

HANSER

Rene Martin

# VBA mit Excel Grundlagen und Profiwissen

ISBN-10: 3-446-41506-8

ISBN-13: 978-3-446-41506-5

Leseprobe

Weitere Informationen oder Bestellungen unter  
<http://www.hanser.de/978-3-446-41506-5>  
sowie im Buchhandel.

### Tipp

Die Kommentare sind überflüssig und können gelöscht werden.

Der Code kann nun weiter verarbeitet werden.

Eine Besonderheit hat der Makrorekorder aufzuweisen. Angenommen, die Zelle F17 ist ausgewählt. Wenn Sie aufzeichnen „klicke auf A1“, dann zeichnet der Makrorekorder auf:

```
Range("A1").Select
```

Das ist eine „absolute“ Sprunganweisung. Das heißt – unabhängig von der Cursorposition wird sich die Markung auf die Zelle A1 des aktuellen Blattes bewegen. Wenn Sie dies nicht möchten, können Sie die Schaltfläche „Relative Aufzeichnung“ aktivieren – dann zeichnet er auf:

```
ActiveCell.Offset(-16, -5).Range("A1").Select
```

Das heißt: ausgehend von der aktuellen Position 16 Zeilen nach oben und fünf Spalten nach links. Dieser Befehl würde zu einem Fehler führen, wenn der Cursor beispielsweise auf der Zelle B7 sitzt.

Das bedeutet: Je nachdem, was Sie benötigen, sollten Sie die relativen oder absoluten Anweisungen aufzeichnen. Auch innerhalb einer Makroaufzeichnung ist ein Wechseln möglich.

Und: Bis Excel 2003 befanden sich die Befehle für den Makrorekorder im Menü Extras | Makro.

## 4.2 Der zweifelhafte Code des Makrorekorders

---

Mithilfe des Makrorekorders kommt man an (fast) alle internen Befehle. Das Ergebnis des Makrorekorders hat jedoch einige entscheidende Nachteile.

### 4.2.1 Zu viel Code

Wenn Sie einen Excel-Befehl aufzeichnen, dann zeichnet das Programm zu viel Code auf. Angenommen, Sie möchten eine Papierseite ins Querformat drehen. Dann zeichnet Excel (in der Version 2007) auf:

```
Sub Makro1()  
  
    With ActiveSheet.PageSetup  
  
        .PrintTitleRows = ""  
  
        .PrintTitleColumns = ""  
  
    End With
```

```
ActiveSheet.PageSetup.PrintArea = ""

With ActiveSheet.PageSetup

    .LeftHeader = ""

    .CenterHeader = ""

    .RightHeader = ""

    .LeftFooter = ""

    .CenterFooter = ""

    .RightFooter = ""

    .LeftMargin = Application.InchesToPoints(0.708661417322835)

    .RightMargin = Application.InchesToPoints(0.708661417322835)

    .TopMargin = Application.InchesToPoints(0.78740157480315)

    .BottomMargin = Application.InchesToPoints(0.78740157480315)

    .HeaderMargin = Application.InchesToPoints(0.31496062992126)

    .FooterMargin = Application.InchesToPoints(0.31496062992126)

    .PrintHeadings = False

    .PrintGridlines = False

    .PrintComments = xlPrintNoComments

    .PrintQuality = 360

    .CenterHorizontally = False

    .CenterVertically = False

    .Orientation = xlLandscape

[...]    .Draft = False

    .PaperSize = xlPaperA4

    .FirstPageNumber = xlAutomatic

    .Order = xlDownThenOver

    .BlackAndWhite = False

    .Zoom = 100

[...]
```

Der Rest braucht hier nicht wiederholt zu werden.

