Kapitel 5

Überblick über die Arbeit mit Objekten

In diesem Kapitel:

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Im ersten Teil dieses Buches haben Sie in den jeweiligen Kapiteln – von der Entwicklungsumgebung bis hin zu den Grundlagen von VBA – einen ersten Eindruck vom Programmieren in Word erhalten. Im zweiten Teil versuchen wir Ihnen nun das Objektmodell von Word sowie die Dokument- bzw. Programmereignisse näher zu bringen.

Von den einzelnen Microsoft Office-Anwendungen weist Word das umfangreichste Objektmodell auf. Und es liegt wohl an der Komplexität dieses Objektmodells, dass jeder, der zum ersten Mal Word automatisieren muss, mit Startschwierigkeiten zu kämpfen hat. Schon vielfach wurden wir mit der Aussage konfrontiert: »Ich weiß nicht, welche Objekte in Word vorhanden sind.«

Die passende Antwort liegt auf der Hand. Doch hilft sie dem Fragesteller selten weiter, sondern verwirrt ihn zusätzlich. Denn jeder Buchstabe, jedes Wort, jeder Absatz, jeder Abschnitt, die Kopf- und Fußzeilen, die Fußnoten, jede Tabelle und jede Grafik, alle Dokumente und deren Dokumentvorlagen sind Objekte – ja, sogar Word selbst ist ein Objekt. Eigentlich ganz logisch.

Vom Anfänger hingegen werden oft ganz andere Objekte gesucht: eine bestimmte Zeile oder eine bestimmte Seite. Doch diese Objekte gibt es nicht. Bei Word handelt es sich um eine Textverarbeitung. Und deren oberstes Gebot ist der so genannte Fließtext. Die gesuchten Objekte können gar nicht existieren, weil eine bestimmte Zeile nur eine Momentaufnahme darstellt. Das Ändern eines einzigen Wortes wirkt sich auf den ganzen Text aus, so dass die gewünschten Objekte bereits neu definiert werden müssten. Die Nicht-Existenz dieser Objekte ist also ebenfalls ganz logisch.

Arbeiten mit Objekten

Die nachstehenden Abschnitte vermitteln zunächst einen kurzen Einblick in die Arbeit mit Objekten. Das eigentliche Objektmodell wird anschließend in Kapitel 6 ausführlich erörtert.

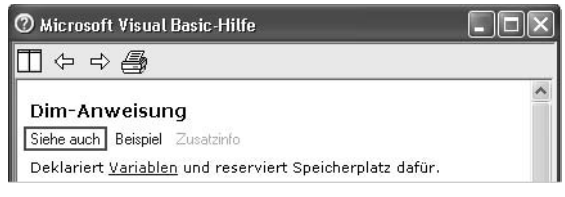
Das gesuchte Objekt aufspüren

Word selber stellt uns einige Hilfsmittel zur Verfügung, um ein gesuchtes Objekt schnell aufzuspüren. Kombiniert mit einer gewissen Portion Neugierde haben sie bisher noch jedes gut versteckte Objekt ans Tageslicht befördert und dürfen hier keinesfalls unerwähnt bleiben.

Online-Hilfe

- Die *Online-Hilfe* wurde bereits in Kapitel 2 im Abschnitt »Die VBA-Hilfe – eine versteckte Schatzkammer« vorgestellt. Aus unserer Sicht ist der wichtigste Hinweis auf jeder Hilfeseite die Verknüpfung *Siehe auch*. Denn hier geht es zu den verwandten Themen, und wenn die Neugierde erst einmal geweckt wurde, dann werden viele interessante Sachen entdeckt.

Abbildg. 5.1 Lesen Sie die verwandten Themen, um ein Objekt aufzuspüren und besser kennen zu lernen



Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)


Objekt-katalog

- Ein weiteres Hilfsmittel ist der *Objektkatalog*, der einen umfassenden Einblick in jede Software-Bibliothek freigibt und dessen Möglichkeiten schon in Kapitel 2 im Abschnitt»Wo alle Fäden zusammenlaufen: Der Objektkatalog« vorgestellt wurden.


Intelli-Sense

- Die *IntelliSense*-Funktion unterstützt den Programmierer während des Erstellens von Programmzeilen und legt gleichzeitig die »Unterobjekte« zu einem Objekt offen. Denn in vielen Fällen entspricht die Eigenschaft eines Objekts einem weiteren Objekt.


