

# VBA mit Excel

Das umfassende Handbuch. Konzepte und Techniken der VBA-Programmierung. Das Standardwerk für  
Einsteiger und fortgeschrittene Anwender

Bearbeitet von  
Bernd Held

2., aktualisierte und erweiterte Auflage 2015. Buch. 950 S. Hardcover

ISBN 978 3 8362 3821 2

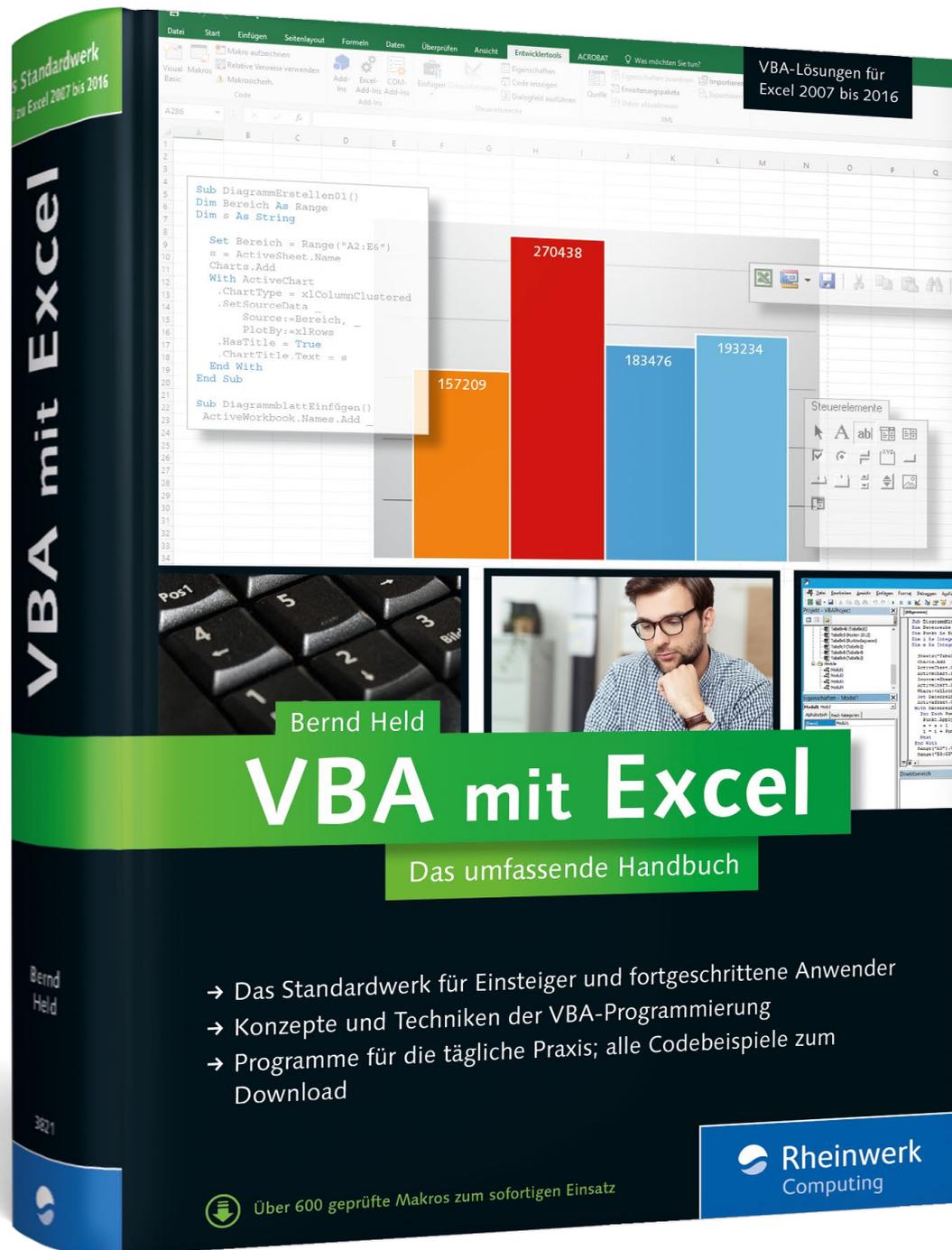
Format (B x L): 16 x 24 cm

[Weitere Fachgebiete > EDV, Informatik > Datenbanken, Informationssicherheit,  
Geschäftssoftware > Tabellenkalkulation](#)

schnell und portofrei erhältlich bei

  
DIE FACHBUCHHANDLUNG

Die Online-Fachbuchhandlung [beck-shop.de](http://beck-shop.de) ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.



## Leseprobe

In diesem Auszug erfahren Sie von Bernd Held alles über die praktische Anwendung der VBA-Sprachelemente. Außerdem lernen Sie, wie Sie Diagramme programmieren und können einen Blick in das vollständige Inhalts- und Stichwortverzeichnis des Buchs werfen.



»Die Sprachelemente der VBA«  
»Tabellen und Diagramme programmieren«



Inhaltsverzeichnis



Index



Der Autor



Leseprobe weiterempfehlen

Bernd Held

### VBA mit Excel – Das umfassende Handbuch

950 Seiten, gebunden, 2. Auflage 2015

49,90 Euro, ISBN 978-3-8362-3821-2



[www.rheinwerk-verlag.de/3891](http://www.rheinwerk-verlag.de/3891)

## Kapitel 3

# Die Sprachelemente von Excel-VBA

*Das Wesentliche, was eine Programmiersprache ausmacht, sind ihre Sprachelemente. In diesem Kapitel erfahren Sie, wie Sie mit Hilfe von Abfragen, Schleifen und anderen Anweisungen Ihre Makros flexibel gestalten.*

Die Sprachelemente lassen sich nicht mit dem Makrorekorder aufzeichnen, sondern Sie müssen sie selbst erstellen. Der richtige Einsatz der Sprachelemente macht letztendlich die Kunst der Programmierung aus.

### Die Beispiele

Sie finden alle Beispiele zum Download unter <http://www.rheinwerk-verlag.de/3891> in der Datei Sprachelemente.xlsm.



### Fragen zum Download?

Nähere Angaben zum Download finden Sie bei Bedarf im betreffenden Hinweiskasten in der Einleitung von Kapitel 1, »Die Entwicklungsumgebung von Excel«.

## 3.1 Bedingungen

Mit Bedingungen können Sie in Excel bestimmte Zustände abfragen und je nach Zustand anders reagieren. Es ist aus meiner Sicht eines der wichtigsten Elemente in der Programmierung. Jede Verzweigung hat eine oder mehrere Bedingungen, dann einen Zweig, der eintritt, wenn die Bedingung zutrifft, und einen Zweig, der abgearbeitet wird, wenn die Bedingung nicht zutrifft.

Bei den Verzweigungen kann zwischen zwei Formen unterschieden werden:

### Normalform:

```
IF Bedingung Then Aktion1 Else Aktion2
```

**Blockform:**

```

If Bedingung Then
    Aktion1a
    Aktion1b
    Aktion1c
Else
    Aktion2a
    Aktion2b
End if

```

Verwenden Sie die Blockform, können mehrere Schritte nacheinander durchgeführt werden. Dabei müssen Sie aber beachten, dass Sie die Anweisung mit einem `End If` abschließen müssen.

Bedingungen werden in der Praxis unter anderem für Prüfungen aller Art eingesetzt. Sie sind in fast jeder Schleife als »Innerei« vorhanden.

Es folgen nun einige typische Beispiele für den Einsatz von Verzweigungen.

## 3.2 Typische Aufgaben aus der Praxis

Lernen Sie auf den nächsten Seiten ausgewählte Praxisbeispiele kennen, die mit Bedingungen arbeiten.

### 3.2.1 Wert in einer Spalte suchen

Bei der folgenden Aufgabenstellung soll ein eindeutiger Wert in einer Spalte gefunden und anschließend gekennzeichnet werden. Der zu suchende Wert soll über eine `InputBox` vom Anwender eingegeben werden. Schauen Sie sich dazu das Makro aus Listing 3.1 an.

```

Sub WertInSpalteSuchen()
    Dim rngTreffer As Range
    Dim strSuchbegriff As String

    Tabelle1.Range("A:A").Interior.ColorIndex = xlColorIndexNone
    strSuchbegriff = InputBox("Suchbegriff eingeben!", "Direktsuche", 4720)

    If Len(strSuchbegriff) <> 0 Then

        Set rngTreffer = Tabelle1.Range("A:A").Find _
            (What:=strSuchbegriff, LookIn:=xlValues, LookAt:=xlWhole)

```

```

If rngTreffer Is Nothing Then
    MsgBox "Wert nicht gefunden"
Else
    rngTreffer.Interior.ColorIndex = 4
End If

End If
End Sub

```

#### Listing 3.1 Einen bestimmten Wert in einer Spalte finden

Deklarieren Sie im ersten Schritt des Makros aus Listing 3.1 eine Objektvariable mit dem Namen `rngTreffer` vom Typ `Range` sowie eine `String` Variable mit dem Namen `strSuchbegriff`.

Entfärben Sie Spalte A über die Eigenschaft `ColorIndex`, der Sie die Konstante `xlColorIndexNone` zuweisen.

Rufen Sie danach die Funktion `InputBox` auf. Dabei geben Sie im ersten Argument die Meldung an, die im Meldungsfenster erscheinen soll. Im zweiten Argument definieren Sie den Titel, der oberhalb des Meldungsfensters angezeigt werden soll. Beim dritten Argument können Sie eine Vorbelegung für das Textfeld im Meldungsfenster bestimmen.

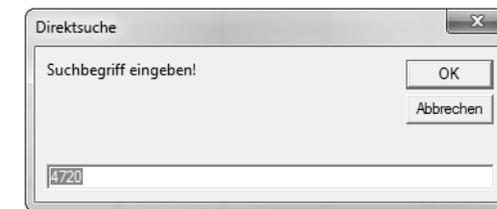


Abbildung 3.1 Über eine `InputBox` mit Vorbelegung eine Eingabe verlangen

Werten Sie die Eingabe des Anwenders aus. Wenn er nichts eingegeben hat, dann liefert die Funktion `Len` den Wert 0. In diesem Fall soll nichts weiter passieren.

Im anderen Fall setzen Sie eine `Direktsuche` in Excel auf, indem Sie die Methode `Find` einsetzen. Diese Methode hat drei wichtige Parameter:

- ▶ **What:** Bei diesem Parameter muss der zu suchende Begriff eingegeben werden. Die Angabe dieses Parameters ist Pflicht.
- ▶ **LookIn:** Dieser Parameter bestimmt, dass beispielsweise in Werten, nicht in Formeln gesucht werden soll.
- ▶ **LookAt:** Über diesen Parameter legen Sie fest, ob Sie vollqualifiziert oder teilqualifiziert suchen möchten. Wenn Sie diesen Parameter nicht angeben, dann wird

dafür die Konstante `xlPart` angenommen, was bedeutet, dass Excel auch Werte, die dem Suchbegriff ähnlich sind, findet. Weisen Sie diesem Parameter die Konstante `xlWhole` zu, damit die Suche wirklich nur eine eindeutige Übereinstimmung findet.

Nachdem die Suche aufgesetzt ist, muss geprüft werden, ob sie erfolgreich war. Auch zu diesem Zweck kommt die Anweisung `If` ins Spiel. War die Folge erfolglos, dann ist die Objektvariable `rngTreffer` leer. Im anderen Falle ist die Objektvariable erfolgreich gesetzt und zeigt genau auf die Fundstelle. Damit haben Sie Zugriff auf die gefundene Zelle, die Sie im Anschluss daran über die Eigenschaft `ColorIndex` einfärben.

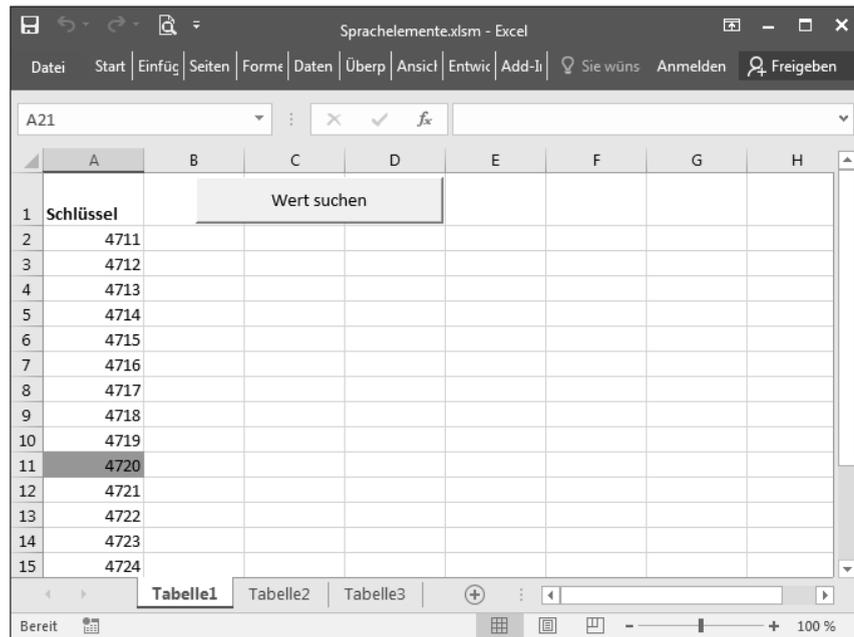


Abbildung 3.2 Der gesuchte Wert wurde gefunden und gekennzeichnet.

### 3.2.2 Liegt die aktive Zelle in einem vorgegebenen Bereich?

Bei dieser Fragestellung liegt in TABELLE2 der Bereich A1:D10 vor. Das Makro aus Listing 3.2 prüft, ob die aktive Zelle in diesem Bereich liegt.

```
Sub LiegtZelleImBereich()  
    Dim rngBereich As Range
```

```
    Set rngBereich = Tabelle2.Range("A1:D10")
```

```
    If Intersect(ActiveCell, rngBereich) Is Nothing Then  
        MsgBox "Die Zelle " & ActiveCell.Address & _
```

```
        " liegt außerhalb des Zielbereichs " & rngBereich.Address  
    Else  
        MsgBox "Die Zelle " & ActiveCell.Address & _  
            " liegt im Zielbereich " & rngBereich.Address  
    End If
```

```
End Sub
```

Listing 3.2 Prüfung, ob eine Zelle in einem Bereich liegt

Deklarieren Sie zu Beginn des Makros aus Listing 3.2 eine Objektvariable vom Typ Range mit dem Namen `rngBereich`. Geben Sie danach über die Anweisung `Set` bekannt, wo sich der Bereich in TABELLE2 befinden soll.

Mit Hilfe der Methode `Intersect` können Sie überprüfen, ob die aktive Zelle in dem vorgegebenen Bereich liegt. Wenn nicht, dann liefert die `If`-Bedingung als Rückgabe den Wert `Nothing`, was Sie mit einer Meldung am Bildschirm über die Funktion `MsgBox` quittieren.

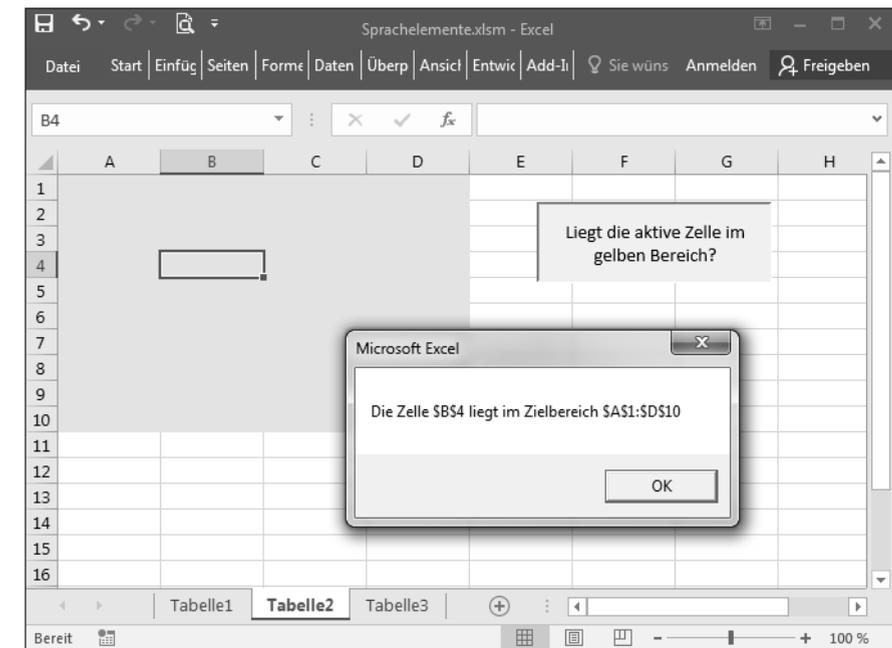


Abbildung 3.3 Die Bereichsprüfung ist in diesem Fall positiv.

### 3.2.3 Prüfung, ob eine bestimmte Datei verfügbar ist

Das Beispiel aus Listing 3.3 prüft, ob eine bestimmte Datei in einem vorgegebenen Verzeichnis existiert.

```

Sub IstDateiVorhanden()
    Dim strDatei As String
    Const ConstDatei = _
        "C:\Users\Bernd.HeId\Desktop\Excel-VBA-Handbuch\Beispiele\Kundenliste.txt"

    strDatei = Dir(ConstDatei)
    If strDatei <> "" Then
        MsgBox "Datei vorhanden!", vbExclamation
    Else
        MsgBox "Datei " & strDatei & " nicht da!", vbCritical
    End If
End Sub

```

### Listing 3.3 Prüfung, ob eine bestimmte Datei in einem vorgegebenen Verzeichnis liegt

Deklarieren Sie im ersten Schritt des Makros aus Listing 3.3 eine String-Variable mit dem Namen `strDatei`. Als Konstante geben Sie den Pfad und Dateinamen zu der Datei an, deren Existenz Sie prüfen möchten. Selbstverständlich müssen Sie den Pfad- und Dateinamen anpassen.

Mit Hilfe der Funktion `Dir` können Sie prüfen, ob die angefragte Datei im vorgegebenen Verzeichnis existiert. Wenn ja, dann ergibt die Prüfung über die Anweisung `If`, dass der Name der Datei in der Variablen `strDatei` steht. Kann die Datei nicht gefunden werden, dann bleibt der Inhalt der Variablen leer.

### 3.2.4 Spalteninhalte direkt nach der Eingabe umsortieren

Auch bei Ereignissen, die ich in Kapitel 9, »Ereignisse programmieren«, behandeln werde, können Sie mit Bedingungen arbeiten. Bei der nächsten Aufgabenstellung liegen in TABELLE3 Zahlenwerte vor. Neue Eingaben in Spalte A sollen direkt nach der Eingabe automatisch sortiert werden. Sehen Sie sich dazu einmal die Ausgangssituation in Abbildung 3.4 an.

Um die automatische Umsortierung einzubauen, stellen Sie das Ereignis `Worksheet_Change` wie folgt ein:

1. Klicken Sie in der Excel-Oberfläche mit der rechten Maustaste auf den Tabellennamen, und wählen Sie den Befehl `CODE ANZEIGEN` aus dem Kontextmenü. Sie landen jetzt direkt hinter der Tabelle in der Entwicklungsumgebung.
2. Stellen Sie im Codefenster auf der rechten Seite oben im ersten Dropdown den Eintrag `WORKSHEET` ein. Dadurch wird das Standardereignis `Worksheet_SelectionChange` eingestellt, das auf jedes Verschieben des Cursors reagiert. Dieses Ereignis brauchen wir nicht.

3. Wählen Sie im zweiten Dropdown oberhalb des Codefensters das Ereignis `CHANGE` aus. Dadurch wird der noch leere Ereignisrahmen des Ereignisses eingestellt.
4. Kompletieren Sie das noch leere Ereignis wie in Listing 3.4 gezeigt.

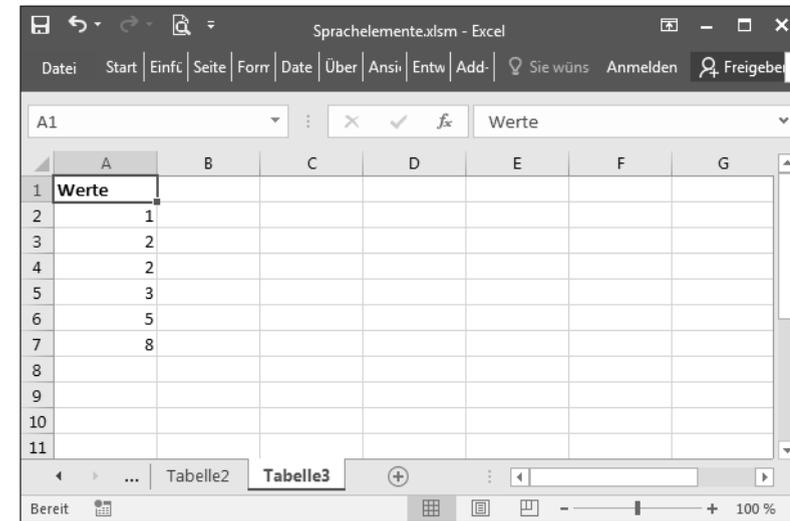


Abbildung 3.4 Neue Werte sollen am Ende der Liste erfasst und automatisch umsortiert werden.

```

Private Sub Worksheet_Change(ByVal Target As Range)
    Dim lngZeileMax As Long

    If Target.Column = 1 Then

        lngZeileMax = Cells(Rows.Count).End(xlUp).Row

        Range("A1:A" & lngZeileMax).Sort _
            Key1:=Range("A1"), Order1:=xlAscending, Header:=xlYes

    End If
End Sub

```

Listing 3.4 Eine automatische Sortierung wird angestoßen, wenn ein Eintrag in Spalte A vorgenommen wird.

Im Kopf des Ereignisses sehen Sie die Variable `Target` vom Typ `Range`. Über diese Variable weiß Excel genau, in welcher Zelle der Anwender eine Eingabe vorgenommen

hat. Falls die Eingabe in Spalte A vorgenommen wurde, liefert die Prüfung über die Eigenschaft `Column` die Spaltennummer 1. In diesem Fall wenden Sie die Methode `Sort` an, um die belegten Zellen aus Spalte A zu sortieren. Wichtig bei der Sortierung sind folgende Parameter der Methode:

- ▶ **Key1:** Über diesen Parameter legen Sie fest, nach welcher Spaltenüberschrift sortiert werden soll. Hier spricht man auch vom *Sortierkriterium*.
- ▶ **Order1:** Bei diesem Parameter legen Sie fest, ob aufsteigend oder absteigend sortiert wird. Die Sortierreihenfolge wird über die beiden Konstanten `xlAscending` bzw. `xlDescending` geregelt.
- ▶ **Header:** Dieser Parameter bestimmt, ob es in der zu sortierenden Liste eine Überschrift gibt oder nicht. Mögliche Konstanten dabei sind: `xlYes`, `xlNo` und interessanterweise `xlGuess`. Beim Gebrauch der letzten Konstante überlassen Sie es Excel, einzuschätzen, ob eine Überschrift verfügbar ist oder nicht.

### 3.2.5 Spalten mit Wochenenden kennzeichnen

Bei der folgenden Aufgabe liegt in TABELLE4 in der ersten Zeile eine Datumsleiste vor. Die Aufgabe besteht darin, zunächst zu prüfen, ob in der jeweiligen Zelle ein Datum steht. Wenn ja, dann erfolgt eine zweite Prüfung, ob es sich bei dem Datum um ein Wochenende handelt. Sehen Sie sich dazu das Makro aus Listing 3.5 an.

```
Sub WochenendenKennzeichnen()
    Dim lngSpalte As Long
    Dim lngSpalteMax As Long

    With Tabelle4

        .Rows(1).Interior.ColorIndex = xlColorIndexNone
        lngSpalteMax = .Cells(1, .Columns.Count).End(xlToLeft).Column

        For lngSpalte = 1 To lngSpalteMax

            If IsDate(.Cells(1, lngSpalte).Value) Then

                If Weekday(.Cells(1, lngSpalte).Value, vbMonday) > 5 Then
                    .Cells(1, lngSpalte).Interior.ColorIndex = 4
                End If

            End If

        Next lngSpalte

    End With
End Sub
```

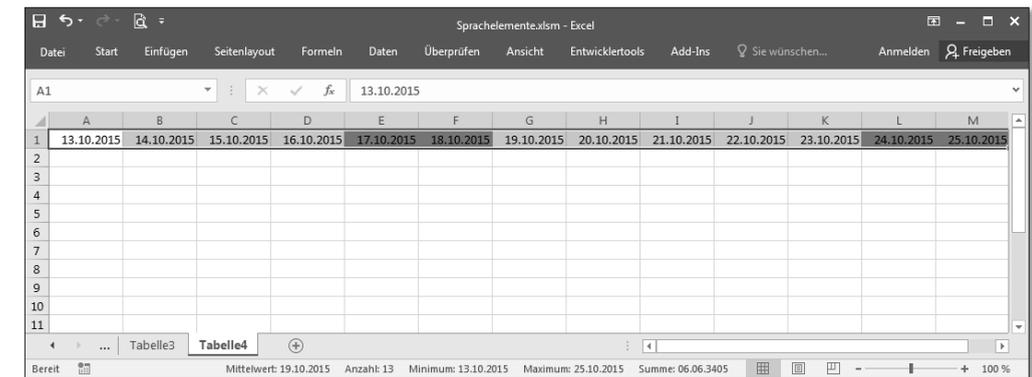
```
End With
```

```
End Sub
```

**Listing 3.5** Alle Wochenendspalten soll farbig hinterlegt werden.

Deklarieren Sie zu Beginn des Makros aus Listing 3.5 zwei Variablen vom Typ `Long`. Die Variable `lngSpalte` wird die Schleifensteuerung übernehmen und gleichermaßen auf die jeweils zu verarbeitende Zelle zeigen. Die Variable `lngSpalteMax` wird im Verlaufe des Makros die letzte belegte Zelle in der ersten Zeile erkunden und von dieser Zelle die Spaltennummer aufnehmen.

Entfärben Sie die erste Zeile über die Eigenschaft `ColorIndex`, der Sie die Konstante `xlColorIndexNone` zuweisen. Danach ermitteln Sie, wie gerade schon angedeutet, die letzte verwendete Zelle in der ersten Zeile, fragen mit der Eigenschaft `Column` die dazugehörige Spaltennummer ab und speichern diese Information in der Variablen `lngSpalteMax`. In der anschließenden `For ... Next`-Schleife bildet diese Variable das Schleifenabbruchkriterium. Innerhalb der Schleife prüfen Sie mit Hilfe der Anweisung `If` und der Funktion `IsDate`, ob in der jeweiligen Zelle überhaupt ein gültiges Datum steht. Wenn ja, dann erfolgt eine zweite Prüfung über die Funktion `Weekday`. Dieser Funktion übergeben Sie das jeweilige Datum. Im zweiten Argument dieser Funktion müssen Sie über die Konstante `vbMonday` angeben, dass die Woche bei uns mit dem Montag beginnt – für uns selbstverständlich, für Amerikaner fängt die eigentliche Woche mit dem Sonntag an. Wenn also die Woche mit dem Montag als erstem Tag der Woche beginnt, dann liefert uns die Funktion `Weekday` für den Montag die Zahl 1 und für den Sonntag den Wert 7 zurück. Daher fragen Sie mit der `If`-Anweisung ab, ob die Funktion `Weekday` für das jeweilige Datum einen Rückgabewert größer 5 zurückgibt. In diesem Fall handelt es sich um ein Wochenende, und die Spalte wird über die Eigenschaft `ColorIndex` eingefärbt.



**Abbildung 3.5** Alle Wochenenden wurden grün eingefärbt.

Neben der Funktion `IsDate` gibt es weitere Prüffunktionen, die Sie Tabelle 3.1 entnehmen können.

Funktion	Beschreibung
<code>IsEmpty</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob eine Variable initialisiert wurde.
<code>IsArray</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob eine Variable ein Datenfeld ist.
<code>IsDate</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob ein Ausdruck in ein Datum umgewandelt werden kann.
<code>IsError</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob ein Ausdruck ein Fehlerwert ist.
<code>IsNull</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob ein Ausdruck keine gültigen Daten ( <code>Null</code> ) enthält.
<code>IsNumeric</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob ein Ausdruck als Zahl ausgewertet werden kann.
<code>IsObject</code>	Gibt einen Wert vom Typ <code>Boolean</code> zurück, der angibt, ob ein Bezeichner eine Objektvariable darstellt.

**Tabelle 3.1** Die Prüffunktionen von Excel

### 3.3 Die Kurzform einer Bedingung

Neben der Verzweigung `If ... Then ... Else` gibt es eine weitere Möglichkeit, Werte zu überprüfen. Die Funktion lautet `IIf`.

Die Funktion `IIf` hat folgende Syntax:

```
IIf(expr, truepart, falsepart)
```

Mit dem Argument `expr` geben Sie den auszuwertenden Ausdruck an.

Das Argument `truepart` liefert den zurückgegebenen Wert oder Ausdruck, wenn `expr` den Wert `True` ergibt.

Das Argument `falsepart` stellt den zurückgegebenen Wert oder Ausdruck dar, wenn `expr` den Wert `False` liefert.

Diese Funktion wertet immer sowohl den Teil `truepart` als auch den Teil `falsepart` aus, auch dann, wenn nur einer von beiden Teilen zurückgegeben wird.

#### 3.3.1 Den Doppelklick auf eine Zelle abfangen

In der folgenden Aufgabe wird in `TABELLE1` der Doppelklick auf eine Zelle abgefangen. Dabei soll automatisch ein `X` gesetzt werden, wenn der Doppelklick auf eine leere Zelle in Spalte `B` durchgeführt wird. Steht in einer Zelle bereits der Buchstabe `X`, dann soll die Zelle wieder geleert werden.

Diese Aufgabe können Sie über ein Tabellenereignis lösen. Stellen Sie das Ereignis wie folgt ein:

- Führen Sie im `PROJEKT-Explorer` der Entwicklungsumgebung einen Doppelklick auf den Tabellennamen `TABELLE1` durch.
- Sie gelangen dadurch direkt hinter die Tabelle. Wählen Sie im Codefenster auf der rechten Seite im ersten Dropdown oben den Eintrag `Worksheet` aus. Dadurch wird zunächst das Ereignis `Worksheet_SelectionChange` eingestellt.
- Stellen Sie im zweiten Ereignis rechts daneben das Ereignis `Worksheet_BeforeDoubleClick` ein, und entfernen Sie danach das nicht benötigte Ereignis `Worksheet_SelectionChange`.
- Kompletieren Sie den noch leeren Ereignisrahmen wie folgt:

```
Private Sub Worksheet_BeforeDoubleClick _
    (ByVal Target As Range, Cancel As Boolean)

    If Target.Column = 2 Then
        Target.Value = IIf(Target.Value = "X", "", "X")
        Cancel = True
    End If

End Sub
```

**Listing 3.6** Im Wechsel eine Zelle mit einem `X` versehen bzw. das `X` wieder löschen

Das Tabellenereignis `Worksheet_BeforeDoubleClick` hat zwei Argumente. Das Argument `Target` gibt Auskunft darüber, welche Zelle doppelt angeklickt wurde. Über das Argument `Cancel` kann die standardmäßig dem Doppelklick zugewiesene Aktion, den Wechsel in den Editiermodus der Zelle, aufgehoben werden.

Mit Hilfe der Anweisung `If` und der Eigenschaft `Column` wird abgefragt, ob der gerade durchgeführte Doppelklick auf eine Zelle in der zweiten Spalte erfolgte. Wenn ja, dann wird mit der `IIf` Anweisung geprüft, ob nicht bereits ein `X` in der Zelle steht. Wenn ja, dann wird die Zelle geleert, ansonsten eben mit einem `X` bestückt. Setzen Sie das Argument `Cancel` auf den Wert `True`, um zu verhindern, dass Excel in den Editiermodus der Zelle wechselt.

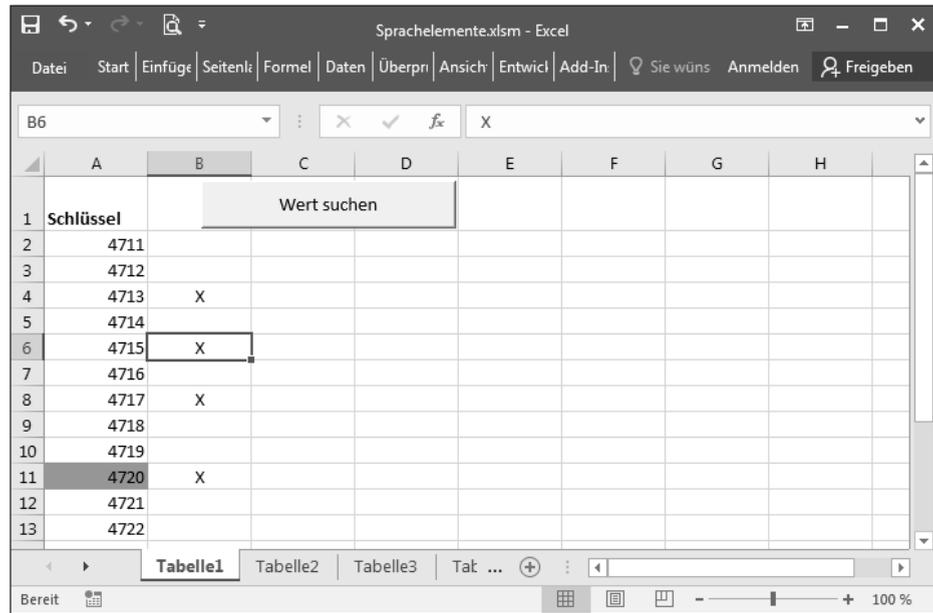


Abbildung 3.6 Ein toller Effekt – mit einem Doppelklick jeweils nach dem Füllen oder Leeren einer Zelle

### 3.4 Die Anweisung »Select Case« einsetzen

Mit Hilfe der Anweisung `Select Case` können Abfragen und Bedingungen leichter erfasst und übersichtlicher gelesen werden. Überhaupt bietet diese Anweisung, wenn es um einige Ausnahmen und Sonderfälle bei der Formulierung von Kriterien geht, im Vergleich zur `If`-Anweisung deutliche Vorteile.

Die Syntax dieser Anweisung lautet:

```
Select Case Ausdruck
Case Ausdrucksliste-n
    Anweisungen-n
Case Else
    elseAnweisungen
End Select
```

Unter dem Argument `Ausdruck` erfassen Sie einen beliebigen numerischen Ausdruck oder Zeichenfolgenausdruck, der ausgewertet werden soll. Im Argument `Ausdrucksliste-n` spezifizieren Sie die Abfrageoptionen näher. Dabei sind auch Vergleichsoperatoren, wie `To`, `Is` oder `Like` möglich.

Unter dem Argument `Anweisungen-n` geben Sie eine oder mehrere Anweisungen an, die ausgeführt werden sollen, wenn der abgefragte Ausdruck mit irgendeinem Element in `Ausdrucksliste-n` übereinstimmt.

Das Argument `elseAnweisungen` ist optional, insbesondere dann, wenn der abgefragte Ausdruck mit keinem Element im `Case`-Abschnitt übereinstimmen sollte.

Lernen Sie nun ganz konkret ein paar Beispiele kennen, wie Sie diese Anweisung in der Praxis einsetzen können.

#### 3.4.1 Excel-Version abfragen

In der folgenden Aufgabe soll ermittelt werden, mit welcher Excel-Version Sie arbeiten. Für diese Aufgabe wird die Eigenschaft `Version` ausgewertet, die über einen numerischen Wert die aktuelle Excel-Installation identifiziert.

```
Sub ExcelVersionAbfragen()
```

```
    MsgBox Application.Version
```

```
    Select Case Left(Application.Version, 2)
        Case 10
            MsgBox "Excel 2002"
        Case 11
            MsgBox "Excel 2003"
        Case 12
            MsgBox "Excel 2007"
        Case 14
            MsgBox "Excel 2010"
        Case 15
            MsgBox "Excel 2013"
        Case 16
            MsgBox "Excel 2016"
        Case Else
            MsgBox "Unbekannte Version von Excel", vbInformation

    End Select
```

```
End Sub
```

**Listing 3.7** Die eingesetzte Excel-Version ermitteln

Über die Funktion `Left` werden die ersten beiden Ziffern der Versionsnummer ausgewertet, die über die Eigenschaft `Version` abgefragt wurde. Innerhalb der `Select Case`-Anweisung wird die Versionsnummer überprüft. Trifft eine Bedingung zu, wird die entsprechende Meldung am Bildschirm ausgegeben, die angibt, um welche Excel-Version es sich handelt.

#### Info

Haben Sie es im Listing gesehen? Zwischen den beiden Versionen Excel 2007 und Excel 2010 wurde die Versionsnummer 13 übersprungen. Ob da wohl jemand abergläubisch ist?

### 3.4.2 Zahlenwerte prüfen

Im nächsten Beispiel werden Eingaben geprüft. Dabei soll ermittelt werden, in welchem Wertebereich die Eingabe vorgenommen wurde. Sehen Sie sich dazu das Makro aus Listing 3.8 an.

```
Sub ZahlUnWerteBereichAuswerten()
    Dim strEingabe As String

    strEingabe = InputBox("Geben Sie einen Wert ein!")

    If Len(strEingabe) <> 0 Then

        Select Case strEingabe

            Case 1 To 5
                MsgBox "Wert liegt zwischen 1 und 5"

            Case 6, 7, 8
                MsgBox "Wert ist entweder 6, 7 oder 8"

            Case 9 To 15
                MsgBox "Wert liegt zwischen 9 und 15"

            Case 16 To 100
                MsgBox "Wert liegt zwischen 16 und 100"

            Case Is > 100
                MsgBox "Wert liegt über 100"
```

```
        Case Else
            MsgBox "Es wurde kein gültiger Wert eingegeben!"

    End Select

End If

End Sub
```

#### Listing 3.8 In welchem Wertebereich liegt die eingegebene Zahl?

Wenden Sie die `Select Case`-Anweisung an, um die eingegebenen Werte zu überprüfen. In der ersten Abfrage wird kontrolliert, ob der eingegebene Wert im Bereich von 1 bis 5 liegt. Für die Prüfung eines Wertebereiches kann der Vergleichsoperator `To` eingesetzt werden. In der zweiten Abfrage werden einzelne Zahlenwerte durch Komma getrennt eingegeben. Wurde kein gültiger Zahlenwert eingegeben, kommt die Anweisung `Case Else` zum Tragen. Dieser Zweig würde beispielsweise dann angesteuert, wenn die eingegebene Zahl entweder 0 ist oder wenn es sich um eine negative Zahl handelt.

#### Reihenfolge der Zweige

Die Reihenfolge, in der Sie die einzelnen `Case`-Zweige anordnen, ist Ihnen frei überlassen. Wenn beispielsweise häufig Werte größer 100 eingegeben werden, dann kann der letzte Zweig aus Listing 3.8 auch als erste Bedingung eingesetzt werden. Dadurch wird die Geschwindigkeit in der Abarbeitung der Abfragen erhöht, da Zweig für Zweig von oben nach unten abgearbeitet wird. Wird eine Entsprechung gefunden, dann wird nach Erfüllung der Bedingung und Ausführen der damit definierten Aktion sofort direkt ans Ende der `Select Case`-Anweisung gesprungen.

Vergleichsoperator	Erklärung
<	kleiner als
<=	kleiner oder gleich
>	größer als
>=	größer oder gleich
=	gleich
<>	ungleich

Tabelle 3.2 Die Vergleichsoperatoren in Excel

### 3.4.3 Den Wochentag eines bestimmten Datums auslesen

Das Beispiel aus Listing 3.9 ermittelt, ob es sich beim aktuellen Tagesdatum um einen Werktag oder ein Wochenende handelt.

```
Sub WochentagErmittleIn()

    MsgBox "Heute ist " & Date

    Select Case Weekday(Date, vbMonday)
        Case Is < 6
            MsgBox "Werktag"

        Case Else
            MsgBox "Wochenende"

    End Select

End Sub
```

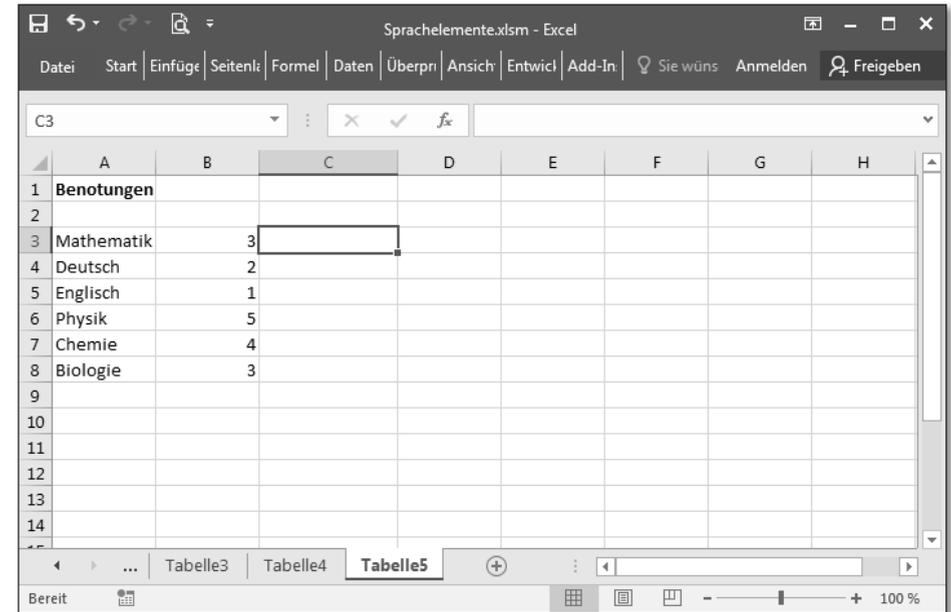
#### Listing 3.9 Das aktuelle Datum auswerten

Sie sollten immer den am häufigsten vorkommenden Wert an den Beginn der Select Case-Struktur stellen. In diesem Beispiel wird der aktuelle Tag ausgewertet. Es ist wahrscheinlicher, dass der aktuelle Tag ein Werktag ist, als dass der aktuelle Tag auf ein Wochenende trifft. Da bei Select Case die Prüfung von Zweig zu Zweig erfolgt, ist es natürlich besser, wenn gleich der erste Vergleich erfüllt wird. In diesem Fall werden weitere Zweige nicht mehr abgearbeitet, und die Verarbeitung der Prozedur geht nach End Select weiter.

