10 - Grafik laden abgebrochen

11 - Grafik laden fehlerhaft

BEISPIEL:

```
Sub ExampleGetFrameStyleMacroLib
Const sStyleName = "NeueVorlage"
Const sMessage_1 = "Die Bibliothek des verwendeten Makros
lautet : "
```

```
Const sMessage_2 = "Die Prozedure des verwendeten Makros lautet
: "
Const sMessage_3 = "Wenn die Maus den Frame verläßt, wird ein
Makro ausgeführt"
Dim sFrameName As String
Documents.Add( "sWriter4" )
Activedocument.NewStyle( sStyleName, 4)
ActiveDocument.SetFrameStyleMacro( sStyleName, 3, "StartChange-
sUserfields(Gimmicks.Userfields)", "StarOffice" )
Print sMessage_1 & ActiveDocument.GetFrameStyleMacroLib( sSty-
leName, 3 )
Print sMessage_2 & ActiveDocument.GetFrameStyleMacroSub( sSty-
leName, 3 )
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Insert sMessage_3
Selection.StyleApply( sStyleName, 4 )
Selection.AlignLeft = "2cm"
Selection.AlignTop = "2cm"
End Sub
```

GetFramestyleMacroSub-Methode

Liefert die Sub oder Function des Makros, daß einem bestimmten Ereignis zugeordnet wurde.

KLASSE:
BaseTextDocument

SYNTAX:
GetFramestyleMacroSub(StyleName As String, Event As Long)

RÜCKGABE:
String

PARAMETER:
StyleName: Name der Rahmenvorlage

Event: Nummer des Ereignisses dem ein Makro zugeordnet wurde. Folgende Ereignisse werden unterstützt:

Eventnummer - Ereignis

1 - Maus über Objekt

2 - Hyperlink auslösen

3 - Maus verläßt Objekt

4 - Objekt anklicken

5 - Eingabe von Alphazeichen

6 - Eingabe von nicht Alphazeichen

7 - Rahmengröße verändern

8 - Rahmen verschieben

9 - Grafik laden erfolgreich

10 - Grafik laden abgebrochen

11 - Grafik laden fehlerhaft

Beispiel:

```
Sub ExampleGetFrameStyleMacroLib
Const sStyleName = "NeueVorlage"
Const sMessage_1 = "Die Bibliothek des verwendeten Makros
lautet : "
Const sMessage_2 = "Die Prozedure des verwendeten Makros lautet
: "
Const sMessage_3 = "Wenn die Maus den Frame verläßt, wird ein
Makro ausgeführt"
Dim sFrameName As String
Documents.Add( "sWriter4" )
Activedocument.NewStyle( sStyleName, 4)
ActiveDocument.SetFrameStyleMacro( sStyleName, 3, "StartChange-
sUserfields(Gimmicks.Userfields)", "StarOffice" )
Print sMessage_1 & ActiveDocument.GetFrameStyleMacroLib( sSty-
leName, 3 )
Print sMessage_2 & ActiveDocument.GetFrameStyleMacroSub( sSty-
leName, 3 )
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Insert sMessage_3
Selection.StyleApply( sStyleName, 4 )
Selection.AlignLeft = "2cm"
Selection.AlignTop = "2cm"
End Sub
```

GetFramestyleScript-Methode

Liefert das Script, das einem bestimmten Ereignis zugeordnet wurde.

Klasse:
BaseTextDocument

Syntax:
GetFramestyleScript(StyleName As String, Event As Long)

Rückgabe:
String

Parameter:
StyleName: Name der Rahmenvorlage

Event: Nummer des Ereignisses, dem ein Script zugeordnet wurde. Folgende Ereignisse werden unterstützt:

Eventnummer - Ereignis

1 - Maus über Objekt

2 - Hyperlink auslösen

3 - Maus verläßt Objekt

4 - Objekt anklicken

5 - Eingabe von Alphazeichen

6 - Eingabe von nicht Alphazeichen

7 - Rahmengröße verändern

8 - Rahmen verschieben

9 - Grafik laden erfolgreich

10 - Grafik laden abgebrochen

11 - Grafik laden fehlerhaft

Beispiel:

```
Sub ExampleSetFramestyleScript
Const sStyleName = "NeueVorlage"
Const sScriptText = "Dies ist ein Javascript"
Const sMessage_1 = "Das JavaScript für dieses Ereignis lautet :
"
Const sMessage_2 = "Wenn die Maus den Frame verläßt, wird ein
Javascript ausgeführt"
Dim sFrameName As String
Documents.Add( "sWriter4" )
Activedocument.NewStyle( sStyleName, 4)
ActiveDocument.SetFramestyleScript( sStyleName, 3, "alert(""" &
sScriptText & """)" )
Print sMessage_1 & ActiveDocument.GetFramestyleScript( sSty-
leName, 3 )
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Insert sMessage_2
Selection.StyleApply( sStyleName, 4 )
Selection.AlignLeft = "2cm"
Selection.AlignTop = "2cm"
End Sub
```

GetGlossaryBlockCount-Methode

Liefert die Anzahl der Textbausteine innerhalb der Gruppe Group.

siehe auch: GetGlossaryGroupCount, GetGlossaryBlockName,GetGlossary-GroupName

KLASSE:
BaseText

SYNTAX:
GetGlossaryBlockCount(Group As String)

RÜCKGABE:
Long

PARAMETER:
Group: Name der AutoText-Gruppe

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
```

```
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

GetGlossaryBlockName-Methode

Liefert den Namen des Textbausteins an der Position Index innerhalb der AutoText-Gruppe GroupName.

siehe auch: GetGlossaryGroupCount, GetGlossaryGroupName, GetGlossaryBlockCount

KLASSE:
BaseText

SYNTAX:
GetGlossaryBlockName(GroupName As String, Index As Long)

RÜCKGABE:
String

PARAMETER:
GroupName: Name der AutoText-Gruppe

Index: Position innerhalb der Gruppe

Wertebereich: von 0 bis GetGlossaryBlockCount -1

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
```

```
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

GetGlossaryGroupCount-Methode

Liefert die Anzahl der vorhandenen AutoText-Gruppen.

KLASSE:
BaseText

SYNTAX:
GetGlossaryGroupCount()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
```

```
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

GetGlossaryGroupName-Methode

Liefert den Namen der AutoText-Gruppe an der Position Index.

siehe auch: GetGlossaryGroupCount

KLASSE:
BaseText

SYNTAX:
GetGlossaryGroupName(Index As Long)

RÜCKGABE:
String

PARAMETER:
Index: Position der AutoText-Gruppe

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
```

```
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

GetGlossaryShortName-Methode

Ermittelt das Kürzel eines AutoTextes und liefert es als String zurück.

KLASSE:
BaseText

SYNTAX:
GetGlossaryShortName(GroupName As String, Name As String)

RÜCKGABE:
String

PARAMETER:
GroupName: Zeichenfolgeausdruck, der die AutoText-Gruppe angibt, in der sich der AutoText befindet.

Name: Zeichenfolgeausdruck, der den Namen des AutoTextes angibt, dessen Kürzel ermittelt werden soll.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
```

```
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

GetHandler-Methode

Führt das Makro aus, das dem Ereignis an dem selektierten Kontrollfeld zugewiesen wurde.

Diese Methode gilt nur für Kontrollfelder, die sich in einem Dokument befinden. Kontrollfelder eines Dialoges sind mit GETHANDLER nicht ansprechbar.

KLASSE:
BaseTextDrawControl

SYNTAX:
GetHandler(EventId As Long)

RÜCKGABE:
String

PARAMETER:
EventId: Long-Integer-Ausdruck, der das Ereignis bestimmt, zu dem das Makro ermittelt werden soll. Folgende Werte sind möglich:

2 - OnChange

3 - OnClick

4 - OnSelect

6 - OnGotFocus

7 - OnLostFocus

14 - OnSpinUp

15 - OnSpinDown

16 - OnScroll

BEISPIEL:

```
Sub ExampleSetHandler
Const sFileName = "\Test.sdw"
Const sMessage = "An dem Commandbutton hängt folgendes Makro :
"
Dim sPath As String
Documents.Add( "sWriter4" )
sPath = Application.PathSettings.Work & sFileName
REM Einen Commandbutton auf dem Dokument erstellen
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
Activewindow.ControlObject.Caption = "Makro ausführen"
Selection.SetHandler( 3, "StartChangesUserfields(Gimmicks.User-
fields)" )
Print sMessage & Selection.GetHandler( 3 )
Selection.Escape
ActiveDocument.SaveAs sPath
End Sub
```

GetHyperlinkMacroLib-Methode

Liefert den Namen der Library, die das Makro enthält, das ausgeführt wird wenn das definierte Ereignis (Event) eintritt.

KLASSE:
TextFont

SYNTAX:
GetHyperlinkMacroLib(Event As Long)

RÜCKGABE:
String

PARAMETER:
Event: Nummer des Ereignisses das erfragt werden soll.

Ereignis	Eventnummer
Maus über Objekt	1
Hyperlink auslösen	2
Maus verläßt Objekt	3

BEISPIEL:

```
Sub ExampleHyperLinkMacro
Dim sProcName As String
Dim sLibName As String
Documents.Add "sWriter4"
Selection.SetHyperlink("Stardivision", "http://www.Stardivi-
sion.de", "_blank", 1)
Selection.GotoStartOfLine FALSE
```

```
Selection.Font.SetHyperlinkMacro(2, "Gimmicks.Userfields.Start-
ChangesUserfields", "Gimmicks")
sProcName = Selection.Font.GetHyperlinkMacroSub(2)
sLibName = Selection.Font.GetHyperlinkMacroLib(2)
Print "Folgende Prozedure ist mit dem Hyperlink verknüpft : " &
sProcName
Print "Folgende Bibliothek ist mit dem Hyperlink verknüpft : "
& sLibName
End Sub
```

GetHyperlinkMacroSub-Methode

Liefert den Namen der Sub oder Funktion, welche ausgeführt wird, wenn das definierte Ereignis (Event) eintritt.

KLASSE:
TextFont

SYNTAX:
GetHyperlinkMacroSub(Event As Long)

RÜCKGABE:
String

PARAMETER:
Event: Nummer des Ereignisses das erfragt werden soll.

Ereignis	Eventnummer
Maus über Objekt	1
Hyperlink auslösen	2
Maus verläßt Objekt	3

BEISPIEL:

```
Sub ExampleHyperLinkMacro
Dim sProcName As String
Dim sLibName As String
Documents.Add "sWriter4"

Selection.SetHyperlink("Stardivision", "http://www.Stardivi-
sion.de", "_blank", 1)
Selection.GotoStartOfLine FALSE

Selection.Font.SetHyperlinkMacro(2, "Gimmicks.Userfields.Start-
ChangesUserfields", "Gimmicks")
sProcName = Selection.Font.GetHyperlinkMacroSub(2)
sLibName = Selection.Font.GetHyperlinkMacroLib(2)
Print "Folgende Prozedure ist mit dem Hyperlink verknüpft : " &
sProcName
```

```
Print "Folgende Bibliothek ist mit dem Hyperlink verknüpft : "
& sLibName
End Sub
```

GetHyperlinkScript-Methode

Liefert den Inhalt der Scriptes, welches ausgeführt wird, wenn das definierte Ereignis (Event) eintritt.

KLASSE:
TextFont

SYNTAX:
GetHyperlinkScript(Event As Long)

RÜCKGABE:
String

PARAMETER:
Event: Nummer des Ereignisses das erfragt werden soll.

Ereignis	Eventnummer
Maus über Objekt	1
Hyperlink auslösen	2
Maus verläßt Objekt	3

BEISPIEL:

```
Sub ExampleHyperLinkScript
Dim sScript As String
Documents.Add "sWriter4"
Selection.SetHyperlink("Stardivision", "http://www.Stardivi-
sion.de", "_blank", 1)
Selection.GotoStartOfLine FALSE
Selection.Font.SetHyperlinkScript(1, "alert( JavaScript )")
sScript = Selection.Font.GetHyperlinkScript(1)
Print "Folgendes JavaScript ist mit dem Hyperlink verknüpft : "
& sScript
End Sub
```

GetLineCount-Methode

Liefert die Anzahl der Textzeilen im Dokument. Der Übergabeparameter regelt, ob alle Zeilen im Dokument gezählt werden sollen oder nur bis zur Cursorposition.

KLASSE:
BaseText

Syntax:
GetLineCount(AtActPos As Boolean)

Rückgabe:
Long

Parameter:
[AtActPos]: Boolescher Ausdruck, der angibt, ob die Zeilen bis zur Cursorposition (True) oder alle Zeilen im Dokument (False) ermittelt werden sollen. Dieser Parameter ist optional. Wird kein Parameter übergeben, werden standardmäßig alle Zeilen im Dokument gezählt (AtActPos = False).

Beispiel:

```
Sub ExampleGetLineCount
Const sMessage_1 = "Diese ist die erste Zeile."
Const sMessage_2 = "Anzahl der auf dem Dokument enthaltenen
Zeilen : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.Insert Selection.GetLineCount( FALSE )
End Sub
```

GetPageStyleCount-Eigenschaft

Liefert die Anzahl der am Dokument verwendeten Seitenvorlagen.

Klasse:
BaseTextEditView

Syntax:
GetPageStyleCount

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
Sub ExamplePageStyle
Const sPageStyle = "Neue Vorlage"
Const sMessage = "Folgende Seitenvorlagen sind verwendet : "
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.StyleNewByExample( sPageStyle, 8)
ActiveWindow.PageStyleApply( sPageStyle )
For iCount=0 to Activewindow.GetPageStyleCount-1
```

```
print sMessage & Activewindow.GetPageStyleNameAt(iCount)
Next iCount
End Sub
```

GetPageStyleNameAt-Methode

Liefert den Namen einer Seitenvorlage über den Index.

siehe auch: GetPageStyleCount

KLASSE:
BaseTextEditView

SYNTAX:
GetPageStyleNameAt(Idx As Long)

RÜCKGABE:
String

PARAMETER:
Idx: Long-Integer-Ausdruck, der den Index der Seitenvorlage angibt, dessen Name ermittelt werden soll. Die Indizierung beginnt bei Null; Idx muß kleiner sein als der mit GETPAGESTYLECOUNT ermittelte Wert.

BEISPIEL:

```
Sub ExamplePageStyle
Const sPageStyle = "Neue Vorlage"
Const sMessage = "Folgende Seitenvorlagen sind verwendet : "
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.StyleNewByExample( sPageStyle, 8)
ActiveWindow.PageStyleApply( sPageStyle )
For iCount=0 to Activewindow.GetPageStyleCount-1
print sMessage & Activewindow.GetPageStyleNameAt(iCount)
Next iCount
End Sub
```

GetParaStyleName-Methode

Liefert den Namen einer Absatzvorlage über den Index.

Mit ParaStyleCount ermittelt man die Anzahl aller verwendeten Absatzvorlagen. Mit GetParaStyleName(Index) ermittelt man den Namen. Der Index beginnt bei 0.

Siehe auch: ParaStyleCount

KLASSE:
BaseText

Syntax:
GetParaStyleName(Idx As Long)

Rückgabe:
String

Parameter:
Idx: Index der Vorlage, deren Namen ermittelt werden soll.

Beispiel:

```
Sub ExampleGetParaStyleName
Const sHeader = "Es sind folgende Vorlagen verwendet worden : "
Const sStyle = "Vorlage : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.StyleNewByExample( "NeueVorlage", 2 )
Selection.Insert sHeader
Selection.InsertPara
for iCount=0 to Activewindow.ParaStyleCount-1
Selection.Insert sStyle &
Activewindow.GetParaStyleName(iCount)
Selection.InsertPara
next iCount
End Sub
```

GetQueryColumns-Methode

Liefert die Spaltenbeschreibungen der Datenbank-Abfrage. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: GetTableColumns

Klasse:
BaseTextEditView

Syntax:
GetQueryColumns(QueryName As String)

Rückgabe:
Object

Parameter:
QueryName: Name der Datenbank-Abfrage, deren Spaltenbeschreibung ermittelt werden soll.

Beispiel:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
```

```
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
```

```
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetQueryNames-Methode

Liefert die Namen aller Abfragen einer geöffneten Datenbank. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: OpenDatabaseConnection

KLASSE:
BaseTextEditView

SYNTAX:
GetQueryNames()

RÜCKGABE:
Object

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
```

```
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetRegionCondition-Methode

Liefert für den Bereich RegionName den Ausdruck für die Bedingung, unter der der Bereich versteckt wird.

KLASSE:
BaseTextSelection

SYNTAX:
GetRegionCondition(RegionName As String)

RÜCKGABE:
String

PARAMETER:
RegionName: Bereichsname, dessen Eigenschaft erfragt werden soll.

BEISPIEL:

```
Sub ExampleGetRegionCondition
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Die Bedingung von dem Bereich REM "
Const sMessage_3 = "REM  lautet : "
Dim sRegionCond As String
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sRegionName, "False", FALSE, FALSE)
Selection.Escape
Selection.SetRegionCondition( sRegionName, "True" )
Selection.InsertPara
sRegionCond = Selection.GetRegionCondition( sRegionName )
Selection.Insert sMessage_2 & sRegionName & sMessage_3 &
sRegionCond
End Sub
```

GetRegionCount-Methode

Liefert die Anzahl der im Dokument enthaltenen Bereiche.

KLASSE:
BaseTextSelection

SYNTAX:
GetRegionCount()

RÜCKGABE:
Long

Beispiel:

```
Sub ExampleGetRegionName
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich "
Const sMessage_3 = " hat folgenden Namen : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
For iCount = 1 to Selection.GetRegionCount
Selection.Insert sMessage_2 & iCount & sMessage_3 &
Selection.GetRegionName(iCount-1)
Next iCount
Selection.DeleteRegion sRegionName
End Sub
```

GetRegionLink-Methode

Liefert für den angegebenen Bereich Pfad und Namen der verknüpften Datei.

Nur sinnvoll bei Bereichen, die als Verknüpfung vorhanden sind.

Klasse:
BaseTextSelection

Syntax:
GetRegionLink(RegionName As String)

Rückgabe:
String

Parameter:
RegionName: Bereichsname, dessen Verknüpfung erfragt werden soll.

Beispiel:

```
Sub ExampleSetRegionLink
Const sRegionName = "Bereich"
Const sRegionDocName = "Bereich.sdw"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der verknüpfte Bereich besteht aus
folgender Datei : "
Const sMessage_3 = "und es wurde folgender Filter verwendet : "
Dim sRegionPath As String
sRegionPath = Application.PathSettings.Work & sRegionDocName
Documents.Add( "sWriter4" )
```

```
Selection.InsertPara
Selection.Insert sMessage_1
ActiveDocument.Close TRUE, sRegionPath
Documents.Add( "sWriter4" )
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionLink( sRegionName, sRegionPath, "StarWriter
4.0")
Selection.Insert sMessage_2 & Selection.GetRegionLink( sRegion-
Name )
Selection.InsertPara
Selection.Insert sMessage_3 & Selection.GetRegionLinkFilter(
sRegionName )
End Sub
```

GetRegionLinkBookmark-Methode

Ist ein Bereich mit einer Textmarke oder einem Bereich innerhalb einer Datei verknüpft, so liefert diese Methode den Namen der Textmarke oder des Bereichs.

KLASSE:
BaseTextSelection

SYNTAX:
GetRegionLinkBookmark(RegionName As String)

RÜCKGABE:
String

PARAMETER:
RegionName: Name des Bereichs

BEISPIEL:

```
Sub ExampleSetRegionLinkBookmark
Const sRegionName = "Bereich"
Const sRegionInDoc = "DokBereich"
Const sRegionDocName = "Bereich.sdw"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Diese Zeile wird nicht eingefügt"
Const sMessage_3 = "Der Bereich im Dokument heißt : "
Dim sRegionPath As String
sRegionPath = Application.PathSettings.Work & sRegionDocName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionInDoc, "True", FALSE, FALSE)
ActiveDocument.Close TRUE, sRegionPath
Documents.Add( "sWriter4" )
```

```
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionLink( sRegionName, sRegionPath, "StarWriter
4.0")
Selection.SetRegionLinkBookmark( sRegionName, sRegionInDoc )
Selection.Insert sMessage_3 & Selection.GetRegionLinkBookmark(
sRegionName )
End Sub
```

GetRegionLinkFilter-Methode

Liefert den Namen des Dateifilters für die Dateiverknüpfung des angegebenen Bereichs. Für verknüpfte Bereiche muß der Filtername nicht gesetzt sein.

KLASSE:
BaseTextSelection

SYNTAX:
GetRegionLinkFilter(RegionName As String)

RÜCKGABE:
String

PARAMETER:
RegionName: Bereichsname, dessen Verknüpfung erfragt werden soll.

BEISPIEL:

```
Sub ExampleSetRegionLink
Const sRegionName = "Bereich"
Const sRegionDocName = "Bereich.sdw"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der verknüpfte Bereich besteht aus
folgender Datei : "
Const sMessage_3 = "und es wurde folgender Filter verwendet : "
Dim sRegionPath As String
sRegionPath = Application.PathSettings.Work & sRegionDocName
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
ActiveDocument.Close TRUE, sRegionPath
Documents.Add( "sWriter4" )
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionLink( sRegionName, sRegionPath, "StarWriter
4.0")
Selection.Insert sMessage_2 & Selection.GetRegionLink( sRegion-
Name )
```

```
Selection.InsertPara
Selection.Insert sMessage_3 & Selection.GetRegionLinkFilter(
sRegionName )
End Sub
```

GetRegionName-Methode

Liefert den Namen eines Bereichs an der Position Index.

Siehe auch: GetRegionCount

KLASSE:
BaseTextSelection

SYNTAX:
GetRegionName(Index As Long)

RÜCKGABE:
String

PARAMETER:
Index: Interne Position des Bereichs. Der Wert darf zwischen Null und GetRegionCount -1 liegen.

BEISPIEL:

```
Sub ExampleGetRegionName
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich "
Const sMessage_3 = " hat folgenden Namen : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
For iCount = 1 to Selection.GetRegionCount
Selection.Insert sMessage_2 & iCount & sMessage_3 &
Selection.GetRegionName(iCount-1)
Next iCount
Selection.DeleteRegion sRegionName
End Sub
```

GetReportNames-Methode

Liefert eine Liste alle Berichte der geöffneten Datenbank. Der Rückgabeparameter ist vom Datentyp Strings.

Siehe auch: OpenDatabaseConnection, CreateReport, DeleteReport.

KLASSE:
BaseTextEditView

SYNTAX:
GetReportNames()

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleCreateForm
Const sFileName = "\Test.sdw"
Const sFormName = "NeuesFormular"
Const sReportName = "NeuerBericht"
Const sQueryName = "NeueAbfrage"
Const sMessage_1 = "Dieses Dokument wurde Automatisch erstellt"
Const sMessage_2 = "Der Name des ersten Formulars : "
Const sMessage_3 = "Der Name der ersten Abfrage : "
Const sMessage_4 = "Der Name des ersten Berichts : "
Dim stStringsColl As New Strings
Dim sDataName As String, sTabName As String, sPath As String
Dim sStatus As Integer
sPath = Application.Pathsettings.Work & sFileName
Documents.Add "sWriter4"
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
sStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
Documents.Add "sWriter4"
Selection.Insert sMessage_1
REM Das Dokument wird gespeichert und danach geschlossen
Activedocument.Close True, sPath
ActiveWindow.CreateForm sFormName, sPath
ActiveWindow.CreateReport sReportName, sPath
ActiveWindow.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
stStringsColl = ActiveWindow.GetFormNames
Print sMessage_2 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetQueryNames
Print sMessage_3 & stStringsColl.Item(1)
stStringsColl = ActiveWindow.GetReportNames
Print sMessage_2 & stStringsColl.Item(1)
ActiveWindow.DeleteForm sFormName
```

```
ActiveWindow.DeleteReport sReportName
ActiveWindow.DeleteQuery sQueryName
End Sub
```

GetSQL_Statement-Methode

Liefert das SQL-Statement einer Datenbank-Abfrage.

KLASSE:
BaseTextEditView

SYNTAX:
GetSQL_Statement(QueryName As String)

RÜCKGABE:
String

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
```

```
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
```

```
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetTableColumns-Methode

Liefert die Beschreibung aller Spalten einer Datenbank-Tabelle als Variable vom Datentyp Strings zurück. Die einzelnen Bestandteile der ermittelten Strings-Variablen beschreiben je eine Spalte der Tabelle und sind vom gleichen Format wie bei der CreateTable-Methode ("Feldname, Feldtyp, Feldlänge, Eingabe").

KLASSE:
BaseTextEditView

SYNTAX:
GetTableColumns(TableName As String)

RÜCKGABE:
Object

PARAMETER:
TableName: Name der Tabelle, deren Spaltenbeschreibungen erfragt werden sollen.

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
```

```
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
```

```
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

GetUserFieldValue-Methode

Gibt den Inhalt eines benutzerdefinierten Feldbefehls zurück.

KLASSE:
BaseTextSelection

SYNTAX:
GetUserFieldValue(Name As String)

RÜCKGABE:
String

PARAMETER:
Name: Zeichenfolgenausdruck, der die Bezeichnung des Feldes aufnimmt, dessen Inhalt übergeben werden soll.

Beispiel:

```
Sub ExampleUserFieldCount
Const sMessage_1 = "Erstes Benutzerfeld : "
Const sMessage_2 = "Zweites Benutzerfeld : "
Const sMessage_3 = "Anzahl der Benutzerfelder auf dem Dokument
: "
Const sMessage_4 = "Der Inhalt vom ersten Benutzerfeld : "
Const sUserName_1 = "Benutzerfeld_1"
Const sUserName_2 = "Benutzerfeld_2"
Const sUserContent_1 = "Benutzerfeld 1"
Const sUserContent_2 = "Benutzerfeld 2"
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_1, sUserCon-
tent_1, 0 )
Selection.InsertPara
Selection.Insert( sMessage_2 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_2, sUserCon-
tent_2, 0 )
Selection.InsertPara
Selection.Insert sMessage_4
Selection.Insert Selection.GetUserFieldValue( sUserName_1 )
Selection.InsertPara
iCount = ActiveWindow.FieldCount( TYP_USERFLD )
MsgBox sMessage_3 & iCount, 64, ""
End Sub
```

GlobalContents-Eigenschaft

Ist die Collection für die Inhalte eines Globaldokuments.

Klasse:
BaseTextEditView

Syntax:
GlobalContents

Rückgabe:
Object

Beispiel:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\GLOBAL.SGL"
Const sClientDocName = "\CLIENT.SDW"
Const sMessage_1 = "Der Titel des Globaldokumentsbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentsbereichs : "
```

```
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
```

```
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

GoDown-Methode

Bewegt die Schreibmarke um die angegebene Anzahl an Zeilen nach unten.

KLASSE:
BaseText

SYNTAX:
GoDown(Count As Integer, Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Count]: Ein beliebiger Integer-Ausdruck, der angibt, um wieviele Zeilen die Schreibmarke nach unten bewegt werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird als Standardwert Count = 1 genommen.

[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGoUpDown
Const sMessage_1 = "Dies ist die erste Zeile"
Const sMessage_2 = "Diese ist die letzte Zeile"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.GoUp( 1, FALSE )
Selection.Insert sMessage_1
Selection.GoDown 1, FALSE
Selection.Insert sMessage_2
End Sub
```

GoIntoFrame-Methode

Setzt den Cursor in den angegebenen Rahmen. Der Inhalt wird ggf. selektiert (SelectAll = True).

Die Methode liefert den Wert True zurück, wenn die Aktion erfolgreich war.

KLASSE:
BaseText

SYNTAX:
GoIntoFrame(FrameName As String, SelectAll As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
FrameName: Zeichenfolge, die den Namen des Rahmens angibt, in den der Cursor gesetzt werden soll.

SelectAll: Boolescher Ausdruck, der angibt, ob der Inhalt des Rahmens selektiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

GoldCut-Eigenschaft

Schaltet die Option DYNAMISCHER UMLAUF bei Textrahmen ein oder aus.

KLASSE:
BaseTextFrame

SYNTAX:
GoldCut

RÜCKGABE:
Boolean

PARAMETER:
True: Option DYNAMISCHER UMLAUF aktivieren

False: Option DYNAMISCHER UMLAUF ausschalten

BEISPIEL:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
   Selection.Insert(sText)
   if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALS
Selection.Printable = TRUE
Selection.SendToBack
Selection.BringToFront
end sub
```

GoLeft-Methode

Bewegt die Schreibmarke um die angegebene Anzahl an Zeichen nach links. Befindet sich der Cursor bereits am Anfang des Dokuments, so hat diese Methode keine Wirkung.

KLASSE:
BaseText

SYNTAX:
GoLeft(Count As Integer, Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Count]: Ein beliebiger Integer-Ausdruck, der angibt, wieviele Zeichen die Schreibmarke nach links bewegt werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird als Standardwert Count = 1 genommen.

[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleThesaurusdialog
Documents.Add("swriter4")
ActiveWindow.Selection.Insert("Das Wort wird thesauriert.")
ActiveWindow.Selection.GoLeft 4
ActiveWindow.Thesaurusdialog
end sub
```

GoRight-Methode

Bewegt die Schreibmarke um die angegebene Anzahl an Zeichen nach rechts.

KLASSE:
BaseText

SYNTAX:
GoRight(Count As Integer, Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Count]: Ein beliebiger Integer-Ausdruck, der angibt, wieviele Zeichen die Schreibmarke nach rechts bewegt werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird als Standardwert Count = 1 genommen.

[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleSelectSentence
REM Dieses Beispiel selektiert den Satz, in dem sich die Einfü-
gemarke befindet.
Const sText1="Das ist ein Satz. "
Const sText2="Dieser Satz wird selektiert. "
Const sText3="Hier kommt das Ende. "
Documents.Add "sWriter4"
Selection.Insert( sText1 & sText2 & sText3)
Selection.JumpToStartOfDoc
Selection.GoRight Len(sText1 + 5 ) REM Cursor in den zweiten
Satz stellen
With ActiveWindow.Selection
.GotoPrevSentence
.GotoNextSentence TRUE   REM mit Selektieren
end with
end sub
```

GoToAnchor-Methode

Springt mit dem Cursor zum Anker des selektierten Objekts.

Klasse:
BaseTextSelection, BaseTextFrame

Syntax:
GoToAnchor()

Beispiel:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
```

```
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

GotoBookmark-Methode

Springt zur angegebenen Textmarke und liefert einen Rückgabewert zurück. Wird die Textmarke gefunden, liefert die Methode den Wert True, andernfalls wird False zurückgegeben.

Siehe auch: JumpToBookmark

KLASSE:
BaseTextSelection, BaseText

SYNTAX:
GotoBookmark(Marke As String, Select As Boolean, ToStart As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
Marke: Name der Textmarke.

[Select]: Ein Boolescher Ausdruck, der angibt, ob der Bereich von der aktuellen Cursorposition bis zur Textmarke markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

[ToStart]: Ein Boolescher Ausdruck, der angibt, ob an den Start (True) oder an das Ende (False) der Textmarke gesprungen werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird an das Ende der Textmarke gesprungen.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
```

```
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

GotoCell-Methode

Setzt den Textcursor in die angegebene Zelle der aktuellen Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
GotoCell(Cell As String, Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
Cell: Zeichenfolge, die den Namen der Zielzelle angibt.

[Select]: Boolescher Ausdruck, der bestimmt, ob der Sprungbereich innerhalb der Tabelle selektiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
```

```
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

GoToEnd-Methode

Bewegt die Schreibmarke innerhalb einer Tabelle an das Ende der aktuellen Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
GoToEnd()

BEISPIEL:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

GoToEndOfColumn-Methode

Bewegt die Schreibmarke innerhalb einer mehrspaltigen Seite an das Ende der aktuellen Spalte.

KLASSE:
BaseText

SYNTAX:
GoToEndOfColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die

sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToEndOfDoc-Methode

Bewegt die Schreibmarke an das Ende des aktuellen Dokuments, der Tabelle oder des Rahmens.

Befindet sich der Cursor innerhalb einer Tabelle oder eines Rahmens, wird zum Ende der Tabelle oder des Rahmens gesprungen. Um den Cursor von hier aus an das Ende des Dokuments zu setzen, verwenden Sie den Befehl JumpToEndOfDoc.

Siehe auch: GoToStartOfDoc

Klasse:
TextPrintPreview, BaseText

Syntax:
GoToEndOfDoc(Select As Boolean)

Rückgabe:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToEndOfLine-Methode

Bewegt die Schreibmarke an das Ende der Zeile.

KLASSE:
BaseText

SYNTAX:
GoToEndOfLine(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

GoToEndOfNextColumn-Methode

Bewegt die Schreibmarke bei einem mehrspaltigen Dokument an das Ende der nächsten Spalte.

KLASSE:
BaseText

SYNTAX:
GoToEndOfNextColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToEndOfNextPage-Methode

Bewegt die Schreibmarke an das Ende der nächsten Seite.

KLASSE:
BaseText

Syntax:
GoToEndOfNextPage(Select As Boolean)

Parameter:
Select: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToEndOfPage-Methode

Bewegt die Schreibmarke an das Ende der Seite.

Klasse:
BaseText

Syntax:
GoToEndOfPage(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToEndOfPara-Methode

Bewegt die Schreibmarke an das Ende des Absatzes.

KLASSE:
BaseText

SYNTAX:
GoToEndOfPara(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
REM Diese Routine dreht die Reihenfolge der Zeichen innerhalb
REM eines Absatzes um.
Sub ExampleInvertParagraph
Const sMessage = "Dieser Absatz wird verdreht"
Dim sText As String, sTemp As String
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.Insert sMessage
With ActiveWindow.Selection
.GotoStartOfPara FALSE
.GotoEndOfPara TRUE
sText = .Value
sText = Trim( sText )
For iCount = 1 To Len(sText)/2
sTemp = Mid( sText, iCount , 1 )
Mid(sText,iCount,1, Mid( sText, Len( sText )-iCount+1 , 1 ) )
Mid(sText,Len(sText) -iCount+1,1,sTemp )
Next iCount
.Insert (sText )
End With
End Sub
```

GoToEndOfPrevColumn-Methode

Bewegt die Schreibmarke an das Ende der vorhergehenden Spalte.

KLASSE:
BaseText

SYNTAX:
GoToEndOfPrevColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
```

```
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToEndOfPrevPage-Methode

Bewegt die Schreibmarke an das Ende der vorhergehenden Seite.

KLASSE:
BaseText

SYNTAX:
GoToEndOfPrevPage(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
```

```
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GotoFrame-Methode

Selektiert den angegebenen Rahmen.

KLASSE:
BaseTextSelection

SYNTAX:
GotoFrame(FrameName As String)

RÜCKGABE:
Boolean

PARAMETER:
FrameName: Zeichenfolgeausdruck, der die Bezeichnung des Rahmens aufnimmt, der selektiert werden soll.

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
```

```
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

GotoNextInputField-Methode

Springt im Dokument zum nächsten Feldbefehl des Typs EINGABEFELD. Bei Anwendung dieser Methode wird automatisch der Dialog EINGABEFELD gestartet. Das ist der Dialog, den Sie erhalten, wenn Sie als Feldbefehl ein Eingabefeld einfügen oder wenn Sie auf einen Feldbefehl des Typs EINGABEFELD klicken.

Siehe auch: GoToPrevInputField

KLASSE:
BaseText

SYNTAX:
GotoNextInputField()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleInputFields
Const sFieldContent_1 = "Eingabefeldinhalt"
Const sFieldContent_2 = "Es wird jetzt Eingabefeld_2 geändert"
Const sFieldContent_3 = "Sie können jetzt Eingabefeld_1 ändern"
Const sFieldContent_4 = "Ändern Sie jetzt Eingabefeld_2"
Const sFieldName_1 = "Eingabefeldname_1"
Const sFieldName_2 = "Eingabefeldname_2"
Documents.Add( "sWriter4" )
Selection.InsertField( 16, 1, sFieldContent_1, sFieldName_1, 0
)
Selection.InsertPara
Selection.InsertField( 16, 1, sFieldContent_2, sFieldName_2, 0
)
Selection.Goleft 1, TRUE
REM Es werden nur die selektierten Felder aufgerufen
Selection.UpdateInputFields TRUE
Selection.GoUp 1, FALSE
Selection.SetCurField( 16, sFieldContent_3, sFieldName_1 )
Selection.GoDown 1, FALSE
Selection.GotoPrevInputField()
Selection.GoDown 1, FALSE
Selection.SetCurField( 16, sFieldContent_4, sFieldName_2 )
```

```
Selection.GoUp 1, FALSE
Selection.GotoNextInputField()
End Sub
```

GotoNextObject-Methode

Selektiert das nächste Zeichenobjekt. Dabei wird ab der Cursorposition die Reihenfolge der Objekte folgendermaßen durchlaufen: Objekte in gleicher Höhe werden von links nach rechts selektiert, bei Objekten auf unterschiedlicher Höhe erfolgt die Selektion von oben nach unten, wobei sich die Methode an der Position der oberen Objektkante orientiert. Wird das Ende des Dokuments erreicht, wird die Selektion automatisch am Dokumentanfang fortgesetzt.

Diese Methode liefert den Wert True zurück, wenn ein Zeichenobjekt gefunden wurde, andernfalls False.

Siehe auch: GotoPrevObject

KLASSE:
BaseTextSelection

SYNTAX:
GotoNextObject()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleSetObjectToGround
Const sMessage_1 = "Erster Commandbutton"
Const sMessage_2 = "Zweiter Commandbutton"
Const sMessage_3 = "Springt zum ersten Commandbutton"
Const sMessage_4 = "Springt zum zweiten Commandbutton"
Dim sFirstControl As String, sSecondControl As String
Documents.Add( "sWriter4" )
Activewindow.InsertVCControl( 2, "5cm", "4cm", "4cm", "1cm")
sFirstControl = Selection.ObjectName
Selection.Escape
Activewindow.InsertVCControl( 2, "5cm", "6cm", "4cm", "1cm")
sSecondControl = Selection.ObjectName
Selection.Escape
Activewindow.Controls(sFirstControl).ControlObject.Caption =
sMessage_1
Activewindow.Controls(sSecondControl).ControlObject.Caption =
sMessage_2
Print sMessage_3
Selection.GotoNextObject
```

```
Print sMessage_4
Selection.GotoPrevObject
End Sub
```

GoToNextPara-Methode

Bewegt den Cursor zum Anfang des nächsten Absatzes. Die Methode liefert den Wert True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToNextPara(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Boolescher Ausdruck, der angibt, ob der Sprungbereich selektiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
REM Diese Routine verbindet den aktuellen mit dem folgenden
Absatz
Sub ExampleGotoEndOfPara
Const sFirstPara = "Dies ist der erste Absatz."
Const sSecondPara = "Dies ist der zweite Absatz."
Documents.Add( "sWriter4" )
Selection.Insert sFirstPara
Selection.InsertPara
Selection.Insert sSecondPara
Selection.InsertPara
Selection.JumpToStartOfDoc
With ActiveWindow.Selection
.GotoEndOfPara FALSE
.GotoNextPara TRUE
.Insert(" ")
End With
End Sub
```

GotoNextPlacemarker-Methode

Setzt den Cursor zum nächsten Feldbefehl-Platzhalter. Die Methode liefert den Wert True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseTextSelection

Syntax:
GotoNextPlacemarker()

Rückgabe:
Boolean

Beispiel:

```
Sub ExamplePlacemarker
Const sPlacemarker_1 = "Platzhalter_1"
Const sPlacemarker_2 = "Platzhalter_2"
Documents.Add( "sWriter4" )
Selection.InsertField( 38, 0, sPlacemarker_1, sPlacemarker_1, 0
)
Selection.InsertPara
Selection.InsertField( 38, 0, sPlacemarker_2, sPlacemarker_2, 0
)
Selection.JumpToStartOfDoc
Selection.GotoNextPlacemarker
Wait 100
Selection.GotoPrevPlacemarker
End Sub
```

GoToNextSentence-Methode

Bewegt die Schreibmarke an den Anfang des nächsten Satzes. Die Methode liefert den Wert True zurück, wenn der Cursor bewegt wurde, sonst False.

Klasse:
BaseText

Syntax:
GoToNextSentence(Select As Boolean)

Rückgabe:
Boolean

Parameter:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleSelectSentence
REM Dieses Beispiel selektiert den Satz in dem sich die Einfü-
gemarke befindet.
Const sText1="Das ist ein Satz. "
Const sText2="Dieser Satz wird selektiert. "
Const sText3="Hier kommt das Ende. "
Documents.Add "sWriter4"
```

```
Selection.Insert( sText1 & sText2 & sText3)
Selection.JumpToStartOfDoc
Selection.GoRight Len(sText1 + 5 ) REM Cursor in den zweiten
Satz stellen
With ActiveWindow.Selection
.GotoPrevSentence
.GotoNextSentence TRUE   REM mit Selektieren
end with
end sub
```

GoToNextWord-Methode

Bewegt die Schreibmarke an den Anfang des nächsten Wortes. Die Methode liefert den Wert True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToNextWord(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
REM Die Procedur färbt den ersten Buchstaben jedes Wortes Rot
Sub ExampleGoToNextWord
Const sMessage = "Dies ist ein Satz"
Documents.Add( "sWriter4" )
Selection.Insert sMessage
With ActiveWindow.Selection
.GoToStartOfPara
Do
.GoRight 1, TRUE
.Font.Color = RGB(255,0,0)
.GoToNextWord
Loop Until .IsEndOfPara
End With
End Sub
```

GotoPage-Methode

Bewegt die Schreibmarke zur angegebenen Seite. Wird kein Parameter übergeben, wird der Navigator aufgerufen.

KLASSE:
BaseTextEditView

SYNTAX:
GotoPage(Page As Long)

PARAMETER:
[Page]: Gibt die Seitennummer der Seite an, zu der die Schreibmarke bewegt werden soll. Dieser Parameter ist optional.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GotoPrevInputField-Methode

Springt zum vorherigen Eingabefeld des Dokuments. Der Dialog EINGABEFELD wird bei Anwendung dieser Methode automatisch gestartet.

Siehe auch: GoToNextInputField

KLASSE:
BaseText

SYNTAX:
GotoPrevInputField()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleInputFields
Const sFieldContent_1 = "Eingabefeldinhalt"
Const sFieldContent_2 = "Es wird jetzt Eingabefeld_2 geändert"
Const sFieldContent_3 = "Sie können jetzt Eingabefeld_1 ändern"
Const sFieldContent_4 = "Ändern Sie jetzt Eingabefeld_2"
Const sFieldName_1 = "Eingabefeldname_1"
Const sFieldName_2 = "Eingabefeldname_2"
Documents.Add( "sWriter4" )
Selection.InsertField( 16, 1, sFieldContent_1, sFieldName_1, 0
)
Selection.InsertPara
Selection.InsertField( 16, 1, sFieldContent_2, sFieldName_2, 0
)
Selection.Goleft 1, TRUE
REM Es werden nur die selektierten Felder aufgerufen
Selection.UpdateInputFields TRUE
Selection.GoUp 1, FALSE
Selection.SetCurField( 16, sFieldContent_3, sFieldName_1 )
Selection.GoDown 1, FALSE
Selection.GotoPrevInputField()
Selection.GoDown 1, FALSE
Selection.SetCurField( 16, sFieldContent_4, sFieldName_2 )
Selection.GoUp 1, FALSE
Selection.GotoNextInputField()
End Sub
```

GotoPrevObject-Methode

Selektiert das vorherige Zeichenobjekt. Dabei wird ab der Cursorposition die Reihenfolge der Objekte folgendermaßen durchlaufen: Objekte in gleicher Höhe werden von rechts nach links selektiert, bei Objekten auf unterschiedlicher Höhe erfolgt die Selektion von unten nach oben, wobei sich die Methode an der Position der oberen Objektkante orientiert. Wird der Anfang

des Dokuments erreicht, wird die Selektion automatisch am Dokumentende fortgesetzt.

Diese Methode liefert den Wert True zurück, wenn ein Zeichenobjekt gefunden wurde, andernfalls False.

Siehe auch: GotoNextObject

KLASSE:
BaseTextSelection

SYNTAX:
GotoPrevObject()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleSetObjectToGround
Const sMessage_1 = "Erster Commandbutton"
Const sMessage_2 = "Zweiter Commandbutton"
Const sMessage_3 = "Springt zum ersten Commandbutton"
Const sMessage_4 = "Springt zum zweiten Commandbutton"
Dim sFirstControl As String, sSecondControl As String
Documents.Add( "sWriter4" )
Activewindow.InsertVCControl( 2, "5cm", "4cm", "4cm", "1cm")
sFirstControl = Selection.ObjectName
Selection.Escape
Activewindow.InsertVCControl( 2, "5cm", "6cm", "4cm", "1cm")
sSecondControl = Selection.ObjectName
Selection.Escape
Activewindow.Controls(sFirstControl).ControlObject.Caption =
sMessage_1
Activewindow.Controls(sSecondControl).ControlObject.Caption =
sMessage_2
Print sMessage_3
Selection.GotoNextObject
Print sMessage_4
Selection.GotoPrevObject
End Sub
```

GoToPrevPara-Methode

Bewegt den Cursor zum vorigen Absatz. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToPrevPara(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Boolescher Ausdruck, der angibt, ob der Sprungbereich selektiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExamplePrevPara
Documents.Add( "sWriter4" )
Selection.Insert "Dies ist der erste Absatz. "
Selection.InsertPara
Selection.Insert "Dies ist der zweite Absatz. "
Selection.GotoPrevPara FALSE
End Sub
```

GotoPrevPlacemarker-Methode

Setzt den Cursor zum vorigen Feldbefehl-Platzhalter. Die Methode liefert den Wert True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseTextSelection

SYNTAX:
GotoPrevPlacemarker()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExamplePlacemarker
Const sPlacemarker_1 = "Platzhalter_1"
Const sPlacemarker_2 = "Platzhalter_2"
Documents.Add( "sWriter4" )
Selection.InsertField( 38, 0, sPlacemarker_1, sPlacemarker_1, 0
)
Selection.InsertPara
Selection.InsertField( 38, 0, sPlacemarker_2, sPlacemarker_2, 0
)
Selection.JumpToStartOfDoc
Selection.GotoNextPlacemarker
Wait 100
Selection.GotoPrevPlacemarker
End Sub
```

GoToPrevSentence-Methode

Bewegt die Schreibmarke an den Anfang des vorangehenden Satzes. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToPrevSentence(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleSelectSentence
REM Dieses Beispiel selektiert den Satz, in dem sich die Einfü-
gemarke befindet.
Const sText1="Das ist ein Satz. "
Const sText2="Dieser Satz wird selektiert. "
Const sText3="Hier kommt das Ende. "
Documents.Add "sWriter4"
Selection.Insert( sText1 & sText2 & sText3)
Selection.JumpToStartOfDoc
Selection.GoRight Len(sText1 + 5 ) REM Cursor in den zweiten
Satz stellen
With ActiveWindow.Selection
.GotoPrevSentence
.GotoNextSentence TRUE   REM mit Selektieren
end with
end sub
```

GoToPrevWord-Methode

Bewegt die Schreibmarke an den Anfang des vorhergehenden Wortes. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToPrevWord(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
REM Die Prozedur stellt sicher, daß der Cursor am Anfang eines
Wortes stehen bleibt
Sub SetCursorToStartOfWord
Const sMessage = "Dies ist ein Satz"
Documents.Add( "sWriter4" )
Selection.Insert sMessage
If Selection.IsStartOfWord = FALSE Then
Selection.GoToPrevWord
End If
End Sub
```

GoToStart-Methode

Bewegt die Schreibmarke innerhalb einer Tabelle an den Anfang der aktuellen Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
GoToStart()

BEISPIEL:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

GoToStartOfColumn-Methode

Bewegt die Schreibmarke an den Anfang der aktuellen Spalte eines mehrspaltigen Dokuments.

KLASSE:
BaseText

SYNTAX:
GoToStartOfColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToStartOfDoc-Methode

Bewegt die Schreibmarke an den Anfang des aktuellen Dokuments, der Tabelle oder des Rahmens.

Befindet sich der Cursor innerhalb einer Tabelle oder eines Rahmens, wird zum Anfang der Tabelle oder des Rahmens gesprungen. Um den Cursor von hier aus an den Anfang des Dokuments zu setzen, verwenden Sie den Befehl JumpToStartOfDoc.

Siehe auch: GoToEndOfDoc

KLASSE:
TextPrintPreview, BaseText

SYNTAX:
GoToStartOfDoc(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
```

```
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToStartOfLine-Methode

Bewegt die Schreibmarke an den Anfang der Zeile.

KLASSE:
BaseText

SYNTAX:
GoToStartOfLine(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
```

```
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

GoToStartOfNextColumn-Methode

Bewegt die Schreibmarke an den Anfang der nächsten Spalte eines mehrspaltigen Dokuments.

KLASSE:
BaseText

SYNTAX:
GoToStartOfNextColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
```

```
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToStartOfNextPage-Methode

Bewegt die Schreibmarke an den Anfang der nächsten Seite.

KLASSE:
BaseText

SYNTAX:
GoToStartOfNextPage(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
```

```
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToStartOfPage-Methode

Bewegt die Schreibmarke an den Anfang der Seite.

KLASSE:
BaseText

SYNTAX:
GoToStartOfPage(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
```

```
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GoToStartOfPara-Methode

Bewegt die Schreibmarke an den Anfang des aktuellen Absatzes. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

KLASSE:
BaseText

SYNTAX:
GoToStartOfPara(Select As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
REM Dieses Beispiel zählt die Sätze innerhalb eines Absatzes
Sub ExampleGotoNextSentence
Const sSentence = "Dies ist ein Satz. Dies ist noch ein Satz."
Const sMessage = "Anzahl der Absätze im aktuellen Absatz : "
Dim iSentenceCount As Integer
Documents.Add( "sWriter4" )
Selection.Insert sSentence
With ActiveWindow.Selection
.GoToStartOfPara
iSentenceCount = 0
Do
iSentenceCount = iSentenceCount + 1
.GotoNextSentence
Loop Until .IsEndOfPara
End With
print sMessage & iSentenceCount
End Sub
```

GoToStartOfPrevColumn-Methode

Bewegt die Schreibmarke an den Anfang der vorhergehenden Spalte eines mehrspaltigen Dokuments.

KLASSE:
BaseText

SYNTAX:
GoToStartOfPrevColumn(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

GoToStartOfPrevPage-Methode

Bewegt die Schreibmarke an den Anfang der vorangehenden Seite.

KLASSE:
BaseText

SYNTAX:
GoToStartOfPrevPage(Select As Boolean)

PARAMETER:
[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

GotoTable-Methode

Setzt den Cursor in die erste Zelle der angegebenen Tabelle. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

Klasse:
BaseTextSelection

Syntax:
GotoTable(TableName As String)

Rückgabe:
Boolean

Parameter:
TableName: Zeichenfolgeausdruck, der die Bezeichnung der Tabelle aufnimmt.

Beispiel:

```
Sub ExampleGotoTable
Const sTableName = "TabellenName"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertNamedTable( sTableName, 2, 2 )
Selection.JumpToStartOfDoc
Selection.GotoTable sTableName
End Sub
```

GoUp-Methode

Bewegt die Schreibmarke um die angegebene Anzahl von Zeilen nach oben. Befindet sich der Cursor bereits in der ersten Zeile des Dokuments, so hat diese Methode keine Wirkung. Die Methode gibt True zurück, wenn der Cursor bewegt wurde, sonst False.

Klasse:
BaseText

Syntax:
GoUp(Count As Integer, Select As Boolean)

Rückgabe:
Boolean

Parameter:
[Count]: Ein beliebiger Integer-Ausdruck, der die Anzahl der Zeilen angibt, um die die Schreibmarke nach oben bewegt werden soll. Dieser Parameter ist optional. Wird er nicht angegeben wird standardmäßig der Wert 1 genommen.

[Select]: Ein Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

Beispiel:

```
Sub ExampleGoUpDown
Const sMessage_1 = "Dies ist die erste Zeile"
Const sMessage_2 = "Dies ist die letzte Zeile"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.GoUp( 1, FALSE )
Selection.Insert sMessage_1
Selection.GoDown 1, FALSE
Selection.Insert sMessage_2
End Sub
```

Graphic-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von Grafiken im Dokument.

KLASSE:
ViewSettings

SYNTAX:
Graphic

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Grafiken im Original (True) oder in Ersatzdarstellung (False) angezeigt werden sollen.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

GraphicDialog-Methode

Ruft bei einer im Dokument markierten Grafik den Dialog GRAFIK auf, in dem die Einstellungen für die selektierte Grafik vorgenommen werden können.

KLASSE:
BaseTextGraphic

SYNTAX:
GraphicDialog()

BEISPIEL:

```
Sub ExampleGraphicDialog
Dim sGrfPath as String
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Selection.GraphicDialog
End Sub
```

GraphicHeight-Eigenschaft

Ermittelt die Original-Höhe der selektierten Grafik.

Die mit der GraphicHeight-Eigenschaft ermittelte Höhe bezieht sich auf die Originalgröße der Grafik. Differiert die aktuelle Größe von der Originalgröße, so kann die aktuelle Höhe über die Height-Eigenschaft des Graphic-Objekts bestimmt werden.

Siehe auch: GraphicWidth, Width.

KLASSE:
BaseTextList, BaseTextGraphic

SYNTAX:
GraphicHeight

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GraphicMirror-Eigenschaft

Bestimmt oder ermittelt die Option zum Spiegeln einer selektierten Grafik.

Klasse:
BaseTextGraphic

Syntax:
GraphicMirror

Rückgabe:
Long

Parameter:
[= MirrorValue]: Long-Integer-Ausdruck, der die Einstellung zum Spiegeln der selektierten Grafik angibt. Folgende Werte sind möglich:

0: Grafik wird nicht gespiegelt.

1: Grafik wird vertikal gespiegelt.

2: Grafik wird horizontal gespiegelt.

3: Grafik wird horizontal und vertikal gespiegelt.

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GraphicName-Eigenschaft

Ermittelt oder setzt den Namen der selektierten Grafik.

Klasse:
BaseTextGraphic

Syntax:
GraphicName

Rückgabe:
String

Parameter:
[= Name]: Zeichenfolge, die den Namen der Grafik angibt. Der zu setzende Name darf nicht bereits für eine andere Grafik im Dokument vergeben sein.

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
```

```
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

Graphics-Eigenschaft

Ermöglicht den Zugriff auf die Collection für Grafiken.

siehe auch: Tables

KLASSE:
BaseTextEditView

SYNTAX:
Graphics

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GraphicURL-Eigenschaft

Bestimmt oder ermittelt den URL einer selektierten Grafik, der als Hyperlink angesprungen wird.

KLASSE:
BaseTextGraphic

SYNTAX:
GraphicURL

RÜCKGABE:
String

PARAMETER:
[= URL]: Zeichenfolge, die den URL der Grafik angibt.

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GraphicWidth-Eigenschaft

Ermittelt die Original-Breite der selektierten Grafik.

Die mit der GraphicWidth-Eigenschaft ermittelte Breite bezieht sich auf die Originalgröße der Grafik. Differiert die aktuelle Größe von der Originalgröße, so kann die aktuelle Breite über die Width-Eigenschaft des TextGraphic-Objekts bestimmt werden.

Siehe auch: GraphicHeight, Height.

KLASSE:
BaseTextList, BaseTextGraphic

SYNTAX:
GraphicWidth

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GrfFilename-Eigenschaft

Bestimmt oder ermittelt Pfad und Dateinamen der Hintergrundgrafik einer Seite oder eines Rahmens, welche als verknüpfte Grafik eingefügt worden ist.

Siehe auch: GrfFiltername, GrfLinkName, GrfPosition.

KLASSE:
TextPage, BaseTextFrame

SYNTAX:
GrfFilename

Rückgabe:
String

Parameter:
[= FileName]: Zeichenfolgenausdruck, der Pfad und Dateinamen der verknüpften Hintergrundgrafik angibt.

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

GrfFiltername-Eigenschaft

Bestimmt oder ermittelt den Filter, der verwendet wird, um eine Grafik als Seitenhintergrund oder als Hintergrund eines Rahmens anzuzeigen.

Siehe auch: GrfFilename, GrfLinkFilterName, GrfPosition.

Klasse:
TextPage, BaseTextFrame

Syntax:
GrfFiltername

Rückgabe:
String

Parameter:
[= FilterName]: Zeichenfolgenausdruck, der den Namen des Filters der Hintergrundgrafik angibt.

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
```

```
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

GrfLinkFilterName-Eigenschaft

Bestimmt oder ermittelt den Filternamen der verknüpften Grafik.

Siehe auch: GrfLinkName, GrfFiltername

KLASSE:
BaseTextGraphic

SYNTAX:
GrfLinkFilterName

RÜCKGABE:
String

PARAMETER:
[= Filter]: Zeichenfolgeausdruck, der den Namen des Filters angibt.

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GrfLinkName-Eigenschaft

Bestimmt oder ermittelt Pfad und Dateinamen der selektierten Grafik, welche als Verknüpfung eingefügt worden ist.

Siehe auch: GrfLinkFilterName, GrfFilename

KLASSE:
BaseTextGraphic

SYNTAX:
GrfLinkName

RÜCKGABE:
String

PARAMETER:
[= Name]: Zeichenfolgeausdruck, der den Pfad und den Dateinamen der verknüpften Grafik angibt.

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

GrfPosition-Eigenschaft

Bestimmt oder ermittelt die Position der verknüpften Hintergrundgrafik des selektierten Objekts.

Siehe auch: GrfFilename, GrfFiltername.

KLASSE:
TextPage, BaseTextFrame

SYNTAX:
GrfPosition

RÜCKGABE:
Long

PARAMETER:
[= Position]: Long-Integer-Ausdruck, der die Position der Hintergrundgrafik angibt. Folgende Werte sind möglich:

0 - Keine Grafik

1 - Oben links

2 - Oben mittig

3 - Oben rechts

4 - Links vertikal zentriert

5 - Horizontal und vertikal zentriert

6 - Rechts vertikal zentriert

7 - Unten links

8 - Unten mittig

9 - Unten rechts

10 - Über die gesamte Fläche

11 - Gekachelt

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
```

```
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

GroupCount-Methode

Liefert die Anzahl von Kontrollfeldern mit gleichem Namen, wobei der Name die durch die Name-Eigenschaft definierte Bezeichnung ist. Gruppen von Kontrollfeldern werden z. B. für RadioButtons benötigt.

KLASSE:
Controls

SYNTAX:
GroupCount(GroupName As String)

RÜCKGABE:
Long

PARAMETER:
GroupName: Zeichenfolge, die den Namen der Kontrollfelder angibt, deren Anzahl ermittelt werden soll.

BEISPIEL:

```
Sub ExampleGroupObject
Const sControlName = "Controlgroup"
Const sControlCaption = "Control "
Const sMessage = "Beschriftung des Controls an der Position "
Dim iCount As Integer
Dim sCaption As String
Documents.Add( "sWriter4" )
For iCount=0 to 4
Activewindow.InsertVCControl( 1, "17cm", iCount+3, "3cm",
"1cm")
Activewindow.ControlObject.Name = sControlName
Activewindow.ControlObject.Caption = sControlCaption & iCount+1
Selection.Escape
Next iCount
For iCount=1 To ActiveWindow.Controls.GroupCount (sControlName)
sCaption = Activewindow.Controls.GroupObject(sControlName,
iCount).Caption
Selection.Escape
Selection.Insert sMessage & iCount & " : "
Selection.Insert sCaption
Selection.InsertPara
Next iCount
End Sub
```

GroupObject-Methode

Diese Methode wird benutzt, um auf das Basic-Objekt eines Kontrollfeldes innerhalb einer Gruppe zugreifen zu können.

Siehe auch: GroupCount.

KLASSE:
Controls

SYNTAX:
GroupObject(GroupName As String, Index As Long)

RÜCKGABE:
Object

PARAMETER:
GroupName: Zeichenfolge, die den Namen der Kontrollfelder angibt, die aufgrund ihrer identischen Bezeichung eine Gruppe bilden.

Index: Gibt den Index des Kontrollfeldes innerhalb der Gruppe an.

BEISPIEL:

```
Sub ExampleGroupObject
Const sControlName = "Controlgroup"
Const sControlCaption = "Control "
Const sMessage = "Beschriftung des Controls an der Position "
Dim iCount As Integer
Dim sCaption As String
Documents.Add( "sWriter4" )
For iCount=0 to 4
Activewindow.InsertVCControl( 1, "17cm", iCount+3, "3cm",
"1cm")
Activewindow.ControlObject.Name = sControlName
Activewindow.ControlObject.Caption = sControlCaption & iCount+1
Selection.Escape
Next iCount
For iCount=1 To ActiveWindow.Controls.GroupCount (sControlName)
sCaption = Activewindow.Controls.GroupObject(sControlName,
iCount).Caption
Selection.Escape
Selection.Insert sMessage & iCount & " : "
Selection.Insert sCaption
Selection.InsertPara
Next iCount
End Sub
```

GroupUndoOnOff-Methode

Faßt alle folgenden Aktionen in einer Rückgängig-Aktion zusammen.

KLASSE:
BaseTextSelection

SYNTAX:
GroupUndoOnOff(On As Boolean)

PARAMETER:
On: TRUE faßt alle folgenden Aktionen in einer Rückgängig-Aktion zusammen. FALSE sorgt dafür, daß alle weiteren Dokumentänderungen wieder als einzelne Rückgängig-Aktionen gemerkt werden.

BEISPIEL:

```
Sub ExampleGroupUndoOnOff
Const sMessage_1 = "Alle folgende Aktionen werden zu einem
Undo-Schritt zusammengefaßt"
Const sMessage_2 = "Die erste zusammengefaßte Aktion"
Const sMessage_3 = "Die zweite zusammengefaßte Aktion"
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GroupUndoOnOff(TRUE)
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.GroupUndoOnOff(FALSE)
Selection.Undo
End Sub
```

Grow-Methode

Vergrößert die Schrift auf die jeweils nächstmögliche Größe.

KLASSE:
TextFont

SYNTAX:
Grow()

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
```

```
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
```

```
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Eigenschaften und Methoden mit "H"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "H" beginnen.

HeadlineRepeat-Eigenschaft

Bestimmt oder ermittelt bei einer im Dokument selektierten Tabelle, ob nach einem Seitenumbruch innerhalb der Tabelle die Tabellenüberschrift wiederholt werden soll.

KLASSE:
BaseTextTable

SYNTAX:
HeadlineRepeat

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Tabellenüberschrift nach einem Seitenumbruch wiederholt wird (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleHeadlineRepeat
Documents.Add "sWriter4"
Selection.InsertTable(37, 2)
Selection.HeadlineRepeat = TRUE
Selection.Insert("Diese Kopfzeile wiederholt sich auf jeder
Seite")
Selection.GotoEnd
Selection.JumpToPrevCell
Selection.Insert("Hier wurde Kopfzeile automatisch einge-
tragen")
End Sub
```

Height-Eigenschaft

Bestimmt oder ermittelt die Höhe des selektierten Objekts. Dabei kann es sich um einen Rahmen, eine Grafik oder ein OLE-Objekt handeln.

Siehe auch: GraphicHeight, Width.

KLASSE:
BaseTextFrame

SYNTAX:
Height

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Höhe des Objekts angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

HeightPercent-Eigenschaft

Bestimmt oder ermittelt die prozentuale Höhe des selektierten Objekts relativ zur Gesamthöhe der Seite.

Haben Sie mit der Height-Eigenschaft einen metrischen Wert für die Höhe des Objekts angegeben, so wird dieser Wert bei Anwendung der HeightPercent-Eigenschaft ignoriert. Er wird erst wieder verwendet, wenn die prozentuale Höhe wieder auf 0 gesetzt wird.

Siehe auch: Height, Width, WidthPercent.

Klasse:
BaseTextFrame

Syntax:
HeightPercent

Rückgabe:
Integer

Parameter:
[= Value]: Integer-Ausdruck, der die Höhe des Objekts in Prozent angibt. Als Prozentwerte werden nur Zahlen von 0 bis 100 akzeptiert.
Wird Value = 255 gesetzt, so erfolgt ein automatischer Abgleich der Höhe.

Beispiel:

```
REM Dieses Beispiel vergrößert einen Rahmen bis auf
REM die Gesamtgröße der Seite
Sub ExampleHeightPercent
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertFrame(2, "0cm", "0cm", "1cm", "1cm")
For iCount = 0 To 100
Selection.HeightPercent = iCount
Selection.WidthPercent = iCount
Next iCount
End Sub
```

HiddenText-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für versteckten Text.

Klasse:
ViewSettings

Syntax:
HiddenText

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob versteckter Text angezeigt werden soll (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

HideSpellMark-Eigenschaft

Bestimmt oder ermittelt, ob bei eingeschalteter Online-Rechtschreibprüfung die Markierung fehlerhafter Wörter ein- oder ausgeblendet wird. Fehlerhaft in diesem Sinne heißt, daß das entsprechende Wort unbekannt ist, da es im verwendeten Wörterbuch nicht gefunden wird.

Siehe auch: SpellOnline

KLASSE:
TextDrawText, StarWriter

SYNTAX:
HideSpellMark

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Markierungen unbekannter Wörter aus- (True) oder eingeblendet (False) werden.

BEISPIEL:

```
sub ExampleSpellOmline
Documents.Add "sWriter4"
Selection.Insert "Quaeramus aliquod non in speciem bonum. "
Activemodule.SpellOnline = True
Activemodule.HideSpellMark = True
Wait 5000
Activemodule.HideSpellMark = False
Selection.Insert " Jetzt wurden die Markierungen für die unbe-
kannten Wörter aktiviert."
end sub
```

HorzOrientation-Eigenschaft

Bestimmt oder ermittelt die Einstellung für die horizontale Ausrichtung des selektierten Objekts.

Siehe auch: XPos, HorzRelation.

KLASSE:
BaseTextFrame

SYNTAX:
HorzOrientation

RÜCKGABE:
Integer

PARAMETER:
[= Orientation]: Integer-Ausdruck, der die Einstellung für die horizontale Ausrichtung des selektierten Objekts angibt. Orientation können die folgenden Werte zugewiesen werden:

0: Positioniert den linken Rand des selektierten Objekts in einer angegebenen Entfernung vom linken Seitenrand, sofern dieser an der Seite verankert ist. Ist das Objekt am Absatz verankert, wird es in einer angegebenen Entfernung vom linken Einzug dieses Absatzes angeordnet. Die Entfernung des Objekts zum Absatz oder zur Seite wird mit der XPos-Eigenschaft bestimmt.

1: Ordnet den rechten Rand des selektierten Objekts rechts am Dokument oder Absatz an. Die genaue Position des Objekts wird über die HorzRelation-Eigenschaft bestimmt.

2: Positioniert das selektierte Objekt in der Mitte des rechten und linken Seitenrands, sofern dieses an der Seite verankert ist. Ist das selektierte Objekt am Absatz verankert, wird es in der Mitte des rechten und linken Einzugs angeordnet.

3: Ordnet den linken Rand des selektierten Objekts links am Dokument oder Absatz an. Die genaue Position des Objekts wird über die HorzRelation-Eigenschaft bestimmt.

BEISPIEL:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebunden Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
```

```
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

HorzRelation-Eigenschaft

Bestimmt oder ermittelt detailliert die Ausrichtung des linken oder rechten Randes des selektierten Objekts.

Siehe auch: HorzOrientation

KLASSE:
BaseTextFrame

SYNTAX:
HorzRelation

RÜCKGABE:
Integer

PARAMETER:
[= FramePos]: Integer-Ausdruck, der die mit der HorzOrientation-Eigenschaft gesetzte Ausrichtung des selektierten Objekts spezifiziert. Der HorzOrientation-Eigenschaft muß entweder der Wert 1 für die Ausrichtung des rechten Objektrandes oder der Wert 3 für die Ausrichtung des linken Objektrands zugewiesen sein. FramePos können die folgenden Einstellungen zugewiesen werden:

0: Positioniert den linken oder rechten Rand des Objekts an der linken oder rechten Papierkante (Verankerung an der Seite) bzw. am linken oder rechten Seitenrand (Verankerung am Absatz).

1: Positioniert den linken oder rechten Rand des Objekts am linken oder rechten Seitenrand (Verankerung an der Seite) bzw. am linken oder rechten Einzug des Absatzes (Verankerung am Absatz).

BEISPIEL:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebundenen Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
```

```
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

HorzRuler-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige des horizontalen Lineals.

KLASSE:
ViewSettings

SYNTAX:
HorzRuler

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das horizontale Lineal ein- (True) oder ausgeblendet (False) werden soll.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

HorzScrollbar-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige der horizontalen Bildlaufleiste.

KLASSE:
ViewSettings

SYNTAX:
HorzScrollbar

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die horizontale Bildlaufleiste ein- (True) oder ausgeblendet (False) werden soll.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Hyphen-Eigenschaft

Bestimmt oder ermittelt, ob die automatische Silbentrennung aktiv ist. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetHyphenZone gesetzte Option zur Silbentrennung abgefragt.

Siehe auch: SetHyphenZone, IsHyphenZone, PageEnd, MinLead, MinTrail, MaxHyphens.

KLASSE:
HyphenZone, ReplaceAttributes, SearchAttributes

SYNTAX:
Hyphen

RÜCKGABE:
Boolean

PARAMETER:
Bei HyphenZone-Objekten:

[= Switch]: Boolescher Ausdruck, der angibt, ob die Silbentrennung aktiviert (True) oder deaktiviert (False) ist.

Beim Suchen und Ersetzen von Attributen:

(nur Lesen)

BEISPIEL:

```
sub ExampleHyphenZone
Dim iCount as Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umbrochen"
Selection.GotoStartOfLine FALSE
```

```
For iCount=1 to 53
Selection.Insert "  "
Next iCount
end sub
```

Hyphenate-Methode

Aktiviert die Silbentrennung von StarOffice. Nach Aufruf wird das Dokument daraufhin durchsucht, ob Wörter getrennt werden können. Wird mindestens ein solches Wort gefunden, wird der Dialog SILBENTRENNUNG geöffnet.

KLASSE:
TextEditView

SYNTAX:
Hyphenate()

BEISPIEL:

```
Sub ExampleHyphenate
Const sPara = "Dieser Absatz enthält ein Wort das getrennt
wird"
Documents.Add( "sWriter4" )
ActiveWindow.PageHeight = "11cm"
ActiveWindow.PageWidth = "11cm"
Selection.Insert sPara
Selection.JumpToStartOfDoc
ActiveWindow.Hyphenate
End Sub
```

HyphenZone-Eigenschaft

HyphenZone enthält die Eigenschaften des Absatzumbruchs. Folgende Eigenschaften gibt es:

Hyphen - automatische Silbentrennung an/aus

PageEnd - Silbentrennung über zwei Seiten

MinLead - Anzahl der Zeichen vor der Silbentrennung am Zeilenende

MinTrail - Anzahl der Zeichen nach der Silbentrennung am Zeilenanfang
MaxHyphens - maximale Anzahl aufeinanderfolgender Trennstellen

KLASSE:
Paragraph

SYNTAX:
HyphenZone

RÜCKGABE:
Object

BEISPIEL:

```
sub ExampleHyphenZone
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umbrochen"
Selection.GotoStartOfLine FALSE
For iCount=1 To 53
Selection.Insert "  "
Next iCount
End Sub
```

Eigenschaften und Methoden mit "I"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "I" beginnen.

ImageMapDialog-Eigenschaft

Liefert oder setzt den Status des ImageMap-Editors.

KLASSE:
BaseTextSelection

SYNTAX:
ImageMapDialog

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der ImageMap-Editor ein- (True) oder ausgeblendet (False) ist.

BEISPIEL:

```
Sub ExampleImageMap
Const sText="Das ist eine ImageMap in HTML"
Const sAltTag1="Nach Hause"
Const sAltTag2="Hier ist es interessant"
Const sAltTag3="Bild von Pamela"
Dim bServer As Boolean, bClient As Boolean REM Typ der Imagemap
Dim iMax As Integer, iCount As Integer
```

```
Dim sLink As String REM Inhalt der Imagemap URLs
Documents.Add("sWriter4")
REM erstelle eine ImageMap per HTML - einfügen
with  ActiveWindow.Selection
.Insert(chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)& sText &
chr(60)&"/BODY"&chr(62))
.Insert(
chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)&chr(60)&"P"&chr(6
2))
.Insert( chr(60)&"MAP NAME=""MAP1"""&chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""77,5,107,32""
HREF=""www.stardivision.de"" TARGET=""_self"" ALT=""" & sAlt-
Tag1 & """ & chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""33,5,66,44""
HREF=""another interesting place"" TARGET=""_self"" ALT=""" &
sAltTag2 & """  &chr(62)
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""1,2,31,34""
HREF=""Pamela Pic"" TARGET=""_self"" ALT=""" & sAltTag3 & """
& chr(62))
.Insert( chr(60)&"MAP"&chr(62))
.Insert( chr(60)&"IMG
SRC=""file:///D|/Office40/gallery/baer.tif"" NAME=""Grafik1""
ALIGN=LEFT WIDTH=108 HEIGHT=63 BORDER=0
USEMAP=""#MAP1"""&chr(62)&chr(60)&"BR CLEAR=LEFT"&chr(62))
.Insert(
chr(60)&"BR"&chr(62)&chr(60)&"BR"&chr(62)&chr(60)&"P"&chr(62)&c
hr(60)&"/BODY"&chr(62))
end with
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut
ActiveDocument.Close "",False
Documents.Add("sWriter4/Web")
ActiveDocument.FramesetSource = True
ActiveWindow.SelectAll
ActiveWindow.Paste
ActiveDocument.FramesetSource = False
REM Die Grafik bearbeiten
with ActiveWindow.Graphics(1)
.GRFLinkName = Application.Pathsettings.Config & "\End.bmp" REM
Grafik neu setzen
.ImageMapDialog=TRUE  REM ImageMap Dialog anzeigen
Wait 500
bServer = .IsServerImageMap
bClient = .IsClientImageMap
iMax =  .ImageMapURLCount
for iCount = 0 to iMax-1
sLink = .ImageMapURLAt(iCount)
MsgBox sLink,64,"ImageMap URL " & iCount+1
next
.ImageMapDialog=False
.Escape
end with
end sub
```

ImageMapURLAt-Methode

Ermittelt die indizierte Adresse, die in einer Client ImageMap vorhanden ist.

Siehe auch: ImageMapURLCount, IsClientImageMap.

KLASSE:
BaseTextFrame

SYNTAX:
ImageMapURLAt(Position As Long)

RÜCKGABE:
String

PARAMETER:
Position: Long-Integer-Ausdruck, der den Index der URL-Adresse angibt, die in der Abfrage ermittelt werden soll. Die erste Adresse ist dabei mit dem Index 0 belegt.

BEISPIEL:

```
Sub ExampleImageMap
Const sText="Das ist eine ImageMap in HTML"
Const sAltTag1="Nach Hause"
Const sAltTag2="Hier ist es interessant"
Const sAltTag3="Bild von Pamela"
Dim bServer As Boolean, bClient As Boolean REM Typ der Imagemap
Dim iMax As Integer, iCount As Integer
Dim sLink As String REM Inhalt der Imagemap URL s
Documents.Add("sWriter4")
REM erstelle eine ImageMap per HTML - einfügen
with  ActiveWindow.Selection
.Insert(chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)& sText &
chr(60)&"/BODY"&chr(62))
.Insert(
chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)&chr(60)&"P"&chr(6
2))
.Insert( chr(60)&"MAP NAME=""MAP1"""&chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""77,5,107,32""
HREF=""www.stardivision.de"" TARGET=""_self"" ALT="""" & sAlt-
Tag1 & """" & chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""33,5,66,44""
HREF=""another interesting place"" TARGET=""_self"" ALT="""" &
sAltTag2 & """"  &chr(62)
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""1,2,31,34""
HREF=""Pamela Pic"" TARGET=""_self"" ALT="""" & sAltTag3 & """"
& chr(62))
.Insert( chr(60)&"MAP"&chr(62))
.Insert( chr(60)&"IMG
SRC=""file:///D|/Office40/gallery/baer.tif"" NAME=""Grafik1""
ALIGN=LEFT WIDTH=108 HEIGHT=63 BORDER=0
USEMAP=""#MAP1"""&chr(62)&chr(60)&"BR CLEAR=LEFT"&chr(62))
```

```
.Insert(
chr(60)&"BR"&chr(62)&chr(60)&"BR"&chr(62)&chr(60)&"P"&chr(62)&c
hr(60)&"/BODY"&chr(62))
end with
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut
ActiveDocument.Close "",False
Documents.Add("sWriter4/Web")
ActiveDocument.FramesetSource = True
ActiveWindow.SelectAll
ActiveWindow.Paste
ActiveDocument.FramesetSource = False
REM Die Grafik bearbeiten
with ActiveWindow.Graphics(1)
.GRFLinkName = Application.Pathsettings.Config & "\End.bmp" REM
Grafik neu setzen
.ImageMapDialog=TRUE  REM ImageMap Dialog anzeigen
Wait 500
bServer = .IsServerImageMap
bClient = .IsClientImageMap
iMax =  .ImageMapURLCount
for iCount = 0 to iMax-1
sLink = .ImageMapURLAt(iCount)
MsgBox sLink,64,"ImageMap URL " & iCount+1
next
.ImageMapDialog=False
.Escape
end with
end sub
```

ImageMapURLCount-Eigenschaft

Zählt die Links, die in einer Client ImageMap vorhanden sind.

Siehe auch: ImageMapURLAt, IsClientImageMap

KLASSE:
BaseTextFrame

SYNTAX:
ImageMapURLCount

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleImageMap
Const sText="Das ist eine ImageMap in HTML"
Const sAltTag1="Nach Hause"
```

```
Const sAltTag2="Hier ist es interessant"
Const sAltTag3="Bild von Pamela"
Dim bServer As Boolean, bClient As Boolean REM Typ der Imagemap
Dim iMax As Integer, iCount As Integer
Dim sLink As String REM Inhalt der Imagemap URLs
Documents.Add("sWriter4")
REM erstelle eine ImageMap per HTML - einfügen
with  ActiveWindow.Selection
.Insert(chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)& sText &
chr(60)&"/BODY"&chr(62))
.Insert(
chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)&chr(60)&"P"&chr(6
2))
.Insert( chr(60)&"MAP NAME=""MAP1"""&chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""77,5,107,32""
HREF=""www.stardivision.de"" TARGET=""_self"" ALT="""" & sAlt-
Tag1 & """" & chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""33,5,66,44""
HREF=""another interesting place"" TARGET=""_self"" ALT="""" &
sAltTag2 & """"  &chr(62)
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""1,2,31,34""
HREF=""Pamela Pic"" TARGET=""_self"" ALT="""" & sAltTag3 & """"
& chr(62))
.Insert( chr(60)&"MAP"&chr(62))
.Insert( chr(60)&"IMG
SRC=""file:///D|/Office40/gallery/baer.tif"" NAME=""Grafik1""
ALIGN=LEFT WIDTH=108 HEIGHT=63 BORDER=0
USEMAP=""#MAP1"""&chr(62)&chr(60)&"BR CLEAR=LEFT"&chr(62))
.Insert(
chr(60)&"BR"&chr(62)&chr(60)&"BR"&chr(62)&chr(60)&"P"&chr(62)&c
hr(60)&"/BODY"&chr(62))
end with
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut
ActiveDocument.Close "",False
Documents.Add("sWriter4/Web")
ActiveDocument.FramesetSource = True
ActiveWindow.SelectAll
ActiveWindow.Paste
ActiveDocument.FramesetSource = False
REM Die Grafik bearbeiten
with ActiveWindow.Graphics(1)
.GRFLinkName = Application.Pathsettings.Config & "\End.bmp" REM
Grafik neu setzen
.ImageMapDialog=TRUE  REM ImageMap Dialog anzeigen
Wait 500
bServer = .IsServerImageMap
bClient = .IsClientImageMap
iMax =  .ImageMapURLCount
for iCount = 0 to iMax-1
sLink = .ImageMapURLAt(iCount)
MsgBox sLink,64,"ImageMap URL " & iCount+1
next
.ImageMapDialog=False
```

```
.Escape
end with
end sub
```

IncrementIndent-Methode

Erhöht den Absatzeinzug um einen festgelegten Wert. Dabei wird der unter EXTRAS - OPTIONEN - TEXTDOKUMENT - LAYOUT als Tabulatorenabstand definierte Standardwert genommen.

KLASSE:
Paragraph

SYNTAX:
IncrementIndent()

BEISPIEL:

```
Sub ExampleParagraphIndent
Documents.Add("sWriter4")
Wait 1
With Selection.Paragraph
   Selection.Insert("Einrückung eines Textes erhöhen")
   Wait 500
   .IncrementIndent
   Wait 500
   .IncrementIndent
   Wait 500
   .IncrementIndent
   Wait 500
   .SelectText
   Selection.Insert("Einrückung eines Textes verringern")
   Wait 500
   .DecrementIndent
   Wait 500
   .DecrementIndent
   Wait 500
   .DecrementIndent
   Wait 500
End With
```

IncrementLevel-Methode

Stuft eine Aufzählung oder Numerierung um eine Ebene nach oben.

KLASSE:
BaseTextList

SYNTAX:
IncrementLevel()

Beispiel:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

IncrementSubLevels-Methode

Stuft eine Aufzählung oder Numerierung einschließlich ihrer Unterpunkte um eine Ebene nach oben.

Klasse:
BaseTextList

Syntax:
IncrementSubLevels()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

IndentFirstLine-Eigenschaft

Bestimmt oder ermittelt den Einzug der ersten Zeile des selektierten Absatzes. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLeftRightParaMargin gesetzte Erstzeileneinzug abgefragt.

Eine Abfrage der IndentFirstLine-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLeftRightParaMargin bestimmt wurde, daß nach einem bestimmten Absatzeinzug gesucht werden soll, d. h. IsLeftRightParaMargin = 2.

Siehe auch (für SearchAttributes, ReplaceAttributes):

SetLeftRightParaMargin, IsLeftRightParaMargin, LeftMargin, RightMargin.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
IndentFirstLine

RÜCKGABE:
Integer

PARAMETER:
[= DistUnit]: Bestimmt den Einzug der ersten Zeile des selektierten Absatzes. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleIndentFirstLine
Documents.Add "sWriter4"
With ActiveWindow.Selection.Paragraph
.IndentFirstLine = "-1cm"
.LeftMargin = "1cm"
End With
Selection.Insert "Die erste Zeile ist weiter eingerückt"
Selection.InsertLinebreak
Selection.Insert "als die zweite Zeile"
End Sub
```

Index-Eigenschaft

Liefert den Index des selektierten Objekts oder der selektierten Tabelle innerhalb der jeweiligen Collection.

KLASSE:
BaseTextFrame, BaseTextTable

SYNTAX:
Index

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleIndex
Dim iCount As Integer
Dim sFrmName As String
Documents.Add "sWriter4"
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Es werden alle Rahmen auf dem Dokument angesprungen
For iCount=1 To ActiveWindow.Frames.Count
sFrmName = Activewindow.Frames(iCount).FrameName
Print "Der Rahmenname von Rahmen " & iCount & " lautet : " &
sFrmName
If Selection.Index <> iCount Then
Print "Fehler ! Index stimmt nicht mit Collectionindex überein"
End If
next iCount
End Sub
```

IndexEntry-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für Verzeichniseinträge.

KLASSE:
ViewSettings

SYNTAX:
IndexEntry

RÜCKGABE:
Boolean

PARAMETER:
[Switch]: Boolescher Ausdruck, der angibt, ob Verzeichniseinträge angezeigt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

IndexEntryDialog-Methode

Ruft den Dialog zum Bearbeiten der Verzeichniseinträge auf.

Klasse:
Text

Syntax:
IndexEntryDialog()

Beispiel:

```
Sub ExampleIndexEntryDialog
Const sTOCEntry = "Inhaltsverzeichnis Eintrag"
Const sTOC = "Inhaltsverzeichnis"
Documents.Add( "sWriter4" )
Selection.InsertTOCEntry( 1, sTOCEntry )
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sTOC )
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfLine FALSE
Selection.IndexEntryDialog
End Sub
```

IndexMarkToIndex-Methode

Springt von der Benutzer- oder Stichwort-Verzeichnismarke im Dokument zum Anfang des zugehörigen Verzeichnisses.

KLASSE:
Text

SYNTAX:
IndexMarkToIndex()

BEISPIEL:

```
Sub ExampleIndexMarkToIndex
Const sHeadingName = "Überschrift "
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sNewIndex)
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfLine FALSE
Selection.IndexMarkToIndex
End Sub
```

InsDelRowCol-Methode

Fügt in eine Tabelle eine Zeile, eine Spalte oder eine Zelle ein oder löscht diese. Die Art und Weise, wie die Tabelle bearbeitet wird, hängt von den übergebenen Parametern ab. Insbesondere bestimmt der letzte Übergabeparameter, ob ganze Zeilen oder Spalten betrachtet werden oder ob nur eine einzelne Zelle bearbeitet wird.

Wenn der Befehl erfolgreich ausgeführt werden konnte, wird der Wert True zurückgeliefert, andernfalls False.

KLASSE:
BaseTextTable

SYNTAX:
InsDelRowCol(Row As Boolean, Left_Top As Boolean, Delete As Boolean, CellOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
Row: Boolescher Ausdruck der angibt, ob sich die Operation auf eine Zeile (True) oder eine Spalte (False) bezieht oder ob eine Zelle horizontal (True) oder vertikal (False) bearbeitet werden soll.

[Left_Top]: Boolescher Ausdruck, der angibt, ob eine Spalte (oder Zelle) links von der aktuellen Zelle bzw. eine Zeile (oder Zelle) oberhalb der aktuellen Zelle (True) eingefügt oder gelöscht werden soll oder ob eine Spalte (oder Zelle) rechts von der aktuellen Zelle bzw. eine Zeile (oder Zelle) unterhalb der aktuellen Zelle (False) eingefügt oder gelöscht werden soll.

Bezieht sich die Operation nicht auf eine gesamte Zeile oder Spalte, sondern nur auf eine Zelle (CellOnly = True), dann gibt Left_Top zusammen mit dem Parameter Row die Position an, an der die aktuelle Zelle bearbeitet werden soll. Wird eine Zelle horizontal bearbeitet (Row = True), dann gibt Left_Top an, ob oberhalb (True) oder unterhalb (False) der aktuellen Zelle eine Zelle eingefügt oder gelöscht werden soll, wird sie vertikal bearbeitet (Row = False), bestimmt Left_Top, ob das Einfügen oder Löschen einer Zelle links (True) oder rechts (False) neben der aktuellen Zelle erfolgt.

Dieser Parameter ist optional. Wird er nicht angegeben, wird Left_Top = True genommen.

[Delete]: Boolescher Ausdruck, der angibt, ob eine Zeile, Spalte oder Zelle gelöscht (True) oder eingefügt (False) werden soll.

Dieser Parameter ist optional. Wird er nicht angegeben, wird Delete = True genommen.

[CellOnly]: Boolescher Ausdruck, der bestimmt, ob sich die Operation nur auf eine Zelle (True) oder auf eine gesamte Zeile bzw. Spalte (False) bezieht.

Dieser Parameter ist optional. Wird er nicht angegeben, wird CellOnly = False genommen.

Beispiel:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

Insert-Methode

Fügt den angegebenen Text an der Position der Schreibmarke ein.

Siehe auch: ExpandGlossary, NewGlossary, SetActGlossaryGroup.

Klasse:
BaseText, BaseTextFrame

Syntax:
Insert(StringName As String)

Parameter:
StringName: Ein beliebiger Zeichenfolgeausdruck, der den einzufügenden Text angibt.

Beispiel:

```
sub ExampleFrameForBack
Dim s1 as String , s2 as String , s3 as String
Documents.add("sWriter4")
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "1cm", "1cm" , "5cm" , "4cm" )
s1 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s1)
Selection.Insert "Dieser Rahmen ist der erste. "
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "2cm", "2cm" , "5cm" , "4cm" )
s2 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s2)
Selection.Insert " Dieser Rahmen ist der zweite. "
Wait 2000
Selection.GotoFrame(s1)
```

```
Selection.ObjectBackOne
Selection.Insert " Der erste Rahmen liegt nun vor dem zweiten.
"
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "3cm", "3cm" , "5cm" , "4cm" )
s3 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s3)
Selection.Insert "Das ist der dritte Rahmen. "
Wait 2000
Selection.GotoFrame(s3)
Selection.ObjectForwardOne
Selection.Insert "Der dritte Rahmen liegt hinter dem zweiten. "
Selection.GotoAnchor
end sub
```

InsertAnnotation-Methode

Fügt an der Position der Schreibmarke eine Notiz ein. Wird kein Parameter übergeben, ruft diese Methode zum Einfügen einer Notiz den Dialog NOTIZ BEARBEITEN auf.

KLASSE:
BaseText

SYNTAX:
InsertAnnotation(Note As String)

PARAMETER:
[Note]: Zeichenfolge, die die einzufügende Notiz aufnimmt. Dieser Parameter ist optional.

BEISPIEL:

```
Sub ExampleInsertAnnotation
Const sMessage = "Hier steht eine Notiz "
Const sAnnotation = "Ich bin eine Notiz"
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.InsertAnnotation( sAnnotation )
End Sub
```

InsertApplet-Methode

Ruft den Dialog APPLET EINFÜGEN auf.

KLASSE:
BaseText

SYNTAX:
InsertApplet()

Beispiel:

```
Sub ExampleInsertApplet
Documents.Add "sWriter4"
Selection.Insert("Das ist der Applet-Einfügen-Dialog")
Selection.InsertApplet()
End Sub
```

InsertAuthorField-Methode

Fügt den Feldbefehl "Autor" an der aktuellen Cursorpostion ein.

Klasse:
BaseText

Syntax:
InsertAuthorField()

Beispiel:

```
Sub ExampleInsertAuthorField
Const sTitle = "Titel des Dokumentes"
Const sTopicField = "Thema des Dokumentes"
Const sMessage_1 = "Der Autor des Dokumentes : "
Const sMessage_2 = "Der Titel des Dokumentes : "
Const sMessage_3 = "Das Thema des Dokumentes : "
Documents.Add( "sWriter4" )
ActiveDocument.DocumentInfo.Title = sTitle
ActiveDocument.DocumentInfo.Subject = sTopicField
Selection.Insert sMessage_1
Selection.InsertAuthorField
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTitleField
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertTopicField
End Sub
```

InsertBookmark-Methode

Fügt eine Textmarke an der Position der Schreibmarke ein. Wird kein Parameter übergeben, ruft diese Methode den Dialog Textmarke einfügen auf.

Klasse:
BaseText

Syntax:
InsertBookmark(Name As String)

PARAMETER:
[Name]: Ein beliebiger Zeichenfolgeausdruck, der die Bezeichnung der Textmarke aufnimmt. Dieser Parameter ist optional.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

InsertBreak-Methode

Ruft den Dialog zum Einfügen eines manuellen Umbruchs auf.

Klasse:
BaseText

Syntax:
InsertBreak()

Beispiel:

```
Sub ExampleInsertBreak
Documents.Add "sWriter4"
Selection.Insert("So können Sie Umbrüche einfügen.")
ActiveWindow.InsertBreak
End Sub
```

InsertCaptionDialog-Methode

Ruft den Dialog Beschriftung einfügen auf.

Klasse:
TextEditView

Syntax:
InsertCaptionDialog()

Beispiel:

```
sub ExampleInsertCaptionDialog
Documents.Add("swriter4")
REM leeren Absatz sonst kann man die Tabelle mit der Beschrif-
tung niocht selektieren
Selection.InsertPara
Selection.InsertTable(3, 2)
Selection.Alignment=2
Selection.Width = "10cm"
REM Jetzt die Beschriftung einfügen
Selection.InsertCaptionDialog
ActiveWindow.JumpToStartOfDoc
REM jetzt alls in einen Rahmen packen.
ActiveWindow.SelectAll
Selection.InsertFrame(2,"4cm","5cm","8cm","5cm")
End Sub
```

InsertColumnBreak-Methode

Fügt an der Position der Schreibmarke einen Spaltenumbruch ein. Diese Methode ist nur dann sinnvoll anzuwenden, wenn in das Dokument Spalten eingefügt wurden.

KLASSE:
BaseText

SYNTAX:
InsertColumnBreak()

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

InsertColumns-Methode

Fügt an der Position der Schreibmarke die angegebene Anzahl an Spalten in die aktuelle Tabelle ein.

KLASSE:
BaseTextTable

SYNTAX:
InsertColumns(Count As Long, After As Boolean)

PARAMETER:
Count: Ein beliebiger Long-Integer-Ausdruck, der die Anzahl der Spalten angibt, die in die Tabelle eingefügt werden sollen.

[After]: Boolescher Ausdruck, der bestimmt, ob die Spalten hinter (True) oder vor (False) der Position der Schreibmarke eingefügt werden sollen. Dieser Parameter ist optional. Wird er nicht angegeben, werden die Spalten hinter der Cursorposition eingefügt.

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

InsertDateField-Methode

Fügt den Feldbefehl "Datum/fix" an der aktuellen Cursorpostion ein.

KLASSE:
BaseText

SYNTAX:
InsertDateField()

BEISPIEL:

```
Sub ExampleFileDialog
Documents.Add("swriter4")
with ActiveWindow.Selection
.Insert("Die Uhrzeit ist ")
.InsertTimeField()
.InsertPara
.Insert("Das Datum ist ")
.InsertDateField()
.GoLeft 1
.FieldDialog
end with
end sub
```

InsertDBField-Methode

Fügt ein Datenbankfeld in das Dokument ein. War das Einfügen des Feldes erfolgreich, wird der Wert True zurückgeliefert, ansonsten False.

KLASSE:
BaseText

SYNTAX:
InsertDBField(Type As Long, DBName As String, TableName As String, ColumnName As String, Content As String, Format As Long)

RÜCKGABE:
Boolean

PARAMETER:
Type: Long-Integer-Ausdruck, der angibt, welches Feld eingefügt werden soll. Folgende Werte sind möglich:

3: Datenbankname

19: Serienbrieffeld

24: Nächster Datensatz

25: Beliebiger Datensatz

26: Datensatznummer

DBName: Name der Datenbank, in der sich das einzufügende Feld befindet.

TableName: Name der Tabelle, die das einzufügende Feld enthält.

ColumnName: Name des Feldes (Spaltenname), welches eingefügt werden soll.

Content: Zeichenfolge, die den Bedingungstext angibt (nur für "Nächster Datensatz" und "Beliebiger Datensatz" möglich).

Format: Numerisches Darstellungsformat (nur für "Serienbrieffeld" und "Datensatznummer" möglich).

Formatwerte für "Datensatznummer":

0: A B C

1: a b c

2: Römisch (I II III)

3: Römisch (i ii iii)

4: Arabisch (1 2 3)

Formatwerte für "Serienbrieffeld":

0: System

1: Währung

2: Text

3: ####

4: ####,#

5: ####,##

6: #.###,#

7: #.###,##

8: #.###,###

9: DM ####

10: DM #.###,##

11: DM #.###,--

12: #### DM

13: #.###,## DM

14: #.###,-- DM

15: #### %

16: ####,## %

Beispiel:

```
Sub ExampleDataUpdateFields
Const sMessage_1 = "Verwendete Datenbank Tabelle : "
Const sMessage_2 = "Folgende Datenbank ist am Dokument ange-
meldet : "
Dim sDataName As String, sTableName As String, sSQL As String
Dim iCount As Integer
Dim vStatus As Variant
Documents.Add("sWriter")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
vStatus = ActiveWindow.OpenTableConnection(sTableName)
Selection.Insert sMessage_1 & sTableName
Selection.InsertPara
for iCount = 1 To ActiveWindow.DataColumnNumber
Selection.Insert ActiveWindow.DataColumnName(iCount) & " : "
```

```
Selection.InsertDBField(TYP_DBFLD, sDataName, sTableName, Acti-
veWindow.DataColumnName(iCount), "", 1)
Selection.InsertPara
Next iCount
sSQL = "SELECT * FROM " & sTableName
REM Es wird der zweite Dateensatz in das Dokument eingemischt
Selection.DataUpdateFields(sDataName, sTableName, sSQL, 2)
REM Alle Datensätze der Tabelle auf dem Standarddrucker
ausdrucken
Selection.DataUpdatePrint(sDataName, sTableName, sSQL, "",
FALSE)
ActiveWindow.CloseTableConnection(sTableName)
ActiveWindow.CloseDatabaseConnection(sDataName)
End Sub
```

InsertDocument-Methode

Fügt ein Dokument mit dem angegebenen Textformat in ein Globaldokument ein.

KLASSE:
GlobalContents

SYNTAX:
InsertDocument(Position As Long, FileName As Long)

RÜCKGABE:
Boolean

PARAMETER:
Position: Index, der die Position innerhalb der Globaldokument-Collection bestimmt, an der das Dokument eingefügt werden soll. Der Index beginnt bei 1.

FileName: Ein Zeichenfolgenausdruck, der den Pfad und den Dateinamen des einzufügenden Dokumentes angibt. Es können Dateien im Format SDW, HTML und TXT verwendet werden.

BEISPIEL:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\global.sgl"
Const sClientDocName = "\client.sdw"
Const sMessage_1 = "Der Titel des Globaldokumentbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentbereichs : "
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
```

```
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
```

```
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

InsertEnvelop-Methode

Öffnet den Dialog BRIEFUMSCHLAG, mit dem ein Briefumschlag gestaltet, ausgedruckt oder an der Cursorposition in ein Dokument eingefügt werden kann.

KLASSE:
StarWriter

SYNTAX:
InsertEnvelop()

BEISPIEL:

