# Publisher-Objekte

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 12.03.2025

Überarbeitung:

Beschreibung:

Ab 01.10.2026 wird Microsoft Publisher als Teil von Microsoft 365 nicht mehr unterstützt. Daher habe ich das Kapitel Publisher aus meinem Buch „Die Welt der VBA-Objkte“ entfernt. Als Ersatz dient dieses Dokument.

10-06-02_Anwendung.pub
10-06-03_Ereignisse.pub
10-06-04_Window.pub
10-06-05_Dokumente.pub
10-06-06_Seiten.pub
10-06-07_Shapes.pub
10-06-08_ExcelToPublisher.xlsm
10-06-09_ImportAccessBericht.pub

# 1 Publisher-Anwendungen

Ähnlich wie in Word besitzt die Publisher-Anwendung ein Dokument zur Aufnahme einer Publikation. Mit dem Öffnen einer neuen Anwendung existiert auch ein leeres Dokument das weitere Objekte und Unterobjekte aufnehmen kann, entsprechend der Objekt-Hierarchie von Publisher (Bild 1).

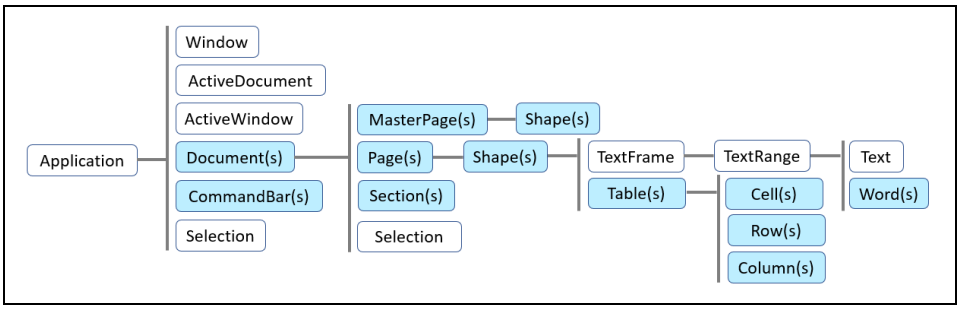



*Bild 1. Die wichtigsten Unterobjekte und Objektlisten (blau) des Application-Objekts*

Das Anwendungs-Objekt *Application* ist das oberste Objekt der Publisher-Objekt-Hierarchie. Darunter steht das *Document*-Objekt, das in einer Liste *Documents* existiert. Entsprechend können die einzelnen Dokumente mit der *For-Each-Next* Schleife gelesen werden, oder ein direkter Zugriff mit dem Index über die *Item*-Eigenschaft erfolgen. Die Prozedur *ReadAllDocuments* (Codeliste 1) liest alle vorhandenen Dokumente der Anwendung und gibt ihre Namen im Direktfenster aus.

*Codeliste 1. Die Prozedur bestimmt alle vorhandenen Dokumente der Anwendung*

```
Sub ReadAllDocuments()
   Dim pubApp  As Application
   Dim pubDocs As Documents
   Dim pubDoc  As Document

   Set pubApp = Application
   Set pubDocs = pubApp.Documents
   For Each pubDoc In pubDocs
      Debug.Print pubDoc.Name
   Next
   Set pubDoc = Nothing
   Set pubDocs = Nothing
   Set pubApp = Nothing
End Sub
```

## 1.1 Eigenschaften

Eigenschaften sind erst mit dem Objekt *Document* wirklich interessant. Die Prozedur *ReadAppAttributs* (Codeliste 2) zeigt einige der wenigen Eigenschaften der Anwendung.

*Codeliste 2. Die Prozedur gibt einige Eigenschaften der aktuellen Anwendung aus*

```
Sub ReadAppAttributs()
   With Application
      Debug.Print "Name......: "; .Name
      Debug.Print "Version...: "; .Version
      Debug.Print "Pfad......: "; .Path
```

```
      Debug.Print "Templates.: "; .TemplateFolderPath
   End With
End Sub
```

Unter *Templates* befinden sich viele interessante Vorlagen, die wir nachfolgend noch betrachten werden.

## 1.2 Methoden

Die Methode *Help* ruft die Microsoft Hilfe zu Top-Anwendungsthemen für den Publisher im aktuellen Browser auf (Codeliste 3).

*Codeliste 3. Die Prozedur ruft die Webseite Top-Kategorien auf*

```
Sub ReadAppHelp()
   Application.Help (pbHelpActiveWindow)
End Sub
```

Die nachfolgende Prozedur setzt das aktuelle Fenster auf eine minimale Darstellung und öffnet danach eine neue Anwendung. Mit der Methode *NewDocument* wird ein neues Dokument nach der Vorlage einer Broschüre (Codeliste 4) erstellt. Die Methode *MsgBox* stoppt den Prozedurablauf, um danach mit der Methode *Quit* die aktuelle Anwendung zu schließen.

*Codeliste 4. Die Prozedur startet ein neues Dokument mit einer Dokumentvorlage*

```
Sub CreateNewDoc()
   Dim pubApp As Application
   Dim pubDoc As Document

   ActiveWindow.WindowState = pbWindowStateMinimize
   Set pubApp = New Publisher.Application
   Set pubDoc = pubApp.NewDocument(pbWizardBrochures)
   pubApp.ActiveWindow.Visible = True
   MsgBox "Bitte Beenden bestätigen!", _
      vbApplicationModal = vbOKOnly
   pubApp.Quit
   ActiveWindow.WindowState = pbWindowStateNormal
   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

Alle *pbWizard-Enumerations* stehen im Anhang 6-1.

Die nächste Prozedur erstellt eine zweite Instanz von Publisher und öffnet die angegebene Publikation schreibgeschützt (Codeliste 5). Voraussetzung ist allerdings, dass diese Anwendung unter dem angegebenen Pfad existiert.

*Codeliste 5. Die Prozedur startet eine schreibgeschützte Anwendung*

```
Sub OpenSavedNewPub()
   Dim pubApp As New Application
   Dim sFile As String

   sFile = "C:\Temp\Demo1.pub"
   pubApp.Open Filename:=sFile, ReadOnly:=True, _
      AddToRecentFiles:=False, _
      SaveChanges:=pbPromptToSaveChanges
   pubApp.ActiveWindow.Visible = True
```

```
End Sub
```

Um eine Datei zu öffnen, sucht Publisher immer zuerst in einem vorgegebenen Ordner nach Dokumenten. Dieser Ordner kann mit der Methode *ChangeFileOpenDirectory* geändert werden (Codeliste 6).

*Codeliste 6. Die Prozedur ändert den Standardordner für Publisher-Dokumente*

```
Sub ChangeAppPath()
   Dim sPath As String
   sPath = "C:\Temp"
   ChangeFileOpenDirectory Dir:=sPath
End Sub
```

10-06-02_Anwendung.pub

## 1.3 Ereignisse

Wie bei allen anderen Anwendungen auch, gibt es keinen Code-Container für die Ereignisse der Publisher-Anwendung. Auch hier muss man also wieder den Weg über eine Objektvariable gehen und mit *WithEvents* die Ereignisse von der Anwendung erben. Wir nutzen wieder ein Klassenmodul. Es bekommt den Namen *clsAppEvents*. Außerdem wählen wir die Ereignis-Prozedur *WindowsActivate* aus (Bild 2) und geben ihr eine Anweisung, die das Anwendungsfenster auf Maximalgröße setzt (Codeliste 7).

*Codeliste 7. Definition eines Ereignis-Objekts und der Ereignis-Prozedur WindowsActivate*

```
Public WithEvents appEvent As Application

Private Sub appEvent_WindowActivate(ByVal Wn As Window)
   Wn.WindowState = pbWindowStateMaximize
End Sub
```

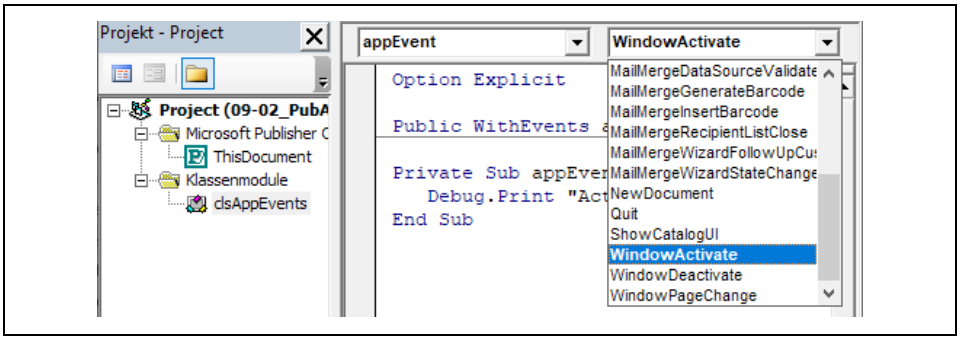



*Bild 2. Auswahl der Anwendungs-Ereignisprozeduren*

Alternativ erweitern wir die Klasse noch um das *WindowsDeactivate*-Ereignis, das die Größe des Anwendungsfensters auf ein Minimum setzt, wenn eine andere Anwendung aktiviert wird (Codeliste 8).

*Codeliste 8. Instanziierung eines Application-Objekts*

```
Private Sub appevent_WindowDeactivate(ByVal Wn As Window)
   Wn.WindowState = pbWindowStateMinimize
End Sub
```

Damit die Events genutzt werden können, muss zuvor eine Instanziierung stattfinden. Diese koppeln wir am Besten direkt mit den Ereignis-Prozeduren *Document_Open* und *Document_BeforeClose* des *ThisDocument*-Objekts (Codeliste 9).

*Codeliste 9. Instanziierung eines Application-Objekts*

```
Dim pubObject As New clsAppEvents

'Create Events
Private Sub Document_Open()
   Set pubObject.appEvent = Application
End Sub

'Delete Events
Private Sub Document_BeforeClose(Cancel As Boolean)
   Set pubObject.appEvent = Nothing
End Sub
```

10-06-03_Ereignisse.pub

### 1.4 ActiveWindow

Das Fenster-Objekt *ActiveWindow* ist ein Unterobjekt des Objekts *VBE*, das Stammobjekt für alle Objekte und Objektlisten in VBA. Das Objekt *ActiveWindows* wird an dieser Stelle erwähnt, weil es das Fenster unter allen geöffneten ist, auf dem der Focus liegt.

Die Prozedur *ChangeWindowCaption* (Codeliste 10) erlaubt die Veränderung der Eigenschaft *Caption* (Text in der Kopfzeile) über einen Dialog mit dem Anwender.

*Codeliste 10. Die Prozedur ändert den Kopftext des aktiven Fensters*

```
Sub ChangeWindowCaption()
   Dim sText As String

   sText = InputBox("Eingabe eines neuen Fenstertextes" & _
      vbCrLf & "für das aktuelle Publisher-Fenster." & vbCrLf & _
      "Nur OK oder Abbrechen " & vbCrLf & _
      "lässt den alten Text bestehen")
   If sText = Empty Then
      Exit Sub
   Else
      ActiveWindow.Caption = sText
   End If
End Sub
```

10-06-04_Window.pub

## 2 Publisher-Dokumente

Das *Document*-Objekt ist der Container für eine Publikation, wie sich auch unschwer aus dem Objektmodell ablesen lässt. Bereits im Kapitel 9.1 wurden die geöffneten Dokumente der aktuellen Publisher-Anwendung bestimmt. Mit *Documents* handelt es sich um eine Objektliste, sodass die einzelnen Dokumente mit der *For-Each-Next*-Schleife oder durch ihren *Index* ansprechbar sind. Eine Besonderheit stellt das Objekt *ActiveDocument* dar, es ist das aktuelle Dokument in der aktuellen Anwendung.

Die Prozedur *CreateNewDocAndClose* (Codeliste 11) öffnet eine neue Anwendung mit einem neuen Dokument. Wichtig ist die Angabe *New*, dadurch lässt sich die Anwendung auch wieder löschen. Ohne diese Angabe wird ein zweites Dokument in der Anwendung erstellt, auch mit einem Anwendungsfenster, aber ein *Quit* würde die aktuelle Anwendung schließen in der noch die Prozedur offen ist. Die Folge ist eine Fehlermeldung.

*Codeliste 11. Die Prozedur öffnet eine neue Anwendung mit neuem Dokument*

```
Sub CreateNewDocAndClose()
   Dim pubApp As Application
   Dim pubDoc As Document

   Set pubApp = New Application
   pubApp.ActiveWindow.Visible = True
   Set pubDoc = pubApp.NewDocument

   MsgBox "Bitte Beenden bestätigen!"

   pubDoc.Close
   pubApp.Quit
   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

Wird ein zweites Dokument in der aktuellen Anwendung erstellt und gespeichert, dann lässt sich die aktuelle Anwendung anschließend auch speichern (Codeliste 12).