Makro-rekorder

- Das wohl wichtigste Hilfsmittel aber ist der *Makrorekorder*. Er wurde bereits in Kapitel 1 im Abschnitt »Den Makrorekorder einsetzen« vorgestellt. Im aktuellen Fall jedoch wird er nicht zum Generieren der benötigten Programmzeilen, sondern zum Aufspüren der entsprechenden Objekte verwendet.

Die Kombination dieser vier Hilfsmittel, der persönlichen Neugierde sowie der wachsenden Erfahrung im Umgang mit dem Objektmodell bringt jedes Objekt zum Vorschein.

Objekte als Variablen

Den Datentyp *Object* haben Sie in Kapitel 3 im Abschnitt »Variablen« bereits kennen gelernt. Doch statt eine allgemeine Variable vom Typ *Object* anzulegen, ist es sinnvoller, eine Variable des benötigten Typs zu deklarieren:

```
Dim rng_1 As Object      'nicht optimal
Dim rng_2 As Range       'besser
Dim rng_3 As Word.Range  'optimal
```

PROFITIPP

Wir Autoren empfehlen Ihnen, bei der Deklaration von Objektvariablen nebst der benötigten Klasse (beispielsweise `Range`) grundsätzlich die zugehörige Bibliothek (beispielsweise Word) zu nennen:

```
Dim rng As Word.Range
```

Die Gründe sind nahe liegend und Sie können sich bei konsequenter Anwendung viel Ärger ersparen:

- Die Bezeichnung einer Klasse muss nicht eindeutig sein. In Abhängigkeit der in das VBA-Projekt eingebundenen Bibliotheken kann die Klasse mehrmals vorkommen. Die Klasse `Range` beispielsweise ist sowohl in der Bibliothek von Word als auch in jener von Microsoft Excel definiert.
- Der Compiler verfügt über eine klare Anweisung, die angeforderte Klasse muss nicht innerhalb aller eingebundenen Bibliotheken zusammengesucht werden.
- Die Unterstützung durch IntelliSense funktioniert für alle Objekte richtig, denn ohne Definition der Bibliothek werden nur die Eigenschaften und Methoden des ersten Treffers in der Liste der eingebundenen Bibliotheken aufgelistet.

Neben dem zusätzlichen Aufwand, den die Deklaration dieser Objektvariablen mit sich bringt, steht der daraus resultierende Nutzen im Vordergrund:

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

- Die Programmzeilen werden kürzer, da das entsprechende Objekt direkt bearbeitet wird. Sie bleiben übersichtlicher und für den Programmierer besser lesbar. Gleichzeitig erfolgt eine indirekte Dokumentation des Programms:

```
tbl.Rows.Add
```

  anstelle von

```
ActiveDocument.Tables(1).Rows.Add
```

- Der Bezug zum Dokument wird eindeutig, denn zusammen mit der Zuweisung des Objekts an die Variable wird diese festgelegt und bleibt bis zu ihrer Freigabe bestehen.
- Nur bei deklarierten Objektvariablen erfolgt eine Unterstützung durch IntelliSense. Bei allgemeinen Variablen vom Typ `Object` steht sie nicht zur Verfügung, da das eigentliche Objekt erst während der Zuweisung, also zur Laufzeit des Programms, interpretiert wird.

HINWEIS Bevor eine Objektvariable einer bestimmten Klasse deklariert werden kann, muss dem VBA-Projekt ein so genannter *Verweis* auf die entsprechende Bibliothek hinzugefügt werden. Mehr zum Thema »Verweise auf externe Bibliotheken« erfahren Sie in Kapitel 8.

Zuweisen von Objektvariablen

Dim xyz
Set yxz =

Damit eine Objektvariable in einem Programmteil verwendet werden kann, muss sie zuerst deklariert und in einem zweiten Schritt mittels der `Set`-Anweisung einem gültigen Objekt zugewiesen werden:

```
Dim doc As Word.Document
Set doc = ActiveDocument
```

Die Zuweisung des Objekts an die Objektvariable setzt nur einen Verweis auf die Speicheradresse des entsprechenden Objekts.

In der Syntax der `Dim`-Anweisung kann das optionale Schlüsselwort `New` angegeben werden. Die Verwendung von `New` weist den Compiler an, ein neues Objekt implizit zu erstellen. Es wird eine neue Instanz auf das Objekt erstellt. (Damit die nachstehende Zeile erfolgreich getestet werden kann, muss vorab ein Verweis auf die »Microsoft Excel 11.0 Object Library« gesetzt werden.)

```
Dim xls As New Excel.Application
```

Es zeugt jedoch von einem unsauberen Programmierstil, wenn innerhalb der gleichen Programmzeile das Objekt deklariert und gleichzeitig eine neue Instanz desselben angelegt wird. Denn in diesem Fall werden Deklarations- und Programmzeilen vermischt und der Anweisung kommt eine doppelte Bedeutung zu. In Listing 5.1 wurden die Deklaration der Objektvariablen und das Anlegen einer neuen Instanz von Excel bewusst getrennt.

