

VBA2007

Autorin: Andrea Weikert

Überarbeitete Ausgabe vom 3. September 2008

© HERDT-Verlag für Bildungsmedien GmbH, Bodenheim

Internet: [www.herdt.com](http://www.herdt.com)

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Herausgebers reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Die in diesem Buch und den abgebildeten bzw. zum Download angebotenen Dateien genannten Personen und Organisationen, Adress- und Telekommunikationsangaben, Bankverbindungen etc. sind frei erfunden. Übereinstimmungen oder Ähnlichkeiten mit lebenden oder toten Personen sowie tatsächlich existierenden Organisationen oder Informationen sind unbeabsichtigt und rein zufällig.

Die Bildungsmedien des HERDT-Verlags enthalten Links bzw. Verweise auf Internetseiten anderer Anbieter. Auf Inhalt und Gestaltung dieser Angebote hat der HERDT-Verlag keinerlei Einfluss. Hierfür sind alleine die jeweiligen Anbieter verantwortlich.

## Microsoft **VBA-Programmierung**

Integrierte Lösungen mit Office 2007

VBA2007

**Mit VBA programmieren**

<b>1 Über dieses Buch .....</b>	<b>4</b>
1.1 Aufbau und Konventionen .....	4
1.2 Verwendete Dateien und Software .....	5
1.3 Empfohlene Vorkenntnisse .....	5
<b>2 Office programmieren.....</b>	<b>6</b>
2.1 Einsatzgebiete von VBA .....	6
2.2 Integrierte Lösungen .....	7
2.3 Einsatzgebiete der Office-Anwendungen .....	8
2.4 Ein einfaches Beispiel für eine integrierte Lösung.....	9
2.5 Makros mit der Symbolleiste für den Schnellzugriff verknüpfen.....	11
2.6 Schnellübersicht.....	12
<b>3 Die VBA-Entwicklungsumgebung .....</b>	<b>14</b>
3.1 VBA-Entwicklungsumgebung verwenden... ..	14
3.2 Bestandteile der Entwicklungsumgebung... ..	16
3.3 Der Projekt-Explorer.....	16
3.4 Das Eigenschaftenfenster .....	18
3.5 Das Code-Fenster benutzen .....	18
3.6 Eingabehilfen für Visual-Basic-Anweisungen .....	20
3.7 Automatische Arbeiten bei Anweisungsende .....	22
3.8 Mit Prozeduren arbeiten.....	23
3.9 Tipps zum Arbeiten mit dem Code-Fenster .....	24
3.10 Mit dem Direktfenster arbeiten.....	26
3.11 Das Lokalfenster .....	27
3.12 Schnellübersicht.....	27
3.13 Übung .....	28
<b>4 Die Sprachelemente von VBA.....</b>	<b>30</b>
4.1 Module verwenden .....	30
4.2 Mit Prozeduren programmieren.....	32
4.3 Sub-Prozeduren .....	33
4.4 Property-Prozeduren .....	35
4.5 Function-Prozeduren .....	35
4.6 Variablen verwenden .....	36
4.7 Konstanten verwenden.....	38
4.8 Datentypen von VBA.....	39
4.9 Operatoren .....	42
4.10 Programmablauf mit Kontrollstrukturen steuern .....	43
4.11 Schnellübersicht.....	47
4.12 Übung .....	48

**5 Objektorientierte Programmierung mit VBA..... 50**

5.1 Was sind Objekte?.....	50
5.2 Die Objekthierarchien (Objektmodelle).....	51
5.3 Eigenschaften und Methoden .....	51
5.4 Objektvariablen.....	54
5.5 Auflistungen.....	55
5.6 Den Objektkatalog verwenden .....	57
5.7 Schnellübersicht.....	59
5.8 Übung .....	59

**6 Gemeinsam genutzte VBA-Elemente ... 60**

6.1 VBA-Elemente für Office-Anwendungen....	60
6.2 Anwendungsfenster programmieren.....	61
6.3 Dokumentenfenster programmieren .....	62
6.4 Dateien suchen .....	65
6.5 Dialogfenster zur Datei- und Ordnerauswahl.....	66
6.6 Mit dem Dateisystem programmieren .....	67
6.7 Eingabedialoge und Meldungsfenster .....	68
6.8 Schnellübersicht.....	72
6.9 Übung .....	72

**Objektmodelle der Office-Anwendungen****7 Programmieren der Office-Anwendungen..... 74**

7.1 Das Word-Objektmodell .....	74
7.2 Mit Dokumenten arbeiten .....	75
7.3 Auf Dokumenteninhalte zugreifen .....	76
7.4 Das Excel-Objektmodell .....	78
7.5 Mit Arbeitsmappen und Tabellenblättern arbeiten .....	79
7.6 Zugriff auf Zellen und Zellbereiche.....	80
7.7 Das Access-Objektmodell .....	83
7.8 Mit Access-Datenbanken arbeiten.....	83
7.9 Zugriff auf Steuerelemente in Formularen und Berichten .....	84
7.10 Das PowerPoint-Objektmodell .....	86
7.11 Mit Präsentationen und Folien arbeiten .....	86
7.12 Zugriff auf den Folieninhalt .....	87
7.13 Das Outlook-Objektmodell .....	88
7.14 Auf Ordner zugreifen .....	89
7.15 Auf Elemente zugreifen.....	89
7.16 Schnellübersicht.....	91
7.17 Übung .....	91

