

16 Objektreferenz

Das größte Problem im Umgang mit VBA besteht in der nahezu unüberschaubaren Zahl von Schlüsselwörtern. Dieses Referenzkapitel macht nicht den Versuch, die weit über 1000 Schlüsselwörter aufzuzählen oder gar zu beschreiben. (1000 Schlüsselwörter mal zwei Absätze ergäbe etwa 500 Seiten!) Diese Aufgabe erfüllt die Hilfe trotz mancher Unzulänglichkeiten weitaus besser als jeder gedruckte Text.

Das Kapitel gibt vielmehr einen Überblick über etwa 200 Objekte, die in der VBA-Programmierung ständig vorkommen. In alphabetischer Reihenfolge werden die wichtigsten Objekte der Excel-, ADO-, MS-Forms-, Office-, Binder-, Scripting-, VBA- und VBE-Bibliothek beschrieben. Dabei wird vor allem versucht, die Querverbindungen zwischen den Objekten deutlich zu machen und Eigenschaften und Methoden zu nennen, die diese Verbindungen herstellen.

16.1 Objekthierarchie

Dieser Abschnitt enthält einige hierarchische Listen, in denen die logische Beziehung, die VBA-Objekte zueinander haben, dargestellt ist. Die Listen beschränken sich jeweils auf Teilbereiche der Objekthierarchie. Sie erheben keinen Anspruch auf Vollständigkeit! Es ist aus Platzgründen unmöglich, eine vollständige Hierarchie *aller* Objekte darzustellen, weil die Objektliste zahllose Verästelungen aufweist, die zum Teil wieder auf ein gemeinsames Objekt führen, sich abermals teilen etc. Eine vollständige Objekthierarchie wäre viel zu lang, um dem Anspruch nach Übersichtlichkeit noch gerecht zu werden.

Im Zusammenhang mit der Objekthierarchie müssen die Begriffe *oben* und *unten* geklärt werden. Ein übergeordnetes Objekt steht in den folgenden Listen oben und in Abschnitt 16.2 links vom Ausgangsobjekt. Das in der Objekthierarchie höchste Objekt ist demnach das *Application*-Objekt, über dessen Methoden und Eigenschaften alle weiteren Objekte erreicht werden können. Dazu ein Beispiel: Dem Objekt *Workbook* sind zuerst *Workbooks* und schließlich *Application* übergeordnet, während *Sheets* und einzelne Blätter untergeordnet sind.

In den folgenden Listen sind ausschließlich Objekte genannt, nicht aber Eigenschaften und Methoden, die von einem Objekt zu einem anderen führen. Fallweise irritiert dabei der Umstand, dass es oft gleichnamige Objekte und Methoden bzw. Objekte und Eigenschaften gibt.

Innerhalb der Listen treten gelegentlich die Pfeilsymbole \Rightarrow auf. Diese Symbole zeigen an, dass an dieser Stelle eine weitere Verästelung auftritt, die etwas weiter unten separat dargestellt wird.

Excel

<i>Application</i>	Spitze der Objekthierarchie
AddIns	Add-Ins
AddIn	
Charts	nur Diagrammblätter
Chart	
CommandBars	Menü- und Symbolleisten
CommandBar	
Dialogs	vordefinierte Dialoge
Dialog	
Range	über <i>Selection</i> , <i>ActiveCell</i>
RecentFiles	zuletzt bearbeitete Dateien
RecentFile	
Sheets	Fortsetzung alle Blätter der aktiven Arbeitsmappe
Chart, Worksheet	Diagramm- und Tabellenblätter
Module, DialogSheet	Module und Dialoge in alten Excel 5/7-Dateien
SmartArtLayouts	Auflistung aller Layouts für SmartArt-Diagramme
SmartArtLayout	Layout für SmartArt-Diagramm
VBE	Verweis zur VBIDE-Bibliothek
Workbooks	alle Arbeitsmappen
Workbook	
Windows	alle Fenster (auch ausgeblendete)
Window	
WorksheetFunctions	Zugriff auf Tabellenfunktionen
Worksheets	nur Tabellenblätter
Worksheet	

Arbeitsmappe

<i>Workbook</i>	Blätter eines bestimmten Typs
Charts	
Chart	
CustomViews	Ansichten
CustomView	
Names	benannte Zellbereiche etc.
Name	
Sheets	Blätter jeden Typs der Arbeitsmappe
Chart, Worksheet	Diagramm- und Tabellenblätter
Module, DialogSheet	Module und Dialoge in alten Excel 5 /7-Dateien
Styles	Formatvorlagen
Style	
Windows	Fenster der Arbeitsmappe (auch ausgeblendete)
Window	
Worksheets	
Worksheet	

Fenster

<i>Window</i>	aktives Diagramm
Chart	
Panes	für geteilte Fenster
Pane	
Range	sichtbarer Zellbereich
PageSetup	Druckoptionen, Druckbereich
Range	über <i>ActiveCell</i> , <i>Selection</i>
SelectedSheets	Gruppe ausgewählter Blätter
Chart, Worksheet	Diagramm- und Tabellenblätter
Worksheet	aktives Blatt

Tabellenblatt

Worksheet	
— ChartObjects	eingebettete Diagramme
— Comments	Kommentare (Notizen)
— Comment	
— H/VBreaks	horizontale und vertikale Seitenumbrüche
— H/VBreak	
— HyperLinks	Querverweise/Internetlinks
— Hyperlinks	
— OLEObjects	eingebettete OLE-Objekte
— Outline	Gliederung der Tabelle
— PageSetup	Seitenlayout/Drucker
— PivotTables	Pivottabellen
⇒ PivotTable	
— Protection	Blattschutzoptionen (Excel 2002)
— AllowEditRange	passwortgeschützter Zellbereich (Excel 2002)
— UserAccess	Benutzer mit Zugriff ohne Passwort (Excel 2002)
— QueryTables	externe Daten
— QueryTable	
⇒ Range	über <i>Selection, Range, Columns, UsedRange, Cells</i> etc.
— Scenarios	Szenarios
— Scenario	
⇒ Shapes	Zeichnungs- und Steuerelemente, Objektgruppen

Zellbereich

Range	
⇒ Areas	Teilbereich bei Mehrfachauswahl
— Range	
— Borders	Umrandung
— Border	
— Characters	einzelne Zeichen des Texts
— Font	
— Comment	Kommentar (Notiz)
— Font	Schriftart
— FormatConditions	Auflistung aller bedingten Formatierungen
— FormatCondition	bedingte Formatierung
— ColorScale	Farbskalendiagramm
— Databar	Datenbalkendiagramm
— IconSetCondition	Symbolsatzdiagramm
— Interior	Hintergrundfarbe und -muster
⇒ PivotTable	Pivottabellen
— PivotField	Pivotfelder
— PivotItem	Pivotelemente
⇒ Range	Teilbereich über <i>CurrentArray, CurrentRegion, Range, SpecialCells, End, EntireRow, EntireColumn, Dependents, Precedents, Previous</i> etc.
— SparklineGroups	Auflistung aller Sparklines-Diagramme
— SparklineGroup	Sparklines-Diagramm
— Style	Formatierung des Bereichs
— Validation	Validitätsbedingungen

Pivottabelle

<i>PivotTable</i>	
— <i>PivotCache</i>	Speicher für die zugrunde liegenden Daten
— <i>PivotFields</i>	Pivotfelder bei normalen Pivottabellen
— <i>PivotField</i>	
— <i>Range</i>	Daten, Beschriftung
— <i>PivotItems</i>	Pivotelemente
— <i>PivotItem</i>	
— <i>Range</i>	Daten, Beschriftung
— <i>ColumnFields</i>	Pivotfelder im Spaltenbereich
— <i>DataFields</i>	Pivotfelder im Datenbereich
— <i>PageFields</i>	Pivotfelder im Seitenbereich
— <i>RowFields</i>	Pivotfelder im Zeilenbereich
— <i>VisibleFields</i>	sichtbare Pivotfelder
— <i>CubeFields</i>	Pivotfelder bei OLAP-Pivottabellen
— <i>PivotFormulas</i>	Berechnungsfelder (Formelfelder)
— <i>PivotFormula</i>	
— <i>TableRange1, -2 ...</i> ⇒ <i>Range</i>	Hauptdaten-, Spaltenbereich etc.

Shape-Objekte

<i>Worksheet/Chart</i>	
— <i>Shapes</i>	alle <i>Shape</i> -Objekte innerhalb des Blatts
— <i>Shape</i>	ein <i>Shape</i> -Objekt
— <i>ConnectorFormat</i>	Verbindung zu anderen Objekten
— <i>ControlFormat</i>	zusätzliche Eigenschaften für Steuerelemente
— <i>FillFormat</i>	Hintergrundmuster (via <i>Fill</i> -Eigenschaft)
— <i>GroupShapes</i>	Einzelobjekte (via <i>GroupItems</i> , wenn <i>Type=msoGroup</i>)
— <i>Shape</i>	
— <i>HyperLink</i>	Querverweise und Internetlinks
— <i>LineFormat</i>	Linienattribute (via <i>Line</i>)
— <i>LinkFormat</i>	zusätzliche Eigenschaften für OLE-Objekte
— <i>OLEFormat</i>	noch mehr Eigenschaften für OLE-Objekte
— <i>PictureFormat</i>	Eigenschaften für Bildobjekte
— <i>Range</i>	Verankerungszellen (via <i>TopLeft-/BottomRightCell</i>)
— <i>Shadow</i>	Eigenschaften für den Schatten
— <i>ShapeNodes</i>	Liniensegmente (via <i>Nodes</i> , wenn <i>Type=msoFreeform</i>)
— <i>ShapeNode</i>	
— <i>ShapeRange</i>	Einzelobjekte bei Mehrfachbearbeitung (via <i>Range</i>)
— <i>Shape</i>	
— <i>SmartArt</i>	SmartArt-Diagramm in einem Shape-Objekt
— <i>TextEffectFormat</i>	Eigenschaften für WordArt-Objekt
— <i>TextFrame</i>	Textbox innerhalb eines AutoForm-Objekts
— <i>ThreeDFormat</i>	3D-Effekte (via <i>ThreeD</i>)

Diagrammobjekte

<pre> Workbook ├── Charts │ └── Chart │ └── ChartObjects │ └── Chart Worksheet ├── ChartObjects │ └── ChartObject │ └── Chart </pre>	<p>Teil 1: Zugriff</p> <p>Hauptdiagramm im Diagrammblatt</p> <p>zusätzliche eingelagerte Diagramme</p> <p>eingelagertes Diagramm in einem Tabellenblatt</p>
<pre> Chart ├── Axes │ └── Axis │ ├── TickLabels │ ├── DisplayUnitLabel │ └── Gridlines ├── ChartArea ├── ChartTitle ├── Corners ├── DataLabels │ └── DataLabel ├── DataTable ├── Floor, Walls ├── Legend ├── PlotArea ├── SeriesCollection │ └── Series │ ├── Points │ │ └── Point │ ├── Trendlines │ │ └── Trendline │ └── ErrorBars ├── PivotLayout ├── Shapes │ └── Shape </pre>	<p>Teil 2: elementare Diagrammelemente</p> <p>Achsen</p> <p>Markierung von Achsenpunkten</p> <p>Beschriftung der Skalierungseinheit</p> <p>Orientierungslinien</p> <p>gesamtes Diagramm</p> <p>Titel</p> <p>Ecken eines 3D-Diagramms</p> <p>Beschriftung einzelner Datenpunkte</p> <p>Quelldaten innerhalb des Diagramms anzeigen</p> <p>Boden und Seitenwände eines 3D-Diagramms</p> <p>Beschriftung der Datenreihen</p> <p>nur Zeichnungsbereich</p> <p>Datenreihen</p> <p>einzelne Datenpunkte</p> <p>Trend- und Ausgleichslinien</p> <p>Kennzeichnung des Fehlerbereichs</p> <p>Steuerung der Pivotelemente bei Pivotdiagrammen</p> <p>eigenständige (Zeichnungs-)Objekte</p>
<pre> Chart ├── ChartGroups │ └── ChartGroup │ ├── SeriesCollection │ ├── DropLines │ ├── HiLoLines │ ├── SeriesLines │ ├── DownBars │ └── UpBars </pre>	<p>Teil 3: Diagrammgruppen für Verbunddiagramme</p> <p>Gruppe von Diagrammen gleichen Typs</p> <p>siehe oben</p> <p>nur Linien- und Flächendiagramm</p> <p>nur Liniendiagramm</p> <p>nur bei Balken- und Säulendiagrammen</p> <p>nur Liniendiagramm</p> <p>nur Liniendiagramm</p>

Datenbankprogrammierung (ADO-Bibliothek)

ADO-Objekthierarchie	
<pre> Connection ├── Command │ └── Parameter[s] ├── Error[s] ├── Recordset │ └── Field[s] </pre>	<p>stellt die Verbindung zur Datenbank her</p> <p>Abfragedetails (SQL-Kommando etc.)</p> <p>variable Parameter der Abfrage</p> <p>Fehlermeldungen zur letzten Datenbankoperation</p> <p>Datensatzliste (Tabellen, Abfrageergebnis etc.)</p> <p>einzelne Felder des Datensatzes</p>

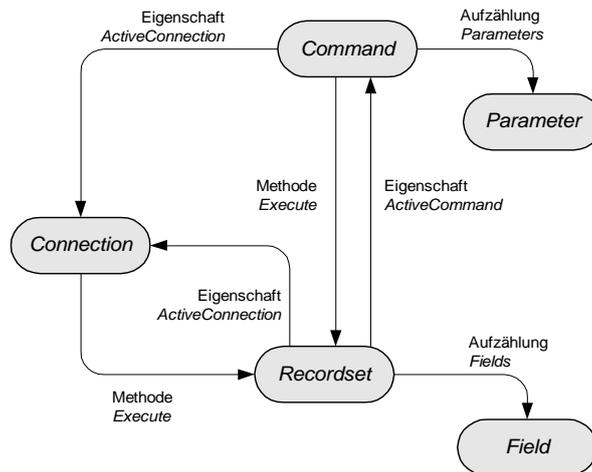


Bild 16.1: Verbindung zwischen den ADO-Objekten

Steuerung der Entwicklungsumgebung (VBIDE-Bibliothek)

VBE	Startobjekt (Zugriff via <i>Application.VBE</i>)
CodePanes[s]	Codebereich
CodeModule	Code verändern
Window	Fenster
CommandBar[s]	Menü- und Symbolleisten
VBProject[s]	Projekte (Excel-Dateien)
Reference[s]	Verweise auf Bibliotheken
VBComponent[s]	Module, Klassen, Dialoge der Datei etc.
CodeModule	Codeanteil der Komponente
Properties/Property	Zugriff auf Objekteigenschaften
Window[s]	Fenster

Benutzerdefinierte Dialoge und Steuerelemente (MS-Forms-Bibliothek)

UserForm	Dialog (Formular)
Controls	Steuerelemente im Dialog
CheckBox	Auswahlkästchen
ComboBox	Kombinationslistenfeld
CommandButton	gewöhnlicher Button
Frame	Rahmenfeld
Controls	darin enthaltene Steuerelemente
Image	Bildfeld
Label	Bezeichnungsfeld
ListBox	Listenfeld
MultiPage	mehrblättrige Dialoge
Pages	Auflistung aller Dialogblätter
Page	ein Dialogblatt
Controls	darin enthaltene Steuerelemente
OptionButton	Optionsfeld
ScrollBar	Bildlaufleiste
SpinButton	Drehfeld
TextBox	Textfeld (Ein- und Ausgabe)
ToggleButton	Umschaltbutton

Dateizugriff mit den File System Objects (Scripting-Bibliothek)

<i>FileSystemObjects</i>	Basisobjekt für alle weiteren Operationen
└─ <i>Drive[s]</i>	Laufwerke (Festplatte, CD-ROM, Diskette)
└─ <i>Folder[s]</i>	Verzeichnisse
└─ <i>File[s]</i>	Dateien
└─ <i>TextStream</i>	Bearbeitung von Textdateien (ANSI/Unicode)

Menü- und Symbolleisten (Office-Bibliothek)

<i>CommandBar[s]</i>	Symbol- und Menüleisten, Kontextmenüs
└─ <i>CommandBarControls</i>	Auflistung aller Einträge (Eigenschaft <i>Controls</i>)
└─ <i>CommandBarButton</i>	Menükommando oder Symbol
└─ <i>CommandBarComboBox</i>	Listenfeld
└─ <i>CommandBarPopup</i>	Menü, Untermenü etc.
└─ <i>CommandBarControls</i>	Auflistung aller Einträge (Eigenschaft <i>Controls</i>)
└─ ...	siehe oben

XML-Programmierung

Zwischen den XML-Objekten gibt es zahlreiche Querverweise. Daher bestehen viele Möglichkeiten zur Darstellung der Hierarchie. Hier wird als Ausgangspunkt ein *XPath*-Objekt verwendet.

<i>XPath</i>	gibt an, welche XML-Elemente in einen Zellbereich (<i>Range</i>) bzw. in eine Spalte einer Liste (<i>ListColumn</i>) importiert werden.
└─ <i>XmlMap</i>	beschreibt die Zuordnung zwischen den XML-Daten und dem Excel-Tabellenblatt.
└─ <i>XmlDataBinding</i>	beschreibt die Datenquelle der XML-Daten.
└─ <i>XmlSchema</i>	beschreibt den Aufbau (die Struktur) von XML-Daten.
└─ <i>XmlNamespace</i>	definiert das Präfix zur Identifizierung zusammengehörender XML-Elemente, die einen Namensraum bilden.

16.3 Alphabetische Referenz

In der folgenden Referenz werden alle Grundobjekte von Excel sowie die wichtigsten Objekte der Office-, MS-Forms- und der ADO-Bibliothek beschrieben. Dabei wurde versucht, die jeweils wichtigsten bzw. die charakteristischen Eigenschaften und Methoden hervorzuheben und kurz ihre Funktion zu erläutern.

Auflistungsobjekte

Bei Auflistungsobjekten werden links die gleichnamige Methode und rechts die Parameter dieser Methode genannt. Genau genommen handelt es sich rechts um die *Item*-Methode, die aber fast nie benutzt wird (d. h., statt *Axes.Item(xlValue)* schreiben Sie in der Regel die Kurzform *Axes(xlValue)*). In den meisten Fällen erfolgt die Identifizierung eines Objekts aus einer Auflistung durch die Angabe einer Indexnummer oder eines Namens (*Sheets(1)* oder *Sheets("Tabelle 1")*).

Bei den meisten Methoden, die zu Auflistungsobjekten führen, sind auch Datenfelder als Argumente erlaubt. Beispielsweise liefert die Methode *Sheets(Array(1,2,3))* ein *Sheets*-Objekt, das die Blätter 1, 2 und 3 enthält. Jetzt können durch *Select* alle drei Blätter gemeinsam markiert werden etc.

Bei fast allen Auflistungsobjekten der *Excel*-Bibliothek wird das erste Objekt mit *auflistung(1)* angesprochen (also z.B. *Sheets(1)*), das letzte mit der Nummer *Count* (also z.B. *Sheets(Sheets.Count)*). Eine Ausnahme von dieser Regel ist die *Watches*-Auflistung, bei der das erste Element den Index 0 hat (*Watches(0)*), das letzte den Index *Count-1*. Auch bei vielen externen Bibliotheken (etwa ADO) ist es üblich, dass der Index von 0 bis *Count-1* reicht.

Gleichbleibende Eigenschaften vieler Objekte

Eigenschaften, die praktisch bei allen Objekten in gleicher oder ähnlicher Form vorliegen, werden nicht jedes Mal neu beschrieben. Dazu zählen insbesondere *Count* (gibt die Anzahl der Elemente eines Auflistungsobjekts an), *Index* (gibt die Indexnummer eines Objekts innerhalb der Auflistung an), *Parent* (verweist auf das jeweils übergeordnete Objekt), *Application* (verweist auf das Objekt *Application*) und *Creator* (enthält eine Kennnummer des Programms, aus dem das Objekt stammt, in der Regel also die Kennnummer von Excel).

Defaulteigenschaften, Defaultobjekte, Kurzschreibweisen

Bei zahlreichen Eigenschaften und Methoden gilt das Objekt *Application* als Defaultobjekt. Ein Objekt muss nur dann genannt werden, wenn sich die Eigenschaft oder Methode auf ein anderes Objekt (z. B. auf ein bestimmtes *Workbook*-Objekt) bezieht.

Bei allen Objekten, die eine *Value*-Eigenschaft besitzen, gilt diese Eigenschaft als Defaulteigenschaft, die nicht explizit genannt werden muss.

Beim Zugriff auf Zellbereiche sowie auf Objekte in Tabellenblättern ist eine Kurzschreibweise in der Form *[A1]* erlaubt. Die vollständige Notation wäre *Range("A1")*. Die Kurzschreibweise mit den eckigen Klammern entspricht intern dem Aufruf der *Evaluate*- bzw. der *Item*-Methode.

Formalitäten

Damit Objekte von häufig gleichnamigen Methoden und Eigenschaften ohne weitere Hinweise unterschieden werden können, sind in diesem Abschnitt alle Objekte in fetter Schrift gesetzt.

Im linken Teil der Syntaxbox sind Methoden und Eigenschaften aufgezählt, die zum jeweils behandelten Objekt führen (Symbol ↗). Im rechten Teil der Syntaxbox sind Methoden, Eigenschaften oder Indizes aufgezählt, die vom behandelten Objekt zu anderen Objekten führen (Symbol ↘).

Im rechten oberen Eck der Syntaxboxen wird die Bibliothek angegeben, aus der die Objekte stammen. (Damit Objekte der ADO-, Binder-, Office- und MS-Forms-Bibliothek verwendet werden können, muss die jeweilige Bibliothek via [EXTRAS | VERWEISE](#) aktiviert werden.)