Innerhalb der ersten Zeilen der Prozedur wird die benötigte Objektvariable angelegt:

```
Dim xls As Excel.Application
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

In einem getrennten zweiten Schritt wird die neue Instanz auf das gewünschte Objekt, in diesem Fall auf Excel, erzeugt:

```
Set xls = CreateObject("excel.application")
```

Listing 5.1 Das Erzeugen der Objektvariablen und das Anlegen der Instanz wurden getrennt

```
Sub ExcelObjekt()
'Verweis auf »Mircosoft Excel 11.0 Object Library« nötig
  Dim xls As Excel.Application
  Dim xwb As Excel.Workbook
  Dim xws As Excel.Worksheet

  Set xls = CreateObject("excel.application")
  xls.Visible = True

  Set xwb = xls.Workbooks.Add
  Set xws = xwb.Sheets.Add

  xws.Range("A1").Formula = "Hallo Welt"
  xws.PrintOut
  xwb.Close SaveChanges:=False

  xls.Quit

  Set xws = Nothing
  Set xwb = Nothing
  Set xls = Nothing
End Sub
```


HINWEIS Im .NET Framework sieht man oft, dass ein Objekt in der gleichen Codezeile deklariert und ihm eine neue Instanz einer Klasse zugewiesen wird. Hier ist das möglich, weil die Instanz gleichzeitig ins Leben gerufen werden kann, anders als im klassischen VBA. Kann die neue Instanz aus irgendeinem Grund nicht angelegt werden, besteht auch hier das gleiche Problem und das Einsetzen des Schlüsselworts `New` muss in einer separaten Zeile erfolgen:

```
Dim xls as Excel.Application = New Excel.Application
```

In C#:

```
Excel.Application xls = new Excel.Application();
```

Standardeigenschaft von Objekten

Jedes Objekt aus dem Objektmodell von Word verfügt über eine Standardeigenschaft. Als Beispiel dient das `Range`-Objekt, für das standardmäßig die `Text`-Eigenschaft definiert ist.

Beim Erfassen der Programmzeilen kann somit die Zuweisung eines Textes an einen festgelegten Range mittels

```
rng = "Hallo Welt"
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

oder

```
rng.Text = "Hallo Welt"
```

erfolgen. Aus Sicht der Programmlogik besteht kein Unterschied zwischen den beiden Anweisungen. Egal welche der beide Zeilen zum Einsatz kommt, im Dokument wird an der gewünschten Stelle der Gruß »Hallo Welt« eingefügt.

Wir empfehlen jedoch, der Verlockung, die Standardeigenschaft zu verwenden, zu widerstehen und stattdessen jede Eigenschaft voll zu qualifizieren:

- Die Programmzeile ist nicht selbst erklärend, da nur geübte Entwickler die Standardeigenschaften aller Objekte kennen. Die Dokumentation der Programmlogik wird umso wichtiger.
- Es ist nicht gewährleistet, dass in allen und auch zukünftigen Programmversionen von Word die Standardeigenschaft eines Objekts immer die gleiche ist.
- Das Portieren des Programms nach C# wird bedeutend aufwändiger, denn in C# muss jede Eigenschaft voll qualifiziert werden; C# kennt keine Standardeigenschaften.

Freigeben von Objektvariablen

Set xyz = Nothing

Objektvariablen müssen grundsätzlich nach deren Verwendung wieder freigegeben werden. Mit dem Schlüsselwort `Nothing` wird veranlasst, dass die Verbindung einer Objektvariablen zum entsprechenden Objekt aufgehoben wird. Die Freigabe erfolgt explizit durch die Verwendung der `Set`-Anweisung oder implizit, nachdem die letzte Objektvariable den Gültigkeitsbereich verlässt und somit auf `Nothing` gesetzt wird:

```
Set xls = Nothing
```

HINWEIS Wir empfehlen, Objektvariablen, die auf andere Applikationen verweisen, grundsätzlich durch eine explizite Verwendung der `Set`-Anweisung freizugeben. Sie ersparen sich so die mühsame Suche nach Programmfehlern, die nur sporadisch und in Abhängigkeit Ihrer eigenen Applikation auftreten werden. Dies vor allem dann, wenn die Anwendung mehr als einmal in einer Sitzung ausgeführt wird.

