R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Export von Diagrammen nach Powerpoint

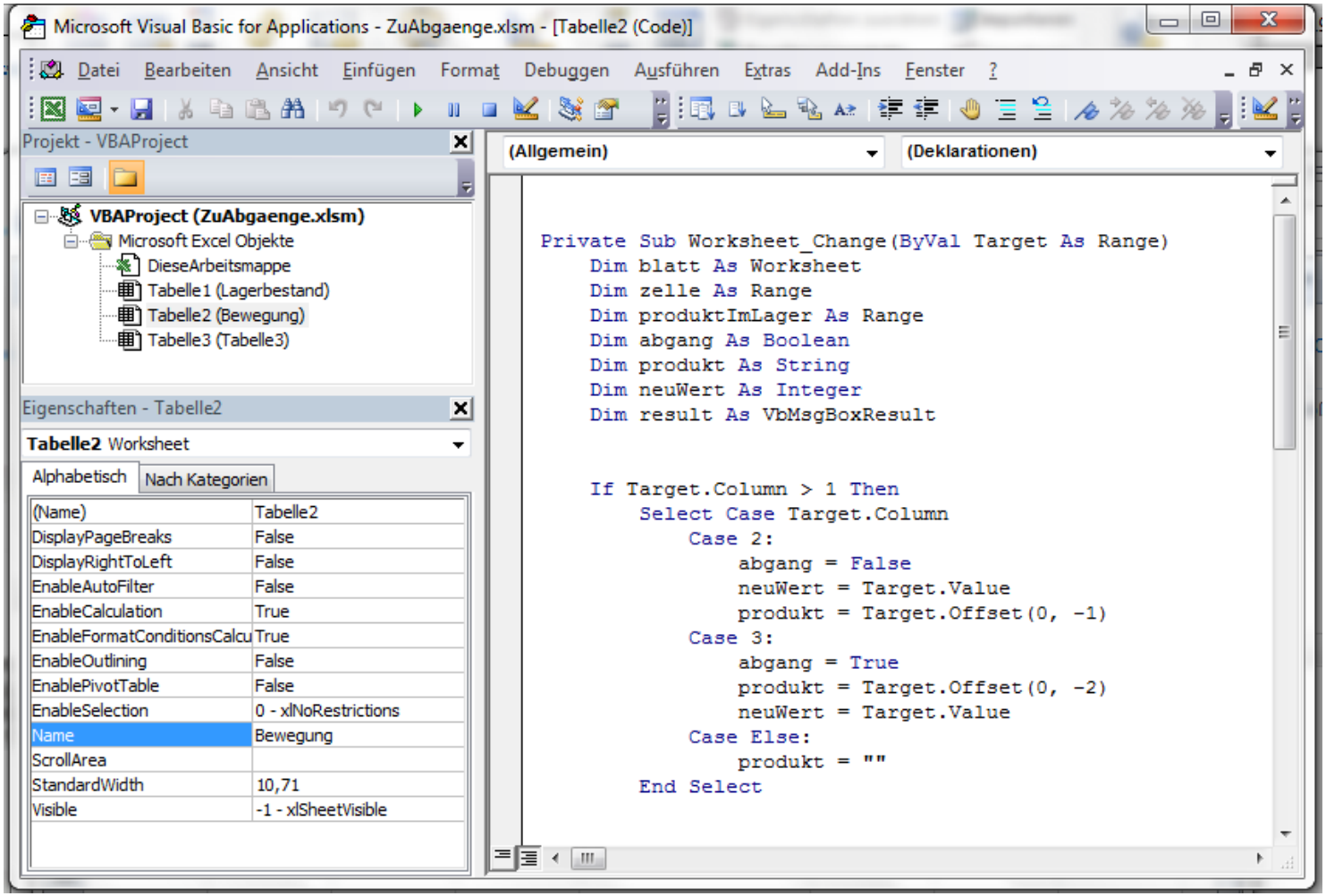
Export nach PowerPoint ...

- ist nur über die Zwischenablage möglich. Hinweis: In einem Excel-Makro werden Aktionen in PowerPoint nicht aufgezeichnet.
- 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 ...