```
Sub ExampleInsertEnvelop
Documents.Add "sWriter4"
Selection.Insert("So können Sie Briefumschläge definieren: ")
StarWriter.InsertEnvelop
End Sub
```

InsertField-Methode

Fügt an der aktuellen Position der Schreibmarke ein Feld ein. Wird kein Parameter übergeben, ruft diese Methode den Dialog FELDBEFEHL EINFÜGEN auf.

KLASSE:
BaseText

SYNTAX:
InsertField(Type As Long, SubType As Long, Name As String, Content As String, Format As Long)

RÜCKGABE:
Boolean

PARAMETER:
[Type]: Long-Integer-Ausdruck, der angibt, welches Feld eingefügt werden

soll. Sollen Dokumentinfo-Felder eingefügt werden, werden diese mit SubType angegeben. Type bestimmt in diesem Fall lediglich den Feldbereich, also DOKUMENTINFO. Die folgenden Felder stehen zur Auswahl:

0: Datum

1: Uhrzeit

2: Dateiname

3: Datenbankname

4: Kapitel

5: Seitennummer

6: Statistik

7: Autor

8: Variable setzen

9: Variable anzeigen

10: Formel einfügen

11: Versteckter Text

12: Referenz setzen

13: Referenz einfügen

14: DDE-Feld

15: Makro ausführen

16: Eingabefeld

17: Versteckter Absatz

18: Dokumentinfo-Bereich (Die einzufügenden Felder werden mit SubType übergeben.)

19: Serienbrieffeld

20: Benutzerdefiniertes Feld

21: Notizfeld

22: Dokumentvorlage

23: Nummernkreis

24: Nächster Datensatz

25: Beliebiger Datensatz

26: Datensatznummer

27: Bedingter Text

28: Nächste Seite

29: Vorherige Seite

30: Absender

35: Seitenvariable setzen

36: Seitenvariable anzeigen

38: Platzhalter

39: Skriptfeld

Dieser Parameter ist optional. Wird er nicht angegeben, wird standardmäßig ein Datumsfeld eingefügt (Type = 0).

[SubType]: Long-Integer-Ausdruck, der das einzufügende Feld spezifiziert. SubType bestimmt vorwiegend eine der in dem Listenfeld AUSWAHL des Feldbefehl-Dialoges aufgeführten Funktionen oder Darstellungen des aktuellen Feldes in der Reihenfolge ihrer Anordnung, wobei der Wert 0 den ersten Eintrag bezeichnet. Überdies spezifiziert SubType auch die Felder des mit Type übergebenen Dokumentinfo-Bereiches, welche über die folgenden Werte bestimmt werden können (Type wurde mit dem Wert 18 für den Bereich DOKUMENTINFO übergeben):

0: Titel

1: Thema

2: Schlüsselwörter

3: Beschreibung

4: Info 0

5: Info 1

6: Info 2

7: Info 3

8: Erzeugung

9: Änderung

10: Letzter Druck

11: Dokumentnummer

12: Bearbeitungszeit

Für Felder vom Typ ABSENDER (Type = 30) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - Firma

1 - Vorname

2 - Name

3 - Kürzel

4 - Straße

5 - Land

6 - PLZ

7 - Ort

8 - Titel

9 - Position

10 - Telefon Privat

11 - Telefon Geschäft

12 - FAX

13 - E-Mail

14 - Staat

Wird ein Feld vom Typ ABSENDER eingefügt, dann haben die Übergabeparameter Name, Content und Format keinerlei Wirkung.

Für Felder vom Typ DATUM (Type = 0) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - Datum fix

1 - Datum variabel

Für Felder vom Typ UHRZEIT (Type = 1) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - Uhrzeit fix

1 - Uhrzeit variabel

Wird ein Feld vom Typ DATUM oder vom Typ UHRZEIT eingefügt, dann haben die Übergabeparameter Name und Content keinerlei Wirkung. Über den Parameter Format kann man jedoch die Darstellung des Datums oder der Uhrzeit beeinflussen.

Für Felder vom Typ STATISTIK (Type = 6) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - Seiten

1 - Absätze

2 - Wörter

3 - Zeichen

4 - Tabellen

5 - Grafiken

6 - Objekte

Mit Feldbefehlen vom Typ Statistik werden nur Zahlen in das Dokument eingefügt, eine Übergabe der Parameter Name und Content ist hier ohne Wirkung. Über den Formatparameter kann man die Darstellung der Werte festlegen und ändern.

Für Felder vom Typ NUMMERNKREIS (Type = 23) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - Abbildung

1 - Tabelle

2 - Text

Alle neu hinzugekommenen Nummernkreise, die selbst definiert wurden, bekommen jeweils den nächstfolgenden Index, der noch nicht belegt ist.

Beim Einfügen eines Nummernkreises, wird stets der aktuelle Wert um 1 erhöht, wenn mit dem Parameter Content nicht ein anderer Feldinhalt definiert wird. Wird der Wert des einzufügenden Nummernkreises mit dem Parameter Content übergeben, dann wird dieser übergebene Wert zum neuen aktuellen Wert des Nummernkreises, und beim nächsten Einfügen eines solchen Nummernkreises wird dann der neue aktuelle Wert um 1 erhöht.

Beim Einfügen eines Feldbefehls vom Typ NUMMERNKREIS ist zu beachten, daß der Parameter Name unbedingt angegeben werden muß.

Für Felder vom Typ SEITENVARIABLE SETZEN (Type = 35) gibt es für SubType folgende Auswahlmöglichkeiten:

0 - aus

1 - an

Mit diesen Optionen ist es möglich, eine Zählung in den Seitenvariablen zu steuern. Mit SubType = 0 wird eine Zählung ausgeschaltet und erst dann wieder aufgenommen, wenn eine Seitenvariable mit SubType = 1 eingefügt wird. Alle Felder vom Typ SEITENVARIABLE ANZEIGEN (Type = 36), die zwischen einem Feld SEITENVARIABLE SETZEN im Zustand "aus" und einem Feld SEITENVARIABLE SETZEN im Zustand "an" stehen, haben keinerlei Inhalt und erscheinen auf dem Bildschirm nur als unterlegtes Leerzeichen (wenn die Option zur Hinterlegung von Feldern unter EXTRAS - OPTIONEN - TEXTDOKUMENT... - INHALTE - HINTERLEGUNG selektiert ist). Wird das Dokument gedruckt, dann erscheint im Ausdruck an dieser Stelle nichts.

Der Parameter SubType ist optional. Wird er nicht angegeben, wird als Standardwert SubType = 0 gesetzt. Die Definition des Subtyps wird dann durch den jeweiligen Wert des Parameters Type bestimmt.

Name: Zeichenfolgenausdruck, der die Bezeichnung des einzufügenden Feldes, eines benutzerdefiniertes Feldes oder die Bedingung eines Funktionsfeldes angibt. Die Feldbezeichnungen entnehmen Sie bitte dem Listenfeld AUSWAHL oder dem Feld NAME im Dialog FELDBEFEHL EINFÜGEN.

Beim Einfügen von Datenbankfeldern muß die Bezeichnung des einzufügenden Feldes folgendermaßen eingegeben werden:
Name = DBName.Tabellenname.Feldname
DBName: Name der Datenbank, in der sich das einzufügende Feld befindet.

Tabellenname: Name der Tabelle, die das einzufügende Feld enthält.

Feldname: Name des Feldes (Spaltenname), welches eingefügt werden soll. Bei einigen Datenbankfeldern müssen Sie statt des Feldnamens den Bedingungstext eingeben.

Als Zeichen zwischen Datenbankname, Tabellenname und Feldname ist der nach unten versetzte Trennstrich zu verwenden.

[Content]: Zeichenfolgeausdruck, der die vom Feldtyp abhängigen Feldinhalte oder Feldwerte aufnimmt.

Dieser Parameter ist optional. Wird er nicht angegeben, wird standardmäßig ein Leerstring übergeben (Content = "").

[Format]: Long-Integer-Ausdruck, der die gewünschte Darstellungsform des aktuellen Feldtyps angibt. Format bestimmt eine der in dem Listenfeld FORMAT des Feldbefehl-Dialoges aufgeführten Darstellungsformen des aktuellen Feldes in der Reihenfolge ihrer Anordnung, wobei der Wert 0 den ersten Eintrag bezeichnet.

Dieser Parameter ist optional. Wird er nicht angegeben, wird als Standardwert Format = 0 genommen.

BEISPIEL:

```
Sub ExampleUserFieldCount
Const sMessage_1 = "Erstes Benutzerfeld : "
Const sMessage_2 = "Zweites Benutzerfeld : "
Const sMessage_3 = "Anzahl der Benutzerfelder auf dem Dokument
: "
Const sMessage_4 = "Der Inhalt vom ersten Benutzerfeld : "
Const sUserName_1 = "Benutzerfeld_1"
Const sUserName_2 = "Benutzerfeld_2"
Const sUserContent_1 = "Benutzerfeld 1"
Const sUserContent_2 = "Benutzerfeld 2"
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_1, sUserCon-
tent_1, 0 )
Selection.InsertPara
Selection.Insert( sMessage_2 )
Selection.InsertField( TYP_USERFLD, 1, sUserName_2, sUserCon-
tent_2, 0 )
Selection.InsertPara
Selection.Insert sMessage_4
Selection.Insert Selection.GetUserFieldValue( sUserName_1 )
Selection.InsertPara
iCount = ActiveWindow.FieldCount( TYP_USERFLD )
MsgBox sMessage_3 & iCount, 64, ""
End Sub
```

InsertFile-Methode

Fügt an der Position der Schreibmarke ein Dokument mit dem angegebenen Textformat ein. Werden keine Parameter übergeben, ruft diese Methode den Dialog DATEI EINFÜGEN auf.

KLASSE:
BaseText

SYNTAX:
InsertFile(Name As String, Filter As String)

RÜCKGABE:
Boolean

PARAMETER:
[Name]: Ein Zeichenfolgenausdruck, der den Pfad und den Dateinamen des einzufügenden Dokumentes angibt. Wird dieser Parameter nicht übergeben, wird der Dialog DATEI EINFÜGEN geöffnet.

[Filter]: Ein Zeichenfolgenausdruck, über den der Textfilter angegeben werden kann, mit dem das einzufügende Dokument eingelesen werden soll. Die Namen der Filter können über den Dialog DATEI ÖFFNEN ermittelt werden. Die Bezeichnung des Textfilters entspricht dem jeweiligen Eintrag im Listenfeld DATEITYP, diese kann optional auch ohne den Modulnamen angegeben werden.

Dieser Parameter ist optional. Wird ein Leerstring oder kein Parameter übergeben oder wird der Textfilter nicht korrekt angegeben, dann erfolgt eine automatische Filtererkennung.

BEISPIEL:

```
Sub ExampleInsertFile
Const sFileName = "Test.sdw"
Const sMessage_1 = "Dieser Absatz wurde aus einem Dokument
eingefügt"
Const sMessage_2 = "Hier wird ein Absatz aus einem anderen
Dokument eingefügt"
Dim sWPath As String
sWPath = Application.Pathsettings.Work & sFileName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Activedocument.Close TRUE, sWPath
Documents.Add( "sWriter4" )
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertFile(sWPath, "swriter4: StarWriter 4.0")
End Sub
```

InsertFootnote-Methode

Fügt auf der aktuellen Seite eine Fußnote ein. Wird kein Parameter übergeben, dann wird der Dialog FUßNOTE EINFÜGEN geöffnet.

KLASSE:
Text

SYNTAX:
InsertFootnote(Text As String)

PARAMETER:
[Text]: Ein Zeichenfolgenausdruck, der die Bezeichnung der einzufügenden Fußnote aufnimmt. Wird ein Leerstring angegeben, so wird die Fußnote mit einer fortlaufenden Nummer versehen.

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert(( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

InsertFootnoteDirect-Methode

Fügt an der Cursorposition eine Fußnote ein. Der Cursor steht danach im Fußnotenbereich am Anfang der neuen Fußnote.

KLASSE:
Text

SYNTAX:
InsertFootnoteDirect()

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert(( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
```

```
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

InsertFrame-Methode

Fügt einen Rahmen in das Dokument ein. Ohne Übergabeparameter wird der Dialog RAHMEN zum Einfügen eines Rahmens aufgerufen.

KLASSE:
BaseText

SYNTAX:
InsertFrame(AnchorType As Long, X As Long, Y As Long, Width As Long, Height As Long)

PARAMETER:
[AnchorType]: Long-Integer-Ausdruck, der den Ankerpunkt angibt. Folgende Werte können AnchorType zugewiesen werden:

0: Setzt den Ankerpunkt an dem Absatz, auf dem sich die Schreibmarke gegenwärtig befindet.

1: Setzt den Ankerpunkt an dem Zeichen, auf dem sich die Schreibmarke befindet.

2: Setzt den Ankerpunkt an die aktuelle Seite.

Dieser Parameter ist optional. Wird er nicht angegeben, wird standardmäßig AnchorType = 0 gesetzt.

[X]: Long-Integer-Ausdruck, der die horizontale Position des Rahmens, gemessen vom Ankerpunkt, aufnimmt. Dieser Parameter ist optional. Wird er nicht angegeben, wird der Rahmen zentriert eingefügt, und die horizontale Position ergibt sich dann automatisch aus der mit dem Parameter Width übergebenen Breite.

[Y]: Long-Integer-Ausdruck, der die vertikale Position des Rahmens, gemessen vom Ankerpunkt, aufnimmt. Dieser Parameter ist optional. Wird er nicht angegeben, wird der Rahmen zentriert eingefügt, und die vertikale Position ergibt sich dann automatisch aus der mit dem Parameter Height übergebenen Höhe.

[Width]: Long-Integer-Ausdruck, der die Breite des Rahmens angibt. Wird dieser Parameter nicht angegeben, wird eine Standardbreite von 2 cm genommen.

[Height]: Long-Integer-Ausdruck, der die Höhe des Rahmens angibt. Wird dieser Parameter nicht angegeben, wird eine Standardhöhe von 0,5 cm genommen.

Wenn Sie die Parameter X, Y, Width und Height numerisch übergeben, so werden diese immer in der aktuell definierten Maßeinheit gesetzt. Die Werte können auch mit Angabe der gewünschten Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

InsertGlossary-Methode

Fügt in den Text an die aktuelle Position der Schreibmarke einen AutoText-Eintrag ein.

Siehe auch: ExpandGlossary, NewGlossary, SetActGlossaryGroup.

Klasse:
BaseText

Syntax:
InsertGlossary(Group As String, ShortName As String)

PARAMETER:
Group: Ein Zeichenfolgeausdruck, der die Gruppe angibt, der der neue Eintrag zugewiesen werden soll.

ShortName: Ein Zeichenfolgeausdruck, der das Kürzel des AutoText-Eintrages angibt.

BEISPIEL:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
```

```
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

InsertGraphic-Methode

Importiert eine Grafik und fügt diese an der Position der Schreibmarke in das Dokument ein. Ohne Übergabeparameter wird der Dialog GRAFIK EINFÜGEN geöffnet.

KLASSE:
BaseText, BaseTextGraphic

SYNTAX:
InsertGraphic(FileName As String, FilterName As String, Link As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
[FileName]: Ein Zeichenfolgenausdruck, der den Dateinamen der zu importierenden Grafik angibt. Wird hier kein Parameter übergeben, wird der Dialog GRAFIK EINFÜGEN geöffnet.

[FilterName]: Ein Zeichenfolgenausdruck, der den zum Import der Grafik notwendigen Filtertyp angibt. Die mit FilterName angegebene Bezeichnung des Filters muß exakt dem Eintrag des Listenfelds DATEITYP im Dialog GRAFIK EINFÜGEN entsprechen, also z. B. "TIF - Tag Image File" für Grafiken im TIF-Format.

Die Angabe des Grafikfilters ist optional. Wird kein Parameter oder ein Leerstring übergeben, so erfolgt eine automatische Erkennung des benötigen Grafikfilters.

[Link]: Ein Boolescher Ausdruck, der angibt, ob die einzufügende Grafik mit dem Dokument verknüpft (True) oder im Dokument gespeichert (False) werden soll. Dieser Parameter ist optional, die Standardeinstellung ist Link = False.

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

InsertGraphicRuler-Methode

Fügt eine Grafik als horizontale Linie ein. Dabei wird die gewünschte Grafik in einem neuen, zentrierten Absatz positioniert. Als Rückgabewert erhalten Sie True, wenn das Einfügen erfolgreich war, sonst False.

Ohne Übergabeparameter wird der Dialog zum Einfügen einer horizontalen Linie aufgerufen.

Klasse:
BaseText

Syntax:
InsertGraphicRuler(Path As String)

Rückgabe:
Boolean

Parameter:
[Path]: Pfadangabe oder URL der Grafikdatei, die als horizontale Linie eingefügt werden soll. Dieser Parameter ist optional.

Beispiel:

```
Sub ExampleInsertGraphicRuler
Dim sGraphic as String
Documents.Add("sWriter4")
```

```
Selection.Insert("Unter diesem Absatz befindet sich ein Ruler
:Eine horizontale Linie, die aus einem Bild erstellt wird.")
sGraphic = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphicRuler( sGraphic )
End Sub
```

InsertHardHyphen-Methode

Fügt an der aktuellen Cursorposition einen geschützten Bindestrich ein. Dabei handelt es sich um ein Steuerzeichen, das einen Bindestrich darstellt, an dem keine Trennung durchgeführt wird.

KLASSE:
BaseText

SYNTAX:
InsertHardHyphen()

BEISPIEL:

```
Sub ExampleInsertHardHyphen
Const sSentence_1 = "Das Wort EDV"
Const sSentence_2 = "Markt wird nicht getrennt"
Dim iCount As Integer
Documents.Add( "sWriter4" )
ActiveWindow.PageHeight = "11cm"
ActiveWindow.PageWidth = "11cm"
REM Schaltet die automatische Silbentrennung ein
Selection.Paragraph.HyphenZone.Hyphen = TRUE
Selection.Insert sSentence_1
Selection.InsertHardHyphen
Selection.Insert sSentence_2
Selection.GotoStartOfPara( FALSE )
For iCount = 0 To 30
Selection.Insert " "
Next iCount
End Sub
```

InsertHyperlinkDlg-Methode

Ruft zum Einfügen eines Hyperlinks das Register HYPERLINK im Dialog FORMAT - ZEICHEN auf.

KLASSE:
BaseText

SYNTAX:
InsertHyperlinkDlg()

BEISPIEL:

```
Sub ExampleInsertHyperlinkDlg
Documents.Add("sWriter4")
Selection.Insert("So können Sie Hyperlinks definieren.")
Selection.InsertHyperlinkDlg
End Sub
```

InsertIndexEntry-Methode

Fügt einen neuen Eintrag in ein Stichwortverzeichnis ein.

Der neue Eintrag wird erst nach Aktualisieren des Verzeichnisses sichtbar. Die Aktualisierung erfolgt entweder mit der UpdateAllIndexes-Methode oder durch Einfügen eines neuen Stichwortverzeichnisses mittels InsertIndexesIndex.

Siehe auch: InsertIndexesEntry

KLASSE:
Text

SYNTAX:
InsertIndexEntry(Key As String, AltKey As String, Entry As String)

PARAMETER:
Key: Zeichenfolgeausdruck, der den Oberbegriff der ersten Ebene aufnimmt, sofern ein mehrstufiges Stichwortverzeichnis verwendet wird.

AltKey: Zeichenfolgeausdruck, der den Oberbegriff der zweiten Ebene aufnimmt, sofern ein mehrstufiges Stichwortverzeichnis verwendet wird.

Entry: Zeichenfolgeausdruck, der den Text des neuen Eintrages aufnimmt, der in das Stichwortverzeichnis eingefügt werden soll.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
```

```
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesEntry-Methode

Fügt einem Verzeichnis einen neuen Eintrag hinzu. Wird kein Parameter übergeben, wird der Dialog VERZEICHNISEINTRAG EINFÜGEN aufgerufen.

Der neue Eintrag wird erst nach Aktualisieren des Verzeichnisses sichtbar. Die Aktualisierung erfolgt entweder mit der UpdateAllIndexes-Methode oder durch Einfügen eines neuen Verzeichnis mittels InsertIndexesIndex,InsertIndexesTOC oder InsertIndexesUserIndex.

Siehe auch: InsertIndexEntry

KLASSE:
Text

SYNTAX:
InsertIndexesEntry(IndexType As Long, Entry As String, Key1 As String, Key2 As String, Level As Long)

PARAMETER:
[IndexType]: Long-Integer-Ausdruck, der angibt, welchem Verzeichnis ein Eintrag hinzugefügt werden soll:

0: Fügt dem Stichwortverzeichnis einen Eintrag hinzu.

1: Fügt dem Inhaltsverzeichnis einen Eintrag hinzu.

2: Fügt dem Benutzerverzeichnis einen Eintrag hinzu.

Dieser Parameter ist optional. Wird er nicht angegeben, wird standardmäßig IndexType = 2 gesetzt.

[Entry]: Zeichenfolgenausdruck, der den Text des neuen Eintrages aufnimmt. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird die aktuelle Selektion als Eintrag interpretiert.

[Key1]: Zeichenfolgenausdruck, der den Oberbegriff der ersten Ebene aufnimmt, sofern ein mehrstufiges Verzeichnis verwendet wird.

[Key2]: Zeichenfolgenausdruck, der den Oberbegriff der zweiten Ebene aufnimmt, sofern ein mehrstufiges Verzeichnis verwendet wird.

[Level]: Long-Integer-Ausdruck, der bei Verwendung eines mehrstufigen Verzeichnisses die Kapitelebene angibt, die nach rechts eingerückt dargestellt werden soll. Dieser Parameter ist optional. Wird er nicht angegeben, wird als Standardwert Level = 1 genommen.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
```

```
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesIndex-Methode

Fügt ein Stichwortverzeichnis an der Position der Schreibmarke ein. Ohne Übergabeparameter wird der Dialog STICHWORTVERZEICHNIS EINFÜGEN geöffnet.

Siehe auch: InsertIndexEntry, InsertIndexesEntry, InsertIndexesTOC, InsertIndexesUserIndex.

KLASSE:
Text

SYNTAX:
InsertIndexesIndex(CreateForm As Long, Title As String)

PARAMETER:
[CreateForm]: Bestimmt die Zusätze, die bei der Erzeugung des Stichwortverzeichnisses zu berücksichtigen sind. Die nachstehenden möglichen Einstellungen können durch Addition der entsprechenden Werte kombiniert werden:

1: Gleiche Einträge werden zusammengefaßt.

2: Gleiche Einträge werden mit dem Vermerk "ff" zusammengefaßt.

4: Groß-/Kleinschreibung wird bei der Zusammenfassung von Einträgen berücksichtigt.

8: Schlüsseleinträge bilden eigene Einträge.

16: Die alphabetisch zusammengefaßten Einträge des Stichwortverzeichnisses werden durch alphabetische Zeichen getrennt.

Dieser Parameter ist optional. Wird er nicht übergeben, wird als Standardwert CreateForm = 2 genommen.

[Title]: Zeichenfolgenausdruck, der die Bezeichnung des Stichwortverzeichnisses angibt.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
```

```
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesIndex-Methode

Fügt ein Stichwortverzeichnis an der Position der Schreibmarke ein. Ohne Übergabeparameter wird der Dialog STICHWORTVERZEICHNIS EINFÜGEN geöffnet.

Siehe auch: InsertIndexEntry, InsertIndexesEntry, InsertIndexesTOC, InsertIndexesUserIndex.

KLASSE:
Text

SYNTAX:
InsertIndexesIndex(CreateForm As Long, Title As String)

PARAMETER:
[CreateForm]: Bestimmt die Zusätze, die bei der Erzeugung des Stichwortverzeichnisses zu berücksichtigen sind. Die nachstehenden möglichen Einstellungen können durch Addition der entsprechenden Werte kombiniert werden:

1: Gleiche Einträge werden zusammengefaßt.

2: Gleiche Einträge werden mit dem Vermerk "ff" zusammengefaßt.

4: Groß-/Kleinschreibung wird bei der Zusammenfassung von Einträgen berücksichtigt.

8: Schlüsseleinträge bilden eigene Einträge.

16: Die alphabetisch zusammengefaßten Einträge des Stichwortverzeichnisses werden durch alphabetische Zeichen getrennt.

Dieser Parameter ist optional. Wird er nicht übergeben, wird als Standardwert CreateForm = 2 genommen.

[Title]: Zeichenfolgenausdruck, der die Bezeichnung des Stichwortverzeichnisses angibt.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
```

```
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesTOC-Methode

Fügt ein Inhaltsverzeichnis an der Position der Schreibmarke ein. Ohne Übergabeparameter wird der Dialog INHALTSVERZEICHNIS EINFÜGEN aufgerufen.

Siehe auch: InsertIndexesEntry, InsertIndexesIndex, InsertIndexesUserIndex.

KLASSE:
Text

SYNTAX:
InsertIndexesTOC(CreateForm As Long, MaxLevel As Long, Title As String)

PARAMETER:
[CreateForm]: Gibt an, woraus das Inhaltsverzeichnis erstellt werden soll. Die folgenden Einstellungen sind möglich:

1: Das Inhaltsverzeichnis wird aus der aktuellen Markierung erzeugt.

2: Das Inhaltsverzeichnis wird aus der Gliederung des Dokumentes erzeugt.

3: Das Inhaltsverzeichnis wird aus der aktuellen Markierung und aus der Gliederung des Dokumentes erzeugt.

Dieser Parameter ist optional. Wird er nicht übergeben, wird als Standardwert CreateForm = 2 gesetzt.

[MaxLevel]: Ein beliebiger Long-Integer-Ausdruck, der die maximale Anzahl der Ebenen des Inhaltsverzeichnisses angibt. Wird MaxLevel nicht übergeben, werden maximal 5 Ebenen genommen.

[Title]: Ein Zeichenfolgenausdruck, der die Bezeichnung des Inhaltsverzeichnisses angibt.

Beispiel:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
```

```
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesTOC-Methode

Fügt ein Inhaltsverzeichnis in ein Globaldokument ein.

KLASSE:
GlobalContents

SYNTAX:
InsertIndexesTOC(Position As Long, CreateForm As Long, Title As String, MaxLevel As Long)

RÜCKGABE:
Boolean

PARAMETER:
Position: Index, der die Position innerhalb der Globaldokument-Collection bestimmt, an der das Inhaltsverzeichnis eingefügt werden soll. Der Index beginnt bei 1.

CreateForm: Gibt an, woraus das Inhaltsverzeichnis erstellt werden soll. Die folgenden Einstellungen sind möglich:

1: Das Inhaltsverzeichnis wird aus der aktuellen Markierung erzeugt.

2: Das Inhaltsverzeichnis wird aus der Gliederung des Dokumentes erzeugt.

3: Das Inhaltsverzeichnis wird aus der aktuellen Markierung und aus der Gliederung des Dokumentes erzeugt.

Title: Ein Zeichenfolgeausdruck, der die Bezeichnung des Inhaltsverzeichnisses angibt.

[MaxLevel]: Ein beliebiger Long-Integer-Ausdruck, der die maximale Anzahl der Ebenen des Inhaltsverzeichnisses angibt. Dieser Parameter ist optional. Wird er nicht übergeben, werden standardmäßig 5 Ebenen genommen.

BEISPIEL:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
```

```
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\GLOBAL.SGL"
Const sClientDocName = "\CLIENT.SDW"
Const sMessage_1 = "Der Titel des Globaldokumentsbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentsbereichs : "
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
```

```
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

InsertIndexesUserIndex-Methode

Fügt ein Benutzerverzeichnis an der Position der Schreibmarke ein. Ohne Übergabeparameter wird der Dialog BENUTZERVERZEICHNIS EINFÜGEN aufgerufen.

Siehe auch: InsertIndexesEntry, InsertIndexesIndex, InsertIndexesTOC.

KLASSE:
Text

SYNTAX:
InsertIndexesUserIndex(CreateForm As Long, IndexName As String, MaxLevel As Long, Title As String, Template As String)

PARAMETER:
[CreateForm]: Gibt an, woraus das Benutzerverzeichnis erstellt werden soll. Die folgenden Einstellungen sind möglich:

1: Das Benutzerverzeichnis wird aus der aktuellen Markierung erzeugt.

2: Das Benutzerverzeichnis wird aus Überschriften erzeugt.

4: Das Benutzerverzeichnis wird aus Absatzvorlagen erzeugt.

8: Das Benutzerverzeichnis wird aus OLE-Objekten erzeugt.

16: Das Benutzerverzeichnis wird aus Tabellen erzeugt.

32: Das Benutzerverzeichnis wird aus Grafiken erzeugt.

64: Das Benutzerverzeichnis wird aus Tabellen erzeugt.

Dieser Parameter ist optional. Wird er nicht übergeben, wird als Standardwert CreateForm = 1 genommen.

[IndexName]: Zeichenfolgenausdruck, der die Bezeichnung des Benutzerverzeichnisses aufnimmt, über die das Verzeichnis angesprochen werden kann.

[MaxLevel]: Ein beliebiger Long-Integer-Ausdruck, der die maximale Anzahl der Ebenen des Benutzerverzeichnisses angibt. Wird MaxLevel nicht angegeben, werden standardmäßig 5 Ebenen genommen.

[Title]: Ein Zeichenfolgenausdruck, der den Titel des Benutzerverzeichnisses angibt.

[Template]: Zeichenfolgenausdruck, der die Absatzvorlage angibt, nach deren Vorgaben das Benutzerverzeichnis erstellt werden soll. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, dann wird die aktuelle Absatzvorlage genommen.

Beispiel:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
```

```
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertIndexesUserIndex-Methode

Fügt ein Benutzerverzeichnis in ein Globaldokument ein.

Siehe auch: InsertIndexesEntry, InsertIndexesIndex, InsertIndexesTOC.

KLASSE:
GlobalContents

SYNTAX:
InsertIndexesUserIndex(Position As Long, CreateForm As Long, Title As String, MaxLevel As Long, IndexName As String, Template As String)

RÜCKGABE:
Boolean

PARAMETER:
Position: Index, der die Position innerhalb der Globaldokument-Collection bestimmt, an der das Benutzerverzeichnis eingefügt werden soll. Der Index beginnt bei 1.

CreateForm: Gibt an, woraus das Benutzerverzeichnis erstellt werden soll. Die folgenden Einstellungen sind möglich:

1: Das Benutzerverzeichnis wird aus der aktuellen Markierung erzeugt.

2: Das Benutzerverzeichnis wird aus Überschriften erzeugt.

4: Das Benutzerverzeichnis wird aus Absatzvorlagen erzeugt.

8: Das Benutzerverzeichnis wird aus OLE-Objekten erzeugt.

16: Das Benutzerverzeichnis wird aus Tabellen erzeugt.

32: Das Benutzerverzeichnis wird aus Grafiken erzeugt.

64: Das Benutzerverzeichnis wird aus Tabellen erzeugt.

[IndexName]: Zeichenfolgeausdruck, der die Bezeichnung des Benutzerverzeichnisses aufnimmt, über die das Verzeichnis angesprochen werden kann. [MaxLevel]: Ein beliebiger Long-Integer-Ausdruck, der die maximale Anzahl der Ebenen des Benutzerverzeichnisses angibt. Wird dieser Parameter nicht angegeben, werden standardmäßig 5 Ebenen genommen.

[Title]: Ein Zeichenfolgeausdruck, der den Titel des Benutzerverzeichnisses angibt.

[Template]: Zeichenfolgeausdruck, der die Absatzvorlage angibt, nach deren Vorgaben das Benutzerverzeichnis erstellt werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, wird die aktuelle Absatzvorlage genommen.

Beispiel:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\GLOBAL.SGL"
Const sClientDocName = "\CLIENT.SDW"
Const sMessage_1 = "Der Titel des Globaldokumentsbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentsbereichs : "
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
```

```
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
```

```
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

InsertLabels-Methode

Ruft den Dialog ETIKETTEN auf.

KLASSE:
StarWriter

SYNTAX:
InsertLabels()

BEISPIEL:

```
Sub ExampleInsertLabels
Documents.Add("swriter4")
Selection.Insert("So können SIe Etiketten einfügen. ")
StarWriter.InsertLabels
End Sub
```

InsertLinebreak-Methode

Fügt an der Position der Schreibmarke einen Zeilenumbruch ein. Ein neuer Absatz wird dabei nicht erzeugt.

KLASSE:
BaseText

SYNTAX:
InsertLinebreak()

BEISPIEL:

```
Sub ExampleDropCap
Documents.Add("sWriter4")
Selection.Insert "Das erstes Zeichen"
Selection.InsertLineBreak
Selection.Insert "wird über drei"
Selection.InsertLineBreak
Selection.Insert "Zeilen dargestellt"
Selection.InsertLineBreak
Selection.GotoStartOfDoc FALSE
Selection.Paragraph.DropCap.Count = 1
Selection.Paragraph.DropCap.Lines = 3
Selection.Paragraph.DropCap.Distance = "0.5cm"
End Sub
```

InsertMode-Eigenschaft

Bestimmt oder ermittelt den Zustand des Einfügemodus.

KLASSE:
BaseTextEditView

SYNTAX:
InsertMode

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Einfügemodus (True) oder der Überschreibmodus (False) aktiv ist.

BEISPIEL:

```
Sub ExampleInsertMode
Documents.Add( "sWriter4" )
If ActiveWindow.InsertMode = TRUE Then
   Print "Der Einfügemodus ist bei dem Dokument eingeschaltet"
Else
   Print "Der Einfügemodus ist bei dem Dokument ausgeschaltet"
End If
End Sub
```

InsertNamedTable-Methode

Fügt eine Tabelle an der Position der Schreibmarke ein.

KLASSE:
BaseText

SYNTAX:
InsertNamedTable(aName As String, Row As Long, Col As Long)

PARAMETER:
aName: Ein Zeichenfolgeausdruck, der den Namen der Tabelle aufnimmt.

Row: Ein beliebiger Long-Integer-Ausdruck, der die Anzahl der Zeilen der Tabelle festlegt.

Col: Ein beliebiger Long-Integer-Ausdruck, der die Anzahl der Spalten der Tabelle festlegt.

BEISPIEL:

```
Sub ExampleInsertNamedTable
Const sTableName = "TestTabelle"
Const sMessage = "Die Tabelle hat folgenden Namen : "
```

```
Documents.Add "sWriter4"
Selection.InsertNamedTable(sTableName, 3, 3)
Print sMessage & Selection.TableName
End Sub
```

InsertNeutralParagraph-Methode

Fügt innerhalb einer Numerierung oder Aufzählung einen neuen Absatz ohne voranstehende Nummer oder voranstehendes Aufzählungszeichen an der Position der Schreibmarke ein.

KLASSE:
BaseTextList

SYNTAX:
InsertNeutralParagraph()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
```

```
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

InsertNonBreakingSpace-Methode

Fügt an der Position der Schreibmarke ein geschütztes Leerzeichen ein.

KLASSE:
BaseText

SYNTAX:
InsertNonBreakingSpace()

BEISPIEL:

```
Sub ExampleInsertNonBreakingSpace
Const sSentence_1 = "Der Baum hat eine Höhe von 5"
Const sSentence_2 = "Metern."
Dim iCount As Integer
Documents.Add( "sWriter4" )
ActiveWindow.PageHeight = "11cm"
ActiveWindow.PageWidth = "11cm"
REM Schaltet die automatische Silbentrennung ein
Selection.Paragraph.HyphenZone.Hyphen = TRUE
Selection.Insert sSentence_1
Selection.InsertNonBreakingSpace
Selection.Insert sSentence_2
Selection.GotoStartOfPara( FALSE )
For iCount = 0 To 10
Selection.Insert " "
Next iCount
End Sub
```

InsertNumbering-Methode

Schaltet die Numerierung ein.

KLASSE:
BaseText

SYNTAX:
InsertNumbering(Name As String)

PARAMETER:
Name: Zeichenfolgeausdruck, der die Bezeichnung der Numerierung aufnimmt, die verwendet werden soll.

Beispiel:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

InsertObject-Methode

Ruft den Dialog OLE-Objekt einfügen auf.

Klasse:
BaseText

Syntax:
InsertObject()

Beispiel:

```
Sub ExampleInsertObject
Documents.Add "sWriter4"
Selection.Insert("Das ist der Objekt-Einfügen-Dialog")
Selection.InsertObject()
End Sub
```

InsertObjectFloatingFrame-Methode

Ruft den Dialog Frame-Eigenschaften zum Einfügen eines Frames auf.

Klasse:
BaseText

Syntax:
InsertObjectFloatingFrame()

Beispiel:

```
Sub ExampleInsertObjectFloatingFrame
Documents.Add "sWriter4"
Selection.Insert("Das ist der Frame-Einfügen-Dialog")
Selection.InsertObjectFloatingFrame()
End Sub
```

InsertObjectStarChart-Methode

Erzeugt ein StarChart-Objekt, wenn sich die Schreibmarke in einer Tabelle befindet. Dabei wird das StarChart-Objekt aus den Daten der aktuellen Tabelle gebildet und vor der Tabelle in das Dokument eingefügt.

Klasse:
BaseText

Syntax:
InsertObjectStarChart()

Beispiel:

```
Sub ExampleOLEObjects
Const sMessage = "Der Name des Charts lautet : "
Dim oActwin As Object
Documents.Add( "sWriter4" )
Set oActwin = ActiveWindow REM Merkt sich die View für Deacti-
vate
If ActiveWindow.OLEObjects.Count = 0 Then
ActiveWindow.InsertObjectStarChart()
Else
ActiveWindow.OLEObjects(1).Activate
End If
oActwin.DeActivate REM Hier muß die View des Dokumentes
```

```
verwendet werden
Print sMessage & ActiveWindow.OLEObjects(1).OLEObject.Name
End Sub
```

InsertObjectStarImage-Methode

Fügt an der aktuellen Zeile ein (leeres) StarImage-Objekt ein. Dabei öffnet sich der Dialog Neues Bild zur Festlegung von Größe und Farbtiefe des Bildes.

Klasse:
BaseText

Syntax:
InsertObjectStarImage()

Beispiel:

```
Sub ExampleInsertObjectStarImage
Documents.Add "sWriter4"
Selection.Insert("Jetzt wird ein StarImage OLE-Objekt eintge-
fügt")
Selection.InsertObjectStarImage()
End Sub
```

InsertObjectStarMath-Methode

Fügt an der Position der Schreibmarke ein (leeres) Math-Objekt ein. Dabei öffnen sich die Fenster Kommandos und Operatoren zum Eingeben einer Formel.

Klasse:
BaseText

Syntax:
InsertObjectStarMath()

Beispiel:

```
Sub ExampleInsertObjectStarMath
Documents.Add "sWriter4"
Selection.Insert("So können Sie Formeln eingeben: ")
Selection.InsertObjectStarMath()
End Sub
```

InsertPagebreak-Methode

Erzeugt an der Position der Schreibmarke einen harten Seitenumbruch. Soll der neuen Seite dieselbe Vorlage wie der vorangehenden Seite zugewiesen werden, muß die Methode ohne Parameter aufgerufen werden.

KLASSE:
BaseText

SYNTAX:
InsertPagebreak(PageDesc As String)

PARAMETER:
[PageDesc]: Zeichenfolgenausdruck, der die Bezeichnung einer der im Stylisten aufgeführten Seitenvorlagen angibt, die der neuen Seite zugewiesen werden soll. Wird dieser Parameter nicht übergeben, wird die Seitenvorlage der vorangehenden Seite übernommen.

BEISPIEL:

```
Sub ExampleGotoPage
Const sPageStyle = "Neue Vorlage"
Const sMessage_1 = "Dies ist ein Absatz auf der ersten Seite. "
Const sMessage_2 = "Dies ist ein Absatz auf der zweiten Seite.
"
Dim iCount As Integer
Documents.Add( "sWriter4" )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
REM Erzeugt eine neue Seitenvorlage.
Selection.StyleNewByExample(sPageStyle, 8)
REM Fügt eine neue Seite ein und verwendet dafür die neue
Vorlage.
Selection.InsertPageBreak(sPageStyle)
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfDoc FALSE
Selection.GotoStartOfPrevPage FALSE
Selection.GotoEndOfPage FALSE
Selection.GotoEndOfNextPage FALSE
Selection.GotoEndOfPrevPage FALSE
Selection.GotoStartOfNextPage FALSE
Selection.GotoStartOfPage FALSE
Activewindow.GotoPage(1)
ActiveDocument.DeleteStyle( sPageStyle, 8 )
End Sub
```

InsertPageCountField-Methode

Fügt den Feldbefehl "Seiten" des Feldtyps Statistik an der aktuellen Cursorposition ein.

KLASSE:
Text

SYNTAX:
InsertPageCountField()

BEISPIEL:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

InsertPageNumberField-Methode

Fügt den Feldbefehl "Seitennummer" an der aktuellen Cursorpostion ein. Dieser Feldbefehl liefert die logische Seitennummer, sofern dieser Wert nicht manuell um einen Korrekturwert geändert wurde.

Siehe auch: LogicalPageNumber, PageNumber.

KLASSE:
Text

Syntax:
InsertPageNumberField()

Beispiel:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

InsertPara-Methode

Fügt an der Position der Schreibmarke ein Absatzende ein und erzeugt somit einen neuen Absatz.

Klasse:
BaseText

Syntax:
InsertPara()

Beispiel:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
```

```
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

InsertPlugin-Methode

Ruft den Dialog PlugIn einfügen auf.

Klasse:
BaseText

Syntax:
InsertPlugin()

Beispiel:

```
Sub ExampleInsertPlugIn
Documents.Add("swriter4")
Selection.Insert("So können Sie ein PlugIn einfügen")
Selection.InsertPlugIn
End Sub
```

InsertRegion-Methode

Erstellt einen neuen Bereich aus der aktuellen Selektion. Ohne Übergabeparameter ruft diese Methode den Dialog Bereich einfügen auf.

Wurde ein neuer Bereich erstellt, dann wird als Rückgabe der Name des eingefügten Bereichs übergeben. Dieser kann vom Parameter "RegionName" abweichen, wenn "RegionName" bereits im Dokument vergeben wurde.

Klasse:
BaseTextSelection

SYNTAX:
InsertRegion(RegionName As String, RegionCondition As String, RegionHidden As Boolean, RegionProtect As Boolean, LinkName As String, FilterName As String, SubRegion As String)

RÜCKGABE:
String

PARAMETER:
[RegionName]: Zeichenfolgenausdruck, der die Bezeichnung des Bereiches aufnimmt, der aus der Selektion erstellt werden soll. Wird RegionName nicht angegeben, wird ein Leerstring übergeben.

[RegionCondition]: Zeichenfolgenausdruck, der die Bedingung für die Anzeige des neuen Bereiches aufnimmt. Wird dieser Parameter nicht angegeben, wird ein Leerstring übergeben.

[RegionHidden]: Boolescher Ausdruck, der angibt, ob der neue Bereich aus- (True) oder eingeblendet (False) werden soll. Wird dieser Parameter nicht übergeben, wird der neue Bereich automatisch eingeblendet.

[RegionProtect]: Boolescher Ausdruck, der angibt, ob der Text im neuen Bereich schreibgeschützt ist (True) oder nicht (False). Wird RegionProtect nicht definiert, ist der Text nicht schreibgeschützt.

[LinkName]: Soll der Bereich eine Dateiverknüpfung enthalten, dann ist hier der Dateiname zu übergeben.

[FilterName]: Name des Dateifilters für die Datei "LinkName". Wird FilterName nicht angegeben, wird automatisch ein Filter ermittelt.

[SubRegion]: Soll die Verknüpfung mit einer Textmarke oder einem Bereich innerhalb der Datei "LinkName" erfolgen, wird mit SubRegion der Name der Textmarke oder des Bereichs übergeben. Wird SubRegion nicht angegeben, erfolgt die Verknüpfung mit dem gesamten Dokument, welches mit dem Parameter LinkName angegeben wurde.

BEISPIEL:

```
Sub ExampleGetRegionName
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich "
Const sMessage_3 = " hat folgenden Namen : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
```

```
Selection.InsertPara
For iCount = 1 to Selection.GetRegionCount
Selection.Insert sMessage_2 & iCount & sMessage_3 &
Selection.GetRegionName(iCount-1)
Next iCount
Selection.DeleteRegion sRegionName
End Sub
```

InsertRows-Methode

Fügt an der Position der Schreibmarke eine bestimmte Zeilenzahl in die aktuelle Tabelle ein.

KLASSE:
BaseTextTable

SYNTAX:
InsertRows(Count As Long, After As Boolean)

PARAMETER:
Count: Ein beliebiger Long-Integer-Audruck, der die Anzahl der einzufügenden Zeilen angibt.

[After]: Boolescher Ausdruck, der angibt, ob die Zeilen unter (True) oder über (False) der Position der Schreibmarke eingefügt werden sollen. Wird dieser Parameter nicht angegeben, wird die Standardeinstellung After = True genommen.

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

InsertScript-Methode

Fügt ein Scriptfeld an der aktuellen Cursorposition ein. Ohne Übergabeparameter wird der Dialog SCRIPT EINFÜGEN geöffnet.

KLASSE:
BaseText

SYNTAX:
InsertScript(Script As String, AsURL As Boolean, ScriptType As String)

PARAMETER:
Script: Zeichenfolge, die den Inhalt des Scripts oder den URL einer Datei, die das Script enthält, angibt.

[AsURL]: Boolescher Ausdruck, der angibt, ob das Script als Text (False) oder als URL (True) interpretiert wird. Die Angabe des Parameters AsURL ist optional. Wird kein Parameter übergeben, so wird das Script als Text interpretiert.

[ScriptType]: Gibt den Typ der Scriptsprache an. Dieser Parameter ist optional. Wird kein Parameter übergeben, so wird als Voreinstellung "JavaScript" genommen.

BEISPIEL:

```
Sub ExampleInsertScript
   Documents.Add "sWriter4/Web"
   Selection.Insert("Hier wird")
   Selection.InsertScript("function GoAlert(){ alert(""Das ist
ein JavaScript"");}")
   Selection.Insert(" ein ")
   Selection.SetHyperlink( "JavaScript" ,
"JavaScript:GoAlert()", "", 1 )
   Selection.Insert(" ausgeführt.")
   ActiveDocument.SaveAs Application.Pathsettings.Work &
"\scriptmp.htm"
   REM Einige Ansichts-Änderungen
   ActiveWindow.ReloadForEdit = TRUE
   Wait 100
   ActiveWindow.ReloadForEdit = FALSE
   Wait 100
REM  ActiveDocument.Save
   ActiveDocument.FrameSetSource = TRUE
   Wait 100
   ActiveDocument.FrameSetSource = FALSE
   Wait 100
   ActiveDocument.Save
   ActiveWindow.ReloadForEdit = TRUE
End Sub
```

InsertSoftHyphen-Methode

Erzeugt an der Position der Schreibmarke eine weiche Silbentrennung.

KLASSE:
BaseText

SYNTAX:
InsertSoftHyphen()

BEISPIEL:

```
Sub ExampleInsertSoftHyphen
Const sSentence_1 = "Das Wort EDV"
Const sSentence_2 = "Markt wird getrennt"
Dim iCount As Integer
Documents.Add( "sWriter4" )
ActiveWindow.PageHeight = "11cm"
ActiveWindow.PageWidth = "11cm"
REM Schaltet die automatische Silbentrennung ein
Selection.Paragraph.HyphenZone.Hyphen = TRUE
Selection.Insert sSentence_1
Selection.InsertSoftHyphen
Selection.Insert sSentence_2
Selection.GotoStartOfPara( FALSE )
For iCount = 0 To 35
Selection.Insert " "
Next iCount
End Sub
```

InsertSymbol-Methode

Fügt entweder ein Sonderzeichen oder ein Symbol an der Position der Schreibmarke ein. Ohne Übergabeparamter wird der Dialog SONDERZEICHEN zum Einfügen eines Sonderzeichens oder eines Symbols aufgerufen.

KLASSE:
TextDrawText, BaseText

SYNTAX:
InsertSymbol(Chars As String, Font As String)

PARAMETER:
Chars: Ein Zeichenfolgenausdruck, der die Zeichen enthält, die als Sonderzeichen dargestellt werden sollen.

Font: Ein Zeichenfolgenausdruck, der die Bezeichnung des Zeichensatzes angibt, aus dem die Sonderzeichen entnommen werden sollen.

Beispiel:

```
Sub ExampleInsertSymbol
Const sMessage = "Es als Sonderzeichen ein Briefsymbol einge-
fügt :  "
Documents.Add( "sWriter4" )
Selection.Insert sMessage
Selection.GoLeft( 1, FALSE )
ActiveWindow.InsertSymbol(CHR$(42), "WingDings")
End Sub
```

InsertTable-Methode

Fügt an der Position der Schreibmarke eine Tabelle ein. Ohne Übergabeparameter wird der Dialog Tabelle einfügen aufgerufen.

Klasse:
BaseText

Syntax:
InsertTable(Row As Long, Col As Long)

Parameter:
[Row]: Ein beliebiger Long-Integer-Ausdruck, der die Anzahl der Zeilen der neuen Tabelle angibt. Wird dieser Parameter nicht angegeben oder wird Row = 0 übergeben, öffnet sich der Dialog Tabelle einfügen.

[Col]: Ein beliebiger Long-Integer-Ausdruck, der die Anzahl der Spalten der neuen Tabelle angibt. Wird dieser Parameter nicht angegeben oder wird Col = 0 übergeben, öffnet sich der Dialog Tabelle einfügen.

Beispiel:

```
Sub ExampleInsertCaptionDialog
Documents.Add("swriter4")
REM Leeren Absatz, sonst kann man die Tabelle mit der Beschrif-
tung nicht selektieren.
Selection.InsertPara
Selection.InsertTable(3, 2)
Selection.Alignment=2
Selection.Width = "10cm"
REM Jetzt die Beschriftung einfügen.
Selection.InsertCaptionDialog
ActiveWindow.JumpToStartOfDoc
REM Jetzt alles in einen Rahmen packen.
ActiveWindow.SelectAll
Selection.InsertFrame(2,"4cm","5cm","8cm","5cm")
End Sub
```

InsertTimeField-Methode

Fügt den Feldbefehl "Zeit/fix" an der aktuellen Cursorpostion ein.

KLASSE:
BaseText

SYNTAX:
InsertTimeField()

BEISPIEL:

```
Sub ExampleFileDialog
Documents.Add("swriter4")
with ActiveWindow.Selection
.Insert("Die Uhrzeit ist ")
.InsertTimeField()
.InsertPara
.Insert("Das Datum ist ")
.InsertDateField()
.GoLeft 1
.FieldDialog
end with
end sub
```

InsertTitleField-Methode

Fügt den Feldbefehl "Titel" an der aktuellen Cursorpostion ein.

KLASSE:
Text

SYNTAX:
InsertTitleField()

BEISPIEL:

```
Sub ExampleInsertAuthorField
Const sTitle = "Titel des Dokumentes"
Const sTopicField = "Thema des Dokumentes"
Const sMessage_1 = "Der Autor des Dokumentes : "
Const sMessage_2 = "Der Titel des Dokumentes : "
Const sMessage_3 = "Das Thema des Dokumentes : "
Documents.Add( "sWriter4" )
ActiveDocument.DocumentInfo.Title = sTitle
ActiveDocument.DocumentInfo.Subject = sTopicField
Selection.Insert sMessage_1
Selection.InsertAuthorField
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTitleField
Selection.InsertPara
```

```
Selection.Insert sMessage_3
Selection.InsertTopicField
End Sub
```

InsertTOCEntry-Methode

Fügt einen neuen Eintrag in ein Inhaltsverzeichnis ein, das aus Markierungen erstellt wurde. Der Eintrag steht dann im Inhaltsverzeichnis, ohne daß es ein gleichlautendes Wort im Dokumententext geben muß.

Siehe auch: InsertUserEntry

KLASSE:
Text

SYNTAX:
InsertTOCEntry(Level As Long, Entry As String)

PARAMETER:
Level: Beliebiger Long-Integer-Ausdruck, der die Ebene des Inhaltsverzeichnisses angibt, sofern ein mehrstufiges Inhaltsverzeichnis verwendet wird.

Entry: Zeichenfolgeausdruck, der den Text des Eintrages aufnimmt, der dem Inhaltsverzeichnis zugewiesen werden soll.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
```

```
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertTopicField-Methode

Fügt den Feldbefehl "Thema" an der aktuellen Cursorpostion ein.

KLASSE:
Text

SYNTAX:
InsertTopicField()

BEISPIEL:

```
Sub ExampleInsertAuthorField
Const sTitle = "Titel des Dokumentes"
Const sTopicField = "Thema des Dokumentes"
Const sMessage_1 = "Der Autor des Dokumentes : "
Const sMessage_2 = "Der Titel des Dokumentes : "
Const sMessage_3 = "Das Thema des Dokumentes : "
Documents.Add( "sWriter4" )
ActiveDocument.DocumentInfo.Title = sTitle
ActiveDocument.DocumentInfo.Subject = sTopicField
Selection.Insert sMessage_1
```

```
Selection.InsertAuthorField
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTitleField
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertTopicField
End Sub
```

InsertUserEntry-Methode

Fügt einen neuen Eintrag in ein Benutzerverzeichnis ein, das aus Markierungen erstellt wurde. Der Eintrag steht dann im Benutzerverzeichnis, ohne daß es ein gleichlautendes Wort im Dokumententext geben muß.

Siehe auch: InsertTOCEntry

KLASSE:
Text

SYNTAX:
InsertUserEntry(IndexName As String, Level As Long, Entry As String)

PARAMETER:
IndexName: Zeichenfolgeausdruck, der die Bezeichnung des Benutzerverzeichnisses aufnimmt, dem ein neuer Eintrag zugewiesen werden soll.

Level: Beliebiger Long-Integer-Ausdruck, der die Ebene des Benutzerverzeichnisses angibt, sofern ein mehrstufiges Inhaltsverzeichnis verwendet wird.

Entry: Zeichenfolgeausdruck, der den Text des Eintrages aufnimmt, der dem Benutzerverzeichnis zugewiesen werden soll.

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
```

```
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

InsertVCControl-Methode

Fügt ein Kontrollfeld seitengebunden an der angegebenen Position und mit angegebener Größe ein. Die Methode liefert einen Rückgabewert zurück, der angibt, ob das Kontrollfeld erfolgreich eingefügt werden konnte (True) oder nicht (False).

Siehe auch: InsertVCControlAnchored

KLASSE:
BaseText

Syntax:
InsertVCControl(Type As Long, X As Long, Y As Long, Width As Long, Height As Long)

Rückgabe:
Boolean

Parameter:
Type: Long-Integer-Ausdruck, der den Typ des Kontrollfeldes angibt. Dabei wird zwischen Kontrollfeldern unterschieden, die in ein Dokument oder in einen Dialog eingefügt werden. Folgende Werte sind möglich:

X: Long-Integer-Ausdruck, der die horizontale Position des Kontrollfeldes angibt.

Y: Long-Integer-Ausdruck, der die vertikale Position des Kontrollfeldes angibt.

Width: Long-Integer-Ausdruck, der die Breite des Kontrollfeldes angibt.

Height: Long-Integer-Ausdruck, der die Höhe des Kontrollfeldes angibt.

Beispiel:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
```

```
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

InsertVCControlAnchored-Methode

Fügt ein Kontrollfeld an der aktuellen Cursorposition in das Dokument ein, wobei die Art der Verankerung und die Größe des Kontrollfeldes mit angegeben werden.

Siehe auch: InsertVCControl

KLASSE:
BaseText

SYNTAX:
InsertVCControlAnchored(Type As Long, Anchor As Long, Width As Long, Height As Long)

RÜCKGABE:
Boolean

PARAMETER:
Type: Long-Integer-Ausdruck, der den Typ des Kontrollfeldes angibt. Folgende Werte sind möglich:

0 = Markierfeld (CheckBox)

1 = Optionsfeld (OptionButton)

2 = Befehlsschaltfläche (CommandButton)

3 = Drehfeld (SpinButton)

4 = Bezeichnungsfeld (Label)

5 = Gruppierungsrahmen (FrameControl)

6 = Listenfeld (ListBox)

7 = Kombinationsfeld (ComboBox)

8 = Textfeld (TextBox)

9 = horizontale Bildlaufleiste (HScrollBar)

10 = vertikale Bildlaufleiste (VScrollBar)

25 = URL-Schaltfläche (URL-Button)

Anchor: Long-Integer-Ausdruck, der die Verankerung des Kontrollfeldes angibt. Folgende Werte sind möglich:

0: Am Absatz

1: Am Zeichen

2: An der Seite

Width: Long-Integer-Ausdruck, der die Breite des Kontrollfeldes angibt.

Height: Long-Integer-Ausdruck, der die Höhe des Kontrollfeldes angibt.

BEISPIEL:

```
Sub ExampleInsertVCControlAnchored
Const sMessage = "Seitengebundener Commandbutton"
Documents.Add( "sWriter4" )
Activewindow.InsertVCControlAnchored( 2, 2, "6cm", "1cm")
Activewindow.Controls(1).ControlObject.Caption = sMessage
End Sub
```

```
Sub ExampleInsertVCControlAnchored
Const sMessage = "Seitengebundener Commandbutton"
Documents.Add( "sWriter4" )
Activewindow.InsertVCControlAnchored( 2, 2, "6cm", "1cm")
Activewindow.Controls(1).ControlObject.Caption = sMessage
End Sub
```


Interior-Eigenschaft

Interior enthält die Eigenschaften für den Hintergrund eines Objekts. Folgende Eigenschaften gibt es:

Style - Hintergrundmuster

Transparent - transparenter Hintergrund

BackColor - Hintergrundfarbe

FillColor - Musterfarbe

GrfFilename - URL der verknüpften Hintergrundgrafik

GrfFiltername - Name des Grafikfilters

GrfPosition - Position der Grafik

KLASSE:
BaseTextSelection

SYNTAX:
Interior

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

InterLineSpace-Eigenschaft

Bestimmt oder ermittelt, welcher Zeilenabstand im aktuellen Absatz verwendet werden soll, bzw. verwendet wird.

Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetLineSpacing gesetzte Zeilenhöhe abgefragt.

Eine Abfrage der InterLineSpace-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLineSpacing bestimmt wurde, daß nach einem bestimmten Zeilenabstand gesucht werden soll, d. h. IsLineSpacing = 2.

Siehe auch: SetLineSpacing, IsLineSpacing, LineSpace, LineHeight, PropLineHeight, InterSpace.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
InterLineSpace

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Absätzen:

[= Value]: Integer-Ausdruck, der angibt, welcher Zeilenabstand im aktuellen Absatz verwendet werden soll. Die folgenden Einstellungen sind möglich:

0 - In dem aktuellen Absatz wird ein einzeiliger Zeilenabstand oder ein Mindestzeilenabstand verwendet. Der Wert für den Mindestzeilenabstand wird mit der LineHeight-Eigenschaft bestimmt.

1 - In dem aktuellen Absatz wird ein proportionaler, 1,5-zeiliger oder 2-zeiliger Zeilenabstand verwendet. Die Prozentzahl des zur Zeilenhöhe prozentualen Zeilenabstandes wird mit der PropLineHeight-Eigenschaft festgesetzt.

2 - In dem aktuellen Absatz wird ein Durchschuß verwendet, also ein zusätzlicher Abstand zum Zeilenabstand. Der zusätzliche Abstand wird über die Eigenschaft InterSpace festgesetzt.

Beispiel:

```
Sub ExampleParagraphSpacing2
REM den lateinischen Blindtext verwenden
Const sBlindText = "BT"
Documents.Add("sWriter4")
With ActiveWindow.Selection
.ExpandGlossary( sBlindText )
wait 1000
.Paragraph.InterLineSpace = 1   REM Typ des Zeilenzwischenraume
auf REM relativREM  einstellen.
.Paragraph.PropLineHeight= 185  REM Angabe in Prozent
end with
end sub
```

InterSpace-Eigenschaft

Bestimmt oder ermittelt den Wert eines Durchschusses, also eines zusätzlichen Abstands zum bereits bestehenden Zeilenabstand. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLineSpacing gesetzte Durchschuß abgefragt.

Eine Abfrage der InterSpace-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLineSpacing bestimmt wurde, daß nach einem bestimmten Zeilenabstand gesucht werden soll, d. h. IsLineSpacing = 2.

Siehe auch: SetLineSpacing, IsLineSpacing, LineSpace, LineHeight, PropLineHeight, InterLineSpace.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
InterSpace

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Absätzen:

[= AddDist]: Integer-Ausdruck, der den Durchschuß in Twips angibt.

BEISPIEL:

```
Sub ExampleInterSpace
Const sMessage_1 = "Dieser Absatz zeigt"
Const sMessage_2 = " einen Durchschuß."
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.Font.Size = 20
Selection.Insert sMessage_2
With ActiveWindow.Selection.Paragraph
.InterLineSpace = 2
.InterSpace = 852
End With
End Sub
```

IsAlignment-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Ausrichtung.

Siehe auch: SetAlignment, Alignment.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsAlignment

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Ausrichtung gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Ausrichtung gesucht.

1 - Es wird nach Absätzen mit dem Attribut Ausrichtung gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Ausrichtung gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsAlignment = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetAlignment ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

Beispiel:

```
Sub ExampleSetAlignment
Const sMessage_1 = "Dies ist ein linksbündig formatierter
Absatz"
Const sMessage_2 = "Dies ist ein rechtsbündig formatierter
Absatz"
Const sMessage_3 = "Es wird der erste rechtsbündige Absatz
gesucht"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Alignment = 1
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsAlignment = 2
ActiveWindow.SearchAttributes.SetAlignment(1)
If ActiveWindow.SearchAttributes.Alignment = 1 Then
Print sMessage_3
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsBold-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Fett.

Siehe auch: Bold, SetBold.

Klasse:
ReplaceAttributes, SearchAttributes

Syntax:
IsBold

Rückgabe:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Fett gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Fett gesucht.

1 - Es wird nach Zeichen mit dem Attribut Fett gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Fontstärke gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsBold = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetBold ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSearchAndReplaceBold
Documents.Add("sWriter4")
REM erstmal eine fette Zeile erstellen
With ActiveWindow.Selection
   .Insert("a bBold c dBold e fBold g hBold i")
   .GotoPrevWord
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
End With
ActiveWindow.SearchAttributes.Reset
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search ""
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
```

```
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
ActiveWindow.ReplaceAttributes.Reset
REM ab jetzt werden die gefundenen Stellen gelöscht
ActiveWindow.Replace
ActiveWindow.Replace
End Sub
```

IsCaseMap-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Auszeichnung (Schrifteffekt).

Siehe auch: CaseMap, SetCaseMap.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsCaseMap

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Auszeichnung gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Auszeichnung gesucht.

1 - Es wird nach Zeichen mit dem Attribut Auszeichnung gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Auszeichnung gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsCaseMap = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetCaseMap ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetCaseMap
Const sMessage_1 = "Es wird ein, mit Kapitälchen formatierter
Text gesucht"
Const sMessage_2 = "Dieser Absatz ist als Kapitälchen forma-
tiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
```

```
Selection.InsertPara
Selection.Font.CaseMap = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsAlignment = 2
ActiveWindow.SearchAttributes.SetCaseMap(4)
If ActiveWindow.SearchAttributes.CaseMap = 4 Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsClientImageMap-Eigenschaft

Stellt fest, ob die aktuelle Selektion eine Client ImageMap ist. Ist das der Fall wird der Wert True zurückgegeben, andernfalls False.

Siehe auch: ImageMapURLAt, ImageMapURLCount, IsServerImageMap.

KLASSE:
BaseTextFrame

SYNTAX:
IsClientImageMap

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleImageMap
Const sText="Das ist eine ImageMap in HTML"
Const sAltTag1="Nach Hause"
Const sAltTag2="Hier ist es Interessant"
Const sAltTag3="Bild von Pamela"
Dim bServer As Boolean, bClient As Boolean REM Typ der Imagemap
Dim iMax As Integer, iCount As Integer
Dim sLink As String REM Inhalt der Imagemap URLREM s
Documents.Add("sWriter4")
REM erstelle eine ImageMap per HTML-einfügen
with  ActiveWindow.Selection
.Insert(chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)& sText &
chr(60)&"/BODY"&chr(62))
.Insert(
chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)&chr(60)&"P"&chr(6
2))
.Insert( chr(60)&"MAP NAME=""MAP1"""&chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""77,5,107,32""
HREF=""www.stardivision.de"" TARGET=""_self"" ALT="""" & sAlt-
```

```
Tag1 & """" & chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""33,5,66,44""
HREF=""another interesting place"" TARGET=""_self"" ALT="""" &
sAltTag2 & """"  &chr(62)
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""1,2,31,34""
HREF=""Pamela Pic"" TARGET=""_self"" ALT="""" & sAltTag3 & """"
& chr(62))
.Insert( chr(60)&"MAP"&chr(62))
.Insert( chr(60)&"IMG
SRC=""file:///D|/Office40/gallery/baer.tif"" NAME=""Grafik1""
ALIGN=LEFT WIDTH=108 HEIGHT=63 BORDER=0
USEMAP=""#MAP1"""&chr(62)&chr(60)&"BR CLEAR=LEFT"&chr(62))
.Insert(
chr(60)&"BR"&chr(62)&chr(60)&"BR"&chr(62)&chr(60)&"P"&chr(62)&c
hr(60)&"/BODY"&chr(62))
end with
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut
ActiveDocument.Close "",False
Documents.Add("sWriter4/Web")
ActiveDocument.FramesetSource = True
ActiveWindow.SelectAll
ActiveWindow.Paste
ActiveDocument.FramesetSource = False
REM Die Grafik bearbeiten
with ActiveWindow.Graphics(1)
.GRFLinkName = Application.Pathsettings.Config & "\End.bmp" REM
Grafik neu setzen
.ImageMapDialog=TRUE  REM ImageMap Dialog hochreisen
Wait 500
bServer = .IsServerImageMap
bClient = .IsClientImageMap
iMax =  .ImageMapURLCount
for iCount = 0 to iMax-1
sLink = .ImageMapURLAt(iCount)
MsgBox sLink,64,"ImageMap URL " & iCount+1
next
.ImageMapDialog=False
.Escape
end with
end sub
```

IsContour-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Zeichenattribut Kontur.

Siehe auch: SetContour, Contour.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsContour

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Integer-Ausdruck, der die Suchoption für das Attribut Kontur bestimmt. Folgende Werte sind möglich:

0: Es wird nicht nach dem Attribut Kontur gesucht.

1, 2: Es wird nach dem Attribut Kontur gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsContour = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetContour ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetContour
Const sMessage_1 = "Es wird ein, mit Konturschrift formatierter
Text gesucht"
Const sMessage_2 = "Bei diesem Absatz ist der Text als Kontur-
schrift formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.OutlineFont = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsContour = 2
ActiveWindow.SearchAttributes.SetContour( TRUE )
If ActiveWindow.SearchAttributes.Contour = TRUE Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsControl-Eigenschaft

Ermittelt, ob das selektierte Objekt ein Kontrollfeld ist. Ist das der Fall, wird der Wert True zurückgegeben, ansonsten False.

Siehe auch: Type

KLASSE:
BaseTextDrawBase

SYNTAX:
IsControl

Rückgabe:
Boolean

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

IsEndOfDoc-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Ende des Dokuments befindet. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

Klasse:
BaseText

Syntax:
IsEndOfDoc

Rückgabe:
Boolean

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleIsStartOfDoc
Const sPara = "Dies ist ein Beispiel um die Funktion von
IsStartOfDoc zuzeigen."
Const sMessage_1 = "Dies ist der Dokumentenanfang"
Const sMessage_2 = "Das Dokumentenende"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sPara
Selection.InsertPara
Selection.JumpToStartOfDoc
If Selection.IsStartOfDoc = TRUE Then
Selection.Insert sMessage_1
End If
Selection.JumpToEndOfDoc
If Selection.IsEndOfDoc = TRUE Then
Selection.Insert sMessage_2
End If
End Sub
```

IsEndOfLine-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Ende der aktuellen Zeile befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, ansonsten False.

Klasse:
BaseText

Syntax:
IsEndOfLine

Rückgabe:
Boolean

Parameter:
(nur lesen)

Beispiel:

```
REM Dieses Beispiel ändert die Farbe des letzte Buchstaben vom
REM Wort->Rot
REM Satz->Grün
REM Zeile->Blau
Sub ExampleIsEndOfLine
Const sPara_1 = "Dies ist ein Absatz, bei dem die Farbe des
letzten Buchstabens "
```

```
Const sPara_2 = "des Wortes auf REM RotREM , des Satzes auf REM
GrünREM , der Zeile auf "
Const sPara_3 = "REM BlauREM  gesetzt wird. "
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.Insert sPara_2
Selection.Insert sPara_3
Selection.JumpToStartOfDoc
Do
If Selection.IsEndOfLine Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 0, 255 )
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfSentence Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 255, 0 )
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfWord Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.GoRight( 2, FALSE )
Else
Selection.GoRight( 1, FALSE )
End If
Loop Until Selection.IsEndOfDoc
End Sub
```

IsEndOfPara-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Ende des aktuellen Absatzes befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, ansonsten False.

KLASSE:
BaseText

SYNTAX:
IsEndOfPara

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
REM Dieses Beispiel zählt die Sätze innerhalb eines Absatzes
Sub ExampleGotoNextSentence
Const sSentence = "Dies ist ein Satz. Dies ist noch ein Satz."
Const sMessage = "Anzahl der Absätze im aktuellen Absatz : "
Dim iSentenceCount As Integer
```

```
Documents.Add( "sWriter4" )
Selection.Insert sSentence
With ActiveWindow.Selection
.GoToStartOfPara
iSentenceCount = 0
Do
iSentenceCount = iSentenceCount + 1
.GotoNextSentence
Loop Until .IsEndOfPara
End With
print sMessage & iSentenceCount
End Sub
```

IsEndOfSentence-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Ende des aktuellen Satzes befindet. Ist das der Fall, gibt die Eigenschaft den Wert TRUE zurück, ansonsten FALSE.

KLASSE:
BaseText

SYNTAX:
IsEndOfSentence

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
REM Dieses Beispiel ändert die Farbe des letzte Buchstaben vom
REM Wort->Rot
REM Satz->Grün
REM Zeile->Blau
Sub ExampleIsEndOfLine
Const sPara_1 = "Dies ist ein Absatz, bei dem die Farbe des
letzten Buchstabens "
Const sPara_2 = "des Wortes auf REM RotREM , des Satzes auf REM
GrünREM , der Zeile auf "
Const sPara_3 = "REM BlauREM  gesetzt wird. "
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.Insert sPara_2
Selection.Insert sPara_3
Selection.JumpToStartOfDoc
Do
If Selection.IsEndOfLine Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 0, 255 )
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfSentence Then
```

```
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 255, 0 )
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfWord Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.GoRight( 2, FALSE )
Else
Selection.GoRight( 1, FALSE )
End If
Loop Until Selection.IsEndOfDoc
End Sub
```

IsEndOfWord-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Ende des aktuellen Wortes befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, sonst wird False zurückgegeben.

KLASSE:
BaseText

SYNTAX:
IsEndOfWord

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
REM Dieses Beispiel ändert die Farbe des letzte Buchstaben vom
REM Wort->Rot
REM Satz->Grün
REM Zeile->Blau
Sub ExampleIsEndOfLine
Const sPara_1 = "Dies ist ein Absatz, bei dem die Farbe des
letzten Buchstabens "
Const sPara_2 = "des Wortes auf REM RotREM , des Satzes auf REM
GrünREM , der Zeile auf "
Const sPara_3 = "REM BlauREM  gesetzt wird. "
Documents.Add( "sWriter4" )
Selection.Insert sPara_1
Selection.Insert sPara_2
Selection.Insert sPara_3
Selection.JumpToStartOfDoc
Do
If Selection.IsEndOfLine Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 0, 255 )
```

```
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfSentence Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 0, 255, 0 )
Selection.GoRight( 2, FALSE )
ElseIf Selection.IsEndOfWord Then
Selection.GoLeft( 1, TRUE )
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.GoRight( 2, FALSE )
Else
Selection.GoRight( 1, FALSE )
End If
Loop Until Selection.IsEndOfDoc
End Sub
```

IsEscapement-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Zeichenposition.

Siehe auch: SetEscapement, Escapement, RelativeSize.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsEscapement

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Zeichenposition gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Zeichenposition gesucht.

1 - Es wird nach Zeichen mit dem Attribut Zeichenposition gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Zeichenposition gesucht, welche durch Schriftgröße und relative Hoch- oder Tiefstellung der Zeichen festgelegt ist.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsEscapement = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetEscapement ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

Beispiel:

```
Sub ExampleSetEscapement
Const sMessage_1 = "Es wird ein, als hochgestellt formatierter
Text gesucht"
Const sMessage_2 = "Bei diesem Absatz ist der Text hochgestellt
formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsEscapement = 2
ActiveWindow.SearchAttributes.SetEscapement( 101, 62 )
If ActiveWindow.SearchAttributes.Escapement = 101 Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsFirstRecord-Methode

Ermittelt, ob der aktuelle Datensatz einer Datenbank-Tabelle der erste Datensatz ist. Ist das der Fall, liefert diese Methode den Wert True zurück, andernfalls False.

Um diese Methode anwenden zu können, muß die Datenbank-Tabelle geöffnet sein.

Siehe auch: OpenTableConnection, IsLastRecord.

Klasse:
BaseTextEditView

Syntax:
IsFirstRecord()

Rückgabe:
Boolean

Beispiel:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
```

```
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
```

```
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsFont-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Font.

Siehe auch: SetFont, StyleName, Pitch, CharSet,

Family,
FamilyName.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsFont

RÜCKGABE:
Boolean

PARAMETER:
[Value]: Bestimmt, ob und wie nach dem Attribut Font gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Font gesucht.

1 - Es wird nach Zeichen mit dem Attribut Font gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Font gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsFont = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetFont ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetFont
Const sMessage_1 = "Es wird ein Text mit der Schrift REM Helve-
ticaREM  gesucht"
Const sMessage_2 = "Dieser Absatz ist mit der Schrift REM
HelveticaREM  formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.FamilyName = "Helvetica"
Selection.Font.Size = 14
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.IsFontHeight = 2
ActiveWindow.SearchAttributes.SetFont("", 14,,,"Helvetica")
ActiveWindow.SearchAttributes.SetFontHeight( 14, 100 )
If ActiveWindow.SearchAttributes.FamilyName = "Helvetica" Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsFontColor-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Zeichenfarbe.

Siehe auch: SetFontColor, FontColor.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsFontColor

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Zeichenfarbe gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Zeichenfarbe gesucht

1 - Es wird nach Zeichen mit dem Attribut Zeichenfarbe gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Zeichenfarbe gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsFontColor = Value gesetzt werden Der Wert 2 wird eingestellt, wenn über SetFontColor ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetFontColor
Const sMessage_1 = "Es wird ein Text, der mit der Farbe REM
RotREM  formatiert ist, gesucht"
Const sMessage_2 = "Bei diesem Absatz hat die Schrift die Farbe
REM RotREM "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.SetFontColor( RGB( 255, 0, 0 ) )
If ActiveWindow.SearchAttributes.FontColor = RGB( 255, 0, 0 )
Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsFontHeight-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Schriftgröße.

Siehe auch: SetFontHeight, Size, Prop.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsFontHeight

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt oder ermittelt, ob und wie nach dem Attribut Schriftgröße gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Schriftgröße gesucht.

1 - es wird nach Zeichen mit dem Attribut Schriftgröße gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Schriftgröße gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsFontHeight = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetFontHeight ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetFont
Const sMessage_1 = "Es wird ein Text mit der Schrift REM Helve-
ticaREM  gesucht"
Const sMessage_2 = "Dieser Absatz ist mit der Schrift REM
HelveticaREM  formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.FamilyName = "Helvetica"
Selection.Font.Size = 14
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.IsFontHeight = 2
ActiveWindow.SearchAttributes.SetFont("", 14,,,"Helvetica")
```

```
ActiveWindow.SearchAttributes.SetFontHeight( 14, 100 )
If ActiveWindow.SearchAttributes.FamilyName = "Helvetica" Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsHyphenZone-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Silbentrennung.

Siehe auch: SetHyphenZone, Hyphen.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsHyphenZone

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Silbentrennung gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Silbentrennung gesucht.

1 - Es wird nach Absätzen mit dem Attribut Silbentrennung gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Regel zur Silbentrennung gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsHyphenZone = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetHyphenZone ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetHyphenZone
Const sMessage_1 = "Es wird ein Absatz mit eingeschalteter
Silbentrennung gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die automtische
Silbentrennung aktiv"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.HyphenZone.Hyphen = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
```

```
ActiveWindow.SearchAttributes.IsHyphenZone = 2
ActiveWindow.SearchAttributes.SetHyphenZone( TRUE, TRUE, 2, 2,
0 )
If ActiveWindow.SearchAttributes.Hyphen = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsItalic-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Kursiv.

Siehe auch: SetItalic, Italic

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsItalic

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Kursiv gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Kursiv gesucht.

1 - Es wird nach Zeichen mit dem Attribut Kursiv gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Einstellung für das Attribut Kursiv gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsItalic = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetItalic ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSearchAndReplaceBold
Documents.Add("sWriter4")
REM erstmal eine fette Zeile erstellen
With ActiveWindow.Selection
   .Insert("a bBold c dBold e fBold g hBold i")
   .GotoPrevWord
   .GotoPrevWord TRUE
```

```
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
End With
ActiveWindow.SearchAttributes.Reset
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search ""
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
ActiveWindow.ReplaceAttributes.Reset
REM ab jetzt werden die gefundenen Stellen gelöscht
ActiveWindow.Replace
ActiveWindow.Replace
End Sub
```

IsLanguage-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Sprache.

Siehe auch: SetLanguage, Language.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsLanguage

RÜCKGABE:
Boolean

PARAMETER:
[Value]: Bestimmt, ob und wie nach dem Attribut Sprache gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Sprache gesucht.

1 - Es wird nach dem Attribut Sprache gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einer bestimmten Sprache gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsLanguage = Value gesetzt werden Der Wert 2 wird eingestellt, wenn über SetLanguage ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2

BEISPIEL:

```
Sub ExampleSetLanguage
Const sMessage_1 = "Es wird ein Absatz bei dem die Sprache auf
Finnisch gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist als Sprache REM
FinnischREM  gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Language = 1035 REM Finnisch
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLanguage = 2
ActiveWindow.SearchAttributes.SetLanguage( 1035 )
If ActiveWindow.SearchAttributes.Language = 1035 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsLastRecord-Methode

Ermittelt, ob der aktuelle Datensatz einer Datenbank-Tabelle der letzte Datensatz ist. Ist das der Fall, liefert diese Methode den Wert True zurück, andernfalls False.

Um diese Methode anwenden zu können, muß die Datenbank-Tabelle geöffnet sein.

Siehe auch: OpenTableConnection, IsFirstRecord.

KLASSE:
BaseTextEditView

SYNTAX:
IsLastRecord()

RÜCKGABE:
Boolean

BEISPIEL:

```
sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
```

```
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
```

```
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsLeftRightParaMargin-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Absatzeinzüge.

Siehe auch: SetLeftRightParaMargin, LeftMargin, RightMargin, IndentFirstLine.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsLeftRightParaMargin

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Absatzeinzüge gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Absatzeinzüge gesucht.

1 - Es wird nach Absätzen mit dem Attribut Absatzeinzüge gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach bestimmten Absatzeinzügen gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsLeftRightParaMargin = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetLeftRightParaMargin ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub Main
ActiveWindow.SearchAttributes.IsLeftRightParaMargin = 2
Activewindow.SearchAttributes.SetLeftRightParaMargin(1.2, 0, 0)
ExVar = ActiveWindow.SearchAttributes.LeftMargin
Activewindow.Search
End Sub
```

IsLineSpacing-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Zeilenabstand.

Siehe auch: SetLineSpacing, LineSpace, LineHeight, PropLineHeight, InterLineSpace, InterSpace.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsLineSpacing

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Zeilenabstand gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Zeilenabstand gesucht.

1 - Es wird nach Absätzen mit dem Attribut Zeilenabstand gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Zeilenabstand gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsLineSpacing = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetLineSpacing ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetLineSpacing
Const sMessage_1 = "Es wird ein Absatz mit einer Zeilehöhe von
1.5 cm, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Zeilenhöhe auf
1.5 cm gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.LineSpace = 1
Selection.Paragraph.PropLineHeight = 200
Selection.Paragraph.LineHeight = 852 REM 1,5cm
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLineSpacing = 2
ActiveWindow.SearchAttributes.SetLineSpacing( 1, 852, 0, 0, 0 )
If ActiveWindow.SearchAttributes.LineHeight = 852 Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsOrphan-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Hurenkindregelung.

Siehe auch: SetOrphan, Orphan

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsOrphan

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Hurenkindregelung gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Hurenkindregelung gesucht.

1 - Es wird nach Absätzen mit dem Attribut Hurenkindregelung gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach bestimmten Werten für die Hurenkindregelung gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsOrphan = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetOrphan ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetOrphan
Const sMessage_1 = "Es wird ein Absatz, bei dem die Huren-
kindregel auf 4 Zeilen gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Hurenkindregel
auf 4 Zeilen gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Orphan = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsOrphan = 2
ActiveWindow.SearchAttributes.SetOrphan( 4 )
If ActiveWindow.SearchAttributes.Orphan = 4 Then
Print sMessage_1
```

```
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsPaintLocked-Eigenschaft

Ermittelt, ob die Bildschirmanzeige des Dokumentenfensterinhalts aktualisiert wird. Ist das Neuzeichnen der aktuellen Ansicht nicht aktiv, liefert IsPaintLocked den Wert True, andernfalls False.

Siehe auch: LockPaint

KLASSE:
BaseTextEditView

SYNTAX:
IsPaintLocked

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleLockPaint
Const sMessage_1 = "Den Aufbau der beiden Absätze ist nicht zu
sehen."
Const sMessage_2 = "Das ist der zweite Absatz."
Const sError = "Das Abschalten der Bildschirmaktualisierung hat
nicht funktioniert"
Documents.Add( "sWriter4" )
ActiveWindow.LockPaint(TRUE)
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
If ActiveWindow.IsPaintLocked = FALSE Then
Print sError
End If
ActiveWindow.LockPaint(FALSE)
End Sub
```

IsParaSplit-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut ABSATZ TRENNEN.

Siehe auch: SetParaSplit, ParaSplit.

KLASSE:
ReplaceAttributes, SearchAttributes

Syntax:
IsParaSplit

Rückgabe:
Boolean

Parameter:
[= Value]: Integer-Ausdruck, der die Suchoption für das Attribut Absatz trennen bestimmt. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Absatz trennen gesucht.

1 - Es wird nach Absätzen mit dem Attribut Absatz trennen gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Absatz trennen gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsParaSplit = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetParaSplit ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

Beispiel:

```
Sub ExampleSetParaSplit
Const sMessage_1 = "Es wird ein Absatz, bei dem ein Seitenum-
bruch im Absatz erfolgen darf, gesucht."
Const sMessage_2 = "Bei diesem Absatz darf ein Seitenumbruch
innerhalb eines Absatzes erfolgen."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.ParaSplit = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsParaSplit = 2
ActiveWindow.SearchAttributes.SetParaSplit( TRUE )
If ActiveWindow.SearchAttributes.ParaSplit = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsQueryNative-Methode

Ermittelt, ob eine Datenbank-Abfrage eine Native SQL ist. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
IsQueryNative(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
```

```
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
```

```
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsQueryReadOnly-Methode

Gibt an, ob eine Datenbank-Abfrage verändert werden kann. Ist die Datenbank schreibgeschützt, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
IsQueryReadOnly(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleAddDatabaseConnection
Const sDBName = "TestDB"
Const sTabName = "Tabelle"
Const sTextName = "TEXT_"
Const sNumericName = "NUMERIC_"
Const sQueryName = "Abfrage"
Const sMessage_1 = "Fehler in der Tabellenstruktur"
Const sMessage_2 = "Es wurde nicht zum ersten Datensatz
gesprungen"
Const sMessage_3 = "Der Feldtyp des dritten Feldes lautet : "
Const sMessage_4 = "Das dritte Feld hat eine Größe von : "
Const sMessage_5 = "Es wurde nicht zum letzten Datensatz
gesprungen"
Const sMessage_6 = "Die Tabelle enthält "
Const sMessage_6a = " Spalten"
Const sMessage_7 = "IsQueryNative liefert : "
Const sMessage_8 = "IsQueryReadOnly liefert : "
Const sMessage_9 = "Die Abfrage verwendet folgendes SQL-State-
ment : "
```

```
Dim stDataRow As New Strings
Dim stNewDataRow As New Strings
Dim stAllQueries As New Strings
Dim stQueryRows As New strings
Dim stAppendRec As New Strings
Dim stColumnsStrings As New Strings
Dim sDBPfad As string, sConfigStr As String, sSQLStatement As
String
Dim vStatus As Variant, iCount As Integer, iQueCount As Integer
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Damit die neue Datenbank im Explorer erscheint, muß die
Datenbank im
REM Explorerverzeichnis angelegt werden.
sDBPfad = Application.PathSettings.Modules & "\explorer\"
sConfigStr = "typ=DBF;dsn=" & sDBPfad &
";charset=IBMPC;deleted=False"
Application.AddDatabaseConnection(sDBPfad & sDBName, sCon-
figStr, sDBName)
vStatus = ActiveWindow.OpenDatabaseConnection(sDBName)
For iCount=1 To 3
stDataRow.Add sTextName & iCount & ",TEXT,10,FALSE"
stDataRow.Add sNumericName & iCount & ",NUMERIC,INTEGER,FALSE"
Next iCount
vStatus = ActiveWindow.CreateTable(sTabName, stDataRow)
vStatus = ActiveWindow.OpenTableConnection(sTabName)
REM Nachdem öffnen muß man zuerst zu einen Datensatz springen
vStatus = ActiveWindow.DataFirstRecord
REM Die stNewDataRow-Stringscollection enthält die gleichen
REM Informationen wie die stDataRow-Stringscollection
stNewDataRow = ActiveWindow.GetTableColumns(sTabName)
If Left$(stNewDataRow.Item(2), 9) <> ActiveWindow.DataCo-
lumnName(2) Then
Print sMessage_1 & Left$(stNewDataRow.Item(2), 9)
End if
vStatus = ActiveWindow.DataAppendRecord("Test_1,1,Test_2,2")
vStatus = ActiveWindow.DataFirstRecord
If ActiveWindow.IsFirstRecord = FALSE Then
Print sMessage_2
End if
For iCount=1 To 3
stAppendRec.Add sTextName & iCount
stAppendRec.Add iCount
stColumnsStrings.Add sTextName & iCount
stColumnsStrings.Add sNumericName & iCount
Next iCount
vStatus = ActiveWindow.DataAppendWholeRecord(stColumnsStrings,
stAppendRec)
Print sMessage_3 & ActiveWindow.DataColumnType(3)
Print sMessage_4 & ActiveWindow.DataColumnSubType(3)
ActiveWindow.DataLastRecord
If ActiveWindow.IsLastRecord = FALSE Then
```

```
Print sMessage_5
end if
vStatus = ActiveWindow.DataDeleteRecord
Print sMessage_6 & ActiveWindow.DataColumnNumber & sMessage_6a
vStatus = ActiveWindow.CreateQuery(sQueryName, "SELECT * FROM "
& sTabName , FALSE)
stAllQueries = ActiveWindow.GetQueryNames()
iQueCount = stAllQueries.Count
if iQueCount > 0 then
Print sMessage_7 & ActiveWindow.IsQueryNative(sQueryName)
ActiveWindow.CloseQueryConnection(sQueryName)
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
stQueryRows = ActiveWindow.GetQueryColumns(sQueryName)
If stQueryRows.Count = 6 Then
Print sMessage_8 & ActiveWindow.IsQueryReadOnly(sQueryName)
sSQLStatement = ActiveWindow.GetSQL_Statement(sQueryName)
Print sMessage_9 & sSQLStatement
End if
ActiveWindow.DeleteQuery sQueryName
End If
ActiveWindow.CloseTableConnection(sTabName)
REM Es wird die angelegte Tabelle gelöscht
vStatus = ActiveWindow.DeleteTable(sTabName)
ActiveWindow.CloseDatabaseConnection(sDBName)
REM Es wird die Datenbank aus dem Explorer entfernt
Application.DeleteDatabaseConnection(sDBName)
End Sub
```

IsRegionHidden-Methode

Gibt an, ob der Bereich RegionName versteckt ist.

KLASSE:
BaseTextSelection

SYNTAX:
IsRegionHidden(RegionName As String)

RÜCKGABE:
Boolean

PARAMETER:
RegionName: Bereichsname, dessen Eigenschaft erfragt werden soll.

BEISPIEL:

```
Sub ExampleSetRegionHidden
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  ist unsichtbar"
Const sMessage_4 = "REM  ist sichtbar"
Dim iCount As Integer
```

```
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionHidden(sRegionName, TRUE)
If Selection.IsRegionHidden(sRegionName) = TRUE Then
Selection.Insert sMessage_2 & sRegionName & sMessage_3
Else
Selection.Insert sMessage_2 & sRegionName & sMessage_4
End If
End Sub
```

IsRegionPassword-Eigenschaft

Gibt an, ob der Dialog Bereiche bearbeiten über ein Paßwort geschützt ist.

Klasse:
BaseTextSelection

Syntax:
IsRegionPassword

Rückgabe:
Boolean

Beispiel:

```
Sub ExampleSetRegionPassword
Const sRegionName = "Bereich"
Const sPassword = "Passwort"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der BereichsDialog ist Passwort geschützt"
Const sMessage_3 = "Der BereichsDialog ist nicht Passwort
geschützt"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionPassword(sPassword)
If Selection.IsRegionPassword() = TRUE Then
Selection.Insert sMessage_2
Else
Selection.Insert sMessage_3
End If
End Sub
```

IsRegionProtected-Methode

Gibt an, ob der Bereich RegionName geschützt ist.

KLASSE:
BaseTextSelection

SYNTAX:
IsRegionProtected(RegionName As String)

RÜCKGABE:
Boolean

PARAMETER:
RegionName: Bereichsname, dessen Eigenschaft erfragt werden soll.

BEISPIEL:

```
Sub ExampleSetRegionProtected
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  ist geschützt"
Const sMessage_4 = "REM  ist nicht geschützt"
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionProtected(sRegionName, TRUE)
If Selection.IsRegionProtected(sRegionName) = TRUE Then
Selection.Insert sMessage_2 & sRegionName & sMessage_3
Else
Selection.Insert sMessage_2 & sRegionName & sMessage_4
End If
End Sub
```

IsServerImageMap-Eigenschaft

Ermittelt, ob an Grafiken oder OLE-Objekten eines Server-Dokuments das Flag für eine ImageMap gesetzt ist. Ist das der Fall, wird der Wert True zurückgegeben, andernfalls False.

Siehe auch: IsClientImageMap

KLASSE:
BaseTextFrame

SYNTAX:
IsServerImageMap

Rückgabe:
Boolean

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleImageMap
Const sText="Das ist eine ImageMap in HTML"
Const sAltTag1="Nach Hause"
Const sAltTag2="Hier ist es Interessant"
Const sAltTag3="Bild von Pamela"
Dim bServer As Boolean, bClient As Boolean REM Typ der Imagemap
Dim iMax As Integer, iCount As Integer
Dim sLink As String REM Inhalt der Imagemap URLREM s
Documents.Add("sWriter4")
REM erstelle eine ImageMap per HTML-einfügen
with  ActiveWindow.Selection
.Insert(chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)& sText &
chr(60)&"/BODY"&chr(62))
.Insert(
chr(60)&"HTML"&chr(62)&chr(60)&"BODY"&chr(62)&chr(60)&"P"&chr(6
2))
.Insert( chr(60)&"MAP NAME=""MAP1"""&chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""77,5,107,32""
HREF=""www.stardivision.de"" TARGET=""_self"" ALT="""" & sAlt-
Tag1 & """" & chr(62))
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""33,5,66,44""
HREF=""another interesting place"" TARGET=""_self"" ALT="""" &
sAltTag2 & """"  &chr(62)
.Insert( chr(60)&"AREA SHAPE=RECT COORDS=""1,2,31,34""
HREF=""Pamela Pic"" TARGET=""_self"" ALT="""" & sAltTag3 & """"
& chr(62))
.Insert( chr(60)&"MAP"&chr(62))
.Insert( chr(60)&"IMG
SRC=""file:///D|/Office40/gallery/baer.tif"" NAME=""Grafik1""
ALIGN=LEFT WIDTH=108 HEIGHT=63 BORDER=0
USEMAP=""#MAP1"""&chr(62)&chr(60)&"BR CLEAR=LEFT"&chr(62))
.Insert(
chr(60)&"BR"&chr(62)&chr(60)&"BR"&chr(62)&chr(60)&"P"&chr(62)&c
hr(60)&"/BODY"&chr(62))
end with
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut
ActiveDocument.Close "",False
Documents.Add("sWriter4/Web")
ActiveDocument.FramesetSource = True
ActiveWindow.SelectAll
ActiveWindow.Paste
ActiveDocument.FramesetSource = False
REM Die Grafik bearbeiten
with ActiveWindow.Graphics(1)
.GRFLinkName = Application.Pathsettings.Config & "\End.bmp" REM
```

```
Grafik neu setzen
.ImageMapDialog=TRUE  REM ImageMap Dialog hochreisen
Wait 500
bServer = .IsServerImageMap
bClient = .IsClientImageMap
iMax =  .ImageMapURLCount
for iCount = 0 to iMax-1
sLink = .ImageMapURLAt(iCount)
MsgBox sLink,64,"ImageMap URL " & iCount+1
next
.ImageMapDialog=False
.Escape
end with
end sub
```

IsShadowed-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Zeichenattribut Schattiert.

Siehe auch: SetShadowed, Shadowed.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsShadowed

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Integer-Ausdruck, der die Suchoption für das Attribut Schattiert bestimmt. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Schattiert gesucht.

1 - Es wird nach Absätzen mit dem Attribut Schattiert gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Schattiert gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsShadowed = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetShadowed ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetShadowed
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
schattiert dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift schat-
```

```
tiert dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Shadowed = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsShadowed = 2
ActiveWindow.SearchAttributes.SetShadowed( TRUE )
If ActiveWindow.SearchAttributes.Shadowed = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsSpacing-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Laufweite.

Siehe auch: SetSpacing, Spacing.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsSpacing

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Laufweite gesucht werden soll:

0 - Es wird nicht nach dem Attribut Laufweite gesucht.

1 - Es wird nach Absätzen mit dem Attribut Laufweite gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Laufweite gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsSpacing = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetSpacing ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSpacing
Const sMessage_1 = "Es wird ein Absatz, bei dem die Laufweite
des Fonts verändert wurde, gesucht."
```

```
Const sMessage_2 = "Bei diesem Absatz wurde die Laufweite des
Fonts verändert."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Spacing = 20
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsSpacing = 2
ActiveWindow.SearchAttributes.SetSpacing( 20 )
If ActiveWindow.SearchAttributes.Spacing = 20 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsStartOfDoc-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Anfang des Dokuments befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, andernfalls wird False zurückgegeben.

KLASSE:
BaseText

SYNTAX:
IsStartOfDoc

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleIsStartOfDoc
Const sPara = "Dies ist ein Beispiel um die Funktion von
IsStartOfDoc zuzeigen."
Const sMessage_1 = "Dies ist der Dokumentenanfang"
Const sMessage_2 = "Das Dokumentenende"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sPara
Selection.InsertPara
Selection.JumpToStartOfDoc
If Selection.IsStartOfDoc = TRUE Then
Selection.Insert sMessage_1
End If
Selection.JumpToEndOfDoc
If Selection.IsEndOfDoc = TRUE Then
```

```
Selection.Insert sMessage_2
End If
End Sub
```

IsStartOfPara-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Anfang des aktuellen Absatzes befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, sonst wird False zurückgegeben.

KLASSE:
BaseText

SYNTAX:
IsStartOfPara

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleValueAndBreaks
Documents.Add("swriter4")
With Selection
.Insert("Hier kommt ein harter Umbruch.")
.InsertPara
.Insert("Hier kommt ein weicher Umbruch.")
.InsertLineBreak
.Insert("Der Text geht weiter.")
.InsertPara
.InsertPara
.Insert("Das ist das Ende.")
.InsertPara
End With
ActiveWindow.JumpToStartOfDoc
Do While NextParaBegin() = TRUE
Wait 1000
Loop
End Sub
Function NextParaBegin() as Boolean
REM Die Funktion springt zu dem nächsten Umbruch, egal ob hart
oder weich
REM Der Umbruch hängt ev. vom System ab.
Const C_LineBreak = 10
Const C_CarridgeReturn = 13
Dim sSel As String
If ActiveWindow.IsStartOfPara Then ActiveWindow.GoRight 1
ActiveWindow.SearchSettings.RegExp = TRUE
ActiveWindow.Search "^.|\n|^."
```

```
sSel = Selection.ValueAndBreaks
If sSel = "" Then
NextParaBegin = FALSE
Else
If sSel = Chr(C_LineBreak) Then
Selection.Escape
Elseif sSel = Chr(C_CarridgeReturn) Then
Selection.Escape
ActiveWindow.GoLeft 1
Else
ActiveWindow.GoLeft 1
End If
NextParaBegin = TRUE
End If
End Function
```

IsStartOfWord-Eigenschaft

Ermittelt, ob sich die Schreibmarke am Anfang des aktuellen Wortes befindet. Ist das der Fall, gibt die Eigenschaft den Wert True zurück, ansonsten wird False zurückgegeben.

KLASSE:
BaseText

SYNTAX:
IsStartOfWord

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
REM Die Prozedur stellt sicher das der Cursor am Anfang eines
Wortes stehen bleibt
Sub SetCursorToStartOfWord
Const sMessage = "Dies ist ein Satz"
Documents.Add( "sWriter4" )
Selection.Insert sMessage
If Selection.IsStartOfWord = FALSE Then
Selection.GoToPrevWord
End If
End Sub
```

IsStrikeOut-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Durchstrichen.

Siehe auch: SetStrikeOut, StrikeOut

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsStrikeOut

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Durchstrichen gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Durchstrichen gesucht.

1 - Es wird nach Zeichen mit dem Attribut Durchstrichen gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Durchstrichen gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsStrikeOut = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetStrikeOut ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetItalic
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
kursiv dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift kursiv
dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Italic = soItalicOblique
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsItalic = 2
ActiveWindow.SearchAttributes.SetItalic( soItalicOblique )
If ActiveWindow.SearchAttributes.Italic = soItalicOblique Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsTableReadOnly-Methode

Gibt an, ob eine Datenbank-Tabelle verändert werden kann. Ist sie schreibgeschützt, wird der Wert True zurückgegeben, ansonsten False.

KLASSE:
BaseTextEditView

SYNTAX:
IsTableReadOnly(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Name der Tabelle.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
```

```
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

IsTopBottomMargin-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Absatzattribut Abstand.

Siehe auch: SetTopBottomMargin, BottomMargin, TopMargin.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsTopBottomMargin

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Abstand gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Abstand gesucht.