*Codeliste 12. Die Prozedur erstellt und speichert ein neues Dokument*

```
Sub CreateNewDocAndSave()
   Dim pubApp As New Application
   Dim pubDoc As Document

   pubApp.ActiveWindow.Visible = True
   Set pubDoc = pubApp.NewDocument
   MsgBox "Dokument speichern und schließen!"
'Speichern und schließen
   pubDoc.SaveAs "C:\Temp\Demo2.pub"

   pubApp.Quit
   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

In der VBA-Hilfe unter *Elemente des Project-Objekts* werden alle Eigenschaften, Methoden und Ereignisse aufgeführt.

## 2.1 Eigenschaften

Drei der wenigen Eigenschaften eines aktiven Dokuments zeigt die Prozedur *ShowPubDocAttributs* (Codeliste 13).

*Codeliste 13. Die Prozedur zeigt Eigenschaften des ActiveDocument-Objekts im Direktfenster*

```
Sub ShowPubDocAttributs()
   With ActiveDocument
      Debug.Print "Name.........: "; .Name
```

```
      Debug.Print "Pfad.........: "; .Path
      Debug.Print "FullName.....: "; .FullName
   End With
End Sub
```

Bezugnehmend auf das Dokument, in dem sich der VBA-Code befindet, kann auch das *ThisDocument*-Objekt verwendet werden (Codeliste 14).

*Codeliste 14. Die Prozedur zeigt Eigenschaften des ThisDocument-Objekts im Direktfenster*

```
Sub ShowThisDocAttributs()
   With ThisDocument
      Debug.Print "Name.........: "; .Name
      Debug.Print "Pfad.........: "; .Path
      Debug.Print "FullName.....: "; .FullName
   End With
End Sub
```

## 2.2 Methoden

Ein Dokument verfügt über die Eigenschaft *Saved*. Dabei handelt es sich um einen schreibgeschützten booleschen Wert, der angibt, ob es Änderungen in der Publikation seit der letzten Speicherung gibt. Die Eigenschaft lässt sich gut mit der Methode *Save* verbinden. So prüft die nachfolgende Prozedur *CheckChangeApp*, ob eine Änderung vorliegt und speichert das aktive Dokument bei Änderung (Codeliste 15).

*Codeliste 15. Speichert eine Publikation nach Änderung*

```
Sub CheckChangeApp()
   If ActiveDocument.Saved = False Then _
      ActiveDocument.Save
End Sub
```

Die Methode *ChangeDocument* (Codeliste 16) ändert das Dokument-Layout nach dem Vorlagetyp *Broschüre* (Bild 3). Für den *pbWizard*-Typ siehe Anhang 6-1.

*Codeliste 16. Änderung eines Dokument-Layouts*

```
Public Sub ChangeDoc()
    ThisDocument.ChangeDocument pbWizardBrochures
End Sub
```


*Bild 3. Dokumentaufbau nach der Broschüren-Vorlage*

## 2.3 Ereignisse

Ein Publisher-Dokument verfügt über wenige Ereignisprozeduren (Bild 4). Die Ereignisse *Open* und *BeforeClose* haben wir bereits für die Anwendungs-Ereignisse genutzt.

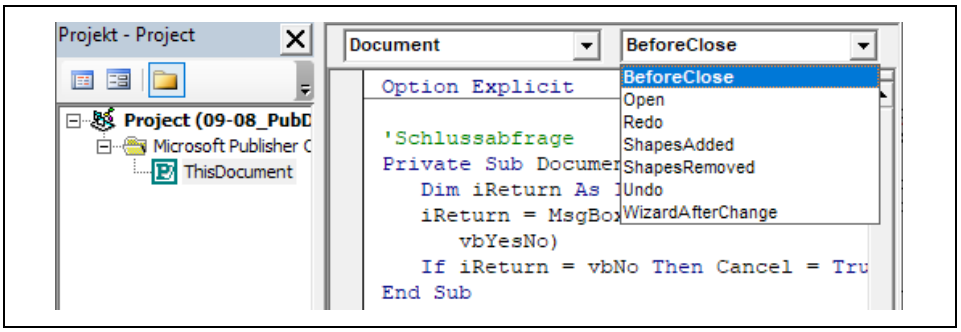



*Bild 4. Ereignis-Prozeduren des Document-Objekts*

Das *BeforeClose*-Ereignis lässt sich auch für ein gezieltes Schließen eines Dokuments im *ThisDocument*-Objekt verwenden (Codeliste 17).

*Codeliste 17. Die Ereignisprozedur reagiert auf das Schließen eines Dokuments*

```
'Schlussabfrage
Private Sub Document_BeforeClose(Cancel As Boolean)
   Dim iReturn As Integer
   iReturn = MsgBox("Sie wollen das Dokument schließen?", _
      vbYesNo)
   If iReturn = vbNo Then Cancel = True
End Sub
```

Wird der Parameter *Cancel* der Ereignis-Prozedur auf *True* gesetzt, dann wird das Schließen des Dokuments verhindert.

10-06-05_Dokumente.pub

# 3 Publisher-Seiten

Die Seiten (Pages) einer Publikation sind die eigentlichen Arbeitsobjekte. Ähnlich wie Folien in PowerPoint werden sie auf der Anwendungsoberfläche in einer Übersichtleiste verwaltet. Die Auswahl ist über die Übersicht möglich.

## 3.1 Seiten

Mit einer neuen Anwendung existiert auch eine leere Seite (Bild 5).

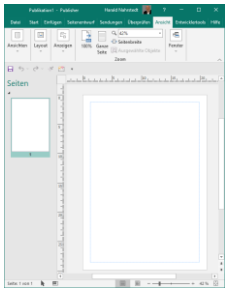


*Bild 5. Aufbau des Anwendungs-Fensters*

Die Prozedur *CreateDocPages* (Codeliste 18) erzeugt zwei weitere Seiten im aktiven Dokument.

*Codeliste 18. Prozedur erzeugt zwei neue Seiten im Dokument*

```
Sub CreateDocPages()
   Dim pubApp As Application
   Dim pubDoc As Document
   Dim pubPages As Pages
   Dim pubPage As Page

   Set pubApp = Application
   Set pubDoc = pubApp.ActiveDocument
   Set pubPages = pubDoc.Pages
   Set pubPage = pubPages.Add(Count:=2, After:=1)

   Set pubPage = Nothing
   Set pubPages = Nothing
   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

Die neuen Seiten gehören ebenfalls zur Objektliste *Pages* (Bild 6).

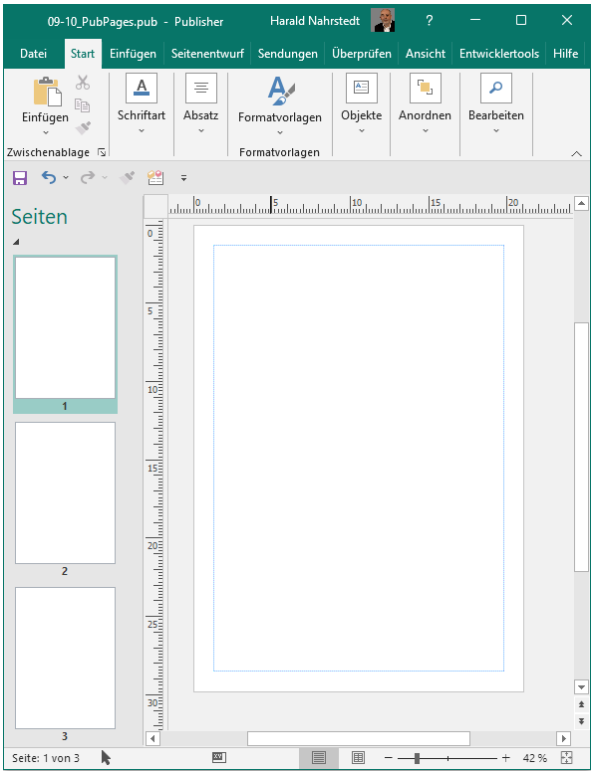



*Bild 6. Weitere Seiten in der Übersichtsleiste*

Über die Eigenschaft *Count* der Pages-Objektliste kann auf einzelne Seiten zugegriffen werden. Mit der Methode *Delete* werden Seiten gelöscht (Codeliste 19).

*Codeliste 19. Prozedur löscht alle Seiten bis auf die erste Seite*

```
Sub DeleteDocPages()
   Dim pubApp As Application
   Dim pubDoc As Document
   Dim iPages As Integer
   Dim iCount As Integer

   Set pubApp = Application
   Set pubDoc = pubApp.ActiveDocument
   iPages = pubDoc.Pages.Count
   If iPages > 1 Then
      For iCount = iPages To 2 Step -1
         pubDoc.Pages(iCount).Delete
      Next iCount
   End If

   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

Publikationen verfügen oft über eine Vielzahl ähnlicher Seiten mit gleichen Strukturen und Inhalten. Daher existieren auch hier, ähnlich wie in PowerPoint, Master-Objekte. Im Menüband werden sie als Gestaltungsvorlagen bezeichnet und dienen für die Publikations-Seiten als Vorlage.

## 3.2 Master-Seiten

Master-Seiten, im Menüband unter *Ansicht / Ansichten / Gestaltungsvorlage*, enthalten Elemente wie Kopf- und Fußzeilen, Seitenzahlen, Rahmen und andere Designelemente, die in der Publikation für alle oder einen Teil der Seiten das Design vorgeben (Codeliste 20).

*Codeliste 20. Die Prozedur erzeugt eine Master-Seite aus der ersten Seite*

```
Sub CreateDocMaster()
   Dim pubApp  As Application
   Dim pubDoc  As Document
   Dim pubPage As Page

   Set pubApp = Application
   Set pubDoc = pubApp.Documents(1)
   Set pubPage = pubDoc.Pages(1).Master
   With pubPage.Shapes.AddShape _
      (Type:=msoShapeSmileyFace, _
      Left:=50, Top:=50, Width:=50, Height:=50)
      .Fill.ForeColor.RGB = RGB(Red:=255, Green:=215, Blue:=0)
      With .TextFrame.TextRange
         .InsertPageNumber
         .ParagraphFormat.Alignment = pbParagraphAlignmentCenter
         With .Font
            .Bold = msoTrue
            .Color.RGB = RGB(Red:=255, Green:=255, Blue:=255)
            .Size = 12
         End With
      End With
   End With
   Set pubPage = Nothing
   Set pubDoc = Nothing
   Set pubApp = Nothing
End Sub
```

In diesem Beispiel wird ein goldener Smiley in der linken oberen Ecke erstellt. Eine *For-Each-Next* Schleife findet alle Shapes (Codeliste 21) und löscht sie.

*Codeliste 21. Die Prozedur löscht alle Shapes auf einer Seite*