ADO	Ein Objekt der ActiveX-Data-Objects-Bibliothek (Zugriff auf externe Datenbanken).
Excel	Ein normales Excel-Objekt. Diese Objekte stehen in jedem VBA-Programm unter Excel ab Version 2002 zur Verfügung.
Office	Ein Objekt der Office-Bibliothek (Dokumenteigenschaften, Menü- und Symbolleisten etc.).
MSForms	Steuerelemente und andere Objekte der MS-Forms-Bibliothek (Gestaltung von Formularen).
Scripting	Ein Objekt aus der Microsoft-Scripting-Runtime-Bibliothek (betrifft vor allem die File Scripting Objects).
StdOLE	Ein Objekt aus der OLE-Automation-Bibliothek (betrifft nur <i>StdPicture</i> und <i>StdFont</i>).
VBA	Ein Objekt, das von der Programmiersprache VBA zur Verfügung gestellt wird (betrifft nur <i>Collection</i> und <i>ErrObject</i>).
VBE	Ein Objekt aus der VBA-Extension-Bibliothek (VBIDE im Objektkatalog).

Einordnung in die Objekthierarchie

In der grau unterlegten Syntaxbox jedes Objekts werden links die wichtigsten übergeordneten Objekte mit jener Eigenschaft oder Methode angeführt, die auf das beschriebene Objekt führen. Rechts werden die wichtigsten Eigenschaften oder Methoden des beschriebenen Objekts genannt, die zu weiteren untergeordneten Objekten führen. Auf diese Weise ist eine unmittelbare Einordnung des Objekts in die Objekthierarchie möglich.

Werfen Sie etwa einen Blick auf das *Axis*-Objekt: Die Methode *Axes* des übergeordneten *Chart*-Objekts führt auf das *Axis*-Objekt. Die beiden Eigenschaften *Major*- und *MinorGridlines* verweisen auf das untergeordnete Objekt *Gridlines*.

Die Querverweise im linken wie im rechten Teil der Syntaxbox sind in vielen Fällen unvollständig. Zum *Range*-Objekt führen beispielsweise Dutzende übergeordneter Methoden und Eigenschaften. Umgekehrt sind über die Eigenschaften und Methoden des *Workbook*-Objekts zahllose weitere Objekte erreichbar. Eine vollständige Querreferenz ist daher aus Platz- und Übersichtsgründen zumeist unmöglich.

AddIn Excel

Application.AddIns(.) ↗

Das Objekt **AddIn** enthält einige Daten zu den in Excel registrierten Add-Ins. Die Eigenschaft *Installed* gibt an, ob das Add-In gerade aktiviert oder deaktiviert ist. Eine Veränderung des Werts dieser Eigenschaft verändert auch den Zustand des Add-Ins. *Path*, *Name* und *FullName* geben den Dateinamen der Add-In-Datei an, *Title* den im ADD-IN-Dialog angezeigten Titel und *Comments* eine kurze dazugehörige Erklärung.

AddIns Excel

Application ↗

(*index* oder *name*) ↘ **AddIn**

Das Objekt verweist auf die Liste aller in Excel registrierten Add-Ins. Ob diese Add-Ins tatsächlich aktiviert sind, können Sie über die *Installed*-Eigenschaft des **AddIn**-Objekts feststellen. Mit der Methode *Add* können Sie neue Add-Ins installieren. Dazu müssen Sie lediglich den Dateinamen des Add-Ins angeben.

Adjustments

Excel

[Shape](#) ↗
[ShapeRange](#) ↗

Viele *Shape*-Objekte (zu denen unter anderem alle Auto-Formen gehören, siehe [ZEICHNEN](#)-Symbolleiste) besitzen gelbe Adjustierungspunkte, mit denen im interaktiven Betrieb das Aussehen des Objekts verändert werden kann (etwa die Breite eines Pfeils oder die Form der Spitze). Diese Veränderungen werden in mehreren Fließkommazahlen gespeichert, die mit *Adjustments(n)* angesprochen werden. Ob und wie viele Adjustierungsparameter zur Verfügung stehen, können Sie mit *Adjustments.Count* feststellen. Schwieriger ist es, die Bedeutung des *n*-ten Parameters für das jeweilige Objekt festzustellen – die ist nämlich nicht dokumentiert. Folgen Sie dem Rat der Hilfe, und verwenden Sie die Makroaufzeichnung!

AllowEditRange

Excel

[AllowEditRanges\(n\)](#) ↗ [.Range](#) ↘ [Range](#)
[AllowEditRanges.Add](#) ↗ [.Users](#) ↘ [UserAccessList](#)
[Worksheet.Protection.AllowEditRanges\(n\)](#) ↗

Das Objekt enthält einen Zellbereich eines Tabellenblatts und ein zugehöriges Passwort. Der Anwender kann auch bei geschützten Tabellenblättern innerhalb dieses Zellbereichs Veränderungen durchführen, wenn er das Passwort kennt. (Damit können unterschiedliche Bereiche einer Tabelle mit unterschiedlichen Passwörtern geschützt werden. Das kann praktisch sein, wenn mehrere Personen Zugriff auf die Datei haben.)

Der Zellbereich kann der *Range*-Eigenschaft entnommen werden, der Name des Bereichs der *Title*-Eigenschaft. Das Passwort kann zwar nicht unmittelbar ausgelesen, aber mit der Methode *ChangePassword* verändert werden. Optional können mit *Users.Add* Benutzer angegeben werden, die den Zellbereich auch ohne Passwort verändern dürfen.

AllowEditRanges

Excel

[Protection](#) ↗ (index) ↘ [AllowEditRange](#)

Die Auflistung verweist auf alle *AllowEditRange*-Objekte eines Tabellenblatts.

Application

Excel

[Excel](#) ↗

- [.Workbooks](#) ↘ [Workbook](#)
- [.ActiveWorkbook](#) ↘ [Workbook](#)
- [.Windows](#) ↘ [Window](#)
- [.ActiveWindow](#) ↘ [Window](#)
- [.ActiveSheet](#) ↘ [Worksheet, Chart](#)
- [.Commandbars](#) ↘ [Commandbar](#)
- [.AddIns](#) ↘ [AddIn](#)

Das Objekt *Application* steht an der Spitze aller Excel-Objekte. Über diverse Eigenschaften und Methoden gelangen Sie zum Teil direkt und zum Teil indirekt zu allen anderen in diesem Kapitel beschriebenen Objekten. In der Syntaxbox oben wurden nur einige besonders häufig benötigte Eigenschaften und Methoden aufgezählt.

Application verweist auf zahlreiche Eigenschaften, die globale Excel-Optionen steuern (beispielsweise *DisplayFormulaBar*, *Calculation*, *ScreenUpdating*, *WindowState*, *DisplayFullScreen*, *DisplayStatusBar*, *DisplayAlerts*). *Path* gibt den Pfad zur *Excel.exe* an. *Version* enthält die Nummer der aktuellen Excel-Version.

Über *Application* kann auf einige Tabellenfunktionen zugegriffen werden, sodass diese auch im VBA-Code verwendet werden können (etwa *Count*, *Index*, *Sum*, *VLookup* und *Lookup*).

Application gilt bei vielen Methoden und Eigenschaften als Defaultobjekt, sodass es nicht immer explizit genannt werden muss (etwa *ActiveSheet*, *ActiveWindow* etc.).

Wenn Sie in einem fremden Programm via ActiveX-Automation Excel steuern, müssen Sie der **Application**-Eigenschaft *Excel* voranstellen, um so anzugeben, dass sich das Kommando auf Excel bezieht. *Excel* darf auch im VBA-Code innerhalb von Excel verwendet werden, hat dort aber keine Wirkung.

Areas Excel	
<i>Range</i> ↗	<i>(index)</i> ↘ <i>Range</i>

Wenn Zellbereiche aus mehreren Teilbereichen zusammengesetzt sind, verweist das Objekt **Areas** auf die rechteckigen Teilflächen dieses Bereichs. Ob ein zusammengesetzter Bereich vorliegt, kann über die *Count*-Eigenschaft von **Areas** festgestellt werden.

AutoCorrect Excel	
<i>Application</i> ↗	

Das **AutoCorrect**-Objekt steuert die in Excel (optional) stattfindende automatische Korrektur während der Eingabe von Texten. Die Eigenschaft *ReplaceText* gibt an, ob die automatische Korrektur aktiv ist. *ReplacementList* enthält ein Datenfeld mit den zu ersetzenden Texten und den dazugehörigen Korrekturen. Mit den Methoden *AddReplacement* und *DeleteReplacement* können dieser Liste Einträge hinzugefügt bzw. aus ihr gelöscht werden. *CapitalizeNamesOfDays* gibt an, ob die Anfangsbuchstaben von Wochentagen automatisch großgeschrieben werden sollen; *TwoInitialCapitals* bestimmt, ob solche Tippfehler automatisch korrigiert werden sollen.

AutoFilter Excel	
<i>Worksheet</i> ↗	<i>.Filters</i> ↘ <i>Filter</i>

Das **AutoFilter**-Objekt ist zwar im Objektkatalog aufgezählt, aber leider nicht dokumentiert. (Gleiches gilt auch für die verwandten Objekte **Filters** und **Filter**). Insofern sind die folgenden Ausführungen mit Vorsicht zu genießen.

Mit der **AutoFilter**-Methode des **Range**-Objekts können Sie Filter für Datenbanken bzw. Listen in Tabellenblättern einrichten. Innerhalb eines Tabellenblatts kann immer nur ein Autofilter aktiv sein; dieser kann aber mehrere Filter enthalten (für jede Spalte der Datenbank einen). Das **AutoFilter**-Objekt ermöglicht den Zugriff auf die entsprechenden **Filter**-Objekte (*AutoFilter.Filters.Count* bzw. *AutoFilter.Filters(n)*). Die Eigenschaft *Range* gibt den Zellbereich der gesamten Datenbank an, die durch den Autofilter bearbeitet wird.

AutoRecover Excel	
<i>Application</i> ↗	

Das Objekt steuert, ob (Eigenschaft *Enabled*), in welchem Intervall (*Time*) und in welches Verzeichnis (*Path*) Excel Sicherheitskopien aller offenen Dateien speichert.

Axes Excel	
<i>Chart</i> ↗	<i>(typ, gruppe)</i> ↘ <i>Axis</i>

Das Auflistungsobjekt **Axes** verweist auf die Koordinatenachsen eines Diagramms (siehe **Axis**). Die Identifizierung erfolgt durch die Angabe des Typs (*xlCategory* für die x-Achse, *xlValue* für die y-Achse und *xlSeries* für die dritte Achse bei 3D-Diagrammen). Bei 2D-Diagrammen mit zwei y-Achsen muss durch den zweiten Parameter die Achsengruppe angegeben werden (*xlPrimary* oder *xlSecondary*).

Es gibt keine Methoden zum Hinzufügen oder Löschen von Achsen. Achsen werden durch die Veränderung der *AxisGroup*-Eigenschaften der Objekte *Chart* und *Series* erzeugt bzw. entfernt.

<i>Axis</i> Excel	
<i>Chart.Axes(..)</i> ↗	<i>.AxisTitle</i> ↘ <i>AxisTitle</i> <i>.MajorGridlines</i> ↘ <i>Gridlines</i> <i>.MinorGridlines</i> ↘ <i>Gridlines</i> <i>.TickLabels</i> ↘ <i>TickLabels</i>

Das *Axis*-Objekt wird in Diagrammen zur Beschreibung der Merkmale von Koordinatenachsen verwendet. Diagramme sind normalerweise mit je einer x- und y-Achse ausgestattet, 3D-Diagramme zusätzlich mit einer z-Achse. Bei Verbunddiagrammen sind auch je zwei voneinander unabhängige x- bzw. y-Achsen möglich (Zuordnung durch *AxisGroup*). Der Zugriff auf sämtliche Achsen erfolgt über die Methode *Axes*.

Die wichtigsten Eigenschaften sind *Minimum*- und *MaximumScale* (Wertebereich), *Major*- und *MinorUnit* (Abstände für Haupt- und Hilfsteilstriche sowie für Gitterlinien), *Major*- und *MinorTickMark* (zum Ein- und Ausschalten der Teilstriche) und *ScaleType* (zur Auswahl zwischen linearer und logarithmischer Skalierung).

Optische Details der Beschriftung können über die untergeordneten Objekte *AxisTitle* und *TickLabels* eingestellt werden.

<i>AxisTitle</i> Excel	
<i>Axis</i> ↗	<i>.Interior</i> ↘ <i>Interior</i> <i>.Border</i> ↘ <i>Border</i> <i>.Font</i> ↘ <i>Font</i>

Das Objekt *AxisTitle* beschreibt das Aussehen des Titels einer Koordinatenachse eines Diagramms. Der Text des Achsentitels wird über die *Caption*- oder die *Text*-Eigenschaft gesteuert, die Schriftart über das untergeordnete *Font*-Objekt. Der Ort des Titels kann über *Left* und *Top* verändert werden. Damit einer noch nicht beschrifteten Achse überhaupt ein Titel zugeordnet werden kann, muss zuerst die *HasTitle*-Eigenschaft des *Axis*-Objekts auf *True* gestellt werden.

<i>Border</i> Excel	
<i>Range.Borders(..)</i> ↗ <i>diagrammobjekt</i> ↗	

Das Objekt *Border* steuert die Umrandung bzw. die Linienform von einzelnen Zellen, Zellbereichen sowie von zahllosen Diagrammobjekten (beispielsweise *Series*, *ChartObject*, *ChartTitle*, *Oval*, *Legend* etc.). Die drei wesentlichen Eigenschaften sind *Color*, *LineStyle* (Linienform, z. B. punktiert) und *Weight* (Breite der Linien). Zur bequemen Einstellung der *Border*-Eigenschaften für einen Zellbereich steht auch die *Range*-Methode *BorderAround* zur Verfügung.

<i>Borders</i> Excel	
<i>FormatCondition</i> ↗ <i>Range</i> ↗ <i>Style</i> ↗	<i>(index)</i> ↘ <i>Border</i>

Das Auflistungsobjekt verweist auf die sechs *Border*-Objekte, die das Aussehen der Umrandungslinien eines Zellbereichs sowie zweier diagonaler Linien bestimmen. Als Index werden die *XlBordersIndex*-Konstanten angegeben.

BuiltInDocumentProperties

Excel

[Workbook](#)*(name oder index)* ↘ [DocumentProperty](#)

Das Auflistungsobjekt enthält eine Liste aller Eigenschaften der angegebenen Excel-Datei. Diese Eigenschaften dienen zur Identifizierung und Suche nach Dokumenten. Zu den zahlreichen vordefinierten Eigenschaften für Excel-Arbeitsmappen zählen unter anderem "Title", "Subject", "Author", "Last Author", "Revision Number" etc. Daneben können mit der [Add](#)-Methode auch eigene Eigenschaften definiert werden.

CalculatedFields

Excel

[PivotTable](#)*(name oder index)* ↘ [PivotField](#)

Das Auflistungsobjekt verweist auf jene Pivotfelder einer Pivottabelle, die nicht unmittelbar aus den Quelldaten stammen, sondern errechnet sind. (Wenn in den Quelldaten Spalten mit [price](#) und [quantity](#) vorgesehen sind, kann aus dem Produkt dieser Zahlen ein neues Pivotfeld [sales](#) errechnet werden. Für berechnete Pivotfelder gilt [IsCalculated=True](#). Die Eigenschaft [Formula](#) enthält die Berechnungsformel.

CalculatedItems

Excel

[PivotField](#)*(name oder index)* ↘ [PivotItem](#)

Das Auflistungsobjekt verweist auf Pivotelemente (nicht Felder), die sich aus einer Formel ergeben. Für berechnete Pivotelemente gilt wie bei Pivotfeldern [IsCalculated=True](#). Die Eigenschaft [Formula](#) enthält die Berechnungsformel.

CalculatedMember[s]

Excel

[PivotTable](#)

Das Auflistungsobjekt verweist auf Felder einer Pivottabelle, die berechnete Daten aus einer OLAP-Datenquelle enthalten. Die Eigenschaft [CalculatedMember.Formula](#) enthält die Berechnungsformel, wobei eine spezielle Syntax zur Adressierung der mehrdimensionalen Daten erlaubt ist.

CalloutFormat

Excel

[Shape](#)[ShapeRange](#)

Laut Hilfe steuert das [CalloutFormat](#)-Objekt diverse Eigenschaften bei [Shape](#)-Objekten des Typs „Legende mit Linie“. Es ist allerdings trotz mehrfacher Versuche nicht gelungen, ein [AutoForm](#)-Objekt zu erzeugen, das einen Zugriff auf [CalloutFormat](#) ohne Fehlermeldung erlaubt hätte.

CellFormat

Excel

[Application.FindFormat](#)*.Font* ↘ [Font](#)[Application.ReplaceFormat](#)

Das Objekt beschreibt das Zellformat, das beim Suchen bzw. Ersetzen von Zellen berücksichtigt wird. Zur Einstellung des Formats greifen Sie über die Eigenschaften [FindFormat](#) bzw. [ReplaceFormat](#) auf das Objekt zu. [Clear](#) löscht alle bisher geltenden Einstellungen. Damit die Formateinstellungen beim Suchen bzw. Ersetzen berücksichtigt werden, müssen Sie bei den Methoden [Find](#) bzw. [Replace](#) die optionalen Parameter [SearchFormat:=True](#) und [ReplaceFormat:=True](#) angeben.

Characters		Excel
Range ↗ objekt ↗	.Font ↘ Font	

Über die [Characters](#)-Methode können Sie auf einzelne Zeichen eines Texts in einer Zelle oder in einem Objekt (etwa [ChartTitle](#), [TextBox](#) etc.) zugreifen und deren Schriftart verändern. Die Eigenschaften [Text](#) und [Caption](#) enthalten den ausgewählten Text. An die [Characters](#)-Methode müssen die Nummer des ersten Zeichens und die Anzahl der Zeichen übergeben werden.

Chart		Excel
Workbook.Charts(..) ↗ ChartObject ↗ Workbook.ActiveChart ↗	.ChartArea ↘ ChartArea .Axes ↘ Axis .SeriesCollection ↘ Series	

Das [Chart](#)-Objekt kann sowohl ein eingebettetes Diagramm in einem Tabellenblatt als auch das Hauptdiagramm eines Diagrammblatts sein. (Es existiert kein eigener Objekttyp für Diagrammblätter.) In jedem Fall stellt [Chart](#) das Ausgangsobjekt für den Inhalt und die optische Gestaltung eines Diagramms dar. Bei Diagrammen, die in Tabellenblättern eingebettet sind, steht zwischen dem Tabellenblatt und dem Diagramm noch das [ChartObject](#), das Position und Größe des Diagramms bestimmt.

Vom [Chart](#)-Objekt verweisen über 30 Eigenschaften und Methoden auf untergeordnete Objekte, über welche die meisten inhaltlichen und formalen Details des Diagramms gesteuert werden. Die drei wichtigsten Eigenschaften/Methoden sind in der Syntaxbox oben genannt. Einen vollständigen Überblick gibt die Objekthierarchieliste zum Thema Diagrammobjekte in Abschnitt 16.1 [!!!](#).

Unmittelbar dem [Chart](#)-Objekt zugeordnet sind die Eigenschaften [Type](#) und [SubType](#), durch die der Diagrammtyp eingestellt wird. Die Methode [ChartWizard](#) erstellt ein neues Diagramm. [Elevation](#), [Rotation](#) und [Perspective](#) bestimmen die Blickrichtung auf 3D-Diagramme.

ChartArea		Excel
Chart ↗	.Interior ↘ Interior .Border ↘ Border	

Das Objekt [ChartArea](#) beschreibt die optische Gestaltung des gesamten Diagramms (inklusive Achsen, Legende, Titel etc.). Im Gegensatz zu [ChartArea](#) steht das Objekt [PlotArea](#), das nur den Hintergrund des grafischen Teils des Diagramms betrifft.

Die Gestaltung der Diagrammfläche erfolgt im Wesentlichen über die Subobjekte [Interior](#) (Farbe und Muster) und [Border](#) (Umrandung). Das Objekt erhält durch die Methoden [Copy](#), [ClearFormats](#), [ClearContents](#) und [Clear](#) eine zusätzliche Bedeutung: Diese Methoden betreffen nämlich nicht nur die Diagrammfläche, sondern das gesamte Diagramm.

ChartColorFormat		Excel
ChartFillFormat.Fore-/BackColor ↗		

Ermöglicht die Einstellung von Farbübergängen in diversen Diagrammobjekten. Ein Unterschied zu [ColorFormat](#) ist nicht erkennbar. Siehe auch [ChartFillFormat](#) und [ColorFormat](#).

ChartFillFormat

Excel

[diagrammobjekt.Fill ↗](#)[.Fore-/BackColor](#) ↘ [ChartColorFormat](#)

Mit dem Objekt können wie mit *FillFormat* (siehe dort) Hintergrundeffekte eingestellt werden. Allerdings ist *ChartFillFormat* speziell für Diagrammobjekte vorgesehen, *FillFormat* dagegen nur für *Shape* und *ShapeRange*.

ChartGroup

Excel

[Chart.ChartGroups\(..\) ↗](#)[.SeriesCollection](#) ↘ [Series](#)[Chart.XxxGroups\(..\) ↗](#)

Diagrammgruppen fassen innerhalb eines Diagramms mehrere Datenreihen mit einem gemeinsamen Diagrammtyp zusammen. Diagrammgruppen sind nur für Verbunddiagramme erforderlich. (Verbunddiagramme sind Diagramme, in denen zwei Diagrammtypen vereint sind – z. B. ein Linien- und ein Punktdiagramm oder zwei Liniendiagramme mit unterschiedlichen y-Achsen.) Die wichtigsten Eigenschaften sind *Type* und *SubType*, durch die der Diagrammtyp der Gruppe bestimmt wird.

