


# Workshop 6: VBA-Programmierung mit MS Excel

1 Standardspeicherort für Arbeitsmappen .....	1
2 Das Application-Objekt.....	2
3 Mit Arbeitsmappen arbeiten .....	4
3.1 Zugriff auf Arbeitsmappen .....	4
3.2 Arbeitsmappen anlegen .....	4
3.3 Arbeitsmappen speichern.....	6
3.4 Arbeitsmappen öffnen .....	6
3.5 Liste der geöffneten Arbeitsmappen beeinflussen .....	7
3.6 Mehrere Arbeitsmappen öffnen.....	7
3.7 Geöffnete Arbeitsmappen ermitteln .....	8
3.8 Arbeitsmappe ohne Rückfrage schließen.....	9
3.9 Arbeitsmappe schließen ohne zu speichern.....	9
3.10 Arbeitsmappe löschen.....	9
3.11 Arbeitsmappe schützen.....	11
4. Dokumenteneigenschaften .....	12
4.1 Dokumenteigenschaften auslesen .....	12
4.2 Dokumenteigenschaften anzeigen.....	13
4.3 Dokumenteigenschaften setzen.....	14
5 Externe Verknüpfungen ausgeben .....	14
6. Aufgaben .....	15
7. Lösungen.....	15

## 1 Standardspeicherort für Arbeitsmappen

Der Standardspeicherort von Arbeitsmappen kann in Excel wie folgt festgelegt werden:

- auf die Office-Schaltfläche  klicken,
- EXCEL-OPTIONEN auswählen,
- Rubrik SPEICHERN auswählen,
- den Standardspeicherort im dafür vorgesehenen Feld eingeben

Mit VBA kann der Standardspeicherort über das Direktfenster mit folgender Codezeile bestimmt werden:

```
Application.DefaultFilePath = "C:\MeinPfad\MeinOrdner"  
oder, etwas konkreter  
Application.DefaultFilePath = "C:\Eigene Dateien\Excel Dateien"
```

Für das Arbeiten mit Arbeitsmappen ist es vorteilhaft, dass der Standardspeicherort gesetzt ist. Probieren sie bitte folgende Prozedur aus:

```
Sub Speichern_in_Standardordner()  
    ' Zum Standarspeicherort wechseln
```

# Workshop 6: VBA-Programmierung mit MS Excel

```
ChDir Application.DefaultFilePath
' Dialogfeld 'Speichern unter' aufrufen
Application.Dialogs(xlDialogSaveAs).Show
End Sub
```

## 2 Das Application-Objekt

Mit dem Application-Objekt (quasi Excel selbst) kann nicht nur auf sogen. Auflistungen zugegriffen werden, sondern auch auf Objekte zur Programmsteuerung (z. B. integrierte Dialogfelder). Für verschiedene aktive Objekte, wie die aktive Arbeitsmappe, das aktive Tabellenblatt oder die aktuell selektierte Zelle, sind verkürzte Zugriffswege vorhanden.

Aufgabe der Prozedur `MappeErstellen` ist die Erstellung einer neuen Arbeitsmappe mit 52 Tabellenblättern. Die Tabellenblätter sollen `KW01`, ..., `KW52` heißen. Die Spaltenköpfe der neu erstellten Tabellenblätter sollen mit *Nord*, *Süd*, *Ost* und *West* belegt werden. Zum Beschriften der Spaltenköpfe in den neuen Tabellenblättern ruft die Prozedur jeweils die Unterprozedur `SpaltenKoepfeEinfügen` auf.

```
Sub MappeErstellen()
    Const conDateiname As String = "IchBinNeuHier"
    Const conKW        As String = "KW"
    Dim objWkb         As Workbook   ' Objektvariable für Arbeitsmappe
    Dim objSht         As Worksheet  ' Objektvariable für Arbeitsblatt
    Dim intWoche       As Integer    ' Schleifenzähler
    Dim strPfadname    As String     ' Vollständiger Pfadname
    Dim strExt         As String     ' Zusatz zum Dateinamen
    ' Bei Fehler ...
    On Error GoTo Fehler
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
    If Val(Application.Version) < 12 Then
        strExt = ".xls"      ' Excel 2003 und früher
    Else
        strExt = ".xlsx"    ' Excel 2007
    End If
    ' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateinzusatz
    strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
    If Dir(strPfadname) <> vbNullString Then
        ' Datei ist vorhanden.
        If MsgBox(Prompt:="Datei '" & strPfadname & "' ist bereits vorhanden!" & _
            vbNewLine & "Datei löschen?", _
            Buttons:=vbCritical + vbYesNo, Title:="Löschen?") = vbYes Then
            Kill strPfadname ' Datei löschen
        Else
            ' Datei nicht löschen
            GoTo Ausgang    ' Prozedur verlassen
        End If
    End If
    ' Neue Arbeitsmappe erzeugen mit der Methode Add
    Set objWkb = Workbooks.Add
    ' Neue Arbeitsmappe im Standardspeicherort speichern
    objWkb.SaveAs Filename:=strPfadname
    Application.DisplayAlerts = False ' Warnungen ausschalten
    ' Alle standardmäßig angelegten Tabellenblätter löschen
    ' außer dem aktiven Tabellenblatt
    For Each objSht In Worksheets
        If Not objSht Is ActiveSheet Then
            objSht.Delete
        End If
    End If
```

## Workshop 6: VBA-Programmierung mit MS Excel

```
Next objSht
Application.DisplayAlerts = True ' Warnungen wieder einschalten
' Die Objektvariable 'objSht' referenziert d. aktuelle Tabellenblatt
Set objSht = ActiveSheet
' Das aktive Tabellenblatt wird benannt
objSht.Name = conKW & "01"
' Spaltenüberschriften einfügen für das 1. Tabellenblatt
Call SpaltenKoepfeEinfügen
' Zählschleife 51 mal durchlaufen
For intWoche = 2 To 52
    ' Neues Tabellenblatt nach dem aktiven einfügen
    Set objSht = Worksheets.Add(After:=ActiveSheet)
    ' Neues Tabellenblatt benennen
    objSht.Name = conKW & Format(intWoche, "00")
    ' Spaltenüberschriften einfügen für das aktuelle Tabellenblatt
    Call SpaltenKoepfeEinfügen
Next intWoche
' Arbeitsmappe schließen
objWkb.Close True
Ausgang:
' Objektvariablen freigeben
Set objWkb = Nothing
Set objSht = Nothing
Exit Sub
Fehler:
MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
    Buttons:=vbCritical, Title:="Laufzeitfehler"
Resume Ausgang
End Sub
```

```
Sub SpaltenKoepfeEinfügen()
Dim varRegion As Variant
Dim intSpalte As Integer
varRegion = Array("Nord", "Süd", "Ost", "West")
For intSpalte = 1 To UBound(varRegion) + 1
    With ActiveSheet
        .Cells(1, intSpalte).Value = varRegion(intSpalte - 1)
    End With
Next intSpalte
End Sub
```

In der Hauptprozedur wird das `Application`-Objekt u.a. dazu genutzt, die aktuell eingesetzte Excel-Version zu bestimmen. In Abhängigkeit davon wird der Zusatz zum Dateinamen (xls bzw.xlsx). Ohne diese Bestimmung würde die Prozedur mit einem Laufzeitfehler abgebrochen.

Einige Excel-Versionen sind in Tab. 1 zusammengestellt:

Name	Version
Excel 2007	12.0
Excel 2003	11.0
Excel XP	10.0
Excel 2000	9.0
Excel 97	8.0
Excel 95	7.0

Tabelle 1: Excel-Versionen

# Workshop 6: VBA-Programmierung mit MS Excel

Haupt- und Unterprozedur sind ausführlich kommentiert. Nur noch ein Hinweis: Mit der Schleifen-Anweisung `For Each shtNeu In Worksheets` wird die Auflistung der Tabellenblätter in der neuen Arbeitsmappe durchlaufen, wobei `shtNeu` eine Objektvariable repräsentiert. Beim Löschen von Tabellenblättern ist darauf zu achten, dass mindestens eines bestehen bleibt. Sonst reagiert Excel mit einer Fehlermeldung.

## 3 Mit Arbeitsmappen arbeiten

### 3.1 Zugriff auf Arbeitsmappen

Der Zugriff auf Arbeitsmappen kann auf verschieden Art und Weise erfolgen. In Tabelle 2 sind die verschiedenen Möglichkeiten zusammengestellt:

<b>ActiveWorkbook</b>	Sie haben Zugriff auf die aktuelle Arbeitsmappe.
<b>ThisWorkbook</b>	Sie haben Zugriff auf die Arbeitsmappe, in der die aktuelle Prozedur ausgeführt wird.
<b>Workbooks (1)</b>	Über ein Element der <code>Workbooks</code> -Auflistung oder eine Objektvariable von Type <code>Workbook</code> besteht ebenfalls Zugriff auf eine bestimmte Arbeitsmappe
<b>Workbooks . Item (1)</b>	
<b>Workbooks ("Name")</b>	
<b>Workbooks . Item ("Name")</b>	
<b>Workbooks-Objektvariable</b>	
<b>Workbooks-Auflistung</b>	Damit können Aktionen durchgeführt werden, die sich allgemein auf die Verwaltung von Arbeitsmappen beziehen.

Tabelle 1: Zugriffsmöglichkeiten auf Arbeitsmappen

In der folgenden Tabelle sind wichtige Eigenschaften und Methoden des `Workbook`-Objekts zusammengestellt:

Eigenschaft		Methode	
FullName	Vollständiger Dateiname	Activate	Aktivieren
Name	Dateiname	Close	Schließen
Theme	Design der Mappe	Save / SaveAs	Speichern
Saved	Speicherstatus	PrintOut	Ausdrucken
Sheets	Auflistung aller Blätter	Protect	Schützen
Worksheets	Auflistung Arbeitsblätter	Unprotect	Schutz aufheben
Charts	Auflistung Diagrammblätter	ApplyTheme	Design zuweisen

Tabelle 2: Ausgewählte Eigenschaften und Methoden des `Workbook`-Objekts

### 3.2 Arbeitsmappen anlegen

Mit der `Add`-Methode kann eine neue Arbeitsmappe angelegt werden. Mit der Eigenschaft `SheetsInWorkbook` wird die Zahl der zur Verfügung gestellten Arbeitsblätter bestimmt. Die Prozedur `NeueArbeitsmappe` erstellt eine neue Arbeitsmappe mit sieben Tabellenblättern. Diese werden dabei auch gleich nach Wochentagen benannt. Zum Benennen der Tabellenblätter wird die eingebaute Textfunktion `Choose` angewandt. In der kopfgesteuerten Schleife `Do While Dir(strPfadname) <> vbNullString` wird ein gültiger vollständiger Pfadname für mithilfe der eingebauten Funktion `Dir` erzeugt.

```
Sub NeueArbeitsmappe()  
    ' Erzeugt eine neue Arbeitsmappe mit 7 Tabellenblättern  
    ' (Montag, ..., Sonntag).  
    Const conSheets As Integer = 7  
    Dim intSht As Integer ' Zählnummer für Tabellenblätter
```

## Workshop 6: VBA-Programmierung mit MS Excel

```
Dim intIdx As Integer ' intIdx zum Dateinamen
Dim strPfadname As String ' Vollständiger Pfadname
Dim strDateiname As String ' Dateiname
Dim strExt As String ' Zusatz zum Dateinamen
Dim strName As String ' Name für Tabellenblatt
' Bei Fehler ...
On Error GoTo Fehler
Application.ScreenUpdating = False ' Bildschirmaktualisierung deaktivieren
strDateiname = "IchBinNeuHier"
' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
If Val(Application.Version) < 12 Then
    strExt = ".xls" ' Excel 2003 und früher
Else
    strExt = ".xlsx" ' Excel 2007
End If
' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateizusatz
strPfadname = Application.DefaultFilePath & "\" & strDateiname & strExt
Do While Dir(strPfadname) <> vbNullString
    ' Dateinamen ändern
    intIdx = intIdx + 1
    strDateiname = strDateiname & CStr(intIdx)
    ' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateizusatz
    strPfadname = Application.DefaultFilePath & "\" & strDateiname & strExt
Loop

MsgBox Prompt:="Standardeinstellung für Zahl der Tabellenblätter" & _
        vbNewLine & _
        "in einer Arbeitsmappe: " & Application.SheetsInNewWorkbook, _
    Buttons:=vbInformation, Title:="Standardeinstellung"
' Neue Arbeitsmappe hinzufügen
Workbooks.Add
With ActiveWorkbook
    ' Die noch fehlende Anzahl Tabellenblätter hinzufügen
    .Sheets.Add Count:=conSheets - Worksheets.Count
    MsgBox Prompt:="Aktuelle Zahl der Tabellenblätter " & vbNewLine & _
        "in der neu angelegten Arbeitsmappe: " & _
        Application.Worksheets.Count, _
        Buttons:=vbInformation, Title:="Anzahl Tabellenblätter"
    ' Zählschleife über alle neu eingefügten Tabellenblätter
    For intSht = 1 To .Worksheets.Count
        strName = Choose(intSht, "Montag", "Dienstag", "Mittwoch", _
            "Donnerstag", "Freitag", "Samstag", "Sonntag")
        With .Worksheets(intSht)
            .Name = strName
            With .[A1] 'Zelle A1 füllen und rot umranden
                .Value = strName
                .BorderAround LineStyle:=xlContinuous, _
                    ColorIndex:=3, Weight:=xlMedium
                .Interior.ColorIndex = 6 ' gelb
            End With
        End With
    Next intSht
    .SaveAs Filename:=strPfadname
    .Close
End With
Application.ScreenUpdating = True ' Bildschirmaktualisierung aktiviert
Ausgang:
Exit Sub
Fehler:
    MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
```

# Workshop 6: VBA-Programmierung mit MS Excel

```
Buttons:=vbCritical, Title:="Laufzeitfehler"  
Resume Ausgang  
End Sub
```

## 3.3 Arbeitsmappen speichern

In den beiden vorstehenden Codebeispielen wurde die neu angelegte Arbeitsmappe mit der `SaveAs`-Methode gespeichert. Dabei wurde das Argument `FileName` vorher mit der Variablen `strPfadName` festgelegt.

Bei Bedarf kann auch auf den integrierten Dialog *Speichern unter* zurückgegriffen werden, der mit der Methode `Show` ausgeführt wird.

```
Sub SpeichernUnterDialog()  
    ' Anwendung des integriertes Dialogfelds 'Speichern unter'  
    If Application.Dialogs(xlDialogSaveAs).Show = True Then  
        MsgBox Prompt:="Gespeichert!", Buttons:=vbInformation, _  
            Title:="Speichern unter"  
    End If  
End Sub
```

Die Auflistung `Dialogs` enthält alle integrierten Dialoge von Excel. Die Konstante `xlDialogSaveAs` dient zum Aufrufen des *Speichern unter*-Dialogs von Excel.

## 3.4 Arbeitsmappen öffnen

Zum Öffnen einer Excel-Arbeitsmappe muss bekannt sein, wie der vollständige Pfadname heißt. Für das Öffnen wird die Methode `Open` angewandt. Die komplette Syntax der Methode `Open` kann in der Online-Hilfe nachgelesen werden. Besonders wichtig ist dabei das Argument `UpdateLinks`. Die Bedeutung der verschiedenen Werte enthält Tab. 1.

Konstante	Bedeutung
1	<code>xlUpdateLinksUserSetting</code> Benutzer gibt an, wie Verknüpfungen aktualisiert werden.
2	<code>xlUpdateLinksNever</code> Verknüpfungen für diese Arbeitsmappe werden beim Öffnen niemals aktualisiert.
3	<code>xlUpdateLinksAlways</code> Verknüpfungen für diese Arbeitsmappe werden beim Öffnen immer aktualisiert.

Quelle: Excel-Online-Hilfe

Tabelle 1: Bedeutung der Konstante `UpdateLinks`

Die folgende Prozedur öffnet die Datei `IchBinNeuHier`, die sich im Standardspeicherort befindet. Falls das nicht zutrifft, sorgt die `On Error`-Anweisung dafür, dass zur Fehlerbehandlung gesprungen wird.

```
Sub ArbeitsmappeOeffnen()  
    Const conDateiname As String = "IchBinNeuHier"  
    Dim strPfadname As String ' Vollständiger Pfadname  
    Dim strExt As String ' Zusatz zum Dateinamen  
    On Error GoTo Fehler  
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen  
    If Val(Application.Version) < 12 Then  
        strExt = ".xls" ' Excel 2003 und früher  
    Else  
        strExt = ".xlsx" ' Excel 2007  
    End If  
End Sub
```

# Workshop 6: VBA-Programmierung mit MS Excel

```
End If
strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
Application.Workbooks.Open Filename:=strPfadname, _
                           UpdateLinks:=xlUpdateLinksNever

Ausgang:
Exit Sub

Fehler:
MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
        Buttons:=vbCritical, Title:="Laufzeitfehler"
Resume Ausgang
End Sub
```

Zum Öffnen von Excel-Arbeitsmappen kann ebenfalls ein integrierter Dialog eingesetzt werden. Dabei kann auch schon der Pfadname mitgegeben werden, den der Öffnen-Dialog anzeigen soll. Die Prozedur dafür befindet sich im folgenden Code-Listing.

```
Sub DialogOeffnenAnzeigen()
Const conDateiname As String = "IchBinNeuHier"
Const conExt      As String = ".xlsx" ' oder ".xls"
Dim strPfadname  As String          ' Vollständiger Pfadname
strPfadname = Application.DefaultFilePath & "\" & conDateiname & conExt
Application.Dialogs(xlDialogOpen).Show strPfadname
End Sub
```

## 3.5 Liste der geöffneten Arbeitsmappen beeinflussen

Beim Klick auf die Office-Schaltfläche werden die zuletzt verwendeten Dokumente angezeigt. Die Anzahl der angezeigten Arbeitsmappen kann wie folgt selbst bestimmt werden.

```
Sub DateilisteManipulieren()
' Liste der zuletzt geöffneten Dokumente beeinflussen
On Error Resume Next
With Application
    .DisplayRecentFiles = True
    .RecentFiles.Maximum = 9 ' Kann ein Wert von 0 bis 9 sein.
End With
End Sub
```

Mit `DisplayRecentFiles = False` wird die Liste der zuletzt verwendeten Dokumente deaktiviert. Sie kann bis zu neun Listeneinträge umfassen.

## 3.6 Mehrere Arbeitsmappen öffnen

Zum Öffnen mehrerer Arbeitsmappen auf einmal kann der integrierte *Öffnen*-Dialog von Excel verwendet werden. Die folgende Prozedur demonstriert, wie's geht:

```
Sub MehrereMappenOeffnen()
Dim varBooks As Variant ' Arbeitsmappen
Dim strFilter As String ' Dateifilterkriterien
Dim intWkb As Integer ' Zähler für Arbeitsmappen
strFilter = "Excel Dateien (*.xls;*.xlsx;*.xlsm;*.xlsb)" & _
           ",*.xls;*.xlsx;*.xlsm;*.xlsb"
varBooks = Application.GetOpenFilename(FileFilter:=strFilter, MultiSelect:=True)
If IsArray(varBooks) Then
    For intWkb = LBound(varBooks) To UBound(varBooks)
        Workbooks.Open varBooks(intWkb)
    Next
End If
End Sub
```

## Workshop 6: VBA-Programmierung mit MS Excel

```
Next intWkb
Else
    Workbooks.Open varBooks
End If
End Sub
```

Das erste Argument `FileFilter` bestimmt, welche Dateien angezeigt werden sollen. Das Argument `MultiSelect` gibt an, ob eine Mehrfachauswahl von Arbeitsmappen möglich ist (`True`) oder nicht (`False`).

### 3.7 Geöffnete Arbeitsmappen ermitteln

Die Prozedur `IstArbeitsmappeGeoeffnet` öffnet eine Arbeitsmappe und prüft anschließend mit der Funktion `IstArbeitsmappe`, ob die Arbeitsmappe wirklich geöffnet wurde.

```
Sub IstArbeitsmappeGeoeffnet()
    Const conDateiname As String = "IchBinNeuHier"
    Dim strPfadname As String ' Vollständiger Pfadname
    Dim strExt As String ' Zusatz zum Dateinamen
    Dim strMsg As String ' Meldung
    On Error GoTo Fehler
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
    If Val(Application.Version) < 12 Then
        strExt = ".xls" ' Excel 2003 und früher
    Else
        strExt = ".xlsx" ' Excel 2007
    End If
    strMsg = "Die Arbeitsmappe " & conDateiname & " ist "
    strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
    Application.Workbooks.Open Filename:=strPfadname, _
        UpdateLinks:=xlUpdateLinksNever
    If IstArbeitsmappe(conDateiname) Then
        MsgBox Prompt:=strMsg & "geöffnet.", Buttons:=vbInformation, _
            Title:="Arbeitsmappe geöffnet."
    Else
        MsgBox Prompt:=strMsg & "nicht geöffnet.", Buttons:=vbInformation, _
            Title:="Arbeitsmappe nicht geöffnet."
    End If
Ausgang:
    Exit Sub
Fehler:
    MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
        Buttons:=vbCritical, Title:="Laufzeitfehler"
    Resume Ausgang
End Sub
```

```
Function IstArbeitsmappe(strPfadname As String) As Boolean
    Dim objWkb As Workbook ' Arbeitsmappe
    On Error Resume Next
    Set objWkb = Workbooks(strPfadname)
    If Err = 0 And Not objWkb Is Nothing Then
        IstArbeitsmappe = True
    End If
End Function
```

Mit der Prozedur `GeoeffneteArbeitsmappenErmitteln` kann über das Auflistungsojekt `Workbooks` festgestellt werden, welche Arbeitsmappen gerade geöffnet sind:



# Workshop 6: VBA-Programmierung mit MS Excel

```
Sub GeoeffneteArbeitsmappenErmitteln()  
    Dim objWkb As Workbook ' Arbeitsmappe  
    Dim intWkb As Integer ' Zähler für Arbeitsmappen  
    Sheets.Add ' Neues Tabellenblatt einfügen  
    For intWkb = 1 To Application.Workbooks.Count  
        Cells(intWkb, 1) = Workbooks(intWkb).Name  
    Next intWkb  
End Sub
```

## 3.8 Arbeitsmappe ohne Rückfrage schließen

Die Prozedur `MappeSchliessen` schließt eine Arbeitsmappe ohne weitere Rückfrage.

```
Sub MappeSchliessen()  
    Application.DisplayAlerts = False ' Warnungen ausschalten  
    ActiveWorkbook.Close SaveChanges:=True ' Mappe schließen und speichern  
    Application.DisplayAlerts = True ' Warnungen einschalten  
End Sub
```

## 3.9 Arbeitsmappe schließen ohne zu speichern

Wenn eine bearbeitete Arbeitsmappe geschlossen werden soll ohne die Änderungen zu speichern, kann die Prozedur `MappeSchliessenOhneSpeichern` eingesetzt werden.

```
Sub MappeSchliessenOhneSpeichern()  
    With Application  
        .DisplayAlerts = False ' Warnungen ausschalten  
        ActiveWorkbook.Close ' Mappe schließen ohne zu speichern  
        .DisplayAlerts = True ' Warnungen einschalten  
    End With  
End Sub
```

## 3.10 Arbeitsmappe löschen

Ohne Einsatz des Datei-Explorers von Windows kann eine Arbeitsmappe mit der Anweisung `Kill` gelöscht werden. Die zu löschende Arbeitsmappe darf dabei nicht geöffnet sein. Das Löschen erfolgt ohne vorherige Warnung. Die Prozedur `MappeLoeschen` demonstriert die Vorgehensweise:

## Workshop 6: VBA-Programmierung mit MS Excel

```
Sub MappedLoeschen()
    Const conDateiname As String = "IchBinNeuHier"
    Dim strPfadname As String ' Vollständiger Pfadname
    Dim strExt As String ' Zusatz zum Dateinamen
    On Error GoTo Fehler
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
    If Val(Application.Version) < 12 Then
        strExt = ".xls" ' Excel 2003 und früher
    Else
        strExt = ".xlsx" ' Excel 2007
    End If
    ' Vollständiger Pfadname
    strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
    If Dir(strPfadname) <> vbNullString Then
        Kill strPfadname ' Arbeitsmappe ohne Warnung löschen
    Else
        MsgBox Prompt:="Eine Arbeitsmappe mit dem Namen " & vbNewLine & _
            conDateiname & strExt & vbNewLine & _
            "konnte nicht gefunden werden!", _
            Buttons:=vbExclamation, Title:="Arbeitsmappe loeschen"
    End If
Ausgang:
    Exit Sub
Fehler:
    MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
        Buttons:=vbCritical, Title:="Laufzeitfehler"
    Resume Ausgang
End Sub
```

Die zusammengehörigen Prozeduren `MappeAnlegen` und `MappeEntfernen` demonstrieren nochmals, wie die Arbeitsmappe `LöschMich` angelegt und danach wieder gelöscht werden kann. Der zugehörige Pfad wird dem Standardspeicherort entnommen. Der Zusatz zum Dateinamen wird mittels Excel-Version bestimmt.

```
Sub MappedAnlegen()
    ' Arbeitsmappe 'LöschMich' anlegen
    Const conDateiname As String = "LöschMich"
    Dim strPfadname As String ' Vollständiger Pfadname
    Dim strExt As String ' Zusatz zum Dateinamen
    On Error GoTo Fehler
    Application.ScreenUpdating = False ' Bildschirmaktualisierung deaktivieren
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
    If Val(Application.Version) < 12 Then
        strExt = ".xls" ' Excel 2003 und früher
    Else
        strExt = ".xlsx" ' Excel 2007
    End If
    ' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateizusatz
    strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
    If Dir(strPfadname) = vbNullString Then
        Workbooks.Add
        ' Angelegte Mappe schließen und Speichern
        ActiveWorkbook.SaveAs Filename:=strPfadname
        ActiveWorkbook.Close
    Else
        MsgBox Prompt:="Die Arbeitsmappe '" & strPfadname & "' existiert bereits!", _
            Buttons:=vbCritical, Title:="Mappe anlegen"
    End If
End Sub
```

# Workshop 6: VBA-Programmierung mit MS Excel

```
Ausgang:
  Application.ScreenUpdating = True ' Bildschirmaktualisierung aktivieren
Exit Sub
Fehler:
  MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
    Buttons:=vbCritical, Title:="Laufzeitfehler"
Resume Ausgang
End Sub
```

```
Sub MappeEntfernen()
  ' Arbeitsmappe 'LöschMich' löschen
  Const conDateiname As String = "LöschMich"
  Dim strPfadname As String ' Vollständiger Pfadname
  Dim strExt As String ' Zusatz zum Dateinamen
  On Error GoTo Fehler ' Fehlerbehandlung
  Application.ScreenUpdating = False ' Bildschirmaktualisierung deaktivieren
  ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen
  If Val(Application.Version) < 12 Then
    strExt = ".xls" ' Excel 2003 und früher
  Else
    strExt = ".xlsx" ' Excel 2007
  End If
  ' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateizusatz
  strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt
  If Dir(strPfadname) <> vbNullString Then
    Kill strPfadname ' Mappe löschen
    MsgBox Prompt:="Die Arbeitsmappe '" & strPfadname & "' wurde gelöscht!", _
      Buttons:=vbInformation, Title:="Mappe löschen"
  Else
    MsgBox Prompt:="Die Arbeitsmappe '" & strPfadname & "' existiert nicht!", _
      Buttons:=vbCritical, Title:="Mappe löschen"
  End If
Ausgang:
  Application.ScreenUpdating = True ' Bildschirmaktualisierung aktivieren
Exit Sub
Fehler:
  MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
    Buttons:=vbCritical, Title:="Laufzeitfehler"
Resume Ausgang
End Sub
```


## 3.11 Arbeitsmappe schützen

Die beiden folgenden Prozeduren demonstrieren, wie alle Blätter eine Arbeitsmappe geschützt bzw. wie der Blattschutz wieder aufgehoben werden kann. Für `Protect` existieren zahlreiche optionale Parameter, siehe Excel-Online-Hilfe.

Blattschutz setzen	Blattschutz aufheben
<pre>Sub BlattschutzSetzen()   'Schutz vor Änderungen setzen   Dim objSht As Worksheet   For Each objSht In ActiveWorkbook.Sheets     objSht.Protect   Next objSht End Sub</pre>	<pre>Sub BlattSchutzAufheben()   'Schutz vor Änderungen aufheben   Dim objSht As Worksheet   For Each objSht In ActiveWorkbook.Sheets     objSht.Unprotect   Next objSht End Sub</pre>

# Workshop 6: VBA-Programmierung mit MS Excel

## 4. Dokumenteneigenschaften

Die Dokumenteneigenschaften einer Arbeitsmappe können angezeigt und auch verändert werden, indem zunächst auf die Office-Schaltfläche  geklickt wird. Über VORBEREITEN und EIGENSCHAFTEN gelangt man zu den Dokumenteigenschaften: Klicken Sie im Dokumentinformationsbereich auf den Pfeil neben Dokumenteigenschaften, um die Eigenschaften auszuwählen, die Sie anzeigen möchten (z. B. Erweiterte Eigenschaften).

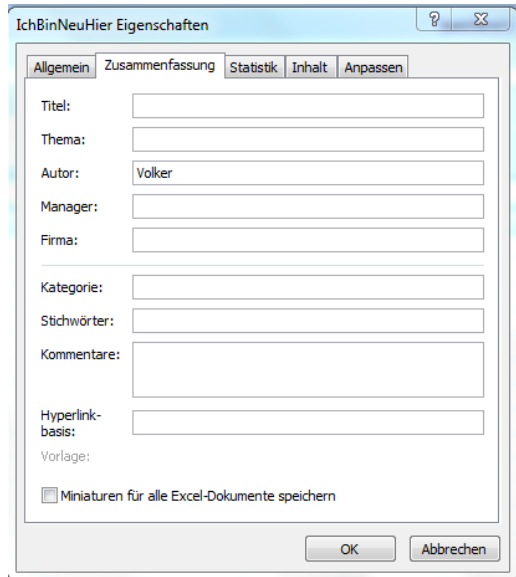


Abb. 1: Dokumenteigenschaften einer Arbeitsmappe

### 4.1 Dokumenteigenschaften auslesen

Mit Hilfe der Prozedur `EigenschaftenAuslesen` können die genauen Bezeichnungen der Felder des Formular (vgl. Abb. 1) in die neue Arbeitsmappe `DocProperties` geschrieben werden.

```
Sub EigenschaftenAuslesen()  
    ' Dokumenteigenschaften in Tabellenblatt schreiben  
    Const conDateiname As String = "DocProperties"  
    Dim objWkb As Workbook ' Arbeitsmappe  
    Dim intZeile As Integer ' Zeilennummer  
    Dim objProp As DocumentProperty ' Eigenschaft  
    Dim strPfadname As String ' Vollständiger Pfadname  
    Dim strExt As String ' Zusatz zum Dateinamen  
    On Error GoTo Fehler  
    Application.ScreenUpdating = False ' Bildschirmaktualisierung deaktivieren  
    ' Dateizusatz in Abhängigkeit von der Excel-Version bestimmen  
    If Val(Application.Version) < 12 Then  
        strExt = ".xls" ' Excel 2003 und früher  
    Else  
        strExt = ".xlsx" ' Excel 2007  
    End If  
    ' Vollständiger Pfadname = Standardspeicherort + Dateiname + Dateizusatz  
    strPfadname = Application.DefaultFilePath & "\" & conDateiname & strExt  
    If Dir(strPfadname) <> vbNullString Then  
        Kill strPfadname  
    End If  
    Set objWkb = Workbooks.Add  
    With objWkb
```

## Workshop 6: VBA-Programmierung mit MS Excel

```
.Sheets(1).Activate
intZeile = 1
For Each objProp In objWkb.BuiltinDocumentProperties
    With objProp
        Cells(intZeile, 1).Value = .Name
        On Error Resume Next
        Cells(intZeile, 2).Value = .Value
        On Error GoTo Fehler
    End With
    intZeile = intZeile + 1
Next objProp
.SaveAs Filename:=strPfadname
End With
Ausgang:
Set objProp = Nothing
Set objWkb = Nothing
Application.ScreenUpdating = True ' Bildschirmaktualisierung aktivieren
Exit Sub
Fehler:
MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
    Buttons:=vbCritical, Title:="Laufzeitfehler"
Resume Ausgang
End Sub
```

	A	B
1	Title	
2	Subject	
3	Author	Volker
4	Keywords	
5	Comments	
6	Template	
7	Last author	
8	Revision number	
9	Application name	Microsoft Excel
10	Last print date	
11	Creation date	15.03.2011 13:56
12	Last save time	
13	Total editing time	0
14	Number of pages	
15	Number of words	
16	Number of characters	

Abb. 2 : Dokumenteigenschaften der aktuellen Arbeitsmappe

### 4.2 Dokumenteigenschaften anzeigen

Mit der Prozedur `DisplayDocProperties` lassen sich mittels eingebautem Dialogfeld die Dokumenteigenschaften anzeigen und pflegen.

```
Sub DisplayDocProperties ()
    Dim dlg As Dialog ' Eingebauter Dialog
    Dim bolErgebnis As Boolean ' Rückgabewert
    Set dlg = Application.Dialogs(xlDialogProperties)
    bolErgebnis = dlg.Show
End Sub
```

# Workshop 6: VBA-Programmierung mit MS Excel

## 4.3 Dokumenteigenschaften setzen

Die Prozedur `SetDocProperties` veranschaulicht, wie die Dokumenteigenschaften gesetzt werden können.

```
Sub SetDocProperties()  
    With ThisWorkbook  
        .BuiltinDocumentProperties("Title").Value = "Titel" '  
        .BuiltinDocumentProperties("Subject").Value = "Thema"  
        .BuiltinDocumentProperties("Author").Value = "Autor"  
        .BuiltinDocumentProperties("Manager").Value = "Manager"  
        .BuiltinDocumentProperties("Company").Value = "Firma"  
        .BuiltinDocumentProperties("Category").Value = "Kategorie"  
        .BuiltinDocumentProperties("Keywords").Value = "Stichwörter"  
        .BuiltinDocumentProperties("Comments").Value = "Kommentar"  
        .BuiltinDocumentProperties("Creation Date").Value = Date  
        .BuiltinDocumentProperties("Last Save Time").Value = Date  
        .BuiltinDocumentProperties("Content Status").Value = "Entwurf"  
    End With  
End Sub
```

## 5 Externe Verknüpfungen ausgeben

In Excel besteht die Möglichkeit, mehrere Arbeitsmappen miteinander zu verknüpfen. Um zu ermitteln, welche Verknüpfungen eine Arbeitsmappe enthält, kann die Prozedur `VerknuepfungenAuflisten` eingesetzt werden. Die einzelnen Verknüpfungen werden nacheinander in ein neues Tabellenblatt geschrieben. Die eingesetzte Methode `LinkSources` gibt mit der Konstanten `xlExcelLinks` die Verknüpfungen zu einem Excel-Tabellenblatt zurück. Wenn keine Verknüpfungen vorhanden sind, wird `Empty` zurückgegeben.

```
Sub VerknuepfungenAuflisten()  
    ' Die Namen verknüpfter Dokumente in einem Tabellenblatt ausgeben  
    Dim objWkb As Workbook ' Arbeitsmappe  
    Dim objSht As Worksheet ' Tabellenblatt  
    Dim varLink As Variant ' externe Verknüpfung  
    Dim intLink As Integer ' Schleifenzähler  
    Set objWkb = ThisWorkbook  
    With objWkb  
        varLink = .LinkSources(xlExcelLinks)  
        ' Wenn keine Verknüpfungen vorhanden sind, wird Empty zurückgegeben.  
        If Not IsEmpty(varLink) Then  
            Set objSht = .Worksheets.Add ' Tabellenblatt hinzufügen  
            For intLink = 1 To UBound(varLink)  
                objSht.Cells(intLink, 1) = varLink(intLink)  
            Next intLink  
        Else  
            MsgBox Prompt:="Diese Arbeitsmappe hat keine " & _  
                "Verknüpfungen zu anderen Mappen!", _  
                Buttons:=vbInformation, Title:="Externe Verknüpfungen"  
        End If  
    End With  
    Set objWkb = Nothing  
    Set objSht = Nothing  
End Sub
```

# Workshop 6: VBA-Programmierung mit MS Excel

## 6. Aufgaben

1. Wie wird die Anzahl der Tabellenblätter festgelegt, die bei Neuanlage einer Arbeitsmappe eingefügt werden?
2. Welche Anweisung kommt zum Einsatz, um eine Arbeitsmappe zu löschen?
3. Wie kann eine Arbeitsmappe geschlossen werden, ohne dass die vorgenommenen Änderungen gespeichert werden?
4. Welche Möglichkeiten bietet Excel, um eine Arbeitsmappe unter einem anderen Namen zu speichern?
5. Wie kann der Name der zu öffnenden Arbeitsmappe zur Laufzeit vom Benutzer angefragt werden?
6. Wie kann erreicht werden, dass eine Arbeitsmappe nur eingesehen, aber nicht verändert werden kann?

## 7. Lösungen

1. Mit Hilfe der Eigenschaft `Application.SheetsInNewWorkbook` wird bestimmt, wie viele Tabellenblätter in einer neuen Arbeitsmappe angelegt werden.
2. Um eine Arbeitsmappe zu löschen, wird die `Kill`-Anweisung benutzt.
3. Zum Schließen einer Arbeitsmappe ohne zu speichern wird die Anweisung `ActiveWorkbook.Close SaveChanges:=False` eingesetzt.
4. Dazu wird die Methode `SaveAs` mit dem Parameter `FileName:= ...` angewandt. Eine weitere Möglichkeit ist die Verwendung des *Speichern unter*-Dialogfelds in Verbindung mit der `Show`-Methode: `Application.Dialogs(xlDialogSaveAS).Show`
5. Der Öffnen-Dialog des `Application`-Objekts kann dazu benutzt werden, um den Namen der zu öffnenden Arbeitsmappe abzufragen: `Application.Dialogs(xlDialogOpen).Show`
6. Sie müssen das betreffende Arbeitsblatt mit folgenden Parametern öffnen:  
`Application.Workbooks.Open Filename:=..., ReadOnly:=True`