Bei der Freigabe von Objektvariablen ist, wie in Listing 5.1 ersichtlich, darauf zu achten, dass diese grundsätzlich in der umgekehrten Reihenfolge, wie diese erzeugt wurden, freigegeben werden. Nur so ist sichergestellt, dass wirklich alle Verbindungen auf die betreffenden Speicheradressen entfernt und die reservierten System- und Speicherressourcen ebenfalls freigegeben werden:

```
Set xws = Nothing
Set xwb = Nothing
Set xls = Nothing
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Auflistung von Objekten

Beim Erforschen des Objektmodells von Word stößt der Anwender schnell auf eine Besonderheit. Von den meisten aufgeführten Objekten sind im Objektkatalog innerhalb der gleichen Bibliothek zwei Einträge vorhanden (beispielsweise `Document` und `Documents`, `Paragraph` und `Paragraphs` usw.).

Bei jenen Objekten, deren Bezeichnung im Singular steht (beispielsweise `Document`, `Paragraph` usw.) handelt es sich um ein einzelnes spezifisches Objekt.

Bei den anderen Objekten, deren Bezeichnung im Plural steht (beispielsweise `Documents`, `Paragraphs` usw.), handelt es sich um eine so genannte Auflistung eines spezifischen Objektes.

Ein einzelnes Objekt kann entweder vorhanden oder eben nicht mehr vorhanden sein (beispielsweise das Dokument wurde geschlossen, der Absatz wurde entfernt). Die Auflistung eines Objekts hingegen ist dynamisch und beinhaltet immer die Werte des aktuellen Zustands (beispielsweise die Auflistung aller geöffneten Dokumente, die Auflistung aller Absätze im Dokument).

Zugreifen auf ein spezifisches Objekt

Um auf ein spezifisches Objekt aus einer Auflistung heraus zuzugreifen, kann dieses mit seinem Index oder über seinen Namen, sofern ein solcher vorhanden ist, angesprochen und einer Objektvariablen zugewiesen werden:

```
Set doc = Application.Documents(1)               'via Index
Set doc = Application.Documents("Document1.doc")  'via Name
```

Index eines Objekts ermitteln

Ein Objekt kann also über seinen Index direkt angesprochen werden. Es gibt jedoch Situationen, in welchen der Index eines markierten Objekts ermittelt werden muss, um sicherzustellen, ob sich die Einfügemarke an oder innerhalb der gewünschten Position befindet.

Der Index eines Objekts ist jedoch meistens nur innerhalb der Auflistung vorhanden und ist auch keine Eigenschaft des einzelnen Objekts. Da die einzelnen Objekte innerhalb der Auflistung ab Dokumentanfang durchnummeriert werden, ist es leicht, die Nummer des markierten Objekts zu bestimmen.

In Listing 5.2 wird der Index der markierten Tabelle bestimmt. Dies, nachdem in einem ersten Programmschritt festgestellt wurde, dass sich die Einfügemarke innerhalb einer Tabelle befindet. Dieses Programmbeispiel kann analog auf andere Objekte angewendet werden.

Listing 5.2 Indexnummer der aktuellen Tabelle ermitteln

```
Sub IndexDerTabelleErmitteln()
  Dim rng As Word.Range
  Dim lngIndex As Long

  If Selection.Information(wdWithInTable) Then
    Set rng = Selection.Tables(1).Range
    rng.Start = ActiveDocument.Range.Start
    lngIndex = rng.Tables.Count
  End If

  MsgBox "Dies ist Tabelle " & CStr(lngIndex)
End Sub
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Die Prozedur setzt in einem ersten Schritt ein Range-Objekt auf die markierte Tabelle. Im folgenden Schritt wird der Bereich dieses Objekts angepasst. Die Startposition wird der Startposition des Dokuments gleichgesetzt. Im dritten Schritt werden die Tabellen des angepassten Range-Objekts gezählt. Da die markierte Tabelle gleichzeitig die letzte Tabelle innerhalb des Range-Objekts ist, stimmt die Anzahl der Tabellen mit der Indexnummer der markierten Tabelle überein.

Auflistung bearbeiten

For...Next
For Each... Next

Um alle zugehörigen Objekte einer Auflistung zu bearbeiten, gibt es zwei Arten von Schleifen. Die eine ist die For…Next-Anweisung, die andere die For Each…Next-Anweisung (die For…Next-Anweisung wurde bereits in Kapitel 3 detailliert vorgestellt).