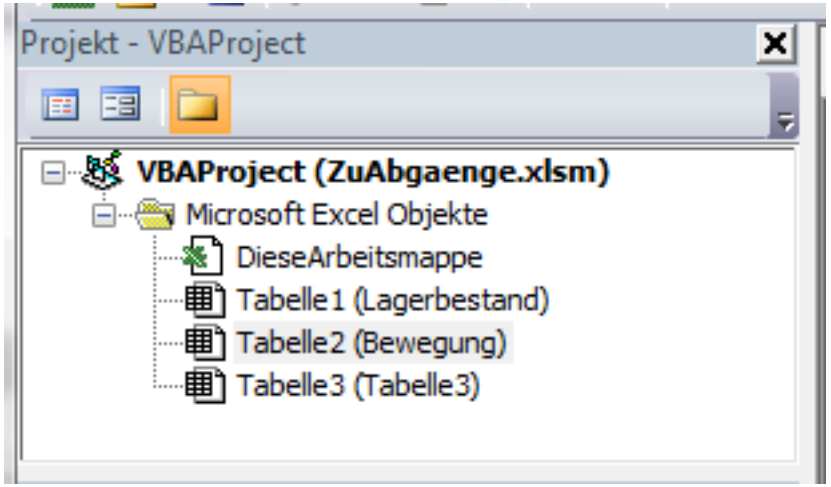
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle 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

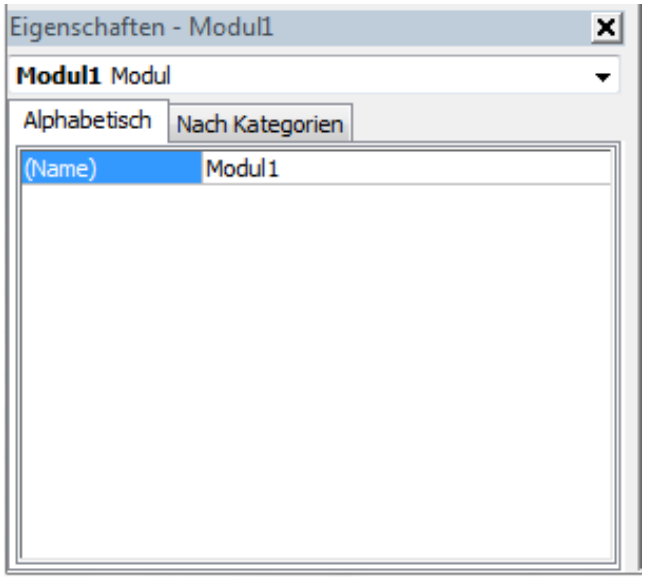
- In der Titelleiste wird die Schließen-Schaltfläche angezeigt.
- Darunter befindet sich die Symbolleiste mit den Icons
  - „Zeige zum gewählten Element den Code an“.
  - „Zeige das passende Objekt zum Code an“.
  - „Sortierung mit Hilfe von Ordnern“.
- Unterhalb der Symbolleiste werden die Module alphabetisch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