Die Funktion Weekday hat wie vorher schon beschrieben zwei Argumente. Im ersten Argument übergeben Sie der Funktion ein Datum. Dieses Datum können Sie beispielsweise über die Funktion Date einsteuern, die Ihnen das aktuelle Tagesdatum liefert. Im zweiten Argument teilen Sie der Funktion mit, dass die Woche mit dem Montag beginnt. Das klingt auf den ersten Blick selbstverständlich, aber bei der Standardeinstellung dieser Funktion gilt der Sonntag als erster Tag der Woche. Die Funktion Weekday liefert Ihnen einen Wert zwischen 1 und 7 zurück. Wenn die Woche mit dem Montag beginnt, dann ist der Tag 1 der Woche der Montag und Tag 7 eben der Sonntag.

### 3.4.4 Benotungen über einen Autotext durchführen

Bei der folgenden Aufgabe liegt TABELLE5 wie in Abbildung 3.7 gezeigt vor. Darin finden Sie für die einzelnen Fächer die dazugehörigen Noten.



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H
1	Benotungen							
2								
3	Mathematik	3						
4	Deutsch	2						
5	Englisch	1						
6	Physik	5						
7	Chemie	4						
8	Biologie	3						
9								
10								
11								
12								
13								
14								

Abbildung 3.7 Fächer mit dazugehörigen Noten

In Spalte C soll jetzt eine Übersetzung der Noten in Texte stattfinden. Schreiben Sie dazu eine eigene benutzerdefinierte Funktion. Sehen Sie sich die Funktion aus Listing 3.10 an.

Function Benotung(rngZelle As Range)

```
    Select Case rngZelle.Value
        Case Is = 1: Benotung = "Sehr gut"
        Case Is = 2: Benotung = "Gut"
        Case Is = 3: Benotung = "Befriedigend"
        Case Is = 4: Benotung = "Ausreichend"
        Case Is = 5: Benotung = "Mangelhaft"
        Case Is = 6: Benotung = "Ungenügend"
        Case Else: Benotung = "keine gültige Zensur"
    End Select
```

End Function

#### Listing 3.10 Die Noten in Text aufschlüsseln

Wie Sie in Listing 3.10 sehen, wird die Select Case-Anweisung auch derart eingesetzt, dass über den Doppelpunkt ein Zeilenumbruch erspart werden kann. Dies ist aber nur dann sinnvoll, wenn es wirklich nur eine Aktion ist, die durchgeführt werden soll, wenn ein Zweig angesteuert wird.

Der Aufruf dieser Funktion kann direkt in der Zelle erfolgen. Markieren Sie dazu den Zellenbereich C3:C8, geben Sie die Formel =Benotung(B3) ein, und schließen Sie sie über die Tastenkombination **[Strg] + [↵]** ab.

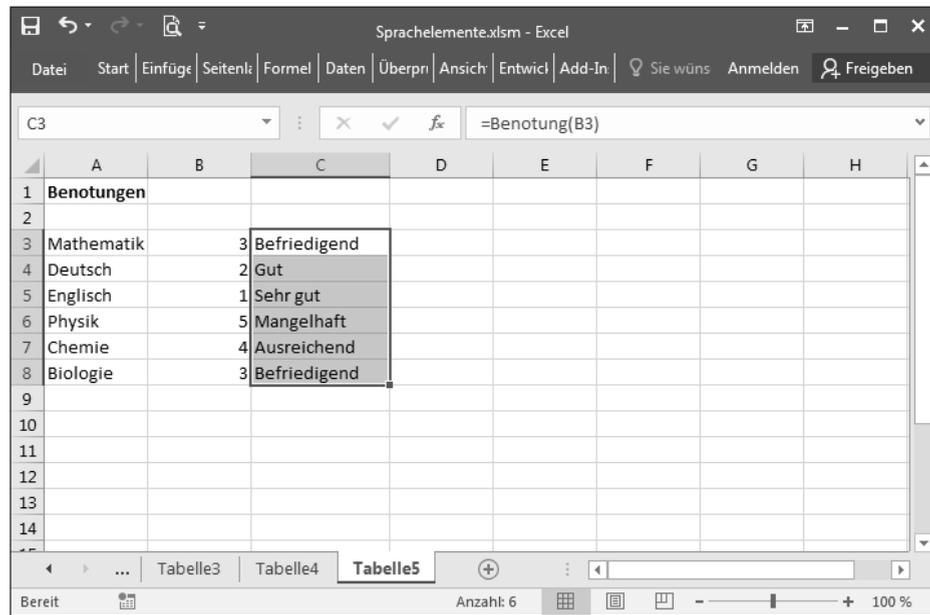


Abbildung 3.8 Ein Zeugnis mit Noten in Zahlen und in Textform

#### Info

Lernen Sie mehr über Funktionen in Kapitel 8, »Eigene Funktionen, reguläre Ausdrücke und API-Funktionen«.

### 3.5 Die »Enum«-Anweisung

Über die Enum-Anweisung generieren Sie eine Aufzählungsliste. Enum hat folgende Syntax:

```
[Public | Private] Enum Name
Elementname [= Konstantenausdruck]
Elementname [= Konstantenausdruck]
End Enum
```

Die Syntax der Enum-Anweisung besteht aus folgenden Bestandteilen:

- ▶ **Public:** optional. Legt fest, dass der Enum-Typ im gesamten Projekt sichtbar sein soll. Enum-Typen sind standardmäßig Public.
- ▶ **Private:** optional. Legt fest, dass der Enum-Typ nur innerhalb des Moduls sichtbar sein soll, in dem er angezeigt wird.
- ▶ **Name:** erforderlich. Der Name des Enum-Typs. Name muss ein zulässiger Visual-Basic-Kennzeichner sein und wird beim Deklarieren von Variablen oder Parametern vom Enum-Typ als Typ angegeben.
- ▶ **Elementname:** erforderlich. Ein zulässiger Visual-Basic-Kennzeichner, der den Namen angibt, der für ein konstituierendes Element des Enum-Typs verwendet werden soll.
- ▶ **Konstantenausdruck:** optional. Repräsentiert den Wert des Elements.

#### 3.5.1 Umsatz klassifizieren mit »Enum«

Im Beispiel aus Listing 3.11 wird eine Umsatzgruppe definiert, die aus drei »Untervariablen« besteht. Diese können Sie nach der Deklaration in einer Prozedur direkt ansprechen und ausgeben.

```
Public Enum MeinUmsatz
    Klein = 0
    Mittel = 1
    Groß = 2
End Enum

Dim Umsatz As MeinUmsatz

Sub VariableFüllenUndAusgeben()

    Select Case Tabelle6.Range("A1").Value
        Case Is >= 5000
            Umsatz = Groß
        Case 1000 To 5000
            Umsatz = Mittel
        Case Is < 1000
            Umsatz = Klein
        End Select

        MsgBox "Der Umsatz gehört in die Klasse " & Umsatz

    End Sub
```

Listing 3.11 Die »Enum«-Anweisung für die Klassifizierung von Umsätzen einsetzen

Deklarieren Sie zu Beginn außerhalb des Makros aus Listing 3.10 eine Aufzählung, indem Sie Texte mit Zahlen gleichsetzen und in die Enum-Struktur packen.

Danach deklarieren Sie eine Variable, die genau auf diese Struktur verweist. Im Makro selbst werten Sie Zelle A1 in TABELLE6 über die Select Case-Anweisung aus. Je nach Wert füllen Sie die Variable Umsatz. Beim Editieren werden Ihnen die drei möglichen Aufzählungen elegant in einem Dropdown angeboten. Geben Sie am Ende die ermittelte Zuordnung über die Funktion MsgBox am Bildschirm aus.

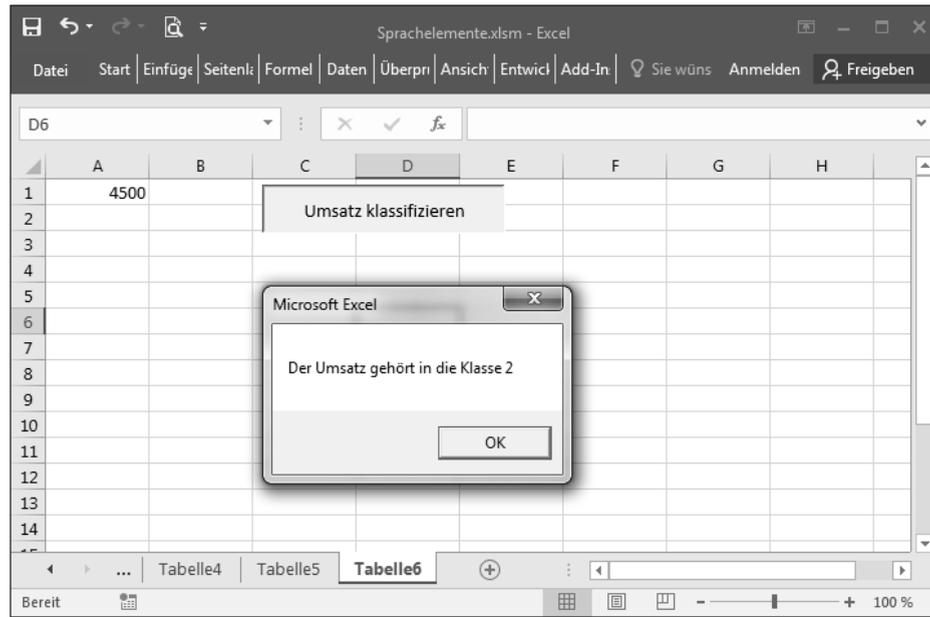


Abbildung 3.9 Der erfasste Umsatz wurde in die dazugehörige Gruppe eingegliedert.

### 3.6 Schleifen erstellen und verstehen

Eine Schleife ist ein Vorgang, der immer wieder gleich abläuft und so oft wiederholt wird, bis er beendet wird.

In Excel werden Schleifen dazu verwendet, Abläufe mehrmals hintereinander durchzuführen. Die Schleifen werden so lange durchlaufen, bis eine oder mehrere Bedingungen zutreffen, die dann einen Abbruch der Schleife bewirken. Je nach verwendeter Schleife findet die Abbruchprüfung am Anfang oder am Ende der Schleife statt.

Eine typische Schleife in Excel wäre das zeilenweise Abarbeiten einer Tabelle von oben nach unten. Für jede Zeile werden in der Schleife bestimmte Prüfungen durchgeführt. Eine typische Prüfung wäre beispielsweise der Vergleich einer Zelle mit einem Vorgabewert. Die Schleife sorgt dafür, dass jede Zeile gleichermaßen verarbei-

tet wird, bis die Verarbeitung an der letzten gefüllten Zelle der Tabelle angekommen ist. Hier erfährt die Schleife dann einen Abbruch. Manche Schleifen können diesen »normalen« Abbruch auch innerhalb der Schleife quasi zwischendurch erfahren, wenn die Verarbeitung beispielsweise auf ein Problem stößt, bei dem es besser ist, die Schleife vorzeitig zu beenden.

In Excel gibt es eine ganze Anzahl verschiedener Schleifentypen, die Sie auf den nächsten Seiten kennenlernen werden.

#### 3.6.1 Die »For ... Next«-Schleife

Sie können die Schleife For ... Next verwenden, um einen Block von Anweisungen eine unbestimmte Anzahl von Wiederholungen auszuführen. For ... Next-Schleifen verwenden eine Zählervariable, deren Wert mit jedem Schleifendurchlauf erhöht oder verringert wird. Sie müssen daher nicht daran denken, den Zähler selbst hoch- oder herunterzusetzen.

Die Syntax dieser Schleife lautet:

```
For Zähler = Anfang To Ende [Step Schritt]
    [Anweisungen]
[Exit For]
    [Anweisungen]
Next [Zähler]
```

Das Argument Zähler ist erforderlich und besteht aus einer numerischen Variablen, die als Schleifenzähler dient.

Das Argument Anfang repräsentiert den Startwert von Zähler.

Mit dem Argument Ende wird der Endwert des Zählers festgelegt. Das Argument Schritt ist optional. Hier können Sie den Wert bestimmen, um den Zähler bei jedem Schleifendurchlauf verändert wird. Falls kein Wert angegeben wird, ist die Voreinstellung 1.

Unter Anweisungen stehen eine oder mehrere Anweisungen zwischen For und Next, die so oft wie angegeben ausgeführt werden.

Innerhalb einer Schleife können Sie eine beliebige Anzahl von Exit For-Anweisungen setzen, die eine alternative Möglichkeit darstellen, die Schleife vorzeitig zu verlassen.

Lernen Sie jetzt einige typische Aufgabenstellungen für diese Schleife kennen.

#### Die Farbpalette von Excel auslesen

Im Beispiel aus Listing 3.12 werden in TABELLE7 alle verfügbaren Farben, die Excel für den Hintergrund einer Zelle zur Verfügung stellt, dargestellt.

```

Sub FarbenErmitteln()
    Dim wksBlatt As Worksheet
    Dim intZ As Integer

    Set wksBlatt = Tabelle7

    For intZ = 1 To 56

        With wksBlatt
            .Cells(intZ, 1).Value = intZ
            .Cells(intZ, 2).Interior.ColorIndex = intZ
        End With

    Next intZ

End Sub

```

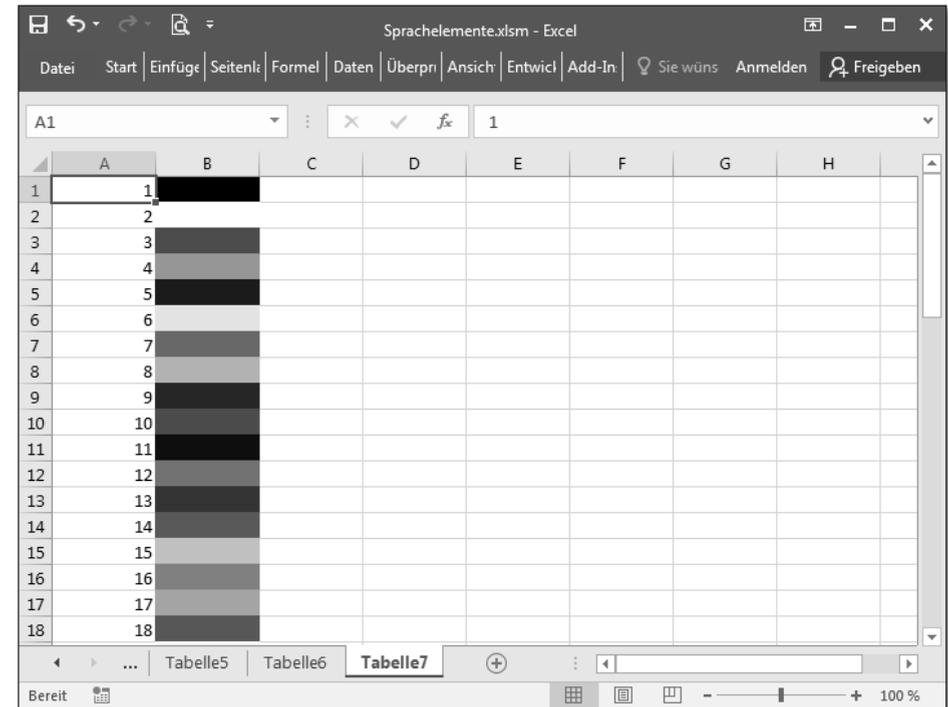
**Listing 3.12** Die 56 Standardfarben von Excel in einer Tabelle ausgeben

Zunächst wird eine Zählvariable vom Typ Integer deklariert. Danach deklarieren Sie die Objektvariable wksBlatt mit dem Datentyp Worksheet. Geben Sie dann über die Anweisung Set bekannt, dass Sie mit TABELLE7 arbeiten möchten. Setzen Sie eine Schleife auf, die genau 56 Mal durchlaufen wird, da es nur 56 Standardfarben für den Hintergrund einer Zelle gibt. Wir haben hier also eine vorher festgelegte Wiederholungszahl für die Schleife. Innerhalb der Schleife können Sie über die With-Anweisung die Schreibarbeit ein wenig reduzieren, da die Anweisungen für die aktive Tabelle gelten.

Über die Cells-Auflistung werden die Zellen gezielt angesteuert und von oben nach unten gefüllt. Diese Auflistung hat genau zwei Argumente: Das erste Argument gibt die Zeilennummer an, das zweite enthält die Spaltennummer. In diesem Beispiel werden die Informationen zeilenweise in die Spalten A (= 1) und B (= 2) geschrieben. Die erste Spalte zeigt die eindeutige Farbnummer. Bei Spalte B wird genau diese Farbnummer genutzt, um den Hintergrund der Zelle (Interior) über die Eigenschaft ColorIndex zu füllen.

#### Hinweis

Wir haben im Beispiel aus Listing 3.12 die Zählvariable doppelt verwendet. Zum einen wird dadurch die Zeilennummer verwaltet (von Zeile 1 bis Zeile 56), zum anderen werden die verfügbaren Farben (von 1 bis 56) den Zellen zugewiesen.



**Abbildung 3.10** Die 56 Standardfarben von Excel auf einen Blick

#### Die Schablone für eine Verarbeitung von Zeilen von oben nach unten

Die allermeisten Aufgaben in Excel befassen sich direkt mit Tabellen, die Zeile für Zeile von oben nach unten abgearbeitet werden. Daher möchte ich Ihnen die Mutter aller Schablonen für eine Schleife vorstellen, über die Sie die meisten Aufgaben lösen können. Diese Schablone ist in 5 Schritte eingeteilt, und das Schöne daran ist, dass die ersten 4 Schritte immer gleich sind, egal, was auch immer Sie mit der Tabelle anstellen. Das bedeutet, dass Sie sich zukünftig voll und ganz auf Schritt 5, die eigentliche Aufgabe, konzentrieren können.

Im Makro aus Listing 3.13 sehen Sie die Schablone für diese Schleife.

```

Sub Schablone_VonObenNachUnten()
    'Verwendung: Zeilenverarbeitung
    'Schritt 1: Deklaration von Variablen
    Dim lngZeile As Long 'Repräsentiert den Zähler f. die Schleife und die Zeile
    Dim lngZeileMax As Long 'Repräsentiert die letzte belegte Zeile

    'Schritt 2: Festlegen der Verarbeitungstabelle
    With Tabelle8

```

```

'Schritt 3: Ermitteln der letzten verwendeten Zeile
lngZeileMax = .UsedRange.Rows.Count

'Schritt 4: Aufsetzen der Schleife
For lngZeile = 2 To lngZeileMax

'Schritt 5: Eigentliche Aufgabe

Next lngZeile

End With

End Sub

```

**Listing 3.13** Die Schablone für eine zeilenweise Verarbeitung einer Tabelle von oben nach unten

In Schritt 1 deklarieren Sie zwei Variablen vom Typ Long mit Hilfe der Anweisung Dim. Danach steht Ihnen ein reservierter Platz mit dem Namen Zeile im Arbeitsspeicher zur Verfügung. Beide Variablen haben jetzt den Wert 0, das heißt, die Variablen müssen nicht gesondert initialisiert werden. Über die Variable Zeile steuern Sie später die Schleife und zeigen direkt auf die zu verarbeitende Zeile. In der Variablen lngZeileMax speichern Sie nachher die Zeilennummer der zuletzt benutzten Zeile in der Tabelle.

In Schritt 2 legen Sie die zu verarbeitende Tabelle über die Anweisung With fest. Immer wenn Sie danach auf die so festgelegte Tabelle zugreifen möchten, reicht es, wenn Sie anstatt des vollen Namens der Tabelle einen Punkt setzen. Excel ergänzt dann den bei With definierten Tabellennamen zur Laufzeit des Makros.

In Schritt 3 ermitteln Sie die Zeilennummer der letzten belegten Zeile der Tabelle. Dazu müssen Sie wissen, dass jede gefüllte Tabelle einen benutzten Bereich hat. Dieser Bereich kann über die Eigenschaft UsedRange abgefragt werden. Bei der Ermittlung dieses Bereiches sucht Excel nach der letzten gefüllten Zelle der Tabelle nach rechts und nach unten. Nehmen wir einmal an, es stünde jeweils ein Wert in Zelle A1 und D10. Dann wäre der benutzte Bereich A1:D10. Sie können das ruhig einmal ausprobieren und dann in das Direktfenster der Entwicklungsumgebung folgende Zeile eingeben, um den benutzten Bereich der aktiven Tabelle abzufragen:

```
?activesheet.usedrange.address
```

Der benutzte Bereich einer Tabelle umfasst eine bestimmte Anzahl von Zeilen, die Sie über den Befehl Rows.Count abfragen und in der Variablen lngZeileMax »parken«. Sie wissen also jetzt, wie lange Sie die Schleife anschließend nach unten »jagen« müssen.

In Schritt 4 wird die Schleife aufgesetzt. In der Regel beginnt diese in Zeile 2, wenn angenommen wird, dass die Überschrift nicht verarbeitet werden soll. Die Schleife

fängt also bei der zweiten Zeile an und arbeitet sich Zeile für Zeile nach unten durch, bis die letzte Zeile (=lngZeileMax) erreicht wird. Über die Anweisung Next lngZeile wird die Variable lngZeile automatisch um den Wert 1 erhöht. Damit zeigen Sie bei jedem Schleifendurchlauf jeweils auf die nächste Zeile.

In Schritt 5 findet die eigentliche Aufgabe statt. Hier können beispielsweise Prüfungen erfolgen, die eine Zelle je nach dem enthaltenen Wert formatieren.

### Die Schablone für eine Verarbeitung von Zeilen von unten nach oben

Eine For ... Next-Schleife können Sie in einer Tabelle auch von unten nach oben Zeile für Zeile durchlaufen lassen. Sie fragen sich vielleicht jetzt: »Warum sollte ich das denn überhaupt wollen?«

Bei dieser umgekehrten Laufrichtung handelt es sich um eine Notwendigkeit beim Löschen von Zeilen aus einer Tabelle. Nur wenn Sie bei diesem Vorhaben unten anfangen und oben aufhören, sind Sie erfolgreich.

Die Begründung dafür liegt am Verhalten von Excel, denn wenn Sie eine Zeile aus einer Tabelle löschen, rutschen die darunterliegenden Zeilen eine Zeile nach oben. Das würde beim standardmäßig üblichen Abarbeiten der Tabelle von oben nach unten zu einer Veränderung des Zählers führen. Excel würde quasi jeweils eine Zeile überspringen und nicht sauber Zeilen löschen, wenn gleich mehrere zu löschende Zeilen direkt untereinander lägen.

Aber auch bei anderen Aufgabenstellungen kann eine Abarbeitung einer Tabelle vom Ende zum Beginn der Tabelle sicherlich reizvoll sein.

Sehen Sie sich jetzt einmal den Makrorahmen aus Listing 3.14 an, der eine Tabelle von unten nach oben abarbeitet.

```

Sub Schablone_VonUntenNachOben()
'In der Regel beim Löschen von Zeilen in Verwendung
'Schritt 1: Deklaration von Variablen
Dim lngZeile As Long
Dim lngZeileMax As Long

'Schritt 2: Festlegen der Verarbeitungstabelle
With Tabelle8

'Schritt 3: Ermitteln der letzten verwendeten Zeile
lngZeileMax = .UsedRange.Rows.Count

'Schritt 4: Aufsetzen der Schleife
For lngZeile = lngZeileMax To 2 Step -1

```

```
'Schritt 5: Löschkriterium festlegen

Next lngZeile

End With

End Sub
```

**Listing 3.14** Die Schablone für eine zeilenweise Verarbeitung einer Tabelle von unten nach oben

In Schritt 1 deklarieren wir zwei Variablen vom Typ Long mit Hilfe der Anweisung Dim. Danach steht uns ein reservierter Platz mit dem Namen Zeile im Arbeitsspeicher zur Verfügung.

In Schritt 2 legen wir die zu verarbeitende Tabelle über die Anweisung With fest.

In Schritt 3 ermitteln wir die Zeilennummer der letzten belegten Zeile der Tabelle.

In Schritt 4 wird die Schleife aufgesetzt. Dabei beginnen wir bei der letzten Zeile der Tabelle und enden vor der Überschrift der Tabelle. Damit wir von unten nach oben kommen, müssen wir die Schrittweite auf den Wert -1 setzen. Damit wird bei jedem Schleifendurchlauf der Wert 1 von der Variablen Zeile abgezogen.

In Schritt 5 legen wir das Löschkriterium fest. Als Löschkriterien könnten wir beispielsweise doppelte Werte, leere Zellen oder Zellen, die bestimmte Inhalte aufweisen, heranziehen.

#### Die Schablone für eine Verarbeitung von Spalten von links nach rechts

Die beiden Laufrichtungen einer Schleife von oben nach unten und von unten nach oben sind jetzt abgehandelt. Wenn es um die Verarbeitung von Spalten geht, dann kommt der Schleifenrahmen aus Listing 3.15 zum Einsatz.

```
Sub Schablone_VonLinksNachRechts()
    'Spaltenverarbeitung
    'Schritt 1: Deklaration von Variablen
    Dim lngSpalte As Long 'Repräsentiert die zu verarbeitende Spalte
    Dim lngSpalteMax As Long 'Repräsentiert die letzte gefüllte Spalte
    'Schritt 2: Festlegen der Verarbeitungstabelle
    With Tabelle12

        'Schritt 3: Ermitteln der letzten gefüllten Spalte
        lngSpalteMax = .UsedRange.Columns.Count

        'Schritt 4: Aufsetzen der Schleife
```

```
For lngSpalte = 1 To lngSpalteMax

    'Schritt 5: Eigentliche Aufgabe

Next lngSpalte

End With

End Sub
```

**Listing 3.15** Die Schablone für eine spaltenweise Verarbeitung einer Tabelle von links nach rechts

In Schritt 1 deklarieren Sie zwei Variablen vom Typ Long mit Hilfe der Anweisung Dim. Danach steht Ihnen ein reservierter Platz mit dem Namen lngSpalte im Arbeitsspeicher zur Verfügung. Beide Variablen haben jetzt den Wert 0. Über die Variable lngSpalte steuern Sie später die Schleife und zeigen direkt auf die zu verarbeitende Spalte. In der Variablen lngSpalteMax speichern Sie nachher die Spaltennummer der letzten benutzten Spalte in der Tabelle.

In Schritt 2 legen Sie die zu verarbeitende Tabelle über die Anweisung With fest.

In Schritt 3 ermitteln Sie die Spaltennummer der letzten belegten Spalte der Tabelle. Dazu müssen Sie wissen, dass jede gefüllte Tabelle einen benutzten Bereich hat. Dieser Bereich kann über die Eigenschaft UsedRange abgefragt werden. Dieser benutzte Bereich besteht aus einer bestimmten Anzahl von Spalten, die Sie über die Funktion Count zählen und in der Variablen lngSpalteMax zwischenspeichern können.

In Schritt 4 setzen Sie die Schleife auf, die bei der ersten Spalte beginnt und sich dann nach rechts Spalte für Spalte durcharbeitet, bis die letzte Spalte (= lngSpalteMax) erreicht ist.

In Schritt 5 formulieren Sie die eigentliche Aufgabe der Schleife. Diese Aufgabe ist in der Schablone noch ausgespart.

#### Die Schablone für eine Verarbeitung von Spalten von rechts nach links

Diese Schablone aus Listing 3.16 wird dann eingesetzt, wenn es darum geht, bestimmte Spalten aus einer Tabelle zu entfernen. Was schon für die Löschung von Zeilen galt, trifft auch auf die Löschung von Spalten 1 : 1 zu.

```
Sub Schablone_VonRechtsNachLinks()
    'bei der Löschung von Spalten in Verwendung
    'Schritt 1: Deklaration von Variablen
    Dim lngSpalte As Long 'Repräsentiert die verarbeitende Spalte
    Dim lngSpalteMax As Long 'Repräsentiert die letzte gefüllte Spalte
    'Schritt 2: Festlegen der Verarbeitungstabelle
```

```

With Tabelle13
'Schritt 3: Ermitteln der letzten gefüllten Spalte
lngSpalteMax = .UsedRange.Columns.Count

'Schritt 4: Aufsetzen der Schleife
For lngSpalte = lngSpalteMax To 1 Step -1

'Schritt 5: Eigentliche Aufgabe

Next lngSpalte

End With
End Sub

```

**Listing 3.16** Die Schablone für eine spaltenweise Verarbeitung einer Tabelle von rechts nach links (Löschen von Spalten)

Da die Schritte 1 bis 3 dieselben wie beim Rahmen für die Verarbeitung einer Tabelle von links nach rechts sind, steigen wir in Schritt 4 ein. Der Schleifenzähler wird auf die letzte Spalte der Tabelle gesetzt, und die Schleife arbeitet sich Spalte für Spalte von rechts nach links durch.

### Praxis, Praxis, Praxis

Auf den folgenden Seiten werden Sie den Einsatz dieser vier vorgestellten Schablonen anhand einiger praktischer Beispiele sehen.

### Werte oberhalb eines Referenzwertes aufspüren

Bei der folgenden Aufgabenstellung werden die Werte in Spalte B von TABELLE9 ausgewertet. Dabei sollen diese Werte immer mit dem Vorgabewert aus Zelle E1 verglichen werden und je nach Wert entsprechend eingefärbt werden.

Verwenden Sie für diese Aufgabe die vorher erstellte Schablone *Schablone\_VonObenNachUnten*, und passen Sie sie an. Das könnte dann so wie in Listing 3.17 gezeigt aussehen.

```

Sub WerteOberhalbReferenzwert()
Dim lngZeile As Long
Dim lngZeileMax As Long

With Tabelle9

lngZeileMax = .UsedRange.Rows.Count

```

```

For lngZeile = 2 To lngZeileMax

If .Range("B" & lngZeile).Value >= .Range("E1").Value Then
.Range("B" & lngZeile).Interior.ColorIndex = 4
Else
.Range("B" & lngZeile).Interior.ColorIndex = xlColorIndexNone
End If

Next lngZeile

End With

End Sub

```

**Listing 3.17** Werte oberhalb des Referenzwertes werden automatisch eingefärbt.

	A	B	C	D	E	F	G	H
1	Datum	Wert		Referenzwert	850			
2	01.09.2015	767						
3	02.09.2015	353						
4	03.09.2015	439						
5	04.09.2015	992						
6	05.09.2015	992						
7	06.09.2015	146						
8	07.09.2015	746						
9	08.09.2015	626						
10	09.09.2015	588						
11	10.09.2015	976						
12	11.09.2015	910						
13	12.09.2015	487						
14	13.09.2015	456						
15	14.09.2015	189						
16	15.09.2015	484						
17								

**Abbildung 3.11** Die Ausgangssituation – welche Werte liegen über dem Referenzwert aus Zelle E1?

Da ich die Schritte 1 bis 4 bereits beim Rahmen in Listing 3.13 beschrieben habe, können wir gleich mitten in die Schleife aus Listing 3.17 gehen und die dort verwendeten Befehle besprechen.

Mit einer If-Anweisung prüfen Sie den Wert einer jeden Zelle aus Spalte B, die Sie ja über die Schleife von oben nach unten durchlaufen. Ist dieser Wert größer als oder gleich dem in Zelle E1, dann färben Sie den Hintergrund der Zelle, indem Sie dem Objekt Interior über die Eigenschaft ColorIndex den Wert 4 zuweisen.

	A	B	C	D	E	F	G	H
1	Datum	Wert		Referenzwert	850			
2	01.09.2015	767						
3	02.09.2015	353						
4	03.09.2015	439						
5	04.09.2015	992						
6	05.09.2015	992						
7	06.09.2015	146						
8	07.09.2015	746						
9	08.09.2015	626						
10	09.09.2015	588						
11	10.09.2015	976						
12	11.09.2015	910						
13	12.09.2015	487						
14	13.09.2015	456						
15	14.09.2015	189						
16	15.09.2015	484						

Abbildung 3.12 Alle Werte oberhalb des Referenzwertes wurden gekennzeichnet.

### Duplikate in einer Liste kennzeichnen

Beim folgenden Beispiel liegt eine Liste mit einigen doppelten Werten vor. Ihre Aufgabe besteht nun darin, diese doppelten Werte aufzuspüren und zu kennzeichnen. Sehen Sie sich dazu einmal Abbildung 3.13 an.

Auf den ersten Blick ist es gar nicht so einfach, die doppelten Werte zu erkennen. Für ein Makro ist das kein Problem. Sehen Sie sich dazu das Makro aus Listing 3.18 an.

```
Sub DuplikateAufspüren()
    Dim lngZeile As Long
    Dim lngZeileMax As Long

    With Tabelle10
        lngZeileMax = .UsedRange.Rows.Count

        For lngZeile = 2 To lngZeileMax
```

```
    If Application.WorksheetFunction.CountIf(.Range("A:A"), _
        .Range("A" & lngZeile).Value) > 1 Then
        .Range("A" & lngZeile).Interior.ColorIndex = 4
    Else
        .Range("A" & lngZeile).Interior.ColorIndex = xlColorIndexNone
    End If
```

```
Next lngZeile
```

```
End With
```

```
End Sub
```

Listing 3.18 Doppelte Werte aufspüren und kenntlich machen

	A	B	C	D	E	F	G	H
1	Nummern							
2		1						
3		5						
4		5						
5		6						
6		7						
7		4						
8		10						
9		6						
10		1						
11		8						
12		3						
13		2						
14		8						
15		5						

Abbildung 3.13 Hier kommen einige Nummern doppelt vor – welche?

Im Inneren der Schleife wird der Zugriff auf alle Tabellenfunktionen von Excel über die Eigenschaft Worksheetfunction angezapft. In dieser Auflistung finden Sie unter anderem die Tabellenfunktion ZÄHLENWENN (englisch countIf), die Sie einsetzen können, um die Duplikate zu ermitteln. Der Rest ist Formsache.

Denken Sie daran, im Else-Zweig der Abfrage die Farbe wieder zurückzusetzen. Es könnte ja sein, dass Sie einen ehemals doppelten in einen einmaligen Wert korrigieren. Dann muss aus der ehemals grünen Zelle schließlich wieder eine farblose Zelle werden. Die Zuweisung »keine Farbe« wird über die Konstante `xlColorIndexNone` erreicht, die der Eigenschaft `ColorIndex` zugewiesen wird.

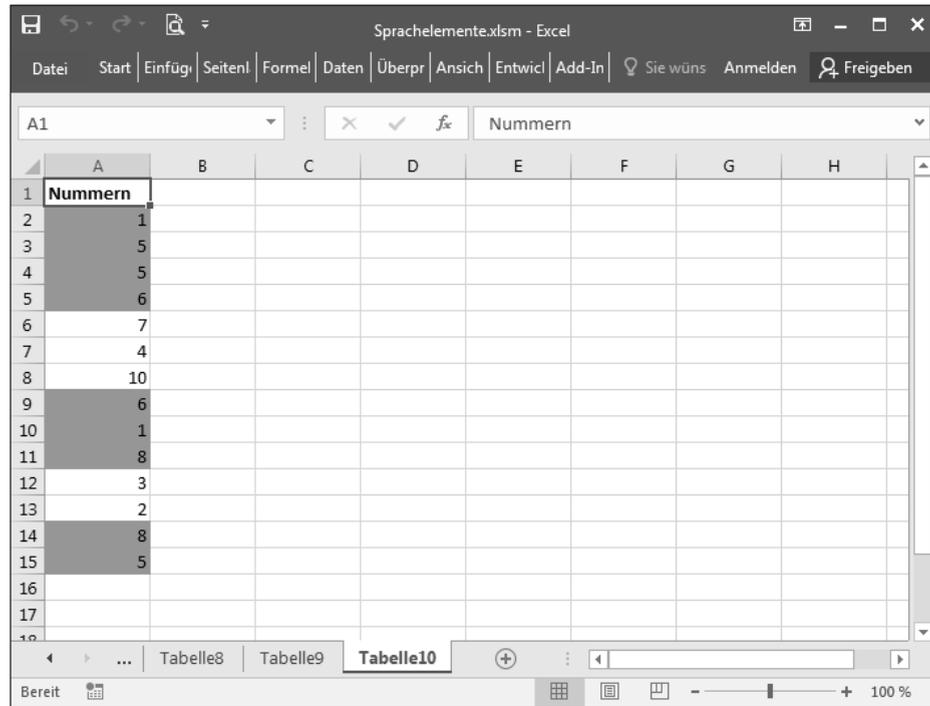


Abbildung 3.14 Alle doppelten Nummern wurden eingefärbt.

### Leere Zeilen entfernen

Bei der folgenden Aufgabe liegen in TABELLE11 Daten vor. Einige Zeilen sind dabei leer, andere teilweise gefüllt.

Zur Lösung der Aufgabe können Sie die Schablone *Schablone\_VonUntenNachOben* verwenden und etwas anpassen. So geschehen im Makro aus Listing 3.19.

```
Sub LeereZeilenLöschen()
    Dim lngZeile As Long
    Dim lngZeileMax As Long

    With Tabelle11
        lngZeileMax = .UsedRange.Rows.Count
        For lngZeile = lngZeileMax To 1 Step -1
```

```
        If Application.WorksheetFunction.CountA(.Rows(lngZeile)) = 0 Then
            .Rows(lngZeile).Delete
            lngZz = lngZz + 1
        End If
    Next lngZeile
End With
```

```
MsgBox lngZz & " Zeilen wurden gelöscht!", vbInformation
End Sub
```

Listing 3.19 Leere Zeilen entfernen

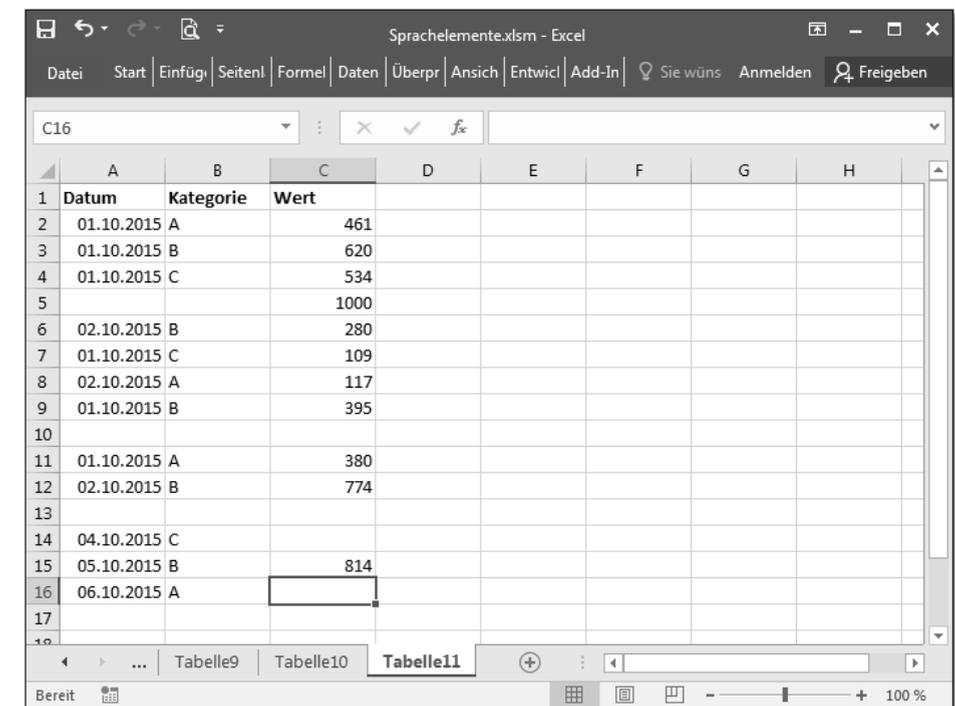


Abbildung 3.15 Nur die wirklich leeren Zeilen sollen entfernt werden.

Die Schleife muss beim Löschen von Zeilen in einer Tabelle von unten nach oben laufen. In der Schleife selbst wenden Sie die Tabellenfunktion `CountA` (deutsch `ANZAHL2`) an, um zu ermitteln, ob in der kompletten Zeile überhaupt irgendein Eintrag (Wert oder Text) steht. Wenn nicht, dann kann die komplette Zeile über die Methode `Delete` entfernt werden. Bei jeder Löschung erhöhen Sie die Zählvariable `lngZz` um den Wert 1.

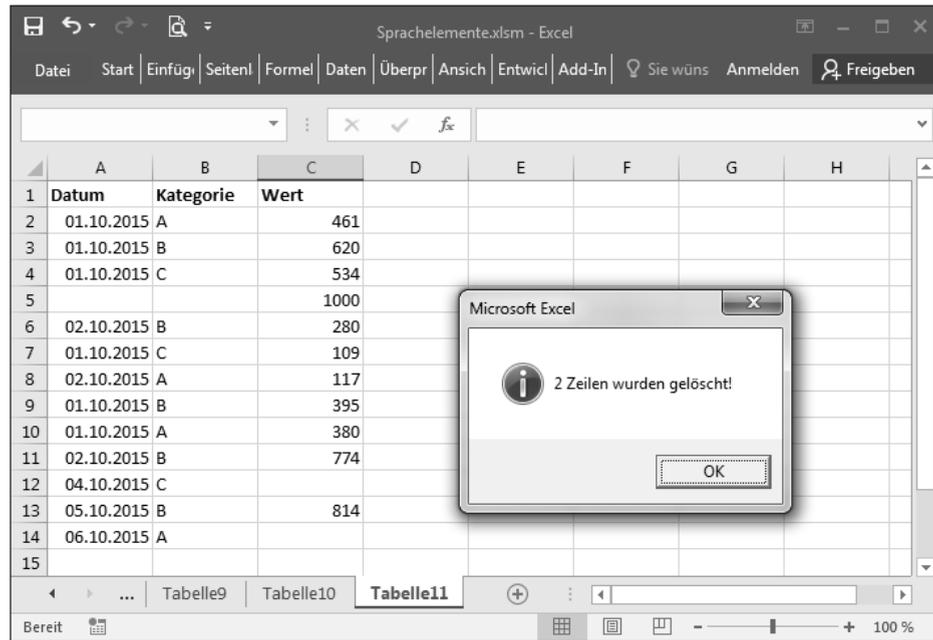


Abbildung 3.16 Die leeren Zeilen wurden entfernt und dabei nebenher noch gezählt.