ChartGroups

Excel

[Chart ↗](#)[\(name oder index\)](#) ↘ [ChartGroup](#)

Das Auflistungsobjekt verweist auf *ChartGroup*-Objekte, die zur Bildung von Verbunddiagrammen erforderlich sind (siehe oben). Neben *ChartGroups* existieren zahlreiche weitere Methoden, die auf spezifische Teilgruppen der *ChartGroup*-Objekte verweisen, beispielsweise *AreaGroups*, *PieGroups* oder *LineGroups*. Es gibt keine eigene Methode zur Bildung von Diagrammgruppen – diese entstehen einfach dadurch, dass die *Type*- oder *SubType*-Eigenschaft einzelner Datenreihen individuell eingestellt wird.

ChartObject

Excel

[Worksheet.ChartObjects\(..\) ↗](#)[.Chart](#) ↘ [Chart](#)[.Interior](#) ↘ [Interior](#)

Das Diagrammobjekt steht zwischen einem eingebetteten *Chart*-Objekt und dem Tabellenblatt. Es bestimmt vor allem Größe und Position des Diagramms. Diagrammobjekte sind prinzipiell auch in Diagrammblättern erlaubt, kommen dort aber selten vor.

Diagrammobjekte können mit den Methoden *Duplicate* vervielfacht und mit *Copy* in die Zwischenablage kopiert werden. Wichtig ist der Unterschied zwischen den Methoden *Select* und *Activate*: *Select* entspricht einem einfachen Mausklick, *Activate* einem doppelten Mausklick. Ein aktiviertes Diagramm muss durch die Zuweisung von *False* an *ActiveWindow.Visible* deaktiviert werden.

ChartObjects

Excel

[Worksheet ↗](#)[\(index oder name\)](#) ↘ [Chart](#)

Die Auflistung verweist auf die eingebetteten *ChartObject*-Objekte eines Tabellen-, Dialog- oder Diagrammblatts. Siehe *ChartObject*.

Charts

Excel

[Workbook ↗](#)[\(index oder name\)](#) ↘ [Chart](#)

Das Auflistungsobjekt verweist auf Diagrammblätter. Beachten Sie dabei, dass es keinen eigenen Objekttyp für Diagrammblätter gibt, weswegen *Charts* genau genommen auf das Hauptdiagramm eines Diagrammblatts verweist. Das Blatt/Diagramm wird mit der Methode *Select* aktiviert und mit *PrintOut* am Drucker ausgegeben bzw. in der

Seitenansicht dargestellt. Auf eingebettete Diagramme in Tabellenblättern müssen Sie über das Objekt *ChartObjects* zugreifen.

	ChartTitle	Excel
<i>Chart</i> ↗	<i>.Interior</i> ↘ <i>Interior</i> <i>.Font</i> ↘ <i>Font</i>	

Das Objekt beschreibt Text, Schriftart, Position und Aussehen des Titels eines Diagramms. Ob ein Diagramm überhaupt einen Titel hat, bestimmt die *Chart*-Eigenschaft *HasTitle*. Anschließend kann über die Eigenschaften/Methoden *Caption*, *Interior*, *Border* und *Font* das Aussehen des Titels eingestellt werden.

	CheckBox	MS-Forms
<i>UserForm.Controls(..)</i> ↗		

Das Objekt repräsentiert ein Auswahlkästchen (Ja- oder Nein-Entscheidung) in MS-Forms-Dialogen. Die wichtigste Eigenschaft ist *Value*, die je nach Zustand *True*, *False* oder *Null* (unbestimmt) sein kann.

	CodeModule	VBE
<i>CodePane</i> ↗ <i>VBComponent</i> ↗		

Das *CodeModule*-Objekt ermöglicht die Veränderung von Programmcode. Dazu stehen Methoden wie *InsertLines*, *DeleteLines*, *AddFromFile* etc. zur Verfügung.

	CodePane[s]	VBE
<i>VBE</i> ↗	<i>.CodeModule</i> ↘ <i>CodeModule</i> <i>.Window</i> ↘ <i>Window</i>	

Die Auflistung *CodePanels* und die daraus abgeleiteten *CodePane*-Objekte beschreiben Codebereiche in der VBA-Entwicklungsumgebung. (Hinweis: Wenn Sie VBA-Code verändern möchten, müssen Sie auf das *CodeModule*-Objekt zurückgreifen.)

	Collection	VBA
	<i>(index oder name)</i> ↘ <i>objekt</i>	

Das Objekt ermöglicht die Definition eigener Auflistungen (Auflistungsobjekte). Neue Objekte können mit *Add* hinzugefügt, vorhandene mit *Remove* entfernt werden. Die Anzahl der gespeicherten Objekte wird mit *Count* ermittelt.

	ColorFormat	Excel
<i>FillFormat.Fore-/BackColor</i> ↗ <i>LineFormat.Fore-/BackColor</i> ↗ <i>ShadowFormat.ForeColor</i> ↗ <i>ThreeDFormat.ExtrusionColor</i> ↗		

Bei einigen Objekten wird die Farbe nicht direkt als RGB-Wert, sondern über den Umweg eines *ColorFormat*-Objekts eingestellt. Die Defaulteigenschaft von *ColorFormat* lautet *RGB*, je nach *Type*-Einstellung kann die Farbe aber auch über die Eigenschaft *SchemeColor* eingestellt werden. *SchemeColor* erwartet Indexnummern für die gültige Farbpalette (deren Einstellung allerdings weder gelesen noch verändert werden kann).

	ColorScale	Excel
<i>Range.FormatConditions(..)</i> ↗		

Das Objekt verweist auf ein neues oder vorhandenes Farbskalendiagramm der Bedingten Formatierung.

ComAddIn[s]

Office

[Application](#) ↗

Application.ComAddIns verweist auf die gleichnamige Auflistung aller für Excel registrierten COM-Add-Ins. **ComAddIns** verweist auf die einzelnen **ComAddIn**-Objekte. Deren Eigenschaft **Description** enthält den Namen des COM-Add-Ins (also den Text, der im Dialog **EXTRAS | COM-ADD-INS** angezeigt wird). Die Eigenschaft **Connect** gibt an, ob das Add-In gerade aktiv ist oder nicht. Eine Veränderung von **Connect** hat dieselbe Wirkung wie die Veränderung des entsprechenden Auswahlkästchens im **COM-ADD-INS**-Dialog.

ComboBox

MS-Forms

[UserForm.Controls\(..\)](#) ↗

Das Steuerelement bietet eine Kombination aus einem ausklappbaren Listefeld mit einem Textfeld. **List(n)** ermöglicht den Zugriff auf die Liste. Über **RowSource** kann die Liste einem Zellbereich der Tabelle entnommen werden. (Wenn das Steuerelement in einem Tabellenblatt verwendet wird, muss stattdessen **ListFillRange** verwendet werden. **LinkedCell** gibt dann an, in welche Zelle das Ergebnis der Auswahl übertragen werden soll.) **Text** enthält den ausgewählten bzw. eingegebenen Text, **Value** je nach Einstellung von **BoundColumn** ebenfalls den Text oder aber die Nummer des ausgewählten Listenelements.

Command

ADO

[.ActiveConnection](#) ↘ **Connection**[.Parameters\(...\)](#) ↘ **Parameter**[.Execute](#) ↘ **Recordset**

Das Objekt ermöglicht es, SQL-Kommandos mit Parametern und sogenannte *stored procedures* (SQL-Prozeduren, die vom Datenbank-Server verwaltet werden) auszuführen. Der SQL-Code der Abfrage bzw. der Name der *stored procedure* wird in **CommandText** angegeben. Anschließend werden die Werte der Parameter eingestellt. Schließlich kann das Kommando mit der Methode **Execute** ausgeführt werden. Falls es sich bei dem Kommando um eine Abfrage handelt, liefert **Execute** als Ergebnis ein **Recordset**-Objekt.

CommandBar

Office

[Application.CommandBars\(..\)](#) ↗ [.Controls](#) ↘ **CommandBarControls**

Das Objekt beschreibt eine Menü- oder Symbolleiste. Genau genommen gibt es drei Typen (Eigenschaft **Type**): „normale“ Symbolleisten (**msoBarTypeNormal**), Menüleisten (**msoBarTypeMenuBar**) und Kontextmenüs (**msoBarTypePopup**).

CommandBarButton

Office

[CommandBar.Controls\(..\)](#) ↗

Das Objekt repräsentiert einen normalen Eintrag in einem Menü bzw. in einer Symbolleiste. Je nach Einstellung von **Style** wird das Objekt als Symbol und/oder Text dargestellt.

CommandBarComboBox

Office

[CommandBar.Controls\(..\)](#) ↗

Das Menüelement kann sowohl als Texteingabefeld als auch als Listefeld (in Symbolleisten) verwendet werden. Über den Verwendungstyp entscheidet **Style** (**msoControlDropDown**, **msoControlEdit** oder **msoControlComboBox**). Bei der Verwendung als Listefeld wird die Liste mit den Methoden **AddItem**, **RemoveItem** und **Clear** bearbeitet. In jedem

Fall kann der eingegebene Text bzw. der ausgewählte Eintrag aus *Text* entnommen werden.

CommandBarControl	Office
<i>CommandBar.Controls(..)</i> ↗	(index oder name) ↘ <i>CommandBarControl</i>

CommandBarControl dient als übergeordnetes Objekt für *CommandBarButton*, *-ComboBox* oder *-Popup*. Welcher Objekttyp nun tatsächlich gilt, können Sie mit *Type* feststellen.

Wichtige Eigenschaften sind *Caption* (der Beschriftungstext), *TooltipText* (der gelbe Infotext, falls dieser von *Caption* abweichen sollte) und *OnAction* (die aufzurufende Ereignisprozedur).

CommandBarControls	Office
<i>CommandBar</i> ↗	(index oder name) ↘ <i>CommandBarControl</i>

Die Auflistung führt zu den einzelnen Einträgen einer Symbol- oder Menüleiste bzw. eines Menüs oder Untermenüs. Formal handelt es sich bei den untergeordneten Objekten um *CommandBarControl*-Objekte. Tatsächlich erhalten Sie aber zumeist ein *CommandBarButton*, *-ComboBox* oder *-Popup*-Objekt (siehe *CommandControl*). Mit *Add* können neue Menüelemente hinzugefügt werden.

CommandBarPopup	Office
<i>CommandBar.Controls(..)</i> ↗	<i>.Controls</i> ↘ <i>CommandBarControls</i>

Dieses Objekt ist der Schlüssel zu eigenen Menüs in Menü- oder Symbolleisten, zu Untermenüs in Menüs etc. *Controls* verweist auf die untergeordneten Einträge, bei denen es sich um *CommandBarButton*, *-ComboBox* oder *-Popup*-Objekte handeln kann.

Die Möglichkeit, das Objekt in jeder Hierarchieebene neuerlich einzusetzen, macht den Objektzugriff oft recht unübersichtlich. Das Objekt eignet sich übrigens nicht für Kontextmenüs – die müssen als *CommandBar*-Objekte mit *Position=msoBarPopup* definiert werden.

CommandBars	Office
<i>Application</i> ↗	(index oder name) ↘ <i>CommandBar</i>

Das Objekt zählt alle vor- und benutzerdefinierten Symbol- und Menüleisten auf. Bei vordefinierten Symbol- und Menüleisten muss als Index der englische Name verwendet werden, also etwa *CommandBars("Worksheet Menu Bar")* für die (in Excel 2007 nicht mehr angezeigte) Menüleiste.

CommandButton	MS-Forms
<i>UserForm.Controls(..)</i> ↗	<i>.Picture</i> ↘ <i>StdPicture</i>

Die wichtigste Buttoneigenschaft lautet *Caption* für den Beschriftungstext. Optional kann im Button auch ein Bild dargestellt werden (Eigenschaften *Picture* und *PicturePosition*). Das Anklicken löst – wenig überraschend – ein *Click*-Ereignis aus.

Wenn Sie das Steuerelement in Tabellenblättern verwenden, sollten Sie *TakeFocusOnClick* auf *False* stellen. Damit vermeiden Sie, dass der Eingabefokus nach dem Anklicken des Buttons dort bleibt (was im VBA-Code zu Problemen führen kann).

Comment		Excel
Range ↗	.Next/.Previous ↘ Comment	
Range.AddComment(.) ↗	.Shape ↘ Shape	

Das Objekt speichert den Inhalt und andere Informationen zu Zellkommentaren (ehemals Notizen). Über die Methode [Text](#) kann der Kommentar gelesen und verändert werden. Die Methoden [Previous](#) und [Next](#) verweisen auf weitere Kommentare im Tabellenblatt. Neue Kommentare können mit der Methode [AddComment](#) des [Range](#)-Objekts erzeugt werden. [Range.ClearComments](#) löscht alle Kommentare im angegebenen Zellbereich.

Comments		Excel
Worksheet ↗	(index oder name) ↘ Comment	

Die Auflistung ermöglicht den Zugriff auf alle Kommentare innerhalb eines Tabellenblatts, ohne dazu alle Zellen überprüfen zu müssen.

Connection		ADO
Recordset.ActiveConnection ↗	.Execute ↘ Recordset	

Bevor Daten aus der Datenbank gelesen werden können, muss über das [Connection](#)-Objekt der Zugang hergestellt werden. Dazu wird mit der Methode [Open](#) eine Zeichenkette übergeben, die alle erforderlichen Parameter enthält (den gewünschten Datenbanktreiber, den Namen der Datenbank, eventuell den Netzwerknamen des Datenbank-Servers, eventuell Login-Name und Passwort etc.).

ConnectorFormat		Excel
Shape/ShapeRange ↗	.Begin-/EndConnectedShape ↘ Shape	

[ConnectorFormat](#) beschreibt die Verbindung zwischen zwei [Shape](#)-Objekten. Beispielsweise können zwei AutoForm-Rechtecke durch eine AutoForm-Verbindungsline verbunden werden. In diesem Fall wird die Verbindung durch das [ConnectorFormat](#)-Objekt des [Shape](#)-Objekts der Verbindungsline beschrieben.

Die wichtigsten Eigenschaften sind [Begin](#)- und [EndConnectedShape](#), die auf die beiden Objekte verweisen, die miteinander verbunden werden. Zum Aufbau bzw. zum Lösen der Verbindung stehen die Methoden [BeginConnect/EndConnect](#) bzw. [BeginDisconnect/EndDisconnect](#) (jeweils für das Start- und das Zielobjekt) zur Verfügung.

Control MS-Forms	
-------------------------	--

Das Objekt stellt gemeinsame Eigenschaften, Methoden und Ereignisse für alle MS-Forms-Steuererelemente zur Verfügung. Es wird selten direkt verwendet (höchstens zur Deklaration von Variablen oder Parametern).

ControlFormat		Excel
Shape ↗		

Wenn ein [Shape](#)-Objekt dazu verwendet wird, um ein Steuerelement (zumeist aus der MS-Forms-Bibliothek) in ein Tabellenblatt einzubetten, stellt das [ControlFormat](#)-Objekt einige Eigenschaften für das Steuerelement zur Verfügung. Das Objekt ermöglicht so die Kommunikation zwischen Tabellenblatt und Steuerelement. Zu den Eigenschaften zählen unter anderem [LinkedCell](#), [ListFillRange](#) und [PrintObject](#). Siehe auch Abschnitt [7.5!!!](#).

ControlsMS-Forms

[Frame ↗](#) [\(index oder name\) ↘ steuerelement](#)
[Page ↗](#)
[UserForm ↗](#)

Die Auflistung verweist auf alle Steuerelemente eines Rahmenfelds, einer Dialogseite oder eines ganzen Dialogs. Als einzige Eigenschaft steht *Count* zur Verfügung. Mit den Methoden *Add* und *Remove* können per Programmcode neue Steuerelemente erzeugt bzw. wieder entfernt werden.

Corners Excel

[Chart ↗](#)

Das Objekt bezeichnet die Eckpunkte des Quaders, der ein 3D-Diagramm umrahmt. Als einzig sinnvolle Methode steht *Select* zur Verfügung. Das Objekt ist zum Programmieren nicht von Bedeutung. Bei der manuellen Bearbeitung von Diagrammen können die Ecken ausgewählt und das ganze Diagramm anschließend mit der Maus verdreht werden.

CubeField

Excel

[CubeFields\(...\) ↗](#) [.TreeViewControl ↘ TreeViewControl](#)
[PivotField ↗](#)

Das Objekt beschreibt einige Merkmale von Pivotfeldern, die spezifisch für OLAP-Datenquellen gelten (z.B. *CubeFieldType=xlHierarchy* oder *xlMeasure*). Eine Menge weiterer Eigenschaften haben dieselbe Bedeutung wie bei *PivotField*, das sich auf Pivotfelder herkömmlicher Datenquellen bezieht.

CubeFields

Excel

[PivotTable ↗](#) [\(index oder name\) ↘ CubeField](#)

Die Auflistung verweist auf alle OLAP-Pivotfelder einer Pivottable und entspricht im Wesentlichen *PivotFields* für Pivotfelder herkömmlicher Datenquellen.

CustomProperty/CustomProperties

Excel

[Worksheet.CustomProperties ↗](#)

Mit der *CustomProperties*-Auflistung können Sie das Tabellenblatt mit zusätzlichen Informationen ausstatten. Jedes *CustomProperty*-Objekt besteht aus einem Namen (Eigenschaft *Name*) und einem beliebigen Objekt (*Value*).

Welchen Zweck diese in Excel 2002 eingeführte Auflistung hat, ist allerdings schleierhaft. Die lapidare Information aus der Hilfe hilft wie so oft nicht weiter: *Diese Informationen können als Metadaten für XML verwendet werden.* Tests haben auf jeden Fall ergeben, dass eigene *CustomProperty*-Objekte, die einem Excel-Tabellenblatt hinzugefügt werden, beim Speichern der Arbeitsmappe als XML-Datei *gerade nicht* gespeichert werden. Die Informationen bleiben nur erhalten, wenn die Datei im gewöhnlichen **.xls?*-Format gespeichert wird.

CustomView

Excel

[Workbook.CustomViews\(..\) ↗](#)

Seit Excel 97 können zu einer Excel-Datei mehrere Einstellungen für den Drucker und für die Anzeige von Zeilen/Spalten (ein-/ausblenden) gespeichert werden.

Über das *CustomView*-Objekt kann lediglich festgestellt werden, ob für eine bestimmte Ansicht Druckereinstellungen (*PrintSettings*) oder Zeilen-/Spalteneinstellungen (*RowColSettings*) gespeichert sind. Wie diese Einstellungen aussehen, kann nur ermittelt werden, wenn die jeweilige Ansicht durch *Show* aktiviert wird.

<i>CustomViews</i>		Excel
<i>Workbook</i> ↗	<i>(index oder name)</i> ↘ <i>CustomView</i>	

Zählt alle zu einer Arbeitsmappe gespeicherten Ansichten auf. Mit der *Add*-Methode werden die aktuellen Einstellungen für den Drucker bzw. für Zeilen- und Spaltenansichten als neue Ansicht gespeichert. Siehe auch *CustomView*.

<i>Databar</i>		Excel
<i>Range.FormatConditions(..)</i> ↗		

Das Objekt verweist auf ein neues oder vorhandenes Datenbalkendiagramm der Bedingungen Formatierung.

<i>DataLabel</i>		Excel
<i>Series.DataLabels(..)</i> ↗	<i>.Interior</i> ↘ <i>Interior</i>	
<i>Point</i> ↗	<i>.Border</i> ↘ <i>Border</i>	
<i>Trendline</i> ↗	<i>.Font</i> ↘ <i>Font</i>	

Über das Objekt kann die Beschriftung einzelner Datenpunkte einer Datenreihe eines Diagramms eingestellt werden. Bevor das Objekt *DataLabel* verändert werden kann, muss die Eigenschaft *HasDataLabels* des Objekts *Series* auf *True* gestellt werden.

Der Typ der Beschriftung (Wert, Prozent oder individueller Beschriftungstext) wird über die *Type*-Eigenschaft eingestellt. Individuelle Texte werden über *Caption* oder *Text* angegeben. Die optische Formatierung erfolgt über *Orientation*, *Interior*, *Border* und *Font*.

In vielen Fällen ist der Einsatz der Methode *ApplyDataLabels* einfacher als eine Beschriftung jedes einzelnen Datenpunkts. Damit können alle Punkte eines *Series*-Objekts oder alle Datenreihen eines *Chart*-Objekts einheitlich durch Werte, Prozente etc. beschriftet werden.

<i>DataLabels</i>		Excel
<i>Series</i> ↗	<i>(index)</i> ↘ <i>DataLabel</i>	

Das Auflistungsobjekt verweist auf *DataLabel*-Objekte einer Datenreihe. Als *index* muss die Nummer des Datenpunkts angegeben werden. Siehe *DataLabel*.

<i>DataObject</i>		MS-Forms

Das Objekt wird bei OLE-Drop-Ereignissen (*BeforeDragOver*, *BeforeDropOrPaste*) an die jeweilige Ereignisprozedur übergeben. Es ermöglicht die Auswertung von Drag&Drop-Operationen, also beispielsweise das Ablegen einer Datei aus dem Explorer in einem MS-Forms-Dialog. Das Objekt kann auch dazu verwendet werden, Daten aus der Zwischenablage zu lesen bzw. dorthin zu schreiben (Methoden *GetFromClipboard* bzw. *PutInClipboard*).