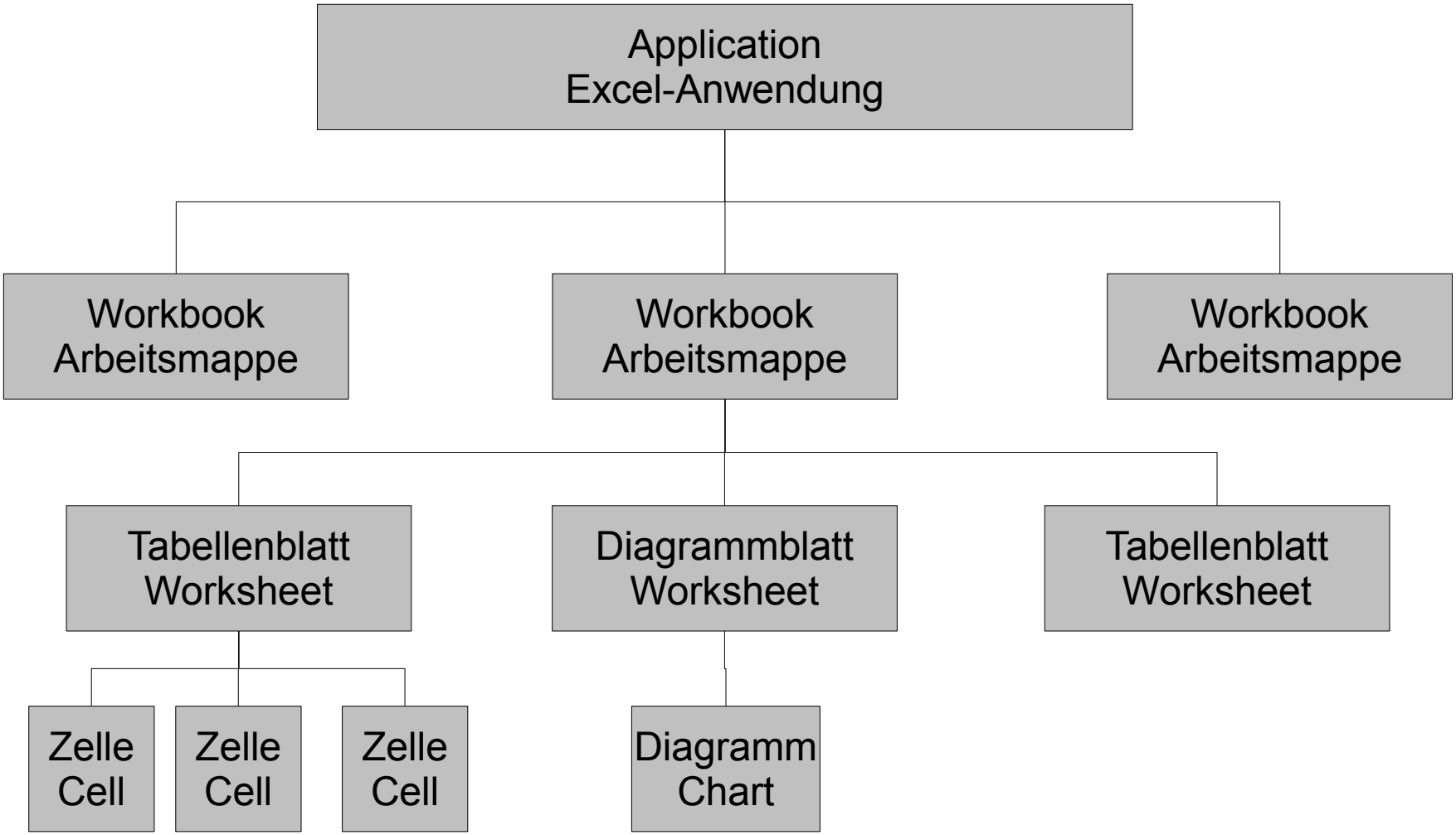
- DieseArbeitsmappe enthält Code, der auf Ereignisse der Arbeitsmappe reagiert. Zum Beispiel kann Code beim Öffnen der Arbeitsmappe ausgeführt werden.
- Jede Arbeitsmappe ...
  - enthält standardmäßig drei Arbeitsblätter
  - muss mindestens ein Arbeitsblatt enthalten.

  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 Folien in PowerPoint.
- 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 ...