### Wochenenden und Werktage kennzeichnen

Bei der Aufgabe aus Listing 3.20 sollen in TABELLE12 die Wochenenden sowie die normalen Werktage gekennzeichnet werden. Dabei erhalten Wochenenden die Farbe Grün, die Werktage hingegen werden mit der Hintergrundfarbe Gelb formatiert. Für diese Aufgabe können Sie die *Schablone\_VonLinksNachRechts* verwenden und etwas anpassen.

```
Sub WochenendenKennzeichnenSpalten()
```

```
    Dim lngSpalte As Long
```

```
    Dim lngSpalteMax As Long
```

```
    With Tabelle12
```

```
        lngSpalteMax = .UsedRange.Columns.Count
```

```
        For lngSpalte = 1 To lngSpalteMax
```

```
            If Weekday(.Cells(1, lngSpalte).Value, vbMonday) > 5 Then
```

```
                .Cells(1, lngSpalte).Interior.ColorIndex = 4
```

```
            Else
```

```
                .Cells(1, lngSpalte).Interior.ColorIndex = 6
```

```
            End If
```

```
        Next lngSpalte
```

```
    End With
```

```
End Sub
```

### Listing 3.20 Wochenende und Werktage in Spalten kennzeichnen

In der Schleife selbst kommt die Funktion `Weekday` zum Einsatz, die ich in diesem Kapitel schon besprochen habe. Daher gehe ich an dieser Stelle nicht mehr darauf ein.

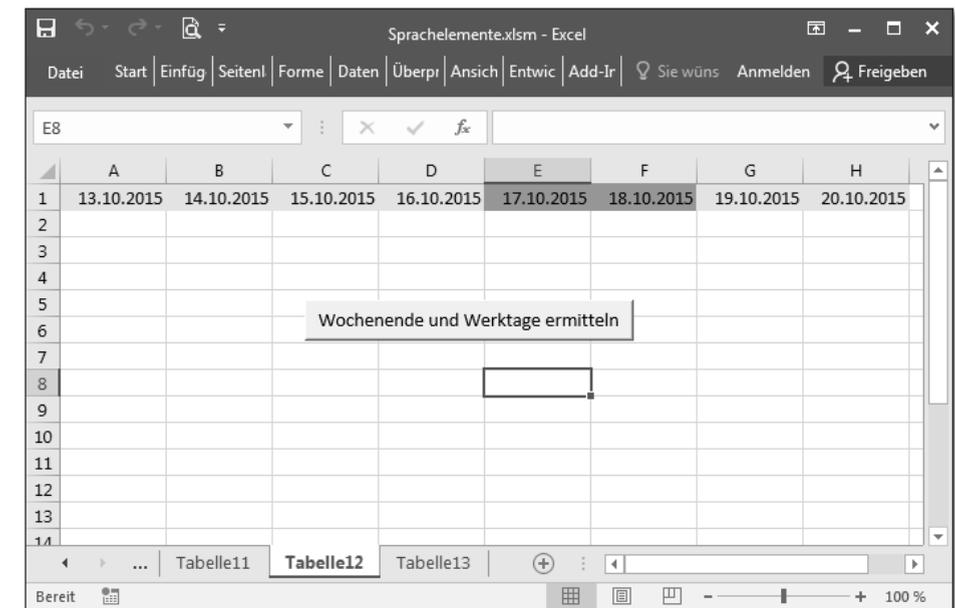


Abbildung 3.17 Alle Tage in der Datumsleiste wurden gekennzeichnet.

### Bestimmte Spalten aus einer Tabelle entfernen

Bei der folgenden Aufgabenstellung liegen in TABELLE13 einige Daten vor. Einige Spalten in dieser Tabelle werden nicht mehr benötigt und können gelöscht werden.

Für diese Aufgabe können Sie die *Schablone\_VonRechtsNachLinks* heranziehen und anpassen. Sehen Sie sich dazu die Umsetzung in Listing 3.21 an:

```
Sub BestimmteSpaltenLöschen()
```

```
    Dim lngSpalte As Long
```

```
    Dim lngSpalteMax As Long
```

```
    With Tabelle13
```

```
        lngSpalteMax = .UsedRange.Columns.Count
```

```

For lngSpalte = lngSpalteMax To 1 Step -1

    Select Case UCase(.Cells(1, lngSpalte).Value)

        Case "SUMME", "Nr"
            .Columns(lngSpalte).Delete
        Case Else
            'keine Aktion
    End Select

    Next lngSpalte

End With

End Sub

```

Listing 3.21 Bestimmte Spalten ersatzlos entfernen

	A	B	C	D	E	F	G	H
1	Datum	Nr	Kst	Bezeichnung	Konto	Summe	Kommentar	
2	01.05.2015	1	4788	KX1235	3416	1.851,00 €		
3	01.05.2015	2	4788	KX1235	3414	3.089,00 €	bitte prüfen	
4	01.05.2015	3	4788	KX1235	3414	3.661,00 €		
5	01.05.2015	4	4788	KX1235	3414	2.211,00 €		
6	01.05.2015	5	4789	KX1236	3417	4.709,00 €		
7	01.05.2015	6	4789	KX1236	3417	3.884,00 €		
8	01.05.2015	7	4789	KX1236	3416	3.247,00 €	bitte prüfen	
9	01.05.2015	8	4789	KX1237	3416	1.653,00 €		
10	01.05.2015	9	4799	KX1237	3417	3.639,00 €		
11	01.05.2015	10	4799	KX1237	3417	3.202,00 €	bitte prüfen	
12	01.05.2015	11	4799	KX1237	3415	3.187,00 €		
13	01.05.2015	12	4799	KX1241	3414	1.503,00 €		

Abbildung 3.18 Die Spalten »Nr« und »Summe« sollen entfernt werden.

In der rückwärts laufenden Schleife wenden Sie die `Select Case`-Anweisung an, um gleich mehrere Spaltenentitel abzuhandeln. Mit Hilfe der Funktion `UCase` können Sie Excel dazu bewegen, nicht zwischen Groß- und Kleinschreibung zu unterscheiden. Löschen Sie nicht benötigte Spalten, indem Sie die Methode `Delete` auf die entsprechende Spalte anwenden.

	A	B	C	D	E	F	G	H
1	Datum	Nr	Kst	Bezeichnung	Konto	Kommentar		
2	01.05.2015	1	4788	KX1235	3416			
3	01.05.2015	2	4788	KX1235	3414	bitte prüfen		
4	01.05.2015	3	4788	KX1235	3414			
5	01.05.2015	4	4788	KX1235	3414			
6	01.05.2015	5	4789	KX1236	3417			
7	01.05.2015	6	4789	KX1236	3417			
8	01.05.2015	7	4789	KX1236	3416	bitte prüfen		
9	01.05.2015	8	4789	KX1237	3416			
10	01.05.2015	9	4799	KX1237	3417			
11	01.05.2015	10	4799	KX1237	3417	bitte prüfen		
12	01.05.2015	11	4799	KX1237	3415			
13	01.05.2015	12	4799	KX1241	3414			

Abbildung 3.19 Die nicht relevanten Spalten wurden entfernt.

### 3.6.2 Die »For Each ... Next«-Schleife

Bei der `For Each ... Next`-Schleife handelt es sich um die schnellste Schleife, die Excel zur Verfügung hat. Diese Schleife wird mit Objekten wie Zellen, Tabellen und Mappen eingesetzt. Alles, was Sie dafür brauchen, sind das Objekt an sich und die dazugehörige Objektvariable. Diese Schleife gehört zu den leicht verständlichen Schleifen, wenn Sie bei der Deklaration der Variablen sinnvolle Namen definieren. Die Schleife `For Each ... Next` wiederholt eine Gruppe von Anweisungen für jedes Element in einem Datenfeld oder einer Auflistung.

Die Syntax dieser Schleife lautet:

```

For Each Element In Gruppe
    Anweisungen
Exit For
Next Element

```

Das Argument `Element` stellt die Variable zum Durchlauf durch die Elemente der Auflistung oder des Datenfeldes dar. Bei Auflistungen sind für `Element` nur eine Variable vom Typ `Variant`, eine allgemeine Objektvariable oder eine beliebige spezielle Objektvariable zulässig. Bei Datenfeldern ist für `Element` nur eine Variable vom Typ `Variant` zulässig. Das nächste Argument, `Gruppe`, steht für den Namen einer Objektauflistung oder eines Datenfeldes. Das letzte Argument, `Anweisungen`, ist optional und führt eine oder mehrere Anweisungen durch, die für jedes Element in der Gruppe ausgeführt werden sollen.

### Alle leeren Zellen einer Markierung ansprechen

Beim folgenden Beispiel sollen alle Zellen, die vorher markiert wurden, durchsucht und leere Zellen mit der Hintergrundfarbe Gelb formatiert werden. Dazu können Sie das Makro aus Listing 3.22 einsetzen.

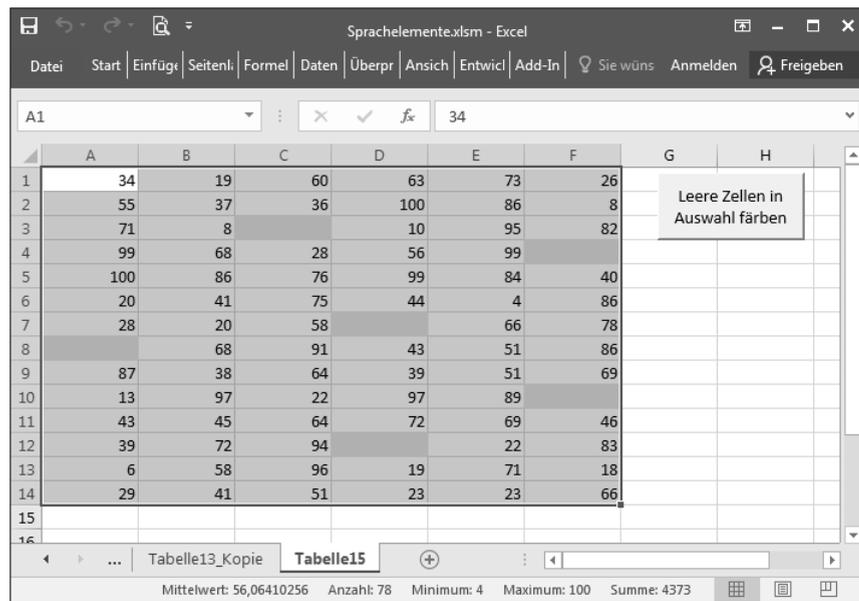
```
Sub AlleLeerenZellenInAuswahlIdentifizieren()
    Dim rngZelle As Range

    If TypeName(Selection) <> "Range" Then Exit Sub

    For Each rngZelle In Selection.SpecialCells(xlCellTypeBlanks)
        rngZelle.Interior.ColorIndex = 6
    Next rngZelle

End Sub
```

**Listing 3.22** Leere Zellen in einer Auswahl von Zellen einfärben



**Abbildung 3.20** Alle leeren Zellen in der Auswahl wurden eingefärbt.

Mit Hilfe der Funktion `TypeName` stellen Sie sicher, dass überhaupt eine Zellenmarkierung vorliegt. In diesem Fall liefert die Funktion als Rückgabe den Text `Range`. Diese Sicherheitsmaßnahme ist dann wichtig, wenn Sie beispielsweise eingebettete Diagrammobjekte in der Tabelle haben und aus Versehen anstatt der Zellen eben ein Diagramm markieren. In einem solchen Fall würde das Makro abstürzen.

In einer anschließenden Schleife vom Typ `For Each ... Next` werden alle Zellen der Markierung, wenn Sie denn leer sind, verarbeitet. Um zu ermitteln, ob die Zellen leer sind, können Sie die Methode `SpecialCells` mit der Konstanten `xlCellTypeBlanks` verwenden. Innerhalb der Schleife färben Sie den Innenraum der jeweiligen leeren Zelle, indem Sie dem Objekt `Interior` über die Eigenschaft `ColorIndex` die Farbnnummer 6 (= Gelb) zuweisen.

### Alle Zellen mit dem gleichen Inhalt markieren

Ganz nützlich ist auch das Makro aus Listing 3.23. Dabei wird der Inhalt der aktiven Zelle genommen und kontrolliert, ob sich im benutzten Bereich der Tabelle weitere Zellen mit dem gleichen Inhalt befinden. Wenn ja, dann werden diese dynamisch markiert.

```
Sub AlleZellenMitWertMarkieren()
    Dim rngZelle As Range
    Dim rngBereich As Range
    Dim lngZ As Long

    For Each rngZelle In Tabelle15.UsedRange
        If rngZelle.Value = ActiveCell.Value Then
            If lngZ = 0 Then
                Set rngBereich = rngZelle
                lngZ = 1
            Else
                Set rngBereich = Union(rngBereich, rngZelle)
            End If
        End If
    Next rngZelle

    rngBereich.Select

End Sub
```

**Listing 3.23** Ausgehend von der aktiven Zelle weitere Zellen mit gleichem Inhalt markieren

In einer Schleife der Form `For Each ... Next` werden alle Zellen im benutzten Bereich von `TABELLE15` abgearbeitet. Innerhalb der Schleife findet zunächst ein Vergleich der aktiven Zelle mit der jeweils durch die Schleife angesprochenen Zelle statt. Tritt eine Übereinstimmung das erste Mal auf, dann wird der Objektvariablen `rngBereich` die gerade verarbeitete Zelle über die Anweisung `Set` zugewiesen. Danach wird der Zähler `lngZ` auf den Wert 1 gesetzt. Bei weiteren Übereinstimmungen wird jeweils die dazukommende Zelle dem Bereich `rngBereich` über die Methode `Union` hinzugefügt. Am Ende des Makros wird die Methode `Select` verwendet, um die vorher ermittelten Zellen zu markieren.

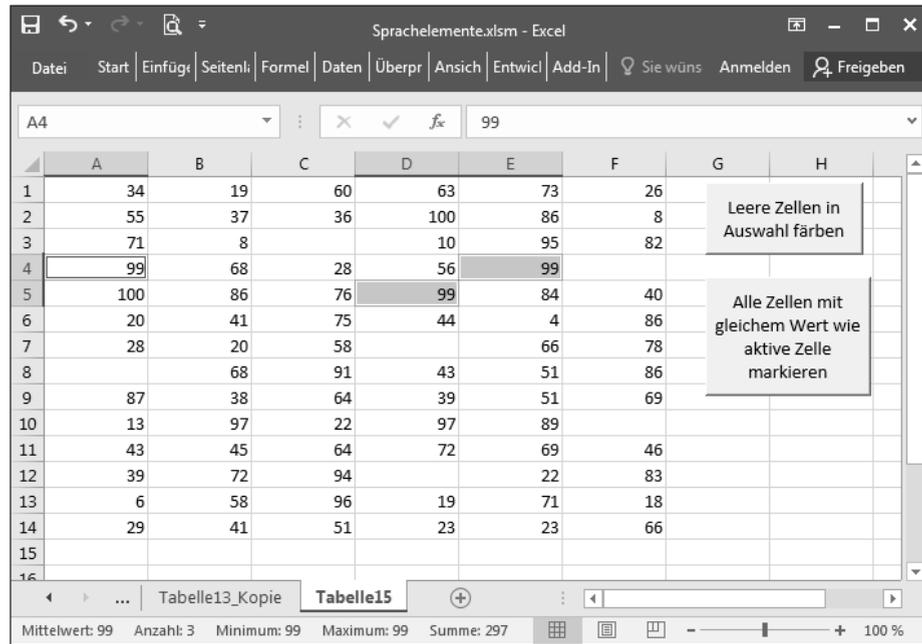


Abbildung 3.21 Alle Zellen mit dem Inhalt der aktiven Zelle werden markiert.

### Die bedingte Formatierung per Makro einstellen

Bei der folgenden Aufgabenstellung sollen in TABELLE16 zwei Spalten miteinander verglichen werden. Dabei soll die bedingte Formatierung von Excel in Form von Symbolsätzen zum Einsatz kommen. Sehen Sie sich zunächst die Ausgangssituation aus Abbildung 3.22 an. Stellen Sie die bedingte Formatierung für Spalte B über das Makro aus Listing 3.24 ein:

```
Sub BedingteFormatierungPerMakroEinfügen()
    Dim rngZelle As Range
    Dim lngZeileMax As Long

    With Tabelle16
        lngZeileMax = .Cells(.Rows.Count, 1).End(xlUp).Row

        With .Range("B1:B" & lngZeileMax)

            .FormatConditions.Delete

            For Each rngZelle In .Cells

                rngZelle.FormatConditions.AddIconSetCondition
                With rngZelle.FormatConditions(1)
```

```
.IconSet = ActiveWorkbook.IconSets(xl3Arrows)
With .IconCriteria(2)
    .Type = xlConditionValueFormula
    .Value = "=" & rngZelle(1, 0).Address
    .Operator = xlGreaterEqual
End With
With .IconCriteria(3)
    .Type = xlConditionValueFormula
    .Value = "=" & rngZelle(1, 0).Address
    .Operator = xlGreater
End With
End With

Next rngZelle

End With

End With
```

End Sub

Listing 3.24 Für den schnellen Überblick einen Symbolsatz per Makro einfügen

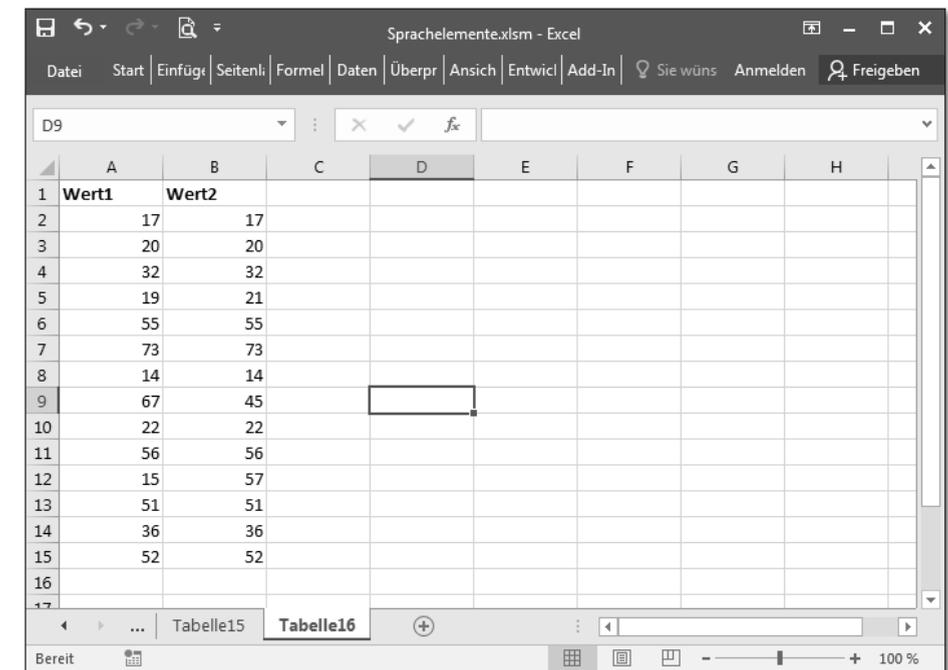


Abbildung 3.22 Wo sind die Unterschiede? Welcher Wert ist kleiner oder größer?

Deklarieren Sie zu Beginn des Makros aus Listing 3.24 die Objektvariable `rngZelle` als Range und die Variable `lngZeileMax` mit dem Datentyp `Long`. Danach ermitteln Sie, wie viele Zeilen in Spalte B gefüllt sind. Setzen Sie dazu die Eigenschaft `End` ein, über die Sie mittels der Konstanten `xlUp` die Richtung festlegen. Sie schauen dabei aus der letzten Zeile von Spalte A nach oben und fragen von der letzten gefüllten Zelle in Spalte A die Zeilennummer über die Eigenschaft `Row` ab.

Geben Sie jetzt über die Anweisung `With` den Bereich derjenigen Zellen in Spalte B an, die Sie mit einer bedingten Formatierung ausstatten möchten. Danach löschen Sie alle eventuell bereits eingestellten bedingten Formate, indem Sie das Auflistungsobjekt `FormatConditions` komplett über die Methode `Delete` entfernen. Über die Methode `AddIconSetCondition` legen Sie fest, dass Sie mit einem Symbolsatz arbeiten möchten.

	A	B	C	D	E	F	G
1	Wert1	Wert2					
2		17 →					
3		20 →					
4		32 →					
5		19 ↑					
6		55 →					
7		73 →					
8		14 →					
9		67 ↓					
10		22 →					
11		56 →					
12		15 ↑					
13		51 →					
14		36 →					
15		52 →					
16							

Abbildung 3.23 Veränderungen aller Art werden über Symbole veranschaulicht.

Mit einem Symbolsatz können Sie Daten kennzeichnen und in drei bis fünf Kategorien einteilen, die durch einen Schwellenwert getrennt werden. Jedes Symbol stellt einen Wertebereich dar. Im Symbolsatz »3 Pfeile« (`xl3arrows`) stellt der rote Pfeil nach unten beispielsweise niedrigere Werte dar, der gelbe Pfeil zur Seite gleiche Werte und der grüne Pfeil nach oben größere Werte. Die Einteilung findet über die Eigenschaft `IconCriteria` statt. In der Eigenschaft `Type` geben Sie bekannt, dass Sie zum Vergleichen der Spalte auf eine Formel zurückgreifen möchten. Diese Formel geben Sie danach über die Eigenschaft `Value` bekannt. Mit Hilfe der Eigenschaft `Operator` defi-

nieren Sie, wie Excel beim Vergleich vorgehen soll. Dabei werden in diesem Beispiel die Konstanten `xlGreaterEqaul` und `xlGreater` verwendet.

### Einen Bereich in einer Tabelle rahmen

Bei der folgenden Aufgabenstellung soll in TABELLE17 ein bestimmter Bereich eingrahmt werden. Zusätzlich soll ein dickerer Rahmen um den kompletten Bereich gezogen werden. Sehen Sie sich die Umsetzung in Listing 3.25 an.

```
Sub BereichRahmen()
    Dim rngZelle As Range
    Dim rngBereich As Range

    Set rngBereich = Tabelle17.Range("B2:D10")

    For Each rngZelle In rngBereich

        With rngZelle.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With rngZelle.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With rngZelle.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With rngZelle.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With

    Next rngZelle

    rngBereich.BorderAround Weight:=xlThick, ColorIndex:=xlAutomatic

End Sub
```

Listing 3.25 Einen bestimmten Bereich in einer Tabelle mit einem Rahmen versehen

Deklarieren Sie im ersten Schritt des Makros aus Listing 3.25 zwei Objektvariablen vom Typ `Range`. Über die Anweisung `Set` geben Sie an, in welcher Tabelle sich der zu verarbeitende Bereich befindet, und gleichzeitig legen Sie fest, wo er genau liegt.

In einer Schleife des Typs `For Each ... Next` wird Zelle für Zelle abgearbeitet. Innerhalb der Schleife wird auf das Objekt `Borders` zugegriffen. Dabei wird über die Konstanten `xlEdgeLeft`, `xlEdgeTop`, `xlEdgeBottom` und `xlEdgeRight` festgelegt, welcher Rahmen angesprochen wird.

Mit Hilfe der Eigenschaft `LineStyle` legen Sie die Art der Linie fest. Über die Konstante `xlContinuous` definieren Sie eine normale durchgezogene Linie.

Über die Eigenschaft `Weight` legen Sie die Dicke der Linie fest. Zur Verfügung stehen dabei die selbstsprechenden Konstanten `xlHairline`, `xlThick`, `xlThin` und `xlMedium` zur Verfügung.

Mit Hilfe der Eigenschaft `ColorIndex` bestimmen Sie die Farbe des Rahmens. Die Konstante `xlAutomatic` färbt dabei die Linie schwarz (= Standard).

Mit Hilfe der Methode `BorderAround` legen Sie einen Gesamtrahmen um den Bereich. Dabei können Sie im Parameter `Weight` die Dicke der Linie festlegen. Hier stehen dieselben Konstanten wie bei der gleichnamigen Eigenschaft zur Verfügung. Über den Parameter `ColorIndex` legen Sie die Farbe des Gesamtrahmens fest.

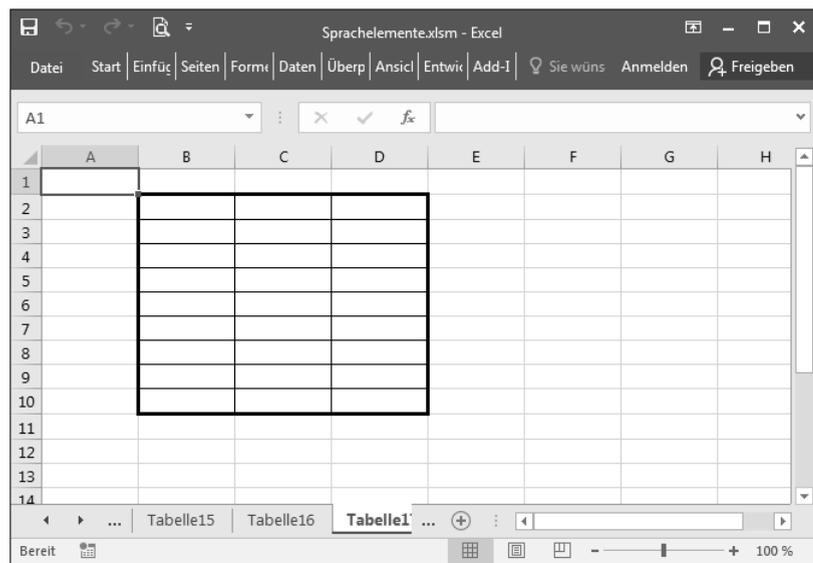


Abbildung 3.24 Der Bereich B2:D10 wurde eingerahmt.

### Alle Tabellen einer Arbeitsmappe als separate Mappen abspeichern

Bei der folgenden Aufgabenstellung sollen alle Tabellen der Arbeitsmappe kopiert und als eigenständige Dateien gespeichert werden. Diese Arbeit möchten Sie sicher

nicht händisch machen, oder? Zur Umsetzung dieser Aufgabenstellung können Sie das Makro aus Listing 3.26 einsetzen.

```
Sub AlleTabellenExportieren()
    Dim wksBlatt As Worksheet

    Application.DisplayAlerts = False

    For Each wksBlatt In ThisWorkbook.Worksheets

        Application.StatusBar = "Tabelle " & wksBlatt.Name & " wird exportiert"
        wksBlatt.Copy
        ActiveWorkbook.SaveAs Filename:="C:\Windows\Temp\" & _
                               wksBlatt.Name & ".xlsx"

        ActiveWorkbook.Close

    Next wksBlatt

    Application.DisplayAlerts = True
    Application.StatusBar = False

End Sub
```

Listing 3.26 Alle Tabellen als separate Mappen speichern

Deklarieren Sie im ersten Schritt des Makros aus Listing 3.26 eine Objektvariable vom Typ `Worksheet` mit dem Namen `wksBlatt`. Schalten Sie über die Eigenschaft `DisplayAlerts` Excel-Warnmeldungen ab, indem Sie dieser Eigenschaft den Wert `False` zuweisen.

In einer Schleife der Art `For Each ... Next` arbeiten Sie alle Tabellen der Arbeitsmappe nacheinander ab. Innerhalb der Schleife dokumentieren Sie den Exportvorgang, indem Sie die Statusleiste von Excel beschreiben. Dazu weisen Sie der Eigenschaft `StatusBar` einen begleitenden Text zu.

Mit Hilfe der Methode `Copy` kopieren Sie die einzelnen Tabellen aus der Arbeitsmappe. Dadurch werden diese Tabellen jeweils zur aktiven Arbeitsmappe, die Sie über die Methode `SaveAs` im Ordner `Temp` von `Windows` speichern können. Schließen Sie danach die aktive Arbeitsmappe über die Methode `Close`.

Vergessen Sie nicht, der Eigenschaft `DisplayAlerts` den Wert `True` zuzuweisen, um zukünftig Warnmeldungen von Excel wieder anzuzeigen. Geben Sie die Steuerung der Statusleiste wieder an Excel zurück, indem Sie die Eigenschaft `StatusBar` auf den Wert `False` setzen.

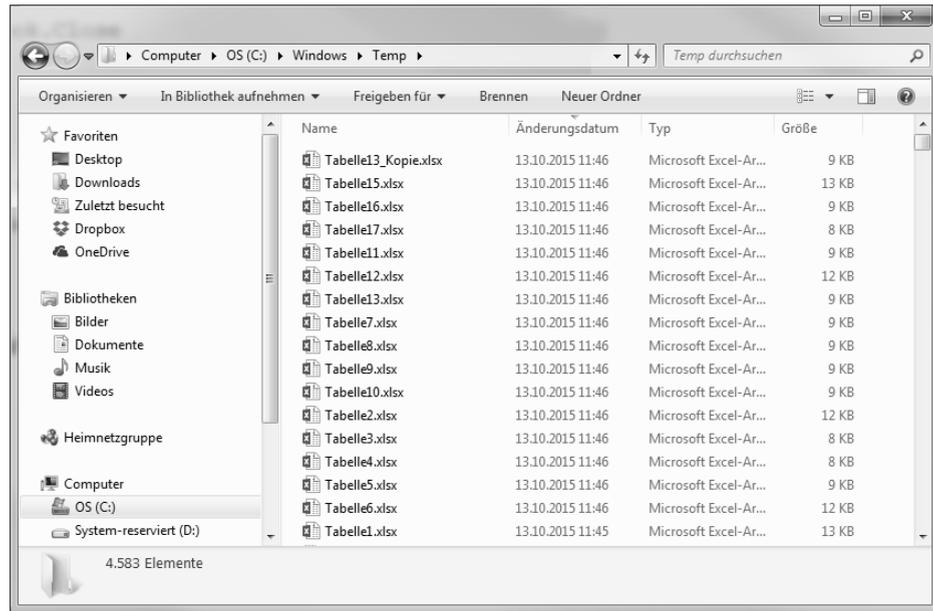


Abbildung 3.25 Die neuen Mappen liegen im Zielverzeichnis vor.

### Alle Kommentare einer Tabelle nachformatieren

Bei der folgenden Aufgabenstellung liegen in TABELLE18 einige Zellenkommentare vor, die umformatiert werden sollen.

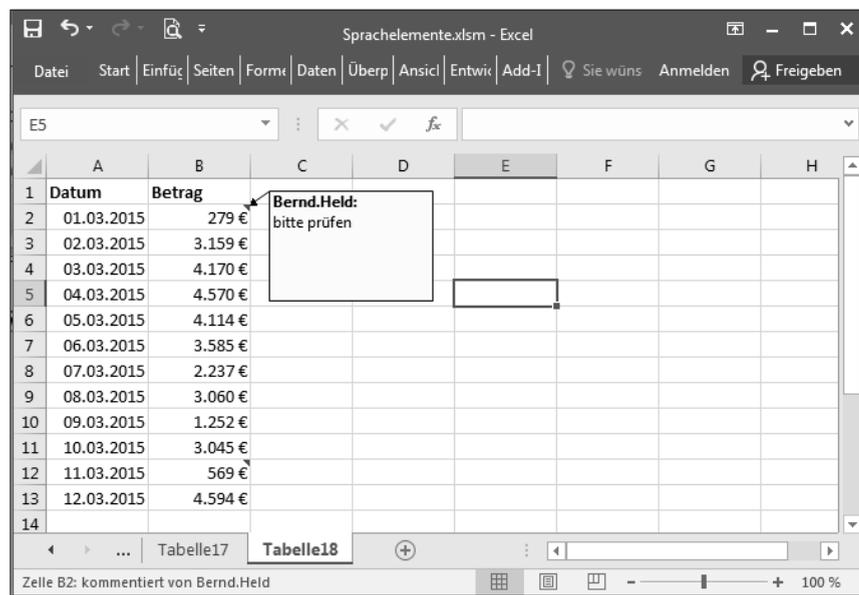


Abbildung 3.26 Die Kommentare der Tabelle sollen anders formatiert werden.

Starten Sie das Makro aus Listing 3.27, um alle Kommentare in TABELLE18 umzuformatieren.

```
Sub KommentareFormatieren()
    Dim cmtNotiz As Comment

    For Each cmtNotiz In Tabelle18.Comments
        With cmtNotiz.Shape.TextFrame.Characters.Font
            .Name = "Arial"
            .Size = 14
            .Italic = True
            .Underline = True
            .Bold = True
        End With
    Next cmtNotiz
End Sub
```

End Sub

Listing 3.27 Alle Notizen in einer Tabelle umformatieren

Deklariieren Sie zu Beginn des Makros aus Listing 3.27 eine Objektvariable vom Typ Comment mit dem Namen cmtNotiz. In einer anschließenden Schleife des Typs For Each ... Next durchlaufen Sie alle Kommentare aus TABELLE18, die automatisch im Auflistungsobjekt Comments verzeichnet sind.

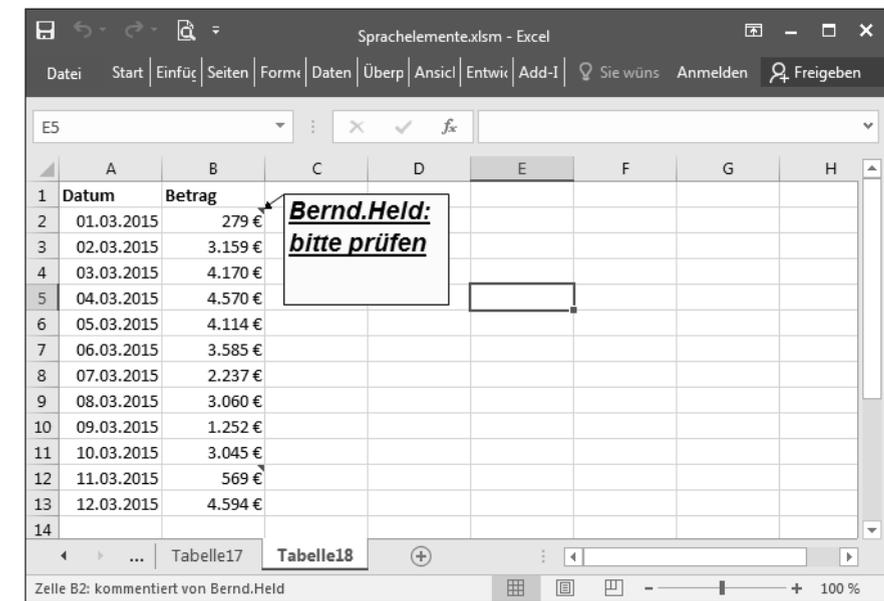


Abbildung 3.27 Die Kommentare sind nun besser lesbar.

Innerhalb der Schleife greifen Sie auf den Kommentar zu, indem Sie über das Shape-Objekt gehen, da der Kommentar ja quasi in einem Rechteck liegt. Innerhalb dieser Form gibt es ein Textfeld, das Sie über die Eigenschaft `TextFrame` ansteuern können. Innerhalb dieses Textfelds können Sie über die Auflistung `Characters` auf alle erfassten Zeichen zugreifen.

Über das Objekt `Font` können Sie danach ganz gezielt die Schriftformatierung des Kommentars beeinflussen. Dazu stehen Ihnen einige Eigenschaften zur Verfügung. Über die Eigenschaft `Name` stellen Sie die gewünschte Schriftart ein. Die Eigenschaft `Size` legt fest, wie groß die Schrift ist. Über die Eigenschaft `Italic` formatieren Sie den Kommentartext kursiv. Mit Hilfe der Eigenschaft `Underline` unterstreichen lassen Sie den Kommentartext automatisch unterstreichen. Über die Eigenschaft `Bold` definieren Sie den Fettdruck.

### 3.6.3 Die Schleife »Do Until ... Loop«

Die Schleife des Typs `Do Until ... Loop` wiederholt einen Block mit Anweisungen, solange eine Bedingung den Wert `True` erhält. Die Bedingung wird jeweils am Ende der Schleife geprüft. Als Abbruchbedingung lassen sich unterschiedliche Bedingungen abfragen.

So können Sie z. B. eine Abbruchbedingung festlegen, wenn ein bestimmter Wert erreicht ist oder eine Zelle einen bestimmten Text aufweist. Beispielsweise könnten Sie eine solche Schleife so oft wiederholen, wie sich die Zellenformatierung der Zellen nicht ändert.

Die Syntax dieser Schleife sieht wie folgt aus:

```
Do Until Bedingung
    Anweisungen
Exit Do
Loop
```

Die Bedingung stellt einen numerischen Ausdruck oder Zeichenfolgenausdruck dar, der entweder erfüllt (`True`) oder nicht erfüllt (`False`) ist. Liefert die Bedingung den Wert `0`, so wird die Bedingung als `False` interpretiert. Hinter den Anweisungen verbergen sich eine oder mehrere Anweisungen, die wiederholt werden, solange oder bis Bedingung durch `True` erfüllt ist.

#### Eine CSV-Datei mit Umsätzen einlesen

Beim folgenden Beispiel liegt eine CSV-Datei wie in Abbildung 3.28 gezeigt vor. Diese Textdatei soll in TABELLE19 eingelesen werden.

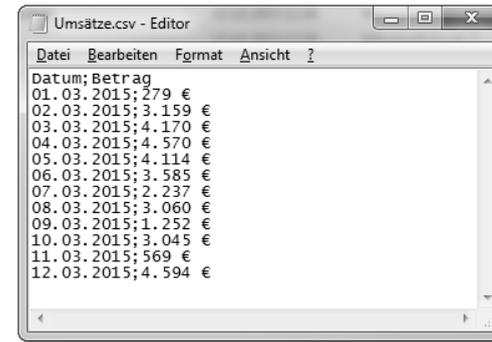


Abbildung 3.28 Die Daten aus dieser CSV-Datei sollen importiert werden.

Um die CSV-Datei in TABELLE19 zu übertragen, starten Sie das Makro aus Listing 3.28.

```
Sub TextdateiEinlesen()
    Dim objFSO As Object
    Dim ts As Object
    Dim strSatz As String
    Dim intZ As Integer

    With Tabelle19
        .Rows.Delete
        Set FSO = CreateObject("Scripting.FileSystemObject")

        'hier gegebenenfalls den Pfad anpassen
        Set ts = FSO.OpenTextFile(ThisWorkbook.Path & "\Umsätze.csv")

        intZ = 1

        Do Until ts.AtEndOfStream
            strSatz = ts.ReadLine
            Cells(intZ, 1).Value = strSatz
            intZ = intZ + 1
        Loop

        ts.Close

        .Columns("A:A").TextToColumns Destination:=.Range("A1"), _
            DataType:=xlDelimited, semicolon:=True
    End With

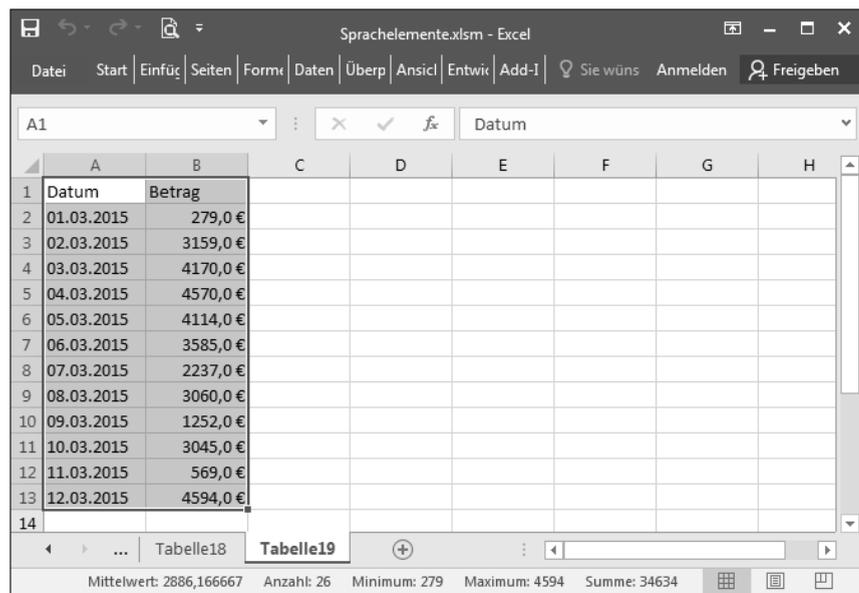
End Sub
```

Listing 3.28 Eine CSV-Datei in einer Excel-Tabelle einlesen (Zeile für Zeile)

Für das Importieren von Textdaten können Sie auf eine Windows-eigene Bibliothek zurückgreifen, die alle Methoden und Eigenschaften anbietet, die Sie für die Bearbeitung von Textdateien benötigen. Diese Bibliothek mit dem Namen `SCRIPTING RUNTIME` binden Sie nach dem Prinzip des *Late Bindings* ein. Das bedeutet, dass Sie erst während des Makrolaufs einen Verweis auf diese Bibliothek mittels der Funktion `CreateObject` vornehmen. Danach haben Sie Zugriff auf alle Befehle, die in dieser Bibliothek enthalten sind. Unter anderem ist das die Methode `OpenTextFile`, über die Sie eine Textdatei öffnen.

Danach setzen Sie eine `Do Until`-Schleife auf, die Zeile für Zeile aus der Textdatei liest und in `TABELLE19` schaufelt. Innerhalb der Schleife wenden Sie die Methode `ReadLine` an, um eben zeilenweise die Textdatei auszulesen. Dabei speichern Sie Satz für Satz temporär in der `String`-Variablen `strSatz`. Den Inhalt der Variablen übertragen Sie dann zunächst in die erste Spalte der Tabelle. Die Auftrennung des Strings nehmen Sie später vor.