1 - Es wird nach Absätzen mit dem Attribut Abstand gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Abstand gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsTopBottomMargin = Value gesetzt werden Der Wert 2 wird eingestellt, wenn über SetTopBottomMargin ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub Main
ActiveWindow.SearchAttributes.IsTopBottomMargin = 2
Activewindow.SearchAttributes.SetTopBottomMargin("1.2cm", "")
ExVar = ActiveWindow.SearchAttributes.BottomMargin
Activewindow.Search
End Sub
```

IsUnderline-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Unterstrichen.

Siehe auch: SetUnderline, Underline.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsUnderline

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Unterstrichen gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Unterstrichen gesucht.

1 - Es wird nach dem Attribut Unterstrichen gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Unterstrichen gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsUnderline = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetUnderline ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetUnderline
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
unterstrichen dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift unter-
strichen dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Underline = soUnderlineSingle
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsUnderline = 2
ActiveWindow.SearchAttributes.SetUnderline( soUnderlineSingle )
If ActiveWindow.SearchAttributes.Underline = soUnderlineSingle
Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsWidow-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Schusterjungenregelung.

siehe auch: SetWidow, Widow

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsWidow

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Bestimmt, ob und wie nach dem Attribut Schusterjungenregelung gesucht werden soll. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Schusterjungenregelung gesucht.

1 - Es wird nach Absätzen mit dem Attribut Schusterjungenregelung gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach bestimmten Werten für die Schusterjungenregelung gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsWidow = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetWidow ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetWidow
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schuster-
jungenregel auf 4 Zeilen gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Schusterjungen-
regel auf 4 Zeilen gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Widow = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsWidow = 2
ActiveWindow.SearchAttributes.SetWidow( 4 )
If ActiveWindow.SearchAttributes.Widow = 4 Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

IsWordMode-Eigenschaft

Bestimmt oder ermittelt die Suchoption für das Attribut Wortweise.

Siehe auch: SetWordMode, WordMode.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
IsWordMode

RÜCKGABE:
Boolean

PARAMETER:
[= Value]: Integer-Ausdruck, der die Suchoption für das Attribut Wortweise bestimmt. Folgende Werte sind möglich:

0 - Es wird nicht nach dem Attribut Wortweise gesucht.

1 - Es wird nach dem Attribut Wortweise gesucht, ein spezieller Wert ist nicht gesetzt.

2 - Es wird nach einem bestimmten Wert für das Attribut Wortweise gesucht.

Die Werte 0 und 1 können über ActiveWindow.SearchAttributes.IsWordMode = Value gesetzt werden. Der Wert 2 wird eingestellt, wenn über SetWordMode ein Attribut gesetzt wird.

An den ReplaceAttributes gibt es nur die Werte 0 und 2.

BEISPIEL:

```
Sub ExampleSetWordMode
Const sMessage_1 = "Es wird ein Absatz, bei dem die jedes Wort
einzeln unterstrichen dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird jedes Wort einzeln
unterstrichen dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Underline = soUnderlineSingle
Selection.Font.WordMode = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsWordMode = 2
```

```
ActiveWindow.SearchAttributes.SetWordMode( TRUE )
If ActiveWindow.SearchAttributes.WordMode = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

Italic-Eigenschaft

Bestimmt oder ermittelt die Option für die kursive Darstellung des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetItalic gesetzte Kursiv-Darstellung abgefragt.

Eine Abfrage der Italic-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsItalic bestimmt wurde, daß nach einer bestimmten Einstellung für das Attribut Kursiv gesucht werden soll, d. h. IsItalic = 2.

Siehe auch: SetItalic, IsItalic.

KLASSE:
TextDrawText, ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Italic

RÜCKGABE:
Integer

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Text-Objekten:

[= ItalNum]: Integer-Ausdruck, der den Grad der Kursiv-Darstellung angibt. Folgende Werte sind möglich:

0: Der Text wird nicht kursiv dargestellt.

1 - 2: Der Text wird kursiv dargestellt. Der Grad der Kursiv-Darstellung erhöht sich mit zunehmendem Wert. Der Wert 2 gibt die Standardeinstellung des Attributs Kursiv wieder.

Nicht alle Fonts unterstützen verschiedene Stufen des Italic-Attributs. Der Standardwert 2 ist jedoch fast immer verfügbar.

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
```

```
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Eigenschaften und Methoden mit "J"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "J" beginnen.

JumpDownThisLevel-Methode

Springt innerhalb einer Aufzählung oder Numerierung zum nächsten Absatz der gleichen Ebene.

KLASSE:
BaseTextList

SYNTAX:
JumpDownThisLevel()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
```

```
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

JumpToBookmark-Methode

Springt zu der angegebenen Textmarke.

Siehe auch: GoToBookmark

KLASSE:
BaseTextSelection

SYNTAX:
JumpToBookmark(Marke As String)

RÜCKGABE:
Boolean

PARAMETER:
Marke: Zeichenfolgeausdruck, der die Bezeichnung der Textmarke aufnimmt, zu der gesprungen werden soll. Die Bezeichnung der Textmarke muß unter Berücksichtigung der Groß- und Kleinschreibung übergeben werden, wie sie bei der Definition der Textmarke angegeben wurde.

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
```

```
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

JumpToEndOfDoc-Methode

Setzt den Cursor von der aktuellen Position an das Ende des Dokuments, wobei das Dokumentende durch den letzten Absatz definiert ist. Ist vor Anwendung dieser Methode Text markiert, wird zusätzlich der gesamte Bereich ab Beginn der Markierung bis zum Sprungziel selektiert.

Bei Anwendung dieser Methode kann sich der Cursor sowohl in einem Absatz oder in einer Tabelle oder einem Rahmen befinden, im Gegensatz zur GoToEndOfDoc-Methode springt JumpToEndOfDoc stets an das Ende des aktuellen Dokuments.

Siehe auch: GoToEndOfDoc, JumpToStartOfDoc.

KLASSE:
BaseTextSelection

SYNTAX:
JumpToEndOfDoc()

BEISPIEL:

```
Sub ExampleJumpToEndOfDoc
Const sMessage_1 = "Hier beginnt das Dokument"
Const sMessage_2 = "Hier endet das Dokument"
Dim ICount As Integer
Documents.Add( "sWriter4" )
For iCount = 1 To 10
Selection.InsertPara
Next iCount
Selection.JumpToStartOfDoc
Selection.Insert sMessage_1
```

```
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
End Sub
```

JumpToFooter-Methode

Bewegt die Schreibmarke in die Fußzeile der aktuellen Seite. Befindet sich die Schreibmarke bereits in der Fußzeile oder wird die JumpToFooter-Methode wiederholt angewendet, so springt die Schreibmarke von der Fußzeile an das Ende des letzten Absatzes der aktuellen Seite.

Siehe auch: JumpToHeader

KLASSE:
BaseText

SYNTAX:
JumpToFooter()

BEISPIEL:

```
Sub ExampleSetPageStyleHeader
Const sMessage_1 = "Dies ist eine Kopfzeile"
Const sMessage_2 = "Dies ist eine Fußzeile"
Documents.Add( "sWriter4" )
REM Legt eine Kopfzeile an
Selection.SetPageStyleHeader(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToHeader()
Selection.Insert sMessage_1
Selection.JumpToEndOfDoc
REM Legt eine Fußzeile an
Selection.SetPageStyleFooter(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToFooter()
Selection.Insert sMessage_2
End Sub
```

JumpToFootnoteOrAnchor-Methode

Bewegt die Schreibmarke von einer Fußnote zum entsprechenden Anker und umgekehrt.

KLASSE:
Text

SYNTAX:
JumpToFootnoteOrAnchor()

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
```

```
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert(( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

JumpToHeader-Methode

Bewegt die Schreibmarke in die Kopfzeile der aktuellen Seite. Befindet sich die Schreibmarke bereits in der Kopfzeile oder wird die JumpToHeader-Methode wiederholt angewendet, so springt die Schreibmarke von der Kopfzeile an den Anfang des ersten Absatzes der aktuellen Seite.

Siehe auch: JumpToFooter

KLASSE:
BaseText

SYNTAX:
JumpToHeader()

BEISPIEL:

```
Sub ExampleSetPageStyleHeader
Const sMessage_1 = "Dies ist eine Kopfzeile"
Const sMessage_2 = "Dies ist eine Fußzeile"
Documents.Add( "sWriter4" )
REM Legt eine Kopfzeile an
Selection.SetPageStyleHeader(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToHeader()
Selection.Insert sMessage_1
Selection.JumpToEndOfDoc
REM Legt eine Fußzeile an
Selection.SetPageStyleFooter(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToFooter()
Selection.Insert sMessage_2
End Sub
```

JumpToMark-Methode

Setzt den Cursor an den Anfang des Textes im angesprungenen Objekt (zum Beispiel Tabelle, Textrahmen) oder markiert das Objekt (zum Beispiel Grafik).

Dabei wird die Marke im Anzeigefenster an den Anfang gescrollt.

Diese Methode ist nur an der View zu verwenden, wenn die View eines der Objekte aus den genannten Klassen enthält.

KLASSE:
BaseTextEditView

SYNTAX:
JumpToMark(Marke As String)

PARAMETER:
Marke: Der Name der Marke ist zusammengesetzt aus dem Namen der Markierung, dem | und dem Bezeichner des Typs der Markierungsart. Folgende Typen gibt es:

Table: Tabellen

Frame: Textrahmen

Region: Bereiche

Text: Textmarken innerhalb des Dokuments

Outline: Überschriften (als Name der Markierung wird der Anfangstext der Überschrift angegeben)

Graphic: Grafiken

Ole: OLE-Objekte

Wird kein Typ übergeben, wird der Name als Textmarkenname interpretiert.

Die Namen der Markierungen, die in der Syntax zu übergeben sind, entsprechen den Einträgen im Navigator.

BEISPIEL:

```
Sub ExampleJumpToMark
Dim sFrameName As String
Documents.Add( "sWriter4" )
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFrameName = Selection.FrameName
Selection.GotoAnchor
ActiveWindow.JumpToMark( sFrameName & "|Frame" )
End Sub
```

JumpToNextBookmark-Methode

Springt zur nächsten Textmarke, die der aktuellen Selektion oder der Position der Schreibmarke folgt.

Die Reihenfolge, in der die Textmarken angesprungen werden, richtet sich nach der Reihenfolge, in der die Textmarken in das Dokument eingefügt wurden.

KLASSE:
BaseTextSelection

SYNTAX:
JumpToNextBookmark()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
```

```
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

JumpToNextCell-Methode

Springt zur nächsten Tabellenzelle, die der aktuellen Position der Schreibmarke folgt. Steht die Schreibmarke in der letzten Zelle einer Tabelle, so wird eine neue Zeile an die Tabelle angehängt und die Schreibmarke in der ersten Zelle der neuen Zeile positioniert.

KLASSE:
BaseTextTable

SYNTAX:
JumpToNextCell()

Beispiel:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

JumpToNextFootnote-Methode

Bewegt den Cursor zum nächsten Fußnotenzeichen im Text.

Klasse:
Text

Syntax:
JumpToNextFootnote()

Rückgabe:
Boolean

Beispiel:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert(( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
```

```
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

JumpToNextFrame-Methode

Setzt den Cursor in den nächsten Rahmen.

KLASSE:
BaseText

SYNTAX:
JumpToNextFrame()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleFrames
Dim sFirstFrame As String, sLastFrame As String
Documents.Add("sWriter4")
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
sFirstFrame = Selection.FrameName
REM Beim zweiten Rahmen zwei Spalten einfügen
Selection.FrameColumns(2, "0.2cm", 2, 75, 3)
REM Rahmen Deselektieren
Selection.GotoAnchor
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.GotoAnchor
REM Name des letzten Frames ermitteln
sLastFrame =
Activewindow.Frames(Activewindow.Frames.count).FrameName
```

```
Selection.GotoAnchor
Selection.GoIntoFrame(sLastFrame)
REM Selektiere den ersten Rahmen
Selection.GotoFrame(sFirstFrame)
Selection.GotoAnchor
REM Springt in den ersten Rahmen
Selection.JumpToNextFrame
End Sub
```

JumpToNextRegion-Methode

Der Cursor springt von der aktuellen Cursorposition vor den nächsten Bereich. Dabei werden verschachtelte Bereiche nicht angesprungen.

Liefert TRUE, wenn ein Sprung ausgeführt wurde.

KLASSE:
BaseText

SYNTAX:
JumpToNextRegion()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleJumpRegion
Const sFirstRegionName = "Bereich 1"
Const sSecondRegionName = "Bereich 2"
Const sMessage_1 = "Diese Zeile ist der erste Bereich"
Const sMessage_2 = "Diese Zeile ist der zweite Bereich"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sFirstRegionName, "True", FALSE, FALSE)
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sSecondRegionName, "True", FALSE,
FALSE)
Selection.Escape
Selection.InsertPara
Selection.JumpToPrevRegion
Selection.JumpToNextRegion
Selection.JumpToRegion sFirstRegionName
End Sub
```

JumpToNextSelection-Methode

Bewegt die Schreibmarke zur nächsten Selektion einer Mehrfachselektion. Es gilt die Reihenfolge, in der die Selektionen angelegt wurden, nicht die Position innerhalb des Dokuments.

KLASSE:
BaseText

SYNTAX:
JumpToNextSelection()

BEISPIEL:

```
Sub ExampleReplaceAll
Const sWord_1 = "Beispielwort"
Const sWord_2 = "Geändertes Wort"
Dim iCount As Integer
Documents.Add( "Swriter4" )
For iCount=1 to 5
Selection.Insert sWord_1
Selection.InsertPara
next iCount
ActiveWindow.SearchAll sWord_1
For iCount=1 To Selection.SelectionCount
Selection.JumpToNextSelection
Next iCount
For iCount=1 To Selection.SelectionCount
Selection.JumpToPrevSelection
Next iCount
ActiveWindow.ReplaceAll sWord_1, sWord_2
End Sub
```

JumpToNextTable-Methode

Bewegt die Schreibmarke zum Anfang der nächsten Tabelle.

KLASSE:
BaseText

SYNTAX:
JumpToNextTable()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleJumpToNextTable
Const sTableName_1 = "ErsteTabelle"
Const sTableName_2 = "ZweiteTabelle"
Const sMessage_1 = "Dies ist die erste Tabelle"
```

```
Const sMessage_2 = "Dies ist die zweite Tabelle"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.InsertNamedTable( sTableName_1, 3, 3 )
Selection.JumpToEndOfDoc
Selection.InsertNamedTable( sTableName_2, 3, 3 )
Selection.JumpToPrevTable
Selection.Insert( sMessage_1 )
Selection.JumpToNextTable
Selection.Insert( sMessage_2 )
End Sub
```

JumpToPrevBookmark-Methode

Springt zu der Textmarke, die der aktuellen Selektion oder der Position der Schreibmarke vorangeht.

Die Reihenfolge, in der die Textmarken angesprungen werden, richtet sich nach der Reihenfolge, in der die Textmarken in das Dokument eingefügt wurden.

KLASSE:
BaseTextSelection

SYNTAX:
JumpToPrevBookmark()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
```

```
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

JumpToPrevCell-Methode

Springt zu der Tabellenzelle, die der Position der Schreibmarke vorangeht.

KLASSE:
BaseTextTable

SYNTAX:
JumpToPrevCell()

BEISPIEL:

```
Sub ExampleTableNavigate
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoEnd
Selection.Insert Selection.CellName
Selection.GotoStart
Selection.Insert Selection.CellName
Selection.JumpToNextCell
Selection.Insert Selection.CellName
Selection.GotoCell("C2", FALSE)
Selection.Insert Selection.CellName
Selection.JumpToPrevCell
Selection.Insert Selection.CellName
End Sub
```

JumpToPrevFootnote-Methode

Bewegt die Schreibmarke zur vorhergehenden Fußnote.

KLASSE:
Text

SYNTAX:
JumpToPrevFootnote()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleFootnote
Const sFootnote_1 = "Fußnote 1"
Const sFootnote_2 = "Fußnote 2"
Const sMessage_1 = "Dies ist erste Fußnote"
Const sMessage_2 = "Dieser Satz enthält die zweite Fußnote"
Const sMessage_3 = "Eine Fußnote die vom System benannt wird"
Const sMessage_4 = "Die Fußnoten sind folgendermaßen definiert"
Documents.Add("sWriter4")
Selection.Insert(( sMessage_1 )
Selection.InsertFootnote( sFootnote_1 )
ActiveWindow.PageUp
Selection.InsertPara
Selection.Insert( sMessage_1 )
Selection.InsertFootnote( sFootnote_2 )
Selection.JumpToFootnoteOrAnchor
Selection.JumpToPrevFootnote
Selection.JumpToNextFootnote
Selection.GotoEndOfDoc FALSE
Selection.InsertPara
Selection.Insert( sMessage_3 )
ActiveWindow.InsertFootnoteDirect
Selection.JumpToFootnoteOrAnchor
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_4
Selection.InsertPara
Selection.Insert "FtnHeight : " & ActiveWindow.FtnHeight
Selection.InsertPara
Selection.Insert "FtnPenWidth : " & ActiveWindow.FtnPenWidth
Selection.InsertPara
Selection.Insert "FtnTopDistance : " &  ActiveWindow.FtnTopDi-
stance
Selection.InsertPara
Selection.Insert "FtnBottomDistance : " & ActiveWindow.FtnBot-
tomDistance
Selection.InsertPara
Selection.Insert "FtnWidth : " & ActiveWindow.FtnWidth
Selection.InsertPara
```

```
Selection.Insert "FtnAdjust : " & ActiveWindow.FtnAdjust
Selection.InsertPara
Selection.Insert "FtnPenStyle : " & ActiveWindow.FtnPenStyle
End Sub
```

JumpToPrevRegion-Methode

Der Cursor springt von der aktuellen Cursorposition vor den vorherigen Bereich. Dabei werden verschachtelte Bereiche nicht angesprungen. Steht der Cursor innerhalb eines Bereichs, so springt der Cursor an dessen Anfang.

Liefert TRUE, wenn ein Sprung ausgeführt wurde.

KLASSE:
BaseText

SYNTAX:
JumpToPrevRegion()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleJumpRegion
Const sFirstRegionName = "Bereich 1"
Const sSecondRegionName = "Bereich 2"
Const sMessage_1 = "Diese Zeile ist der erste Bereich"
Const sMessage_2 = "Diese Zeile ist der zweite Bereich"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sFirstRegionName, "True", FALSE, FALSE)
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sSecondRegionName, "True", FALSE,
FALSE)
Selection.Escape
Selection.InsertPara
Selection.JumpToPrevRegion
Selection.JumpToNextRegion
Selection.JumpToRegion sFirstRegionName
End Sub
```

JumpToPrevSelection-Methode

Bewegt die Schreibmarke zur vorherigen Selektion einer Mehrfachselektion. Es gilt die Reihenfolge, in der die Selektionen angelegt wurden, nicht die Position innerhalb des Dokuments.

KLASSE:
BaseText

SYNTAX:
JumpToPrevSelection()

BEISPIEL:

```
Sub ExampleReplaceAll
Const sWord_1 = "Beispielwort"
Const sWord_2 = "Geändertes Wort"
Dim iCount As Integer
Documents.Add( "Swriter4" )
For iCount=1 to 5
Selection.Insert sWord_1
Selection.InsertPara
next iCount
ActiveWindow.SearchAll sWord_1
For iCount=1 To Selection.SelectionCount
Selection.JumpToNextSelection
Next iCount
For iCount=1 To Selection.SelectionCount
Selection.JumpToPrevSelection
Next iCount
ActiveWindow.ReplaceAll sWord_1, sWord_2
End Sub
```

JumpToPrevTable-Methode

Bewegt die Schreibmarke zum Anfang der vorhergehenden Tabelle.

KLASSE:
BaseText

SYNTAX:
JumpToPrevTable()

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleJumpToNextTable
Const sTableName_1 = "ErsteTabelle"
Const sTableName_2 = "ZweiteTabelle"
Const sMessage_1 = "Dies ist die erste Tabelle"
```

```
Const sMessage_2 = "Dies ist die zweite Tabelle"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.InsertNamedTable( sTableName_1, 3, 3 )
Selection.JumpToEndOfDoc
Selection.InsertNamedTable( sTableName_2, 3, 3 )
Selection.JumpToPrevTable
Selection.Insert( sMessage_1 )
Selection.JumpToNextTable
Selection.Insert( sMessage_2 )
End Sub
```

JumpToReference-Methode

Springt zur Referenz des Feldes, in dem sich die Schreibmarke befindet. Der Cursor muß direkt vor einem REFERENZ EINFÜGEN-Feld stehen und springt dann zu dem REFERENZ SETZEN-Zielfeld.

KLASSE:
BaseText

SYNTAX:
JumpToReference()

BEISPIEL:

```
Sub ExampleJumpToReference
Const sReference = "Referenz"
Const sRefName = "Referenz"
Const sMessage_1 = "In diesem Absatz befindet sich eine Refe-
renz."
Const sMessage_2 = "Die Referenz befindet sich auf Seite : "
Documents.Add( "sWriter4" )
Selection.Insert( sMessage_1 )
Selection.InsertPara
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertField( 13, 0, sRefName, "", 0 )
Selection.JumpToStartOfDoc
Activewindow.Search sReference
Selection.InsertField( 12, 1, sRefName, "", 0 )
Selection.UpdateFields
Selection.JumpToEndOfDoc
Selection.GoLeft( 1, FALSE )
Selection.JumpToReference
End Sub
```

JumpToRegion-Methode

Springt zu dem angegebenen Bereich.

KLASSE:
BaseTextSelection

SYNTAX:
JumpToRegion(Name As String)

RÜCKGABE:
Boolean

PARAMETER:
Name: Zeichenfolgeausdruck, der die Bezeichnung des Bereiches aufnimmt, zu dem gesprungen werden soll.

BEISPIEL:

```
Sub ExampleJumpRegion
Const sFirstRegionName = "Bereich 1"
Const sSecondRegionName = "Bereich 2"
Const sMessage_1 = "Diese Zeile ist der erste Bereich"
Const sMessage_2 = "Diese Zeile ist der zweite Bereich"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sFirstRegionName, "True", FALSE, FALSE)
Selection.GoRight 1, FALSE
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.GoUp 1, FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sSecondRegionName, "True", FALSE,
FALSE)
Selection.Escape
Selection.InsertPara
Selection.JumpToPrevRegion
Selection.JumpToNextRegion
Selection.JumpToRegion sFirstRegionName
End Sub
```

JumpToStartOfDoc-Methode

Setzt den Cursor von der aktuellen Position an den Anfang des Dokuments, wobei der Dokumentanfang durch den ersten Absatz definiert ist. Ist vor Anwendung dieser Methode Text markiert, dann wird diese Markierung aufgehoben und der gesamte Bereich ab Beginn der ursprünglichen Markierung bis zum Sprungziel selektiert.

Bei Anwendung dieser Methode kann sich der Cursor sowohl in einem Absatz oder in einer Tabelle oder einem Rahmen befinden, im Gegensatz zur GoToStartOfDoc-Methode springt JumpToStartOfDoc stets an den Anfang des aktuellen Dokuments.

Siehe auch: GoToStartOfDoc, JumpToEndOfDoc.

KLASSE:
BaseTextSelection

SYNTAX:
JumpToStartOfDoc()

BEISPIEL:

```
Sub ExampleJumpToEndOfDoc
Const sMessage_1 = "Hier beginnt das Dokument"
Const sMessage_2 = "Hier endet das Dokument"
Dim ICount As Integer
Documents.Add( "sWriter4" )
For iCount = 1 To 10
Selection.InsertPara
Next iCount
Selection.JumpToStartOfDoc
Selection.Insert sMessage_1
Selection.JumpToEndOfDoc
Selection.Insert sMessage_2
End Sub
```

JumpToURL-Methode

Setzt den Cursor vor den angegebenen Hyperlink.

siehe auch: URLFieldCount

KLASSE:
BaseTextEditView

SYNTAX:
JumpToURL(Index As Long)

Rückgabe:
Boolean

Parameter:
Index: Index des Hyperlinks, zu dem der Cursor bewegt werden soll. Die Hyperlinks sind fortlaufend durchnumeriert, wobei der erste Hyperlink mit dem Index 1 belegt ist. Die Reihenfolge richtet sich nicht nach der Position im Text.

Beispiel:

```
Sub ExampleJumpToURL
Const sMessage = "Der aktuelle Hyperlink hat folgende URL : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.SetHyperlink( "Adaptec", "http://www.Adaptec.com/",
"", 1 )
Selection.InsertPara
Selection.SetHyperlink( "IBM", "http://www.IBM.com/", "", 1 )
Selection.InsertPara
Selection.SetHyperlink( "Compaq", "http://www.compaq.com/", "",
1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
For iCount=1 to ActiveWindow.URLFieldCount
ActiveWindow.JumpToURL(iCount)
Print sMessage & Selection.LinkURL
next iCount
End Sub
```

JumpUpThisLevel-Methode

Springt innerhalb einer Aufzählung oder Numerierung zum vorangehenden Absatz der gleichen Ebene.

Klasse:
BaseTextList

Syntax:
JumpUpThisLevel()

Beispiel:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
```

```
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

Eigenschaften und Methoden mit "K"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "K" beginnen.

Eigenschaften und Methoden mit "L"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "L" beginnen.

Language-Eigenschaft

Bestimmt oder ermittelt die Sprache, die für die Schriftart des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen verwendet werden soll. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetLanguage gesetzte Sprache abgefragt.

Eine Abfrage der Language-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLanguage bestimmt wurde, daß nach einer bestimmten Sprache gesucht werden soll, d. h. IsLanguage = 2.

Siehe auch: SetLanguage, IsLanguage.

KLASSE:
ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Language

RÜCKGABE:
Long

PARAMETER:
Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Bei Text-Objekten:

[= Value]: Long-Integer-Ausdruck, der die Sprache angibt, die über die Language-Eigenschaft gesetzt oder ermittelt werden soll. Folgende Einstellungen stehen zur Verfügung:

255 - Keine

1030 - Dänisch

1031 - Deutsch

2055 - Deutsch (CH)

2057 - Englisch (UK)

1033 - Englisch (US)

1035 - Finnisch

1036 - Französisch

1040 - Italienisch

1043 - Niederländisch

1044 - Norwegisch Bokmal

2068 - Norwegisch Nynorsk

2070 - Portugiesisch

1053 - Schwedisch

1034 - Spanisch

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
```

```
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

LaunchStarImage-Methode

Startet StarImage zur Bearbeitung der im Dokument selektierten Grafik. Dabei ändert sich der Rahmen der Grafik, und die Symbolleisten von StarImage werden eingeblendet.

Diese Methode funktioniert nur, wenn ein importiertes Pixelbild selektiert ist, andernfalls erscheint eine Fehlermeldung.

KLASSE:
BaseTextGraphic

SYNTAX:
LaunchStarImage()

BEISPIEL:

```
Sub ExampleGraphicEdit
Dim sGrfPath as String
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows" )
Selection.LaunchStarImage
End Sub
```

Left-Eigenschaft

Bestimmt oder ermittelt den Wert für das Skalieren der Ansicht der selektierten Grafik an der linken Seite. Negative Werte beschneiden die Ansicht der Grafik, positive verbreitern den Rand zwischen Rahmen und Grafik und verkleinern die Grafik dabei.

Siehe auch: Right, Top und Bottom.

KLASSE:
BaseTextGraphic

SYNTAX:
Left

RÜCKGABE:
Long

PARAMETER:
[= Value]: Gibt den Wert an, um den die Grafik an der linken Seite geändert werden soll. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExampleGraphicCrop
Dim sGrfPath as String
Dim iCount As Integer
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Wait 3000
With Selection  REM in zwei Stufen wird das Bild verkleinert
For iCount = 2 to -1 Step -1
   .Bottom = Trim(Str(iCount) & "cm")
   .Top= Trim(Str(iCount) & "cm")
   .Left= Trim(Str(iCount) & "cm")
   .Right= Trim(Str(iCount) & "cm")
   Wait 3000
Next iCount
End With
Selection.Escape
End Sub
```

LeftMargin-Eigenschaft

Bestimmt oder ermittelt den Abstand vom linken Rand des selektierten oder aktiven Objekts zu seiner Umgebung. Dieser Abstand ist für verschiedene Objekte unterschiedlich definiert: Bei einem TextEditView-Objekt ist es der Abstand bis zur linken Papierkante, bei einem TextTable-Objekt der bis zum linken Seitenrand. Bei Objekten der Klassen TextFrame, TextGraphic, TextOLEObject ist es der Abstand bis zu dem danebenliegenden umlaufenden Text. Ist ein Absatz selektiert (Paragraph, TextInTable, ListInTable), so liefert LeftMargin den linken Einzug des Absatzes.

Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLeftRightParaMargin gesetzte linke Einzug abgefragt.

Eine Abfrage der LeftMargin-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLeftRightParaMargin bestimmt wurde, daß nach einem bestimmten Absatzeinzug gesucht werden soll, d. h. IsLeftRightParaMargin = 2.

Siehe auch: BottomMargin, RightMargin, TopMargin.

Für SearchAttributes, ReplaceAttributes:

SetLeftRightParaMargin, IsLeftRightParaMargin, IndentFirstLine.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
LeftMargin

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der den jeweiligen Abstand angibt. Der Wert kann auch mit einer Maßeinheit als Zeichenfolge übergeben werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleParagraphMargin
REM Das Beispiel setzt den aktuellen Absatz optisch ab, z.B.
für Zitate.
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert("Dieser Absatz erscheint beid-
seitig eingerückt und mit einem Abstand zum vorigen und
folgenden Absatz")
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert("Hier geht es weiter")
ActiveWindow.Selection.GoUp 1
With ActiveWindow.Selection.Paragraph
.LeftMargin  = "3cm"
.RightMargin = "3cm"
.TopMargin   = "1cm"
.BottomMargin   = "1cm"
End With
End Sub
```

LineColor-Eigenschaft

Bestimmt oder ermittelt die Farbe der Umrandung des selektierten Objekts.

Die LineColor-Eigenschaft kann die Linienfarbe eines Objekts nur dann ändern, wenn das Objekt zuvor selektiert wurde. Bei einer Tabelle funktioniert diese Eigenschaft, sobald der Cursor in einer Tabellenzelle steht. Allerdings wird dann die Linienfarbe nur bei den mit der jeweiligen Zelle korrespondierenden Linien geändert. Um die Farbe aller Linien einer Tabelle zu ändern, muß die Tabelle vor Anwendung der LineColor-Eigenschaft komplett selektiert werden.

KLASSE:
BaseTextFrame

Syntax:
LineColor

Rückgabe:
Long

Parameter:
[= Farbe]: Numerischer Ausdruck, der die Farbe angibt, mit der die Umrandung erfolgen soll.

Beispiel:

```
Sub ExampleLineColor
Dim sTabName As String
Documents.Add "sWriter4"
Selection.InsertTable(4,3)
InsertTable = Selection.TableName
Selection.JumpToStartOfDoc
Selection.GotoTable(TableName$)
Selection.SelectAll
Selection.LineColor = RGB(255,0,0) REM Den rahmen Rot einfärben
Selection.Escape
Selection.JumpToStartOfDoc
Selection.Insert("Die Tabelle hat einen roten Rand.")
End Sub
```

LineDash-Eigenschaft

Liefert oder setzt den Linienstil des selektierten Zeichnungsobjekts.

Klasse:
TextDraw

Syntax:
LineDash

Rückgabe:
String

Parameter:
[= Style]: Zeichenfolge, die den Linienstil des selektierten Zeichnungsobjekts angibt.

Beispiel:

```
REM Es muß bereits ein Zeichenobject eingefügt worden sein.
Const sLineStyle = "Linienstil9"
Selection.LineDash=sLineStyle
```

LineHeight-Eigenschaft

Bestimmt oder ermittelt den Wert des Mindestzeilenabstandes im aktuellen Absatz. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLineSpacing gesetzte Mindestzeilenabstand oder der feste Zeilenabstand abgefragt. Ob sich diese Eigenschaft beim Suchen und Ersetzen von Attributen auf den minimalen oder den festen Zeilenabstand bezieht, hängt von den Eigenschaften InterLineSpace und LineSpace ab.

LineHeight verträgt keine Angabe in cm, sondern nur in twips.

Eine Abfrage der LineHeight-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLineSpacing bestimmt wurde, daß nach einem bestimmten Zeilenabstand gesucht werden soll, d. h. IsLineSpacing = 2.

Siehe auch: SetLineSpacing, IsLineSpacing, LineSpace, PropLineHeight, InterLineSpace, InterSpace.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
LineHeight

RÜCKGABE:
Long

PARAMETER:
Bei Absätzen:

[= Value]: Long-Integer-Ausdruck, der den Mindestzeilenabstand angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge gesetzt werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleSetLineSpacing
Const sMessage_1 = "Es wird ein Absatz mit einer Zeilehöhe von
1.5 cm, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Zeilenhöhe auf
1.5 cm gesetzt"
Documents.Add( "sWriter4" )
```

```
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.LineSpace = 1
Selection.Paragraph.PropLineHeight = 200
Selection.Paragraph.LineHeight = 852 REM 1,5cm
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLineSpacing = 2
ActiveWindow.SearchAttributes.SetLineSpacing( 1, 852, 0, 0, 0 )
If ActiveWindow.SearchAttributes.LineHeight = 852 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

Lines-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Textzeilen, über die eine Initiale dargestellt wird.

Siehe auch: Count, Distance.

KLASSE:
DropCap

SYNTAX:
Lines

RÜCKGABE:
Integer

PARAMETER:
[= IniLines]: Numerischer Ausdruck vom Typ Integer, der die Anzahl der Textzeilen angibt, über die eine Initiale dargestellt werden soll. Die Initiale kann maximal über 9 Zeilen dargestellt werden. Wird IniLines mit dem Wert 1 übergeben, wird die Darstellung der Initiale ausgeschaltet.

BEISPIEL:

```
Sub ExampleDropCap
Documents.Add("sWriter4")
Selection.Insert "Das erste Zeichen"
Selection.InsertLineBreak
Selection.Insert "wird über drei"
Selection.InsertLineBreak
Selection.Insert "Zeilen dargestellt"
Selection.InsertLineBreak
Selection.GotoStartOfDoc FALSE
Selection.Paragraph.DropCap.Count = 1
```

```
Selection.Paragraph.DropCap.Lines = 3
Selection.Paragraph.DropCap.Distance = "0.5cm"
End Sub
```

LineSpace-Eigenschaft

Bestimmt oder ermittelt die Zeilenhöhe im aktuellen Absatz. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLineSpacing gesetzte Zeilenabstand abgefragt.

Eine Abfrage der LineSpace-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLineSpacing bestimmt wurde, daß nach einem bestimmten Zeilenabstand gesucht werden soll, d. h. IsLineSpacing = 2.

Siehe auch: SetLineSpacing, IsLineSpacing, LineHeight, PropLineHeight, InterLineSpace, InterSpace.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
LineSpace

RÜCKGABE:
Integer

PARAMETER:
Bei Absätzen:

[Value]: Integer-Ausdruck, der die Zeilenhöhe des aktuellen Absatzes angibt. Folgende Werte sind möglich:

0 - Automatische Zuweisung der Zeilenhöhe. Die Zeilenhöhe entspricht der Zeichenhöhe der größten Schriftart im Absatz.

1 - Größtmögliche Zeilenhöhe

2 - Mindestzeilenhöhe

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleSetLineSpacing
Const sMessage_1 = "Es wird ein Absatz mit einer Zeilenhöhe von
1.5 cm gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Zeilenhöhe auf
1.5 cm gesetzt"
```

```
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.LineSpace = 1
Selection.Paragraph.PropLineHeight = 200
Selection.Paragraph.LineHeight = 852 REM 1.5cm in twips
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLineSpacing = 2
ActiveWindow.SearchAttributes.SetLineSpacing( 1, 852, 0, 0, 0 )
If ActiveWindow.SearchAttributes.LineHeight = 852 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

LineText-Eigenschaft

Liefert den Inhalt der aktuellen Zeile, in der der Cursor steht, als Zeichenfolge.

KLASSE:
BaseText

SYNTAX:
LineText

RÜCKGABE:
String

PARAMETER:
Rückgabe: der Inhalt der aktuellen Zeile.

BEISPIEL:

```
Sub ExampleTableCalc
Const sMessage_1 = "Die Formel aus der Zelle B2 berechnet : "
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoCell("A2", FALSE)
Selection.Insert "2"
Selection.GotoCell("A3", FALSE)
Selection.Insert "3"
Selection.GotoCell("B2", FALSE)
Selection.Formula = "<A2>+<A3>"
Selection.GotoCell("A2", FALSE)
Selection.Insert "4"
Selection.Calc
Selection.GotoCell("B2", FALSE)
Print sMessage_1 & Selection.LineText
End Sub
```

LinkDialog-Methode

Ruft den Dialog auf, mit dem die Einstellungen für VERKNÜPFUNGEN bestimmt werden.

KLASSE:
BaseTextEditView

SYNTAX:
LinkDialog()

BEISPIEL:

```
Sub ExampleLinkDialog
Dim sGrfPath As String
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
ActiveWindow.LinkDialog
End Sub
```

LinkName-Eigenschaft

Liefert den Namen des Hyperlinks an der aktuellen Cursorposition. Der Cursor muß innerhalb des Hyperlinks stehen. Wurde der Hyperlink als Button definiert, muß letzterer selektiert sein, sonst wird ein Leerstring zurückgeliefert.

KLASSE:
BaseText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
LinkName

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleLinkTyp
Const sMessage = "Hier geht es zur internationalen Homepage
der: "
Const sTarget = "Neues_Dokument" REM Hyperlink in ein neues
benanntes Fenster öffnen
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert(sMessage)
Selection.SetHyperlink( "Star Division GmbH", "http://www.star-
```

```
division.com",sTarget , 1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GoRight (Len( sMessage ) +5 ) REM Textmarke in den
Hyperlink stellen
MsgBox Selection.LinkName,0,"Der Text des Hyperlinks lautet"
MsgBox Selection.LinkTarget,0,"Das Ziel des Hyperlinks lautet"
MsgBox "Der Hyperlink ist ein " &
IIF(Selection.LinkType=2,"Button","Text"),0,"Hyperlinktyp"
End Sub
```

LinkTarget-Eigenschaft

Liefert die Einstellungen des Zeilrahmens (Frame) für den Hyperlink an der aktuellen Cursorposition. Der Cursor muß innerhalb des Hyperlinks stehen. Wurde der Hyperlink als Button definiert, muß letzterer selektiert sein, sonst wird ein Leerstring zurückgeliefert.

KLASSE:
BaseText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
LinkTarget

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleLinkTyp
Const sMessage = "Hier geht es zur internationalen Homepage
der: "
Const sTarget = "Neues_Dokument" REM Hyperlink in ein neues
benanntes Fenster öffnen
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert(sMessage)
Selection.SetHyperlink( "Star Division GmbH", "http://www.star-
division.com",sTarget , 1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GoRight (Len( sMessage ) +5 ) REM Textmarke in den
Hyperlink stellen
MsgBox Selection.LinkName,0,"Der Text des Hyperlinks lautet"
MsgBox Selection.LinkTarget,0,"Das Ziel des Hyperlinks lautet"
MsgBox "Der Hyperlink ist ein " &
IIF(Selection.LinkType=2,"Button","Text"),0,"Hyperlinktyp"
End Sub
```

LinkType-Eigenschaft

Liefert den Typ des eingefügten Hyperlinks. Der Cursor muß innerhalb des Hyperlinks stehen. Wurde der Hyperlink als Button definiert, muß letzterer selektiert sein, sonst wird eine "0" zurückgeliefert.

KLASSE:
BaseText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
LinkType

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

Return:

0 - kein Hyperlink

1 - Hyperlink als Text

2 - Hyperlink als Button

BEISPIEL:

```
Sub ExampleLinkTyp
Const sMessage = "Hier geht es zur internationalen Homepage
der: "
Const sTarget = "Neues_Dokument" REM Hyperlink in ein neues
benanntes Fenster öffnen
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert(sMessage)
Selection.SetHyperlink( "Star Division GmbH", "http://www.star-
division.com",sTarget , 1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GoRight (Len( sMessage ) +5 ) REM Textmarke in den
Hyperlink stellen
MsgBox Selection.LinkName,0,"Der Text des Hyperlinks lautet"
MsgBox Selection.LinkTarget,0,"Das Ziel des Hyperlinks lautet"
MsgBox "Der Hyperlink ist ein " &
IIF(Selection.LinkType=2,"Button","Text"),0,"Hyperlinktyp"
End Sub
```

LinkURL-Eigenschaft

Liefert die URL-Adresse des Hyperlinks an der aktuellen Cursorposition. Der Cursor muß innerhalb des Hyperlinks stehen. Wurde der Hyperlink als Button definiert, muß letzterer selektiert sein, sonst wird ein Leerstring zurückgeliefert.

KLASSE:
BaseText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
LinkURL

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleJumpToURL
Const sMessage = "Der aktuelle Hyperlink hat folgende URL : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.SetHyperlink( "Adaptec", "http://www.Adaptec.com/",
"", 1 )
Selection.InsertPara
Selection.SetHyperlink( "IBM", "http://www.IBM.com/", "", 1 )
Selection.InsertPara
Selection.SetHyperlink( "Compaq", "http://www.compaq.com/", "",
1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
For iCount=1 to ActiveWindow.URLFieldCount
ActiveWindow.JumpToURL(iCount)
Print sMessage & Selection.LinkURL
next iCount
End Sub
```

LoadStyles-Methode

Lädt die Vorlagen aus einem externen StarWriter-Dokument in das aktuelle. Über die Parameter kann definiert werden, ob die im Dokument vorhandenen Vorlagen überschrieben werden sollen.

Ohne Übergabeparameter ruft diese Methode den Dialog VORLAGEN LADEN auf.

KLASSE:
BaseTextDocument

SYNTAX:
LoadStyles(File As String, Select As Long)

PARAMETER:
File: Laufwerk, Pfad und Dateiname des Dokumentes, aus dem die Vorlagen importiert werden sollen.

Select: Gibt an, welche Vorlagenart geladen werden soll. Folgende Werte sind möglich:

1 - Textvorlagen (Absatz- und Zeichenvorlagen)

2 - Rahmenvorlagen

4 - Seitenvorlagen

8 - Überschreiben

Eine Kombination der verschiedenen Möglichkeiten wird durch Addition der einzelnen Werte erreicht:

Wird im Parameter Select eine 15 übergeben, werden alle Text-, Rahmen- und Seitenvorlagen übernommen und die vorhandenen überschrieben.

Wird in Select eine 7 übergeben, werden alle Text-, Rahmen- und Seitenvorlagen übernommen, die vorhandenen werden nicht überschrieben.

BEISPIEL:

```
Sub ExampleLoadStyles
Const sTemp="Fetter Text"  REM Vorlagenname
Const sDoc ="DocFett.SdW"  REM Dokumentname
REM zuerst ein Dokument mit einer Absatzvorlage erstellen
Documents.Add "sWriter4"
Selection.Insert("Es wird eine Absatzvorlage " & sTemp & "
erstellt.")
Selection.GoToStartOfLine TRUE
Selection.Font.Bold=8
Selection.Escape
Selection.StyleNewByExample( sTemp, 2)
Selection.InsertPara
ActiveDocument.SaveAs Application.PathSettings.Work & "\" &
sDoc
ActiveDocument.Close "",False
REM jetzt ein Dokument erstellen, in das die erstellte Vorlage
importiert wird.
Documents.Add "sWriter4"
ActiveDocument.LoadStyles( Application.PathSettings.Work & "\"
& sDoc, 1 )
Selection.Insert("Es wird die zuvor erstellte Absatzvorlage  "
```

```
& sTemp & "  importiert und angewandt.")
Selection.StyleApply( sTemp, 2 )
Selection.InsertPara
End Sub
```

Location-Eigenschaft

Bestimmt oder ermittelt die Position des Schattens, der dem selektierten Bereich zugewiesen ist.

KLASSE:
Shadow, BaseTextEditView

SYNTAX:
Location

RÜCKGABE:
Integer

PARAMETER:
[= ShadPos]: Integer-Ausdruck, der die Position des Schattens angibt. Folgende Werte sind möglich:

0 - Kein Schatten

1 - Schatten links/oben

2 - Schatten rechts/oben

3 - Schatten links/unten

4 - Schatten rechts/unten

BEISPIEL:

```
Sub ExampleShadow
Documents.Add("sWriter4")
Selection.InsertFrame(1, "3cm", "3cm", "4cm", "4cm")
Selection.Shadow.Style = 2
Selection.Shadow.Fillcolor = RGB(255, 255, 0) REM Gelb
Selection.Shadow.Location = 2
End Sub
```

LockPaint-Methode

Schaltet die Aktualisierung der Bildschirmanzeige des Dokumentenfensterinhalts aus oder ein. Ist LockPaint aktiv, so werden Eingaben in dem Dokument nicht dargestellt. Die Methode erlaubt eine Verschachtelung, so daß ein paarweises Setzen zwingend ist. Um LockPaint zu deaktivieren, muß die Methode genauso oft zurückgesetzt werden wie sie zuvor aktiviert wurde.

Siehe auch: IsPaintLocked

KLASSE:
BaseTextEditView

SYNTAX:
LockPaint(Lock As Boolean)

PARAMETER:
Lock: Boolescher Ausdruck, der angibt, ob das Neuzeichnen des Dokumenteninhalts aus- (True) oder eingeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleLockPaint
Const sMessage_1 = "Den Aufbau der beiden Absätze ist nicht zu
sehen."
Const sMessage_2 = "Das ist der zweite Absatz."
Const sError = "Das Abschalten der Bildschirmaktualisierung hat
nicht funktioniert"
Documents.Add( "sWriter4" )
ActiveWindow.LockPaint(TRUE)
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
If ActiveWindow.IsPaintLocked = FALSE Then
Print sError
End If
ActiveWindow.LockPaint(FALSE)
End Sub
```

LogicalPageNumber-Eigenschaft

Liefert die logische Seitennumer der aktuellen Seite, in der sich der Cursor befindet. Die logische Seitennummer ist manipulierbar. Solange Sie die automatische Seitennumerierung, die sich an den tatsächlich vorhandenen Seiten orientiert, nicht manuell ändern, stimmt die logische Seitennummer mit der physikalischen überein. Haben Sie aber beispielsweise der ersten Seite in Ihrem Dokument die Seitennummer 3 zugewiesen, so liefert Logical PageNumber für die erste Seite den Wert 3.

Siehe auch: PageNumber, PageOffset, PageOffsetDialog, InsertPageNumberField.

KLASSE:
BaseText

SYNTAX:
LogicalPageNumber

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

Eigenschaften und Methoden mit "M"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "M" beginnen.

ManualBreak-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für manuelle Umbrüche.

Klasse:
ViewSettings

Syntax:
ManualBreak

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob manuelle Umbrüche angezeigt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

MaxHyphens-Eigenschaft

Bestimmt oder ermittelt die maximale Anzahl aufeinanderfolgender Silbentrennungen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetHyphenZone gesetzte Option zur Silbentrennung abgefragt.

Siehe auch: SetHyphenZone, IsHyphenZone, Hyphen, PageEnd, MinLead, MinTrail.

KLASSE:
HyphenZone, ReplaceAttributes, SearchAttributes

SYNTAX:
MaxHyphens

RÜCKGABE:
Integer

PARAMETER:
Bei HyphenZone-Objekten:

[= Value]: Integer-Ausdruck, der die maximale Anzahl aufeinanderfolgender Silbentrennungen angibt.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
sub ExampleHyphenZone
Dim iCount as Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umgebrochen"
Selection.GotoStartOfLine FALSE
For iCount=1 to 53
Selection.Insert "  "
Next iCount
end sub
```

MergeCells-Methode

Verbindet die selektierten Tabellenzellen zu einer Zelle.

KLASSE:
BaseTextTable

SYNTAX:
MergeCells()

BEISPIEL:

```
Sub ExampleMergeCell
Documents.Add("sWriter4")
Selection.InsertTable( 4 , 4 )
REM Die Zellen A1 bis D1 werden horizontal verbunden
Selection.GotoCell("D1", TRUE)
Selection.MergeCells
End Sub
```

MergeDialog-Methode

Ruft den Dialog SERIENBRIEF auf.

KLASSE:
StarWriter

SYNTAX:
MergeDialog()

BEISPIEL:

```
Sub ExampleMergeDialog
Documents.Add("swriter4")
ActiveModule.MergeDialog
end sub
```

MetricUnit-Eigenschaft

Bestimmt oder ermittelt die Maßeinheit für das aktive Modul.

KLASSE:
StarWriter

SYNTAX:
MetricUnit

RÜCKGABE:
Long

PARAMETER:
Wert - Einheit

1 - mm

2 - cm

6 - Punkt

7 - Pica

8 - Inch (Zoll)

BEISPIEL:

```
Sub ExampleMetricUnit
Const sMessage="Die Textbreite ist "
Documents.Add "sWriter4"
StarWriter.MetricUnit = 1
Print sMessage & ActiveWindow.PageWidth
StarWriter.MetricUnit = 2
Print sMessage & ActiveWindow.PageWidth
StarWriter.MetricUnit = 6
Print sMessage & ActiveWindow.PageWidth
StarWriter.MetricUnit = 8
Print sMessage & ActiveWindow.PageWidth
End Sub
```

MinLead-Eigenschaft

Bestimmt oder ermittelt die Mindestanzahl der Zeichen, die einer Worttrennung voranstehen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetHyphenZone gesetzte Option zur Silbentrennung abgefragt.

Siehe auch: SetHyphenZone, IsHyphenZone, Hyphen, PageEnd, MinTrail, MaxHyphens.

KLASSE:
HyphenZone, ReplaceAttributes, SearchAttributes

SYNTAX:
MinLead

RÜCKGABE:
Integer

PARAMETER:
Bei HyphenZone-Objekten:

[= Value]: Integer-Ausdruck, der die Mindestanzahl der Zeichen angibt, die vor einem Trennstrich angeordnet sein müssen.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
sub ExampleHyphenZone
Dim iCount as Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umbrochen"
Selection.GotoStartOfLine FALSE
For iCount=1 to 53
Selection.Insert "  "
Next iCount
end sub
```

MinTrail-Eigenschaft

Bestimmt oder ermittelt die Mindestanzahl der Zeichen, die nach einer Worttrennung am Anfang der nächsten Zeile stehen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetHyphenZone gesetzte Option zur Silbentrennung abgefragt.

Siehe auch: SetHyphenZone, IsHyphenZone, Hyphen, PageEnd, MinLead, MaxHyphens.

KLASSE:
HyphenZone, ReplaceAttributes, SearchAttributes

SYNTAX:
MinTrail

RÜCKGABE:
Integer

PARAMETER:
Bei HyphenZone-Objekten:

[= Value]: Integer-Ausdruck, der die Mindestanzahl der Zeichen angibt, die einem Trennstrich folgen müssen.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Beispiel:

```
sub ExampleHyphenZone
Dim iCount as Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umbrochen"
Selection.GotoStartOfLine FALSE
For iCount=1 to 53
Selection.Insert "  "
Next iCount
end sub
```

MoveDown-Methode

Verschiebt die selektierte Aufzählung oder Numerierung um eine Position nach unten.

Klasse:
BaseText

Syntax:
MoveDown()

Beispiel:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
```

```
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

MoveDownSubItems-Methode

Verschiebt die selektierte Aufzählung oder Numerierung einschließlich ihrer Unterpunkte um eine Position nach unten.

KLASSE:
BaseTextList

SYNTAX:
MoveDownSubItems()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
```

```
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

MoveRowCol-Methode

Bewirkt eine schrittweise Vergrößerung oder Verkleinerung von Tabellenzellen, -zeilen oder -spalten.

Im Office ist dieses Verhalten durch entsprechende Tastatursteuerungen zu erreichen.

KLASSE:
BaseTextTable

SYNTAX:
MoveRowCol(Hori As Boolean, Left_Top As Boolean, Greater As Boolean, CellOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
Hori: Gibt an, ob sich die Größenänderung auf die aktuelle Spalte (True) oder die aktuelle Zeile (False) auswirkt.

[Left_Top]: Gibt an, ob sich die Größenänderung auf die Zellen vor (True) oder hinter (False) der aktuellen Zelle auswirkt. Bei der Größenänderung einer Spalte sind dann die linken (True) oder rechten (False) Spalten betroffen, bei einer Zeile die oberen (True) oder unteren (False) Zeilen.

Wird dieser Parameter nicht übergeben, wird die Standardeinstellung Left_Top = True genommen.

[Greater]: Gibt an, ob eine Vergrößerung (True) oder eine Verkleinerung (False) erfolgen soll. Wird dieser Parameter nicht übergeben, findet eine Vergrößerung statt (Greater = True).

[CellOnly]: Gibt an, ob sich die Größenänderung nur auf eine einzelne Zelle beziehen soll (True) oder auf die aktuelle Zeile oder Spalte (False). Ist CellOnly = False, dann gibt der erste Übergabeparameter an, ob sich diese Methode auf eine Spalte (Hori = True) oder eine Zeile (Hori = False) auswirkt.

Der Parameter CellOnly ist optional. Wird er nicht übergeben, wirkt sich eine Größenänderung standardmäßig auf die aktuelle Zeile oder Spalte aus (CellOnly = False).

Beispiel:

```
Sub ExampleMoveRowCol
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
REM Die erste Zeile der Tabelle vergrößern
Selection.MoveRowCol(FALSE, TRUE, TRUE, FALSE)
End Sub
```

MoveUp-Methode

Verschiebt die selektierte Aufzählung oder Numerierung um eine Position nach oben.

Klasse:
BaseText

Syntax:
MoveUp()

Beispiel:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
```

```
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

MoveUpSubItems-Methode

Verschiebt die selektierte Aufzählung oder Numerierung einschließlich ihrer Unterpunkte um eine Position nach oben.

KLASSE:
BaseTextList

SYNTAX:
MoveUpSubItems()

BEISPIEL:

```
Sub ExampleNumberingLevels
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Dies ist der vierte Eintrag der Numerie-
rung"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.InsertPara
Selection.Insert sMessage_4
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 2, FALSE )
Selection.DecrementLevel
Selection.JumpToStartOfDoc
Selection.JumpDownThisLevel
Selection.DecrementSubLevels
Selection.IncrementSubLevels
Selection.JumpUpThisLevel
Selection.MoveDownSubItems
Selection.MoveUpSubItems
Selection.MoveDown
```

```
Selection.MoveUp
Selection.GoDown( 1, FALSE )
Selection.IncrementLevel
Selection.JumpToEndOfDoc
Selection.InsertNeutralParagraph
End Sub
```

MultiLine-Eigenschaft

Bestimmt oder ermittelt, ob ein Textfeld einen Zeilenumbruch darstellen kann.

KLASSE:
BaseTextDrawControl

SYNTAX:
MultiLine

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob eine mehrzeilige Darstellung möglich ist (True) oder nicht (False).

BEISPIEL:

```
sub ExampleMultiline
Dim sName As String
Documents.add "sWriter4"
Activewindow.InsertVCControl( 8, "3cm", "3cm", "4cm", "2cm")
sName = Activewindow.Controls(1).ObjectName
Activewindow.Controls(sName).Tag = "Test"
Activewindow.Controls(sName).Multiline = True
Activewindow.Controls(sName).ControlObject.Text="Das ist ein
Text der in dem Multiline Edit-Feld steht."
Selection.Escape
end sub
```

Eigenschaften und Methoden mit "N"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "N" beginnen.

NewField-Methode

Legt im Dialog Feldbefehl einfügen ein neues benutzerdefiniertes Feld an. Diese Methode kann z. B. dazu genutzt werden, dem Anwender bestimmte Felder zur Verfügung zu stellen, die er in sein Textdokument einfügen und dann mittels eines Makros weiterverarbeiten kann.

Siehe auch: InsertField

Klasse:
BaseTextSelection

Syntax:
NewField(Name As String)

Parameter:
Name: Ein Zeichenfolgeausdruck, der den Namen des neuen Feldes angibt.

Beispiel:

```
Sub ExampleDefineUserFields
Const sFieldName = "NeuesFeld"
Const sMessage = "Hier erscheint ein Feld "
Documents.Add( "swriter4" )
With ActiveWindow.Selection
.Insert( sMessage )
.NewField( sFieldName )
.SetUserFieldSubType( sFieldName, 0 )
.SetUserFieldValue( sFieldName, "3,1415" )
End With
End Sub
```

NewGlobalDoc-Methode

Erzeugt aus einem bestehenden Dokument ein neues Globaldokument. Ohne Übergabeparameter erscheint der Dialog Name und Pfad des Globaldokuments zum Speichern des Dokuments als Globaldokument.

Klasse:
TextDocument

Syntax:
NewGlobalDoc(Path As String)

Parameter:
[Path]: Gibt Namen und Pfad des neuen Globaldokuments an.

Beispiel:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\GLOBAL.SGL"
Const sClientDocName = "\CLIENT.SDW"
Const sMessage_1 = "Der Titel des Globaldokumentsbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentsbereichs : "
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

NewGlossary-Methode

Fügt einer AutoText-Gruppe einen neuen AutoText-Eintrag hinzu. Dabei wird der aktuell selektierte Text aus dem Dokument als Textbaustein in die AutoText-Datei übernommen.

Siehe auch: ExpandGlossary, InsertGlossary, SetActGlossaryGroup.

KLASSE:
BaseText

SYNTAX:
NewGlossary(Group As String, Name As String, ShortName As String)

PARAMETER:
Group: Ein Zeichenfolgeausdruck, der angibt, welcher Gruppe bzw. welchem Bereich der neue Eintrag hinzugefügt werden soll.

Name: Ein Zeichenfolgeausdruck, der den Namen des AutoText-Eintrags angibt.

ShortName: Ein Zeichenfolgeausdruck, der die Kurzbezeichnung des Eintrages angibt.

Beispiel:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
```

```
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

NewStyle-Methode

Ruft den Dialog der angegebenen Vorlagenart auf und legt eine neue Vorlage im Vorlagenkatalog an. Sie sehen die Vorlage unter "Benutzervorlagen".

Bei erfolgreicher Ausführung wird der Wert 1 zurückgeliefert, im Falle eines Fehlers der Wert 0.

KLASSE:
BaseTextDocument

SYNTAX:
NewStyle(Param As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Param: Name der Vorlage, die neu erzeugt werden soll.

Family: Bestimmt die Vorlagenfamilie, der die neue Vorlage zugehören soll. Folgende Werte sind möglich:

1 - Zeichenvorlage

2 - Absatzvorlage

4 - Rahmenvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExamplePageStyle
Const sTemp="Meine neue Seite"
Documents.Add "sWriter4"
REM Erst mal eine neue Seitenvorlage anlegen, da Seitenattri-
bute immer direkt auf die Vorlage wirken.
ActiveDocument.NewStyle( sTemp, 8)
REM ... und deswegen auch gleich anwenden, damit bereits defi-
nierte Vorlagen nicht überschrieben werden
Selection.StyleApply( sTemp, 8)
Selection.Insert("Hier können Sie Seitenformatierungen
ändern.")
Selection.InsertPara
Selection.PageDialog REM Format-Seite aufrufen
Selection.InsertPageBreak
Selection.Insert("Die Formatierung wurden in die Seitenvorlage
" &_
```

```
Activewindow.PageStyleName &_
" übertragen."
End Sub
```

NewWindow-Methode

Erzeugt ein neues Dokumentfenster.

KLASSE:
WebSourceView, BaseTextEditView

SYNTAX:
NewWindow()

BEISPIEL:

```
Sub ExampleNewWindow
Const sMessage=" Fenster "
Dim sDoc As String
Documents.Add "sWriter4"
sDoc = ActiveDocument.Name
For iCount = 1 to 10
Selection.Insert(sMessage & iCount)
ActiveWindow.NewWindow
Next iCount
Windows.Tile
Wait 2000
Documents(sDoc ).Close "",FALSE
End Sub
```

NextFieldByType-Methode

Springt im Dokument zum nächsten, durch ID-Nummer und Name gekennzeichneten Feldtyp.

Siehe auch: GetFieldTypeCount, GetFieldTypeID, GetFieldTypeName, PrevFieldByType.

KLASSE:
BaseText

SYNTAX:
NextFieldByType(Type As Long, Name As String)

RÜCKGABE:
Boolean

PARAMETER:
Type: Long-Integer-Ausdruck, der die ID-Nummer des Feldtyps angibt, zu dem gesprungen werden soll.

Name: Zeichenfolgeausdruck, der den Namen des Feldtyps angibt, zu dem gesprungen werden soll.

Dieser Parameter ist nur für solche Feldtypen anzugeben, wo ein Name definiert wurde, also z.B. Nummernkreise. Hier ist er allerdings zwingend erforderlich, damit zum nächsten Feld gesprungen werden kann.

Bei Feldtypen, die keine Namensdefinition haben, also z.B. Absender-Felder, springt man mit dieser Methode automatisch zum nächsten Feld des Typs Absender, wenn kein Name oder ein Leerstring übergeben wird.

Beispiel:

```
Sub ExampleNextFieldByType
Const sVar ="Wichtig"
Dim sContent As String
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert(" Fülltext ")
.InsertField( 8, 1,sVar , "10" , 0 )
.Insert(" Fülltext ")
.InsertField( 8, 1, sVar, "100" , 0 )
.Insert(" Fülltext ")
.InsertField( 8, 1, sVar, "1000" , 0 )
.InsertPara
End With
ActiveWindow.Selection.JumpToStartOfDoc
Do While ActiveWindow.NextFieldByType( 8 , sVar )
   sContent = ActiveWindow.Selection.GetFieldPar2()
   ActiveWindow.Selection.SetCurField( 0 ,sVar, Val(sCon-
tent) +1 )
Loop
ActiveWindow.Selection.JumpToEndOfDoc
Do While ActiveWindow.PrevFieldByType( 8 , sVar )
   sContent = ActiveWindow.Selection.GetFieldPar2()
   ActiveWindow.Selection.SetCurField( 0 ,sVar, Val(sCon-
tent) +1 )
Loop
End Sub
```

NonbreakingSpace-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von geschützten Leerzeichen.

Klasse:
ViewSettings

Syntax:
NonbreakingSpace

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob geschützte Leerzeichen angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Notes-Eigenschaft

Bestimmt oder ermittelt, ob Notizen im Dokument markiert werden sollen. Dies entspricht der Eigenschaft "Notizen anzeigen" im Register INHALTE des Menüs EXTRAS - OPTIONEN - TEXTDOKUMENT.

KLASSE:
ViewSettings

SYNTAX:
Notes

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Markierungen für Notizen auf dem Bildschirm ein- (True) oder ausgeblendet (False) werden sollen.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption "
Const sMessage_2 = " ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

NumberFormat-Eigenschaft

Bestimmt oder ermittelt das Zahlenformat der aktuellen Tabellenzelle. Das Zahlenformat entspricht dem im Dialog Zahlenformat.

KLASSE:
BaseTextTable

SYNTAX:
NumberFormat

RÜCKGABE:
String

PARAMETER:
[= Format]: Zeichenfolge, die das Zahlenformat angibt.

BEISPIEL:

```
Sub ExampleTableNumberFormat
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With ActiveWindow.Selection
.Insert("112")
.NumberFormat="ZZ #_##0+00 " REM Ein exotisches Format
.JumpToNextCell
.Insert("Benutzerdefiniert")
.JumpToNextCell
End With
End Sub
```

NumberFormatCurrency-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte Währungsformat.

KLASSE:
BaseTextTable

SYNTAX:
NumberFormatCurrency()

Beispiel:

```
Sub ExampleTableFormatCurrency
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With ActiveWindow.Selection
.Insert("112")
.NumberFormatCurrency
.JumpToNextCell
.Insert("Währung")
.JumpToNextCell
End With
End Sub
```

NumberFormatDate-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte Datumsformat.

Klasse:
BaseTextTable

Syntax:
NumberFormatDate()

Beispiel:

```
Sub ExampleTableFormatDate
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With activeWindow.Selection
.Insert("112233")
.NumberFormatDate
.JumpToNextCell
.Insert("Datum")
.JumpToNextCell
End With
End Sub
```

NumberFormatDecimal-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf zwei Dezimalstellen.

Klasse:
BaseTextTable

Syntax:
NumberFormatDecimal()

Beispiel:

```
Sub ExampleTableFormatDecimal
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With activeWindow.Selection
.Insert("12345,6789")
.NumberFormatDecimal
.JumpToNextCell
.Insert("Dezimal")
.JumpToNextCell
End With
End Sub
```

NumberFormatPercent-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte Prozentformat.

Klasse:
BaseTextTable

Syntax:
NumberFormatPercent()

Beispiel:

```
Sub ExampleTableFormatPercent
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With activeWindow.Selection
.Insert("112")
.NumberFormatPercent
.JumpToNextCell
.Insert("Prozent")
.JumpToNextCell
End With
End Sub
```

NumberFormatScientific-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte wissenschaftliche Format zur Darstellung von Exponentialzahlen.

Klasse:
BaseTextTable

Syntax:
NumberFormatScientific()

Beispiel:

```
Sub ExampleTableFormatScientific
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With activeWindow.Selection
.Insert("112")
.NumberFormatScientific
.JumpToNextCell
.Insert("Wissenschaftlich")
.JumpToNextCell
End With
End Sub
```

NumberFormatStandard-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte Standard-Format.

Klasse:
BaseTextTable

Syntax:
NumberFormatStandard()

Beispiel:

```
Sub ExampleTableFormatStandard
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
Selection.Insert "1234"
Selection.NumberFormat = "#.#0,0"
Selection.NumberFormatStandard
End Sub
```

NumberFormatTime-Methode

Setzt das Zahlenformat der aktuellen Tabellenzelle auf das vordefinierte Zeitformat.

Klasse:
BaseTextTable

Syntax:
NumberFormatTime()

Beispiel:

```
Sub ExampleTableFormatTime
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.GotoCell("A2", FALSE)
With ActiveWindow.Selection
.Insert("11,21")
.NumberFormatTime
.JumpToNextCell
.Insert("Zeit")
.JumpToNextCell
End With
End Sub
```

NumberLevel-Eigenschaft

Ermittelt die Numerierungsebene der aktuellen Cursorposition.

Klasse:
BaseTextList

Syntax:
NumberLevel

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleNumberLevel
Const sMessage_1 = "Dies ist der erste Eintrag der Numerierung"
Const sMessage_2 = "Dies ist der zweite Eintrag der Numerie-
rung"
Const sMessage_3 = "Dies ist der dritte Eintrag der Numerie-
rung"
Const sMessage_4 = "Der Cursor befindet sich in der "
Const sMessage_5 = ". Numerierungsebene"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_3
Selection.JumpToStartOfDoc
Selection.GoDown( 1, FALSE )
Selection.DecrementLevel
Selection.GoDown( 1, FALSE )
```

```
Selection.DecrementLevel
print sMessage_4 & Selection.Numberlevel & sMessage_5
End Sub
```

NumberOfPages-Eigenschaft

Ermittelt die Anzahl aller Seiten des aktuellen Dokuments.

KLASSE:
BaseText

SYNTAX:
NumberOfPages

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitennummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

NumberOrNoNumber-Eigenschaft

Schaltet die Aufzählung/Numerierung aus, wenn sich der Cursor in einer Aufzählung/Numerierung befindet.

KLASSE:
BaseTextList

SYNTAX:
NumberOrNoNumber

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Numerierung/Aufzählung ausgeschaltet werden soll (False) oder nicht (True).

BEISPIEL:

```
Sub ExampleNumberOrNoNumber
Const sMessage_1 = "Dies ist eine Numerierung"
Const sMessage_2 = "die in dieser Zeile abgeschaltet ist "
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
Selection.Insert sMessage_1
Selection.DecrementLevel
Selection.InsertPara
Selection.NumberOrNoNumber = FALSE
Selection.Insert sMessage_2
End Sub
```

NumberType-Eigenschaft

Bestimmt oder ermittelt den Numerierungstyp an der aktuellen Cursorposition.

KLASSE:
BaseTextList

SYNTAX:
NumberType

RÜCKGABE:
Long

PARAMETER:
[= Type]: Long-Integer-Ausdruck, der den Numerierungstyp angibt. Folgende Werte sind möglich:

0 - Großbuchstaben

1 - Kleinbuchstaben

2 - große römische Zahlen

3 - kleine römische Zahlen

4 - arabische Zahlen

5 - keine Zeichen

6 - Bullets

7 - Numerierung wie in der Seitenvorlage

8 - Grafiken

BEISPIEL:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
```

```
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

Eigenschaften und Methoden mit "O"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "O" beginnen.

ObjectBackOne-Methode

Stellt das selektierte Objekt um eine Ebene weiter nach vorne, wenn mehrere Objekte übereinander liegen. Vorne bedeutet hierbei, daß das Objekt zurück zur Oberfläche geholt wird.

KLASSE:
BaseTextDrawBase, BaseTextFrame

SYNTAX:
ObjectBackOne()

BEISPIEL:

```
sub ExampleFrameForBack
Dim s1 as String , s2 as String , s3 as String
Documents.add("sWriter4")
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "1cm", "1cm" , "5cm" , "4cm" )
s1 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s1)
Selection.Insert "Dieser Rahmen ist der erste. "
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "2cm", "2cm" , "5cm" , "4cm" )
s2 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s2)
Selection.Insert " Dieser Rahmen ist der zweite. "
Wait 2000
Selection.GotoFrame(s1)
Selection.ObjectBackOne
Selection.Insert " Der erste Rahmen liegt nun vor dem zweiten.
"
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "3cm", "3cm" , "5cm" , "4cm" )
s3 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s3)
```

```
Selection.Insert "Das ist der dritte Rahmen. "
Wait 2000
Selection.GotoFrame(s3)
Selection.ObjectForwardOne
Selection.Insert "Der dritte Rahmen liegt hinter dem zweiten. "
Selection.GotoAnchor
end sub
```

ObjectForwardOne-Methode

Stellt das selektierte Objekt um eine Ebene weiter nach hinten, wenn mehrere Objekte übereinander liegen. Hinten bedeutet hierbei, daß das Objekt in Blickrichtung von der Oberfläche weg plaziert wird.

KLASSE:
BaseTextDrawBase, BaseTextFrame

SYNTAX:
ObjectForwardOne()

BEISPIEL:

```
sub ExampleFrameForBack
Dim s1 as String , s2 as String , s3 as String
Documents.add("sWriter4")
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "1cm", "1cm" , "5cm" , "4cm" )
s1 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s1)
Selection.Insert "Dieser Rahmen ist der erste. "
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "2cm", "2cm" , "5cm" , "4cm" )
s2 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s2)
Selection.Insert " Dieser Rahmen ist der zweite. "
Wait 2000
Selection.GotoFrame(s1)
Selection.ObjectBackOne
Selection.Insert " Der erste Rahmen liegt nun vor dem zweiten.
"
Selection.JumpToStartOfDoc
Selection.InsertFrame( 2, "3cm", "3cm" , "5cm" , "4cm" )
s3 = Selection.FrameName
Selection.GotoAnchor
Selection.GoIntoFrame(s3)
Selection.Insert "Das ist der dritte Rahmen. "
Wait 2000
Selection.GotoFrame(s3)
Selection.ObjectForwardOne
```

```
Selection.Insert "Der dritte Rahmen liegt hinter dem zweiten. "
Selection.GotoAnchor
end sub
```

ObjectMirrorHorizontal-Methode

Bewirkt eine horizontale Spiegelung der ausgewählten Zeichenobjekte.

KLASSE:
TextDraw

SYNTAX:
ObjectMirrorHorizontal()

BEISPIEL:

```
REM Es muß bereits ein Zeichenobject eingefügt worden sein.
Selection.ObjectMirrorHorizontal
```

ObjectMirrorVertical-Methode

Bewirkt eine vertikale Spiegelung der ausgewählten Zeichenobjekte.

KLASSE:
TextDraw

SYNTAX:
ObjectMirrorVertical()

BEISPIEL:

```
REM Es muß bereits ein Zeichenobject eingefügt worden sein.
Selection.ObjectMirrorVertical
```

ObjectName-Eigenschaft

Ermittelt die Name-Eigenschaft des selektierten Kontrollfeldes, die zur Entwurfszeit festgelegt wurde, und gibt sie als Zeichenfolge zurück.

KLASSE:
BaseTextDrawControl

SYNTAX:
ObjectName

RÜCKGABE:
String

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

OleObject-Eigenschaft

Ist ein OLE-Objekt selektiert, wird mit OleObject dieses herausgereicht. Daran sind dann die StarBasic-Funktionen dieses Objekts verfügbar.

Handelt es sich z.B. um ein StarChart-Objekt, können daran Eigenschaften und Methoden zur Diagrammveränderung gerufen werden.

Klasse:
BaseTextOLEObject

Syntax:
OleObject

Rückgabe:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleOLEObjects
Const sMessage = "Der Name des Charts lautet : "
Dim oActwin As Object
Documents.Add( "sWriter4" )
Set oActwin = ActiveWindow REM Merkt sich die View für Deacti-
vate
If ActiveWindow.OLEObjects.Count = 0 Then
ActiveWindow.InsertObjectStarChart()
Else
ActiveWindow.OLEObjects(1).Activate
End If
oActwin.DeActivate REM Hier muß die View des Dokumentes
verwendet werden
Print sMessage & ActiveWindow.OLEObjects(1).OLEObject.Name
End Sub
```

OLEObjects-Eigenschaft

Ermöglicht den Zugriff auf die Collection der OLE-Objekte.

siehe auch: Tables

KLASSE:
BaseTextEditView

SYNTAX:
OLEObjects

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleOLEObjects
Const sMessage = "Der Name des Charts lautet : "
Dim oActwin As Object
Documents.Add( "sWriter4" )
Set oActwin = ActiveWindow REM Merkt sich die View für Deacti-
vate
If ActiveWindow.OLEObjects.Count = 0 Then
ActiveWindow.InsertObjectStarChart()
Else
ActiveWindow.OLEObjects(1).Activate
End If
```

```
oActwin.DeActivate REM Hier muß die View des Dokumentes
verwendet werden
Print sMessage & ActiveWindow.OLEObjects(1).OLEObject.Name
End Sub
```

OnlineAutoFormat-Eigenschaft

Bestimmt oder ermittelt, ob die automatische Korrekturfunktion aktiv ist.

KLASSE:
BaseText

SYNTAX:
OnlineAutoFormat

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die AutoKorrektur ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleOnlineAutoFormat
Documents.Add "sWriter4"
Selection.Insert( "Das automatische Korrigieren während der
Eingabe ist " )
Selection.Insert( IIF(ActiveWindow.Selection.OnlineAuto-
Format,"ausgeschaltet.","angeschaltet.")
End Sub
```

OnlineOptions-Methode

Ruft den Dialog OPTIONEN HTML-DOKUMENT auf. Dieser Dialog ist modal, d. h. das Makro wird solange unterbrochen, bis der Dialog von Anwender geschlossen wird.

KLASSE:
StarWriter

SYNTAX:
OnlineOptions()

BEISPIEL:

```
Sub ExampleOnlineOptions
StarWriter.OnlineOptions
End Sub
```

Opaque-Eigenschaft

Gibt an, ob das Objekt opaque (undurchsichtig) ist.

KLASSE:
BaseTextFrame

SYNTAX:
Opaque

RÜCKGABE:
Boolean

PARAMETER:
Rückgabe:

True: Objekt ist undurchsichtig.

False: Objekt ist durchsichtig.

BEISPIEL:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
Selection.Insert(sText)
if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALSE
Selection.Printable = TRUE
Selection.SendToBack
Selection.BringToFront
end sub
```

OpenDatabaseConnection-Methode

Öffnet eine Datenbank, die im StarOffice bereits registriert ist. Dabei wird lediglich eine Verbindung zur Datenbank hergestellt, sie wird nicht sichtbar geöffnet.

Bei Textdokumenten kann eine Datenbank sowohl am ActiveWindow als auch an der Application geöffnet werden. Öffnet man eine Datenbank an der Application, dann müssen alle nachfolgenden Datenbank-Methoden ebenfalls an der Application angewendet werden.

Siehe auch: AddDatabaseConnection, GetActiveDatabaseConnection, OpenTableConnection, OpenQueryConnection.

KLASSE:
BaseTextEditView

SYNTAX:
OpenDatabaseConnection(DBName As String)

RÜCKGABE:
Boolean

PARAMETER:
DBName: Ein Zeichenfolgeausdruck, der die Bezeichnung der Datenbank angibt, die geöffnet werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
DIM iDataNumber as Integer, iTableCount as Integer,
iTableNumber as Integer
DIM sDataName as String, sTableName as String, sFieldName as
String
DIM sOldTableName as String
DIM vStatus as Variant
DIM lRecordCount as Long
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)   REM
Datenbank öffnen
Print "Es ist folgende Datenbank geöffnet : " &
Activewindow.GetActiveDatabaseConnection
Print "QualifierSeparator liefert : " & Activewindow.Qualifier-
Separator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
```

```
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print "IsTableReadonly liefert : " &  Activewindow.IsTableRea-
dOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)  REM Die
Tabelle öffnen
Print "Es ist folgende Tabelle geöffnet : " &
Activewindow.GetActiveTableConnection
Print "GetDatabaseLimits liefert : " &  Activewindow.GetDataba-
seLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print "Die Abfrage hat " & lRecordCount & " Datensätze ermit-
telt"
If lRecordCount > 0 Then
Print "DataRecordID liefert : " & Activewindow.DataRecordID
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print "Der Inhalte des ersten Feldes : " & Activewindow.DataCo-
lumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord  REM Springt zum nächsten
Datensatz
vStatus = Activewindow.DataRecord(2)   REM Geht zum zweiten
Datensatz
vStatus = Activewindow.DataPreviousRecord  REM Springt zum
vorherigen Datensatz
vStatus = Activewindow.DataLastRecord  REM Springt zum letzten
Datensatz
End if
Print "DataPhysicalRecordID liefert : " & Activewindow.DataPhy-
sicalRecordID
Print "DataPhysicalRecordID liefert : " & Activewindow.DataRe-
cordID
End if
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End if
End if
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
```

```
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

OpenForm-Methode

Öffnet ein an einer Datenbank angemeldetes Formular in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenReport, OpenQuery.

KLASSE:
BaseTextEditView

SYNTAX:
OpenForm(FormName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
FormName: Name des Formulares, das geöffnet werden soll.

ReadOnly: Boolescher Ausdruck, der angibt, ob das Formular im normalen Dateneingabemodus (True) oder im Entwurfsmodus (False) geöffnet werden soll.

BEISPIEL:

```
Sub Main
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sWindowName As String, sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
```

```
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
sWindowName = ActiveWindow.Name
ActiveWindow.CreateForm( sFormName, sPathName )
ActiveWindow.CreateReport( sReportName, sPathName )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Windows(sWindowName).OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Windows(sWindowName).OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Windows(sWindowName).OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).DeleteForm( sFormName )
Windows(sWindowName).DeleteReport( sReportName )
Windows(sWindowName).DeleteQuery( sQueryName )
End Sub
```

OpenQuery-Methode

Öffnet eine Datenbank-Abfrage in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenReport, OpenForm.

KLASSE:
BaseTextEditView

SYNTAX:
OpenQuery(QueryName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage, die geöffnet werden soll.

Readonly: Boolescher Ausdruck, der angibt, ob die Daten aus der Abfrage geändert werden können (False) oder nicht (True).

Beispiel:

```
Sub Main
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sWindowName As String, sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
sWindowName = ActiveWindow.Name
ActiveWindow.CreateForm( sFormName, sPathName )
ActiveWindow.CreateReport( sReportName, sPathName )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Windows(sWindowName).OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Windows(sWindowName).OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Windows(sWindowName).OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).DeleteForm( sFormName )
Windows(sWindowName).DeleteReport( sReportName )
Windows(sWindowName).DeleteQuery( sQueryName )
End Sub
```

OpenQueryConnection-Methode

Stellt eine Verbindung zu einer Datenbank-Abfrage her, wenn die zugehörige Datenbank zuvor mit der OpenDatabaseConnection-Methode geöffnet wurde. OpenQueryConnection stellt lediglich eine Verbindung zur Abfrage her, sie wird nicht sichtbar geöffnet.

Siehe auch: OpenDatabaseConnection, OpenQuery, OpenTableConnection.

KLASSE:
BaseTextEditView

SYNTAX:
OpenQueryConnection(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Name der Datenbank-Abfrage.

BEISPIEL:

```
Sub ExampleQueries
Const sQueryName = "TestAbfrage"
Const sMessage = "Der Inhalt des ersten Feldes : "
Dim sDataName As String, sTabName As String, sTPath As String
Dim vStatus As Variant
Documents.Add( "sWriter4" )
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTabName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateQuery sQueryName, "SELECT * FROM " & sTab-
Name, FALSE
vStatus = ActiveWindow.OpenQueryConnection(sQueryName)
ActiveWindow.DataFirstRecord
Print sMessage & ActiveWindow.DataColumn(1)
ActiveWindow.CloseQueryConnection(sQueryName)
ActiveWindow.DeleteQuery sQueryName
End Sub
```

OpenReport-Methode

Öffnet einen Datenbank-Bericht in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenTable, OpenForm, OpenQuery.

KLASSE:
BaseTextEditView

SYNTAX:
OpenReport(ReportName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
ReportName: Zeichenfolge, die den Namen des Berichts angibt.

ReadOnly: Gibt an, ob die Daten des Berichts schreibgeschützt angezeigt werden (True) oder ob der Bericht im Entwurfsmodus geöffnet werden soll (False).

BEISPIEL:

```
Sub Main
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sWindowName As String, sTmpWindowName As String
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
sWindowName = ActiveWindow.Name
ActiveWindow.CreateForm( sFormName, sPathName )
ActiveWindow.CreateReport( sReportName, sPathName )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Windows(sWindowName).OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).OpenForm( sFormName, FALSE )
Wait 1000
```

```
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Windows(sWindowName).OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Windows(sWindowName).OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).DeleteForm( sFormName )
Windows(sWindowName).DeleteReport( sReportName )
Windows(sWindowName).DeleteQuery( sQueryName )
End Sub
```

OpenTable-Methode

Öffnet eine Datenbank-Tabelle in der Dokumentansicht. Voraussetzung ist, daß vor Aufruf dieser Methode mittels OpenDatabaseConnection eine Verbindung zur zugehörigen Datenbank hergestellt wurde.

Siehe auch: OpenDatabaseConnection, OpenReport, OpenForm, OpenQuery.

KLASSE:
BaseTextEditView

SYNTAX:
OpenTable(TableName As String, ReadOnly As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Zeichenfolgeausdruck, der den Namen der Datenbank-Tabelle angibt, die geöffnet werden soll.

ReadOnly: Gibt an, ob die Daten schreibgeschützt sind (True) oder ob Daten hinzugefügt und geändert werden können (False).

BEISPIEL:

```
Sub Main
Const sFileName = "\Test.sdw"
Const sFormName = "Formular"
Const sReportName = "Bericht"
Const sQueryName = "Abfrage"
Const Message = "Dieses Dokument wird ein Formular bzw. Bericht
werden."
Dim sDataName As String, sTableName As String, sPathName As
String
Dim sWindowName As String, sTmpWindowName As String
```

```
Dim vStatus As Variant
REM Das Databaseverzeichnis ist für die Erstellung des Berichts
nötig
sPathName = Application.Pathsettings.Database & sFileName
REM Erstellt eine Datei die als Formular verwendet wird.
Documents.Add("sWriter4")
Selection.Insert Message
ActiveDocument.Close TRUE, sPathName
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
sWindowName = ActiveWindow.Name
ActiveWindow.CreateForm( sFormName, sPathName )
ActiveWindow.CreateReport( sReportName, sPathName )
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Windows(sWindowName).OpenTable( sTableName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).OpenForm( sFormName, FALSE )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Topwindow.Document.Close( FALSE, "" )
Windows(sWindowName).OpenReport( sReportName, 1 )
Wait 1000
sTmpWindowName = ActiveDocument.Name
Documents(sTmpWindowName).Close( FALSE, "" )
Windows(sWindowName).OpenQuery( sQueryName )
Wait 1000
sTmpWindowName = Activewindow.Name
Windows(sTmpWindowName).Close
Windows(sWindowName).DeleteForm( sFormName )
Windows(sWindowName).DeleteReport( sReportName )
Windows(sWindowName).DeleteQuery( sQueryName )
End Sub
```

OpenTableConnection-Methode

Stellt eine Verbindung zu einer Datenbank-Tabelle her, wenn die zugehörige Datenbank zuvor mit der OpenDatabaseConnection-Methode geöffnet wurde. OpenTableConnection stellt lediglich eine Verbindung zur Tabelle her, sie wird nicht sichtbar geöffnet.

Um den Zugriff auf die Daten der Tabelle zu ermöglichen, muß zunächst auf einen Datensatz positioniert werden. Es ist also erforderlich, nach dem Öffnen der Tabelle vor einem ersten Verwenden der Daten zu einem gewünschten Datensatz zu springen, z.B. mittels DataRecord.

Siehe auch: GetActiveTableConnection, OpenDatabaseConnection, OpenQueryConnection, OpenTable.

KLASSE:
BaseTextEditView

SYNTAX:
OpenTableConnection(TableName As String)

RÜCKGABE:
Boolean

PARAMETER:
TableName: Zeichenfolgenausdruck, der den Namen der Tabelle angibt, die geöffnet werden soll.

BEISPIEL:

```
Sub ExampleDataRecord
Const sMessage_1 = "Es ist folgende Datenbank geöffnet : "
Const sMessage_2 = "QualifierSeparator liefert : "
Const sMessage_3 = "IsTableReadonly liefert : "
Const sMessage_4 = "Es ist folgende Tabelle geöffnet : "
Const sMessage_5 = "GetDatabaseLimits liefert : "
Const sMessage_6a = "Die Abfrage hat "
Const sMessage_6b = " Datensätze ermittelt"
Const sMessage_7 = "DataRecordID liefert : "
Const sMessage_8 = "Der Inhalt des ersten Feldes : "
Const sMessage_9 = "DataPhysicalRecordID liefert : "
Const sMessage_10 = "DataRecordID liefert : "
Dim iDataNumber As Integer, iTableCount As Integer,
iTableNumber As Integer
Dim sDataName As String, sTableName As String, sFieldName As
String
Dim sOldTableName As String
Dim vStatus As Variant
Dim lRecordCount As Long
```

```
REM Alle nachfolgenden Datenbank Methoden/Eigenschaften funk-
tionieren nur
REM an einem geöffneten Textdokument.
Documents.Add "sWriter4"
REM Alle angemeldete Datenbanken nacheinander verwenden
For iDataNumber=1 To Activewindow.DataDefinedDatabaseNumber()
sDataName = Activewindow.DataDefinedDatabase(iDataNumber)  REM
Datenbankname ermitteln
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
Print sMessage_1 & Activewindow.GetActiveDatabaseConnection
Print sMessage_2 & Activewindow.QualifierSeparator
REM Anzahl der Tabellen in der Datenbank
iTableCount = Activewindow.DataTableNumber(sDataName)
REM Es werden alle Tabellen innerhalb der Datenbank nachein-
ander verwendet
For iTableNumber=1 To iTableCount
REM Tabellenname ermitteln
sTableName = Activewindow.DataGetTableName(sDataName,
iTableNumber)
Print sMessage_3 &  Activewindow.IsTableReadOnly(sTableName)
vStatus = Activewindow.OpenTableConnection(sTableName)
Print sMessage_4 & Activewindow.GetActiveTableConnection
Print sMessage_5 &  Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & sTableName & " ;")
REM  Wichtig, es muß als erstes zu einen Datensatz positioniert
werden
vStatus = Activewindow.DataFirstRecord
lRecordCount = Activewindow.GetDataRecordCount
Print sMessage_6a & lRecordCount & sMessage_6b
If lRecordCount > 0 Then
Print sMessage_7 & Activewindow.DataRecordID
Print sMessage_8 & Activewindow.DataColumn(1)
sFieldName = Activewindow.DataColumnName(1)  REM Wie lautet der
Name des ersten Feldes
Print sMessage_8 & Activewindow.DataColumn(sFieldName)
If lRecordCount > 0 Then
vStatus = Activewindow.DataNextRecord
vStatus = Activewindow.DataRecord(2)
vStatus = Activewindow.DataPreviousRecord
vStatus = Activewindow.DataLastRecord
End If
Print sMessage_9 & Activewindow.DataPhysicalRecordID
Print sMessage_10 & Activewindow.DataRecordID
End If
REM DataSwitch kann man nur bei mehr als einer Tabelle sinnvoll
verwenden
If iTableCount > 1 Then
If Len(sOldTableName) > 0 Then
REM Schaltet zur vorherigen Tabelle zurück
vStatus = Activewindow.DataSwitch(sDataName, sOldTableName)
Else
sOldTableName = sTableName
End If
End If
```

```
Activewindow.CloseTableConnection(sTableName)
Next iTableNumber
Activewindow.CloseDatabaseConnection(sDataName)
Next iDataNumber
End Sub
```

Orphan-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Zeilen eines auf die nächste Seite umbrochenen Absatzes, die am oberen Rand dieser Seite aufgeführt werden dürfen (Hurenkindregelung). Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetOrphan gesetzte Hurenkindregel abgefragt.

Eine Abfrage der Orphan-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsOrphan bestimmt wurde, daß nach einer bestimmten Hurenkindregel gesucht werden soll, d. h. IsOrphan = 2.

Siehe auch: Widow, SetOrphan, IsOrphan.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
Orphan

RÜCKGABE:
Integer

PARAMETER:
Bei Absätzen:

[= Rows]: Integer-Ausdruck, der die Anzahl der Zeilen eines auf die nächste Seite umgebrochenen Absatzes angibt, die am oberen Rand dieser Seite aufgeführt werden dürfen.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleOrphanAndWidow
REM den lateinischen Blindtext verwenden
Const sBlindText = "BT"
Dim iCount As Integer
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Dies ist ein Absatz, der mit der Schusterjungenregel
auf 4 und Hurenkinderregel auf 4 formatiert wurde.  ")
```

```
.Insert("Der Absatz wird langsam über die Seitengrenze
geschoben. Verfolgen Sie, was passiert, indem Sie die vertikale
")
.Insert("Bildlaufleiste nachstellen.  ")
.Insert(" *** *** *** *** *** *** *** *** *** *** *** *** ***
*** ***   ")
.ExpandGlossary( sBlindText )
End With
With ActiveWindow.Selection.Paragraph
.Widow = 4
.Orphan = 4
End With
ActiveWindow.JumpToStartOfDoc
For iCount = 1 to 56
Wait 20
Selection.InsertPara
Next
End Sub
```

OutlineFont-Eigenschaft

Bestimmt oder ermittelt die Option für die Darstellung des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen als Konturschrift.

KLASSE:
TextDrawText, TextFont

SYNTAX:
OutlineFont

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen als Konturschrift dargestellt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
```

```
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
```

```
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Outside-Eigenschaft

Schaltet die Option Kontur beim Umlauf von Textrahmen ein.

Klasse:
BaseTextFrame

Syntax:
Outside

Rückgabe:
Boolean

Parameter:
True: Option Kontur aktivieren

False: Option Kontur ausschalten

Beispiel:

```
Sub ExampleOutside
Documents.Add( "sWriter4" )
Selection.Insert "Dieser Text läuft nicht in den Textrahmen
rein"
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "4cm", "2cm", "4cm", "4cm")
Selection.Contour = TRUE
Selection.Outside = TRUE
End Sub
```

Eigenschaften und Methoden mit "P"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "P" beginnen.

PageColumnDialog-Methode

Ruft das Register Spalten auf, in dem die Einstellungen für die Spalten einer Seite gesetzt werden können.

Klasse:
BaseTextSelection

Syntax:
PageColumnDialog()

Beispiel:

```
Sub ExamplePageColumnDialog
Documents.Add("swriter4")
ActiveWindow.PageColumnDialog
end sub
```

PageDialog-Methode

Ruft den Dialog Seitenvorlage auf, in dem die Einstellungen für die aktuelle Seite gesetzt werden können.

Klasse:
BaseTextSelection

Syntax:
PageDialog()

Beispiel:

```
Sub ExamplePageStyle
Const sTemp="Meine neue Seite"
Documents.Add "sWriter4"
```

```
REM Erst mal eine neue Seitenvorlage anlegen, da Seitenattri-
bute immer direkt auf die Vorlage wirken.
ActiveDocument.NewStyle( sTemp, 8)
REM ... und desegen auch gleich anwende, damit bereits defi-
nierte Vorlagen nicht überschrieben werden
Selection.StyleApply( sTemp, 8)
Selection.Insert("Hier können Sie Seitenformatierungen
ändern.")
Selection.InsertPara
Selection.PageDialog REM Format-Seite aufrufen
Selection.InsertPageBreak
Selection.Insert("Die Formatierung wurden in die Seitenvorlage
REM " &_
   Activewindow.PageStyleName &_
   "REM  übertragen."
End Sub
```

PageDown-Methode

Bewegt die Schreibmarke um eine Seite nach unten.

KLASSE:
TextPrintPreview, BaseTextEditView

SYNTAX:
PageDown(Select As Boolean)

PARAMETER:
[Select]: Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExamplePageUp
Const sMessage_1 = "Seite "
Const sMessage_2 = "Dies ist die dritte Seite"
Const sMessage_3 = "Dies ist die erste Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
For iCount=1 To 2
ActiveWindow.Selection.Insert( sMessage_1 & iCount )
ActiveWindow.Selection.InsertPageBreak
Next iCount
Selection.JumpToStartOfDoc
ActiveWindow.PageDown FALSE
ActiveWindow.PageDown FALSE
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.InsertPara
```

```
ActiveWindow.PageUp TRUE
ActiveWindow.PageUp TRUE
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.InsertPara
End Sub
```

PageEnd-Eigenschaft

Bestimmt oder ermittelt, ob die Silbentrennung eines Wortes über zwei Seiten erfolgen darf. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetHyphenZone gesetzte Option zur Silbentrennung abgefragt.

Siehe auch: SetHyphenZone, IsHyphen, Hyphen, MinLead, MinTrail, MaxHyphens.

KLASSE:
HyphenZone, ReplaceAttributes, SearchAttributes

SYNTAX:
PageEnd

RÜCKGABE:
Boolean

PARAMETER:
Bei HyphenZone-Objekten:

[= Switch]: Boolescher Ausdruck, der angibt, ob die Silbentrennung eines Wortes über zwei Seiten erfolgen darf (True) oder nicht (False).

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
sub ExampleHyphenZone
Dim iCount as Integer
Documents.Add("sWriter4")
Selection.Paragraph.HyphenZone.Hyphen = True
Selection.Paragraph.HyphenZone.PageEnd = True
Selection.Paragraph.HyphenZone.MinLead = 3
Selection.Paragraph.HyphenZone.MinTrail = 3
Selection.Paragraph.HyphenZone.MaxHyphens = 3
Selection.Insert "Das letzte Wort wird umgebrochen"
Selection.GotoStartOfLine FALSE
For iCount=1 to 53
Selection.Insert "  "
Next iCount
end sub
```

PageHeight-Eigenschaft

Setzt oder ermittelt die Seitenhöhe an der aktuellen Seitenvorlage.

Siehe auch: PageWidth

KLASSE:
BaseTextEditView

SYNTAX:
PageHeight

RÜCKGABE:
Long

PARAMETER:
[= Height]: Long-Integer-Ausdruck, der die Seitenhöhe angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExamplePageUp
Const sMessage_1 = "Seite "
Const sMessage_2 = "Dies ist die dritte Seite"
Const sMessage_3 = "Dies ist die erste Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
For iCount=1 To 2
ActiveWindow.Selection.Insert( sMessage_1 & iCount )
ActiveWindow.Selection.InsertPageBreak
Next iCount
Selection.JumpToStartOfDoc
ActiveWindow.PageDown FALSE
ActiveWindow.PageDown FALSE
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.InsertPara
ActiveWindow.PageUp TRUE
ActiveWindow.PageUp TRUE
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.InsertPara
End Sub
```

PageNumber-Eigenschaft

Ermittelt die aktuelle physikalische Seitennummer, unabhängig von geänderten Seitennummern z. B. durch manuelle Seitenumbrüche. Die physikalische Seitennummer wird automatisch vergeben und ist nicht manipulierbar. Haben Sie beispielsweise für die aktuelle Seite die logische Seitennummer 3 vergeben und ist diese Seite die erste Seite in Ihrem Dokument, so liefert PageNumber den Wert 1.

Siehe auch: LogicalPageNumber

KLASSE:
BaseText

SYNTAX:
PageNumber

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

PageOffset-Eigenschaft

Liefert den Offset zurück, der durch das Setzen einer logischen Seitennummer entstanden ist. Haben Sie beispielsweise ein Dokument mit 10 Seiten, und Sie definieren Seite 3 (physikalisch) als Seite 5 (logisch) und Seite 7 (physikalisch) als Seite 10 (logisch), so liefern die Seiten 1 bis 2 den Offset 0, Seite 3 bis 6 den Offset 5 und Seite 7 bis 10 den Offset 10. Der Offset stellt damit den logischen Nullpunkt für selbstdefinierte Seitennummern dar. Er wird durch die PageOffsetDialog-Methode festgelegt.

Siehe auch: LogicalPageNumber, PageNumber, PageOffsetDialog.

KLASSE:
TextEditView

SYNTAX:
PageOffset

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
```

```
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

PageOffsetDialog-Methode

Setzt die logische Seitennummer auf der aktuellen Seite. Ohne Übergabeparameter ruft diese Methode den Dialog SEITENNUMMER SETZEN auf.

Der übergebene Offset läßt sich durch die PageOffset-Eigenschaft ermitteln.

Siehe auch: PageOffset, LogicalPageNumber.

KLASSE:
TextEditView

SYNTAX:
PageOffsetDialog(nOffset As Long)

PARAMETER:
[nOffset]: Long-Integer-Ausdruck, der den Wert der einzustellenden logischen Seitennummer (den Offset) angibt.