Grundsätzlich verfügen alle Auflistungen über eine Eigenschaft mit der Bezeichnung Count. In dieser Eigenschaft ist die jeweils aktuelle Anzahl der in der Auflistung vorhandenen Elemente aufgeführt. Anhand dieser Eigenschaft und einem Zähler kann eine Schleife aufgebaut werden, welche alle Elemente der betreffenden Auflistung durchläuft. Die einzelnen Elemente der Auflistung werden durch deren Index angesprochen. Ein entsprechendes Beispiel ist in Listing 5.3 dargestellt.

Listing 5.3 Auflistung aller Elemente einer Auflistung mittels einer *For...Next*-Schleife

```
Sub Demo_For_Next()
  Dim intZähler As Integer
  Dim strText As String

  With Application.Documents
    For intZähler = 1 To .Count
      strText = strText & .Item(intZähler).Name & vbCr
    Next intZähler
  End With

  MsgBox strText, , "Alle geöffneten Dokumente"
End Sub
```

Anhand der For Each…Next-Anweisung können jedoch die einzelnen Objekte der Auflistung direkt angesprochen werden. Die Zuweisung innerhalb der Schleife weist das einzelne Objekt der betreffenden Objektvariablen zu. Das Objekt kann direkt bearbeitet werden. Ein entsprechendes Beispiel ist in Listing 5.4 dargestellt.

Listing 5.4 Auflistung aller Elemente einer Auflistung mittels einer *For Each...Next*-Schleife

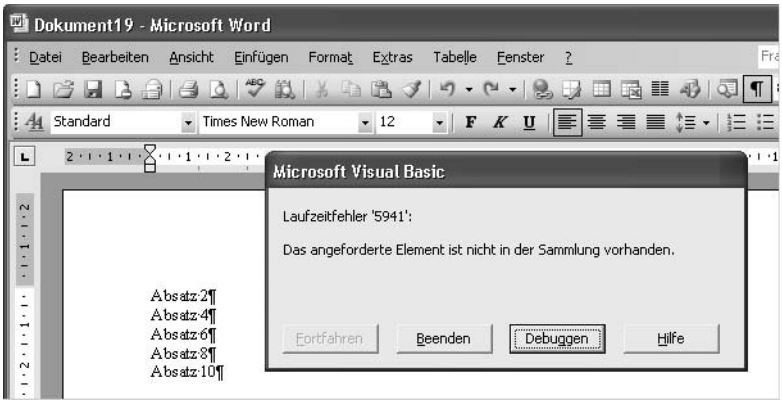
```
Sub Demo_ForEach_Next()
  Dim doc As Word.Document
  Dim strText As String

  For Each doc In Application.Documents
      strText = strText & doc.Name & vbCr
  Next doc

  MsgBox strText, , "Alle geöffneten Dokumente"
End Sub
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Elemente aus der Auflistung entfernen

Das Entfernen einzelner oder auch aller Elemente aus einer Auflistung kann heimtückisch sein. Wird dieses Vorhaben mit einer `For…Next`-Anweisung umgesetzt, wird regelmäßig der Laufzeitfehler 5941, »Das angeforderte Element ist nicht in der Sammlung vorhanden«, auftreten. Die Programmzeilen aus Listing 5.5 erzeugen eben diesen Fehler.

Abbildg. 5.2 Laufzeitfehler bei der Verwendung der *For…Next*-Schleife



Die Problematik liegt in der `Count`-Eigenschaft und deren Zuweisung an die `For…Next`-Schleife. Das Beispiel erzeugt ein Dokument mit zehn Absätzen. Bei der Zuweisung an die Schleife beträgt der Wert der `Count`-Eigenschaft 10. Dies bedeutet, dass die Schleife zehnmal durchlaufen wird. Bei jedem Durchgang wird ein Absatz (jener mit der Nummer der Variable `intZähler`) aus dem Dokument entfernt. Beim sechsten Schleifendurchgang sollte der sechste Absatz aus dem Dokument entfernt werden, im Dokument selber sind jedoch nur noch deren fünf enthalten, da in den ersten fünf Durchgängen je ein Absatz entfernt wurde. Der sechste Absatz ist nicht vorhanden, das Makro quittiert mit der Fehlermeldung 5941 seinen Dienst.

HINWEIS Beachten Sie in Abbildung 5.2, welche Absätze durch die `For…Next`-Schleife tatsächlich bearbeitet wurden.

Der Effekt, dass nur jeder zweite Absatz bearbeitet wird, liegt daran, dass nach dem Löschen des ersten Absatzes der zweite an dessen Stelle rückt. Der zweite Schleifendurchgang löscht dann den zweiten Absatz, also jenen Absatz mit der Zahl drei im Text, usw.

Listing 5.5 Entfernen von Elementen aus der Auflistung erzeugt den Laufzeitfehler 5941