Nach dem Schleifenaustritt wenden Sie die Methode `Close` an, um die Textdatei zu schließen. Danach kommt die Methode `TextToColumns` zum Einsatz, die auf Basis des Trennzeichens (in diesem Fall ist das ein Semikolon) die Daten trennt.



	A	B	C	D	E	F	G	H
1	Datum	Betrag						
2	01.03.2015	279,0 €						
3	02.03.2015	3159,0 €						
4	03.03.2015	4170,0 €						
5	04.03.2015	4570,0 €						
6	05.03.2015	4114,0 €						
7	06.03.2015	3585,0 €						
8	07.03.2015	2237,0 €						
9	08.03.2015	3060,0 €						
10	09.03.2015	1252,0 €						
11	10.03.2015	3045,0 €						
12	11.03.2015	569,0 €						
13	12.03.2015	4594,0 €						
14								

Abbildung 3.29 Alle Datensätze aus der CSV-Datei wurden importiert.

### Arbeitsmappe nach Untätigkeit automatisch schließen

Für viele ein großes Ärgernis: Der Kollege »Dauerspaziergänger« arbeitet an einer Datei und geht zwischendurch einen Kaffee trinken. Er hat die Datei aber leider nicht geschlossen, und Sie kommen nicht an sie heran. Abhilfe schafft ein kleines Skript,

das Sie beispielsweise beim Öffnen der Arbeitsmappe über das Ereignis `Workbook_Open` starten.

Schauen Sie sich die Umsetzung des Vorhabens in Listing 3.29 an. Wir geben dem Kollegen mal 120 Sekunden, bis die Datei automatisch schließt und dabei gespeichert wird.

```
Sub Zeitschaltuhr()
    Dim dteBeginn As Date
    Dim dtePause As Date

    dteBeginn = Timer
    dtePause = 120

    Do Until Timer > dteBeginn + dtePause
        DoEvents
    Loop

    ThisWorkbook.Close savechanges:=True

End Sub
```

Listing 3.29 Eine Mappe nach zwei Minuten Untätigkeit einfach mal schließen

Deklarieren Sie zu Beginn des Makros aus Listing 3.29 zwei Datumsvariablen, um die Zeit messen zu können. Über die Funktion `Timer` holen Sie sich sekundenscharf die aktuelle Uhrzeit und speichern diese in der Variablen `dteBeginn`. Legen Sie in der Variablen `dtePause` die Anzahl der Sekunden fest, die Sie dem Kollegen auf Wanderschaft gewähren möchten.

In einer `Do Until`-Schleife wird jetzt bereits im Kopf der Schleife geprüft, ob die Zeit abgelaufen ist. In der Schleife selbst wird die Funktion `DoEvents` eingesetzt. Diese Funktion sorgt dafür, dass die Kontrolle wieder zurück an das Betriebssystem gegeben wird.

Bei Schleifenaustritt, also wenn die Zeit abgelaufen ist, wenden Sie die Methode `Close` an, um die Arbeitsmappe zu schließen. Über den Parameter `SaveChanges` können Sie dabei selbst entscheiden, ob Sie die Mappe noch einmal speichern möchten oder nicht.

### 3.6.4 Die Schleife »Do While ... Loop«

Die Schleife des Typs `Do While ... Loop` wiederholt einen Block mit Anweisungen, solange eine Bedingung den Wert `True` enthält. Die Prüfung der angegebenen Bedingung erfolgt immer zu Beginn der Schleife. Als Abbruchbedingung können Sie z. B.

eine Abbruchbedingung festlegen, wenn ein bestimmter Wert erreicht ist oder eine Zelle einen bestimmten Text aufweist.

Die Syntax dieser Schleife sieht wie folgt aus:

```
Do While Bedingung
    Anweisungen
Exit Do
Loop
```

Die Bedingung stellt einen numerischen Ausdruck oder Zeichenfolgenausdruck dar, der entweder erfüllt (True) oder nicht erfüllt (False) wird. Liefert die Bedingung den Wert 0, so wird die Bedingung als False interpretiert. Hinter den Anweisungen verbergen sich eine oder mehrere Anweisungen, die wiederholt werden, solange oder bis die Bedingung True erfüllt ist. Innerhalb einer Schleife der Art Do While ... Loop können Sie eine beliebige Anzahl von Exit Do-Anweisungen an beliebiger Stelle als Alternative zum Verlassen einer Do ... Loop-Anweisung verwenden.

#### Anzahl von Dateien eines Verzeichnisses ermitteln

Beim Beispiel aus Listing 3.30 sollen alle Dateien aus einem bestimmten Verzeichnis gezählt werden. Dazu setzen wir die Funktion Dir ein, die DOS-Anwender aus »alten Tagen« sicherlich noch kennen werden.

```
Sub DateienZaehlen()
    Dim strOrdnerName As String
    Dim strName As String
    Dim intz As Integer

    strOrdnerName = "C:\Windows\Temp\"
    strName = Dir(strOrdnerName & "*.*.")

    Do While strName <> ""
        strName = Dir
        intz = intz + 1
    Loop

    MsgBox "Anzahl der Dateien: " & intz

End Sub
```

#### Listing 3.30 Alle Dateien in einem Verzeichnis zählen

Geben Sie zu Beginn des Makros den Namen des Ordners an, in dem die darin enthaltenen Dateien gezählt werden sollen. Übergeben Sie diesen Ordner sowie den »Dateifilter« an die Funktion Dir.

Nun kommt die Do While-Schleife zum Einsatz. Innerhalb der Schleife wird wiederum die Funktion Dir angewendet, um die nächste Datei zu ermitteln. Erhöhen Sie dann jeweils den Zähler intz um den Wert 1. Die Ende-Bedingung für die Schleife wird dann erreicht, wenn keine weitere Datei mehr gefunden wird. In diesem Fall wird eine leere Zeichenfolge zurückgegeben.

#### Alle Dateien aus einem Verzeichnis auslesen

Im Beispiel aus Listing 3.31 werden die Namen aller Dateien aus einem bestimmten Verzeichnis in TABELLE20 übertragen.

```
Sub DateienAusOrdnerInTabelleAusgeben()
    Dim strPfad As String
    Dim strDatei As String
    Dim lngZeile As Long

    With Tabelle20
        .Rows.Delete
        lngZeile = 1
        strPfad = ThisWorkbook.Path

        strDatei = Dir(strPfad & "\*.xls*")

        Do While strDatei <> ""
            .Cells(lngZeile, 1).Value = strPfad & "\" & strDatei
            lngZeile = lngZeile + 1
            strDatei = Dir
        Loop

        .Columns(1).AutoFit

    End With

End Sub
```

#### Listing 3.31 Alle Dateien aus einem Verzeichnis auslesen

Deklarieren Sie zu Beginn des Makros aus Listing 3.31 zwei String-Variablen. In der Variablen strPfad wird der Ordner angegeben, der durchsucht werden soll. Über die Variable strDatei wird später der Dateifilter gebildet.

Ermitteln Sie den Pfad des auszulesenden Ordners, indem Sie über die Eigenschaft Path den Pfad der Mappe ableiten, in der die Makros gespeichert sind. Das bedeutet, dass Excel genau in diesem Pfad nachsehen soll, in dem auch die Datei *Sprachelemente.xlsm* steht.

Über die Funktion `Dir` lesen Sie zunächst einmal die erste Datei im Verzeichnis aus. Anschließend setzen Sie die `Do While`-Schleife auf, die so lange über die Funktion `Dir` nach weiteren Dateien sucht, bis diese Funktion eine leere Zeichenfolge zurückliefert. Das ist in etwa so, als wenn Sie im Gebirge jodeln, und es kommt kein Echo zurück. Wenn also die Funktion `Dir` eine leere Zeichenfolge zurückliefert, dann werden keine weiteren Dateien mehr gefunden. Demnach erfolgt ein Schleifenaustritt. Passen Sie am Ende die Breite von Spalte A automatisch über die Methode `AutoFit` an.

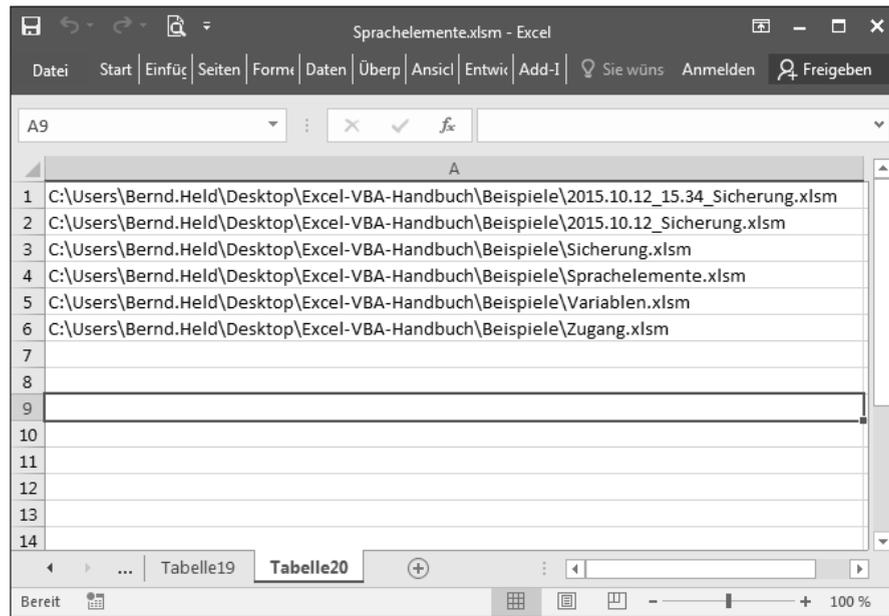


Abbildung 3.30 Eine Auflistung aller Dateien aus einem bestimmten Verzeichnis

## 3.7 Sonstige Sprachelemente

Neben Bedingungen und Schleifen gibt es noch ein anderes gängiges Sprachelement, das wir bereits hin und wieder in vorangegangenen Beispielen eingesetzt haben.

### 3.7.1 Die Anweisung »With«

Die Anweisung `With` wird eingesetzt, um sich viel Schreibarbeit zu sparen und mehr Übersichtlichkeit in den Quellcode zu bringen. Dabei wird das Objekt, auf das mehrere Befehle angewendet werden sollen, einmalig definiert, ohne dieses Objekt für jeden Befehl zu wiederholen.

Die Syntax dieser Anweisung lautet:

```
With Objekt
    Anweisungen
End With
```

Im Argument `Objekt` geben Sie den Namen des Objekts an, das Sie ansprechen möchten. Im Argument `Anweisungen` erfassen Sie ein oder mehrere Anweisungen, die für das Objekt ausgeführt werden sollen.

### Alle verwendeten Zellen einer Spalte formatieren

Beim der folgenden Aufgabenstellung sollen alle verwendeten Zellen von Spalte A in TABELLE21 formatiert werden, und zwar neben dem Zahlenformat auch die Schriftfarbe, den Schriftschnitt sowie die Hintergrundfarbe. Damit Sie so wenig Schreibarbeit wie möglich haben, setzen Sie die `With`-Anweisung wie in Listing 3.32 gezeigt ein:

```
Sub BestimmteZelleAnprechenUndFormatieren()
    Dim lngZeileMax As Long

    lngZeileMax = Tabelle21.UsedRange.Rows.Count

    With Tabelle21.Range("A2:A" & lngZeileMax)

        'Datumsformat
        .NumberFormat = "DD.MM.YYYY"

        'Hintergrundfarbe festlegen (Grün)
        .Interior.ColorIndex = 4

        'Schriftschnitt Fett
        .Font.Bold = True

        'Schriftfarbe bestimmen (Weiss)
        .Font.ColorIndex = 2

    End With
End Sub
```

Listing 3.32 Alle verwendeten Zellen von Spalte A formatieren

Ermitteln Sie zu Beginn des Makros aus Listing 3.32, wie viele Zeilen in Spalte A belegt sind. Speichern Sie diese Information in der Variablen `lngZeileMax`. Danach geben Sie an, mit welcher Tabelle und mit welchem Bereich Sie arbeiten möchten. Setzen Sie dazu die Anweisung `With` ein. Immer wenn Sie danach auf diese »Zusammenfassung« zugreifen möchten, genügt es, wenn Sie als erstes Zeichen der Zeile einen Punkt erfassen. Excel weiß dann, welches Objekt Sie meinen.

Über die Eigenschaft `NumberFormat` können Sie die Formatierung des Datums festlegen. Die Formatierung erfolgt über Buchstabenkürzel (`DD` = zweistellige Tagesangabe, `MM` = zweistellige Monatsangabe, `YYYY` = vierstellige Jahresangabe).

Über die Eigenschaft `ColorIndex`, die Sie auf das Objekt `Interior` und das Objekt `Font` anwenden, färben Sie die verwendeten Zellen ein.

Über das Objekt `Font` haben Sie Zugriff auf alle Eigenschaften, die für die Formatierung der Schrift verantwortlich sind. Über die Eigenschaft `Bold` formatieren Sie einen Fettdruck.

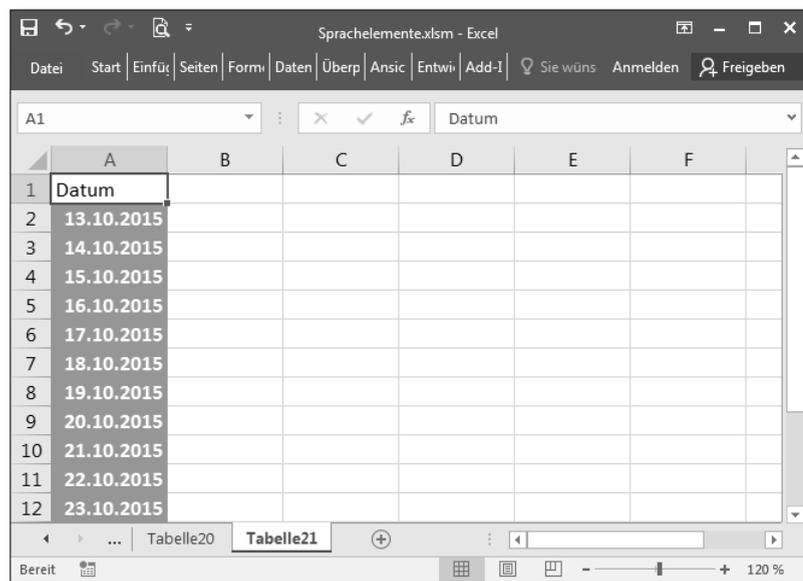


Abbildung 3.31 Das Datumsformat sowie die restlichen Formatierungen wurden eingestellt.

### Eine eigene Gültigkeitsliste erstellen

Bei der folgenden Aufgabenstellung soll in TABELLE22 im Bereich A1:A10 eine Gültigkeitsliste in Form eines Zellen-Dropdowns automatisch angelegt werden. Dazu können Sie das Makro aus Listing 3.33 einsetzen:

```
Sub GültigkeitslisteEinfügen()  
    Dim strAuswahl As String
```

```
strAuswahl = "Deutschland, Spanien, Frankreich, Italien"
```

```
With Tabelle22.Range("A1:A10").Validation  
    .Delete  
    .Add Type:=xlValidateList, _  
        AlertStyle:=xlValidAlertStop, Formula1:=strAuswahl  
    .InCellDropdown = True  
    .InputTitle = "Land auswählen"  
    .InputMessage = "Bitte das gewünschte Land auswählen"  
    .ErrorTitle = "Fehler"  
    .ErrorMessage = "Dieses Land ist nicht vorgesehen"  
    .ShowInput = True  
    .ShowError = True  
End With
```

```
End Sub
```

Listing 3.33 Eine Gültigkeitsliste als Zellen-Dropdown automatisch anlegen

Deklarieren Sie zunächst zu Beginn des Makros aus Listing 3.33 eine String-Variable mit dem Namen `strAuswahl`. Danach füllen Sie sie, indem Sie die gewünschten Länder mit Komma getrennt in diese Variable packen.

Danach sparen Sie sich eine Menge Schreiarbeit, indem Sie das Objekt `Validation` für den Zellenbereich A1:A10 für TABELLE22 über die Anweisung `With` angeben.

Wenden Sie zunächst die Methode `Delete` an, um gegebenenfalls eine bereits existierende Gültigkeitsregel in diesem Bereich zu entfernen.

Über die Methode `Add` fügen Sie eine neue Gültigkeitsregel ein. Dabei geben Sie über die Konstante `xlValidateList` an, dass Sie gerne eine Liste anlegen möchten. Im Parameter `AlertStyle` geben Sie über die Konstante `xlValidAlertStop` bekannt, dass Sie keine anderen Länder als die in der Liste befindlichen akzeptieren. Im Parameter `Formula1` verweisen Sie auf die vorher gefüllte String-Variable `strAuswahl`.

Um das Dropdown-Symbol in den Zellen anzuzeigen, weisen Sie der Eigenschaft `InCellDropdown` den Wert `True` zu.

Den Titel, der automatisch in einer QuickInfo angezeigt wird, wenn Sie den Cursor auf eine Zelle setzen, legen Sie über die Eigenschaft `InputTitle` fest. Die dazugehörige Meldung erfassen Sie mit Hilfe der Eigenschaft `InputMessage`.

Analog dazu können Sie den Fehlertitel und die Fehlermeldung über die Eigenschaften `ErrorTitle` und `ErrorMessage` festlegen.

Damit beide Meldungen überhaupt verfügbar sind, setzen Sie den Wert `True` für die Eigenschaften `ShowInput` und `ShowError`.

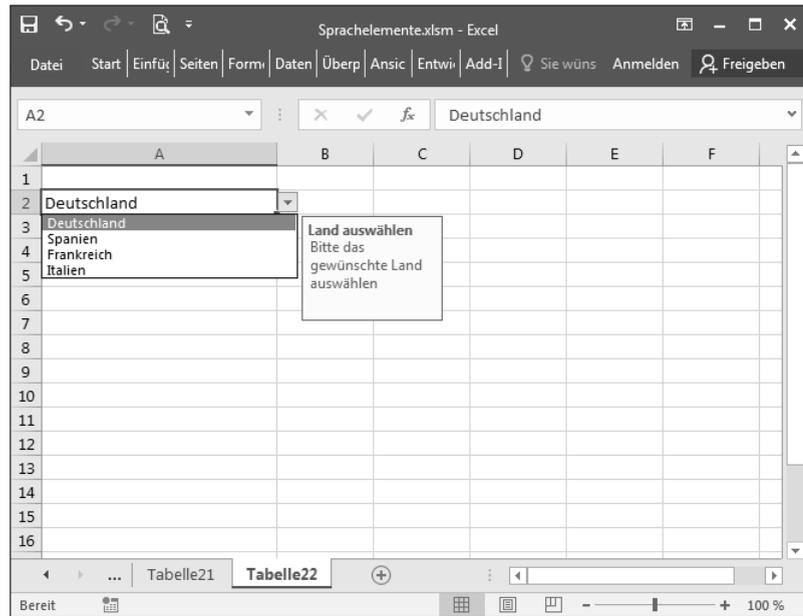


Abbildung 3.32 Die Gültigkeitsliste wurde erfolgreich eingefügt.

### Eine Ampelfunktion für Monatsumsätze erstellen

Im letzten Beispiel in diesem Kapitel integrieren wir eine Umsatz-Monatstabelle mit einer Ampelfunktion mit Hilfe der bedingten Formatierung von Excel. Sehen Sie sich dazu einmal Abbildung 3.33 an.

Abbildung 3.33 Die Ausgangssituation – eine Umsatzliste nach Monaten und Konten

Für die Ampel sollen folgende Regeln gelten:

- ▶ Werte > 67 % → grüne Ampel
- ▶ Werte 33–67 % → gelbe Ampel
- ▶ Werte < 33 % → rote Ampel

Setzen Sie diese Regeln im Makro aus Listing 3.34 um:

```
Sub AmpelfunktionBedingtesFormat()
    Dim rngBereich As Range

    Set rngBereich = Tabelle23.Range("B2:H11")
    rngBereich.FormatConditions.Delete
    rngBereich.FormatConditions.AddIconSetCondition
    rngBereich.FormatConditions(1).IconSet = _
        ActiveWorkbook.IconSets(xl3TrafficLights2)
    With rngBereich.FormatConditions(1).IconCriteria(2)
        .Type = xlConditionValuePercent
        .Value = 33
        .Operator = xlGreater
    End With
    With rngBereich.FormatConditions(1).IconCriteria(3)
        .Type = xlConditionValuePercent
        .Value = 67
        .Operator = xlGreater
    End With
End Sub
```

End Sub

### Listing 3.34 Eine Ampelfunktion für einen Bereich einfügen

Deklariert Sie im ersten Schritt eine Objektvariable mit dem Namen `rngBereich` vom Typ `Range`. Geben Sie im nächsten Schritt mit Hilfe der Anweisung `Set` bekannt, welche Tabelle und welchen Bereich darin Sie verarbeiten möchten.

Wenden Sie danach die Methode `Delete` an, um eventuell bereits existierende bedingte Formate in diesem Bereich zu entfernen. Nutzen Sie danach die Methode `AddIconSetCondition`, um mit einem Symbolsatz arbeiten zu können. Den Symbolsatz legen Sie über das Objekt `IconSet` fest. Über die Konstante `xl3TrafficLights2` geben Sie vor, dass Sie mit dreidimensionalen Ampelsymbolen arbeiten möchten. Danach stellen Sie die Formatierungskriterien über das Objekt `IconCriteria` fest. In der Eigenschaft `Type` legen Sie fest, dass Excel die prozentuale Berechnung anwenden soll. Über die Eigenschaft `Value` legen Sie diesen Prozentwert selber fest. In der Eigenschaft `Operator` geben Sie über die Konstante `xlGreater` an, dass die Formatierung dann gelten soll, wenn der in `Value` eingestellte Prozentsatz überschritten wird.

The screenshot shows an Excel spreadsheet with the following data:

	Januar	Februar	März	April	Mai	Juni	Juli
Konto 1	72	76	91	28	91	1	8
Konto 2	3	8	84	84	90	84	22
Konto 3	93	35	33	65	63	3	74
Konto 4	14	38	94	37	78	96	71
Konto 5	74	57	72	54	10	22	95
Konto 6	12	79	99	51	6	97	9
Konto 7	34	12	30	88	1	60	39
Konto 8	22	13	27	3	94	36	85
Konto 9	45	30	26	41	16	81	58
Konto 10	78	45	67	1	7	80	51

Abbildung 3.34 Die Ampeln wurden erfolgreich integriert.

## Kapitel 6

# Tabellen und Diagramme programmieren

Das Objekt »Worksheet« symbolisiert das Tabellenblatt. Tabellenblätter lassen sich individuell modifizieren. Sie können Tabellenblätter einfügen, umbenennen, löschen, drucken, kopieren, verschieben und vieles mehr. Über das Objekt »ChartObject« erstellen Sie Diagramme, die Sie in Tabellen einbetten.

In diesem Kapitel erfahren Sie anhand ausgesuchter Beispiele aus der täglichen Praxis mehr über den Einsatz von Eigenschaften und Methoden des Objekts `Worksheet`. Auch die Themen Pivot-Tabellen und Diagramme werde ich in diesem Kapitel behandeln.

### Kapitelbegleitende Beispiele zum Download

Sie finden alle Beispiele in der Datei `Tabellen.xlsm` aus dem Downloadpaket, das Sie von <http://www.rheinwerk-verlag.de/3891> herunterladen können.



## 6.1 Tabellen einfügen

### Hinweis

Die Datei `Tabellen.xlsm` enthält im Modul `mdl_Allgemein` alle folgenden Makros.

Standardmäßig bietet Excel Ihnen bei der Erstellung einer neuen Arbeitsmappe drei Tabellenblätter an. Wenn Sie weitere hinzufügen möchten, setzen Sie die Methode `Add` ein. Das neu eingefügte Tabellenblatt wird immer vor dem aktiven Tabellenblatt der Arbeitsmappe eingefügt.

```
Sub TabelleEinfügen()
```

```
    Worksheets.Add
```

```
End Sub
```

**Listing 6.1** Neues Tabellenblatt einfügen

Möchten Sie ein Tabellenblatt an einer bestimmten Position einfügen, starten Sie das Makro aus Listing 6.2:

```
Sub TabelleAnPositionEinfügen()

    Worksheets.Add Before:=ThisWorkbook.Worksheets(1)

End Sub
```

#### Listing 6.2 Neues Tabellenblatt als erstes Blatt in eine Mappe einfügen

In Listing 6.2 wurde die neue Tabelle zu Beginn der Arbeitsmappe, also als erste Tabelle, eingefügt. Das bisherige Tabellenblatt mit dem Index 1 wird dann eine Position nach rechts geschoben. Möchten Sie die neue Tabelle ganz am Ende, also ganz rechts, einfügen, setzen Sie das Makro aus Listing 6.3 ein:

```
Sub TabelleAmEndeEinfügen()

    Worksheets.Add After:=Worksheets(Worksheets.Count)

End Sub
```

#### Listing 6.3 Neues Tabellenblatt am Ende der Arbeitsmappe einfügen

Um zu ermitteln, welche die gewünschte Einfügeposition des neuen Tabellenblattes ist, müssen Sie zuerst herausfinden, wie viele Tabellenblätter bereits in der Arbeitsmappe enthalten sind. Dabei hilft Ihnen die Eigenschaft `Count`. Sie liefert die Anzahl der Tabellenblätter. Danach brauchen Sie nur noch den Parameter `After` anzugeben, und das neue Tabellenblatt wird als letztes Tabellenblatt in die Arbeitsmappe eingefügt.

## 6.2 Tabellenblätter benennen

Excel vergibt beim Einfügen von Tabellennamen selbständig Namen, die sich aus dem Ausdruck `TABELLE` und einer fortlaufenden Zahl zusammensetzen. Wenn Sie andere Namen verwenden möchten, können Sie dies jederzeit tun.

### 6.2.1 Eine neue Mappe mit 12 Monatstabellen anlegen und benennen

Bei der nächsten Aufgabe – siehe Listing 6.4 – soll eine neue Mappe mit 12 Tabellen erstellt werden. Diese Tabellen sollen danach nach den Monatsnamen benannt werden.

```
Sub MappeMit12MonatenAnlegen()
    Dim intAnz As Integer
    Dim wkbMappe As Workbook
```

```
Dim wksBlatt As Worksheet

intAnz = Application.SheetsInNewWorkbook
Application.SheetsInNewWorkbook = 12
Set wkbMappe = Workbooks.Add
Application.SheetsInNewWorkbook = intAnz

For Each wksBlatt In wkbMappe.Worksheets

    wksBlatt.Name = MonthName(wksBlatt.Index)

Next wksBlatt

End Sub
```

#### Listing 6.4 Eine neue Mappe mit 12 Tabellen anlegen und nach Monatsnamen benennen

Ermitteln Sie zunächst einmal über die Eigenschaft `SheetsInNewWorkbook`, welche Applikationseinstellung bezüglich der Anzahl Tabellen festgelegt ist, wenn eine neue Mappe angelegt ist. Speichern Sie diesen Wert in der Variablen `intAnz` zwischen. Jetzt ändern Sie den Wert dieser Eigenschaft in 12 Tabellen. Durch die Methode `Add`, die auf das Auflistungsobjekt `Workbooks` angewendet wird, wird nun eine neue Arbeitsmappe mit 12 Tabellen erstellt. Stellen Sie jetzt am besten gleich wieder die vorher eingestellte Anzahl der angebotenen Tabellen ein. Dazu weisen Sie der Eigenschaft `SheetsInNewWorkbook` den Inhalt der Variablen `intAnz` zu.

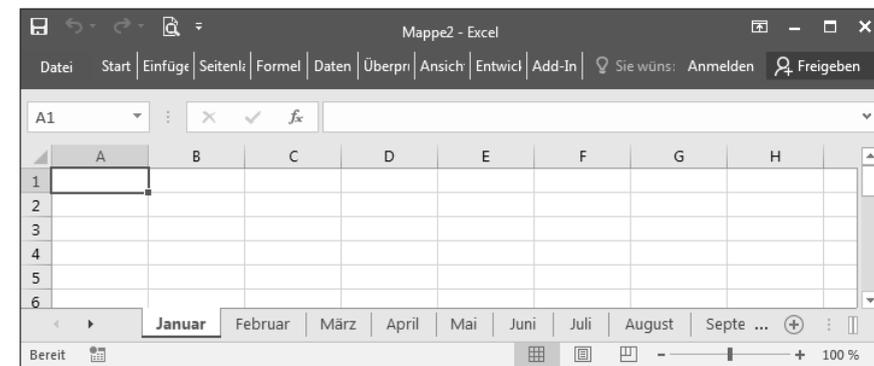


Abbildung 6.1 Die Monatstabellen wurden automatisch angelegt.

Setzen Sie danach eine Schleife des Typs `For Each . . . Next` ein, in der Sie alle 12 Tabellen nacheinander verarbeiten. In der Schleife führen Sie die Benennung der Tabellen durch. Dazu setzen Sie die Funktion `MonthName` ein. Diese Funktion benötigt einen Wert zwischen 1 und 12, um den gewünschten Monat direkt aus der Windows-Sys-

temsteuerung in der dort eingestellten Landessprache zu holen. Diesen Wert leiten Sie über die Eigenschaft `Index` der jeweiligen Tabelle ab.

### 6.2.2 Eine neue Mappe mit den nächste 14 Tagen anlegen

Im Beispiel aus Listing 6.5 wird eine neue Arbeitsmappe zunächst mit nur einer Tabelle angelegt. Danach werden über eine Schleife 14 weitere Tabellen hinzugefügt und mit einem fortlaufenden Datum versehen.

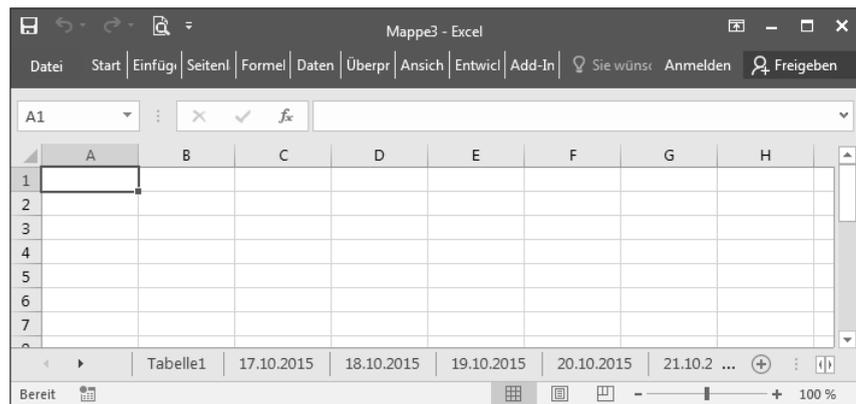
```
Sub TabellenMitDatumEinfügen()
    Dim inTabz As Integer
    Dim intAnz As Integer
    Dim wkbMappe As Workbook
    Dim wksBlatt As Worksheet

    intAnz = Application.SheetsInNewWorkbook
    Application.SheetsInNewWorkbook = 1
    Set wkbMappe = Workbooks.Add
    Application.SheetsInNewWorkbook = intAnz

    For inTabz = 1 To 14
        wkbMappe.Worksheets.Add after:=wkbMappe.Worksheets(Worksheets.Count)
        wkbMappe.Worksheets(Worksheets.Count).Name = Date + inTabz
    Next inTabz

End Sub
```

**Listing 6.5** Eine neue Mappe für die nächsten 14 Tage wird angelegt.



**Abbildung 6.2** Eine neue Mappe für die nächsten 14 Tage steht bereit.

Setzen Sie die Eigenschaft `SheetsInNewWorkbook` auf den Wert 1, und fügen Sie anschließend in einer `For ... Next`-Schleife weitere 14 Tabellen über den Einsatz der Methode `Add` hinzu. Innerhalb der Schleife führen Sie die Benennung beginnend vom aktuellen Tagesdatum durch. Mit jedem Schleifendurchlauf wird der Schleifenzähler, der gleichzeitig auch der Tageszähler ist, um den Wert 1 inkrementiert.

### 6.2.3 Tabelle einfügen und benennen kombinieren

Selbstverständlich können Sie das Einfügen von neuen Tabellenblättern und deren Benennung auch in einem Aufwasch erledigen:

```
Sub TabelleEinfügenUndBenennen()

    Worksheets.Add.Name = "Tabelle10"

End Sub
```

**Listing 6.6** Neue Tabelle einfügen und benennen in einem Schritt

Allerdings ist hierbei zu beachten, dass Sie sich sicher sein müssen, ob der Name nicht schon in Verwendung ist, da es sonst zu einem Laufzeitfehler kommt.

## 6.3 Tabellen markieren

Um eine einzige Tabelle zu markieren, können Sie den Befehl `Worksheets("Tabelle1").Select` anwenden. Sollen es ein paar Tabellen mehr sein, dann wenden Sie das Makro aus Listing 6.7 an:

```
Sub MehrereTabellenMarkieren()

    Sheets(Array("Tabelle1", "Tabelle2")).Select

End Sub
```

**Listing 6.7** Mehrere Tabellen markieren

Mit Hilfe der Funktion `Array` bilden Sie ein Datenfeld, in das Sie die Namen der Tabellen aufnehmen, die Sie markieren möchten.

Soll diese Lösung ein wenig dynamischer sein, dann markieren Sie in der nächsten Aufgabe einmal alle Tabellen einer Arbeitsmappe mit Ausnahme der ersten Tabelle. Wie das geht, entnehmen Sie dem Makro aus Listing 6.8:

```

Sub MehrereTabellenMarkierenÜberArray()
    Dim lngZ As Long
    Dim intTab As Integer
    Dim Vardat() As Long

    intTab = ThisWorkbook.Worksheets.Count

    ReDim Vardat(1 To intTab - 1)

    For lngZ = 2 To intTab
        Vardat(lngZ - 1) = lngZ
    Next lngZ

    ThisWorkbook.Worksheets(Vardat).Select

End Sub

```

**Listing 6.8** Mehrere Tabellen markieren (nur nicht die erste)

Ermitteln Sie im ersten Schritt einmal die Gesamtzahl der Tabellen, die sich in der aktiven Arbeitsmappe befinden, und speichern Sie diese Information in der Variablen `intTab`. Danach definieren Sie mit der Anweisung `ReDim` ein Datenfeld in der Größe der Anzahl der Tabellen in der Arbeitsmappe. Von dieser ermittelten Größe subtrahieren Sie den Wert 1, da Sie die erste Tabelle nicht markieren möchten. In einer Schleife füllen Sie dann das Datenfeld. Am Ende der Schleife stehen die Namen aller Tabellen im Datenfeld. Markieren Sie anschließend alle im Datenfeld stehenden Tabellen mit der Methode `Select`.

**6.4 Tabellenblätter gruppieren**

In Excel haben Sie die Möglichkeit, Ihre Arbeit an einem Tabellenblatt automatisch auch für andere Tabellenblätter gültig zu machen. Dazu gruppieren Sie die einzelnen Tabellenblätter. Manuell klappt das, indem Sie die `[Strg]`-Taste gedrückt halten und die einzelnen Tabellenregister mit der linken Maustaste anklicken. Das Ergebnis dieser Aktion können Sie selbstverständlich auch mit VBA erreichen. Im Folgenden erfahren Sie, wie Sie das machen.

**6.4.1 Mehrere Tabellen gruppieren**

```

Sub MehrereTabellenblätterMarkieren()

    On Error Resume Next

```

```

    Sheets(Array("Tabelle2", "Tabelle3", "Tabelle5")).Select

```

```

End Sub

```

**Listing 6.9** Mehrere Tabellenblätter gruppieren

Die Funktion `Array` ermöglicht es Ihnen, eine durch Kommas getrennte Liste von Werten (hier Tabellennamen) anzugeben. Auch hier ist wieder die `On Error`-Anweisung wichtig, um eine Fehlermeldung zu vermeiden, falls eines der Tabellenblätter nicht vorhanden ist.

**6.4.2 Alle Tabellen gruppieren**

Möchten Sie alle Tabellenblätter einer Arbeitsmappe gruppieren, können Sie die Tabellenblätter in ein Array einlesen und anschließend gruppieren. Dazu wenden Sie das Makro aus Listing 6.10 an:

```

Sub AlleTabellenMarkieren()
    Dim lngZ As Long
    Dim lngTab As Long
    Dim TabArray() As Long

    lngTab = ThisWorkbook.Worksheets.Count

    ReDim TabArray(1 To lngTab)
    On Error Resume Next

    For lngZ = 1 To lngTab
        TabArray(lngZ) = 1
    Next lngZ

    ThisWorkbook.Worksheets(TabArray).Select

End Sub

```

**Listing 6.10** Alle Tabellenblätter einer Arbeitsmappe gruppieren

Ermitteln Sie mit der Eigenschaft `Count` die Anzahl der Tabellenblätter, die in der Arbeitsmappe enthalten sind. Mit der Anweisung `ReDim` reservieren Sie Speicherplatz für die dynamische Datenfeldvariable `TabArray`. Danach füllen Sie das Array mit Hilfe einer `For ... Next`-Schleife. Im Anschluss daran werden alle Tabellenblätter der Arbeitsmappe gruppiert.

### 6.4.3 Gruppierete Tabellen übertragen

Im nächsten Beispiel werden alle gruppierten Tabellen in eine neue Arbeitsmappe eingefügt:

```
Sub GruppiereteTabellenInNeueMappeTransferieren()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWindow.SelectedSheets
        wksBlatt.Copy
    Next wksBlatt

End Sub
```

#### Listing 6.11 Gruppierete Tabellen in neue Arbeitsmappe überführen

Setzen Sie die Eigenschaft `SelectedSheets` ein, um alle markierten Tabellenblätter zu ermitteln. Kopieren Sie all diese Tabellen mit Hilfe der Methode `Copy`.

### 6.4.4 Gruppierete Tabellen ermitteln

Möchten Sie herausfinden, welche Tabellen in Ihrer Arbeitsmappe gruppiert sind, dann starten Sie das folgende Makro:

```
Sub GruppiereteBlätterErmitteln()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Windows(1).SelectedSheets
        MsgBox wksBlatt.Name
    Next wksBlatt

End Sub
```

#### Listing 6.12 Gruppierete Tabellen ermitteln

Mit Hilfe der Eigenschaft `Name` ermitteln Sie die Namen der gruppierten Tabellen. Gruppierete Tabellen fragen Sie über die Eigenschaft `SelectedSheets` ab.

## 6.5 Tabellenblätter löschen

Wie Sie Tabellenblätter einfügen, wissen Sie jetzt. Aber wie löschen Sie Tabellenblätter? Dafür setzen Sie die Methode `Delete` ein.

### 6.5.1 Eine Tabelle löschen

Beim Beispiel aus Listing 6.13 wird TABELLE1 in der Arbeitsmappe gelöscht. Hierbei müssen Sie zwischen dem »normalen« Tabellennamen und dem Codenamen der Tabelle unterscheiden.

```
Sub TabellenblattLöschen()
    On Error GoTo fehler
    Sheets("Tabelle1").Delete

    'oder eben über den Codenamen
    'Tabelle1.Delete

Exit Sub
```

```
fehler:
    MsgBox "Es gibt keine Tabelle1 zum Löschen"
```

```
End Sub
```

#### Listing 6.13 Ein benanntes Tabellenblatt löschen

Zu Beginn sorgt die Anweisung `On Error` dafür, dass im Fehlerfall sofort zur Textmarke `fehler` gesprungen wird. Ein Fehler kann z. B. auftreten, wenn die Tabelle gar nicht in der Arbeitsmappe enthalten ist. Danach wird versucht, die Tabelle TABELLE1 über den Einsatz der Methode `Delete` zu löschen. Sollte der Vorgang erfolgreich sein, wird die nächste Zeile abgearbeitet, wenn nicht, wird zur Textmarke `fehler` gesprungen. Die Anweisung `Exit Sub` sorgt dafür, dass nach dem erfolgreichen Löschen des Tabellenblattes das Makro sofort beendet, also die Textmarke `fehler` nicht mehr abgearbeitet wird. Die Textmarke `fehler` leitet die Fehlerbehandlung ein. Sie wird nur ausgeführt, wenn z. B. versucht wurde, eine Tabelle zu löschen, die es gar nicht mehr gibt. Als Fehlerreaktion wird eine einfache Meldung auf dem Bildschirm ausgegeben.