BEISPIEL:

```
Sub ExamplePageNumber
Documents.Add "sWriter4"
Selection.Insert("Das ist Seite 1")
Selection.InsertPageBreak
Activewindow.PageOffsetDialog(20)
Selection.Insert("Jetzt wurde die logische Seitenummer geän-
dert.")
Selection.InsertPara
ActiveWindow.Selection.ResetAttributes
Selection.Insert("Seitenoffset=" & Activewindow.PageOffset )
Selection.InsertPara
Selection.Insert("Pysikalische Seitennummer=" &
Activewindow.PageNumber )
Selection.InsertPara
Selection.Insert("Logische Seitennummer=" & Activewindow.Logi-
calPageNumber )
Selection.InsertPara
Selection.Insert("Gesamte Seitenanzahl=" & Activewindow.Number-
OfPages )
Selection.InsertPara
Selection.Insert("Die Seitennumerierung über Felder lautet:
Seite ")
Selection.InsertPageCountField
Selection.Insert(" von ")
```

```
Selection.InsertPageNumberField
Selection.InsertPara
End Sub
```

PagePaperBin-Eigenschaft

Bestimmt oder ermittelt den Druckerschacht an der aktuellen Seitenvorlage. Die verschiedenen Auswahlmöglichkeiten hängen vom verwendeten Druckertreiber ab.

KLASSE:
BaseTextEditView

SYNTAX:
PagePaperBin

RÜCKGABE:
Integer

PARAMETER:
[= Value]: Numerischer Ausdruck vom Typ Integer, der den Index des Druckerschachts angibt. Numeriert sind die Einträge des Listenfelds SCHACHT im Dialog SEITENVORLAGE, wobei der erste Eintrag mit dem Index 0 belegt ist.

BEISPIEL:

```
Sub ExamplePagePaperBin
Const sMessage = "Der Druckerschacht bei der aktuellen Seiten-
vorlage ist : "
Documents.Add("sWriter4")
Print sMessage & Activewindow.PagePaperBin
End Sub
```

PageStyle-Eigenschaft

Bestimmt oder ermittelt den Namen der aktuellen Seitenvorlage.

Hängt an einem Absatz ein Seitenumbruch (Menü FORMAT - ABSATZ... - TEXTFLUß - ZUSÄTZE - UMBRUCH - Option SEITE) und wurde dieser Umbruch mit einer Seitenvorlage definiert (das ist dann der Fall, wenn im Dialog das Markierfeld MIT SEITENVORLAGE aktiviert ist), dann wird die am Absatz definierte Seitenvorlage gesetzt oder ermittelt.

Siehe auch: Auto, Break.

KLASSE:
Paragraph, BaseTextSelection

SYNTAX:
PageStyle

Rückgabe:
String

Parameter:
[= Style]: Zeichenfolge, die die Seitenvorlage aufnimmt. Die möglichen Einstellungen entsprechen den Eintragungen im Listenfeld des Textfluß-Registers. Wird ein Leerstring übergeben, wird eine bereits gesetzte Seitenvorlage am Absatz gelöscht.

Beispiel:

```
sub ExampleParagraphBreak
Const sM1 = "Vor diesem Absatz wird ein Seitenumbruch mit
Vorlage: "
Const sTemp="Rechte Seite"  REM Vorlage Linke Seite wird beim
Seitenumbruch verwendet.
Const sM2 = " eingefügt"
Const sTempWithBreak="Erster Absatz einer Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
With Activewindow.Selection
   .Insert("Erster Absatz hat Standardformat.")
   .InsertPara
   .Insert( sM1 & sTemp & sM2 )
   .Paragraph.Break = 4 REM Seitenumbruch vor dem Absatz
   .Paragraph.Auto = TRUE
   .Paragraph.PageStyle=sTemp REM ... mit dieser Seitenvorlage
   .StyleNewByExample( sTempWithBreak, 2 ) REM die Absatzattri-
bute in eine Vorlage räumen
   .InsertPara
   .Paragraph.PageStyle=""  REM Mit "" wird die gesamte
Umbruch-Attributierung hart gelöscht !
   .Insert("Der letzte Absatz in diesem Dokument hat wieder
Standardformat.")
   .InsertPara
End With
```

PageStyleApply-Methode

Weist der aktuellen Seite die angegebene Seitenvorlage zu. Ist keine Seitenvorlage angegeben, so öffnet diese Methode den Dialog Seitenvorlage.

Klasse:
BaseTextEditView

Syntax:
PageStyleApply(Style As String)

Parameter:
[Style]: Zeichenfolgenausdruck, der die Bezeichnung der Seitenvorlage aufnimmt, die der aktuellen Seite zugewiesen werden soll.

Beispiel:

```
Sub ExamplePageStyle
Const sPageStyle = "Neue Vorlage"
Const sMessage = "Folgende Seitenvorlagen sind verwendet : "
Dim iCount As Integer
Documents.Add("sWriter4")
Selection.StyleNewByExample( sPageStyle, 8)
ActiveWindow.PageStyleApply( sPageStyle )
For iCount=0 to Activewindow.GetPageStyleCount-1
print sMessage & Activewindow.GetPageStyleNameAt(iCount)
Next iCount
End Sub
```

PageStyleName-Eigenschaft

Ermittelt die Bezeichnung der Seitenvorlage der aktuellen Seite.

Klasse:
BaseTextEditView

Syntax:
PageStyleName

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExamplePageStyle
Const sTemp="Meine neue Seite"
Documents.Add "sWriter4"
REM Erst mal eine neue Seitenvorlage anlegen, da Seitenattri-
bute immer direkt auf die Vorlage wirken.
ActiveDocument.NewStyle( sTemp, 8)
REM ... und desegen auch gleich anwende, damit bereits defi-
nierte Vorlagen nicht überschrieben werden
Selection.StyleApply( sTemp, 8)
Selection.Insert("Hier können Sie Seitenformatierungen
ändern.")
Selection.InsertPara
Selection.PageDialog REM Format-Seite aufrufen
Selection.InsertPageBreak
Selection.Insert("Die Formatierung wurden in die Seitenvorlage
REM " &_
   Activewindow.PageStyleName &_
   "REM  übertragen."
End Sub
```

PageUp-Methode

Bewegt die Schreibmarke um eine Seite nach oben.

KLASSE:
TextPrintPreview, BaseTextEditView

SYNTAX:
PageUp(Select As Boolean)

PARAMETER:
[Select]: Boolescher Ausdruck, der angibt, ob die Textstelle, über die sich die Schreibmarke bewegt, markiert werden soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht angegeben, erfolgt keine Selektion.

BEISPIEL:

```
Sub ExamplePageUp
Const sMessage_1 = "Seite "
Const sMessage_2 = "Dies ist die dritte Seite"
Const sMessage_3 = "Dies ist die erste Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
For iCount=1 To 2
ActiveWindow.Selection.Insert( sMessage_1 & iCount )
ActiveWindow.Selection.InsertPageBreak
Next iCount
Selection.JumpToStartOfDoc
ActiveWindow.PageDown FALSE
ActiveWindow.PageDown FALSE
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.InsertPara
ActiveWindow.PageUp TRUE
ActiveWindow.PageUp TRUE
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.InsertPara
End Sub
```

PageWidth-Eigenschaft

Setzt oder ermittelt die Seitenbreite an der aktuellen Seitenvorlage.

Siehe auch: PageHeight

KLASSE:
BaseTextEditView

SYNTAX:
PageWidth

RÜCKGABE:
Long

PARAMETER:
[= Width]: Long-Integer-Ausdruck, der die Breite der Seite bestimmt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExamplePageUp
Const sMessage_1 = "Seite "
Const sMessage_2 = "Dies ist die dritte Seite"
Const sMessage_3 = "Dies ist die erste Seite"
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
For iCount=1 To 2
ActiveWindow.Selection.Insert( sMessage_1 & iCount )
ActiveWindow.Selection.InsertPageBreak
Next iCount
Selection.JumpToStartOfDoc
ActiveWindow.PageDown FALSE
ActiveWindow.PageDown FALSE
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.InsertPara
ActiveWindow.PageUp TRUE
ActiveWindow.PageUp TRUE
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.InsertPara
End Sub
```

PairKerning-Eigenschaft

Bestimmt oder ermittelt, ob paarweises Kerning aktiv ist.

KLASSE:
TextFont

SYNTAX:
PairKerning

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob paarweises Kerning ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
```

```
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

PaperFromSetup-Eigenschaft

Bestimmt oder ermittelt die Druckereinstellung für den Papierschacht.

KLASSE:
PrintSettings

SYNTAX:
PaperFromSetup

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der im Druckertreiber vermerkte Papierschacht verwendet werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
```

```
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

Paragraph-Eigenschaft

Am Objekt Paragraph stehen Eigenschaften und Methoden für Absatzeinstellungen zur Verfügung.

KLASSE:
BaseText, BaseTextEditView

SYNTAX:
Paragraph

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleJustifyAll
REM Formatiert das gesamte Dokument auf Blocksatz
Const sMotherTemp="Standard" REM Die Mutter aller Absatzvor-
lagen
Const sBlindText = "BT" REM den Blindtext verwenden
Documents.Add("sWriter4")
With ActiveWindow.Selection
.ExpandGlossary( sBlindText )
sStyle = .ParaStyle REM rette aktuelle Vorlage
.Paragraph.ParaStyleApply(sMotherTemp)
.Paragraph.Alignment = 2
.StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
Wait 1000
```

```
.Paragraph.Alignment = 0 REM aus Standardwert zurücksetzen
.StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
.Paragraph.ParaStyleApply( sStyle ) REM wieder auf ursprüng-
liche Vorlage setzen
end with
End Sub
```

ParagraphDialog-Methode

Ruft den Dialog Absatz auf. Der zuvor selektierte Text wird nach Auswahl der neuen Einstellungen formatiert.

Klasse:
TextDrawText, Paragraph

Syntax:
ParagraphDialog()

Beispiel:

```
Sub ExampleParagraphDialog
Documents.Add("sWriter4")
ActiveWindow.Selection.Paragraph.ParagraphDialog
End Sub
```

ParagraphEnd-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von Absatzendezeichen.

Klasse:
ViewSettings

Syntax:
ParagraphEnd

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Absatzendezeichen angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

ParaSplit-Eigenschaft

Bestimmt oder ermittelt, ob ein Seitenumbruch innerhalb eines Absatzes erfolgen darf. Beim Suchen und Ersetzen von Attributen ermittelt diese Eigenschaft die mit SetParaSplit gesetzte Suchoption für das Attribut ABSATZ TRENNEN, wenn IsParaSplit den Wert 2 liefert.

Absätze, die länger als die gesamte Seite sind, werden unabhängig von der ParaSplit-Eigenschaft stets umgebrochen.

Siehe auch: SetParaSplit, IsParaSplit.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
ParaSplit

RÜCKGABE:
Boolean

PARAMETER:
Bei Absätzen:

[= Switch]: Boolescher Ausdruck, der angibt, ob ein Seitenumbruch innerhalb eines Absatzes verhindert wird (True) oder ob er zulässig ist (False). Ist ParaSplit = True und kann ein am Ende einer Seite angeordneter Absatz nicht vollständig auf dieser Seite angeordnet werden, dann wird er vollständig auf die nächste Seite umgebrochen.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
sub ExampleParagraphSplit
Const sTempWithBreak="AbsatzOhneUmbruch"
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "8cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.TopMargin = "1cm"
ActiveWindow.BottomMargin = "1cm"
With Activewindow.Selection
   .StyleNewByExample( sTempWithBreak, 2 ) REM die Absatzattri-
bute in eine Vorlage räumen
   For iCount = 1 To 10
   .Insert("Das ist der erste Absatz. ")
   Next iCount
   .InsertPara
   For iCount = 1 To 10
   .Insert("Das wird ein weiterer Absatz. ")
   Next iCount
   .InsertPara
   ActiveWindow.JumpToStartOfDoc
   .Paragraph.ParaSplit=FALSE
   .StyleUpdateByExample( sTempWithBreak, 2 ) REM die Absatz-
vorlage aktualisieren
Wait 2000
.Paragraph.ParaSplit=TRUE
   .StyleUpdateByExample( sTempWithBreak, 2 ) REM die Absatz-
vorlage aktualisieren
End With
End Sub
```

ParaStyle-Eigenschaft

Liefert den Namen der an der Selektion gültigen Absatzvorlage als String zurück.

KLASSE:
BaseTextSelection

SYNTAX:
ParaStyle

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleParaStyle
Const sStyleName = "Neuer Style"
Const sMessage_1 = "Hier wird die Absatzvorlage neu gesetzt"
Const sMessage_2 = "Die neue Absatzvorlage heißt : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine True
REM Neue Absatzvorlage anlegen
Selection.StyleNewByExample( sStyleName, 2 )
Selection.Paragraph.ParaStyleApply( sStyleName )
Print sMessage_2 & Selection.ParaStyle
End Sub
```

ParaStyleApply-Methode

Weist dem aktuellen Absatz eine neue Absatzvorlage zu. Ist keine Absatzvorlage angegeben, so öffnet diese Methode den Dialog ABSATZ, in dem die Einstellungen für den aktuellen Absatz vorgenommen werden können.

KLASSE:
Paragraph

SYNTAX:
ParaStyleApply(Style As String)

PARAMETER:
[Style]: Zeichenfolge, die die Bezeichnung der Absatzvorlage angibt.

BEISPIEL:

```
Sub ExampleJustifyAll
REM Formatiert das gesamte Dokument auf Blocksatz
Const sMotherTemp="Standard" REM Die Mutter aller Absatzvor-
```

```
lagen
Const sBlindText = "BT" REM den Blindtext verwenden
Documents.Add("sWriter4")
With ActiveWindow.Selection
   .ExpandGlossary( sBlindText )
   sStyle = .ParaStyle      REM rette aktuelle Vorlage
   .Paragraph.ParaStyleApply(sMotherTemp)
   .Paragraph.Alignment = 2
   .StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
   Wait 1000
   .Paragraph.Alignment = 0     REM aus Standardwert zurück-
setzen
   .StyleUpdateByExample(sMotherTemp ,2) REM setze auf Block-
satz -> und alle Kindvorlagen.
   .Paragraph.ParaStyleApply( sStyle )   REM wieder auf
ursprüngliche Vorlage setzen
end with
End Sub
```

ParaStyleCount-Eigenschaft

Liefert die Anzahl der im Dokument verwendeten Absatzvorlagen.

Siehe auch: GetParaStyleName

KLASSE:
BaseText

SYNTAX:
ParaStyleCount

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleGetParaStyleName
Const sHeader = "Es sind folgende Vorlagen verwendet worden : "
Const sStyle = "Vorlage : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.StyleNewByExample( "NeueVorlage", 2 )
Selection.Insert sHeader
Selection.InsertPara
for iCount=0 to Activewindow.ParaStyleCount-1
Selection.Insert sStyle &
Activewindow.GetParaStyleName(iCount)
```

```
Selection.InsertPara
next iCount
End Sub
```

Paste-Methode

Fügt den Inhalt der Zwischenablage an der Position der Schreibmarke ein.

Siehe auch: PasteSpecial

KLASSE:
TextDrawText, WebSourceView, BaseTextSelection

SYNTAX:
Paste()

BEISPIEL:

```
Sub ExampleCutCopyPaste
Dim sWin As String REM für den  Fensternamen
REM TextDokument öffnen
Documents.Add("sWriter4")
sWin = ActiveWindow.Name
REM jetzt wird ein wenig HTML-Code eingegeben
ActiveWindow.Selection.Insert(Chr(60)&"HTML"&Chr(62)&Chr(60)&"B
ODY"&Chr(62)&"This is HTML !"&Chr(60)&"/BODY"&Chr(62))
ActiveWindow.SelectAll
ActiveWindow.Selection.Cut REM HTML-Code ausschneiden
Documents.Add("sWriter4/Web") REM HTML-Dokument öffnen und ...
Wait 100
ActiveDocument.FrameSetSource = True REM ... in die Quelltex-
tansicht ...
ActiveWindow.SelectAll
ActiveWindow.Paste REM .. den HTML-Code einfügen.
ActiveDocument.FrameSetSource = False
REM und zurück ...
ActiveDocument.FrameSetSource = TRUE
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveDocument.FrameSetSource = False
Windows( sWin ).Activate
ActiveWindow.Paste
end sub
```

PasteSpecial-Methode

Fügt den Inhalt der Zwischenablage in einem bestimmten Format ein, sofern dieses angegeben wird. Ohne Übergabe eines Parameters ruft die Methode zum Einfügen des Inhaltes der Zwischenablage den Dialog INHALT EINFÜGEN auf, in dem das Format des einzufügenden Inhaltes bestimmt werden kann.

Siehe auch: Paste, GetClipboardFormat, GetClipboardFormatCount, GetClipboardFormatName.

KLASSE:
TextDrawText, BaseTextSelection

SYNTAX:
PasteSpecial(Format As Long)

RÜCKGABE:
Integer

PARAMETER:
[Format]: Format-ID, die das Format bestimmt, in dem der Clipboardinhalt eingefügt wird.

Mit der Methode SELECTION.GETCLIPBOARDFORMATCOUNT ermitteln Sie die Anzahl der vorhandenen Clipboardformate.

Mit der Methode SELECTION.GETCLIPBOARDFORMAT(<Index%>) bekommen Sie die Format-ID für diesen Index zurück.

Mit dieser Format-ID setzen Sie bei PASTESPECIAL, in welchem Format der Clipboardinhalt eingefügt wird.

BEISPIEL:

```
Sub ExamplePasteSpecial
Const sWarn1="Das HTML-Format wurde nicht gefunden."
Const sWarn2="Es konnte die StarCalc-Tabelle nicht korrekt
eingefügt werden."
Dim sFormat As String
Dim iFormat As Long
Dim iState As Integer
Documents.Add("sCalc4")
wait 100
with ActiveDocument
.PutCell(1,1,1,"Text")   REM Ein wenig Text zum Testen
.PutCell(1,2,1,"=a1")    REM und eine Formel
.Range("a1:b2").Copy     REM ins Clipboard stellen
end with
Documents.Add("sWriter4")
wait 100
REM Jetzt muessen alle ClipBoardFormate untersucht werden.
for iState = 0 to ActiveWindow.Selection.GetClipboardFormat-
Count() -1
sFormat = Selection.GetClipboardFormatName( ActiveWindow.Selec-
tion.GetClipboardFormat(iState) )
REM Um dem Benutzer klar zu erläutern, welche Formate im Clip-
board zur Verfügung stehen,
REM werden die Formate mit den Identifikatoren in den Text
eingetragen.
```

```
ActiveWindow.Selection.Insert( sFormat & " : " & iState)
ActiveWindow.Selection.InsertPara
REM Wir verwenden das HTML Format, um die Tabelle aus StarCalc
in StarWriter zu kopieren.
if InStr( 1, Ucase(sFormat) , "HTML" ) > 0 then iFormat = Acti-
veWindow.Selection.GetClipboardFormat(iState)
next iState
ActiveWindow.Selection.InsertPara
iState = ActiveWindow.Selection.PasteSpecial( iFormat )
if iState<>1 then MsgBox sWarn1 & Chr(13) & sWarn2  ,64,"Clip-
board"
end sub
```

Pitch-Eigenschaft

Ermittelt, ob die Schriftart des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen proportional ist. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetFont gesetzte Schriftart abgefragt.

Falls ein im Dokument genannter Font im System nicht vorhanden ist, wird ein ähnlicher Font genommen. Diese Eigenschaft dient als Hilfsmittel dazu. Mit dieser Eigenschaft prüft man, ob der Originalfont ein propertionaler Font ist oder nicht. Diese Eigenschaft kann keinen nichtproportionalen Font in einen proportionalen umwandeln.

Siehe auch: SetFont, IsFont, StyleName, CharSet, Family, FamilyName.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
Pitch

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= PropFont]: Integer-Ausdruck, der angibt, ob die Schriftart proportional ist. Folgende Werte sind möglich:

0: Unbekannte Laufweite

1: Nichtproportionale Schriftart

2: Proportionale Schriftart

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExamplePitch
Documents.Add( "sWriter4" )
Selection.Insert "Die verwendete Schriftart ist eine "
Selection.InsertPara
Select Case Selection.Font.Pitch
Case 0
Selection.Insert "Schriftart mit unbekannten Laufweite "
Case 1
Selection.Insert "nichtproportionale Schriftart "
Case 2
Selection.Insert "proportionale Schriftart "
End Select
End Sub
```

PrevFieldByType-Methode

Springt im Dokument zum vorherigen, durch ID-Nummer und Name gekennzeichneten Feldtyp.

Siehe auch: GetFieldTypeCount, GetFieldTypeID, GetFieldTypeName, NextFieldByType.

KLASSE:
BaseText

SYNTAX:
PrevFieldByType(Type As Long, Name As String)

RÜCKGABE:
Boolean

PARAMETER:
Type: Long-Integer- Ausdruck, der die ID-Nummer des Feldtyps angibt, zu dem gesprungen werden soll.

Name: Zeichenfolgenausdruck, der den Namen des Feldtyps angibt, zu dem gesprungen werden soll.

Dieser Parameter ist nur für solche Feldtypen anzugeben, wo ein Name definiert wurde, also z.B. Nummernkreise. Hier ist er allerdings zwingend erforderlich, damit zum vorherigen Feld gesprungen werden kann.

Bei Feldtypen, die keine Namensdefinition haben, also z.B. Absender-Felder, springt man mit dieser Methode automatisch zum vorherigen Feld des Typs Absender, wenn kein Name oder ein Leerstring übergeben wird.

Beispiel:

```
Sub ExampleNextFieldByType
Const sVar ="Wichtig"
Dim sContent As String
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert(" Fülltext ")
.InsertField( 8, 1,sVar , "10" , 0 )
.Insert(" Fülltext ")
.InsertField( 8, 1, sVar, "100" , 0 )
.Insert(" Fülltext ")
.InsertField( 8, 1, sVar, "1000" , 0 )
.InsertPara
End With
ActiveWindow.Selection.JumpToStartOfDoc
Do While ActiveWindow.NextFieldByType( 8 , sVar )
   sContent = ActiveWindow.Selection.GetFieldPar2()
   ActiveWindow.Selection.SetCurField( 0 ,sVar, Val(sCon-
tent) +1 )
Loop
ActiveWindow.Selection.JumpToEndOfDoc
Do While ActiveWindow.PrevFieldByType( 8 , sVar )
   sContent = ActiveWindow.Selection.GetFieldPar2()
   ActiveWindow.Selection.SetCurField( 0 ,sVar, Val(sCon-
tent) +1 )
Loop
End Sub
```

Print-Methode

Druckt das aktuelle Dokument auf dem Drucker aus. Werden dieser Funktion keine Parameter übergeben, so wird der Dialog Drucken aufgerufen.

Siehe auch: PrintDefault

Klasse:
TextPrintPreview, WebSourceView, BaseTextEditView

Syntax:
Print(from As Integer, to As Integer, copies As Integer)

Rückgabe:
Boolean

Parameter:
[From]: Beliebiger Integer-Ausdruck, der die Seitenzahl der ersten zu druckenden Seite definiert. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung From = 1 genommen.

[To]: Beliebiger Integer-Ausdruck, der die Seitenzahl der letzten zu druckenden Seite definiert. Wird dieser Wert nicht übergeben, wird als Standardwert To = 9999 genommen.

[Copies]: Beliebiger Integer-Ausdruck, der die Anzahl der zu druckenden Kopien des Dokuments angibt. Wird dieser Wert nicht übergeben, wird in der Standardeinstellung Copies = 1 genommen.

Beispiel:

```
Sub ExamplePrint
Const sMessage_1 = "Dieses Dokument wird zweimal auf dem  "
Const sMessage_2 = "  ausgedruckt."
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.Insert ActiveWindow.Printer
Selection.Insert sMessage_2
REM Die erste Seiten mit 2 Kopien ausdrucken
ActiveWindow.Print( 1, 1, 2 )
End Sub
```

Printable-Eigenschaft

Setzt oder ermittelt die Eigenschaft "Drucken".

Klasse:
BaseTextFrame

Syntax:
Printable

Rückgabe:
Boolean

Beispiel:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
   Selection.Insert(sText)
   if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALS
```

```
Selection.Printable = TRUE
Selection.SendToBack
Selection.BringToFront
end sub
```

PrintAnnotations-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von Notizen.

KLASSE:
PrintSettings

SYNTAX:
PrintAnnotations

RÜCKGABE:
Long

PARAMETER:
[= Notes]: Long-Integer-Ausdruck, der angibt, ob und wie die Notizen im Ausdruck erscheinen. Folgende Werte sind möglich:

0: Im Ausdruck erscheinen keine Notizen.

1: Es werden nur die Notizen ausgedruckt.

2: Alle Notizen werden gesammelt am Ende des Dokumentes ausgedruckt.

3: Notizen werden am Ende der entsprechenden Seite ausgedruckt.

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
```

```
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintBackground-Eigenschaft

Bestimmt oder ermittelt die Druckoption für Hintergrundgrafiken.

KLASSE:
PrintSettings

SYNTAX:
PrintBackground

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Hintergrundgrafiken ausgedruckt werden (True) oder nicht (False).

Beispiel:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
```

```
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintBlackFont-Eigenschaft

Bestimmt oder ermittelt, ob die Druckeigenschaft "Schwarz drucken" aktiv ist.

KLASSE:
PrintSettings

SYNTAX:
PrintBlackFont

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Texte immer in schwarz gedruckt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
```

```
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintControl-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von Kontrollfeldern.

KLASSE:
PrintSettings

SYNTAX:
PrintControl

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Kontrollfelder im Ausdruck erscheinen (True) oder ob statt der Kontrollfelder leere Rahmen gedruckt werden (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
```

```
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintDefault-Methode

Druckt das aktuelle Dokument mit den Standard-Einstellungen. Ein Drucken-Dialog erscheint nicht.

Siehe auch: Print

Klasse:
TextPrintPreview, WebSourceView, BaseTextEditView

Syntax:
PrintDefault()

Rückgabe:
Boolean

Beispiel:

```
Sub ExamplePrintDefault
Const sMessage_1 = "Dieses Dokument wird auf dem REM "
Const sMessage_2 = "REM  ausgedruckt."
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.Insert ActiveWindow.Printer
Selection.Insert sMessage_2
ActiveWindow.PrintDefault
End Sub
```

PrintDrawing-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von Zeichnungen.

Klasse:
PrintSettings

Syntax:
PrintDrawing

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob Zeichnungen im Ausdruck erscheinen (True) oder nicht (False).

Beispiel:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
```

```
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintGraphic-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von Grafiken.

KLASSE:
PrintSettings

SYNTAX:
PrintGraphic

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Grafiken im Ausdruck erscheinen sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
```

```
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintLayout-Eigenschaft

Schaltet die Ansicht DRUCK-LAYOUT ein. Kann nur bei HTML Dokumenten verwendet werden.

KLASSE:
WebDocument

SYNTAX:
PrintLayout

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExamplePrintLayout
Documents.add "sWriter4/Web"
Selection.Insert("Das Dokument wird jetzt in den Druck Layout
Modus geschaltet.")
Wait 3000
ActiveDocument.PrintLayout = True
End Sub
```

PrintLeftPage-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von linken Seiten. Möchten Sie alle Seiten ausdrucken, so müssen PrintLeftPage und PrintRightPage gesetzt sein.

KLASSE:
PrintSettings

SYNTAX:
PrintLeftPage

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die linken Seiten des Dokuments gedruckt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
```

```
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintPagePreView-Methode

Öffnet bei angezeigter Seitenansicht des aktuellen Dokuments den Drucken-Dialog.

KLASSE:
TextPrintPreview

SYNTAX:
PrintPagePreView()

BEISPIEL:

```
Sub ExamplePrintPreview
Dim iCount As Integer
Documents.Add("sWriter4")
wait 100
For iCount= 1 to 6
   ActiveWindow.Selection.Insert("Das ist Seite " & iCount)
   ActiveWindow.InsertPageBreak
Next iCount
ActiveDocument.PrintPreview = True
ActiveWindow.GotoStartOfDoc
ActiveWindow.ShowTwoPages()
Wait 2000  REM Zeigt zweiseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
   ActiveWindow.PageDown
   Wait 1000
Next iCount
ActiveWindow.GotoEndOfDoc
```

```
ActiveWindow.ShowFourPages()
Wait 2000 REM zeigt vierseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
   ActiveWindow.PageUp
   Wait 1000
Next iCount
ActiveWindow.PrintPagePreView
ActiveDocument.PrintPreview = False
End Sub
```

PrintPreview-Eigenschaft

Schaltet die Seitenansicht ein oder aus.

KLASSE:
TextDocument

SYNTAX:
PrintPreview

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Seitenansicht ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExamplePrintPreview
Dim iCount As Integer
Documents.Add("sWriter4")
wait 100
For iCount= 1 to 6
ActiveWindow.Selection.Insert("Das ist Seite " & iCount)
ActiveWindow.InsertPageBreak
Next iCount
ActiveDocument.PrintPreview = True
ActiveWindow.GotoStartOfDoc
ActiveWindow.ShowTwoPages()
Wait 2000 REM Zeigt zweiseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageDown
Wait 1000
Next iCount
ActiveWindow.GotoEndOfDoc
ActiveWindow.ShowFourPages()
Wait 2000 REM zeigt vierseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageUp
Wait 1000
Next iCount
```

```
ActiveWindow.PrintPagePreView
ActiveDocument.PrintPreview = False
End Sub
```

PrintProspect-Eigenschaft

Bestimmt oder ermittelt, ob der Prospektdruck aktiv ist.

KLASSE:
PrintSettings

SYNTAX:
PrintProspect

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Prospektdruck ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
```

```
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintReverse-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den umgekehrten Ausdruck.

KLASSE:
PrintSettings

SYNTAX:
PrintReverse

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Seiten des Dokumentes in umgekehrter Reihenfolge gedruckt werden sollen.

True: Der Ausdruck beginnt mit der letzten Seite und endet mit der ersten Seite des Dokuments.

False: Der Ausdruck beginnt mit der ersten Seite und endet mit der letzten Seite des Dokuments.

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
```

```
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintRightPage-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von rechten Seiten. Möchten Sie alle Seiten ausdrucken, so müssen PrintRightPage und PrintLeftPage gesetzt sein.

KLASSE:
PrintSettings

SYNTAX:
PrintRightPage

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die rechten Seiten des Dokuments gedruckt werden sollen (True) oder nicht (False) .

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
```

```
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintSettings-Eigenschaft

Ermöglicht einen Zugriff auf die Druckoptionen.

Siehe Hilfe zu "StarBasic - Textdokumente - Klassen - Eigenschaften und Methoden der Klasse PrintSettings".

KLASSE:
StarWriter

SYNTAX:
PrintSettings

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
```

```
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

PrintTable-Eigenschaft

Bestimmt oder ermittelt die Druckoption für den Ausdruck von Tabellen.

KLASSE:
PrintSettings

SYNTAX:
PrintTable

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Tabellen im Ausdruck erscheinen (True) oder nicht (False).

BEISPIEL:

```
Sub ExamplePrintSettings
Const sNo = "Nein"
Const sYes = "Ja"
Const sMessage_1 = "Keine"
Const sMessage_2 = "Nur Notizen"
Const sMessage_3 = "Dokumentende"
Const sMessage_4 = "Seitenende"
Const sMessage_5 = "Starwriter Druckereinstellungen"
Const sMessage_6 = "Papierschachteinstellung "
Const sMessage_7 = "Druckereinstellungen"
Dim sInfo As String, sInfo_2 As String
Dim bInfo As Boolean
Dim iInfo As Integer
Documents.Add("swriter4")
With StarWriter.PrintSettings
sInfo_2 = .FaxName
If sInfo_2="" Then sInfo_2=sNo
sInfo = sInfo + "Fax: " + sInfo_2 + Chr(13)
bInfo = .PaperFromSetup
sInfo = sInfo + sMessage_6 + iif( bInfo , sMessage_7,
"StarWriter") +Chr(13)
iInfo = .PrintAnnotations REM Keine ...
Select Case iInfo
Case 0: sInfo_2 = sMessage_1
Case 1: sInfo_2 = sMessage_2
Case 2: sInfo_2 = sMessage_3
Case 3: sInfo_2 = sMessage_4
End Select
sInfo = sInfo + "Annotations: " + sInfo_2 +Chr(13)
bInfo = .PrintBackground
sInfo = sInfo + "Background: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintBlackFont
sInfo = sInfo + "Black Font: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintControl
sInfo = sInfo + "Controls: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintDrawing
sInfo = sInfo + "Drawings: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintGraphic
sInfo = sInfo + "Graphics: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintLeftPage
sInfo = sInfo + "Left pages: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintProspect
sInfo = sInfo + "Prospect: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintReverse
```

```
sInfo = sInfo + "Reverse: " + iif( bInfo ,sYes,sNo) +Chr(13)
bInfo = .PrintRightPage
sInfo = sInfo + "Right pages: " + iif( bInfo ,sYes,sNo) +
Chr(13)
bInfo = .PrintTable
sInfo = sInfo + "Table: " + iif( bInfo ,sYes,sNo) +Chr(13)
End With
MsgBox sInfo, 64, sMessage_5
End Sub
```

Prop-Eigenschaft

Bestimmt oder ermittelt die zur Schriftgröße der verknüpften Absatzvorlage relative Schriftgröße der selektierten Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetFontHeight gesetzte relative Schriftgröße abgefragt.

Eine Abfrage der Prop-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsFontHeight bestimmt wurde, daß nach einer bestimmten Schriftgröße gesucht werden soll, d. h. IsFontHeight = 2.

Siehe auch: SetFontHeight, IsFontHeight, Size.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
Prop

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Value]: Integer-Ausdruck, der die relative Schriftgröße in Prozent angibt.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Selection.Font.Prop
```

PropLineHeight-Eigenschaft

Bestimmt oder ermittelt für den aktuellen Absatz den prozentualen Wert des Zeilenabstandes relativ zur Zeilenhöhe, wenn InterLineSpace den Wert 1 liefert; beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLineSpacing gesetzte prozentuale Zeilenabstand abgefragt.

Der zur Zeilenhöhe prozentuale Zeilenabstand muß gesetzt werden, wenn dem aktuellen Absatz mit der InterLineSpace-Eigenschaft ein 1,5-zeiliger, ein 2-zeiliger oder ein proportionaler Zeilenabstand zugewiesen wird, d. h. InterLineSpace = 1. Umgekehrt kann man mittels PropLineHeight nur dann einen prozentualen Zeilenabstand setzen, wenn InterLineSpace = 1 ist.

Eine Abfrage der PropLineHeight-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLineSpacing bestimmt wurde, daß nach einem bestimmten Zeilenabstand gesucht werden soll, d. h. IsLineSpacing = 2.

Siehe auch: SetLineSpacing, IsLineSpacing, LineSpace, LineHeight, InterLineSpace, InterSpace.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
PropLineHeight

RÜCKGABE:
Long

PARAMETER:
Bei Absätzen:

[= Percent]: Long-Integer-Ausdruck, der den zur Zeilenhöhe relativen Zeilenabstand in Prozent angibt.

Die folgenden Beispiele zeigen, welche Werte der PropLineHeight-Eigenschaft zugewiesen werden können:

Zeilenabstand : Wert

1,5-zeilig : PropLineHeight = 150 (150% der Zeilenhöhe).

2-zeilig : PropLineHeight = 200 (200% der Zeilenhöhe).

Proportional : Prozentzahl des zur Zeilenhöhe relativen Zeilenabstandes. Soll der Zeilenabstand beispielsweise 80% der Zeilenhöhe betragen, muß der PropLineHeight-Eigenschaft der Wert 80 zugewiesen werden.

Beim Suchen und Ersetzen von Attributen:
(nur lesen)

Beispiel:

```
Sub ExampleParagraphSpacing2
REM den lateinischen Blindtext verwenden
Const sBlindText = "BT"
Documents.Add("sWriter4")
With ActiveWindow.Selection
.ExpandGlossary( sBlindText )
wait 1000
.Paragraph.InterLineSpace = 1   REM Typ des Zeilenzwischenraume
auf REM relativREM  einstellen.
.Paragraph.PropLineHeight= 185  REM Angabe in Prozent
end with
end sub
```

Protect-Methode

Schaltet den Schutz der aktuellen Tabellenzelle oder der Zellen des selektierten Tabellenbereichs ein.

Klasse:
TextTable

Syntax:
Protect()

Beispiel:

```
Sub ExampleUnsetCellsReadonly
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
Selection.JumpToNextCell
Selection.Insert "Geschützte Zelle"
Selection.Protect
Selection.GotoStart
REM Geschützte Zellen werden übersprungen
Selection.GoRight 1, True
Selection.UnsetCellsReadonly
Selection.JumpToPrevCell
End Sub
```

Protected-Eigenschaft

Bestimmt oder ermittelt, ob Inhalt, Position und/oder Größe des selektierten Objekts geschützt sind.

KLASSE:
BaseTextFrame

SYNTAX:
Protected

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die angibt, ob und wie das selektierte Objekt geschützt ist:

Leerstring: Das Objekt ist nicht geschützt.

C: Der Inhalt ist geschützt.

P: Die Position ist geschützt.

S: Die Größe ist geschützt.

Durch Verkettung der einzelnen Buchstaben, kann das Objekt geschützt werden. Wenn am Objekt z. B. Inhalt und Größe geschützt sind, wird CS zurückgegeben.

BEISPIEL:

```
Sub ExampleProtected
Documents.Add "sWriter4"
Selection.InsertFrame(0, "3cm", "1cm", "4cm", "2cm")
Selection.Protected = "SP"
Selection.Insert("Dieser Rahmen ist geschützt:" &
Selection.Protected)
Wait 1000
Selection.Protected = "CSP"
Selection.GotoAnchor
Selection.Insert("Der Inhalt dieses Rahmens kann jetzt auch
nicht mehr geändert werden.")
End Sub
```

Eigenschaften und Methoden mit "Q"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Q" beginnen.

QualifierSeparator-Methode

Liefert den für SQL-Statements nötigen Separator. Dieser ist in der Regel ein Komma, kann bei einigen Datenbanktreibern aber auch ein anderes Zeichen sein. Als Rückgabe wird die Nummer des ASCII-Zeichens geliefert. Bei einem Komma wäre das der Wert 44. Dieser Wert kann mit der CHR$ -Funktion in ein "normales" Zeichen gewandelt werden.

KLASSE:
BaseTextEditView

SYNTAX:
QualifierSeparator()

RÜCKGABE:
Integer

BEISPIEL:

```
Sub Main
Dim DataName as string, TableName as string, FieldName as
string
Dim DataNumber as integer, Status as integer, TableCount as
integer, TableNumber as integer
Dim ExVar as Variant
for DataNumber=1 to Activewindow.DataDefinedDatabaseNumber()
DataName = Activewindow.DataDefinedDatabase(DataNumber)
Status = ActiveWindow.OpenDatabaseConnection(DataName)
ExVar = Activewindow.QualifierSeparator
TableCount = Activewindow.DataTableNumber(DataName)
for TableNumber=1 to TableCount
TableName = Activewindow.DataGetTableName(DataName,
TableNumber)
ExVar = Activewindow.IsTableReadOnly(TableName)
Status = ActiveWindow.OpenTableConnection(TableName)
ExVar = Activewindow.GetDatabaseLimits(1)
Activewindow.DataSelect("SELECT * FROM " & TableName & " ;")
Status = Activewindow.DataFirstRecord  REM  Wichtig da sonst
keine definierte Position in der Datenbank
Status = Activewindow.DataRecordID
ExVar = Activewindow.DataColumn(1)
FieldName = Activewindow.DataColumnName(1)
ExVar = Activewindow.DataColumn(FieldName)
Status = Activewindow.DataNextRecord
Status = Activewindow.DataRecord(2)
```

```
Status = Activewindow.DataPreviousRecord
Status = Activewindow.DataLastRecord
ExVar = activewindow.GetDataRecordCount
ExVar = activewindow.DataPhysicalRecordID
ExVar = activewindow.DataRecordID
Activewindow.DataUpdate
Status = Activewindow.DataSwitch(DataName, TableName)
Activewindow.CloseTableConnection(TableName)
next TableNumber
ExVar = Activewindow.GetActiveDatabaseConnection
Activewindow.CloseDatabaseConnection(DataName)
next DataNumber
End Sub
```

QuoteIdentifier-Methode

Gibt den Namen einer Datenbank-Tabelle so zurück, daß er vom verwendeten Datenbank-Treiber interpretiert werden kann.

Je nach Datenbank-Treiber müssen Tabellennamen oder Spaltennamen in SQL-Statements unterschiedlich gequoted werden (i.d.R. einfache oder doppelte Anführungszeichen). Diese Methode gibt den übergebenen Namen korrekt gequoted zurück.

KLASSE:
BaseTextEditView

SYNTAX:
QuoteIdentifier(TableName As String)

RÜCKGABE:
String

PARAMETER:
TableName: Name der Tabelle.

BEISPIEL:

```
sub Main
Dim DataName as string, Status as integer, ExVar as variant
Documents.add "sWriter"
DataName = Activewindow.DataDefinedDatabase(1)
status = ActiveWindow.OpenDatabaseConnection(DataName)
ExVar = Activewindow.QuoteIdentifier("Test") REM  ExVar =
"Test"
end sub
```

Eigenschaften und Methoden mit "R"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "R" beginnen.

Redo-Methode

Wiederholt die mittels Undo rückgängig gemachte Aktion.

Siehe auch: Undo

KLASSE:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection

SYNTAX:
Redo()

BEISPIEL:

```
Sub ExampleDoUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sMessage_5 = "RedoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.DoUndo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
Print sMessage_5 & ActiveWindow.Selection.RedoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.DoUndo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
```

```
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

RedoAction-Eigenschaft

Ermittelt die im Menü Bearbeiten unter dem Menüpunkt Wiederholen aufgeführte Beschreibung der zuvor rückgängig gemachten Aktion.

Klasse:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection

Syntax:
RedoAction

Rückgabe:
String

Parameter:
(nur lesen)

Beispiel:

```
Sub ExampleDoUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite "
Const sMessage_2 = "Neue Seite "
Const sMessage_3 = "Kein weiteres Undo möglich "
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sMessage_5 = "RedoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.DoUndo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
Print sMessage_5 & ActiveWindow.Selection.RedoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.DoUndo
StarWriter.UndoCount = 50
```

```
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

Refresh-Methode

Berrechnet das Seitenlayout neu und zeigt anschließend die aktualisierte Darstellung.

KLASSE:
TextPrintPreview, BaseTextEditView

SYNTAX:
Refresh()

BEISPIEL:

```
Sub ExampleRefresh
Documents.Add("sWriter4")
ActiveWindow.Refresh
End Sub
```

RelativeSize-Eigenschaft

Bestimmt oder ermittelt die relative Schriftgröße der hoch- oder tiefgestellten Zeichen bezogen auf die aktuelle Schriftgröße. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetEscapement gesetzte Schriftgröße abgefragt.

Siehe auch: SetEscapement, IsEscapement, Escapement.

KLASSE:
ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
RelativeSize

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= RelSize]: Integer-Ausdruck, der die Prozentzahl aufnimmt, die die Schriftgröße der selektierten Zeichen in Relation zur aktuellen Schriftgröße angibt.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
```

```
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Remove-Methode

Schaltet die Aufzählung oder Numerierung aus.

KLASSE:
BaseTextList

SYNTAX:
Remove()

BEISPIEL:

```
Sub ExampleNumbering
Const sMessage_1 = "Dies ist die normale Numerierung"
Const sMessage_2 = "Dies ist eine normale Aufzählung"
Const sMessage_3 = "Dies ist eine Aufzählung aus Sonderzeichen"
Const sMessage_4 = "Dies ist eine Aufzählung aus Grafiken"
Const sMessage_5 = "Dies ist eine Numerierung mit römischen
Zahlen"
Dim sBulletPath As String
sBulletPath = Application.Pathsettings.Graphics &
"\CLIPART\BULLET\APPLE.GIF"
Documents.Add( "sWriter4" )
Selection.InsertNumbering()
selection.Insert sMessage_1
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_2
Selection.DefaultBullet
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_3
Selection.InsertNumbering()
Selection.BulletCode = 109
Selection.BulletFont = "WingDings"
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_4
Selection.InsertNumbering()
Selection.BulletImageURL = sBulletPath
Selection.InsertPara
Selection.Remove
Selection.InsertPara
selection.Insert sMessage_5
Selection.InsertNumbering()
Selection.NumberType = 2
Selection.InsertPara
Selection.Remove
Selection.InsertPara
End Sub
```

RenameGlossary-Methode

Mit dieser Methode können Sie Name und Kürzel eines AutoTextes ändern. Der AutoText-Baustein mit dem Kürzel ShortName innerhalb der AutoText-Gruppe GroupName erhält den Namen NewName und das neue Kürzel NewShortName.

Die Methode liefert TRUE, wenn die Umbenennung erfolgreich war, sonst FALSE.

KLASSE:
BaseText

SYNTAX:
RenameGlossary(GroupName As String, ShortName As String, NewName As String, NewShortName As String)

RÜCKGABE:
Boolean

PARAMETER:
GroupName: AutoText-Gruppe, die den gewünschten Textbaustein enthält.

ShortName: aktuelles Kürzel des Textbausteins.

NewName: Name, der dem Baustein zugewiesen wird.

NewShortName: Kürzel, das dem Baustein zugewiesen wird.

BEISPIEL:

```
Sub ExampleRenameGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sNewName = "Neuer Name"
Const sNewShortName = "NN"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es wurde folgender Autotext angelegt : "
Const sMessage_3 = "Nun heißt der Autotext : "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.Insert Selection.GetGlossaryBlockName( sGroupName, 0
)
Selection.RenameGlossary(sGroupName, sShortName, sNewName,
sNewShortName)
Selection.InsertPara
Selection.Insert sMessage_3
```

```
Selection.Insert Selection.GetGlossaryBlockName( sGroupName, 0
)
Selection.DeleteGlossary( sGroupName, sNewShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

RenameRegion-Methode

Weist dem angegebenen Bereich eine neue Bezeichnung zu.

KLASSE:
BaseTextSelection

SYNTAX:
RenameRegion(RegionName As String, NewName As String)

PARAMETER:
RegionName: Zeichenfolgeausdruck, der die Bezeichnung des Bereiches angibt, dem eine neue Bezeichnung zugewiesen werden soll.

NewName: Zeichenfolgeausdruck, der die neue Bezeichnung des angegebenen Bereichs aufnimmt.

BEISPIEL:

```
Sub ExampleEditRegion
Const sRegionName = "Bereich"
Const sNewRegionName = "NeuerBereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  heißt jetzt : "
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.Escape
Selection.InsertPara
Selection.RenameRegion( sRegionName, sNewRegionName )
Selection.Insert sMessage_2 & sRegionName & sMessage_3 & Selec-
tion.GetRegionName(0)
End Sub
```

RenameUserFields-Methode

Dient zur Umbenennung von Variablen- oder Benutzerfeldnamen. Diese Methode ändert automatisch alle Felder, in denen die Variablen- oder Benutzerfeldnamen verwendet wurden. Diese Methode liefert den Wert False, wenn OldName nicht im Dokument existiert.

KLASSE:
BaseTextSelection

SYNTAX:
RenameUserFields(OldName As String, NewName As String)

RÜCKGABE:
Boolean

PARAMETER:
OldName: alter Name der Variablen oder des Benutzerfelds.

NewName: neuer Name der Variablen oder des Benutzerfelds.

BEISPIEL:

```
Sub ExampleRenameUserfields
Const sUsername = "Benutzerfeld"
Const sUserContent = "BenutzerInhalt"
Const sNewName = "NeuerName"
Documents.Add( "sWriter4" )
Selection.InsertField( 20, 0, sUsername, sUserContent, 0)
Selection.GoLeft( 1, TRUE )
Selection.RenameUserFields( sUsername, sNewName )
End Sub
```

Repaginate-Methode

Führt eine Neuformatierung des Dokuments durch.

KLASSE:
BaseTextSelection

SYNTAX:
Repaginate()

BEISPIEL:

```
Sub ExampleRepaginate
Documents.Add "sWriter4"
With ActiveWindow.Selection
   .Insert("Manchmal ist es wichtig, daß die Seiten neu forma-
tiert werden. ")
   .InsertPara
   .Insert("Zum Beispiel bei großen Dokumenten, in denen man
```

```
viel Text oder viele Objekte gelöscht und verschoben hat.")
   .InsertPara
   .Repaginate
End With
End Sub
```

Repeat-Methode

Wiederholt die letzte Aktion.

KLASSE:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection

SYNTAX:
Repeat()

BEISPIEL:

```
Sub ExampleRepeatAction
Const sMessage_1 = "Dieser Text wird durch RepeatAction mehr-
fach eingefügt"
Const sMessage_2 = "RepeatAction liefert den Text: "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveWindow.Paste REM Diese Action wird mehrfach wiederholt
Print sMessage_2 & ActiveWindow.Selection.RepeatAction
ActiveWindow.Selection.Repeat
ActiveWindow.Selection.Repeat
End Sub
```

RepeatAction-Eigenschaft

Liefert eine Zeichenkette für die Aktion, die der Befehl Repeat ausführen kann.

Siehe auch: Repeat

KLASSE:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection

SYNTAX:
RepeatAction

RÜCKGABE:
String

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExampleRepeatAction
Const sMessage_1 = "Dieser Text wird durch RepeatAction mehr-
fach eingefügt"
Const sMessage_2 = "RepeatAction liefert den Text: "
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
ActiveWindow.SelectAll
ActiveWindow.Copy
ActiveWindow.Paste REM Diese Action wird mehrfach wiederholt
Print sMessage_2 & ActiveWindow.Selection.RepeatAction
ActiveWindow.Selection.Repeat
ActiveWindow.Selection.Repeat
End Sub
```

RepeatSearch-Methode

Setzt die Suche nach dem zuletzt gesuchten Ausdruck fort.

Siehe auch: Replace, Search, SearchAll

Klasse:
BaseTextEditView

Syntax:
RepeatSearch()

Rückgabe:
Boolean

Beispiel:

```
Sub ExampleSearchAndReplace
Const sMessage = "a bFett c dFett e fFett g"
Documents.Add("sWriter4")
with  ActiveWindow.Selection
.Insert( sMessage )
.GotoPrevWord
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
```

```
.Font.Bold = soWeightBold
end with
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
End Sub
```

Replace-Methode

Sucht und markiert eine Zeichenfolge, die bei erneutem Anwenden der Replace-Methode ersetzt wird. Konnte die Aktion erfolgreich durchgeführt werden, wird der Wert True zurückgegeben, andernfalls False.

Sind keine Übergabeparameter definiert, wird der Dialog SUCHEN & ERSETZEN geöffnet.

Wenden Sie diese Methode auf einen Text an, der bereits selektiert ist, dann wird dieser auf jeden Fall durch die mit dem Parameter ReplaceString angegebene Zeichenfolge ersetzt.

Wurden vor Aufruf dieser Methode Suchattribute gesetzt, dann sind die Übergabeparameter optional. In diesem Fall werden die fehlenden Parameter automatisch mit den aktuellen Einstellungen der Search- und ReplaceAttributes gesetzt. Sonst wird der Suchdialog geöffnet.

Siehe auch: RepeatSearch, ReplaceAll, Search.

KLASSE:
BaseTextEditView

SYNTAX:
Replace(SearchString As String, ReplaceString As String)

RÜCKGABE:
Boolean

PARAMETER:
[SearchString]: Ein Zeichenfolgeausdruck, der den zu suchenden Text aufnimmt. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den SearchAttributes ausgelesen, wenn diese zuvor gesetzt wurden. Ist im Dokument Text selektiert, ist die Angabe des Parameters SearchString nicht zwingend erforderlich, da dann der selektierte Text statt des Suchtextes mit der Replace-Methode ersetzt wird.

[ReplaceString]: Ein Zeichenfolgeausdruck, der angibt, durch welchen Text der mit dem ersten Parameter übergebene Text ersetzt werden soll. Ist im Dokument Text selektiert, wird dieser ersetzt. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den ReplaceAttributes ausgelesen, wenn diese zuvor gesetzt wurden.

BEISPIEL:

```
Sub ExampleSearchAndReplace
Const sMessage = "a bFett c dFett e fFett g"
Documents.Add("sWriter4")
with  ActiveWindow.Selection
.Insert( sMessage )
.GotoPrevWord
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
end with
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
End Sub
```

ReplaceAll-Methode

Sucht und ersetzt alle gefundenen Zeichenfolgen mit einer anderen Zeichenfolge.

Sind keine Übergabeparameter definiert, wird der Dialog SUCHEN & ERSETZEN geöffnet.

Bei Aufruf dieser Methode werden gefundene Zeichenfolgen in jedem Fall ersetzt. Es findet keine Abfrage mehr statt, in der ein Durchführen dieser Aktion bestätigt werden kann.

Wurden vor Aufruf dieser Methode Suchattribute gesetzt, dann sind die Übergabeparameter optional. In diesem Fall werden die fehlenden Parameter automatisch mit den aktuellen Einstellungen der Search- und ReplaceAttributes gesetzt. Sonst wird der Suchdialog geöffnet.

Siehe auch: Replace, SearchAll

KLASSE:
BaseTextEditView

SYNTAX:
ReplaceAll(SearchString As String, ReplaceString As String)

PARAMETER:
[SearchString]: Ein Zeichenfolgeausdruck, der den zu ersetzenden Text aufnimmt. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den SearchAttributes ausgelesen, wenn diese zuvor gesetzt wurden.

[ReplaceString]: Ein Zeichenfolgeausdruck, der angibt, durch welchen Text der mit dem ersten Parameter übergebene Text ersetzt werden soll. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den ReplaceAttributes ausgelesen, wenn diese zuvor gesetzt wurden.

BEISPIEL:

```
Sub ExampleReplaceAll
Const sWord_1 = "Beispielwort"
Const sWord_2 = "Geändertes Wort"
Dim iCount As Integer
Documents.Add( "Swriter4" )
For iCount=1 to 5
Selection.Insert sWord_1
Selection.InsertPara
```

```
next iCount
ActiveWindow.SearchAll sWord_1
For iCount=1 To Selection.SelectionCount
Selection.JumpToNextSelection
Next iCount
For iCount=1 To Selection.SelectionCount
Selection.JumpToPrevSelection
Next iCount
ActiveWindow.ReplaceAll sWord_1, sWord_2
End Sub
```

ReplaceAttributes-Eigenschaft

Das Objekt enthält die Information, welche Attribute beim Ersetzen angewendet werden sollen. Im Gegensatz zu den SearchAttributes gibt es hier nur zwei mögliche Zustände für jedes Attribut: nicht gesetzt und gesetzt. Der Zustand kann erfragt und auf nicht gesetzt eingestellt werden. Er wird durch Setzen eines Attributwertes auf "gesetzt" umgeschaltet.

Der Wert der gesetzten Attribute kann auch erfragt werden.

KLASSE:
BaseTextEditView

SYNTAX:
ReplaceAttributes

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleSearchAndReplace
Const sMessage = "a bFett c dFett e fFett g"
Documents.Add("sWriter4")
with  ActiveWindow.Selection
.Insert( sMessage )
.GotoPrevWord
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
```

```
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
end with
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
End Sub
```

Reset-Methode

Setzt alle für das Suchen und Ersetzen von Formaten gesetzten Attribute zurück. Diese Methode hat die gleiche Funktion wie die Schaltfläche KEIN FORMAT im Dialog SUCHEN & ERSETZEN.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
Reset()

BEISPIEL:

```
Sub ExampleSearchAndReplaceBold
Documents.Add("sWriter4")
REM erstmal eine fette Zeile erstellen
With ActiveWindow.Selection
   .Insert("a bBold c dBold e fBold g hBold i")
   .GotoPrevWord
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
```

```
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
End With
ActiveWindow.SearchAttributes.Reset
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search ""
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
ActiveWindow.ReplaceAttributes.Reset
REM ab jetzt werden die gefundenen Stellen gelöscht
ActiveWindow.Replace
ActiveWindow.Replace
End Sub
```

ResetAllActions-Methode

Setzt alle Aktionen zurück, die mittels ActionStart aufgerufen wurden. Die Anzahl der zurückgesetzten Aktionen wird als Rückgabewert geliefert.

Siehe auch: ActionStart, SetActions.

KLASSE:
BaseTextSelection

SYNTAX:
ResetAllActions()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleActions
Const Message = "Gleich kommt noch mehr Text"
Const Message_1 = "Dieser Text erscheint erst nach einer Verzö-
gerung"
Const Message_2 = "Anzahl der Actionebenen : "
Dim iAction As Integer
Documents.Add( "sWriter4" )
Selection.Insert Message
Selection.InsertPara
Selection.ActionStart
Selection.ActionStart
```

```
Selection.ActionStart
Selection.Insert Message_1
Selection.InsertPara
Wait 100
iAction = Selection.ResetAllActions
Selection.Insert Message_2 & iAction
Selection.SetActions( iAction )
Selection.ActionEnd
Selection.ActionEnd
Selection.ActionEnd
End Sub
```

ResetAllLocks-Methode

Setzt alle mit LockPaint gesetzten Lock-Ebenen zurück.

Die Anzahl der zurückgesetzten Ebenen wird als Long-Integer-Wert zurückgeliefert.

siehe auch: LockPaint

KLASSE:
BaseTextEditView

SYNTAX:
ResetAllLocks()

RÜCKGABE:
Long

BEISPIEL:

```
sub Main
Documents.add "sWriter"
Activewindow.Lockpaint(True)
selection.Insert "Test"
selection.InsertPara
selection.Insert "Test"
Print Activewindow.IsPaintLocked
Activewindow.Lockpaint(False)
Activewindow.ResetAllLocks
end sub
```

ResetAttributes-Methode

Entfernt alle harten Textattribute des selektierten Textes.

KLASSE:
TextDrawText, BaseText

SYNTAX:
ResetAttributes()

Beispiel:

```
Sub ExampleStyles
REM Das Beispiel legt drei Absatzvorlagen an, von denen eine
ein Kind der anderen ist.
Const sMessage= "Anwendung der Vorlage: "
Const sDefaultTemp="Standard"
Const sTemp1="Stil_Rot"
Const sTemp1a="Stil_Rot_Zentriert"
Const sTemp2="Stil_Zentriert"
Dim sStyle As String
Documents.Add("sWriter4")
with ActiveWindow.Selection
.ResetAttributes      REM Für den aktuellen Absatz die harten
Attribute löschen
.StyleApply( sDefaultTemp ,2)    REM Absatz auf Standard setzen
sStyle = .StyleApplyState
if sStyle <> sDefaultTemp then     REM ist tatsächlich auf
Standard  ?
MsgBox("Konnte die Absatzvorlage nicht
zurücksetzen",64,"Warnung")  REM gibt Warnung aus
end if
.StyleNewByExample( sTemp1 ,2)    REM setzt die Absatzvorlage
zurück
.Interior.BackColor = RGB (255,127,127)
.Interior.Style = 1
.StyleUpdateByExample( sTemp1 ,2)  REM setzt die Vorlage auf
rot
.ResetAttributes      REM Zelle auf Standard setzen
.StyleApply( sDefaultTemp ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp2 ,2)
.ResetAttributes
.StyleApply( sTemp1 ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp1a ,2)  REM Kind-Vorlage von sTemp1a ,
mit zusätzlich zentriertem Text
.ResetAttributes
  REM jetzt werden die Vorlagen angewendet
.Insert( sMessage & sTemp1 )
.StyleApply( sTemp1 ,2)
.InsertPara
.Insert(sMessage & sTemp1a )
.StyleApply( sTemp1a ,2)
.InsertPara
.Insert(sMessage & sTemp2 )
.StyleApply( sTemp2 ,2)
.InsertPara
end with
End Sub
```

ResetTableProtection-Methode

Hebt den Zellenschutz für die gesamte Tabelle, in der sich der Cursor befindet, auf.

KLASSE:
BaseText

SYNTAX:
ResetTableProtection()

BEISPIEL:

```
sub ExampleTableProtect
Dim sTabName as string
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
sTabName = Selection.TableName
REM Die ganze Spalte
Selection.GoDown 2, True
Selection.Protect
Selection.GotoTable sTabName
Selection.Insert("Das steht eigentlich in Zelle A1.")
Selection.ResetTableProtection
end sub
```

Right-Eigenschaft

Bestimmt oder ermittelt den Wert für das Skalieren der Ansicht der selektierten Grafik an der rechten Seite. Negative Werte beschneiden die Ansicht der Grafik, positive verbreitern den Rand zwischen Rahmen und Grafik und verkleinern die Grafik dabei.

Siehe auch: Left, Top und Bottom.

KLASSE:
BaseTextGraphic

SYNTAX:
Right

RÜCKGABE:
Long

PARAMETER:
[= Value]: Gibt den Wert an, um den die Grafik an der rechten Seite geändert werden soll. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

BEISPIEL:

```
Sub ExampleGraphicCrop
Dim sGrfPath as String
Dim iCount As Integer
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Wait 3000
With Selection  REM in zwei Stufen wird das Bild verkleinert
For iCount = 2 to -1 Step -1
   .Bottom = Trim(Str(iCount) & "cm")
   .Top= Trim(Str(iCount) & "cm")
   .Left= Trim(Str(iCount) & "cm")
   .Right= Trim(Str(iCount) & "cm")
   Wait 3000
Next iCount
End With
Selection.Escape
End Sub
```

RightMargin-Eigenschaft

Bestimmt oder ermittelt den Abstand vom rechten Rand des selektierten oder aktiven Objekts zu seiner Umgebung. Dieser Abstand ist für verschiedene Objekte unterschiedlich definiert: Bei einem TextEditView-Objekt ist es der Abstand bis zur rechten Papierkante, bei einem TextTable-Objekt zum rechten Seitenrand. Bei Objekten der Klassen TextFrame, TextGraphic, TextOLEObject ist es der Abstand bis zu dem danebenliegenden umlaufenden Text. Ist ein Absatz selektiert (Paragraph, TextInTable, ListInTable), so liefert RightMargin den rechten Einzug des Absatzes.

Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetLeftRightParaMargin gesetzte rechte Einzug abgefragt.

Eine Abfrage der RightMargin-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsLeftRightParaMargin bestimmt wurde, daß nach einem bestimmten Absatzeinzug gesucht werden soll, d. h. IsLeftRightParaMargin = 2.

Siehe auch: BottomMargin, LeftMargin, TopMargin.

Für SearchAttributes, ReplaceAttributes:

SetLeftRightParaMargin, IsLeftRightParaMargin, IndentFirstLine.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
RightMargin

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der den jeweiligen Abstand angibt. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleTableMargin
Documents.Add("sWriter4")
Selection.InsertTable( 4 , 4 )
Selection.Select
REM Tabellenausrichtung auf REM ManuellREM  stellen
Selection.Alignment = 0
Selection.BottomMargin = "3cm"
Selection.TopMargin = "2cm"
Selection.RightMargin = "1cm"
Selection.LeftMargin = "1cm"
End Sub
```

Rows-Eigenschaft

Liefert die Anzahl der Tabellenzeilen zurück, die auch in der Statusleiste angezeigt wird, wenn man sich mit dem Cursor in der letzten Zelle befindet.

Die Funktion arbeitet nur dann korrekt, wenn in der Tabelle noch keine Zellen geteilt oder zusammengefaßt wurden.

KLASSE:
BaseTextTable

SYNTAX:
Rows

RÜCKGABE:
Long

PARAMETER:
Rückgabe: Anzahl der Tabellenzeilen.

BEISPIEL:

```
Sub ExampleTableInsertDelete
Dim sPString As String
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.InsertRows(1, TRUE)
Selection.InsertColumns(1, TRUE)
REM Es wird eine Zeile links oben eingefügt
Selection.InsDelRowCol(TRUE, TRUE, FALSE, FALSE)
Selection.DeleteRows
Selection.DeleteColumns
sPString =  "Die Tabelle besteht aus : " & Selection.Columns &
" Spalten und "
sPString = sPString & Selection.Rows & " Zeilen."
Print sPString
End Sub
```

Eigenschaften und Methoden mit "S"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "S" beginnen.

SaveAs-Methode

Speichert das Dokument unter dem angegebenen Namen. Optional können weitere Parameter übergeben werden, um das Dokument in einem Fremdformat oder mit einem Paßwort abzuspeichern.

KLASSE:
WebSourceView

SYNTAX:
SaveAs(FileName As String, FilterName As String, PassWord As String, FilterOptions As String)

RÜCKGABE:
Boolean

PARAMETER:
[FileName]: Dateiname, unter dem das Dokument gespeichert werden soll. Wird er nicht angegeben, wird der Dialog SPEICHERN UNTER aufgerufen.

[FilterName]: Name des Filters, der beim Speichern verwendet werden soll. Die Angabe des Dateityps, unter dem das Dokument gespeichert werden soll, ist optional.

[PassWord]: Zeichenfolge, die das Paßwort angibt, mit dem das Dokument gespeichert werden soll. Dieser Parameter ist optional.

[FilterOptions]: Applikationsspezifische Filterangabe, die im Writer nicht ausgewertet wird. Dieser optionale Parameter ist hier ohne Bedeutung, da es im StarWriter-Modul keinen Filter gibt, der diesen Parameter verwendet.

BEISPIEL:

```
Sub ExampleUpdateAllLinks
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Datei wird ein Bereich"
Const sMessage_2 = "Hier wird der Bereich eingefügt."
Const sMessage_3 = "So, nun hat sich der Bereich geändert. "
Dim sDocName As String
sDocName = "BerDoc"
REM Document definierem was als Bereich gelinkt wird.
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
ActiveDocument.SaveAs Application.PathSettings.Work &"\"&
sDocName &".SDW"
sDocName = ActiveDocument.FullName
sWin = ActiveWindow.Name
REM neues Dokument mit einem Bereich anlegen
Documents.Add( "sWriter4" )
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
REM Bereich als Link einfügen
Selection.InsertRegion( sRegionName, "True", FALSE, TRUE,
sDocName)
REM den Bereich im Hintergrund ändern
Windows(sWin).JumpToEndOfDoc
Windows(sWin).Insert(sMessage_3)
Windows(sWin).InsertPara
REM das Dokument was zu dem Fenster gehört Speichern.
Windows(sWin).Document.Save
Wait 1000
Selection.UpdateAllLinks(TRUE, FALSE)
End Sub
```

Search-Methode

Sucht ab der Position der Schreibmarke nach einem bestimmten Text. Wird der Suchbegriff gefunden, setzt die Methode die Schreibmarke an die Fundstelle und gibt den Wert True zurück, ansonsten False.

Ohne Übergabeparameter wird der Dialog SUCHEN & ERSETZEN aufgerufen.

Wurden vor Aufruf dieser Methode Suchattribute gesetzt, dann ist der Übergabeparameter optional. In diesem Fall wird der fehlende Parameter automatisch mit den aktuellen Einstellungen der SearchAttributes gesetzt.

Siehe auch: RepeatSearch, Replace, SearchAll.

KLASSE:
BaseTextEditView

SYNTAX:
Search(SearchString As String)

RÜCKGABE:
Boolean

PARAMETER:
[SearchString]: Ein Zeichenfolgeausdruck, der den Text aufnimmt, der gesucht werden soll. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den SearchAttributes ausgelesen, wenn diese zuvor gesetzt wurden. Sonst wird der Suchdialog geöffnet.

BEISPIEL:

```
Sub ExampleSearchAndReplace
Const sMessage = "a bFett c dFett e fFett g"
Documents.Add("sWriter4")
with  ActiveWindow.Selection
.Insert( sMessage )
.GotoPrevWord
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
end with
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
```

```
Activewindow.Replace
ActiveWindow.Selection.Escape
End Sub
```

SearchAll-Methode

Sucht im gesamten Dokument nach einem bestimmten Text. War die Suche erfolgreich, wird der Wert True zurückgegeben, ansonsten False.

Ohne Übergabeparameter wird der Dialog SUCHEN & ERSETZEN aufgerufen.

Wurden vor Aufruf dieser Methode Suchattribute gesetzt, dann ist der Übergabeparameter optional. In diesem Fall wird der fehlende Parameter automatisch mit den aktuellen Einstellungen der SearchAttributes gesetzt.

Siehe auch: ReplaceAll, Search.

KLASSE:
BaseTextEditView

SYNTAX:
SearchAll(SearchString As String)

RÜCKGABE:
Boolean

PARAMETER:
[SearchString]: Ein Zeichenfolgeausdruck, der den Text aufnimmt, der gesucht werden soll. Wird dieser Parameter nicht übergeben, dann wird automatisch die Einstellung aus den SearchAttributes ausgelesen, wenn diese zuvor gesetzt wurden. Sonst wird der Suchdialog geöffnet.

BEISPIEL:

```
Sub ExampleReplaceAll
Const sWord_1 = "Beispielwort"
Const sWord_2 = "Geändertes Wort"
Dim iCount As Integer
Documents.Add( "Swriter4" )
For iCount=1 to 5
Selection.Insert sWord_1
Selection.InsertPara
next iCount
ActiveWindow.SearchAll sWord_1
For iCount=1 To Selection.SelectionCount
Selection.JumpToNextSelection
Next iCount
For iCount=1 To Selection.SelectionCount
Selection.JumpToPrevSelection
```

```
Next iCount
ActiveWindow.ReplaceAll sWord_1, sWord_2
End Sub
```

SearchAttributes-Eigenschaft

Das Objekt informiert, nach welchen Attributen gesucht werden soll. Dabei gibt es für jedes Attribut drei Zustände: nicht nach dem Attribut suchen, nach dem Attribut mit einem beliebigen Wert suchen, nach dem Attribut mit einem gesetzten Wert suchen. Für jedes Attribut gibt es eine Property, womit der Zustand erfragt und die ersten beiden Zustände gesetzt werden können. Durch das Setzen des Wertes für ein Attribut wird der dritte Zustand eingestellt. Ist ein Attribut gesetzt, kann sein Wert erfragt werden.

KLASSE:
BaseTextEditView

SYNTAX:
SearchAttributes

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleSearchAndReplace
Const sMessage = "a bFett c dFett e fFett g"
Documents.Add("sWriter4")
with  ActiveWindow.Selection
.Insert( sMessage )
.GotoPrevWord
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
.GotoPrevWord FALSE
.GotoPrevWord TRUE
.Font.Bold = soWeightBold
end with
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
```

```
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
End Sub
```

SearchCount-Methode

Sucht nach der Zeichenkette SearchString und liefert die Anzahl der gefundenen Positionen zurück.

KLASSE:
BaseTextEditView

SYNTAX:
SearchCount(SearchString As String)

RÜCKGABE:
Long

PARAMETER:
SearchString: Zeichenkette, nach der gesucht werden soll.

BEISPIEL:

```
Sub ExampleSearchCount
Const sWord = "Satz"
Const sMessage_1 = "Dies ist ein Satz"
Const sMessage_2 = "und nun kommt der zweite Satz"
Const sMessage_3 = "Anzahl des Wortes REM SatzREM  auf dem
Dokument : "
Documents.Add("sWriter4")
Selection.Insert( sMessage_1 )
Selection.InsertPara
Selection.Insert( sMessage_2 )
Print sMessage_3 & ActiveWindow.SearchCount( sWord )
End Sub
```

SearchSettings-Eigenschaft

Liefert die Suchoptionen als SearchSettings-Objekt zurück und ermöglicht somit einen Zugriff auf folgende Eigenschaften:

Backward: TRUE - rückwärts

InSelection: TRUE - nur in Selektionen

CaseSensitive: TRUE - Groß-/Kleinschreibung beachten

RegExp: TRUE- reguläre Ausdrücke

WordOnly: TRUE- ganze Wörter

Mode: nur für StarCalc (0 in Formeln;1 - in Werten; 2 - in Notizen)

Order: nur für StarCalc (TRUE: in Zeilenrichtung/FALSE in Spaltenrichtung)

Similarity: TRUE- Ähnlichkeitssuche

Relaxed: kombinierte Einstellungen für Ähnlichkeitssuche

Other: Anzahl der zu tauschenden Zeichen (Ähnlichkeitssuche)

Shorter: Anzahl der zu entfernenden Zeichen (Ähnlichkeitssuche)

Longer: Anzahl der hinzuzufügenden Zeichen (Ähnlichkeitssuche)

KLASSE:
WebSourceView, BaseTextEditView

SYNTAX:
SearchSettings

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleBackward
Documents.Add "sWriter4"
Selection.InsertPara
Selection.Insert "Test"
Selection.InsertPara
Selection.JumpToStartOfDoc
Selection.GotoEndOfDoc TRUE
ActiveWindow.SearchSettings.Backward = TRUE
ActiveWindow.SearchSettings.InSelection = TRUE
ActiveWindow.SearchSettings.CaseSensitive = TRUE
ActiveWindow.SearchSettings.WordOnly = TRUE
```

```
ActiveWindow.Search "Test"
Print Selection.Value
End Sub
```

Select-Methode

Selektiert die gesamte Tabelle, in der sich die Schreibmarke gegenwärtig befindet.

KLASSE:
BaseTextTable

SYNTAX:
Select()

BEISPIEL:

```
Sub ExampleSelect
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
Selection.Select
End Sub
```

SelectAll-Methode

Markiert das gesamte Dokument.

KLASSE:
TextDrawText, WebSourceView, BaseText

SYNTAX:
SelectAll()

BEISPIEL:

```
Sub ExampleSortDialog
Documents.Add("swriter4")
REM Daten eingeben, selektieren und Dialog starten
With ActiveWindow.Selection
.Insert("Text : 1")
.InsertPara
.Insert("Text : 2")
.InsertPara
.SelectALL
.SortDialog
End With
End Sub
```

SelectionCount-Eigenschaft

Ermittelt die Anzahl der Markierungen oder die Anzahl der selektierten Kontrollfelder im gesamten Dokument.

KLASSE:
BaseText, BaseTextDrawBase

SYNTAX:
SelectionCount

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleReplaceAll
Const sWord_1 = "Beispielwort"
Const sWord_2 = "Geändertes Wort"
Dim iCount As Integer
Documents.Add( "Swriter4" )
For iCount=1 to 5
Selection.Insert sWord_1
Selection.InsertPara
next iCount
ActiveWindow.SearchAll sWord_1
For iCount=1 To Selection.SelectionCount
Selection.JumpToNextSelection
Next iCount
For iCount=1 To Selection.SelectionCount
Selection.JumpToPrevSelection
Next iCount
ActiveWindow.ReplaceAll sWord_1, sWord_2
End Sub
```

SelectionMode-Eigenschaft

Bestimmt oder ermittelt den Selektionsmodus.

KLASSE:
BaseTextEditView

SYNTAX:
SelectionMode

RÜCKGABE:
Long

PARAMETER:
[= SelMode]: Long-Integer-Ausdruck, der den Selektionsmodus angibt:

0: Normaler Selektionsmodus. Nur eine Markierung ist möglich.

1: Erweiterter Selektionsmodus. Ermöglicht das Markieren ganzer Blöcke, von der aktuellen Position der Schreibmarke bis zu ihrer neuen Position.

2: Ergänzender Selektionsmodus. Mehrere voneinander unabhängige Markierungen sind möglich.

BEISPIEL:

```
Sub ExampleSelectionMode
Const sMessage = "Der Selektionsmodus auf dem akuellen Dokument
ist : "
Documents.Add("sWriter4")
Print sMessage & ActiveWindow.SelectionMode
End Sub
```

SelectText-Methode

Selektiert den aktuellen Absatz.

KLASSE:
Paragraph

SYNTAX:
SelectText()

BEISPIEL:

```
Sub ExampleSelectText
REM den lateinischen Blindtext verwenden
Const sBlindText = "BT"
Documents.Add("sWriter4")
With ActiveWindow.Selection
.ExpandGlossary( sBlindText )
Wait 500
.Paragraph.SelectText REM Absatz wird Selektiert
Wait 500
.Insert("Diesen Text wollte ich eigentlich an dieser Stelle
haben.")
End With
End Sub
```

SelectWord-Methode

Selektiert das Wort, über dem sich die Schreibmarke befindet.

KLASSE:
ReplaceAttributes, BaseText

SYNTAX:
SelectWord()

BEISPIEL:

```
Sub ExampleSelectWord
Const sText ="Das ist ein Satz mit nicht zu wenigen Wörtern,
die zufällig rot gefärbt werden."
Dim iCount As Integer
Documents.Add("swriter4")
With ActiveWindow.Selection
   .Insert( sText )
   For iCount = 1 to 20
   .ResetAttributes
   ActiveWindow.JumpToStartOfDoc
   .GoRight (Rnd()*Len(sText))
   .SelectWord
   .Font.Color=RGB(255,0,0)
   .Escape
   Wait 500
   .SelectWord
   Next iCount
   .Escape
End With
End Sub
```

SendAbstractToStarImpress-Methode

Ruft den Dialog AUTOABSTRACT ERZEUGEN auf, um ein AutoAbstract als Gliederung in ein neues Präsentationsdokument zu übernehmen.

KLASSE:
TextDocument

SYNTAX:
SendAbstractToStarImpress()

BEISPIEL:

```
Sub ExampleAutoOutline
Const sHeader1="Überschrift 1" REM Die Überschriften Vorlagen
werden verwendet.
Const sHeader2="Überschrift 2"
Const sHeader3="Überschrift 3"
REM zuerst ein kleines Dokument erstellen
Documents.Add("swriter4")
```

```
sWin = ActiveWindow.Name
With ActiveWindow.Selection
   .Insert("Wichtig")
   .StyleApply( sHeader1, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Wichtig")
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Wichtig")
   .InsertPara
   .Insert("Weniger Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
   .InsertPara
   .Insert("Unwichtig")
   .StyleApply( sHeader3, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Unwichtig")
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Unwichtig")
   .InsertPara
   .InsertPara
   .Insert("Etwas Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
End With
Wait 2000
ActiveDocument.SendOutlineToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendAbstractToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendOutlineToClipboard
Documents.Add("swriter4")
Selection.Insert("Gliederung in neues Dokument kopieren.")
Selection.InsertPara
ActiveWindow.Selection.Paste
end sub
```

SendFax-Methode

Druckt das aktuelle Dokument an das Standard-Fax-Gerät, daß unter Extras - Optionen - Textdokument - Drucken eingestellt wurde.

Klasse:
BaseTextEditView

Syntax:
SendFax()

Beispiel:

```
Sub ExampleDDE  REM Steuerung des Programm Winfax
Dim sFaxPath As String, sDDE As String , iCon As integer
Dim sWFaxStatus As String
Dim iTra As integer ,iStat As integer , sC As String
sC = CHR$(34)
sFaxPath = "E:\Winfax\"  REM  Pfad muß angepasst werden
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
If iTra = 0 OR iCon = 0 Then
Shell(sFaxPath + "Faxmng.exe")    REM   Faxprogramm starten
iCon = DDEInitiate("FAXMNG", "CONTROL")
iTra = DDEInitiate("FAXMNG", "TRANSMIT")
End If
sWFaxStatus = DDERequest(iTra, "Status")
sDDE = "recipient(" & sC & "040122345" & sC & ", " & sC & sC &
",  " & sC & sC & ", "
sDDE = sDDE & sC & "  " & sC & ", " & sC & "  " & sC & ", " &
sC & "  " & sC
sDDE = sDDE & ")"
iStat = DDEPoke( iTra, "Sendfax", sDDE)
ActiveWindow.SendFax
DDEExecute(iTra, "GoActive")
DDETerminate(iTra) REM Schließt den iTra-DDEKanal
DDETerminateAll
End Sub
```

SendOutlineToClipboard-Methode

Stellt die Gliederung des aktuellen Dokuments in die Zwischenablage.

Klasse:
TextDocument

Syntax:
SendOutlineToClipboard()

Beispiel:

```
Sub ExampleAutoOutline
Const sHeader1="Überschrift 1" REM Die Überschriften Vorlagen
werden verwendet.
Const sHeader2="Überschrift 2"
Const sHeader3="Überschrift 3"
REM zuerst ein kleines Dokument erstellen
Documents.Add("swriter4")
sWin = ActiveWindow.Name
With ActiveWindow.Selection
   .Insert("Wichtig")
   .StyleApply( sHeader1, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Wichtig")
```

```
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Wichtig")
   .InsertPara
   .Insert("Weniger Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
   .InsertPara
   .Insert("Unwichtig")
   .StyleApply( sHeader3, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Unwichtig")
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Unwichtig")
   .InsertPara
   .InsertPara
   .Insert("Etwas Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
End With
Wait 2000
ActiveDocument.SendOutlineToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendAbstractToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendOutlineToClipboard
Documents.Add("swriter4")
Selection.Insert("Gliederung in neues Dokument kopieren.")
Selection.InsertPara
ActiveWindow.Selection.Paste
end sub
```

SendOutlineToStarImpress-Methode

Erzeugt aus der Gliederung des aktuellen Dokuments ein neues Präsentationsdokument.

KLASSE:
TextDocument

SYNTAX:
SendOutlineToStarImpress()

BEISPIEL:

```
Sub ExampleAutoOutline
Const sHeader1="Überschrift 1" REM Die Überschriften Vorlagen
werden verwendet.
Const sHeader2="Überschrift 2"
Const sHeader3="Überschrift 3"
REM zuerst ein kleines Dokument erstellen
Documents.Add("swriter4")
```

```
sWin = ActiveWindow.Name
With ActiveWindow.Selection
   .Insert("Wichtig")
   .StyleApply( sHeader1, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Wichtig")
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Wichtig")
   .InsertPara
   .Insert("Weniger Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
   .InsertPara
   .Insert("Unwichtig")
   .StyleApply( sHeader3, 2 )
   .InsertPara
   .Insert("Das ist der erste Absatz von Kapitel Unwichtig")
   .InsertPara
   .Insert("Das ist der zweite Absatz von Kapitel Unwichtig")
   .InsertPara
   .InsertPara
   .Insert("Etwas Wichtig")
   .StyleApply( sHeader2, 2 )
   .InsertPara
End With
Wait 2000
ActiveDocument.SendOutlineToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendAbstractToStarImpress
Wait 3000
Windows(sWin).Activate
ActiveDocument.SendOutlineToClipboard
Documents.Add("swriter4")
Selection.Insert("Gliederung in neues Dokument kopieren.")
Selection.InsertPara
ActiveWindow.Selection.Paste
end sub
```

SendToBack-Methode

Ordnet das selektierte Objekt hinter allen anderen Objekten im Dokument an.

KLASSE:
BaseTextDrawBase, BaseTextFrame

SYNTAX:
SendToBack()

Beispiel:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
   Selection.Insert(sText)
   if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALS
Selection.Printable = TRUE
Selection.SendToBack
Selection.BringToFront
end sub
```

SetActGlossaryGroup-Methode

Setzt den aktuellen AutoText-Bereich.

Siehe auch: ExpandGlossary, InsertGlossary, NewGlossary.

Klasse:
BaseText

Syntax:
SetActGlossaryGroup(Group As String)

Parameter:
Group: Ein Zeichenfolgeausdruck, der den Bereich angibt, dem der Auto-Text zugeordnet werden soll.

Beispiel:

```
Sub ExampleGlossary
Const sGroupName = "Gruppe"
Const sGlossaryName = "Name"
Const sShortName = "TT"
Const sMessage_1 = "Dieser Satz wird ein Autotext"
Const sMessage_2 = "Es sind folgende Autotexte im System defi-
niert"
Const sMessage_3 = "Gruppe"
Const sMessage_4 = "Name"
Const sMessage_5 = "Kürzel"
Const sMessage_6 = "InsertGlossary liefert folgendes : "
Const sMessage_7 = "ExpandGlossary liefert folgendes : "
```

```
Dim iCount As Integer, iBlockCount As Integer
Dim sBlockGroup As String, sBlockName as String
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.GotoStartOfLine( TRUE )
Selection.CreateGlossaryGroup( sGroupName )
Selection.NewGlossary( sGroupName, sGlossaryName, sShortName )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.Insert sMessage_6
Selection.InsertGlossary( sGroupName, sShortName )
Selection.InsertPara
Selection.SetActGlossaryGroup( sGroupName )
Selection.Insert sMessage_7
Selection.ExpandGlossary( sShortName )
Selection.InsertPara
Selection.Insert sMessage_2
Selection.InsertTable( 3, 3 )
Selection.Insert sMessage_3
Selection.JumpToNextCell
Selection.Insert sMessage_4
Selection.JumpToNextCell
Selection.Insert sMessage_5
Selection.JumpToNextCell
For iCount = 0 To Selection.GetGlossaryGroupCount - 1
sBlockGroup = Selection.GetGlossaryGroupName( iCount )
For iBlockCount = 0  To Selection.GetGlossaryBlockCount(
sBlockGroup ) - 1
Selection.Insert sBlockGroup
Selection.JumpToNextCell
sBlockName = Selection.GetGlossaryBlockName( sBlockGroup,
iBlockCount )
Selection.Insert sBlockName
Selection.JumpToNextCell
Selection.Insert Selection.GetGlossaryShortName( sBlockGroup,
sBlockName )
Selection.JumpToNextCell
Next iBlockCount
Next iCount
Selection.DeleteGlossary( sGroupName, sShortName )
Selection.DeleteGlossaryGroup( sGroupName )
End Sub
```

SetActions-Methode

Setzt eine bestimmte Anzahl von Aktionen.

Siehe auch: ActionStart, ResetAllActions.

KLASSE:
BaseTextSelection

SYNTAX:
SetActions(Depth As Long)

PARAMETER:
Depth: Anzahl der Aktionen, die gesetzt werden sollen.

BEISPIEL:

```
Sub ExampleSetActions
Const sHeader = "SetAction setzt drei Action-Ebenen auf
einmal."
Const sMessage_1 = "Es sind jetzt nur noch zwei Action-Ebenen
gesetzt."
Const sMessage_2 = "Es ist jetzt nur noch eine Action-Ebenen
gesetzt."
Const sMessage_3 = "Es ist keine Action-Ebene gesetzt."
Documents.Add( "sWriter4" )
Selection.Insert sHeader
Selection.InsertPara
Selection.SetActions( 3 )
Wait 100
Selection.ActionEnd
Selection.Insert sMessage_1
Selection.InsertPara
Selection.ActionEnd
Selection.Insert sMessage_2
Selection.InsertPara
Selection.ActionEnd
Selection.Insert sMessage_3
End Sub
```

SetAlignment-Methode

Setzt das Attribut Ausrichtung für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsAlignment, Alignment.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetAlignment(Alignment)

PARAMETER:
Alignment: Integer-Ausdruck, der die Ausrichtungsoption bestimmt. Folgende Werte sind möglich:

0: Linksbündig

1: Rechtsbündig

2: Blocksatz

3: Zentriert

Beispiel:

```
Sub ExampleSetAlignment
Const sMessage_1 = "Dies ist ein linksbündig formatierter
Absatz"
Const sMessage_2 = "Dies ist ein rechtsbündig formatierter
Absatz"
Const sMessage_3 = "Es wird der erste rechtsbündige Absatz
gesucht"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Alignment = 1
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsAlignment = 2
ActiveWindow.SearchAttributes.SetAlignment(1)
If ActiveWindow.SearchAttributes.Alignment = 1 Then
Print sMessage_3
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetAnchorToChar-Methode

Mit dieser Methode wird die Verankerung geändert (Zeichen). Textrahmen, Grafiken, OLE-Objekte, Controls und Zeichenobjekte werden auf Verankerung "Am Zeichen" gesetzt.

Klasse:
BaseTextSelection

Syntax:
SetAnchorToChar()

Beispiel:

```
Sub ExampleSetObjectAnchor
Documents.Add( "swriter4" )
Activewindow.InsertFrame( 1, "3cm", "3cm", "5cm", "2cm" )
Wait 100
Activewindow.Selection.SetAnchorToPara
Wait 100
Activewindow.Selection.SetAnchorToChar
```

```
Wait 100
Activewindow.Selection.SetAnchorToPage
End Sub
```

SetAnchorToPage-Methode

Mit dieser Methode wird die Verankerung geändert (Seite). Textrahmen, Grafiken, OLE-Objekte, Controls und Zeichenobjekte werden auf Verankerung "An der Seite" gesetzt.

KLASSE:
TextSelection

SYNTAX:
SetAnchorToPage()

BEISPIEL:

```
Sub ExampleSetObjectAnchor
Documents.Add( "swriter4" )
Activewindow.InsertFrame( 1, "3cm", "3cm", "5cm", "2cm" )
Wait 100
Activewindow.Selection.SetAnchorToPara
Wait 100
Activewindow.Selection.SetAnchorToChar
Wait 100
Activewindow.Selection.SetAnchorToPage
End Sub
```

SetAnchorToPara-Methode

Mit dieser Methode wird die Verankerung geändert (Absatz). Textrahmen, Grafiken, OLE-Objekte, Controls und Zeichenobjekte werden auf Verankerung "Am Absatz" gesetzt.

KLASSE:
BaseTextSelection

SYNTAX:
SetAnchorToPara()

BEISPIEL:

```
Sub ExampleSetObjectAnchor
Documents.Add( "swriter4" )
Activewindow.InsertFrame( 1, "3cm", "3cm", "5cm", "2cm" )
Wait 100
Activewindow.Selection.SetAnchorToPara
Wait 100
Activewindow.Selection.SetAnchorToChar
```

```
Wait 100
Activewindow.Selection.SetAnchorToPage
End Sub
```

SetBold-Methode

Setzt das Attribut Fontstärke für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsBold, Bold.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetBold(Bold)

PARAMETER:
Bold: Kann die Werte 1-10, TRUE oder FALSE enthalten. Wird TRUE angegeben, wird nach der Intensität 8 gesucht. Der Standardwert für Bold ist 8.

BEISPIEL:

```
Sub ExampleSearchAndReplaceBold
Documents.Add("sWriter4")
REM erstmal eine fette Zeile erstellen
With ActiveWindow.Selection
   .Insert("a bBold c dBold e fBold g hBold i")
   .GotoPrevWord
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
End With
ActiveWindow.SearchAttributes.Reset
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
```

```
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search ""
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
ActiveWindow.ReplaceAttributes.Reset
REM ab jetzt werden die gefundenen Stellen gelöscht
ActiveWindow.Replace
ActiveWindow.Replace
End Sub
```

SetCaseMap-Methode

Definiert das Attribut Schrifteffekt für das Suchen und Ersetzen von Attributen. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der angegebene Schrifteffekt gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: CaseMap, IsCaseMap.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetCaseMap(CaseMap)

PARAMETER:
CaseMap: Integer-Ausdruck, der den Schrifteffekt festlegt. Folgende Werte sind möglich:

0 - Keine Auszeichnung

1 - Versalien

2 - Gemeine

3 - Titelschrift

4 - Kapitälchen

BEISPIEL:

```
Sub ExampleSetCaseMap
Const sMessage_1 = "Es wird ein, mit Kapitälchen formatierter
Text gesucht"
Const sMessage_2 = "Dieser Absatz ist als Kapitälchen forma-
tiert"
```

```
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.CaseMap = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsAlignment = 2
ActiveWindow.SearchAttributes.SetCaseMap(4)
If ActiveWindow.SearchAttributes.CaseMap = 4 Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetColumnWidth-Methode

Mit dieser Methode kann die Spaltenbreite einer Tabelle gesetzt werden. Ohne Übergabeparameter wird der Dialog Spaltenbreite aufgerufen.

Klasse:
BaseTextTable

Syntax:
SetColumnWidth(Col As Long, Width As Long)

Parameter:
[Col]: Index der Tabellenspalte, die verändert werden soll. Dabei hat die am weitesten links stehende Spalte den Index Null.

[Width]: Gibt die gewünschte Tabellenbreite an. Der Wert kann auch mit Angabe einer Maßeinheit als Zeichenfolge übergeben werden.

Beispiel:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

SetContour-Methode

Setzt das Zeichenattribut Kontur zum Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird mit dem Übergabeparameter True nach dem Attribut Kontur gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsContour, Contour.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetContour(Contour As Boolean)

PARAMETER:
Contour: Boolescher Ausdruck, der angibt, ob nach dem Attribut Kontur gesucht wird oder ob das Attribut beim Ersetzen angewendet werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSetContour
Const sMessage_1 = "Es wird ein, mit Konturschrift formatierter
Text gesucht"
Const sMessage_2 = "Bei diesem Absatz ist der Text als Kontur-
schrift formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.OutlineFont = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsContour = 2
ActiveWindow.SearchAttributes.SetContour( TRUE )
If ActiveWindow.SearchAttributes.Contour = TRUE Then
Print sMessage_1
End if
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetCurField-Methode

Weist dem selektierten Feldbefehl die angegebenen Parameter zu.

KLASSE:
BaseText

SYNTAX:
SetCurField(Format As Long, Par1 As String, Par2 As String)

RÜCKGABE:
Boolean

PARAMETER:
Format: Long-Integer-Ausdruck, der die Darstellungsform des selektierten Feldes angibt. Format kann eine der in dem Listenfeld FORMAT des Feldbefehl-Dialogs aufgeführten Darstellungsformen in der Reihenfolge ihrer Anordnung zugewiesen werden, wobei der Wert 0 den ersten Eintrag kennzeichnet.

Par1: Zeichenfolgeausdruck, der dem selektierten Feld eine Bezeichnung zuweist, sofern dieses benutzerdefiniert ist. Ist das selektierte Feld ein Funktionsfeld, übergibt Par1 die Funktionsbedingung. Par1 kann nicht bei allen Feldern gesetzt werden.

Par2: Zeichenfolgeausdruck, der dem selektierten Feld den vom Feldtyp abhängigen Feldinhalt oder Feldwert zuweist.

BEISPIEL:

```
Sub ExampleCurFieldFormat
Dim iFmt As Integer
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert("Nächster Start um: ")
ActiveWindow.Selection.InsertField( 1, 1, "", "" , 0 )
ActiveWindow.Selection.GoLeft 1
ActiveWindow.Selection.InsertPara
Selection.Escape
Activewindow.Selection.JumpToStartOfDoc
ActiveWindow.NextFieldByType( 1 , "" )
ActiveWindow.Selection.SetCurField(1,"","") REM setze auf 24
Stundenformat
iFmt = ActiveWindow.Selection.GetFieldFmt()
Activewindow.Selection.JumpToEndOfDoc
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert("Feld-Format-Identifikation ist:
" & iFmt)
End Sub
```

SetDefaultBorder-Methode

Setzt eine Standardlinie der Farbe Schwarz und der Linienbreite 0,5 pt an markierte Zellen oder Zellbereiche einer Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
SetDefaultBorder(Position As Long)

PARAMETER:
Position: Long-Integer-Ausdruck, der die Position der Linie im selektierten Tabellenbereich angibt:

1: Linienposition oberhalb der selektierten Zelle(n).

2: Linienposition unterhalb der selektierten Zelle(n).

4: Linienposition links neben der markierten Zelle oder dem markierten Zellbereich.

8: Linienposition rechts neben der markierten Zelle oder dem markierten Zellbereich.

16: Linienposition horizontal zwischen den markierten Zellen.

32: Linienposition vertikal zwischen den markierten Zellen.

Um eine Kombination der verschiedenen Linienpositionen zu erreichen, werden die jeweiligen Werte von Position addiert.

BEISPIEL:

```
Sub ExampleTableBorder
Documents.Add("sWriter4")
Selection.InsertTable(4, 4)
Selection.GotoStart
Selection.EntireRow
Selection.SetDefaultBorder(0) REM Rahmen aus
REM Umrandung: Oben+Unten+Links+Rechts+Horiz.+Vertikal
Selection.SetDefaultBorder( 1 + 2 + 4 + 8 + 16 + 32 )
Selection.LineColor = RGB(255,0,0) REM Den Rahmen Rot einfärben
Selection.EntireColumn
Selection.GoLeft 1, TRUE
Selection.SetDefaultBorder(0) REM Border aus
Selection.Border.TopLine.Style = 3
Selection.Border.TopLine.Color = RGB(0, 255, 0) REM Grün
Selection.Border.LeftLine.Style = 3
Selection.Border.LeftLine.Color = RGB(255, 0, 0)  REM Rot
Selection.Border.BottomLine.Style = 4
Selection.Border.BottomLine.Color = RGB(0, 0, 255) REM Blau
Selection.Border.RightLine.Style = 4
```

```
Selection.Border.RightLine.Color = RGB(255, 0, 255) REM Lila
Selection.Border.VerticalLine.Style = 5
Selection.Border.VerticalLine.Color = RGB(128, 128, 255) REM
Blau
Selection.Border.HorizontalLine.Style = 6
Selection.Border.HorizontalLine.Color = RGB(0, 0, 0) REM
Schwarz
Selection.Border.Distance = "0.5cm"
End Sub
```

SetEscapement-Methode

Setzt das Attribut Zeichenposition für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt, werden die angegebenen Werte für die prozentuale Hoch- oder Tiefstellung und die relative Schriftgröße gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt werden die übergebenen Werte beim Ersetzen angewendet.

Siehe auch: IsEscapement , Escapement, RelativeSize.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetEscapement(Escapement As Integer, RelativeSize As Integer)

PARAMETER:
Escapement: Integer-Ausdruck, der die Hoch- oder Tiefstellung des Textes prozentual zur Zeilenhöhe angibt.

RelativeSize: Gibt die relative Schriftgröße der hoch- oder tiefgestellten Zeichen an.