Excel zeichnet sämtliche Informationen der vier Registerblätter Seitenlayout | Seite einrichten (bis Excel 2003: Datei | Seite einrichten) auf. Nun liegt es an Ihnen, den überflüssigen Code zu entfernen und auf den nötigen Befehl zu reduzieren. In diesem Fall lautet er:

```
Sub Makro1()  
  
    With ActiveSheet.PageSetup  
  
        .Orientation = xlLandscape  
  
    End With  
  
End Sub
```

oder noch knapper:

```
ActiveSheet.PageSetup.Orientation = xlLandscape
```

### 4.2.2 Nicht immer der beste Code

Häufig zeichnet Excel die US-amerikanische Schreibweise auf. Wird in eine Zelle eine Summe geschrieben, dann lautet der aufgezeichnete Befehl:

```
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
```

Es würde auch mit der deutschen Schreibweise

```
ActiveCell.FormulaLocal = "=SUMME(A1:A3)"
```

funktionieren. Wenn Sie eine Zelle benutzerdefiniert mit der Kategorie „Währung“ formatieren, dann lautet der aufgezeichnete Befehl:

```
Selection.NumberFormat = "#,##0.00 $"
```

Besser lesbar für uns Mitteleuropäer ist sicherlich:

```
Selection.NumberFormatLocal = "#.##0,00 €"
```

Auch das im oberen Abschnitt beschriebene Beispiel „gehe zu Zelle A1“ in Abhängigkeit von der aktuellen Position – oder genauer: gehe 15 Zeilen nach oben und fünf Spalten nach links:

```
ActiveCell.Offset(-16, -5).Range("A1").Select
```

ist einer der vielen Befehle, die überarbeitet werden sollten. Der interne Verweis `Range("A1")` ist überflüssig. Es genügt die Zeile:

```
ActiveCell.Offset(-16, -5).Select
```

### 4.2.3 Nicht immer Code

Einige Dinge zeichnet Excel gar nicht auf. Beispielsweise wenn Sie versuchen, die Eigenschaften der Datei (Office-Menü Vorbereiten | Eigenschaften, bis Excel 2003: Datei | Eigenschaften) aufzuzeichnen, dann liefert der Makrorekorder ein leeres Makro. Man muss nun wissen, wie der Befehl „Eigenschaften“ heißen könnte und vor allem – zu welchem Objekt er gehört. Ähnlich ist es mit dem Löschen eines Tabellenblattes. Wenn Sie mit dem Makrorekorder aufzeichnen „lösche Tabelle1“, dann lauten die Codezeilen:

```
Sheets("Tabelle1").Select
ActiveWindow.SelectedSheets.Delete
```

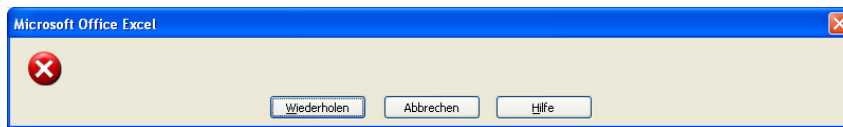
Würden Sie dies dem Anwender weitergeben oder in ein Programm einbauen, dann würde er stets gefragt werden, ob er das Blatt wirklich löschen möchte. Das ist kein sauberes Vorgehen!

### 4.2.4 Fehlerhafter Code

Wenn Sie die Datenüberprüfung (bis Excel 2003: Gültigkeit) mit einer benutzerdefinierten Formel aufzeichnen, dann erhalten Sie folgenden Code:

```
With Selection.Validation
    .Delete
    .Add Type:=xlValidateCustom, AlertStyle:=xlValidAlertStop, _
        Operator:=xlBetween, Formula1:="<heute()"
    .IgnoreBlank = True
    .InCellDropdown = True
    .InputTitle = ""
    .ErrorTitle = ""
    .InputMessage = ""
    .ErrorMessage = ""
    .ShowInput = True
    .ShowError = True
End With
```

Wenden Sie nun diesen Code auf einen anderen Bereich an, dann sieht die Fehlermeldung folgendermaßen aus:



**Abbildung 4.4** Der Ergebnis des Makrorekorders

Was bei dieser Funktion noch klappt, das scheitert beispielsweise bei folgendem Problem: Der Benutzer darf am Ende eines Textes kein Leerzeichen eingeben. Der Makrorekorder zeichnet auf:

```
With Selection.Validation
    .Delete
    .Add Type:=xlValidateCustom, AlertStyle:=xlValidAlertStop, _
        Operator:=xlBetween, Formula1:="=rechts (A1;1) <>" " "
```