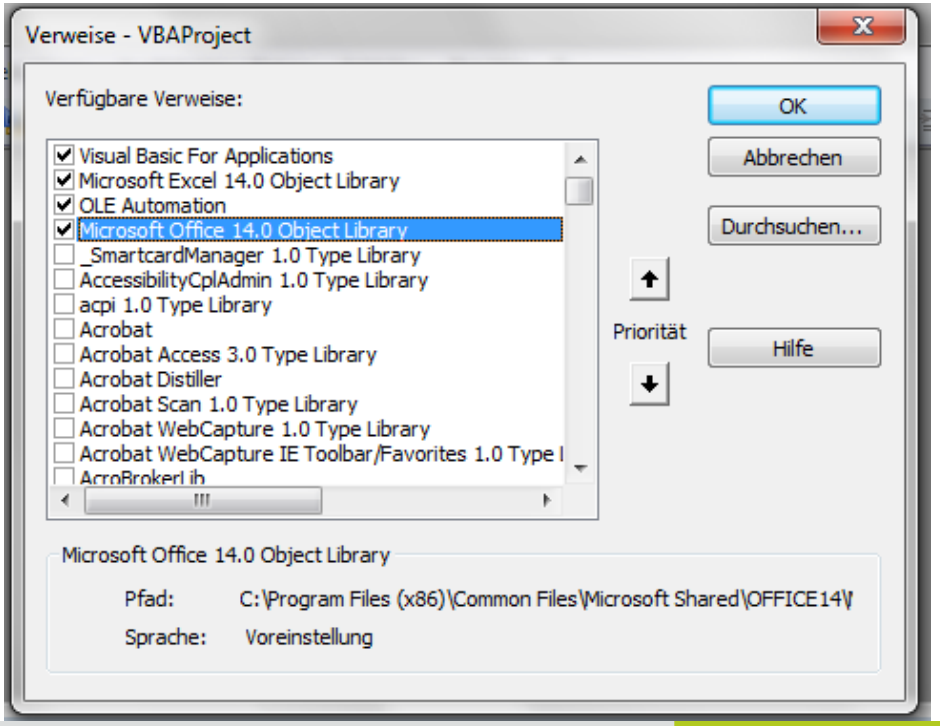
- bilden die Elemente einer Anwendung für VBA nach.
- sind eine Auflistung von Objekten einer Office-Anwendung.
- ordnen Elemente einer Anwendung hierarchisch.
- werden für die Programmierung in Bibliotheken mit der Dateiendung „.olb“ gespeichert.
- können im VBA-Editor mit Hilfe von *Ansicht* – *Objektkatalog*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