BEISPIEL:

```
Sub ExampleSetEscapement
Const sMessage_1 = "Es wird ein, als hochgestellt formatierter
Text gesucht"
Const sMessage_2 = "Bei diesem Absatz ist der Text hochgestellt
formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsEscapement = 2
ActiveWindow.SearchAttributes.SetEscapement( 101, 62 )
If ActiveWindow.SearchAttributes.Escapement = 101 Then
Print sMessage_1
End if
```

```
Activewindow.Search
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetFixDateTimeFields-Methode

Die Methode setzt alle Datum- und Zeitfelder mit fixem Format auf den aktuellen Wert.

KLASSE:
BaseTextSelection

SYNTAX:
SetFixDateTimeFields()

BEISPIEL:

```
Sub ExampleFixToVarTime
Documents.Add( "sWriter4" )
ActiveWindow.Selection.InsertTimeField
ActiveWindow.Selection.SetFixDateTimeFields
End Sub
```

SetFont-Methode

Setzt das Attribut Schrift für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der angegebene Zeichensatz gesucht, bei einem ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsFont, StyleName, Pitch, CharSet, Family, FamilyName.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetFont(StyleName As String, Pitch, CharSet, Family, FamilyName As String)

PARAMETER:
StyleName: Zeichenfolgeausdruck, der den Schriftstil bestimmt. StyleName können die folgenden Werte zugewiesen werden:

Leerstring - Voreinstellung. Der Text wird normal dargestellt.

Bold - Die Schrift wird fett dargestellt.

Italic - Die Schrift wird kursiv dargestellt.

Pitch: Integer-Ausdruck, der angibt, ob die Schriftart proportional ist. Folgende Werte sind möglich:

0 - Unbekannte Laufweite

1 - Nichtproportionale Schriftart

2 - Proportionale Schriftart

Dieser Parameter ist optional.

CharSet: Integer-Ausdruck, der den Zeichensatz bestimmt, der der Schrift zugewiesen werden kann:

0 - Voreinstellung, unbekannter Zeichensatz

1 - ANSI-Zeichensatz

2 - Macintosh-Zeichensatz

3 - IBM PC Zeichensatz 437

4 - IBM PC Zeichensatz 850

5 - IBM PC Zeichensatz 860

6 - IBM PC Zeichensatz 861

7 - IBM PC Zeichensatz 863

8 - IBM PC Zeichensatz 865

9 - System-Zeichensatz

10 - Symbol-Zeichensatz

Family: Integer-Ausdruck, der die Schriftfamilie kennzeichnet, der die Schriftart zugewiesen werden kann:

0 - Voreinstellung, unbekannte Schriftartgruppe

1 - Decorative

2 - Modern

3 - Roman

4 - Script

5 - Swiss

6 - System

Wird eine Schriftart gewählt, die nicht vorhanden ist, dann wird eine adäquate Schriftart aus der angegebenen Schriftfamilie verwendet.

FamilyName: Zeichenfolge, die den Namen der Schriftfamilie angibt, z. B. "Arial".

Beispiel:

```
Sub ExampleSetFont
Const sMessage_1 = "Es wird ein Text mit der Schrift REM Helve-
ticaREM  gesucht"
Const sMessage_2 = "Dieser Absatz ist mit der Schrift REM
HelveticaREM  formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.FamilyName = "Helvetica"
Selection.Font.Size = 14
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.IsFontHeight = 2
ActiveWindow.SearchAttributes.SetFont("", 14,,,"Helvetica")
ActiveWindow.SearchAttributes.SetFontHeight( 14, 100 )
If ActiveWindow.SearchAttributes.FamilyName = "Helvetica" Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetFontColor-Methode

Setzt das Attribut Fontfarbe für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird die angegebene Farbe gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird sie beim Ersetzen angewendet.

Siehe auch: IsFontColor, FontColor.

Klasse:
ReplaceAttributes, SearchAttributes

Syntax:
SetFontColor(FontColor As Long)

Parameter:
FontColor: Long-Integer-Ausdruck, der den Farbcode der Zeichenfarbe angibt.

Beispiel:

```
Sub ExampleSetFontColor
Const sMessage_1 = "Es wird ein Text, der mit der Farbe REM
RotREM  formatiert ist, gesucht"
```

```
Const sMessage_2 = "Bei diesem Absatz hat die Schrift die Farbe
REM RotREM "
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Color = RGB( 255, 0, 0 )
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.SetFontColor( RGB( 255, 0, 0 ) )
If ActiveWindow.SearchAttributes.FontColor = RGB( 255, 0, 0 )
Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetFontHeight-Methode

Setzt das Attribut Schriftgröße für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt, wird der übergebene Wert gesucht, bei einem ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsFontHeight, Size, Prop.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetFontHeight(Size As Long, Prop As Integer)

PARAMETER:
Size: Long-Integer-Ausdruck, der die Schriftgröße in Punkt angibt.

[Prop]: Integer-Ausdruck, der die zur Schriftgröße der verknüpften Absatzvorlage relative Schriftgröße in Prozent angibt. Dieser Parameter ist optional. Es genügt den ersten Übergabeparameter anzugeben, um die Schriftgröße festzulegen.

BEISPIEL:

```
Sub ExampleSetFont
Const sMessage_1 = "Es wird ein Text mit der Schrift REM Helve-
ticaREM  gesucht"
Const sMessage_2 = "Dieser Absatz ist mit der Schrift REM
HelveticaREM  formatiert"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.FamilyName = "Helvetica"
```

```
Selection.Font.Size = 14
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsFont = 2
ActiveWindow.SearchAttributes.IsFontHeight = 2
ActiveWindow.SearchAttributes.SetFont("", 14,,,"Helvetica")
ActiveWindow.SearchAttributes.SetFontHeight( 14, 100 )
If ActiveWindow.SearchAttributes.FamilyName = "Helvetica" Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetFramestyleMacro-Methode

Setzt an einer Rahmenvorlage ein Makro, das bei einem definierten Ereignis ausgeführt wird, oder löscht ein bereits zugewiesenes Makro.

KLASSE:
BaseTextDocument

SYNTAX:
SetFramestyleMacro(StyleName As String, Event As Long, SubName As String, LibName As String)

PARAMETER:
StyleName: Name der Rahmenvorlage

Event: Nummer des Ereignisses, das eintritt. Folgende Ereignisse werden unterstützt:

Werte:

1 - Maus über Objekt

2 - Hyperlink auslösen

3 - Maus verläßt Objekt

4 - Objekt anklicken

5 - Eingabe von Alphazeichen

6 - Eingabe von nicht Alphazeichen

7 - Rahmengröße verändern

8 - Rahmen verschieben

9 - Grafik laden erfolgreich

10 - Grafik laden abgebrochen

11 - Grafik laden fehlerhaft

[SubName]: Name der SUB oder FUNCTION, die gestartet werden soll. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird ein an der Rahmenvorlage bereits gesetztes Makro gelöscht.

[LibName]: Name der StarBasic-Bibliothek, die die mit dem Parameter SubName angegebene SUB oder FUNCTION enthält. Dieser Parameter ist nicht erforderlich, wenn SubName nicht übergeben wurde.

Beispiel:

```
Sub ExampleGetFrameStyleMacroLib
Const sStyleName = "NeueVorlage"
Const sMessage_1 = "Die Bibliothek des verwendeten Makros
lautet : "
Const sMessage_2 = "Die Prozedure des verwendeten Makros lautet
: "
Const sMessage_3 = "Wenn die Maus den Frame verläßt, wird ein
Makro ausgeführt"
Dim sFrameName As String
Documents.Add( "sWriter4" )
Activedocument.NewStyle( sStyleName, 4)
ActiveDocument.SetFrameStyleMacro( sStyleName, 3, "StartChange-
sUserfields(Gimmicks.Userfields)", "StarOffice" )
Print sMessage_1 & ActiveDocument.GetFrameStyleMacroLib( sSty-
leName, 3 )
Print sMessage_2 & ActiveDocument.GetFrameStyleMacroSub( sSty-
leName, 3 )
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Insert sMessage_3
Selection.StyleApply( sStyleName, 4 )
Selection.AlignLeft = "2cm"
Selection.AlignTop = "2cm"
End Sub
```

SetFrameStyleScript-Methode

Setzt an einer Rahmenvorlage zu einem bestimmten Ereignis ein JavaScript oder löscht ein bereits gesetztes JavaScript.

Klasse:
BaseTextDocument

Syntax:
SetFrameStyleScript(StyleName As String, Event As Long, Script As String)

Parameter:
StyleName: Name der Rahmenvorlage

Event: Nummer des eintretenden Ereignisses. Folgende Ereignisse werden unterstützt:

Werte:

1 - Maus über Objekt

2 - Hyperlink auslösen

3 - Maus verläßt Objekt

4 - Objekt anklicken

5 - Eingabe von Alphazeichen

6 - Eingabe von nicht Alphazeichen

7 - Rahmengröße verändern

8 - Rahmen verschieben

9 - Grafik laden erfolgreich

10 - Grafik laden abgebrochen

11 - Grafik laden fehlerhaft

[Script]: Zeichenfolge, die das JavaScript aufnimmt. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird ein zuvor an der Rahmenvorlage gesetztes Script gelöscht.

Beispiel:

```
Sub ExampleSetFramestyleScript
Const sStyleName = "NeueVorlage"
Const sScriptText = "Dies ist ein Javascript"
Const sMessage_1 = "Das JavaScript für dieses Ereignis lautet :
"
Const sMessage_2 = "Wenn die Maus den Frame verläßt, wird ein
Javascript ausgeführt"
Dim sFrameName As String
Documents.Add( "sWriter4" )
Activedocument.NewStyle( sStyleName, 4)
ActiveDocument.SetFramestyleScript( sStyleName, 3, "alert(""" &
sScriptText & """)" )
Print sMessage_1 & ActiveDocument.GetFramestyleScript( sSty-
leName, 3 )
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Insert sMessage_2
Selection.StyleApply( sStyleName, 4 )
Selection.AlignLeft = "2cm"
Selection.AlignTop = "2cm"
End Sub
```

SetHandler-Methode

Weist einem Kontrollfeld ein Makro zu, das bei einem bestimmten Ereignis ausgeführt werden soll.

Diese Methode gilt nur für Kontrollfelder, die sich in einem Dokument befinden. Kontrollfelder eines Dialoges sind mit SetHandler nicht ansprechbar.

KLASSE:
BaseTextDrawControl

SYNTAX:
SetHandler(EventId As Long, Handler As String)

PARAMETER:
EventId: Long-Integer-Ausdruck, der das Ereignis bestimmt, dem das Makro zugewiesen werden soll. Folgende Werte sind möglich (Bedeutung der Events):

2 - OnChange

3 - OnClick

4 - OnSelect

6 - OnGotFocus

7 - OnLostFocus

14 - OnSpinUp

15 - OnSpinDown

16 - OnScroll

Handler: Makroname, dabei ist das Format entweder

Macro(Bibliothek.Modul) oder Bibliothek.Modul.Macro

BEISPIEL:

```
Sub ExampleSetHandler
Const sFileName = "\Test.sdw"
Const sMessage = "An dem Commandbutton hängt folgendes Makro :
"
Dim sPath As String
Documents.Add( "sWriter4" )
sPath = Application.PathSettings.Work & sFileName
REM Einen Commandbutton auf dem Dokument erstellen
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
Activewindow.ControlObject.Caption = "Makro ausführen"
```

```
Selection.SetHandler( 3, "StartChangesUserfields(Gimmicks.User-
fields)" )
Print sMessage & Selection.GetHandler( 3 )
Selection.Escape
ActiveDocument.SaveAs sPath
End Sub
```

SetHyperlink-Methode

Weist der aktuellen Selektion einen Hyperlink zu.

KLASSE:
BaseText

SYNTAX:
SetHyperlink(LinkName As String, LinkURL As String, LinkTarget As String, LinkType As Long)

PARAMETER:
LinkName: Name des Hyperlinks

LinkURL: Ziel-URL

LinkTarget: Target-Frame

LinkType: Gibt an, ob der Hyperlink als Text oder als Button eingefügt werden soll:

1 - als Text

2 - als Button

BEISPIEL:

```
Sub ExampleJumpToURL
Const sMessage = "Der aktuelle Hyperlink hat folgende URL : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.SetHyperlink( "Adaptec", "http://www.Adaptec.com/",
"", 1 )
Selection.InsertPara
Selection.SetHyperlink( "IBM", "http://www.IBM.com/", "", 1 )
Selection.InsertPara
Selection.SetHyperlink( "Compaq", "http://www.compaq.com/", "",
1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
For iCount=1 to ActiveWindow.URLFieldCount
ActiveWindow.JumpToURL(iCount)
Print sMessage & Selection.LinkURL
next iCount
End Sub
```

SetHyperlink-Methode

Weist der aktuellen Selektion einen Hyperlink zu.

KLASSE:
TextDrawText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
SetHyperlink(LinkName As String, LinkURL As String, LinkTarget As String, LinkType As Long)

PARAMETER:
Name: Name des Hyperlinks

URL: Ziel-URL

Target: Target-Frame

Type: Gibt an, ob der Hyperlink als Text oder als Button eingefügt werden soll:

1 - als Text

2 - als Button

BEISPIEL:

```
Sub ExampleJumpToURL
Const sMessage = "Der aktuelle Hyperlink hat folgende URL : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.SetHyperlink( "Adaptec", "http://www.Adaptec.com/",
"", 1 )
Selection.InsertPara
Selection.SetHyperlink( "IBM", "http://www.IBM.com/", "", 1 )
Selection.InsertPara
Selection.SetHyperlink( "Compaq", "http://www.compaq.com/", "",
1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
For iCount=1 to ActiveWindow.URLFieldCount
ActiveWindow.JumpToURL(iCount)
Print sMessage & Selection.LinkURL
next iCount
End Sub
```

SetHyperlinkMacro-Methode

Weist einem Hyperlink ein Makro zu, das bei einem definierten Ereignis ausgelöst wird. Steht der Cursor nicht innerhalb des Hyperlinks wird keine Zuordnung vorgenommen. Hier sollte vorher mit der LinkType-Eigenschaft geprüft werden, ob an der aktuellen Cursorposition ein Hyperlink vorhanden ist.

KLASSE:
TextFont

SYNTAX:
SetHyperlinkMacro(Event As Long, SubName As String, LibName As String)

PARAMETER:
Event: Nummer des Ereignisses, das ausgelöst werden soll. Folgende Ereignisse werden unterstützt:

Ereignis	Eventnummer
Maus über Objekt	1
Hyperlink auslösen	2
Maus verläßt Objekt	3

[SubName]: Name der SUB oder FUNCTION, die gestartet werden soll. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird ein bereits gesetztes Makro gelöscht.

[LibName]: Name der StarBasic-Bibliothek, die die mit dem Parameter SubName angegebene SUB oder FUNCTION enthält. Dieser Parameter ist nicht erforderlich, wenn SubName nicht übergeben wurde.

BEISPIEL:

```
Sub ExampleHyperLinkMacro
Dim sProcName As String
Dim sLibName As String
Documents.Add "sWriter4"

Selection.SetHyperlink("Stardivision", "http://www.Stardivi-
sion.de", "_blank", 1)
Selection.GotoStartOfLine FALSE

Selection.Font.SetHyperlinkMacro(2, "Gimmicks.Userfields.Start-
ChangesUserfields", "Gimmicks")
sProcName = Selection.Font.GetHyperlinkMacroSub(2)
sLibName = Selection.Font.GetHyperlinkMacroLib(2)
Print "Folgende Prozedure ist mit dem Hyperlink verknüpft : " &
sProcName
```

```
Print "Folgende Bibliothek ist mit dem Hyperlink verknüpft : "
& sLibName
End Sub
```

SetHyperlinkScript-Methode

Weist einem Hyperlink ein Script zu, das bei einem bestimmten Ereignis ausgelöst wird, oder hebt die Zuordnung zu einem Script wieder auf.

KLASSE:
TextFont

SYNTAX:
SetHyperlinkScript(Event As Long, Script As String)

PARAMETER:
Event: Nummer des Ereignisses, das ausgelöst werden soll. Folgende Ereignisse werden unterstü

Ereignis	Eventnummer
Maus über Objekt	1
Hyperlink auslösen	2
Maus verläßt Objekt	3

[Script]: Zeichenfolge, die das JavaScript aufnimmt. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird ein zuvor gesetztes Script gelöscht.

BEISPIEL:

```
Sub ExampleHyperLinkScript
Dim sScript As String
Documents.Add "sWriter4"
Selection.SetHyperlink("Stardivision", "http://www.Stardivi-
sion.de", "_blank", 1)
Selection.GotoStartOfLine FALSE
Selection.Font.SetHyperlinkScript(1, "alert(REM JavaScrip-
tREM )")
sScript = Selection.Font.GetHyperlinkScript(1)
Print "Folgendes JavaScript ist mit dem Hyperlink verknüpft : "
& sScript
End Sub
```

SetHyphenZone-Methode

Setzt das Attribut Silbentrennung für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt, werden Absätze mit der angegebenen Trennregel gesucht, bei einem ReplaceAttributes-Objekt wird die übergebene Regel beim Ersetzen angewendet.

Siehe auch: IsHyphenZone, Hyphen, PageEnd, MinLead, MinTrail, MaxHyphens.

KLASSE:
SearchAttributes

SYNTAX:
SetHyphenZone(Hyphen As Boolean, PageEnd As Boolean, MinLead As Integer, MinTrail As Integer, MaxHyphens As Integer)

PARAMETER:
[Hyphen]: Boolescher Ausdruck, der angibt, ob die automatische Silbentrennung ein- (True) oder ausgeschaltet (False) ist. Dieser Parameter ist optional. Wird er nicht übergeben, ist die Silbentrennung standardmäßig ausgeschaltet (Hyphen = False).

[PageEnd]: Boolescher Ausdruck, der angibt, ob die Silbentrennung eines Wortes über zwei Seiten erfolgen darf (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht übergeben, erfolgt die Silbentrennung über zwei Seiten (PageEnd = True).

[MinLead]: Gibt die minimale Anzahl der Zeichen am Zeilenende an. Wird dieser Wert nicht übergeben, wird für das Suchen und Ersetzen der Wert MinLead = 0 genommen.

[MinTrail]: Gibt die minimale Anzahl der Zeichen am Zeilenanfang an. Wird dieser Wert nicht übergeben, wird für das Suchen und Ersetzen der Wert MinTrail = 0 genommen.

[MaxHyphens]: Gibt die maximale Anzahl aufeinanderfolgender Silbentrennungen an. Wird dieser Wert nicht übergeben, wird ein Standardwert von 255 genommen.

BEISPIEL:

```
Sub ExampleSetHyphenZone
Const sMessage_1 = "Es wird ein Absatz mit eingeschalteter
Silbentrennung gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die automtische
Silbentrennung aktiv"
Documents.Add( "sWriter4" )
```

```
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.HyphenZone.Hyphen = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsHyphenZone = 2
ActiveWindow.SearchAttributes.SetHyphenZone( TRUE, TRUE, 2, 2,
0 )
If ActiveWindow.SearchAttributes.Hyphen = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetItalic-Methode

Setzt das Attribut Kursiv für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsItalic, Italic.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetItalic(Italic)

PARAMETER:
Italic: Integer-Ausdruck, der die Neigung des Zeichens bestimmt. (True und False sind ebenfalls zugelassen.) Folgende Werte sind möglich:

0: Der Text wird normal dargestellt (False).

1 - 2: Der Text wird kursiv dargestellt (True). Der Grad der Kursiv-Darstellung erhöht sich mit zunehmendem Wert. Der Wert 2 gibt die Standardeinstellung des Attributs Kursiv wieder.

Nicht alle Fonts unterstützen verschiedene Stufen des Italic-Attributs. Der Standardwert 2 ist jedoch fast immer verfügbar.

BEISPIEL:

```
Sub ExampleSearchAndReplaceBold
Documents.Add("sWriter4")
REM erstmal eine fette Zeile erstellen
With ActiveWindow.Selection
   .Insert("a bBold c dBold e fBold g hBold i")
   .GotoPrevWord
```

```
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
   .GotoPrevWord False
   .GotoPrevWord TRUE
   .Font.Bold = soWeightBold
End With
ActiveWindow.SearchAttributes.Reset
ActiveWindow.Selection.Escape
ActiveWindow.Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsBold = 2
ActiveWindow.SearchAttributes.SetBold(soWeightBold)
Activewindow.Search ""
Wait 2000
Activewindow.RepeatSearch
ActiveWindow.ReplaceAttributes.IsBold = 2
ActiveWindow.ReplaceAttributes.SetBold( 0 )
ActiveWindow.ReplaceAttributes.IsItalic = 2
ActiveWindow.ReplaceAttributes.SetItalic( 6 )
Activewindow.Replace
ActiveWindow.Selection.Escape
ActiveWindow.ReplaceAttributes.Reset
REM ab jetzt werden die gefundenen Stellen gelöscht
ActiveWindow.Replace
ActiveWindow.Replace
End Sub
```

SetLanguage-Methode

Setzt das Attribut Sprache zum Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: Language, IsLanguage.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetLanguage(Language As Long)

PARAMETER:
Language: Long-Integer-Ausdruck, der den Nummerncode für die Sprache angibt. Folgende Werte sind möglich:

255 - Keine

1030 - Dänisch

1031 - Deutsch

2055 - Deutsch (CH)

2057 - Englisch (UK)

1033 - Englisch (US)

1035 - Finnisch

1036 - Französisch

1040 - Italienisch

1043 - Niederländisch

1044 - Norwegisch Bokmal

2068 - Norwegisch Nynorsk

2070 - Portugiesisch

1053 - Schwedisch

1034 - Spanisch

BEISPIEL:

```
Sub ExampleSetLanguage
Const sMessage_1 = "Es wird ein Absatz bei dem die Sprache auf
Finnisch gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist als Sprache REM
FinnischREM  gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Language = 1035 REM Finnisch
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLanguage = 2
ActiveWindow.SearchAttributes.SetLanguage( 1035 )
If ActiveWindow.SearchAttributes.Language = 1035 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetLeftRightParaMargin-Methode

Setzt das Attribut Absatzeinzüge für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsLeftRightParaMargin, LeftMargin, IndentFirstLine, RightMargin.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetLeftRightParaMargin(LeftMargin As Long, IndentFirstLine As Integer, RightMargin As Long)

PARAMETER:
LeftMargin: Bestimmt den linken Einzug.

IndentFirstLine: Bestimmt den Erstzeileneinzug.

RightMargin: Bestimmt den rechten Einzug.

Die Werte werden in der aktuell definierten Maßeinheit übergeben.

BEISPIEL:

```
Sub Main
ActiveWindow.SearchAttributes.IsLeftRightParaMargin = 2
Activewindow.SearchAttributes.SetLeftRightParaMargin(1.2, 0, 0)
ExVar = ActiveWindow.SearchAttributes.LeftMargin
Activewindow.Search
End Sub
```

SetLineSpacing-Methode

Setzt das Attribut Zeilenabstand zum Suchen und Ersetzen von Formaten. Beim Anwenden dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei einem ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsLineSpacing, LineSpace, LineHeight, PropLineHeight, InterLineSpace, InterSpace.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetLineSpacing(LineSpace, LineHeight As Long, PropLineHeight As Long, InterLineSpace, InterSpace As Integer)

PARAMETER:
LineSpace: Legt die Option für den Zeilenabstand fest. Folgende Werte sind möglich:

0 - Automatisch

1 - Fest

2 - Mindestens

LineHeight: Wert für minimalen (InterLineSpace = 0 und LineSpace = 2) oder festen (InterLineSpace = 2 und LineSpace = 1) Zeilenabstand.

PropLineHeight: Wert für prozentualen Zeilenabstand.

InterLineSpace: Legt die Option für die Zeilenhöhe fest. Folgende Werte sind möglich:

0 - Ausgeschaltet

1 - Proportional

2 - Fest

InterSpace: Durchschußwert, gültig für LineSpace = 0 und InterLineSpace = 2.

Die Parameter LineHeight und InterSpace werden in der aktuell definierten Standard-Maßeinheit übergeben.

BEISPIEL:

```
Sub ExampleSetLineSpacing
Const sMessage_1 = "Es wird ein Absatz mit einer Zeilehöhe von
1.5 cm, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Zeilenhöhe auf
1.5 cm gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.LineSpace = 1
Selection.Paragraph.PropLineHeight = 200
Selection.Paragraph.LineHeight = 852 REM 1,5cm
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsLineSpacing = 2
ActiveWindow.SearchAttributes.SetLineSpacing( 1, 852, 0, 0, 0 )
If ActiveWindow.SearchAttributes.LineHeight = 852 Then
Print sMessage_1
```

```
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetLocks-Methode

Entspricht dem mehrfachen Aufrufen von LockPaint.

Siehe auch: LockPaint

KLASSE:
BaseTextEditView

SYNTAX:
SetLocks(Depth As Long)

PARAMETER:
Depth: Anzahl der einzustellenden Lock-Ebenen.

BEISPIEL:

```
Sub ExampleSetLocks
Const sMessage_1 = "Den Aufbau der beiden Absätze ist nicht zu
sehen."
Const sMessage_2 = "Das ist der zweite Absatz."
Const sError = "Das Abschalten der Bildschirmaktualisierung hat
nicht funktioniert"
Dim lLockCount as Long, lCount As Long
Documents.Add( "sWriter4" )
lLockCount = 3
ActiveWindow.SetLocks(lLockCount)
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Insert sMessage_2
If ActiveWindow.IsPaintLocked = FALSE Then
Print sError
End If
For lCount = 1 To lLockCount
ActiveWindow.LockPaint(FALSE)
Next lCount
End Sub
```

SetObjectToBackground-Methode

Setzt das selektierte Objekt in den Hintergrund.

KLASSE:
BaseTextDrawBase

SYNTAX:
SetObjectToBackground()

Beispiel:

```
Sub ExampleSetObjectToGround
Const sMessage_1 = "Dieser Commandbutton steht im Hintergrund"
Const sMessage_2 = "Dieser Commandbutton steht im Vordergrund"
Dim sFirstControl As String, sSecondControl As String
Documents.Add( "sWriter4" )
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
sFirstControl = Selection.ObjectName
Selection.Escape
Activewindow.InsertVCControl( 2, "6cm", "6cm", "7cm", "3cm")
sSecondControl = Selection.ObjectName
Selection.Escape
Activewindow.Controls(sSecondControl).ControlObject.Caption =
sMessage_1
Selection.SetObjectToBackground
Activewindow.Controls(sFirstControl).ControlObject.Caption =
sMessage_2
ActiveWindow.Selection.SetObjectToForeground
End Sub
```

SetObjectToForeground-Methode

Setzt das selektierte Objekt in den Vordergrund.

Klasse:
BaseTextDrawBase

Syntax:
SetObjectToForeground()

Beispiel:

```
Sub ExampleSetObjectToGround
Const sMessage_1 = "Dieser Commandbutton steht im Hintergrund"
Const sMessage_2 = "Dieser Commandbutton steht im Vordergrund"
Dim sFirstControl As String, sSecondControl As String
Documents.Add( "sWriter4" )
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
sFirstControl = Selection.ObjectName
Selection.Escape
Activewindow.InsertVCControl( 2, "6cm", "6cm", "7cm", "3cm")
sSecondControl = Selection.ObjectName
Selection.Escape
Activewindow.Controls(sSecondControl).ControlObject.Caption =
sMessage_1
Selection.SetObjectToBackground
Activewindow.Controls(sFirstControl).ControlObject.Caption =
sMessage_2
ActiveWindow.Selection.SetObjectToForeground
End Sub
```

SetOptimalColumnWidth-Methode

Setzt die optimale Spaltenbreite für die selektierte Tabellenspalte.

KLASSE:
BaseTextTable

SYNTAX:
SetOptimalColumnWidth()

BEISPIEL:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

SetOptimalRowHeight-Methode

Setzt die optimale Zeilenhöhe für die selektierte Tabellenzeile.

KLASSE:
BaseTextTable

SYNTAX:
SetOptimalRowHeight()

BEISPIEL:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
```

```
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

SetOrphan-Methode

Setzt das Absatzattribut HURENKINDREGELUNG für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei einem ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: Orphan, IsOrphan.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetOrphan(Orphan As Integer)

PARAMETER:
[Orphan]: Anzahl der Zeilen eines auf die nächste Seite umgebrochenen Absatzes, die am oberen Rand dieser Seite aufgeführt werden dürfen (Hurenkindregel). Wird dieser Wert nicht übergeben, wird der Standardwert Orphan = 0 genommen.

BEISPIEL:

```
Sub ExampleSetOrphan
Const sMessage_1 = "Es wird ein Absatz, bei dem die Huren-
kindregel auf 4 Zeilen gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Hurenkindregel
auf 4 Zeilen gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Orphan = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsOrphan = 2
ActiveWindow.SearchAttributes.SetOrphan( 4 )
If ActiveWindow.SearchAttributes.Orphan = 4 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetPageStyleCols-Methode

Setzt die Anzahl der Spalten an der aktuellen Seitenvorlage. Ohne Übergabeparameter wird der Dialog zur Bearbeitung der Seitenvorlage aufgerufen.

KLASSE:
BaseTextSelection

SYNTAX:
SetPageStyleCols(Cols As Long)

PARAMETER:
[Cols]: Long-Integer-Ausdruck, der die Anzahl der gewünschten Spalten angibt. Dieser Parameter ist optional.

BEISPIEL:

```
Sub ExampleTextColumns
Const sMessage_1 = "Dies sind Beispielsätze für die erste
Spalte. "
Const sMessage_2 = "Dies sind Beispielsätze für die zweite
Spalte. "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.SetPageStyleCols( 2 )
For iCount=1 To 10
Selection.Insert sMessage_1
Next iCount
Selection.InsertColumnBreak
For iCount=1 To 10
Selection.Insert sMessage_2
Next iCount
Selection.GotoStartOfDoc FALSE
Selection.GotoEndOfNextColumn FALSE
Selection.GotoStartOfPrevColumn FALSE
Selection.GotoStartOfNextColumn FALSE
Selection.GotoEndOfPrevColumn FALSE
Selection.GotoStartOfColumn FALSE
Selection.GotoEndOfColumn FALSE
End Sub
```

SetPageStyleFooter-Methode

Mit dieser Methode wird die Fußzeile der aktuellen Seitenvorlage eingestellt.

KLASSE:
BaseTextSelection

SYNTAX:
SetPageStyleFooter(On As Boolean, Dist As Long, Height As Long, ShareContent As Boolean, Dynamic As Boolean)

PARAMETER:
[On]: Gibt an, ob in der Seitenvorlage eine Fußzeile eingerichtet werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Voreinstellung On = False genommen.

[Dist]: Legt den Abstand der Fußzeile vom Textbereich in der Standard-Maßeinheit fest. Wird dieser Parameter nicht übergeben, wird Dist = 0 gesetzt.

[Height]: Gibt die Höhe der Fußzeile in der Standard-Maßeinheit an. Wird dieser Wert nicht übergeben, wird eine Höhe von 0,5 cm genommen.

[ShareContent]: Gibt an, ob der Inhalt der Fußzeile auf linken und rechten Seiten identisch sein soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Voreinstellung ShareContent = False genommen.

[Dynamic]: Gibt an, ob die Höhe der Fußzeile dynamisch angepaßt werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Höhe dynamisch angepaßt.

Die Parameter sind jeweils optional. Ein Aufruf dieser Methode ist aber nicht ohne Übergabeparameter möglich, mindestens 1 Parameter muß übergeben werden.

BEISPIEL:

```
Sub ExampleSetPageStyleHeader
Const sMessage_1 = "Dies ist eine Kopfzeile"
Const sMessage_2 = "Dies ist eine Fußzeile"
Documents.Add( "sWriter4" )
REM Legt eine Kopfzeile an
Selection.SetPageStyleHeader(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToHeader()
Selection.Insert sMessage_1
Selection.JumpToEndOfDoc
REM Legt eine Fußzeile an
Selection.SetPageStyleFooter(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToFooter()
Selection.Insert sMessage_2
End Sub
```

SetPageStyleHeader-Methode

Mit dieser Methode wird die Kopfzeile der aktuellen Seitenvorlage eingestellt.

KLASSE:
BaseTextSelection

SYNTAX:
SetPageStyleHeader(On As Boolean, Dist As Long, Height As Long, ShareContent As Boolean, Dynamic As Boolean)

PARAMETER:
[On]: Gibt an, ob in der Seitenvorlage eine Kopfzeile eingerichtet werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Voreinstellung On = False genommen.

[Dist]: Legt den Abstand der Kopfzeile vom Textbereich in der aktuell definierten Maßeinheit fest. Wird dieser Parameter nicht übergeben, wird Dist = 0 gesetzt.

[Height]: Gibt die Höhe der Kopfzeile in der Standard-Maßeinheit an. Wird dieser Wert nicht übergeben, wird eine Höhe von 0,5 cm genommen.

[ShareContent]: Bestimmt, ob die Kopfzeilentexte auf linken und rechten Seiten doppelseitig bedruckter Dokumente identisch (True) oder verschieden (False) sein sollen. Wird dieser Parameter nicht übergeben, wird die Voreinstellung ShareContent = False genommen.

[Dynamic]: Bestimmt, ob die Höhe der Kopfzeile dynamisch an Textmenge und Schriftgröße angepaßt werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Höhe dynamisch angepaßt.

Die Parameter sind jeweils optional. Ein Aufruf dieser Methode ist aber nicht ohne Übergabeparameter möglich, mindestens 1 Parameter muß übergeben werden.

BEISPIEL:

```
Sub ExampleSetPageStyleHeader
Const sMessage_1 = "Dies ist eine Kopfzeile"
Const sMessage_2 = "Dies ist eine Fußzeile"
Documents.Add( "sWriter4" )
REM Legt eine Kopfzeile an
Selection.SetPageStyleHeader(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToHeader()
Selection.Insert sMessage_1
Selection.JumpToEndOfDoc
REM Legt eine Fußzeile an
```

```
Selection.SetPageStyleFooter(TRUE, "0.5cm", "2cm", FALSE, TRUE)
Selection.JumpToFooter()
Selection.Insert sMessage_2
End Sub
```

SetParaSplit-Methode

Setzt das Absatzattribut Absatz trennen für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei einem ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsParaSplit, ParaSplit.

Klasse:
ReplaceAttributes, SearchAttributes

Syntax:
SetParaSplit(ParaSplit As Boolean)

Parameter:
[ParaSplit]: Boolescher Ausdruck, der angibt, ob nach dem Attribut Absatz trennen gesucht wird (True) oder nicht (False) oder ob das Attribut beim Ersetzen angewendet werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, wird die Voreinstellung ParaSplit = True genommen.

Beispiel:

```
Sub ExampleSetParaSplit
Const sMessage_1 = "Es wird ein Absatz, bei dem ein Seitenum-
bruch im Absatz erfolgen darf, gesucht."
Const sMessage_2 = "Bei diesem Absatz darf ein Seitenumbruch
innerhalb eines Absatzes erfolgen."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.ParaSplit = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsParaSplit = 2
ActiveWindow.SearchAttributes.SetParaSplit( TRUE )
If ActiveWindow.SearchAttributes.ParaSplit = TRUE Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetRegionCondition-Methode

Setzt die Bedingung, unter der ein Bereich im Dokument versteckt werden soll.

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionCondition(RegionName As String, Condition As String)

PARAMETER:
RegionName: Name des Bereichs, der unter einer bestimmten Bedinungung ausgeblendet werden soll.

[Condition]: Logischer Ausdruck, der die Bedingung angibt, unter der der Bereich ausgeblendet werden soll. Wird dieser Parameter nicht angegeben oder wird ein Leerstring übergeben, dann wird eine bereits vorhandene Bedingung zurückgesetzt.

BEISPIEL:

```
Sub ExampleGetRegionCondition
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Die Bedingung von dem Bereich REM "
Const sMessage_3 = "REM  lautet : "
Dim sRegionCond As String
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine FALSE
Selection.GotoEndOfLine TRUE
Selection.InsertRegion( sRegionName, "False", FALSE, FALSE)
Selection.Escape
Selection.SetRegionCondition( sRegionName, "True" )
Selection.InsertPara
sRegionCond = Selection.GetRegionCondition( sRegionName )
Selection.Insert sMessage_2 & sRegionName & sMessage_3 &
sRegionCond
End Sub
```

SetRegionHidden-Methode

Bestimmt, ob ein Bereich im Dokument eingeblendet oder versteckt werden soll. Diese Option läßt sich auch an Bedinungungen knüpfen. Ist mit SetRegionCondition eine Bedingung für das Ausblenden gesetzt, wird diese ausgewertet.

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionHidden(RegionName As String, Protect As Boolean)

PARAMETER:
RegionName: Name des Bereichs, dessen Eigenschaft bestimmt werden soll.

[Protect]: Gibt an, ob der Bereich versteckt werden soll (True) oder nicht (False). Ist eine Bedingung gesetzt, wird diese ausgewertet. Wird dieser Parameter nicht übergeben, wird der Bereich standardmäßig versteckt (Protect = True).

BEISPIEL:

```
Sub ExampleSetRegionHidden
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  ist unsichtbar"
Const sMessage_4 = "REM  ist sichtbar"
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionHidden(sRegionName, TRUE)
If Selection.IsRegionHidden(sRegionName) = TRUE Then
Selection.Insert sMessage_2 & sRegionName & sMessage_3
Else
Selection.Insert sMessage_2 & sRegionName & sMessage_4
End If
Selection.DeleteRegion sRegionName
End Sub
```

SetRegionLink-Methode

Fügt eine Datei als Verknüpfung in einen Bereich des aktuellen Dokuments ein oder hebt eine bestehende Verknüpfung wieder auf.

Siehe auch: SetRegionLinkBookmark

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionLink(RegionName As String, LinkName As String, FilterName As String)

PARAMETER:
RegionName: Name des Bereichs im aktuellen Dokument, dessen Eigenschaft verändert werden soll.

[LinkName]: Name der Datei, mit der der Bereich verknüpft werden soll. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, dann wird eine bestehende Verknüpfung aufgehoben.

[FilterName]: Name des Dateifilters, mit dem die Datei "LinkName" geladen werden soll. Dieser Parameter ist optional. Wird er nicht übergeben, erfolgt eine automatische Filterkennung.

BEISPIEL:

```
Sub ExampleSetRegionLink
Const sRegionName = "Bereich"
Const sRegionDocName = "Bereich.sdw"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der verknüpfte Bereich besteht aus
folgender Datei : "
Const sMessage_3 = "und es wurde folgender Filter verwendet : "
Dim sRegionPath As String
sRegionPath = Application.PathSettings.Work & sRegionDocName
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
ActiveDocument.Close TRUE, sRegionPath
Documents.Add( "sWriter4" )
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionLink( sRegionName, sRegionPath, "StarWriter
4.0")
Selection.Insert sMessage_2 & Selection.GetRegionLink( sRegion-
Name )
Selection.InsertPara
Selection.Insert sMessage_3 & Selection.GetRegionLinkFilter(
sRegionName )
End Sub
```

SetRegionLinkBookmark-Methode

Fügt einen Bereich oder eine Textmarke einer anderen Datei als Verknüpfung in einen Bereich des aktuellen Dokuments ein oder hebt eine bestehende Verknüpfung wieder auf.

Vor Aufruf dieser Methode muß bereits eine Verknüpfung mit der Datei bestehen, die den einzufügenden Bereich enthält.

Siehe auch: SetRegionLink

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionLinkBookmark(RegionName As String, Bookmark As String)

PARAMETER:
RegionName: Name des Bereichs im aktuellen Dokument, dem ein Bereich einer verknüpften, externen Datei zugewiesen werden soll.

[Bookmark]: Name eines Bereichs (oder einer Textmarke) innerhalb der verknüpften, externen Datei, der (oder die) dem Bereich des aktuellen Dokuments zugewiesen werden soll. Wird dieser Parameter nicht gesetzt oder wird ein Leerstring übergeben, wird eine bereits bestehende Verknüpfung zu dem Bereich (oder der Textmarke) der externen Datei wieder aufgehoben.

BEISPIEL:

```
Sub ExampleSetRegionLinkBookmark
Const sRegionName = "Bereich"
Const sRegionInDoc = "DokBereich"
Const sRegionDocName = "Bereich.sdw"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Diese Zeile wird nicht eingefügt"
Const sMessage_3 = "Der Bereich im Dokument heißt : "
Dim sRegionPath As String
sRegionPath = Application.PathSettings.Work & sRegionDocName
Documents.Add( "sWriter4" )
Selection.Insert sMessage_2
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionInDoc, "True", FALSE, FALSE)
ActiveDocument.Close TRUE, sRegionPath
Documents.Add( "sWriter4" )
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionLink( sRegionName, sRegionPath, "StarWriter
4.0")
Selection.SetRegionLinkBookmark( sRegionName, sRegionInDoc )
```

```
Selection.Insert sMessage_3 & Selection.GetRegionLinkBookmark(
sRegionName )
End Sub
```

SetRegionPassword-Methode

Setzt das Paßwort für einen im Dokument definierten Bereich.

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionPassword(Password As String)

PARAMETER:
PassWord: Zeichenfolge, die das Paßwort für den Bereich angibt.

BEISPIEL:

```
Sub ExampleSetRegionPassword
Const sRegionName = "Bereich"
Const sPassword = "Passwort"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der BereichsDialog ist Passwort geschützt"
Const sMessage_3 = "Der BereichsDialog ist nicht Passwort
geschützt"
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionPassword(sPassword)
If Selection.IsRegionPassword() = TRUE Then
Selection.Insert sMessage_2
Else
Selection.Insert sMessage_3
End If
End Sub
```

SetRegionProtected-Methode

Bestimmt, ob ein im Dokument definierter Bereich geschützt werden soll.

KLASSE:
BaseTextSelection

SYNTAX:
SetRegionProtected(RegionName As String, Protect As Boolean)

PARAMETER:
RegionName: Name des Bereichs, dessen Eigenschaft verändert werden soll.

[Protect]: Gibt an, ob der Bereich geschützt sein soll (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht übergeben, wird der Bereich geschützt (Protect = True).

Beispiel:

```
Sub ExampleGetRegionName
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Zeile wird ein Bereich"
Const sMessage_2 = "Der Bereich REM "
Const sMessage_3 = "REM  ist geschützt"
Const sMessage_4 = "REM  ist nicht geschützt"
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.Insert sMessage_1
Selection.GotoStartOfLine TRUE
Selection.InsertRegion( sRegionName, "True", FALSE, FALSE)
Selection.InsertPara
Selection.SetRegionProtected(sRegionName, TRUE)
If Selection.IsRegionProtected(sRegionName) = TRUE Then
Selection.Insert sMessage_2 & sRegionName & sMessage_3
Else
Selection.Insert sMessage_2 & sRegionName & sMessage_4
End If
End Sub
```

SetRowHeight-Methode

Setzt die Höhe der aktuellen Tabellenzeile. Ohne Übergabeparameter wird der Dialog Zeilenhöhe aufgerufen.

Klasse:
BaseTextTable

Syntax:
SetRowHeight(Height As Long, AutoHeight As Boolean)

Parameter:
Height: Long-Integer-Audruck, der die neue Höhe der aktuellen Zeile in der Standard-Maßeinheit angibt.

[AutoHeight]: Boolescher Ausdruck, der angibt, ob die Höhe dynamisch angepaßt werden soll (True) oder nicht (False). Wird dieser Parameter nicht übergeben, erfolgt keine dynamische Anpassung (AutoHeight = False).

Beispiel:

```
Sub ExampleTableWidthHeight
Documents.Add("sWriter4")
Selection.InsertTable(3, 5)
Selection.SetRowHeight("4cm")
```

```
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.SetOptimalColumnWidth
Selection.SetOptimalRowHeight
Selection.Escape
Selection.SetRowHeight("4cm")
Selection.SetColumnWidth(1, "5cm")
Selection.Select
Selection.DistributeColumns
Selection.DistributeRows
End Sub
```

SetShadowed-Methode

Setzt das Zeichenattribut Schattiert für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsShadowed, Shadowed.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetShadowed(Shadowed As Boolean)

PARAMETER:
Shadowed: Boolescher Ausdruck, der angibt, ob nach dem Zeichenattribut SCHATTIERT gesucht wird oder ob das Attribut beim Ersetzen angewendet werden soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSetShadowed
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
schattiert dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift schat-
tiert dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Shadowed = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsShadowed = 2
ActiveWindow.SearchAttributes.SetShadowed( TRUE )
If ActiveWindow.SearchAttributes.Shadowed = TRUE Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetSpacing-Methode

Setzt das Attribut Kerning zum Suchen und Ersetzen von Formaten. Beim Anwenden dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht. Bei der Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsSpacing, Spacing.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetSpacing(Spacing As Integer)

PARAMETER:
Spacing: Integer-Ausdruck, der den Wert für die Laufweite in Twips angibt. Wird ein positiver Wert übergeben, werden die gesperrt dargestellten Zeichen für das Suchen und Ersetzen genommen, bei einem negativen Wert die Zeichen mit schmaler Darstellung. Der Wert 0 stellt für das Kerning die Option STANDARD ein.

BEISPIEL:

```
Sub ExampleSpacing
Const sMessage_1 = "Es wird ein Absatz, bei dem die Laufweite
des Fonts verändert wurde, gesucht."
Const sMessage_2 = "Bei diesem Absatz wurde die Laufweite des
Fonts verändert."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Spacing = 20
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsSpacing = 2
ActiveWindow.SearchAttributes.SetSpacing( 20 )
If ActiveWindow.SearchAttributes.Spacing = 20 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetStrikeOut-Methode

Setzt das Attribut Durchstrichen für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsStrikeOut, StrikeOut.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetStrikeOut(StrikeOut)

PARAMETER:
Strikeout: Gibt den Typ des Attributs Durchstrichen an. Folgende Werte sind möglich:

0 - Keine Durchstreichung

1 - Einfach

2 - Doppelt

BEISPIEL:

```
Sub ExampleSetItalic
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
kursiv dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift kursiv
dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Italic = soItalicOblique
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsItalic = 2
ActiveWindow.SearchAttributes.SetItalic( soItalicOblique )
If ActiveWindow.SearchAttributes.Italic = soItalicOblique Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetTopBottomMargin-Methode

Setzt das Absatzattribut Abstand zum Suchen und Ersetzen von Formaten. Das Attribut Abstand bezieht sich auf den oberen und unteren Abstand eines Absatzes zu seiner Umgebung. Es wird bestimmt durch die Werte in den Drehfeldern Oben und Unten unter Suchen & Ersetzen - Format - Einzüge und Abstände - Abstand. Diese Werte werden mit der SetTopBottomMargin-Methode übergeben. Beim Anwenden dieser Methode auf ein SearchAttributes-Objekt werden die übergebenen Werte gesucht. Bei der Anwendung auf ein ReplaceAttributes-Objekt werden sie beim Ersetzen angewendet.

Siehe auch: IsTopBottomMargin, TopMargin, BottomMargin.

Klasse:
ReplaceAttributes, SearchAttributes

Syntax:
SetTopBottomMargin(TopMargin As Long, BottomMargin As Long)

Parameter:
TopMargin: Abstand zwischen der ersten Zeile des Absatzes, der gesucht oder beim Ersetzen angewendet werden soll, und der letzten Textzeile des vorangehenden Absatzes.

BottomMargin: Abstand zwischen der letzten Zeile des Absatzes, der gesucht oder beim Ersetzen angewendet werden soll, und der ersten Textzeile des nachfolgenden Absatzes

Beispiel:

```
Sub Main
ActiveWindow.SearchAttributes.IsTopBottomMargin = 2
Activewindow.SearchAttributes.SetTopBottomMargin("1.2cm", "")
ExVar = ActiveWindow.SearchAttributes.BottomMargin
Activewindow.Search
End Sub
```

SetUnderline-Methode

Setzt das Attribut Unterstrichen für das Suchen und Ersetzen von Formaten. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: IsUnderline, Underline.

Klasse:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetUnderline(Underline)

PARAMETER:
Underline: Gibt den Wert des Attributs Unterstrichen an, nach dem gesucht oder durch den ersetzt werden soll. Folgende Werte sind möglich:

0 - Keine Unterstreichung

1 - Einfach

2 - Doppelt

3 - Punktiert

BEISPIEL:

```
Sub ExampleSetUnderline
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schrift
unterstrichen dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird die Schrift unter-
strichen dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Underline = soUnderlineSingle
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsUnderline = 2
ActiveWindow.SearchAttributes.SetUnderline( soUnderlineSingle )
If ActiveWindow.SearchAttributes.Underline = soUnderlineSingle
Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetUserFieldSubType-Methode

Setzt den Subtyp eines benutzerdefinierten Feldes. Dieser Subtyp bestimmt, ob es sich um ein numerisches Feld oder ein Textfeld handelt.

KLASSE:
BaseTextSelection

SYNTAX:
SetUserFieldSubType(FieldName As String, SubType As Long)

RÜCKGABE:
Boolean

PARAMETER:
FieldName: Gibt den Namen eines im Dokument definierten Benutzerfeldes an.

[SubType]: Bestimmt den Subtyp des Feldes. Folgende Einstellungen sind möglich:

0 - numerisch

1 - Text

Dieser Paramter ist optional. Wird er nicht angegeben, wird das Feld als numerisches Feld interpretiert (SubType = 0).

BEISPIEL:

```
Sub ExampleDefineUserFields
Const sFieldName = "NeuesFeld"
Const sMessage = "Hier erscheint ein Feld "
Documents.Add( "swriter4" )
With ActiveWindow.Selection
.Insert( sMessage )
.NewField( sFieldName )
.SetUserFieldSubType( sFieldName, 0 )
.SetUserFieldValue( sFieldName, "3,1415" )
End With
End Sub
```

SetUserFieldValue-Methode

Bestimmt einen neuen Wert für ein benutzerdefiniertes Feld.

Änderungen, die Sie mit SetUserFieldValue vorgenommen haben, werden erst nach einer Aktualisierung der Felder sichtbar. Felder im Dokument können mit der UpdateFields-Methode aktualisiert werden.

KLASSE:
BaseTextSelection

SYNTAX:
SetUserFieldValue(Name As String, Value As String)

RÜCKGABE:
Boolean

PARAMETER:
Name: Ein Zeichenfolgenausdruck, der den Namen des benutzerdefinierten Feldes angibt.

[Value]: Ein Zeichenfolgenausdruck, der den neuen Feldwert enthält. Wird dieser Parameter nicht gesetzt, wird ein Leerstring übergeben.

Beispiel:

```
Sub ExampleDefineUserFields
Const sFieldName = "NeuesFeld"
Const sMessage = "Hier erscheint ein Feld "
Documents.Add( "swriter4" )
With ActiveWindow.Selection
.Insert( sMessage )
.NewField( sFieldName )
.SetUserFieldSubType( sFieldName, 0 )
.SetUserFieldValue( sFieldName, "3,1415" )
End With
End Sub
```

SetWidow-Methode

Setzt das Absatzattribut Schusterjungenregelung zum Suchen und Ersetzen von Formaten. Beim Anwenden dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht. Bei der Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: Widow, IsWidow.

Klasse:
ReplaceAttributes, SearchAttributes

Syntax:
SetWidow(Widow As Integer)

Parameter:
Widow: Anzahl der Zeilen eines auf die nächste Seite umgebrochenen Absatzes, die noch am unteren Rand der ersten Seite aufgeführt werden dürfen (Schusterjungenregel).

Beispiel:

```
Sub ExampleSetWidow
Const sMessage_1 = "Es wird ein Absatz, bei dem die Schuster-
jungenregel auf 4 Zeilen gesetzt ist, gesucht."
Const sMessage_2 = "Bei diesem Absatz ist die Schusterjungen-
regel auf 4 Zeilen gesetzt"
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Paragraph.Widow = 4
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsWidow = 2
ActiveWindow.SearchAttributes.SetWidow( 4 )
```

```
If ActiveWindow.SearchAttributes.Widow = 4 Then
Print sMessage_1
End if
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

SetWordMode-Methode

Setzt das Zeichenattribut Wortweise für das Suchen und Ersetzen von Formaten. Mit dieser Methode wird bestimmt, ob die Attribute Unterstreichung und Durchstreichung wortweise gesucht oder ersetzt werden sollen. Bei Anwendung dieser Methode auf ein SearchAttributes-Objekt wird der übergebene Wert gesucht, bei Anwendung auf ein ReplaceAttributes-Objekt wird er beim Ersetzen angewendet.

Siehe auch: WordMode, IsWordMode.

KLASSE:
ReplaceAttributes, SearchAttributes

SYNTAX:
SetWordMode(WordMode As Boolean)

PARAMETER:
WordMode: Boolescher Ausdruck, der angibt, ob das Attribut Unterstreichung oder Durchstreichung wortweise gesucht wird oder ob beim Ersetzen diese Attribute wortweise angewendet werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSetWordMode
Const sMessage_1 = "Es wird ein Absatz, bei dem die jedes Wort
einzeln unterstrichen dargestellt wird, gesucht."
Const sMessage_2 = "Bei diesem Absatz wird jedes Wort einzeln
unterstrichen dargestellt."
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
Selection.Font.Underline = soUnderlineSingle
Selection.Font.WordMode = TRUE
Selection.Insert sMessage_2
Selection.JumpToStartOfDoc
ActiveWindow.SearchAttributes.IsWordMode = 2
ActiveWindow.SearchAttributes.SetWordMode( TRUE )
If ActiveWindow.SearchAttributes.WordMode = TRUE Then
Print sMessage_1
End if
```

```
Activewindow.Search ""
ActiveWindow.SearchAttributes.Reset
End Sub
```

Shadow-Eigenschaft

Interface für Schattierung, enthält die Eigenschaften:

Location

Width

Style

Transparent

BackColor

FillColor

KLASSE:
BaseTextSelection

SYNTAX:
Shadow

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleShadow
Documents.Add("sWriter4")
Selection.InsertFrame(1, "3cm", "3cm", "4cm", "4cm")
Selection.Shadow.Style = 2
Selection.Shadow.Fillcolor = RGB(255, 255, 0) REM Gelb
Selection.Shadow.Location = 2
End Sub
```

Shadowed-Eigenschaft

Bestimmt oder ermittelt, ob das Zeichenattribut Schattiert gesetzt ist. Beim Suchen und Ersetzen von Formaten ermittelt diese Eigenschaft die mit SetShadowed gesetzte Suchoption für das Attribut Schattiert, wenn IsShadowed den Wert 2 liefert.

Siehe auch: SetShadowed, IsShadowed.

KLASSE:
TextDrawText, ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Shadowed

RÜCKGABE:
Boolean

PARAMETER:
Bei Text-Objekten:

[= Switch]: Boolescher Ausdruck, der angibt, ob das Attribut Schattiert ein- (True) oder ausgeschaltet (False) ist.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
```

```
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
```

```
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

ShowCursor-Eigenschaft

Schaltet die Anzeige des Cursors aus.

KLASSE:
BaseTextSelection

SYNTAX:
ShowCursor

RÜCKGABE:
Boolean

PARAMETER:
True: Schaltet die Anzeige des Cursors ein.

False: Schaltet die Anzeige des Cursors aus.

BEISPIEL:

```
Sub ExampleTableSort
Const sTable="Tabelle 100"
Const sT1="Wörter"
Const sT2="Zahlen"
Documents.Add("sWriter4")
REM Tabelle zum Sortieren erstellen
Selection.InsertNamedTable( sTable, 4,2 )
Selection.GotoTable(sTable): Selection.Insert(sT1)
Selection.JumpToNextCell : Selection.Insert(sT2)
Selection.JumpToNextCell : Selection.Insert("ABC")
Selection.JumpToNextCell : Selection.Insert("53")
Selection.JumpToNextCell : Selection.Insert("XYZ")
Selection.JumpToNextCell : Selection.Insert("234")
Selection.JumpToNextCell : Selection.Insert("GHT")
Selection.JumpToNextCell : Selection.Insert("1254")
REM Nach erster Spalte sortieren
ActiveWindow.ShowCursor = False REM Anzeige ausschalten
Selection.GotoTable(sTable)
Wait 3000
Selection.EntireColumn
Selection.GoRight 1, TRUE
Selection.Sort( 0,1,,1,0)
REM nach zweiter Spalte Sortieren
Selection.GotoTable(sTable)
```

```
Wait 2000
Selection.EntireColumn
Selection.GoRight 1, TRUE
Selection.Sort( 0,1,,2,1)
ActiveWindow.ShowCursor = True REM Anzeige einschalten
End Sub
```

ShowFourPages-Methode

Zeigt in der Seitenvorschau vier Seiten gleichzeitig an.

Bevor diese Methode aufgerufen wird, muß die Befehlszeile ActiveDocument.PrintPreview = True ausgeführt werden, die die Seitenvorschau einschaltet.

KLASSE:
TextPrintPreview

SYNTAX:
ShowFourPages()

BEISPIEL:

```
Sub ExamplePrintPreview
Dim iCount As Integer
Documents.Add("sWriter4")
wait 100
For iCount= 1 to 6
ActiveWindow.Selection.Insert("Das ist Seite " & iCount)
ActiveWindow.InsertPageBreak
Next iCount
ActiveDocument.PrintPreview = True
ActiveWindow.GotoStartOfDoc
ActiveWindow.ShowTwoPages()
Wait 2000 REM Zeigt zweiseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageDown
Wait 1000
Next iCount
ActiveWindow.GotoEndOfDoc
ActiveWindow.ShowFourPages()
Wait 2000 REM zeigt vierseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageUp
Wait 1000
Next iCount
ActiveWindow.PrintPagePreView
ActiveDocument.PrintPreview = False
End Sub
```

ShowHiddenPara-Eigenschaft

Schaltet die Option Extras - Optionen - Textdokument... - Register Inhalte "Versteckte Absätze" ein oder aus.

Klasse:
ViewSettings

Syntax:
ShowHiddenPara

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob versteckte Absätze angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

ShowQuery-Methode

Zeigt eine Datenbank-Abfrage im Beamer an. Die Datenbankabfrage muß vorher für diese Datenbank definiert worden sein, und die Datenbank muß geöffnet sein.

Konnte die Methode erfolgreich ausgeführt werden, wird der Wert True zurückgegeben, andernfalls False.

KLASSE:
BaseTextEditView

SYNTAX:
ShowQuery(QueryName As String)

RÜCKGABE:
Boolean

PARAMETER:
QueryName: Zeichenfolge, die den Namen der Abfrage angibt.

BEISPIEL:

```
Sub ExampleShowQuery
Const sQueryName = "Abfrage"
Dim sDataName As String, sTableName As String
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.CreateQuery( sQueryName, "SELECT * FROM " &
sTableName, FALSE )
Activewindow.ShowQuery( sQueryName )
Activewindow.DeleteQuery( sQueryName )
End Sub
```

ShowStatement-Methode

Zeigt eine SQL-Selektion im Beamer an. Voraussetzung ist, daß vor Aufruf dieser Methode die zugehörige Datenbank, an der die SQL-Abfrage durchgeführt wird, mittels OpenDatabaseConnection geöffnet wurde.

Siehe auch: OpenDatabaseConnection

KLASSE:
BaseTextEditView

SYNTAX:
ShowStatement(StatementName As String)

RÜCKGABE:
Boolean

PARAMETER:
StatementName: Zeichenfolge, die einen gültigen SQL-Ausdruck aufnimmt, z. B. "Select * from Adressen".

BEISPIEL:

```
Sub ExampleShowStatement
Dim sDataName As String, sTableName As String
Dim vStatus As Variant
Documents.Add("sWriter4")
REM Die erste angemeldete Datenbank wird ermittelt
sDataName = ActiveWindow.DataDefinedDatabase(1)
vStatus = ActiveWindow.OpenDatabaseConnection(sDataName)
REM Die erste Tabelle innerhalb der Datenbank wird ermittelt
sTableName = ActiveWindow.DataGetTableName(sDataName, 1)
ActiveWindow.ShowStatement("SELECT * FROM " & sTableName)
End Sub
```

ShowTwoPages-Methode

Zeigt in der Seitenvorschau zwei Seiten gleichzeitig an.

Bevor diese Methode aufgerufen wird, muß die Befehlszeile ActiveDocument.PrintPreview = True ausgeführt werden, die die Seitenvorschau einschaltet.

KLASSE:
TextPrintPreview

SYNTAX:
ShowTwoPages()

BEISPIEL:

```
Sub ExamplePrintPreview
Dim iCount As Integer
Documents.Add("sWriter4")
wait 100
For iCount= 1 to 6
ActiveWindow.Selection.Insert("Das ist Seite " & iCount)
ActiveWindow.InsertPageBreak
Next iCount
```

```
ActiveDocument.PrintPreview = True
ActiveWindow.GotoStartOfDoc
ActiveWindow.ShowTwoPages()
Wait 2000 REM Zeigt zweiseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageDown
Wait 1000
Next iCount
ActiveWindow.GotoEndOfDoc
ActiveWindow.ShowFourPages()
Wait 2000 REM zeigt vierseitige Seitenvorschau 2 Sek. lang an
For iCount= 1 to 2
ActiveWindow.PageUp
Wait 1000
Next iCount
ActiveWindow.PrintPagePreView
ActiveDocument.PrintPreview = False
End Sub
```

Shrink-Methode

Verringert die aktuelle Schriftgröße auf die jeweils nächstmögliche Größe.

KLASSE:
TextFont

SYNTAX:
Shrink()

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
```

```
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
```

```
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Size-Eigenschaft

Bestimmt oder ermittelt die Schriftgröße des selektierten Textes oder der Zeichen, die an der Position der Schreibmarke eingefügt werden. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetFontHeight gesetzte Schriftgröße abgefragt.

Eine Abfrage der Size-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsFontHeight bestimmt wurde, daß nach einer bestimmten Schriftgröße gesucht werden soll, d. h. IsFontHeight = 2.

Siehe auch: SetFontHeight, IsFontHeight, Prop.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
Size

RÜCKGABE:
Long

PARAMETER:
Bei Text-Objekten::

[= Value]: Long-Integer-Ausdruck, der die Größe der Schrift in Punkt angibt. Der Wert kann auch mit Angabe der gewünschten Maßeinheit als Zeichenfolge gesetzt werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
```

```
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

SmoothScroll-Eigenschaft

Bestimmt oder ermittelt die Option für Weiches Scrollen in Textdokumenten. Diese Option ist unter Extras - Optionen - Textdokument... - Register "Layout" zu erreichen.

Klasse:
ViewSettings

Syntax:
SmoothScroll

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Option Weiches Scrollen eingeschaltet ist (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

SoftHyphen-Eigenschaft

Bestimmt oder ermittelt die Ansichtsoption für manuell eingefügte weiche Trennstellen.

KLASSE:
ViewSettings

SYNTAX:
SoftHyphen

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob manuell eingefügte weiche Trennstellen angezeigt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

SolidHandles-Eigenschaft

Bestimmt und ermittelt die Art der Anzeige der Handles, zum Beispiel an Textrahmen, Zeichenobjekten und Kontrollfeldern.

KLASSE:
ViewSettings

SYNTAX:
SolidHandles

RÜCKGABE:
Boolean

PARAMETER:
True = Farbige Handles werden gezeigt

False = Schwarze Handles werden gezeigt

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Sort-Methode

Sortiert den selektierten Bereich.

KLASSE:
BaseText

SYNTAX:
Sort(Order As Long, Direction As Long, Delimiter As Long, Col1 As Long, Typ1 As Long, Col2 As Long, Typ2 As Long, Col3 As Long, Typ3 As Long)

PARAMETER:
[Order]: Long-Integer-Ausdruck, der die Sortierreihenfolge festlegt. Order können die folgenden Werte zugewiesen werden:

0: Aufsteigende Sortierreihenfolge.

1: Absteigende Sortierreihenfolge.

Dieser Parameter ist optional. Wird er nicht gesetzt, wird der Standardwert Order = 0 genommen.

[Direction]: Long-Integer-Ausdruck, der die Sortierrichtung innerhalb einer Tabelle angibt:

0: Sortiert die Spalten einer Tabelle.

1: Sortiert die Zeilen einer Tabelle.

Wird dieser Parameter nicht gesetzt, wird der Standardwert Direction = 0 genommen.

[Delimiter]: Long-Integer-Ausdruck, der das Trennzeichen zum Sortieren von Absätzen bestimmt:

0: Der Tabulator wird als Trennzeichen verwendet.

1: Ein selbstdefiniertes Zeichen wird als Trennzeichen verwendet.

Wird dieser Parameter nicht gesetzt, wird in der Standardeinstellung der Tabulator als Trennzeichen verwendet.

[Col1/Col2/Col3]: Long-Integer-Ausdruck, der die Spalte des ersten/zweiten/dritten Sortierschlüssels angibt, nach dem die Tabelle sortiert werden soll.

Wird dieser Parameter nicht übergeben, wird als Standardwert 0 genommen, also kein Sortierschlüssel definiert. Wird eine Spalte angegeben, muß der zugehörige Sortiertyp unbedingt mit angegeben werden.

[Typ1/Typ2/Typ3]: Long-Integer-Ausdruck, der den Sortiertyp des ersten/zweiten/dritten Sortierschlüssels angibt, nach dem die Tabelle sortiert werden soll.

0: Alphanumerische Sortierung

1: Numerische Sortierung

Dieser Parameter ist nur dann erforderlich, wenn der zugehörige Sortierschlüssel mittels Col1/Col2/Col3 übergeben wurde.

Beispiel:

```
Sub ExampleTableSort
Const sTable="Tabelle 100"
Const sT1="Wörter"
Const sT2="Zahlen"
Documents.Add("sWriter4")
REM Tabelle zum Sortieren erstellen
Selection.InsertNamedTable( sTable, 4,2 )
Selection.GotoTable(sTable): Selection.Insert(sT1)
Selection.JumpToNextCell : Selection.Insert(sT2)
Selection.JumpToNextCell : Selection.Insert("ABC")
Selection.JumpToNextCell : Selection.Insert("53")
Selection.JumpToNextCell : Selection.Insert("XYZ")
Selection.JumpToNextCell : Selection.Insert("234")
Selection.JumpToNextCell : Selection.Insert("GHT")
Selection.JumpToNextCell : Selection.Insert("1254")
REM Nach erster Spalte sortieren
ActiveWindow.ShowCursor = False REM Anzeige ausschalten
Selection.GotoTable(sTable)
Wait 3000
Selection.EntireColumn
Selection.GoRight 1, TRUE
Selection.Sort( 0,1,,1,0)
REM nach zweiter Spalte Sortieren
Selection.GotoTable(sTable)
Wait 2000
```

```
Selection.EntireColumn
Selection.GoRight 1, TRUE
Selection.Sort( 0,1,,2,1)
ActiveWindow.ShowCursor = True REM Anzeige einschalten
End Sub
```

SortDialog-Methode

Ruft den Dialog Sortieren auf. Die Sortierung bezieht sich auf den selektierten Bereich.

Klasse:
BaseText

Syntax:
SortDialog()

Beispiel:

```
Sub ExampleSortDialog
Documents.Add("swriter4")
REM Daten eingeben, selektieren und Dialog starten
With ActiveWindow.Selection
.Insert("Text : 1")
.InsertPara
.Insert("Text : 2")
.InsertPara
.SelectALL
.SortDialog
End With
End Sub
```

Space-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von Leerzeichen.

Klasse:
ViewSettings

Syntax:
Space

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob Leerzeichen durch einen kleinen Punkt dargestellt angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Spacing-Eigenschaft

Bestimmt oder ermittelt die Laufweite des aktuellen Fonts. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetSpacing gesetzte Laufweite abgefragt.

Eine Abfrage der Spacing-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsSpacing bestimmt wurde, daß nach einer bestimmten Fontstärke gesucht werden soll, d. h. IsSpacing = 2.

Siehe auch (SearchAttributes, ReplaceAttributes): SetSpacing, IsSpacing.

Siehe auch (Font):

PairKerning.

KLASSE:
ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Spacing

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Value]: Integer-Ausdruck, der die Laufweite in Twips angibt. Ein positiver Wert vergrößert den Buchstabenabstand, ein negativer Wert verkleinert diesen. Mit 0 wird die Standardlaufweite angegeben.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
```

```
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
```

```
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Spelling-Methode

Führt eine Rechtschreibkontrolle ab der Position der Schreibmarke durch.

KLASSE:
TextDrawText, BaseTextEditView

SYNTAX:
Spelling()

BEISPIEL:

```
Sub ExampleSpelling
Const sMessage = "Dieser Satz wird gleich kontrolliert"
Documents.Add("sWriter4")
ActiveWindow.Insert sMessage
Selection.JumpToStartOfDoc
ActiveWindow.Spelling
End Sub
```

SpellOnline-Eigenschaft

Bestimmt oder ermittelt, ob die Online-Rechtschreibprüfung aktiv ist.

Siehe auch: HideSpellMark

KLASSE:
TextDrawText, StarWriter

SYNTAX:
SpellOnline

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Online-Rechtschreibprüfung ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
sub ExampleSpellOmline
Documents.Add "sWriter4"
Selection.Insert "Quaeramus aliquod non in speciem bonum. "
Activemodule.SpellOnline = True
Activemodule.HideSpellMark = True
Wait 5000
Activemodule.HideSpellMark = False
Selection.Insert " Jetzt wurden die Markierungen für die unbekannten Wörter aktiviert."
end sub
```

SplitCell-Methode

Unterteilt die selektierte Zelle in mehrere Zellen. Ohne Übergabeparameter wird der Dialog ZELLEN TEILEN aufgerufen.

KLASSE:
BaseTextTable

SYNTAX:
SplitCell(eDir As Boolean, Count As Long)

PARAMETER:
eDir: Boolescher Ausdruck, der angibt, ob die Zelle vertikal in mehrere Spalten (True) oder horizonal in mehrere Zeilen (False) aufgeteilt werden soll.

Count: Anzahl der neuen Zellen.

BEISPIEL:

```
Sub ExampleSplitCell
Documents.Add("sWriter4")
Selection.InsertTable( 4 , 4 )
REM Die Zelle A1 wird vertikal geteilt
Selection.SplitCell(TRUE, 1)
End Sub
```

StartAutoCorrect-Methode

Diese Methode fügt das per Parameter übergebene Zeichen ein und führt dann eine Autokorrektur durch.

KLASSE:
BaseText

Syntax:
StartAutoCorrect(Char As String)

Parameter:
[Char]: Das einzufügende Zeichen. Wird dieser Parameter nicht gesetzt, dann wird ein Leerzeichen eingefügt.

Beispiel:

```
Sub ExampleStartAutoCorrect
Dim sControl As String
Const sAutoKorr="mfg"
Documents.Add("sWriter4")
Selection.Insert("Mit folgendem Kürzel wird eine AutoKorrektur
durchgeführt. " & sAutoKorr)
Wait 2000
Selection.StartAutoCorrect()
End Sub
```

StatusLine-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige der Statusleiste.

Klasse:
ViewSettings

Syntax:
StatusLine

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Statusleiste ein- (True) oder ausgeblendet (False) werden soll.

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

StrikeOut-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Striche, mit denen der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen durchgestrichen dargestellt werden sollen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetStrikeOut gesetzte Art der Durchstreichung abgefragt.

Eine Abfrage der StrikeOut-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsStrikeOut bestimmt wurde, daß nach einer bestimmten Art der Durchstreichung gesucht werden soll, d. h. IsStrikeOut = 2.

Siehe auch: SetStrikeOut, IsStrikeOut.

KLASSE:
TextDrawText, ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
StrikeOut

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Strikes]: Integer- Ausdruck, der die Anzahl der Striche angibt, mit denen der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen durchgestrichen dargestellt werden sollen:

0 - Keine Durchstreichung

1 - Einfach

2 - Doppelt

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
```

```
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
```

```
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Style-Eigenschaft

Bestimmt oder ermittelt das Hintergrundmuster des selektierten Textes, bzw. das Füllmuster des Schattens, der dem selektierten Text zugewiesen ist.

Siehe auch: BackColor.

KLASSE:
TextPage, Shadow, BaseTextFrame, BaseTextEditView

SYNTAX:
Style

RÜCKGABE:
Integer

PARAMETER:
[= Pattern]: Integer-Ausdruck, der angibt, welches Muster im Hintergrund des selektierten Textes erscheinen soll bzw. welches Füllmuster dem Schatten des selektierten Textes zugewiesen werden soll. Pattern können die folgenden Muster zugewiesen werden:

Wert	Hintergrund / Schatten
0	Der Hintergrund oder der Schatten wird transparent dargestellt.
1	Ein Hintergrund wird weiß, ein Schatten schwarz dargestellt.
2	
3	
4	
5	
6	
7	

Wert	Hintergrund / Schatten
8	
9	
10	

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

StyleApply-Methode

Wendet eine bestimmte Vorlage auf die aktuelle Selektion an.

Bei erfolgreicher Ausführung wird der Wert 1 zurückgegeben, andernfalls eine 0.

Ohne Übergabeparameter wird der Stylist geöffnet.

Bei Aufruf dieser Methode dürfen Rahmenvorlagen nur bei Selektionen von Rahmen, Grafiken und OLE-Objekten angewendet werden, Zeichen- und Absatzvorlagen nur in Textselektionen.

KLASSE:
BaseTextSelection

SYNTAX:
StyleApply(Template As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Template: Name der gewünschten Vorlage.

Family: Gibt den Vorlagenbereich an, aus dem die Vorlage genommen werden soll. Folgende Werte sind möglich:

1 - Zeichenvorlagen

2 - Absatzvorlagen

4 - Rahmenvorlagen

8 - Seitenvorlagen

BEISPIEL:

```
Sub ExampleStyles
REM Das Beispiel legt drei Absatzvorlagen an, von denen eine
ein Kind der anderen ist.
Const sMessage= "Anwendung der Vorlage: "
Const sDefaultTemp="Standard"
Const sTemp1="Stil_Rot"
Const sTemp1a="Stil_Rot_Zentriert"
Const sTemp2="Stil_Zentriert"
Dim sStyle As String
Documents.Add("sWriter4")
with ActiveWindow.Selection
.ResetAttributes      REM Für den aktuellen Absatz die harten
Attribute löschen
.StyleApply( sDefaultTemp ,2)    REM Absatz auf Standard setzen
sStyle = .StyleApplyState
if sStyle <> sDefaultTemp then     REM ist tatsächlich auf
Standard  ?
MsgBox("Konnte die Absatzvorlage nicht
zurücksetzen",64,"Warnung")  REM gibt Warnung aus
end if
.StyleNewByExample( sTemp1 ,2)    REM setzt die Absatzvorlage
zurück
.Interior.BackColor = RGB (255,127,127)
.Interior.Style = 1
.StyleUpdateByExample( sTemp1 ,2)  REM setzt die Vorlage auf
rot
.ResetAttributes      REM Zelle auf Standard setzen
.StyleApply( sDefaultTemp ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp2 ,2)
.ResetAttributes
.StyleApply( sTemp1 ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp1a ,2)  REM Kind-Vorlage von sTemp1a ,
mit zusätzlich zentriertem Text
.ResetAttributes
REM jetzt werden die Vorlagen angewendet
.Insert( sMessage & sTemp1 )
```

```
.StyleApply( sTemp1 ,2)
.InsertPara
.Insert(sMessage & sTemp1a )
.StyleApply( sTemp1a ,2)
.InsertPara
.Insert(sMessage & sTemp2 )
.StyleApply( sTemp2 ,2)
.InsertPara
end with
End Sub
```

StyleApplyState-Eigenschaft

Die Eigenschaft liefert den Namen der aktuellen Absatz- oder Rahmenvorlage.

KLASSE:
BaseTextSelection

SYNTAX:
StyleApplyState

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleStyles
REM Das Beispiel legt drei Absatzvorlagen an, von denen eine
ein Kind der anderen ist.
Const sMessage= "Anwendung der Vorlage: "
Const sDefaultTemp="Standard"
Const sTemp1="Stil_Rot"
Const sTemp1a="Stil_Rot_Zentriert"
Const sTemp2="Stil_Zentriert"
Dim sStyle As String
Documents.Add("sWriter4")
with ActiveWindow.Selection
.ResetAttributes      REM Für den aktuellen Absatz die harten
Attribute löschen
.StyleApply( sDefaultTemp ,2)    REM Absatz auf Standard setzen
sStyle = .StyleApplyState
if sStyle <> sDefaultTemp then     REM ist tatsächlich auf
Standard  ?
MsgBox("Konnte die Absatzvorlage nicht
zurücksetzen",64,"Warnung")  REM gibt Warnung aus
end if
.StyleNewByExample( sTemp1 ,2)    REM setzt die Absatzvorlage
zurück
.Interior.BackColor = RGB (255,127,127)
```

```
.Interior.Style = 1
.StyleUpdateByExample( sTemp1 ,2)  REM setzt die Vorlage auf
rot
.ResetAttributes      REM Zelle auf Standard setzen
.StyleApply( sDefaultTemp ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp2 ,2)
.ResetAttributes
.StyleApply( sTemp1 ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp1a ,2)  REM Kind-Vorlage von sTemp1a ,
mit zusätzlich zentriertem Text
.ResetAttributes
   REM jetzt werden die Vorlagen angewendet
.Insert( sMessage & sTemp1 )
.StyleApply( sTemp1 ,2)
.InsertPara
.Insert(sMessage & sTemp1a )
.StyleApply( sTemp1a ,2)
.InsertPara
.Insert(sMessage & sTemp2 )
.StyleApply( sTemp2 ,2)
.InsertPara
end with
End Sub
```

StyleCatalog-Methode

Ruft den Vorlagenkatalog auf.

KLASSE:
TextDrawText, BaseTextDrawBase

SYNTAX:
StyleCatalog()

BEISPIEL:

```
Sub ExampleStyleCatalog
Documents.Add("swriter4")
Selection.Insert("Hier können Sie Ihre Vorlagen bearbeiten.")
ActiveWindow.StyleCatalog
End Sub
```

StyleName-Eigenschaft

Bestimmt oder ermittelt den Schriftstil des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetFont gesetzte Schriftstil abgefragt.

Siehe auch: SetFont, IsFont, Pitch, CharSet, Family, FamilyName.

KLASSE:
TextDrawFont, ReplaceAttributes, SearchAttributes, WebSourceView, TextFont

SYNTAX:
StyleName

RÜCKGABE:
String

PARAMETER:
Bei Text-Objekten:

[= FontStyle]: Zeichenfolgeausdruck, der den Schriftstil bestimmt. FontStyle können die folgenden Werte zugewiesen werden:

Leerstring: Voreinstellung. Der Text wird normal dargestellt.

Bold: Die Schrift wird fett dargestellt.

Italic: Die Schrift wird kursiv dargestellt.

Falls ein Font in dem System nicht vorhanden ist, in dem das Dokument geladen wurde, dann wird ein ähnlicher Font genommen. Diese Eigenschaft dient nur als Hilfsmittel dazu. Setzen Sie bei dieser Eigenschaft BOLD, dann weiß das System, daß der Originalfont ein fetter Font war.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleStyleName
Documents.Add "sWriter4"
Selection.Font.StyleName = "Bold"
print "StyleName liefert : " & Selection.Font.StyleName
End Sub
```

StyleNewByExample-Methode

Erzeugt eine neue Vorlage aus der aktuellen Selektion. Ohne Übergabeparameter wird der Dialog VORLAGE ERZEUGEN geöffnet.

KLASSE:
BaseTextSelection

SYNTAX:
StyleNewByExample(Param As String, Family As Long)

PARAMETER:
Param: Name der neuen Vorlage. Dieser Name darf noch nicht existieren.

Family: Gibt den Vorlagenbereich an, in dem die Vorlage angelegt werden soll. Folgende Werte sind möglich:

1 - Zeichenvorlagen

2 - Absatzvorlagen

4 - Rahmenvorlagen

8 - Seitenvorlagen

BEISPIEL:

```
Sub ExampleStyles
REM Das Beispiel legt drei Absatzvorlagen an, von denen eine
ein Kind der anderen ist.
Const sMessage= "Anwendung der Vorlage: "
Const sDefaultTemp="Standard"
Const sTemp1="Stil_Rot"
Const sTemp1a="Stil_Rot_Zentriert"
Const sTemp2="Stil_Zentriert"
Dim sStyle As String
Documents.Add("sWriter4")
with ActiveWindow.Selection
.ResetAttributes      REM Für den aktuellen Absatz die harten
Attribute löschen
.StyleApply( sDefaultTemp ,2)    REM Absatz auf Standard setzen
sStyle = .StyleApplyState
if sStyle <> sDefaultTemp then     REM ist tatsächlich auf
Standard  ?
MsgBox("Konnte die Absatzvorlage nicht
zurücksetzen",64,"Warnung")  REM gibt Warnung aus
end if
.StyleNewByExample( sTemp1 ,2)    REM setzt die Absatzvorlage
zurück
.Interior.BackColor = RGB (255,127,127)
.Interior.Style = 1
.StyleUpdateByExample( sTemp1 ,2)  REM setzt die Vorlage auf
rot
.ResetAttributes      REM Zelle auf Standard setzen
.StyleApply( sDefaultTemp ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp2 ,2)
.ResetAttributes
.StyleApply( sTemp1 ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp1a ,2)  REM Kind-Vorlage von sTemp1a ,
mit zusätzlich zentriertem Text
.ResetAttributes
  REM jetzt werden die Vorlagen angewendet
.Insert( sMessage & sTemp1 )
```

```
.StyleApply( sTemp1 ,2)
.InsertPara
.Insert(sMessage & sTemp1a )
.StyleApply( sTemp1a ,2)
.InsertPara
.Insert(sMessage & sTemp2 )
.StyleApply( sTemp2 ,2)
.InsertPara
end with
End Sub
```

StyleUpdateByExample-Methode

Mit dieser Methode wird die Vorlage aus der Selektion aktualisiert.

Zur Aktualisierung von Absatzvorlagen ist diese Methode auch ohne Übergabeparameter aufrufbar: Ist ein Absatz selektiert und wird diese Methode ohne Parameter aufgerufen, dann werden die aktuellen Einstellungen des Absatzes in die zugehörige Absatzvorlage übernommen.

KLASSE:
BaseTextSelection

SYNTAX:
StyleUpdateByExample(Param As String, Family As Long)

PARAMETER:
Param: Name der Vorlage, die aus der Selektion aktualisiert werden soll. Die Vorlage muß bereits existieren.

Family: Gibt den Vorlagenbereich an, in dem die Vorlage abgelegt ist. Folgende Werte sind möglich:

1 - Zeichenvorlagen

2 - Absatzvorlagen

4 - Rahmenvorlagen

8 - Seitenvorlagen

BEISPIEL:

```
Sub ExampleStyles
REM Das Beispiel legt drei Absatzvorlagen an, von denen eine
ein Kind der anderen ist.
Const sMessage= "Anwendung der Vorlage: "
Const sDefaultTemp="Standard"
Const sTemp1="Stil_Rot"
Const sTemp1a="Stil_Rot_Zentriert"
Const sTemp2="Stil_Zentriert"
Dim sStyle As String
```

```
Documents.Add("sWriter4")
with ActiveWindow.Selection
.ResetAttributes      REM Für den aktuellen Absatz die harten
Attribute löschen
.StyleApply( sDefaultTemp ,2)    REM Absatz auf Standard setzen
sStyle = .StyleApplyState
if sStyle <> sDefaultTemp then     REM ist tatsächlich auf
Standard  ?
MsgBox("Konnte die Absatzvorlage nicht
zurücksetzen",64,"Warnung")  REM gibt Warnung aus
end if
.StyleNewByExample( sTemp1 ,2)    REM setzt die Absatzvorlage
zurück
.Interior.BackColor = RGB (255,127,127)
.Interior.Style = 1
.StyleUpdateByExample( sTemp1 ,2)  REM setzt die Vorlage auf
rot
.ResetAttributes      REM Zelle auf Standard setzen
.StyleApply( sDefaultTemp ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp2 ,2)
.ResetAttributes
.StyleApply( sTemp1 ,2)
.Paragraph.Alignment = 3    REM Text zentriert
.StyleNewByExample( sTemp1a ,2)  REM Kind-Vorlage von sTemp1a ,
mit zusätzlich zentriertem Text
.ResetAttributes
  REM jetzt werden die Vorlagen angewendet
.Insert( sMessage & sTemp1 )
.StyleApply( sTemp1 ,2)
.InsertPara
.Insert(sMessage & sTemp1a )
.StyleApply( sTemp1a ,2)
.InsertPara
.Insert(sMessage & sTemp2 )
.StyleApply( sTemp2 ,2)
.InsertPara
end with
End Sub
```

StyleWatercanMode-Eigenschaft

Ermittelt, ob der Gießkannenmodus des Stylist aktiv ist.

KLASSE:
BaseTextSelection

SYNTAX:
StyleWatercanMode

RÜCKGABE:
Boolean

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleWatercan
Dim bCan As Boolean
Const sMessage = "Status des Gießkannenmodus : "
Documents.Add( "sWriter4" )
With ActiveWindow.Selection
.WatercanStyleModeOnOff("Standard" ,2 )
bCan = .StyleWatercanMode
MsgBox sMessage & bCan, 64, "StyleWatercanMode"
wait 2000
.WatercanStyleModeOnOff("" ,2 )
End With
End Sub
```

SubScript-Eigenschaft

Bestimmt oder ermittelt, ob der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen tiefergestellt werden sollen.

KLASSE:
TextDrawText, TextFont

SYNTAX:
SubScript

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen tiefergestellt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
```

```
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
```

```
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

SuperScript-Eigenschaft

Bestimmt oder ermittelt, ob der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen hochgestellt werden sollen.

KLASSE:
TextDrawText, TextFont

SYNTAX:
SuperScript

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der selektierte Text oder die an der Schreibmarke einzufügenden Zeichen hochgestellt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
```

```
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Surround-Eigenschaft

Steuert das Umlaufverhalten an Zeichenobjekten.

KLASSE:
BaseTextFrame

SYNTAX:
Surround

RÜCKGABE:
Integer

PARAMETER:
Mögliche Werte:

1 - kein Umlauf

2 - paralleler Umlauf

4 - Umlauf nur links

5 - Umlauf nur rechts

BEISPIEL:

```
sub ExampleFrameSurround
Const sText="Das ist ein Text, der um die Rahmen fließt. "
Documents.Add("sWriter4")
For iCount=1 to 25
   Selection.Insert(sText)
   if iCount Mod 5 = 0 then Selection.InsertPara
Next
Selection.InsertFrame(2, "5cm", "3cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen1.")
Selection.GotoAnchor
Selection.InsertFrame(1, "4cm", "4cm", "4cm", "4cm")
Selection.Insert("Das ist Text in Rahmen2.")
Selection.Surround = 4
Selection.AnchorType = 2
Selection.GoldCut = TRUE
Selection.Opaque = FALS
Selection.Printable = TRUE
Selection.SendToBack
Selection.BringToFront
end sub
```

SwEditOptions-Methode

Ruft den Dialog OPTIONEN TEXTDOKUMENT auf.

KLASSE:
StarWriter

SYNTAX:
SwEditOptions()

BEISPIEL:

```
Sub ExampleStarWriter
Application.StarWriter.SwEditOptions
End Sub
```

Eigenschaften und Methoden mit "T"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "T" beginnen.

Tab-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von Tabulatorzeichen.

KLASSE:
ViewSettings

SYNTAX:
Tab

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Tabulatorzeichen angezeigt werden sollen (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

Table-Eigenschaft

Mit dieser Eigenschaft kann die Darstellung der Tabellen im Dokument gewählt werden.

KLASSE:
ViewSettings

SYNTAX:
Table

RÜCKGABE:
Boolean

PARAMETER:
True: Tabellen werden angezeigt.

False: Tabellen werden nicht angezeigt.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
```

```
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

TableBounds-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige von Tabellenbegrenzungen im Dokument.

KLASSE:
ViewSettings

SYNTAX:
TableBounds

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob Tabellenbegrenzungen angezeigt werden sollen (True) oder nicht (False).

Beispiel:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

TableDialog-Methode

Ruft bei selektierter Tabelle den Dialog TABELLENFORMAT auf, mit dessen Hilfe die Einstellungen für die aktuelle Tabelle gesetzt werden können.

KLASSE:
BaseTextTable

SYNTAX:
TableDialog()

Beispiel:

```
Sub ExampleTableDialog
Documents.Add("sWriter4")
ActiveWindow.InsertTable(4,5)
Selection.TableDialog
end sub
```

TableModeFix-Methode

Schaltet die Tastatursteuerung für Tabellen in den Modus "fix".

Klasse:
BaseTextTable

Syntax:
TableModeFix()

Beispiel:

```
Sub ExampleTableMode
Const sMessage_1 = "Modus der Tastatursteuerung ist : "
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.SetColumnWidth( 0, "1.5cm" )
Selection.GotoCell("A2", FALSE)
Selection.TableModeFix
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fixREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fix proportionalREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM variabelREM "
End Sub
```

TableModeFixProp-Methode

Schaltet die Tastatursteuerung für Tabellen in den Modus "fix proportional".

Klasse:
BaseTextTable

Syntax:
TableModeFixProp()

Beispiel:

```
Sub ExampleTableMode
Const sMessage_1 = "Modus der Tastatursteuerung ist : "
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.SetColumnWidth( 0, "1.5cm" )
Selection.GotoCell("A2", FALSE)
Selection.TableModeFix
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fixREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fix proportionalREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM variabelREM "
End Sub
```

TableModeVariable-Methode

Schaltet die Tastatursteuerung für Tabellen in den Modus "variabel".

Klasse:
BaseTextTable

Syntax:
TableModeVariable()

Beispiel:

```
Sub ExampleTableMode
Const sMessage_1 = "Modus der Tastatursteuerung ist : "
Documents.Add("sWriter4")
Selection.InsertTable( 3 , 2 )
Selection.SetColumnWidth( 0, "1.5cm" )
Selection.GotoCell("A2", FALSE)
Selection.TableModeFix
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fixREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM fix proportionalREM "
Selection.JumpToNextCell
Selection.TableModeFixProp
```

```
Selection.Insert "12"
Selection.JumpToNextCell
Selection.Insert sMessage_1 & "REM variabelREM "
End Sub
```

TableName-Eigenschaft

Bestimmt oder ermittelt die Bezeichnung der selektierten Tabelle.

KLASSE:
BaseTextTable

SYNTAX:
TableName

RÜCKGABE:
String

PARAMETER:
[= TabName]: Zeichenfolgeausdruck, der die Bezeichnung der selektierten Tabelle angibt.

BEISPIEL:

```
Sub ExampleInsertNamedTable
Const sTableName = "TestTabelle"
Const sMessage = "Die Tabelle hat folgenden Namen : "
Documents.Add "sWriter4"
Selection.InsertNamedTable(sTableName, 3, 3)
Print sMessage & Selection.TableName
End Sub
```

TableNumberFormatDialog-Methode

Ruft den Dialog ZAHLENFORMAT BEARBEITEN auf.

KLASSE:
BaseTextTable

SYNTAX:
TableNumberFormatDialog()

BEISPIEL:

```
Sub ExampleTableNumberDialog
Documents.Add("sWriter4")
Selection.InsertTable( 4, 5)
ActiveWindow.Table.TableNumberFormatDialog
end sub
```

TableOptimizeMinMax-Methode

Paßt die Spaltenbreiten einer Tabelle an den Inhalt an.

KLASSE:
BaseTextTable

SYNTAX:
TableOptimizeMinMax()

BEISPIEL:

```
Sub ExampleTableOptimizeMinMax
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
Selection.Insert "In dieser Tabelle wird gleich"
Selection.JumpToNextCell
Selection.Insert "die Breite der Zellen optimal angepaßt"
Selection.TableOptimizeMinMax
End Sub
```

Tables-Eigenschaft

Ermöglicht den Zugriff auf die Tables-Collection.

Über Tables(Index) wird die Tabelle an der Position "Index" selektiert und das Selection.Object übergeben. Index ist ein Integerausdruck.

Tables.Remove(Index) löscht die Tabelle an der Position "Index".

Tables.Count liefert die Anzahl der im Dokument vorhandenen Tabellen.

Die erste Tabelle im Dokument wird über den Index 1 angesprochen.

KLASSE:
BaseTextEditView

SYNTAX:
Tables

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleTables
Const sMessage_1 = "Der Name der Tabelle "
Const sMessage_2 = " lautet : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
```

```
Selection.InsertPara
Selection.InsertPara
Selection.GoUp( 1, FALSE )
Selection.InsertTable(3, 3)
Selection.JumpToStartOfDoc
Selection.InsertTable(2, 2)
For iCount=1 To Selection.Tables.Count
Print sMessage_1 & iCount & sMessage_2 &
Selection.Tables(iCount).TableName
Next iCount
End Sub
```

Tabstops-Eigenschaft

Das TabStops-Interface enthält die kompletten Information für die Tabulatoren.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
Tabstops

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleTabStops
Dim iCount As Integer, iTabCount As Integer
Dim sText As String, sTAB As String, sBrandNewStyle As String
Dim s1 As String,s2 As String,s3 As String,s4 As String
sTAB = CHR(9)REM Der Tabulator
sBrandNewStyle = "BrandNewStyle"
Documents.Add("sWriter4")
With ActiveWindow.Selection.Paragraph.Tabstops      REM erst
einmal drei Tabulatoren setzen
REM rechtbündiger Tabulator
.Insert( 1700, "R" )
REM Dezimaler Tabular, mit REM KommaREM  als dezimalen Trenn-
zeichen und Punktfüllung
.Insert( 3000, "D", ",", ".")
REM linksbündiger Tabulator
.Insert( 3370, "L" )
end with
With ActiveWindow.Selection
REM einfacher als jeden Abstz zu formatieren ist es ist es,
REM eine Vorlage zu definieren und anzuwenden
.StyleNewByExample(sBrandNewStyle,2)
.Insert( sTAB & "Auto" & sTAB & "65,54" & sTAB & "DM"   )  REM
Punkt und Komma sind relevant
.InsertPara
.StyleApply(sBrandNewStyles,2)
```

```
.Insert( sTAB & "Haus" & sTAB & "4.565" & sTAB & "DM"   )
.InsertPara
.StyleApply(sBrandNewStyle,2)
end with
REM Kontrolle: Tabulatoren ausgeben
sText = ""
With ActiveWindow.Selection.Paragraph.Tabstops
iTabCount = .Count
for iCount =  0 to iTabCount-1
s1 = .GetPosition( iCount )
s2 = .GetAdjust( iCount )
s3 = .GetDecimal( iCount )
s4 = .GetFillChar( iCount )
sText  = sText &  "Position: " & s1 & "  Justierung: " & s2 & "
Dezimaltrennzeichen: " & s3 &"  Füllzeichen: " & s4 & Chr(13)
next
end with
MsgBox(sText,64,"" & iTabCount & " Tabulatoren")    REM Hier
die Tabulatoren ausgeben
end sub
```

Tag-Eigenschaft

Mit Hilfe dieser Eigenschaft können zusätzliche Daten zu einem Kontrollfeld in Form einer Zeichenkette abgelegt werden. Verwenden Sie diese Eigenschaft, um einem Element bestimmte Informationen zuzuweisen, die es beispielsweise identifizieren, oder die Ihnen als Anmerkung zu diesem Element dienen.

KLASSE:
BaseTextDrawControl

SYNTAX:
Tag

RÜCKGABE:
String

BEISPIEL:

```
Sub ExampleControlTag
Dim sControl As String
Documents.Add("sWriter4")
Activewindow.InsertVCControl( 8, "3cm", "3cm", "3cm", "5cm")
sControl = Activewindow.Controls(1).ObjectName
Activewindow.Controls(sControl).Tag = "Der Tag ist gegangen,
..."
Activewindow.Controls(sControl).Multiline = TRUE
end sub
```

TargetFrame-Eigenschaft

Bestimmt oder ermittelt, in welchem Frame das über die URL-Eigenschaft zu ladende Dokument in der Ansicht geöffnet werden soll.

KLASSE:
TextFont, BaseTextDrawControl

SYNTAX:
TargetFrame

RÜCKGABE:
String

PARAMETER:
[= Frame]: Zeichenfolgeausdruck, der den Namen des Frames angibt, in dem das Dokument geladen werden soll.

BEISPIEL:

```
Sub ExampleHyperLinkScript
Dim sScript As String
Documents.Add "sWriter4"
Selection.Insert "Stardivision.de"
Selection.GotoStartOfLine TRUE

Selection.Font.URL = "http://www.Stardivision.de"
Selection.Font.SetHyperlinkScript(1, "alert(REM JavaScrip-
tREM )")
Selection.Font.TargetFrame = "_blank"
sScript = Selection.Font.GetHyperlinkScript(1)
Print "Folgendes JavaScript ist mit dem Hyperlink verknüpft : "
& sScript
End Sub
```

TextAttributes-Methode

Öffnet den Zeichenattribut-Dialog für Zeichenobjekte.

KLASSE:
TextDraw, TextDrawText

SYNTAX:
TextAttributes()

BEISPIEL:

```
REM Muß ein bereits definiertes Zeichen-Objekt sein.
Selection.TextAttributes
```

TextWrap-Methode

Ruft den Dialog UMLAUF für Zeichenobjekte oder Kontrollfelder auf.

KLASSE:
BaseTextDrawBase

SYNTAX:
TextWrap()

BEISPIEL:

```
Sub ExampleTextWrapDialog
Documents.Add( "sWriter4" )
REM Einen Commandbutton auf dem Dokument erstellen
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
ActiveWindow.Selection.TextWrap
End Sub
```

ThesaurusDialog-Methode

Ruft den Thesaurus auf, mit dem für das selektierte Wort ein Synonom gesucht werden kann.

KLASSE:
TextDrawText, BaseTextEditView

SYNTAX:
ThesaurusDialog()

BEISPIEL:

```
Sub ExampleThesaurusdialog
Documents.Add("swriter4")
ActiveWindow.Selection.Insert("Das Wort wird thesauriert.")
ActiveWindow.Selection.GoLeft 4
ActiveWindow.Thesaurusdialog
end sub
```

Title-Methode

Liefert den Titel des über die Position angegebenen Teils der Globaldokument-Collection.

Beinhaltet das Globaldokument an der angegebenen Position einen Bereich mit einer Dateiverknüpfung, so wird der Titel des Bereichs zurückgegeben.

Für ein Verzeichnis wird der Verzeichnistitel zurückgeliefert. Handelt es sich um einen Textabschnitt, wird ein Leerstring zuückgegeben.

KLASSE:
GlobalContents

SYNTAX:
Title(Position As Long)

RÜCKGABE:
String

PARAMETER:
Position: Ist die Positionsnummer innerhalb der Globaldokument-Collection, von der der Titel erfragt werden soll.

BEISPIEL:

```
Sub ExampleGlobalContents
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sIndexesName = "Stichwortverzeichnis"
Const sGlobalDocName = "\GLOBAL.SGL"
Const sClientDocName = "\CLIENT.SDW"
Const sMessage_1 = "Der Titel des Globaldokumentsbereichs : "
Const sMessage_2 = " lautet : "
Const sMessage_3 = "Der Typ des Globaldokumentsbereichs : "
Dim iContentCount As Integer, iCount As Integer
Dim sGlobalDocPath As String, sClientPath As String
sGlobalDocPath = Application.PathSettings.Work & sGlobalDocName
sClientPath = Application.PathSettings.Work & sClientDocName
Call CreateGlobalClientDoc( sClientPath )
Documents.Add( "sWriter4" )
ActiveDocument.NewGlobalDoc( sGlobalDocPath )
REM Dokument mit dem Verzeichniseinträgen einfügen
Activewindow.GlobalContents.InsertDocument( 1, sClientPath )
REM Fügt ein Inhaltsverzeichnis ein
Activewindow.GlobalContents.InsertIndexesTOC(1, 2, sIndexe-
sTOCName, 20)
REM Benutzerverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesUserIndex(2, 5, sUse-
rIndexName, 5, sUserIndexName, sUserIndexName & "1" )
iContentCount = Activewindow.GlobalContents.count
REM Stichwortverzeichnis einfügen
Activewindow.GlobalContents.InsertIndexesIndex( iContentCount,
7, sIndexesName )
For iCount=1 To iContentCount
Print sMessage_1 & iCount & sMessage_2 &
Activewindow.Globalcontents.Title(iCount)
Print sMessage_3 & iCount & sMessage_2 &
Activewindow.GlobalContents.Type(iCount)
Next iCount
End Sub
Sub CreateGlobalClientDoc( sClientPath )
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
```

```
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sUserIndexName = "Benutzerverzeichnis "
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
ActiveDocument.Close TRUE, sClientPath
End Sub
```

Top-Eigenschaft

Bestimmt oder ermittelt den Wert für das Skalieren der Ansicht der selektierten Grafik an der oberen Seite. Negative Werte beschneiden die Ansicht der Grafik, positive verbreitern den Rand zwischen Rahmen und Grafik und verkleinern die Grafik dabei.

Siehe auch: Left, Right und Bottom.

KLASSE:
BaseTextGraphic

SYNTAX:
Top

RÜCKGABE:
Long

PARAMETER:
[= Cut]: Gibt an, um welchen Betrag die markierte Grafik an der oberen Seite verändert werden soll. Der Wert kann auch mit Angabe der gewünschten Maßeinheit als Zeichenfolge gesetzt werden.

BEISPIEL:

```
Sub ExampleGraphicCrop
Dim sGrfPath as String
Dim iCount As Integer
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Wait 3000
With Selection  REM in zwei Stufen wird das Bild verkleinert
For iCount = 2 to -1 Step -1
   .Bottom = Trim(Str(iCount) & "cm")
   .Top= Trim(Str(iCount) & "cm")
   .Left= Trim(Str(iCount) & "cm")
   .Right= Trim(Str(iCount) & "cm")
   Wait 3000
Next iCount
End With
Selection.Escape
End Sub
```

TopMargin-Eigenschaft

Bestimmt oder ermittelt den Abstand vom oberen Rand des selektierten oder aktiven Objekts zu seiner Umgebung. Dieser Abstand ist für verschiedene Objekte unterschiedlich definiert: Bei einem TextEditView-Objekt ist es der Abstand bis zur oberen Papierkante. Bei Objekten der Klassen TextFrame, TextTable, TextGraphic, TextOLEObject ist es der Abstand bis zur letzten, darüberliegenden Zeile des umlaufenden Textes oder des vorangehenden Absatzes. Ist ein Absatz selektiert (Paragraph, TextInTable, ListInTable), so liefert TopMargin den Abstand zwischen der ersten Zeile des selektierten und der letzten Textzeile des vorangehenden Absatzes.

Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetTopBottomMargin gesetzte Abstand abgefragt.

Eine Abfrage der TopMargin-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsTopBottomMargin bestimmt wurde, daß nach einem bestimmten Absatzabstand gesucht werden soll., d. h. IsTopBottomMargin = 2.

Siehe auch: BottomMargin, LeftMargin, RightMargin.

Nur bei SearchAttributes, ReplaceAttributes:

SetTopBottomMargin, IsTopBottomMargin.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
TopMargin

RÜCKGABE:
Long

PARAMETER:
[= Distance]: Long-Integer-Ausdruck, der den jeweiligen Abstand angibt. Der Wert kann auch mit Angabe der gewünschten Maßeinheit als Zeichenfolge gesetzt werden.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleParagraphMargin
REM Das Beispiel setzt den aktuellen Absatz optisch ab, z.B.
für Zitate.
Documents.Add("sWriter4")
ActiveWindow.Selection.Insert("Dieser Absatz erscheint beid-
seitig eingerückt und mit einem Abstand zum vorigen und
folgenden Absatz")
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert("Hier geht es weiter")
ActiveWindow.Selection.GoUp 1
With ActiveWindow.Selection.Paragraph
.LeftMargin  = "3cm"
.RightMargin = "3cm"
.TopMargin   = "1cm"
.BottomMargin   = "1cm"
End With
End Sub
```

TransformDialog-Methode

Ruft den Dialog Position und Größe für Zeichenobjekte und Kontrollfelder auf.