Diese `On Error`-Geschichte sollten Sie aber auch nicht übertreiben. Oft werden mir Quellcodes zur Begutachtung vorgelegt, bei denen es davon nur so wimmelt. Einen möglichen Fehler mehrfach zu ignorieren, ist keine gute Reaktion auf einen Fehler. Besser wäre es, eine Funktion zu haben, die immer vorher prüft, ob eine Tabelle sich in der Arbeitsmappe befindet. In diesem Fall bräuchten Si keine Fehlerbehandlung mehr. In meinen Software-Projekten habe ich immer eine solche Funktion zur Verfügung, siehe Listing 6.14:

```
Function TabDa(strBlatt As String) As Boolean
    Dim wksBlatt As Worksheet
```

```

For Each wksBlatt In ThisWorkbook.Worksheets

    If wksBlatt.Name = strBlatt Then
        TabDa = True
        Exit For
    End If

Next wksBlatt

End Function

```

**Listing 6.14** Funktion, die prüft, ob eine bestimmte Tabelle in der Mappe existiert

Immer, wenn Sie eine Tabelle löschen oder auf eine Tabelle zugreifen müssen, rufen Sie vorher die Funktion `TabDa` auf und übergeben ihr den Namen der entsprechenden Tabelle. In der Funktion selbst wird eine Schleife des Typs `For Each ... Next` durchlaufen, die den Namen der an die Funktion übergebenen Tabelle mit dem jeweiligen Namen der Tabelle vergleicht, den Sie durch die Schleife ansprechen. Wird die gesuchte Tabelle in der Arbeitsmappe gefunden, dann setzen Sie den Rückgabewert der Funktion auf `True` und verlassen vorzeitig die Schleife, indem Sie die Anweisung `Exit For` einsetzen. Damit springen Sie direkt im Anschluss auch aus der Funktion und landen wieder im aufrufenden Makro aus Listing 6.15.

```

Sub TabelleBenennenMitVorherigerPrüfung()

    If TabDa("Tabelle10") = False Then
        Worksheets.Add.Name = "Tabelle10"
    Else
        MsgBox "Die Tabelle10 ist bereits in der Mappe!"
    End If

End Sub

```

**Listing 6.15** Das Makro ruft eine Funktion auf und verarbeitet ihre Rückmeldung.

### 6.5.2 Bestimmte Tabellen aus einer Mappe entfernen

Im Beispiel aus Listing 6.16 werden alle Tabellen aus der Arbeitsmappe entfernt, die im Tabellennamen das Kürzel »neu« aufweisen.

```

Sub BestimmteTabellenEntfernen()
    Dim wksBlatt As Worksheet

    Application.DisplayAlerts = False

```

```

For Each wksBlatt In ActiveWorkbook.Worksheets

    If wksBlatt.Name Like "*neu*" Then

        wksBlatt.Delete

    End If

Next wksBlatt

Application.DisplayAlerts = True

End Sub

```

**Listing 6.16** Alle Tabellen mit einer bestimmten Zeichenfolge im Namen werden gelöscht.

Mit Hilfe der Eigenschaft `DisplayAlerts` schalten Sie die Excel-Warnmeldungen temporär aus, indem Sie dieser Eigenschaft den Wert `False` zuweisen. Dies ist wichtig, da Sie sonst die Löschung einer jeden Tabelle bestätigen müssten. In einer Schleife der Form `For Each ... Next` arbeiten Sie alle Tabellen der Arbeitsmappe nacheinander ab. Im Innern der Schleife prüfen Sie mit dem Operator `Like`, ob im Tabellennamen der Begriff »neu« vorkommt. Dabei unterscheidet dieser Operator zwischen Groß- und Kleinschreibung. Wird eine Tabelle gefunden, deren Name die Zeichenfolge »neu« enthält, dann wird die Methode `Delete` verwendet, um die Tabelle zu löschen.

#### Hinweis

Möchten Sie, dass Excel nicht zwischen Groß- und Kleinschreibung unterscheidet, dann tauschen Sie die Bedingung in Listing 6.16 durch diese Bedingung aus:

```
If UCase(wksBlatt.Name) Like "*NEU*" Then
```

### 6.5.3 Tabellen mit gefärbten Registerlaschen entfernen

Zur Identifizierung zu löschender Tabellen könnten Sie auch die Farbe der Tabellenreiter heranziehen. Das Makro aus Listing 6.17 entfernt alle Tabellen, die mit rotem Tabellenreiter formatiert wurden, ohne Rückfrage aus der Arbeitsmappe.

```

Sub FarbTabellenEntfernen()
    Dim wksBlatt As Worksheet

    Application.DisplayAlerts = False

    For Each wksBlatt In ActiveWorkbook.Worksheets

```

```

    If wksBlatt.Tab.ColorIndex = 3 Then

        wksBlatt.Delete

    End If

Next wksBlatt

Application.DisplayAlerts = True

End Sub

```

**Listing 6.17** Alle Tabellen mit einem roten Tabellenreiter werden entfernt.

Über das Objekt `Tab` sprechen Sie den Tabellenreiter einer Tabelle an. Mit Hilfe der Eigenschaft `ColorIndex` lesen Sie die Farbe des Tabellenreiters aus. Ist der Tabellenreiter rot eingefärbt, dann gibt die Eigenschaft die Farbnummer 3 zurück. In diesem Fall wenden Sie die Methode `Delete` an, um die Tabelle zu löschen.

#### Hinweis

Möchten Sie nicht nur rote, sondern alle farbigen Tabellen aus der Mappe entfernen, dann passen Sie das Löschkriterium wie folgt an:

```
If wksBlatt.Tab.ColorIndex > 0 Then
```

#### 6.5.4 Leere Tabellen aus Arbeitsmappen entfernen

Bei der nächsten Lösung sehen Sie im Makro aus Listing 6.18, wie Sie leere Tabellen aus einer Arbeitsmappe entfernen.

```

Sub LeereTabellenAusMappeEntfernen()
    Dim Blatt As Worksheet

    For Each Blatt In ActiveWorkbook.Worksheets
        If Application.WorksheetFunction.CountA(Blatt.Cells) = 0 Then
            Application.DisplayAlerts = False
            Blatt.Delete
            Application.DisplayAlerts = True
        End If
    Next Blatt

End Sub

```

**Listing 6.18** Leere Tabellen aus der Arbeitsmappe entfernen

In einer Schleife des Typs `For Each ... Next` arbeiten Sie sich Tabelle für Tabelle durch die aktive Arbeitsmappe hindurch. Innerhalb der Schleife prüfen Sie mit Hilfe der Tabellenfunktion `ANZAHL2` (englisch `CountA`), ob die Anzahl der Zellen, die einen Eintrag enthalten, null ist. Ist dies der Fall, dann sind in der Tabelle keine Daten vorhanden, und Sie wenden die Methode `Delete` an, um die leere Tabelle zu löschen.

## 6.6 Tabellenblätter ein- und ausblenden

Wenn Sie bestimmte Tabellenblätter nicht mit einem Passwort schützen, jedoch trotzdem einen gewissen Schutz Ihrer Daten erreichen möchten, können Sie Tabellenblätter auch ausblenden. Das Ein- und Ausblenden von Tabellenblättern erreichen Sie mit der Eigenschaft `Visible`:

```

Sub TabelleAusblenden()

    On Error Resume Next

    Worksheets("Tabelle1").Visible = False

    'oder

    Tabelle1.Visible = xlSheetHidden

End Sub

```

**Listing 6.19** Tabellenblatt ausblenden

Nachdem Sie das Makro `TabelleAusblenden` ausgeführt haben, wird die Tabelle in der Arbeitsmappe nicht mehr angezeigt.

Anwender der Versionen Excel 2007 bis 2016 klicken mit der rechten Maustaste auf einen beliebigen Tabellenreiter und wählen den Befehl `EINBLENDEN` aus dem Kontextmenü aus.

Das Einblenden eines ausgeblendeten Tabellenblatts funktioniert in VBA wie folgt:

```

Sub TabelleEinblenden()

    On Error Resume Next

    Sheets("Tabelle1").Visible = True

    'oder

```

```
Tabelle1.Visible = xlSheetVisible
```

```
End Sub
```

**Listing 6.20** Tabellenblatt wieder einblenden

### 6.6.1 Tabellenblätter sicher ausblenden

Wenn Sie verhindern möchten, dass der Anwender Ihre ausgeblendeten Tabellenblätter über die Benutzeroberfläche wieder einblendet, dann verwenden Sie bei der Eigenschaft `Visible` die Konstante `xlSheetVeryHidden`:

```
Sub TabelleSicherAusblenden()
```

```
On Error Resume Next
```

```
Tabelle1.Visible = xlSheetVeryHidden
```

```
End Sub
```

**Listing 6.21** Tabelle ausblenden (sichere Methode)

In diesem Fall können Sie Ihre ausgeblendete Tabelle nur mit einem Makro wieder verfügbar machen. Dazu setzen Sie das Makro aus Listing 6.21 ein.

### 6.6.2 Tabellen je nach Status ein- oder ausblenden

In einer Arbeitsmappe sollen alle eingeblendeten Tabellenblätter ausgeblendet bzw. alle ausgeblendeten Tabellenblätter eingeblendet werden. Das Makro zur Umsetzung dieser Aufgabe sehen Sie in Listing 6.22:

```
Sub TabellenJeNachStatusEinAusblenden()
```

```
Dim wksBlatt As Worksheet
```

```
For Each wksBlatt In ActiveWorkbook.Worksheets
```

```
    Select Case wksBlatt.Visible
```

```
        Case xlSheetHidden: Blatt.Visible = xlSheetVisible
```

```
        Case xlSheetVisible: Blatt.Visible = xlSheetHidden
```

```
    End Select
```

```
Next wksBlatt
```

```
End Sub
```

**Listing 6.22** Tabellenblätter je nach Status ein- oder ausblenden

In einer Schleife der Art `For Each ... Next` überprüfen Sie mit Hilfe einer `Select Case`-Anweisung, wie der Status der Eigenschaft `Visible` für das jeweilige Tabellenblatt ist. Je nach Status wird der Eigenschaft dann entweder die Konstante `xlSheetVisible` bzw. `xlSheetHidden` zugewiesen.

#### Achtung

Achten Sie darauf, dass Sie die Anweisung `On Error` in das Makro integrieren. In einer Arbeitsmappe muss immer wenigstens eine Tabelle sichtbar bleiben. Versucht nun das Makro, das letzte Tabellenblatt auszublenden, kommt es zum Fehlerfall, der Sie aber mit dieser Anweisung abfangen.

### 6.6.3 Alle Tabellenblätter anzeigen

Ausgeblendete Tabellenblätter werden oft vergessen. Diese versteckten Tabellenblätter schlummern dann jahrelang in Arbeitsmappen. Eines Tages erfahren Sie mehr durch Zufall, dass es in der Arbeitsmappe versteckte Tabellenblätter gibt.

Schreiben Sie daher ein Makro, das in der aktiven Arbeitsmappe alle Tabellenblätter wieder sichtbar macht:

```
Sub VersteckteBlätterEinblenden()
```

```
Dim wksBlatt As Worksheet
```

```
For Each wksBlatt In ActiveWorkbook.Worksheets
```

```
    wksBlatt.Visible = True
```

```
Next wksBlatt
```

```
End Sub
```

**Listing 6.23** Alle Tabellenblätter einblenden

In einer `For Each ... Next`-Schleife setzen Sie die Eigenschaft `Visible` aller Tabellenblätter auf den Wert `True`.

### 6.6.4 Alle Tabellen außer der aktiven Tabelle ausblenden

Wenn Sie möchten, können Sie alle Tabellenblätter einer Arbeitsmappe mit Ausnahme des aktiven Tabellenblattes ausblenden, indem Sie das Makro aus Listing 6.24 starten:

```
Sub NurAktivesBlattSichtbar()
```

```
Dim wksBlatt As Worksheet
```

```

For Each wksBlatt In ActiveWorkbook.Worksheets

    If wksBlatt.Name <> ActiveSheet.Name Then
        wksBlatt.Visible = xlSheetHidden
    End If

Next wksBlatt

End Sub

```

**Listing 6.24** Alle Tabellen außer der aktiven Tabelle werden ausgeblendet.

Definieren Sie zuerst eine Objektvariable vom Typ `Worksheet`. Danach greifen Sie in einer Schleife des Typs `For Each ... Next` auf das Auflistungsobjekt `Worksheets` zu, das alle Tabellenblätter der aktiven Arbeitsmappe enthält. Innerhalb der Schleife vergleichen Sie den Namen des aktiven Tabellenblattes mit dem jeweiligen Tabellenblatt aus dem Auflistungsobjekt. Mit der Eigenschaft `Visible`, die Sie auf den Wert `False` oder die Konstante `xlSheetHidden` setzen, blenden Sie alle Tabellenblätter aus der Arbeitsmappe mit Ausnahme des aktiven Tabellenblattes aus.

## 6.7 Tabellenblätter schützen

Haben Sie wichtige Daten auf Ihrem Tabellenblatt erfasst und möchten Sie sie vor Veränderung durch andere schützen, so können Sie Ihre Tabelle mit einem Passwort belegen:

```

Sub BlattschutzEinschalten()

    ActiveSheet.Protect Password:="test", _
        DrawingObjects:=True, Contents:=True, Scenarios:=True

End Sub

```

**Listing 6.25** Tabellen schützen

### Die Syntax

Es lohnt sich, die Syntax der Methode `Protect` einmal näher zu betrachten. Dabei beschränken wir uns zunächst auf die Argumente, die in allen Versionen von Excel 97 bis Excel 2016 verfügbar sind:

```

ActiveSheet.Protect _
(Passwort, DrawingObjects, Contents, _
Scenarios, UserInterfaceOnly)

```

### Die Argumente der Methode `Protect`

- ▶ Im Argument `Password` geben Sie eine Zeichenfolge an, die das groß-/kleinschreibungsabhängige Kennwort für das Blatt oder die Arbeitsmappe festlegt. Wenn Sie dieses Argument weglassen, kann der Schutz des Blattes oder der Arbeitsmappe ohne Angabe eines Kennworts aufgehoben werden. Weisen Sie dagegen ein Kennwort zu, muss das Kennwort angegeben werden, um den Schutz des Blattes oder der Arbeitsmappe aufzuheben.
- ▶ Mit dem Argument `DrawingObjects` legen Sie fest, ob Sie zusätzlich zu Ihren Zellen auch Formen – wie z. B. Blockpfeile, Sterne oder Banner – schützen möchten. Diese Formen werden standardmäßig jedoch nicht geschützt. Wenn Sie Formen schützen möchten, setzen Sie das Argument auf den Wert `True`.
- ▶ Bei dem Argument `Contents`, das standardmäßig auf `True` gesetzt ist, werden die Zellen eines Tabellenblattes geschützt.
- ▶ Das Argument `Scenarios` gilt nur für Arbeitsblätter und bedeutet, dass bestimmte Ansichten und Einstellungen, wie z. B. der eingestellte Zoom, geschützt werden. Die Standardeinstellung ist dabei ebenfalls `True`.
- ▶ Das letzte Argument, `UserInterfaceOnly`, nimmt den Wert `True` an. Damit schützen Sie die Benutzeroberfläche, jedoch keine Makros. Ohne Angabe dieses Arguments gilt der Schutz für Makros und die Benutzeroberfläche.

#### 6.7.1 Tabellenschutz aufheben

Zum Deaktivieren des Tabellenschutzes reicht es, wenn Sie bei der Methode `Unprotect` das Passwort angeben. Sollten Sie Ihr Tabellenblatt ohne Passwort geschützt haben, reicht der Befehl `ActiveSheet.Unprotect`.

```

Sub BlattschutzAusschalten()

    ActiveSheet.Unprotect ("test")

End Sub

```

**Listing 6.26** Tabellenschutz aufheben

#### 6.7.2 Alle Tabellen einer Arbeitsmappe schützen

Wenn Sie alle Tabellenblätter einer Arbeitsmappe schützen und dabei dasselbe Passwort verwenden möchten, können Sie das Makro aus Listing 6.27 nutzen:

```

Sub PasswortFürAlleBlätterEinstellen()
    Dim intTabz As Integer
    Dim intz As Integer

```

```

intTabz = ActiveWorkbook.Worksheets.Count

For i = intz To intTabz

    Worksheets(intz).Protect DrawingObjects:=True, _
        Contents:=True, Scenarios:=True, Password:="test"

Next intz

End Sub

```

**Listing 6.27** Alle Tabellenblätter einer Arbeitsmappe schützen

Um den Blattschutz für alle Tabellenblätter in der Mappe wieder aufzuheben, starten Sie das Makro aus Listing 6.28:

```

Sub PasswortAlleBlätterEntfernen()
    Dim intTabz As Integer
    Dim intz As Integer

    intTabz = ActiveWorkbook.Worksheets.Count

    For intz = 1 To Tabz
        Worksheet(intz).Unprotect "test"
    Next intz

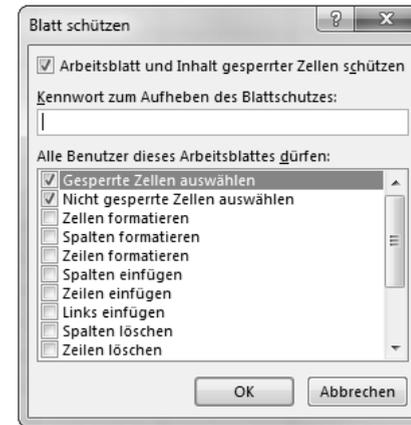
End Sub

```

**Listing 6.28** Blattschutz auf allen Tabellenblättern der Mappe entfernen

### 6.7.3 Weitere Schutzfunktionen ab Excel 2002

Eine sehr gute Verbesserung gegenüber den Vorversionen von Excel können Sie auch beim Schützen Ihrer Tabellen ab der Version Excel 2002 feststellen. Sie haben seitdem die Möglichkeit, zwar einen Blattschutz einzustellen, aber einzelne Aktionen trotz eingestelltem Blattschutz durchzuführen. So können Sie zum Beispiel festlegen, dass Anwender in einer geschützten Tabelle die Filter verwenden sowie Formattierungen durchführen und Zeilen und Spalten einfügen dürfen. Diese und weitere Möglichkeiten sehen Sie, wenn Sie aus dem Menü EXTRAS den Befehl BLATT SCHÜTZEN wählen (Excel 2002 bis Excel 2003). Anwender der Versionen Excel 2007 bis 2016 klicken im Ribbon ÜBERPRÜFEN auf die Schaltfläche BLATT SCHÜTZEN.



**Abbildung 6.3** Erweiterte Schutzmöglichkeiten ab Excel 2002

Diese manuellen Einstellungen können Sie aber auch über ein Makro durchführen. Das folgende Makro lässt in einer geschützten Tabelle alle aktivierten Optionen zu:

```

Sub TabelleSchützen()

    Tabelle2.Protect _
        DrawingObjects:=True, _
        Contents:=True, Scenarios:=True, _
        AllowFormattingCells:=True, _
        AllowFormattingColumns:=True, _
        AllowFormattingRows:=True, _
        AllowInsertingColumns:=True, _
        AllowInsertingRows:=True, _
        AllowInsertingHyperlinks:=True, _
        AllowDeletingColumns:=True, _
        AllowDeletingRows:=True, _
        AllowSorting:=True, _
        AllowFiltering:=True, _
        AllowUsingPivotTables:=True

End Sub

```

**Listing 6.29** Tabelle schützen (ab Excel 2002)

### 6.7.4 Passwort – Einstellungsdialog mit verschlüsseltem Passwort aufrufen

Das Makro aus Listing 6.30 ruft den Dialog BLATT SCHÜTZEN bereits mit voreingestelltem Passwort auf:

```
Sub DialogBlattSchutzAufrufen()

    Application.Dialogs(xlDialogProtectDocument).Show 1, 1, "TEST"

End Sub
```

**Listing 6.30** Blattschutzdialog mit verschlüsseltem Passwort aufrufen



**Abbildung 6.4** Der integrierte Dialog »Blatt schützen« wurde aufgerufen.

## 6.8 Tabellen einstellen

Wenn Sie mehrere Tabellen in einer Arbeitsmappe verwalten, dann ist es mitunter nützlich, bestimmte Einstellungen für alle Tabellen gleichermaßen vorzunehmen. Ich denke da beispielsweise an eine einheitliche Zoomeinstellung, eine überall gleiche Kopf- und Fußzeile, an eine einheitliche Positionierung des Cursors und des Bildlaufs und vieles mehr.

### Hinweis

Sie finden alle folgenden Makros in *Tabellen.xlsm* im Modul `mdl_Einstellen`.

### 6.8.1 Registerlaschen ein- und ausblenden

Standardmäßig werden die Registerlaschen am unteren Bildrand von Excel angezeigt. Wenn diese Standardeinstellung Sie stört, dann können Sie die Anzeige der Registerlaschen ausblenden. Im Makro aus Listing 6.31 werden die Registerlaschen in der kompletten Arbeitsmappe ein- bzw. ausgeblendet:

```
Sub RegisterAusEinblenden()

    ActiveWindow.DisplayWorkbookTabs = Not ActiveWindow.DisplayWorkbookTabs

End Sub
```

**Listing 6.31** Tabellenregister ein- oder ausblenden

Über die Eigenschaft `DisplayWorkbookTabs` können Sie die Registerlaschen Ihrer Tabelle ein- und ausblenden. Setzen Sie diese Eigenschaft auf den Wert `True`, wenn Sie die Registerlaschen anzeigen möchten. Weisen Sie der Eigenschaft den Wert `False` zu, um die Registerlaschen wieder auszublenden. Den dynamischen Wechsel zwischen Anzeigen und Ausblenden der Registerlaschen bekommen Sie über eine Gegenüberstellung hin. Dabei können Sie sich diese Gegenüberstellung wie einen Lichtschalter vorstellen, also an und aus.

### 6.8.2 Tabellenansicht anpassen

In einer Tabelle können Sie sich die Daten jederzeit etwas größer oder auch verkleinert anzeigen lassen. Um diese Aufgabe schrittweise zu erledigen, können Sie ein Makro einsetzen, das die Ansicht um jeweils 10 % vergrößert oder verkleinert:

```
Sub TabelleVergrößern()
    Dim intFaktor As Integer

    intFaktor = ActiveWindow.Zoom
    ActiveWindow.Zoom = intFaktor + 10

End Sub
```

**Listing 6.32** Tabellenansicht vergrößern

Über die Eigenschaft `Zoom` können Sie einen Vergrößerungsfaktor bis zu 400 % einstellen, aber bei der Verkleinerung einer Tabelle können Sie keine niedrigeren Werte Wert 10 % einstellen. Bei 100 % ist eine 1 : 1-Ansicht gegeben.

```
Sub VerkleinernTabelle()
    Dim intFaktor As Integer

    intFaktor = ActiveWindow.Zoom
    ActiveWindow.Zoom = intFaktor - 10

End Sub
```

**Listing 6.33** Tabellenansicht verkleinern

### 6.8.3 Einen einheitliche Zoomeinstellung vornehmen

Beim nächsten Beispiel aus Listing 6.34 stellen wir für alle Tabellen für eine bessere Lesbarkeit die Zoomeinstellung auf 120 % ein.

```
Sub ZoomEinstellungVornehmen()
    Dim wksBlatt As Worksheet
    Dim IntTab As Integer

    IntTab = ActiveSheet.Index
    For Each wksBlatt In ActiveWorkbook.Worksheets

        wksBlatt.Activate
        ActiveWindow.Zoom = 120

    Next wksBlatt

    Worksheets(IntTab).Select

End Sub
```

**Listing 6.34** Eine einheitliche Zoomeinstellung für alle Tabellen vornehmen

Da die Zoomeinstellung immer nur für eine Tabelle einstellbar ist, setzen Sie eine Schleife des Typs `For Each ... Next` auf und durchlaufen Tabelle für Tabelle. Innerhalb der Schleife wenden Sie die Methode `Activate` an, um die jeweilige Tabelle zu aktivieren. Danach weisen Sie der Eigenschaft `Zoom` des aktiven Fensters die gewünschte Einstellung zu.

Da Sie im Prinzip durch die ganze Mappe hüpfen, sollten Sie sich zu Beginn des Makros in einer Variablen merken, von welcher Tabelle aus Sie starten. Dies können Sie über die Eigenschaft `Index` abfragen. Am Ende selektieren Sie diese Startabelle wieder, indem Sie sie über die Methode `Select` auswählen.

#### Hinweis

Die Methode `Activate` funktioniert übrigens auch bei ausgeblendeten Tabellen.

Die Methode `Select` verursacht einen Laufzeitfehler, wenn versucht wird, eine ausgeblendete Tabelle zu selektieren.

### 6.8.4 Tabellenblätter sortieren

In umfangreichen Excel-Arbeitsmappen geht leicht einmal der Überblick verloren. Aus diesem Grund ist es vorteilhaft, die Tabellen alphabetisch nach Tabellennamen sortiert in der Arbeitsmappe anzuordnen.

Das Makro für die Sortierung der Tabellenblätter lautet:

```
Sub ArbeitsblätterSortieren()
    Dim intMax As Integer
    Dim intz As Integer
    Dim intn As Integer

    Application.ScreenUpdating = False

    intMax = ActiveWorkbook.Worksheets.Count

    For intz = 1 To intMax

        For intn = intz To intMax

            If UCase(Worksheets(intn).Name) < UCase(Worksheets(intz).Name) Then
                Worksheets(intn).Move before:=Worksheets(intz)
            End If

        Next intn

    Next intz

    Application.ScreenUpdating = True

End Sub
```

**Listing 6.35** Alle Tabellen werden alphabetisch in der Arbeitsmappe angeordnet.

Um Arbeitsblätter zu sortieren, durchlaufen Sie zwei verschachtelte `For ... Next`-Schleifen. Beide haben als Endbedingung immer die Anzahl der Tabellen, die in der Mappe enthalten sind. Innerhalb der zweiten Schleife werden die Namen der Tabellenblätter verglichen. Beim Vergleich der Tabellennamen werden diese erst einmal in Großbuchstaben umgewandelt, um sicherzustellen, dass die Groß- und Kleinschreibung beim Sortiervorgang keine Rolle spielt. Je nach Vergleichsergebnis werden die einzelnen Tabellen dann innerhalb der Arbeitsmappe mit Hilfe der Methode `Move` verschoben oder nicht.

### 6.8.5 Kopf- und Fußzeilen einrichten

Standardmäßig werden in Excel 2000 bis 2016 keine Kopf- und Fußzeilen ausgedruckt. Um diese müssen Sie sich selbst kümmern. Dazu verwenden Sie das Objekt `PageSetup`, das Sie für das Tabellenblatt anwenden können.

**Fußzeile mit Anwendernamen**

So fügen Sie beispielsweise den Namen des Anwenders, den genauen Speicherpfad, das heutige Datum oder andere Angaben aus den Dokumenteigenschaften als Kopf- oder Fußzeile ein.

```
Sub BenutzerNameInFußzeile()

    ActiveSheet.PageSetup.RightFooter = Environ("username")

End Sub
```

**Listing 6.36** Fußzeile mit Benutzername generieren**Fußzeile mit Pfad**

Wenn Sie eine Fußzeile mit dem Namen der Arbeitsmappe definieren, können Sie leider aus dieser Angabe nicht ersehen, wo diese Arbeitsmappe gespeichert ist. Daher erstellen Sie ein Makro, das Ihnen eine Fußzeile mit dem Namen des kompletten Pfades der Datei ausgibt:

```
Sub FußzeileMitPfad()

    ActiveSheet.PageSetup.LeftFooter = _
    ActiveWorkbook.FullName

End Sub
```

**Listing 6.37** Fußzeile mit kompletter Pfadangabe der Datei erstellen**Kopfzeile mit Datums- und Zeitangabe**

Im nächsten Beispiel fügen Sie ein vierstelliges Datum in die Kopfzeile sowie die aktuelle Uhrzeit in die Fußzeile ein:

```
Sub KopfzeileMit4stelligemDatum()

    With ActiveSheet.PageSetup
        .LeftHeader = ""
        .CenterHeader = Format(Date, "dd.mm.yyyy")
        .RightHeader = ""
        .LeftFooter = ""
        .CenterFooter = Time
        .RightFooter = ""
    End With
```

```
ActiveWindow.SelectedSheets.PrintPreview
```

```
End Sub
```

**Listing 6.38** Kopf- und Fußzeilen mit Datums- und Zeitangaben bestücken

Mit der Anweisung `With` führen Sie eine Reihe von Anweisungen für ein bestimmtes Objekt aus, ohne den Namen des Objekts mehrmals angeben zu müssen. Dadurch sparen Sie eine Menge Schreibarbeit, und das Ganze sieht auch noch übersichtlicher aus. Um das Datum in eine bestimmte Form zu bringen, setzen Sie die Funktion `Format` ein. Möglich wäre auch die Anweisung

```
CenterHeader = Format(Date, "Long Date"),
```

die zur Folge hätte, dass das Datum ausgeschrieben wird (z. B. »Donnerstag, 16. Oktober 2016«).

Mit der Methode `PrintPreview` zeigen Sie direkt nach dem Festlegen der Kopf- und Fußzeilen das Ergebnis, so wie es in der Seitenansicht aussieht.

**Fußzeile mit Dokumenteigenschaften füllen**

Im nächsten Beispiel greifen Sie auf die Dokumenteigenschaften Ihrer Excel-Arbeitsmappe zurück.

Anwender der Version Excel 2007 klicken auf die runde Office-Schaltfläche links oben und wählen aus dem Menü den Befehl **VORBEREITEN • EIGENSCHAFTEN**.

Anwender der Versionen Excel 2010 und 2016 klicken auf die **DATEI**-Schaltfläche links oben und wählen aus dem Menü den Befehl **INFORMATIONEN**. Dort klicken Sie auf **EIGENSCHAFTEN** und wählen **ERWEITERTE EIGENSCHAFTEN**. In dem neuen Fenster **EIGENSCHAFTEN** wählen Sie das Register **ZUSAMMENFASSUNG**.

```
Sub DateieigenschaftInFußzeile()

    With ActiveSheet.PageSetup
        .LeftFooter = _
        ActiveWorkbook.BuiltinDocumentProperties("Company")
        .RightFooter = _
        ActiveWorkbook.BuiltinDocumentProperties("Author")
    End With

    ActiveWindow.SelectedSheets.PrintPreview

End Sub
```

**Listing 6.39** Fußzeile mit Dokumenteigenschaften füllen

Da Sie die Dokumenteigenschaften auf Englisch ansprechen müssen und in der Onlinehilfe lediglich die deutschen Begriffe aufgeführt werden, orientieren Sie sich an Abbildung 6.5.

	Englisch	Deutsch	Englisch	Deutsch
1	Title	Titel	Number of bytes	Anzahl Bytes
2	Subject	Thema	Number of lines	Anzahl Zeilen
3	Author	Autor	Number of paragraphs	Anzahl Absätze
4	Keywords	Schlüsselworte	Number of slides	Anzahl Blätter
5	Comments	Kommentare	Number of notes	Anzahl Kommentare
6	Template	Vorlage	Number of hidden Slides	Anzahl versteckter Blätter
7	Last author	Letzter Autor	Number of multimedia clips	Anzahl von Multi-Media-Clips
8	Revision number	Revisionsnummer	Hyperlink base	Hyperlink Basis
9	Application name	Anwendungsname	Number of characters	Zeichenanzahl (mit Leerzeichen)
10	Last print date	Letztes Druckdatum		
11	Creation date	Erstellungsdatum		
12	Last save time	Zuletzt gespeichert		
13	Total editing time	Gesamtbearbeitungszeit		
14	Number of pages	Seitenanzahl		
15	Number of words	Wortanzahl		
16	Number of characters	Zeichenanzahl		
17	Security	Sicherheit		
18	Category	Kategorie		
19	Format	Format		
20	Manager	Manager		
21	Company	Unternehmen		
22				
23				

Abbildung 6.5 Die Gegenüberstellung der Dokumenteigenschaften (Englisch – Deutsch)

### Kopfzeile mit Logo einrichten

Sie können ab der Excel-Version 2002 standardmäßig Grafiken in die Kopf- und Fußzeile integrieren. Um beispielsweise eine Grafik in die Kopfzeile der aktiven Tabelle einzufügen, nutzen Sie den folgenden Code:

```
Sub GrafikInKopfzeileEinfügen()
    With ActiveSheet.PageSetup
        .RightHeaderPicture.FileName = ThisWorkbook.Path & "\Logo.jpg"
        .RightHeader = "&G"
    End With
End Sub
```

Listing 6.40 Kopfzeile mit einem Logo ausstatten (Tabelle)



Abbildung 6.6 Ein Logo in die Kopfzeile der aktiven Tabelle einfügen

Über die Eigenschaft `RightHeaderPicture` weisen Sie dem rechten Rand der Kopfzeile Ihrer Tabelle die angegebene Grafik zu. Neben dieser Eigenschaft gibt es fünf weitere, die Sie in Tabelle 6.1 entdecken können:

Eigenschaft	Beschreibung
<code>RightHeaderPicture</code>	Bild rechts in der Kopfzeile
<code>CenterHeaderPicture</code>	Bild in der Mitte der Kopfzeile
<code>LeftFooterPicture</code>	Bild links in der Fußzeile
<code>CenterFooterPicture</code>	Bild in der Mitte der Fußzeile
<code>RightFooterPicture</code>	Bild rechts in der Fußzeile

Tabelle 6.1 Die Positionen in den Kopf- und Fußzeilen bestimmen

Mit Hilfe der Eigenschaft `FileName` geben Sie bekannt, wo die Grafik zu finden ist und wie diese heißt. Mit der Eigenschaft `RightHeader` definieren Sie, was Sie konkret in der Kopfzeile machen möchten. Dazu weisen Sie dieser Eigenschaft einen Formatcode zu. In Tabelle 6.2 finden Sie die dabei möglichen Formatcodes und deren Bedeutung.

Formatcode	Beschreibung
&L	Richtet folgende Zeichen links aus.
&C	Zentriert das folgende Zeichen.
&R	Richtet folgende Zeichen rechts aus.
&E	Schaltet doppeltes Unterstreichen ein oder aus.
&X	Schaltet Hochstellen ein oder aus.
&Y	Schaltet Tiefstellen ein oder aus.
&B	Schaltet Fettdruck ein oder aus.
&I	Schaltet Kursivdruck ein oder aus.
&U	Schaltet Unterstreichen ein oder aus.
&S	Schaltet Durchstreichen ein oder aus.
&D	das aktuelle Datum
&T	Druckt die aktuelle Zeit.
&F	Druckt den Namen des Dokuments.
&A	Druckt den Namen des Registers einer Arbeitsmappe.
&P	Druckt die Seitenzahl.
&P+Zahl	Druckt die Seitenzahl zuzüglich der angegebenen Zahl.
&&	Druckt ein einzelnes kaufmännisches Und-Zeichen (&).
&"Schriftart"	Druckt die folgenden Zeichen in der angegebenen Schriftart; die Schriftart muss von Anführungszeichen eingeschlossen sein.
&nn	Druckt die folgenden Zeichen im angegebenen Schriftgrad. Geben Sie eine zweistellige Zahl an, um den Schriftgrad anzugeben.
&N	Druckt die Gesamtanzahl der Seiten eines Dokuments.

**Tabelle 6.2** Die Formatcodes für die Kopf- und Fußzeile

Sie brauchen sich übrigens keine Sorgen zu machen, wenn Sie eine Arbeitsmappe mit integrierten Grafiken verschicken. Sie müssen keineswegs auch die Grafiken mitversenden. Ist die Grafik einmal in der Kopf- oder Fußzeile integriert, bleibt sie auch darin.

Bei der Lösung aus Listing 6.40 haben wir das Logo lediglich in die Kopfzeile der aktiven Tabelle integriert. Möchten Sie das Logo auf allen Tabellen Ihrer Arbeitsmappe einfügen, dann starten Sie das Makro aus Listing 6.41:

```
Sub GrafikInKopfzeileinfügenAlleTabellen()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets

        wksBlatt.PageSetup.LeftHeaderPicture.FileName = _
            ThisWorkbook.Path & "\Logo.Jpg"
        wksBlatt.PageSetup.LeftHeader = "&G"

    Next wksBlatt

End Sub
```

**Listing 6.41** Kopfzeile mit Logo ausstatten (ganze Arbeitsmappe)

### Mehrzeilige Fußzeile anlegen

Oft findet man bei Geschäftspapieren und offiziellen Briefen eine mehrzeilige Fußzeile. Diese Fußzeile ist aber in den meisten Fällen bereits fest auf das Papier aufgedruckt und wird demnach von Excel nicht mehr erstellt. Die Standardeinstellung bei Kopf- und Fußzeilen in Excel sieht normalerweise mehrzeilige Fußzeilen nicht vor, beziehungsweise es ist relativ aufwendig, solche Mehrzeiler zu erstellen, weil dazu zum einen die Schriftgröße herabgesetzt und zum anderen mehr Platz für die Fußzeile einkalkuliert werden muss. Darüber hinaus kommt es darauf an, welche Informationen Sie in der Fußzeile ausgeben möchten. Um beispielsweise das Erstellungsdatum oder das letzte Änderungsdatum einer Arbeitsmappe in die Fußzeile zu bringen, bedarf es schon eines Makros:

```
Sub FußzeileSpezialAktiveTabelle()

    With ActiveSheet.PageSetup
        .BottomMargin = 56
        .FooterMargin = 42
        .LeftFooter = "&8" & _
            Application.WorksheetFunction.Rept("_", 60) & vbCr & _
            "Erstellungsdatum: " & _
            ActiveWorkbook.BuiltinDocumentProperties _
            ("Creation date") & vbCr & _
            "Letzte Änderung: " & _
            ActiveWorkbook.BuiltinDocumentProperties _
```

```

("last save time") & vbCr & _
"Ersteller der Mappe: " & _
ActiveWorkbook.BuiltinDocumentProperties _
("author") & vbCr & "Pfad: " & ActiveWorkbook.FullName
End With

End Sub

```

#### Listing 6.42 Mehrzeilige Fußzeile erstellen (aktive Tabelle)

Im ersten Schritt des Makros wird die Anweisung `With` auf das Objekt `PageSetup` der aktiven Tabelle (`ActiveSheet`) angewendet, um den Code übersichtlicher zu machen und um Schreibarbeit zu sparen. Danach müssen Sie den Befehl `ActiveSheet.PageSetup` nicht in jeder Zeile wiederholen. Stattdessen genügt es, wenn Sie einen Punkt als erstes Zeichen vor die Eigenschaften setzen, die auf die Seiteneinrichtung angewendet werden sollen. Über die Eigenschaft `BottomMargin` wird der Abstand zum unteren Papierrand angegeben. Mit Hilfe der Eigenschaft `FooterMargin` wird der Abstand der Fußzeile ebenfalls vom unteren Papierrand eingestellt. Diese Abstände müssen Sie in der Einheit *Punkt* angeben. Ein Punkt entspricht dabei in etwa 0,35 mm. So entsprechen 56 Punkt ungefähr 2 cm, und 42 Punkt sind in etwa 1,5 cm.

Über die Eigenschaft `LeftFooter` legen Sie den Inhalt des linken Teils der Fußzeile fest. Standardmäßig ist die Fußzeile in drei Teile gegliedert: Der linke Teil wird durch die Eigenschaft `LeftFooter`, der mittlere Teil durch die Eigenschaft `CenterFooter` und der rechte Teil durch die Eigenschaft `RightFooter` repräsentiert. Insgesamt dürfen jedoch nicht mehr als 255 Zeichen in der Fußzeile stehen. Aus diesem Grund wird im vorgestellten Beispiel nur der linke Teil der Fußzeile befüllt, und die restlichen Teile bleiben leer. Mit dem Steuerzeichen `"&8"` sorgen Sie dafür, dass der Schriftgrad auf 8 heruntergestellt wird, um Platz zu sparen. Alle folgenden Informationen werden danach in der Schriftgröße 8 (Standard ist 10) in die Fußzeile geschrieben.

In der ersten Zeile wird ein horizontaler Trennstrich eingefügt. Dazu wird die Tabellenfunktion `WIEDERHOLEN` eingesetzt, die in VBA über die Anweisung `WorksheetFunction.Rept` angesprochen wird. Dieser Funktion übergeben Sie im ersten Argument das Zeichen (hier der Unterstrich), das wiederholt werden soll. Im zweiten Argument geben Sie die Anzahl der Wiederholungen an. Immer im Hinblick darauf, dass insgesamt nur 255 Zeichen in der Fußzeile/Kopfzeile verwendet werden dürfen, kann dieses zweite Argument je nach sonstigem Füllgehalt der Fußzeile höher beziehungsweise niedriger eingestellt werden. Über die Konstante `VbCr` werden die folgenden Informationen jeweils in der nächsten Zeile der Fußzeile ausgegeben. Unter anderem wird das Erstellungsdatum der aktiven Arbeitsmappe mit Hilfe der Dokumenteigenschaft »Erstellt am« ermittelt. Um diese Dokumenteigenschaft aus der Arbeitsmappe per VBA zu ermitteln, setzen Sie das Auflistungsobjekt `BuiltinDocu-`

`mentProperties` ein, dem Sie die gewünschte Dokumenteigenschaft als Text übergeben. Um das Erstellungsdatum einer Arbeitsmappe zu ermitteln, übergeben Sie den Text `"Creation date"`. Das Datum der letzten Änderung ermitteln Sie über den Text `"last save time"`. Über den Text `"author"` fragen Sie den Ersteller der Arbeitsmappe ab. In der letzten Zeile der Fußzeile werden der komplette Pfad und der Dateiname mit Hilfe der Eigenschaft `FullName` ermittelt.

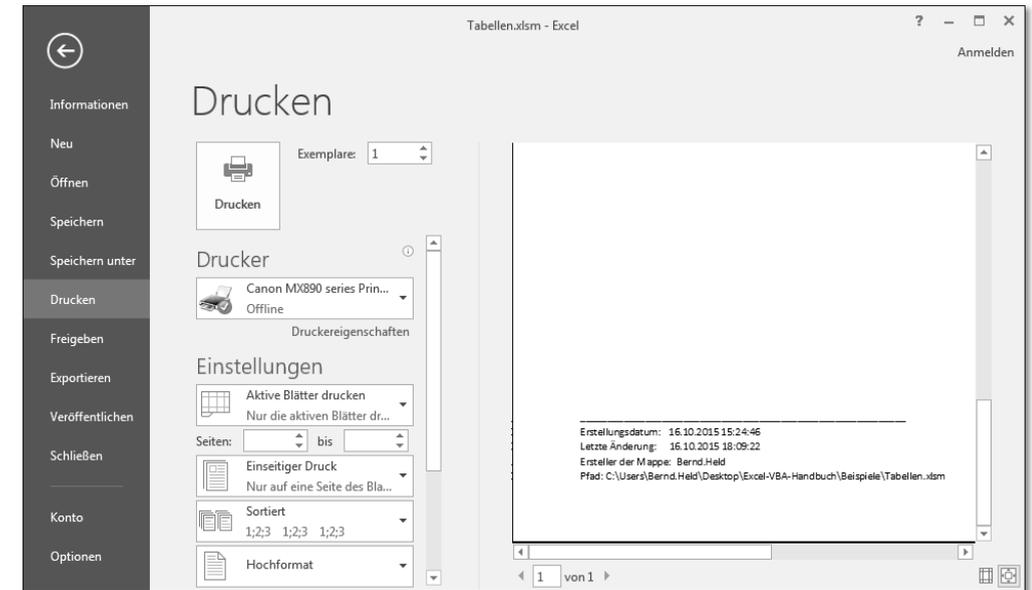


Abbildung 6.7 Eine mehrzeilige Fußzeile einfügen

#### Kopf- und Fußzeileneinträge leeren

Das Makro aus Listing 6.43 entfernt alle Einträge aus der Kopf- und Fußzeile, sogar das vorher eingefügte Logo.

```

Sub KopffussLeeren()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets

        With wksBlatt.PageSetup
            .LeftHeader = ""
            .CenterHeader = ""
            .RightHeader = ""
            .LeftFooter = ""
            .CenterFooter = ""
            .RightFooter = ""
        End With
    End With

```

```
Next wksBlatt
```

```
End Sub
```

#### Listing 6.43 Alle Inhalte der Kopf- und Fußzeile leeren

In einer Schleife des Typs `For Each ... Next` arbeiten Sie alle Tabellen der Arbeitsmappe ab. Über das Objekt `PageSetup` haben Sie Zugriff auf alle Kopf- und Fußzeilen der jeweiligen Tabelle. Weisen Sie den Eigenschaften jeweils eine leere Zeichenfolge zu, um die Inhalte der Kopf- und Fußzeile zu entfernen.