<i>DataTable</i>		Excel
<i>Chart</i> ↗	<i>.Border</i> ↘ <i>Border</i> <i>.Font</i> ↘ <i>Font</i>	

Seit Excel 97 können innerhalb eines Diagrammobjekts (üblicherweise unterhalb des eigentlichen Diagramms) auch die Quelldaten als Zahlenwerte angezeigt werden. Per Programmcode wird die Datentabelle durch *Chart.HasDataTable=True/False* aktiviert

bzw. wieder entfernt. Das Aussehen der Datentabelle kann durch die Eigenschaften von *DataTable* gesteuert werden, etwa durch *HasBorderOutline*, *ShowLegendKey* etc.

Debug VBA

Beim Objekt *Debug* handelt es sich um ein allgemeines VBA-Objekt. *Debug* verweist auf das Testfenster der Programmierumgebung. Zu *Debug* existieren nur zwei Methoden: *Print* führt Ausgaben im Testfenster durch, *Assert* unterbricht je nach Wert einer als Parameter übergebenen Boolean-Variablen die Programmausführung in der Zeile, in der die Methode auftritt.

DefaultWebOptions Excel

Application ↗

Die Eigenschaften dieses Objekts steuern die Parameter des HTML-Exports von Excel. Trotz der Endung *-s* handelt es sich nicht um ein Auflistungsobjekt. Wenn die Weboigenschaften nicht global für Excel, sondern individuell für eine Datei eingestellt werden sollen, kann dafür das *WebOptions*-Objekt verwendet werden (Zugriff via *Workbook.WebOptions*).

Dialog Excel

Application.Dialogs(..) ↗

Das Objekt dient zur internen Verwaltung der vordefinierten Excel-Dialoge. Diese Dialoge können mit der *Show*-Methode angezeigt werden. Selbst definierte Dialoge werden über das *DialogSheet*-Objekt verwaltet.

Dialogs Excel

Application ↗

(index) ↘ *Dialog*

Das Auflistungsobjekt enthält eine Liste aller vordefinierten Excel-Dialoge. Die Auswahl erfolgt durch die Indexangabe über *xlDialogNamexxx*-Konstanten.

Dictionary Scripting

(index oder name) ↘ *objekt*

Das Objekt entspricht im Wesentlichen dem VBA-Objekt *Collection*, ist aber etwas leistungsfähiger. Es ermöglicht die Definition eigener Auflistungen (Auflistungsobjekte). Neue Objekte können mit *Add* hinzugefügt, vorhandene mit *Remove* entfernt werden. Die Anzahl der gespeicherten Objekte wird mit *Count* ermittelt.

DisplayUnitLabel Excel

Axis ↗

.Font ↘ *Font*

Das Objekt beschreibt Text, Schriftart, Position und Aussehen der Beschriftung der Skalierungseinheit einer Koordinatenachse eines Diagramms. Ob die Achse überhaupt skaliert ist, bestimmt die *Axis*-Eigenschaft *HasTitle*. Wenn einer der vordefinierten Faktoren verwendet wird (z.B. *Axis.DisplayUnit = xlMillions*), enthält *Axis.DisplayUnitLabel.Text* automatisch eine passende Beschriftungszeichenkette (z.B. "Millionen"). Wenn als Skalierungsfaktor dagegen ein beliebiger anderer Faktor verwendet wird (Eigenschaft *DisplayUnitCustom*), muss *DisplayUnitLabel.Text* entsprechend eingestellt werden.

DocumentProperty Excel

Workbook.BuiltinDocumentProperties(..) ↗

Das *DocumentProperty*-Objekt beschreibt eine Eigenschaft einer Excel-Datei. Dabei gibt *Name* den Eigenschaftsnamen, *Type* seinen Typ und *Value* die aktuelle Einstellung

an. Mit den Eigenschaften *LinkSource* und *LinkToContent* kann der Wert einer eigenen (benutzerdefinierten) Eigenschaft direkt mit dem Inhalt eines Tabellenblatts verbunden werden. Dazu muss *LinkToContent* auf *True* gesetzt und *LinkSource* ein benannter Zellbereich zugewiesen werden.

<i>DownBars</i>		Excel
<i>ChartGroup</i> ↗	<i>.Interior</i> ↘ <i>Interior</i> <i>.Border</i> ↘ <i>Border</i>	

Das Objekt beschreibt das Aussehen von negativen Abweichungsbalken zwischen zwei Datenreihen eines Liniendiagramms. (Analog gibt es ein *UpBars*-Objekt, das positive Abweichungsbalken beschreibt.) Beachten Sie, dass es sich hier trotz des Plurals im Objektnamen nicht um eine Auflistung handelt (d.h., es gibt kein eigenes *DownBar*-Objekt); daher können die Abweichungsbalken nur für die ganze Datenreihe, nicht aber für einen einzelnen Datenpunkt verändert werden.

Ob in einem Diagramm überhaupt Abweichungsbalken dargestellt werden, kann durch *ChartGroup.HasUpDownBars* festgestellt werden.

Das Aussehen der Abweichungsbalken wird über die Eigenschaften *Interior* und *Border* eingestellt. Damit werden die Innenfarbe und die Umrandung beeinflusst.

<i>Drive</i> Scripting	
<i>Drives(...)</i> ↗	<i>RootFolder</i> ↘ <i>Folder</i>

Das Objekt beschreibt eine Festplatte, ein Diskettenlaufwerk, ein CD-ROM-Laufwerk etc. *DriveType* gibt den Laufwerkstyp an, *TotalSize* die Gesamtkapazität, *FreeSpace* den noch freien Speicherplatz. *RootFolder* verweist auf das Wurzelverzeichnis (über das die Dateien und alle anderen Verzeichnisse angesprochen werden können).

<i>Drives</i> Scripting	
<i>FileScriptingObject</i> ↗	<i>(name oder index)</i> ↘ <i>Drive</i>

Die Auflistung verweist auf alle Laufwerke des Computers.

<i>DropLines</i>		Excel
<i>ChartGroup</i> ↗	<i>.Border</i> ↘ <i>Border</i>	

Bei Linien- und Flächendiagrammen können zu den einzelnen Datenpunkten vertikale Bezugslinien gezeichnet werden. Dazu muss die *ChartGroup*-Eigenschaft *HasDropLines* auf *True* gestellt werden. Die Bezugslinien reichen von der x-Achse bis zum Datenpunkt. Über das Objekt *DropLines* (bzw. über dessen Subobjekt *Border*) kann dann das Aussehen der Bezugslinien eingestellt werden.

<i>ErrObject</i>		VBA
------------------	--	-----

Das Objekt enthält Informationen zum letzten Fehler und kann beispielsweise in Fehlerbehandlungsroutinen ausgewertet werden. Die beiden wichtigsten Eigenschaften sind *Number* mit der Fehlernummer und *Description* mit einer kurzen Beschreibung. Das Objekt wird selten verwendet, weil dieselben Informationen auch über die schon bisher verfügbaren Funktionen *Err* und *Error* verfügbar sind.

<i>Error/Errors</i>		ADO
<i>Connection.Errors</i> ↗		

Die Auflistung verweist auf die Fehler, die bei der letzten ADO-Datenbankoperation aufgetreten sind. (Bei einem einzigen Kommando können mehrere Fehler auftreten, die von unterschiedlichen Datenbankbibliotheken bzw. vom Datenbanksystem selbst gemeldet werden.) Der meist kryptische Fehlertext befindet sich in *Description*. Darüber

hinaus enthalten *Number* und *NativeNumber* die interne Provider-Fehlernummer sowie die ADO-Fehlernummer.

Error/Errors

Excel

Range.Errors ↗

Die Auflistung verweist auf mögliche Fehler innerhalb einer Zelle, soweit diese durch die automatische Fehlerüberprüfung festgestellt wurden. Dabei kann es sich z.B. um Zahlen handeln, die irrtümlich als Zeichenketten gespeichert sind, Daten mit nur zweistelliger Jahreszahl etc. Das Ausmaß der Fehlerüberprüfung wird durch das *ErrorCheckingOptions*-Objekt bestimmt (siehe unten).

Beachten Sie, dass die Eigenschaft *Range.Errors* nur auf eine einzelne Zelle angewendet werden darf, nicht auf einen Zellbereich. Untypisch und nicht intuitiv ist auch das Verhalten der *Errors*-Auflistung: Es gibt weder *Count*, um die Anzahl der Fehler innerhalb der Zelle festzustellen, noch kann eine Schleife mit *For-Each* gebildet werden. Stattdessen muss als Index eine der sieben *XLErrorChecks*-Konstanten verwendet werden, so wie im folgenden Muster. Wenn die Eigenschaft *Value* des so adressierten *Error*-Objekts *True* enthält, liegt ein Fehler vor. (Warum die *Error[s]*-Objekte derart umständlich realisiert wurden, weiß nur Microsoft ...)

```
Dim c As Range
Set c = [a2]
If c.Errors(xlEmptyCellReferences).Value = True Then
    MsgBox "Formel verweist auf leere Zelle"
End If
If c.Errors(xlInconsistentFormula).Value = True Then
    MsgBox "Formel ist nicht konsistent"
End If
```

ErrorBars

Excel

Series ↗

.Border ↘ *Border*

Fehlerindikatoren sind kleine vertikale oder horizontale Linien bei jedem Datenpunkt eines 2D-Diagramms, die den möglichen Fehlerbereich des Datenpunkts anzeigen. Fehlerindikatoren werden normalerweise mit der *ErrorBars*-Methode des *Series*-Objekts erzeugt. Über das *Border*-Subobjekt können Sie die optische Gestaltung der Fehlerlinien einstellen. Durch die Zuweisung von *False* an die *Series*-Eigenschaft *HasErrorBars* können die Fehlerlinien wieder entfernt werden.

ErrorCheckingOptions

Excel

Application ↗

Die verschiedenen Eigenschaften dieses Objekts (z.B. *NumberAsText* oder *TextDate*) steuern den Umfang der automatischen Fehlerüberprüfung.

Field ADO

Recordset!name ↗
Fields("name") ↗

Das Objekt ermöglicht den Zugriff auf ein einzelnes Feld des gerade aktiven Datensatzes eines *Recordset*-Objekts. Dabei enthält die Eigenschaft *Value* den Inhalt des Felds. *Name*, *Type*, *Attributes* etc. geben zusätzliche Informationen über den Typ des Felds. *Recordset!name* ist die übliche Kurzschreibweise für *Recordsets.Fields("name").Value*.

Fields ADO

Recordset ↗ *(index oder name)* ↘ *Field*

Die Auflistung verweist auf alle Datenfelder eines *Recordset*-Objekts.

File Scripting

[Files\(...\)](#) ↗ [.Drive](#) ↘ [Drive](#)
[.OpenAsStream](#) ↘ [TextStream](#)

Das Objekt beschreibt eine Datei auf der Festplatte (oder einem anderen Laufwerk). Wichtige Eigenschaften sind [Name](#) (der Dateiname), [Path](#) (Kombination aus Laufwerks-, Verzeichnis- und Dateiname), [Size](#) (Dateigröße), [Attributes](#) (Attribute, z.B. schreibgeschützt). Dateien können mit [Copy](#) kopiert, mit [Move](#) verschoben und mit [Delete](#) gelöscht werden. Textdateien können zudem als [TextStream](#)-Objekt geöffnet werden.

FileDialog

Office

[Application](#) ↗ [.Filters](#) ↘ [FileDialogFilters](#)
[.SelectedItems](#) ↘ [FileDialogSelectedItems](#)

Das Objekt ermöglicht die Auswahl einer Datei oder eines Verzeichnisses. Durch die [Show](#)-Methode wird ein entsprechender Dialog ([DATEI ÖFFNEN](#), [SPEICHERN UNTER](#)) etc. angezeigt. Vorher können diverse Eigenschaften des Dialogs eingestellt werden, z.B. der Fenstertitel ([Title](#)), die Beschriftung des OK-Buttons ([ButtonName](#)) oder die im Dialog anzuzeigenden Dateitypen ([Filters](#)). Wenn der Dialog mit [OK](#) abgeschlossen wird, liefert [Show](#) als Ergebnis [True](#). Die ausgewählten Namen können nun über die [SelectedItems](#)-Eigenschaft ermittelt werden.

FileDialogFilter

Office

[FileDialog.Filters\(n\)](#) ↗
[FileDialogFilters.Add](#) ↗

Das Objekt beschreibt einen Dateityp (z.B. [*.txt](#)) für die Dateiauswahl mit dem [FileDialog](#)-Objekt. Der Dateityp wird durch die Eigenschaften [Extension](#) (z.B. ["txt"](#)) und [Description](#) (z.B. ["Textdateien"](#)) beschrieben.

FileDialogFilters

Office

[FileDialog.Filters](#) ↗ [\(index\)](#) ↘ [FileDialogFilter](#)
[.Add](#) ↘ [FileDialogFilter](#)

Die Auflistung verweist auf alle [FileDialogFilter](#)-Objekte eines Dateiauswahldialogs. Bei den Dialogen [msoFileDialogOpen](#) und [msoFileDialogSaveAs](#) sind bereits eine Menge Filter vordefiniert, die nicht verändert werden können. Wenn Sie eigene Filter definieren möchten, müssen Sie den Dialog [msoFileDialogFilePicker](#) verwenden. Neue Filter fügen Sie dann so hinzu: [fd.Filters.Add\("Textdateien", "*.txt"\)](#).

FileDialogSelectedItems

Office

[FileDialog.SelectedItems](#) ↗

Das Objekt enthält alle ausgewählten Dateien bzw. Verzeichnisse als Zeichenketten. Die Anzahl kann mit [Count](#) festgestellt werden, der Zugriff auf die einzelnen Zeichenketten erfolgt durch [\(index\)](#), wobei das erste Element mit [index=1](#) angesprochen wird.

Files Scripting

[Folder](#) ↗ [.Drives](#) ↘ [Drive](#)

Die Auflistung verweist auf alle Dateien innerhalb eines Verzeichnisses.

FileSystemObject

Scripting

.Drives ↘ *Drive*
.GetSpecialFolder ↘ *Folder*

Das Objekt bildet das Fundament der *File System Objects* (kurz FSO), die einen objekt-orientierten Zugang auf Verzeichnisse und Dateien ermöglichen. Untergeordnete Objekte sind *Drive[s]*, *Folder[s]* und *File[s]*.

FillFormat

Excel

Shape/ShapeRange.Fill ↗ *.Fore-/BackColor* ↘ *ColorFormat*

Mit dem Objekt können für diverse Zeichnungsobjekte (*Shape*) Hintergrundeffekte eingestellt werden. Für Farbübergänge können über *Fore-* und *BackColor* zwei Farben angegeben werden; der Farbübergang wird dann durch die Eigenschaften *GradientDegree* und *GradientStyle* gesteuert. Dem Objekt kann mit den Eigenschaften *TextureName* und *TextureType* auch eine Textur (Hintergrundbitmap) zugeordnet werden. Für Diagrammobjekte steht das verwandte Objekt *ChartFillFormat* zur Verfügung.

Filter

Excel

AutoFilter.Filters(..) ↗

Die drei Eigenschaften *Operator*, *Criteria1* und *Criteria2* des *Filter*-Objekts beschreiben das Filterkriterium für eine Spalte eines *AutoFilter*-Objekts. *On* gibt an, ob der Filter aktiv ist oder nicht.

Filters

Excel

AutoFilter ↗ *(index oder name)* ↘ *Filter*

Das Auflistungsobjekt verweist auf alle Filter eines Autofilters (für jede Spalte der Datenbank einen).

Floor

Excel

Chart ↗ *.Interior* ↘ *Interior*
.Border ↘ *Border*

Floor beschreibt die Bodenfläche von 3D-Diagrammen. Die optische Formatierung erfolgt über die beiden Subobjekte *Interior* und *Border*. Die Methode *ClearFormats* stellt die Standardformatierung der Bodenfläche wieder her. Siehe auch *Walls* für die Seitenwände eines 3D-Diagramms.

Folder

Scripting

Folders(...) ↗ *.Files* ↘ *Files*
FileSystemObject.GetSpecialFolder ↗ *.SubFolders* ↘ *Folders*

Das Objekt beschreibt ein Verzeichnis der Festplatte (oder eines anderen Laufwerks). Wichtige Eigenschaften sind *Name* und *Path* (Kombination aus Laufwerks- und Verzeichnisname), *Size* (Größe aller enthaltenen Dateien), *Attributes* (Attribute, z.B. schreibgeschützt).

Folders

Scripting

Folder.SubFolders ↗ *.Drives* ↘ *Drive*

Die Auflistung verweist auf alle Unterverzeichnisse eines Verzeichnisses.

Font Excel

[Range](#) ↗
[Characters](#) ↗
[diagrammobjekt](#) ↗

Das **Font**-Objekt dient zur Einstellung der Schriftart von Zellbereichen, von einzelnen Textzeichen sowie von diversen Diagrammobjekten und Steuerelementen. Die wesentlichen Eigenschaften sind *Name* (für den Zeichensatznamen), *Size*, *Bold*, *Italic*, *Underline*, *Subscript*, *Superscript*, *Color* und *Background*.

FormatCondition Excel

[Range.FormatConditions\(..\)](#) ↗
.Borders(..) ↘ **Border**
.Font ↘ **Font**
.Interior ↘ **Interior**

Die Formatierung einer Zelle oder eines Zellbereichs kann von mehreren Bedingungen abhängig gemacht werden (**START | BEDINGTE FORMATIERUNG**). Auf diese Weise können etwa automatisch alle Werte, die einen Grenzwert überschreiten, durch fette Schrift hervorgehoben werden.

Die Eigenschaften *Formula1* und *-2* sowie *Operator* beschreiben die Bedingung, *Borders*, *Font* und *Interior* die daraus resultierende Formatierung. Mit *Delete* kann die bedingte Formatierung gelöscht werden.

FormatConditions Excel

[Range](#) ↗
(index oder name) ↘ **FormatCondition**
 ↘ **ColorScale**
 ↘ **Databar**
 ↘ **IconSetCondition**

Die Auflistung verweist auf die bedingten Formate einer Zelle bzw. eines Zellbereichs. Durch *Add* kann ein neues Format hinzugefügt werden. *Delete* löscht alle bedingten Formate.

Frame MS-Forms

[UserForm.Controls\(..\)](#) ↗
.Controls(..) ↘ **stueurelement**

Das Rahmenfeld dient zur optischen Gliederung von MS-Forms-Dialogen. Im Rahmen enthaltene Steuerelemente werden mit diesem Container-Feld verschoben und können unabhängig vom restlichen Dialog durch *Zoom* vergrößert oder verkleinert werden. *Controls* ermöglicht den Zugriff auf die Steuerelemente.

FreeFormBuilder Excel

[Shapes.BuildFreeForm\(..\)](#) ↗

Freihandformen sind aus einer beliebigen Anzahl von Linien oder Kurven zusammengesetzt und werden normalerweise im interaktiven Betrieb gezeichnet. Wenn Sie Freihandobjekte per Code erzeugen möchten, müssen Sie die Methode *BuildFreeForm* verwenden. Die Methode liefert ein **FreeFormBuilder**-Objekt zurück, das anschließend mit der Methode *AddNodes* erweitert werden kann. Die Methode *ConvertToShape* wandelt das Objekt schließlich in ein *Shape*-Objekt um.

Gridlines Excel

[Axis.MajorGridlines](#) ↗
[Axis.MinorGridlines](#) ↗
.Border ↘ **Border**

Das Objekt beschreibt die Gitternetzlinien im Hintergrund von Diagrammen. Gitternetzlinien sind jeweils einer Koordinatenachse zugeordnet, d. h. horizontale Linien der

y-Achse und vertikale Linien der x-Achse. Ob und welche Gitternetzlinien angezeigt werden, bestimmen die beiden *Axis*-Eigenschaften *HasMajor*- und *HasMinorGridlines*. Der Abstand von Hauptlinien wird durch die *Axis*-Eigenschaft *MajorUnit* bestimmt. Dazwischen werden Hilfslinien gezeichnet, und zwar im Abstand *MinorUnit*. Die optische Gestaltung von Gitternetzlinien (Farbe und Linienform) erfolgt durch das Subobjekt *Border*.

Das Objekt *Gridlines* hat nichts mit den Gitternetzlinien in Tabellenblättern zu tun. Ob und wie dieses Gitternetz angezeigt wird, ist durch die *Window*-Eigenschaften *DisplayGridlines* und *GridlineColor* bestimmt.

GroupShapes

Excel

Shape/ShapeRange.GroupItems ↗ (index oder name) ↘ *Shapes*

Wenn mehrere Objekte durch das Kontextmenükommando GRUPPIERUNG zu einer Objektgruppe zusammengefasst werden, erzeugt Excel ein neues *Shape*-Objekt, dem die bisherigen Einzelobjekte untergeordnet werden. Der Zugriff auf die Einzelobjekte erfolgt durch die *GroupShapes*-Auflistung, deren zwei wichtigste Eigenschaften *Count* und *Item* sind. Siehe auch *Shape*.

HiLoLines

Excel

ChartGroup ↗

.Border ↘ *Border*

Das Objekt beschreibt das Aussehen von Spannweitenlinien in Liniendiagrammen. Spannweitenlinien sind vertikale Linien, welche die jeweils kleinsten und größten Datenpunkte aus mehreren Datenreihen verbinden. Ob im Diagramm Spannweitenlinien angezeigt werden, wird über die *ChartGroup*-Eigenschaft *HasHiLoLines* bestimmt. Das Aussehen der Linien wird durch das Subobjekt *Border* eingestellt.

HPageBreak

Excel

Worksheet.HPageBreaks(..) ↗ *.Location* ↘ *Range*

Das Objekt bezeichnet einen horizontalen Seitenumbruch in einem Tabellenblatt.

HPageBreaks

Excel

Worksheet ↗

(index oder name) ↘ *HPageBreak*

Die Auflistung ermöglicht den Zugriff auf alle manuellen Seitenumbrüche im Tabellenblatt.