```
Sub DeleteMasterShapes()
   Dim pubApp     As Application
   Dim pubDoc     As Document
   Dim pubPage    As Page
   Dim pubShape   As Shape

   Set pubApp = Application
   Set pubDoc = pubApp.Documents(1)
   Set pubPage = pubDoc.Pages(1).Master
   For Each pubShape In pubPage.Shapes
      pubShape.Delete
   Next
```

```
    Set pubApp = Nothing
    Set pubDoc = Nothing
    Set pubPage = Nothing
End Sub
```

Die Prozedur *AddNewPages* (Codeliste 22) erzeugt auf der Master-Seite einen goldenen Smiley nach Größen- und Farbangaben. Erzeugt dann eine weitere Seite unter Beachtung der Master-Seite. Danach wird eine Seite nach der ersten Seite eingefügt, ohne die Vorgaben der Master-Seite zu beachten.

*Codeliste 22. Die Prozedur erstellt Seiten mit und ohne Master-Seiten-Vorgabe*

```
Sub AddNewPages()
   Dim pubPage As Page

   With ActiveDocument
      .MasterPages(1).Shapes.AddShape(Type:=msoShapeSmileyFace, _
         Left:=50, Top:=50, Width:=50, Height:=50).Fill.ForeColor _
         .CMYK.SetCMYK Cyan:=0, Magenta:=40, Yellow:=255, Black:=0
      .Pages.Add Count:=1, After:=1
      Set pubPage = .Pages.Add(Count:=1, After:=1)
      pubPage.IgnoreMaster = True
   End With
End Sub
```

Die Prozedur *AddNewMasterShape* (Codeliste 23) erzeugt ein Shape auf der Master-Seite.

*Codeliste 23. Die Prozedur erstellt ein Shape auf der Master-Seite*

```
Sub AddNewMasterShape()
   With ActiveDocument.Pages(1).Master.Shapes.AddShape _
      (Type:=msoShapeSmileyFace, Left:=500, _
      Top:=50, Width:=50, Height:=50)
      .Fill.ForeColor.CMYK.SetCMYK Cyan:=0, _
         Magenta:=40, Yellow:=255, Black:=0
   End With
End Sub
```

Die nächsten Prozeduren *CreateHeaderFooter* und *DeleteHeaderFooter* (Codeliste 24) stellen einen benutzerdefinierten Text in die Kopf- und Fußzeile bzw. löschen diesen.

*Codeliste 24. Die Prozedur stellt benutzerdefinierten Text in die Kopf- und Fußzeile*

```
Sub CreateHeaderFooter()
   With ActiveDocument.MasterPages(1)
      .Header.TextRange.Text = "Header-Text"
      .Footer.TextRange.Text = "Footer-Text"
   End With
End Sub

Sub DeleteHeaderFooter()
   With ActiveDocument.MasterPages(1)
      .Header.Delete
      .Footer.Delete
   End With
End Sub
```

## 3.3 Abschnitte

Ein Dokument kann auch in Abschnitte aufgeteilt werden. Abschnitts-Objekte (*Sections*) sind in einer *Sections*-Objektliste zusammengefasst, die über einen Zähler *Count* und die Methode *Add* das Hinzufügen neuer Abschnitt erlaubt. Einzelne Abschnitt können somit über einen Index angesprochen werden

Für unser nachfolgendes Beispiel verwenden wir die Vorlage *Magazine Karteikarte*. Sie besitzt zwei Seiten, die für das nachfolgende Beispiel eine Voraussetzung sind (Bild 7).

In der Prozedur *CreateNewSectionPage2* (Codeliste 25) wird der zweiten Seite ein neuer Abschnitt hinzugefügt.

*Codeliste 25. Die Prozedur erzeugt einen Abschnitt auf der zweiten Seite*

```
Sub CreateNewSectionPage2()
   Dim pubSection    As Section
   Dim pubSections   As Sections

   Set pubSections = ActiveDocument.Sections
   Set pubSection = pubSections.Add(StartPageIndex:=2)
   With pubSection
      .PageNumberFormat = pbPageNumberFormatLCRoman
      .PageNumberStart = 1
   End With
   Set pubSection = Nothing
   Set pubSections = Nothing
End Sub
```

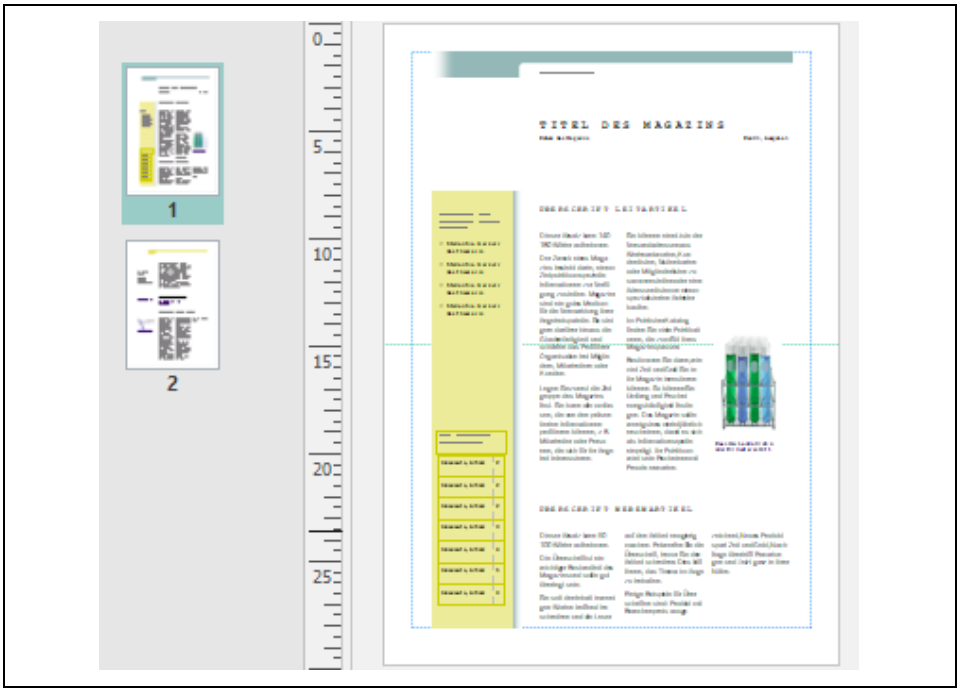



*Bild 7. Vorlage Magazin Karteikarte*

Das Seitenzahlenformat wird auf römische Kleinbuchstaben und die Seitenzahl des neuen Abschnitts auf 1 festgelegt (Bild 8).

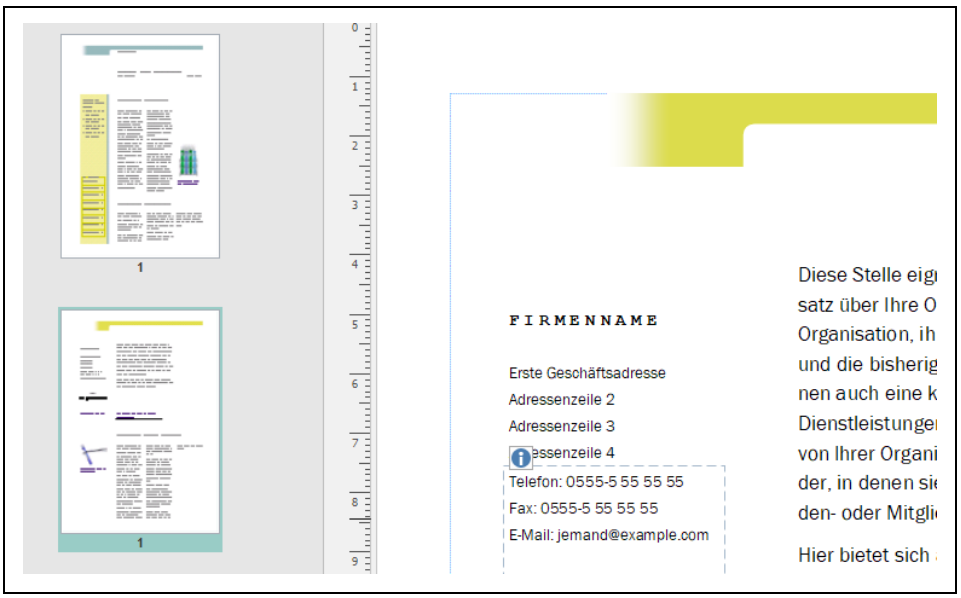



*Bild 8. Die zweite Seite der Publikation besitzt nun ebenfalls die Seitennummer 1*

In der Navigationsleiste ist zu erkennen, dass unter der zweiten Seite als Seitennummer eine Eins angegeben wird.

Das *Section*-Objekt darf nicht mit dem *Selection*-Objekt verwechselt werden, das eine Anwahl in einem Fenster oder Bereich beschreibt. Es kann nur ein *Selection*-Objekt in einer Publikation geben, wie ein *Selection*-Objekt in den anderen Office-Anwendungen auch. Die Prozedur *ReadDocSelection* (Codeliste 26) schreibt einen im Dokument ausgewählten Text ins Direktfenster.

*Codeliste 26. Die Prozedur schreibt einen ausgewählten Text ins Direktfenster*

```
Sub ReadDocSelection()
   Dim pubApp  As Application
   Set pubApp = Application
   Debug.Print pubApp.Selection.TextRange.Text
End Sub
```

10-06-06_Seiten.pub

# 4 Formen

Formen (Shapes) sind die tragenden Objekte der Seitengestaltung. In ihren Formen können sie Texte, Bilder und Grafiken aufnehmen. Die Unterobjekte sind Tabellen, Textrahmen und Textbereiche. Zum nächsten Beispiel verwenden wir die Vorlage *Katalog Brokat*. Die Prozedur *SearchShapes* (Codeliste 27) sucht nach verwendeten Shapes in der Vorlage. Sind sie vom Typ *pbTextFrame*, wird ihr Text im Direktfenster ausgegeben, sonst der Shape-Name.

*Codeliste 27. Die Prozedur sucht nach vorhandenen Shapes*

```
Sub SearchShapes()
   Dim pubPage As Page
   Dim pubShape As Shape
   Dim iShapes As Integer
   Dim iCount As Integer

   For Each pubPage In ActiveDocument.Pages
      For Each pubShape In pubPage.Shapes
         If pubShape.Type = pbTextFrame Then
            Debug.Print pubShape.TextFrame.TextRange.text
         Else
            Debug.Print "Shape-Name: " & pubShape.Name
         End If
      Next
   Next
End Sub
```

Die Anwendung der Prozedur auf die Vorlage *Katalog Brokat* liefert:

```
Kurzartikelüberschrift
Beschreiben Sie hier kurz Ihr Produkt/Ihre Dienstleistung
Beschreiben Sie hier kurz Ihr Produkt/Ihre Dienstleistung
Beschreiben Sie hier kurz Ihr Produkt/Ihre Dienstleistung

Datum: 00.00.00

Firmenname

Katalogtitel

Kataloguntertitel

Shape-Name: Picture 82
Shape-Name: Rectangle 83
Shape-Name: Rectangle 84
Shape-Name: Line 85
Shape-Name: Picture 86
Shape-Name: Picture 87
Shape-Name: Picture 88
Shape-Name: Picture 89
Shape-Name: Picture 90
Shape-Name: Picture 91
Shape-Name: Picture 92
Shape-Name: Picture 93
Shape-Name: Picture 94
Shape-Name: Picture 95
Telefon: 0555-5 55 55 55
Fax: 0555-5 55 55 55
E-Mail: jemand@example.com

Erste Geschäftsadresse
Adressenzeile 2
Adressenzeile 3
Adressenzeile 4

Shape-Name: Group 98
Shape-Name: Picture 102
Shape-Name: Line 103
```

```
Firmenname
```

Die verschiedenen *Shape*-Typen zeigt Anhang 6-2. Auch Autoformen können eingefügt und bearbeitet werden. Die vielen Autoformen zeigt Anhang 6-3.

Die Prozedur *AddFillRectangle* (Codeliste 28) fügt ein Rechteck mit Positions- und Größenangaben auf die erste Seite der Publikation ein. Mit der Angabe von zwei RGB-Farben erhält das Rechteck einen graduellen Farbverlauf (Bild 9).

*Codeliste 28. Die Prozedur fügt ein Rechteck auf der ersten Seite ein*

```
Sub AddFillRectangle()
   With ActiveDocument.Pages(1).Shapes.AddShape _
      (Type:=msoShapeRectangle, _
      Left:=100, Top:=200, Width:=100, Height:=150).Fill
      .ForeColor.RGB = RGB(255, 255, 128)
      .BackColor.RGB = RGB(128, 255, 128)
      .TwoColorGradient msoGradientHorizontal, 1
   End With
End Sub
```

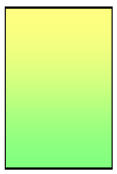


*Bild 9. Eingefügtes Rechteck mit zweifarbigem Farbverlauf*

Auch 3D-Autoformen können verwendet werden. Die Prozedur *AddFillCube* (Codeliste 29) fügt einen Würfel mit einem einfarbig graduellem Farbverlauf ein (Bild 10).

*Codeliste 29. Die Prozedur fügt einen Würfel auf der ersten Seite ein*

```
Sub AddFillCube()
   With ActiveDocument.Pages(1).Shapes. _
      AddShape(Type:=msoShapeCube, _
      Left:=300, Top:=200, Width:=200, Height:=150).Fill
      .ForeColor.RGB = RGB(Red:=255, Green:=255, Blue:=128)
      .OneColorGradient Style:=msoGradientHorizontal, _
      Variant:=1, Degree:=1
   End With
End Sub
```

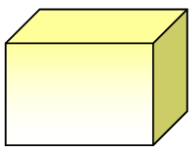


*Bild 10. Eingefügter Würfel mit einfarbigem Farbverlauf*

Die rechteckige Autoform kann auch als Textcontainer verwendet werden. Die Prozedur *AddTextShape* (Codeliste 30) zeigt ein Muster (Bild 11) mit formatiertem Text.

*Codeliste 30. Die Prozedur fügt ein Rechteck mit Text auf der ersten Seite ein*

```
Sub AddTextShape()
   With ActiveDocument.Pages(1).Shapes.AddShape _
      (Type:=msoShapeRectangle, _
      Left:=200, Top:=400, Width:=200, Height:=150)
      .Fill.ForeColor.RGB = vbWhite
      With .TextFrame.TextRange
         .text = "Texteingabe!"
         .Font.Name = "Lucida Sans"
         .Font.Bold = msoTrue
         .Font.Size = 16
      End With
   End With
End Sub
```

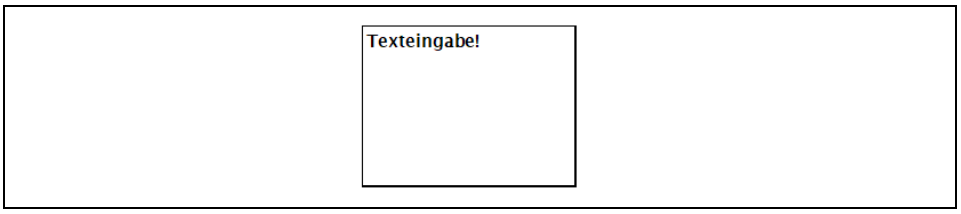



*Bild 11. Eingefügtes Rechteck mit Text*

Als Beispiel für die Verwendung einer weiteren Autoform fügt die Prozedur *AddShapeLine* (Codeliste 31) eine Linie auf der ersten Seite ein.

*Codeliste 31. Die Prozedur fügt eine Line mit Vorder- und Hintergrundfarbe ein*

```
Sub AddShapeLine()
   With ActiveDocument.Pages(1).Shapes.AddLine _
      (BeginX:=200, BeginY:=400, EndX:=500, EndY:=400).Line
      .Weight = 6
      .ForeColor.RGB = vbYellow
      .BackColor.RGB = vbGreen
      .Pattern = msoPatternDarkDownwardDiagonal
   End With
End Sub
```

Eine komplexere Linie als Pfeil fügt die Prozedur *AddFormatLine* (Codeliste 32) auf der ersten Seite ein.

*Codeliste 32. Die Prozedur fügt einen Pfeil ein*

```
Sub AddFormatLine()
   With ActiveDocument.Pages(1).Shapes.AddLine( _
      BeginX:=200, BeginY:=200, EndX:=400, EndY:=300).Line
      .DashStyle = msoLineDashDot
      .ForeColor.RGB = RGB(50, 0, 128)
      .BeginArrowheadLength = msoArrowheadShort
      .BeginArrowheadStyle = msoArrowheadOval
      .BeginArrowheadWidth = msoArrowheadNarrow
      .EndArrowheadLength = msoArrowheadLong
      .EndArrowheadStyle = msoArrowheadTriangle
      .EndArrowheadWidth = msoArrowheadWide
   End With
```

```
End Sub
```

Es wird eine Strich-Punkt-Linie dargestellt, deren Enden durch Pfeil-Attribute beschrieben werden. Eine weitere Shape-Form sind gespeicherte Bilder. Die Prozedur *AddPicture* (Codeliste 33) fügt ein gespeichertes Bild auf der ersten Seite ein.

*Codeliste 33. Die Prozedur fügt ein Bild ein*

```
Sub AddPicture()
   Dim pubShape   As Shape
   Dim sFile      As String

   sFile = "C:\Temp\Koala.png"
   Set pubShape = ActiveDocument.Pages(1).Shapes. _
      AddPicture(Filename:=sFile, LinkToFile:=msoTrue, _
      savewithdocument:=msoFalse, _
      Left:=200, Top:=400, Width:=200, Height:=160)
   Set pubShape = Nothing
End Sub
```

## 4.1 Textrahmen

*TextFrames* sind ein Unterobjekt eines *Shape*-Objekts. Die nachfolgende Prozedur *AddTextToTextFrame* (Codeliste 34) ersetzt im ersten Shape auf der ersten Seite einen Text durch einen neuen (Bild 12). Der alte Text geht damit verloren.

*Codeliste 34. Die Prozedur weist einem Text-Shape einen neuen formatierten Text zu*

```
Sub AddTextToTextFrame()
   With ActiveDocument.Pages(1).Shapes(1). _
      TextFrame.TextRange
      .text = "Neuer Text!"
      With .Font
         .Bold = msoTrue
         .Size = 25
         .Name = "Arial"
      End With
   End With
End Sub
```

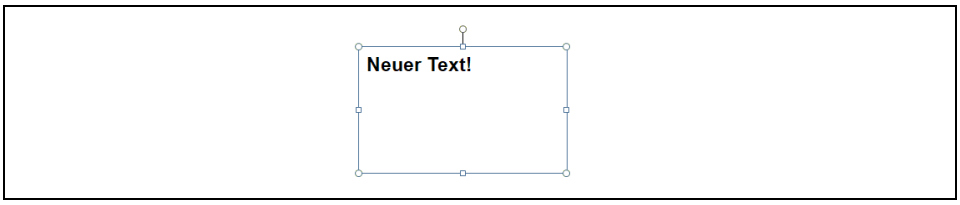



*Bild 12. Textfeld mit neuem Text*

Vorausgesetzt wird, dass die erste Form ein Textcontainer ist. Eine Abfrage kann einen Fehler verhindern. Die Prozedur *GetTextFromTextFrame* (Codeliste 35) prüft alle Shapes auf der ersten Seite auf Textinhalt und zeigt gefundenen Text in einer *MsgBox*.

*Codeliste 35. Die Prozedur überprüft Shapes auf der ersten Seite auf TextFrames*

```
Sub GetTextFromTextFrame()
   Dim shpText As Shape
```

```
    For Each shpText In ActiveDocument.Pages(1).Shapes
       If shpText.HasTextFrame = msoTrue Then
          With shpText.TextFrame
             If .HasText Then MsgBox .TextRange.Text
          End With
       End If
    Next
End Sub
```

Im nächsten Beispiel werden zwei Textfelder miteinander verlinkt (Codeliste 36).

*Codeliste 36. Die Prozedur verlinkt zwei Textfelder*

```
Sub LinkTextBoxes()
    Dim shpTextBox1 As Shape
    Dim shpTextBox2 As Shape

    Set shpTextBox1 = ActiveDocument.Pages(1). _
       Shapes.AddTextbox _
       (msoTextOrientationHorizontal, 70, 70, 70, 35)
    shpTextBox1.TextFrame.TextRange.text = _
       "Dies ist der erste Satz. Dies ist der zweite Satz."

    Set shpTextBox2 = ActiveDocument.Pages(1). _
       Shapes.AddTextbox _
       (msoTextOrientationHorizontal, 160, 70, 70, 35)
    shpTextBox1.TextFrame.NextLinkedTextFrame = _
       shpTextBox2.TextFrame
End Sub
```

Zuerst fügt die Prozedur *LinkTextBoxes* eine *Textbox1* mit einem Text aus zwei Sätzen ein. Dabei reicht die Größe der Textbox nicht zur Darstellung des gesamten Textes aus (Bild 13).

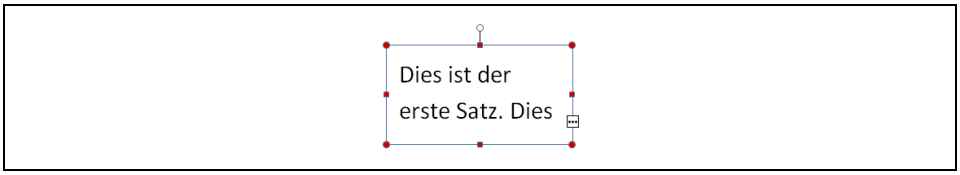



*Bild 13. Erste Textbox mit längerem Text*

Nach der zweiten eingefügten *Textbox2* ohne Inhalt erfolgt eine Verknüpfung, sodass der Text nun beide Textboxen nutzt (Bild 14).

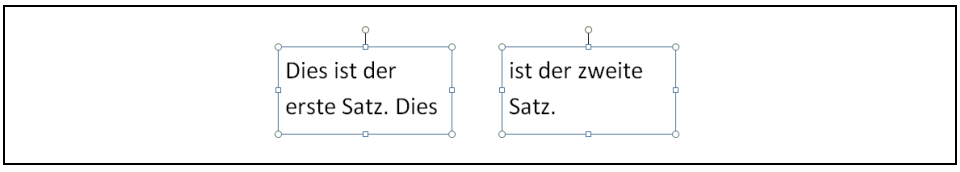



*Bild 14. Zwei verknüpfte Textboxen*

## 4.2 Textbereiche

Textbereiche (*TextRange*-Objekte) sind Unterobjekte der *TextFrame*-Objekte und werden zur gezielten Bearbeitung von *TextFrames* eingesetzt. Im ersten Beispiel nutzt

die Prozedur *CopyAndPasteText* (Codeliste 37) die *Copy and Paste*-Methode, um einen vorhandenen Text in einem *TextRange* zu kopieren (Bild 15).

*Codeliste 37. Die Prozedur kopiert den Text im TextRange*

```
Sub CopyAndPasteText()
   With ActiveDocument.Pages(1).Shapes(1)
      .TextFrame.TextRange.Copy
      .TextFrame.TextRange.InsertAfter vbLf
      .TextFrame.TextRange.Paste
   End With
End Sub
```

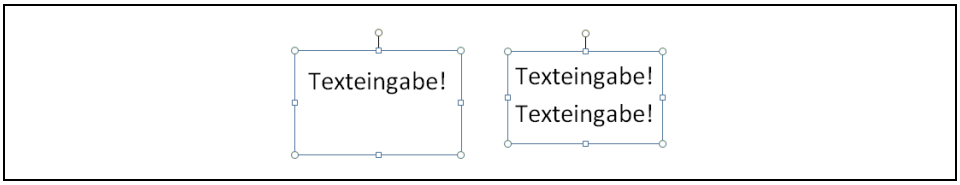



*Bild 15. Textfeld vor und nach dem Kopiervorgang*

Im Textbereich sind Worte ein Unterobjekt. Die Prozedur *FormatWord* (Codeliste 38) formatiert das zweite Wort im ersten Shape auf der ersten Seite (Bild 16).

*Codeliste 38. Die Prozedur formatiert ein einzelnes Wort*

```
Sub FormatWord()
   With ActiveDocument.Pages(1).Shapes(1). _
      TextFrame.TextRange.Words(2).Font
      .Bold = msoTrue
      .Size = 15
      .Name = "Text Name"
   End With
End Sub
```

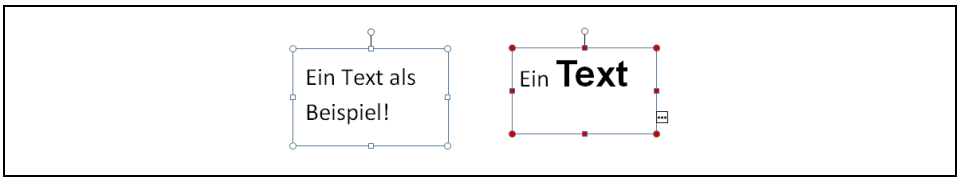



*Bild 16. Textfeld vor und nach der Formatierung*

Die nächste Prozedur *InsertNewText* (Codeliste 39) fügt einem Text in einem Textfeld eine neue Zeile an (Bild 17).

*Codeliste 39. Die Prozedur fügt eine neue Zeile ein*

```
Sub InsertNewText()
   Dim intCount As Integer
   With ActiveDocument.Pages(1).Shapes(1). _
      TextFrame.TextRange
      For intCount = 1 To 2
         .InsertAfter vbLf & "Neue Zeile!"
      Next intCount
   End With
End Sub
```

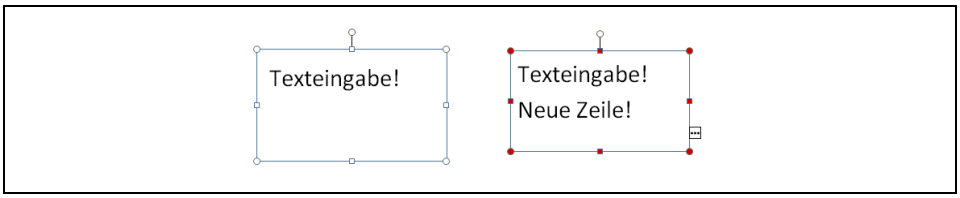



*Bild 17. Textfeld vor und nach dem Einfügen*

## 4.3 Tabellen

Tabellen (*Tables*) erlauben nicht nur die Verwaltung von Daten im ursprünglichen Sinn, sondern auch die Einteilung von Darstellungsbereichen. Dabei sind die Tabellenlinien oft nicht sichtbar dargestellt.

Die Prozedur *CreateNewTable* (Codeliste 40) erstellt auf der ersten Seite eine Tabelle aus fünf Zeilen und fünf Spalten. Anschließend wird die erste Spalte ausgewählt (Bild 18).

*Codeliste 40. Die Prozedur erstellt eine Tabelle auf der ersten Seite*

```
Sub CreateNewTable()
   With ActiveDocument.Pages(1). _
      Shapes.AddTable(NumRows:=5, NumColumns:=5, _
      Left:=100, Top:=200, Width:=300, Height:=100)
      .Table.Columns(1).Cells.Select
   End With
End Sub
```

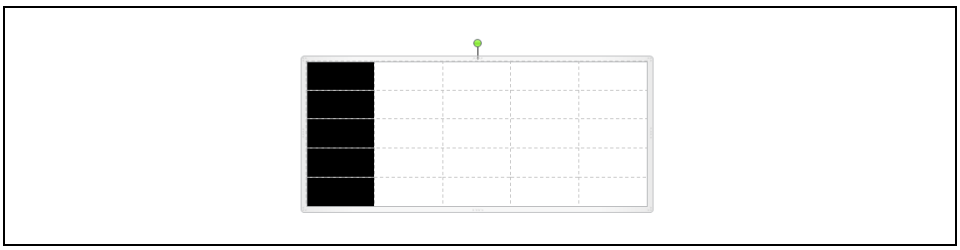


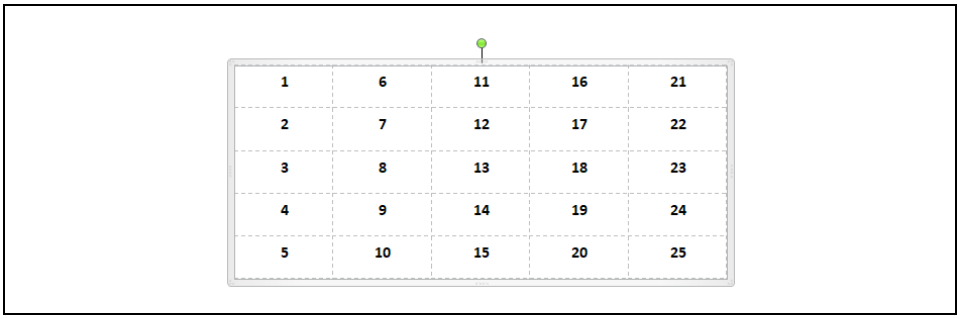
*Bild 18. Erstellte Tabelle auf einer Seite*

Einzelne Zellen einer Tabelle können mit der Merge-Methode verbunden werden. Die Prozedur *MergeCells* (Codeliste 41) fasst in der Tabelle oben links zwei Zellen in Zeilen und Spalten zusammen.

*Codeliste 41. Die Prozedur fasst vier Zellen zusammen*

```
Sub MergeCells()
   ActiveDocument.Pages(1).Shapes(1).Table _
      .Cells(StartRow:=1, StartColumn:=1, _
      EndRow:=2, EndColumn:=2).Merge
End Sub
```

Die Zellen einer Tabelle verfügen ebenfalls über ein *TextRange*-Unterobjekt. Die Prozedur *WriteToCells* (Codeliste 42) nutzt das *TextRange*-Objekt und nummeriert die einzelnen Zellen mit einem Zähler (Bild 19).



| | | | | |
|---|---|---|---|---|
| 1 | 6 | 11 | 16 | 21 |
| 2 | 7 | 12 | 17 | 22 |
| 3 | 8 | 13 | 18 | 23 |
| 4 | 9 | 14 | 19 | 24 |
| 5 | 10 | 15 | 20 | 25 |

*Bild 19. Nummerierte Zellen*

*Codeliste 42. Die Prozedur nummeriert die einzelnen Zellen*

```
Sub WriteToCells()
   Dim pubTable   As Shape
   Dim pubColumn  As Column
   Dim pubCell    As Cell
   Dim iCount     As Integer

   iCount = 1
   Set pubTable = ActiveDocument.Pages(1).Shapes(1)
'Schleife über alle Spalten in der Tabelle
   For Each pubColumn In pubTable.Table.Columns
'Schleife über jede Zelle in der Spalte
      For Each pubCell In pubColumn.Cells
         With pubCell.TextRange
            .text = iCount
            .ParagraphFormat.Alignment = _
               pbParagraphAlignmentCenter 'zentriert
            .Font.Bold = msoTrue
            iCount = iCount + 1
         End With
      Next pubCell
   Next pubColumn
   Set pubTable = Nothing
End Sub
```

Zellen haben auch farbige Eigenschaften. Die Prozedur *FillRowCells* (Codeliste 43) erstellt eine Tabelle mit verschieden farbigen Zellen (Bild 20).

*Codeliste 43. Die Prozedur erstellt eine farbige Tabelle*

```
Sub FillRowCells()
   Dim pubTable As Shape
   Dim pubRow As Row
   Dim pubCell As Cell

   Set pubTable = ActiveDocument.Pages(1).Shapes _
      .AddTable(NumRows:=5, NumColumns:=5, _
      Left:=100, Top:=300, Width:=100, Height:=50)
   For Each pubRow In pubTable.Table.Rows
      For Each pubCell In pubRow.Cells
         If pubCell.Row Mod 2 = 0 Then
            pubCell.Fill.ForeColor.RGB = vbYellow
         Else
            pubCell.Fill.ForeColor.RGB = vbCyan
```

```
         End If
      Next pubCell
   Next pubRow
   Set pubTable = Nothing
End Sub
```

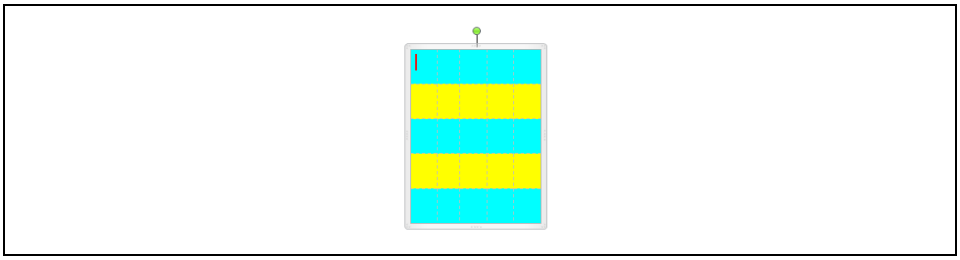


*Bild 20. Erstellte farbige Tabelle*

Den Einfluss auf die Zellgröße zeigt die Prozedur *DontEnlargeTableCells* (Codeliste 44). Sie stellt die Höhe der Zellen auf einen vorgegebenen Wert ein.

*Codeliste 44. Die Prozedur stellt die Zellhöhe auf einen Wert ein*

```
Sub DontEnlargeTableCells()
   Dim rowTable As Row
   With ActiveDocument.Pages(1).Shapes(1).Table
      .GrowToFitText = False
      For Each rowTable In .Rows
         rowTable.Height = 14
      Next
   End With
End Sub
```

Die Prozedur *OutlineBorderCell* (Codeliste 45) ändert den Rahmen einer ausgesuchten Zelle (Bild 21).

*Codeliste 45. Die Prozedur ändert die Begrenzung einer ausgesuchten Zelle*

```
Sub OutlineBorderCell()
   With ActiveDocument.Pages(1).Shapes(1). _
      Table.Columns(2).Cells(2)
      .BorderLeft.Weight = 5
      .BorderRight.Weight = 5
      .BorderTop.Weight = 5
      .BorderBottom.Weight = 5
   End With
End Sub
```

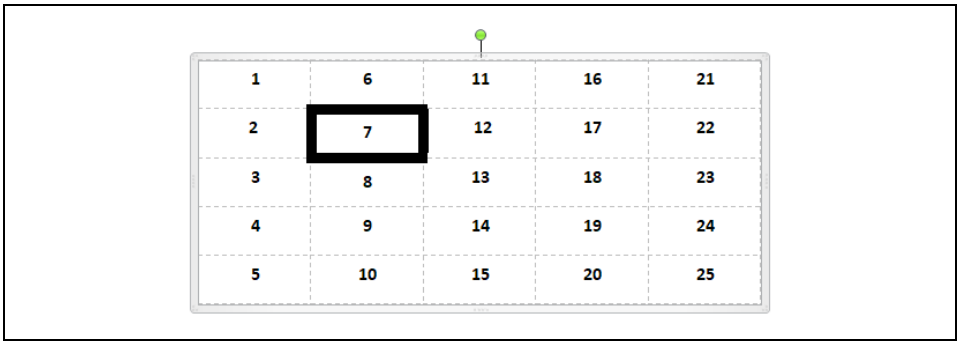



*Bild 21. Geänderter Zellrahmen*

10-06-07_Shapes.pub

# 5 Publisher-Interaktionen

Publisher ist der Spezialist für eine professionelle Druckgestaltung. Daher ist das Einsammeln von Daten aus anderen Anwendungen eine oft genutzte Möglichkeit.

## 5.1 Excel-Daten auf Publisher-Seiten übertragen

Eine Richtung der Übertragung ist die Daten in Excel auf eine Publisher-Seite zu übertragen. Dazu verwenden wir die späte Bindung der Publisher-Objekte, da ein Verweis in der Regel nicht gegeben ist und Übertragen die Tabelle als *UsedRange*-Objekt mit der *Copy-And-Paste*-Methode (Codeliste 46).

*Codeliste 46. Die Prozedur überträgt einen Excel-Bereich auf eine Publisher-Seite*

```
Sub SendDataToPage()
   Dim pubApp     As Object
   Dim exlSheet   As Worksheet
   Dim exlData    As Range
   Dim exlChart   As Chart

   Set pubApp = CreateObject("Publisher.Application")
   Set exlSheet = ActiveSheet
'Überträgt den UsedRange
   pubApp.ActiveWindow.Visible = True
   Set exlData = exlSheet.UsedRange
   exlData.Copy
   pubApp.ActiveDocument.Pages(1).Shapes.Paste
'Erzeugt ein Diagramm und überträgt es
'aus der Zwischenablage
   Set exlChart = ThisWorkbook.Charts.Add
   With exlChart
      .ChartType = xl3DColumn
      .SetSourceData exlData
      .Location xlLocationAsNewSheet
      .CopyPicture
   End With
   pubApp.ActiveDocument.Pages(1).Shapes.Paste
'Shapes positionieren
   With pubApp.Documents(1).Pages(1).Shapes(1)
      .Top = 100
```

```
        .Left = 50
    End With
    With pubApp.Documents(1).Pages(1).Shapes(2)
        .Top = 300
        .Left = 50
        .Width = 400
        .Height = 250
    End With
    Set exlData = Nothing
    Set exlSheet = Nothing
    Set pubApp = Nothing
End Sub
```

10-06-08_ExcelToPublisher.xlsm

Das Ergebnis sind zwei Shapes auf dem Dokument, die noch verschoben und in ihrer Größe angepasst werden (Bild 22).

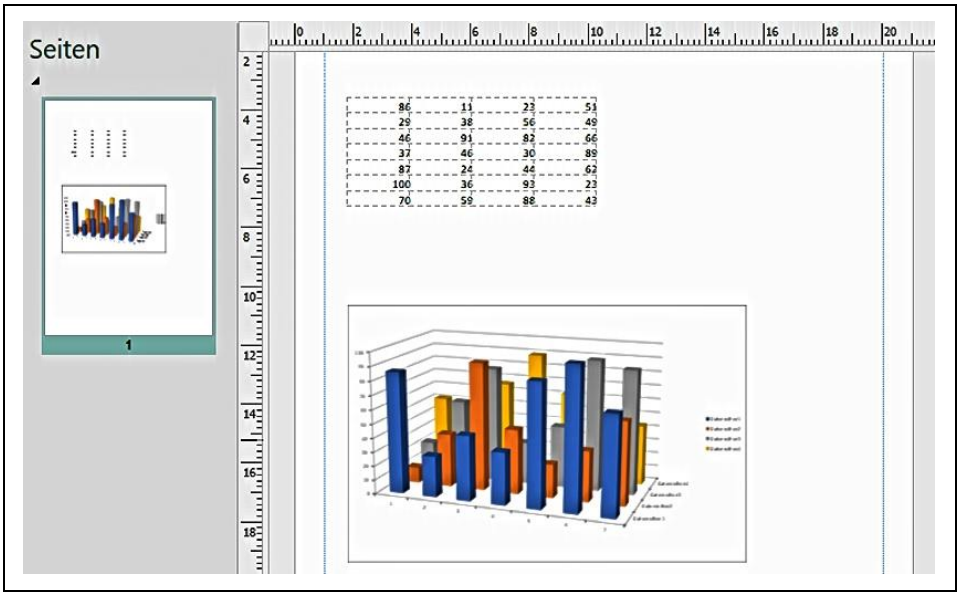



*Bild 22. Die erzeugte Publisher-Seite mit zwei Formen*

Zusätzlich erzeugt die Prozedur in Excel noch ein externes Diagramm-Sheet mit einem Stapel-Diagramm zur Tabelle (Bild 23).

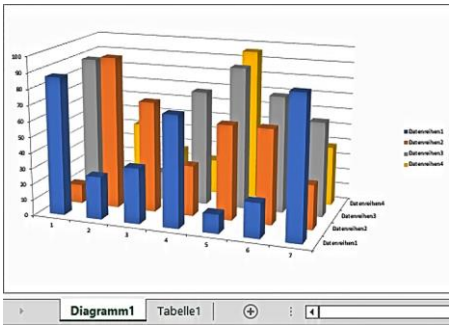



*Bild 23. Neues Diagramm-Sheet in der Excel-Anwendung*

## 5.2 Access-Bericht nach Publisher importieren

An dieser Interaktion sind neben der Publisher- und Access- auch die Word-Anwendung beteiligt. Die Prozedur *ImportAccessReport* (Codeliste 47) im Publisher Modul *modImport* verwendet die späte Bindung, da nicht immer ein Verweis auf die Access- und Word-Bibliothek vorausgesetzt werden kann.

*Codeliste 47. Die Prozedur importiert einen Access-Bericht*

```
Sub ImportAccessReport()
   Dim pubApp     As Application
   Dim pubDoc     As Document
   Dim accApp     As Object
   Dim accReport  As Object
   Dim wrdApp     As Object
   Dim sPfad      As String
   Dim sFile      As String
   Dim sTemp      As String

   sPfad = "C:\Temp\"
   sFile = "Berichte.accdb"
   sTemp = "BerichtAblage.rtf"
   Set pubApp = Publisher.Application
   Set pubDoc = pubApp.ActiveDocument

'Bericht in Access auswählen und kopieren
   Set accApp = CreateObject("Access.Application")
   With accApp
      .Visible = False
      .OpenCurrentDatabase filepath:=sPfad & sFile
      .DoCmd.OpenReport ReportName:="VerkaufUmsatz", _
         View:=2 'acViewPreview
      'acOutputReport = 3
      .DoCmd.OutputTo 3, "VerkaufUmsatz", _
         "Rich Text Format (*.rtf)", sPfad & sTemp
   End With

'Bericht aus Datei nach Word einlesen
   Set wrdApp = CreateObject("Word.Application")
   With wrdApp
      .Visible = False
      .ChangeFileOpenDirectory sPfad
```

```
        .Documents.Open sTemp
        With .Selection
           .WholeStory
           .Copy
        End With
    End With
'Bericht aus der Ablage ins Dokument einfügen
    'Type msoShapeRectangle = 1
    With pubDoc.Pages(1).Shapes.AddShape _
        (Type:=1, _
        Left:=50, Top:=50, Width:=300, Height:=200).Fill
        .ForeColor.RGB = vbWhite
    End With
    With pubDoc.Pages(1).Shapes(1)
        .TextFrame.TextRange.Paste
    End With
'Anwendungen schließen
    accApp.Quit 2 'acQuitSaveNone
    Set accReport = Nothing
    Set accApp = Nothing
    wrdApp.Quit SaveChanges:=0 'wdDoNotSaveChanges
    Set wrdApp = Nothing
    Set pubDoc = Nothing
    Set pubApp = Nothing
End Sub
```

Im ersten Schritt wird die Datenbank *Berichte.accdb* aus Kapitel 6 geöffnet und der Report *VerkaufUmsatz* in der Berichtsansicht dargestellt (Bild 24). Der Bericht wird anschließend im rtf-Format in eine Datei gespeichert.

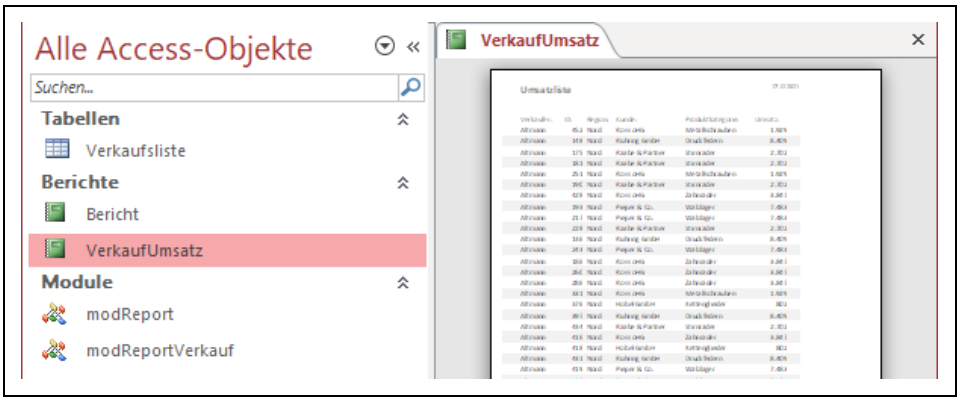



*Bild 24. Report VerkaufUmsatz in der Access-Datenbank*

Wer den Vorgang verfolgen möchte, muss die Eigenschaft *Visible* der jeweiligen Anwendung auf *False* setzen. Im nächsten Schritt wird der Inhalt der Ablagedatei nach Word importiert (Bild 25) und in die Zwischenablage kopiert.

Im letzten Schritt wird der Inhalt der Zwischenablage in eine Form auf der ersten Seite der Publikation eingefügt (Bild 26). Ich habe den Umweg über eine Datei gewählt, weil sich diese Methode auch für andere Anwendungen verwenden lässt.

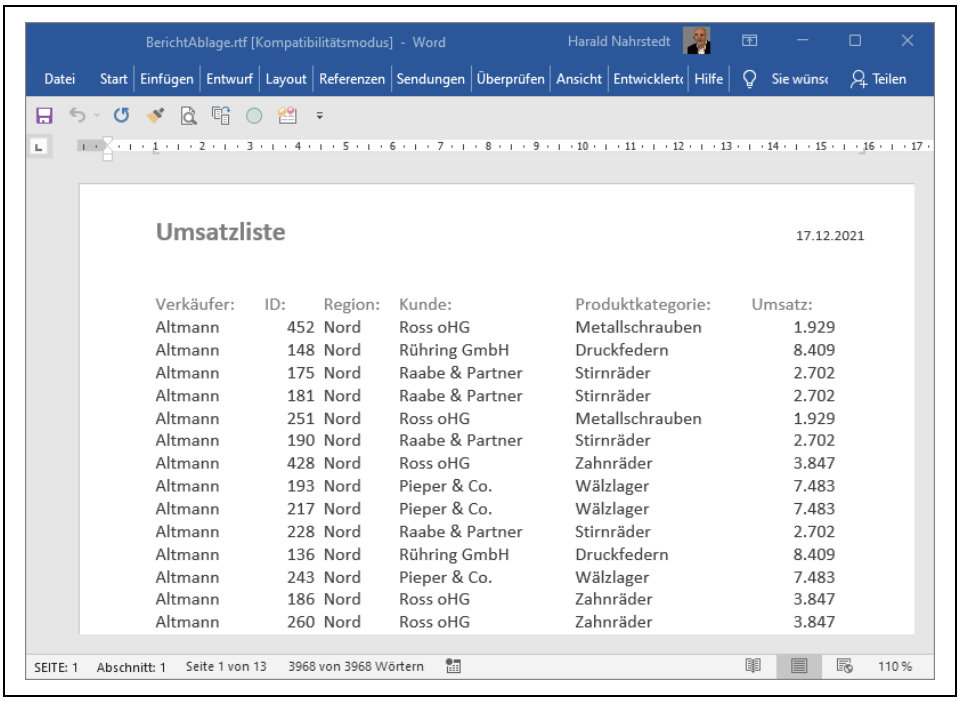



*Bild 25. In Word importierter Inhalt aus der Ablagedatei*

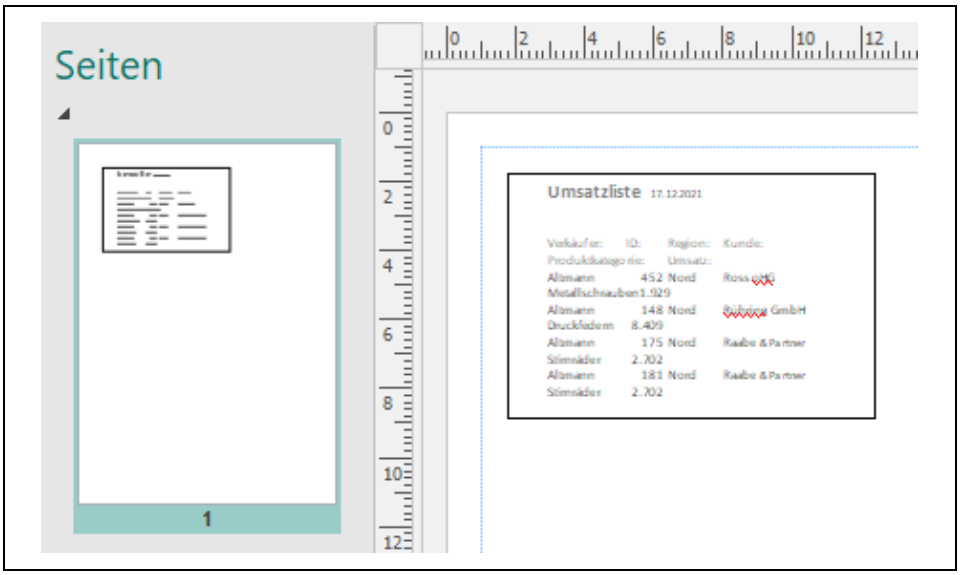



*Bild 26. In Publisher importierter Access-Bericht*

10-06-09_ImportAccessBericht.pub

# 6 Tabellen

*Anhang 6-1. PbWizard-Enumeration*

| Name | Wert | Vorlage für … |
|---|---|---|
| pbWizardAdvertisements | 12 | Werbung |
| pbWizardAirplanes | 23 | Flugzeuge |
| pbWizardBanners | 21 | Banner |
| pbWizardBrochures | 8 | Broschüren |
| pbWizardBusinessCards | 3 | Visitenkarten |
| pbWizardBusinessForms | 20 | Geschäftsformulare (Spesenabrechnung) |
| pbWizardCalendars | 13 | Kalender |
| pbWizardCatalogs | 161 | Kataloge |
| pbWizardCertificates | 62 | Zertifikate |
| pbWizardEmailActivityEvent | 302 | E-Mail-Aktivitätsereignisse |
| pbWizardEmailAutomatic | 305 | automatisch E-Mail-Nachrichten |
| pbWizardEmailFeaturedProduct | 304 | E-Mail-Nachrichten für vorgestellte Produkte |
| pbWizardEmailLetter | 300 | E-Mail-Briefe |
| pbWizardEmailNewsletter | 39 | E-Mail-Newsletter |
| pbWizardEmailProductList | 303 | E-Mail-Produktlisten |
| pbWizardEmailSpeakerEvent | 301 | E-Mail-Speaker-Event |
| pbWizardEnvelopes | 7 | Umschläge |
| pbWizardFlyers | 16 | Flyer |
| pbWizardGiftCertificates | 63 | Geschenkgutscheine |
| pbWizardGreetingCard | 40 | Grußkarten |
| pbWizardInvitation | 41 | Einladungen |
| pbWizardJapaneseAdvertisements | 165 | japanische Werbung |
| pbWizardJapaneseAirplanes | 164 | japanische Flugzeuge |
| pbWizardJapaneseBanners | 121 | japanische Banner |
| pbWizardJapaneseBrochures | 92 | japanische Broschüren |
| pbWizardJapaneseBusinessCards | 91 | japanische Visitenkarten |
| pbWizardJapaneseBusinessForms | 123 | japanische Geschäftsformulare |

| | | |
|---|---|---|
| pbWizardJapaneseCalendars | 82 | japanische Kalender |
| pbWizardJapaneseCatalogs | 177 | japanische Kataloge |
| pbWizardJapaneseCertificates | 119 | japanische Zertifikate |
| pbWizardJapaneseEnvelopes | 93 | japanische Umschläge |
| pbWizardJapaneseFlyers | 94 | japanische Flyer |
| pbWizardJapaneseGiftCertificates | 122 | japanische Geschenkgutscheine |
| pbWizardJapaneseGreetingCards | 80 | japanische Grußkarten |
| pbWizardJapaneseInvitations | 81 | japanische Einladungen |
| pbWizardJapaneseLabels | 118 | japanische Etiketten |
| pbWizardJapaneseLetterheads | 95 | japanische Briefköpfe |
| pbWizardJapaneseMenus | 116 | japanische Menüs |
| pbWizardJapaneseNewsletters | 117 | japanische Newsletter |
| pbWizardJapaneseOrigami | 163 | japanisches Origami |
| pbWizardJapanesePostcards | 78 | japanische Postkarten |
| pbWizardJapanesePrograms | 115 | japanische Programme |
| pbWizardJapaneseSigns | 149 | japanische Zeichen |
| pbWizardJapaneseWebSites | 120 | japanische Websites |
| pbWizardLabels | 19 | Etiketten |
| pbWizardLetterheads | 6 | Briefköpfe |
| pbWizardMenus | 59 | Menüs |
| pbWizardNewsletters | 9 | Newsletter |
| pbWizardNone | 0 | Standard |
| pbWizardOrigami | 22 | Origami |
| pbWizardPostcards | 10 | Postkarten |
| pbWizardPrograms | 76 | Programme |
| pbWizardQuickPublications | 179 | QuickPublikationen |
| pbWizardResumes | 18 | Lebensläufe |
| pbWizardSigns | 17 | Zeichen |
| pbWizardWebSiteBlank | 203 | eine leere Website |
| pbWizardWebSiteHomePage | 5 | eine Homepage für eine Website |
| pbWizardWebSiteProductSales | 201 | eine Produktverkaufswebsite |

| | | |
|---|---|---|
| pbWizardWebSiteServices | 202 | eine Service-Website |
| pbWizardWebSiteThreePage | 200 | eine dreiseitige Website |
| pbWizardWithComplimentsCards | 73 | mit Komplimentkarten |
| pbWizardWordDocument | 189 | ein Microsoft Office Word-Dokument |

*Anhang 6-2. pbShapeType-Enumeration*

| Name | Wert | Beschreibung |
|---|---|---|
| pbAutoShape | 1 | AutoShape |
| pbCallout | 2 | Legende |
| pbCatalogMergeArea | 111 | Datensatzkatalogbereich |
| pbChart | 3 | Diagramm |
| pbComment | 4 | Kommentar |
| pbEmbeddedOLEObject | 7 | Eingebettetes OLE-Objekt |
| pbFormControl | 8 | Formularsteuerelement |
| pbFreeform | 5 | Freihandform |
| pbGroup | 6 | Group |
| pbGroupWizard | 108 | Gruppen-Assistent |
| pbLine | 9 | Linie |
| pbLinkedOLEObject | 10 | Verknüpftes OLE-Objekt |
| pbLinkedPicture | 11 | Grafik |
| pbMedia | 16 | Medien |
| pbOLEControlObject | 12 | OLE-Steuerelementobjekt |
| pbPicture | 13 | Grafik |
| pbPlaceholder | 14 | Platzhalter |
| pbShapeTypeMixed | -2 | Gemischter Formtyp |
| pbTable | 18 | Tabelle |
| pbTextEffect | 15 | Texteffekt |
| pbTextFrame | 17 | Textrahmen |
| pbWebCheckBox | 100 | Webkontrollkästchen |
| pbWebCommandButton | 101 | Webbefehlsschaltfläche |
| pbWebHotSpot | 110 | Webhotspot |
| pbWebHTMLFragment | 107 | Web-HTML-Fragment |

| | | |
|---|---|---|
| pbWebListBox | 102 | Weblistenfeld |
| pbWebMultiLineTextBox | 103 | Mehrzeiliges Webtextfeld |
| pbWebNavigationBar | 112 | Webnavigationsleiste |
| pbWebOptionButton | 104 | Web-Optionsschaltfläche |
| pbWebSingleLineTextBox | 105 | Einzeiliges Webtextfeld |
| pbWebWebComponent | 106 | Web-Webkomponente |

*Anhang 6-3. msoShapeType-Enumeration*

| msoShape … | Wert | Beschreibung |
|---|---|---|
| 10pointStar | 149 | 10-zackiger Stern |
| 12pointStar | 150 | 12-zackiger Stern |
| pointStar | 94 | 16-zackiger Stern |
| 24pointStar | 95 | 24-zackiger Stern |
| 32pointStar | 96 | 32-zackiger Stern |
| 4pointStar | 91 | 4-zackiger Stern |
| 5pointStar | 92 | 5-zackiger Stern |
| 6pointStar | 147 | 6-zackiger Stern |
| 7pointStar | 148 | 7-zackiger Stern |
| 8pointStar | 93 | 8-zackiger Stern |
| ActionButtonBackorPrevious | 129 | Schaltfläche Zurück |
| ActionButtonBeginning | 131 | Schaltfläche Start |
| ActionButtonCustom | 125 | Schaltfläche Benutzerdefiniert |
| ActionButtonDocument | 134 | Schaltfläche Dokument |
| ActionButtonEnd | 132 | Schaltfläche Ende |
| ActionButtonForwardorNext | 130 | Schaltfläche Weiter oder Nächstes |
| ActionButtonHelp | 127 | Schaltfläche Hilfe |
| ActionButtonHome | 126 | Schaltfläche Startseite |
| ActionButtonInformation | 128 | Schaltfläche Information |
| ActionButtonMovie | 136 | Schaltfläche Film |
| ActionButtonReturn | 133 | Schaltfläche Zurück |
| ActionButtonSound | 135 | Schaltfläche Lied |
| Arc | 25 | Bogen |

| | | |
|---|---|---|
| Balloon | 137 | Ballon |
| BentArrow | 41 | Gebogener Pfeil |
| BentUpArrow | 44 | Nach oben gebogener Pfeil |
| Bevel | 15 | Fase |
| BlockArc | 20 | Blockierter Pfeil |
| Can | 13 | Können |
| ChartPlus | 182 | Diagramm Plus |
| ChartStar | 181 | Diagramm Stern |
| ChartX | 180 | Diagramm X |
| Chevron | 52 | Chevron |
| Chord | 161 | Akkord |
| CircularArrow | 60 | Kreisförmiger Pfeil |
| Cloud | 179 | Wolke |
| CloudCallout | 108 | Cloud Aufruf |
| Corner | 162 | Ecke |
| CornerTabs | 169 | Eckige Tabs |
| Cross | 11 | Überqueren |
| Cube | 14 | Kubus |
| CurvedDownArrow | 48 | Gebogener Abwärtspfeil |
| CurvedDownRibbon | 100 | Abgebogenes Band |
| CurvedLeftArrow | 46 | Gebogener linker Pfeil |
| CurvedRightArrow | 45 | Gebogener rechter Pfeil |
| CurvedUpArrow | 47 | Gebogener Aufwärtspfeil |
| CurvedUpRibbon | 99 | Aufgebogenes Band |
| Decagon | 144 | Zehneck |
| DiagonalStripe | 141 | Diagonaler Streifen |
| Diamond | 4 | Diamant |
| Dodecagon | 146 | Zwölfeck |
| Donut | 18 | Donut |
| DoubleBrace | 27 | Doppelstrebe |
| DoubleBracket | 26 | Doppelhalterung |

| | | |
|---|---|---|
| DoubleWave | 104 | Doppelwelle |
| DownArrow | 36 | Pfeil nach unten |
| DownArrowCallout | 56 | Legende mit Pfeil nach unten |
| DownRibbon | 98 | Band abwärts |
| Explosion1 | 89 | Explosion 1 |
| Explosion2 | 90 | Explosion 2 |
| FlowchartAlternateProcess | 62 | Flussdiagramm Alternativer Prozess |
| FlowchartCard | 75 | Karte |
| FlowchartCollate | 79 | Sortieren |
| FlowchartConnector | 73 | Verbindung |
| FlowchartData | 64 | Daten |
| FlowchartDecision | 63 | Entscheidung |
| FlowchartDelay | 84 | Verzögerung |
| FlowchartDirectAccessStorage | 87 | Zugriffsspeicher |
| FlowchartDisplay | 88 | Anzeige |
| FlowchartDocument | 67 | Dokument |
| FlowchartExtract | 81 | Auszug |
| FlowchartInternalStorage | 66 | Interner Speicher |
| FlowchartMagneticDisk | 86 | Magnetplatten |
| FlowchartManualInput | 71 | Manuelle Eingabe |
| FlowchartManualOperation | 72 | Manuelle Operation |
| FlowchartMerge | 82 | Zusammenführung |
| FlowchartMultidocument | 68 | Mehrere Dokumente |
| FlowchartOfflineStorage | 139 | Offlinespeicher |
| FlowchartOffpageConnector | 74 | Offpage-Verbindung |
| FlowchartOr | 78 | OR-Verbindung |
| FlowchartPredefinedProcess | 65 | Definierter Prozessfluss |
| FlowchartPreparation | 70 | Vorbereitungs-Fluss |
| FlowchartProcess | 61 | Prozessfluss |
| FlowchartPunchedTape | 76 | Lochstreifen |
| FlowchartSequentialAccessStorage | 85 | Sequentieller Zugriffsspeicher |

| | | |
|---|---|---|
| FlowchartSort | 80 | Sortierung |
| FlowchartStoredData | 83 | Gespeicherte Daten |
| FlowchartSummingJunction | 77 | Summierende Kreuzung |
| FlowchartTerminator | 69 | Terminator |
| FoldedCorner | 16 | Gefaltete Ecke |
| Frame | 158 | Rechteckiger Bilderrahmen |
| Funnel | 174 | Trichter |
| Gear6 | 172 | Zahnrad mit 6 Zähnen |
| Gear9 | 173 | Zahnrad mit 9 Zähnen |
| HalfFrame | 159 | Hälfte eines rechteckigen Bilderrahmens |
| Heart | 21 | Herz |
| Heptagon | 145 | Heptagon |
| Hexagon | 10 | Hexagon |
| HorizontalScroll | 102 | Horizontales Scrollen |
| IsoscelesTriangle | 7 | Gleichschenkliges Dreieck |
| LeftArrow | 34 | Linker Pfeil |
| LeftArrowCallout | 54 | Legende mit Pfeil nach links |
| LeftBrace | 31 | Linke Klammer |
| LeftBracket | 29 | Rechte Klammer |
| LeftCircularArrow | 176 | Runder Pfeil gegen den Uhrzeigersinn |
| LeftRightArrow | 37 | Links-Rechts-Pfeil |
| LeftRightArrowCallout | 57 | Legende mit Links-Rechts-Pfeil |
| LeftRightCircularArrow | 177 | Kreisförmiger Pfeil mit Pfeilenden |
| LeftRightRibbon | 140 | Band mit Pfeil an beiden Enden |
| LeftRightUpArrow | 40 | Blockpfeil nach links, rechts und oben |
| LeftUpArrow | 43 | Blockpfeil nach links und oben |
| LightningBolt | 22 | Blitz |
| LineCallout1 | 109 | Legende mit Rahmen und hor. Legendenlinie |
| LineCallout1AccentBar | 113 | Legende mit hor. Akzentleiste |
| LineCallout1BorderandAccentBar | 121 | Callout with border and horizontal accent bar |
| LineCallout1NoBorder | 117 | Callout with horizontal line |

| | | |
|---|---|---|
| LineCallout2 | 110 | Callout with diagonal straight line |
| LineCallout2AccentBar | 114 | Callout with diagonal callout line and accent bar |
| LineCallout2BorderandAccentBar | 122 | Callout with border, diagonal straight line, and accent bar |
| LineCallout2NoBorder | 118 | Callout with no border and diagonal callout line |
| LineCallout3 | 111 | Callout with angled line |
| LineCallout3AccentBar | 115 | Callout with angled callout line and accent bar |
| LineCallout3BorderandAccentBar | 123 | Callout with border, angled callout line, and accent bar |
| LineCallout3NoBorder | 119 | Callout with no border and angled callout line |
| LineCallout4 | 112 | Callout with callout line segments forming a U-shape |
| LineCallout4AccentBar | 116 | Callout with accent bar and callout line segments forming a U-shape |
| LineCallout4BorderandAccentBar | 124 | Callout with border, accent bar, and callout line segments forming a U-shape |
| LineCallout4NoBorder | 120 | Callout with no border and callout line segments forming a U-shape |
| LineInverse | 183 | Line inverse |
| MathDivide | 166 | Division symbol ÷ |
| MathEqual | 167 | Equivalence symbol = |
| MathMinus | 164 | Subtraction symbol - |
| MathMultiply | 165 | Multiplication symbol x |
| MathNotEqual | 168 | Non-equivalence symbol ≠ |
| MathPlus | 163 | Addition symbol + |
| Mixed | -2 | Return value only; indicates a combination of the other states. |
| Moon | 24 | Moon |
| NonIsoscelesTrapezoid | 143 | Trapezoid with asymmetrical non-parallel sides |
| NoSymbol | 19 | "No" symbol |
| NotchedRightArrow | 50 | Notched block arrow that points right |
| NotPrimitive | 138 | Not supported |
| Octagon | 6 | Octagon |

| | | |
|---|---|---|
| Oval | 9 | Oval |
| OvalCallout | 107 | Oval-shaped callout |
| Parallelogram | 2 | Parallelogram |
| Pentagon | 51 | Pentagon |
| Pie | 142 | Circle ('pie') with a portion missing |
| PieWedge | 175 | Quarter of a circular shape |
| Plaque | 28 | Plaque |
| PlaqueTabs | 171 | Four quarter-circles defining a rectangular shape |
| QuadArrow | 39 | Block arrows that point up, down, left, and right |
| QuadArrowCallout | 59 | Callout with arrows that point up, down, left, and right |
| Rectangle | 1 | Rectangle |
| RectangularCallout | 105 | Rectangular callout |
| RegularPentagon | 12 | Pentagon |
| RightArrow | 33 | Block arrow that points right |
| RightArrowCallout | 53 | Callout with arrow that points right |
| RightBrace | 32 | Right brace |
| RightBracket | 30 | Right bracket |
| RightTriangle | 8 | Right triangle |
| Round1Rectangle | 151 | Rectangle with one rounded corner |
| Round2DiagRectangle | 157 | Rectangle with two rounded corners, diagonally-opposed |
| Round2SameRectangle | 152 | Rectangle with two-rounded corners that share a side |
| RoundedRectangle | 5 | Rounded rectangle |
| RoundedRectangularCallout | 106 | Rounded rectangle-shaped callout |
| SmileyFace | 17 | Smiley face |
| Snip1Rectangle | 155 | Rectangle with one snipped corner |
| Snip2DiagRectangle | 157 | Rectangle with two snipped corners, diagonally-opposed |
| Snip2SameRectangle | 156 | Rectangle with two snipped corners that share a side |

| | | |
|---|---|---|
| SnipRoundRectangle | 154 | Rectangle with one snipped corner and one rounded corner |
| SquareTabs | 170 | Four small squares that define a rectangular shape |
| StripedRightArrow | 49 | Block arrow that points right with stripes at the tail |
| Sun | 23 | Sun |
| SwooshArrow | 178 | Curved arrow |
| Tear | 160 | Water droplet |
| Trapezoid | 3 | Trapezoid |
| UpArrow | 35 | Block arrow that points up |
| UpArrowCallout | 55 | Callout with arrow that points up |
| UpDownArrow | 38 | Block arrow that points up and down |
| UpDownArrowCallout | 58 | Callout with arrows that point up and down |
| UpRibbon | 97 | Ribbon banner with center area above ribbon ends |
| UTurnArrow | 42 | Block arrow forming a U shape |
| VerticalScroll | 101 | Vertical scroll |
| Wave | 103 | Wave |

# Literaturverzeichnis

[1] Balzert, Heide, UML 2 kompakt mit Checklisten, 3. Aufl.,
Spektrum Akademischer Verlag, 2010
[2] Weilkins, Tim, System Engineering mit SysMl/UML, 2. Auflage, dpunkt verlag, 2008
[3] Nahrstedt, Harald, Excel + VBA für Ingenieure, 8. Auflage, Springer Verlag, 2025
[4] Freßdorf / Gahler / Meister, Mikrosoft Word Programmierung, 4. Auflage,
Microsoft Press, 2014
[5] Mansfield, Richard, Mastering VBA for Office 2016, 3. Auflage, John Wiley & Sons, 2016
[6] Slovak, Ken, Outlook 2007 Programming, John Wiley & Sons, 1. Auflage, 2007
[7] Hölscher, Lorenz, Access 2013 VBA-Programming, Microsoft Press, 2013
[8] Martin, Rene, Visio anpassen und programmieren, BoD, 2021
[9] Gill, Rod, VBA Programming for MS Project, Msprojectexperts, 2011