### 6.8.6 Druckbereiche festlegen

Um Papier zu sparen, können Sie vor dem Drucken einen Druckbereich festlegen. Im ersten Beispiel setzen Sie einen Druckbereich, der der momentanen Markierung entspricht. Markieren Sie also den Bereich, den Sie drucken möchten, und starten Sie danach folgendes Makro:

```
Sub DruckbereichSetzen()
```

```
    ActiveSheet.PageSetup.PrintArea = Selection.Address
```

```
End Sub
```

#### Listing 6.44 Druckbereich in Tabelle festlegen

Mit der Eigenschaft `PrintArea` legen Sie den Druckbereich fest. Wenn Sie Ihren Druckbereich fix gestalten möchten, setzen Sie das Makro aus Listing 6.45 ein:

```
Sub DruckbereichFestlegen()
```

```
    Worksheets("Tabelle1").PageSetup.PrintArea = "$A$1:$E$80"
    'oder
    Tabelle1.PageSetup.PrintArea = "$A$1:$E$80"
```

```
End Sub
```

#### Listing 6.45 Druckbereich in Tabelle konstant festlegen

Eine weitere Variante ist, den Druckbereich nach dem verwendeten Bereich zu bestimmen. Dazu setzen Sie die Eigenschaft `CurrentRegion` ein. Diese Eigenschaft ermittelt, beginnend von einer Zelle, den umliegenden Bereich. Sobald eine Leerzeile bzw. Leerspalte kommt, wird der Bereich abgeschlossen.

```
Sub DruckbereichNachVerwendungFestlegen()
```

```
    With Tabelle1
```

```
        .PageSetup.PrintArea = .Range("A1").CurrentRegion.Address
```

```
    End With
```

```
End Sub
```

#### Listing 6.46 Druckbereich nach Verwendung festlegen

#### Tip

Um einen Druckbereich wieder aufzuheben, setzen Sie die Eigenschaft `PrintArea` auf den Wert `False` oder auf die leere Zeichenfolge `""`. Damit wird das gesamte Blatt wieder als Druckbereich festgelegt.

### 6.8.7 Das Tabellen Gitternetz ein- und ausschalten

Sollten Sie das Gitternetz einer Tabelle ausschalten wollen, dann können Sie hierfür die Eigenschaft `DisplayGridlines` verwenden.

Das Makro aus Listing 6.47 schaltet die Anzeige der Gitternetzlinien für die aktive Tabelle ein und aus.

```
Sub UmschaltenGitternetzEinAus()
```

```
    ActiveWindow.DisplayGridlines = _
    Not ActiveWindow.DisplayGridlines
```

```
End Sub
```

#### Listing 6.47 Die Gitternetzanzeige für Tabellen ein- und ausschalten

### 6.8.8 Zeilen- und Spaltenköpfe ein- und ausblenden

Die Sichtbarkeit der Zeilen- und Spaltenbeschriftung regeln Sie über die Eigenschaft `DisplayHeadings`. Das Makro aus Listing 6.48 blendet die Spalten- und Zeilenköpfe im Wechsel ein- und aus.

```
Sub SpaltenUndZeilenEinUndAusblenden()
```

```
    If ActiveWindow.DisplayHeadings = False Then
        ActiveWindow.DisplayHeadings = True
```

```

Else
    ActiveWindow.DisplayHeadings = False
End If

End Sub

```

**Listing 6.48** Spalten- und Zeilenköpfe ein- und ausschalten

### 6.8.9 Cursor einstellen auf Zelle A1

Bei der Lösung aus Listing 6.49 wird bei allen Tabellen der Arbeitsmappe der Cursor in Zelle A1 gesetzt. Gegebenenfalls wird auch nach oben gescrollt.

```

Sub CursorEinstellenA1()
    Dim wksBlatt As Worksheet

    For Each wksBlatt In ActiveWorkbook.Worksheets

        Application.GoTo Reference:=wksBlatt.Range("A1"), scroll:=True

    Next wksBlatt

End Sub

```

**Listing 6.49** Cursor in jeder Tabelle einheitlich positionieren

Über eine Schleife des Typs `For Each ... Next` werden alle Tabellen der Arbeitsmappe nacheinander verarbeitet. Innerhalb der Schleife wenden Sie die Methode `GoTo` an, um in der jeweiligen Tabelle den Cursor in Zelle A1 zu setzen. Über den Parameter `Scroll` legen Sie fest, ob Sie einen Bildlauf durchführen, also scrollen möchten.

## 6.9 Tabellenblätter drucken und PDF erstellen

Wenn Sie die Kopf- und Fußzeilen gesetzt und eventuell auch einen Druckbereich eingestellt haben, gehen Sie zum Thema Drucken über. Drucken können Sie entweder ein oder mehrere Tabellenblätter, die ganze Arbeitsmappe, einen Druckbereich oder eine Markierung. Je nach Wunsch schreiben Sie dazu Makros.

Seit der Excel-Version 2007 gibt es eine PDF-Schnittstelle, die Sie über die Methode `ExportAsFixedFormat` aufrufen.

### Hinweis

Sie finden alle folgenden Makros in der Datei *Tabellen.xlsm* im Modul `mdl_Drucken`.

```

Sub TabellenblattDrucken()

    Sheets("Tabelle1").PrintOut
    'oder
    Tabelle1.PrintOut

End Sub

```

**Listing 6.50** Eine bestimmte Tabelle drucken

### 6.9.1 Mehrere Kopien drucken

Mit Hilfe der Methode `PrintOut` drucken Sie aus. Möchten Sie gleich mehrere Kopien ausgeben, so stellen Sie beim Parameter `Copies` die gewünschte Anzahl der Kopien ein.

```

Sub TabellenblattDruckenMitKopie()

    Worksheets("Tabelle1").PrintOut Copies:=2

    'oder

    Tabelle1.PrintOut Copies:=2

End Sub

```

**Listing 6.51** Mehrere Kopien einer Tabelle drucken

### 6.9.2 Markierte Bereiche drucken

Beim folgenden Makro markieren Sie entweder vorher den Bereich, den Sie drucken möchten, mit der Maus, oder weisen den Bereich per VBA zu, zum Beispiel: `Range("A1:D10").Select`. Führen Sie dann das Makro aus Listing 6.52 aus:

```

Sub MarkierungDrucken()

    Selection.PrintOut Copies:=1, Collate:=True

End Sub

```

**Listing 6.52** Markierten Bereich drucken

### 6.9.3 Mehrere Tabellenblätter drucken

Möchten Sie mehrere Tabellenblätter drucken, so bilden Sie ein Array mit den gewünschten Tabellenblättern und schicken den Druckauftrag auf den Weg:

```
Sub MehrereTabellenblätterDrucken()

    Sheets(Array("Tabelle4", "Tabelle1", _
        "Tabelle2")).PrintOut

End Sub
```

**Listing 6.53** Mehrere Tabellenblätter drucken

#### 6.9.4 Tabelle als PDF ablegen

Bei der Lösung aus Listing 6.54 wird von TABELLE3 zunächst eine eigenständige Excel-Arbeitsmappe erstellt und danach ein PDF gezogen:

```
Sub ExportAlsPDFundxlsx()
    'Warnmeldungen temporär ausschalten
    Application.DisplayAlerts = False

    'Bildschirmaktualisierung temporär ausschalten
    Application.ScreenUpdating = False

    'prüfen, ob ein Unterordner Exporte verfügbar ist.
    If Dir(ThisWorkbook.Path & "\Exporte", vbDirectory) = "" Then
        MkDir ThisWorkbook.Path & "\Exporte"
    End If

    'Kopiert die Tabelle automatisch in eine neue Mappe
    '(Copy ohne weitere Argumente)
    Tabelle3.Copy

    'Export als xlsx
    ActiveWorkbook.SaveAs Filename:=ThisWorkbook.Path & _
        "\Exporte\" & Format(Date, "YYYY.MM.DD_") & ActiveSheet.Name & ".xlsx"

    'Export als PDF
    ActiveWorkbook.ExportAsFixedFormat Type:=xlTypePDF, _
        Filename:=ThisWorkbook.Path & "\Exporte\" & _
        Format(Date, "YYYY.MM.DD_") & ActiveSheet.Name & ".pdf"

    ActiveWorkbook.Close

    Application.DisplayAlerts = True
End Sub
```

```
Application.ScreenUpdating = True
```

```
End Sub
```

**Listing 6.54** Tabelle exportieren, ein PDF erstellen und Tabelle als Mappe ablegen

Schalten Sie zu Beginn des Makro aus Listing 6.54 die Rückfragen von Excel temporär aus, indem Sie der Eigenschaft `DisplayAlerts` den Wert `False` zuweisen. Auch die Bildschirmaktualisierung können Sie ausschalten. Setzen Sie dazu die Eigenschaft `ScreenUpdating` auf den Wert `False`.

Prüfen Sie als Nächstes, ob der gewünschte Speicherpfad für das PDF und die Excel-Mappe überhaupt zur Verfügung steht. Dazu setzen Sie die Funktion `Dir` ein. Sollte das Zielverzeichnis noch nicht vorhanden sein, dann wenden Sie die Anweisung `MkDir` an, um es zu erstellen.

Danach wenden Sie die Methode `Copy` an, um TABELLE3 aus der Mappe herauszukopieren und in einer neuen Arbeitsmappe anzubieten.

Wenden Sie dann die Methode `SaveAs` an, um die Mappe zu speichern. Ergänzen Sie dabei mit Hilfe der Funktionen `Format` und `Date` den Dateinamen um einen Datumsstempel.

Mit Hilfe der Methode `ExportAsFixedFormat` wandeln Sie diese neue Mappe, die nur aus einer Tabelle besteht, in ein PDF um.

Schließen Sie am Ende des Makros die aktive Arbeitsmappe, indem Sie die Methode `Close` einsetzen. Vergessen Sie nicht, die beiden Schalter, die Sie zu Beginn des Makros ausgeschaltet haben, wieder einzuschalten.

## 6.10 Tabelleninhaltsverzeichnis erstellen

Bei der nächsten Aufgabe aus Listing 6.55 wird mit Hilfe eines Makros ein Inhaltsverzeichnis erstellt. Dabei wird zu Beginn der Arbeitsmappe eine neue Tabelle eingefügt. Danach werden die Namen aller in der Mappe enthaltenen Tabellen untereinander eingefügt und verlinkt.

```
Sub TabellenVerzeichnisErstellenHyperlinks()
    Dim wkbMappe As Workbook
    Dim wksBlatt As Worksheet
    Dim intTab As Integer
    Dim intZeile As Integer

    Set wkbMappe = ActiveWorkbook
    Set wksBlatt = wkbMappe.Worksheets.Add(Before:=Worksheets(1))
    intZeile = 1
End Sub
```

## Auf einen Blick

1	Die Entwicklungsumgebung von Excel .....	27
2	Datentypen, Variablen und Konstanten .....	65
3	Die Sprachelemente von Excel-VBA .....	99
4	Zellen und Bereiche programmieren .....	159
5	Spalten und Zeilen programmieren .....	289
6	Tabellen und Diagramme programmieren .....	355
7	Arbeitsmappen und Dateien programmieren .....	429
8	Eigene Funktionen, reguläre Ausdrücke und API-Funktionen .....	491
9	Ereignisse programmieren .....	563
10	Die VBE-Programmierung .....	599
11	Dialoge, Meldungen und UserForms programmieren .....	619
12	Excel im Umfeld von Office programmieren .....	751
13	Datenfelder, ADO, Dictionaries und Collections programmieren .....	815
14	Die Programmierung der Excel-Oberfläche .....	875
15	Fehlerbehandlung, Tuning und der Schutz von VBA-Projekten .....	893
16	Typische Verarbeitungsaufgaben aus der Praxis .....	905

# Inhalt

Vorwort .....	23
<b>1 Die Entwicklungsumgebung von Excel</b> .....	<b>27</b>
<b>1.1 Excel auf die Programmierung vorbereiten</b> .....	<b>27</b>
1.1.1 Heruntersetzen der Sicherheitsstufe .....	27
1.1.2 Die Entwicklertools einbinden .....	28
<b>1.2 Die Bestandteile der Entwicklungsumgebung</b> .....	<b>30</b>
1.2.1 Der Projekt-Explorer und das Eigenschaften-Fenster .....	30
1.2.2 Das Direktfenster zum Testen verwenden .....	38
1.2.3 Das Überwachungsfenster .....	43
1.2.4 Das »Lokal«-Fenster .....	45
1.2.5 Die Symbolleiste »Bearbeiten« .....	46
1.2.6 Automatische Syntaxprüfung .....	51
1.2.7 Befehle in der nächsten Zeile fortsetzen .....	52
1.2.8 Automatische Anpassung der einzelnen Befehle .....	53
1.2.9 Schnelles Arbeiten über Tastenkombinationen .....	54
1.2.10 Der Objektkatalog .....	56
1.2.11 Der Makrorekorder .....	57
1.2.12 Sonstige wichtige Einstellungen in der Entwicklungsumgebung .....	62
1.2.13 Die Onlinehilfe .....	64
<b>2 Datentypen, Variablen und Konstanten</b> .....	<b>65</b>
<b>2.1 Der Einsatz von Variablen</b> .....	<b>65</b>
2.1.1 Regeln für die Syntax von Variablen .....	66
2.1.2 Variablen am Beginn vom Makro deklarieren .....	68
2.1.3 Die wichtigsten Variablentypen .....	68
2.1.4 Variablendeklarationen erzwingen .....	70
2.1.5 Noch kürzere Deklaration von Variablen .....	70
2.1.6 Die unterschiedlichen Variablentypen .....	71
2.1.7 Typische Beispiele für den Einsatz von Variablen .....	73
2.1.8 Die Objektvariablen .....	80
<b>2.2 Die Verwendung von Konstanten</b> .....	<b>84</b>
2.2.1 Typische Aufgaben für die Verwendung von Konstanten .....	85

<b>3</b>	<b>Die Sprachelemente von Excel-VBA</b>	99
<b>3.1</b>	<b>Bedingungen</b>	99
<b>3.2</b>	<b>Typische Aufgaben aus der Praxis</b>	100
3.2.1	Wert in einer Spalte suchen	100
3.2.2	Liegt die aktive Zelle in einem vorgegebenen Bereich?	102
3.2.3	Prüfung, ob eine bestimmte Datei verfügbar ist	103
3.2.4	Spalteninhalte direkt nach der Eingabe umsortieren	104
3.2.5	Spalten mit Wochenenden kennzeichnen	106
<b>3.3</b>	<b>Die Kurzform einer Bedingung</b>	108
3.3.1	Den Doppelklick auf eine Zelle abfangen	109
<b>3.4</b>	<b>Die Anweisung »Select Case« einsetzen</b>	110
3.4.1	Excel-Version abfragen	111
3.4.2	Zahlenwerte prüfen	112
3.4.3	Den Wochentag eines bestimmten Datums auslesen	114
3.4.4	Benotungen über einen Autotext durchführen	114
<b>3.5</b>	<b>Die »Enum«-Anweisung</b>	116
3.5.1	Umsatz klassifizieren mit »Enum«	117
<b>3.6</b>	<b>Schleifen erstellen und verstehen</b>	118
3.6.1	Die »For ... Next«-Schleife	119
3.6.2	Die »For Each ... Next«-Schleife	135
3.6.3	Die Schleife »Do Until ... Loop«	146
3.6.4	Die Schleife »Do While ... Loop«	149
<b>3.7</b>	<b>Sonstige Sprachelemente</b>	152
3.7.1	Die Anweisung »With«	152
<b>4</b>	<b>Zellen und Bereiche programmieren</b>	159
<b>4.1</b>	<b>Zahlenformat einstellen und/oder konvertieren</b>	159
4.1.1	Zahlenformate einstellen (Datum und Zahl)	159
4.1.2	Zahlenformate einstellen (Text)	161
4.1.3	Zahlenformate übertragen	163
4.1.4	Zellen mit Nullen auffüllen	163
4.1.5	Einheitliches Datumsformat einstellen	165
4.1.6	Unerwünschte führende und nachgestellte Leerzeichen entfernen	167
4.1.7	Korrektur nach fehlerhaftem Import von Daten	169

4.1.8	Die Position des Minuszeichens umstellen	171
4.1.9	Daten umschlüsseln	174
4.1.10	Einen eindeutigen Schlüssel aus mehreren Spalten basteln	175
<b>4.2</b>	<b>Zellen, Rahmen und Schriften formatieren</b>	178
4.2.1	Schriftart ermitteln	178
4.2.2	Schriftart ändern	178
4.2.3	Zelleninhalte löschen	181
4.2.4	Schriftfarbe teilweise ändern	182
4.2.5	Automatisch runden und formatieren	184
4.2.6	Zwei Bereiche miteinander vergleichen	186
4.2.7	Einen Bereich »mustern«	188
4.2.8	Einen Bereich einrahmen	190
4.2.9	Einen Bereich umrahmen	192
<b>4.3</b>	<b>Die bedingte Formatierung von Excel</b>	194
4.3.1	Eine bedingte Formatierung als Standard einstellen	194
4.3.2	Duplikate mit dem bedingten Format aufspüren	198
4.3.3	Wertgrößen über einen Datenbalken darstellen	199
4.3.4	Eine Farbskala definieren und einsetzen	201
4.3.5	Daten über Pfeilsymbole bewerten	202
4.3.6	Die Top-Werte in einem Bereich hervorheben	203
4.3.7	Die einzugebende Textlänge überwachen	205
4.3.8	Eine bedingte Formatierung mit Wertgrenzen ausstatten	206
4.3.9	Sparklines einsetzen	208
<b>4.4</b>	<b>Bereiche und Zellen benennen</b>	209
4.4.1	Einen Bereich benennen	209
4.4.2	Mehrere Zellen einzeln benennen	211
4.4.3	Konstante als Namen vergeben	212
4.4.4	Verwendete Namen auslesen	213
4.4.5	Versteckte Namen sichtbar machen	214
4.4.6	Einen geheimen Namen anlegen	215
4.4.7	Einen dynamischen Namen anlegen	215
4.4.8	Ein Zellen-Dropdown auf Basis einer benannten Liste anlegen	217
4.4.9	Eine Prüfung auf Namen vornehmen	218
4.4.10	Benannte Bereiche löschen	220
4.4.11	Benutzerdefinierte Listen erstellen	221
<b>4.5</b>	<b>Tabellenfunktionen einsetzen</b>	222
4.5.1	Bedingtes Zählen von Werten (ZÄHLENWENN)	223
4.5.2	Bedingtes Zählen von Werten bei mehreren Bedingungen (ZÄHLENWENN)	224
4.5.3	Bedingte Summierung von Umsätzen	226

4.5.4	Bedingte Summierung von Umsätzen mit mehreren Kriterien .....	227
4.5.5	Den SVERWEIS im Makro einsetzen .....	229
4.5.6	Extremwerte finden und kennzeichnen .....	231
4.5.7	Prüfen, ob ein bestimmter Bereich leer ist .....	234
4.5.8	Einen Bereich mit Zahlenwerten mit vorangestellten Nullen auffüllen .....	235
4.5.9	Die Top-3-Werte in einem Bereich aufspüren .....	236
4.5.10	Automatische Prüfung und Überwachung eines Budgets .....	238
4.5.11	Mussfelder in einer Tabelle überprüfen .....	240
4.5.12	Mittelwert über eine InputBox ermitteln .....	241
4.5.13	Werte oberhalb und unterhalb des Durchschnitts ermitteln und kennzeichnen .....	243
<b>4.6</b>	<b>Matrixformeln in der Programmierung nutzen .....</b>	<b>244</b>
4.6.1	Werte bedingt zählen .....	245
4.6.2	Werte bedingt summieren .....	246
4.6.3	Mittelwert bilden ohne Berücksichtigung von Nullen .....	247
4.6.4	Extremwerte bedingt bilden .....	248
4.6.5	Den am meisten genannten Wert ermitteln .....	248
<b>4.7</b>	<b>Texte und Zahlen manipulieren .....</b>	<b>249</b>
4.7.1	Texte finden und umstellen .....	249
4.7.2	Mehrere Spalten anhand von Trennzeichen splitten .....	252
4.7.3	Daten nach einem Datentransfer bereinigen .....	254
4.7.4	Zeichenfolge(n) aus Zellen entfernen .....	255
4.7.5	Buchstaben aus Zellen entfernen .....	257
4.7.6	Dateinamen aus Pfad extrahieren .....	259
4.7.7	Alle Formelzellen einer Tabelle schützen .....	260
<b>4.8</b>	<b>Gültigkeitsprüfung in Excel .....</b>	<b>261</b>
4.8.1	Gültigkeitskriterien erstellen .....	262
4.8.2	Zellen mit Gültigkeitsfunktion auswählen .....	264
4.8.3	Datumsgrenzen festlegen .....	265
4.8.4	Nur Werkzeuge für die Eingabe zulassen .....	267
4.8.5	Eine Gültigkeitsliste aus einem Datenfeld befüllen .....	268
4.8.6	Uhrzeiten mit einer Gültigkeitsüberprüfung regeln .....	270
4.8.7	Ein Zellen-Dropdown aus einer Konstanten befüllen .....	272
4.8.8	Einen Standardeintrag für Zellen-Dropdowns setzen .....	273
4.8.9	Gültigkeitskriterien löschen .....	274
<b>4.9</b>	<b>Kommentare in Excel einsetzen .....</b>	<b>274</b>
4.9.1	Kommentare einfügen .....	275
4.9.2	Kommentare im Direktfenster auslesen .....	276
4.9.3	Kommentare formatieren .....	277

4.9.4	Kommentare aus der aktiven Tabelle löschen .....	280
4.9.5	Alle Kommentare einer Arbeitsmappe löschen .....	281
4.9.6	Einen Kommentar einem Namen in einer Arbeitsmappe zuweisen .....	282
4.9.7	Den Autor von Kommentaren anpassen .....	283
4.9.8	Kommentarkennzeichnung ein- und ausschalten .....	284
4.9.9	Kommentare einer Arbeitsmappe in eine Textdatei schreiben .....	284
4.9.10	Kommentare vor Veränderung schützen .....	285
4.9.11	Ein Bild in einen Kommentar einfügen .....	286
<b>5</b>	<b>Spalten und Zeilen programmieren .....</b>	<b>289</b>
<b>5.1</b>	<b>Zeilen und Spalten ansprechen, ansteuern und anpassen .....</b>	<b>289</b>
5.1.1	Mehrere Zeilen auf dem Tabellenblatt markieren .....	290
5.1.2	Mehrere Spalten auf dem Tabellenblatt markieren .....	290
5.1.3	Zeilen und Spalten markieren .....	291
5.1.4	Markierte Spalten zählen .....	291
5.1.5	Letzte freie Zelle in Spalte anspringen .....	293
5.1.6	Anzahl der benutzten Spalten und Zeilen ermitteln .....	293
5.1.7	Zeilen und Spalten im umliegenden Bereich zählen .....	295
5.1.8	Zeilenhöhe und Spaltenbreite einstellen .....	296
<b>5.2</b>	<b>Zeilen einfügen und löschen .....</b>	<b>299</b>
5.2.1	Zeile einfügen .....	299
5.2.2	Mehrere Zeilen einfügen .....	299
5.2.3	Leere Zeilen dynamisch einfügen .....	300
5.2.4	Wirklich leere Zeilen löschen .....	302
5.2.5	Doppelte Sätze löschen .....	303
<b>5.3</b>	<b>Spalten einfügen, löschen und bereinigen .....</b>	<b>304</b>
5.3.1	Spalte einfügen .....	304
5.3.2	Mehrere Spalten einfügen .....	304
5.3.3	Spalte löschen .....	305
5.3.4	Mehrere Spalten löschen .....	305
<b>5.4</b>	<b>Zeilen ein- und ausblenden .....</b>	<b>306</b>
5.4.1	Leere Zeilen ausblenden .....	306
5.4.2	Alle Zellen einblenden .....	307
<b>5.5</b>	<b>Spalten ein- und ausblenden .....</b>	<b>307</b>
5.5.1	Bestimmte Spalten ausblenden .....	308
<b>5.6</b>	<b>Spalten und Zeilen formatieren .....</b>	<b>308</b>

<b>5.7</b>	<b>Daten sortieren</b> .....	310
5.7.1	Daten sortieren mit der klassischen Variante .....	310
5.7.2	Daten sortieren mit der modernen Methode .....	311
5.7.3	Daten sortieren nach Farbe der Zellen .....	314
5.7.4	Daten nach einer Überschrift spaltenweise sortieren .....	315
<b>5.8</b>	<b>Spalte(n) vergleichen</b> .....	315
5.8.1	Zelle mit Spalte vergleichen .....	316
5.8.2	Spalten über eine bedingte Formatierung miteinander vergleichen .....	317
<b>5.9</b>	<b>Zeilen filtern</b> .....	318
5.9.1	AutoFilter aktivieren und deaktivieren .....	319
5.9.2	Filterkriterien setzen .....	321
5.9.3	Nur Texte filtern .....	324
5.9.4	Daten filtern, die eine Zeichenfolge enthalten .....	325
5.9.5	Der Top-10-Filter .....	326
5.9.6	Filtern nach Zellenfarbe .....	328
5.9.7	Gefilterte Zeilen entfernen .....	329
5.9.8	Filterkriterien als Datenfeld übergeben .....	331
5.9.9	Alle gesetzten Filter sichtbarer machen .....	336
5.9.10	Wie lauten die Filterkriterien? .....	338
5.9.11	Doppelte Werte mit dem Spezialfilter ermitteln .....	339
5.9.12	Doppelte Werte mit dem »Dictionary«-Objekt entfernen .....	341
5.9.13	Daten über einen Kriterienbereich filtern .....	342
5.9.14	Mehrere Spalten über einen Kriterienbereich filtern .....	344
5.9.15	Wildcards im Spezialfilter einsetzen .....	345
5.9.16	Filtern von Umsätzen in einem vorgegebenen Zeitraum .....	347
5.9.17	Gefilterte Daten transferieren .....	349
<b>5.10</b>	<b>Zeilen über das Teilergebnis gruppieren</b> .....	351
<b>6</b>	<b>Tabellen und Diagramme programmieren</b> .....	355
<b>6.1</b>	<b>Tabellen einfügen</b> .....	355
<b>6.2</b>	<b>Tabellenblätter benennen</b> .....	356
6.2.1	Eine neue Mappe mit 12 Monatstabellen anlegen und benennen .....	356
6.2.2	Eine neue Mappe mit den nächste 14 Tagen anlegen .....	358
6.2.3	Tabelle einfügen und benennen kombinieren .....	359
<b>6.3</b>	<b>Tabellen markieren</b> .....	359

<b>6.4</b>	<b>Tabellenblätter gruppieren</b> .....	360
6.4.1	Mehrere Tabellen gruppieren .....	360
6.4.2	Alle Tabellen gruppieren .....	361
6.4.3	Gruppierte Tabellen übertragen .....	362
6.4.4	Gruppierte Tabellen ermitteln .....	362
<b>6.5</b>	<b>Tabellenblätter löschen</b> .....	362
6.5.1	Eine Tabelle löschen .....	363
6.5.2	Bestimmte Tabellen aus einer Mappe entfernen .....	364
6.5.3	Tabellen mit gefärbten Registerlaschen entfernen .....	365
6.5.4	Leere Tabellen aus Arbeitsmappen entfernen .....	366
<b>6.6</b>	<b>Tabellenblätter ein- und ausblenden</b> .....	367
6.6.1	Tabellenblätter sicher ausblenden .....	368
6.6.2	Tabellen je nach Status ein- oder ausblenden .....	368
6.6.3	Alle Tabellenblätter anzeigen .....	369
6.6.4	Alle Tabellen außer der aktiven Tabelle ausblenden .....	369
<b>6.7</b>	<b>Tabellenblätter schützen</b> .....	370
6.7.1	Tabellenschutz aufheben .....	371
6.7.2	Alle Tabellen einer Arbeitsmappe schützen .....	371
6.7.3	Weitere Schutzfunktionen ab Excel 2002 .....	372
6.7.4	Passwort – Einstellungsdialog mit verschlüsseltem Passwort aufrufen .....	373
<b>6.8</b>	<b>Tabellen einstellen</b> .....	374
6.8.1	Registerlaschen ein- und ausblenden .....	374
6.8.2	Tabellenansicht anpassen .....	375
6.8.3	Einen einheitliche Zoomeinstellung vornehmen .....	376
6.8.4	Tabellenblätter sortieren .....	376
6.8.5	Kopf- und Fußzeilen einrichten .....	377
6.8.6	Druckbereiche festlegen .....	386
6.8.7	Das Tabellen Gitternetz ein- und ausschalten .....	387
6.8.8	Zeilen- und Spaltenköpfe ein- und ausblenden .....	387
6.8.9	Cursor einstellen auf Zelle A1 .....	388
<b>6.9</b>	<b>Tabellenblätter drucken und PDF erstellen</b> .....	388
6.9.1	Mehrere Kopien drucken .....	389
6.9.2	Markierte Bereiche drucken .....	389
6.9.3	Mehrere Tabellenblätter drucken .....	389
6.9.4	Tabelle als PDF ablegen .....	390
<b>6.10</b>	<b>Tabelleninhaltsverzeichnis erstellen</b> .....	391
<b>6.11</b>	<b>Intelligente Tabellen</b> .....	393
6.11.1	Tabelle umwandeln .....	393

6.11.2	Tabelle um eine Spalte ergänzen .....	395
6.11.3	Tabelle um eine Zeile ergänzen .....	396
6.11.4	Tabelle filtern .....	397
6.11.5	Tabellen sortieren .....	399
6.11.6	Tabelle um Ergebniszeile erweitern .....	400
6.11.7	Tabelle entfernen .....	401
<b>6.12</b>	<b>Pivot-Tabellen erstellen</b> .....	401
6.12.1	Pivot-Tabellen aktualisieren .....	405
6.12.2	Eine einzelne Pivot-Tabelle aktualisieren .....	405
6.12.3	Mehrere Pivot-Tabellen auf einem Tabellenblatt aktualisieren .....	406
6.12.4	Alle Pivot-Tabellen in Arbeitsmappe aktualisieren .....	406
6.12.5	Pivot-Tabellen dynamisch erweitern .....	407
6.12.6	Pivot-Tabellen formatieren .....	408
6.12.7	Slicer einfügen und bedienen .....	409
<b>6.13</b>	<b>Diagramme programmieren</b> .....	411
6.13.1	Umsätze in einem Säulendiagramm darstellen .....	412
6.13.2	Tagesumsätze im Liniendiagramm anzeigen .....	417
6.13.3	Tagesgenaue Formatierung im Punktdiagramm .....	419
6.13.4	Diagramme als Grafiken speichern .....	421
6.13.5	Gewinn und Verlust in einem Säulendiagramm präsentieren .....	423
6.13.6	Linienstärke unabhängig von den Markierungssymbolen formatieren .....	424
6.13.7	Sparklines automatisch erstellen .....	426
<b>7</b>	<b>Arbeitsmappen und Dateien programmieren</b> .....	429
<b>7.1</b>	<b>Arbeitsmappen ansprechen</b> .....	429
<b>7.2</b>	<b>Arbeitsmappen anlegen</b> .....	430
7.2.1	Eine Arbeitsmappe auf Basis einer Dokumentvorlage erstellen .....	430
7.2.2	Arbeitsmappe mit x Tabellen anlegen .....	431
7.2.3	Mappe mit Wochentabellen anlegen .....	432
<b>7.3</b>	<b>Arbeitsmappen speichern</b> .....	433
7.3.1	Arbeitsmappe unter aktuellem Tagesdatum speichern .....	434
7.3.2	Alle Tabellen einer Mappe als eigenständige Mappen speichern .....	434
7.3.3	Mappe erstellen und »Speichern unter«-Dialog aufrufen .....	436
7.3.4	Individuellen Speichern-Dialog aufrufen .....	437
7.3.5	Kopie der aktuellen Mappe zur Laufzeit erstellen .....	438

<b>7.4</b>	<b>Arbeitsmappen öffnen</b> .....	439
7.4.1	Mehrere Arbeitsmappen öffnen .....	439
7.4.2	Die aktuelle Datei in einem Verzeichnis öffnen .....	441
7.4.3	Regelmäßig die Dateixistenz prüfen .....	442
7.4.4	Alle verknüpften Mappen automatisch öffnen .....	442
<b>7.5</b>	<b>Arbeitsmappen schließen</b> .....	444
7.5.1	Arbeitsmappe schließen – Änderungen akzeptieren .....	444
7.5.2	Alle Arbeitsmappen bis auf eine schließen .....	445
<b>7.6</b>	<b>Arbeitsmappe löschen</b> .....	446
7.6.1	Arbeitsmappe nach Verfallsdatum löschen .....	446
7.6.2	Alle Excel-Mappen in einem Verzeichnis löschen .....	447
7.6.3	Mappe löschen, die älter als 14 Tage ist .....	448
<b>7.7</b>	<b>Arbeitsmappen drucken</b> .....	449
7.7.1	Nur bestimmte Tabellen drucken .....	449
7.7.2	Alle Mappen eines Verzeichnisses drucken .....	450
7.7.3	Nur sichtbare Blätter ausdrucken .....	451
<b>7.8</b>	<b>Dokumenteigenschaften verarbeiten</b> .....	452
7.8.1	Dokumenteigenschaftsnamen abfragen .....	452
7.8.2	Letztes Speicherdatum abfragen .....	454
7.8.3	Erstelldatum herausfinden und manipulieren .....	454
7.8.4	Zugriffsdaten einer Arbeitsmappe ermitteln .....	455
7.8.5	Eigene Dokumenteigenschaften verwenden .....	456
<b>7.9</b>	<b>Arbeitsmappen und Verknüpfungen</b> .....	457
7.9.1	Verknüpfungen in Hyperlinks umwandeln .....	458
7.9.2	Verknüpfungen aus der Arbeitsmappe entfernen .....	459
7.9.3	Verknüpfungen ändern .....	461
7.9.4	Verknüpfungen aktualisieren .....	462
<b>7.10</b>	<b>Arbeitsmappe durchsuchen</b> .....	463
<b>7.11</b>	<b>Arbeitsmappen miteinander vergleichen</b> .....	464
<b>7.12</b>	<b>Arbeitsmappenübersicht erstellen</b> .....	467
<b>7.13</b>	<b>Textdateien importieren</b> .....	468
<b>7.14</b>	<b>Makros für das Dateimanagement</b> .....	480
7.14.1	Ein Jahresverzeichnis automatisch anlegen .....	480
7.14.2	Eine bestimmte Datei nach Rückfrage löschen .....	481
7.14.3	Einen Ordner archivieren .....	483
7.14.4	Eine bestimmte Datei kopieren .....	484
7.14.5	Ordner anlegen und entfernen .....	485
7.14.6	Verzeichnisstruktur in einer Tabelle anzeigen .....	487

## 8 Eigene Funktionen, reguläre Ausdrücke und API-Funktionen 491

<b>8.1 Benutzerdefinierte Funktionen</b>	491
8.1.1 Aktive Arbeitsmappe ermitteln	492
8.1.2 Aktives Tabellenblatt ermitteln	493
8.1.3 Ist eine Tabelle leer?	494
8.1.4 Ist eine Tabelle geschützt?	495
8.1.5 Befinden sich Daten in einer bestimmten Spalte?	495
8.1.6 Den letzten Wert einer Spalte ermitteln	496
8.1.7 Den letzten Wert einer Zeile ermitteln	496
8.1.8 Den aktiven Bearbeiter identifizieren	497
8.1.9 Funktion zum Umsetzen von Noten	497
8.1.10 Rangfolge als Text ausgeben	499
8.1.11 Enthält eine bestimmte Zelle ein Gültigkeitskriterium?	499
8.1.12 Enthält eine Zelle einen Kommentar?	500
8.1.13 Ist eine bestimmte Zelle verbunden?	501
8.1.14 Initialen aus Namen erstellen	501
8.1.15 Nur Zellen mit Fettdruck addieren	502
8.1.16 Mit Uhrzeiten rechnen	503
8.1.17 Erweitertes Runden durchführen	505
8.1.18 Schnelles Umrechnen von Geschwindigkeiten	506
8.1.19 Extremwerte berechnen	507
8.1.20 Erste Ziffer in einer Zelle ermitteln	508
8.1.21 Buchstaben aus Zellen entfernen	510
8.1.22 Anzahl der Ziffern einer Zelle ermitteln	511
8.1.23 Römische Zahlen in arabische umwandeln	512
8.1.24 Einen Kommentartext in eine Zelle holen	514
8.1.25 Angabe eines optionalen Parameters bei einer Funktion	514
8.1.26 Leerzeichen in einen String integrieren	515
<b>8.2 Modulare Funktionen schreiben</b>	516
8.2.1 Dateien in einem Verzeichnis zählen	516
8.2.2 Fehlerüberwachung umleiten	517
8.2.3 Prüfen, ob eine bestimmte Datei existiert	519
8.2.4 Prüfen, ob eine bestimmte Datei geöffnet ist	519
8.2.5 Prüfen, ob eine Datei gerade bearbeitet wird	520
8.2.6 Prüfen, ob ein bestimmter Name in der Arbeitsmappe verwendet wird	521
8.2.7 Dokumenteigenschaften einer Arbeitsmappe ermitteln	522
8.2.8 Den letzten Wert einer Spalte ermitteln	524
8.2.9 Grafikelemente in einem definierten Bereich löschen	525

8.2.10 Kalenderwoche nach DIN ermitteln	527
8.2.11 Unerwünschte Zeichen aus Zellen entfernen	528
<b>8.3 Funktionen verfügbar machen</b>	530
8.3.1 Speichern der Funktionen in der persönlichen Arbeitsmappe	530
8.3.2 Speichern der Funktionen in einem Add-In	531
8.3.3 Ein Add-In einbinden	532
<b>8.4 Mit regulären Ausdrücken programmieren</b>	532
8.4.1 Funktionen für die Arbeit mit regulären Ausdrücken erstellen	533
8.4.2 Bestimmte Zeichenfolgen in Texten finden	536
8.4.3 Spezielle Zeichen nutzen	537
8.4.4 Zeichenfolgen aus Texten extrahieren	539
8.4.5 Eine E-Mail-Adresse prüfen	541
8.4.6 Konten prüfen	543
8.4.7 Zahlen aus Texten extrahieren	544
<b>8.5 API-Funktionen einsetzen</b>	546
8.5.1 Ermittlung der Laufwerke am PC	546
8.5.2 Bedienung des CD-ROM-Laufwerks	549
8.5.3 Bildschirmauflösung ermitteln	550
8.5.4 Ist ein externes Programm gestartet?	550
8.5.5 Excel schlafen schicken	551
8.5.6 Verzeichnisbaum anzeigen und auswerten	551
8.5.7 Cursorposition in Pixel angeben	554
8.5.8 Sounds per API-Funktion ausgeben	554
8.5.9 PC piepsen lassen	555
8.5.10 Eine E-Mail erstellen	555
8.5.11 Eine UserForm bildschirmfüllend anzeigen	556
8.5.12 Eine UserForm unverrückbar machen	557
8.5.13 Ein Label einer UserForm mit einem Hyperlink ausstatten	558
8.5.14 Den Standarddrucker per API abfragen	559
8.5.15 Texte über MD5 verschlüsseln	560
<b>9 Ereignisse programmieren</b>	563

<b>9.1 Ereignisse für die Arbeitsmappe</b>	563
9.1.1 Allgemeine Vorgehensweise beim Erstellen von Arbeitsmappen-Ereignissen	564
9.1.2 Die wichtigsten Ereignisse für die Arbeitsmappe im Überblick	565
9.1.3 Zugriff beim Öffnen der Mappe festhalten (»Workbook_Open«)	567
9.1.4 Das Schließen der Arbeitsmappe bedingt verhindern (»Workbook_BeforeClose«)	572