```
Sub Demo_Entfernen_ForNext()
  Dim doc As Word.Document
  Dim intZähler As Integer

  Set doc = fktNeuesDokumentErzeugen()

  For intZähler = 1 To doc.Paragraphs.Count
    doc.Paragraphs(intZähler).Range.Delete
  Next intZähler
End Sub
```


Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Dennoch ist es möglich, mit einer For…Next-Schleife eine Auflistung zu bearbeiten und einzelne oder alle Elemente zu löschen. Bei der Definition kann das optionale Schlüsselwort Step verwendet werden.

Das Schlüsselwort Step dient zur Definition der Schrittweite innerhalb der Schleife. In Listing 5.6 wird als Wert für die Schrittweite minus Eins (–1) verwendet. Dies hat zur Folge, dass die Schleife vom größten Wert zum kleinsten Wert rückwärts durchlaufen wird.

Auf diese Weise wird das Problem mit dem Laufzeitfehler 5941 elegant gelöst. In dieser Konstellation wird die Schleife vom höchsten Wert zum niedrigsten Wert durchlaufen. Das Beispiel erzeugt wiederum ein Dokument mit zehn Absätzen. Bei der Zuweisung an die Schleife beträgt der Wert der Count-Eigenschaft 10. Dies bedeutet, dass die Schleife zehnmal durchlaufen wird. Bei jedem Durchgang wird ein Absatz (jener mit der Nummer der Variable intZähler) aus dem Dokument entfernt. Da jetzt aber mit dem Wert 10 gestartet wird, wird im ersten Durchgang der zehnte Absatz entfernt, beim zweiten Durchgang wird der neunte Absatz entfernt. Dies geht so weiter, bis die Schleife abgearbeitet ist.

Listing 5.6 Entfernen von Elementen aus der Auflistung mittels *For…Next* und *Step –1*

```
Sub Demo_Entfernen_ForNext_Step1()
  Dim doc As Word.Document
  Dim intZähler As Integer

  Set doc = fktNeuesDokumentErzeugen()

  For intZähler = doc.Paragraphs.Count To 1 Step -1
    doc.Paragraphs(intZähler).Range.Delete
  Next intZähler
End Sub
```

WICHTIG Beim Entfernen von Elementen aus einer Auflistung muss die For…Next-Schleife unbedingt vom größten zum kleinsten Element durchlaufen werden. Dies kann durch die Angabe einer negativen Schrittweite unter Verwendung des Schlüsselworts Step -1 erreicht werden.

Wird wie in Listing 5.7 statt einer For…Next-Schleife eine For Each…Next-Schleife verwendet, taucht die genannte Problematik gar nicht erst auf. In diesem Fall wird jedes Objekt direkt der Variablen para zugewiesen.

Im Gegensatz zur statischen Zuweisung der benötigten Durchgänge bei der For…Next-Schleife ist die For Each…Next-Schleife dynamisch. Bei jedem Schleifendurchgang werden die entsprechenden Objekte definiert. Dies hat jedoch zur Folge, dass nach dem Entfernen eines Elements aus der Auflistung dessen Indizierung neu aufgebaut werden muss. Dieses Verhalten wird bei größeren Dokumenten zu Problemen führen, weil die Schleife dadurch immer langsamer abgearbeitet wird.

Dieses Verhalten kann sehr einfach überprüft werden. Das Makro aus Listing 5.7 muss im Einzelschritt abgearbeitet werden. Sobald die ersten Absätze entfernt wurden, wird das Dokument bearbeitet. Es können zusätzliche Absätze aufgenommen oder bestehende manuell entfernt werden. Wird die Bearbeitung durch das Makro weiter ausgeführt, ist das problemlos möglich, da die benötigte Anzahl der Schleifendurchgänge dynamisch ermittelt wird.

Wird dieser Versuch bei den anderen beiden Beispielen durchgeführt, wird die Schleife einen Laufzeitfehler erzeugen oder eben keinen Fehler mehr erzeugen, da entsprechend viele Absätze eingefügt

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

wurden. Aber egal, wie Sie das Dokument modifizieren, es werden garantiert nicht alle Absätze entfernt.

Listing 5.7 Entfernen von Elementen aus der Auflistung mittels *For Each...Next*