Beachten Sie dabei, dass einfache Anführungszeichen als doppelte Anführungszeichen geschrieben werden müssen. Auch wenn der Makrorekorder die deutschen Funktionsnamen aufzeichnet, muss in den VBA-Code der englische Funktionsname eingegeben werden. Und: In der deutschen Schreibweise steht das Semikolon als Trennzeichen – in der US-amerikanischen das Komma. Es ist also nur so korrekt:

```
Selection.Validation.Add _
    Type:=xlValidateCustom, AlertStyle:=xlValidAlertStop, _
    Formula1:="=Right (A1,1) <>" " "
```

### 4.2.5 Excel-VBA „hilft“ in VBA nicht immer

Wenn Sie den aufgezeichneten Code weiter verarbeiten möchten und schreiben selbst das Objekt „Selection“, tippen einen Punkt, dann kann man Excel nicht dazu bewegen, die komplette Liste der Eigenschaften und Methoden anzuzeigen. Wenn Sie dagegen „sauber“ programmieren, wie ab dem nächsten Kapitel beschrieben, dann werden automatisch die Elemente aufgelistet.

### 4.2.6 Excel zeichnet „unscharf“ auf

Und damit sind wir beim zweiten Problem. Angenommen, Sie suchen den Befehl, mit dem ein Zellohintergrund gelb eingefärbt wird. Excel wird Ihnen – wenn Sie das Symbol verwenden – folgende Zeile aufzeichnen:

```
With Selection.Interior
    .ColorIndex = 6
```

```
.Pattern = xlSolid
```

```
End With
```

Zugegeben: Die Eigenschaft „Pattern“, die auf „Solid“ gesetzt wird, stört an dieser Stelle wenig. Störender wirkt sich hingegen das Objekt „Selection“ aus. Es bedeutet, dass der Cursor auf dieser Zelle sitzen muss und dass das zugehörige Tabellenblatt selektiert ist. Wenn Sie dies möchten – beispielsweise dem Benutzer ein Symbol zur Verfügung stellen, mit dessen Hilfe er komplexe, immer wiederkehrende Formatierungen mit einem Mausklick durchführen kann, so ist der Befehl gerechtfertigt. In den meisten anderen Fällen jedoch nicht. Sie kopieren beispielsweise Daten von einer Datei in eine andere und markieren das Ergebnis gelb. Dies sollte ohne „Selection“ geschehen, da Sie nicht am Bildschirm programmieren, sondern Werte oder Formate in eine Zelle schreiben.

An vielen Stellen zeichnet der Makrorekorder Dinge wie „ActiveCell“, „Selection“ oder „ActiveChart“ auf. Auch andere per Makrorekorder aufgezeichnete Befehle (Objekte) wie „Sheets("Tabelle1")“, „Range("A2")“ und so weiter sind im Sinne einer Objektorientierung nicht elegant und bergen eine Menge Fehler.

#### Hinweis

Vermeiden Sie das Objekt „Selection“ und Ähnliche und die Methode „Activate“ (oder „Select“). Der Code wird langsam, unübersichtlich, ist schlecht zu pflegen und fehleranfällig. In meinen gesamten Programmen tauchen diese Befehle überhaupt nicht auf. Zugegeben – vielleicht in der letzten Codezeile, damit der Cursor nach der Auswertung der Tabellenblätter auf einem bestimmten Blatt in Zelle „A1“ sitzt. Aber sonst nicht!

Damit man völlig ohne diese Methoden auskommen kann beziehungsweise die Objekte korrekt setzt, muss man das Objektmodell kennen.

## 4.3 Fazit

Der Makrorekorder ist ein praktisches Werkzeug, mit dem man schnell an Code – oder genauer – und an die eigentlichen Objekte, Eigenschaften und Methoden gelangt. Für einen Anfänger sicherlich völlig ausreichend; ein Profi sollte jedoch immer den Code überarbeiten, beziehungsweise den Code lediglich als „Baustelle“ verwenden. Das heißt, ein guter Programmierer sollte:

- Voraussetzungen überprüfen
- mehrfach durchgeführte Aktionen in Schleifen verwalten
- Fehler abfangen
- dem Anwender bequeme Eingabemöglichkeiten zur Verfügung stellen (den Start über Symbole, die Eingabe über UserForms, ... organisieren)
- unnötigen Ballast löschen
- Select und Activate entfernen