Klasse:
BaseTextDrawBase

Syntax:
TransformDialog()

Beispiel:

```
Sub ExampleTransformDialog
Documents.Add( "sWriter4" )
REM Einen Commandbutton auf dem Dokument erstellen
Activewindow.InsertVCControl( 2, "5cm", "4cm", "7cm", "3cm")
ActiveWindow.Selection.TransformDialog
End Sub
```

Transparent-Eigenschaft

Bestimmt oder ermittelt die Einstellung für einen transparenten Hintergrund oder Schatten.

Klasse:
TextPage, Shadow, BaseTextFrame, BaseTextEditView

Syntax:
Transparent

Rückgabe:
Boolean

Parameter:
[= Bool]: Boolescher Ausdruck, der angibt, ob der Hintergrund oder der Schatten transparent dargestellt werden soll (True) oder nicht (False).

Beispiel:

```
Sub ExampleStyle
Dim sGrfPath As String
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Documents.Add("sWriter4")
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "4cm")
Selection.Interior.Style = 2
Selection.Interior.Transparent = TRUE
Selection.Interior.Backcolor = RGB(0, 0, 255)
Selection.Interior.FillColor = RGB(0, 255,0)
Selection.GotoAnchor
Selection.InsertFrame(2, "7cm", "7cm", "4cm", "4cm")
Selection.Interior.GrfFilename = sGrfPath
```

```
Selection.Interior.GrfFiltername = "BMP - MS Windows"
Selection.Interior.GrfPosition = 10
End Sub
```

Type-Eigenschaft

Ermittelt den Typ des selektierten Kontrollfeldes.

KLASSE:
BaseTextDrawBase

SYNTAX:
Type

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

Rückgabe: Long-Integer-Ausdruck, der den Typ des Kontrollfeldes angibt. Folgende Werte sind möglich:

Wert	U91Typ
0	Markierfeld (CheckBox) im Dokument
1	Optionsfeld (OptionButton) im Dokument
2	Befehlsschaltfläche (CommandButton) im Dokument
3	Drehfeld (SpinButton) im Dokument
4	Bezeichnungsfeld (Label) im Dokument
5	Gruppierungsrahmen (FrameControl) im Dokument
6	Listenfeld (ListBox) im Dokument
7	Kombinationsfeld (ComboBox) im Dokument
8	Textfeld (TextBox) im Dokument
9	horizontale Bildlaufleiste (HScrollBar) im Dokument
10	vertikale Bildlaufleiste (VScrollBar) im Dokument
11	Markierfeld (CheckBox) im Dialog
12	Optionsfeld (OptionButton) im Dialog
13	Befehlsschaltfläche (CommandButton) im Dialog
14	Drehfeld (SpinButton) im Dialog
15	Bezeichnungsfeld (Label) im Dialog
16	Gruppierungsrahmen (FrameControl) im Dialog
17	Listenfeld (ListBox) im Dialog

Wert	U91Typ
18	Kombinationsfeld (ComboBox) im Dialog
19	Textfeld (TextBox) im Dialog
20	horizontale Bildlaufleiste (HScrollBar) im Dialog
21	vertikale Bildlaufleiste (VScrollBar) im Dialog
22	Dialog-Element
23	Vorschaufeld (Preview) im Dialog
24	URL-Schaltfläche (URL-Button) im Dialog
25	URL-Schaltfläche (URL-Button) im Dokument

Beispiel:

```
Sub ExampleControls
Const sValue = "Eintrag"
Const sMessage_1 = "Die Beschriftung vom Label lautet : "
Const sMessage_2 = "Bei der Listbox ist folgendes markiert : "
Dim iYPos As Integer, iXPos As Integer, iCount As Integer
Dim sControlName As String
iYPos = 5
iXPos = 2
Documents.Add "sWriter4"
For iCount=0 To 10
Activewindow.InsertVCControl( iCount, iXPos, iYPos, "3cm",
"1cm")
If Activewindow.Controls(iCount+1).Type = iCount Then
if Selection.Iscontrol = TRUE Then
Selection.Escape
If iXPos <= 14 Then
iXPos = iXPos + 3
Else
iXPos = 2
iYPos = iYPos + 3
End If
End If
End If
Next iCount
sControlName = Activewindow.Controls(5).ObjectName
Print sMessage_1 & Activewindow.Controls(sControlName).Caption
Selection.Escape
sControlName = Activewindow.Controls(7).ObjectName
Activewindow.Controls(sControlName).ControlObject.AddItem
sValue
Activewindow.Controls(sControlName).ControlObject.Listindex = 0
Print sMessage_2 & Activewindow.Controls(sControlName).Value
End Sub
```

Eigenschaften und Methoden mit "U"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "U" beginnen.

Underline-Eigenschaft

Bestimmt oder ermittelt, ob und wie die selektierten Zeichen unterstrichen werden sollen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft der mit SetUnderline gesetzte Wert abgefragt.

Eine Abfrage der Underline-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsUnderline bestimmt wurde, daß nach einer bestimmten Unterstreichung gesucht werden soll, d. h. IsUnderline = 2.

Siehe auch: SetUnderline, IsUnderline.

KLASSE:
TextDrawText, ReplaceAttributes, SearchAttributes, TextFont

SYNTAX:
Underline

RÜCKGABE:
Integer

PARAMETER:
Bei Text-Objekten:

[= Style]: Integer-Ausdruck, der die Art der Unterstreichung angibt. Folgende Werte sind möglich:

0 - Keine Unterstreichung

1 - Einfach

2 - Doppelt

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
```

```
Selection.Escape
Selection.Font.Bold = 0
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
```

```
Selection.Font.Spacing = 20
Selection.Escape
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

UnderlineDouble-Eigenschaft

Bestimmt oder ermittelt, ob das Zeichenattribut Doppelt unterstrichen aktiv ist.

Klasse:
TextFont

Syntax:
UnderlineDouble

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das Zeichenattribut ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub ExampleUnderlineDouble
Documents.Add("sWriter4")
Selection.Insert("Doppelt unterstrichen")
Selection.GotoStartOfLine TRUE
Selection.Font.Underlinedouble = TRUE
End Sub
```

Undo-Methode

Macht die letzte Aktion rückgängig.

KLASSE:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection, BaseTextEditView

SYNTAX:
Undo()

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
```

```
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UndoAction-Eigenschaft

Ermittelt die im Menü BEARBEITEN unter dem Menüpunkt RÜCKGÄNGIG aufgeführte Beschreibung der zuvor ausgeführten Aktion.

KLASSE:
TextDrawText, TextPrintPreview, WebSourceView, BaseTextSelection, BaseTextEditView

SYNTAX:
UndoAction

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
```

```
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UndoCount-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Schritte, die per Undo rückgängig gemacht werden können. Diese Zahl entspricht dem Wert im Dialog EXTRAS - OPTIONEN - ALLGEMEIN... - SPEICHERN - RÜCKGÄNGIG.

KLASSE:
StarWriter

SYNTAX:
UndoCount

RÜCKGABE:
Long

PARAMETER:
[= Step]: Long-Integer-Ausdruck, der die Anzahl der möglichen Rückwärts-Schritte angibt.

BEISPIEL:

```
Sub ExampleUndo
On error goto UndoError
Dim iCount as integer
Const sMessage_1 = "Erste Seite"
Const sMessage_2 = "Neue Seite"
Const sMessage_3 = "Kein weiteres Undo möglich"
Const sMessage_4 = "UndoAction liefert folgenden Text zurück :
"
Const sError_1 = "Es konnte kein Undo ausgeführt werden"
Documents.Add("sWriter4")
StarWriter.UndoCount = 50
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth = "10cm"
ActiveWindow.Selection.Insert(sMessage_1)
ActiveWindow.Selection.InsertPara
For iCount = 1 to 2
ActiveWindow.Selection.Undo
Next
Print sMessage_4 & ActiveWindow.Selection.UndoAction
```

```
ActiveWindow.Selection.Redo
ActiveWindow.Selection.Insert(sMessage_2)
ActiveWindow.Selection.InsertPara
StarWriter.UndoCount = 0
ActiveWindow.Selection.Insert(sMessage_3)
ActiveWindow.Selection.InsertPara
ActiveWindow.Undo
ActiveWindow.Undo
StarWriter.UndoCount = 50
Exit Sub
undoerror:
MsgBox sError_1, 64, ""
Resume Next
End Sub
```

UndoMode-Eigenschaft

Bestimmt oder ermittelt die Einstellung für den Rückgängig-Modus.

Mit ActiveWindow.UndoMode = False schalten Sie für das aktive Window die Rückgängig-Funktion (und damit auch die Wiederholen-Funktion) aus. Die entsprechenden Befehle im Bearbeiten-Menü und die Buttons in der Funktionsleiste sind jetzt grau.

KLASSE:
BaseTextSelection

SYNTAX:
UndoMode

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob im aktiven Window die Rückgängig-Funktion ausgeführt werden kann (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleUndoMode
REM Es wird der Laufzeitfehler abgefangen
On Error Goto UNDOERROR
Const sMessage_1 = "Das Undo wird ausgeschaltet"
Const sMessage_2 = "Dieser Text kann nicht rückgängig gemacht
werden"
Const sMessage_3 = "Bei ausgeschaltetem Undo kann nichts rück-
gängig gemacht werden"
Documents.Add("sWriter4")
Selection.Insert sMessage_1
Selection.InsertPara
Selection.UndoMode = FALSE
Selection.Insert sMessage_2
```

```
Selection.InsertPara
REM Es kommt hier zu einem Laufzeitfehler, da das Undo ausge-
schaltet ist
Selection.Undo
Selection.UndoMode = TRUE
Exit Sub
UNDOERROR:
Selection.Insert sMessage_3
Selection.UndoMode = TRUE
End Sub
```

UnsetCellsReadOnly-Methode

Hebt den Zellschutz für Tabellenzellen innerhalb der Selektion auf.

KLASSE:
BaseTextTable

SYNTAX:
UnsetCellsReadOnly()

BEISPIEL:

```
Sub ExampleUnsetCellsReadonly
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
Selection.JumpToNextCell
Selection.Insert "Geschützte Zelle"
Selection.Protect
Selection.GotoStart
REM Geschützte Zellen werden übersprungen
Selection.GoRight 1, True
Selection.UnsetCellsReadonly
Selection.JumpToPrevCell
End Sub
```

UnvisitedFmt-Eigenschaft

Bestimmt oder ermittelt, welche Zeichenvorlage verwendet werden soll, um einen noch nicht besuchten Hyperlink darzustellen.

Siehe auch: VisitedFmt

KLASSE:
TextFont

SYNTAX:
UnvisitedFmt

RÜCKGABE:
String

PARAMETER:
[= Format]: Zeichenfolgeausdruck, der die Zeichenvorlage angibt, mit der ein unbesuchter Link dargestellt werden soll.

BEISPIEL:

```
Sub ExampleVistingHype
Documents.Add("sWriter4")
Selection.Insert("Stardivision.de")
Selection.GotoStartOfLine TRUE
With Selection.Font
.UnderlineDouble = TRUE
Selection.StyleNewByExample("UnvisitedHype" , 1 )
Selection.ResetAttributes
Selection.Font.Color = RGB(0, 255, 0)
Selection.StyleNewByExample("VisitedHype" , 1 )
Selection.ResetAttributes
.URL ="http://www.stardivision.de"
.TargetFrame "_blank"
.VisitedFmt = "VisitedHype"
.UnvisitedFmt = "UnvisitedHype"
End With
End Sub
```

UpdateAllIndexes-Methode

Aktualisiert alle Verzeichnisse innerhalb eines Dokuments.

Siehe auch: InsertIndexEntry, InsertIndexesEntry, InsertIndexesIndex, InsertIndexesTOC, InsertIndexesUserIndex.

KLASSE:
BaseTextEditView

SYNTAX:
UpdateAllIndexes()

BEISPIEL:

```
Sub ExampleIndexes
Const sHeadingName = "Überschrift "
Const sStandardParaStyle = "Standard"
Const sUserIndexName = "Benutzerverzeichnis "
Const sIndexesTOCName = "Inhaltsverzeichnis"
Const sNewUserEntry = "Neuer Benutzereintrag"
Const sNewIndex = "Neues Verzeichnis"
Const sNewEntry = "Neuer Eintrag"
Const sIndexesName = "Stichwortverzeichnis"
Const sMessage_1 = "Inhaltsverzeichniszeile : "
Const sMessage_2 = "Benutzerverzeichniszeile "
Const sMessage_3 = "Stichwortverzeichniszeile "
Dim iCount As Integer
Documents.Add( "sWriter4" )
```

```
Selection.InsertPara
Selection.InsertPara
For iCount=1 To 5
Selection.Insert sHeadingName & iCount
Selection.Paragraph.ParaStyleApply sHeadingName & iCount
Selection.InsertPara
Selection.Insert sMessage_1 & iCount
Selection.GoRight 3, TRUE
Selection.InsertIndexesEntry( 0, "", "", "", 0 )
Selection.InsertPara
Selection.InsertPara
Next iCount
Selection.Insert sUserIndexName & "1"
Selection.Paragraph.ParaStyleApply sUserIndexName & "1"
Selection.InsertPara
Selection.Paragraph.ParaStyleApply sStandardParaStyle
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
Selection.GotoStartOfDoc FALSE
Selection.GoDown 1, FALSE
Selection.InsertIndexesTOC(3, 20, sIndexesTOCName)
Selection.GotoStartOfDoc FALSE
Selection.InsertIndexesUserIndex(5, sUserIndexName, 5, sUserIn-
dexName, sUserIndexName & "1" )
Selection.GotoEndOfDoc FALSE
Selection.InsertUserEntry( sUserIndexName, 2, sNewUserEntry )
Selection.InsertPara
Selection.InsertTOCEntry( 1, sNewIndex )
Selection.InsertPara
Selection.InsertIndexesEntry( 0, sNewEntry, "", "", 1 )
Selection.InsertPara
Selection.Insert sMessage_3
Selection.GotoStartOfLine TRUE
Selection.InsertIndexEntry( "", "", "" )
Selection.GotoEndOfLine FALSE
Selection.InsertPara
Selection.InsertIndexesIndex( 1, sIndexesName )
Activewindow.UpdateAllIndexes
End Sub
```

UpdateAllLinks-Methode

Mit dieser Methode können alle verknüpften Bereiche, Grafiken oder Felder aktualisiert werden.

KLASSE:
BaseTextSelection

SYNTAX:
UpdateAllLinks(UpdateGrfLinks As Boolean, CallErrHdl As Boolean)

PARAMETER:
[UpdateGrfLinks]: Bestimmt, ob alle verknüpften Bereiche, Grafiken oder Felder neu geladen werden sollen (True) oder nicht (False). Wird dieser Parameter nicht übergeben, findet keine Aktualisierung statt.

[CallErrorHdl]: Gibt an, ob ein Fehlermeldung erfolgen soll, wenn bei einer DDE-Verknüpfung der Server nicht gestartet werden kann (True), oder ob eine Fehlermeldung nicht erfolgen soll (False). Wird dieser Parameter nicht übergeben, findet keine Fehlermeldung statt.

BEISPIEL:

```
Sub ExampleUpdateAllLinks
Const sRegionName = "Bereich"
Const sMessage_1 = "Diese Datei wird ein Bereich"
Const sMessage_2 = "Hier wird der Bereich eingefügt."
Const sMessage_3 = "So, nun hat sich der Bereich geändert. "
Dim sDocName As String
sDocName = "BerDoc"
REM Document definierem was als Bereich gelinkt wird.
Documents.Add( "sWriter4" )
Selection.Insert sMessage_1
Selection.InsertPara
ActiveDocument.SaveAs Application.PathSettings.Work &"\"&
sDocName &".SDW"
sDocName = ActiveDocument.FullName
sWin = ActiveWindow.Name
REM neues Dokument mit einem Bereich anlegen
Documents.Add( "sWriter4" )
Selection.Insert sMessage_2
Selection.InsertPara
Selection.InsertPara
REM Bereich als Link einfügen
Selection.InsertRegion( sRegionName, "True", FALSE, TRUE,
sDocName)
REM den Bereich im Hintergrund ändern
Windows(sWin).JumpToEndOfDoc
Windows(sWin).Insert(sMessage_3)
Windows(sWin).InsertPara
REM das Dokument was zu dem Fenster gehört Speichern.
Windows(sWin).Document.Save
Wait 1000
Selection.UpdateAllLinks(TRUE, FALSE)
End Sub
```

UpdateDocstat-Methode

Aktualisiert die Statistikdaten des Dokuments, die auch für die entsprechenden Felder verwendet werden.

KLASSE:
BaseTextSelection

SYNTAX:
UpdateDocstat()

BEISPIEL:

```
Sub ExampleStatisticFields
Const sMessage_1 = "Anzahl der Wörter im Dokument : "
Const sMessage_2 = "Nach diesem Satz sind dreizehn Wörter
vorhanden, "
Const sMessage_3 = "und jetzt sechzehn."
Documents.Add( "sWriter4" )
ActiveWindow.Selection.Insert( sMessage_1 )
ActiveWindow.Selection.InsertField( 6, 2, "", "" , 4 )
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.UpdateFields
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.UpdateDocStat
End Sub
```

UpdateFields-Methode

Aktualisiert alle Felder im Dokument, mit Ausnahme der Eingabefelder.

KLASSE:
BaseTextSelection

SYNTAX:
UpdateFields()

BEISPIEL:

```
Sub ExampleStatisticFields
Const sMessage_1 = "Anzahl der Wörter im Dokument : "
Const sMessage_2 = "Nach diesem Satz sind dreizehn Wörter
vorhanden, "
Const sMessage_3 = "und jetzt sechzehn."
Documents.Add( "sWriter4" )
ActiveWindow.Selection.Insert( sMessage_1 )
ActiveWindow.Selection.InsertField( 6, 2, "", "" , 4 )
ActiveWindow.Selection.InsertPara
ActiveWindow.Selection.Insert( sMessage_2 )
ActiveWindow.Selection.UpdateFields
```

```
ActiveWindow.Selection.Insert( sMessage_3 )
ActiveWindow.Selection.UpdateDocStat
End Sub
```

UpdateInputFields-Methode

Aktualisiert die Eingabefelder im Dokument.

KLASSE:
BaseTextSelection

SYNTAX:
UpdateInputFields(OnlyInSelection As Boolean)

PARAMETER:
[OnlyInSelection]: Boolescher Ausdruck, der bestimmt, welche Eingabefelder aktualisiert werden:

True: Es werden nur die Eingabefelder, die sich in der aktuellen Selektion befinden, aktualisiert.

False: Alle Eingabefelder werden aktualisiert (Voreinstellung).

BEISPIEL:

```
Sub ExampleInputFields
Const sFieldContent_1 = "Eingabefeldinhalt"
Const sFieldContent_2 = "Es wird jetzt Eingabefeld_2 geändert"
Const sFieldContent_3 = "Sie können jetzt Eingabefeld_1 ändern"
Const sFieldContent_4 = "Ändern Sie jetzt Eingabefeld_2"
Const sFieldName_1 = "Eingabefeldname_1"
Const sFieldName_2 = "Eingabefeldname_2"
Documents.Add( "sWriter4" )
Selection.InsertField( 16, 1, sFieldContent_1, sFieldName_1, 0
)
Selection.InsertPara
Selection.InsertField( 16, 1, sFieldContent_2, sFieldName_2, 0
)
Selection.Goleft 1, TRUE
REM Es werden nur die selektierten Felder aufgerufen
Selection.UpdateInputFields TRUE
Selection.GoUp 1, FALSE
Selection.SetCurField( 16, sFieldContent_3, sFieldName_1 )
Selection.GoDown 1, FALSE
Selection.GotoPrevInputField()
Selection.GoDown 1, FALSE
Selection.SetCurField( 16, sFieldContent_4, sFieldName_2 )
Selection.GoUp 1, FALSE
Selection.GotoNextInputField()
End Sub
```

URL-Eigenschaft

Bestimmt oder ermittelt den URL eines Hyperlinks. Dieser kann einem Text oder einer Schaltfläche zugewiesen sein. Wird diese Eigenschaft gesetzt, wenn ein Text selektiert ist, so wird die selektierte Zeichenfolge in einen Hyperlink umgewandelt, so daß bei Anklicken des Hyperlinks zu der angegebenen URL-Adresse gesprungen wird.

Siehe auch: TargetFrame

KLASSE:
TextFont, BaseTextDrawControl

SYNTAX:
URL

RÜCKGABE:
String

PARAMETER:
[= Link]: Zeichenfolgeausdruck, der den URL angibt, zu dem der Hyperlink verweisen soll.

BEISPIEL:

```
Sub ExampleHyperLinkScript
Dim sScript As String
Documents.Add "sWriter4"
Selection.Insert "Stardivision.de"
Selection.GotoStartOfLine TRUE

Selection.Font.URL = "http://www.Stardivision.de"
Selection.Font.SetHyperlinkScript(1, "alert(REM JavaScrip-
tREM )")
Selection.Font.TargetFrame = "_blank"
sScript = Selection.Font.GetHyperlinkScript(1)
Print "Folgendes JavaScript ist mit dem Hyperlink verknüpft : "
& sScript
End Sub
```

URLFieldCount-Methode

Ermittelt die Anzahl der im Dokument befindlichen URL-Felder.

KLASSE:
BaseTextEditView

SYNTAX:
URLFieldCount()

RÜCKGABE:
Long

BEISPIEL:

```
Sub ExampleJumpToURL
Const sMessage = "Der aktuelle Hyperlink hat folgende URL : "
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertPara
Selection.SetHyperlink( "Adaptec", "http://www.Adaptec.com/",
"", 1 )
Selection.InsertPara
Selection.SetHyperlink( "IBM", "http://www.IBM.com/", "", 1 )
Selection.InsertPara
Selection.SetHyperlink( "Compaq", "http://www.compaq.com/", "",
1 )
Selection.InsertPara
Selection.JumpToStartOfDoc
For iCount=1 to ActiveWindow.URLFieldCount
ActiveWindow.JumpToURL(iCount)
Print sMessage & Selection.LinkURL
next iCount
End Sub
```

Eigenschaften und Methoden mit "V"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "V" beginnen.

Value-Eigenschaft

Ermittelt den Inhalt der aktuellen Selektion. Wenn sich Umbrüche in einem selektierten Text befinden, enthält der zurückgelieferte String für jedes Umbruchzeichen ein Leerzeichen (Sonderzeichen mit dem ASCII-Code 32 im Dezimalsystem). Dabei ist es gleichgültig, ob es sich bei dem jeweiligen Umbruchzeichen um einen weichen Umbruch oder ein Absatzende handelt.

Siehe auch: ValueAndBreaks

KLASSE:
BaseText, BaseTextDrawControl, BaseTextFrame

SYNTAX:
Value

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleBookmark
Const sMessage = "Testsatz mit mindestens vier Worten"
Const sError = "Es wurde das Bookmark nicht angesprungen"
Dim sBookmarkArray(4) As String
Dim iCount As Integer
Documents.Add "sWriter4"
Selection.Insert sMessage
Selection.GotoStartOfLine FALSE
sBookmarkArray(1) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(1))
Selection.GoRight 1, FALSE
sBookmarkArray(2) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(2))
Selection.GotoEndOfLine FALSE
sBookmarkArray(4) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(4))
Selection.GoLeft (Len(sBookmarkArray(4)+1), FALSE
sBookmarkArray(3) = ActiveWindow.SelectionTextExt
Selection.InsertBookmark(sBookmarkArray(3))
Selection.GotoStartOfLine False
Selection.JumpToNextBookmark
If Selection.Value <> sBookmarkArray(2) Then
Print sError
End If
selection.GoLeft (Len(sBookmarkArray(2)+1), FALSE
Selection.JumpToPrevBookmark
If Selection.Value <> sBookmarkArray(1) Then
Print sError
End If
Selection.JumpToBookmark(sBookmarkArray(3))
If Selection.Value <> sBookmarkArray(3) Then
Print sError
End If
Selection.GotoBookmark(sBookmarkArray(4), TRUE, FALSE)
If Selection.Value <> " " & sBookmarkArray(4) Then
Print sError
End If
For iCount=Selection.Bookmarkcount To 0 Step -1
Selection.DeleteBookmark(Selection.BookmarkName(iCount))
Next iCount
End Sub
```

ValueAndBreaks-Eigenschaft

Liefert den Text der aktuellen Selektion einschließlich der Zeichen für Zeilen- und Absatzumbrüche.

Der zurückgelieferte String enthält alle Umbruchzeichen der Selektion. Diese Umbruchzeichen sind betriebssystemabhängig. Unter Windows werden folgende Werte zurückgegeben:

Ein weicher Umbruch (Shift+Return) liefert das Sonderzeichen mit dem ASCII-Code 10 Dezimal, und ein Absatzende die beiden ASCII-Zeichen 10 und 13.

Siehe auch: Value

KLASSE:
BaseText

SYNTAX:
ValueAndBreaks

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleValueAndBreaks
   Documents.Add("swriter4")
   With Selection
   .Insert("Hier kommt ein harter Umbruch.")
   .InsertPara
   .Insert("Hier kommt ein weicher Umbruch.")
   .InsertLineBreak
   .Insert("Der Text geht weiter.")
   .InsertPara
   .InsertPara
   .Insert("Das ist das Ende.")
   .InsertPara
   End With
   ActiveWindow.JumpToStartOfDoc
   Do While NextParaBegin() = TRUE
   Wait 1000
   Loop
End Sub
Function NextParaBegin() as Boolean
REM Die Funktion springt zu dem nächsten Umbruch, egal ob hart
oder weich
REM Der Umbruch hängt ev. vom System ab.
Const C_LineBreak = 10
```

```
Const C_CarridgeReturn = 13
Dim sSel As String
If ActiveWindow.IsStartOfPara Then ActiveWindow.GoRight 1
ActiveWindow.SearchSettings.RegExp = TRUE
ActiveWindow.Search "^.|\n|^."
sSel = Selection.ValueAndBreaks
If sSel = "" Then
NextParaBegin = FALSE
Else
If sSel = Chr(C_LineBreak) Then
Selection.Escape
Elseif sSel = Chr(C_CarridgeReturn) Then
Selection.Escape
ActiveWindow.GoLeft 1
Else
ActiveWindow.GoLeft 1
End If
NextParaBegin = TRUE
End If
End Function
```

ValueAtCursor-Eigenschaft

Liefert den Inhalt einer Selektion als String zurück.

Ist Text selektiert, dann wird der selektierte Text zurückgegeben, vorausgesetzt, die Selektion geht nicht über einen Absatzumbruch hinweg, dann wird ein Leerstring zurückgegeben.

Steht der Cursor vor einem Feldbefehl, dann wird dessen Inhalt zurückgegeben.

Sonst wird das Wort rechts neben dem Cursor als String zurückgegeben.

KLASSE:
BaseText

SYNTAX:
ValueAtCursor

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleValueAtCursor
Documents.Add "sWriter4"
ActiveWindow.Selection.Insert("Rechts steht ein ")
ActiveWindow.Selection.InsertField( 8, 1, "aVariable", "Feld" ,
0 )
```

```
ActiveWindow.Selection.GotoStartOfDoc TRUE
MsgBox ActiveWindow.Selection.ValueAtCursor ,0,"Der Inhalt der
Selektion lautet"
end sub
```

VerticalCellAlignment-Eigenschaft

Bestimmt oder ermittelt die Art der vertikalen Ausrichtung des Textes innerhalb von Tabellenzellen.

KLASSE:
BaseTextTable

SYNTAX:
VerticalCellAlignment

RÜCKGABE:
Long

PARAMETER:
[= Type]: Long-Integer-Ausdruck, der die Art der Ausrichtung angibt. Folgende Werte sind möglich:

0 - unten

1 - oben

2 - Mitte

BEISPIEL:

```
Sub ExampleVerticalCellAlignment
Documents.Add "sWriter4"
Selection.InsertTable(3, 3)
Selection.SetRowHeight("2cm")
Selection.SetColumnWidth(0, "6cm")
REM Text wird oben an der Zelle angeordnet
Selection.VerticalCellAlignment(1)
Selection.Insert "Dieser Text ist oben angeordnet"
End Sub
```

VertOrientation-Eigenschaft

Bestimmt oder ermittelt die Einstellung für die vertikale Ausrichtung des selektierten Objekts.

KLASSE:
BaseTextList, BaseTextFrame

SYNTAX:
VertOrientation

RÜCKGABE:
Integer

PARAMETER:
[=Orientation]: Integer-Ausdruck, der die Einstellung für die vertikale Ausrichtung des selektierten Objekts angibt. Orientation können die folgenden Werte zugewiesen werden:

0: Positioniert den oberen Rand des selektierten Objekts in einer angegebenen Entfernung vom oberen Seitenrand, sofern dieses an der Seite verankert ist. Ist das Objekt am Absatz verankert, wird es in einer angegebenen Entfernung vom oberen Absatzrand angeordnet. Ein an einem Zeichen verankertes, selektiertes Objekt wird in einer angegebenen Entfernung von der Grundlinie der vorangehenden Zeile positioniert. Die Entfernung des Objekts zum Absatz oder zur Seite wird mit der XPos-Eigenschaft bestimmt.

1: Ordnet den oberen Rand des selektierten Objekts am oberen Rand des Dokuments, Absatzes oder Zeichens an. Die genaue Position des Objekts wird über die VertRelation-Eigenschaft bestimmt.

2: Positioniert das selektierte Objekt in der Mitte der oberen und unteren Seitenränder, sofern dieses an der Seite verankert ist. Ist das selektierte Objekt am Absatz verankert, wird es in der Mitte des oberen und unteren Absatzrandes angeordnet. Ein an einem Zeichen verankertes, selektiertes Objekt wird in der Mitte der Grundlinien der vorangehenden und aktuellen Zeile positioniert.

3: Ordnet den unteren Rand des selektierten Objekts am unteren Rand des Dokuments, Absatzes oder Zeichens an. Die genaue Position des Objekts wird über die VertRelation-Eigenschaft bestimmt.

BEISPIEL:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebundenen Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

VertRelation-Eigenschaft

Bestimmt oder ermittelt detailliert die Ausrichtung des oberen oder unteren Randes des selektierten Objekts.

KLASSE:
BaseTextList, BaseTextFrame

SYNTAX:
VertRelation

RÜCKGABE:
Integer

PARAMETER:
[= FramePos]: Integer-Ausdruck, der die mit der VertOrientation-Eigenschaft gesetzte Ausrichtung des selektierten Objekts spezifiziert. Der VertOrientation-Eigenschaft muß entweder der Wert 1 für die Ausrichtung des oberen Objektrandes oder der Wert 3 für die Ausrichtung des unteren Objektrandes zugewiesen sein. FramePos können die folgenden Einstellungen zugewiesen werden:

0: Positioniert den oberen oder unteren Rand des Objekts an der oberen oder unteren Papierkante (Verankerung an der Seite).

1: Positioniert den oberen oder unteren Rand des Objekts an dem oberen oder unteren Seitenrand (Verankerung an der Seite).

BEISPIEL:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebundenen Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

VertRuler-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige des vertikalen Lineals.

KLASSE:
ViewSettings

SYNTAX:
VertRuler

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob das vertikale Lineal ein- (True) oder ausgeblendet (False) werden soll.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
```

```
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

VertScrollbar-Eigenschaft

Bestimmt oder ermittelt die Option für die Anzeige der vertikalen Bildlaufleiste.

KLASSE:
ViewSettings

SYNTAX:
VertScrollbar

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die vertikale Bildlaufleiste ein- (True) oder ausgeblendet (False) werden soll.

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtsoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
```

```
Selection.InsertPara
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

ViewSettings-Eigenschaft

Liefert die ViewSettings als Objekt zurück und ermöglicht somit einen Zugriff auf die Ansichtsoptionen des Textdokuments. Das sind die Einstellungen, die im Dialog Optionen Textdokument in den Registern Inhalte und Layout vorgenommen werden.

Das Objekt ViewSettings enthält folgende Ansichtsoptionen:

ParagraphEnd - Anzeige Absatzendezeichen

Tab - Anzeige Tabulatoren

Space - Anzeige Leerzeichen

NonbreakingSpace - Anzeige harter Leerzeichen

SoftHyphen - Anzeige weicher Trenner

HiddenText - Anzeige versteckten Textes

ManualBreak - Anzeige harter Zeilenumbrüche

IndexEntry - Anzeige Verzeichniseinträge

FootnoteBackground - Anzeige Fußnoten

Field - Anzeige Feldbefehle

Table - Anzeige Tabellen

Graphic - Anzeige Grafiken

Drawing - Anzeige Zeichenobjekte

FieldName - Anzeige Felder

Notes - Anzeige Notizen

ShowHiddenPara - Anzeige versteckter Absätze

HorzScrollbar - Anzeige horizontaler Bildlaufleiste

VertScrollbar - Anzeige vertikaler Bildlaufleiste

HorzRuler - Anzeige horizontales Lineal

VertRuler - Anzeige vertikales Lineal

TableBounds - Anzeige Tabellenbegrenzungen

Crosshair - Anzeige Hilfslinien

Bounds - Anzeige Textbegrenzungen

StatusLine - Anzeige Statuszeile

SolidHandles - Anzeige farbiger Handles

SmoothScroll - Weiches Scrollen

KLASSE:
StarWriter

SYNTAX:
ViewSettings

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleViewSettings
Const sHeader = "Folgende Ansichtoptionen sind bei Ihnen
gesetzt : "
Const sMessage_1 = "Die Ansichtoption REM "
Const sMessage_2 = "REM  ist wie folgt gesetzt : "
Documents.Add("sWriter4")
Selection.Insert sHeader
Selection.InsertPara
Selection.Insert sMessage_1 & "Bounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Bounds
Selection.InsertPara
Selection.Insert sMessage_1 & "Crosshair" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Crosshair
Selection.InsertPara
Selection.Insert sMessage_1 & "Drawing" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Drawing
Selection.InsertPara
Selection.Insert sMessage_1 & "Field" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Field
Selection.InsertPara
Selection.Insert sMessage_1 & "FieldName" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FieldName
Selection.InsertPara
Selection.Insert sMessage_1 & "FootnoteBackground" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.FootnoteBackground
Selection.InsertPara
Selection.Insert sMessage_1 & "Graphic" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Graphic
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "HiddenText" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HiddenText
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "HorzScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.HorzScrollbar
Selection.InsertPara
Selection.Insert sMessage_1 & "IndexEntry" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.IndexEntry
Selection.InsertPara
Selection.Insert sMessage_1 & "ManualBreak" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ManualBreak
Selection.InsertPara
Selection.Insert sMessage_1 & "NonBreakingSpace" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.NonBreakingSpace
Selection.InsertPara
Selection.Insert sMessage_1 & "Notes" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Notes
Selection.InsertPara
Selection.Insert sMessage_1 & "ParagraphEnd" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ParagraphEnd
Selection.InsertPara
Selection.Insert sMessage_1 & "ShowHiddenPara" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.ShowHiddenPara
Selection.InsertPara
Selection.Insert sMessage_1 & "SmoothScroll" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SmoothScroll
Selection.InsertPara
Selection.Insert sMessage_1 & "SoftHyphen" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SoftHyphen
Selection.InsertPara
Selection.Insert sMessage_1 & "SolidHandles" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.SolidHandles
Selection.InsertPara
Selection.Insert sMessage_1 & "Space" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Space
Selection.InsertPara
Selection.Insert sMessage_1 & "StatusLine" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.StatusLine
Selection.InsertPara
Selection.Insert sMessage_1 & "Tab" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Tab
Selection.InsertPara
Selection.Insert sMessage_1 & "Table" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.Table
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounderies" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounderies
Selection.InsertPara
Selection.Insert sMessage_1 & "TableBounds" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.TableBounds
Selection.InsertPara
```

```
Selection.Insert sMessage_1 & "VertRuler" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertRuler
Selection.InsertPara
Selection.Insert sMessage_1 & "VertScrollbar" & sMessage_2
Selection.Insert ActiveModule.ViewSettings.VertScrollbar
Selection.InsertPara
End Sub
```

VisitedFmt-Eigenschaft

Bestimmt oder ermittelt, welche Zeichenvorlage verwendet werden soll, um einen bereits besuchten Hyperlink darzustellen.

Siehe auch: UnvisitedFmt

KLASSE:
TextFont

SYNTAX:
VisitedFmt

RÜCKGABE:
String

PARAMETER:
[= Format]: Zeichenfolgeausdruck, der die Zeichenvorlage angibt, mit der der besuchte Link dargestellt werden soll.

BEISPIEL:

```
Sub ExampleVistingHype
Documents.Add("sWriter4")
Selection.Insert("Stardivision.de")
Selection.GotoStartOfLine TRUE
With Selection.Font
.UnderlineDouble = TRUE
Selection.StyleNewByExample("UnvisitedHype" , 1 )
Selection.ResetAttributes
Selection.Font.Color = RGB(0, 255, 0)
Selection.StyleNewByExample("VisitedHype" , 1 )
Selection.ResetAttributes
.URL ="http://www.stardivision.de"
.TargetFrame "_blank"
.VisitedFmt = "VisitedHype"
.UnvisitedFmt = "UnvisitedHype"
End With
End Sub
```

Eigenschaften und Methoden mit "W"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "W" beginnen.

WatercanStyleModeOnOff-Methode

Schaltet den Gießkannenmodus des Stylisten ein. Wollen Sie den Modus wieder ausschalten, dann rufen Sie die Methode ohne Übergabe von Parametern auf.

KLASSE:
BaseTextSelection

SYNTAX:
WatercanStyleModeOnOff(Param As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Param: Name der Vorlage, die im Gießkannenmodus angewendet werden soll.

Family: Bestimmt die Familie der Vorlage, die im Gießkannenmodus angewendet werden soll. Folgende Werte sind möglich:

1 - Zeichenvorlage

2 - Absatzvorlage

4 - Rahmenvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExampleWatercan
Dim bCan As Boolean
Const sMessage = "Status des Gießkannenmodus : "
Documents.Add( "sWriter4" )
With ActiveWindow.Selection
.WatercanStyleModeOnOff("Standard" ,2 )
bCan = .StyleWatercanMode
MsgBox sMessage & bCan, 64, "StyleWatercanMode"
wait 2000
.WatercanStyleModeOnOff("" ,2 )
End With
End Sub
```

Widow-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Zeilen eines auf die nächste Seite umbrochenen Absatzes, die noch am unteren Rand der ersten Seite aufgeführt werden dürfen (Schusterjungenregelung). Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetWidow gesetzte Schusterjungenregel abgefragt.

Eine Abfrage der Widow-Eigenschaft am SearchAttributes- oder ReplaceAttributes-Objekt ist nur dann sinnvoll, wenn mit IsWidow bestimmt wurde, daß nach einer bestimmten Schusterjungenregel gesucht werden soll, d. h. IsWidow = 2.

Siehe auch: Orphan, SetWidow, IsWidow.

KLASSE:
Paragraph, ReplaceAttributes, SearchAttributes

SYNTAX:
Widow

RÜCKGABE:
Integer

PARAMETER:
Bei Absätzen:

[= Rows]: Integer-Ausdruck, der die Anzahl der Zeilen eines auf die nächste Seite umbrochenen Absatzes angibt, die noch am unteren Rand der ersten Seite aufgeführt werden dürfen.

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

BEISPIEL:

```
Sub ExampleOrphanAndWidow
REM den lateinischen Blindtext verwenden
Const sBlindText = "BT"
Dim iCount As Integer
Documents.Add("sWriter4")
With ActiveWindow.Selection
.Insert("Diese ist ein Absatz der mit der Schusterjungenregel
auf 4 und Hurenkinderregel auf 4 formatiert wurde.  ")
.Insert("Der Absatz wird langsam über die Seitengrenze
geschoben. Verfolgen Sie was passiert, indem Sie die vertikale
")
.Insert("Bildlaufleiste nachstellen.  ")
.Insert(" *** *** *** *** *** *** *** *** *** *** *** *** ***
```

```
*** ***   ")
.ExpandGlossary( sBlindText )
End With
With ActiveWindow.Selection.Paragraph
.Widow = 4
.Orphan = 4
End With
ActiveWindow.JumpToStartOfDoc
For iCount = 1 to 56
Wait 20
Selection.InsertPara
Next
End Sub
```

Width-Eigenschaft

Bestimmt oder ermittelt die Breite des aktiven Objekts oder die Größe des Schattens, der dem selektierten Bereich zugewiesen ist.

Siehe auch: Height, GraphicWidth.

KLASSE:
Shadow, BaseTextFrame, BaseTextTable, BaseTextEditView

SYNTAX:
Width

RÜCKGABE:
Long

PARAMETER:
Bei Rahmen, Grafiken, OLE-Objekten, Tabellen:

[= Value]: Long-Integer-Ausdruck, der die Breite des Objekts in der Standardmaßeinheit angibt. Der Wert kann auch mit einer Maßeinheit als Zeichenfolge übergeben werden.

Bei Schatten:

[= Value]: Long-Integer-Ausdruck, der die Größe des Schattens in Twips angibt.

Bei Fenstern:

[= Value]: Long-Integer-Ausdruck, der die Größe des Fensters in Pixel angibt.

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath as String, sGrfName As String
Dim fHeight As Single, fWidth As Single
```

```
Documents.Add("sWriter4")
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Selection.InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
sGrfName = Selection.GraphicName
fHeight = Selection.GraphicHeight
fWidth = Selection.GraphicWidth
Selection.Height = fHeight + 2
Selection.Width = fWidth + 2
Selection.GraphicURL = "http://www.Stardivision.de"
Print "Der vollständige Grafikname lautet : " &
Selection.GrfLinkName
Print "Die Grafik hat folgendes Grafikformat : " &
Selection.GrfLinkFilterName
ActiveWindow.Graphics(sGrfName).GraphicMirror = 3
If MsgBox("Soll diese Grafik gelöscht werden ?",4+32, "Grafik
löschen") = 6 Then
ActiveWindow.Graphics(sGrfName).Delete
End If
End Sub
```

WidthPercent-Eigenschaft

Bestimmt oder ermittelt die prozentuale Breite eines selektierten Objekts relativ zur Gesamtbreite der Seite.

Haben Sie mit der Width-Eigenschaft einen metrischen Wert für die Breite des Objekts angegeben, so wird dieser Wert bei Anwendung der WidthPercent-Eigenschaft ignoriert. Er wird erst wieder verwendet, wenn die prozentuale Breite wieder auf 0 gesetzt wird.

Siehe auch: Height, HeightPercent, Width.

KLASSE:
BaseTextFrame, BaseTextTable

SYNTAX:
WidthPercent

RÜCKGABE:
Long

PARAMETER:
[= Value]: Integer-Ausdruck, der die Breite des Objekts in Prozent angibt. Als Prozentwerte werden nur Zahlen von 0 bis 100 akzeptiert.
Wird Value = 255 gesetzt, so erfolgt ein automatischer Abgleich der Breite.

Beispiel:

```
REM Dieses Beispiel vergrößert einen Rahmen bis auf
REM die Gesamtgröße der Seite
Sub ExampleHeightPercent
Dim iCount As Integer
Documents.Add( "sWriter4" )
Selection.InsertFrame(2, "0cm", "0cm", "1cm", "1cm")
For iCount = 0 To 100
Selection.HeightPercent = iCount
Selection.WidthPercent = iCount
Next iCount
End Sub
```

WordMode-Eigenschaft

Bestimmt oder ermittelt die Option für wortweises Unterstreichen oder Durchstreichen. Beim Suchen und Ersetzen von Attributen wird mit dieser Eigenschaft die mit SetWordMode gesetzte Suchoption abgefragt, wenn die Eigenschaft IsWordMode den Wert 2 liefert.

Siehe auch: SetWordMode, IsWordMode.

Klasse:
ReplaceAttributes, SearchAttributes, TextFont

Syntax:
WordMode

Rückgabe:
Boolean

Parameter:
Bei Text-Objekten:

[= Switch]: Boolescher Ausdruck, der angibt, ob der selektierte Text wortweise unterstrichen bzw. durchgestrichen dargestellt werden soll (True) oder nicht (False).

Beim Suchen und Ersetzen von Attributen:

(nur lesen)

Beispiel:

```
Sub ExampleFont
Dim iFontSize As Integer
Documents.Add "sWriter4"
Call WriteAndMark("Bold  ")
Selection.Font.Bold = soWeightBold
Selection.Escape
Selection.Font.Bold = 0
```

```
Call WriteAndMark("Italic  ")
Selection.Font.Italic = soItalicOblique
Selection.Escape
Selection.Font.Italic = 0
Call WriteAndMark("Underline  ")
Selection.Font.Underline = soUnderlineSingle
Selection.Escape
Selection.Font.Underline = 0
Call WriteAndMark("StrikeOutThroughSingle  ")
Selection.Font.StrikeOut = soStrikeThroughSingle
Selection.Escape
Selection.Font.StrikeOut = 0
Call WriteAndMark("OutlineFont  ")
Selection.Font.OutlineFont = TRUE
Selection.Escape
Selection.Font.OutlineFont = FALSE
Call WriteAndMark("Shadowed  ")
Selection.Font.Shadowed = TRUE
Selection.Escape
Selection.Font.Shadowed = FALSE
Call WriteAndMark("PairKerning  ")
Selection.Font.PairKerning = TRUE
Selection.Escape
Selection.Font.PairKerning = FALSE
Call WriteAndMark("SuperScript  ")
Selection.Font.SuperScript = TRUE
Selection.Escape
Selection.Font.SuperScript = FALSE
Call WriteAndMark("SubScript  ")
Selection.Font.SubScript = TRUE
Selection.Escape
Selection.Font.SubScript = FALSE
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = 101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Escapement+RelativeSize  ")
Selection.Font.Escapement = -101
Selection.Font.RelativeSize = 62
Selection.Escape
Selection.Font.Escapement = 0
Selection.Font.RelativeSize = 100
Call WriteAndMark("Language  ")
Selection.Font.Language = 1031
Selection.Escape
Call WriteAndMark("CaseMap ")
Selection.Font.CaseMap = 1
Selection.Escape
Selection.Font.CaseMap = 0
Call WriteAndMark("Spacing  ")
Selection.Font.Spacing = 20
Selection.Escape
```

```
Selection.Font.Spacing = 0
Call WriteAndMark("Fonthöhe + 2pt  ")
iFontSize = Selection.Font.Size
Selection.Font.Size = iFontSize + 2
Selection.Escape
Selection.Font.Size = iFontSize
Call WriteAndMark("Grow  ")
Selection.Font.Grow
Selection.Escape
Call WriteAndMark("Shrink  ")
Selection.Font.Shrink
Selection.Escape
Call WriteAndMark("WordMode ")
Selection.Font.WordMode = TRUE
Selection.Font.Underline = 1
Selection.Escape
Selection.Font.WordMode = FALSE
Selection.Font.Underline = 0
Call WriteAndMark("CharFontNameObject ")
Selection.Font.FamilyName = "Helvetica"
Selection.Escape
Call WriteAndMark("Flash ")
Selection.Font.Flash = TRUE
Selection.Escape
Selection.Font.Flash = FALSE
Call WriteAndMark("Color (Grün) ")
Selection.Font.Color = RGB(0, 255, 0)
Selection.Escape
Selection.Font.Color = RGB(0, 0, 0)
End Sub
Sub WriteAndMark(sParaString as String)
Dim iParaStringLen As Integer
iParaStringLen = Len(sParaString)
Selection.Insert(sParaString)
Selection.GoLeft(iParaStringLen, FALSE)
Selection.GoRight(iParaStringLen, TRUE)
End Sub
```

Wrap-Methode

Stellt den Umlauf für das selektierte Objekt ein.

KLASSE:
BaseTextFrame

SYNTAX:
Wrap(Wrap As Long, GoldCut As Boolean, Opaque As Boolean)

PARAMETER:
Wrap: Long-Integer-Ausdruck, der die Art des Umlaufs angibt. Folgende Werte sind möglich:

0 - Kein Umlauf

1 - Transparenter Durchlauf

2 - Paralleler Umlauf

4 - Umlauf links

5 - Umlauf rechts

GoldCut: Schaltet den parallelen Umlauf (False) in den dynamischen Umlauf (True) um und umgekehrt.

Opaque: Gibt an, ob die Option "Im Hintergrund" aktiv ist (True) oder nicht (False). Dieser Parameter wirkt sich nur für die Einstellung "Durchlauf" aus.

Beispiel:

```
Sub ExampleFrameAnchorPage
Const sText1="Das ist ein seitengebundener Rahmen."
Const sText2="Das ist der Text der außerhalb des Rahmens steht.
"
Dim sFirstFrame As String, sLastFrame As String
Dim sName
Dim iCount As Integer
Documents.Add("sWriter4")
ActiveWindow.PageHeight = "10cm"
ActiveWindow.PageWidth  = "12cm"
With ActiveWindow
.TopMargin = "1cm"
.BottomMargin = "1cm"
.LeftMargin = "1cm"
.RightMargin = "1cm"
End With
REM Seitengebundener Rahmen einfügen
Selection.InsertFrame(2, "3cm", "3cm", "4cm", "2cm")
sFirstFrame = Selection.FrameName
sName = ActiveWindow.Frames.Item( sFirstFrame ).Index
Selection.Insert( sText1 )
With ActiveWindow.Frames( sFirstFrame )
.AnchorType = 2 REM An der Seite gebunden
Wait 500
.AlignHorizontalCenter
Wait 500
.AlignBottom
Wait 500
.AlignLeft
Wait 500
.AlignVerticalCenter
Wait 500
.AlignTop
Wait 500
.AlignHorizontalCenter
Wait 500
```

```
.AlignRight
Wait 500
.AlignVerticalCenter
Wait 500
.AlignHorizontalCenter
Wait 500
End With
ActiveWindow.JumpToStartOfDoc
For iCount=1 to 10
Selection.Insert(  sText2 )
if (iCount Mod 2 ) = 0 then
Selection.InsertPara
End If
Next iCount
With ActiveWindow.Frames( sFirstFrame )
wait 500
.Wrap(0) REM Kein Umlauf
wait 500
.Wrap(1,TRUE, TRUE) REM Transparenter Durchlauf
wait 500
.Wrap(1,FALSE, FALSE) REM Transparenter Durchlauf
wait 500
.Wrap(2) REM Paralleler Umlauf
wait 500
.Wrap(4) REM Umlauf links
wait 500
.Wrap(5) REM Umlauf rechts
wait 500
.GotoAnchor
End With
End Sub
```

Eigenschaften und Methoden mit "X"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "X" beginnen.

XPos-Eigenschaft

Bestimmt oder ermittelt die horizontale Position des selektierten Objekts.

Um das selektierte Objekt in der horizontalen Ebene positionieren zu können, muß der HorzOrientation-Eigenschaft sowie der HorzRelation-Eigenschaft der Wert 0 zugewiesen werden.

KLASSE:
BaseTextFrame

Syntax:
XPos

Rückgabe:
Long

Parameter:
[= HorzPos]: Long-Integer-Ausdruck, der die Entfernung des linken Objektrandes zum linken Seitenrand angibt, sofern das selektierte Objekt an der Seite verankert ist. Ist das Objekt am Absatz verankert, gibt HorzPos die Entfernung des linken Objektrandes zum linken Einzug des Absatzes an. Die Entfernung wird in der aktuell definierten Maßeinheit bestimmt.

Beispiel:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebundenen Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

Eigenschaften und Methoden mit "Y"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Y" beginnen.

YPos-Eigenschaft

Bestimmt oder ermittelt die vertikale Position des selektierten Objekts.

Um das selektierte Objekt in der vertikalen Ebene positionieren zu können, muß der VertOrientation-Eigenschaft sowie der VertRelation-Eigenschaft der Wert 0 zugewiesen werden.

Klasse:
BaseTextList, BaseTextFrame

Syntax:
YPos

Rückgabe:
Long

Parameter:
[= VertPos]: Long-Integer-Ausdruck, der die Entfernung des oberen Objektrandes zum oberen Seitenrand angibt, sofern das selektierte Objekt an der Seite verankert ist. Ist das Objekt am Absatz verankert, gibt VertPos die Entfernung des oberen Objektrandes zum oberen Absatzrand an. Die Entfernung wird in der aktuell definierten Maßeinheit bestimmt.

Beispiel:

```
Sub ExampleHorzRelation
Documents.Add( "sWriter4" )
REM Fügt einen seitengebundenen Rahmen ein.
Selection.InsertFrame( 2, "6.00cm", "4.00cm", "7.00cm",
"5.00cm" )
Selection.HorzRelation = 0
Selection.HorzOrientation = 0
Activewindow.XPos = "5cm"
Selection.VertRelation = 0
Selection.VertOrientation = 0
Activewindow.YPos = "7cm"
End Sub
```

Eigenschaften und Methoden mit "Z"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Z" beginnen.

Zoom-Eigenschaft

Ändert die Maßstabseinstellungen für die Dokumentanzeige.

Klasse:
TextPrintPreview, BaseTextEditView

Syntax:
Zoom

Rückgabe:
Integer

Parameter:
= Value (nur Schreiben): Integer-Ausdruck, der die Maßstabseinstellung angibt. Folgende Werte sind möglich:

-1 = Optimal

-2 = ganze Seite

-3 = Seitenbreite

> 0 = Angaben in Prozent, mindestens 20 maximal 600

Beispiel:

```
ExampleZoom
Dim iZoom As Integer
Documents.Add( "sWriter4" )
For iZoom = 10 to 400 Step 5
ActiveWindow.SelectAll
Selection.Insert("Zoomfaktor " & iZoom)
ActiveWindow.Zoom = iZoom
Wait 500
Next iZoom
End Sub
```

Referenzhilfe für Tabellendokumente

Hier erhalten Sie in alphabetischer Ordnung Informationen über sämtliche Eigenschaften und Methoden, die Ihnen für Tabellendokumente zur Verfügung stehen.

Eigenschaften und Methoden mit "A"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "A" beginnen.

Activate-Methode

Aktiviert ein selektiertes OLE-Objekt.

Der Zugriff erfolgt über die Selection oder über die entsprechende Collection "OLEObjects".

KLASSE:
TableOLEObject, TableSelection

SYNTAX:
Activate()

BEISPIEL:

```
Sub ExampleActivate
Dim oDoc as Object
Documents.Add("scalc4")
Wait 1000
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Männer")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Frauen")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("b5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
Wait 2000
oDoc = ActiveDocument REM Objekt merken
oDoc.OLEObjects(1).Activate
wait 5000
oDoc.OLEObjects(1).Deactivate
End Sub
```

ActiveCell-Eigenschaft

Gibt ein Range-Objekt für die Zelle zurück, in der der Cursor in der aktiven View des Dokuments steht.

Unabhängig von der Selektion, bezieht sich diese Funktion nur auf die Cursorposition. ActiveCell.Name liefert dabei die Cursorposition zurück, z. B. "A4" als String.

KLASSE:
TableDocument

SYNTAX:
ActiveCell

RÜCKGABE:
Object

PARAMETER:
Return: Range-Objekt

BEISPIEL:

```
Sub ExampleActiveCell
Dim sAddress As String
Documents.Add("scalc4")
sAddress = ActiveCell.Address
ActiveCell.Insert( "Diese Zelle hat die Referenz: " & sAddress
)
end sub
```

ActiveTable-Eigenschaft

Gibt ein Table-Objekt für die Tabelle zurück, die in der aktiven View des Dokuments angezeigt wird.

KLASSE:
TableDocument

SYNTAX:
ActiveTable

RÜCKGABE:
Object

PARAMETER:
Return: Table-Objekt

Beispiel:

```
Sub ExampleActiveTable
Documents.Add("scalc4")
ActiveCell.Insert( "Der Tabellenname lautet: " & ActiveDocu-
ment.Tables( ActiveTable.Index ).Name
End Sub
```

Add-Methode

Hängt eine neue Tabelle hinten an das Dokument an, zu dem die Tables-Collection gehört.

Klasse:
Tables

Syntax:
Add(Name As String)

Parameter:
Name: Name der neuen Tabelle.

Beispiel:

```
Sub ExampleCalcTables
Const sTab1="Tabelle_Versteckt"
Const sTab2="Tabelle_Neu"
Const sTab3="Tabelle_Eingefuegt"
Dim iCount As Integer
Dim sTable As String
Documents.Add("scalc4")
REM alle ausser einer Tabelle verstecken
For iCount = 1 To ActiveDocument.Tables.Count -1
   ActiveDocument.Tables.Hide(
ActiveDocument.Tables.Item(iCount).Name )
Next iCount
sTable = ActiveDocument.Tables(1).Name
With ActiveDocument.Tables
   .Add( sTab1 )
   .Add( sTab2 )
   .Insert( sTab3 ,2)
   .Hide( sTab1 )
   .Show( sTable )
End With
End Sub
```

AddChartData-Methode

Fügt zu einem bestehenden Chart Datenbereiche aus einer Tabelle hinzu.

KLASSE:
TableDocument

SYNTAX:
AddChartData(Name As String, Range As String)

PARAMETER:
Name: Zeichenfolge, die den Namen des Charts angibt. (Da dem Chart selbst kein Name zugewiesen werden kann, muß der Name des OLE-Objekts angegeben werden.)

Range: Zeichenfolge, die den hinzuzufügenden Zellbereich angibt.

BEISPIEL:

```
Sub ExampleStarChart
Documents.Add("scalc4")
Wait 2000
REM Tabelle mit Daten füllen
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Frauen")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Männer")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("e5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
wait 2000
REM Daten verändern
Range("a4").Insert("Jungens")
Range("b4").Insert("1300")
Range("c4").Insert("150")
Range("d4").Insert( 0.5 )
wait 3000
Range("e5").UpdateChart(TRUE)
REM neue Daten hinzufügen
Range("b7").Insert( "Mädchen" )
Range("b8").Insert( "Babies" )
Range("c7").Insert( 23 )
Range("e8").Insert( 120 )
ActiveDocument.AddChartData( OLEObjects(1).Name , "b7:e8")
wait 3000
REm anderen Chartbereich festlegen
Range("c6").Insert("Dubai")
Range("d6").Insert("Paris")
```

```
Range("e6").Insert("Hong Kong")
ActiveDocument.ChangeChartData( OLEObjects(1).Name , "b6:e8" ,
TRUE, TRUE)
End Sub
```

AddPrintArea-Methode

Legt einen Zellbereich als Druckbereich für die zugehörige Tabelle fest. Wurden zuvor bereits Druckbereiche definiert, so wird mit dieser Methode ein neuer Druckbereich hinzugefügt.

KLASSE:
RangeSelection

SYNTAX:
AddPrintArea()

BEISPIEL:

```
Sub ExamplePrintArea
Documents.Add("scalc4")
ActiveCell.Insert("=a2+1")
ActiveCell.Cut
ActiveDocument.Range("a1:h8").Paste
ActiveDocument.Range("a1:b2").AddPrintArea
ActiveDocument.Range("c3:d4").AddPrintArea
ActiveDocument.Range("e5:f6").AddPrintArea
ActiveDocument.Range("g7:h8").AddPrintArea
ActiveDocument.Range("c3:d4").PrintPreview = TRUE
End Sub
```

Address-Eigenschaft

Liefert den selektierten oder den angegebenen Tabellenbereich als Zeichenfolge zurück.

KLASSE:
RangeSelection

SYNTAX:
Address

RÜCKGABE:
String

PARAMETER:
(nur lesen)

Beispiel:

```
Sub ExampleActiveCell
Dim sAddress As String
Documents.Add("scalc4")
sAddress = ActiveCell.Address
ActiveCell.Insert( "Diese Zelle hat die Referenz: " & sAddress
)
End Sub
```

AlignBlock-Methode

Setzt die horizontale Ausrichtung für den selektierten Zellbereich auf Blocksatz.

Diese Methode funktioniert nur für eine einfache Selektion. Mehrfachselektionen im Tabellendokument sind vom StarBasic aus i.a. nicht ansprechbar.

Siehe auch: HorizontalAlignment.

Klasse:
RangeSelection

Syntax:
AlignBlock()

Beispiel:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
```

```
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignBottom-Methode

Setzt die vertikale Ausrichtung für den selektierten Zellbereich so, daß eine Ausrichtung an den unteren Zellbegrenzungen erfolgt.

Siehe auch: AlignTop, VerticalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignBottom()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
```

```
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignHCenter-Methode

Setzt die horizontale Ausrichtung für den selektierten Zellbereich so, daß eine zentrierte Ausrichtung erfolgt.

Siehe auch: AlignVCenter, HorizontalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignHCenter()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
```

```
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignLeft-Methode

Setzt die horizontale Ausrichtung für den selektierten Zellbereich auf Linksbündig.

Siehe auch: AlignRight, HorizontalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignLeft()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
```

```
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignRight-Methode

Setzt die horizontale Ausrichtung für den selektierten Zellbereich auf Rechtsbündig.

Siehe auch: AlignLeft, HorizontalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignRight()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
```

```
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignTop-Methode

Setzt die vertikale Ausrichtung für den selektierten Zellbereich so, daß eine Ausrichtung an den oberen Zellbegrenzungen erfolgt.

Siehe auch: AlignBottom, VerticalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignTop()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
```

```
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AlignVCenter-Methode

Setzt die vertikale Ausrichtung für den selektierten Zellbereich so, daß eine zentrierte Ausrichtung erfolgt.

Siehe auch: AlignHCenter, VerticalAlignment.

KLASSE:
RangeSelection

SYNTAX:
AlignVCenter()

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
```

```
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

AuditingFillMode-Methode

Schaltet den Detektiv-Füll-Modus ein oder aus.

KLASSE:
TableEditView

SYNTAX:
AuditingFillMode()

BEISPIEL:

```
Sub Main
Documents.Add "sCalc4"
Activewindow.AuditingFillMode = FALSE
End Sub
```

AutoComplete-Eigenschaft

Bestimmt oder ermittelt, ob "AutoEingabe" aktiv ist. Der Zugriff erfolgt mit "StarCalc.AutoComplete".

KLASSE:
StarCalc

SYNTAX:
AutoComplete

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob AutoEingabe ein- (True) oder ausgeschaltet (False) ist.

BEISPIEL:

```
Sub  ExampleAutoComplete
Documents.add "sCalc4"
StarCalc.AutoComplete
Range("A3").Insert("Die automatische Eingabeergänzung ist " &
IIf( StarCalc.AutoComplete,"an","aus") & "." )
End Sub
```

AutoFill-Methode

Füllt die Zellen von der aktuellen Zelle bis zur angegebenen Tabellenspalte oder Tabellenzeile aus. In der Voreinstellung wird mit einem Inkrement von 1 eine arithmetische Reihe erzeugt, mit dem Inhalt der aktuellen Zelle als Startwert. Beim Ausfüllen nach oben oder links ist das Inkrement -1. Gefüllt wird nur die aktuelle Zeile oder Spalte, kein zweidimensionaler Bereich.

KLASSE:
RangeSelection

SYNTAX:
AutoFill(EndCell As String)

PARAMETER:
EndCell: Zeichenfolge, die die Adresse der Endzelle angibt. Der Bezeichnung der Tabellenspalte und der Nummer der Tabellenzeile kann das $-Zeichen voranstehen, z.B. "A3".

BEISPIEL:

```
Sub ExampleAutoFill
Documents.Add("scalc4")
ActiveCell.Insert("=a2+1")
REM Die Formel wird in alle Zellen gefüllt.
ActiveCell.AutoFill ("$A$5" )
ActiveWindow.GoToCell("b1")
ActiveCell.Insert("Wort_1")
REM Hier wird ein Wort mit Zahl aufsteigend gefüllt.
ActiveCell.AutoFill ("$b$5" )
End Sub
```

AutoFormat-Methode

Weist dem selektierten oder dem bezeichneten Bereich das angebene AutoFormat zu. Dabei muß der Bereich, auf den das gewählte Format angewendet werden soll, mindestens 3 mal 3 Zellen groß sein.

KLASSE:
RangeSelection

SYNTAX:
AutoFormat(aFormatName As String)

PARAMETER:
aFormatName: Zeichenfolge, die die Bezeichnung des Formates angibt, das dem selektierten oder dem bezeichneten Bereich zugewiesen werden soll. aFormatName kann die Bezeichnungen eines im Dialog AUTOFORMAT aufgeführten Formates aufnehmen.

Beispiel:

```
Sub ExampleAutoFormat
Const sFormat="Standard"  REM ein Tabellen-AutoFormat defi-
nieren
Documents.Add("scalc4")
Range("b2:e7").AutoFormat("Standard")
End Sub
```

AutomaticCalculation-Eigenschaft

Schaltet die Option für die automatische Neuberechnung von Zellen in einem Toggle-Verhalten um oder ermittelt, ob eine automatische Berechnung aktiv ist. Ist die automatische Berechnung eingeschaltet, liefert diese Eigenschaft den Wert True zurück, ist sie ausgeschaltet, den Wert False. Ist die automatische Berechnung eingeschaltet, wird sie durch Setzen dieser Eigenschaft ausgeschaltet und umgekehrt, dabei spielt es keine Rolle, welcher Parameter übergeben wird.

Klasse:
TableDocument

Syntax:
AutomaticCalculation

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die automatische Berechnung ein- (True) oder ausgeschaltet (False) ist. Die Zuordnung der Werte True und False gilt nur, wenn die AutomaticCalculation-Eigenschaft abgefragt wird. Wird dieser Parameter gesetzt, entscheidet der ursprüngliche Zustand, ob die automatische Berechnung ein- oder ausgeschaltet wird.

Beispiel:

```
Sub ExampleCalculation
Documents.Add "sCalc4"
Wait 500
ActiveDocument.AutomaticCalculation = FALSE
ActiveTable.Cells(3,2).Insert( "Automatisch-Berechnen
abschalten.")
Range("a1").Insert("400")
Range("a2").Insert("=a1*11")
ActiveTable.Cells(3,3).Insert( "Bei Eingabe wird berechnet.")
Wait 500
Range("a1").Insert("123")
ActiveTable.Cells(3,4).Insert( "Hat sich ein Argument verän-
dert, wird nicht neu berechnet")
```

```
Wait 3000
ActiveTable.Cells(3,5).Insert("Neu Berechnen.")
Wait 500
ActiveDocument.Calculate
ActiveTable.Cells(3,6).Insert( "Automatisch-Berechnen wieder
anschalten.")
Wait 500
ActiveDocument.AutomaticCalculation = TRUE
Wait 3000
Range("a1").Insert("5500")
Wait 1000
ActiveTable.Cells(3,7).Insert( "Alles noch einmal durch-
rechnen.")
ActiveDocument.CalculateHard
End Sub
```

AutoOutline-Methode

Wendet die Funktion AutoGliederung auf eine Tabelle an. Diese Methode entspricht dem Menübefehl DATEN - GLIEDERUNG - AUTOGLIEDERUNG.

KLASSE:
TableSelection

SYNTAX:
AutoOutline()

BEISPIEL:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
```

```
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

Eigenschaften und Methoden mit "B"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "B" beginnen.

BackgroundColor-Eigenschaft

Liefert oder setzt die Hintergrundfarbe des Zell-Hintergrundes.

Siehe auch: BackColor

KLASSE:
RangeSelection

SYNTAX:
BackgroundColor

RÜCKGABE:
Long

PARAMETER:
[= Color]: Long-Integer-Ausdruck, der den Farbcode für die Hintergrundfarbe angibt.

BEISPIEL:

```
Sub ExampleBorder
Documents.add "sCalc4"
With ActiveDocument.Range("b2:c4").Border
   .TopLine.Style = 2 REM 1pt-dicke Linie
   .TopLine.Color = RGB(0,0,255) REM in Blau
   .BottomLine.Style = 2
   .BottomLine.Color = RGB(0,0,255)
   .LeftLine.Style = 2
   .LeftLine.Color = RGB(0,0,255)
   .RightLine.Style = 2
   .RightLine.Color = RGB(0,0,255)
   .HorizontalLine.Style = 2
   .HorizontalLine.Color = RGB(0,0,255) REM im starCalc auch
die inneren Linien
   .VerticalLine.Style = 2
   .VerticalLine.Color = RGB(0,0,255)
End With
```

```
ActiveDocument.Range("b2:c4").BackGroundColor =
RGB(255,128,128)
End Sub
```

Backward-Eigenschaft

Bestimmt oder ermittelt die Suchoption für die Richtung der Suche.

KLASSE:
TableEditView

SYNTAX:
Backward

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche ab der Cursorposition rückwärts (True) oder vorwärts (False) erfolgen soll.

BEISPIEL:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

Bold-Eigenschaft

Bestimmt oder ermittelt die Intensität der Fett-Darstellung der Schrift des selektierten Bereiches.

KLASSE:
TableFont

SYNTAX:
Bold

RÜCKGABE:
Integer

PARAMETER:
[= BoldIntens]: Integer-Ausdruck, der die Intensität der Fett-Darstellung der Schrift des selektierten Bereiches angibt. Folgende Werte sind möglich:

0: Bestimmt die normale Darstellung für die Schrift des selektierten Bereiches.

1 - 10: Bestimmt die Fett-Darstellung für die Schrift des selektierten Bereiches. Die Intensität der Darstellung erhöht sich mit zunehmendem Wert.

BEISPIEL:

```
Sub ExampleFontBold
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 10
For iCount = 0 To 10
ActiveDocument.Cells(2, iCount+1).Font.Bold = iCount
ActiveDocument.Cells(2, iCount+1).Insert "Das ist ein Text."
Next iCount
End Sub
```

Border-Eigenschaft

Gibt ein Border-Objekt zurück, mit dem die Umrandung einer Zelle oder eines Zellbereichs eingestellt werden kann. Das Border-Interface enthält die komplette Information für die Umrandungslinien. Die Linien sind ihrerseits Objekte, folgende Linien-Objekte gibt es:

TopLine

BottomLine

LeftLine

RightLine

HorizontalLine

VerticalLine

Die Linien-Objekte können mit folgenden Eigenschaften angesprochen werden:

Style, Color

KLASSE:
RangeSelection

SYNTAX:
Border

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleBorder
Documents.add "sCalc4"
With ActiveDocument.Range("b2:c4").Border
   .TopLine.Style = 2 REM 1pt-dicke Linie
   .TopLine.Color = RGB(0,0,255) REM in Blau
   .BottomLine.Style = 2
   .BottomLine.Color = RGB(0,0,255)
   .LeftLine.Style = 2
   .LeftLine.Color = RGB(0,0,255)
   .RightLine.Style = 2
   .RightLine.Color = RGB(0,0,255)
   .HorizontalLine.Style = 2
   .HorizontalLine.Color = RGB(0,0,255) REM im starCalc auch
die inneren Linien
   .VerticalLine.Style = 2
   .VerticalLine.Color = RGB(0,0,255)
End With
ActiveDocument.Range("b2:c4").BackGroundColor =
RGB(255,128,128)
End Sub
```

Eigenschaften und Methoden mit "C"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "C" beginnen.

Calculate-Methode

Führt eine Neuberechnung des aktuellen Fensters oder des angegebenen Dokumentes durch. Es werden nur die unberechneten Zellen berechnet. Zum Berechnen aller Zellen verwenden Sie CalculateHard.

KLASSE:
TableDocument

SYNTAX:
Calculate()

BEISPIEL:

```
Sub ExampleCalculation
Documents.Add "sCalc4"
Wait 500
ActiveDocument.AutomaticCalculation = FALSE
ActiveTable.Cells(3,2).Insert( "Automatisch-Berechnen
abschalten.")
Range("a1").Insert("400")
Range("a2").Insert("=a1*11")
ActiveTable.Cells(3,3).Insert( "Bei Eingabe wird berechnet.")
Wait 500
Range("a1").Insert("123")
ActiveTable.Cells(3,4).Insert( "Hat sich ein Argument verän-
dert, wird nicht neu berechnet")
Wait 3000
ActiveTable.Cells(3,5).Insert("Neu Berechnen.")
Wait 500
ActiveDocument.Calculate
ActiveTable.Cells(3,6).Insert( "Automatisch-Berechnen wieder
anschalten.")
Wait 500
ActiveDocument.AutomaticCalculation = TRUE
Wait 3000
Range("a1").Insert("5500")
Wait 1000
ActiveTable.Cells(3,7).Insert( "Alles noch einmal durch-
rechnen.")
ActiveDocument.CalculateHard
End Sub
```

CalculateHard-Methode

Berechnet alle Formeln im Dokument neu.

KLASSE:
TableDocument

SYNTAX:
CalculateHard()

Beispiel:

```
Sub ExampleCalculation
Documents.Add "sCalc4"
Wait 500
ActiveDocument.AutomaticCalculation = FALSE
ActiveTable.Cells(3,2).Insert( "Automatisch-Berechnen
abschalten.")
Range("a1").Insert("400")
Range("a2").Insert("=a1*11")
ActiveTable.Cells(3,3).Insert( "Bei Eingabe wird berechnet.")
Wait 500
Range("a1").Insert("123")
ActiveTable.Cells(3,4).Insert( "Hat sich ein Argument verän-
dert, wird nicht neu berechnet")
Wait 3000
ActiveTable.Cells(3,5).Insert("Neu Berechnen.")
Wait 500
ActiveDocument.Calculate
ActiveTable.Cells(3,6).Insert( "Automatisch-Berechnen wieder
anschalten.")
Wait 500
ActiveDocument.AutomaticCalculation = TRUE
Wait 3000
Range("a1").Insert("5500")
Wait 1000
ActiveTable.Cells(3,7).Insert( "Alles noch einmal durch-
rechnen.")
ActiveDocument.CalculateHard
End Sub
```

CaseSensitive-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche Groß- und Kleinschreibung berücksichtigt werden soll. Diese Funktion entspricht der Suchoption Exakte Suche im Dialog Suchen & Ersetzen.

Klasse:
TableEditView

Syntax:
CaseSensitive

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob bei der Suche Groß- und Kleinschreibung berücksichtigt wird (True) oder nicht (False).

Beispiel:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

Cells-Methode

Gibt ein Range-Objekt für die angegebene Zelle des Dokuments zurück.

Klasse:
TableDocument

Syntax:
Cells(Column As Long, Row As Long, Table As Long)

Rückgabe:
Object

Parameter:
Column: Gibt die Spaltennummer an (1-256).

Row: Gibt die Zeilennummer an (1-32000).

Table: Gibt die Tabellennummer an (ab 1 gezählt).

Beispiel:

```
Sub ExampleCellText
Const sText1="Das ist ein "
Const sText2="Baum"
Dim sText As String
```

```
Documents.add "sCalc4"
ActiveDocument.Cells(1,1,1).Insert( sText1 )
ActiveDocument.Cells(2,1,1).Insert( sText2 )
REM Texte verketten mit einer Formel
ActiveDocument.Cells(2,3,1).Insert( "=VERKETTEN(A1;B1)" )
sText= ActiveDocument.CellText(2,3,1)
ActiveDocument.PutCell(2,4,1, sText )
End Sub
```

Cells-Methode

Die Methode gibt ein Range-Objekt für die angegebene (einzelne) Zelle zurück.

KLASSE:
TableSelection

SYNTAX:
Cells(Column As Long, Row As Long)

RÜCKGABE:
Object

PARAMETER:
Column: Spaltennummer (ab 1 gezählt)

Row: Zeilennummer (ab 1 gezählt)

Return: Range-Objekt

BEISPIEL:

```
Sub ExampleCells
Documents.add "sCalc4"
Wait 100
Dim iX As Integer, iY As Integer
Documents.add "sCalc4"
For iX = 1 To 10
   For iY = 1 To 10
      ActiveTable.Cells(iX,iY).Insert "Index= " &  iX & ":" &
iY
   Next iY
Next IX
end sub
```

CellText-Methode

Liefert den Inhalt einer Zelle als Zeichenfolge, wie in der Tabelle angezeigt, zurück.

Diese Methode hat die gleiche Funktion wie die TextValue-Eigenschaft. Im Vergleich bietet die CellText-Methode jedoch den Vorteil, daß sie deutlich schneller arbeitet.

KLASSE:
TableDocument

SYNTAX:
CellText(Column As Long, Row As Long, Table As Long)

RÜCKGABE:
String

PARAMETER:
Column: Long-Integer-Ausdruck, der die Spalte angibt, in der sich die Zelle befindet.

Row: Long-Integer-Ausdruck, der die Zeile angibt, in der sich die Zelle befindet.

[Table]: Long-Integer-Ausdruck, der den Index der Tabelle in der Tables-Collection angibt, auf welche sich die Parameter Column und Row beziehen. Die Angabe des Parameters Table ist optional und nur dann erforderlich, wenn sich die Zelle, deren Inhalt zurückgegeben werden soll, nicht in der aktuellen Tabelle befindet.

BEISPIEL:

```
Sub ExampleCellText
Const sText1="Das ist ein "
Const sText2="Baum"
Dim sText As String
Documents.Add "sCalc4"
ActiveDocument.Cells(1,1,1).Insert( sText1 )
ActiveDocument.Cells(2,1,1).Insert( sText2 )
REM Texte verketten mit einer Formel
ActiveDocument.Cells(2,3,1).Insert( "=VERKETTEN(A1;B1)" )
sText= ActiveDocument.CellText(2,3,1)
ActiveDocument.PutCell(2,4,1, sText )
End Sub
```

ChangeChartData-Methode

Ändert den Datenbereich für ein Chart im Dokument.

KLASSE:
TableDocument

SYNTAX:
ChangeChartData(Name As String, Range As String, ColHeaders As Boolean, RowHeaders As Boolean)

PARAMETER:
Name: Name des Charts (wie im Navigator angezeigt).

Range: String, der den Zellbereich angibt ("a1:a3", oder mehrere Bereiche durch Semikolon getrennt: "a1:a3;c1:c3").

ColHeaders: Flag, ob die erste Zeile Spaltenköpfe enthält (True) oder nicht (False).

RowHeaders: Flag, ob die erste Spalte Zeilenköpfe enthält (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleStarChart
Documents.Add("scalc4")
Wait 2000
REM Tabelle mit Daten füllen
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Frauen")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Männer")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("e5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
wait 2000
REM Daten verändern
Range("a4").Insert("Jungens")
Range("b4").Insert("1300")
Range("c4").Insert("150")
Range("d4").Insert( 0.5 )
wait 3000
Range("e5").UpdateChart(TRUE)
REM neue Daten hinzufügen
Range("b7").Insert( "Mädchen" )
Range("b8").Insert( "Babies" )
Range("c7").Insert( 23 )
```

```
Range("e8").Insert( 120 )
ActiveDocument.AddChartData( OLEObjects(1).Name , "b7:e8")
wait 3000
REM anderen Chartbereich festlegen
Range("c6").Insert("Dubai")
Range("d6").Insert("Paris")
Range("e6").Insert("Hong Kong")
ActiveDocument.ChangeChartData( OLEObjects(1).Name , "b6:e8" ,
TRUE, TRUE)
End Sub
```

ChangePrintArea-Methode

Legt Druckbereiche, Wiederholungszeilen und Wiederholungsspalten für die Tabelle fest.

KLASSE:
TableSelection

SYNTAX:
ChangePrintArea(PrintArea As String, PrintRepeatRow As String, PrintRepeatCol As String)

PARAMETER:
PrintArea: String, der einen oder mehrere Zellbereiche (mit Semikolon getrennt) für den Druckbereich angibt.

PrintRepeatRow: String, der einen Zellbereich für Wiederholungszeilen angibt.

PrintRepeatCol: String, der einen Zellbereich für Wiederholungsspalten angibt.

BEISPIEL:

```
Sub ExampleChangePrintArea
Documents.add "sCalc4"
Wait 100
ActiveTable.ChangePrintArea("B2:D5", "B2:D2", "B2:B5")
Wait 3000
ActiveTable.DeletePrintArea()
End Sub
```

CharSet-Eigenschaft

Bestimmt oder ermittelt den Zeichensatz der selektierten Zelle oder des selektierten Bereichs.

KLASSE:
TableFont

SYNTAX:
CharSet

RÜCKGABE:
Integer

PARAMETER:
[= Set]: Integer-Ausdruck, der den Zeichensatz bestimmt, der der Schrift zugewiesen werden kann:

0 - Voreinstellung, unbekannter Zeichensatz

1 - ANSI-Zeichensatz

2 - Macintosh-Zeichensatz

3 - IBM PC Zeichensatz 437

4 - IBM PC Zeichensatz 850

5 - IBM PC Zeichensatz 860

6 - IBM PC Zeichensatz 861

7 - IBM PC Zeichensatz 863

8 - IBM PC Zeichensatz 865

9 - System-Zeichensatz

10 - Symbol-Zeichensatz

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
```

```
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

Clear-Methode

Löscht den Inhalt respektive das Format des selektierten Bereiches.

KLASSE:
RangeSelection

SYNTAX:
Clear(Flags As String)

PARAMETER:
Flags: Zeichenfolgenausdruck, der angibt, ob das Format oder welche Inhalte des selektierten Bereiches gelöscht werden sollen. Wird Flag nicht übergeben, wird der Dialog INHALTE LÖSCHEN aufgerufen. Flag können die folgenden Werte zugewiesen werden:

A: Der gesamte Inhalt und das Format des selektierten Bereiches wird gelöscht.

S: Im selektierten Bereich werden lediglich alle Zeichenfolgen gelöscht. Zahlen, Daten, Zeiten, Notizen und Formate bleiben erhalten.

V: Im selektierten Bereich werden lediglich alle Zahlen gelöscht. Zeichenfolgen, Daten, Zeiten, Notizen und Formate bleiben erhalten.

D: Im selektierten Bereich werden lediglich alle Daten und Zeiten gelöscht. Zeichenfolgen, Zahlen, Notizen und Formate bleiben erhalten.

N: Im selektierten Bereich werden lediglich alle Notizen gelöscht. Zeichenfolgen, Daten, Zeiten, Zahlen und Formate bleiben erhalten.

F: Das Format des selektierten Bereiches wird gelöscht. Zeichenfolgen, Daten, Zeiten, Notizen und Zahlen bleiben erhalten.

BEISPIEL:

```
Sub ExampleClear
Documents.Add("scalc4")
Range("a1").Insert("314") REM Das wird wieder gelöscht.
Range("b1").Insert("=a1")
```

```
Range("a2").Insert("Not a Number")
Range("b2").Insert("REM 555")
Range("a1:b2").Select
Range("a1:b2").Clear("V") REM Nur Zahlen löschen
End Sub
```

ClearArrowDependents-Methode

Entfernt die Spur der aktuellen Zelle zu ihrem Nachfolger.

KLASSE:
RangeSelection

SYNTAX:
ClearArrowDependents()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ClearArrowPrecedents-Methode

Entfernt die Spur der aktuellen Zelle zu ihrem Vorgänger.

KLASSE:
RangeSelection

SYNTAX:
ClearArrowPrecedents()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ClearArrows-Methode

Löscht alle Detektivlinien innerhalb der aktuellen Tabelle.

Methoden, die für die TableSelection gelten, lassen sich auch auf die Tables-Collection übertragen. So lassen sich mit der ClearArrows-Methode beispielsweise Detektivlinien einer Tabelle löschen, die nicht der aktiven Tabelle entspricht. So wird z.B. mit "ActiveDocument.Tables(2).ClearArrows" die zweite Tabelle im StarCalc-Dokument angesprochen.

KLASSE:
TableSelection

SYNTAX:
ClearArrows()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
```

```
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ClearContents-Methode

Löscht den gesamten Zelleninhalt des selektierten Bereiches. Formatierungen werden nicht gelöscht.

KLASSE:
RangeSelection

SYNTAX:
ClearContents()

BEISPIEL:

```
Sub ExampleClearContents
Documents.Add("scalc4")
Range("a1").Insert("314")
Range("a2").Insert("Not a Number")
Range("a2").BackgroundColor = RGB(255,0,0)
REM Inhalte ohne Formate löschen
Range("a1:b2").ClearContents
End Sub
```

ClearOutline-Methode

Entfernt alle Gliederungen von einer Tabelle. Dabei kann es sich um die aktive Tabelle handeln (siehe Beispiel) oder um eine bestimmte Tabelle des aktiven Dokuments. So löscht beispielsweise "ActiveDocument.Tables("Table2").ClearOutline" alle Gliederungen in der Tabelle "Table2".

KLASSE:
TableSelection

SYNTAX:
ClearOutline()

BEISPIEL:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

Color-Eigenschaft

Bestimmt oder ermittelt die Farbe der Schrift im selektierten Bereich.

KLASSE:
TableFont

SYNTAX:
Color

RÜCKGABE:
Long

PARAMETER:
[= Value]: Numerischer Ausdruck, der die Schriftfarbe im selektierten Bereich angibt.

Beispiel:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

Column-Eigenschaft

Die Eigenschaft gibt die Nummer der letzten Spalte des Zellbereichs eines Bereichs an (ab 1 gezählt).

Klasse:
RangeSelection

Syntax:
Column

Rückgabe:
Long

Parameter:
(nur lesen)

Beispiel:

```
REM Folgendes Beispiel rechnet einen Bereich in Integer-Größen
um.
REM Einige Funktionen wie Cells(...) benötigen Integer-Werte um
Zellen zu referenzieren,
REM dazu kann man an zentraler Stelle den zu bearbeiteten
Bereich in Indexgrenzen umrechnen.
Sub ExampleRCT
DIM iLeft As Integer,iTop As Integer,iRight As Integer,iBottom
As Integer, iTable As Integer
```

```
DIM sText As String
Documents.Add("scalc4")
Wait 20
Range("b2:c4").Select
sText = Selection.Address REM gibt den selektierten Bereich als
Bereichstext zurück
AddressRC( sText, iLeft, iTop, iRight, iBottom, iTable )   REM
Wait 1000
ActiveDocument.Cells(1,5,1).Insert("Index")
ActiveDocument.Cells(2,5,1).Insert("Wert")
ActiveDocument.Cells(1,6,1).Insert("Links")
ActiveDocument.Cells(2,6,1).Insert(iLeft)
ActiveDocument.Cells(1,7,1).Insert("Rechts")
ActiveDocument.Cells(2,7,1).Insert(iRight)
ActiveDocument.Cells(1,8,1).Insert("Oben")
ActiveDocument.Cells(2,8,1).Insert(iTop)
ActiveDocument.Cells(1,9,1).Insert("Unten")
ActiveDocument.Cells(2,9,1).Insert(iBottom)
ActiveDocument.Cells(1,10,1).Insert("Tabellenindex")
ActiveDocument.Cells(2,10,1).Insert(iTable)
End Sub
Sub AddressRC(ByVal srAddress  As String, srLeft As Integer,
srTop As Integer, srRight As Integer, srBottom As Integer,
srTab As integer)
DIM iC As Integer, sText As String
iC = instr( 1, srAddress , ".")
If iC > 0 Then
srTab = Range( srAddress ).Table
srAddress = Mid(  srAddress, iC+1 )
Else
srTab = 0 REM Bereichstext enthält keinen Tabellen-Namen
End If
iC = instr( 1, srAddress , ":")
If iC > 0 Then
sText = Left( srAddress, iC -1 ) REM Start des Bereiches
srLeft = Range(sText).Column
srTop = Range(sText).Row
End If
srRight  = Range(srAddress).Column  REM End des Bereiches
srBottom = Range(srAddress).Row
End Sub
```

ColumnWidth-Eigenschaft

Bestimmt oder ermittelt die Spaltenbreite der aktiven Zelle. Bei einem selektierten oder angegebenen Bereich wird die Spaltenbreite der rechten unteren Zelle des Bereichs bestimmt.

KLASSE:
RangeSelection

SYNTAX:
ColumnWidth

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Spaltenbreite in Twips angibt.