9.1.5	Letztes Bearbeitungsdatum festhalten (»Workbook_BeforeSave«) ....	572
9.1.6	Die Lösung für das sparsame Drucken (»Workbook_BeforePrint«) ....	573
9.1.7	Einfügen von Blättern verhindern (»Workbook_NewSheet«) .....	574
<b>9.2</b>	<b>Ereignisse für das Tabellenblatt</b> .....	<b>575</b>
9.2.1	Allgemeine Vorgehensweise bei der Einstellung von Tabellereignissen .....	575
9.2.2	Die wichtigsten Ereignisse für Tabellen im Überblick .....	576
9.2.3	Passworteingabe beim Aktivieren einer Tabelle (»Worksheet_Activate«) .....	576
9.2.4	Vergleich von zwei Spalten (»Worksheet_Change«) .....	577
9.2.5	AutoTexte über Kürzel abrufen (»Worksheet_Change«) .....	579
9.2.6	Symbole nach der Eingabe verändern (»Worksheet_Change«) .....	580
9.2.7	Die Spaltensumme überwachen (»Worksheet_Change«) .....	582
9.2.8	Nur einmalige Eingabe zulassen (»Worksheet_Change«) .....	583
9.2.9	Die Eingabe von Dubletten verhindern (»Worksheet_Change«) .....	583
9.2.10	Eingabe verhindern (»Worksheet_SelectionChange«) .....	585
9.2.11	Markierung überwachen (»Worksheet_SelectionChange«) .....	586
9.2.12	Mausklicks überwachen (»Worksheet_BeforeRightClick«) .....	587
9.2.13	Die Aktualisierung von Pivot-Tabellen überwachen (»Worksheet_PivotTableUpdate«) .....	587
<b>9.3</b>	<b>Reaktion auf Tastendruck</b> .....	<b>588</b>
9.3.1	Texte einfügen .....	590
9.3.2	Blattsperre ohne Blattschutz erstellen .....	591
9.3.3	Nur Werte einfügen .....	592
<b>9.4</b>	<b>Zeitsteuerung in Excel</b> .....	<b>593</b>
9.4.1	Regelmäßig die Uhrzeit anzeigen .....	593
9.4.2	Die Zeit läuft ... .....	594
9.4.3	Zellen blinken lassen .....	595
9.4.4	Eingaben nach Ablauf einer Minute löschen .....	596
<b>10</b>	<b>Die VBE-Programmierung</b> .....	<b>599</b>
<b>10.1</b>	<b>Die VBE-Bibliothek einbinden</b> .....	<b>600</b>
10.1.1	Die VBE-Bibliothek deaktivieren .....	601
10.1.2	Weitere Bibliotheken einbinden .....	602
10.1.3	Objektbibliotheken deaktivieren .....	603
10.1.4	Informationen zu Objektbibliotheken ausgeben .....	604
10.1.5	VBE-Editor aufrufen .....	604
10.1.6	Das Direktfenster aufrufen .....	605

<b>10.2</b>	<b>Die VBE ein- und ausschalten</b> .....	<b>605</b>
10.2.1	Neue Module einfügen .....	606
10.2.2	Einzelne Module löschen .....	607
<b>10.3</b>	<b>Einzelnes Makro löschen</b> .....	<b>607</b>
<b>10.4</b>	<b>Alle Makros aus einer Arbeitsmappe entfernen</b> .....	<b>608</b>
10.4.1	Module mit Makros bestücken .....	609
10.4.2	Makro zeilenweise in ein Modul übertragen .....	610
10.4.3	Makros aus einer Textdatei in ein Modul überführen .....	611
10.4.4	Export von VBA-Modulen in Textdateien .....	612
<b>10.5</b>	<b>Identifikation von Komponenten</b> .....	<b>613</b>
<b>10.6</b>	<b>Ein bestimmtes Makro auskommentieren</b> .....	<b>614</b>
<b>10.7</b>	<b>Das Direktfenster löschen</b> .....	<b>615</b>
<b>10.8</b>	<b>Den Status des VBA-Projekt abfragen</b> .....	<b>615</b>
<b>10.9</b>	<b>Makros und Ereignisse dokumentieren</b> .....	<b>616</b>
<b>11</b>	<b>Dialoge, Meldungen und UserForms programmieren</b> .....	<b>619</b>
<b>11.1</b>	<b>»MsgBox«-Meldung</b> .....	<b>620</b>
11.1.1	Welche Schaltfläche wurde angeklickt? .....	622
11.1.2	Löschrückfrage .....	623
<b>11.2</b>	<b>Die »InputBox«-Eingabemaske</b> .....	<b>623</b>
11.2.1	Einen Suchbegriff über eine InputBox abfragen .....	624
11.2.2	Abfrage des Spaltenbuchstabens .....	626
<b>11.3</b>	<b>Integrierte Dialoge einsetzen</b> .....	<b>627</b>
11.3.1	Den »Öffnen«-Dialog aufrufen .....	628
11.3.2	Den Dialog »Optionen« aufrufen .....	631
<b>11.4</b>	<b>Eigene UserForms entwerfen</b> .....	<b>631</b>
11.4.1	UserForm einfügen .....	632
11.4.2	UserForm beschriften .....	633
11.4.3	UserForm aufrufen .....	634
11.4.4	Die verfügbaren Steuerelemente .....	634
11.4.5	Steuerelemente einfügen .....	635
11.4.6	Die wichtigsten Eigenschaften .....	636
11.4.7	Ereignisse einstellen .....	640

<b>11.5 Programmierung von Textfeldern</b> .....	640
11.5.1 Passwort über einen Dialog abfragen .....	641
11.5.2 Textfelder leeren .....	644
11.5.3 Textfelder kennzeichnen .....	647
11.5.4 Prüfung auf numerischen Inhalt .....	649
11.5.5 Länge eines Textfeldes prüfen .....	651
11.5.6 Prüfen und rechnen mit Textfeldern .....	652
11.5.7 Prüfen und widerrufen .....	657
11.5.8 Eine AutoAusfüllen-Funktion programmieren .....	659
11.5.9 Rechtschreibprüfung vornehmen .....	662
11.5.10 Daten über ein Textfeld suchen .....	664
<b>11.6 Programmierung von Listefeldern</b> .....	668
11.6.1 Listefeld mit Tabellen füllen .....	668
11.6.2 Listefeld mit Monaten füllen .....	671
11.6.3 Mehrspaltiges Listefeld mit Daten aus Tabelle füllen .....	673
11.6.4 Listefeld transponiert füllen .....	678
11.6.5 Listefelder im Duett .....	681
11.6.6 Listefeld und Textfelder im Zusammenspiel .....	685
<b>11.7 Programmierung von Kombinationsfeldlisten</b> .....	687
11.7.1 Dropdown mit Tagen füllen .....	687
11.7.2 Unikate Einträge im Dropdown anzeigen .....	689
11.7.3 Dropdowns synchronisieren .....	693
11.7.4 Dropdown und Listefeld im Duett .....	696
<b>11.8 Die Programmierung von Optionsschaltflächen</b> .....	699
11.8.1 Mehrwertsteuersatz als Option anwenden .....	699
11.8.2 Optionsfelder und Listefeld im Zusammenspiel .....	703
<b>11.9 Die Programmierung von Kontrollkästchen</b> .....	705
11.9.1 Kontrollkästchen über eine Tabelle speisen .....	706
11.9.2 Ansichtseinstellungen über Kontrollkästchen vornehmen .....	709
<b>11.10 Die Programmierung von Bildelementen</b> .....	713
11.10.1 Der eigene Bildbetrachter .....	713
<b>11.11 Die Programmierung sonstiger Steuerelemente</b> .....	717
11.11.1 Bilder in MultiPage laden .....	717
11.11.2 Umschaltfläche programmieren .....	719
11.11.3 Drehfeld programmieren .....	724
11.11.4 Die Programmierung des »ListView«-Steuerelements .....	727
11.11.5 Die Programmierung des »TreeView«-Steuerelements .....	732
11.11.6 Die Programmierung des »ProgressBar«-Steuerelements .....	736

<b>11.12 Das Verwaltungstool</b> .....	738
11.12.1 Die hinterlegte Datentabelle .....	739
11.12.2 Die beteiligten Steuerelemente .....	740
11.12.3 Vorbereitende Aufgaben .....	740
11.12.4 Daten suchen .....	742
11.12.5 Mit Klick auf das Listefeld die Textfelder ausfüllen .....	744
11.12.6 Den Dialog initialisieren .....	745
11.12.7 Datensatz löschen .....	746
11.12.8 Datensatz ändern .....	747
11.12.9 Neue Kundennummer ermitteln .....	748
11.12.10 Datensatz anlegen .....	748
<b>12 Excel im Umfeld von Office programmieren</b> .....	751
<b>12.1 Excel im Zusammenspiel mit PowerPoint</b> .....	751
12.1.1 Excel-Bereich nach PowerPoint exportieren .....	751
12.1.2 Bereich aus Excel in eine bestehende Präsentation einfügen .....	754
12.1.3 Excel-Bereich verknüpft in eine neue Präsentation integrieren .....	756
12.1.4 PowerPoint-Folie als Objekt in Excel einbinden .....	758
12.1.5 Diagrammobjekte in eine Präsentation exportieren .....	760
<b>12.2 Excel im Zusammenspiel mit Word</b> .....	762
12.2.1 Excel-Bereich in Dokument exportieren .....	763
12.2.2 Excel-Tabelle in ein leeres Dokument überführen .....	765
12.2.3 Markierten Bereich einer Excel-Tabelle in ein Dokument exportieren .....	769
12.2.4 Bereich als Grafik an eine bestimmte Stelle eines Dokuments einfügen .....	771
<b>12.3 Excel im Zusammenspiel mit Outlook</b> .....	773
12.3.1 Kontaktdaten aus Excel nach Outlook exportieren .....	774
12.3.2 Termine aus Excel in den Outlook-Kalender schieben .....	778
12.3.3 Aktive Tabelle aus Excel heraus versenden .....	782
12.3.4 Aktive Tabelle als Anhang aus Excel heraus versenden .....	783
12.3.5 Aktive Arbeitsmappe per E-Mail versenden .....	785
12.3.6 Alle Dokumente aus einem Verzeichnis per E-Mail versenden .....	788
<b>12.4 Excel im Zusammenspiel mit Access</b> .....	790
12.4.1 Toolfrage und Randbedingungen .....	791
12.4.2 Anforderungen an das Tool .....	791
12.4.3 Die Umsetzung der Kernfunktionen .....	791
12.4.4 Befüllung der UserForm mit den wichtigsten Daten .....	793

12.4.5	Suche nach Therapeut über das Kürzel/den Patientennamen .....	795
12.4.6	Suche nach Datum .....	801
12.4.7	Termine erfassen .....	803
12.4.8	Änderung von Terminen .....	805
12.4.9	Termine löschen .....	807
12.4.10	Felder löschen .....	808
<b>12.5</b>	<b>Excel im Zusammenspiel mit dem Internet Explorer .....</b>	<b>809</b>
12.5.1	Eine Internetseite aus Excel aufrufen .....	809
12.5.2	Texte übersetzen mit Google .....	810
12.5.3	Bild aus Internet laden und in UserForm anzeigen .....	812
12.5.4	Ein PDF aus dem Internet laden .....	814

## **13 Datenfelder, ADO, Dictionaries und Collections programmieren** 815

<b>13.1</b>	<b>Aufgaben mit Hilfe von ADO und SQL-Statements lösen .....</b>	<b>815</b>
13.1.1	Daten filtern und in einer anderen Tabelle ausgeben .....	816
13.1.2	Umsätze nach Datum verdichten .....	819
13.1.3	Umsätze nach Datum und Warengruppe verdichten .....	822
13.1.4	Daten aus einer Arbeitsmappe beziehen, ohne diese zu öffnen .....	825
13.1.5	Daten aus einer Tabelle löschen .....	828
13.1.6	Top-Werte ermitteln .....	830
13.1.7	Mehrere Tabellen zusammenfassen .....	832
13.1.8	Eine Unikatsliste bilden .....	834
13.1.9	Excel-Daten per ADO verändern .....	837
<b>13.2</b>	<b>Arbeiten mit Arrays .....</b>	<b>840</b>
13.2.1	Aktionen im Arbeitsspeicher ausführen lassen .....	841
13.2.2	Bestimmte Daten aus einer Tabelle löschen .....	844
13.2.3	Daten konvertieren .....	847
<b>13.3</b>	<b>Arbeiten mit dem »Dictionary«-Objekt .....</b>	<b>851</b>
13.3.1	Daten verdichten .....	852
13.3.2	Bedingte Summierung mit mehreren Kriterien .....	855
13.3.3	Eine Unikatsliste erstellen .....	859
13.3.4	Anzahl von Bestellungen ermitteln .....	861
13.3.5	Doppelte Daten in einem Bereich ermitteln .....	866
<b>13.4</b>	<b>Arbeiten mit Collections .....</b>	<b>868</b>
13.4.1	Unikate Einträge über eine Collection bilden .....	869
13.4.2	Eine Collection aus einer Tabelle befüllen .....	872

## **14 Die Programmierung der Excel-Oberfläche** 875

<b>14.1</b>	<b>Die Programmierung von Kontextmenüs .....</b>	<b>875</b>
14.1.1	Kontextmenü deaktivieren .....	876
14.1.2	Das Zellen-Kontextmenü erweitern .....	877
14.1.3	Kontextmenü aufbauen (dreistufig) .....	879
14.1.4	Kontextmenü zurücksetzen .....	881
<b>14.2</b>	<b>Die Ribbon-Programmierung .....</b>	<b>881</b>
14.2.1	Der Custom UI Editor .....	881
14.2.2	Weitere wichtige Quellen und Hilfen .....	884
14.2.3	Ribbon mit Schaltflächen erstellen .....	885
14.2.4	Ribbon mit ComboBox erstellen .....	887
14.2.5	Ribbon mit bereits verfügbaren Funktionen bestücken .....	889
14.2.6	Den Backstage-Bereich programmieren .....	890
14.2.7	Eine Galerie mit Fotos erstellen .....	892

## **15 Fehlerbehandlung, Tuning und der Schutz von VBA-Projekten** 893

<b>15.1</b>	<b>Kleinere Fehler beheben .....</b>	<b>893</b>
15.1.1	Stimmt die Syntax? .....	893
15.1.2	Ist die Variablendefinition gegeben? .....	894
15.1.3	Objekt vorhanden? .....	894
15.1.4	Methode, Eigenschaft verfügbar? .....	895
<b>15.2</b>	<b>Schwerwiegendere Fehler .....</b>	<b>895</b>
15.2.1	Fehler im Vorfeld erkennen und reagieren .....	895
15.2.2	Fehler ignorieren .....	896
15.2.3	Fehlerursache ermitteln .....	896
15.2.4	Die Funktion »IsError« .....	897
<b>15.3</b>	<b>Das Add-In MZ-Tools .....</b>	<b>898</b>
15.3.1	Zeilennummern automatisch einfügen .....	899
15.3.2	Eine Fehlerbehandlung mit den MZ-Tools hinzufügen .....	900
<b>15.4</b>	<b>Laufzeiten verkürzen .....</b>	<b>901</b>
15.4.1	Variablen und Konstanten einsetzen .....	901
15.4.2	Berechnung und Bildschirmaktualisierung ausschalten .....	902
15.4.3	Integrierte Tabellenfunktionen anwenden .....	903
<b>15.5</b>	<b>VBA-Projekte schützen .....</b>	<b>903</b>

<b>16 Typische Verarbeitungsaufgaben aus der Praxis</b>	905
<b>16.1 Daten übertragen</b>	905
<b>16.2 Daten im Batch verarbeiten</b>	908
<b>16.3 Daten verteilen</b>	910
16.3.1 Die Tabellen entfernen	911
16.3.2 Die Verteilung der Zeilen auf die Tabellen	912
16.3.3 Die Plausibilität prüfen	915
16.3.4 Der Export der Tabellen	917
<b>16.4 Berichtfilterseiten erstellen</b>	918
<b>16.5 Daten löschen</b>	921
16.5.1 Daten entfernen – Variante 1	922
16.5.2 Daten entfernen – Variante 2	922
16.5.3 Daten entfernen – Variante 3	923
<b>16.6 Daten kennzeichnen</b>	924
16.6.1 Doppelte Daten kennzeichnen (der Standard)	925
16.6.2 Doppelte Daten kennzeichnen (die Erweiterung)	927
16.6.3 Top-10-Werte aus einem Bereich ermitteln	929
Index	933

# Index

## A

- Access  
     *mit Excel* ..... 790  
 Activate ..... 376, 889  
 ActiveConnection ..... 818  
 ActiveControl ..... 649  
 ActiveSheet ..... 265, 384  
 ActiveWorkbook ..... 79, 214, 474, 493  
 Add ... 48, 52, 155, 214–216, 263, 315, 355, 403,  
     410, 430–432, 606, 729–730, 735, 769  
 AddChart ..... 414, 427  
 AddColorScale ..... 201  
 AddComment ..... 276, 286  
 addDataBar ..... 200  
 AddFromFile ..... 602  
 AddFromGuid ..... 601  
 AddIconSetCondition ..... 140, 157, 203  
 Add-In einbinden ..... 532  
 AddIns (Auflistung) ..... 602  
 AddItem ..... 670, 682–683, 688  
 AddNew ..... 805  
 Address ..... 82, 268, 276, 285, 525, 581  
 AddTop10 ..... 204  
 AddUniqueValues ..... 198, 927  
 ADO programmieren ..... 815  
 AdvancedFilter ..... 340, 343, 345,  
     347–348, 691, 698  
 Aktion wiederrufen ..... 659  
 Aktive Arbeitsmappe versenden ..... 785  
 Aktivierungsreihenfolge festlegen ..... 645,  
     654, 666  
 AllDayEvent ..... 780  
 AllowMultiSelect ..... 630, 909  
 Ampelfunktion erstellen ..... 156  
 And ..... 527  
 Anmeldenname abfragen ..... 39  
 Anordnung umkehren ..... 203  
 Ansicht umstellen ..... 711  
 Anzeige umstellen ..... 711  
 API-Funktion einsetzen ..... 546  
 Application ..... 46, 711  
 Applikation  
     *anzeigen* ..... 89  
     *beenden* ..... 90  
 Apply ..... 313, 400  
 Arbeitsmappe ..... 433  
     *abfragen* ..... 492  
     *aktive, versenden* ..... 785  
     *anlegen* ..... 430–431, 434  
     *ansprechen* ..... 429  
     *automatisch schließen* ..... 148  
     *drucken* ..... 449  
     *durchsuchen* ..... 463  
     *identifizieren* ..... 493  
     *löschen* ..... 446–447, 785  
     *Name abfragen* ..... 81, 430  
     *öffnen* ..... 79, 439, 465  
     *schließen* ..... 79, 143, 149, 444–445, 572  
     *speichern* ..... 143, 433–435  
     *vergleichen* ..... 464  
 Arbeitsmappen auslesen ..... 80  
 Arbeitsmappenname abfragen ..... 81, 430  
 Arbeitsmappenübersicht erstellen ..... 467  
 Arbeitsspeicher freigeben ..... 90  
 Areas ..... 293, 770  
 Arrange ..... 465  
 Array ..... 361  
     *drehen* ..... 861  
     *verwenden* ..... 840  
 Array (Funktion) ..... 359  
 Array → Datenfeld  
 Artikelnummer  
     *finden* ..... 668  
     *suchen* ..... 668  
 As ..... 80  
 Asc ..... 258  
 Ausgabe formatieren ..... 702  
 AutoAusfüllen-Funktion programmieren ..... 659  
 AutoFilter ..... 75, 319, 321, 325, 334–335  
     *aktivieren* ..... 319  
     *ausschalten* ..... 320  
 AutoFilterMode ..... 75, 319–320, 325  
 AutoFit ..... 152, 297–298, 454, 617, 769  
 AutoFormat ..... 408  
 Autor  
     *anpassen* ..... 283  
     *entfernen* ..... 280  
 AutoSize ..... 276  
 AutoText einfügen ..... 579  
 Average ..... 242, 244, 248  
 AVG ..... 821

**B**

BackColor ..... 637, 649, 652, 670, 672,  
676, 680, 689, 702, 705

Backstage-Bereich programmieren ..... 890

Balkenfarbe festlegen ..... 200

BarColor ..... 200

Batchverarbeitung von Daten ..... 908

Bearbeiter ermitteln ..... 497

Bearbeitungsleiste anzeigen ..... 711

Bedingte Formatierung  
  *anwenden* ..... 157  
  *einfügen* ..... 196  
  *einstellen* ..... 138, 140, 157, 194, 205, 318  
  *löschen* ..... 157, 196

Bedingte Summierung durchführen ..... 227

Bedingte Zählung durchführen ..... 224

Bedingtes Format → Bedingte Formatierung

Bedingung einsetzen ..... 99

Beep ..... 442

Befehl  
  *anpassen* ..... 53  
  *in der nächsten Zeile fortsetzen* ..... 52

BeforeClose ..... 33

BeforePrint ..... 33

BeforeSave ..... 33

BeginGroup ..... 878

Benotungen durchführen ..... 114

Benutzereingabe auswerten ..... 642

Berechnung  
  *ausschalten* ..... 41, 902  
  *einschalten* ..... 42

Bereich  
  *definieren* ..... 84, 207, 209  
  *drehen* ..... 679  
  *einfügen in PowerPoint* ..... 754  
  *einrahmen* ..... 190  
  *erweitern* ..... 137  
  *integrieren in PowerPoint* ..... 756  
  *markieren* ..... 291  
  *mustern* ..... 188  
  *prüfen* ..... 102, 234  
  *umrahmen* ..... 141–142, 192  
  *verbinden* ..... 83, 137, 291  
  *verschieben* ..... 164, 293  
  *zwei Bereiche vergleichen* ..... 186

Berichtsfilterseiten erstellen ..... 918

Beschriftungsfeld beschreiben ..... 643

Bestimmte Daten löschen ..... 844

Bestimmte Spalten entfernen ..... 133

Bezeichnungsfeld  
  *beschreiben* ..... 666  
  *füllen* ..... 666

Bibliothek  
  *deaktivieren* ..... 601  
  *einbinden* ..... 602  
  *einsehen* ..... 56

Bild  
  *einfügen* ..... 287  
  *laden* ..... 715, 719  
  *löschen* ..... 717

Bildanzeige festlegen ..... 715

Bildbetrachter programmieren ..... 713

Bildelement programmieren ..... 713

Bildergalerie erstellen ..... 892

Bildschirmaktualisierung  
  *ausschalten* ..... 334, 435, 902  
  *einschalten* ..... 336

Bildschirmauflösung ermitteln ..... 550

Bildschirmmeldung anzeigen ..... 620

Block auskommentieren ..... 51

Body ..... 570, 787

Bold ..... 146, 179, 185, 287, 670, 672, 676, 701

Boolean ..... 68

BorderAround ..... 48, 142, 193

Borders ..... 191, 193, 205, 425

BottomMargin ..... 384

Breite einstellen ..... 287

Bruttowert berechnen ..... 702

Buchstabe entfernen ..... 257, 510

Budget  
  *prüfen* ..... 238  
  *überwachen* ..... 238

BuiltinDocumentProperties ..... 385, 454, 572

ButtonName ..... 438, 630

Byte ..... 68

**C**

Calculation ..... 42

Call ..... 312, 920

Caption ..... 633, 638, 648, 686, 721, 795

Case Else ..... 113

CDate ..... 56, 166, 266, 447

CDbl ..... 656

CD-ROM-Laufwerk bedienen ..... 549

Cells ..... 95, 120, 191, 196, 296, 496

CenterFooterPicture ..... 381

CenterHeaderPicture ..... 381

Change ..... 640

ChangeLink ..... 461

Characters ..... 183, 278, 287

ChartObject ..... 413

ChartStyle ..... 414

ChartTitle ..... 414, 419

ChartType ..... 411, 413, 419

CheckBoxes ..... 729

CheckSpelling ..... 663

Clean ..... 878

Clear ..... 312, 809, 860

ClearComments ..... 281, 286

ClearContents ..... 77, 239, 334, 698, 731, 845

Close ..... 48, 79, 444–445, 518, 570, 643

Codename verwenden ..... 35

CodePanels (Auflistung) ..... 602

Collection  
  *befüllen* ..... 872  
  *programmieren* ..... 815  
  *verwenden* ..... 868

Color ..... 179, 181, 315

ColorIndex ..... 81, 86–87, 101–102, 120, 128,  
130, 137, 142, 179, 181–182, 187,  
205, 232, 316, 338, 366, 927

Column ..... 106–107, 239, 579,  
687, 743, 768, 800

ColumnCount ..... 675, 698, 705, 795

ColumnDifferences ..... 317

ColumnHeaders ..... 729

Columns ..... 289

ColumnWidth ..... 296, 698, 705, 795

ComboBox → Kombinationsfeldliste

CommandBars (\Column\ ) ..... 876

CommandBars (\Ply\ ) ..... 876

CommandBars (\Row\ ) ..... 876

CommandBars (\System\ ) ..... 876

CommandBars (\Toolbar List\ ) ..... 876

CommandBars (Auflistung) ..... 602

CommandBars(\Cell\ ) ..... 875

Comment ..... 145, 276, 281–282, 500, 514

Const ..... 272

ContactItem ..... 775

Controls ..... 646, 686

ControlTipText ..... 642, 652

ConvertFormula ..... 879

Copy ..... 60, 143, 350, 362, 391, 435–436,  
757, 765, 784, 818, 834, 914, 918

CopyFile ..... 485

CopyFromRecordset ..... 827, 839

CopyPicture ..... 753, 762, 773

Count ..... 61, 125, 211, 293, 295, 303, 309, 334, 356,  
361, 535, 675, 719, 762, 770, 780, 821, 871

CountA ..... 294, 303, 307, 367, 468, 495–496

Countdown erstellen ..... 594

CountIf ..... 129, 226, 249, 584

CountOfLines ..... 611

CreateFolder ..... 486

CreateItem ..... 569, 776, 780, 787

CreateObject ..... 89, 148, 455, 468, 484, 569,  
753, 755, 757, 765, 770, 787, 810–811,  
814, 818, 853, 857, 860, 868, 897

CSV-Datei öffnen ..... 146, 469

Currency ..... 69

CurrentRegion ..... 216, 295, 343, 350, 386, 403

Cursor einstellen ..... 388

CursorPosition abfragen ..... 554

CursorType ..... 818

Custom UI Editor verwenden ..... 881

CutCopyMode ..... 178, 758

**D**

DatabodyRange ..... 396

Date ..... 53, 69, 77, 114, 276, 455, 504, 594, 633, 795

Datei  
  *auslesen* ..... 150  
  *Existenz prüfen* ..... 103–104, 519  
  *kopieren* ..... 484  
  *löschen* ..... 422, 481  
  *Name extrahieren* ..... 259  
  *prüfen* ..... 482  
  *verschieben* ..... 484  
  *zählen* ..... 516

Dateiname extrahieren ..... 259

Daten  
  *abrufen* ..... 475  
  *Batchverarbeitung* ..... 908  
  *bereinigen* ..... 254  
  *bestimmte löschen* ..... 844  
  *doppelte ermitteln* ..... 866  
  *einfügen* ..... 58  
  *filtern* ..... 816  
  *kennzeichnen* ..... 924  
  *konvertieren* ..... 847  
  *kopieren* ..... 58  
  *löschen* ..... 828, 921  
  *sortieren* ..... 106, 310, 694  
  *suchen* ..... 664, 742  
  *suchen über Textfeld* ..... 664  
  *transferieren* ..... 349  
  *Typ überprüfen* ..... 650  
  *umschlüsseln* ..... 174  
  *verdichten* ..... 852  
  *verteilen* ..... 910

Datenbalken einfügen ..... 200  
 Datenfeld  
   *anzeigen* ..... 90  
   *ausgeben* ..... 93  
   *auslesen* ..... 90  
   *bilden* ..... 359  
   *einstellen* ..... 268  
   *erweitern* ..... 269  
   *programmieren* ..... 815  
 Datenfilter einblenden ..... 75  
 Datensatz  
   *ändern* ..... 747  
   *anlegen* ..... 748  
   *Anzahl ermitteln* ..... 821  
   *einfärben* ..... 86  
 Datenschnitt einfügen ..... 409  
 Datensuche ..... 664, 742  
   *über Textfeld* ..... 664  
 Datentyp  
   *prüfen* ..... 218, 298, 650  
   *Regeln* ..... 218  
 Datum  
   *abfragen* ..... 53  
   *auswerten* ..... 114  
   *formatieren* ..... 85, 154  
   *konvertieren* ..... 266  
   *suchen* ..... 801  
 Datumseingaben vereinheitlichen ..... 165  
 Datumsfilter einsetzen ..... 347  
 Datumsformat  
   *vereinheitlichen* ..... 165  
   *verschiedene* ..... 85  
 Datumsgrenze festlegen ..... 265  
 Debug.Print ..... 42, 92–93, 517, 845  
 Default ..... 642  
 Definition  
   *Eigenschaften* ..... 46  
   *Methoden* ..... 46  
   *Objekt* ..... 56  
 Delete ..... 48, 220–221, 302–303, 306,  
   362–363, 365–367, 410  
 DeleteFolder ..... 486  
 DeleteLines ..... 608  
 Description ..... 265  
 Diagramm  
   *exportieren* ..... 421–422  
   *Form festlegen* ..... 414  
   *Objekt einfügen* ..... 414, 419  
   *programmieren* ..... 411  
   *Titel festlegen* ..... 414, 419  
 Diagrammform festlegen ..... 414  
 Diagrammobjekt  
   *einfügen* ..... 414, 419  
   *exportieren* ..... 760  
 Diagrammtitel festlegen ..... 414, 419  
 Dialog  
   *anzeigen* ..... 438, 628  
   *programmieren* ..... 619  
 Dialogs ..... 436  
 Dictionary programmieren ..... 815  
 Dictionary-Objekt  
   *einsetzen* ..... 341  
   *verwenden* ..... 851  
 Dim ..... 68, 80, 122, 124–125  
 Dir ..... 104, 150–151, 391, 482, 517, 519, 719  
 Direktfenster  
   *aufrufen* ..... 38, 605  
   *löschen* ..... 615  
   *schreiben* ..... 92, 122  
   *verwenden* ..... 38  
 Direktsuche durchführen ..... 101, 224, 695  
 Display ..... 787  
 DisplayAlerts ..... 143, 365, 391, 434–435, 444,  
   574, 594, 673, 784, 912  
 DisplayCommentIndicator ..... 284  
 DisplayFormulaBar ..... 711  
 DisplayGridlines ..... 387  
 DisplayHeadings ..... 711  
 DisplayWorkbookTabs ..... 375  
 Do ..... 464  
 Do Until .. 92, 146, 148–149, 441, 776, 797, 803  
 Do While ..... 149–152, 812, 910  
 DoEvents ..... 149, 738  
 Dokument schließen ..... 765  
 Dokumenteigenschaft  
   *abfragen* ..... 385, 452  
   *anlegen* ..... 456  
   *auslesen* ..... 522  
   *löschen* ..... 457  
 Dokumenteigenschaften  
   *eigene erstellen* ..... 456  
 Doppelklick abfangen ..... 109  
 Doppelte Daten ermitteln ..... 866  
 Doppelte Werte entfernen ..... 341  
 Double ..... 69  
 Drehfeld  
   *dimensionieren* ..... 726  
   *programmieren* ..... 715, 724  
 Dropdown  
   *auslesen* ..... 689  
   *Bedeutung der Symbole im* ..... 46  
   *füllen* ..... 687  
   *leeren* ..... 695

Dropdown → Kombinationsfeldliste  
 Dropdowns synchronisieren ..... 693  
 Dropdown-Symbol anzeigen ..... 155, 273  
 Druckbereich festlegen ..... 386  
 Dubletten verhindern ..... 583  
 DupeUnique ..... 198, 927  
 Duplikate  
   *kennzeichnen* ..... 128, 198  
   *löschen* ..... 303  
 Durchschnitt berechnen ..... 243  
**E**  
 Eigene Dokumenteigenschaften erstellen ..... 456  
 Eigenschaft  
   *BackColor* ..... 637  
   *ForeColor* ..... 637  
   *prüfen* ..... 895  
   *Value* ..... 638  
 Eigenschaften ..... 56  
   *anzeigen* ..... 46  
 Einfügen  
   *Daten* ..... 58  
 Eingabe  
   *löschen* ..... 596  
   *verhindern* ..... 585  
   *widerrufen* ..... 586, 657  
 Eingabelänge überwachen ..... 205  
 Eingabemeldung festlegen ..... 155  
 Eingaben  
   *automatisieren* ..... 660  
   *zusammensetzen* ..... 648  
 Einstellungen vornehmen ..... 62  
 Eintrag hinzufügen ..... 692  
 Einzelschrittmodus starten ..... 45  
 Einzug  
   *vergrößern* ..... 50  
   *verkleinern* ..... 50  
 Element ..... 135  
 E-Mail  
   *erstellen* ..... 555  
   *versenden* ..... 569  
 E-Mailadresse prüfen ..... 541  
 EnableEvents ..... 185, 239, 580, 583, 585  
 End ..... 100, 140, 170, 293, 351, 657, 749, 914  
 EntireColumn ..... 296, 304  
 EntireRow ..... 296, 331  
 Entwicklertools einbinden ..... 28  
 Entwicklungsumgebung aufrufen ..... 30  
 Enum ..... 116  
 Enum-Anweisung einsetzen ..... 116  
 Environ ..... 39, 276, 283, 444, 568, 633, 721  
 EOF ..... 479  
 Ereignis ..... 56  
   *dokumentieren* ..... 616  
   *einstellen* ..... 32, 640  
   *programmieren* ..... 563  
 Ereignissteuerung  
   *ausschalten* ..... 185, 239, 580  
   *einschalten* ..... 239, 583  
 Ergebniszeile hinzufügen ..... 400  
 Erinnerungsfunktion einschalten ..... 780  
 Err ..... 901  
 Err ..... 521  
 Err-Objekts ..... 896  
 ErrorMessage ..... 155, 266  
 ErrorTitle ..... 155, 266  
 Erstelldatum anpassen ..... 455  
 Evaluate ..... 176, 245  
 Excel  
   *beenden* ..... 445, 594–595  
   *einschlafen* ..... 551  
   *mit Access* ..... 790  
   *mit Internet-Explorer* ..... 809  
   *mit Outlook* ..... 773  
   *mit PowerPoint* ..... 751  
   *mit Word* ..... 762  
 Excel-Bereich  
   *einfügen in PowerPoint* ..... 754  
   *exportieren in Word* ..... 763  
   *integrieren in PowerPoint* ..... 756  
 Excel-Tabelle  
   *überführen in Word* ..... 765  
 Excel-Version abfragen ..... 111  
 Excel-Warnmeldungen ausschalten ... 143, 365  
 Execute ..... 535  
 Exists ..... 853, 861, 864  
 Exit ..... 658  
 Exit For ..... 364  
 Exit Function ..... 216, 509, 520, 684, 913  
 Exit Sub ..... 219, 363, 464, 568,  
   574, 650, 661, 896  
 Export ..... 422, 612  
 ExportAsFixedFormat ..... 388, 391  
 Extremwert  
   *bedingt bilden* ..... 248  
   *berechnen* ..... 507  
   *finden* ..... 231  
   *kennzeichnen* ..... 231

## F

- Farbe
  - abdunkeln* ..... 200
  - ansprechen* ..... 86
  - aufhellen* ..... 200
- Farbkonstante einsetzen ..... 181
- Farbpalette auslesen ..... 119
- Farbskala definieren ..... 201
- Farbsortierung vornehmen ..... 315
- Fehler
  - abfangen* ..... 230
  - beheben* ..... 893, 895
  - erkennen* ..... 895
  - ignorieren* ..... 896
  - schwerwiegender* ..... 895
- Fehlerbehandlung hinzufügen ..... 900
- Fehlermeldung festlegen ..... 155
- Fehlernummer auslesen ..... 521
- Fehlertitel definieren ..... 266
- Fehlerüberwachung umleiten ..... 517
- Fehlerursache ermitteln ..... 896
- Feld
  - löschen* ..... 808
  - verbinden* ..... 269
- Fettdruck anwenden ..... 146, 154, 660
- FileDateTime ..... 441, 449
- FileDialog ..... 437, 630
- FileExists ..... 485
- FileName ..... 381
- FileSystemObject ..... 455, 468, 483–485, 489, 523–524
- Fill ..... 279, 287, 581
- Filter ..... 422
  - sichtbar machen* ..... 336
- Filterkriterium
  - abfragen* ..... 338
  - angeben* ..... 75, 321
  - übergeben* ..... 331
- Filterung prüfen ..... 338
- Find ..... 101, 224, 464, 625, 668, 695, 907, 931
- FindNext ..... 464
- FindWindow ..... 551
- FIRST ..... 821
- Folder ..... 484
- FolderExists ..... 483
- Font ..... 146, 154, 178–179, 206, 215, 260, 287, 638, 654, 660, 676, 701, 927
- FontStyle ..... 179
- FooterMargin ..... 384
- For ... Next ..... 119, 123, 301, 682, 684, 699, 708, 721, 762, 780, 782, 795, 846, 853, 857, 870, 907
- For Append ..... 518
- For Each ... Next 81–82, 135, 137, 142–143, 145, 164, 182, 185, 187, 196, 214, 216, 219, 232, 234, 238, 244, 251, 260, 267, 271, 273, 276, 278, 280, 282–283, 285, 298, 338, 357, 364, 367, 369–370, 376, 386, 388, 406, 408, 421–422, 424, 445, 457, 464, 466, 468, 475, 580, 595, 617, 646, 657, 670, 709, 762, 770, 861, 868, 910, 913, 918, 931
- For Input ..... 479
- ForeColor ..... 279, 637, 642, 672, 676, 680, 689
- Format ..... 78, 433, 528, 702
  - löschen* ..... 205
- FormatConditions 140, 196, 198, 200, 205, 318
- FormatDateTime ..... 85
- Formatierung
  - bedingte* → *Bedingte Formatierung*
- Formatierungskriterien definieren ..... 157, 207
- Formel
  - abfragen* ..... 39
  - einsetzen* ..... 159
  - prüfen* ..... 255
  - schreiben* ..... 226, 229
  - simulieren* ..... 176
  - verbergen* ..... 260
- Formelzelle
  - aufspüren* ..... 96
  - schützen* ..... 260
- Formula ..... 40, 193, 226, 229, 461
- FormulaArray ..... 245–246, 248
- FormulaHidden ..... 261
- FormulaR1C1 ..... 177
- Foto
  - laden* ..... 715, 719
  - löschen* ..... 717
- FullName ..... 62, 81, 259, 385, 430, 493, 639, 818, 910
- FullRowSelect ..... 729
- Function ..... 498
- Funktion
  - aufrufen* ..... 364
  - beenden* ..... 913
  - IsError* ..... 897
  - verlassen* ..... 216, 364, 509, 684
- Fußzeile
  - einrichten* ..... 377
  - einstellen* ..... 61
  - leeren* ..... 385
  - mehrzeilige* ..... 383

## G

- Gesamtbereich bilden ..... 84
- Gesamtpreis berechnen ..... 655
- Geschwindigkeit umrechnen ..... 506
- GetDefaultFolder ..... 778
- GetDriveType ..... 547
- GetFolder ..... 484, 789
- GetNamespace ..... 782
- GetObject ..... 768, 770, 773, 897
- GetOpenFileName ..... 440, 629
- GetProfileString ..... 559
- GetSetting ..... 570
- Gitternetzlinien anzeigen ..... 387, 711, 729
- Global ..... 535
- Globale Variable verwenden ..... 68
- Google-Translator nutzen ..... 811
- Goto ..... 388
- Grafik
  - einfügen* ..... 380, 753
  - kopieren* ..... 753
- Grafikelement löschen ..... 525
- GridLines ..... 711, 729
- Gültigkeit prüfen ..... 499
- Gültigkeitsfunktion auslesen ..... 264
- Gültigkeitskriterium
  - einstellen* ..... 262
  - löschen* ..... 274
- Gültigkeitsliste erstellen ..... 154
- Gültigkeitsprüfung durchführen ..... 159, 261
- Gültigkeitsregel
  - einstellen* ..... 265, 267, 269–270
  - hinzufügen* ..... 263
  - löschen* ..... 155, 266, 271–272
  - prüfen* ..... 499

## H

- Haltepunkt setzen ..... 50
- HasFormula ..... 255, 260, 879
- HasLegend ..... 414, 419
- HasTitle ..... 414, 419
- Header ..... 313
- Height ..... 287, 639, 755, 762
- Helligkeitsgrad festlegen ..... 201
- Hidden ..... 307
- Hintergrundfarbe festlegen ..... 637
- Höchstwert festlegen ..... 204
- Höhe einstellen ..... 287
- HorizontalAlignment ..... 164
- Hyperlink einfügen ..... 558

## I

- IconCriteria ..... 140, 157
- IconSet ..... 157, 203
- If ..... 100, 102, 104, 107, 109, 128, 164, 246, 248, 293, 508, 878, 897
- IIf ..... 108–109
- Import
  - fehlerhaften korrigieren* ..... 169
- InCellDropdown ..... 155, 269, 273
- Index ..... 249, 358, 376
- Inhalt
  - numerischer, prüfen auf* ..... 649
- Initialen bilden ..... 501
- InitialFileName ..... 438, 630, 909
- InitialView ..... 909
- InnerText ..... 812
- InputBox ..... 76, 100–101, 223, 242, 283, 464, 623, 627
  - aufrufen* ..... 75
  - verwenden* ..... 623
  - vorbelegen* ..... 627
- Inputbox auswerten ..... 101
- InputMessage ..... 155, 266
- InputTitle ..... 155, 266
- Insert ..... 299, 301
- InsertAfter ..... 769
- InsertLines ..... 615
- InStr ..... 461, 743, 789
- Instr ..... 90, 219, 249, 257, 259, 280
- InStrRev ..... 259
- Int ..... 193
- Integer ..... 69
- Integrierte Dialoge einsetzen ..... 627
- Integrierte Tabellenfunktionen
  - anwenden* ..... 903
- IntelliSense ..... 756
- Interior ..... 84, 128, 137, 154, 187, 206, 927
- Internet Explorer
  - beenden* ..... 90
  - mit Excel* ..... 809
  - starten* ..... 89
- Internetabfrage umsetzen ..... 87
- Internetseite
  - aufrufen* ..... 809
  - auslesen* ..... 89
- Intersect ..... 103, 219, 587
- Introduction ..... 783
- IsArray ..... 108, 440
- IsDate ..... 107–108, 166
- IsEmpty ..... 108, 443, 460