HyperLink

Excel

objekt.eigenschaft ↗
Shape ↗

.Range ↘ *Range*
.Shape ↘ *Shape*

Das Objekt beschreibt einen Querverweis auf ein Dokument. Dabei kann es sich sowohl um eine bestimmte Stelle in der aktiven Datei, um eine andere lokale Datei oder um ein Dokument im Internet handeln. Die Adresse bzw. der Dateiname wird in *Address* angegeben, die genaue Position im Dokument (beispielsweise eine Zelladresse) in *SubAddress*.

HyperLinks

Excel

Chart/Worksheet ↗
Range ↗

(index oder name) ↘ *HyperLink*

Die Auflistung ermöglicht den Zugriff auf alle Querverweise bzw. Internetlinks eines Blatts oder eines Zellbereichs.

IconSetCondition

Excel

[Range.FormatConditions\(..\)](#) ↗

Das Objekt verweist auf ein neues oder vorhandenes Symbolsatzdiagramm der Bedingten Formatierung.

Image MS-Forms[UserForm.Controls\(..\)](#) ↗[.Picture](#) ↘ [StdPicture](#)

Das Bildfeld dient zur Anzeige von Bitmaps in Dialogen. Die Eigenschaft [Picture](#) verweist auf ein [StdPicture](#)-Objekt der StdOLE-Bibliothek. Das Bild kann durch die Funktion [LoadPicture](#) aus einer Datei geladen werden. Diverse Eigenschaften von [Image](#) bieten Formatierungsoptionen (Clipping, Anpassung des Bilds an die Größe des Steuerelements etc.).

Interior Excel[Range](#) ↗
[diagrammobjekt](#) ↗

Das Objekt beschreibt Farbe und Muster des Innenbereichs (also des Hintergrunds) von zahllosen Objekten. Beinahe alle Diagrammobjekte, Formatvorlagen etc. verweisen auf das [Interior](#)-Objekt. Die wichtigsten Eigenschaften sind [Color](#), [Pattern](#) und [PatternColor](#). Farben werden durch die Zuweisung eines *RGB*-Werts eingestellt. Alternativ können die Eigenschaften [Colors](#) bzw. [PatternColorIndex](#) verwendet werden, denen ein Farbindeindexwert zwischen 1 und 56 oder die Konstanten [xlNone](#) oder [xlAutomatic](#) zugewiesen werden.

Label MS-Forms[UserForm.Controls\(..\)](#) ↗[.Font](#) ↘ [NewFont](#)

Das Labelfeld (Bezeichnungsfeld) dient zur Beschriftung anderer Steuerelemente in MS-Forms-Dialogen. Der Text wird mit [Caption](#) eingestellt.

LeaderLines

Excel

[Series](#) ↗[.Border](#) ↘ [Border](#)

Das Objekt ermöglicht bei einigen Diagrammtypen (etwa Kreisdiagrammen) die Formatierung von Linien, die zwischen Diagrammelementen und Beschriftungstext gezeichnet werden. Die Linien werden nur angezeigt, wenn die Eigenschaft [HasLeaderLines](#) von [Series](#) auf [True](#) gesetzt wird.

Legend Excel[Chart](#) ↗[LegendEntries](#) ↘ [LegendEntry](#)
[Border](#) ↘ [Border](#)

Das [Legend](#)-Objekt beschreibt die Legende eines Diagramms. (Die Legende ist ein rechteckiges Kästchen, in dem die Linien, Farben oder Muster des Diagramms aufgeschlüsselt und beschriftet sind.)

Über die Eigenschaften des Objekts werden Position, Größe und Format bestimmt ([Left](#), [Top](#), [Width](#), [Height](#), [Interior](#), [Border](#), [Font](#), [Shadow](#)). Ob ein Diagramm überhaupt mit einer Legende ausgestattet ist, bestimmt die [Chart](#)-Eigenschaft [HasLegend](#). Die eigentlichen Details der Legende, d. h. die Muster und deren Beschriftung, werden über das Subobjekt [LegendEntry](#) eingestellt.

LegendEntries		Excel
Legend ↗	(index) ↘ LegendEntry	

Das Auflistungsobjekt verweist auf die Einträge in der Legende eines Diagramms. Es besteht keine Möglichkeit, die Anzahl der Einträge durch [Add](#) zu vergrößern; jeder Datenreihe und Trendlinie ist ein Legendeneintrag fest zugeordnet.

LegendEntry		Excel
Legend.LegendEntries(..) ↗	.Font ↘ Font .LegendKey ↘ LegendKey	

Das Objekt beschreibt einen einzelnen Eintrag innerhalb der Legende eines Diagramms. Das Objekt hat relativ wenige eigene Eigenschaften bzw. Methoden: [Font](#) für den Beschriftungstext, [LegendKey](#) für die optische Darstellung der Datenreihe (siehe etwas weiter unten) und [Delete](#) zum Löschen des ganzen Eintrags. (Der Eintrag kann allerdings nicht wiederhergestellt oder neu eingefügt werden. Sie müssen die ganze Legende mit [Chart.HasLegend=False](#) löschen und mit [.=True](#) wieder neu erzeugen, und zwar mit allen Legendeneinträgen.)

Der Beschriftungstext des Legendeneintrags wird durch die [Name](#)-Eigenschaft des [Series](#)-Objekts bestimmt. Die Anordnung der Legendeneinträge innerhalb der Legende wird von Excel automatisch vorgenommen (je nachdem, wie viel Platz zur Verfügung steht).

LegendKey		Excel
LegendEntry ↗	.Border ↘ Border	

Das Objekt bestimmt die optische Darstellung einer Datenreihe innerhalb der Legende (siehe [Legend](#)-Objekt). Eine Veränderung der Formatierung verändert auch die Formatierung der zugeordneten Datenreihe im Diagramm (und umgekehrt). Die wichtigsten Eigenschaften lauten [MarkerStyle](#), [MarkerBackground](#)- und [MarkerForegroundColor](#) sowie [Border](#).

LineFormat		Excel
Shape/ShapeRange.Line ↗	.Fore-/BackColor ↘ ColorFormat	

Das Objekt beschreibt das Aussehen von Linien und Pfeilen, die durch AutoForm-Objekte (siehe [Shape](#)) dargestellt werden. Zu den wichtigsten Eigenschaften zählen [Fore](#)- und [BackColor](#), [DashStyle](#) und [Weight](#). Pfeileigenschaften werden durch [BeginArrowheadLength](#), [-Style](#) und [-Width](#) bzw. [EndArrowheadLength](#), [-Style](#) und [-Width](#) eingestellt.

LinkFormat		Excel
Shape ↗		

Das Objekt enthält im Wesentlichen die Eigenschaft [AutoUpdate](#), die bei verknüpften OLE-Objekten angibt, ob das Objekt bei einer Veränderung der Quelldaten automatisch aktualisiert wird. Die Methode [Update](#) führt diese Aktualisierung manuell durch.

ListBox MS-Forms	
UserForm.Controls(..) ↗	

Das Listenfeld ermöglicht die bequeme Auswahl eines oder mehrerer Einträge aus einer Liste. Der Zugriff auf die Listenelemente erfolgt über [List](#). Über [RowSource](#) kann die Liste einem Zellbereich der Tabelle entnommen werden. (Wenn das Steuerelement in einem Tabellenblatt verwendet wird, muss stattdessen [ListFillRange](#) verwendet werden. [LinkedCell](#) gibt dann an, in welche Zelle das Ergebnis der Auswahl übertragen werden soll.) [Text](#) enthält den ausgewählten bzw. eingegebenen Text, [Value](#) je nach Einstellung

von *BoundColumn* ebenfalls den Text oder aber die Nummer des ausgewählten Listenelements.

<i>ListColumn</i>		Excel
<i>ListColumns</i> (..) ↗	.DataFormat ↘ <i>ListDataFormat</i> .Range ↘ <i>Range</i> .XPath ↘ <i>XPath</i>	

Das Objekt beschreibt eine Spalte in einer Liste. Mit der Eigenschaft *TotalsCalculation* kann angegeben werden, mit welcher Funktion das Ergebnisfeld der Spalte berechnet werden soll (z.B. *xlTotalsCalculationSum*). *Range* verweist auf den Zellbereich die Spalte.

Wenn die Daten der Liste aus einer XML-Datenquelle stammen, stellt das *XPath*-Objekt die Verbindung zu den XML-Daten her und gibt an, welches XML-Element importiert wurde bzw. später wieder exportiert werden soll.

<i>ListColumns</i>		Excel
<i>ListObject</i> ↗	(index oder name) ↘ <i>ListColumn</i>	

Die Auflistung verweist auf die Spalten einer Liste. Mit *Add* kann an einer vorgegebenen Position eine neue Spalte eingefügt werden.

<i>ListDataFormat</i>		Excel
<i>ListColumn.DataFormat</i> ↗		

Das Objekt beschreibt die Formatierungseigenschaften einer Spalte einer Liste. Diese Eigenschaften sind unveränderlich vorgegeben (d.h., alle Eigenschaften des *ListDataFormat* sind read-only). Aus der Dokumentation geht leider nicht hervor, woher diese Eigenschaften stammen. Bei meinen Tests waren sämtliche Eigenschaften von *ListDataFormat* generell nicht belegt, unabhängig davon, welche Datenquelle der Liste zugrunde lag.

<i>ListObject</i>		Excel
<i>ListObjects</i> (..) ↗ <i>Range</i> ↗ <i>QueryTable</i> ↗	.ListColumns(n) ↘ <i>ListColumn</i> .ListRow(n) ↘ <i>ListRow</i> .Range, .TotalsRowRange ↘ <i>Range</i> .XmlMap ↘ <i>XmlMap</i>	

Das Objekt beschreibt eine Liste in einer Excel-Tabelle. Als Liste gilt dabei ein Zellbereich mit tabellarischen Daten, der seit Excel 2003 besonders komfortabel bearbeitet werden kann. Neue Listen können mit *Workbook.ListObjects.Add* erzeugt werden, wobei als Datenbasis ein existierender Zellbereich, eine XML-Datei oder eine externe Datenquelle (Datenbank) dienen kann.

Die Größe der Liste kann durch *Resize* verändert werden. *ShowTotals* gibt an, ob unterhalb der Liste Ergebniszellen dargestellt werden. *Unlist* löst die Liste auf. (Dabei bleiben alle Daten erhalten.)

<i>ListObjects</i>		Excel
<i>Worksheet</i> ↗	(index oder name) ↘ <i>ListObject</i>	

Die Auflistung verweist auf die Listen eines Tabellenblatts.

<i>ListRow</i>		Excel
<i>ListRows</i> (..) ↗	.Range ↘ <i>Range</i>	

Das Objekt beschreibt eine Zeile einer Liste.

ListRows

Excel

[ListObject ↗](#)[\(index oder name\) ↘ ListRow](#)

Die Auflistung verweist auf die Zeilen einer Liste. Mit [Add](#) kann an einer vorgegebenen Position eine neue Zeile eingefügt werden.

Mailer

Excel

[Workbook ↗](#)

Das Objekt steuert die Weiterleitung (Übertragung) der Arbeitsmappe via Netzwerk. Das Objekt ist nur von Interesse, wenn Sie auf einem Apple Macintosh mit der Netzwerkerweiterung PowerTalk arbeiten. Um eine Wurfsendung durchzuführen, stellen Sie die Eigenschaften von [Mailer](#) ein und starten die Verteilung mit [SendMailer](#).

MultiPage

MS-Forms

[UserForm.Controls\(..\) ↗](#)[.Pages\(..\) ↘ Page](#)

Das Objekt dient zur Verwaltung mehrblättriger Dialoge. Jedes Dialogblatt wird durch ein eigenes [Page](#)-Objekt repräsentiert (Zugriff über [Pages\(n\)](#)). [Value](#) gibt die gerade sichtbare Seite des Dialogs an.

Name

Excel

[Workbook.Names\(..\) ↗](#)

Das Objekt beschreibt normalerweise einen benannten Zellbereich. Dieser Zellbereich kann sowohl in Tabellenformeln als auch im VBA-Code durch die Angabe seines Namens (beispielsweise [\[Gewinn\]](#) statt [\[F17\]](#)) verwendet werden. Aus Kompatibilitätsgründen zu Excel 4 können [Name](#)-Objekte auch auf herkömmliche Makros verweisen, was sich in zahlreichen, ansonsten überflüssigen Eigenschaften niederschlägt.

Die wichtigsten Eigenschaften sind [Name](#) (mit den Namen des [Name](#)-Objekts) und [RefersTo](#) (mit einer Formel, die den Zellverweis enthält, z. B. "[=Tabelle1!\\$A\\$1](#)").

Names

Excel

[Workbook ↗](#)[\(index oder name\) ↘ Name](#)

Das Auflistungsobjekt verweist auf alle definierten Namen einer Arbeitsmappe. Namen werden in der Regel zur Bezeichnung von Zellbereichen verwendet (Kommando [FORMELN|NAMEN DEFINIEREN](#)). Im Programmcode können Bereiche einfach durch die Veränderung der [Name](#)-Eigenschaft des Bereichs benannt werden.

NewFont

MS-Forms

[steuerelement.Font ↗](#)

Das Objekt dient zur internen Darstellung von Zeichensätzen ([Font](#)-Eigenschaft vieler Steuerelemente). Dieselbe Funktion erfüllt auch das Objekt [StdFont](#) der StdOLE-Bibliothek.

ODBCError

Excel

[Application.ODBCErrors\(n\) ↗](#)

Das Objekt enthält in der Eigenschaft [ErrorString](#) eine Fehlermeldung zum letzten ODBC-Fehler. Solche Fehler können beim Datenbankzugriff auf Datenbankserver auftreten. Vielleicht würden Sie das Objekt eher in der ADO-Bibliothek zur Datenbankprogrammierung vermuten. ODBC-Fehler können aber auch direkt in Excel auftreten, etwa wenn in Pivottabellen oder [QueryTable](#)-Objekten auf externe Daten zugegriffen wird.

ODBCErrors	Excel
<i>Application</i> ↗	<i>(index)</i> ↘ <i>ODBCError</i>

Das Objekt listet alle beim letzten ODBC-Zugriff aufgetretenen Fehler auf. (Datenzugriffe via ODBC werden von einem ganzen Konglomerat von Funktionsbibliotheken verarbeitet. Auf jeder Ebene – die unterste stellt der Datenbank-Server selbst dar – können Fehler auftreten. Aus diesem Grund ist es möglich, dass ein ODBC-Zugriff gleich mehrere Fehlermeldungen liefert.) Ob Fehler aufgetreten sind, können Sie über die *Count*-Eigenschaft feststellen.

OLEDBError[s]	Excel
<i>Application</i> ↗	

Die beiden Objekte entsprechen *ODBCError[s]*, gelten aber für Fehler, die durch die OLEDB-Bibliotheken verursacht wurden (z.B. beim Zugriff auf einen OLAP-Cube durch eine Pivottabelle).

OLEFormat	Excel
<i>Shape</i> ↗	<i>.Object</i> ↘ <i>oleprogramm</i>

Das Objekt ist eine verkleinerte Variante von *OLEObject*, d.h., es enthält nur einen Bruchteil der Eigenschaften von *OLEObject*. Das Objekt dient speziell zur Bearbeitung von OLE-Objekten, die als *Shape*-Objekte in ein Excel-Tabellenblatt eingebettet sind.

OLEObject	Excel
<i>Worksheet.OLEObjects(..)</i> ↗	<i>.Object</i> ↘ <i>oleprogramm</i>

OLE-Objekte sind Objekte anderer Windows-Programme, die innerhalb Excels eingebettet sind und dort angezeigt werden (beispielsweise CorelDraw-Zeichnungen). OLE-Objekte können in benutzerdefinierten Dialogen, Diagrammen oder Tabellenblättern enthalten sein.

OLEObject beschreibt einige äußere Eigenschaften des Objekts, die von Excel aus (ohne Aufruf des OLE-Programms) verändert werden können – etwa Position, Größe, Rahmen und Schatten. Die Eigenschaft *OLEType* gibt an, wie das Objekt in Excel eingebunden ist: als eingebettetes, eigenständiges Objekt oder als verknüpftes Objekt einer anderen Datei. *Object* verweist auf das OLE-Programm und ermöglicht Object-Automation-Anwendungen.

Die Methode *Activate* ruft das zugeordnete OLE-Programm auf. *Update* bringt die Daten auf den neuesten Stand (das erfolgt normalerweise in regelmäßigen Abständen automatisch). Über *Verb* können (sehr wenige) vordefinierte Kommandos an das OLE-Programm übergeben werden.

OLEObjects	Excel
<i>Worksheet</i> ↗	<i>(index oder name)</i> ↘ <i>OLEObject</i>

Das Auflistungsobjekt verweist auf alle Objektgruppen in einem benutzerdefinierten Dialog, Diagramm oder Tabellenblatt. Siehe *OLEObject*.

OptionButton	MS-Forms
<i>UserForm.Controls</i> ↗	

Das Optionsfeld ermöglicht die bequeme Auswahl einer von mehreren Optionen. Jede Option wird durch ein eigenes Optionsfeld gebildet. Zusammengehörige Optionsfelder müssen durch eine einheitliche *GroupName*-Einstellung gekennzeichnet werden (nur erforderlich, wenn in einem Dialog mehrere Optionsgruppen verwendet werden). Zur Auswertung muss die *Value*-Eigenschaft aller Optionsfelder getestet werden.

Outline Excel[Worksheet ↗](#)

Das Objekt **Outline** dient zur internen Darstellung einer hierarchischen Gruppierung einer Tabelle. (Solche Gruppierungen werden durch **DATEN|GRUPPIEREN** oder **DATEN|TEILERGEBNIS** gebildet.) Die Methode **ShowLevels** gibt an, wie viele Zeilen- oder Spaltenebenen angezeigt werden sollen. Die Eigenschaften **SummaryColumn** und **SummaryRow** geben an, ob sich Ergebniszellen rechts bzw. unterhalb der Daten befinden (Defaulteinstellung) oder links bzw. oberhalb.

Aufbau und Veränderungen an Gliederungen erfolgen durch Methoden und Eigenschaften des **Range**-Objekts (**AutoOutline**, **ClearOutline**, **Group** und **Ungroup**). **OutlineLevel** gibt die Gliederungsebene einer einzelnen Spalte/Zeile an bzw. verändert sie.

Page MS-Forms[MultiPage.Pages\(..\) ↗](#)[.Controls\(..\) ↘ steuerelement](#)

Das Objekt repräsentiert eine Seite aus einem mehrblättrigen Dialog (**MultiPage**). Die darin enthaltenen Steuerelemente werden über **Controls** angesprochen. Die Seite wird mit **Caption** beschriftet.

Pages MS-Forms[MultiPage ↗](#)[\(index oder name\) ↘ Page](#)

Das Objekt listet alle Seiten eines mehrblättrigen Dialogs auf.

PageSetup

Excel

[Chart ↗](#)[Worksheet ↗](#)

Das Objekt beschreibt alle druckerspezifischen Daten für die Seitengestaltung. Es steht für alle Blatttypen sowie für das **Window**-Objekt zur Verfügung. Die Druckereinstellung muss für jedes einzelne Blatt durchgeführt werden, die Daten gelten also **nicht** für eine ganze Arbeitsmappe!

Kopf- und Fußzeilen werden über die sechs Eigenschaften **Left**-, **Center**- und **RightHeader** bzw. **-Footer** eingestellt. **Left**-, **Right**-, **Top**- und **BottomMargin** bestimmen das Ausmaß der Seitenränder. **Orientation** bestimmt, ob im Hoch- oder Querformat gedruckt werden soll. **Zoom** definiert einen generellen Skalierungsfaktor für den Ausdruck (10 bis 400 Prozent).

Bei Tabellenblättern gibt **PrintArea** den zu druckenden Zellbereich an. **PrintTitleColumns** und **-Line** bestimmen jene Spalten/Zeilen, die auf **jeder** Seite gedruckt werden sollen. Bei Diagrammen bestimmt **ChartSize**, wie die zur Verfügung stehende Seitengröße genutzt werden soll.

Der eigentliche Ausdruck wird durch die Methode **PrintOut** gestartet, die für diverse Objekte (**Range**, alle drei Blatttypen, **Workbook**) zur Verfügung steht. In Tabellenblättern können über die **H/VPPageBreaks**-Auflistungen horizontale und vertikale Seitenumbrüche eingefügt werden.

Pane Excel[Window.Panes\(..\) ↗](#)[.VisibleRange ↘ Range](#)[Window.ActivePane ↗](#)

Das Objekt **Pane** beschreibt einen von maximal vier Fensterausschnitten, die durch das Teilen (und Fixieren) von Fenstern entstehen. Die Methode **Activate** wählt den gerade aktiven Ausschnitt aus. Die beiden Eigenschaften **ScrollColumn** und **ScrollRow** geben die

Nummer der ersten sichtbaren Spalte/Zeile im Ausschnitt an bzw. verändern diese Spalte/Zeile. *VisibleRange* verweist auf den im Ausschnitt sichtbaren Zellbereich.

Panes Excel	
<i>Window</i> ↗	(<i>index</i>) ↘ <i>Pane</i>

Das Auflistungsobjekt *Panes* verweist auf die Ausschnitte eines Fensters (siehe *Pane*). Wenn das Fenster nicht geteilt ist, enthält die Eigenschaft *Count* den Wert 1. Ungeteilte Fenster können durch die Veränderung der *Window*-Eigenschaften *Split* und *FreezePanes* an der aktuellen Cursorposition geteilt werden.

Parameter ADO	
<i>Command!name</i> ↗	
<i>Parameters(...)</i> ↗	

Das Objekt beschreibt einen Parameter eines Datenbankkommandos. *Name* und *Type* geben den Parameternamen und seinen Datentyp an. *Direction* bestimmt, ob es sich um einen Ein- oder Ausgabeparameter handelt. *Value* enthält den Wert des Parameters.

Parameter Excel	
<i>QueryTable.Parameters(..)</i> ↗	<i>.SourceRange</i> ↘ <i>Range</i>

Wenn in einem *QueryTable*-Objekt eine SQL-Abfrage mit Parametern verwendet wird (? im SQL-Text), dann können diese Parameter mit der *SetParam*-Methode des *Parameter*-Objekts eingestellt werden. Eine andere Vorgehensweise besteht darin, den Parameter via *SourceRange* aus einem Tabellenfeld zu lesen. Zur Aktualisierung der Daten muss anschließend in jedem Fall die *Refresh*-Methode von *QueryTable* ausgeführt werden.

Parameters ADO	
<i>Command</i> ↗	(<i>index</i> oder <i>name</i>) ↘ <i>Parameter</i>

Die Auflistung verweist auf alle Parameter eines SQL-Kommandos, das über ein ADO-*Command*-Objekt angesprochen wird.

Parameters Excel	
<i>QueryTable</i> ↗	(<i>index</i> oder <i>name</i>) ↘ <i>Parameter</i>

Die Auflistung listet alle Parameter einer SQL-Abfrage eines *QueryTable*-Objekts auf. (Aus jedem ?-Zeichen im SQL-Kommando resultiert ein Parameter.)

Picture Excel	
<i>PageSetup.Right-/Center-/LeftFooterPicture</i> ↗	
<i>PageSetup.Right-/Center-/LeftHeader</i> ↗	

Das Objekt beschreibt eine Grafik, die in der Kopf- oder Fußzeile der Datei ausgedruckt wird. Dazu müssen zumindest der Dateiname (*FileName*) und die gewünschte Größe des Bilds angegeben werden. Die Einheit von *Height* und *Width* ist nicht dokumentiert; wahrscheinlich sind es Punkt (ca. 0,35 mm). *Width=100* bewirkt, dass die Grafik ca. 35 mm breit ausgedruckt wird. Außerdem muss in der *Footer*- oder *Header*-Eigenschaft des *PageSetup*-Objekts das Kürzel *&G* eingefügt werden. Es gibt die Position der Grafik innerhalb der Kopf- bzw. Fußzeile an.

```
ActiveSheet.PageSetup.LeftFooterPicture.FileName = "C:\test.bmp"
ActiveSheet.PageSetup.LeftFooterPicture.Height = 20
ActiveSheet.PageSetup.LeftFooterPicture.Width = 20
ActiveSheet.PageSetup.LeftFooter = "&G"
```

PictureFormat

Excel

[Shape/ShapeRange ↗](#)

Das Objekt beschreibt Merkmale von Bildern, die in *Shape*-Objekten dargestellt werden (*Type=mso[Linked]Picture* oder *msoXxxOLEObject*). Zu den wichtigsten Eigenschaften zählen *Brightness* und *Contrast*. Mit *CropLeft*, *-Bottom*, *-Top* und *-Right* kann der sichtbare Bildausschnitt eingestellt werden.

Anmerkung: Bitmap-Dateien sollten mit *Shapes.AddPicture* eingefügt werden. Die Makroaufzeichnung greift stattdessen noch auf die nicht mehr unterstützte *Pictures*-Auflistung zurück.

PivotCache

Excel

[Workbook.PivotCaches\(..\) ↗](#)
[PivotTable ↗](#)

Über das *PivotCache*-Objekt werden die einer Pivottabelle zugrunde liegenden Daten sowie (ähnlich wie bei *QueryTable*) die Verbindungsinformationen gespeichert. Das Objekt spielt insbesondere dann eine große Rolle, wenn die Basisdaten nicht aus einer Excel-Tabelle, sondern aus einer externen Datenbank stammen. (Wenn eine Excel-Datei unerwartet groß wird, gibt der Ausdruck von *MemoryUsed* für alle *PivotCache*-Objekte oft eine schlüssige Begründung.)

PivotCaches

Excel

[Workbook ↗](#) (index oder name) ↘ *PivotCache*

Die Auflistung verweist auf die *PivotCache*-Objekte in einer Excel-Arbeitsmappe.

PivotCell

Excel

[Range ↗](#) .Column-/RowItems ↘ *PivotItemList*
.Data-/PivotField ↘ *PivotField*
.PivotTable ↘ *PivotTable*

Das Objekt beschreibt eine einzelne Zelle einer Pivottabelle. Die eigentliche Funktion des Objekts besteht darin, eine Verbindung zwischen einer einzelnen Tabellenzelle (*Range*) und einer Pivottabelle herzustellen. Interessant ist vor allem die Eigenschaft *PivotCellType*, die den Typ der Zelle angibt (z.B. *xlPivotCellValue*, *xlPivotCellSubtotal*). Die Eigenschaften *PivotTable*, *Column-* und *RowItems* bzw. *Data-* und *PivotField* verweisen auf verschiedene Bestandteile der Pivottabelle, in der sich das Feld befindet.

PivotField

Excel

[PivotTable.PivotFields ↗](#) .PivotItems ↘ *PivotItem*

Pivotfelder sind die Gliederungsfelder von Pivottabellen. Es handelt sich dabei um jene Felder, die Sie von der *PivotTable*-Feldliste in die Bereiche „Spaltenbeschriftungen“, „Zeilenbeschriftungen“ etc. verschieben. Die Eigenschaften von Pivotfeldern steuern den eigentlichen Inhalt der Pivottabelle. Über die Methoden *Pivot-*, *Hidden-*, *Visible-*, *Parent-* und *ChildItems* kann auf einzelne Pivotelemente eines Pivotfelds zugegriffen werden.

PivotFields

Excel

[PivotTable ↗](#) (index oder name) ↘ *PivotField*

Das Auflistungsobjekt verweist auf die *PivotField*-Objekte einer Pivottabelle. Die Auflistung erfasst nur die Pivotfelder, die unmittelbar aus der Datenbasis gebildet werden. Nicht berücksichtigt werden beispielsweise berechnete Felder (siehe *CalculatedFields*)

und Datenfelder (*PivotTable.DataFields* bzw. *PivotTable.VisibleFields*), deren zusammengesetzter Name vom ursprünglichen Feldnamen abweicht (etwa "*Summe - quantity*" statt "*quantity*").

PivotFormula

Excel

PivotTable.PivotFormulas(..) ↗

Das *PivotFormula*-Objekt beschreibt ein Formelfeld in einer Pivottabelle. (Im interaktiven Betrieb werden Formelfelder durch das FORMELN-Untermenü in der Befehlsregisterkarte OPTIONEN erstellt.)

PivotFormulas

Excel

PivotTable ↗

(*index*) ↘ *PivotFormula*

Die Auflistung listet alle Formelfelder einer Pivottabelle auf. (Die meisten Pivottabellen kommen ohne Formeln aus.)

PivotItem

Excel

PivotField.PivotItems(..) ↗

.*ChildItems* ↘ *PivotItems*

Pivotelemente sind die kleinsten Dateneinheiten einer Pivottabelle. Sie enthalten die Gruppen, in die ein Pivotfeld aufgeteilt wurde. (Bei einer Artikelliste, in der die Preiskategorien I, II und III vorkommen, existieren zum Pivotfeld „Preis“ die Pivotelemente „I“, „II“ und „III“.) Pivotelemente besitzen kaum eigene Eigenschaften, die Aufbau und Inhalt der Pivottabelle verändern. Einzig *ShowDetail* verändert die Anzeige von untergeordneten Details.

PivotItemList

Excel

PivotField.PivotItems(..) ↗

(*index* oder *name*) ↘ *PivotItem*

Das Auflistungsobjekt ist eine Variante zu *PivotItems* (siehe unten). Der einzige Unterschied besteht darin, dass die *Add*-Methode fehlt.

PivotItems

Excel

PivotField.PivotItems(..) ↗

(*index* oder *name*) ↘ *PivotItem*

Das Auflistungsobjekt verweist auf die *PivotItem*-Objekte eines Pivotfelds. Siehe *PivotItem*.

PivotLayout

Excel

Chart ↗

.*PivotFields* ↘ *PivotField*

.*PivotTable* ↘ *PivotTable*

PivotLayout stellt die Querverbindung zwischen einem Pivotdiagramm (*Chart*) und der dazugehörigen Pivottabelle her. Zudem kann direkt über *PivotLayout* auf all jene Eigenschaften von *PivotTable* zugegriffen werden, die das Layout (den Aufbau) einer Pivottabelle bestimmen. *Chart.PivotLayout.PivotFields* ist also eine Kurzschreibweise für *Chart.PivotLayout.PivotTable.PivotFields*. Wenn ein Diagramm nicht mit einer Pivottabelle verbunden ist (wenn es sich also um ein herkömmliches Diagramm handelt), enthält *Chart.PivotLayout* den Wert *Nothing*.

Beachten Sie bitte, dass der Objektvergleich *Chart.PivotLayout.PivotTable Is PivotTable* unter Umständen *False* liefert, auch wenn es sich um dieselbe Pivottabelle handelt. Wenn Sie einen derartigen Vergleich durchführen müssen, vergleichen Sie die Eigenschaften *.Worksheet.Name* und *.Address* von *PivotTable.TableRange1*.

PivotTable

Excel

[Worksheet.PivotTables](#) ↗[.PivotFields](#) ↘ [PivotField](#)

Pivottabellen (Kreuztabellen) sind das wichtigste Instrument von Excel zur Analyse tabellarischer Daten (Listen). In Pivottabellen werden die Tabelleneinträge nach verschiedenen Kriterien zu Gruppen zusammengefasst und als Raster dargestellt. **PivotTable** dient zur Verwaltung von Pivottabellen. Die Eigenschaft [SourceData](#) verweist auf die Ausgangsdaten der Tabelle. Zahlreiche Eigenschaften wie [DataBodyRange](#), [RowRange](#), [ColumnRange](#) etc. verweisen auf jene Bereiche, in denen die Ergebnisse der Pivottabelle angezeigt werden.

Mit der Methode [AddFields](#) kann die Pivottabelle erweitert werden. [RefreshTable](#) bringt die Tabelle auf den neuesten Stand. [ShowPages](#) erstellt für ein ausgewähltes Seitenfeld eine oder mehrere neue Pivotdetailtabellen (in neuen Tabellenblättern).

Die Methoden [PivotFields](#), [HiddenFields](#), [DataFields](#), [PageFields](#), [ColumnFields](#) und [RowFields](#) verweisen jeweils auf ausgewählte **PivotField**-Objekte.

PivotTables

Excel

[Worksheet](#) ↗[\(index oder name\)](#) ↘ [PivotTable](#)

Das Auflistungsobjekt verweist auf alle Pivottabellen eines Tabellenblatts. Es existiert keine [Add](#)-Methode; neue Pivottabellen werden über die Methode [PivotTableWizard](#) erzeugt. Siehe [PivotTable](#).

PlotArea Excel[Chart](#) ↗[.Interior](#) ↘ [Interior](#)[.Border](#) ↘ [Border](#)

Das Objekt beschreibt den Hintergrund des Zeichnungsbereichs eines Diagramms. Der Zeichnungsbereich ist jener Teilbereich des Diagramms, in dem die eigentliche Grafik angezeigt wird. Das Objekt steht im Gegensatz zu [ChartArea](#), durch das der gesamte Diagrammbereich (inklusive Titel, Legende etc.) beschrieben wird. Die Gestaltung des Zeichnungsbereichs erfolgt im Wesentlichen über die Subobjekte [Interior](#) (Farbe und Muster) und [Border](#) (Umrandung). Siehe auch [Floor](#) und [Walls](#) für die Begrenzungsflächen von 3D-Diagrammen.

Point Excel[Series.Points\(..\)](#) ↗[.Border](#) ↘ [Border](#)[.DataLabel](#) ↘ [DataLabel](#)

Das **Point**-Objekt beschreibt einen einzelnen Datenpunkt eines Diagramms. Datenpunkte sind innerhalb des Diagramms zu Datenreihen zusammengefasst, also zu Gruppen zusammengehöriger Daten. Über das **Point**-Objekt können der Datenpunkt selbst, der Linienabschnitt vom vorigen Datenpunkt zu diesem Datenpunkt und die Beschriftung des Datenpunkts optisch gestaltet werden. Dazu stehen die Eigenschaften [DataLabel](#), [MarkerStyle](#), [MarkerBackground](#)- und [MarkerForegroundColor](#) sowie [Border](#) zur Verfügung.

Points Excel[Series.Points](#) ↗[\(index\)](#) ↘ [Point](#)

Das Auflistungsobjekt verweist auf die Datenpunkte einer Datenreihe. Als [index](#) muss die Nummer des Datenpunkts angegeben werden (also 5 für den fünften Datenpunkt). Siehe [Point](#).

Properties/Property

ADO

[Connection ↗](#)
[Recordset ↗](#)

Die Auflistung listet dynamische Eigenschaften diverser ADO-Objekte ([Connection](#), [Command](#), [Field](#), [Parameter](#), [Recordset](#)) auf. Mit welchen Eigenschaften ein Objekt ausgestattet ist, hängt vor allem vom Datenbanktreiber ab ([Provider](#)-Einstellung in [Connection.ConnectionString](#)). Die wichtigsten [Property](#)-Eigenschaften sind [Name](#), [Type](#), [Attributes](#) und [Value](#).

Properties/Property

VBE

[VBComponent ↗](#)

Die Auflistung listet alle Eigenschaften ([Name](#)) und deren Einstellungen ([Value](#)) einer VBA-Komponente auf.

Protection

Excel

[Worksheet ↗](#) [.AllowEditRanges](#) ↘ [AllowEditRanges](#)

Das Objekt gibt mit den Eigenschaften [AllowFormattingCells](#), [AllowSorting](#) etc. Auskunft über die Blattschutzoptionen. Die [AllowXxx](#)-Eigenschaften können nur gelesen, aber nicht verändert werden. Veränderungen erfolgen mit der [Protect](#)-Methode.

PublishObject

Excel

[Chart/Range.CreatePublisher ↗](#)
[PublishObjects\(..\) ↗](#)

Das Objekt verwaltet die Einstellungen einer HTML-Exportoperation für Excel-Daten (z.B. für einige Tabellenzellen, ein ganzes Tabellenblatt oder ein Diagramm). Der Export kann mit der Methode [Publish](#) jederzeit wiederholt werden. Wichtige Exporteigenschaften sind [FileName](#) (der Dateiname bzw. die HTML-Adresse), [HtmlType](#) (Art der HTML-Datei, z.B. [xlHtmlStatic](#)) sowie [Sheet](#) und [Source](#) zur Beschreibung der Datenquelle.

PublishObjects

Excel

[Workbook ↗](#) [\(index oder name\)](#) ↘ [PublishObject](#)

Die Auflistung verweist auf alle Komponenten der Datei, die im HTML-Format exportiert werden können.

QueryTable

Excel

[Range ↗](#)
[Worksheet.QueryTables\(..\) ↗](#)

Das Objekt speichert alle relevanten Daten zur Durchführung eines Datenimports aus einer Textdatei, aus einer Datenbank, aus einem OLAP-Cube oder aus einer Webseite. Die Eigenschaft [QueryType](#) gibt den Typ der Datenquelle an. Wenn entsprechend der Abfragequelle alle weiteren Eigenschaften korrekt eingestellt sind, kann der Import mit [Refresh](#) durchgeführt bzw. wiederholt werden (zur Aktualisierung der Daten).

Wenn der Import mit MS Query durchgeführt wurde, gilt [QueryType = xlODBCQuery](#). In diesem Fall enthält die Eigenschaft [Connection](#) die Zugriffsinformationen zur Datenbankdatei bzw. zum Datenbank-Server. [CommandText](#) enthält den SQL-Code der Abfrage.

Bei einem Textimport gilt [QueryType=xlTextImport](#). Die Eigenschaft [Connection](#) enthält jetzt die Zeichenkette "[Text](#)"; gefolgt vom vollständigen Namen der zu importierenden

Datei. Die diversen Parameter des Imports werden durch eine ganze Reihe von *Text-FileXxx*-Eigenschaften bestimmt.

QueryTables		Excel
Worksheet.QueryTable ↗	(<i>index</i> oder <i>name</i>) ↘	QueryTable

Die Auflistung verweist auf alle externen Datenquellen in einer Excel-Datei.

Range		Excel
Worksheet.Range(..) ↗	.Cells ↘	Range
Worksheet.Cells(..) ↗	.Areas ↘	Range
Application.ActiveCell ↗	.Font ↘	Font
Application.Selection ↗	.Border ↘	Border

Das Objekt **Range** kann eine einzelne Zelle, einen ganzen (unter Umständen aus mehreren Teilbereichen zusammengesetzten) Zellbereich sowie ganze Zeilen oder Spalten einer Tabelle umfassen. Es gibt zahllose Eigenschaften und Methoden, die von den verschiedensten Objekten auf Bereiche verweisen. Umgekehrt können ausgehend von einem **Range**-Objekt beinahe beliebig viele andere (Teil-)Bereiche angesprochen werden (z. B. über die Eigenschaften bzw. Methoden *CurrentRegion*, *SpecialCells*, *End*, *Dependents*, *EntireColumn*, *EntireRow*, *Precedents* etc.).

Aus mehreren Teilbereichen zusammengesetzte Bereiche müssen generell durch die Auswertung von *Areas* weiterverarbeitet werden. Das **Range**-Objekt liefert in diesem Fall nur den ersten Teilbereich.

Die Zeichenkette zur Beschreibung eines Zellbereichs kann mit *Address* ermittelt werden. (Wenn die Zellen A1:B4 ausgewählt sind, dann liefert *Selection.Address* die Zeichenkette "\$A\$1:\$B\$4".) Über mehrere Parameter kann das gewünschte Adressformat (absolut, relativ, A1 oder Z1S1, extern) ausgewählt werden.

Der Inhalt von Zellen kann über die Eigenschaften *Value* und *Formula* ausgelesen bzw. verändert werden. Bei Bereichen mit mehreren Zellen wird beim Lesen nur der Inhalt der ersten Zelle ermittelt, während das Schreiben alle Zellen verändert. Die *Text*-Eigenschaft einer Zelle kann nur gelesen, aber nicht verändert werden.

Die wichtigsten Eigenschaften und Methoden zur Formatierung von Zellen sind *Font*, *Border*, *Interior* (für die Hintergrundfarbe), *Orientation* (Textrichtung: horizontal, vertikal), *HorizontalAlignment* (links, zentriert, rechts, Blocksatz), *VerticalAlignment* (oben, Mitte, unten), *NumberFormat*, *Style*, *ColumnWidth* und *RowHeight*.

Zur Bewegung des Zellzeigers bzw. zur Veränderung des markierten Bereichs stehen die beiden Methoden *Select* und *Offset* zur Verfügung.

RecentFile		Excel
Application.RecentFiles(..) ↗		

Das Objekt gibt den Dateinamen (Eigenschaften *Name* und *Path*) einer vor kurzem verwendeten Excel-Datei an. Mit der Methode *Open* kann diese Datei wieder geöffnet werden.

RecentFiles		Excel
Application ↗	(<i>index</i>) ↘	RecentFile

Die Auflistung zählt die zuletzt geöffneten Excel-Dateien auf.

Recordset

ADO

[Command.Open ↗](#)[.ActiveConnection ↘](#) [Connection](#)
[!name ↘](#) [Field](#)

Das Objekt ermöglicht die Bearbeitung von Datensatzlisten, die aus einer SQL-Abfrage resultieren. Das Objekt hat damit eine zentrale Bedeutung in ADO-Datenbankanwendungen, weil es sowohl das Lesen als auch das Verändern von Daten ermöglicht.

Die Datensatzliste wird mit der Methode *Open* erstellt. Die drei Eigenschaften *CursorLocation*, *CursorType* und *LockType* bestimmen, welche Funktionen das *Recordset*-Objekt unterstützt und wie effizient die interne Verwaltung erfolgt.

Bei einem geöffneten *Recordset*-Objekt kann über *!name* auf die Datenfelder des gerade aktiven Datensatzes zugegriffen werden. Um einen anderen Datensatz zu aktivieren, stehen die Methoden *MoveNext*, *MovePrevious* etc. zur Verfügung. Die Eigenschaften *EOF* und *BOF* zeigen an, ob die Navigation über das Ende bzw. den Anfang der Datensatzliste hinausgeführt hat.

RefEdit RefEdit[UserForm.Controls\(..\) ↗](#)

Das *RefEdit*-Steuerelement (Formelfeld) ist nicht Teil der MS-Forms-Bibliothek, sondern wird in der eigenständigen RefEdit-Bibliothek zur Verfügung gestellt. Es stellt eine Variante zum Textfeld dar und ermöglicht die komfortable Eingabe von Zellbezügen. *Value* enthält den eingegebenen Text bzw. den Zellbezug als Zeichenkette.

Reference[s]

VBE

[VBProject ↗](#)

References zählt alle Objektverweise (Bibliotheken) auf, die in einem VBA-Projekt benutzt werden. Über die Eigenschaften des untergeordneten *Reference*-Objekts können deren Name, Dateiname, eine kurze Beschreibung etc. ermittelt werden.

RoutingSlip

Excel

[Workbook ↗](#)

Das Objekt steuert die Weiterleitung (Übertragung) der Arbeitsmappe via E-Mail. Die wichtigsten Eigenschaften sind *Recipients* (Liste jener Personen/Rechner, denen die Mappe gesandt werden soll), *Message* und *Subject* (Inhaltsangabe) und *Delivery*. Die Übertragung der Arbeitsmappen wird durch die *Workbook*-Methode *Route* gestartet.

Scenario Excel[Worksheet.Scenarios\(..\) ↗](#)[.ChangingCell ↘](#) [Range](#)

Szenarios sind ein Hilfsmittel, um verschiedene Änderungen an Ausgangsdaten eines Tabellenmodells miteinander zu vergleichen. Das *Scenario*-Objekt beschreibt, welche Zellen der Tabelle wie zu ändern sind. Das Objekt kennt zwei charakteristische Eigenschaften: *ChangingCell* verweist auf einen zumeist zusammengesetzten Zellbereich mit den variablen Werten. *Values* enthält ein Datenfeld, das jene Zahlenwerte angibt, die in die Zellen einzusetzen sind. Die Methode *ChangeScenario* stellt eine Möglichkeit dar, beide Eigenschaften gleichzeitig zu verändern.

Scenarios

Excel

[Worksheet ↗](#)[\(index oder name\) ↘](#) [Scenario](#)

Das Auflistungsobjekt verweist auf die in einem Tabellenblatt definierten Szenarios. *Add* erstellt ein neues Szenario. Die Methode *CreateSummary* erstellt auf der Basis aller

Szenarien einen Gesamtbericht (in einem neuen Tabellenblatt), in dem alle veränderten Zellen und die daraus resultierenden Ergebnisse angegeben werden.

ScrollBar		MS-Forms
UserForm.Controls ↗		

Mit der Bildlaufleiste kann ein Zahlenwert (*Value*) innerhalb eines vorgegebenen Bereichs (*Min* bis *Max*) eingestellt werden.

Series		Excel
Chart.SeriesCollection(..) ↗	.Trendlines ↘ Trendline	
ChartGroup.SeriesCollection(..) ↗	.Points ↘ Point	
	.ErrorBars ↘ ErrorBars	
	.DataLabels ↘ DataLabel	

Datenreihen enthalten Gruppen zusammengehöriger Datenpunkte eines Diagramms. Eine Datenreihe kann beispielsweise alle Werte für einen Linienzug in einem Liniendiagramm enthalten. Falls im Diagramm mehrere Linienzüge dargestellt werden sollen, sind auch mehrere Datenreihen erforderlich.

Über das Objekt *Series* können verschiedene Formatierungsmerkmale für die grafische Anzeige der gesamten Datenreihe verändert werden. Dazu stehen generell dieselben Eigenschaften/Methoden wie für die individuelle Formatierung einzelner *Point*-Objekte zur Verfügung – siehe etwas weiter oben.

Type bestimmt den Diagrammtyp für die einzelne Datenreihe. (Durch die Wahl unterschiedlicher Diagrammtypen für mehrere Datenreihen entstehen aus Diagrammgruppen zusammengesetzte Verbunddiagramme.) Über *AxisGroup* wird die Datenreihe einer von zwei möglichen Koordinatenachsen zugeordnet. *PlotOrder* rückt einzelne Datenreihen in 3D-Diagrammen nach vorne oder nach hinten. *Smooth* bestimmt, ob die Kurve eines Liniendiagramms geglättet wird. Zusätzliche Effekte lassen sich über die Subobjekte *ErrorBars* und *Trendline* erzielen.

SeriesCollection		Excel
Chart ↗	(index oder name) ↘ Series	
ChartGroup ↗		

Das Auflistungsobjekt verweist auf die Datenreihen eines Diagramms bzw. einer Diagrammgruppe in einem Verbunddiagramm (siehe *Series*). Über die Methoden *Paste* bzw. *Add* kann die Anzahl der Datenreihen vergrößert werden. *Extend* vergrößert die Anzahl der Datenpunkte der Datenreihen.

SeriesLines		Excel
ChartGroup ↗	.Border ↘ Border	

Verbindungslinien verbinden die Balken oder Säulen eines gestapelten Balken- oder Säulendiagramms. („Gestapelt“ bedeutet, dass mehrere Datenreihen nicht in eigenen Säulen, sondern in Teilen einer Säule übereinander dargestellt werden.) Verbindungslinien verdeutlichen so die Entwicklung der einzelnen Werte. Das Aussehen der Verbindungslinien wird über das *Border*-Subobjekt gesteuert.

ShadowFormat		Excel
Shape/ShapeRange ↗	.ForeColor ↘ ColorFormat	

Das Objekt beschreibt die Schatteneffekte von *Shape*-Objekten. Die wichtigsten Eigenschaften sind *ForeColor* (die Farbe des Schattens), *OffsetX* und *-Y*.

Shape Excel

[Comment](#) ↗ [ConnectorFormat](#). [Begin/EndConnectedShape](#) ↗ [ConnectorFormat](#) ↘ [ConnectorFormat](#)
[FreeFormBuilder.ConvertToShape\(..\)](#) ↗ [ControlFormat](#) ↘ [ControlFormat](#)
[Hyperlink](#) ↗ [Nodes](#) ↘ [ShapeNodes](#)
[Shapes.AddShape\(..\)](#) ↗ [TopLeftCell](#), [BottomRightCell](#) ↘ [Range](#)
[Worksheet.Shapes\(..\)](#) ↗ ...

Das **Shape**-Objekt dient primär zur Darstellung von AutoFormen (Linien, Rechtecke, Pfeile, Sterne etc. – siehe [EINFÜGEN | FORMEN](#)). Es löst damit die diversen Zeichnungsobjekte aus Excel 5/7 ab. Das Objekt wird aber auch zur Verwaltung vollkommen fremder Objekte (etwa der MS-Forms-Steuer-elemente) eingesetzt.

Es gibt eine ganze Reihe verwandter Objekte: **ShapeRange** ermöglicht die gemeinsame Bearbeitung mehrerer **Shape**-Objekte. Freihandformen stellen eine Sonderform von **Shape**-Objekten dar. In diesem Fall verweist die Eigenschaft **ShapeNodes** auf eine gleichnamige Auflistung von **ShapeNode**-Objekten. Mit dem **GroupShape**-Objekt werden mehrere zu einer Gruppe zusammengefasste Elemente verwaltet.

ShapeRange

Excel

[Shapes.Range\(Array\(...,...\)\)](#) ↗ [\(index oder name\)](#) ↘ [Shape](#)
[Charts/OLEObjects.ShapeRange](#) ↗

Das Objekt ermöglicht die gemeinsame Bearbeitung einer ganzen Ansammlung von **Shape**-Objekten. Dazu stehen die meisten Eigenschaften und Methoden des **Shape**-Objekts zur Verfügung. Darüber hinaus kann durch **Group** bzw. **ReGroup** eine Objektgruppe gebildet werden. (Diese liefert ein neues **Shape**-Objekt, das die Ansammlung vereint. Die Einzelobjekte können jetzt nur noch via **Shape.GroupItems** angesprochen werden. Diese Eigenschaft führt wiederum auf ein **GroupShape**-Objekt.)

ShapeNode

Excel

[Shape/ShapeRange.Nodes\(..\)](#) ↗

Das Objekt beschreibt ein Segment (also ein Linienstück oder eine Kurve) einer Freihandform. Die Koordinatenpunkte werden über **Points** gelesen. Die Eigenschaft liefert ein zweidimensionales **Array** als Ergebnis. (Ein eindimensionales Feld hätte eigentlich auch gereicht.) Der Zugriff auf die beiden Koordinaten wird durch die folgenden Zeilen demonstriert.

```
'erster Koordinatenpunkt des ersten Shape-Objekts
'setzt Type=msoFreeform voraus
pt = Shapes(1).Nodes(1).Points
x = pt(1,1)
y = pt(1,2)
```

Andere wichtige Eigenschaften sind **SegmentType** (Linie oder Kurve) und **EditingType** (z.B. Eckpunkt oder Symmetriepunkt). Alle drei Eigenschaften können nur gelesen werden. Zur Veränderung müssen die **SetXxx**-Methoden des **ShapeNodes**-Objekts verwendet werden.

ShapeNodes

Excel

[Shape/ShapeRange.Nodes](#) ↗ [\(index oder name\)](#) ↘ [ShapeNode](#)

Wenn ein **Shape**-Objekt eine Freihandform enthält (etwa einen frei gezeichneten Linienzug, **Shape.Type=msoFreeform**), dann verweist dessen Eigenschaft **Nodes** auf die Auflistung **ShapeNodes**, über welche die einzelnen Eckpunkte des Linienzugs angesprochen werden können. Außerdem können mit den Methoden **Insert** und **Delete** zusätzliche Liniensegmente eingefügt und mit **SetEditingType**, **SetPosition** und **SetSegmentType** bearbeitet werden. Siehe auch **FreeformBuilder**.

Shapes Excel

[Chart](#) ↗ (index oder name) ↘ *Shape*
[Worksheet](#) ↗ *.Range* ↘ *ShapeRange*

Die Auflistung zählt alle *Shape*-Objekte eines Tabellenblatts oder Diagramms auf. Neben dem Zugriff auf einzelne *Shape*-Objekte kann mit *Range* auch auf mehrere Objekte gleichzeitig zugegriffen werden. Zum Erzeugen neuer *Shape*-Objekte stehen unzählige *Add*-Methoden zur Auswahl, etwa *AddShape*, *AddLine*, *AddCurve*, *AddOLEObject* etc. *SelectAll* selektiert alle *Shape*-Objekte.

Sheets Excel

[Application](#) ↗ (index oder name) ↘ *Chart*
[Workbook](#) ↗ (index oder name) ↘ *Worksheet*
[Window.SelectedSheets](#) ↗

Das Auflistungsobjekt verweist auf alle Blätter einer Arbeitsmappe bzw. der gerade aktiven Arbeitsmappe (wenn das Objekt *Application* oder gar kein Objekt angegeben wird) bzw. auf die in einem Fenster gemeinsam markierte Blattgruppe.

Beachten Sie bitte, dass es kein *Sheet*-Objekt gibt. *Sheets* verweist je nach Blatttyp auf ein *Chart*- oder *Worksheet*-Objekt! In Excel 5/7-Dateien kann *Sheets* auch auf Objekte des Typs *DialogSheet* oder *Module* verweisen.

Über das *Sheets*-Objekt sind alle Eigenschaften bzw. Methoden der jeweiligen *Xxx-Sheets*-Objekte zugänglich. Insbesondere können mit *Add* und *Delete* Blätter erzeugt bzw. gelöscht werden. *Select* aktiviert ein ausgewähltes Blatt. Die Eigenschaft *Visible* gibt an, ob ein Blatt im Blattregister angezeigt werden soll oder nicht. Mit der Methode *FillAcrossSheets* kann ein Zellbereich eines Tabellenblatts in alle durch *Sheets* erfasste Blätter kopiert werden.

SparklineGroup Excel

[SparklineGroups](#) ↗

Verweist auf ein neues oder vorhandenes Sparklines-Diagramm.

SparklineGroups Excel

[Range](#) ↗ (index) ↘ *SparklineGroup*

Auflistung aller Sparklines-Diagramme in einem *Range*-Objekt.

SmartArt Excel

[Shapes](#) ↗

Das Objekt repräsentiert ein SmartArt-Diagramm in einem Shape-Objekt.

SmartArtLayouts Excel

[Application](#) ↗ (index) ↘ *SmartArtLayout*

Die Auflistung verweist auf alle vordefinierten Layouts für SmartArt-Diagramme.

SmartArtLayout Excel

[SmartArtLayouts](#) ↗

Das Objekt steht für ein einzelnes, vordefiniertes Layout für die Gestaltung eines SmartArt-Diagramms.

SmartTag

Excel

[Range.SmartTags\(...\)](#) ↗ [.Range](#) ↘ [Range](#)
[Worksheet.SmartTags\(...\)](#) ↗ [.SmartTagActions](#) ↘ [SmartTagActions](#)
[SmartTags.Add](#) ↗

Smart Tags sind kleine Menüs, deren Inhalt vom Inhalt einer Excel-Zelle abhängt. Über die Menüeinträge können zum Zellinhalt passende Aktionen ausgeführt werden. (Es kann z.B. eine Website mit weiterführenden Informationen angezeigt werden.)

Das *SmartTag*-Objekt weist nur wenige Eigenschaften auf: *Name* enthält eine interne Bezeichnung, die aber nur den Typ angibt (z.B. "[urn:schemas-microsoft-com:office:smart-tags#stockticker](#)"). Mehrere Smart Tags eines Tabellenblatts können also durchaus denselben Namen haben! *SmartTagActions* verweist auf die für das Smart Tag verfügbaren Aktionen (siehe unten). *XML* enthält eine Beschreibung des Smart Tags im XML-Format, wobei darin aber auch nicht viel mehr Informationen als in *Name* enthalten sind. *Range* verweist auf die Zelle, zu dem das *SmartTag* gehört.

SmartTagAction

Excel

[SmartTag.SmartTagActions\(...\)](#) ↗

Das Objekt beschreibt eine Aktion, die für ein bestimmtes *Smart-Tag*-Objekt ausgeführt werden kann. Mit der Methode *Execute* kann diese Aktion tatsächlich ausgeführt werden. Die Eigenschaft *Name* gibt eine kurze Beschreibung der Aktion (wobei Beschreibung eigentlich übertrieben ist: Der Text ist kürzer als der, der im Smart-Tag-Menü angezeigt wird; der Menüttext kann per Code aber nicht ermittelt werden).

SmartTagActions

Excel

[SmartTag](#) ↗ [\(index\)](#) ↘ [SmartTagAction](#)

Auflistung aller *SmartTagAction*-Objekte eines bestimmten Smart Tags. Bitte beachten Sie, dass die Auflistung nicht mit einer *For-Each*-Schleife durchlaufen werden kann. Sie müssen stattdessen eine Schleife von 1 bis *Count* bilden.

SmartTagOptions

Excel

[Workbook](#) ↗

Dieses Objekt steuert über zwei Eigenschaften, welche Smart-Tag-Funktionen in der aktuellen Arbeitsmappe aktiv sind: *DisplaySmartTags* gibt an, ob Smart Tags angezeigt werden. *EmbedSmartTags* gibt an, ob die Smart Tags auch in die Arbeitsmappe eingebettet werden können. Das bedeutet, dass die Smart-Tag-Informationen zusammen mit der Excel-Datei gespeichert und nach dem nächsten Laden selbst dann angezeigt werden, wenn die Smart-Tag-Funktion eigentlich deaktiviert ist.

SmartTagRecognizer[s]

Excel

[Application](#) ↗

Die Auflistung *SmartTagRecognizers* verweist auf alle am Rechner installierten Smart-Tag-Erkennungsmodule (also *SmartTagRecognizer*-Objekte). Dabei handelt es sich normalerweise nur um zwei Module, eines für Outlook-Kontakte und ein zweites zur Verwaltung von Smart-Tag-Listen. (Es besteht aber keine Möglichkeit, per Programmcode festzustellen, welche Smart-Tag-Listen für dieses Modul installiert sind.)

Die Eigenschaft *SmartTagRecognizers.Recognize* gibt an, ob die Smart-Tag-Funktion im Hintergrund ausgeführt wird (d.h., ob bei Neueingaben oder Veränderungen in Zellen automatisch Smart Tags eingefügt werden).

Über das *SmartTagRecognizer*-Objekt können Sie den Dateinamen des Moduls (Eigenschaft *FullName*) feststellen und jedes Modul einzeln aktivieren/deaktivieren (Eigenschaft *Enabled*).

SmartTags

Excel

[Range](#) ↗
[Worksheet](#) ↗[\(index\)](#) ↘ [SmartTag](#)

Die Auflistung verweist auf alle Smart Tags, die innerhalb des Zellbereichs bzw. des gesamten Tabellenblatts zur Verfügung stehen. Mit der Methode [Add](#) kann eine Zelle mit einem Smart Tag ausgestattet werden. An [Add](#) muss der Typname des Smart Tags übergeben werden. [Add](#) kann nur auf [Range](#)-Objekte angewendet werden, wenn diese genau eine einzige Zelle umfassen – andernfalls kommt es zu einem Fehler.

```
ActiveCell.SmartTags.Add( _
    "urn:schemas-microsoft-com:office:smartsheet#stockticker")
```

SpellingOptions

Excel

[Application](#) ↗

Das Objekt gibt Auskunft über diverse Einstellungen der Rechtschreibprüfung. Beispielsweise enthält [DictLang](#) den Sprachcode (1033 für Englisch). [IgnoreCaps](#) gibt an, ob auch Wörter in Großbuchstaben überprüft werden sollen etc.

Speech Excel[Application](#) ↗

Das Objekt steuert die automatische Sprachausgabe von Zellen (Eigenschaft [SpeakCellOnEnter](#)) und ermöglicht es mit der [Speak](#)-Methode, englische Sätze über die Soundkarte auszugeben: [Application.Speech.Speak "I love Excel"](#). Beachten Sie, dass das Objekt nur bei der englischen Excel-Version genutzt werden kann. Bei der deutschen Version tritt ein Fehler 1004 auf ([Anwendungs- oder objektdefinierter Fehler](#)).

SpinButton

MS-Forms

[UserForm.Controls](#) ↗

Mit dem Drehfeld kann ein Zahlenwert ([Value](#)) innerhalb eines vorgegebenen Bereichs ([Min](#) bis [Max](#)) eingestellt werden.

StdFont StdOLE[objekt.Font](#) ↗

Das Objekt dient zur internen Darstellung von Zeichensätzen ([Font](#)-Eigenschaft vieler Objekte). Dieselbe Funktion erfüllt auch das Objekt [NewFont](#) der MS-Forms-Bibliothek.

StdPicture

StdOLE

[Image.Picture](#) ↗

Das Objekt dient zur internen Darstellung von Bildern (Bitmaps, [Picture](#)-Eigenschaft bei diversen MS-Forms-Steuerelementen). Die StdOLE-Bibliothek stellt auch die beiden Funktionen [Load](#)- und [SavePicture](#) zur Verfügung, mit denen Bitmap-Dateien geladen bzw. gespeichert werden können.

Style Excel[Workbook.Styles\(..\)](#) ↗
[Range](#) ↗[.Interior](#) ↘ [Interior](#)
[.Borders](#) ↘ [Borders](#)
[.Font](#) ↘ [Font](#)

Formatvorlagen sind vordefinierte Formate, die zur raschen Formatierung von Zellen bzw. Zellbereichen verwendet werden können. Formatvorlagen gehören zu den Daten einer Arbeitsmappe und werden mit ihr gespeichert. Um einen Zellbereich mit einer

Formatvorlage zu gestalten, muss lediglich der *Range*-Eigenschaft *Style* die entsprechende Vorlage zugewiesen werden.

Formatvorlagen enthalten Informationen über Schriftart, Farben und Muster (Eigenschaft *Interior*), Umrandung der Zelle (Methode *Borders* für die vier Linien links, rechts, oben und unten), Zahlenformat (*NumberFormat*), Textausrichtung (*Orientation*, *Vertical*- und *HorizontalAlignment*, *WrapText*) und Zellschutz (*Locked*). Die sechs Eigenschaften *IncludeAlignment*, *-Pattern* etc. geben an, welche der sechs Teilformate durch die Formatvorlage verändert werden sollen. Sie können also auch Formate definieren, die nur Schriftart und Zahlenformat verändern und die restlichen Formatinformationen des Bereichs unverändert lassen.

Styles Excel

[Workbook ↗](#)

(*index* oder *name*) ↘ *Style*

Das Auflistungsobjekt verweist auf die in einer Arbeitsmappe definierten Formatvorlagen. Über die Methoden *Add* bzw. *Merge* werden weitere Formatvorlagen in die Liste aufgenommen.

Tab Excel

[Worksheet ↗](#)

Das Objekt ermöglicht es, die Hintergrundfarbe des Reiters eines Tabellenblatts zu verändern. Die folgende Anweisung stellt als Hintergrundfarbe Rot ein.

```
ActiveSheet.Tab.Color = RGB(255, 0, 0)
```

Beachten Sie, dass das Objekt keinen Zugriff auf den Text des Reiters gibt. Diesen können Sie mit *Worksheet.Name* lesen bzw. verändern.

TextBox MS-Forms

[UserForm.Controls ↗](#)

.Font ↘ *NewFont*

Das Textfeld ermöglicht die Eingabe (auch mehrzeiligen) Texts in eigenen Dialogen. Die wichtigste Eigenschaft lautet naturgemäß *Text*.

TextEffectFormat Excel

[Shape/ShapeRange.TextEffect ↗](#)

Das Objekt beschreibt Merkmale eines WordArt-Objekts (*Shape*-Objekt mit *Type=mso-TextEffect*).

TextFrame Excel

[Shape/ShapeRange.TextEffect ↗](#)

.Characters ↘ *Characters*

Das Objekt beschreibt den Textbereich eines AutoForm-Objekts (*Shape*-Objekt mit *Type=msoAutoShape*). Der Zugriff auf den eigentlichen Text erfolgt durch das *Characters*-Objekt, das eine individuelle Formatierung jedes einzelnen Buchstabens ermöglicht. Die Textausrichtung wird durch *Horizontal*- und *VerticalAlignment* eingestellt, der Rand zum AutoForm-Objekt durch *MarginLeft*, *-Top*, *-Right*, und *-Bottom*.

TextStream Scripting

[File.OpenAsStream ↗](#)

Das Objekt ermöglicht das Lesen bzw. Schreiben einer Textdatei. Die wichtigsten Methoden sind *Read* und *ReadLine* zum Lesen einiger Zeichen bzw. einer ganzen Zeile sowie *Write* und *WriteLine* zum Schreiben einer Zeichenkette bzw. einer ganzen Zeile.

ThreeDFormat

Excel

[Shape/ShapeRange.ThreeD](#) ↗ [.ExtrusionColor](#) ↘ [ColorFormat](#)

Das Objekt beschreibt das 3D-Aussehen von *Shape*-Objekten. Durch geeignete Einstellungen können die an sich flachen *Shape*-Objekte dreidimensional erweitert werden.

TickLabels

Excel

[Axis](#) ↗ [.Font](#) ↘ [Font](#)

Das Objekt beschreibt, wie die Teilstriche auf der Koordinatenachse eines Diagramms beschriftet werden sollen. Charakteristische Eigenschaften und Methoden sind *Orientation*, *Font*, *NumberFormat* und *NumberFormatLinked* (*True*, wenn die Zahlenformate aus der Tabelle übernommen werden sollen).

Das Objekt hat weder Einfluss auf Ort und Inhalt der Beschriftung noch auf die Anzahl der Beschriftungspunkte. Diese Details werden durch *Axis*-Eigenschaften gesteuert, und zwar durch *TickLabelSpacing* und *-Position* sowie durch *TickMarkSpacing* (für x-Achsen) oder *MajorUnit* (für y-Achsen).

ToggleButton

MS-Forms

[UserForm.Controls](#) ↗ [.Picture](#) ↘ [StdPicture](#)

Der Umschaltbutton ist eine Variante zum gewöhnlichen Button. Die Besonderheit besteht darin, dass der Button nicht automatisch, sondern erst durch ein nochmaliges Anklicken wieder zurückspringt.

TreeviewControl

Excel

[CubeField](#) ↗

Das Objekt verweist auf das hierarchische Listenfeld, das bei OLAP-Pivotfeldern zur Auswahl der angezeigten Hierarchieebenen und -details eingesetzt wird. Über die Eigenschaft *Drilled* kann die Sichtbarkeit der dem *CubeField* entsprechenden Pivotdaten gesteuert werden. Der Eigenschaft wird dazu ein zweidimensionales *Array* übergeben (siehe Beispiel in der Hilfe bzw. Makroaufzeichnung).

Trendline

Excel

[Series.Trendlines\(..\)](#) ↗ [.Border](#) ↘ [Border](#)
[.DataLabel](#) ↘ [DataLabel](#)

Das Objekt beschreibt Trend-, Näherungs- oder Ausgleichskurven in Diagrammen. Trendlinien sind einzelnen Datenreihen zugeordnet und können nur bei einigen zweidimensionalen Diagrammtypen angezeigt werden. *Type* bestimmt den Typ der Trendlinie (z. B. *xlPolynomial*, *xlLogarithmic*). Bei Ausgleichskurven bestimmt *Period* die Anzahl der Datenpunkte, über die gemittelt wird. Bei polynomischen Näherungskurven bestimmt die Eigenschaft *Order* die Ordnung des Polynoms (2 bis 6). *Forward* und *Backward* bestimmen, wie viele Perioden die Kurve über den vorhandenen Datenbereich hinaus in die Zukunft oder in die Vergangenheit gezeichnet wird (zur Trendabschätzung).

Die optische Gestaltung der Kurve erfolgt über die Subobjekte *DataLabel* und *Border*. *DisplayEquation* und *DisplayRSquared* bestimmen, ob die Formel der Kurve und ein Koeffizient für das Ausmaß der erzielten Annäherung an die Daten in einem Textfeld angezeigt werden soll.

Trendlines		Excel
Series ↗	(index oder name) ↘ Trendline	

Das Auflistungsobjekt verweist auf die Trend-, Näherungs- oder Ausgleichskurven einer Datenreihe. Siehe [Trendline](#).

UpBars		Excel
ChartGroup ↗	.Interior ↘ Interior .Border ↘ Border	

Das Objekt beschreibt das Aussehen von positiven Abweichungsbalken zwischen zwei Datenreihen eines Liniendiagramms. Details finden Sie bei [DownBars](#) (für negative Abweichungsbalken).

UsedObjects		Excel
Application ↗	(index) ↘ Object	

Die Auflistung verweist auf alle von Excel zurzeit verwalteten Basisobjekte. Dazu zählen z.B. geladene Arbeitsmappen ([Workbook](#)-Objekte), darin enthaltene Tabellenblätter ([Worksheet](#)), Diagramme und ActiveX-Komponenten.

UserAccess		Excel
UserAccessList(n) ↗ UserAccessList.Add(...) ↗ AllowEditRange.Users(n) ↗		

Das Objekt enthält in der Eigenschaft [Name](#) den Namen eines Computerbenutzers, der einen Zellbereich eines geschützten Tabellenblatts ohne Passwort verändern darf. Der Name muss in der Form "[computername\loginname](#)" angegeben sein. Außerdem muss [AllowEdit=True](#) gelten.

UserAccessList		Excel
AllowEditRange.Users ↗	(index) ↘ UserAccess	

Die Auflistung enthält alle Benutzer, die einen Zellbereich innerhalb eines geschützten Tabellenblatts ohne Passwortangabe verändern dürfen.

UserForm		MS-Forms
	.ActiveControl ↘ steuerelement .Controls ↘ Controls .steuerelementname ↘ steuerelement	

Das [UserForm](#)-Objekt ist das Basisobjekt für alle MS-Forms-Dialoge. Es stellt eine Menge Eigenschaften zur Gestaltung des Dialogs zur Verfügung. [StartupPosition](#) beeinflusst den Ort, an dem der Dialog erscheint. [Picture](#) kann eine Bitmap enthalten, die als Dialoghintergrund angezeigt wird. Mit [Zoom](#) kann der gesamte Dialoginhalt verkleinert oder vergrößert werden.

Über die Auflistung [Controls](#) können alle enthaltenen Steuerelemente angesprochen werden. (Im Programmcode wird aber zumeist direkt der Name der jeweiligen Steuerelemente verwendet.) [ActiveControl](#) verweist auf das gerade aktive Steuerelement (mit Tastaturfokus).

Validation

Excel

[Range ↗](#)

Das Objekt beschreibt, welche Eingaben in einem Zellbereich zulässig sind (interaktive Einstellung durch [DATEN|DATENÜBERPRÜFUNG](#)). Der zulässige Zahlentyp (beispielsweise Ganzzahl, Datum etc.) wird durch [Type](#) eingestellt. Der Zahlenbereich kann durch Grenzwerte in [Formula1](#) und [-2](#) eingeschränkt werden. Dazu muss mit [Operator](#) ein Vergleichsoperator angegeben werden (etwa [xlBetween](#), [xlGreater](#) etc.). [Formula1](#) und [-2](#) können sowohl Werte als auch Zelladressen enthalten. (Im zweiten Fall werden die Grenzwerte aus den Zellen gelesen.) [InputMessage](#) gibt einen kurzen Infotext zur Eingabe, [ErrorMessage](#) enthält die Fehlermeldung, die bei Missachtung der Regeln angezeigt wird.

VBComponent[s]

VBE

[VBProject ↗](#)

[.CodeModule](#) ↘ [CodeModule](#)
[.Properties\(..\)](#) ↘ [Property](#)

[VBComponents](#) enthält eine Auflistung aller Komponenten eines VBA-Projekts (also Module und Dialoge, [Type](#)-Eigenschaft). Der Codeanteil einer Komponente wird via [CodeModule](#) angesprochen, die Eigenschaften via [Properties](#). Wenn die Komponente einen Designer hat (damit ist etwa der Dialogeditor gemeint), kann mit [HasOpenDesigner](#) festgestellt werden, ob dieser aktiv ist.

VBE VBE[Application ↗](#)

[.CodePanels\(..\)](#) ↘ [CodePane](#)
[.CommandBars\(..\)](#) ↘ [CommandBars](#)
[.VBProjects\(..\)](#) ↘ [VBProject](#)
[.Window\(..\)](#) ↘ [Window](#)

VBE ist das Startobjekt der gleichnamigen Bibliothek zur Programmierung der VBA-Entwicklungsumgebung. (Im Objektkatalog wird diese Bibliothek mit VBIDE bezeichnet.) Die Hauptaufgabe von **VBE** besteht darin, den Zugriff auf die untergeordneten Objekte zu ermöglichen (siehe Syntaxbox). Außerdem können über [ActiveCodePane](#), [ActiveVBProject](#), [ActiveWindow](#) und [SelectedVBComponent](#) die gerade aktiven Komponenten der Entwicklungsumgebung angesprochen werden.

VBProject[s]

VBE

[VBE.ActiveVBProject ↗](#)
[VBE ↗](#)

[.References\(..\)](#) ↘ [Reference](#)
[.VBComponents\(..\)](#) ↘ [VBComponent](#)

[VBProjects](#) enthält Verweise auf alle zurzeit geladenen Projekte (Excel-Dateien und Add-Ins). Über die Eigenschaften [VBComponents](#) und [References](#) können die einzelnen Komponenten (gemeint sind Module und Dialoge) und die Verweise auf externe Objektbibliotheken angesprochen werden.

VPageBreak

Excel

[Worksheet.VPageBreaks\(..\) ↗](#)[.Location](#) ↘ [Range](#)

Das Objekt bezeichnet einen vertikalen Seitenumbruch in einem Tabellenblatt.

VPageBreaks

Excel

[Worksheet ↗](#)[\(index oder name\)](#) ↘ [VPageBreak](#)

Die Auflistung ermöglicht den Zugriff auf alle manuellen Seitenumbrüche im Tabellenblatt.

Walls Excel[Chart ↗](#)[.Interior](#) ↘ [Interior](#)
[.Border](#) ↘ [Border](#)

Das Objekt beschreibt die beiden Seitenwände eines 3D-Diagramms. Durch die Subobjekte [Interior](#) und [Border](#) können Farbe und Umrandung dieser Wände eingestellt werden, allerdings nur für beide Wände gemeinsam. Siehe auch [Floor](#) für den Boden eines 3D-Diagramms.

Watch Excel[Application.Watches\(n\)](#) ↗
[Watches.Add](#) ↗[.Source](#) ↘ [Range](#)

Das Objekt beschreibt ein Überwachungsobjekt. Damit können einzelne Zellen von unterschiedlichen Tabellenblättern in einem Überwachungsfenster angezeigt werden (interaktiv über [FORMELN | ÜBERWACHUNGSFENSTER](#)). Die Eigenschaft [Source](#) gibt an, welche Zelle überwacht wird. Es fehlt aber merkwürdigerweise eine Eigenschaft mit der Information, in welchem Tabellenblatt sich die Zelle befindet.

Watches Excel[Application](#) ↗[\(index\)](#) ↘ [Watch](#)

Die Auflistung verweist auf die Überwachungsobjekte von Excel. Beachten Sie, dass [index](#) bei [Watches](#) im Gegensatz zu fast allen anderen Excel-Aufzählungen mit 0 beginnt. [Watches\(0\)](#) verweist also auf das erste Überwachungsobjekt! [Watches](#) kann auch nicht zur Bildung von *For-Each*-Schleifen verwendet werden (*For Each w In Watches*). Stattdessen müssen Sie eine Schleife für den Index von 0 bis [Count-1](#) bilden.

WebOptions

Excel

[Workbook](#) ↗

Die Eigenschaften dieses Objekts steuern die Parameter des HTML-Exports einer Excel-Datei. (Für die globalen Exporteinstellungen von Excel müssen Sie stattdessen die Eigenschaften von [DefaultWebOptions](#) einstellen.)

Window Excel[Workbook.Windows\(..\)](#) ↗
[Application.ActiveWindow](#) ↗[.Panes](#) ↘ [Pane](#)
[.SelectedSheets](#) ↘ [Sheets](#)

Das Objekt verweist auf ein Fenster innerhalb von Excel. Fenster werden mit [Activate](#) zum aktiven Fenster gemacht. Die Methoden [ActivateNext](#) bzw. [-Previous](#) wechseln zum nächsten bzw. vorangegangenen Fenster. Durch [NewWindow](#) und [Close](#) werden neue Fenster erzeugt bzw. geschlossen. Mit [Split](#) und [FreezePanels](#) werden Fenster in mehrere Ausschnitte geteilt.

Die Eigenschaft [Visible](#) bestimmt, ob ein Fenster ausgeblendet (unsichtbar) ist oder nicht. Über [WindowState](#) wird die Größe des Fensters (Icon, normal, maximiert) eingestellt. [Caption](#) enthält den Fenstertitel. Die Eigenschaften [ScrollColumn](#) bzw. [ScrollRow](#) bestimmen die Nummer der ersten sichtbaren Spalte/Zeile. [DisplayGridlines](#) gibt an, ob im Fenster Gitterlinien dargestellt werden.

Die [Window](#)-Eigenschaften bzw. Methoden [ActivePane](#), [ActiveSheet](#), [ActiveChart](#), [ActiveCell](#), [Panels](#), [Selection](#) und [SelectedSheets](#) verweisen auf diverse untergeordnete Objekte.

Windows		Excel
Workbook ↗	(index oder aufschrift) ↘ Window	

Das Auflistungsobjekt verweist auf die Fenster einer Arbeitsmappe. Der Zugriff erfolgt durch die Angabe der (internen) Fensternummer oder des Fenstertitels (*Caption*-Eigenschaft des *Window*-Objekts).

Window[s]		VBE
VBE ↗		

Die Auflistung *Windows* und das daraus abgeleitete *Window*-Objekt ermöglichen den Zugriff auf die Fenster der VBA-Entwicklungsumgebung. Bei Namenskonflikten mit den Excel-*Window[s]*-Objekten müssen Sie *VBE* voranstellen, also etwa *Dim w As VBE.Window*.

Workbook		Excel
Application.Workbooks(..) ↗	.Charts ↘ Chart	
Application.ActiveWorkbook ↗	.DialogSheets ↘ DialogSheet	
	.Modules ↘ Module	
	.Worksheets ↘ Worksheet	

Das Objekt *Workbook* beschreibt eine Excel-Datei. Über die oben angeführten Methoden kann auf die Blätter der Arbeitsmappe zugegriffen werden, wobei nach den vier möglichen Blatttypen differenziert wird. Die Methode *Sheets* ermöglicht den Zugriff auf alle Blätter der Arbeitsmappe (unabhängig von ihrem Typ). *Windows* verweist auf die zur Arbeitsmappe gehörigen Fenster.

Die Arbeitsmappe kann mit den Methoden *Save* bzw. *SaveAs* gespeichert werden. *Close* schließt die Arbeitsmappe, wobei diese zuvor durch die Angabe eines optionalen Parameters ebenfalls gespeichert werden kann.

Workbooks		Excel
Application ↗	(index oder name) ↘ Workbook	

Workbooks enthält die Liste aller geladenen Arbeitsmappen (auch der unsichtbaren, deren Fenster ausgeblendet sind). Der Verweis auf einzelne Arbeitsmappen kann entweder durch die Angabe der Indexnummer oder über den Dateinamen (ohne Pfad) erfolgen. Über *Add* kann eine neue, leere Arbeitsmappe angegeben werden. *Open* lädt eine bereits vorhandene Arbeitsmappe.

Worksheet		Excel
Workbook.Worksheets ↗	.Range ↘ Range	
Application.ActiveSheet ↗	.Cells ↘ Range	
	.XxxObjekts ↘ XxxObjekt	

Tabellenblätter gehören neben Dialog-, Diagramm- und Modulblättern zu den Blättern einer Arbeitsmappe. Zahllose Methoden und Eigenschaften verweisen einerseits auf Zellbereiche (*Range*, *Rows*, *Columns*, *Cells*) und andererseits auf eingelagerte Steuerelemente, Zeichnungsobjekte, Diagramme, Pivottabellen etc.

Der Datenaustausch über die Zwischenablage erfolgt über die Methoden *Copy*, *Paste* und *PasteSpecial*. *Calculate* führt zu einer manuellen Neuberechnung der Tabelle (nur erforderlich, wenn die *Application*-Eigenschaft *Calculation* auf *xlManual* steht). Über die Eigenschaften *ConsolidationFunction*, *-Options* und *-Sources* werden die Details eines Konsolidierungsvorgangs eingestellt, der anschließend mit *Range.Consolidate* ausgeführt werden kann.

WorksheetFunctions

Excel

[Application ↗](#)

Über das Objekt können Excel-Tabellenfunktionen im VBA-Programmcode verwendet werden.

Worksheets

Excel

[Workbook ↗](#)[\(index oder name\) ↘ Worksheet](#)

Das Auflistungsobjekt verweist auf die Tabellenblätter einer Arbeitsmappe. Über die Methoden [Add](#) und [Copy](#) können neue Tabellenblätter erzeugt werden. [Select](#) macht ein Tabellenblatt zum aktiven Blatt.

XmlDataBinding

Excel

[XmlMap.DataBinding ↗](#)

Das Objekt beschreibt den Datenursprung von XML-Daten. Die einzig relevante Eigenschaft lautet [SourceUrl](#) und enthält eine Zeichenkette der Datenquelle (also z.B. einen Dateinamen oder eine Internetadresse). Mit der Methode [Refresh](#) werden die Daten neu eingelesen.

XmlMap Excel[ListObject ↗](#)[.DataBinding ↘ XmlDatabinding](#)[Workbook.XmlMaps\(n\) ↗](#)[.RootElementNamespace ↘ XmlNamespace](#)[XPath.Map ↗](#)[.Schemas\(n\) ↘ XmlSchema](#)

Das Objekt beschreibt die gemeinsamen Eigenschaften (Schema, Namensraum) sowie die Import-/Exportoptionen einer XML-Datenquelle. Das Objekt dient als Verbindungspunkt zwischen den XML-Daten (Objekte [XmlDatabinding](#), [XmlSchema](#), [XmlNamespace](#)) und den Zellen im Tabellenblatt, in welche die Daten importiert werden (Objekt [XPath](#)).

Mit der Methode [Import](#) wird der Datenimport durchgeführt, wobei als Parameter die Datenquelle angegeben werden muss (XML-Dateiname oder Internetadresse). Das ist deswegen erforderlich, weil mit einem [XmlMap](#)-Objekt auch mehrere XML-Dateien importiert werden können, beispielsweise um die Daten aneinanderzufügen. [ImportXml](#) funktioniert analog, liest die Daten aber aus einer als Parameter übergebenen XML-Zeichenkette.

Sofern Excel in der Lage ist, die Daten wieder zu speichern (Eigenschaft [IsExportable](#)), kann dieser Vorgang mit den Methoden [Export](#) oder [ExportXml](#) durchgeführt werden. Die Methode [Delete](#) löscht das [XmlMap](#)-Objekt.

XmlMaps

Excel

[Workbook ↗](#)[\(index oder name\) ↘ XmlMap](#)

Die Auflistung verweist auf alle [XmlMap](#)-Objekte der Excel-Datei. Solche Objekte werden automatisch durch einen XML-Importvorgang (z.B. [ActiveWorkbook.XmlImport](#)) erzeugt. Explizit können Sie ein neues [XmlMap](#)-Objekt auch durch [Add](#) hinzufügen. Dabei müssen Sie als Parameter das XML-Schema übergeben, und zwar wahlweise als XML-Zeichenkette, durch einen Dateinamen oder durch eine Internetadresse.

XmlNamespace

Excel

[Workbook.XmlNamespaces\(n\)](#) ↗
[XmlMap.RootElementNamespace](#) ↗
[XmlSchema.NameSpace](#) ↗

Das Objekt definiert ein Präfix. Das ist eine Zeichenkette, die den Namen von XML-Elementen vorangestellt werden kann. Präfixe ermöglichen es, zwischen unterschiedlichen, aber gleichnamigen XML-Elementen zu unterscheiden (z.B. `<prefix1:name>` und `<prefix2:name>`). Alle XML-Elemente mit einem bestimmten Präfix werden als Namensraum bezeichnet.

Das Objekt kennt nur zwei relevante Eigenschaften: *Prefix* enthält die Präfixzeichenkette und *Uri* die Adresse der Definition des Namensraums. (URI steht für *uniform resource identifier*. Die *Uri*-Zeichenkette sieht wie ein Dateiname oder eine Internetadresse aus.)

XmlNamespaces

Excel

[Workbook](#) ↗ [\(index\)](#) ↘ [XmlNamespace](#)

Die Auflistung verweist auf alle *XmlNamespace*-Objekte der Excel-Datei.

XmlSchema

Excel

[XmlSchema\(..\)](#) ↗ [.Namespace](#) ↘ [XmlNamespace](#)

Das Objekt beschreibt den Aufbau (die Struktur) von XML-Daten. Die Beschreibung erfolgt ebenfalls in XML und ist als Zeichenkette in der Eigenschaft *XML* enthalten. Optional können dem Schema Zusatzinformationen über den Namensraum beigelegt sein; diese befinden sich in einem eigenen Objekt (*XmlNamespace*).

XmlSchemas

Excel

[XmlMap.Schemas](#) ↗ [\(index\)](#) ↘ [XmlSchema](#)

Die Auflistung verweist auf alle Schemas eines *XmlMap*-Objekts. (In vielen Fällen gibt es für jedes *XmlMap*-Objekt nur ein Schema. Wenn mehrere Schemas vorliegen, dann verweist *Schemas(1)* auf das Primärschema für das Wurzelement.)

XPath

Excel

[ListColumn](#) ↗ [.Map](#) ↘ [XmlMap](#)
[Range](#) ↗

Das Objekt stellt die Zuordnung zwischen einzelnen Elementen aus einem XML-Datenstrom (*XmlMap*) und einem Zellbereich bzw. einer Spalte einer Liste her. Welche XML-Elemente importiert werden sollen, bestimmt ein XPath-Suchausdruck, der an die Methode *SetValue* übergeben wird. Der eigentliche Import findet aber erst statt, wenn für das zugrunde liegende *XmlMap*-Objekt die *Import*-Methode ausgeführt wird.

Die Eigenschaften *Value* (für den XPath-Suchausdruck), *Map* und *Repeating* von *XPath* enthalten die an *SetValue* übergebenen Parameter. Die Eigenschaften können nur gelesen werden. Veränderungen müssen mit *SetValue* durchgeführt werden.

Um die Zuordnung zwischen einem *Range/ListColumn*-Objekt und den XML-Daten zu löschen, muss für das betreffende *XPath*-Objekt die Methode *Clear* ausgeführt werden.