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount as Integer
Documents.Add("scalc4")
Wait 1000
with ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
```

```
Next
End With
End Sub
```

ConditionalFormatDialog-Methode

Ruft den Dialog BEDINGTE FORMATIERUNG auf.

Es ist zu beachten, daß die Einstellungen des Dialoges nur für den selektierten Zellbereich gültig sind. Um eine bedingte Formatierung durchzuführen, ist es daher notwendig, vor Aufruf dieser Methode den zu formatierenden Bereich zu selektieren. Zum Beispielt bedeutet der Befehl "Range("a1:a4").ConditionalFormatDialog" nicht, daß eine Formatierung auf den beim Range-Objekt angegebenen Zellbereich angewendet wird, es wird lediglich der Dialog aufgerufen, und der zu formatierende Bereich muß vorher festgelegt werden (siehe Beispiel).

KLASSE:
RangeSelection

SYNTAX:
ConditionalFormatDialog()

BEISPIEL:

```
Sub ExampleConditionalFormatDialog
Documents.Add("scalc4")
Range("a2:b2").Select
Range("a2:b2").ConditionalFormatDialog
End Sub
```

Copy-Methode

Kopiert den Inhalt des selektierten Bereichs in die Zwischenablage.

KLASSE:
RangeSelection

SYNTAX:
Copy()

BEISPIEL:

```
Sub ExampleCopyPaste
Documents.Add("scalc4")
ActiveCell.Insert("10")
ActiveDocument.PutCell(1,2,1,"=a1 * 2")
ActiveDocument.Cells(1,2,1).ToggleRelative REM absolute Refe-
renzen verwenden
```

```
ActiveDocument.Cells(1,2,1).Copy
ActiveDocument.Cells(3,4,1).Paste
End Sub
```

CreateNames-Methode

Ruft den Dialog NAMEN ERZEUGEN auf.

KLASSE:
RangeSelection

SYNTAX:
CreateNames()

BEISPIEL:

```
Sub ExampleCreateNames
Const sText1="Hund"
Const sText2="Katze"
Documents.Add("scalc4")
Range("b2").Insert(sText1)
Range("c2").Insert(sText2)
Range("b3").Insert("1000")
Range("c3").Insert("2000")
Range("b2:c3").Select REM damit man sieht worauf der Dialog
wirkt
Range("b2:c3").CreateNames
end sub
```

CursorEndOfScreen-Methode

Setzt den Cursor in die letzte Zeile, die gerade in der View angezeigt wird.

KLASSE:
TableEditView

SYNTAX:
CursorEndOfScreen()

BEISPIEL:

```
Sub ExampleCursorOfSreen
Const sText = "Hier steht die Einfügemarke"
Documents.add "sCalc4"
Wait 1000
Activewindow.CursorEndOfScreen
ActiveCell.Insert sText
Wait 1000
Activewindow.CursorTopOfScreen
ActiveCell.Insert sText
End Sub
```

CursorTopOfScreen-Methode

Setzt den Cursor in die erste Zeile, die gerade in der View angezeigt wird.

KLASSE:
TableEditView

SYNTAX:
CursorTopOfScreen()

BEISPIEL:

```
Sub ExampleCursorOfSreen
Const sText = "Hier steht die Einfügemarke"
Documents.add "sCalc4"
Wait 1000
Activewindow.CursorEndOfScreen
ActiveCell.Insert sText
Wait 1000
Activewindow.CursorTopOfScreen
ActiveCell.Insert sText
End Sub
```

Cut-Methode

Kopiert den Inhalt des selektierten Bereichs in die Zwischenablage und löscht diesen anschließend.

KLASSE:
RangeSelection

SYNTAX:
Cut()

BEISPIEL:

```
Sub ExamplePrintArea
Documents.Add("scalc4")
ActiveCell.Insert("=a2+1")
ActiveCell.Cut
ActiveDocument.Range("a1:h8").Paste
ActiveDocument.Range("a1:b2").AddPrintArea
ActiveDocument.Range("c3:d4").AddPrintArea
ActiveDocument.Range("e5:f6").AddPrintArea
ActiveDocument.Range("g7:h8").AddPrintArea
ActiveDocument.Range("c3:d4").PrintPreview = TRUE
End Sub
```

Eigenschaften und Methoden mit "D"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "D" beginnen.

DataAreaRefresh-Methode

Wiederholt Datenbank-Import, Filter, Sortierung, Teilergebnisse und Datenpilot-Tabellen für einen Zellbereich, wenn es sich um einen Datenbankbereich handelt, an dem mindestens eine der Operationen angewendet wurde.

Diese Methode funktioniert nur, wenn sie an einem RangeSelection-Objekt aufgerufen wird, das der ersten Zelle des angelegten Datenbankbereichs entspricht. Wird beispielsweise ein Datenbank-Import durchgeführt, so muß das RangeSelection-Objekt bei Aufruf von DataAreaRefresh die erste, also die linke obere Zelle des importierten Bereichs beinhalten. Andernfalls kommt es zu einem Laufzeitfehler.

KLASSE:
RangeSelection

SYNTAX:
DataAreaRefresh()

BEISPIEL:

```
Sub ExampleDataImport
Const SQLStatement="SELECT * FROM ""Adressen"""
Documents.Add("scalc4")
Wait 1000
REM Datenimport aus Adressenliste vornehmen
Range("b2:c8").DataImport("Adreßbuch",SQLStatement)
Wait 1000
Range("b2").DataAreaRefresh
Wait 1000
Range("b2").DataReImport
End Sub
```

DataConsolidate-Methode

Ruft den Dialog KONSOLIDIEREN auf.

KLASSE:
RangeSelection

SYNTAX:
DataConsolidate()

Beispiel:

```
Sub ExampleDataConsolidate
Documents.Add("scalc4")
Wait 1000
Range("c3:e5").Select
Range("c3:e5").DataConsolidate
End Sub
```

DataDataPilotRun-Methode

Ruft den Dialog Datenpilot auf.

Klasse:
RangeSelection

Syntax:
DataDataPilotRun()

Beispiel:

```
Sub ExampleDataDataPilotRun
Documents.Add("sCalc4")
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataDataPilotRun
End Sub
```

DataFilterAutoFilter-Methode

Schaltet für den Zellbereich Autofilter ein oder wieder aus.

Klasse:
RangeSelection

Syntax:
DataFilterAutoFilter()

Beispiel:

```
Sub ExampleDataFilterAutoFilter
Documents.Add("scalc4")
Wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataFilterAutoFilter
ActiveWindow.Deselect
```

```
wait 2000
Range("a1:c5").DataFilterHideAutoFilter
End Sub
```

DataFilterHideAutoFilter-Methode

Blendet Autofilter aus, ohne die Filterung aufzuheben.

KLASSE:
RangeSelection

SYNTAX:
DataFilterHideAutoFilter()

BEISPIEL:

```
Sub ExampleDataFilterAutoFilter
Documents.Add("scalc4")
Wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataFilterAutoFilter
ActiveWindow.Deselect
wait 2000
Range("a1:c5").DataFilterHideAutoFilter
End Sub
```

DataFilterRemoveFilter-Methode

Hebt eine Filterung für einen Bereich auf.

KLASSE:
RangeSelection

SYNTAX:
DataFilterRemoveFilter()

BEISPIEL:

```
REM Im angegebenen Bereich muß bereits eine Filterung vorhanden
sein.
Range("a1:c5").DataFilterRemoveFilter
```

DataFilterSpecialFilter-Methode

Ruft den Dialog SPEZIALFILTER auf.

KLASSE:
RangeSelection

SYNTAX:
DataFilterSpecialFilter()

BEISPIEL:

```
Sub ExampleDataFilterSpecialFilter
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataFilterSpecialFilter
End Sub
```

DataFilterStandardFilter-Methode

Ruft den Dialog STANDARDFILTER auf. Dabei bestimmt die aktuelle Cursorposition die Spalte, die in dem Dialog zur Definition eines Filterkriteriums als Feldname eingetragen wird.

KLASSE:
RangeSelection

SYNTAX:
DataFilterStandardFilter()

BEISPIEL:

```
Sub ExampleDataFilterStandardFilter
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataFilterStandardFilter
End Sub
```

DataImport-Methode

Importiert Daten aus der angegebenen Datenbank in einen Zellbereich. Dabei gibt die erste Zelle (oben links) des angegebenen oder selektierten Zellbereichs an, wo der Datenbankbereich in der Tabelle angelegt werden soll. Weitere Bereichsangaben werden vom Import ignoriert, d.h. der Datenbankbereich, der im Tabellendokument angelegt wird, wird durch die importierten Daten bestimmt.

Siehe auch: DataReImport

KLASSE:
RangeSelection

SYNTAX:
DataImport(DatabaseName As String, SqlCommand As String)

PARAMETER:
DatabaseName: Gibt den Namen der Datenbank an, aus der die Daten importiert werden sollen.

SqlCommand: Gibt einen SQL-String an, der zur Filterung der Daten ausgeführt werden soll.

BEISPIEL:

```
Sub ExampleDataImport
Const SQLStatement="SELECT * FROM ""Adressen"""
Documents.Add("scalc4")
Wait 1000
REM Datenimport aus Adressenliste vornehmen
Range("b2:c8").DataImport("Adreßbuch",SQLStatement)
Wait 1000
Range("b2").DataAreaRefresh
Wait 1000
Range("b2").DataReImport
End Sub
```

DataReImport-Methode

Wiederholt eine externe Abfrage für importierte Daten, wenn diese mit der DataImport-Methode importiert wurden.

Entspricht bei Aufruf dieser Methode am Range-Objekt der angegebene Zellbereich nicht dem importierten Datenbankbereich, so kommt es zu einem Laufzeitfehler. Wird als Range-Objekt dagegen die erste, also die linke obere Zelle des Datenbankbereichs genommen, wird der in der Tabelle angelegte Datenbankbereich automatisch richtig erkannt.

Siehe auch: DataImport

KLASSE:
RangeSelection

SYNTAX:
DataReImport()

Beispiel:

```
Sub ExampleDataImport
Const SQLStatement="SELECT * FROM ""Adressen"""
Documents.Add("scalc4")
Wait 1000
REM Datenimport aus Adressenliste vornehmen
Range("b2:c8").DataImport("Adreßbuch",SQLStatement)
Wait 1000
Range("b2").DataAreaRefresh
Wait 1000
Range("b2").DataReImport
End Sub
```

DataSelect-Methode

Öffnet das Popup-Menü für die Auswahlliste. In einem Tabellendokument erhält man dieses Popup-Menü, wenn man im Kontextmenü einer Zelle, den Befehl Auswahlliste... aufruft. Eine Auswahlliste wird einem stets dann angeboten, wenn es innerhalb derselben Spalte zur aktiven Zelle Nachbarzellen gibt, die einen String enthalten.

Klasse:
TableEditView

Syntax:
DataSelect()

Beispiel:

```
Sub ExampleDataSelect
Const sText="Eintrag"
Dim iCount As Integer
Documents.add "sCalc4"
For iCount=1 To 5
Activedocument.Cells(2, iCount).Insert sText & iCount
next iCount
Activewindow.GotoCell(Activedocument.Cells(2, iCount).Address)
Activewindow.DataSelect
end sub
```

DataSort-Methode

Ruft den Dialog Sortieren auf. Der beim Range-Objekt angegebene Zellbereich wird dabei berücksichtigt. Er bestimmt die erste Spalte, in der eine Sortierung durchgeführt werden soll.

Klasse:
RangeSelection

Syntax:
DataSort()

Beispiel:

```
Sub ExampleDataSort
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataSort
End Sub
```

DataSubTotals-Methode

Ruft den Dialog Teilergebnis auf. Der beim Range-Objekt angegebene Zellbereich wird dabei berücksichtigt.

Klasse:
RangeSelection

Syntax:
DataSubTotals()

Beispiel:

```
Sub ExampleDataSubTotals
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DataSubTotals
End Sub
```

Deactivate-Methode

Deaktiviert ein OLE-Objekt, wenn es gerade InPlace-aktiv ist. InPlace-aktiv bedeutet hierbei, daß das OLE-Objekt bearbeitbar ist, ein lediglich selektiertes OLE-Objekt ist nicht InPlace-aktiv.

Klasse:
TableOLEObject

Syntax:
Deactivate()

Beispiel:

```
Sub ExampleActivate
Dim oDoc as Object
Documents.Add("scalc4")
Wait 1000
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Männer")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Frauen")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("b5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
Wait 2000
oDoc = ActiveDocument REM Objekt merken
oDoc.OLEObjects(1).Activate
wait 5000
oDoc.OLEObjects(1).Deactivate
End Sub
```

DecrementIndent-Methode

Setzt die horizontale Ausrichtung auf linksbündig und vermindert den Einzug um 10pt.

Klasse:
RangeSelection

Syntax:
DecrementIndent()

Beispiel:

```
Sub ExampleIncrementIndent
Documents.Add("scalc4")
ActiveCell.Insert("X")
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.DecrementIndent
```

```
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
End Sub
```

DefineDBName-Methode

Ruft den Dialog DATENBANKBEREICH FESTLEGEN auf.

KLASSE:
RangeSelection

SYNTAX:
DefineDBName()

BEISPIEL:

```
Sub ExampleDefineDBName
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DefineDBName
End Sub
```

DefineLabelRange-Methode

Ruft den Dialog BESCHRIFTUNGSBEREICH FESTLEGEN auf. Die Einstellungen dieses Dialoges gelten nur für einen zuvor selektierten Zellbereich.

KLASSE:
RangeSelection

SYNTAX:
DefineLabelRange()

BEISPIEL:

```
Sub ExampleDefineLabelRange
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
```

```
Range("a1:c5").Select
Range("a1:c5").DefineLabelRange
End Sub
```

DefineName-Methode

Definiert einen benannten Bereich. Der Bereich, an dem die Methode aufgerufen wird, gibt bei relativen Referenzen an, auf welche Zelle sich die Referenz bezieht.

KLASSE:
RangeSelection

SYNTAX:
DefineName(Name As String, Symbol As String, Options As String)

PARAMETER:
Name: Name für den Bereich

Symbol: Inhalt des Bereichsnamens

Option: Flags für den Bereich; Leerstring oder Kombination aus:

P: Druckbereich

R: Wiederholungszeile

C: Wiederholungsspalte

F: Filterbereich

Ohne Parameter wird der Dialog NAMEN FESTLEGEN aufgerufen.

BEISPIEL:

```
Sub ExampleDefineName
Const sName="BereichsName"
Documents.Add("scalc4")
wait 1000
Range("a1").Insert("Hund")
Range("b1").Insert("Katze")
Range("c1").Insert("Maus")
Range("a1:c5").Select
Range("a1:c5").DefineName( sName, "$a$1:$g$7" , "P" )
Range("a1:c5").DefineName()
End Sub
```

DefinePrintArea-Methode

Legt den Bereich als (einzigen) Druckbereich für die zugehörige Tabelle fest.

KLASSE:
RangeSelection

SYNTAX:
DefinePrintArea()

BEISPIEL:

```
Sub ExampleDefinePrintArea
Documents.Add("scalc4")
Wait 500
Range("a1:c5").DefinePrintArea()
End Sub
```

Delete-Methode

Löscht die selektierten Zellen.

KLASSE:
RangeSelection

SYNTAX:
Delete(Flags As String)

PARAMETER:
Flags: Zeichenfolgenausdruck, der angibt, wie die Zellen gelöscht werden:

U: Löscht die selektierten Zellen, indem die darunter angeordneten Zellen nach oben, über die selektierten Zellen verschoben werden.

L: Löscht die selektierten Zellen, indem die rechts davon angeordneten Zellen nach links, über die selektierten Zellen verschoben werden.

R: Löscht alle Tabellenzeilen, in denen die selektierten Zellen angeordnet sind.

C: Löscht alle Tabellenspalten, in denen die selektierten Zellen angeordnet sind.

BEISPIEL:

```
Sub ExampleDelete
Documents.Add("scalc4")
Wait 10
Range("a1").Insert("Text_A")
Range("b1").Insert("Text_B")
Range("c1").Insert("Text_C")
```

```
Range("a2").Insert("Neu_A")
Range("b2").Insert("Wird gelöscht")
Range("c2").Insert("Neu_C")
Wait 2000
Range("b2").Delete("L")
End Sub
```

DeleteColumnbreak-Methode

Löscht den Spaltenumbruch vor der aktiven Zelle. Bei einem selektierten oder angegebenen Bereich wird der Spaltenumbruch vor der rechten unteren Zelle des Bereichs gelöscht.

KLASSE:
RangeSelection

SYNTAX:
DeleteColumnbreak()

BEISPIEL:

```
Sub ExampleBreaks
Documents.Add("sCalc4")
wait 1000
Range("b2").Select
ActiveCell.InsertRowBreak
ActiveCell.InsertColumnBreak
wait 2000
ActiveCell.DeleteRowBreak
ActiveCell.DeleteColumnBreak
End Sub
```

DeleteColumns-Methode

Löscht die aktuelle Tabellenspalte oder die Spalten des angegebenen oder selektierten Bereichs.

KLASSE:
RangeSelection

SYNTAX:
DeleteColumns()

BEISPIEL:

```
Sub ExampleDeleteColumns
Documents.Add("scalc4")
Range("a1").Insert("Text_A")
Range("b1").Insert("Wird gelöscht")
Range("c1").Insert("Text_C")
Range("a2").Insert("Neu_A")
Range("b2").Insert("Wird gelöscht")
```

```
Range("c2").Insert("Neu_C")
Wait 2000
Range("b2").Select
Range("b2").DeleteColumns
End Sub
```

DeletePrintArea-Methode

Hebt Druckbereiche, die für die Tabelle definiert waren, wieder auf. Wenn keine Druckbereiche definiert sind, gibt es eine Fehlermeldung.

KLASSE:
TableSelection

SYNTAX:
DeletePrintArea()

BEISPIEL:

```
Sub ExampleChangePrintArea
Documents.add "sCalc4"
Wait 100
ActiveTable.ChangePrintArea("B2:D5", "B2:D2", "B2:B5")
Wait 3000
ActiveTable.DeletePrintArea()
End Sub
```

DeleteRowbreak-Methode

Löscht den Zeilenumbruch vor der aktiven Zelle. Bei einem selektierten oder angegebenen Bereich wird der Zeilenumbruch vor der rechten unteren Zelle des Bereichs gelöscht.

KLASSE:
RangeSelection

SYNTAX:
DeleteRowbreak()

BEISPIEL:

```
Sub ExampleBreaks
Documents.Add("sCalc4")
wait 1000
Range("b2").Select
ActiveCell.InsertRowBreak
ActiveCell.InsertColumnBreak
wait 2000
ActiveCell.DeleteRowBreak
ActiveCell.DeleteColumnBreak
End Sub
```

DeleteRows-Methode

Löscht die aktuelle Tabellenzeile oder die Zeilen des angegebenen oder selektierten Bereichs.

KLASSE:
RangeSelection

SYNTAX:
DeleteRows()

BEISPIEL:

```
Sub ExampleDeleteRows
Documents.Add("scalc4")
Range("a1").Insert("Text_1")
Range("a2").Insert("Wird gelöscht")
Range("a3").Insert("Text_3")
Range("b1").Insert("Neu_1")
Range("b2").Insert("Wird gelöscht")
Range("b3").Insert("Neu_3")
Wait 2000
Range("b2").Select
Range("b2").DeleteRows
End Sub
```

DeleteTemplate-Methode

Löscht eine Zell- oder Seitenvorlage des Dokuments, zu dem der Range gehört. Dabei spielt es keine Rolle, welcher Zellbereich beim Range-Objekt übergeben wird. Die Vorlage bezieht sich auf das gesamte Dokument.

KLASSE:
RangeSelection

SYNTAX:
DeleteTemplate(Param As String, Family As Long)

PARAMETER:
Param: Gibt den Namen der Vorlage an, die gelöscht werden soll.

Family: Bestimmt die Art der Vorlage. Folgende Werte sind möglich:

2 - Zellvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExampleDeleteTemplate
Const sTemp ="Neue Zell-Vorlage"
Documents.Add("scalc4")
```

```
Application.DesignerDialog = TRUE
Wait 2000
ActiveCell.StyleNewByExample(sTemp,2) REM legt eine Vorlage an
Wait 2000
Range("u45:v45").DeleteTemplate(sTemp,2) REM löscht sie wieder
Application.DesignerDialog = TRUE
End Sub
```

Deselect-Methode

Hebt die aktuelle Selektion auf.

KLASSE:
TableEditView

SYNTAX:
Deselect()

BEISPIEL:

```
Sub ExampleSelect
Documents.Add "sCalc4"
Wait 100
Range("B2:D5").Select
Wait 2000
ActiveWindow.Deselect
End Sub
```

Eigenschaften und Methoden mit "E"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "E" beginnen.

EditHeaderAndFooter-Methode

Öffnet den Dialog zum Bearbeiten der Kopf- und Fußzeilen.

Obwohl diese Methode an einem Objekt der Klasse TableSelection aufgerufen wird, werden Kopf- und Fußzeilen stets für das Tabellendokument festgelegt. Da sie für die Seitenvorlage eingestellt werden, gelten die Einstellungen unabhängig von der aktiven Tabelle für das gesamte Dokument.

KLASSE:
TableSelection

SYNTAX:
EditHeaderAndFooter()

Beispiel:

```
Sub ExampleEditHeaderAndFooter
Documents.Add "sCalc4"
Wait 100
ActiveTable.EditHeaderAndFooter
End Sub
```

EditLinks-Methode

Öffnet den Dialog Verknüpfungen bearbeiten für das Dokument, zu dem der Range gehört. Es müssen Verknüpfungen definiert sein, sonst gibt es eine Fehlermeldung.

Klasse:
RangeSelection

Syntax:
EditLinks()

Beispiel:

```
Sub ExampleInsertGraphic
Dim sGrfPath As String
Documents.Add("scalc4")
Wait 1000
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Range("d5").InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Range("a1").EditLinks
End Sub
```

EditPrintArea-Methode

Ruft den Dialog Druckbereiche bearbeiten für die aktuelle Tabelle auf.

Klasse:
TableSelection

Syntax:
EditPrintArea()

Beispiel:

```
Sub ExampleEditPrintArea
Documents.Add "sCalc4"
Wait 100
ActiveTable.EditPrintArea
End Sub
```

EditTemplate-Methode

Öffnet den Dialog zum Bearbeiten der angegebenen Zell- oder Seitenvorlage für das Dokument, zu dem der Range gehört.

KLASSE:
RangeSelection

SYNTAX:
EditTemplate(Param As String, Family As Long)

PARAMETER:
Param: Name der Vorlage

Family: Bestimmt die Art der Vorlage:

2 - Zellvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExampleEditTemplate
Const sTemp ="Neue Zell-Vorlage"
Documents.Add("scalc4")
Wait 500
ActiveCell.StyleNewByExample(sTemp,2) REM legt eine Vorlage an
Range("u45:v45").EditTemplate(sTemp,2) REM öffnet den Dialog
zur Bearbeitung
End Sub
```

Eigenschaften und Methoden mit "F"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "F" beginnen.

Family-Eigenschaft

Bestimmt oder ermittelt die Schriftfamilie der selektierten Zelle oder des selektierten Bereichs.

KLASSE:
TableFont

SYNTAX:
Family

RÜCKGABE:
Integer

PARAMETER:
[= Group]: Integer-Ausdruck, der die Schriftfamilie kennzeichnet, die der Schriftart zugewiesen werden kann:

0 - unbekannte Schriftartgruppe (Voreinstellung)

1 - Decorative

2 - Modern

3 - Roman

4 - Script

5 - Swiss

6 - System

Wird eine Schriftart gewählt, die nicht vorhanden ist, dann wird eine adäquate Schriftart aus der angegebenen Schriftfamilie verwendet.

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

FamilyName-Eigenschaft

Bestimmt oder ermittelt den Namen der Schrift der selektierten Zelle oder des selektierten Bereichs.

KLASSE:
TableFont

SYNTAX:
FamilyName

RÜCKGABE:
String

PARAMETER:
[= String]: Zeichenfolge, die den Namen der Schriftfamilie angibt, z. B. "Arial".

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

FillDown-Methode

Füllt die selektierten Zellen oder den angegebenen Zellbereich mit dem Inhalt und dem Format der Zellen in der obersten Zeile der Selektion oder des Bereichs aus.

KLASSE:
RangeSelection

Syntax:
FillDown()

Beispiel:

```
Sub ExampleFill
Documents.Add("sCalc4")
Wait 1000
Range("g5").Font.Underline = 2
Wait 10
Range("g5").Insert(100)
Range("a1:g5").FillLeft
Wait 10
Range("g1:g5").FillUp
Wait 10
Range("g5").BackgroundColor = RGB(255,255,0)
Wait 10
Range("g5:g10").FillDown
Wait 10
Range("g5:k5").FillRight
End Sub
```

FillLeft-Methode

Füllt die selektierten Zellen oder den angegebenen Zellbereich mit dem Inhalt und Format der Zellen in der äußerst rechten Spalte der Selektion oder des Bereichs aus.

Klasse:
RangeSelection

Syntax:
FillLeft()

Beispiel:

```
Sub ExampleFill
Documents.Add("sCalc4")
Wait 1000
Range("g5").Font.Underline = 2
Wait 10
Range("g5").Insert(100)
Range("a1:g5").FillLeft
Wait 10
Range("g1:g5").FillUp
Wait 10
Range("g5").BackGroundColor = RGB(255,255,0)
Wait 10
Range("g5:g10").FillDown
Wait 10
Range("g5:k5").FillRight
End Sub
```

FillRight-Methode

Füllt die selektierten Zellen oder den angegebenen Zellbereich mit dem Inhalt und Format der Zellen in der äußerst linken Spalte der Selektion oder des Bereichs aus.

KLASSE:
RangeSelection

SYNTAX:
FillRight()

BEISPIEL:

```
Sub ExampleFill
Documents.Add("sCalc4")
Wait 1000
Range("g5").Font.Underline = 2
Wait 10
Range("g5").Insert(100)
Range("a1:g5").FillLeft
Wait 10
Range("g1:g5").FillUp
Wait 10
Range("g5").BackgroundColor = RGB(255,255,0)
Wait 10
Range("g5:g10").FillDown
Wait 10
Range("g5:k5").FillRight
End Sub
```

FillSeries-Methode

Füllt den Zellbereich nach angegebenen Regeln. Zur Definition einer Regel ist die Übergabe sämtlicher Parameter kaum erforderlich. Nicht benötigte Parameter sind daher optional. Wird kein Parameter übergeben, wird der Dialog REIHE FÜLLEN aufgerufen.

KLASSE:
RangeSelection

SYNTAX:
FillSeries(FillDir As String, FillCmd As String, FillStep As String, FillDateCmd As String, FillStart As String, FillMax As String)

PARAMETER:
FillDir: Gibt die Richtung an, in die gefüllt wird:

B - Nach unten

R - Nach rechts

T - Nach oben

L - Nach links

FillCmd: Bestimmt den Füll-Modus:

S - Einfach (die erste Zeile/Spalte wird kopiert)

L - Linear (um einen festen Wert erhöhen)

G - Geometrisch (mit einem Wert multiplizieren)

D - Datum

A - Automatisch (die belegten Zellen werden analysiert, um zu erkennen, wie gefüllt werden soll)

FillStep: Bestimmt die Schrittweite für die Füll-Modi L, G und D.

FillDateCmd: Bestimmt die Einheit beim Füllen nach Datum:

D - Um FillStep Tage erhöhen

W - Dito, aber Samstage und Sonntage überspringen

M - Um FillStep Monate erhöhen

Y - Um FillStep Jahre erhöhen

FillStart: Legt den Startwert fest. Wird dieser nicht angegeben, wird der Startwert aus der Tabelle ausgelesen.

FillMax: Bestimmt den Endwert für die Füll-Modi L, G und D.

Beispiel:

```
Sub ExampleFillSeries
Documents.Add("scalc4")
ActiveCell.Insert("T1")
REM Automatisch füllen ..
Range("a1:e1").FillSeries("A")
REM Nach rechts linear mit Startwert 10 und Increment 10 füllen
Range("a2:e2").FillSeries("R","L",10,"",10,"")
REM nach unten geometrisch mit Faktor 2 füllen
Range("a2:e5").FillSeries("B","G",2)  REM
Range("a1:e5").StarChartDialog
End Sub
```

FillUp-Methode

Füllt die selektierten Zellen oder den angegebenen Zellbereich mit dem Inhalt und Format der Zellen in der untersten Zeile der Selektion oder des Bereichs aus.

KLASSE:
RangeSelection

SYNTAX:
FillUp()

BEISPIEL:

```
Sub ExampleFill
Documents.Add("sCalc4")
Wait 1000
Range("g5").Font.Underline = 2
Wait 10
Range("g5").Insert(100)
Range("a1:g5").FillLeft
Wait 10
Range("g1:g5").FillUp
Wait 10
Range("g5").BackgroundColor = RGB(255,255,0)
Wait 10
Range("g5:g10").FillDown
Wait 10
Range("g5:k5").FillRight
End Sub
```

Font-Eigenschaft

Gibt ein Font-Objekt zurück, mit dem die Zeichenattribute der Selektion oder des Zellbereichs verändert werden können.

KLASSE:
RangeSelection

SYNTAX:
Font

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

Beispiel:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

FormatCellDialog-Methode

Ruft den Dialog Zellattribute auf. Die Einstellungen, die Sie hier vornehmen, gelten für das gesamte RangeSelection-Objekt.

Klasse:
RangeSelection

Syntax:
FormatCellDialog()

Beispiel:

```
Sub ExampleFormatCellDialog
Documents.Add "sCalc4"
Wait 100
Range("a1:b4").FormatCellDialog REM gilt für den ganzen Bereich
End Sub
```

Formula-Eigenschaft

Gibt für eine Formel-Zelle den Formel-String zurück.

KLASSE:
RangeSelection

SYNTAX:
Formula

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleValues
DIM sText as String
Documents.Add("scalc4")
Wait 500
ActiveCell.ColumnWidth = 3000
ActiveCell.Insert( "=Pi()" )
sText = ActiveDocument.Cells(1,1,1).TextValue
ActiveWindow.GoDown 1
ActiveCell.Insert( "TextValue="& sText )
sText = ActiveDocument.Cells(1,1,1).Value
ActiveWindow.GoDown 1
ActiveCell.Insert( "Value="& sText )
sText = ActiveDocument.Cells(1,1,1).Formula
ActiveWindow.GoDown 1
ActiveCell.Insert( "Formula="& sText )
end sub
```

FreezePanes-Methode

Fixiert das Fenster an der Cursorposition oder hebt die Fixierung wieder auf.

KLASSE:
TableEditView

SYNTAX:
FreezePanes()

BEISPIEL:

```
Sub ExampleFreezePanes
Documents.add "sCalc4"
Wait 100
Activewindow.GotoCell "F7"
Activewindow.FreezePanes
End Sub
```

Eigenschaften und Methoden mit "G"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "G" beginnen.

GoalSeekDialog-Methode

Ruft den Dialog Zielwertsuche auf. Bei Aufruf dieser Methode wird ein durch das Range-Objekt definierter Bereich, z.B. Range("a1:c3"), ignoriert, da die Zielwertsuche für eine einzige Zelle aufgerufen wird. Das ist die Zelle, die die für die Zielwertsuche relevante Formel enthält. Der Dialog wird stets für die selektierte oder aktive Zelle aufgerufen, die man im StarBasic mit Selection oder ActiveCell ansprechen kann.

Klasse:
RangeSelection

Syntax:
GoalSeekDialog()

Beispiel:

```
Sub ExampleGoalSeekDialog
Documents.add "sCalc4"
Wait 100
Activewindow.GotoCell "F7"
Activewindow.GoalSeekDialog
End Sub
```

GoDown-Methode

Bewegt die Einfügemarke um die angegebene Anzahl an Zellen nach unten.

Klasse:
TableEditView

Syntax:
GoDown(By As Integer, Sel As Boolean)

Parameter:
By: Integer-Ausdruck, der die Anzahl der Zellen angibt, um die die Einfügemarke nach unten bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

Beispiel:

```
Sub ExampleGoAhead
Documents.Add "sCalc4"
Wait 100
ActiveWindow.GoDown (2, True)
Wait 100
ActiveWindow.GoRight (2, True)
Wait 100
ActiveWindow.GoUp (2, True)
Wait 100
ActiveWindow.GoLeft (2, True)
End Sub
```

GoDownBlock-Methode

Bewegt die Einfügemarke um die angegebene Anzahl an Seiten nach unten.

Klasse:
TableEditView

Syntax:
GoDownBlock(By As Integer, Sel As Boolean)

Parameter:
By: Beliebiger Integer-Ausdruck, der die Anzahl der Seiten angibt, um die die Einfügemarke nach unten bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

Beispiel:

```
Sub ExampleGoBlock
Documents.Add "sCalc4"
ActiveCell.Insert("Seite") REM Damit den markierten Bereich
füllen
ActiveCell.Copy
Wait 10
ActiveWindow.GoDownBlock (1, True)
Wait 10
ActiveWindow.GoRightBlock (1, True)
Wait 10
Selection.Paste
ActiveWindow.GoUpBlock (1, True)
Wait 10
ActiveWindow.GoLeftBlock (1, True)
End Sub
```

GoDownToEndOfData-Methode

Bewegt die Einfügemarke innerhalb der aktuellen Spalte nach unten zum Ende des aktuellen Datenblocks. Befindet sich der Cursor bereits am Ende des Datenblocks, wird zum Anfang des nächsten Blocks gesprungen. Folgt der aktuellen Zelle keine weitere Zelle mit Inhalt, wird die Einfügemarke in die letzte Tabellenzeile gesetzt.

KLASSE:
TableEditView

SYNTAX:
GoDownToEndOfData(By As Integer, Sel As Boolean)

PARAMETER:
By: Gibt die Anzahl der Blöcke an, die mit dieser Methode übersprungen werden sollen. Da mehrere Blöcke nur schrittweise angesprungen werden können, muß hier eine 1 übergeben werden, also By = 1.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, erfolgt keine Markierung (Sel = False).

BEISPIEL:

```
Sub ExampleGoData
REM Es wird ein rechteckiger Bereich selektiert.
Const sText="Bereich" REM Wort ...
Dim iCount As Integer
Documents.Add "sCalc4"
For iCount=3 to 7
ActiveDocument.Cells(2, iCount).Insert sText & "_B" & iCount
ActiveDocument.Cells(3, iCount).Insert sText &"_C" & iCount
ActiveDocument.Cells(4, iCount).Insert sText &"_D" & iCount
Next iCount
ActiveWindow.GotoCell("c5")
ActiveWindow.GoDownToEndOfData (1, FALSE)
ActiveWindow.GoRightToEndOfData (1, FALSE)
ActiveWindow.GoUpToStartOfData (1, TRUE)
ActiveWindow.GoLeftToStartOfData (1, TRUE)
End Sub
```

GoLeft-Methode

Bewegt die Einfügemarke um die angegebene Anzahl an Zellen nach links.

KLASSE:
TableEditView

SYNTAX:
GoLeft(By As Integer, Sel As Boolean)

PARAMETER:
By: Integer-Ausdruck, der die Anzahl der Zellen angibt, um die die Einfügemarke nach links bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

BEISPIEL:

```
Sub ExampleGoAhead
Documents.Add "sCalc4"
Wait 100
ActiveWindow.GoDown (2, True)
Wait 100
ActiveWindow.GoRight (2, True)
Wait 100
ActiveWindow.GoUp (2, True)
Wait 100
ActiveWindow.GoLeft (2, True)
End Sub
```

GoLeftBlock-Methode

Bewegt die Einfügemarke um die angegebene Seitenanzahl nach links.

KLASSE:
TableEditView

SYNTAX:
GoLeftBlock(By As Integer, Sel As Boolean)

PARAMETER:
By: Beliebiger Integer-Ausdruck, der die Anzahl der Seiten angibt, um die die Einfügemarke nach links bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser

Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

Beispiel:

```
Sub ExampleGoBlock
Documents.Add "sCalc4"
ActiveCell.Insert("Seite") REM Damit den markierten Bereich
füllen
ActiveCell.Copy
Wait 10
ActiveWindow.GoDownBlock (1, True)
Wait 10
ActiveWindow.GoRightBlock (1, True)
Wait 10
Selection.Paste
ActiveWindow.GoUpBlock (1, True)
Wait 10
ActiveWindow.GoLeftBlock (1, True)
End Sub
```

GoLeftToStartOfData-Methode

Bewegt die Einfügemarke innerhalb der aktuellen Zeile nach links zum Anfang des aktuellen Datenblocks. Befindet sich der Cursor bereits am Anfang des Datenblocks, wird zum Ende des vorigen Blocks gesprungen. Befindet sich vor der aktuellen Zelle keine weitere Zelle mit Inhalt, wird die Einfügemarke in die erste Tabellenspalte gesetzt.

Klasse:
TableEditView

Syntax:
GoLeftToStartOfData(By As Integer, Sel As Boolean)

Parameter:
By: Gibt die Anzahl der Blöcke an, die mit dieser Methode übersprungen werden sollen. Da mehrere Blöcke nur schrittweise angesprungen werden können, muß hier eine 1 übergeben werden, also By = 1.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, erfolgt keine Markierung (Sel = False).

Beispiel:

```
Sub ExampleGoData
REM Es wird ein rechteckiger Bereich selektiert.
Const sText="Bereich" REM Wort ...
Dim iCount As Integer
```

```
Documents.Add "sCalc4"
For iCount=3 to 7
ActiveDocument.Cells(2, iCount).Insert sText & "_B" & iCount
ActiveDocument.Cells(3, iCount).Insert sText &"_C" & iCount
ActiveDocument.Cells(4, iCount).Insert sText &"_D" & iCount
Next iCount
ActiveWindow.GotoCell("c5")
ActiveWindow.GoDownToEndOfData (1, FALSE)
ActiveWindow.GoRightToEndOfData (1, FALSE)
ActiveWindow.GoUpToStartOfData (1, TRUE)
ActiveWindow.GoLeftToStartOfData (1, TRUE)
End Sub
```

GoRight-Methode

Bewegt die Einfügemarke um die angegebene Zellenanzahl nach rechts.

KLASSE:
TableEditView

SYNTAX:
GoRight(By As Integer, Sel As Boolean)

PARAMETER:
By: Integer-Ausdruck, der die Anzahl der Zellen angibt, um die die Einfügemarke nach rechts bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

BEISPIEL:

```
Sub ExampleGoAhead
Documents.Add "sCalc4"
Wait 100
ActiveWindow.GoDown (2, True)
Wait 100
ActiveWindow.GoRight (2, True)
Wait 100
ActiveWindow.GoUp (2, True)
Wait 100
ActiveWindow.GoLeft (2, True)
End Sub
```

GoRightBlock-Methode

Bewegt die Einfügemarke um die angegebene Seitenanzahl nach rechts.

KLASSE:
TableEditView

SYNTAX:
GoRightBlock(By As Integer, Sel As Boolean)

PARAMETER:
By: Beliebiger Integer-Ausdruck, der die Anzahl der Seiten angibt, um die die Einfügemarke nach rechts bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

BEISPIEL:

```
Sub ExampleGoBlock
Documents.Add "sCalc4"
ActiveCell.Insert("Seite") REM Damit den markierten Bereich
füllen
ActiveCell.Copy
Wait 10
ActiveWindow.GoDownBlock (1, True)
Wait 10
ActiveWindow.GoRightBlock (1, True)
Wait 10
Selection.Paste
ActiveWindow.GoUpBlock (1, True)
Wait 10
ActiveWindow.GoLeftBlock (1, True)
End Sub
```

GoRightToEndOfData-Methode

Bewegt die Einfügemarke innerhalb der aktuellen Zeile nach rechts zum Ende des aktuellen Datenblocks. Befindet sich der Cursor bereits am Ende des Datenblocks, wird zum Anfang des nächsten Blocks gesprungen. Befindet sich hinter der aktuellen Zelle keine weitere Zelle mit Inhalt, wird die Einfügemarke in die letzte Tabellenspalte gesetzt.

KLASSE:
TableEditView

SYNTAX:
GoRightToEndOfData(By As Integer, Sel As Boolean)

PARAMETER:
By: Gibt die Anzahl der Blöcke an, die mit dieser Methode übersprungen werden sollen. Da mehrere Blöcke nur schrittweise angesprungen werden können, muß hier eine 1 übergeben werden, also By = 1.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, erfolgt keine Markierung (Sel = False).

BEISPIEL:

```
Sub ExampleGoData
REM Es wird ein rechteckiger Bereich selektiert.
Const sText="Bereich"  REM Wort ...
Dim iCount As Integer
Documents.Add "sCalc4"
For iCount=3 to 7
ActiveDocument.Cells(2, iCount).Insert sText & "_B" & iCount
ActiveDocument.Cells(3, iCount).Insert sText &"_C" & iCount
ActiveDocument.Cells(4, iCount).Insert sText &"_D" & iCount
Next iCount
ActiveWindow.GotoCell("c5")
ActiveWindow.GoDownToEndOfData (1, FALSE)
ActiveWindow.GoRightToEndOfData (1, FALSE)
ActiveWindow.GoUpToStartOfData (1, TRUE)
ActiveWindow.GoLeftToStartOfData (1, TRUE)
End Sub
```

GoToCell-Methode

Selektiert einen angegebenen Zellbereich oder eine Zelle. Wird bei dieser Methode ein Zellbereich angegeben, wird dieser selektiert und der Cursor in die erste Zelle des Bereichs (oben links) gesetzt. Wird eine einzige Zelle angegeben, wird diese als Sprungziel genommen und selektiert. Der selektierte Bereich oder die selektierte Zelle wird als String zurückgeliefert.

KLASSE:
TableEditView

SYNTAX:
GoToCell(ToPoint As String)

RÜCKGABE:
String

PARAMETER:
ToPoint: Der Zellbereich, der selektiert werden soll, oder die Zelle, die angesprungen werden soll. Der Zellbereich kann als Bereich ("a1:a3") oder als Bereichs- oder Datenbankbereichsname angegeben werden. Wird ein Zellbe-

reich angegeben, wird die erste Zelle dieses Bereichs (links oben) zur aktiven Zelle.

Beispiel:

```
Sub ExampleGtotCell
Documents.Add "sCalc4"
Wait 1000
ActiveDocument.Range("B2").Insert "=A1+A2"
ActiveWindow.GotoCell "A1"
ActiveWindow.SearchSettings.Mode = 0 REM Formeln Suchen
ActiveWindow.Search "=A1+A2"
Print ActiveCell.Formula
End Sub
```

GoToCurrentCell-Methode

Bewegt die Bildschirmmaske direkt auf die aktive Zelle.

Klasse:
TableEditView

Syntax:
GoToCurrentCell()

Beispiel:

```
REM Die aktive Zelle muß bereits im nicht mehr sichtbaren
Bereich stehen.
ActiveWindow.GotoCurrentCell
```

GotoDocument-Methode

Aktiviert das Fenster eines bereits geladenen Dokuments.

Klasse:
TableEditView

Syntax:
GotoDocument(DocName As String)

Parameter:
DocName: Das Dokument, zu dem umgeschaltet werden soll.

Beispiel:

```
Sub ExampleGotoDocument
Dim sDoc As String
Documents.Add "sWriter4"
Wait 1000
sDoc = ActiveDocument.Name
Documents.Add "sCalc4"
Wait 1000
```

```
ActiveCell.Insert("Jetzt springen wir zu dem eben erstellten
StarWriter-Dokument")
ActiveWindow.GotoDocument(sDoc)
End Sub
```

GoToEndOfData-Methode

Bewegt die Einfügemarke in die letzte mit Inhalt gefüllte Zelle der aktuellen Tabelle.

KLASSE:
TableEditView

SYNTAX:
GoToEndOfData(Sel As Boolean)

PARAMETER:
[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, erfolgt keine Markierung (Sel = False).

BEISPIEL:

```
Sub ExampleGoToEndOfData
Documents.Add("scalc4")
ActiveCell.Insert("Wo ist die letzte beschriebene Zelle?")
Range("iv8000").Insert("Hier")
Wait 2000
ActiveWindow.GoToEndOfData(TRUE)
End Sub
```

GoToEndOfRow-Methode

Bewegt die Einfügemarke innerhalb der aktuellen Tabellenzeile auf die letzte mit Inhalt gefüllte Zelle.

KLASSE:
TableEditView

SYNTAX:
GoToEndOfRow(Sel As Boolean)

PARAMETER:
[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke hinweg bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht übergeben, erfolgt keine Markierung (Sel = False).

BEISPIEL:

```
Sub ExampleGotoEndOfRow
Documents.Add "sCalc4"
ActiveDocument.Range("F1").Insert "Hierher"
ActiveWindow.GotoEndOfRow(True)
Wait 2000
ActiveWindow.GotoStartOfRow
End Sub
```

GoToObject-Methode

Selektiert das angegebene OLE-Objekt im aktiven Fenster.

Um ein OLE-Objekt zu selektieren, steht auch die Select-Methode zur Verfügung, wobei der Zugriff über die Klasse TableOLEObject erfolgt.

Siehe auch: Select

KLASSE:
TableEditView

SYNTAX:
GoToObject(ToObject As String)

RÜCKGABE:
String

PARAMETER:
ToObject: Name des Objekts, das selektiert werden soll.

BEISPIEL:

```
Sub ExampleGotoObject
Dim sObjectName as string
Documents.Add "sCalc4"
Selection.InsertObjectStarChart(True, True, False)
sObjectName = ActiveDocument.OLEObjects(1).Name
ActiveWindow.GotoObject(sObjectName) REM oder
ActiveDocument.OLEObjects( sObjectName ).Select
End Sub
```

GoToStart-Methode

Bewegt die Einfügemarke auf die erste Zelle der Tabelle (A1).

KLASSE:
TableEditView

SYNTAX:
GoToStart(Sel As Boolean)

PARAMETER:
[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht übergeben, erfolgt keine Markierung des Sprungbereichs (Sel = False).

BEISPIEL:

```
Sub ExampleJumpToCell
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 500
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
ActiveWindow.JumpToTable(2)
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
Wait 10
ActiveWindow.JumpToTable(1)
Wait 10
For iCount = 1 to 100
ActiveWindow.JumpToPreviousCell
Next iCount
ActiveWindow.JumpToTable(2)
ActiveWindow.GotoStart
End Sub
```

GoToStartOfRow-Methode

Bewegt die Einfügemarke auf die erste Zelle der aktuellen Tabellenzeile.

KLASSE:
TableEditView

SYNTAX:
GoToStartOfRow()

BEISPIEL:

```
Sub ExampleGotoEndOfRow
Documents.Add "sCalc4"
ActiveDocument.Range("F1").Insert "Hierher"
ActiveWindow.GotoEndOfRow(True)
Wait 2000
ActiveWindow.GotoStartOfRow
End Sub
```

GoUp-Methode

Bewegt die Einfügemarke um die angegebene Anzahl an Zellen nach oben.

KLASSE:
TableEditView

SYNTAX:
GoUp(By As Integer, Sel As Boolean)

PARAMETER:
By: Integer-Ausdruck, der die Anzahl der Zellen angibt, um die die Einfügemarke nach oben bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

BEISPIEL:

```
Sub ExampleGoAhead
Documents.Add "sCalc4"
Wait 100
ActiveWindow.GoDown (2, True)
Wait 100
ActiveWindow.GoRight (2, True)
Wait 100
ActiveWindow.GoUp (2, True)
Wait 100
ActiveWindow.GoLeft (2, True)
End Sub
```

GoUpBlock-Methode

Bewegt die Einfügemarke um die angegebene Seitenanzahl nach oben.

KLASSE:
TableEditView

SYNTAX:
GoUpBlock(By As Integer, Sel As Boolean)

PARAMETER:
By: Integer-Ausdruck, der die Anzahl der Seiten angibt, um die die Einfügemarke nach oben bewegt werden soll.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser

Parameter ist optional. Wird er nicht gesetzt, werden die Zellen standardmäßig nicht markiert (Sel = False).

Beispiel:

```
Sub ExampleGoBlock
Documents.Add "sCalc4"
ActiveCell.Insert("Seite") REM Damit den markierten Bereich
füllen
ActiveCell.Copy
Wait 10
ActiveWindow.GoDownBlock (1, True)
Wait 10
ActiveWindow.GoRightBlock (1, True)
Wait 10
Selection.Paste
ActiveWindow.GoUpBlock (1, True)
Wait 10
ActiveWindow.GoLeftBlock (1, True)
End Sub
```

GoUpToStartOfData-Methode

Bewegt die Einfügemarke innerhalb der aktuellen Spalte nach oben zum Anfang des aktuellen Datenblocks. Befindet sich der Cursor bereits am Anfang des Datenblocks, wird zum Ende des vorigen Blocks gesprungen. Befindet sich vor der aktuellen Zelle keine weitere Zelle mit Inhalt, wird die Einfügemarke in die erste Tabellenzeile gesetzt.

Klasse:
TableEditView

Syntax:
GoUpToStartOfData(By As Integer, Sel As Boolean)

Parameter:
By: Gibt die Anzahl der Blöcke an, die mit dieser Methode übersprungen werden sollen. Da mehrere Blöcke nur schrittweise angesprungen werden können, muß hier eine 1 übergeben werden, also By = 1.

[Sel]: Boolescher Ausdruck, der angibt, ob die Zellen, über die sich die Einfügemarke bewegt, markiert werden sollen (True) oder nicht (False). Dieser Parameter ist optional. Wird er nicht gesetzt, erfolgt keine Markierung (Sel = False).

Beispiel:

```
Sub ExampleGoData
REM Es wird ein rechteckiger Bereich selektiert.
Const sText="Bereich" REM Wort ...
Dim iCount As Integer
```

```
Documents.Add "sCalc4"
For iCount=3 to 7
ActiveDocument.Cells(2, iCount).Insert sText & "_B" & iCount
ActiveDocument.Cells(3, iCount).Insert sText &"_C" & iCount
ActiveDocument.Cells(4, iCount).Insert sText &"_D" & iCount
Next iCount
ActiveWindow.GotoCell("c5")
ActiveWindow.GoDownToEndOfData (1, FALSE)
ActiveWindow.GoRightToEndOfData (1, FALSE)
ActiveWindow.GoUpToStartOfData (1, TRUE)
ActiveWindow.GoLeftToStartOfData (1, TRUE)
End Sub
Activewindow.GotoCell("c5")
Activewindow.GoDownToEndOfData 1, FALSE
Activewindow.GoRightToEndOfData 1, FALSE
Activewindow.GoUpToStartOfData 1, TRUE
Activewindow.GoLeftToStartOfData 1, TRUE
End Sub
```

Group-Methode

Gruppiert die Zeilen oder Spalten des Bereichs für eine Gliederung.

KLASSE:
RangeSelection

SYNTAX:
Group(RowOrCol As String)

PARAMETER:
RowOrCol: Gibt an, ob Zeilen oder Spalten gruppiert werden sollen:

R: Zeilen werden gruppiert.

C: Spalten werden gruppiert.

BEISPIEL:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
```

```
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

Eigenschaften und Methoden mit "H"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "H" beginnen.

Hide-Methode

Blendet die angegebene Tabelle aus.

KLASSE:
Tables

SYNTAX:
Hide(aTableName As String)

PARAMETER:
aTableName: Zeichenfolge, die die Bezeichnung der Tabelle aufnimmt, die ausgeblendet werden soll.

BEISPIEL:

```
Sub ExampleCalcTables
Const sTab1="Tabelle_Versteckt"
Const sTab2="Tabelle_Neu"
Const sTab3="Tabelle_Eingefuegt"
Dim iCount As Integer
Dim sTable As String
Documents.Add("scalc4")
REM Alle außer einer Tabelle verstecken
For iCount = 1 To ActiveDocument.Tables.Count -1
ActiveDocument.Tables.Hide(
ActiveDocument.Tables.Item(iCount).Name )
Next iCount
sTable = ActiveDocument.Tables(1).Name
With ActiveDocument.Tables
.Add( sTab1 )
```

```
.Add( sTab2 )
.Insert( sTab3 ,2)
.Hide( sTab1 )
.Show( sTable )
End With
End Sub
```

HideColumn-Methode

Blendet die aktuelle oder die angegebene Tabellenspalte aus.

KLASSE:
RangeSelection

SYNTAX:
HideColumn()

BEISPIEL:

```
Sub ExampleHideShow
Documents.Add("scalc4")
Wait 500
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("a3").HideRow
Range("a3").Insert("Diese Reihe war versteckt.")
Wait 1000
Range("a3").ShowRow
Range("b2").Insert("15")
Range("c2").Insert("200")
Range("d2").Insert("30")
Wait 1000
Range("c2:d2").HideColumn
Range("c2").Insert("War versteckt")
Wait 1000
Range("c2").ShowColumn
Range("e2").Insert("<-- Spalte D ist versteckt")
End Sub
```

HideDetail-Methode

Blendet die innerste Gliederungs-Gruppe (Zeilen und Spalten) aus, die den Zellbereich enthält.

KLASSE:
RangeSelection

SYNTAX:
HideDetail()

Beispiel:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

HideRow-Methode

Blendet die aktuelle oder die angegebene Tabellenzeile aus.

Klasse:
RangeSelection

Syntax:
HideRow()

Beispiel:

```
Sub ExampleHideShow
Documents.Add("scalc4")
Wait 500
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("a3").HideRow
Range("a3").Insert("Diese Reihe war versteckt.")
Wait 1000
```

```
Range("a3").ShowRow
Range("b2").Insert("15")
Range("c2").Insert("200")
Range("d2").Insert("30")
Wait 1000
Range("c2:d2").HideColumn
Range("c2").Insert("War versteckt")
Wait 1000
Range("c2").ShowColumn
Range("e2").Insert("<-- Spalte D ist versteckt")
End Sub
```

HorizontalAlignment-Eigenschaft

Bestimmt oder ermittelt die horizontale Ausrichtung des aktuellen Zellbereichs.

Bei leeren Zellen, die keinen Inhalt aufweisen, führt diese Eigenschaft bei einer Abfrage zu einem Laufzeitfehler, wenn die betroffenen Zellen mit der horizontalen Ausrichtungsoption "Standard" formatiert sind. Um dieses Problem zu umgehen und eine Weiterverarbeitung nicht wegen eines Laufzeitfehlers abbrechen zu lassen, kann man den Rückgabewert auf "nicht gesetzt" abfragen:

Align = Selection.HorizontalAlignment
If IsNull(Align) Then Align = 0
Damit wird der Variant auf einen gültigen Wert gleichen Typs gesetzt.

Siehe auch: VerticalAlignment, AlignLeft, AlignHCenter, AlignRight, AlignBlock.

KLASSE:
RangeSelection

SYNTAX:
HorizontalAlignment

RÜCKGABE:
Integer

PARAMETER:
[= AlignType]: Integer-Ausdruck, der die Art der Ausrichtung angibt. Folgende Werte sind möglich:

0: Standard (nur Schreiben möglich, wenn Zellen ohne Inhalt)

1: Linksbündig

2: Zentriert

3: Rechtsbündig

4: Blocksatz

Beispiel:

```
Sub ExampleCellAlignment
Const stext="Text"  REM Ein bißchen Text, der die Ausrichtung
zeigt
Documents.Add("scalc4")
Wait 100
ActiveCell.Insert( sText )
ActiveCell.Copy
Range("a1:d5").Paste
Range("a1:iv256").ColumnWidth = 300
Range("a1:iv256").RowHeight = 300
Range("a1:d5").ColumnWidth = 1000
Range("a1:d5").RowHeight = 1000
REM eine kleine Tabelle mit allen Ausrichtungen
For iX=1 To 4
For iY=1 To 5
ActiveTable.Cells(iX,iY).VerticalAlignment = iX -1
ActiveTable.Cells(iX,iY).HorizontalAlignment = iY -1
Next iY
Next iX
End Sub
```

Eigenschaften und Methoden mit "I"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "I" beginnen.

ImageMapDialog-Eigenschaft

Liefert oder setzt den Status des ImageMap-Editors.

Klasse:
RangeSelection

Syntax:
ImageMapDialog

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob der ImageMap-Editor ein- (True) oder ausgeblendet (False) ist.

Beispiel:

```
Sub ExampleImageMap
Dim sGrfPath As String
Documents.Add("scalc4")
Wait 1000
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Range("d5").InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Range("d5").ImageMapDialog = TRUE
End Sub
```

IncrementIndent-Methode

Setzt die horizontale Ausrichtung auf linksbündig und vergrößert den Einzug um 10pt.

Klasse:
RangeSelection

Syntax:
IncrementIndent()

Beispiel:

```
Sub ExampleIncrementIndent
Documents.Add("scalc4")
ActiveCell.Insert("X")
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.IncrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
wait 500
ActiveCell.DecrementIndent
End Sub
```

Index-Eigenschaft

Liefert den Index der aktuellen Tabelle aus der Tables-Collection. Dieser Index wird beispielsweise mittels "ActiveTable.Index" ermittelt.

KLASSE:
TableSelection

SYNTAX:
Index

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleMove
Const sTabName="Tabelle_die_verschoben_wird"
Dim sName1 As String, sName2 As String
Dim iTab As Integer
REM ein Dokument, aus der eine Tabelle verschoben werden soll
Documents.Add "sCalc4"
ActiveTable.Name = sTabName
sName1 = ActiveDocument.Name
REM ein Dokument, in das eine Tabelle verschoben werden soll
Documents.Add "sCalc4"
iTab = ActiveTable.Index + 1
sName2 = ActiveDocument.Name
Documents(sName1).Tables( sTabName ).Move(sName2 , iTab, False)
End Sub
```

InSelection-Eigenschaft

Bestimmt oder ermittelt, ob die Suche nur innerhalb eines markierten Bereichs erfolgen soll. Diese Funktion entspricht der Suchoption NUR IN SELEKTION im Dialog SUCHEN & ERSETZEN.

KLASSE:
TableEditView

SYNTAX:
InSelection

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche nur innerhalb der Selektion erfolgen soll (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

Insert-Methode

Fügt Daten in die selektierte oder angegebene Zelle ein. Diese Methode gilt nur für eine einzelne Zelle, keinen Zellbereich. Die selektierte Zelle läßt sich mit ActiveCell ansprechen, eine bestimmte Zelle, z.B. A5, wird mit Range("A5") angegeben.

Siehe auch: PutCell

KLASSE:
RangeSelection

SYNTAX:
Insert(StringName As String)

PARAMETER:
StringName: Zeichenfolge, die die Daten aufnimmt, die in die Zelle eingefügt werden sollen.

Beispiel:

```
Sub ExampleContents
Documents.Add("scalc4")
Wait 2000
Range("b1").Insert("Zellen mit Formeln")
Range("b2").Insert( 123.4567)
Range("c2").Insert( "=b2 * " & cdbl(123.456))
Range("d2").Insert( "=c2*b2")
Wait 1000
Range("b2:d2").Copy
Tables(2).Activate
Wait 1000
Range("b2:d2").InsertContents("SV")
Range("b1").Insert("Nur Werte und Texte kopieren, damit man
sortieren kann.")
End Sub
```

Insert-Methode

Fügt eine neue Tabelle an der angegebenen Position in das Dokument ein.

Klasse:
Tables

Syntax:
Insert(Name As String, Index As Long)

Parameter:
Name: Gibt den Namen der neuen Tabelle an.

Index: Bestimmt die Position der neuen Tabelle innerhalb der Tables-Collection, an der die Tabelle eingefügt werden soll (ab 1 gezählt).

Beispiel:

```
Sub ExampleCalcTables
Const sTab1="Tabelle_Versteckt"
Const sTab2="Tabelle_Neu"
Const sTab3="Tabelle_Eingefuegt"
Dim iCount As Integer
Dim sTable As String
Documents.Add("scalc4")
REM alle ausser einer Tabelle verstecken
For iCount = 1 To ActiveDocument.Tables.Count -1
   ActiveDocument.Tables.Hide(
ActiveDocument.Tables.Item(iCount).Name )
Next iCount
sTable = ActiveDocument.Tables(1).Name
With ActiveDocument.Tables
   .Add( sTab1 )
   .Add( sTab2 )
   .Insert( sTab3 ,2)
```

```
   .Hide( sTab1 )
   .Show( sTable )
End With
End Sub
```

InsertApplet-Methode

Ruft den Dialog Applet einfügen auf. Ein Applet wird in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

Klasse:
RangeSelection

Syntax:
InsertApplet()

Beispiel:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertApplet
End Sub
```

InsertCell-Methode

Fügt an der aktuellen Position der Einfügemarke eine Zelle ein.

Klasse:
RangeSelection

Syntax:
InsertCell(Flags As String)

Parameter:
Flags: Zeichenfolge, die angibt, wie die umliegenden Zellen verschoben werden sollen. Folgende Werte sind möglich:

V: Die Zellen werden nach unten verschoben.

>: Die Zellen werden nach rechts verschoben.

R: Eine vollständige Zeile wird eingefügt.

C: Eine vollständige Spalte wird eingefügt.

Beispiel:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
```

```
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

InsertCellsDown-Methode

Fügt neue Zellen in der Größe des Range ein und verschiebt die bisherigen Zellen nach unten.

KLASSE:
RangeSelection

SYNTAX:
InsertCellsDown()

BEISPIEL:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
```

```
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

InsertCellsRight-Methode

Fügt neue Zellen in der Größe des Range ein und verschiebt die bisherigen Zellen nach rechts.

KLASSE:
RangeSelection

SYNTAX:
InsertCellsRight()

BEISPIEL:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

InsertColumnBreak-Methode

Fügt einen Spaltenumbruch vor der aktiven Zelle ein. Bei einem selektierten oder angegebenen Bereich wird der Spaltenumbruch vor der rechten unteren Zelle des Bereichs eingefügt.

KLASSE:
RangeSelection

SYNTAX:
InsertColumnBreak()

Beispiel:

```
Sub ExampleBreaks
Documents.Add("sCalc4")
wait 1000
Range("b2").Select
ActiveCell.InsertRowBreak
ActiveCell.InsertColumnBreak
wait 2000
ActiveCell.DeleteRowBreak
ActiveCell.DeleteColumnBreak
End Sub
```

InsertColumns-Methode

Fügt in dem selektierten oder dem angegebenen Bereich Spalten in die Tabelle ein. Die Zahl der einzufügenden Spalten richtet sich nach der Selektion oder dem Range-Objekt.

Siehe auch: InsertRows

Klasse:
RangeSelection

Syntax:
InsertColumns()

Beispiel:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

InsertContents-Methode

Fügt Inhalte aus dem Clipboard ein. (Der Clipboard-Inhalt muß von Calc stammen.)

KLASSE:
RangeSelection

SYNTAX:
InsertContents(Flags As String)

PARAMETER:
Flags: String, der angibt, was eingefügt werden soll:

A: alles

S: Strings

V: Werte

D: Datum/Zeit

F: Formeln

N: Notizen

T: Attribute

Die verschiedenen Flags können auch kombiniert werden (außer "A").

BEISPIEL:

```
Sub ExampleContents
Documents.Add("scalc4")
wait 2000
Range("b1").Insert("Zellen mit Formeln")
Range("b2").Insert( 123.4567)
Range("c2").Insert( "=b2 * " & cdbl(123.456))
Range("d2").Insert( "=c2*b2")
Wait 1000
Range("b2:d2").Copy
Tables(2).Activate
Wait 1000
Range("b2:d2").InsertContents("SV")
Range("b1").Insert("Nur Werte und Texte kopieren, damit man
sortieren kann.")
end sub
```

InsertGraphic-Methode

Importiert eine Grafik und fügt diese an der Position der Schreibmarke in das Dokument ein.

KLASSE:
RangeSelection

SYNTAX:
InsertGraphic(FileName As String, FilterName As String, Link As Boolean)

RÜCKGABE:
Boolean

PARAMETER:
FileName: Ein Zeichenfolgenausdruck, der den Dateinamen der zu importierenden Grafik angibt.

FilterName: Ein Zeichenfolgenausdruck, der den zum Importieren der Grafik notwendigen Filtertyp angibt. Die mit FilterName angegebene Bezeichnung des Filters muß exakt dem Eintrag des Listenfelds DATEITYP im Dialog GRAFIK EINFÜGEN entsprechen, also z. B. "TIF - Tag Image File" für Grafiken im TIF-Format.

Die Angabe des Grafikfilters ist optional. Wird ein Leerstring übergeben, so erfolgt eine automatische Erkennung des benötigten Grafikfilters.

Link: Ein Boolescher Ausdruck, der angibt, ob die einzufügende Grafik mit dem Dokument verknüpft (True) oder im Dokument gespeichert (False) werden soll.

BEISPIEL:

```
Sub ExampleInsertGraphic
Dim sGrfPath As String
Documents.Add("scalc4")
Wait 1000
sGrfPath = Application.PathSettings.Config & "\END.BMP"
Range("d5").InsertGraphic(sGrfPath, "BMP - MS Windows", TRUE )
Range("a1").EditLinks
End Sub
```

InsertMatrix-Methode

Gibt eine Matrix-Formel in den Bereich ein.

KLASSE:
RangeSelection

SYNTAX:
InsertMatrix(Formula As String)

PARAMETER:
Formula: die Formel (ohne die geschweiften Klammern)

BEISPIEL:

```
Sub ExampleMatrix
Wait 500
Documents.Add("scalc4")
Range("a1").Insert("2")
Range("a2").Insert("5")
Range("b1").Insert("3")
Range("b2").Insert("8")
Wait 3000
REM jetzt wird ein MatrixFormel eingefügt
Range("c3:d4").InsertMatrix("=MINV(A1:B2)")
End Sub
```

InsertName-Methode

Ruft den Dialog NAMEN EINFÜGEN auf. Mit diesem Dialog wird durch Auswahl eines zuvor definierten Bereichsnamens ein Zellbereich an der Cursorposition eingefügt. Da der benannte Bereich in der Zelle eingefügt wird, die bei Aufruf des Dialogs aktiv war, läßt sich das entsprechende RangeSelection-Objekt mit ActiveCell oder Selection ansprechen.

KLASSE:
RangeSelection

SYNTAX:
InsertName()

BEISPIEL:

```
Sub ExampleInsertName
   Documents.Add("scalc4")
   ActiveCell.InsertName
End Sub
```

InsertObject-Methode

Ruft den Dialog OLE-Objekt einfügen auf. Ein OLE-Objekt wird in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

Klasse:
RangeSelection

Syntax:
InsertObject()

Beispiel:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertObject
End Sub
```

InsertObjectFloatingFrame-Methode

Ruft den Dialog Frame-Eigenschaften zum Einfügen eines Frames auf. Ein Frame wird jeweils in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

Klasse:
RangeSelection

Syntax:
InsertObjectFloatingFrame()

Beispiel:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertObjectFloatingFrame
End Sub
```

InsertObjectStarChart-Methode

Fügt ein StarChart-Diagramm in die aktuelle Tabelle ein.

Klasse:
RangeSelection

Syntax:
InsertObjectStarChart(ColHeaders As Boolean, RowHeaders As Boolean, InNewTable As Boolean, RangeList As String)

PARAMETER:
ColHeaders: Boolescher Ausdruck, der angibt, ob die erste Zeile des angegebenen Bereiches als Beschriftung verwendet werden soll (True) oder nicht (False).

RowHeaders: Boolescher Ausdruck, der angibt, ob die erste Spalte des angegebenen Bereiches als Beschriftung verwendet werden soll (True) oder nicht (False).

InNewTable: Boolescher Ausdruck, der angibt, ob das Diagramm in eine neue Tabelle eingefügt werden soll (True) oder nicht (False).

RangeList: Zeichenfolge, die den Zellbereich aufnimmt, der die Daten enthält, aus denen das Diagramm erstellt werden soll. Sollen mehrere Zellbereiche in das Diagramm einbezogen werden, so werden diese im Paramter RangeList durch Semikola getrennt aufgelistet.

BEISPIEL:

```
Sub ExampleStarChart
Documents.Add("scalc4")
Wait 2000
REM Tabelle mit Daten füllen
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Frauen")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Männer")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("e5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
wait 2000
REM Daten verändern
Range("a4").Insert("Jungens")
Range("b4").Insert("1300")
Range("c4").Insert("150")
Range("d4").Insert( 0.5 )
wait 3000
Range("e5").UpdateChart(TRUE)
REM neue Daten hinzufügen
Range("b7").Insert( "Mädchen" )
Range("b8").Insert( "Babies" )
Range("c7").Insert( 23 )
Range("e8").Insert( 120 )
ActiveDocument.AddChartData( OLEObjects(1).Name , "b7:e8")
wait 3000
REm anderen Chartbereich festlegen
Range("c6").Insert("Dubai")
```

```
Range("d6").Insert("Paris")
Range("e6").Insert("Hong Kong")
ActiveDocument.ChangeChartData( OLEObjects(1).Name , "b6:e8" ,
TRUE, TRUE)
End Sub
```

InsertObjectStarImage-Methode

Fügt an der aktuellen Zelle ein (leeres) StarImage-Objekt ein. Das Bild wird in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

KLASSE:
RangeSelection

SYNTAX:
InsertObjectStarImage()

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertObjectStarImage
End Sub
```

InsertObjectStarMath-Methode

Fügt ein (leeres) Math-Objekt in die aktuelle Zelle ein. Die Formel wird in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

KLASSE:
RangeSelection

SYNTAX:
InsertObjectStarMath()

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertObjectStarMath
End Sub
```

InsertPlugin-Methode

Ruft den Dialog PLUGIN EINFÜGEN auf. Das PlugIn wird in der Zelle eingefügt, die bei Aufruf des Dialogs aktiv war. Das entsprechende RangeSelection-Objekt läßt sich daher mit ActiveCell oder Selection ansprechen.

KLASSE:
RangeSelection

SYNTAX:
InsertPlugin()

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.InsertPlugin
End Sub
```

InsertRowBreak-Methode

Fügt einen Zeilenumbruch vor der aktiven Zelle ein. Bei einem selektierten oder angegebenen Bereich wird der Zeilenumbruch vor der rechten unteren Zelle des Bereichs eingefügt.

KLASSE:
RangeSelection

SYNTAX:
InsertRowBreak()

BEISPIEL:

```
Sub ExampleBreaks
Documents.Add("sCalc4")
wait 1000
Range("b2").Select
ActiveCell.InsertRowBreak
ActiveCell.InsertColumnBreak
wait 2000
ActiveCell.DeleteRowBreak
ActiveCell.DeleteColumnBreak
End Sub
```

InsertRows-Methode

Fügt in dem selektierten oder dem angegebenen Bereich Zeilen in die Tabelle ein. Die Zahl der einzufügenden Zeilen richtet sich nach der Selektion oder dem Range-Objekt.

Siehe auch: InsertColumns

KLASSE:
RangeSelection

SYNTAX:
InsertRows()

BEISPIEL:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

Interior-Eigenschaft

Gibt ein Interior-Objekt zurück, mit dem der Zellhintergrund eingestellt werden kann.

KLASSE:
RangeSelection

SYNTAX:
Interior

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleInterior
Documents.add "sCalc4"
Wait 100
With Range( "b2:e5" ).Interior REM setzt eingegebenen Bereich
   .Transparent = TRUE REM auf einen blaßroten Hintergrund
   .Style = 10 REM durch ein Muster das rot und weiß enthält
   .FillColor = RGB( 255,0,0)
   .BackColor = RGB( 255,255,255)
End With
End Sub
```

Italic-Eigenschaft

Die Eigenschaft bestimmt oder ermittelt den Grad der kursiven Darstellung der Schrift des selektierten Bereiches.

KLASSE:
TableFont

SYNTAX:
Italic

RÜCKGABE:
Integer

PARAMETER:
[= ItalicExtent]: Integer-Ausdruck, der den Grad der kursiven Darstellung der Schrift des selektierten Bereiches angibt. Folgende Werte sind möglich:

0: Der Text wird nicht kursiv dargestellt.

1 - 2: Der Text wird kursiv dargestellt. Der Grad der Kursiv-Darstellung erhöht sich mit zunehmendem Wert. Der Wert 2 gibt die Standardeinstellung des Attributs Kursiv wieder.

Nicht alle Fonts unterstützen verschiedene Stufen des Italic-Attributs. Der Standardwert 2 ist jedoch fast immer verfügbar.

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
```

```
With ActiveCell.Font
sInfo = .CharSet
.Color=RGB(255,0,0)
sInfo = .Family
sInfo = .FamilyName
.Italic = 7
.Pitch=2
.Size = 24
sInfo = .StyleName
.Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit der Vorlage REM " &
sTemp1 & "REM  geschrieben")
End Sub
```

Eigenschaften und Methoden mit "J"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "J" beginnen.

JumpToNextCell-Methode

Setzt den Cursor in der aktuelle Spalte eine Zeile tiefer.

Siehe auch: JumpToPreviousCell

KLASSE:
TableEditView

SYNTAX:
JumpToNextCell()

BEISPIEL:

```
Sub ExampleJumpToCell
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 500
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
ActiveWindow.JumpToTable(2)
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
Wait 10
ActiveWindow.JumpToTable(1)
Wait 10
For iCount = 1 to 100
ActiveWindow.JumpToPreviousCell
```

```
Next iCount
ActiveWindow.JumpToTable(2)
ActiveWindow.GotoStart
End Sub
```

JumpToNextTable-Methode

Springt zur nächsten Tabelle im aktuellen Dokument.

KLASSE:
TableEditView

SYNTAX:
JumpToNextTable()

BEISPIEL:

```
Sub ExampleJumpToTable
Dim sDoc As String,iCount As Integer
Documents.Add "sCalc4"
Wait 1000
iCount = ActiveDocument.TableCount
Wait 1000
ActiveCell.Insert("Neu erstellte Kalkulationen besitzen immer "
& iCount & " Blätter."
For iCount = 2 To ActiveDocument.TableCount
Wait 1000
ActiveWindow.JumpToNextTable
ActiveCell.Insert("Alle Blätter einmal durchblättern: " &
iCount
Next
For iCount = 1 To ActiveDocument.TableCount-1
Wait 1000
ActiveWindow.JumpToPrevTable
ActiveCell.Insert("Und wieder zurück: " & ActiveDocument.Table-
Count-iCount
Next
End Sub
```

JumpToNextUnprotected-Methode

Springt zur nächsten ungeschützten Zelle (zuerst horizontal), wenn die Tabelle geschützt ist, oder bewegt den Cursor nach rechts zur nächsten Zelle, wenn die Tabelle nicht geschützt ist.

Siehe auch: JumpToPreviousUnprotected

KLASSE:
TableEditView

SYNTAX:
JumpToNextUnprotected()

Beispiel:

```
REM Es müssen bereits ungeschützte Zellen definiert worden
sein.
ActiveWindow.JumpToNextUnprotected
ActiveWindow.JumpToPreviousUnprotected
```

JumpToPreviousCell-Methode

Setzt den Cursor in der aktuelle Spalte eine Zeile höher.

Siehe auch: JumpToNextCell

Klasse:
TableEditView

Syntax:
JumpToPreviousCell()

Beispiel:

```
Sub ExampleJumpToCell
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 500
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
ActiveWindow.JumpToTable(2)
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
Wait 10
ActiveWindow.JumpToTable(1)
Wait 10
For iCount = 1 to 100
ActiveWindow.JumpToPreviousCell
Next iCount
ActiveWindow.JumpToTable(2)
ActiveWindow.GotoStart
End Sub
```

JumpToPreviousUnprotected-Methode

Springt zur vorherigen ungeschützten Zelle (zuerst horizontal), wenn die Tabelle geschützt ist, oder bewegt den Cursor nach links zur vorigen Zelle, wenn die Tabelle nicht geschützt ist.

Siehe auch: JumpToNextUnprotected

Klasse:
TableEditView

SYNTAX:
JumpToPreviousUnprotected()

BEISPIEL:

```
REM Es müssen bereits ungeschützte Zellen definiert worden
sein.
ActiveWindow.JumpToNextUnprotected
ActiveWindow.JumpToPreviousUnprotected
```

JumpToPrevTable-Methode

Springt zur vorangehenden Tabelle im aktuellen Dokument.

KLASSE:
TableEditView

SYNTAX:
JumpToPrevTable()

BEISPIEL:

```
Sub ExampleJumpToTable
Dim sDoc As String,iCount As Integer
Documents.Add "sCalc4"
Wait 1000
iCount = ActiveDocument.TableCount
Wait 1000
ActiveCell.Insert("Neu erstellte Kalkulationen besitzen immer "
& iCount & " Blätter."
For iCount = 2 To ActiveDocument.TableCount
Wait 1000
ActiveWindow.JumpToNextTable
ActiveCell.Insert("Alle Blätter einmal durchblättern: " &
iCount
Next
For iCount = 1 To ActiveDocument.TableCount-1
Wait 1000
ActiveWindow.JumpToPrevTable
ActiveCell.Insert("Und wieder zurück: " & ActiveDocument.Table-
Count-iCount
Next
End Sub
```

JumpToTable-Methode

Die Methode selektiert die angegebene Tabelle.

KLASSE:
TableEditView

SYNTAX:
JumpToTable(Nr As Long)

PARAMETER:
Nr: Index der Tabelle (ab 1 gezählt).

BEISPIEL:

```
Sub ExampleJumpToCell
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 500
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
ActiveWindow.JumpToTable(2)
For iCount = 1 to 100
   ActiveWindow.JumpToNextCell
Next iCount
Wait 10
ActiveWindow.JumpToTable(1)
Wait 10
For iCount = 1 to 100
ActiveWindow.JumpToPreviousCell
Next iCount
ActiveWindow.JumpToTable(2)
ActiveWindow.GotoStart
End Sub
```

Eigenschaften und Methoden mit "L"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "L" beginnen.

Longer-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche zu einem Wort hinzugefügt werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Longer

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, um wieviele Zeichen das Wort im Dokument länger sein darf als der Suchbegriff, um noch als gleich zu gelten.

BEISPIEL:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Eigenschaften und Methoden mit "M"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "M" beginnen.

MergeCells-Methode

Verbindet die selektierten Zellen oder den angegebenen Zellbereich zu einer einzigen Zelle. Dabei kann entschieden werden, ob eventuelle Inhalte der Zellen, die dabei "aufgelöst" werden, in die erste Zelle übernommen werden.

KLASSE:
RangeSelection

SYNTAX:
MergeCells(MoveContents As Boolean)

PARAMETER:
MoveContents: Boolescher Ausdruck, der angibt, ob der Inhalt der "verdeck-

ten" Zellen in die erste Zelle des betroffenen Bereichs verschoben werden soll (True) oder nicht (False).

Beispiel:

```
Sub ExampleSplitCell
On error Goto SplitCellError
Dim sText as String
Documents.Add("scalc4")
Wait 500
Range("a2").Insert(100)
Range("b3").Insert(" Text")
Range("a2:b3").MergeCells False
Wait 1000
ActiveDocument.Cells(1,5).Insert("Zellen verbinden ohne
Textübernahme")
sText = Range("a2").TextValue
ActiveDocument.Cells(1,6).Insert("a2="& sText )
sText = Range("b3").TextValue
ActiveDocument.Cells(1,7).Insert("b3=" & sText )
Wait 2000
Range("b3").SplitCell
Wait 2000
Range("a2:b3").MergeCells TRUE
ActiveDocument.Cells(1,9).Insert("Zellen verbinden mit
Textübernahme")
sText = Range("a2").TextValue
ActiveDocument.Cells(1,10).Insert("a2="& sText )
sText = Range("b3").TextValue
ActiveDocument.Cells(1,11).Insert("b3=" & sText )
Wait 2000
Range("a2:b3").SplitCell
Exit Sub
SplitCellError:
Range("a2").SplitCell
Range("e2").Insert("Fehler beim Versuch eine Zelle zu teilen.")
Resume Next
End Sub
```

MetricUnit-Eigenschaft

Bestimmt oder ermittelt die Maßeinheit für das aktive Modul.

Klasse:
StarCalc

Syntax:
MetricUnit

Rückgabe:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der die Maßeinheit angibt. Folgende Werte sind möglich:

Wert	Einheit
1	mm
2	cm
5	Twip (Windows-System Maßeinheit)
6	Punkt
7	Pica
8	Inch (Zoll)

BEISPIEL:

```
Sub ExampleMetricUnit
Const sMessage="Die Zellenbreite ist "
Dim iUnit As Integer
Documents.Add "sCalc4"
iUnit = StarCalc.MetricUnit
StarCalc.MetricUnit = 5
Range("A2").Insert( sMessage & ActiveCell.ColumnWidth &" Twips"
)
StarCalc.MetricUnit = iUnit
End Sub
```

Mode-Eigenschaft

Die Eigenschaft bestimmt oder ermittelt, ob nur in Formeln, Werten oder Notizen gesucht wird.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Mode

RÜCKGABE:
Long

PARAMETER:
[= Search]: Gibt die zusätzliche Suchoption an. Folgende Werte sind möglich:

0 - in Formeln

1 - in Werten

2 - in Notizen

Beispiel:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Move-Methode

Verschiebt oder kopiert die aktuelle Tabelle in das angegebene Dokument.

Klasse:
TableSelection

Syntax:
Move(DocName As String, Index As Long, Copy As Boolean)

Parameter:
DocName: Zeichenfolgenausdruck, der die Bezeichnung des Dokumentes angibt, in das die Tabelle kopiert oder verschoben werden soll.

Index: Long-Integer-Ausdruck, der die Nummer der Tabelle bestimmt, vor der die aktuelle Tabelle eingefügt werden soll. Der Wert 0 gibt an, daß die Tabelle an das Ende des Dokumentes kopiert oder verschoben werden soll.

Copy: Boolescher Ausdruck, der angibt, ob die Tabelle kopiert (True) oder verschoben (False) werden soll.

Beispiel:

```
Sub ExampleMove
Const sTabName="Tabelle_die_verschoben_wird"
Dim sName1 As String, sName2 as string
Dim iTab as integer
REm ein Dokument, aus der eine Tabelle verschoben werden soll
Documents.add "sCalc4"
ActiveTable.Name = sTabName
sName1 = Activedocument.Name
REM Dokument in das eine Tabelle hinein verschoben werden soll
Documents.add "sCalc4"
iTab = ActiveTable.Index + 1
```

```
sName2 = ActiveDocument.Name
Documents(sName1).Tables( sTabName ).Move(sName2 , iTab, False)
End Sub
```

Eigenschaften und Methoden mit "N"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "N" beginnen.

Name-Eigenschaft

Bestimmt oder ermittelt die Bezeichnung der aktuellen Tabelle.

KLASSE:
TableSelection

SYNTAX:
Name

RÜCKGABE:
String

PARAMETER:
[= TabName]: Zeichenfolgenausdruck, der die Bezeichnung der aktuellen Tabelle aufnimmt.

BEISPIEL:

```
Sub ExampleMove
Const sTabName="Tabelle_die_verschoben_wird"
Dim sName1 As String, sName2 As String
Dim iTab As Integer
REM ein Dokument, aus dem eine Tabelle verschoben werden soll
Documents.Add "sCalc4"
ActiveTable.Name = sTabName
sName1 = ActiveDocument.Name
REM Dokument, in das eine Tabelle verschoben werden soll
Documents.Add "sCalc4"
iTab = ActiveTable.Index + 1
sName2 = ActiveDocument.Name
Documents(sName1).Tables( sTabName ).Move(sName2 , iTab, False)
End Sub
```

NewTemplate-Methode

Legt eine neue Zell- oder Seitenvorlage im Dokument an und öffnet den Dialog zum Bearbeiten der Vorlage.

KLASSE:
RangeSelection

SYNTAX:
NewTemplate(Param As String, Family As Long)

PARAMETER:
[Param]: Optionaler Parameter, die den Namen der neuen Vorlage bestimmen soll.

Family: Gibt die Art der neuen Vorlage an. Folgende Werte sind möglich:

2 - Zellvorlage

8 - Seitenvorlage

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.NewTemplate("NeueSeitenVorlage",8)
End Sub
```

NoteText-Eigenschaft

Gibt den Text einer Notiz an einer Zelle zurück oder setzt ihn. Bei einem Bereich mit mehreren Zellen wird die erste Zelle links oben genommen.

KLASSE:
RangeSelection

SYNTAX:
NoteText

RÜCKGABE:
String

PARAMETER:
[= Note]: Zeichenfolge, die den Text der Notiz angibt.

BEISPIEL:

```
Sub ExampleNotes
Dim sText as String
Documents.Add("sCalc4")
Range("a2").Insert("1234,5678")
Range("a2").NoteText="Diese Notiz erscheint, wenn sie mit der
```

```
Maus über die Zelle fahren."
sText = Range("a2").NoteText
Range("a3").Insert( sText )
Range("a2").NoteVisible = TRUE
Wait 5000
Range("a2").NoteVisible = FALSE
End Sub
```

NoteVisible-Eigenschaft

Blendet eine Notiz an einer Zelle ein oder aus. An der Zelle muß vorher eine Notiz gesetzt worden sein.

Wird diese Eigenschaft an einem Range-Objekt aufgerufen, der einen Zellbereich festgelegt, so wird die letzte Zelle des Bereichs unten rechts genommen.

Siehe auch: NoteText

KLASSE:
RangeSelection

SYNTAX:
NoteVisible

RÜCKGABE:
Boolean

BEISPIEL:

```
Sub ExampleNotes
Dim sText as String
Documents.Add("sCalc4")
Range("a2").Insert("1234,5678")
Range("a2").NoteText="Diese Notiz erscheint, wenn sie mit der
Maus über die Zelle fahren."
sText = Range("a2").NoteText
Range("a3").Insert( sText )
Range("a2").NoteVisible = TRUE
Wait 5000
Range("a2").NoteVisible = FALSE
End Sub
```

NumberFormat-Eigenschaft

Bestimmt oder ermittelt das Zahlenformat für eine Zelle oder einen Zellbereich. Das Zahlenformat ist im Dialog Zellattribute unter Format - Zelle... im Register Zahlen unter Format-Code zu sehen. Wenn in einem Zellbereich unterschiedliche Formate gesetzt sind, kommt bei der Abfrage ein leerer String zurück.

Klasse:
RangeSelection

Syntax:
NumberFormat

Rückgabe:
String

Beispiel:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatCurrency-Methode

Weist dem Zellinhalt des selektierten oder angegebenen Bereiches das aktuelle Währungsformat zu.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatCurrency()

BEISPIEL:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatDate-Methode

Setzt das Zahlenformat für eine Zelle oder einen Zellbereich auf das Standard-Datums-Format.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatDate()

Beispiel:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatDecDecimals-Methode

Setzt die Anzahl der Nachkommastellen im Zahlenformat der Zelle oder des Zellbereichs um 1 zurück.

Siehe auch: NumberFormatIncDecimals

Klasse:
RangeSelection

Syntax:
NumberFormatDecDecimals()

Beispiel:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
```

```
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatDecimal-Methode

Setzt das Zahlenformat für eine Zelle oder einen Zellbereich auf zwei Nachkommastellen und Tausendertrenner.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatDecimal()

BEISPIEL:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
```

```
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatIncDecimals-Methode

Erhöht die Anzahl der Nachkommastellen im Zahlenformat der Zelle oder des Zellbereichs um 1.

Siehe auch: NumberFormatDecDecimals

KLASSE:
RangeSelection

SYNTAX:
NumberFormatIncDecimals()

BEISPIEL:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
```

```
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatPercent-Methode

Weist dem Zellinhalt des selektierten oder angegebenen Bereiches das aktuelle Prozentformat zu.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatPercent()

BEISPIEL:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatScientific-Methode

Setzt das Zahlenformat für eine Zelle oder einen Zellbereich auf das Standard-Exponential-Format.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatScientific()

BEISPIEL:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatStandard-Methode

Setzt das Zahlenformat für eine Zelle oder einen Zellbereich auf das Standard-Format.

KLASSE:
RangeSelection

SYNTAX:
NumberFormatStandard()

Beispiel:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

NumberFormatTime-Methode

Setzt das Zahlenformat für eine Zelle oder einen Zellbereich auf das Standard-Uhrzeit-Format.

Klasse:
RangeSelection

Syntax:
NumberFormatTime()

Beispiel:

```
Sub ExampleNumberFormats
Documents.Add("scalc4")
Range("b2").Insert("1234,5678")
Range("b2").Copy
Range("b2:b10").Paste
Range("a2").Insert("Benutzerdefiniert")
Range("b2").NumberFormat="On;Off;nothing"
Range("a3").Insert("Währung")
Range("b3").NumberFormatCurrency
```

```
Range("a4").Insert("Datum")
Range("b4").NumberFormatDate
Range("a5").Insert("Dezimal")
Range("b5").NumberFormatDecimal
Range("b5").NumberFormatDecDecimals
Range("b5").NumberFormatDecDecimals
Range("a6").Insert("Dezimal")
Range("b6").NumberFormatDecimal
Range("b6").NumberFormatIncDecimals
Range("a7").Insert("Prozent")
Range("b7").NumberFormatPercent
Range("a8").Insert("Wissenschaftlich")
Range("b8").NumberFormatScientific
Range("a9").Insert("Standard")
Range("b9").NumberFormatStandard
Range("a10").Insert("Zeit")
Range("b10").NumberFormatTime
Range("a1:b10").SetOptimalColumnWidth( 100 )
End Sub
```

Eigenschaften und Methoden mit "O"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "O" beginnen.

Offset-Methode

Verschiebt einen Tabellenbereich um die angegebene Zeilen- und Spaltenzahl. Die Verschiebung kann sowohl innerhalb einer Tabelle als auch zwischen verschiedenen Tabellen erfolgen. Damit ist es möglich, einen Bereich (Range) über die Angabe relativer Parameter zu verändern. Das neue Range-Objekt wird zurückgeliefert, wodurch der Zugriff auf diesen neuen Bereich mit anderen Eigenschaften und Methoden ermöglicht wird.

> Verschoben wird nicht der Inhalt der Zellen, sondern der Range, auf den Bezug genommen wird! Im Beispiel wird einmal der Range B2:E5 gefärbt, dann der Range D7:B10, aber die Zellen selbst werden nicht verschoben.

KLASSE:
RangeSelection

SYNTAX:
Offset(Column As Integer, Row As Integer, Table As Integer)

RÜCKGABE:
Object

PARAMETER:
[Column]: Integer-Ausdruck, der die Anzahl der Spalten angibt, um die die Adressierung des Tabellenbereichs verschoben werden soll.

[Row]: Integer-Ausdruck, der die Anzahl der Zeilen angibt, um die die Adressierung des Tabellenbereichs verschoben werden soll.

[Table]: Integer-Ausdruck, der die Anzahl der Tabellen angibt, um die die Adressierung des Tabellenbereichs verschoben werden soll.

Die Angabe der Offset-Parameter Column, Row und Table ist optional. Werden keine Parameter übergeben, wird jeweils der Standardwert 0 genommen (kein Offset).

BEISPIEL:

```
Sub ExampleOffset
Documents.Add("scalc4")
Wait 1000
Range("b2:e5").BackGroundColor=RGB(255,0,0)
Wait 1000
Range("b2:e5").Offset(2,5).BackGroundColor=RGB(0,255,0)
End Sub
```

OLEObject-Eigenschaft

Gibt ein Objekt mit den Eigenschaften und Methoden des enthaltenen Objekts zurück.

KLASSE:
TableOLEObject

SYNTAX:
OleObject

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleOLEObject
Dim oOLE As Object
Documents.add "sCalc4"
ActiveDocument.Range("B1").Insert "London"
ActiveDocument.Range("C1").Insert "New York"
ActiveDocument.Range("D1").Insert "Hamburg"
ActiveDocument.Range("A2").Insert "VW"
```

```
ActiveDocument.Range("A3").Insert "Ford"
ActiveDocument.Range("A4").Insert "Pontiac"
For a=2 to 4
Activedocument.Cells(2, a).Insert  a
Activedocument.Cells(3, a).Insert  a+1
Activedocument.Cells(4, a).Insert  a+2
next a
Range("B6").InsertObjectStarChart(True, True, False, "A1:D4")
With ActiveDocument.OLEObjects(1).OLEObject
Wait 2000
REM Hier kann man das Chart bearbeiten
End With
ActiveDocument.OLEObjects(1).Select
oOLE = ActiveDocument.SelectedObject
MsgBox oOLE.Name,0,"Name des selektierten OLE-Objektes"
end sub
```

OLEObjects-Eigenschaft

Ermöglicht den Zugriff auf die Collection OLEObjects, über die auf OLE-Objekte im Dokument zugegriffen werden kann. So liefert beispielsweise "OLEObjects(1).Name" den Namen des ersten Objekts, z.B. eines Charts, im Dokument.

KLASSE:
TableDocument

SYNTAX:
OLEObjects

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleStarChart
Documents.Add("scalc4")
Wait 2000
REM Tabelle mit Daten füllen
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Frauen")
Range("b3").Insert("1200")
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Männer")
Range("b4").Insert("400")
Range("c4").Insert("200")
```

```
Range("d4").Insert("1000")
Range("e5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
wait 2000
REM Daten verändern
Range("a4").Insert("Jungens")
Range("b4").Insert("1300")
Range("c4").Insert("150")
Range("d4").Insert( 0.5 )
wait 3000
Range("e5").UpdateChart(TRUE)
REM neue Daten hinzufügen
Range("b7").Insert( "Mädchen" )
Range("b8").Insert( "Babies" )
Range("c7").Insert( 23 )
Range("e8").Insert( 120 )
ActiveDocument.AddChartData( OLEObjects(1).Name , "b7:e8")
wait 3000
REm anderen Chartbereich festlegen
Range("c6").Insert("Dubai")
Range("d6").Insert("Paris")
Range("e6").Insert("Hong Kong")
ActiveDocument.ChangeChartData( OLEObjects(1).Name , "b6:e8" ,
TRUE, TRUE)
End Sub
```

Order-Eigenschaft

Bestimmt oder ermittelt die Suchrichtung.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Order

RÜCKGABE:
Long

PARAMETER:
[= Value]: Gibt an, ob zeilen- (1) oder spaltenweise (0) gesucht werden soll.

BEISPIEL:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
```

```
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Other-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche vertauscht werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Other

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, wieviele Zeichen des Suchbegriffs vertauscht sein dürfen, damit die Suche einen Treffer meldet.

BEISPIEL:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Eigenschaften und Methoden mit "P"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "P" beginnen.

PageFormatDialog-Methode

Ruft den Dialog zum Bearbeiten der Seitenvorlage auf.

KLASSE:
TableSelection

SYNTAX:
PageFormatDialog()

BEISPIEL:

```
Sub ExampleEditPageFormatDialog
Documents.Add "sCalc4"
Wait 100
ActiveTable.PageFormatDialog
End Sub
```

ParaStyle-Eigenschaft

Gibt die aktuell eingestellte Zellvorlage der Zelle oder des Zellbereichs als String zurück. Beinhaltet ein Zellbereich mehrere Vorlagen, wird ein Leerstring zurückgegeben.

KLASSE:
RangeSelection

SYNTAX:
ParaStyle

RÜCKGABE:
String

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleParaStyle
Documents.Add("scalc4")
Wait 500
MsgBox ActiveCell.ParaStyle,0,"Die verwendete Zellvorlage:"
End Sub
```

Paste-Methode

Fügt den Inhalt der Zwischenablage an die Position der Einfügemarke ein.

KLASSE:
RangeSelection

SYNTAX:
Paste()

BEISPIEL:

```
Sub ExampleCopyPaste
Documents.Add("scalc4")
ActiveCell.Insert("10")
ActiveDocument.PutCell(1,2,1,"=a1 * 2")
ActiveDocument.Cells(1,2,1).ToggleRelative REM absolute Refe-
renzen verwenden
ActiveDocument.Cells(1,2,1).Copy
ActiveDocument.Cells(3,4,1).Paste
End Sub
```

PasteSpecial-Methode

Ruft den Dialog INHALTE EINFÜGEN auf. Dieser Dialog ist abhängig davon, ob es sich bei den Daten im Clipboard um Calc-Zellen oder anderes handelt.

KLASSE:
RangeSelection

SYNTAX:
PasteSpecial()

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
ActiveCell.PasteSpecial
End Sub
```

Pitch-Eigenschaft

Bestimmt oder ermittelt, ob die Schriftart des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen proportional ist.

KLASSE:
TableFont

SYNTAX:
Pitch

RÜCKGABE:
Integer

PARAMETER:
[= PropFont]: Integer-Ausdruck, der angibt, ob die Schriftart proportional ist. Folgende Werte sind möglich:

0 - Unbekannte Laufweite

1 - Nichtproportionale Schriftart

2 - Proportionale Schriftart

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

PrintPreview-Eigenschaft

Bestimmt oder ermittelt die Option für die Seitenansicht.

Mit "Selection.PrintPreview = TRUE" wird die Seitenansicht geöffnet und die erste Seite der Tabelle angezeigt, auf der der Range liegt.

KLASSE:
RangeSelection

SYNTAX:
PrintPreview

Rückgabe:
Boolean

Beispiel:

```
Sub ExamplePrintArea
Documents.Add("scalc4")
ActiveCell.Insert("=a2+1")
ActiveCell.Cut
ActiveDocument.Range("a1:h8").Paste
ActiveDocument.Range("a1:b2").AddPrintArea
ActiveDocument.Range("c3:d4").AddPrintArea
ActiveDocument.Range("e5:f6").AddPrintArea
ActiveDocument.Range("g7:h8").AddPrintArea
ActiveDocument.Range("c3:d4").PrintPreview = TRUE
End Sub
```

Prop-Eigenschaft

Bestimmt oder ermittelt die zur Schriftgröße der verknüpften Absatzvorlage relative Schriftgröße der selektierten Zeichen.

Klasse:
TableFont

Syntax:
Prop

Rückgabe:
Integer

Parameter:
[= Value]: Integer-Ausdruck, der die relative Schriftgröße in Prozent angibt.

Beispiel:

```
Print ActiveCell.Font.Prop
```

Protect-Methode

Schützt die aktuelle Tabelle oder hebt den Schutz der aktuellen Tabelle auf. Der Schutz ist nur für die jeweilige Tabelle gültig.

Klasse:
TableSelection

Syntax:
Protect(Protect As Boolean, Password As String)

Parameter:
Protect: Boolescher Ausdruck, der angibt, ob die Tabelle geschützt (True)

oder der Tabellenschutz aufgehoben (False) werden soll.
Password: Zeichenfolgenausdruck, der das Paßwort angibt, das den Zugriff auf die aktuelle Tabelle ermöglicht.

BEISPIEL:

```
Sub ExampleProtect
Const sPassWord="Test"
Documents.Add "sCalc4"
Wait 100
Range("A2").Insert("Das zweite Tabellenblatt ist schreibge-
schützt.")
Range("A3").Insert("Das Kennwort lautet: " & sPassWord)
ActiveDocument.Tables(2).Protect( True, sPassWord)
End Sub
```

PutCell-Methode

Schreibt eine Zeichenfolge in die angegebene Zelle.

Diese Methode hat die gleiche Funktion wie die Insert-Methode. Im Vergleich bietet die PutCell-Methode jedoch den Vorteil, daß sie deutlich schneller arbeitet.

KLASSE:
TableDocument

SYNTAX:
PutCell(Column As Long, Row As Long, Table As Long, Text As String)

PARAMETER:
Column: Nummer der Spalte, in der sich die Zelle befindet.

Row: Nummer der Zeile, in der sich die Zelle befindet.

Table: Index der Tabelle in der Tables-Collection, auf die sich die Parameter Column und Row beziehen.

Text: Zeichenfolge, die den Text angibt, der in die Zelle geschrieben werden soll.

BEISPIEL:

```
Sub ExampleCellText
Const sText1="Das ist ein "
Const sText2="Baum"
Dim sText As String
Documents.Add "sCalc4"
ActiveDocument.Cells(1,1,1).Insert( sText1 )
ActiveDocument.Cells(2,1,1).Insert( sText2 )
REM Texte verketten mit einer Formel
```

```
ActiveDocument.Cells(2,3,1).Insert( "=VERKETTEN(A1;B1)" )
sText= ActiveDocument.CellText(2,3,1)
ActiveDocument.PutCell(2,4,1, sText )
End Sub
```

Eigenschaften und Methoden mit "R"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "R" beginnen.

Range-Methode

Gibt ein Range-Objekt für den angegebenen Zellbereich zurück.

KLASSE:
TableDocument

SYNTAX:
Range(Range As String, KeepSelect As Boolean)

RÜCKGABE:
Object

PARAMETER:
Range: Zellbereich (z.B. "Tabelle1.a1" oder "a1:b3")

KeepSelect: wird momentan nicht ausgewertet

BEISPIEL:

```
Sub ExampleInsertRange
Documents.Add("scalc4")
REM Datenblock definieren
ActiveCell.Insert("100")
Range("a1:a10").FillSeries("D","A")
Range("a1:g10").FillSeries("R","D","10")
Wait 1000
Range("b2").Select
Range("b2").InsertCell(">")
Wait 500
Range("c3:e4").Select
Range("c3:e4").InsertCellsDown
Wait 500
Range("d4").Select
Range("d4").InsertColumns
Wait 500
Range("e5:f6").Select
Range("e5:f6").InsertRows
Wait 500
```

```
Range("g7:h8").Select
Range("g7:h8").InsertCellsRight
End Sub
```

RefreshArrows-Methode

Aktualisiert alle Detektiv-Pfeile auf der Tabelle. Es müssen Detektiv-Pfeile vorhanden sein.

KLASSE:
RangeSelection

SYNTAX:
RefreshArrows()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

RegExp-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche nach regulären Ausdrücken gesucht wird.

KLASSE:
TableEditView

SYNTAX:
RegExp

Rückgabe:
Boolean

Parameter:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Suche mit Hilfe von Platzhaltern erfolgen soll (True) oder nicht (False).

Beispiel:

```
Sub ExampleSearchsettings
Const sText="Wort"
Const sText1=" ,das am Anfang steht."
Const sText2="Ein anderes Wort"
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert sText & sText1
ActiveDocument.Range("g8").Insert sText2 & sText1
ActiveDocument.Range("e23").Insert sText & sText1
ActiveWindow.SearchSettings.RegExp = TRUE
REM Text nur am Anfang einer Zelle ersetzen
ActiveWindow.Replace "^" & sText,sText2
REM und wiederholen
ActiveWindow.RepeatSearch
ActiveWindow.SearchSettings.RegExp = FALSE
End Sub
```

Relaxed-Eigenschaft

Bestimmt oder ermittelt, ob die drei Randbedingungen der Shorter-, Longer- und Other-Eigenschaft für die Ähnlichkeitssuche kombiniert werden sollen.

Siehe auch: SearchSettings, Shorter, Longer, Other.

Klasse:
TableEditView

Syntax:
Relaxed

Rückgabe:
Boolean

Parameter:
[= Switch:] Boolescher Ausdruck, der angibt, ob die Ähnlichkeitssuche mit einer Kombination der drei Randbedingungen erfolgen soll:

True: Der Suchbegriff gilt dann als gefunden, wenn das Wort im Dokument durch eine beliebige Kombination der drei Randbedingungen erzeugt werden kann.

False: Die Ähnlichkeitssuche wird ohne eine Kombination der drei Randbedingungen durchgeführt.

Beispiel:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Remove-Methode

Löscht eine Tabelle aus der Arbeitsmappe.

Klasse:
TableSelection

Syntax:
Remove(Index As Long)

Parameter:
Index: Index der Tabelle, die gelöscht werden soll.

Beispiel:

```
Sub ExampleTableRemove
Dim iCount As Integer
REM Löscht alle Blätter bis auf eines
Documents.Add("sCalc4")
Wait 1000
iCount= ActiveDocument.Tables.Count
Do
ActiveDocument.Tables(iCount).Remove
iCount = iCount - 1
Loop until iCount <=1
End Sub
```

Repaint-Methode

Löst einen Repaint für die View aus. Dabei wird die Bildschirmanzeige des Fensterinhalts neu aufgebaut.

KLASSE:
TableEditView

SYNTAX:
Repaint()

BEISPIEL:

```
Sub ExampleRepaint
Documents.Add("sCalc4")
Wait 1000
ActiveWindow.Repaint
End Sub
```

RepeatSearch-Methode

Setzt die Suche nach dem zuletzt gesuchten Ausdruck fort. Dabei wird die Suche, die mittels der Search-Methode durchgeführt wurde, wiederholt. Nach Ausführung der RepeatSearch-Methode steht die Markierung auf dem nächsten Suchbegriff.

Siehe auch: Replace, Search, SearchAll

KLASSE:
TableEditView

SYNTAX:
RepeatSearch()

BEISPIEL:

```
Sub ExampleSearchsettings
Const sText="Wort"
Const sText1=" ,das am Anfang steht."
Const sText2="Ein anderes Wort"
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert sText & sText1
ActiveDocument.Range("g8").Insert sText2 & sText1
ActiveDocument.Range("e23").Insert sText & sText1
ActiveWindow.SearchSettings.RegExp = TRUE
REM Text nur am Anfang einer Zelle ersetzen
ActiveWindow.Replace "^" & sText,sText2
REM und wiederholen
ActiveWindow.RepeatSearch
ActiveWindow.SearchSettings.RegExp = FALSE
End Sub
```

Replace-Methode

Sucht und ersetzt eine Zeichenfolge. Vor dem Ersetzen der gefundenen Zeichenfolge erfolgt eine Abfrage.

Siehe auch: RepeatSearch, ReplaceAll, Search.

KLASSE:
TableEditView

SYNTAX:
Replace(SearchString As String, ReplaceString As String)

PARAMETER:
SearchString: Zeichenfolgenausdruck, der den Suchtext angibt, der ersetzt werden soll.