- IsError ..... 108, 230, 897
- IsNull ..... 108
- IsNumeric ..... 108, 171, 185, 298, 503, 650, 659
- IsObject ..... 108
- Italic ..... 146, 179, 660
- items ..... 861
- J**
- Join ..... 269
- K**
- Kalenderwoche ermitteln ..... 527
- Kill ..... 422, 447, 449, 482, 785
- Kombinationsfeldliste
  - füllen ..... 687
  - programmieren ..... 687
  - Typ festlegen ..... 689
- Kommentar
  - abfragen ..... 276, 279, 285
  - einfügen ..... 50, 275–276
  - einsetzen ..... 274
  - exportieren ..... 284
  - färben ..... 279
  - formatieren ..... 277
  - Größe anpassen ..... 278
  - hinterlassen ..... 34
  - hinzufügen ..... 282
  - löschen ..... 280–281, 286
  - markieren ..... 281
  - nachformatieren ..... 144
  - prüfen ..... 500
  - schützen ..... 285
  - Text festlegen ..... 282
  - Text formatieren ..... 146
  - unterstreichen ..... 146
  - verarbeiten ..... 159
- Kommentarfenster
  - anpassen ..... 276
  - vergrößern ..... 276
- Kommentargröße anpassen ..... 278
- Kommentarkennzeichnung
  - ausschalten ..... 284
  - einschalten ..... 284
- Kommentartext
  - erfassen ..... 276
  - festlegen ..... 282
  - formatieren ..... 146
  - unterstreichen ..... 146
- Komponenten identifizieren ..... 613
- Konstante
  - anzeigen ..... 48
  - einsetzen ..... 901
  - verwenden ..... 84
- Kontakt
  - anlegen ..... 776
  - Daten auslesen ..... 774
  - Daten exportieren ..... 774
  - speichern ..... 776
- Kontaktdaten
  - auslesen ..... 774
  - exportieren ..... 774
- Kontextmenü
  - deaktivieren ..... 876
  - programmieren ..... 875
  - zurücksetzen ..... 881
- Konto prüfen ..... 543
- Kontrollkästchen
  - aktivieren ..... 708
  - beschriften ..... 708
  - deaktivieren ..... 708
  - programmieren ..... 705
  - verknüpfen ..... 706
  - vorbelegen ..... 709
- Kopf- und Fußzeile einstellen ..... 61
- Kopfzeile leeren ..... 385
- Kopfzeileneinträge leeren ..... 385
- Kopie erstellen ..... 438
- Kopieren
  - Daten ..... 58
- Kursivdruck anwenden ..... 146, 660
- Kurzform
  - Variablendeklaration ..... 70
- L**
- Large ..... 236
- LAST ..... 821
- Laufwerk ermitteln ..... 546
- Laufzeit
  - messen ..... 42
  - verkürzen ..... 901
- LBound ..... 440, 850
- Leere Zeilen entfernen ..... 130
- Leere Zellen ansprechen ..... 136
- Leerzeichen
  - einfügen ..... 515
  - entfernen ..... 167
  - führende entfernen ..... 167
  - nachgestellte entfernen ..... 167
  - unerwünschte entfernen ..... 167

- Leerzeile löschen ..... 302
- Left ..... 112, 251, 298, 662, 755, 762, 782, 850
- LeftFooter ..... 384
- LeftFooterPicture ..... 381
- Len ..... 76, 101, 164, 509, 519, 662, 850
- Like ..... 365, 512
- Line Input ..... 92, 479
- LineStyle ..... 49, 142, 191, 193
- Linienart festlegen ..... 142
- Linienbreite festlegen ..... 142
- LinkSources ..... 443, 458, 460
- List ..... 679
- ListColumns ..... 395
- Liste
  - benutzerdefinierte ..... 221
- Listenfeld
  - abfragen ..... 681
  - auslesen ..... 671
  - einfärben ..... 705
  - formatieren ..... 705
  - füllen ..... 668, 670–671, 682
  - füllen (mehrspaltig) ..... 673
  - füllen (transponiert) ..... 678
  - leeren ..... 684, 745, 809
  - löschen ..... 684
  - programmieren ..... 668
  - sortieren ..... 705
  - Spaltenanzahl festlegen ..... 698, 705
  - Spaltenbreite festlegen ..... 698, 705
  - Spaltentitel anzeigen ..... 705
  - Spaltenüberschriften festlegen ..... 729
  - verknüpfen ..... 705
- Listenfeldeintrag entfernen ..... 684
- ListIndex ..... 670, 672, 681, 689, 691, 698, 747, 800
- ListItems ..... 729, 731
- ListObject ..... 394
- ListRows ..... 396, 689
- ListSubItems ..... 730–731
- ListView
  - füllen ..... 730
  - programmieren ..... 727
  - Typ festlegen ..... 730
- LoadPicture ..... 715, 719
- Location ..... 414
- Locked ..... 260, 285, 583, 654, 666, 701, 726
- LockType ..... 819
- LOF ..... 93
- Lokal-Fenster
  - aufrufen ..... 253
  - einblenden ..... 45, 90
- Long ..... 69
- Löschrückfrage einholen ..... 623

**M**

- MacroType ..... 609
- Makro
  - aktivieren ..... 28
  - aufrufen ..... 312
  - aufzeichnen ..... 29
  - ausführen (schrittweise) ..... 59
  - auskommentieren ..... 614
  - beenden ..... 464
  - dokumentieren ..... 616
  - importieren ..... 610
  - löschen ..... 607–608
  - pausieren ..... 594
  - starten ..... 442, 611
  - verlassen ..... 219, 363
- Makrorekorder
  - starten ..... 57
  - verwenden ..... 57
- Mappe
  - anlegen ..... 357
  - schließen ..... 918
  - speichern ..... 391, 673
- Mappen anordnen ..... 465
- MarkerBackgroundColorIndex ..... 421
- MarkerForegroundColorIndex ..... 421
- Markierte Bereiche drucken ..... 389
- Markierung
  - abfragen ..... 293
  - überwachen ..... 586
- Match ..... 249
- MatchCase ..... 313
- MatchCollection ..... 540
- Matrixformel
  - nutzen ..... 244
  - schreiben ..... 246
  - verwenden ..... 245
- Mausklick überwachen ..... 587
- Mauszeiger
  - positionieren ..... 643
  - setzen ..... 646–647
  - voreinstellen ..... 646
- Max ..... 248–249, 507, 726, 738, 821
- Maximalwert berechnen ..... 507, 821
- mdl\_Formtieren ..... 194
- Mehrzeilige Fußzeile anlegen ..... 383
- Meldung
  - anzeigen ..... 49, 77
  - ausgeben ..... 103
  - programmieren ..... 619
- Meldungsfenster anzeigen ..... 53

Meldungstext definieren ..... 266  
 Menüband anpassen ..... 28  
 MergeCells ..... 501  
 Methode ..... 56  
   *anzeigen* ..... 46  
   *prüfen* ..... 895  
 Mid ..... 90, 251, 259, 511–512, 529  
 Min ..... 508, 726, 821  
 Minimalwert ermitteln ..... 821  
 Minuszeichen versetzen ..... 171  
 Mittelwert berechnen ..... 241–242, 247, 821  
 MkDir ..... 391, 422, 435, 481, 918  
 Mod ..... 309  
 Modul  
   *einfügen* ..... 41, 606  
   *exportieren* ..... 612  
   *löschen* ..... 607  
 Monatsname abfragen ..... 357, 672  
 Month ..... 907  
 MonthName ..... 357, 672  
 Move ..... 377  
 MoveFile ..... 484  
 MoveNext ..... 797, 803  
 MsgBox ... 49, 103, 118, 214, 241, 451, 620, 650  
 MsgBox-Meldung einsetzen ..... 620  
 MultiLine ..... 639, 652, 663, 714  
 MultiPage einsetzen ..... 717  
 Mussfeld überprüfen ..... 240  
 Muster festlegen ..... 188

## N

Name ..... 209–222  
   *anzeigen* ..... 214  
   *auslesen* ..... 213–214  
   *entfernen* ..... 220  
   *löschen* ..... 220  
   *prüfen* ..... 218  
   *verbergen* ..... 214  
 Namenskonventionen einhalten ..... 209  
 Namensprüfung durchführen ..... 218  
 Names ..... 212, 221  
 Navigate ..... 811, 814  
 Next lngZeile ..... 123  
 Note vergeben ..... 497  
 Now ..... 71, 78, 444, 593, 611, 845  
 Nullen auffüllen ..... 163, 235  
 Number ..... 521  
 NumberFormat ..... 154, 160, 162, 236  
 NumberFormatLocal ..... 161  
 Numerischer Inhalt  
   *prüfen* ..... 649

## O

Object ..... 69, 267  
 Objekt überprüfen ..... 894  
 Objektbibliothek deaktivieren ..... 603  
 Objektkatalog  
   *aufrufen* ..... 56, 80, 627  
   *einsehen* ..... 56  
 Objektvariable  
   *anlegen* ..... 668  
   *verwenden* ..... 80  
   *Workbook* ..... 80  
 Öffnen-Dialog einsetzen ..... 628  
 Offset ..... 164, 251, 257, 293, 617, 668  
 On ..... 338  
 On Error ..... 97, 242, 265, 282, 361, 363, 500, 514, 656, 895–896  
 On Error Goto Fehler ..... 96  
 On Error Resume Next ..... 220, 770, 896  
 OnAction ..... 878, 886  
 OnKey ..... 588, 590  
 Onlinehilfe benutzen ..... 64  
 OnTime ..... 442, 593, 596  
 Open 439, 441, 473, 479, 521, 755, 797, 818, 910  
 Openlinks ..... 444  
 OpenText ..... 473, 477  
 OpenTextFile ..... 148  
 Operator ..... 140, 157  
 Option Explicit ..... 62, 70, 894  
 Optional ..... 515  
 Optionsfeld vorbelegen ..... 701  
 Optionsschaltfläche programmieren ..... 699  
 Or ..... 527  
 ORDER BY ..... 821  
 Ordner  
   *archivieren* ..... 483  
   *erstellen* ..... 918  
 Orientation ..... 313, 400, 404, 768, 921  
 outerHTML ..... 89  
 OutLineFont ..... 180  
 Outlook mit Excel ..... 773

## P

PageSetup ..... 62, 377, 384, 386  
 Paramterinfo anzeigen ..... 49  
 Parent ..... 276, 736  
 PasswordChar ..... 642  
 Passwortabfrage  
   *über Dialog* ..... 641  
 Passwortheingabe abfragen ..... 576

Paste ..... 753, 765, 773  
 PasteSpecial ..... 178, 757  
 Path ..... 151, 430  
 Pattern ..... 188–189, 543  
 PDF erstellen ..... 390  
 Percent ..... 205  
 Pfad  
   *abfragen* ..... 81, 151, 430  
   *einstellen* ..... 714  
   *festlegen* ..... 714  
 Pfad- und Dateinamen abfragen ..... 62  
 PictureSizeMode ..... 715  
 PivotFields ..... 404  
 Pivot-Tabelle  
   *aktualisieren* ..... 405, 587  
   *erstellen* ..... 401  
   *erweitern* ..... 407  
   *formatieren* ..... 408  
 PivotTableWizard ..... 920  
 Plausibilität prüfen ..... 915  
 Point ..... 425  
 Position ..... 404  
 PowerPoint  
   *Excel-Bereich einfügen* ..... 754, 756  
   *mit Excel* ..... 751  
   *starten* ..... 753  
 Print ..... 285, 351, 518  
 PrintArea ..... 386–387  
 PrintOut ..... 389, 449, 451  
 PrintPreview ..... 379  
 Private ..... 73  
 ProcCountLines ..... 608, 615  
 ProcOfLine ..... 617  
 ProcStartLine ..... 608, 615  
 Programm finden ..... 550  
 ProgressBar-Steuerlement  
   programmieren ..... 736  
 Projekt  
   *schützen* ..... 903  
   *Status abfragen* ..... 615  
 Proper ..... 252  
 Protect ..... 73, 261, 370, 583  
 ProtectContents ..... 495  
 Public ..... 71  
 Punktdiagramm einfügen ..... 419

## Q

QueryTables ..... 475  
 QuickInfo anzeigen ..... 49, 261  
 Quit ..... 90, 445, 595, 765

## R

Rang bestimmen ..... 205  
 Range ..... 82, 84, 101, 142, 182, 191, 289, 668, 679  
 Range-Objekt verwenden ..... 159  
 Rangfolge ausgeben ..... 499  
 Rank ..... 204  
 ReadAll ..... 787  
 ReadLine ..... 148  
 Rechtschreibprüfung  
   *aufrufen* ..... 663  
   *vornehmen* ..... 662  
 RecordCount ..... 801  
 Recordset ..... 797  
 ReDim ..... 360–361, 846  
 ReDim Preserve ..... 269  
 Redundanzen ermitteln ..... 339  
 RefersTo ..... 214  
 RefersToRange ..... 522  
 Refresh ..... 475–476  
 RefreshStyle ..... 476  
 RefreshTable ..... 406  
 RegExp ..... 535  
 Registerkarte einfärben ..... 81  
 Registerlaschen  
   *ausblenden* ..... 374  
   *einblenden* ..... 374  
 Registrierungsdatenbank  
   *auslesen* ..... 570  
   *speichern* ..... 571  
 Registryeintrag schreiben ..... 567  
 Regulärer Ausdruck  
   *Übersicht* ..... 538  
   *verwenden* ..... 532  
 Reihenfolge festlegen ..... 404  
 ReminderMinutesBeforeStart ..... 780  
 Reminderset ..... 780  
 Remove ..... 601, 603, 607  
 RemoveDuplicates ..... 304  
 RemoveItem ..... 684  
 RemoveSubtotal ..... 353  
 Replace ..... 95, 173, 257, 280, 284, 850  
 Rept ..... 235, 384  
 Reset ..... 881  
 ResultRange ..... 475  
 ReverseOrder ..... 203  
 RGB 183, 315, 329, 642, 654, 666, 670, 680, 702  
 Ribbon  
   *bestücken* ..... 889  
   *erstellen* ..... 885

Ribbon (Forts.)  
*mit ComboBox* ..... 887  
*programmieren* ..... 881  
Right ..... 164, 173  
RightFooter ..... 384, 910  
RightFooterPicture ..... 381  
RightHeader ..... 381  
RightHeaderPicture ..... 381  
Round ..... 185, 505  
Row ..... 140, 171, 347, 657, 768, 857, 863, 914  
RowHeight ..... 296  
Rows ..... 61, 289, 293, 299, 334  
RowSource ..... 676, 691, 698, 705, 795  
Rückfrage  
*anzeigen* ..... 48  
*auswerten* ..... 451  
Run ..... 611, 810  
Runden  
*automatisches* ..... 184  
Runtime ..... 791

**S**

Satz hinzufügen ..... 692  
Save ..... 433, 595, 776, 780  
SaveAs ..... 433–435, 438–439, 784, 918  
SaveCopyAs ..... 77–78, 439  
SaveSetting ..... 571  
Schablone verwenden ..... 121  
Schaltflächenbeschriftung festlegen ..... 438  
SchemeColor ..... 581  
Schleife  
*aufsetzen* ..... 81, 185  
*erstellen* ..... 118  
*programmieren* ..... 42  
*Schritt für Schritt* ..... 122, 124–125  
*verstehen* ..... 118  
Schlüssel bilden ..... 175  
Schreibschutz abfragen ..... 495  
Schrift formatieren ..... 154  
Schriftart  
*ermitteln* ..... 178  
*festlegen* ..... 178, 276, 638  
Schriftfarbe festlegen ..... 181, 287, 642  
Schriftformatierung ..... 146  
Schriftgröße  
*anpassen* ..... 276  
*festlegen* ..... 287, 638, 654, 660, 726  
Schriftschnitt festlegen ..... 260, 287, 638, 660, 726  
Schriftweite festlegen ..... 726  
ScreenUpdating ..... 334, 336, 391, 435, 918  
ScrollArea ..... 591  
*einsetzen* ..... 33  
Seitenansicht ..... 379  
Seitenrand einstellen ..... 384  
Select ..... 60, 137, 265, 291, 360, 376  
Select Case ..... 110, 112–115, 134, 498, 509, 523, 580  
Selected ..... 411, 677, 683  
SelectedItems ..... 438, 736, 909  
SelectedSheets ..... 362  
Selection ..... 60, 82  
Send ..... 570  
Series ..... 424  
SeriesColor ..... 208  
Set 79, 84, 94, 120, 137, 142, 157, 163, 208, 227, 291, 458, 625, 675, 782, 873, 914  
SetFocus ..... 642–643, 646–647, 650, 666, 701  
SetRange ..... 313  
SetSourceData ..... 414, 419  
Shadow ..... 180  
Shape ..... 146, 278  
Sheets ..... 38, 370  
SheetsInNewWorkbook ..... 357, 359, 431, 677  
Shell ..... 810  
ShellExecute ..... 556  
Show ..... 438, 628, 630, 634, 644, 909  
ShowAllData ..... 344  
ShowError ..... 155, 266  
ShowIconOnly ..... 203  
ShowInput ..... 155, 266  
ShowPages ..... 921  
ShowTotals ..... 401  
Sicherheitskopie erstellen ..... 77  
Sicherheitsleck beheben ..... 644  
Sicherheitsstufe heruntersetzen ..... 27  
SignaturErmitteln ..... 787  
Single ..... 69  
Size ..... 179, 280, 287, 638, 642, 652, 654, 660, 670, 672, 676, 689, 726  
Slicer  
*abfragen* ..... 411  
*einsetzen* ..... 409–410  
*entfernen* ..... 410  
SmallChange ..... 726  
Sonstige Sprachelemente verwenden ..... 152  
Sort 106, 311–312, 315, 694, 705, 857, 861, 865  
SortFields ..... 312  
Sortieroptionen  
*entfernen* ..... 400  
*festlegen* ..... 400  
Sortierrichtung festlegen ..... 313

Sortierung ausführen ..... 313, 400  
SortOnValue ..... 315  
Sound ausgeben ..... 554  
SourceData ..... 403  
Spalte  
*ansprechen* ..... 289  
*Beite anpassen* ..... 296  
*Beschriftung anzeigen* ..... 387  
*bestimmte entfernen* ..... 133  
*Breite anpassen* ..... 152, 296  
*Breite automatisch anpassen* ..... 297  
*Buchstabe abfragen* ..... 626  
*einsetzen* ..... 304  
*ergänzen* ..... 395  
*Inhalt umsortieren* ..... 104  
*löschen* ..... 305  
*markieren* ..... 290  
*Nummer abfragen* ..... 107  
*Nummer ermitteln* ..... 239, 626  
*Summe überwachen* ..... 582  
*trennen* ..... 148  
*vergleichen* ..... 315, 577  
*Wert suchen* ..... 100  
*zählen* ..... 293–294, 309  
Spalten  
*Köpfe anzeigen* ..... 387  
Spaltenbeschriftung anzeigen ..... 387  
Spaltenbreite  
*anpassen* ..... 152, 296  
*einsetzen* ..... 296  
*einsetzen (automatisch)* ..... 297  
Spaltenbuchstabe abfragen ..... 626  
Spalteninhalt umsortieren ..... 104  
Spaltenköpfe anzeigen ..... 387  
Spaltennummer  
*abfragen* ..... 107  
*ermitteln* ..... 239, 626  
Spaltensumme überwachen ..... 582  
Spaltenwert suchen ..... 100  
Sparkline einfügen ..... 208, 426  
SparklineGroup ..... 208  
SpecialCells ..... 95, 97, 137, 265, 274, 281, 331, 350, 525  
SpecialEffect ..... 714  
Speicherdatum abfragen ..... 454  
Speichern-Dialog aufrufen ..... 437  
Spezialfilter  
*anwenden* ..... 339, 691  
*einsetzen* ..... 345  
Spezielle Zellen ermitteln ..... 95  
Split ..... 90, 253, 334, 479, 487  
Sprachelemente verwenden ..... 99  
SQL-Anweisung einlesen ..... 93  
Standardbrowser öffnen ..... 810  
Standarddrucker abfragen ..... 559  
Standardeintrag setzen ..... 273  
StatusBar ..... 143  
Statusleiste beschreiben ..... 143  
STDEV ..... 821  
Steuerelement  
*einsetzen* ..... 635  
*kennenlernen* ..... 634  
*Typ prüfen* ..... 646, 709  
Strikethrough ..... 180  
String ..... 69, 92  
strText ..... 80  
Style ..... 689  
Styles ..... 769  
Sub ..... 68  
Subfolders ..... 489  
Subject ..... 570, 776–777, 782  
Subscript ..... 180  
Subtotal ..... 336, 352  
Suchbegriff  
*abfragen* ..... 624  
*auswerten* ..... 662  
Suchmuster definieren ..... 543  
Sum ..... 239, 246, 583, 821, 903, 917  
Sumif ..... 227–229  
Summe bilden ..... 239, 821  
Summierung  
*bedingte* → *Bedingte Summierung von Umsätzen* ..... 226  
Superscript ..... 180  
SVERWEIS einsetzen ..... 229  
Symbol anpassen ..... 580  
Symbolbedeutung im Dropdown ..... 46  
Symbolleiste bearbeiten ..... 46  
Symbolsatz festlegen ..... 140, 157, 203  
Syntax überprüfen ..... 893  
Syntaxprüfung  
*automatisch* ..... 51

**T**

Tab ..... 366  
Tabelle  
*abfragen* ..... 493  
*aktivieren* ..... 376  
*Ansicht anpassen* ..... 375  
*ausblenden* ..... 31, 367, 568  
*ausblenden (sicher)* ..... 368

Tabelle (Forts.)	Tabellenname (Forts.)
<i>benennen</i> ..... 433	<i>prüfen</i> ..... 219
<i>drucken</i> ..... 388–389	<i>vergeben</i> ..... 356
<i>einblenden</i> ..... 31, 367, 369	Tabellenposition ermitteln ..... 358
<i>einfügen</i> ..... 355, 359, 392, 432	Tabellenreiter ansprechen ..... 366
<i>einstellen</i> ..... 374	Tabellenschutz
<i>entfernen</i> ..... 362, 366–367, 401, 574, 911	<i>aufheben</i> ..... 371
<i>entsperren</i> ..... 583	<i>einstellen</i> ..... 261
<i>exportieren</i> ..... 142, 390, 917	Tabellenstatus abfragen ..... 368, 452
<i>filtern</i> ..... 397	TabExists ..... 889
<i>gruppieren</i> ..... 360	TabIndex ..... 646, 654, 666
<i>Inhaltsverzeichnis erstellen</i> ..... 391	TableRange1 ..... 408
<i>kopieren</i> ..... 143, 362	Tables ..... 769
<i>leeren</i> ..... 77, 89	Tabulatorsprung erzeugen ..... 219
<i>markieren</i> ..... 359	Tagesdatum abrufen ..... 71
<i>Name vergeben</i> ..... 356	Tagesumsätze anzeigen ..... 417
<i>Position ermitteln</i> ..... 358	Tastenkombinationen verwenden ..... 54
<i>prüfen</i> ..... 670	Teilergebnis
<i>Reiter ansprechen</i> ..... 366	<i>entfernen</i> ..... 353
<i>Schutz aufheben</i> ..... 371	<i>verwenden</i> ..... 351
<i>schützen</i> ..... 73, 370, 583	Telefonnummer erfassen ..... 161
<i>sortieren</i> ..... 376, 399	Termin
<i>Status abfragen</i> ..... 368, 452	<i>ändern</i> ..... 805
<i>überwachen</i> ..... 238	<i>anlegen</i> ..... 778
<i>umwandeln</i> ..... 393	<i>erfassen</i> ..... 803
<i>verschieben</i> ..... 377	<i>löschen</i> ..... 807
<i>zusammenfassen</i> ..... 832	Text 276, 285, 298, 414, 500, 514, 663, 731, 736
Tabellenansicht anpassen ..... 375	<i>ausrichten</i> ..... 164
Tabellenblatt	<i>einfügen</i> ..... 590
<i>ansprechen</i> ..... 81	<i>filtern</i> ..... 324
<i>anzeigen</i> ..... 369	<i>finden</i> ..... 249
<i>ausblenden</i> ..... 367	<i>formatieren</i> ..... 236
<i>ausblenden (sicher)</i> ..... 368	<i>konvertieren</i> ..... 662
<i>benennen</i> ..... 356	<i>manipulieren</i> ..... 249
<i>drucken</i> ..... 388–389	<i>parsen</i> ..... 529
<i>einblenden</i> ..... 367	<i>suchen</i> ..... 625
<i>gruppieren</i> ..... 360	<i>übersetzen mit Google</i> ..... 810
<i>kopieren</i> ..... 362	<i>umwandeln</i> ..... 662
<i>löschen</i> ..... 362	<i>verschlüsseln</i> ..... 560
<i>schützen</i> ..... 370	TextAlign ..... 666, 701
<i>sortieren</i> ..... 376	Textausrichtung festlegen ..... 666
<i>zählen</i> ..... 356	Textdatei
Tabelleneintrag	<i>auslesen</i> ..... 92
<i>aufteilen</i> ..... 90	<i>einlesen</i> ..... 148, 468
<i>splitten</i> ..... 90	<i>einlesen (zeilenweise)</i> ..... 478
Tabellenfunktion	<i>Größe ermitteln</i> ..... 93
<i>einsetzen</i> ..... 159, 222	<i>importieren</i> ..... 468
<i>integrierte</i> ..... 903	<i>öffnen</i> ..... 92–93, 148, 285, 351, 472
Tabelleninhaltsverzeichnis erstellen ..... 391	<i>schließen</i> ..... 92, 351
Tabellenname	<i>schreiben</i> ..... 285
<i>ansprechen</i> ..... 34	<i>verarbeiten</i> ..... 91

Textfeld	<b>U</b>
<i>berechnen</i> ..... 653	Übereinstimmung prüfen ..... 535–536
<i>ein färben</i> ..... 648, 702	Übersicht
<i>einstellen</i> ..... 663	<i>Mappenereignisse</i> ..... 565
<i>füllen</i> ..... 663	<i>reguläre Ausdrücke</i> ..... 538
<i>initialisieren</i> ..... 646	<i>Tabellenergebnisse</i> ..... 576
<i>kennzeichnen</i> ..... 647	Überwachung hinzufügen ..... 43
<i>Länge prüfen</i> ..... 651	Überwachungsfenster einblenden ..... 43
<i>leeren</i> ..... 657, 745	UBound ..... 253, 441, 460, 716, 846, 850
<i>programmieren</i> ..... 640	UCase ..... 134, 175, 365, 513, 580, 662, 744
<i>prüfen</i> ..... 650, 654	Uhrzeit
<i>sperren</i> ..... 654, 666, 701	<i>abfragen</i> ..... 53, 71
Textfeldlänge prüfen ..... 651	<i>anzeigen</i> ..... 593
TextFileFixedColumnWidths ..... 477	Umliegenden Bereich ermitteln ..... 216
TextFileParseType ..... 477	Umsatz klassifizieren ..... 117
TextFilePromptOnRefresh ..... 475	Umsätze
TextFrame ..... 146, 278, 287	<i>summieren</i> ..... 226
TextToColumns ..... 148	<i>verdichten</i> ..... 819, 822
Textübersetzung mit Google ..... 810	Umschaltfläche programmieren ..... 719, 722
ThisWorkbook ..... 79, 81, 214, 429	Umwandlungsfunktion einsetzen ..... 656
ThisWorkbook.Path ..... 909	Underline ..... 146, 180
Tiefstwert festlegen ..... 204	Undo ..... 586
Time ..... 53, 271, 276	UndoAction ..... 659
Timer ..... 149	Unerwünschte Leerzeichen entfernen ..... 167
TimeValue ..... 593	Ungarische Notation einsetzen ..... 67
TintAndShade ..... 200–201	Unikate Einträge bilden ..... 869
Titel festlegen ..... 633	Unikatsliste
Title ..... 630, 909	<i>bilden</i> ..... 834
To ..... 113, 570, 783, 787	<i>erstellen</i> ..... 859
ToggleButton ..... 723	Union ..... 84, 137, 291, 833
Ton ausgeben ..... 442	Unlist ..... 401
Top ..... 755, 762	Unload Me ..... 643
Top-10-Filter anwenden ..... 326–327	Unprotect ..... 73, 260, 371, 583
TopBottom ..... 204	Unterbestand ermitteln ..... 321
Top-Wert ermitteln ..... 203, 236, 830, 929	Update ..... 805
Transparency ..... 279	UpdateLink ..... 462
Transpose ..... 342, 679, 853, 861	UsedRange ..... 122, 125, 182, 244, 293, 347, 495, 675
TreeView	UserForm
<i>auslesen</i> ..... 736	<i>anzeigen (bildschirmfüllend)</i> ..... 556
<i>auswerten</i> ..... 736	<i>aufrufen</i> ..... 634
<i>füllen</i> ..... 733	<i>beenden</i> ..... 643
<i>programmieren</i> ..... 732	<i>befüllen</i> ..... 793
<i>Zweig einfügen</i> ..... 734	<i>beschriften</i> ..... 633
TreeView-Zweig einfügen ..... 734	<i>einfügen</i> ..... 632
Trennzeichen	<i>entwerfen</i> ..... 631
<i>entfernen</i> ..... 94	<i>öffnen</i> ..... 644
<i>suchen</i> ..... 257	<i>programmieren</i> ..... 619
Trim ..... 169, 175, 625, 850	<i>schließen</i> ..... 644
Type ..... 140, 273, 499, 609	<i>starten</i> ..... 639
TypeName ..... 136, 646, 709, 746, 757, 809	
TypeParagraph ..... 768	
TypeText ..... 768	

UserForm\_Click ..... 633  
 UserForm\_Initialize .... 633, 640–641, 645, 662,  
 669, 672, 674, 679, 682, 688, 694,  
 700, 704, 707, 740  
 UserForm\_QueryClose ..... 644  
 UserName ..... 53, 276  
 UserPicture ..... 287

**V**

Validation ..... 155, 218, 269, 272, 499  
 Value .... 638, 643, 656–657, 663, 723, 741, 914  
 Variable  
   Definition kontrollieren ..... 894  
   Deklaration erzwingen ..... 70  
   deklarieren ..... 68  
   einsetzen ..... 65  
   globale → Globale Variable  
   Kurzform ..... 70  
   Name festlegen ..... 66  
   öffentliche Variable ..... 71  
   private ..... 73  
   statische ..... 71  
   Variablentypen ..... 68  
 Variablen einsetzen ..... 901  
 Variant ..... 69, 93  
 VBA  
   Projektstatus abfragen ..... 615  
 VBA-Projekt schützen ..... 903  
 VBE  
   ausschalten ..... 605  
   einschalten ..... 605  
 VBE-Bibliothek einbinden ..... 600  
 VBE-Editor aufrufen ..... 604  
 vbLf ..... 219, 276, 280, 590  
 VBProjects (Auflistung) ..... 602  
 vbTab ..... 219, 472  
 Verbindung herstellen ..... 475  
 Verkettungsoperator einsetzen ..... 648  
 Verknüpfung  
   aktualisieren ..... 462  
   ändern ..... 461  
   entfernen ..... 459  
   prüfen ..... 461  
   umwandeln ..... 458  
 Version ..... 111–112  
 VerticalAlignment ..... 590  
 Verzeichnis  
   anlegen ..... 66, 422, 435, 481  
   auslesen ..... 150  
   prüfen ..... 66, 103, 391, 422  
 Verzeichnisbaum anzeigen ..... 551  
 View ..... 730  
 Visible ..... 31, 210, 214, 367–370, 568, 755  
 VLookup ..... 230  
 Volatile ..... 498, 505

**W**

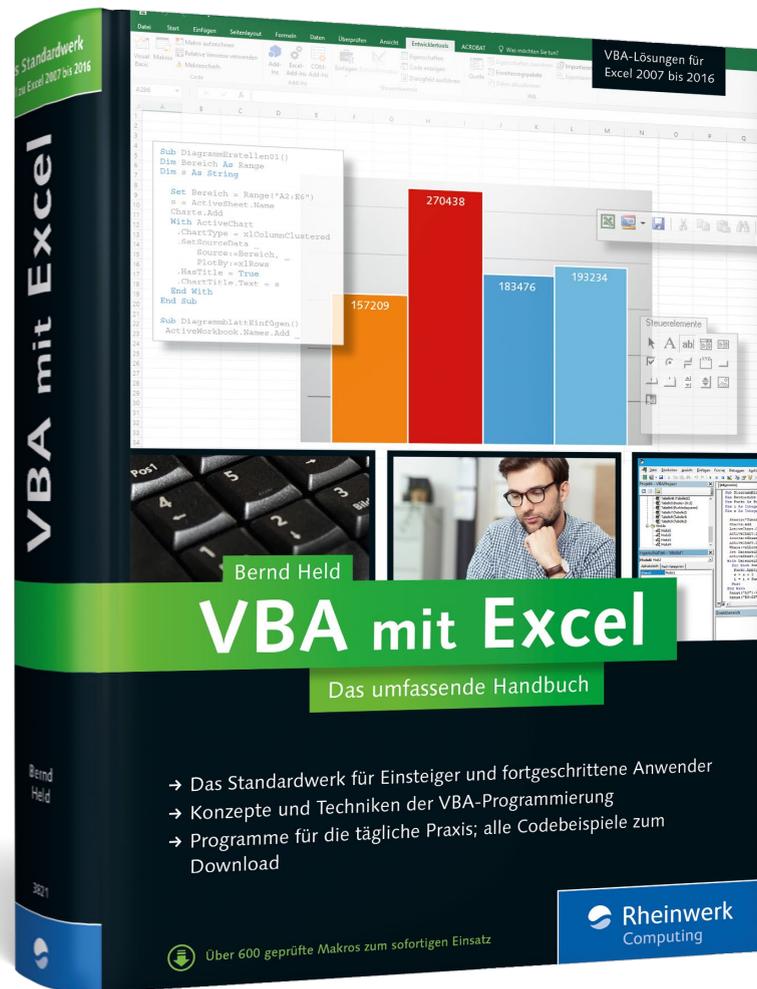
Wait ..... 594  
 Warnmeldung einschalten ..... 673  
 Webpage aufrufen ..... 809  
 Webseite aufrufen ..... 809  
 Webseite → Internetseite  
 Weekday ..... 107, 114, 133, 267–268, 689  
 WeekdayName ..... 688  
 Weight ..... 142, 425  
 Werktage prüfen ..... 267  
 Wert  
   aufspüren ..... 126  
   bedingt summieren ..... 246  
   bedingt zählen ..... 245  
   doppelte Werte entfernen ..... 341  
   einfügen ..... 592  
   runden ..... 185  
 Width ..... 287, 755, 762  
 Wildcard verwenden ..... 345  
 Wildcards einsetzen ..... 74  
 Windows (Auflistung) ..... 602  
 With ..... 84, 120, 122, 124–125,  
 140, 152–153, 379, 675  
 Wochenende  
   abfragen ..... 114  
   kennzeichnen ..... 106  
   prüfen ..... 106  
 Wochenendspalten einfärben ..... 107  
 Wochentabelle anlegen ..... 432  
 Wochentag  
   auslesen ..... 114  
   ermitteln ..... 107, 267  
   Name ermitteln ..... 688  
 Word  
   beenden ..... 765  
   Excel-Bereich exportieren in ..... 763  
   Excel-Tabelle überführen in ..... 765  
   mit Excel ..... 762  
   starten ..... 765  
 WordOffen ..... 773  
 Workbook ..... 81, 458, 673  
 Workbook\_Activate ..... 565  
 Workbook\_AddinInstall ..... 565  
 Workbook\_AddinUninstall ..... 565

Workbook\_AfterSave ..... 565  
 Workbook\_BeforeClose ..... 563, 572, 593, 876  
 Workbook\_BeforePrint ..... 565, 573  
 Workbook\_BeforeSave ..... 565, 572  
 Workbook\_Deactivate ..... 565  
 Workbook\_NewChart ..... 565  
 Workbook\_NewSheet ..... 565, 574  
 Workbook\_Open 32–33, 78, 149, 320, 563–564,  
 567, 570, 591, 593, 605  
 Workbook\_PivotTableCloseConnection .... 566  
 Workbook\_PivotTableOpenConnection .... 566  
 Workbook\_SheetActivate ..... 566  
 Workbook\_SheetBeforeDoubleClick ..... 566  
 Workbook\_SheetBeforeRightClick ..... 566  
 Workbook\_SheetCalculate ..... 566  
 Workbook\_SheetChange ..... 566  
 Workbook\_SheetDeactivate ..... 566  
 Workbook\_SheetFollowHyperlink ..... 566  
 Workbook\_SheetSelectionChange ..... 566  
 Workbook\_WindowActivate ..... 566  
 Workbook\_WindowDeactivate ..... 566  
 Workbook\_WindowResize ..... 566  
 Workbooks ..... 357, 429, 445, 673  
 Worksheet ..... 36, 52, 81, 94, 120, 355, 370  
 Worksheet\_Activate ..... 575–576, 596  
 Worksheet\_BeforeDoubleClick ..... 109,  
 575–576  
 Worksheet\_BeforeDoubleClick ..... 109  
 Worksheet\_BeforeRightClick ..... 575–576, 587  
 Worksheet\_Calculate ..... 576  
 Worksheet\_Change .... 104, 184, 238, 575–576,  
 579–580, 582–583, 585  
 Worksheet\_Deactivate ..... 576  
 Worksheet\_FollowHyperlink ..... 576  
 Worksheet\_PivotTableUpdate ..... 576, 587  
 Worksheet\_SelectionChange ..... 104, 109,  
 576, 585  
 WorksheetFunction ..... 226–227, 468  
 Worksheetfunction ..... 129  
 WrapText ..... 590, 879  
 wsf.SumIf ..... 227

**Z**

Zahl  
   extrahieren ..... 544  
   manipulieren ..... 249  
   runden ..... 505  
   umwandeln ..... 193, 512  
 Zahlenformat  
   einstellen ..... 159, 161  
   übertragen ..... 163  
 Zahlenwerte prüfen ..... 112  
 Zählung  
   bedingte → Bedingte Zählung  
 Zeichen  
   Anzahl ermitteln ..... 164  
   entfernen ..... 95, 254, 528  
   ersetzen ..... 95  
   vergleichen ..... 512  
 Zeichencode auslesen ..... 258  
 Zeichenfolge  
   ersetzen ..... 93, 257  
   extrahieren ..... 539  
   filtern ..... 326  
   finden ..... 90, 249, 508, 536  
 Zeichenlänge  
   ermitteln ..... 164, 662  
   messen ..... 662  
 Zeile  
   ansprechen ..... 289  
   einblenden ..... 306, 344  
   einfügen ..... 299, 396, 479  
   entfernen ..... 131, 329  
   filtern ..... 318  
   Höhe festlegen ..... 296  
   leere Zeilen entfernen ..... 130, 302  
   löschen ..... 123, 131, 299, 302, 331  
   markieren ..... 290  
   zählen ..... 211, 293–294, 334  
 Zeilen  
   Höhe anpassen ..... 296  
 Zeilennummern automatisch einfügen .... 899  
 Zeilenüberschrift anzeigen ..... 711  
 Zeilenumbruch definieren ..... 639  
 Zeilenvorschub  
   definieren ..... 276  
   einfügen ..... 219  
 Zeit  
   überwachen ..... 270  
   umrechnen ..... 503  
 Zelle  
   Adresse abfragen ..... 82, 268  
   ansprechen ..... 82, 120  
   Auswahl abfragen ..... 82  
   benennen ..... 209, 211  
   blinken lassen ..... 595  
   einblenden ..... 307  
   färben ..... 181, 238  
   formatieren ..... 153, 159, 215  
   Grafik einfügen ..... 208  
   Inhalt ausrichten ..... 590  
   Inhalt löschen ..... 181

Zelle (Forts.)	
<i>Kontextmenü erweitern</i> .....	877
<i>konvertieren</i> .....	159
<i>leere Zellen ansprechen</i> .....	136
<i>löschen</i> .....	239
<i>markieren</i> .....	137, 265
<i>schreiben</i> .....	226
<i>schützen</i> .....	260
<i>Umbruch einstellen</i> .....	590
<i>umliegenden Bereich ermitteln</i> .....	216
<i>verschieben</i> .....	293, 668
<i>zählen</i> .....	294
Zellenadresse abfragen .....	82, 268
Zellenauswahl abfragen .....	82
Zellen-Dropdown anlegen .....	217
Zellenfarbe filtern .....	328
Zellengrafik einfügen .....	208
Zelleninhalt	
<i>ausrichten</i> .....	590
<i>löschen</i> .....	181
Zellen-Kontextmenü erweitern .....	877
Zellenumbruch einstellen .....	590
Zielbereich festlegen .....	340
Zoom .....	375–376
Zoomeinstellung vornehmen .....	376
Zufallszahlen erzeugen .....	193
Zugriff dokumentieren .....	567–568
Zugriffsmodus festlegen .....	479
Zusammensetzen	
<i>Eingaben</i> .....	648
Zwei Bereiche vergleichen .....	186



**Bernd Held** ist von Haus aus gelernter Informatiker. Zu seinen Spezialgebieten zählen Excel, VBA-Programmierung, Access und allgemeine Office- und Tool-Themen. Er wurde von Microsoft mehrfach als MVP (Most Valuable Professional) für den Bereich Excel ausgezeichnet. Seit 2008 arbeitet er mit einem eigenen Team aus Experten zusammen, das Projekte und Schulungen durchführt, Unternehmen berät und Bücher sowie Fachartikel veröffentlicht.

Bernd Held

## VBA mit Excel – Das umfassende Handbuch

950 Seiten, gebunden, 2. Auflage 2015

49,90 Euro, ISBN 978-3-8362-3821-2

 [www.rheinwerk-verlag.de/3891](http://www.rheinwerk-verlag.de/3891)

*Wir hoffen sehr, dass Ihnen diese Leseprobe gefallen hat. Sie dürfen sie gerne empfehlen und weitergeben, allerdings nur vollständig mit allen Seiten. Bitte beachten Sie, dass der Funktionsumfang dieser Leseprobe sowie ihre Darstellung von der E-Book-Fassung des vorgestellten Buches abweichen können. Diese Leseprobe ist in all ihren Teilen urheberrechtlich geschützt. Alle Nutzungs- und Verwertungsrechte liegen beim Autor und beim Verlag.*

*Teilen Sie Ihre Leseerfahrung mit uns!*