```
Sub Demo_Entfernen_ForEachNext()
  Dim doc As Word.Document
  Dim para As Word.Paragraph

  Set doc = fktNeuesDokumentErzeugen()

  For Each para In doc.Paragraphs
    para.Range.Delete
  Next para
End Sub
```

HINWEIS Bei einer `For…Next`-Schleife wird die Anzahl der benötigten Schleifendurchgänge statisch bei der Zuweisung der Schleife festgelegt.

Bei einer `For Each…Next`-Schleife wird die Anzahl der benötigten Schleifendurchgänge dynamisch bei jedem Durchgang neu überprüft.

Selection – das verpönte Objekt

Werden für Word Makros erstellt oder wird das Programm von außen durch eine andere Applikation gesteuert, sollte grundsätzlich auf die Verwendung des `Selection`-Objekts verzichtet werden. Als Alternative zu `Selection` wird die Verwendung des `Range`-Objekts empfohlen (mehr zum `Range`-Objekt und dessen Möglichkeiten erfahren Sie in Kapitel 6).

- Das `Selection`-Objekt entspricht der Einfügemarke im Dokument. Werden komplexe Eingaben und Formatierungen vorgenommen, springt die Einfügemarke von Ort zu Ort. Die Bildschirmdarstellung ist sehr unruhig.
- Das `Range`-Objekt ermöglicht es, jeden Bereich des Dokuments direkt anzusprechen. Es können gleichzeitig mehrere Ranges definiert sein und ohne Wechsel der Einfügemarke bearbeitet werden.
- Durch die konsequente Verwendung vom `Range`-Objekt werden die einzelnen Anweisungen einfacher. Das Programm wird übersichtlicher und indirekt auch dokumentiert. Vergleichen Sie dazu Listing 5.8 und Listing 5.9.
- Wird Word als verborgenes Programm durch eine andere Applikation ferngesteuert, steht kein `Selection`-Objekt unmittelbar zur Verfügung. Dies hat zur Folge, dass alle Programmzeilen überarbeitet werden müssen und zumindest durch das Hinzufügen einer Objektvariablen, die auf die Word-Anwendung hinweist, zu ergänzen sind.
- Dem `Selection`-Objekt stehen im Verhältnis zum `Range`-Objekt nur wenige zusätzliche Eigenschaften und Methoden zur Verfügung. Für die professionelle Software-Entwicklung sind diese zusätzlichen Eigenschaften und Methoden meistens von keiner großen Bedeutung.

Trotz dieser Einwände, dass auf die Verwendung des `Selection`-Objekts verzichtet werden sollte, gibt es Konstellationen, die nur oder besser mit diesem Objekt umgesetzt werden können. Diese werden in Kapitel 6 im Abschnitt »Die gegenwärtige Markierung: Selection und ähnliche Objekte« zusammengefasst.

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Listing 5.8 Einfügen eines Textes in die Kopfzeile unter Verwendung des *Selection*-Objekts

```
Sub Demo_Selection()
  If ActiveWindow.View.SplitSpecial <> wdPaneNone Then
    ActiveWindow.Panes(2).Close
  End If
  If ActiveWindow.ActivePane.View.Type = wdNormalView Or ActiveWindow. _
    ActivePane.View.Type = wdOutlineView Then
    ActiveWindow.ActivePane.View.Type = wdPrintView
  End If
  ActiveWindow.ActivePane.View.SeekView = wdSeekCurrentPageHeader
  Selection.TypeText Text:="Text in Kopfzeile"
End Sub
```

Listing 5.9 Einfügen eines Textes in die Kopfzeile unter Verwendung des *Range*-Objekts