ReplaceString: Zeichenfolgenausdruck, der den Text angibt, durch den SearchString ersetzt werden soll.

BEISPIEL:

```
Sub ExampleSearchsettings
Const sText="Wort"
Const sText1=" ,das am Anfang steht."
Const sText2="Ein anderes Wort"
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert sText & sText1
ActiveDocument.Range("g8").Insert sText2 & sText1
ActiveDocument.Range("e23").Insert sText & sText1
ActiveWindow.SearchSettings.RegExp = TRUE
REM Text nur am Anfang einer Zelle ersetzen
ActiveWindow.Replace "^" & sText,sText2
REM und wiederholen
ActiveWindow.RepeatSearch
ActiveWindow.SearchSettings.RegExp = FALSE
End Sub
```

ReplaceAll-Methode

Sucht und ersetzt eine Zeichenfolge im gesamten Dokument. Vor dem Ersetzen der gefundenen Zeichenfolge erfolgt keine Abfrage.

Siehe auch:Replace, SearchAll.

KLASSE:
TableEditView

SYNTAX:
ReplaceAll(SearchString As String, ReplaceString As String)

PARAMETER:
SearchString: Zeichenfolgenausdruck, der den Suchtext angibt, der ersetzt werden soll.

ReplaceString: Zeichenfolgenausdruck, der den Text angibt, durch den SearchString ersetzt werden soll.

BEISPIEL:

```
Sub ExampleReplaceAll
Documents.Add "sCalc4"
ActiveCell.Insert(100)
ActiveCell.AutoFill "a4"
Range("c1").Insert(10)
Range("c1").AutoFill "c4"
ActiveDocument.Range("B2").Insert "=A1+A2"
ActiveDocument.Range("B1").Insert "=A3+A4"
REM Hier müssen eventuell noch mehr Settings beachtet werden.
ActiveWindow.SearchSettings.Mode = 0 REM Nach Formeln suchen
ActiveWindow.ReplaceAll "A", "C"
End Sub
```

ResetAttributes-Methode

Setzt alle Schriftattribute zurück, die dem selektierten Bereich zuvor zugewiesen wurden.

KLASSE:
RangeSelection

SYNTAX:
ResetAttributes()

BEISPIEL:

```
Sub ExampleDefineStyleTemplates
Const sCellTemplate = "iv32000" REM definiert Bereich außerhalb
Const sTempStd="Standard"
Const sTemp1="Rot"
Const sTemp1a="Rot Und Zentriert"
Const sTemp2="Zentriert"
Documents.Add("scalc4")
Wait 500
Application.DesignerDialog = TRUE
with Range( sCellTemplate ) REM über diesen Bereich werden
Zellvorlagen definiert
   .ResetAttributes REM harte Attribute löschen
   .StyleApply( sTempStd ,2) REM Zelle auf Standard setzen
   .StyleNewByExample( sTemp1 ,2) REM setzt die Zellvorlage
zurück

   .Interior.BackColor = RGB (255,127,127)
   .Interior.Style = 1
```

```
   .StyleUpdateByExample( sTemp1 ,2) REM setzt die Zellvorlage
auf rot

   .ResetAttributes REM Zelle auf Standard setzen
   .StyleApply( sTempStd ,2)
   .AlignHCenter  REM Text zentriert
   .StyleNewByExample( sTemp2,2)

   .ResetAttributes
   .StyleApply( sTemp1 ,2)
   .AlignHCenter REM Text zentriert
   .StyleNewByExample( sTemp1a ,2) REM Kind-Vorlage von
Style_1, mit zusätzlich zentriertem Text
   .ResetAttributes
End With
Range("b2").ColumnWidth = 5000
Range("b1").Insert("Das ist rot")
Range("b1").StyleApply( sTemp1 ,2)
Range("b2").Insert("Das ist rot und zentriert")
Range("b2").StyleApply( sTemp1a ,2)
Range("b3").Insert("Das ist nur zentriert")
Range("b3").StyleApply( sTemp2 ,2)
End Sub
```

Row-Eigenschaft

Die Eigenschaft gibt die Nummer der letzten Zeile des Zellbereichs eines Range an (ab 1 gezählt).

KLASSE:
RangeSelection

SYNTAX:
Row

RÜCKGABE:
Long

PARAMETER:
(nur lesen)

BEISPIEL:

```
REM Folgendes Beispiel rechnet einen Bereich in Integer-Größen
um.
REM Einige Funktionen wie Cells(...) benötigen Integer-Werte um
Zellen zu referenzieren,
REM dazu kann man an zentraler Stelle den zu bearbeiteten
Bereich in Indexgrenzen umrechnen.
Sub ExampleRCT
DIM iLeft As Integer,iTop As Integer,iRight As Integer,iBottom
As Integer, iTable As Integer
DIM sText As String
```

```
Documents.Add("scalc4")
Wait 20
Range("b2:c4").Select
sText = Selection.Address REM gibt den selektierten Bereich als
Bereichstext zurück
AddressRC( sText, iLeft, iTop, iRight, iBottom, iTable )   REM
Wait 1000
ActiveDocument.Cells(1,5,1).Insert("Index")
ActiveDocument.Cells(2,5,1).Insert("Wert")
ActiveDocument.Cells(1,6,1).Insert("Links")
ActiveDocument.Cells(2,6,1).Insert(iLeft)
ActiveDocument.Cells(1,7,1).Insert("Rechts")
ActiveDocument.Cells(2,7,1).Insert(iRight)
ActiveDocument.Cells(1,8,1).Insert("Oben")
ActiveDocument.Cells(2,8,1).Insert(iTop)
ActiveDocument.Cells(1,9,1).Insert("Unten")
ActiveDocument.Cells(2,9,1).Insert(iBottom)
ActiveDocument.Cells(1,10,1).Insert("Tabellenindex")
ActiveDocument.Cells(2,10,1).Insert(iTable)
End Sub
Sub AddressRC(ByVal srAddress  As String, srLeft As Integer,
srTop As Integer, srRight As Integer, srBottom As Integer,
srTab As integer)
DIM iC As Integer, sText As String
iC = instr( 1, srAddress , ".")
If iC > 0 Then
srTab = Range( srAddress ).Table
srAddress = Mid(  srAddress, iC+1 )
Else
srTab = 0 REM Bereichstext enthält keinen Tabellen-Namen
End If
iC = instr( 1, srAddress , ":")
If iC > 0 Then
sText = Left( srAddress, iC -1 ) REM Start des Bereiches
srLeft = Range(sText).Column
srTop = Range(sText).Row
End If
srRight  = Range(srAddress).Column  REM End des Bereiches
srBottom = Range(srAddress).Row
End Sub
```

RowHeight-Eigenschaft

Setzt die Höhe der selektierten oder der angegebenen Tabellenzeile.

KLASSE:
RangeSelection

SYNTAX:
RowHeight

RÜCKGABE:
Long

PARAMETER:
= Value (nur Schreiben): Long-Integer-Ausdruck, der die Höhe der selektierten oder angegebenen Tabellenzeile in Twips angibt. Dieser Wert kann nicht abgefragt werden.

BEISPIEL:

```
Sub ExampleTurnAround
Const sHello="Guten Tag "
Dim sText as String,iCount As Integer
Documents.Add("scalc4")
Wait 1000
With ActiveCell
.ResetAttributes
.Insert( sHello )
.Font.Bold = 8
.RowHeight=3000
.ColumnWidth=3000
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignTop
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignRight
Wait( 1000 )
.AlignVCenter
Wait( 1000 )
.AlignBottom
Wait( 1000 )
.AlignHCenter
Wait( 1000 )
.AlignLeft
Wait( 1000 )
.AlignBlock
sText=""
For iCount= 1 To 30
sText=sText+ sHello
.Insert( sText )
Next
End With
End Sub
```

Eigenschaften und Methoden mit "S"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "S" beginnen.

Scale-Methode

Ruft den MAßSTAB-DIALOG auf.

KLASSE:
TableEditView

SYNTAX:
Scale()

BEISPIEL:

```
Sub ExampleInsertApplet
Documents.Add "sCalc4"
Activewindow.Scale
End Sub
```

ScEditOptions-Methode

Ruft den Dialog OPTIONEN TABELLENDOKUMENT auf.

KLASSE:
StarCalc

SYNTAX:
ScEditOptions()

BEISPIEL:

```
Sub ExampleStarCalc
Application.StarCalc.ScEditOptions
End Sub
```

Search-Methode

Sucht ab der Position der Schreibmarke nach einem bestimmten Text.

Siehe auch: RepeatSearch, Replace, SearchAll.

KLASSE:
TableEditView

SYNTAX:
Search(SearchString As String)

PARAMETER:
SearchString: Zeichenfolgenausdruck, der den Text aufnimmt, nach dem gesucht werden soll.

BEISPIEL:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

SearchAll-Methode

Sucht im gesamten Dokument nach einem bestimmten Text.

Siehe auch: ReplaceAll, Search.

KLASSE:
TableEditView

SYNTAX:
SearchAll(SearchString As String)

PARAMETER:
SearchString: Zeichenfolgenausdruck, der den Text aufnimmt, nach dem gesucht werden soll.

BEISPIEL:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
```

```
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

SearchSettings-Eigenschaft

Liefert ein SearchSettings-Objekt zurück und ermöglicht somit einen Zugriff die Suchoptionen.

KLASSE:
TableEditView

SYNTAX:
SearchSettings

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleSearchsettings
Const sText="Wort"
Const sText1=" ,was am Anfang steht."
Const sText2="Ein anderes Wort"
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert sText & sText1
ActiveDocument.Range("g8").Insert sText2 & sText1
ActiveDocument.Range("e23").Insert sText & sText1
Activewindow.Searchsettings.RegExp = TRUE
REM Text nur am Anfang einer Zelle ersetzen
ActiveWindow.Replace "^" & sText,sText2
REM und wiederholen
```

```
ActiveWindow.RepeatSearch
Activewindow.Searchsettings.RegExp = FALSE
End Sub
```

Select-Methode

Selektiert den im Range-Objekt angegebenen Bereich oder das angegebene OLE-Objekt.

KLASSE:
TableOLEObject, RangeSelection

SYNTAX:
Select()

BEISPIEL:

```
Sub ExampleSelect
Documents.Add "sCalc4"
Wait 100
Range("B2:D5").Select
Wait 2000
ActiveWindow.Deselect
End Sub
```

SelectAll-Methode

Markiert die gesamte Tabelle.

KLASSE:
TableEditView

SYNTAX:
SelectAll()

BEISPIEL:

```
Sub ExampleSelection
Documents.Add("scalc4")
Wait 1000
ActiveWindow.SelectAll
ActiveDocument.Selection.BackgroundColor = RGB(255,128,128)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectRow
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectColumn
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
End Sub
```

SelectColumn-Methode

Markiert die gesamte aktuelle Spalte.

KLASSE:
TableEditView

SYNTAX:
SelectColumn()

BEISPIEL:

```
Sub ExampleSelection
Documents.Add("scalc4")
Wait 1000
ActiveWindow.SelectAll
ActiveDocument.Selection.BackgroundColor = RGB(255,128,128)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectRow
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectColumn
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
End Sub
```

SelectedObject-Eigenschaft

Gibt das in der aktiven View gerade selektierte OLE-Objekt als ein Objekt der Klasse OLEObject zurück.

KLASSE:
TableDocument

SYNTAX:
SelectedObject

RÜCKGABE:
Object

BEISPIEL:

```
Sub ExampleOLEObject
Dim oOLE As Object
Documents.Add "sCalc4"
ActiveDocument.Range("B1").Insert "London"
ActiveDocument.Range("C1").Insert "New York"
ActiveDocument.Range("D1").Insert "Hamburg"
ActiveDocument.Range("A2").Insert "VW"
ActiveDocument.Range("A3").Insert "Ford"
ActiveDocument.Range("A4").Insert "Pontiac"
For a=2 to 4
ActiveDocument.Cells(2, a).Insert  a
```

```
ActiveDocument.Cells(3, a).Insert  a+1
ActiveDocument.Cells(4, a).Insert  a+2
next a
Range("B6").InsertObjectStarChart(True, True, False, "A1:D4")
With ActiveDocument.OLEObjects(1).OLEObject
Wait 2000
REM Hier kann man das Chart bearbeiten
End With
ActiveDocument.OLEObjects(1).Select
oOLE = ActiveDocument.SelectedObject
MsgBox oOLE.Name,0,"Name des selektierten OLE-Objektes"
End Sub
```

Selection-Eigenschaft

Gibt ein Range-Objekt für die Selektion in der aktiven View des Dokuments zurück.

KLASSE:
TableDocument

SYNTAX:
Selection

RÜCKGABE:
Object

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleSelection
Documents.Add("scalc4")
Wait 1000
ActiveWindow.SelectAll
ActiveDocument.Selection.BackGroundColor = RGB(255,128,128)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectRow
ActiveDocument.Selection.BackGroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectColumn
ActiveDocument.Selection.BackGroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
End Sub
```

SelectRow-Methode

Markiert die gesamte aktuelle Zeile.

KLASSE:
TableEditView

SYNTAX:
SelectRow()

BEISPIEL:

```
Sub ExampleSelection
Documents.Add("scalc4")
Wait 1000
ActiveWindow.SelectAll
ActiveDocument.Selection.BackgroundColor = RGB(255,128,128)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectRow
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
ActiveWindow.SelectColumn
ActiveDocument.Selection.BackgroundColor = RGB(0,255,0)
ActiveWindow.GoToCell("b2")
End Sub
```

SetHyperlink-Methode

Fügt einen Hyperlink ein. Um einen Hyperlink als Text einzufügen, muß diese Methode an der Selektion aufgerufen werden, also "Selection.SetHyperlink(...)". Wird er als Button eingefügt, so können auch andere RangeSelection-Objekte genommen werden, also ActiveCell oder Range(...).

Wird der Hyperlink als Text eingefügt, verbleibt die Zelle im Editiermodus.

KLASSE:
RangeSelection

SYNTAX:
SetHyperlink(LinkName As String, LinkURL As String, LinkTarget As String, LinkType As Long)

PARAMETER:
LinkName: Gibt den Namen an, der als Hyperlink angezeigt wird.

LinkURL: Gibt die URL-Adresse an, zu der der Link führen soll.

LinkTarget: Bestimmt den Zielrahmen. Folgende TargetFrame-Typen gibt es:

_self: Das Folgedokument wird im selben Frame aufgebaut.

_top: Das Folgedokument wird in einem Top-Window, also im höchsten Frame der Hierarchie, aufgebaut; ist der Ausgangs-Frame bereits ein Top-Window, wird das Dokument im selben Frame aufgebaut.

_parent: Das Folgedokument wird in einem Parent, also einem übergeordneten Frame, aufgebaut; gibt es keinen Parent, wird das Dokument im selben Frame aufgebaut.

_clear: Das Folgedokument wird in einer eigenen View, also in einem leeren neuen Frame, aufgebaut.

LinkType: Gibt an, ob der Hyperlink als Text oder als Button eingefügt werden soll:

1 - als Text (nur eingeschränkte Funktionalität)

2 - als Button

Beispiel:

```
Sub ExampleSetHyperlink
Documents.Add("scalc4")
ActiveWindow.GotoCell("b2")
Range("b3:c5").SetHyperlink( "Star Division GmbH",
"http://www.stardivision.de", "_self" , 2 )
End Sub
```

SetInputMode-Methode

Schaltet für die aktive Zelle, in der der Cursor steht, in den Edit-Modus.

Klasse:
RangeSelection

Syntax:
SetInputMode()

Beispiel:

```
Sub ExampleSetInputMode
Documents.Add("scalc4")
Wait 1000
ActiveCell.Insert("Zelle A1 wird in den Einfügemodus
geschaltet.")
Wait 1000
ActiveCell.SetInputMode
End Sub
```

SetOptimalColumnWidth-Methode

Berechnet für den selektierten oder den angegebenen Bereich die optimale Spaltenbreite und weist diese dem Bereich zu. Ohne Übergabeparameter wird der Dialog Optimale Spaltenbreite aufgerufen.

Klasse:
RangeSelection

Syntax:
SetOptimalColumnWidth(aExtraWidth As Long)

Parameter:
aExtraWidth: Long-Integer-Ausdruck, der die Breite angibt, die dem selektierten oder angegebenen Bereich zusätzlich zur optimalen Spaltenbreite zugewiesen werden soll. Soll zur optimalen Breite kein Extra-Wert addiert werden, muß dieser Parameter mit dem Wert 0 übergeben werden. Wird er nicht gesetzt, öffnet diese Methode den Dialog Optimale Spaltenbreite.

Beispiel:

```
Sub ExampleOptimalSize
Dim sText as String, iCount As Integer
Documents.Add("scalc4")
Wait 1000
With ActiveCell
.Insert("x4u") REM Ein X für ein U !
Wait 1000
.SetOptimalColumnWidthDirect()
Wait 1000
.SetOptimalColumnWidth( 3000 )
.AlignHCenter
sText=""
For iCount= 1 To 30
sText=sText+"Hello "
.Insert( sText )
Next
Wait 1000
.RowHeight = 2000
.WrapText = TRUE
Wait 1000
.SetOptimalRowHeight( 0 )
End With
End Sub
```

SetOptimalColumnWidthDirect-Methode

Stellt die optimale Spaltenbreite ein, mit dem Standard-Extra-Wert von 0,2 cm.

KLASSE:
RangeSelection

SYNTAX:
SetOptimalColumnWidthDirect()

BEISPIEL:

```
Sub ExampleOptimalSize
Dim sText as String, iCount As Integer
Documents.Add("scalc4")
Wait 1000
With ActiveCell
.Insert("x4u") REM Ein X für ein U !
Wait 1000
.SetOptimalColumnWidthDirect()
Wait 1000
.SetOptimalColumnWidth( 3000 )
.AlignHCenter
sText=""
For iCount= 1 To 30
sText=sText+"Hello "
.Insert( sText )
Next
Wait 1000
.RowHeight = 2000
.WrapText = TRUE
Wait 1000
.SetOptimalRowHeight( 0 )
End With
End Sub
```

SetOptimalRowHeight-Methode

Berechnet für den selektierten oder den angegebenen Bereich die optimale Zeilenhöhe und weist diese dem Bereich zu. Ohne Übergabeparameter wird der Dialog OPTIMALE ZEILENHÖHE aufgerufen.

KLASSE:
RangeSelection

SYNTAX:
SetOptimalRowHeight(aExtraHeight As Long)

PARAMETER:
aExtraHeight: Long-Integer-Ausdruck, der die Höhe angibt, die dem selek-

tierten oder angegebenen Bereich zusätzlich zur optimalen Höhe zugewiesen werden soll. Soll zur optimalen Höhe kein Extra-Wert addiert werden, muß dieser Parameter mit dem Wert 0 übergeben werden. Wird er nicht gesetzt, öffnet diese Methode den Dialog OPTIMALE ZEILENHÖHE.

BEISPIEL:

```
Sub ExampleOptimalSize
Dim sText as String, iCount As Integer
Documents.Add("scalc4")
Wait 1000
With ActiveCell
.Insert("x4u") REM Ein X für ein U !
Wait 1000
.SetOptimalColumnWidthDirect()
Wait 1000
.SetOptimalColumnWidth( 3000 )
.AlignHCenter
sText=""
For iCount= 1 To 30
sText=sText+"Hello "
.Insert( sText )
Next
Wait 1000
.RowHeight = 2000
.WrapText = TRUE
Wait 1000
.SetOptimalRowHeight( 0 )
End With
End Sub
```

Shorter-Eigenschaft

Bestimmt oder ermittelt, wieviele Zeichen bei der Ähnlichkeitssuche aus einem Wort entfernt werden dürfen, damit gesuchtes und gefundenes Wort noch als gleich gelten.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Shorter

RÜCKGABE:
Long

PARAMETER:
[= Value]: Long-Integer-Ausdruck, der angibt, um wieviele Zeichen das

Wort im Dokument kürzer sein darf als der Suchbegriff, um noch als gleich zu gelten.

Beispiel:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Show-Methode

Stellt die angegebene Tabelle, die zuvor ausgeblendet wurde, wieder dar.

Klasse:
Tables

Syntax:
Show(aTableName As String)

Parameter:
aTableName: Zeichenfolgenausdruck, der die Bezeichnung der Tabelle angibt, die wieder angezeigt werden soll.

Beispiel:

```
Sub ExampleCalcTables
Const sTab1="Tabelle_Versteckt"
Const sTab2="Tabelle_Neu"
Const sTab3="Tabelle_Eingefuegt"
Dim iCount As Integer
Dim sTable As String
Documents.Add("scalc4")
REM alle außer einer Tabelle verstecken
For iCount = 1 To ActiveDocument.Tables.Count -1
ActiveDocument.Tables.Hide(
ActiveDocument.Tables.Item(iCount).Name )
Next iCount
sTable = ActiveDocument.Tables(1).Name
With ActiveDocument.Tables
.Add( sTab1 )
.Add( sTab2 )
.Insert( sTab3 ,2)
.Hide( sTab1 )
```

```
.Show( sTable )
End With
End Sub
```

ShowColumn-Methode

Stellt die im selektierten oder im angegebenen Bereich enthaltenen Tabellenspalten, die zuvor ausgeblendet wurden, wieder dar.

KLASSE:
RangeSelection

SYNTAX:
ShowColumn()

BEISPIEL:

```
Sub ExampleHideShow
Documents.Add("scalc4")
Wait 500
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("a3").HideRow
Range("a3").Insert("Diese Reihe war versteckt.")
Wait 1000
Range("a3").ShowRow
Range("b2").Insert("15")
Range("c2").Insert("200")
Range("d2").Insert("30")
Wait 1000
Range("c2:d2").HideColumn
Range("c2").Insert("War versteckt")
Wait 1000
Range("c2").ShowColumn
Range("e2").Insert("<-- Spalte D ist versteckt")
End Sub
```

ShowDependents-Methode

Zeigt die Spur zu den Nachfolgern der aktuellen Zelle an.

KLASSE:
RangeSelection

SYNTAX:
ShowDependents()

Beispiel:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ShowDetail-Methode

Blendet die äußerste Gliederungs-Gruppe (Zeilen und Spalten) ein, die komplett im Zellbereich enthalten ist.

Klasse:
RangeSelection

Syntax:
ShowDetail()

Beispiel:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
```

```
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

ShowErrors-Methode

Zeigt die Zellen an, die einen Fehler in der markierten Zelle hervorrufen.

KLASSE:
RangeSelection

SYNTAX:
ShowErrors()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ShowInvalid-Methode

Markiert alle Zellen auf der Tabelle, für die ein eingestelltes Gültigkeitskriterium nicht erfüllt ist.

KLASSE:
RangeSelection

SYNTAX:
ShowInvalid()

BEISPIEL:

```
REm Es müssen bereits Gültigkeitskriterien definiert worden
sein.
Range( ActiveTable.Name & ".a1" ).ShowInvalid
```

ShowPrecedents-Methode

Zeigt eine Spur vom Vorgänger der aktuellen Zelle zu dieser Zelle an.

KLASSE:
RangeSelection

SYNTAX:
ShowPrecedents()

BEISPIEL:

```
Sub ExampleArrows
Documents.Add("scalc4")
Range("a2").Insert( "=a4" )
Range("a4").Insert( "0" )
Range("b1").Insert( "123" )
Range("b4").Insert( "=b1/a2" )
Wait 2000
Range("a2").ShowDependents
Wait 2000
Range("a2").ShowPrecedents
Range("d4").Insert( "=b4" )
Wait 2000
Range("d4").ShowErrors
Wait 2000
Range("d4").ClearArrowPrecedents
Wait 2000
Range("a4").Insert( "10" )
Wait 2000
Range("b4").RefreshArrows
Wait 2000
Range("a2").ClearArrowDependents
Wait 2000
ActiveTable.ClearArrows
End Sub
```

ShowRow-Methode

Stellt die im selektierten oder im angegebenen Bereich enthaltenen Tabellenzeilen, die zuvor ausgeblendet wurden, wieder dar.

KLASSE:
RangeSelection

SYNTAX:
ShowRow()

BEISPIEL:

```
Sub ExampleHideShow
Documents.Add("scalc4")
Wait 500
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("a3").HideRow
Range("a3").Insert("Diese Reihe war versteckt.")
Wait 1000
Range("a3").ShowRow
Range("b2").Insert("15")
Range("c2").Insert("200")
Range("d2").Insert("30")
Wait 1000
Range("c2:d2").HideColumn
Range("c2").Insert("War versteckt")
Wait 1000
Range("c2").ShowColumn
Range("e2").Insert("<-- Spalte D ist versteckt")
End Sub
```

Similarity-Eigenschaft

Bestimmt oder ermittelt, ob die Ähnlichkeitssuche aktiv ist.

Siehe auch: SearchSettings

KLASSE:
TableEditView

SYNTAX:
Similarity

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Ähnlichkeitssuche ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
Sub ExampleSimilarity
Documents.Add "sCalc4"
ActiveDocument.Range("B2").Insert "eTxst"
ActiveWindow.GotoCell "A1"
ActiveWindow.Mode = 1
ActiveWindow.Similarity = True
ActiveWindow.Relaxed = True
ActiveWindow.Other = 2 REM Zeichen ersetzen (x)
ActiveWindow.Shorter = 2 REM Zeichen entfernen (y)
ActiveWindow.Longer = 2 REM Zeichen hinzufügen (z)
ActiveWindow.Search "Test"
MsgBox ActiveCell.TextValue ,0,"Gefundener Text"
End Sub
```

Size-Eigenschaft

Bestimmt oder ermittelt die Schriftgröße der Schrift des selektierten Bereichs.

Klasse:
TableFont

Syntax:
Size

Rückgabe:
Long

Parameter:
[= FontSize]: Long-Integer-Ausdruck, der die Größe der Schrift in Punkt angibt. Der Wert kann auch mit Angabe der gewünschten Maßeinheit als Zeichenfolge gesetzt werden.

Beispiel:

```
Sub ExampleFontSize
Dim iCount As Integer
Documents.Add "sCalc4"
Wait 10
For iCount = 2 To 24 Step 2
ActiveDocument.Cells(2, iCount+1).Font.Size = iCount
ActiveDocument.Cells(2, iCount+1).Insert "Das ist ein Text."
Next iCount
End Sub
```

SortAscending-Methode

Sortiert die Feldinhalte des selektierten oder angegebenen Bereichs in aufsteigender Reihenfolge.

KLASSE:
RangeSelection

SYNTAX:
SortAscending()

BEISPIEL:

```
Sub ExampleSort
Documents.Add("sCalc4")
Wait 10
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("b2").Insert("15")
Range("b3").Insert("200")
Range("b4").Insert("30")
Range("b1").Insert("Aufsteigend")
Range("a1").Insert("Absteigend")
Wait 100
Range("b2:b4").SortAscending
Wait 100
Range("a2:a4").SortDescending
End Sub
```

SortDescending-Methode

Sortiert die Feldinhalte des selektierten oder angegebenen Bereichs in absteigender Reihenfolge.

KLASSE:
RangeSelection

SYNTAX:
SortDescending()

BEISPIEL:

```
Sub ExampleSort
Documents.Add("sCalc4")
Wait 10
Range("a2").Insert("Hallo")
Range("a3").Insert("StarCalc")
Range("a4").Insert("Benutzer")
Range("b2").Insert("15")
Range("b3").Insert("200")
Range("b4").Insert("30")
```

```
Range("b1").Insert("Aufsteigend")
Range("a1").Insert("Absteigend")
Wait 100
Range("b2:b4").SortAscending
Wait 100
Range("a2:a4").SortDescending
End Sub
```

Spelling-Methode

Startet die Rechtschreibprüfung an der Cursorposition.

KLASSE:
TableEditView

SYNTAX:
Spelling()

BEISPIEL:

```
Sub ExampleSpelling
REM Text mit einen Rechtschreibfehler
Const sText = "Das ist ein Text der Korregiert wird."
Documents.Add("sCalc4")
Wait 500
Range("c3").Insert(sText)
ActiveWindow.Spelling
ActiveWindow.Thesaurus
End Sub
```

SplitCell-Methode

Löst einen zusammengefaßten Zellbereich wieder auf. Zusammengefaßte Zellbereiche erhält man durch das Verbinden von Zellen mittels MergeCells.

KLASSE:
RangeSelection

SYNTAX:
SplitCell()

BEISPIEL:

```
Sub ExampleSplitCell
On error Goto SplitCellError
Dim sText As String
Documents.Add("scalc4")
Wait 500
Range("a2").Insert(100)
Range("b3").Insert(" Text")
Range("a2:b3").MergeCells False
Wait 1000
```

```
ActiveDocument.Cells(1,5).Insert("Zellen verbinden ohne
Textübernahme")
sText = Range("a2").TextValue
ActiveDocument.Cells(1,6).Insert("a2="& sText )
sText = Range("b3").TextValue
ActiveDocument.Cells(1,7).Insert("b3=" & sText )
Wait 2000
Range("b3").SplitCell
Wait 2000
Range("a2:b3").MergeCells TRUE
ActiveDocument.Cells(1,9).Insert("Zellen verbinden mit
Textübernahme")
sText = Range("a2").TextValue
ActiveDocument.Cells(1,10).Insert("a2="& sText )
sText = Range("b3").TextValue
ActiveDocument.Cells(1,11).Insert("b3=" & sText )
Wait 2000
Range("a2:b3").SplitCell
Exit Sub
SplitCellError:
Range("a2").SplitCell
Range("e2").Insert("Fehler beim Versuch, eine Zelle zu
teilen.")
Resume Next
End Sub
```

SplitWindow-Methode

Teilt das Fenster an der Cursorposition oder hebt eine Teilung wieder auf.

KLASSE:
TableEditView

SYNTAX:
SplitWindow()

BEISPIEL:

```
Sub Main
Documents.Add "sCalc4"
ActiveWindow.GotoCell "F5"
ActiveWindow.SplitWindow
End Sub
```

StarChartDialog-Methode

Ruft den Dialog zum Einfügen eines StarChart-Diagramms auf.

KLASSE:
RangeSelection

SYNTAX:
StarChartDialog()

BEISPIEL:

```
Sub ExampleFillSeries
Documents.Add("scalc4")
ActiveCell.Insert("T1")
REM Automatisch füllen ...
Range("a1:e1").FillSeries("A")
REM Nach rechts linear mit Startwert 10 und Inkrement 10 füllen
Range("a2:e2").FillSeries("R","L",10,"",10,"")
REM nach unten geometrisch mit Faktor 2 füllen
Range("a2:e5").FillSeries("B","G",2)
Range("a1:e5").StarChartDialog
End Sub
```

StatusDocPos-Eigenschaft

Die Methode gibt einen String zurück, der angibt, welche Tabelle gerade aktiv ist und wieviele Tabellen das Dokument enthält.

KLASSE:
TableEditView

SYNTAX:
StatusDocPos

RÜCKGABE:
String

PARAMETER:
(nur lesen)

BEISPIEL:

```
Sub ExampleStatusDocPos
   Documents.Add("scalc4")
   ActiveCell.Insert( ActiveWindow.StatusDocPos )
End Sub
```

StatusSelectionMode-Eigenschaft

Gibt den aktuellen Selektionsmodus zurück (Read-Only).

KLASSE:
TableEditView

SYNTAX:
StatusSelectionMode

RÜCKGABE:
Long

PARAMETER:
return: Selektionsmodus:

0 = Standard

1 = Erweitern

2 = Ergänzen

BEISPIEL:

```
Sub ExampleStatusSelectionMode
Dim sText As String
Documents.add "sCalc4"
Select Case Activewindow.StatusSelectionMode
Case 0 :sText= "Standard"
Case 1 :sText= "Erweitern"
Case 2 :sText= "Ergänzen"
Case Else sText="unbekannt"
End Select
ActiveCell.Insert("Selektionsmodus: " & sText)
End Sub
```

StyleApply-Methode

Weist dem Zellbereich eine Zellvorlage oder der ganzen Tabelle eine Seitenvorlage zu.

KLASSE:
RangeSelection

SYNTAX:
StyleApply(Template As String, Family As Long)

RÜCKGABE:
Long

PARAMETER:
Template: Name der Vorlage

Family: Bestimmt die Art der Vorlage. Folgende Werte sind möglich:

2 - Zellvorlagen

8 - Seitenvorlagen

return: Flags der Vorlage (uninteressant)

Beispiel:

```
Sub ExampleDefineStyleTemplates
Const sCellTemplate = "iv32000" REM definiert Bereich außerhalb
Const sTempStd="Standard"
Const sTemp1="Rot"
Const sTemp1a="Rot Und Zentriert"
Const sTemp2="Zentriert"
Documents.Add("scalc4")
Wait 500
Application.DesignerDialog = TRUE
with Range( sCellTemplate ) REM über diesen Bereich werden
Zellvorlagen definiert
   .ResetAttributes REM harte Attribute löschen
   .StyleApply( sTempStd ,2) REM Zelle auf Standard setzen
   .StyleNewByExample( sTemp1 ,2) REM setzt die Zellvorlage
zurück

   .Interior.BackColor = RGB (255,127,127)
   .Interior.Style = 1
   .StyleUpdateByExample( sTemp1 ,2) REM setzt die Zellvorlage
auf rot

   .ResetAttributes REM Zelle auf Standard setzen
   .StyleApply( sTempStd ,2)
   .AlignHCenter  REM Text zentriert
   .StyleNewByExample( sTemp2,2)

   .ResetAttributes
   .StyleApply( sTemp1 ,2)
   .AlignHCenter REM Text zentriert
   .StyleNewByExample( sTemp1a ,2) REM Kind-Vorlage von
Style_1, mit zusätzlich zentriertem Text
   .ResetAttributes
End With
Range("b2").ColumnWidth = 5000
Range("b1").Insert("Das ist rot")
Range("b1").StyleApply( sTemp1 ,2)
Range("b2").Insert("Das ist rot und zentriert")
Range("b2").StyleApply( sTemp1a ,2)
Range("b3").Insert("Das ist nur zentriert")
Range("b3").StyleApply( sTemp2 ,2)
End Sub
```

StyleName-Eigenschaft

Bestimmt oder ermittelt den Schriftstil des selektierten Textes oder der an der Schreibmarke einzufügenden Zeichen.

Falls ein Font in dem System, in dem das Dokument geladen wurde, nicht vorhanden ist, dann wird ein ähnlicher Font genommen. Die StyleName-Eigenschaft dient dann nur als Hilfsmittel. Setzen Sie beispielsweise StyleName = Bold, dann weiß das System, daß der Originalfont ein fetter Font war.

KLASSE:
TableFont

SYNTAX:
StyleName

RÜCKGABE:
String

PARAMETER:
[= FontStyle]: Zeichenfolgenausdruck, der den Schriftstil bestimmt. FontStyle können die folgenden Werte zugewiesen werden:

Leerstring (Voreinstellung): Der Text wird normal dargestellt.

Bold: Die Schrift wird fett dargestellt.

Italic: Die Schrift wird kursiv dargestellt.

BEISPIEL:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
```

```
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

StyleNewByExample-Methode

Erzeugt eine neue Vorlage aus den harten Attributen des Zellbereichs. Die neue Vorlage wird dann auf den Bereich angewendet.

KLASSE:
RangeSelection

SYNTAX:
StyleNewByExample(Param As String, Family As Long)

PARAMETER:
Param: Gibt den Namen der neuen Vorlage an.

Family: Bestimmt die Art der Vorlage. Folgende Werte sind möglich:

2 - Zellvorlagen

8 - Seitenvorlagen (Bei Seitenvorlagen ist die neue Vorlage eine Kopie der vorher eingestellten, weil es keine harten Attribute gibt.)

BEISPIEL:

```
Sub ExampleDefineStyleTemplates
Const sCellTemplate = "iv32000" REM definiert Bereich außerhalb
Const sTempStd="Standard"
Const sTemp1="Rot"
Const sTemp1a="Rot Und Zentriert"
Const sTemp2="Zentriert"
Documents.Add("scalc4")
Wait 500
Application.DesignerDialog = TRUE
with Range( sCellTemplate ) REM über diesen Bereich werden
Zellvorlagen definiert
   .ResetAttributes REM harte Attribute löschen
   .StyleApply( sTempStd ,2) REM Zelle auf Standard setzen
   .StyleNewByExample( sTemp1 ,2) REM setzt die Zellvorlage
zurück

   .Interior.BackColor = RGB (255,127,127)
   .Interior.Style = 1
   .StyleUpdateByExample( sTemp1 ,2) REM setzt die Zellvorlage
auf rot

   .ResetAttributes REM Zelle auf Standard setzen
   .StyleApply( sTempStd ,2)
   .AlignHCenter  REM Text zentriert
   .StyleNewByExample( sTemp2,2)
```

```
   .ResetAttributes
   .StyleApply( sTemp1 ,2)
   .AlignHCenter REM Text zentriert
   .StyleNewByExample( sTemp1a ,2) REM Kind-Vorlage von
Style_1, mit zusätzlich zentriertem Text
   .ResetAttributes
End With
Range("b2").ColumnWidth = 5000
Range("b1").Insert("Das ist rot")
Range("b1").StyleApply( sTemp1 ,2)
Range("b2").Insert("Das ist rot und zentriert")
Range("b2").StyleApply( sTemp1a ,2)
Range("b3").Insert("Das ist nur zentriert")
Range("b3").StyleApply( sTemp2 ,2)
End Sub
```

StyleUpdateByExample-Methode

Übernimmt die harten Attribute des Zellbereichs in die eingestellte Vorlage.

KLASSE:
RangeSelection

SYNTAX:
StyleUpdateByExample(Param As String, Family As Long)

PARAMETER:
Param: Gibt die Bezeichnung der neuen Vorlage an.

Family: Gibt den Vorlagentyp an. Nur der Wert 2 (für Zellvorlagen) ist möglich.

BEISPIEL:

```
Sub ExampleDefineStyleTemplates
Const sCellTemplate = "iv32000" REM definiert Bereich außerhalb
Const sTempStd="Standard"
Const sTemp1="Rot"
Const sTemp1a="Rot Und Zentriert"
Const sTemp2="Zentriert"
Documents.Add("scalc4")
Wait 500
Application.DesignerDialog = TRUE
with Range( sCellTemplate ) REM über diesen Bereich werden
Zellvorlagen definiert
   .ResetAttributes REM harte Attribute löschen
   .StyleApply( sTempStd ,2) REM Zelle auf Standard setzen
   .StyleNewByExample( sTemp1 ,2) REM setzt die Zellvorlage
zurück

   .Interior.BackColor = RGB (255,127,127)
   .Interior.Style = 1
```

```
   .StyleUpdateByExample( sTemp1 ,2) REM setzt die Zellvorlage
auf rot

   .ResetAttributes REM Zelle auf Standard setzen
   .StyleApply( sTempStd ,2)
   .AlignHCenter  REM Text zentriert
   .StyleNewByExample( sTemp2,2)

   .ResetAttributes
   .StyleApply( sTemp1 ,2)
   .AlignHCenter REM Text zentriert
   .StyleNewByExample( sTemp1a ,2) REM Kind-Vorlage von
Style_1, mit zusätzlich zentriertem Text
   .ResetAttributes
End With
Range("b2").ColumnWidth = 5000
Range("b1").Insert("Das ist rot")
Range("b1").StyleApply( sTemp1 ,2)
Range("b2").Insert("Das ist rot und zentriert")
Range("b2").StyleApply( sTemp1a ,2)
Range("b3").Insert("Das ist nur zentriert")
Range("b3").StyleApply( sTemp2 ,2)
End Sub
```

StyleWatercanMode-Eigenschaft

Ermittelt, ob der Vorlagen-Gießkannen-Modus aktiv ist.

KLASSE:
RangeSelection

SYNTAX:
StyleWatercanMode

RÜCKGABE:
Boolean

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleWatercan
Documents.Add("sCalc4")
MsgBox (
IIF(ActiveCell.StyleWaterCanMode,"angeschaltet","ausge-
schaltet"),0,"Füllmodus des Stylisten"
end sub
```

Eigenschaften und Methoden mit "T"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "T" beginnen.

Table-Eigenschaft

Gibt die Tabellennummer (ab 1 gezählt) zurück, auf der ein Range liegt, oder die Tabelle, die in einer View angezeigt wird.

KLASSE:
RangeSelection, TableEditView

SYNTAX:
Table

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

BEISPIEL:

```
REM Folgendes Beispiel rechnet einen Bereich in Integer-Größen
um.
REM Einige Funktionen wie Cells(...) benötigen Integer-Werte um
Zellen zu referenzieren,
REM dazu kann man an zentraler Stelle den zu bearbeiteten
Bereich in Indexgrenzen umrechnen.
Sub ExampleRCT
DIM iLeft As Integer,iTop As Integer,iRight As Integer,iBottom
As Integer, iTable As Integer
DIM sText As String
Documents.Add("scalc4")
Wait 20
Range("b2:c4").Select
sText = Selection.Address REM gibt den selektierten Bereich als
Bereichstext zurück
AddressRC( sText, iLeft, iTop, iRight, iBottom, iTable )   REM
Wait 1000
ActiveDocument.Cells(1,5,1).Insert("Index")
ActiveDocument.Cells(2,5,1).Insert("Wert")
ActiveDocument.Cells(1,6,1).Insert("Links")
ActiveDocument.Cells(2,6,1).Insert(iLeft)
ActiveDocument.Cells(1,7,1).Insert("Rechts")
ActiveDocument.Cells(2,7,1).Insert(iRight)
ActiveDocument.Cells(1,8,1).Insert("Oben")
ActiveDocument.Cells(2,8,1).Insert(iTop)
ActiveDocument.Cells(1,9,1).Insert("Unten")
ActiveDocument.Cells(2,9,1).Insert(iBottom)
```

```
ActiveDocument.Cells(1,10,1).Insert("Tabellenindex")
ActiveDocument.Cells(2,10,1).Insert(iTable)
End Sub
Sub AddressRC(ByVal srAddress  As String, srLeft As Integer,
srTop As Integer, srRight As Integer, srBottom As Integer,
srTab As integer)
DIM iC As Integer, sText As String
iC = instr( 1, srAddress , ".")
If iC > 0 Then
srTab = Range( srAddress ).Table
srAddress = Mid(  srAddress, iC+1 )
Else
srTab = 0 REM Bereichstext enthält keinen Tabellen-Namen
End If
iC = instr( 1, srAddress , ":")
If iC > 0 Then
sText = Left( srAddress, iC -1 ) REM Start des Bereiches
srLeft = Range(sText).Column
srTop = Range(sText).Row
End If
srRight  = Range(srAddress).Column  REM End des Bereiches
srBottom = Range(srAddress).Row
End Sub
```

TableCount-Eigenschaft

Liefert die Anzahl der im Dokument enthaltenen Tabellen.

KLASSE:
TableDocument

SYNTAX:
TableCount

RÜCKGABE:
Long

PARAMETER:
(nur Lesen)

BEISPIEL:

```
Sub ExampleJumpToTable
Dim sDoc As String, iCount As Integer
Documents.Add "sCalc4"
Wait 1000
iCount = ActiveDocument.TableCount
Wait 1000
ActiveCell.Insert("Neu erstellte Kalkulationen besitzen immer "
& iCount & " Blätter.")
For iCount = 2 To ActiveDocument.TableCount
Wait 1000
ActiveWindow.JumpToNextTable
```

```
ActiveCell.Insert("Alle Blätter einmal durchblättern: " &
iCount)
Next
For iCount = 1 To ActiveDocument.TableCount-1
Wait 1000
ActiveWindow.JumpToPrevtable
ActiveCell.Insert("Und wieder zurück: " & ActiveDocument.Table-
Count-iCount)
Next
End Sub
```

TableOperationDialog-Methode

Ruft den Dialog MEHRFACHOPERATIONEN auf. Der Zellbereich muß mehr als eine Zelle umfassen.

KLASSE:
RangeSelection

SYNTAX:
TableOperationDialog()

BEISPIEL:

```
Sub DialogValidation
Documents.Add("scalc4")
Range("b2:c4").Select
Range("b2:c4").TableOperationDialog
end sub
```

Tables-Eigenschaft

Gibt ein Tables-Objekt zurück, mit dem auf die einzelnen Tabellen des Dokuments zugegriffen werden kann.

Um die Anzahl der Tabellenblätter zu ermitteln, die in dem aktiven Dokument enthalten sind, kann die Count-Eigenschaft auf die Tables-Collection angewendet werden, also "ActiveDocument.Tables.Count".

KLASSE:
TableDocument

SYNTAX:
Tables

RÜCKGABE:
Object

PARAMETER:
(nur Lesen)

Beispiel:

```
Sub ExampleTableVisible
Dim iCount As Integer
Documents.Add("sCalc4")
Wait 1000
For iCount=2 to ActiveDocument.Tables.Count
ActiveDocument.Tables(iCount).Visible= FALSE
Next iCount
End Sub
```

TableSelectAll-Methode

Selektiert alle Tabellen.

Klasse:
TableEditView

Syntax:
TableSelectAll()

Beispiel:

```
sub Main
Documents.Add "sCalc4"
ActiveWindow.TableSelectAll
end sub
```

TextValue-Eigenschaft

Ermittelt eine in der selektierten oder der angegebenen Zelle enthaltene Zeichenfolge, so wie sie in der Zelle sichtbar dargestellt wird. Diese Zeichenfolge enthält die Zahlen-Formatierung, so wie sie unter Format - Zelle... - Zahlen aufgelistet ist.

Siehe auch: CellText, Value.

Klasse:
RangeSelection

Syntax:
TextValue

Rückgabe:
String

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleValues
DIM sText As String
Documents.Add("scalc4")
Wait 500
ActiveCell.ColumnWidth = 3000
ActiveCell.Insert( "=Pi()" )
sText = ActiveDocument.Cells(1,1,1).TextValue
ActiveWindow.GoDown 1
ActiveCell.Insert( "TextValue="& sText )
sText = ActiveDocument.Cells(1,1,1).Value
ActiveWindow.GoDown 1
ActiveCell.Insert( "Value="& sText )
sText = ActiveDocument.Cells(1,1,1).Formula
ActiveWindow.GoDown 1
ActiveCell.Insert( "Formula="& sText )
End Sub
```

Thesaurus-Methode

Ruft den Thesaurus an der Cursorposition auf. Ist die aktive Tabellenzelle bei Aufruf dieser Methode leer, kommt es zu einem Laufzeitfehler.

Klasse:
TableEditView

Syntax:
Thesaurus()

Beispiel:

```
Sub ExampleSpelling
REM Text mit einen Rechtschreibfehler
Const sText = "Das ist ein Text der Korregiert wird."
Documents.Add("sCalc4")
Wait 500
Range("c3").Insert(sText)
ActiveWindow.Spelling
ActiveWindow.Thesaurus
End Sub
```

ToggleRelative-Methode

Verändert die Referenzen in einer Formelzelle. Die folgenden Kombinationen werden nacheinander eingestellt. Nach achtmaligem Aufruf steht wieder die ursprüngliche Formel in der Zelle:

$Tabelle1.$A$1

$Tabelle1.A$1

$Tabelle1.$A1

$Tabelle1.A1

Tabelle1.A1

Tabelle1.A$1

Tabelle1.$A1

Tabelle1.A1

KLASSE:
RangeSelection

SYNTAX:
ToggleRelative()

BEISPIEL:

```
Sub ExampleToggleRelative
Documents.Add("sCalc4")
Wait 500
Range("a1").Insert(1200)
ActiveWindow.JumpToNextCell
ActiveCell.Insert("=a1")
ActiveWindow.JumpToNextCell
ActiveCell.Insert("=a1")
ActiveCell.ToggleRelative
ActiveWindow.JumpToNextCell
ActiveCell.Insert("=a1")
ActiveCell.ToggleRelative
ActiveCell.ToggleRelative
ActiveWindow.JumpToNextCell
ActiveCell.Insert("=a1")
ActiveCell.ToggleRelative
ActiveCell.ToggleRelative
ActiveCell.ToggleRelative
Range("b1").INsert("Relativität")
Range("b2").Insert("REM " & Range("a2").Formula )
Range("b3").Insert("REM " & Range("a3").Formula )
Range("b4").Insert("REM " & Range("a4").Formula )
Range("b5").Insert("REM " & Range("a5").Formula )
End Sub
```

ToolProtectionDocument-Methode

Wenn das Dokument nicht geschützt ist, wird per Dialog nach einem optionalen Passwort gefragt und das Dokument wird geschützt.

Wenn das Dokument mit Passwort geschützt ist, wird das Passwort abgefragt und der Schutz aufgehoben.

Wenn das Dokument ohne Passwort geschützt ist, wird der Schutz sofort aufgehoben.

KLASSE:
RangeSelection

SYNTAX:
ToolProtectionDocument()

BEISPIEL:

```
Sub DialogProtection
Documents.Add("scalc4")
ActiveCell.ToolProtectionDocument
end sub
```

Eigenschaften und Methoden mit "U"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "U" beginnen.

Underline-Eigenschaft

Bestimmt oder ermittelt die Anzahl der Unterstriche der Schrift des selektierten Bereiches.

KLASSE:
TableFont

SYNTAX:
Underline

RÜCKGABE:
Integer

PARAMETER:
[= LineCount]: Integer-Ausdruck, der die Anzahl der Unterstriche der Schrift des selektierten Bereiches angibt. LineCount kann folgende Werte annehmen:

0: Keine Unterstreichung

1: Einfache Unterstreichung

2: Doppelte Unterstreichung

Beispiel:

```
Sub ExampleTableFont
Const sTemp1 = "Spezial"
Dim sInfo As String
Documents.Add("sCalc4")
REM Vorlage definieren
ActiveCell.StyleNewByExample( sTemp1, 2 )
With ActiveCell.Font
   sInfo = .CharSet
   .Color=RGB(255,0,0)
   sInfo = .Family
   sInfo = .FamilyName
   .Italic = 7
   .Pitch=2
   .Size = 24
   sInfo = .StyleName
   .Underline = 2
End With
REM Vorlage ändern
ActiveCell.StyleUpdateByExample( sTemp1, 2 )
Range("a1").Insert("Dieser Text ist mit Vorlage REM " & sTemp1
& "REM  geschrieben")
End Sub
```

Ungroup-Methode

Hebt eine Gruppierung der Zeilen oder Spalten des Range für eine Gliederung auf.

Klasse:
RangeSelection

Syntax:
Ungroup(RowOrCol As String)

Parameter:
RowOrCol: Zeichenfolge, die angibt, welche Gruppierung aufgehoben werden soll:

R: Gruppierung für Zeilen wird aufgehoben.

C: Gruppierung für Spalten wird aufgehoben.

Beispiel:

```
Sub ExampleOutline
Documents.Add "sCalc4"
Wait 100
```

```
ActiveDocument.Range("A2").Insert "10"
ActiveDocument.Range("A3").Insert "20"
ActiveDocument.Range("A4").Insert "=SUMME(A2:A3)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A5").Insert "100"
ActiveDocument.Range("A6").Insert "200"
ActiveDocument.Range("A7").Insert "=SUMME(A5:A6)"  REM eine
Funktion zum Gliedern
ActiveDocument.Range("A8").Insert "=SUMME(A4;A7)"  REM eine
Funktion zum Gliedern
Range("a1:a10").Select REM Der Berich muß vorher selektiert
werden.
ActiveTable.AutoOutline REM ... automatisch gliedern
ActiveWindow.Deselect
Wait 2000
Range("A2:A7").Group("R") REM Die Gesamte Summe auch noch Grup-
pieren
ActiveDocument.Range("A2").HideDetail
Wait 2000
ActiveDocument.Range("A1:A3").ShowDetail
Wait 2000
Range("A5").UnGroup("R") REM Bei zweiter Untersumme Gruppierung
entfernen.
Wait 2000
ActiveTable.ClearOutline
End Sub
```

UpdateChart-Methode

Aktualisiert das im selektierten oder angegebenen Bereich enthaltene StarChart-Diagramm.

KLASSE:
RangeSelection

SYNTAX:
UpdateChart(AllCharts As Boolean)

PARAMETER:
AllCharts: Boolescher Ausdruck, der angibt, ob die Diagramme aller Tabellen (True) oder nur die der aktiven Tabelle (False) aktualisiert werden sollen.

BEISPIEL:

```
Sub ExampleStarChart
Documents.Add("scalc4")
Wait 2000
REM Tabelle mit Daten füllen
Range("b2").Insert("New York")
Range("c2").Insert("London")
Range("d2").Insert("Hamburg")
Range("a3").Insert("Frauen")
Range("b3").Insert("1200")
```

```
Range("c3").Insert("900")
Range("d3").Insert("300")
Range("a4").Insert("Männer")
Range("b4").Insert("400")
Range("c4").Insert("200")
Range("d4").Insert("1000")
Range("e5").InsertObjectStarChart(TRUE,TRUE,FALSE,"a2:d4")
Wait 2000
REM Daten verändern
Range("a4").Insert("Jungens")
Range("b4").Insert("1300")
Range("c4").Insert("150")
Range("d4").Insert( 0.5 )
Wait 3000
Range("e5").UpdateChart(TRUE)
REM neue Daten hinzufügen
Range("b7").Insert( "Mädchen" )
Range("b8").Insert( "Babies" )
Range("c7").Insert( 23 )
Range("e8").Insert( 120 )
ActiveDocument.AddChartData( OLEObjects(1).Name , "b7:e8")
Wait 3000
REM anderen Chartbereich festlegen
Range("c6").Insert("Dubai")
Range("d6").Insert("Paris")
Range("e6").Insert("Hong Kong")
ActiveDocument.ChangeChartData( OLEObjects(1).Name , "b6:e8" ,
TRUE, TRUE)
End Sub
```

Eigenschaften und Methoden mit "V"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "V" beginnen.

Validation-Methode

Ruft den Dialog GÜLTIGKEIT auf.

KLASSE:
RangeSelection

SYNTAX:
Validation()

Beispiel:

```
Sub DialogValidation
Documents.Add("scalc4")
Range("b2:c4").Select
Range("b2:c4").Validation
end sub
```

Value-Eigenschaft

Ermittelt einen in der selektierten Zelle enthaltenen numerischen Wert.

Im Gegensatz zur TextValue-Eigenschaft wird hier nicht der in der Zelle sichtbare Wert inklusive der textuellen Formatierung, wie Prozent- oder Währungsformat, zurückgegeben, sondern der in dem Eingabefeld der Rechenleiste sichtbare Wert ohne Formatierung. Nur mit dieser Funktion können die genauen Werte für eine Berechnung aus einer Zelle ausgelesen werden. Dabei wird eine länderabhängige Zahlendarstellung (Dezimaltrenner, Tausendertrenner) korrekt berücksichtigt, so daß die ausgelesenen Zahlen für eine Weiterverarbeitung in StarBasic nicht weiter konvertiert werden müssen.

Siehe auch: TextValue

Klasse:
RangeSelection

Syntax:
Value

Rückgabe:
Double

Parameter:
(nur Lesen)

Beispiel:

```
Sub ExampleValues
DIM sText As String
Documents.Add("scalc4")
Wait 500
ActiveCell.ColumnWidth = 3000
ActiveCell.Insert( "=Pi()" )
sText = ActiveDocument.Cells(1,1,1).TextValue
ActiveWindow.GoDown 1
ActiveCell.Insert( "TextValue="& sText )
sText = ActiveDocument.Cells(1,1,1).Value
ActiveWindow.GoDown 1
ActiveCell.Insert( "Value="& sText )
sText = ActiveDocument.Cells(1,1,1).Formula
```

```
ActiveWindow.GoDown 1
ActiveCell.Insert( "Formula="& sText )
End Sub
```

VerticalAlignment-Eigenschaft

Bestimmt oder ermittelt die vertikale Ausrichtung des aktuellen Zellbereichs.

Bei leeren Zellen, die keinen Inhalt aufweisen, führt diese Eigenschaft bei einer Abfrage zu einem Laufzeitfehler, wenn die betroffenen Zellen mit der vertikalen Ausrichtungsoption "Standard" formatiert sind. Um dieses Problem zu umgehen und eine Weiterverarbeitung nicht wegen eines Laufzeitfehlers abbrechen zu lassen, kann man den Rückgabewert auf "nicht gesetzt" abfragen:

Align = Selection.VerticalAlignment
If IsNull(Align) Then Align = 0
Damit wird der Variant auf einen gültigen Wert gleichen Typs gesetzt.

Siehe auch: HorizontalAlignment, AlignTop, AlignVCenter, AlignBottom.

KLASSE:
RangeSelection

SYNTAX:
VerticalAlignment

RÜCKGABE:
Integer

PARAMETER:
[= AlignType]: Integer-Ausdruck, der die Art der Ausrichtung angibt. AlignType können die folgenden Werte zugewiesen werden:

0: Standard (nur Schreiben möglich, wenn Zellen ohne Inhalt)

1: Ausrichtung an den oberen Zellbegrenzungen

2: Zentrierte Ausrichtung

3: Ausrichtung an den unteren Zellbegrenzungen

BEISPIEL:

```
Sub ExampleCellAlignment
Const stext="Text" REM Ein bißchen Text, der die Ausrichtung
zeigt
Documents.Add("scalc4")
Wait 100
```

```
ActiveCell.Insert( sText )
ActiveCell.Copy
Range("a1:d5").Paste
Range("a1:iv256").ColumnWidth = 300
Range("a1:iv256").RowHeight = 300
Range("a1:d5").ColumnWidth = 1000
Range("a1:d5").RowHeight = 1000
REM eine kleine Tabelle mit allen Ausrichtungen
For iX=1 To 4
For iY=1 To 5
ActiveTable.Cells(iX,iY).VerticalAlignment = iX -1
ActiveTable.Cells(iX,iY).HorizontalAlignment = iY -1
Next iY
Next iX
End Sub
```

ViewInputLineOnOff-Methode

Schaltet die Eingabezeile (Rechenleiste) ein oder aus.

KLASSE:
RangeSelection

SYNTAX:
ViewInputLineOnOff()

BEISPIEL:

```
Sub ExampleToggle
Documents.Add "sCalc4"
Wait 1000
Range("b2").Insert("Das ist Text")
Range("b3").Insert( 123.456 )
Range("b4").Insert( "=b3" )
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
Wait 2000
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
End Sub
```

ViewRowColumnHeadersOnOff-Methode

Schaltet die Ansicht der Spalten- und Zeilenköpfe ein oder aus.

KLASSE:
TableEditView

SYNTAX:
ViewRowColumnHeadersOnOff()

Beispiel:

```
Sub ExampleToggle
Documents.add "sCalc4"
Wait 1000
Range("b2").Insert("Das ist Text")
Range("b3").Insert( 123.456 )
Range("b4").Insert( "=b3" )
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
Wait 2000
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
End Sub
```

ViewValueHighlightingOnOff-Methode

Schaltet den "Werte hervorheben"-Modus ein oder aus.

Klasse:
TableEditView

Syntax:
ViewValueHighlightingOnOff()

Beispiel:

```
Sub ExampleToggle
Documents.add "sCalc4"
Wait 1000
Range("b2").Insert("Das ist Text")
Range("b3").Insert( 123.456 )
Range("b4").Insert( "=b3" )
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
Wait 2000
ActiveWindow.ViewInputLineOnOff
ActiveWindow.ViewRowColumnHeadersOnOff
ActiveWindow.ViewValueHighlightingOnOff
End Sub
```

Visible-Eigenschaft

Bestimmt oder ermittelt, ob eine Tabelle eingeblendet werden soll.

KLASSE:
TableSelection

SYNTAX:
Visible

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob die Tabelle sichtbar ist (True) oder nicht (False).

BEISPIEL:

```
Sub ExampleTableVisible
Dim iCount As Integer
Documents.Add("sCalc4")
Wait 1000
For iCount=2 to ActiveDocument.Tables.Count
ActiveDocument.Tables(iCount).Visible= FALSE
Next iCount
End Sub
```

Eigenschaften und Methoden mit "W"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "W" beginnen.

WordOnly-Eigenschaft

Bestimmt oder ermittelt, ob bei der Suche nur nach Zellen gesucht werden soll, deren Inhalte dem Suchbegriff exakt entsprechen. Dies entspricht der Suchoption NUR GANZE ZELLEN im Dialog SUCHEN & ERSETZEN.

KLASSE:
TableEditView

SYNTAX:
WordOnly

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Suchbegriff exakt dem Zellinhalt entsprechen (True) oder als Bestandteil des Zellinhalts gesucht werden (False) soll.

BEISPIEL:

```
Sub ExampleSearchAll
Const sText1="Das ist ein "
Const sText1a="Das ist noch ein "
Const sText2="Test"
Const sText3="Baum"
Documents.Add "sCalc4"
ActiveWindow.InSelection = FALSE
ActiveDocument.Range("B2").Insert sText1 & sText2
ActiveDocument.Range("G5").Insert sText1a & sText2
ActiveWindow.Backward = FALSE
ActiveWindow.CaseSensitive = TRUE
ActiveWindow.WordOnly = FALSE
ActiveWindow.InSelection = FALSE REM alle suchen...
Wait 100
ActiveWindow.SearchAll sText2
ActiveWindow.SearchSettings.InSelection = True
Wait 100
REM Die erste Selektion ersetzen
ActiveWindow.Replace sText2 , sText3
Range("a1").Select
ActiveWindow.SearchSettings.InSelection = FALSE
REM Die zweite Fundstelle markieren
ActiveWindow.Search sText2
End Sub
```

WrapText-Eigenschaft

Gibt für einen Tabellenbereich an, ob ein Zeilenumbruch stattfindet.

KLASSE:
RangeSelection

SYNTAX:
WrapText

RÜCKGABE:
Boolean

PARAMETER:
[= Switch]: Boolescher Ausdruck, der angibt, ob der Zeilenumbruch ein- (True) oder ausgeschaltet (False) ist.

Beispiel:

```
Sub ExampleOptimalSize
Dim sText As String, iCount As Integer
Documents.Add("scalc4")
Wait 1000
With ActiveCell
.Insert("x4u") REM Ein X für ein U !
Wait 1000
.SetOptimalColumnWidthDirect()
Wait 1000
.SetOptimalColumnWidth( 3000 )
.AlignHCenter
sText=""
For iCount= 1 To 30
sText=sText+"Hello "
.Insert( sText )
Next
Wait 1000
.RowHeight = 2000
.WrapText = TRUE
Wait 1000
.SetOptimalRowHeight( 0 )
End With
End Sub
```

Eigenschaften und Methoden mit "Z"

Hier finden Sie alle Eigenschaften und Methoden, die mit dem Buchstaben "Z" beginnen.

Zoom-Eigenschaft

Setzt die Maßstabseinstellung für die Dokumentanzeige.

Klasse:
TableEditView

Syntax:
Zoom

Rückgabe:
Integer

Parameter:
= Value (nur Schreiben): Integer-Ausdruck, der den Maßstab angibt, in dem das Dokument dargestellt werden soll. Folgende Werte sind möglich:

-1 = Optimal (nur möglich, wenn vorher ein Druckbereich festgelegt wurde)

-2 = Ganze Seite

-3 = Seitenbreite

> 0 = Angaben in Prozent, mindestens 20 maximal 600

Beispiel:

```
Sub ExampleZoom
Dim iCount As Integer
Documents.add "sCalc4"
Wait 1000
Range("b2").Insert("Der Maßstab ist eine Sache der Ansicht.")
Range("b1").Insert( ActiveWindow.Zoom )
For iCount = 10 to 400 Step 2
Range("b1").INsert( iCount)
ActiveWindow.Zoom = iCount
Next iCount
End Sub
```

Stichwortverzeichnis

A

AboutDialog 349
Abreißleiste 263
Abs 152
ActionEnd 727
ActionStart 728
Activate 349f., 729, 1563
ActiveCell 1564
ActiveDocument 350
ActiveFrameWindow 351
ActiveHelp 352
ActiveModule 352
ActiveTable 1564
ActiveWindow 353
Add 353f., 1565
AddAllUnknownWords 729
AddBookmark 355
AddChartData 1566
AddDatabaseConnection 356
AddItem 226
AddLibrary 362
AddPrintArea 1567
Address 363, 1567
Aggregation 10
AlignBlock 1568
AlignBottom 730, 1569
AlignCharBottom 731
AlignCharTop 732
AlignHCenter 1570
AlignHorizontalCenter 733
AlignLeft 735, 1571
Alignment 737
Alignment 227
AlignRight 738, 1572
AlignRowBottom 739
AlignRowTop 740
AlignTop 740, 1573
AlignVCenter 1574
AlignVerticalCenter 742
AlignVerticalCharCenter 743
AlignVerticalRowCenter 744
AnchorOnly 744
AnchorType 745
Application 364
ArrangeHorizontal 365
ArrangeIcons 365
ArrangeVertical 366
Asc 204
Atn 144
AuditingFillMode 1575
Auto 747
Auto3D 228
AutoBackup 367
AutoBrowse 368
AutoComplete 1575
AutoCorrectDlg 369
AutoFill 1576
AutoFormat 748, 1576
AutomaticCalculation 1577
AutoOutline 1578
AutoPilotAgenda 369, 748
AutoPilotFax 369, 748
AutoPilotLetter 370, 749
AutoPilotMemo 370, 749
AutoSave 370
AutoSaveInterval 372

B

BackColor 373, 750
BackColor 229
BackgroundColor 751, 1579
BackgroundDialog 751
Backspace 752
BackStyle 230
Backup 374
Backward 375, 1580
Basic 376
BasicIDEAppear 377
BasicStop 377
Beamer 377
BeamerTaskSensitive 378

Beep 221
Blue 86
Bold 752, 1581
BookmarkCount 755
BookmarkName 756
Border 758, 1582
Border 231
BorderDialog 758
Bottom 759
BottomLine 378
BottomMargin 760
Bounds 761
Break 763
BringToFront 764
BrowseBackward 379
BrowseForward 380
BrowseHome 380
BrowserDialog 380
BrowseView 765
BulletCode 765
BulletFont 767
BulletImageURL 768
BulletsAndNumberingDialog 769

C

Calc 769
Calculate 1583
CalculateHard 1583
CalculateSel 770
Cancel 232
CancelDownloads 381
Caption 381, 771
Caption 233
Cascade 382
CaseMap 772
CaseSensitive 383, 1584
CBool 177
CDate 178
CDbl 179
CellName 774
Cells 1585f.
CellText 1587
Change 234
ChangeChartData 1588
ChangeDatabaseField 775
ChangePrintArea 1589
ChapterNumberingDialog 776
CharSet 777, 1590
CharStyle 778
CharStyleApply 778
CharStyleCount 779
ChDir 103
ChDrive 104
Choose 169
Chr 204
CInt 179
City 383
ClassName 385
Clear 236, 386, 1591
ClearArrowDependents 1592
ClearArrowPrecedents 1592
ClearArrows 1593
ClearContents 1594
ClearDiskCache 386
ClearHistory 387, 780
ClearMemCache 387
ClearOutline 1595
CLng 180
Close 388f.
Close 90
CloseDatabaseConnection 390, 781
CloseQueryConnection 392, 783
CloseTableConnection 394, 783
Closing 396
Cls 236
Color 397, 785, 1595
Column 1596
Columns 788
ColumnWidth 1598
CommonTaskBarVisible 397
Company 398
ConditionalFormatDialog 1599
Config 399
ConfigureDialog 401

ConfigureToolboxVisible 401
Const 181
Contour 788
ContourDialog 789
ContourWrap 790
ControlObject 791
Controls 792
ConvertTableText 793
Copy 794, 1599
Cos 145
Count 401ff., 795
Country 404
CountryId 406
CreateAbstract 796
CreateForm 406, 796
CreateGlossaryGroup 797
CreateIndex 408, 798
CreateLink 409
CreateNames 1600
CreateObject 182
CreateQuery 410, 800
CreateReport 410, 801
CreateTable 412, 802
CreationDate 415
CreationTime 416
Creator 417
Crosshair 805
CSng 182
CStr 183
CurDir 104
CurFieldSubType 807
CurFieldType 808
CurrentDate 418
CurrentStep 237
CurrentTime 418
CurrentX 238
CurrentY 238
CursorEndOfScreen 1600
CursorTopOfScreen 1601
CursorXPos 808
CursorYPos 809
Cut 810, 1601

D

DataAppendRecord 419, 811
DataAppendWholeRecord 421, 813
DataAreaRefresh 1602
Database 424
DataBaseBrowser 816
DataColumn 425, 817
DataColumnName 427, 819
DataColumnNumber 428, 821
DataColumnSubType 823
DataColumnType 432, 825
DataConsolidate 1602
DataDataPilotRun 1603
DataDefinedDatabase 434, 829
DataDefinedDatabaseNumber 436, 831
DataDeleteRecord 437, 833
DataFilterAutoFilter 1603
DataFilterHideAutoFilter 1604
DataFilterRemoveFilter 1604
DataFilterSpecialFilter 1605
DataFilterStandardFilter 1605
DataFirstRecord 440, 835
DataGetTableName 441, 836
DataImport 1606
DataLastRecord 443, 838
DataNextRecord 445, 840
DataPhysicalRecordID 447, 842
DataPreviousRecord 449, 844
DataRecord 451, 846
DataRecordID 453, 847
DataReImport 1606
DataSelect 849, 1607
DataSort 1608
DataSQLStatement 851
DataSubTotals 1608
DataSwitch 455, 852
DataTableNumber 456, 854
DataUpdate 856
DataUpdateFields 858
DataUpdatePrint 859
DataUpdateRecord 861

DataUpdateWholeRecord 862
DataUsedDatabase 864
DataUsedDatabaseNumber 865
Date 124
DateSerial 114
DateValue 116
Day 117
DDEExecute 126
DDEInitiate 128
DDEPoke 129
DDERequest 130
DDETerminate 131
DDETerminateAll 131
Deactivate 866, 1608
Declare 170
DecrementIndent 867, 1609
DecrementLevel 867
DecrementSubLevels 868
Default 239
DefaultBullet 869
DefaultChecked 239
DefaultFilePath 458
DefaultSelected 241
DefaultText 243
DefineDBName 1610
DefineLabelRange 1610
DefineName 1611
DefinePrintArea 1612
DefSng 189
DefStr 189
Defxxx 184ff.
Delete 870, 1612
DeleteBookmark 870
DeleteColumnbreak 1613
DeleteColumns 872, 1613
DeleteDatabaseConnection 459
DeleteForm 461, 872
DeleteFrame 462
DeleteGlossary 873
DeleteGlossaryGroup 875
DeleteIndex 464, 876
DeletePrintArea 1614
DeleteQuery 465, 877
DeleteRegion 878
DeleteReport 467, 879
DeleteRowbreak 1614
DeleteRows 880, 1615
DeleteStyle 880
DeleteTable 468, 882
DeleteTemplate 1615
DeleteUserField 884
DeleteVariable 885
DelLine 885
DelToEndOfLine 885
DelToEndOfPara 886
DelToEndOfSentence 887
DelToEndOfWord 888
DelToStartOfLine 888
DelToStartOfPara 889
DelToStartOfSentence 890
DelToStartOfWord 890
Description 471
Deselect 1616
DesignerDialog 472
Dictionary 472
Dir 105
Distance 474, 891
DistributeColumns 892
DistributeRows 892
Document 475
DocumentInfo 475
DocumentManager 477
Documents 477
DontExpand 893
DoUndo 894
DrawBox 244
DrawEllipse 249
DrawFont 895
Drawing 895
DrawLine 249
DrawLineTo 250
DrawMode 250
DrawPicture 251
DrawPoint 252

DrawPolygon 252
DrawStyle 252
DrawText 253
DrawWidth 254
Drehfeld 263
DropCap 897

E

EditGlossary 898
EditHeaderAndFooter 1616
EditLinks 1617
EditPrintArea 1617
EditRegion 898
EditStyle 899
EditTemplate 1618
EditTime 478
EMail 479
Enabled 254
End 171
EnterWait 480
EntireColumn 900
EntireRow 901
Eof 99
Erl 132
Err 133
Error 134
Escape 902
Escapement 902
Exchange 905
ExecHyperlinks 905
Exp 148
ExpandGlossary 906
Explorer 481
ExtendedHelp 481

F

Family 908, 1618
FamilyName 909, 1620
Fax 482
FaxName 911
Field 912
Field1 483
Field1Title 485
Field2 486
Field2Title 487
Field3 489
Field3Title 490
Field4 491
Field4Title 492
FieldCount 915
FieldDialog 917
FieldName 918
FieldPrompt 920
FileAttr 107
FileCopy 108
FileDateTime 109
FileDialog 494
FileDocument 495
FileLen 109
FileName 495
FileUpload 255
FillColor 496, 920
FillDown 1621
FillLeft 1621
FillRight 1622
FillSeries 1622
FillStyle 256
FillUp 1624
FirstName 497
Fix 151
Flash 921
FocusBeamer 499
FocusExplorer 500
FocusUrlBox 500
Font 923, 1624
FontColor 926
FontDialog 926
FontTransparent 258
FontWork 927
FootnoteBackground 927
FootnoteDialog 929
ForeColor 930
ForeColor 259
Format 208

FormatArea 930
FormatCellDialog 1625
FormatColumnValue 500, 931
FormatGroup 932
FormatLine 932
FormatUngroup 933
Formula 933, 1626
FrameColumns 934
FrameContent 502
FrameCount 504
FrameDialog 935
FrameDistance 936
FrameName 506, 937
Frames 938
FramesetSource 507, 938
FrameSpacing 508
FrameStyle 939
FreeFile 91
FreeLibrary 173
FreezePanes 1626
FtnAdjust 940
FtnBottomDistance 941
FtnHeight 943
FtnPenStyle 944
FtnPenWidth 945
FtnTopDistance 946
FtnWidth 948
FullName 510f.
FullScreen 511
FunctionBarVisible 512

G

GeneralOptions 512
Get 94
GetActiveDatabaseConnection 513, 949
GetActiveTableConnection 515, 951
GetAdjust 516
GetAttr 110
GetCharCount 953
GetCharStyleName 954
GetClipboardFormat 955
GetClipboardFormatCount 956
GetClipboardFormatName 957
GetDatabaseFieldValue 958
GetDatabaseLimits 960
GetDataRecordCount 518, 962
GetDecimal 520
GetFieldFmt 964
GetFieldName 965
GetFieldPar1 967
GetFieldPar2 968
GetFieldTypeCount 969
GetFieldTypeId 971
GetFieldTypeName 973
GetFillChar 521
GetFormNames 522, 974
GetFramestyleMacroLib 975
GetFramestyleMacroSub 976
GetFramestyleScript 977
GetFrameWindow 523
GetGlossaryBlockCount 979
GetGlossaryBlockName 980
GetGlossaryGroupCount 982
GetGlossaryGroupName 983
GetGlossaryShortName 984
GetGUIType 221
GetGUIVersion 222
GetHandler 986
GetHyperlinkMacroLib 987
GetHyperlinkMacroSub 988
GetHyperlinkScript 989
GetLineCount 990
GetPageStyleCount 990
GetPageStyleNameAt 991
GetParaStyleName 992
GetPosition 525
GetQueryColumns 526, 992
GetQueryNames 529, 995
GetRegionCondition 997
GetRegionCount 997
GetRegionLink 998
GetRegionLinkBookmark 999
GetRegionLinkFilter 1000
GetRegionName 1001

GetReportNames 531, 1002
GetSQL 532, 1003
GetSystemType 222
GetTableColumns 534, 1005
GetTextHeight 264
GetTextWidth 265
GetUserFieldValue 1007
GlobalContents 1008
Glossary 536
GoalSeekDialog 1627
GoDown 1010, 1627
GoDownBlock 1628
GoDownToEndOfData 1629
GoIntoFrame 1011
GoldCut 1012
GoLeft 1012, 1630
GoLeftBlock 1630
GoLeftToStartOfData 1631
GoRight 1013, 1632
GoRightBlock 1633
GoRightToEndOfData 1633
GoToAnchor 1014
GotoBookmark 1015
GotoCell 1016
GoToCell 1634
GoToCurrentCell 1635
GotoDocument 1635
GoToEnd 1017
GoToEndOfColumn 1017
GoToEndOfData 1636
GoToEndOfDoc 1018
GoToEndOfLine 1019
GoToEndOfNextColumn 1021
GoToEndOfNextPage 1022
GoToEndOfPage 1022
GoToEndOfPara 1023
GoToEndOfPrevColumn 1024
GoToEndOfPrevPage 1025
GoToEndOfRow 1636
GotoFrame 1026
GotoNextInputField 1027
GotoNextObject 1028
GoToNextPara 1029
GotoNextPlacemarker 1030
GoToNextSentence 1030
GoToNextWord 1031
GoToObject 1637
GotoPage 1032
GotoPrevInputField 1033
GotoPrevObject 1034
GoToPrevPara 1034
GotoPrevPlacemarker 1035
GoToPrevSentence 1036
GoToPrevWord 1036
GoToStart 1037, 1637
GoToStartOfColumn 1038
GoToStartOfDoc 1039
GoToStartOfLine 1040
GoToStartOfNextColumn 1041
GoToStartOfNextPage 1042
GoToStartOfPage 1043
GoToStartOfPara 1044
GoToStartOfPrevColumn 1045
GoToStartOfPrevPage 1045
GoToStartOfRow 1638
GotoTable 1046
GoUp 1047, 1639
GoUpBlock 1639
GoUpToStartOfData 1640
Graphic 1048
GraphicDialog 1050
GraphicHeight 1050
GraphicMirror 1051
GraphicName 1052
Graphics 538, 1053
GraphicsIndep 539
GraphicURL 1054
GraphicWidth 1055
Green 87
GrfFilename 540, 1055
GrfFiltername 541, 1056
GrfLinkFilterName 1057
GrfLinkName 1058
GrfPosition 541, 1059

Group 1641
GroupCount 1060
GroupObject 1061
GroupUndoOnOff 1062
Grow 1062

H

HeadlineRepeat 1065
Height 543, 1065
Height 265
HeightPercent 1066
Help 543
HelpContextID 266
HelperDialog 544
HelpIndex 544
HelpOnHelp 545
HelpText 267
HelpTip 545
Hex 154
HiddenText 1067
Hide 267, 1642
HideColumn 1643
HideDetail 1643
HideRow 1644
HideSpellMark 1069
HorizontalAlignment 1645
HorizontalLine 546
HorzOrientation 1070
HorzRelation 1071
HorzRuler 1072
HorzScrollbar 1074
Hour 120
Hyphen 1076
Hyphenate 1077
HyphenZone 1077

I

IIf 157
ImageMapDialog 1078, 1646
ImageMapURLAt 1080
ImageMapURLCount 1081
IncrementIndent 1083, 1647
IncrementLevel 1083
IncrementSubLevels 1084
IndentFirstLine 1086
Index 1086, 1648
IndexEntry 1087
IndexEntryDialog 1089
IndexMarkToIndex 1090
Input 95
InputBox 85
InsDelRowCol 1091
InSelection 548, 1648
Insert 549, 1092, 1649f.
InsertAnnotation 1093
InsertApplet 1093, 1651
InsertAuthorField 1094
InsertBookmark 1094
InsertBreak 1096
InsertCaptionDialog 1096
InsertCell 1651
InsertCellsDown 1652
InsertCellsRight 1653
InsertColumnBreak 1097, 1653
InsertColumns 1097, 1654
InsertContents 1655
InsertDateField 1098
InsertDBField 1099
InsertDocument 1101
InsertEnvelop 1103
InsertField 1103
InsertFile 1109
InsertFootnote 1110
InsertFootnoteDirect 1112
InsertFrame 1113
InsertGlossary 1114
InsertGraphic 1116, 1656
InsertGraphicRuler 1117
InsertHardHyphen 1118
InsertHyperlink 550
InsertHyperlinkDlg 1118
InsertIndexEntry 1119
InsertIndexesEntry 1121
InsertIndexesIndex 1123f.

InsertIndexesTOC 1126, 1128
InsertIndexesUserIndex 1130, 1132
InsertLabels 1135
InsertLinebreak 1135
InsertMatrix 1657
InsertMode 1136
InsertName 1657
InsertNamedTable 1136
InsertNeutralParagraph 1137
InsertNonBreakingSpace 1138
InsertNumbering 1138
InsertObject 1139, 1658
InsertObjectFloatingFrame 1140, 1658
InsertObjectStarChart 1140, 1658
InsertObjectStarImage 1141, 1660
InsertObjectStarMath 1141, 1660
InsertPagebreak 1142
InsertPageCountField 1143
InsertPageNumberField 1144
InsertPara 1144
InsertPlugin 1145, 1661
InsertRegion 1146
InsertRowBreak 1661
InsertRows 1147, 1662
InsertScript 1148
InsertSoftHyphen 1149
InsertSymbol 1149
InsertTable 1150
InsertTimeField 1151
InsertTitleField 1151
InsertTOCEntry 1152
InsertTopicField 1153
InsertUserEntry 1154
InsertVCControl 1156
InsertVCControlAnchored 1157
InStr 218
Int 152
Interactive 551
Interior 1158, 1662
InterLineSpace 1159
InternetDialog 551
InternetOnline 552
InternetUpdate 552
InterSpace 1161
IsAlignment 1161
IsArray 194
IsBold 1162
IsCaseMap 1164
IsClientImageMap 1165
IsContour 1166
IsControl 1167
IsDate 194
IsEmpty 195
IsEndOfDoc 1168
IsEndOfLine 1169
IsEndOfPara 1170
IsEndOfSentence 1171
IsEndOfWord 1172
IsError 195
IsEscapement 1173
IsFirstRecord 553, 1174
IsFont 1177
IsFontColor 1178
IsFontHeight 1179
IsHyphenZone 1180
IsItalic 1181
IsLanguage 1182
IsLastRecord 555, 1184
IsLeftRightParaMargin 1186
IsLineSpacing 1187
IsLoading 558
IsLoadingImages 559
IsNull 196
IsNumeric 196
IsObject 197
IsOrphan 1188
IsPaintLocked 1189
IsParaSplit 1190
IsPrinting 560
IsQueryNative 561, 1191
IsQueryReadOnly 563, 1193
IsRegionHidden 1195
IsRegionPassword 1196
IsRegionProtected 1197

IsServerImageMap 1197
IsShadowed 1199
IsSpacing 1200
IsStartOfDoc 1201
IsStartOfPara 1202
IsStartOfWord 1203
IsStrikeOut 1204
IsTableReadOnly 565, 1205
IsTopBottomMargin 1207
IsTopWindow 567
IsUnderline 1207
IsWait 567
IsWidow 1209
IsWordMode 1210
Italic 1211, 1663
Item 568
ItemData 268

J

JumpDownThisLevel 1214
JumpToBookmark 1215
JumpToEndOfDoc 1216
JumpToFooter 1217
JumpToFootnoteOrAnchor 1217
JumpToHeader 1219
JumpToMark 1219
JumpToNextBookmark 1220
JumpToNextCell 1221, 1664
JumpToNextFootnote 1222
JumpToNextFrame 1223
JumpToNextRegion 1224
JumpToNextSelection 1225
JumpToNextTable 1225, 1665
JumpToNextUnprotected 1665
JumpToPrevBookmark 1226
JumpToPrevCell 1227
JumpToPrevFootnote 1228
JumpToPreviousCell 1666
JumpToPreviousUnprotected 1667
JumpToPrevRegion 1229
JumpToPrevSelection 1230
JumpToPrevTable 1230, 1667
JumpToReference 1231
JumpToRegion 1232
JumpToStartOfDoc 1233
JumpToTable 1668
JumpToURL 1233
JumpUpThisLevel 1234

K

Keywords 569
Kill 111
Kontextmenü 263

L

Language 1236
LanguageId 571
LargeChange 270
LastName 571
LateInitDone 573
LaunchStarImage 1239
LBound 197
LCase 211
LeaveWait 573
Left 574, 1239
Left 211, 271
LeftLine 574
LeftMargin 1241
Len 219
Line 96
LineColor 1242
LineDash 1242
LineHeight 1243
Lines 1244
LineSpace 1245
LineText 1246
Link 272
LinkDialog 1247
LinkName 1247
LinkTarget 1248
LinkType 1249
LinkURL 1250
List 274
ListCount 275

ListIndex 276
Listing 20
Load 276, 575
LoadAccel 577
LoadConfiguration 577
LoadLibrary 578
LoadMenu 579
LoadStatusBar 580
LoadStyles 1251
LoadToolBox 580
Loc 100
Location 1252
LockPaint 1253
Lof 101
Log 148
LogicalPageNumber 1253
Longer 581, 1669
LSet 212
LTrim 213

M

MacroBarVisible 582
MacroDialog 583
MailFollowup 583
MailForward 583
MailGroupReply 584
MailMarkAsRead 584
MailNext 584
MailNextDelete 585
MailNextUnread 585
MailNotifyReset 586
MailPrev 586
MailPrevDelete 586
MailPrevUnread 587
MailReceipt 587
MailReply 587
MailSend 588
Makros aus der Vorgänger-Version 79
ManualBreak 1254
MarkMail 588
MarkThreadAsRead 588
MarkThreadAsUnread 589
Max 277
MaxHyphens 1257
Maximized 589
MaxLength 279
MergeCells 1258, 1669
MergeDialog 1258
MetricUnit 1258, 1670
Mid 214
Min 280
Minimize 590
Minimized 591
MinLead 1259
MinTrail 1260
Minute 120
MkDir 111
Mode 1671
ModificationAuthor 591
ModificationDate 593
ModificationTime 594
Modified 595
Module 596
Modules 597
Month 117
MousePointer 281
Move 1672
Move 282
MoveDown 1261
MoveDownSubItems 1262
MoveRowCol 1263
MoveUp 1264
MoveUpSubItems 1265
MsgBox 81f.
MultiLine 1266
MultiLine 282
MultiQuickSearch 598
MultiSelect 283

N

Name 599, 1673
Name 112
Navigator 600
NewField 1267

NewGlobalDoc 1267
NewGlossary 1269
NewIndex 285
NewMail 600
NewStyle 1271
NewTemplate 1674
NewWindow 601, 1272
NextFieldByType 1272
NonbreakingSpace 1273
Notes 1276
NoteText 1674
NoteVisible 1675
Now 121
NumberFormat 1278, 1676
NumberFormatCurrency 1278, 1677
NumberFormatDate 1279, 1677
NumberFormatDecDecimals 1678
NumberFormatDecimal 1279, 1679
NumberFormatIncDecimals 1680
NumberFormatPercent 1280, 1681
NumberFormatScientific 1280, 1682
NumberFormatStandard 1281, 1682
NumberFormatTime 1281, 1683
NumberLevel 1282
NumberOfPages 1283
NumberOrNoNumber 1284
NumberType 1284

O

ObjectBackOne 1286
ObjectBarVisible 602
ObjectForwardOne 1287
ObjectMirrorHorizontal 1288
ObjectMirrorVertical 1288
ObjectName 1288
Objekt-Collection 55
Objektkatalog 57
Oct 154
Offset 1684
OleObject 1289, 1685
OLEObjects 1290, 1686
On 135
OnActivate 602
OnClose 603
OnCreate 604
OnDeactivate 604
OnDocActivate 605
OnDocClose 606
OnDocCreate 607
OnDocDeactivate 608
OnDocOpen 608
OnDocPrepareClose 609
OnDocPrint 610
OnDocSave 611
OnDocSaveAs 612
OnlineAutoFormat 1291
OnlineOptions 613, 1291
OnlineRegistrationDlg 613
OnOpen 613
OnPrepareClose 614
OnPrint 615
OnQuit 616
OnSave 617
OnSaveAs 617
OnStart 618
Opaque 1292
Open 619
Open 92
OpenDatabaseConnection 621, 1293
OpenForm 622, 1295
OpenQuery 624, 1296
OpenQueryConnection 625, 1298
OpenReport 626, 1299
OpenTable 628, 1300
OpenTableConnection 629, 1302
OpenTemplate 631
Optimize 632
Option 199f.
OptionBarVisible 632
Order 1687
Organizer 633
Orphan 1304
Other 633, 1688
OutlineFont 1305

OutputHeight 286
OutputWidth 287
Outside 1307

P

PageColumnDialog 1308
PageDialog 1308
PageDown 1309
PageEnd 1310
PageFormatDialog 1689
PageHeight 1311
PageNumber 1312
PageOffset 1313
PageOffsetDialog 1314
PagePaperBin 1315
PageStyle 1315
PageStyleApply 1316
PageStyleName 1317
PageUp 1318
PageWidth 1318
PairKerning 1319
Palette 634
PaperFromSetup 1322
Paragraph 1323
ParagraphDialog 1324
ParagraphEnd 1324
ParaSplit 1326
ParaStyle 1328, 1689
ParaStyleApply 1328
ParaStyleCount 1329
Parent 287, 635
ParentWindow 636
PasswordChar 288
Paste 1330, 1690
PasteSpecial 1331, 1690
Path 636f.
PathSettings 638
Pattern 639
Phone 639
PhonePrivate 641
Pitch 1332, 1690
PlayMacro 642
PlugInsActive 643
PopupDocInfo 643
Position 644
PrevFieldByType 1333
Print 646, 1334
Print 84
Printable 1335
Printable 289
PrintAnnotations 1336
PrintBackground 1337
PrintBlackFont 1339
PrintControl 1340
PrintDefault 646, 1342
PrintDrawing 1342
Printer 647
PrintGraphic 1344
PrintLayout 1345
PrintLeftPage 1346
PrintOut 648
PrintPagePreView 1347
PrintPreview 1348, 1691
PrintProspect 1349
PrintReverse 1350
PrintRightPage 1352
PrintSettings 1353
PrintTable 1354
Prop 1356, 1692
Properties 4
PropLineHeight 1357
Protect 1358, 1692
Protected 1359
Public 200
Put 97
PutCell 1693

Q

QBColor 88
QualifierSeparator 1360
QueryAutoSave 649
Quit 650
QuoteIdentifier 1361

R

Randomize 149
Range 1694
ReadOnly 650
ReadOnly 290
Receiver 651
Recording 651
Red 87
Redo 652, 1362
RedoAction 653, 1363
Refresh 290, 1364
RefreshArrows 1695
RegExp 654, 1695
RelativeSize 1364
Relaxed 655, 1696
Reload 655
ReloadForEdit 656
Rem 174
Remove 657, 1367, 1697
RemoveItem 291
RemoveLibrary 658
RenameGlossary 1368
RenameRegion 1369
RenameUserFields 1370
Repaginate 1370
Repaint 1698
Repeat 658, 1371
RepeatAction 659, 1371
RepeatSearch 1372, 1698
Replace 1373, 1699
ReplaceAll 1375, 1699
ReplaceAttributes 1376
Reset 93, 1377
ResetAllActions 1378
ResetAllLocks 1379
ResetAttributes 1379, 1700
ResetTableProtection 1381
Restore 660
Result 136ff., 202
Resume 660
Revision 661
RGB 89
Right 1381
Right 215
RightLine 662
RightMargin 1383
RmDir 112
Rnd 150
Row 1701
RowHeight 1702
Rows 1383
RSet 215
RTrim 217

S

Save 663f.
SaveAll 665
SaveAs 666, 1384
SaveAsTemplate 667
SaveConfiguration 668
Saved 668
SaveSettings 669
SaveWorkingset 670
Scale 1704
ScEditOptions 671, 1704
ScrollBars 292
SdEditOptions 671
Search 1386, 1704
SearchAll 1387, 1705
SearchAttributes 1388
SearchCount 1389
SearchDialog 672
SearchOriginal 672
SearchSettings 1390, 1706
Second 121
see 158, 161, 164, 166, 171, 173, 175
Seek 102f.
Select 1391, 1707
Select 157
SelectAll 1391, 1707
SelectColumn 1708
Selected 292
SelectedObject 1708

Selection 673, 1709
SelectionCount 1392
SelectionMode 1392
SelectionText 674
SelectionTextExt 674
SelectRow 1710
SelectText 1393
SelectWord 1394
SelLength 293
SelStart 294
SelText 295
SendAbstractToStarImpress 1394
SendFax 1395
SendMail 676
SendOutlineToClipboard 1396
SendOutlineToStarImpress 1397
SendToBack 1398
SetActGlossaryGroup 1399
SetActions 1401
SetAlignment 1401
SetAnchorToChar 1402
SetAnchorToPage 1403
SetAnchorToPara 1403
SetAttr 113
SetBold 1404
SetCaseMap 1405
SetColumnWidth 1406
SetContour 1407
SetCurField 1408
SetDefaultBorder 1409
SetEscapement 1410
SetFixDateTimeFields 1411
SetFocus 297
SetFont 1411
SetFontColor 1413
SetFontHeight 1414
SetFrameContent 676
SetFrameName 678
SetFramestyleMacro 1415
SetFrameStyleScript 1416
SetHandler 1418
SetHyperlink 1419f., 1710
SetHyperlinkMacro 1421
SetHyperlinkScript 1422
SetHyphenZone 1423
SetInputMode 1711
SetItalic 1424
SetLanguage 1425
SetLeftRightParaMargin 1427
SetLineSpacing 1428
SetLocks 1429
SetObjectToBackground 1429
SetObjectToForeground 1430
SetOptimalColumnWidth 1431, 1712
SetOptimalColumnWidthDirect 1713
SetOptimalRowHeight 1431, 1713
SetOrphan 1432
SetPageStyleCols 1433
SetPageStyleFooter 1434
SetPageStyleHeader 1435
SetParaSplit 1436
SetRegionCondition 1437
SetRegionHidden 1438
SetRegionLink 1439
SetRegionLinkBookmark 1440
SetRegionPassword 1441
SetRegionProtected 1441
SetRowHeight 1442
SetShadowed 1443
SetSpacing 1444
SetStrikeOut 1445
SetTopBottomMargin 1446
SetUnderline 1447
SetUserFieldSubType 1447
SetUserFieldValue 1448
SetWidow 1449
SetWordMode 1450
Sgn 153
Shadow 1451
Shadowed 1452
Shell 223
Shorter 680, 1714
Show 297, 1715
ShowAddressBook 681

ShowBrowser 681
ShowColumn 1716
ShowCursor 1454
ShowDependents 1716
ShowDetail 1717
ShowErrors 1718
ShowFourPages 1455
ShowHiddenPara 1456
ShowInvalid 1719
ShowPrecedents 1719
ShowQuery 682, 1458
ShowRow 1720
ShowStatement 682, 1459
ShowTwoPages 1459
Shrink 1460
SimEditOptions 683
Similarity 683, 1720
Sin 146
Size 1462, 1721
SmallChange 297
SmEditOptions 684
SmoothScroll 1465
SoftHyphen 1467
SolidHandles 1469
Sort 1471
SortAscending 1722
SortDescending 1722
SortDialog 1473
Sorted 299
Space 1473
Space 206
Spacing 1476
Spelling 1478, 1723
SpellOnline 1478
SplitCell 1479, 1723
SplitHorizontal 685
SplitParentHorizontal 686
SplitParentVertical 688
SplitVertical 690
SplitWindow 1724
Sqr 151
StarCalc 692
StarChartDialog 1725
StarDraw 692
Start 693
StartAutoCorrect 1480
StarWriter 694
Static 201
StatusBar 694
StatusDocPos 1725
StatusLine 1480
StatusSelectionMode 1726
Step 300
Stop 175, 695
Str 205
StrComp 220
Street 695
StrikeOut 1482
String 207
Style 300, 697, 1485
StyleApply 1486, 1726
StyleApplyState 1488
StyleCatalog 699, 1489
StyleName 1490, 1728
StyleNewByExample 1490, 1729
StyleUpdateByExample 1492, 1730
StyleWatercanMode 1493, 1731
SubIni 699
Subject 700, 702
SubScript 1494
SuperScript 1496
Surround 1499
Suspend 703
SwEditOptions 704, 1500
Switch 176
SyncExplorer 704
Syntax 257

T

Tab 1500
TabIndex 302
Table 1502, 1732
TableBounds 1504
TableCount 1733

TableDialog 1506
TableModeFix 1507
TableModeFixProp 1507
TableModeVariable 1508
TableName 1509
TableNumberFormatDialog 1509
TableOperationDialog 1734
TableOptimizeMinMax 1510
Tables 1510, 1734
TableSelectAll 1735
TabStop 302
Tabstops 1511
Tag 1512
Tag 303
Tan 147
Target 304
TargetFrame 1513
TaskBarVisible 704
Template 705
Text 305
TextAttributes 1513
TextValue 1735
TextWrap 1514
Thesaurus 1736
ThesaurusDialog 1514
Tile 706
Time 125
Timer 125
TimeSerial 122
TimeValue 123
Title 707, 709f., 1515
ToggleRelative 1737
ToolBarVisible 711
ToolProtectionDocument 1738
Top 712, 1517
Top 306
TopIndex 307
TopLine 712
TopMargin 1518
TopWindow 713
TransformDialog 1519
Transparent 714, 1519
Trim 217
Type 1520
TypeName 201

U

UBound 198
UCase 218
Underline 1522, 1738
UnderlineDouble 1524
Undo 715, 1525
UndoAction 716, 1526
UndoCount 717, 1527
UndoMode 1528
Ungroup 1739
UnLoad 308
UnloadLibrary 718
UnsetCellsReadOnly 1529
UnvisitedFmt 1529
UpdateAllIndexes 1530
UpdateAllLinks 1531
UpdateChart 1740
UpdateDocstat 1533
UpdateFields 1533
UpdateInputFields 1534
URL 1535
URL 308
URLFieldCount 1535
UseMnemonic 309

V

Val 205
Validation 1741
Value 719, 1536, 1742
Value 310
ValueAndBreaks 1538
ValueAtCursor 1539
VerticalAlignment 1743
VerticalCellAlignment 1540
VerticalLine 719
VertOrientation 1540
VertRelation 1542
VertRuler 1543

VertScrollbar 1545
View 720
ViewInputLineOnOff 1744
ViewRowColumnHeadersOnOff 1744
ViewSettings 1548
ViewValueHighlightingOnOff 1745
Visible 311, 721, 1746
VisitedFmt 1550

W

Wait 224
WatercanStyleModeOnOff 1551
WeekDay 118
While 164
Widow 1552
Width 722, 1553
Width 314
WidthPercent 1554
Windows 722
With 177
WordMode 1555
WordOnly 723, 1746
WordWrap 315
Work 723
Wrap 1557
WrapText 1747
Write 98

X

XPos 1560

Y

Year 119
YPos 1560

Z

ZIPCode 725
Zoom 1561, 1748

Für Ihre Notizen:

Für Ihre Notizen:

Für Ihre Notizen:

Für Ihre Notizen: