R | R | Z | N |
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Export nach Word

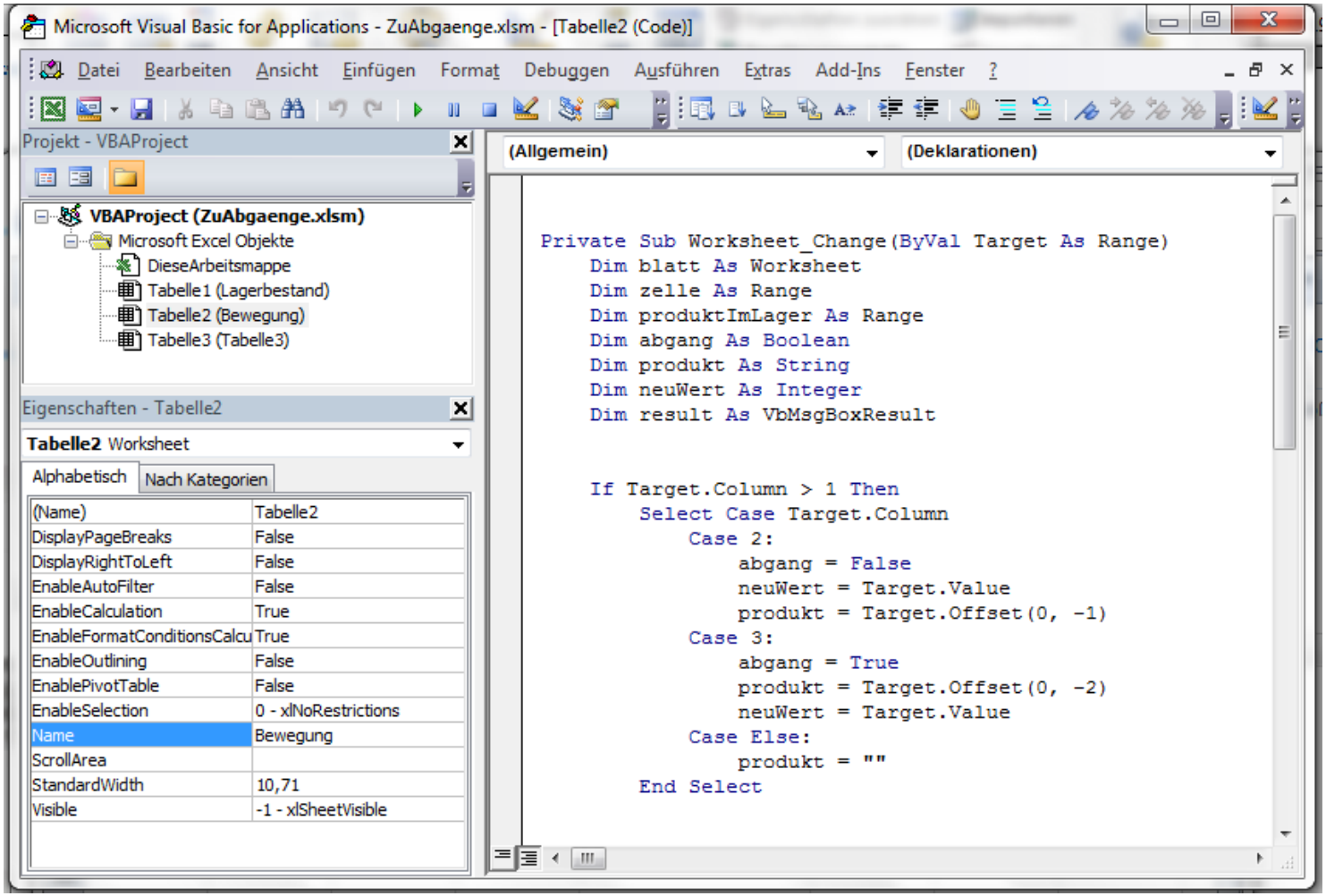
Export nach Word ...

- ist nur über die Zwischenablage möglich. Hinweis: In einem Excel-Makro werden Aktionen in Word nicht aufgezeichnet.
- kann über die Seriendruckfunktion von Word erfolgen.
- ist mit Hilfe von VBA möglich.

VBA-Editor öffnen

- Das Menüband Entwicklertools ist eingeblendet.
- Mit einem Klick auf *Visual Basic* in der Gruppe Code wird der VBA-Editor geöffnet.

Beispiel


Microsoft Visual Basic for Applications - ZuAbgaenge.xlsm - [Tabelle2 (Code)]
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Projekt - VBAProject
VBAProject (ZuAbgaenge.xlsm)
Microsoft Excel Objekte
DieseArbeitsmappe
Tabelle1 (Lagerbestand)
Tabelle2 (Bewegung)
Tabelle3 (Tabelle3)
Eigenschaften - Tabelle2
Tabelle2 Worksheet
Alphabetisch
Nach Kategorien
(Name) Tabelle2
DisplayPageBreaks False
DisplayRightToLeft False
EnableAutoFilter False
EnableCalculation True
EnableFormatConditionsCalcu True
EnableOutlining False
EnablePivotTable False
EnableSelection 0 - xlNoRestrictions
Name Bewegung
ScrollArea
StandardWidth 10,71
Visible -1 - xlSheetVisible
(Allgemein)
(Deklarationen)
Private Sub Worksheet_Change(ByVal Target As Range)
Dim blatt As Worksheet
Dim zelle As Range
Dim produktImLager As Range
Dim abgang As Boolean
Dim produkt As String
Dim neuWert As Integer
Dim result As VbMsgBoxResult
If Target.Column > 1 Then
Select Case Target.Column
Case 2:
abgang = False
neuWert = Target.Value
produkt = Target.Offset(0, -1)
Case 3:
abgang = True
produkt = Target.Offset(0, -2)
neuWert = Target.Value
Case Else:
produkt = ""
End Select

VBA-Editor ...

- ist eine integrierte Entwicklungsumgebung (IDE) für die Programmiersprache V(isual)B(asic for)A(pplication).
- ist in jeder Office-Anwendung vorhanden.
- ist eine eigenständige Anwendung, die in der Taskleiste als Symbol eingeblendet wird.
- bietet die Möglichkeit VBA-Code zu lesen und zu bearbeiten.

Aufbau

- Titelleiste zur Anzeige von Informationen.
- Menüleiste sammelt alle Befehle.
- Symbolleiste zeigt die wichtigsten Befehle als Icon an.
- Projekt-Explorer als Schaltzentrale.
- Eigenschaftenfenster zeigt die Attribute eines gewählten Objekts an.
- Codefenster zeigt den Code zu dem gewählten Objekt an.
- Mit Hilfe des Rahmens um den Editor herum, wird das Fenster vergrößert oder verkleinert.

Titelleiste ...

- befindet sich am oberen Rand der Anwendung.
- hat in der linken Ecke ein Icon, welches die Anwendung symbolisiert. Mit einem Klick auf das Icon wird das dazugehörige Systemmenü geöffnet.
- zeigt den Namen der Excel-Datei sowie des aktiven Moduls an.
- hat am rechten Rand Schaltflächen zum Minimieren, Verkleinern / Maximieren oder Schließen der Anwendung.

Menüleiste ...

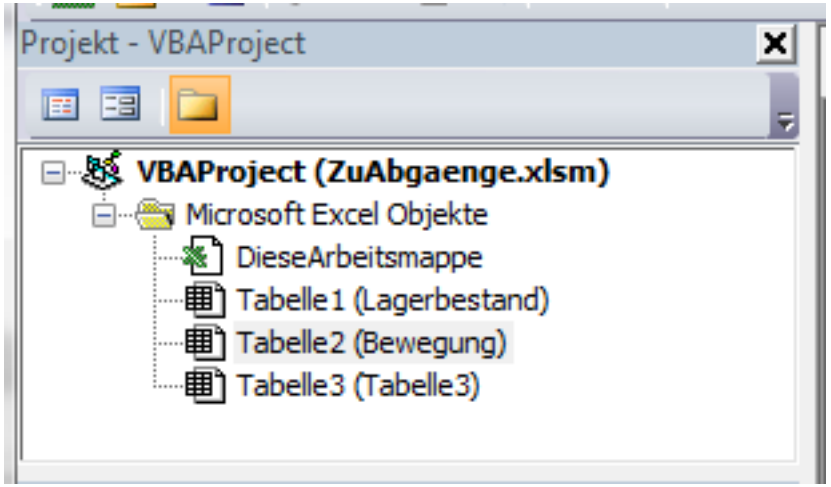
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle zusammengefasst. Der Name der Kategorie gibt einen ersten Hinweis auf die Benutzung der darin enthaltenen Befehle.

Symbolleisten ...

- sammeln häufig genutzte Aktionen zu einem Thema. Die Aktionen werden durch Icons dargestellt.
- beginnen mit der senkrechten gestrichelten Linie am linken Rand. Sobald die Maustaste über diese Linie liegt, kann die Symbolleiste mit Hilfe von Drag (Maustaste gedrückt) & Drop (Maustaste loslassen) verschoben werden.
- werden über das Menü *Ansicht* – *Symbolleiste* ein- oder ausgeblendet.
- haben einen Pfeil nach unten am rechten Rand. Mit einem Klick auf den Pfeil wird ein Menü zum Ein- und Ausblenden von Icons in der Symbolleiste angezeigt.

Projekt-Explorer ...

- ist die Schaltzentrale für die Programmierung einer Excel-Anwendung.
- verwaltet die, zu dem Projekt gehörende Arbeitsmappe sowie die darin enthaltenen Arbeitsblätter.
- zeigt aufgezeichnete Makros in Modulen an.
- ist frei platzierbar.
- kann über das Menü *Ansicht* ein- oder ausgeblendet werden.



Aufbau des Projekt-Explorers

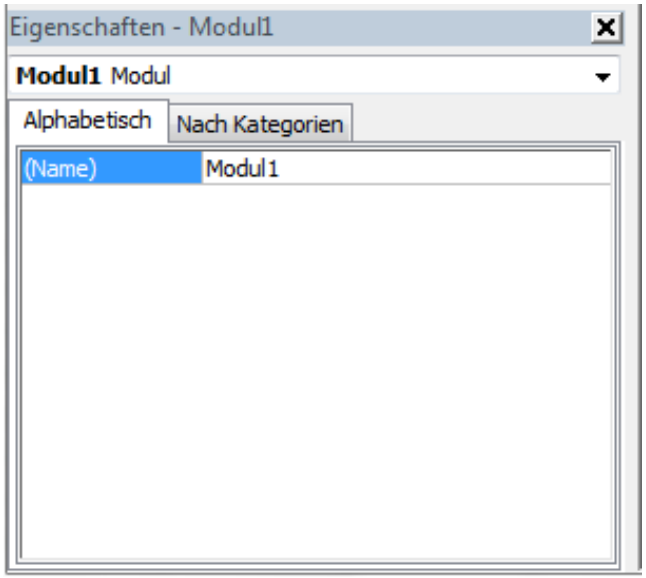
- In der Titelleiste wird die Schließen-Schaltfläche angezeigt.
- Darunter befindet sich die Symbolleiste mit den Icons
  - „Zeige zum gewählten Element den Code an“.
  - „Zeige das passende Objekt zum Code an“.
  - „Sortierung mit Hilfe von Ordnern“.
- Unterhalb der Symbolleiste werden die Module alphabetisch oder in Abhängigkeit ihres Typs sortiert angezeigt.

Module ...

- kapseln Code zu einem Thema.
- sind Container für Code. In dem Container wird eine bestimmte Aufgabe gelöst.
- fassen Programmiercode und Deklarationen zu einem Thema zusammen.
- werden automatisch durch die Aufzeichnung eines Makros angelegt.
- können vom Entwickler oder der Anwendung angelegt werden.

… hinzufügen

- *Einfügen – Modul*.
- Im Eigenschaftenfenster wird der Name des Moduls eingegeben.
- *Datei - … speichern* speichert die Excel-Anwendung und die darin enthaltenen Module.



Modul-Typen ...

- werden mit Hilfe von Ordnern im Projekt-Explorer dargestellt.
- (Microsoft Excel Objekte) enthalten Module, die Code für die Arbeitsmappe oder die darin enthaltenen Arbeitsblätter enthalten
- (Module) enthalten aufgezeichnete Makros oder vom Entwickler selbst geschriebene Prozeduren.

Ordner „Microsoft Excel Objekte“

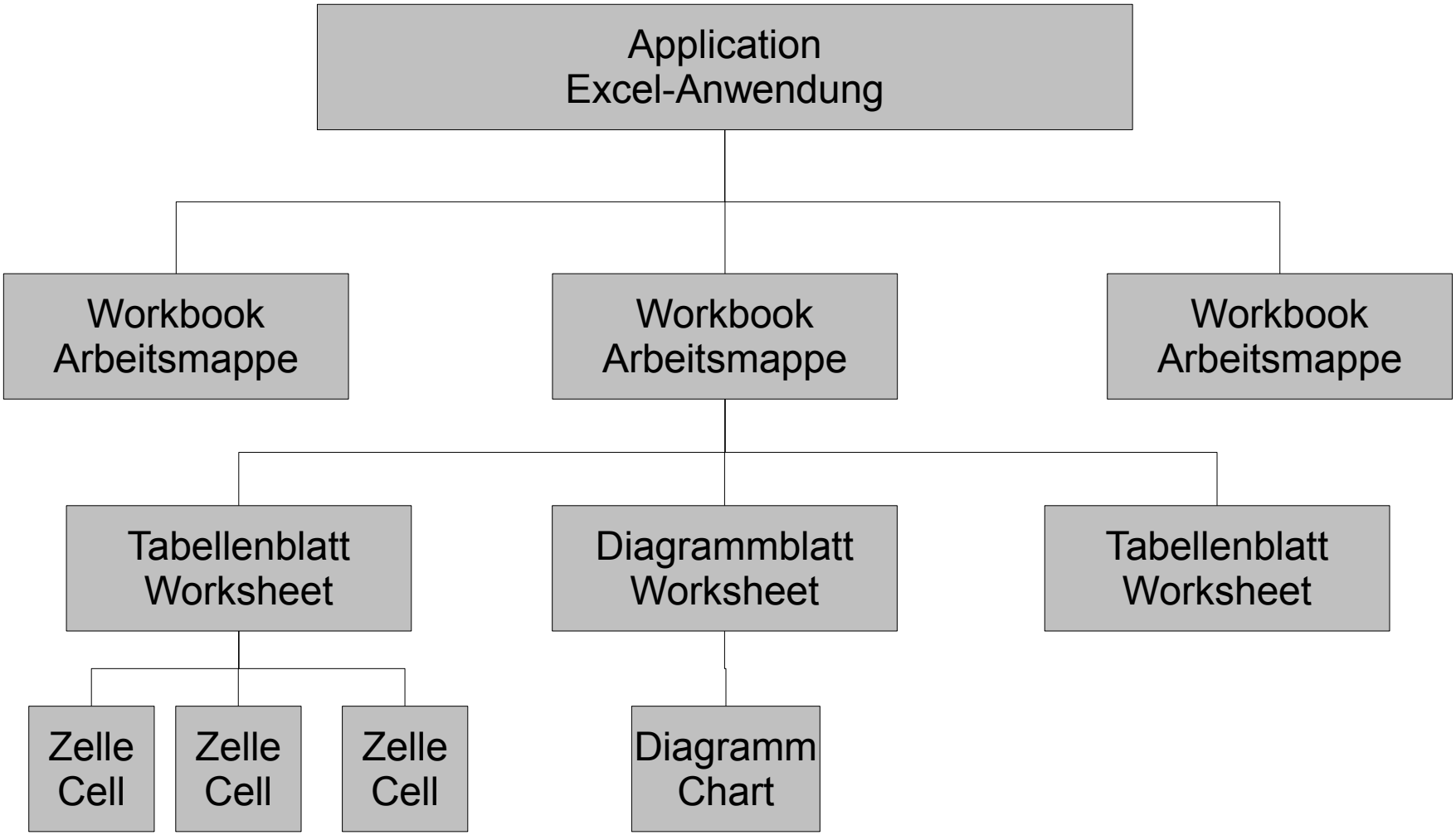
- DieseArbeitsmappe enthält Code, der auf Ereignisse der Arbeitsmappe reagiert. Zum Beispiel kann Code beim Öffnen der Arbeitsmappe ausgeführt werden.
- Jede Arbeitsmappe ...
  - enthält standardmäßig drei Arbeitsblätter
  - muss mindestens ein Arbeitsblatt enthalten.

  Diese Arbeitsblätter werden als Tabelle 1 bis Tabelle n im Projekt-Explorer aufgelistet. In den runden Klammern wird der Name des jeweiligen Tabellenblattes angezeigt.

Objekte ...

- sind zum Beispiel
  - Arbeitsblätter und Zellen in Excel,
  - Dokumente und Absätze in Word.
- beschreiben eindeutig ein bestimmtes Element.
- haben Attribute für die Beschreibung der Eigenschaft.
- haben Methoden, um die Attribute zu verändern.
- reagieren auf Benutzeraktionen mit Hilfe von Ereignisprozeduren.
- werden hierarchisch in einem Modell zusammengefasst.

Excel-Objektmodell



Objektmodelle ...

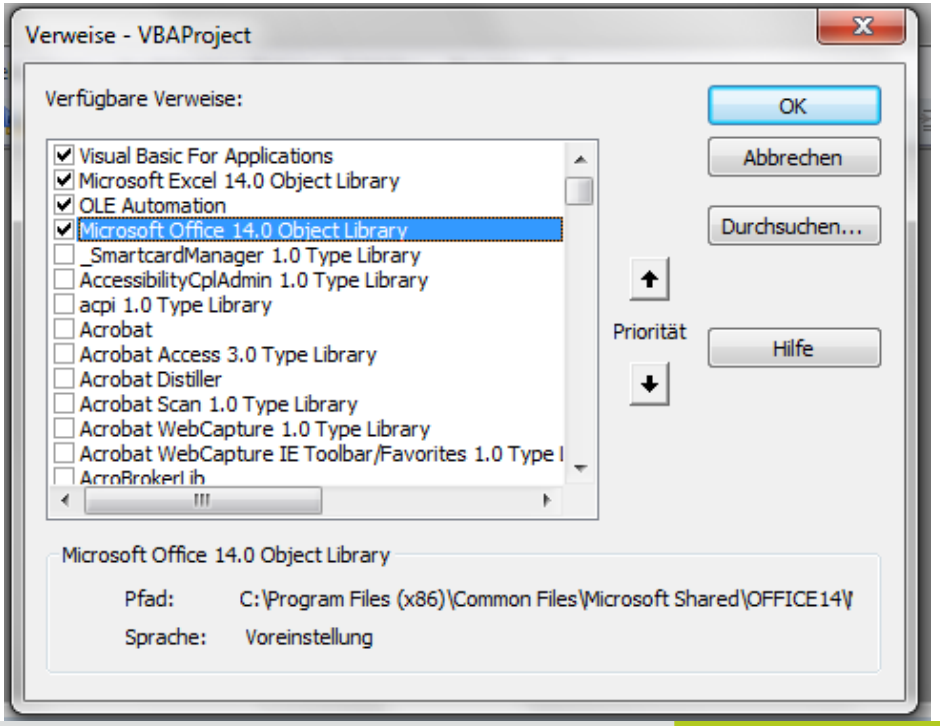
- bilden die Elemente einer Anwendung für VBA nach.
- sind eine Auflistung von Objekten einer Office-Anwendung.
- ordnen Elemente einer Anwendung hierarchisch.
- werden für die Programmierung in Bibliotheken mit der Dateiendung „.olb“ gespeichert.
- können im VBA-Editor mit Hilfe von *Ansicht* – *Objektkatalog* angezeigt werden.

… im Internet

- Office:
  http://msdn.microsoft.com/en-us/library/ff870203.aspx
- Excel:
  http://msdn.microsoft.com/en-us/library/ff846392.aspx
- Word:
  http://msdn.microsoft.com/en-us/library/ff837519.aspx
- PowerPoint:
  http://msdn.microsoft.com/en-us/library/ff743835.aspx
- Outlook:
  http://msdn.microsoft.com/en-us/library/ff870566.aspx

Eingebundene Objektmodelle

- *Extras – Verweise* im VBA-Editor.
- Im oberen Bereich der Tabelle werden alle aktiven Verweise angezeigt. Die Verweise besitzen ein Häkchen im Kontrollkästchen.
- Der Speicherort des ausgewählten Verweises wird am unteren Rand angezeigt.

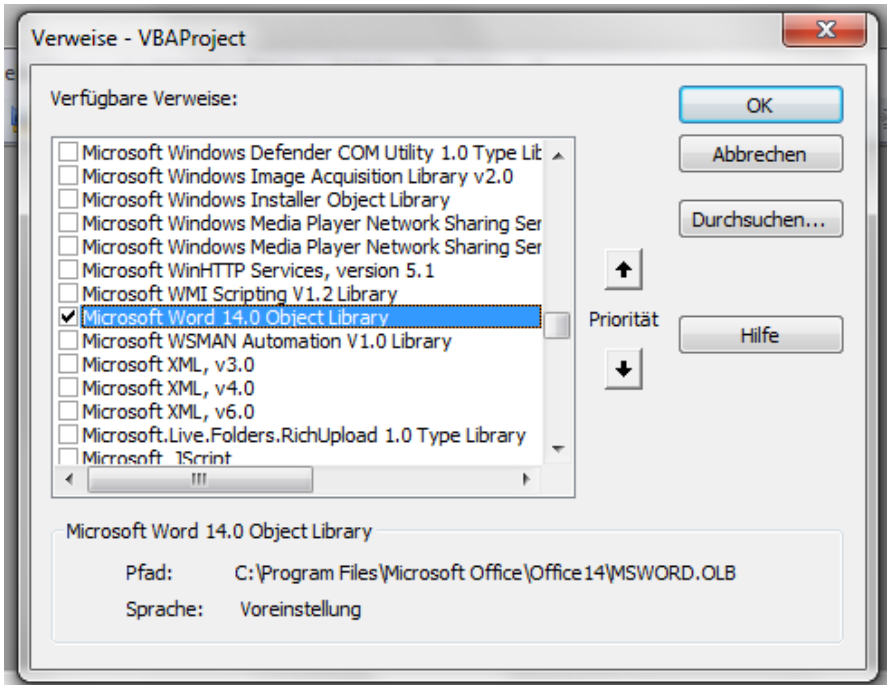


Vorhandene Verweise

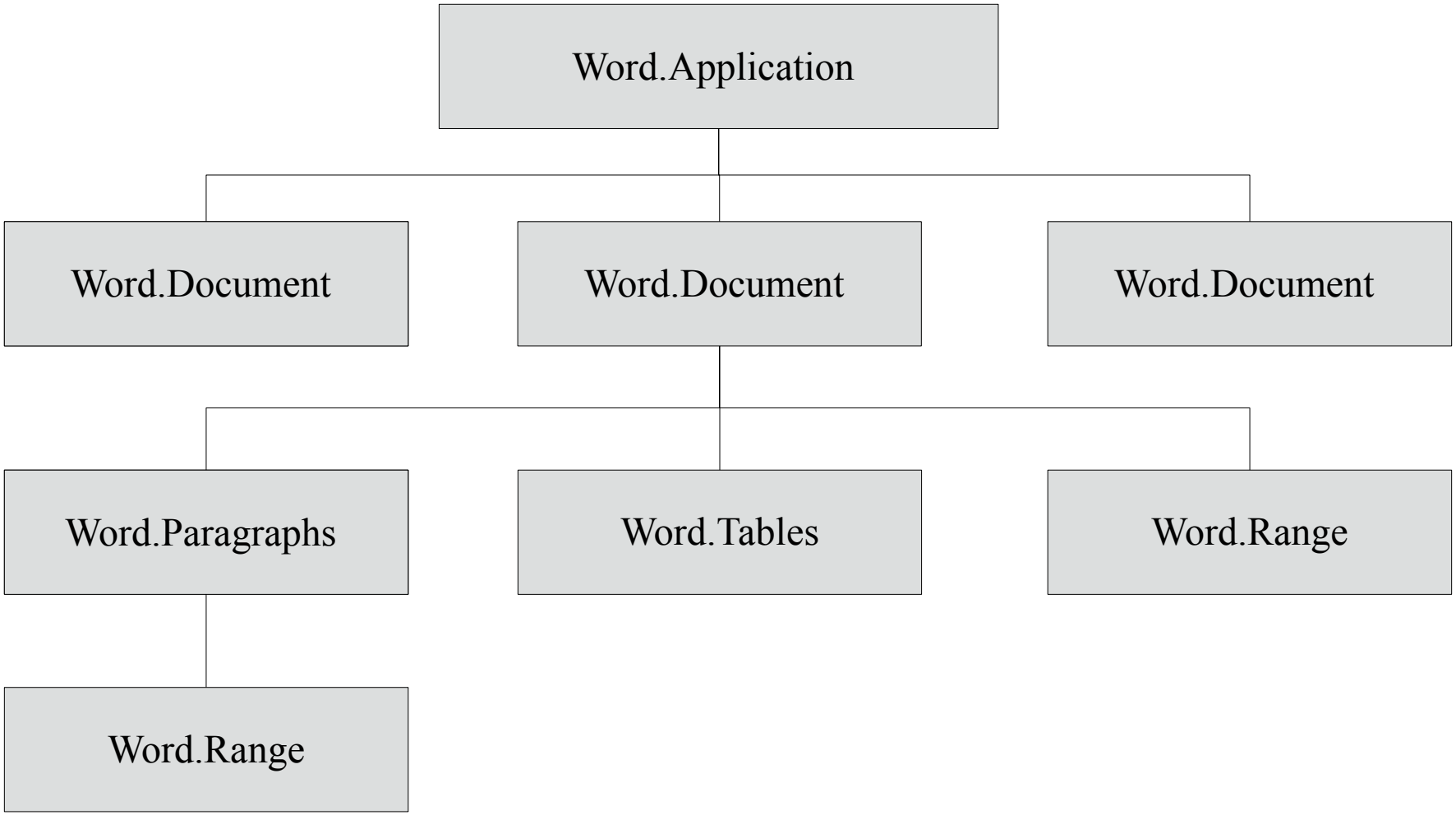
- Visual Basic for Application beschreibt alle Objekte, die die Programmiersprache VBA enthält.
- OLE Automation beschreibt das Einbetten und Verlinken von Objekten wie zum Beispiel die Verknüpfung zu einem Bild.
- Microsoft Office 14.0 Object Library enthält alle Objekte, die alle Office 2010 – Anwendungen gemeinsam nutzen.
- Microsoft Excel 14.0 Object Library beschreibt die Anwendung Excel 2010.

Einbindung eines Objektmodelles

- *Extras* – *Verweise* im VBA-Editor.
- Suche nach Microsoft Word 14.0 Object Library. Hinweis: Die Bibliotheken sind in der Liste alphabetisch sortiert.
- Mit einem Mausklick auf das Kontrollkästchen links von der Bezeichnung wird der Verweis aktiviert.
- Die Schaltfläche *OK* schließt den Dialog.



Objektmodell „Word“



Eigenen Code für den Export erzeugen

- Das Menüband Entwicklertools ist aktiv.
- Mit Hilfe des Symbols *Visual Basic* wird der VBA-Editor geöffnet.
- Mit Hilfe des Menüs *Einfügen* – *Modul* wird ein neue Datei angelegt. Das Modul wird im Codefenster angezeigt.
- In diesem Modul wird der Code mit Hilfe der Tastatur eingegeben.

Eingabe einer Prozedur in das Codefenster

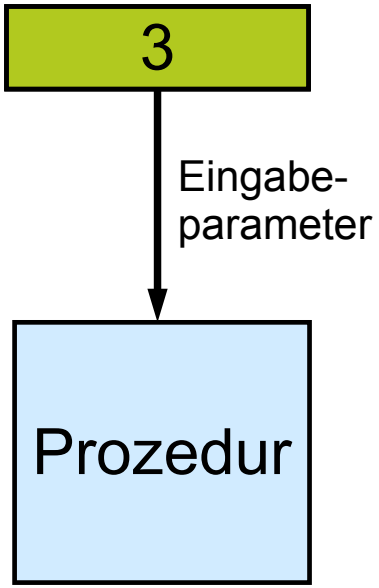
```
Sub Word_Absatz()


End Sub
```

- Durch das Schlüsselwort Sub wird der Beginn einer Prozedur gekennzeichnet.
- Anschließend folgt ein selbsterklärender, benutzerdefinierter Name.
- Dem Namen folgen leere runde Klammern. Der Prozedur werden keine Werte übergeben.
- Die Prozedur endet mit den Schlüsselwörtern End Sub. Diese Schlüsselwörter werden automatisch von VBA gesetzt.

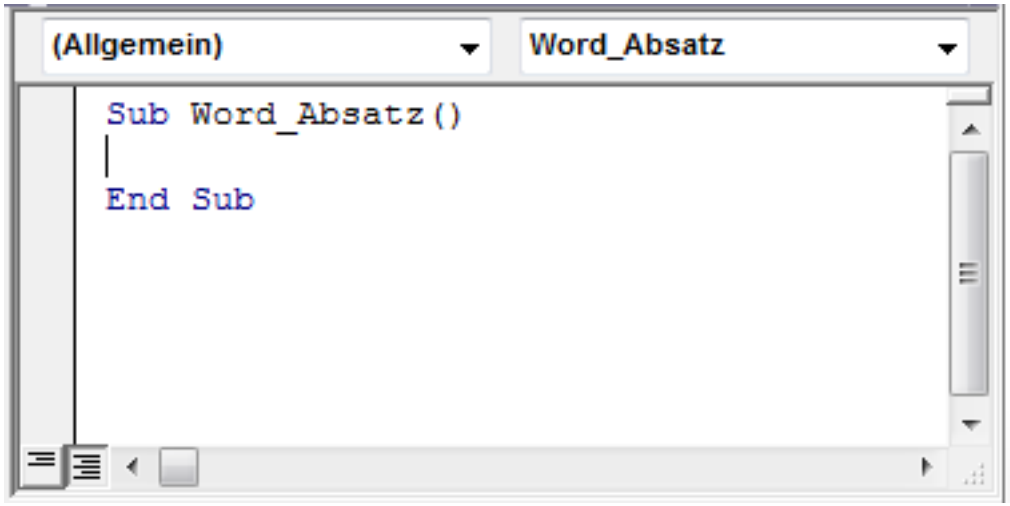
Arbeitsweise einer Prozedur

- Der Prozedur können Eingabeparameter (Argumente) übergeben werden.
- Diese Eingabeparameter werden in der Prozedur verarbeitet.
- Die Verarbeitung selber ist dem Aufrufer aber nicht bekannt. Der Nutzer kennt nur die Schnittstelle (den Prozedurkopf) nach außen.



Anzeige im Codefenster

- Die Schlüsselwörter aus VBA werden standardmäßig mit blauer Schrift angezeigt.
- Im Codefenster wird das Modul mit Hilfe der Prozeduren in kleine Code-Abschnitte unterteilt.



Arbeitsschritte eingeben

```
Sub Word_Absatz()


End Sub
```

- Mit einem Mausklick wird die Einfügemarke zwischen den Schlüsselwörtern Sub und End Sub gesetzt.
- Mit Hilfe der Tastatur werden dort die benötigten Arbeitsschritte zeilenweise eingegeben.

Verweise auf Objekte in VBA

```
Sub Word_Absatz()

    Dim datenblatt As Worksheet
    Dim zelle As Range

    Dim wdApp As Word.Application
    Dim wdDoc As Word.Document
    Dim wdParagraph As Word.Paragraph

End Sub
```

Objektvariablen ...

- verweisen auf ein bestimmtes Objekt in einer Office-Anwendung.
- enthalten eine Referenz auf die Office-Anwendung selbst oder auf Elemente in einem Word-Dokument, Excel-Arbeitsmappe etc..
- werden für Objekte definiert, die häufig in einem Programm genutzt werden.

… definieren

```
Dim datenblatt As Worksheet
Dim wdApp As Word.Application
```

- Jede Objektvariable hat einen eindeutigen Namen (wdApp, datenblatt). Der Name sollte über die Art des Verweises Auskunft geben. Die Bezeichnung ist frei wählbar.
- Jede Objektvariable verweist auf einen bestimmten Typ von Objekt. Mit Hilfe des Schlüsselwortes As wird der Variablen ein Typ zugewiesen.
- Mit Hilfe des Schlüsselwortes Dim wird der Zugriff auf Variablen geregelt. Auf die Variablen kann nur innerhalb der Prozedur Word_Absatz zugegriffen werden. Diese Prozedur ist Besitzer dieser Variablen.

… zurücksetzen

```
Set datenblatt = Nothing
```

- Das Schlüsselwort `Set` leitet eine Zuweisung an eine Objektvariable ein.
- Mit Hilfe des Gleichheitszeichens wird der Objektvariablen ein Verweis zugewiesen.
- Der Wert `Nothing` symbolisiert die leere Objektvariable. Die Variable verweist auf kein Objekt.

Platzhalter nutzen

```
Sub Word_Absatz()

    Dim dateipfad As String
    Dim txtZeile As String
    Dim int_zeileBis As Integer
    Dim int_zeile As Integer

End Sub
```

Variablen ...

- sind Platzhalter für bestimmte Werte. Als Werte können Zahlen, Datums- und Zeitwerte sowie Texte gespeichert werden.
- sind von einem bestimmten Standard-Datentyp. Die verschiedenen Datentypen werden im Internet auf der Seite http://msdn.microsoft.com/en-us/library/gg278937.aspx aufgelistet.
- haben einen eindeutigen Namen. Der Name sollte selbsterklärend sein.

… definieren

```
Dim dateipfad As String
```

- Jede Variable hat einen eindeutigen Namen (dateipfad). Der Name sollte über den zu speichernden Wert Auskunft geben. Die Bezeichnung ist frei wählbar.
- Jede Variable verweist auf einen bestimmten Datentyp. Mit Hilfe des Schlüsselwortes As wird der Variablen ein Typ zugewiesen. In diesem Beispiel wird eine Variable vom Typ String (Text) erzeugt.
- Auf die Variablen kann nur innerhalb der Prozedur Word_Absatz zugegriffen werden (Dim). Diese Prozedur ist Besitzer dieser Variablen.

… zuweisen

```
dateipfad = ThisWorkbook.Path & "\" & "listeKunde" & Day(Date) & ".docx"
```

- Mit Hilfe des Gleichheitszeichens wird einer Variable ein Wert zugewiesen. Der Wert kann durch ein Ausdruck berechnet werden.
- In diesem Beispiel wird der Variablen dateipfad ein Text zugewiesen. Dieser Text setzt sich aus verschiedenen Elementen zusammen. Die Elemente werden mit Hilfe des kaufmännischen Unds verknüpft.

Datentypen für Ganzzahlen

	Speicherbedarf in Bytes	Datenbereich
As Byte	1	0 - 255
As Integer	2	-32.768 - +32.767
As Long	4	-2.147.483.648 - +2.147.483.647
As Boolean	2	0 (falsch, false) <> 0 (wahr, true)

Datentypen für Zeichenfolgen (Text)

	Länge
As String	Variable Länge.
As String * 5	Die Länge des Strings wird auf eine bestimmte Anzahl eingeschränkt. In diesem Beispiel besteht die Zeichenfolge aus fünf Zeichen.

Angabe eines Speicherortes und Dateinamens

```
dateipfad = ThisWorkbook.Path & "\" & "listeKunde" & Day(Date) & ".docx"
```

- Der Pfad zu der aktuell, mit dem Code verknüpften Arbeitsmappe (ThisWorkbook.Path).
- Einen Dateinamen als feststehenden Begriff. Text wird in VBA immer in Anführungsstrichen gesetzt.
- Den aktuellen Tag berechnet aus dem Systemdatum (Date).
- Die Dateiendung.

Excel-Objekte in VBA nutzen

```
Dim datenblatt As Worksheet
 Dim zelle As Range


Set datenblatt = ThisWorkbook.Worksheets("Kunde")
 datenblatt.Activate

 int_zeileBis = datenblatt.UsedRange.SpecialCells(xlCellTypeLastCell).Row

 For int_zeile = 1 To int_zeileBis
    txtZeile = ""

    For Each zelle In datenblatt.UsedRange.Rows(int_zeile).Cells
       txtZeile = txtZeile & vbTab & zelle.Value
    Next zelle
 Next int_zeile
```

Objektvariablen definieren

```
Dim datenblatt As Worksheet
Dim zelle As Range
```

- Worksheet kann einen Verweis auf ein Tabellenblatt in einer Arbeitsmappe von Excel speichern.
- Range verweist auf eine einzelne Zelle, eine Spalte, eine Zeile oder einen Zellbreich.

Arbeitsmappe

- ThisWorkbook ist ein Verweis auf die, zum Code gehörenden Arbeitsmappe.
- ActiveWorkbook ist ein Verweis auf die aktuelle Arbeitsmappe.
- Application.Workbooks("Name") verweist auf eine geöffnete Arbeitsmappe in der Excel-Anwendung. Der Name entspricht häufig dem Dateinamen der Arbeitsmappe.

Tabellenblätter

- ThisWorkbook.ActiveSheet ist ein Verweis auf das aktive Tabellenblatt in der, an den Code gebundenen Arbeitsmappe.
- ThisWorkbook.Worksheets("Kunde") verweist auf ein Tabellenblatt mit dem Namen Kunde. In den runden Klammern wird ein Index in Form eines Textes verwendet. Ein Text beginnt und endet immer mit den Anführungszeichen. Als Index wird die Bezeichnung auf dem Tabellenreiter genutzt.

… aktivieren

```
ThisWorkbook.Worksheets("Kunde").Activate
```

- Die Methode .Activate aktiviert ein Tabellenblatt.
- Die Angabe des Tabellenblattes kann durch eine Objektvariable ersetzt werden.
- Das Objekt und deren Methode werden durch einen Punkt getrennt.

Zellen

- datenblatt.Range("A1") definiert einen bestimmten Zellbereich. "A1" verweist auf die Zelle A1. "A1:B5" verweist auf den zusammenhängenden Zellbereich A1 bis B5. "5:5" verweist auf die Zeile 5.
- datenblatt.UsedRange definiert den genutzten Zellbereich in einem Tabellenblatt.
- datenblatt.UsedRange.SpecialCells(xlCellTypeLastCell) verweist auf bestimmte Zellen in einem Bereich. In diesem Beispiel wird auf die letzte Zelle im benutzten Bereich verwiesen. Die Art der Zelle wird durch eine definierte Konstante in den runden Klammern angegeben.

Word.Application in VBA nutzen

```
Sub Word_Absatz()
    Dim wdApp As Word.Application
    Dim wdDoc As Word.Document

    Dim dateipfad As String

    dateipfad = ThisWorkbook.Path & "\" & "listeKunde" & Day(Date) & ".docx"
    Set wdApp = CreateObject("Word.Application")
    wdApp.Visible = True

    wdApp.Quit
End Sub
```

Verweise auf die Word-Anwendung und -Dokument

```
Dim wdApp As Word.Application
Dim wdDoc As Word.Document
```

- Application verweist auf eine Anwendung.
- Document verweist auf Dokument.
- Alle definierten Objekttypen beziehen sich auf Word. Die dazugehörige Anwendung und das darin enthaltene Objekt werden mit einem Punkt verbunden.

Verweise auf Elemente in einem Word-Dokument

Dim wdParagraph As Word.Paragraph

Dim wdTable As Word.Table

Dim wdBookmark As Word.Bookmark

- Paragraph verweist auf einen Absatz in einem Dokument.
- Table verweist auf eine Tabelle in einem Dokument.
- Bookmark speichert ein Verweis auf eine Textmarke in einem Dokument.
- Alle definierten Objekttypen beziehen sich auf Word. Die dazugehörige Anwendung und das darin enthaltene Objekt werden mit einem Punkt verbunden.

Verweise auf ein Bereich in einem Word-Dokument

Dim wdRange As Word.Range

- Als Bereich kann ein Absatz, eine Tabellenzelle etc. genutzt werden.
- Ein Bereich hat einen definierten Start- und Endpunkt.
- Bereiche können beliebig oft in einem Word-Dokument vorkommen.
- Alle definierten Objekttypen beziehen sich auf Word. Die dazugehörige Anwendung und das darin enthaltene Objekt werden mit einem Punkt verbunden.

Zugriff auf die Word-Anwendung

```
Set wdApp = CreateObject("Word.Application")
```

- Mit Hilfe des Schlüsselwortes `Set` werden Zuweisungen zu einer Objektvariablen eingeleitet.
- Der Operator „Gleichheitszeichen“ wird ein Verweis auf die Word-Anwendung übergeben.
- Die Funktion `CreateObject()` erzeugt ein ActiveX-Objekt. ActiveX-Objekte können nur unter dem Betriebssystem Windows erzeugt. In diesem Beispiel wird eine Word-Anwendung in Excel eingebettet. Die Angabe der zu erstellenden Anwendung wird als Text übergeben.

Anwendung einblenden

```
wdApp.Visible = True
```

- Die Anwendung hat die Eigenschaft „Sichtbar“.
- Die Eigenschaft wird auf `True` (wahr) gesetzt. Word wird am Bildschirm angezeigt.
- Der Punkt wird als Verbindung zwischen der Eigenschaft und seinem Objekt genutzt.

Anwendung schließen

wdApp.Quit

- Mit Hilfe der Methode Quit wird die Anwendung geschlossen.
- Die Methode wird mit dem dazugehörigen Objekt durch ein Punkt verbunden.

Word-Dokumente erstellen und speichern

```
Sub Word_Absatz()
    Dim wdApp As Word.Application
    Dim wdDoc As Word.Document

    Dim dateipfad As String

    dateipfad = ThisWorkbook.Path & "\" & "listeKunde" & Day(Date) & ".docx"
    Set wdApp = CreateObject("Word.Application")
    wdApp.Visible = True
    Set wdDoc = wdApp.Documents.Add(Visible:=True)

    wdDoc.SaveAs Filename:=dateipfad
    wdDoc.Close
    wdApp.Quit
End Sub
```

Neues Word-Dokument erstellen

```
Set wdDoc = wdApp.Documents.Add(Visible:=True)
```

- Mit Hilfe des Schlüsselwortes Set wird die Zuweisung an die Objektvariable eingeleitet.
- Der Operator „Gleichheitszeichen“ weist der Variablen entsprechend des Objekttyps einen Referenz zu.
- Die Anwendung besitzt die Liste Documents. In dieser Liste werden alle, in der Anwendung geöffneten Word-Dokumente aufgelistet.
- Der Liste Documents wird mit Hilfe der Methode .Add ein neues Dokument hinzugefügt. Das Dokument wird am Bildschirm angezeigt (Visible:=True).

Benannte Argumente

.Add(Visible:=True)

- Methoden können in den runden Klammern Parameter übergeben werden. Häufig werden dafür benannte Argumente genutzt.
- Alle Argumente der Methode werden nach Eingabe der runden Klammern in einem gelben Erklärfenster angezeigt.
- Die Namen der Argumente können nicht verändert werden.
- Mit Hilfe des zusammengesetzten Operators := wird dem Argument ein Wert zugewiesen. In diesem Beispiel wird der boolsche Wert True (wahr) genutzt.

Dokument speichern

wdDoc.SaveAs Filename:=dateipfad

- Mit Hilfe der Methode Save As wird ein neues Dokument erstmalig gespeichert.
- Dem Argument Filename wird der Name sowie der Speicherort des neuen Dokuments übergeben.

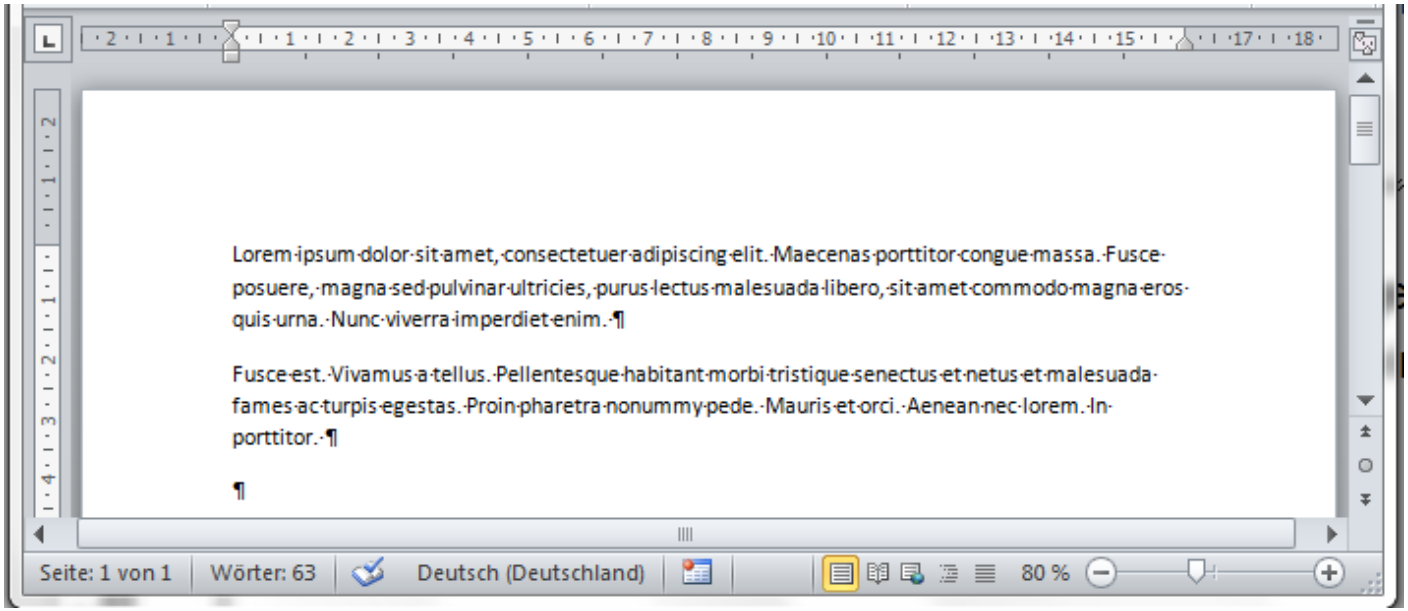
Dokument schließen

wdDoc.Close

- Mit Hilfe der Methode Close wird das Dokument geschlossen.
- Falls das Dokument nicht gespeichert ist, wird nachgefragt, ob alle Änderungen verworfen werden sollen.

Absätze ...

- gliedern größere Texte.
- werden voneinander durch Leerzeichen oder Einzüge getrennt.
- enden mit dem Absatzzeichen. Das Absatzzeichen wird mit Hilfe von <RETURN> erzeugt.



Absatzmarken erstellen

```
Set wdDoc = wdApp.Documents.Add(Visible:=True)

For int_zeile = 1 To int_zeileBis
      wdDoc.Paragraphs.Add
      Set wdParagraph = wdDoc.Paragraphs(wdDoc.Paragraphs.Count)

      txtZeile = ""
      For Each zelle In datenblatt.UsedRange.Rows(int_zeile).Cells
         txtZeile = txtZeile & vbTab & zelle.Value
      Next zelle

      wdParagraph.Range.Text = txtZeile
Next int_zeile
```

Paragraphs ...

- ist eine Sammlung von Absatzmarken. Objektsammlung enden immer den Buchstaben „s“.
- .Count gibt die Anzahl der Absatzmarken in dem angegebenen Dokument zurück.
- .Add fügt eine Absatzmarke der Sammlung hinzu.
- Jedes Element in der Sammlung kann mit Hilfe eines Indizes eindeutig identifiziert werden. Der Index folgt dem Namen der Sammlung, begrenzt durch runde Klammern. Die Anweisung wdDoc.Paragraphs(wdDoc.Paragraphs.Count) bezieht sich auf den letzten Absatz. Als Index wird eine Ganzzahl genutzt.

Text hinzufügen

```
Set wdDoc = wdApp.Documents.Add(Visible:=True)

For int_zeile = 1 To int_zeileBis
      wdDoc.Paragraphs.Add
      Set wdParagraph = wdDoc.Paragraphs(wdDoc.Paragraphs.Count)

      txtZeile = ""
      For Each zelle In datenblatt.UsedRange.Rows(int_zeile).Cells
         txtZeile = txtZeile & vbTab & zelle.Value
      Next zelle

      wdParagraph.Range.Text = txtZeile
Next int_zeile
```

Erläuterung

wdDoc.Paragraphs(wdDoc.Paragraphs.Count).Range.Text

wdParagraph.Range.Text

- .Range bezieht sich hier auf eine Absatzmarke. In dem Objekt wird auf einen bestimmten Absatz im Dokument genommen.
- .Text bezieht sich auf den Text in dem angegebenen Bereich. In diesem Beispiel wird der Text eines Absatzes verändert.

Text und Absätze hinzufügen

```
Set wdDoc = wdApp.Documents.Add(Visible:=True)

For int_zeile = 1 To int_zeileBis
        txtZeile = ""

      For Each zelle In datenblatt.UsedRange.Rows(int_zeile).Cells
         txtZeile = txtZeile & vbTab & zelle.Value
      Next zelle

      wdDoc.Content.InsertAfter txtZeile
      wdDoc.Content.InsertParagraphAfter
Next int_zeile
```

Erläuterung

- wdDoc.Content bezieht sich auf den Inhalt des gesamten Word-Dokuments.
- Die Methode .InsertAfter fügt an das Ende des angegebenen Bereichs Text ein. In diesem Beispiel wird am Ende des Dokuments Text eingefügt. Die Methode .InsertBefore würde Text vor dem angegebenen Bereich einfügen.
- Die Methode .InsertParagraphAfter fügt an das Ende des angegebenen Bereichs eine Absatzmarke ein. In diesem Beispiel wird am Ende des Dokuments eine Absatzmarke eingefügt. Die Methode .InsertParagraphBefore würde eine Absatzmarke vor dem angegebenen Bereich einfügen.

Tabelle hinzufügen und formatieren

```
Sub Word_Tabelle()
    Dim wdDoc As Word.Document
    Dim wdTable As Word.Table
    Dim wdRange As Word.Range

    Set wdDoc = wdApp.Documents.Add(Visible:=True)

    Set wdTable = wdDoc.Tables.Add(wdDoc.Range(0), int_zeileBis, int_spalteBis)
    wdTable.Borders.InsideLineStyle = wdLineStyleSingle
    wdTable.Borders.OutsideLineStyle = wdLineStyleSingle

    wdDoc.SaveAs Filename:=dateipfad
    wdDoc.Close
End Sub
```

Neue Tabelle erstellen

```
Set wdTable = wdDoc.Tables.Add(wdDoc.Range(0),
                               int_zeileBis, int_spalteBis)
```

- Die Sammlung Tables enthält alle Tabellen in einem Dokument.
- Mit Hilfe der Methode .Add wird ein neues Objekt der Sammlung hinzugefügt.
- Die Methode benötigt folgende Eingabeparameter:
  - An welcher Position wird die Tabelle eingefügt? In diesem Beispiel wird die Tabelle an den Anfang des Dokuments eingefügt.
  - Wie viele Zeilen werden benötigt?
  - Wie viele Spalten werden benötigt?

Tabelle formatieren

- .Borders legt das Aussehen der Rahmen der Tabelle fest.
  - InsideLineStyle und InsideColor legt die Rahmenart und -farbe innerhalb der Tabelle fest.
  - OutsideLineStyle und OutsideColor legt die Rahmenart und -farbe der Tabellenumrandung fest.
- Der Eigenschaft .Style kann eine Formatvorlage für Tabellen zugewiesen werden.
- Der Eigenschaft .Shading.BackgroundPatternColor legt eine Hintergrundfarbe für die Tabelle fest.

Zellen nutzen

```
Set wdRange = wdTable.Cell(int_zeile, int_spalte).Range

With wdRange
        .Font.Name = "Arial"
        .Font.Bold = False
        .Font.Size = "12"
        .ParagraphFormat.Alignment = wdAlignParagraphLeft
End With
```

Zellen in einer Tabelle

```
Set wdRange = wdTable.Cell(int_zeile, int_spalte).Range
```

- .Cell ist ein Verweis eine bestimmte Zelle in einer Tabelle. In den runden Klammern wird die Zeile und die Spalte der Zelle übergeben.
- .Range bezieht sich auf die vorhergehende Zelle.

… formatieren

- .Font legt die genutzte Schriftart in einer Zelle fest.
  - .Name → Schriftart.
  - .Size → Schriftgröße
  - .Bold → Fettschrift. Eigenschaften wie kursiv etc. sind vorhanden.
- .ParagraphFormat legt die Formatierungen für einen Absatz in einer Zelle fest.
  - .Alignment → Ausrichtung des Absatzes.

Textmarken ...

- arbeiten ähnlich wie Lesezeichen in einem Internet-Browser.
- verweisen auf bestimmte Texte in einem Dokument.
- einfügen: *Textmarke* in der Kategorie Hyperlinks auf dem Menüband Einfügen.

Mit Textmarken arbeiten

```
For Each wdBookmark In wdDoc.Bookmarks
    Set wdRange = wdBookmark.Range



    Select Case wdBookmark.Name
        Case "marke_Datum":
                wdRange.Text = "Hannover, d. " & Format(Date, "dd.mm.yyyy")
    End Select

Next wdBookmark
```

Erläuterung

- Die Sammlung .Bookmarks enthält alle Textmarken in einem Dokument.
- Die Eigenschaft .Name gibt über die Bezeichnung der Textmarke Auskunft.

For-Each-Schleife

For Each wdBookmark In wdDoc.Bookmarks

For Each [Element] In [Liste/Sammlung]

- Der Schleifenkopf beginnt mit dem Schlüsselwort For Each.
- Dem Schlüsselwort folgt ein Platzhalter für jedes Element in einer Liste oder Sammlung. Falls keine Objektvariable genutzt wird, muss die Variable vom Typ Variant sein.
- Dem Platzhalter folgt das Schlüsselwort In, dem der Name der Liste folgt. Die angegebene Liste muss definiert sein.
- Das Schlüsselwort Next leitet den nächsten Schleifendurchlauf ein.

Auswahlanweisung

```
Select Case variable
   Case wert:
         Anweisung
   Case wert, wert, wert:
         Anweisung
   Case min To max:
         Anweisung
   Case Is operator wert:
         Anweisung
End Select
```

```
Select Case wdBookmark.Name
    Case "marke_Datum":
          wdRange.Text =  Date
End Select
```

Erläuterung

- Die Auswahlanweisung bietet für ein Element verschiedene Möglichkeiten kann.
- Eine Auswahlanweisung beginnt mit Select Case und endet mit End Select. Das zu untersuchende Elemente wird im Kopf Select Case element angegeben. Für element werden verschiedene Werte untersucht.
- Das Schlüsselwort Case leitet eine Auswahlmöglichkeit ein. Mit Hilfe des Doppelpunktes wird die Angabe beendet. In der nächsten Zeile folgen die dazugehörigen Anweisungen.
- Mit Hilfe einer Auswahlanweisung können Texte und Zahlen untersucht werden.

Text formatieren

Format(Date, "dd.mm.yyyy")

- Mit Hilfe der Funktion Format wird ein Text für die Ausgabe formatiert.
- In den runden Klammern werden der Funktion der zu formatierende Text und ein Formatstring übergeben.
- Der Formatstring wird durch Anführungszeichen begrenzt. Auf folgender Internetseite finden Sie Beispiele für Formatstrings: http://office.microsoft.com/de-de/access-help/format-funktion-HA001228839.aspx

Existiert die Textmarke?

```
If wdDoc.Bookmarks.Exists("marke_Datum") Then
    Set wdRange = wdDoc.Bookmarks("marke_Datum").Range
    wdRange.Text = "Hannover, d. " & Format(Date, "dd.mm.yyyy")
End If
```

- Die Methode .Exists gibt den Wert True (Wahr) zurück, wenn der Name vorhanden ist.
- Der gesuchte Name wird in den runden Klammern als Text übergeben.

Bedingte Anweisung

```
If Bedingung = True  Then
    ' Anwendung
End If
```

- Die Bedingung beginnt mit dem Kopf If (Bedingung) Then und endet mit dem Schlüsselwort End If.
- Bedingungen sind Ausdrücke, die einen boolschen Wert zurückliefern.
- Wenn If die Bedingung wahr ist, dann Then führe die Anweisungen aus.
- Wenn die Bedingung nicht wahr ist, werden die Anweisungen nicht ausgeführt.

Textmarken auf Tabellen

```
If wdDoc.Bookmarks.Exists("marke_Empfaenger") Then
        Set wdRange = wdDoc.Bookmarks("marke_Empfaenger").Range
        Set infoEmpfaenger = zelle
        kundeName = Split(infoEmpfaenger.Value, " ")

        If wdRange.Tables.Count > 0 Then
            wdRange.Tables(1).Cell(1, 2).Range.Text = kundeName(0)
            wdRange.Tables(1).Cell(2, 2).Range.Text = kundeName(1)
      End If
End If
```