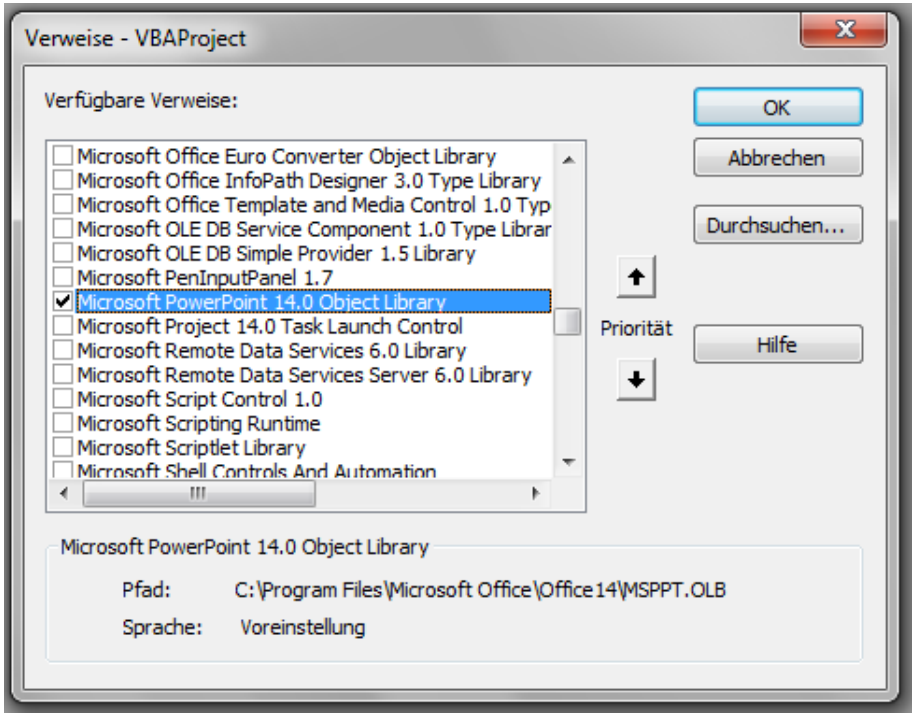


Vorhandene Verweise

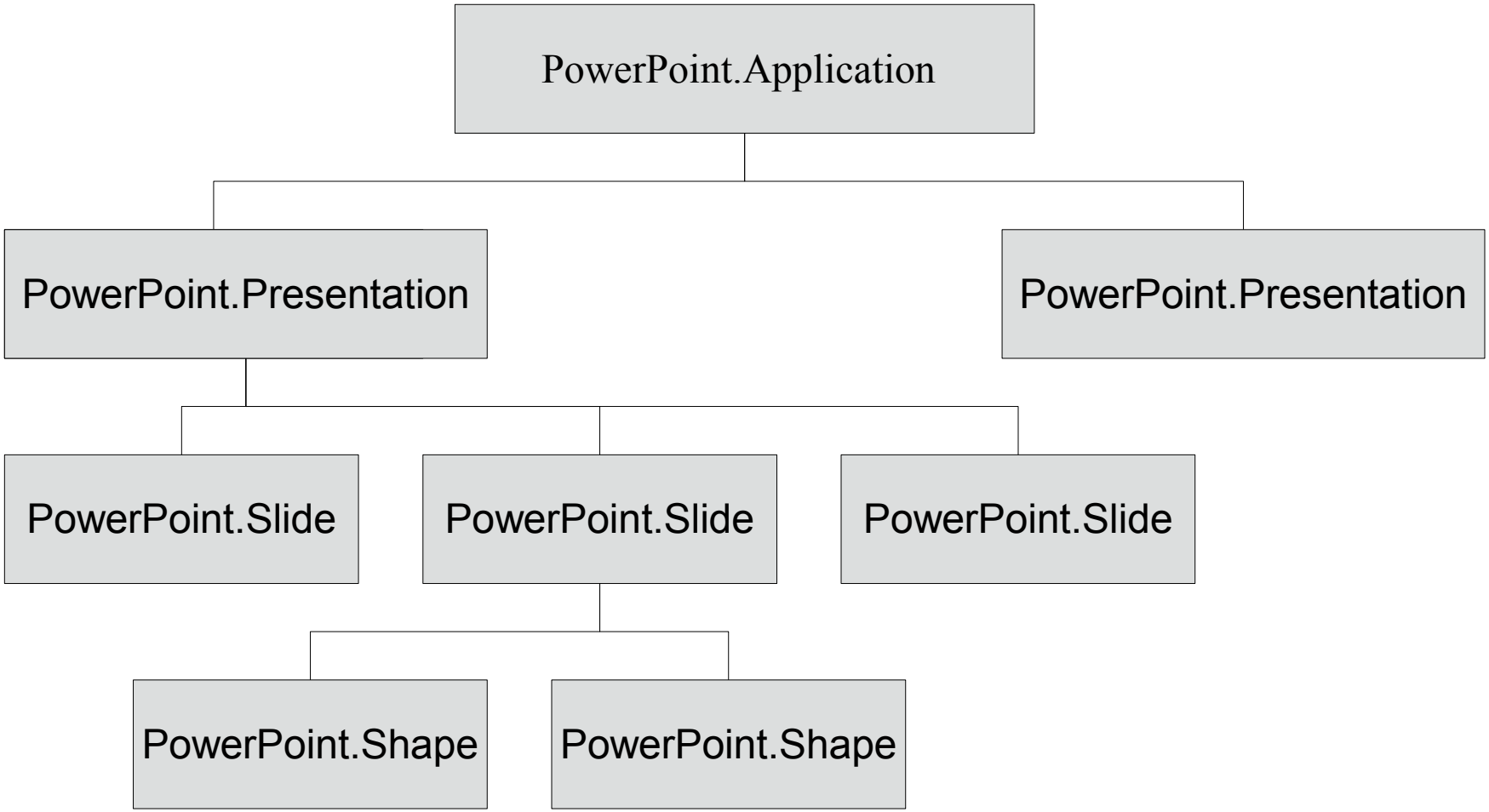
- Visual Basic for Application beschreibt alle Objekte, die die Programmiersprache VBA enthält.
- OLE Automation beschreibt das Einbetten und Verlinken von Objekten wie zum Beispiel die Verknüpfung zu einem Bild.
- Microsoft Office 14.0 Object Library enthält alle Objekte, die alle Office 2010 – Anwendungen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