**8 Kommunikation zwischen Office-Anwendungen..... 92**

8.1 Integrierte Office-Automatisierung .....	92
8.2 Technische Grundlagen.....	93

- 8.3 Verweis auf eine Objektbibliothek erstellen.....94
- 8.4 Deklarieren von Objektvariablen für die Automation.....95
- 8.5 Objektinstanzen erzeugen .....96
- 8.6 Objekt schließen und Arbeitsspeicher freigeben.....98
- 8.7 Meldungen des Automationsobjekts unterdrücken..... 100
- 8.8 Schnellübersicht ..... 102
- 8.9 Übung ..... 103

**Benutzerdefinierte Dialoge und Datenbankzugriffe**

**9 Benutzerdefinierte Dialoge verwenden..... 104**

- 9.1 Kommunikation mit dem Anwender ..... 104
- 9.2 UserForm-Dialoge erstellen ..... 105
- 9.3 UserForm-Dialoge verwenden und programmieren ..... 109
- 9.4 Steuerelemente programmieren ..... 112
- 9.5 Schnellübersicht ..... 113
- 9.6 Übung ..... 114

**10 Datenbankzugriff in Office-Anwendungen ..... 116**

- 10.1 Objektmodelle für den Zugriff auf Datenbanken ..... 116
- 10.2 Datenbankzugriff mit ADO ..... 117
- 10.3 Verbindung zu einer Datenquelle herstellen..... 118
- 10.4 Datensatzgruppe auswählen und öffnen ..... 120
- 10.5 Datensätze auslesen..... 121
- 10.6 Zugriff auf Datenfelder ..... 122
- 10.7 Datensätze hinzufügen und bearbeiten .... 125
- 10.8 Datenbankzugriff mit DAO ..... 126
- 10.9 Schnellübersicht ..... 128
- 10.10 Übung ..... 129

**Integrierte Lösungen**

**11 Integrierte Lösungen mit Word ..... 132**

- 11.1 Möglichkeiten für integrierte Lösungen.... 132
- 11.2 Diagrammfunktion von Excel nutzen ..... 133
- 11.3 Adressen aus Access übernehmen ..... 135
- 11.4 Schnellübersicht ..... 137
- 11.5 Übung ..... 138

**12 Integrierte Lösungen mit Excel ..... 140**

- 12.1 Möglichkeiten für integrierte Lösungen.... 140
- 12.2 Druckfunktion von Word nutzen ..... 141
- 12.3 Daten aus Outlook und Access importieren ..... 144
- 12.4 Schnellübersicht ..... 146
- 12.5 Übung ..... 147

**13 Integrierte Lösungen mit Access ..... 148**

- 13.1 Möglichkeiten für integrierte Lösungen.... 148
- 13.2 Datenbank zur Dokumentenverwaltung... 149
- 13.3 Dokumente in die Datenbank einlesen ..... 150
- 13.4 Dokumente suchen ..... 152
- 13.5 Vorschaufunktion für Word-Dokumente ..... 153
- 13.6 Vorschaufunktion für Excel-Dokumente ..... 155
- 13.7 Dokumente in der Datenbank archivieren..... 156
- 13.8 Liste der gefundenen Dokumente löschen ..... 157
- 13.9 Versteckt geöffnete Word- bzw. Excel-Anwendung schließen..... 157
- 13.10 Dokumente verwalten ..... 158
- 13.11 Dokumente öffnen ..... 161
- 13.12 Schnellübersicht ..... 162
- 13.13 Übung ..... 162

**14 Integrierte Lösungen mit PowerPoint..... 164**

- 14.1 Möglichkeiten für integrierte Lösungen.... 164
- 14.2 Die Präsentationserstellung..... 165
- 14.3 Bearbeiten der Titelfolie ..... 166
- 14.4 Erstellen einer Textfolie mit Word-Unterstützung ..... 168
- 14.5 Erstellen einer Diagrammfolie mit Excel-Unterstützung ..... 171
- 14.6 Suchfunktion für Excel-Arbeitsmappen ..... 174
- 14.7 Schnellübersicht ..... 175
- 14.8 Übung ..... 176

**15 Integrierte Lösungen mit Outlook .... 178**

- 15.1 Möglichkeiten für integrierte Lösungen.... 178
- 15.2 Notizen in Word erstellen ..... 179
- 15.3 Serien-E-Mails mit Word und Outlook versenden..... 180
- 15.4 Schnellübersicht ..... 187
- 15.5 Übung ..... 187

**Stichwortverzeichnis ..... 190**

## 5 Objektorientierte Programmierung mit VBA

### In diesem Kapitel erfahren Sie

- ▶ wie VBA das Programmieren mit Objekten unterstützt
- ▶ wie Sie auf Eigenschaften und Methoden zugreifen
- ▶ wie Sie mit Auflistungen arbeiten
- ▶ wie der Objektkatalog verwendet wird

### Voraussetzungen

- ✓ Grundlagen der VBA-Programmierung
- ✓ Kenntnisse im Umgang mit der VBA-Entwicklungsumgebung

### 5.1 Was sind Objekte?

In der objektorientierten Programmierung stellt ein **Objekt** eine abgeschlossene Einheit dar, die bestimmte Daten sowie die Prozeduren (innerhalb von Objekten als "Methoden" bezeichnet) zum Verarbeiten dieser Daten enthält. Was sich im Inneren des Objekts befindet, bleibt dadurch weitgehend verborgen und geschützt. Auf diese Weise lässt sich eine bestimmte Funktionalität kapseln. Objekte (engl. Objects) kommunizieren mit anderen Objekten über festgelegte Schnittstellen, das sind vor allem die Methoden des Objekts.

Alle Elemente in Office, wie z. B. Dokumente, Tabellen, Arbeitsblätter, Formulare und Steuerelemente, werden als Objekte betrachtet. VBA erschließt Ihnen damit die Möglichkeit, auf alle Objekte der einzelnen Office-Anwendungen von einem VBA-Programm aus zuzugreifen, um z. B. Daten auszulesen oder zu verändern.

Die Office-Anwendungen Word, Excel usw. stellen selbst ebenfalls Objekte dar. Auf diese Weise wird Ihnen der Zugriff auf diese Anwendungen bei der Programmierung integrierter Lösungen vereinfacht.

Die im Programmcode befindliche Definition für bestimmte Objekte wird als **Klasse** (engl. Class) bezeichnet. VBA stellt für die Programmierung viele vordefinierte Klassen zur Verfügung, z. B. die Klasse `Application`. Von einer Klasse können beliebig viele Objekte (auch **Instanzen** genannt) gebildet werden. Wenn Sie z. B. ein neues Dokument anlegen, erzeugt Word eine neue Instanz der Klasse `Document` und fügt es der Auflistung `Documents` der entsprechenden Instanz des `Application`-Objekts hinzu.

Jedes Objekt verfügt über eine Anzahl individueller **Eigenschaften** (engl. Properties) und **Methoden** (engl. Methods). Da die Eigenschaften und Methoden zu der Klasse gehören, werden sie auch als **Member** (Mitglieder) der Klasse bezeichnet. Zu den Eigenschaften gehören z. B. Größe, Position, Farbe und Name eines Objekts. Mit den Methoden können Sie das Verhalten eines Objekts steuern. Ein Word-Dokument besitzt andere Eigenschaften als beispielsweise ein Access-Formular.

### Wichtige Office-Objekte

Microsoft Office besteht aus einer sehr großen Anzahl von Objekten, auf die Sie in VBA Zugriff haben. Die folgende Tabelle gibt einen Überblick über einige Office-Objekte:

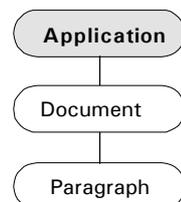
<code>Application</code>	Bezeichnet das Objekt der jeweiligen Anwendung (Word, Excel usw.)
<code>Document</code>	Klasse für Word-Dokumente
<code>Range</code>	Ein Textbereich in einem Word-Dokument bzw. ein Zellenbereich in einer Excel-Tabelle
<code>Workbook</code>	Eine Excel-Arbeitsmappe

Form	Klasse für Access-Formulare
Table	Klasse für Tabellen in Word, Excel oder Access
Presentation	Klasse für eine PowerPoint-Präsentation

## 5.2 Die Objekthierarchien (Objektmodelle)

Objekte existieren in der Regel nicht unabhängig voneinander. Sie sind in Hierarchiestufen angeordnet und in einem Objektmodell zusammengefasst. Für die Programmierung dieser Objekte ist es nicht nur wichtig, ihre Eigenschaften und Methoden zu kennen, sondern auch ihre jeweilige Hierarchiestufe im Objektmodell, da sie nur über diese Stufen angesprochen werden können.

Die höchste Ebene der Office-Objekte bildet das Objekt `Application`, das die jeweilige Anwendung darstellt. Alle anderen Objekte sind diesem untergeordnet und befinden sich dementsprechend in der Hierarchie unterhalb dieses Objektes. Das `Application`-Objekt wird automatisch beim Öffnen einer Anwendung erzeugt.



Teil der Word-Objekthierarchie

### Beispiel: *Kap05-Objekte.docm*, Modul `ThisDocument`

Im folgenden Beispiel sollen mithilfe von VBA zwei Absätze eines Word-Dokuments einmal fett und einmal kursiv formatiert werden.

```

Sub AbsatzFormatieren()
① Application.Documents(1).Paragraphs(2).Range.Bold = True
② Paragraphs(3).Range.Italic = True
End Sub
  
```

- ① Ausgehend vom `Application`-Objekt (der Anwendung Word) wird im ersten geöffneten Dokument der gesamte zweite Absatz fett formatiert.
- ② Hier wird der dritte Absatz kursiv formatiert. Da sich die Anweisungen in einem Dokumentenmodul befinden, erkennt VBA selbstständig das aktuelle Dokument als Bezugsobjekt.

VBA kann am Kontext oft erkennen, welches Objekt gemeint ist, z. B. wenn Sie in einem Dokumentenmodul programmieren. Sie müssen in diesen Fällen nicht die gesamte Hierarchie benennen. VBA stellt zusätzlich das Schlüsselwort `Me` zur Verfügung, welches sich immer auf das gerade aktuelle Objekt bezieht.



## 5.3 Eigenschaften und Methoden

Jedes Objekt verfügt über Eigenschaften und Methoden, über die Sie das Objekt beeinflussen und steuern können. Eigenschaften sind Variablen des Objekts, die Sie auslesen und verändern können. Als Methoden werden die Prozeduren eines Objekts bezeichnet.

### Eigenschaften von Objekten

Die meisten Objekte verfügen über eine sehr große Anzahl Eigenschaften. Innerhalb von Objekten wird der Zugriff auf Eigenschaften meist über Property-Prozeduren (`Get`, `Let`, `Set`) geregelt bzw. kontrolliert, um die Kapselung der Daten sicherzustellen.

Die Entwicklungsumgebung unterstützt Sie bei der Programmierung, indem sie die verfügbaren Eigenschaften und Methoden eines Objekts automatisch auflistet.

```

Sub AbsatzFormatieren()
Application.
End Sub
  
```

Automatisch aufgelistete Eigenschaften und Methoden



Auch der Objektkatalog (Taste **F2**) hilft Ihnen, den Überblick über die verfügbaren Eigenschaften und Methoden eines Objekts zu behalten.

### Wichtige Eigenschaften

Name	Über den Namen wird das Objekt angesprochen und referenziert. Abhängig vom Objekt kann die Eigenschaft zur Laufzeit nur gelesen oder gelesen und geändert werden.
Caption	Beschriftung, z. B. von Schaltflächen, Formularen oder Fenstern
Selection	Bezeichnet die Markierung, beispielsweise einer Textstelle, Grafik usw.
Value	Ermöglicht den Zugriff auf den Inhalt eines Objekts, z. B. den Eintrag in einem Textfeld oder den Inhalt einer Tabellenzelle

### Beispiel: Kap05-Eigenschaften.xlms, Modul Tabelle1

Im folgenden Beispiel wird einigen Zellen eines Excel-Arbeitsblattes mithilfe von VBA ein neuer Wert zugewiesen. Zusätzlich erhält eine Zelle eine neue Hintergrundfarbe, und der Wert einer Zelle wird ausgelesen.

```

Sub Eigenschaften ()
①   Range("A1:C1").Value = 123
②   Range("C1").Interior.Color = RGB(255, 255, 0)
③   MsgBox Range("A1").Value
End Sub

```

- ① Da sich die Anweisungen in einem Tabellenmodul befinden, kann auf die Angabe der übergeordneten Objekthierarchie verzichtet werden. Hier wird mittels der Eigenschaft `Value` in die Zellen A1 bis C1 der Wert 123 eingetragen.
- ② Über die Eigenschaft `Color` wird der Zelle C1 eine gelbe Hintergrundfarbe zugewiesen. Das Objekt `Interior` bezeichnet dabei den Innenbereich der Zelle. Die VBA-Funktion `RGB` erzeugt den benötigten Farbcode.
- ③ Um den Wert der Zelle A1 zu erhalten, wird die Eigenschaft `Value` ausgelesen und das Ergebnis in einem Meldungsfenster angezeigt.

### Syntax des Zugriffs auf Objekteigenschaften

```
Variablenname = Objektverweis.Eigenschaftsname
```

oder

```
Objektverweis.Eigenschaftsname = Ausdruck
```

- ✓ `Objektverweis` ist der Name eines Objektes oder ein Ausdruck, der das entsprechende Objekt bestimmt.
- ✓ Nach einem Punkt folgt der Name der Eigenschaft. Falls eine hierarchische Objektfolge vorliegt, werden die einzelnen Objekte ebenfalls durch Punkte getrennt.
- ✓ Eine Objekteigenschaft kann als Variable betrachtet werden, die Sie auslesen oder der Sie einen Wert zuweisen können.



Eigenschaften besitzen wie Variablen einen bestimmten Datentyp. So liefern viele Eigenschaften, z. B. `Caption`, eine Zeichenkette. Eigenschaften, die eine Auswahl zwischen `True` und `False` erlauben, sind vom Datentyp `Boolean`. Daneben gibt es einige Eigenschaften, bei denen eine Reihe von zulässigen Einstellungen möglich sind. Diese Eigenschaften besitzen den Datentyp `Integer`. Dabei ist jeder Einstellung ein Zahlenwert zugeordnet. Ein Beispiel für eine solche Eigenschaft ist die Eigenschaft `STANDARDANSICHT` (`DefaultView`) eines Access-Formulars. Möglich sind die Einstellungen `EINZELNES FORMULAR`, `ENDLOSFORMULAR` und `DATENBLATTANSICHT`. Bei der Programmierung mit VBA werden dafür intern die Zahlenwerte 0, 1 und 2 verwendet.

## Methoden von Objekten

Methoden sind Prozeduren, die einem Objekt eine Anweisung erteilen, damit es eine Aufgabe erledigt. Sie werden wie normale `Sub`-Prozeduren oder `Function`-Prozeduren verwendet.

### Beispiel: *Kap05-Eigenschaften.xlms*, Modul `Tabelle1`

Der von Daten verwendete Bereich eines Arbeitsblattes soll mit VBA-Befehlen markiert und anschließend in die Zwischenablage kopiert werden.

```
Sub Zwischenablage ()
  ① ActiveSheet.UsedRange.Select
  ② Selection.Copy
End Sub
```

- ① Die Eigenschaft `UsedRange` des `ActiveSheet`-Objekts verweist auf den verwendeten Bereich des aktiven Tabellenblatts. Der Aufruf der Methode `Select` markiert diesen Bereich.
- ② Die Methode `Copy` des `Selection`-Objekts kopiert den markierten Zellbereich in die Zwischenablage.

### Syntax des Zugriffs auf Objektmethoden

- ✓ Objektverweis ① ist der Name eines Objektes oder eine Zeichenfolge, die das Objekt bestimmt. Innerhalb des Objekts kann der Verweis auf das Objekt selbst durch das Schlüsselwort `Me` ② hergestellt werden oder auch entfallen ③.
- ✓ Nach einem Punkt folgt der Name der Methode.
- ✓ Methoden sind Prozeduren des Objekts, die auch Werte zurückliefern oder Argumente erwarten können.

```
Objektverweis.Methodenname ①
Me.Methodenname ②
Methodenname ③
```

### Die `With`-Anweisung zum Zugriff auf Eigenschaften und Methoden

VBA ermöglicht eine vereinfachte Form, um nacheinander auf Eigenschaften und Methoden des gleichen Objekts zuzugreifen. Dazu dient die Anweisung `With`.

### Beispiel: *Kap05-Eigenschaften.xlms*, Modul `Beispiele`

Mithilfe der `With`-Anweisung werden einige Eigenschaften der Arbeitsmappe ausgelesen und im Direktfenster angezeigt.

```
Sub EigenschaftenAnzeige ()
  ① With Workbooks (1)
  ②   Debug.Print .Name
      Debug.Print .FullName
      Debug.Print .Path
  End With
End Sub
```

- ① Mit dem Schlüsselwort `With` wird die Anweisung eingeleitet. Danach folgt die Angabe des Objekts, auf das sich die folgenden Eigenschaften und Methoden beziehen.
- ② Um auf die Eigenschaften des Objekts zuzugreifen, wird der Punkt gefolgt vom Namen angegeben. Hier werden einige Eigenschaften im Direktfenster angezeigt.

### Syntax der With-Anweisung

- ✓ Nach der Angabe des Schlüsselwortes `With` folgt die Angabe des Objektnamens.
- ✓ Innerhalb der `With`-Struktur entfällt beim Zugriff auf Eigenschaften und Methoden die Angabe des Objekts. Nur der Punkt, gefolgt vom Namen der Eigenschaft oder Methode, muss angegeben werden.
- ✓ Soll innerhalb der `With`-Struktur vereinzelt auf ein anderes Objekt zugegriffen werden, darf der Objektverweis dort nicht entfallen.

```
With Objektverweis
    ...
    .Eigenschafts-/Methodenname
    ...
End With
```



`With`-Strukturen können beliebig verschachtelt werden. Beachten Sie jedoch, dass die Übersicht gewahrt bleibt. Eigenschaften und Methoden beziehen sich immer auf die zuletzt eingeleitete `With`-Struktur.

## 5.4 Objektvariablen

Um den Zugriff auf Objekte zu vereinfachen, können Sie Objektvariablen einsetzen. Dies sind spezielle Variablen, die Verweise auf Objekte speichern. Objektvariablen repräsentieren einen Verweis auf ein Objekt, z. B. auf den selektierten Bereich eines Dokuments.



Objektvariablen vereinfachen besonders den Zugriff auf Objekte, die weiter unten in der Objekthierarchie angeordnet sind. Der Aufbau langer Objektreferenzen wird dadurch vermieden, und der Programmcode wird übersichtlicher.

### Beispiel: *Kap05-Objekte.docm*, Modul `ThisDocument`

Mithilfe einer VBA-Prozedur soll für den ersten Absatz eines Word-Dokuments bestimmt werden, wie viele Zeichen, Wörter und Sätze er enthält. Das Ergebnis wird in einem Meldungsfenster angezeigt.

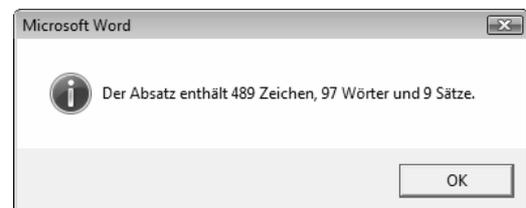
```
Sub AbsatzEigenschaften()
①   Dim Absatz As Range
    Dim Meldung As String
②   Set Absatz = Application.Selection.Paragraphs(1).Range
③   Meldung = "Der Absatz enthält " & _
        Absatz.ComputeStatistics(wdStatisticCharacters) & " Zeichen, "
    Meldung = Meldung & Absatz.Words.Count & " Wörter und " & _
        Absatz.ComputeStatistics(wdStatisticWords)
④   Meldung = Meldung & " Satz/Sätze."
    MsgBox Meldung, vbOKOnly + vbInformation
End Sub
```

① An dieser Stelle wird die Objektvariable `Absatz` deklariert. Als Objekttyp wird `Range` angegeben; die Variable kann damit Verweise auf Bereiche im Dokument aufnehmen. Word verwendet das `Range`-Objekt, um auf Teile des Texts, z. B. einen Absatz, zugreifen zu können.

② Im Unterschied zu normalen Variablen wird der Objektvariablen an dieser Stelle ein Objektverweis mithilfe von `Set` zugewiesen. `Absatz` verweist danach auf den Text des Absatzes, in dem sich gerade der Cursor befindet (`Selection`-Objekt).

③ Die Objektvariable wird wie eine Objektangabe verwendet. Hier werden verschiedene Eigenschaften ausgelesen und in der Zeichenkette `Meldung` gespeichert.

④ Die Informationen werden in einem Meldungsfenster angezeigt.



Informationen über den aktuellen Absatz

## Syntax der Deklaration von Objektvariablen

```
Dim Objektvariablenname As Objekttyp
```

- ✓ Die Deklaration wird wie bei normalen Variablen mit der Anweisung `Dim` eingeleitet. Danach folgt der Name der Objektvariablen.
- ✓ Mit der Anweisung `As` wird der Datentyp des Objekts bestimmt. Jeder verfügbare Objektname kommt dafür in Betracht. Der Datentyp legt fest, welche Verweise später gespeichert werden können.
- ✓ Es gelten alle Regeln, die auch für normale Variablen Gültigkeit haben, beispielsweise die Einschränkungen bei der Wahl des Variablennamens.

## Syntax der Set-Anweisung

Um einer Objektvariablen ein Objekt zuzuweisen, auf das sie verweisen soll, muss die `Set`-Anweisung verwendet werden. Eine einfache Wertzuweisung ist in diesem Fall nicht möglich.

```
Set Variablenname = [New ]Objekt
```

- ✓ Die Zuweisung eines Objekts an eine Objektvariable erfolgt mithilfe der `Set`-Anweisung. Nach dem Schlüsselwort `Set` folgen der Name der Objektvariablen sowie die Zuweisung der Objektreferenz durch das Gleichheitszeichen.
- ✓ Um ein neues Objekt einer Klasse zu erzeugen, werden hinter dem Gleichheitszeichen das Schlüsselwort `New` und der Name der Klasse angegeben.

Wird der Verweis auf das Objekt nicht mehr benötigt, kann er gelöscht werden, indem der Objektvariablen das Schlüsselwort `Nothing` zugewiesen wird.

```
Set Objektvariable = Nothing
```

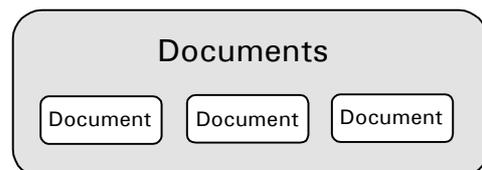
## 5.5 Auflistungen

### Was sind Auflistungen?

Einige der in der Objekthierarchie vorkommenden Objekte sind Auflistungen. Auflistungen bestehen aus einer Gruppe gleicher Objekte, d. h., es existieren mehrere Objekte dieses Typs. Sie stellen eine Art Container für mehrere Einzelobjekte dar.

Die Objektauflistung `Documents` in Word enthält beispielsweise alle geöffneten Dokumente. Jedes einzelne Dokument wird dabei innerhalb dieser Auflistung durch ein eigenes `Document`-Objekt mit seinen Eigenschaften und Methoden repräsentiert. Objektauflistungen werden in VBA durch den Namen des Einzelobjekts in der Plural-Form ("s" am Ende) gekennzeichnet, z. B. `Documents`, `Forms`, `Controls`, `Worksheets` usw.

Auf die Eigenschaften und Methoden eines einzelnen Objektes (z. B. eines Formulars oder Steuerelements) können Sie nur über das Einzelobjekt zugreifen. Jedoch besitzt auch die Auflistung selbst einige Eigenschaften, z. B. `Count` (ermittelt die Anzahl der in der Auflistung enthaltenen Objekte).



*Document-Objekte in der Documents-Auflistung in Word*

### Auf einzelne Objekte einer Auflistung zugreifen

VBA behandelt Auflistungen ähnlich wie Arrays. Auf einzelne Objekte der Auflistung greifen Sie daher über einen Indexwert zu.

Objektauflistungen besitzen eine dynamische Länge, und die Zählung des Indexwerts beginnt bei 1.

### Beispiel: *Kap05-Eigenschaften.xlsm*, Modul *Beispiele*

In einer Schleife werden die Namen aller Arbeitsblätter einer Excel-Arbeitsmappe ausgelesen und im Direktfenster angezeigt.

```

Sub Arbeitsblätter()
    Dim Anzahl As Integer, Index As Integer
    ① Anzahl = Worksheets.Count
    ② For Index = 1 To Anzahl
    ③     Debug.Print Worksheets(Index).Name
        Next
    End Sub

```

- ① Die Eigenschaft `Count` der Auflistung `Worksheets` gibt die Anzahl der Arbeitsblätter an und bestimmt damit den größten Indexwert für den Zugriff auf die Auflistung. Der Wert wird in der Variablen `Anzahl` gespeichert.
- ② In einer `For-Next`-Schleife werden die einzelnen Elemente der Auflistung `Worksheets` vom Indexwert 1 bis `Anzahl` durchlaufen.
- ③ Mittels der Variablen `Index` wird ein bestimmtes `Worksheet`-Objekt der Auflistung angesprochen und die Eigenschaft `Name` im Direktfenster angezeigt.



Ausgabe im Direktfenster



Das Direktfenster können Sie benutzen, um einfache Ausgaben zu testen. Falls es noch nicht angezeigt wird, öffnen Sie es über den Menüpunkt ANSICHT - DIREKTFENSTER bzw. den Tastaturbefehl **STRG** **G**.

### Syntax des Zugriffs auf einzelne Objekte einer Auflistung

```
Auflistung(Index).Eigenschaftsname
```

oder

```
Auflistung(Index).Methodenname
```

- ✓ Der Zugriff auf einzelne Objekte einer Auflistung erfolgt ähnlich wie bei einem Array.
- ✓ Nach dem Namen der Auflistung folgt in runden Klammern ein Indexwert. Der Index beginnt bei dem Wert 1.
- ✓ Durch einen Punkt getrennt werden die Eigenschaften und Methoden des Einzelobjekts angesprochen, beispielsweise des `Document`-Objekts bei einer `Documents`-Auflistung.

### Namen beim Zugriff auf Auflistungen verwenden

Außer dem Zugriff über Indexwerte kann bei vielen Auflistungen auch der Name des gewünschten Objekts verwendet werden, beispielsweise der Name eines Word-Dokuments oder eines Excel-Arbeitsblatts.

### Beispiel für den Zugriff auf Objekte einer Auflistung über Namen

```

Documents("Markierung.docm")
Workbooks("Mappel.xlsm").Worksheets("Tabelle2").Range("A1:B5")
Forms("Mitarbeiter")

```



Der Name ist in den genannten Beispielen die Bezeichnung, die bei dem gewünschten Objekt in der Titelleiste der Anwendung angezeigt wird. Falls es sich um einen Dateinamen handelt, muss die Dateinamenserweiterung, z. B. `*.docm`, angegeben werden.