```
Sub Demo_Range()
  Dim rng As Word.Range
  Set rng = ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range
  rng.Text = "Ein anderer Text in der Kopfzeile"
End Sub
```

Die Zeilen in Listing 5.8 wurden mit dem Makrorekorder aufgezeichnet. Um einen Text mittels `Selection` in die Kopfzeile einzufügen, muss sichergestellt sein, dass sich die Einfügemarke auch wirklich in der Kopfzeile befindet. Hierfür werden acht zusätzliche Programmzeilen benötigt.

Das Gleiche wird in Listing 5.9 unter Verwendung eines `Range`-Objekts mit drei Programmzeilen erreicht. Zusätzlich ist auf den ersten Blick ersichtlich, auf welchen Bereich sich der definierte `Range` bezieht.

Die aufgeführten Codesequenzen finden Sie in der Beispieldatei *Bsp05_01.doc* auf der CD-ROM zum Buch im Ordner *\Beispiele\Kap05*.

Auto-Makros als Pseudo-Ereignisse

Im aktuellen Teil dieses Buchs werden die Möglichkeiten zu den Dokument- und Programmereignissen aufgezeichnet (siehe Kapitel 7). Neben den eigentlichen Ereignissen, die zu einem festgelegten Zeitpunkt ausgeführt werden, existieren in Word spezielle Makros, die automatisch ohne zusätzliche Interaktion des Anwenders angestoßen werden: die so genannten Auto-Makros.

Damit Word diese automatisch auszuführenden Makros als solche erkennt, müssen sie besonders benannt werden.

Tabelle 5.1 Zusammenstellung der Auto-Makros und deren Ausführungszeitpunkt

Bezeichnung	Zeitpunkt der Ausführung
`AutoOpen`	Beim Öffnen eines bestehenden Dokuments.
`AutoClose`	Beim Schließen eines Dokuments.
`AutoNew`	Beim Erstellen eines neuen Dokuments.

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)

Tabelle 5.1 Zusammenstellung der Auto-Makros und deren Ausführungszeitpunkt *(Fortsetzung)*

Bezeichnung	Zeitpunkt der Ausführung
`AutoExec`	Beim Starten von Word bzw. beim Laden eines Add-Ins.
`AutoExit`	Beim Beenden von Word bzw. beim Entladen eines Add-Ins.

Auto-Open
Auto-Close

Die Makros `AutoOpen` und `AutoClose` können im aktuellen Dokument, der zugehörigen Dokumentvorlage oder in der *Normal.dot* abgespeichert werden.

AutoNew

Das Makro `AutoNew` kann in einer Dokumentvorlage oder in der *Normal.dot* abgespeichert werden.

AutoExec
AutoExit

Die Makros `AutoExec` und `AutoExit` können in der *Normal.dot* oder einer globalen Vorlage (Add-In) abgespeichert werden.

HINWEIS Besteht ein Namenskonflikt, da mehrere Makros mit der gleichen Bezeichnung vorhanden sind, wird nur das Makro ausgeführt, welches eher zum Kontext passt. Demzufolge wird ein Makro im aktuellen Dokument vor jenem der zugehörigen Dokumentvorlage, ein Makro innerhalb der Dokumentvorlage vor jenem aus der Normal.dot und das Makro aus der Normal.dot vor jenem aus dem Add-In abgearbeitet.

Diese Regelung gilt nicht nur für die Auto-Makros, sondern grundsätzlich für alle Makros. Spezialfälle sind lediglich `AutoExec` und `AutoExit`. Diese beiden Makros werden für die *Normal.dot* und für jedes geladene Add-In einzeln abgearbeitet. Die Reihenfolge für das Laden der im Autostart-Ordner gespeicherten Add-Ins kann nicht beeinflusst werden.

Durch Drücken der ⇧-Taste kann verhindert werden, dass ein Auto-Makro gestartet wird (beispielsweise beim Starten von Word, beim Öffnen eines Dokuments usw.).

Das spätere Ausführen von Auto-Makros kann innerhalb einer VBA-Prozedur unterbunden werden. Die Anweisung `WordBasic.DisableAutoMacros True` verhindert das automatische Starten so lange, bis diese Anweisung zurückgesetzt oder Word erneut gestartet wird (mehr zu WordBasic finden Sie in Kapitel 6).

Zusammenfassung

In diesem Kapitel soll dem Leser der Einstieg in die Welt der Objekte vermittelt werden. Das Objektmodell von Word ist mächtig und deshalb sind die Themen in diesem Kapitel wichtig.

- Es wurde vermittelt, wie ein gesuchtes Objekt einfach ermittelt (Seite 174) und dieses einer Variablen zugewiesen werden kann (Seite 175).
- Es wurde aufgezeigt wie eine Auflistung von Objekten bearbeitet und wie auf deren einzelne Objekte zugegriffen wird (Seite 179 ff.).
- Es wurde näher auf das `Selection`-Objekt eingegangen und dessen Vor- und Nachteile gegenüber dem `Range`-Objekt erläutert (Seite 183 ff.).
- Mit der Behandlung der speziellen Auto-Makros wird das Kapitel abgeschlossen (Seite 184 ff.).

Cindy Meister, Thomas Gahler, Peter Jamieson, Christian Freßdorf: Microsoft Word-Programmierung - Das Handbuch. Microsoft Press 2006 (ISBN 3-86063-989-7)