Einbindung eines Objektmodelles

- *Extras* – *Verweise* im VBA-Editor.
- Suche nach Microsoft PowerPoint 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 „PowerPoint“



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

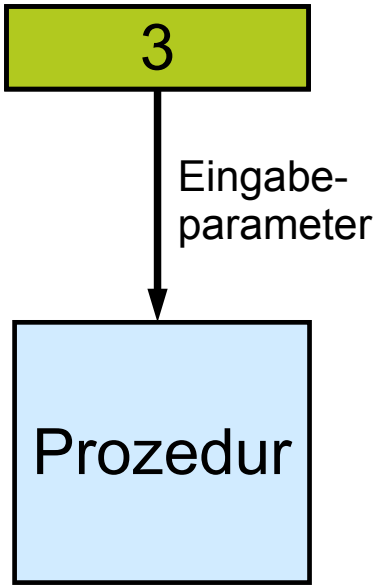
```
Sub DiagrammToPowerPoint()


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.

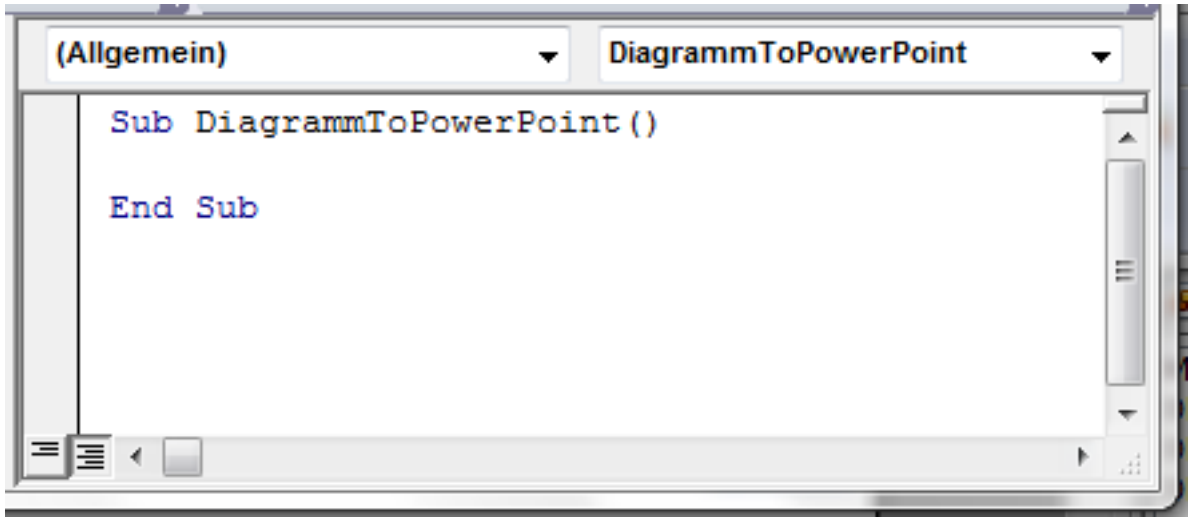
Arbeitsweise einer Prozedur

- 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 DiagrammToPowerPoint()


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 DiagrammToPowerPoint()
    Dim pptApp As PowerPoint.Application
    Dim pptPresentation As PowerPoint.Presentation
    Dim pptSlide As PowerPoint.Slide

    Dim pfad As String



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 PowerPoint-Dokument, Excel-Arbeitsmappe etc..
- werden für Objekte definiert, die häufig in einem Programm genutzt werden.

… definieren

```
Dim pptSlide As PowerPoint.Slide
Dim pfad As String
```

- Jede Objektvariable hat einen eindeutigen Namen (pptSlide, pfad). 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 DiagrammToPowerPoint zugegriffen werden. Diese Prozedur ist Besitzer dieser Variablen.

… zurücksetzen

```
Set pptSlide = 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.

Variablen

```
Sub DiagrammToPowerPoint()
    Dim pfad As String
End Sub
```

- 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 pfad As String
```

- Jede Variable hat einen eindeutigen Namen (pfad). 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 DiagrammToPowerPoint zugegriffen werden (Dim). Diese Prozedur ist Besitzer dieser Variablen.

… zuweisen

```
dateipfad = ThisWorkbook.Path & "\" & "praesentation.pptx"
```

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

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 & "\" & "praesentation.pptx"
```

- 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.
- Die Dateiendung muss vom Entwickler gesetzt werden. Andernfalls ist die Datei nicht lesbar.

Excel-Objekte in VBA nutzen

```
Sub DiagrammToPowerPoint()

    ActiveSheet.ChartObjects(1).Copy

End Sub
```

Objektvariablen definieren

```
Dim datenblatt As Worksheet
Dim zelle As Range
Dim diagramm As ChartObject
```

- 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.
- ChartObject stellt immer ein, in das Arbeitsblatt eingebettetes Diagramm dar.

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. Das Objekt ActiveSheet kann auch ohne Angabe des Eltern-Objekts verwendet werden.
- 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.

Eingebettete Diagramme

ActiveSheet.ChartObjects(1).Copy

- Die Sammlung ChartObjects enthält alle eingebetteten Objekte auf einem Arbeitsblatt. In diesem Beispiel wird auf das aktive Arbeitsblatt Bezug genommen.
- In den runden Klammern wird zur Identifizierung des Diagramms eine Ganzzahl angegeben. Das erste, eingebettete Diagramm hat die Ganzzahl 1 und so weiter.
- Mit Hilfe der Methode .Copy wird ein Diagramm in die Zwischenablage kopiert.

Verweise auf Elemente in PowerPoint

```
Dim pptApp As PowerPoint.Application
Dim pptPresentation As PowerPoint.Presentation
Dim pptSlide As PowerPoint.Slide
```

- Application verweist auf eine Office-Anwendung.
- Presentation verweist auf eine PowerPoint-Präsentation..
- Slide bezieht sich auf die Folien einer Präsentation.
- Alle definierten Objekttypen beziehen sich auf PowerPoint. Die dazugehörige Anwendung und das darin enthaltene Objekt werden mit einem Punkt verbunden.

PowerPoint.Application in VBA nutzen

```
Sub DiagrammToPowerPoint()
    Dim pptApp As PowerPoint.Application

   Set pptApp = CreateObject("Powerpoint.Application")

    With pptApp
        .Visible = True
        .WindowState = ppWindowNormal
        .Activate
    End With

     pptApp.Quit
     Set pptApp = Nothing

End Sub
```

Zugriff auf die PowerPoint-Anwendung

```
Set pptApp = CreateObject("Powerpoint.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 PowerPoint-Anwendung in Excel eingebettet. Die Angabe der zu erstellenden Anwendung wird als Text übergeben.

Anwendung schließen

pptApp.Quit

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

Eigenschaften und Methoden der Anwendung

```
With pptApp
  .Visible = True
  .WindowState = ppWindowNormal
  .Activate
End With
```

- Mit Hilfe der Methode .Activate wird die Anwendung aktiviert. Die Anwendung wird in den Vordergrund des Bildschirms gelegt.
- Die Eigenschaft .Visible blendet die Anwendung am Bildschirm ein.
- Die Eigenschaft .WindowState legt fest, ob das Fenster der Anwendung als Vollbild, minimiert oder normal angezeigt wird.

Nutzung von With

```
With pptApp
  .Visible = True
  .WindowState = ppWindowNormal
  .Activate
End With
```

- Mit Hilfe der Schlüsselwörter With und End With können Anweisungen, die sich auf ein bestimmtes Objekt beziehen, zusammengefasst werden.
- Dem Schlüsselwörter With folgt der Name des Objekts.
- Alle Ausdrücke, die mit einem Punkt beginnen, beziehen sich auf dieses Objekt.

Powerpoint-Präsentationen öffnen oder erstellen