Wenn Sie die Namen der in einer Auflistung enthaltenen Objekte nicht kennen, können Sie sich diese in einer `For-Each`-Schleife ausgeben lassen (vgl. nachfolgendes Beispiel).

## Auf alle Objekte einer Auflistung nacheinander zugreifen

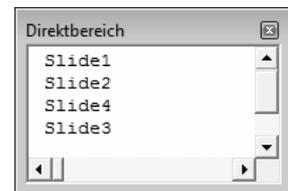
Eine besondere Form der `For-Next`-Schleife in VBA ist die `For-Each-Next`-Schleife. Mit ihr können Sie jedes Element, das in einem Array oder einer Auflistung (Liste von Objekten) vorkommt, ansprechen. Dazu müssen Sie die Anzahl der Elemente nicht kennen.

### Beispiel: *Kap05-Computerkurs.pptm*, Modul *Beispiele*

In einer Sub-Prozedur werden die Foliennamen der aktuellen PowerPoint-Präsentation ermittelt und im Direktfenster ausgegeben.

```
Sub FolienNamen()
  ① Dim Folie As Slide
  ② For Each Folie In ActivePresentation.Slides
  ③   Debug.Print Folie.Name
  Next
End Sub
```

- ① Die Objektvariable `Folie`, die ein Objekt vom Typ `Slide` (PowerPoint-Folie) referenzieren kann, wird deklariert.
- ② Nacheinander wird bei jedem Durchlauf der `For-Each-Next`-Schleife der Variablen `Folie` ein neues Objekt der Auflistung `Slides` zugewiesen. Das Objekt `ActivePresentation` verweist auf die aktuelle PowerPoint-Präsentation.
- ③ Der Name der Folie wird mit der Eigenschaft `Name` ermittelt und im Direktfenster angezeigt.



Ausgabe im Direktfenster

## Syntax der `For-Each-Next`-Schleife

```
For Each Objektvariable In Auflistung
  ...
Next
```

- ✓ Mit den Schlüsselwörtern `For Each` wird die Schleife eingeleitet. Danach folgt eine Objektvariable, der nacheinander die einzelnen Objekte der Auflistung zugewiesen werden.
- ✓ Nach dem Schlüsselwort `In` folgt der Name der Auflistung, die durchlaufen werden soll.
- ✓ Innerhalb der Schleife kann über die Objektvariable auf die Methoden und Eigenschaften des Einzelobjekts zugegriffen werden.

Der Objektvariablen muss bei der Deklaration der Objekttyp der Einzelobjekte der Auflistung oder der allgemeine Objekttyp `Object` zugewiesen werden.



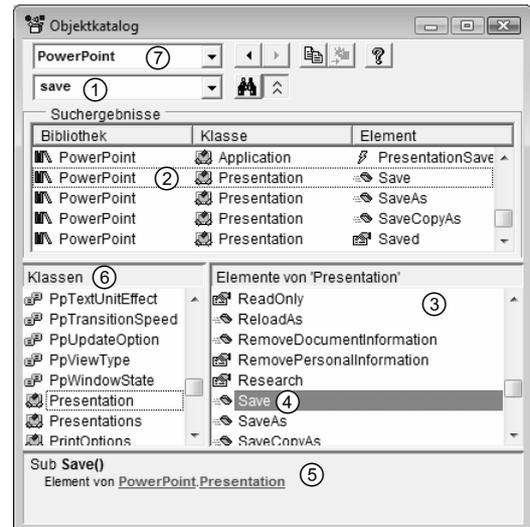
## 5.6 Den Objektkatalog verwenden

VBA verfügt über eine Vielzahl Objekte, die wiederum viele Eigenschaften und Methoden zur Verfügung stellen. Der Objektkatalog der VBA-Entwicklungsumgebung hilft Ihnen, den Überblick zu behalten.

- ▶ Wählen Sie den Menüpunkt **ANSICHT - OBJEKT-KATALOG**, um den Objektkatalog anzuzeigen.  
Alternativen: **F2** oder 
- ▶ Mit **F7** wechseln Sie wieder zur Anzeige des vorher aktiven Code-Fensters.

Sie können, im Objektkatalog der Hierarchie folgend, ein bestimmtes Objekt nachschlagen. Um schnell die Eigenschaften und Methoden eines Objekts anzuzeigen, verwenden Sie die Suchfunktion.

- ▶ Tragen Sie im Listenfeld ① den gewünschten Suchbegriff ein, z. B. save.
- ▶ Betätigen Sie nun die Taste **RETURN** , um den eingegebenen Begriff zu suchen.  
Alternative:   
Die Auflistung zeigt alle gefundenen Begriffe.
- ▶ Wählen Sie einen davon aus ②.  
Im Listenfeld ③ sind nun alle Eigenschaften und Methoden des gewählten Objekts aufgeführt. Wählen Sie einen Eintrag aus ④, werden im unteren Textbereich ⑤ eine kurze Erklärung und die Syntax des Aufrufs angezeigt.
- ▶ Im Listenfeld KLASSEN ⑥ können Sie auch ohne vorherige Suche eine Klasse nachschlagen.



Der Objektkatalog nach einer Suche



Standardmäßig werden im Objektkatalog die aktuell unterstützten Elemente angezeigt. Um z. B. Elemente anzuzeigen, die in Vorgängerversionen gültig waren und jetzt verborgen sind, können Sie im Listenfeld ⑦ den Kontextmenüpunkt **VERBORGENE ELEMENTE ANZEIGEN** wählen.

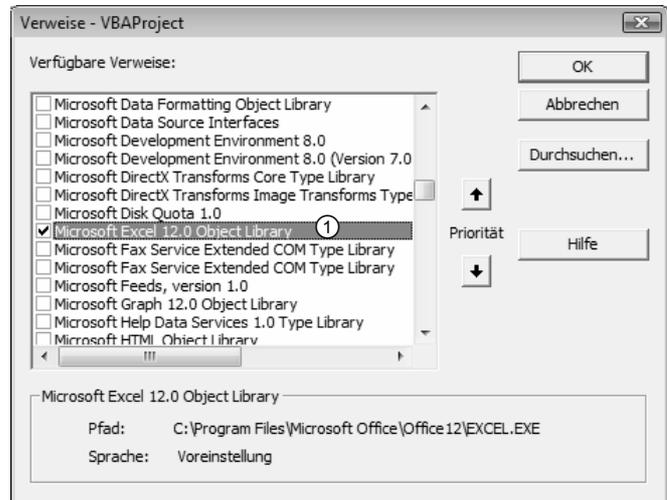
## Zusätzliche Objektbibliotheken in der Entwicklungsumgebung aktivieren

Die VBA-Entwicklungsumgebung kennt standardmäßig nur die Objekte von VBA und der Office-Anwendung, in der gerade programmiert wird. Damit die Programmierhilfen, wie z. B. der Objektkatalog und das automatische Auflisten, auch mit den Objekten der anderen Anwendungen funktionieren, müssen Sie die entsprechenden Bibliotheken einbinden.

- ▶ Wählen Sie in der VBA-Entwicklungsumgebung den Menüpunkt **EXTRAS - VERWEISE**.
- ▶ Aktivieren Sie im Dialogfenster **VERWEISE** im Listenfeld ① den Eintrag für die benötigte Bibliothek.
- ▶ Bestätigen Sie mit **OK**.

Folgende Bibliotheken enthalten die wichtigsten Konstanten und zugehörigen Objekte:

- ✓ Visual Basic for Applications
- ✓ Microsoft Word 12.0 Object Library
- ✓ Microsoft Excel 12.0 Object Library
- ✓ Microsoft Access 12.0 Object Library
- ✓ Microsoft PowerPoint 12.0 Object Library
- ✓ Microsoft Outlook 12.0 Object Library
- ✓ Microsoft Publisher 12.0 Object Library
- ✓ Microsoft Office 12.0 Object Library



Zusätzliche Bibliotheken einbinden

## 5.7 Schnellübersicht

Was bedeutet ...	
Klasse (Objektyp)	Abstrakte Beschreibung (Bauanleitung) für Objekte desselben Typs
Objekt	Instanz (Ausprägung) einer Klasse
Eigenschaft	Daten des Objekts können über Eigenschaften angesprochen und verändert werden.
Methode	Prozeduren des Objekts, die mit den Daten des Objekts arbeiten können
Auflistung	Zusammenfassung mehrerer Objekte desselben Typs
Objektvariable	Variable, um Verweise auf Objekte zu speichern

Sie möchten ...	
eine Objekteigenschaft auslesen	Variablenname = Objektverweis.Eigenschaftsname
eine Objekteigenschaft verändern	Objektverweis.Eigenschaftsname = Ausdruck
eine Objektmethode aufrufen	Objektverweis.Methodename [Argument1, Argument2, ...]
auf mehrere Eigenschaften und Methoden eines Objekts zugreifen	<b>With ... End With</b>
die Anzahl der Objekte einer Auflistung bestimmen	Auflistung.Count
auf ein einzelnes Objekt einer Auflistung zugreifen	Auflistung(Index) Auflistung("Name")
nacheinander auf alle Objekte einer Auflistung zugreifen	<b>For Each ... In ... Next</b>
einer Objektvariablen ein Objekt zuweisen	<b>Set</b> Objektvariable = Objektverweis

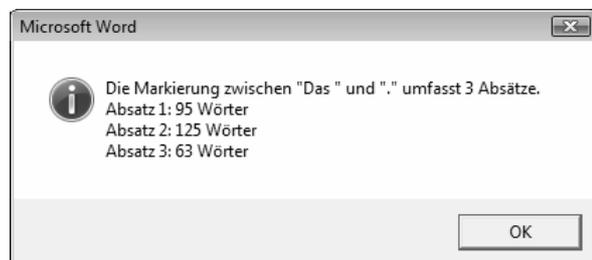
## 5.8 Übung

### Mit Objekten arbeiten

Übungsdatei: *Wordobjekte.docm*

Ergebnisdatei: *Wordobjekte-E.docm*

- ① Erstellen Sie im Dokumentenmodul der Datei *Wordobjekte.docm* eine Prozedur `MarkierungInformationen`. Die Prozedur soll anzeigen, wie viele Absätze die aktuelle Markierung im Dokument umfasst. Für jeden Absatz soll zusätzlich die Wörterzahl angezeigt werden.
- ② Erstellen Sie eine Objektvariable, die einen Verweis auf ein `Selection`-Objekt aufnehmen kann, und weisen Sie ihr die aktuelle Markierung zu.
- ③ Ermitteln Sie die Anzahl der Absätze in der Markierung.
- ④ Programmieren Sie eine `For-Next`-Schleife, die nacheinander alle Absätze der Markierung durchläuft und die Wörterzahl anzeigt.
- ⑤ Sammeln Sie alle Ausgaben in einer Zeichenkette, und zeigen Sie diese in einem Meldungsfenster an (vgl. Abb.).



Meldungsfenster mit den Absatzinformationen