```
Set pptApp = CreateObject("Powerpoint.Application")

With pptApp
    .Visible = True
    .WindowState = ppWindowNormal
    .Activate

    If pfad <> "" Then
       .Presentations.Open Filename:=pfad, ReadOnly:=msoFalse
    Else
        .Presentations.Add
    End If

    Set pptPresentation = .ActivePresentation
End With
```

Neue Präsentation erstellen

pptApp.Presentations.Add

- Die Anwendung besitzt die Liste Presentations. In dieser Liste werden alle, in der Anwendung geöffneten PowerPoint-Dokumente aufgelistet.
- Der Liste Presentations wird mit Hilfe der Methode .Add eine neue Präsentation hinzugefügt.

Präsentation öffnen

```
pptApp.Presentations.Open Filename:=pfad, ReadOnly:=msoFalse
```

- Die Anwendung besitzt die Liste Presentations. In dieser Liste werden alle, in der Anwendung geöffneten PowerPoint-Dokumente aufgelistet.
- Mit Hilfe der Methode .Open wird eine vorhandene Präsentation geöffnet und der Sammlung hinzugefügt.
- Der Methode werden folgende Eingabeparameter übergeben:
  - Welche Datei wird geöffnet?
  - Ist die Präsentation schreibgeschützt?

Benannte Argumente

Filename:=pfad

- … werden einer Methode übergeben. Alle Argumente der Methode werden nach Eingabe der runden Klammern in einem gelben Erklärfenster angezeigt. Die Namen der Argumente können nicht verändert werden.
- Der Zuweisungsoperator := wird aus dem Doppelpunkt und dem Gleichheitszeichen zusammengesetzt. Der Operator weist einem Argument ein Wert zu.
- In diesem Beispiel wird dem Argument Filename der Methode .Open die zu öffnende Datei als Wert übergeben. Der übergebende Wert hat entsprechend des Arguments einen Dateityp.

Zuweisung an eine Objektvariable

```
Set pptPresentation = .ActivePresentation
```

- Mit Hilfe des Schlüsselwortes `Set` wird eine Zuweisung an eine Objektvariable eingeleitet.
- Das Gleichheitszeichen übergibt einen Verweis auf eine bestimmtes Objekt an die Variable.
- In diesem Beispiel wird ein Verweis auf die aktive Präsentation übergeben.

Präsentation speichern

pptPresentation.SaveAs Filename:=pfad

- Mit Hilfe der Methode Save As wird eine neue Präsentation erstmalig gespeichert.
- Dem Argument Filename wird der Name sowie der Speicherort des neuen Dokuments übergeben.

Präsentation schließen

pptPresentation.Close

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

Folien hinzufügen und bearbeiten

```
Dim pptSlide As PowerPoint.Slide

Set pptSlide = pptPresentation.Slides.AddSlide(pptPresentation.Slides.Count + 1, _
                                               pptPresentation.SlideMaster.CustomLayouts(2))


pptSlide.Select
pptSlide.Shapes(1).TextFrame.TextRange = "Klimadiagramm"
```

Folie hinzufügen

```
pptPresentation.Slides.AddSlide(pptPresentation.Slides.Count + 1,
                                pptPresentation.SlideMaster.CustomLayouts(2))
```

- Slides ist eine Liste aller Folien einer Präsentation. Mit Hilfe von .AddSlide wird dieser Liste eine neue Folie hinzugefügt.
- Der Methode werden folgende Eingabeparameter in Abhängigkeit der Position übergeben:
  - Welcher Index wird für die neue Folie genutzt. In diesem Beispiel wird die Anzahl der Folien plus eins genutzt.
  - Welches Layout hat die Folie?
- Mit Hilfe der Methode .Select kann die neue Folie ausgewählt werden.

Text in eine Folie einfügen

```
pptSlide.Shapes(1).TextFrame.TextRange = "Klimadiagramm"
```

- Shapes ist eine Liste aller Elemente auf der gewählten Folie.
- Die Elemente werden durch einen Index eindeutig identifiziert. In diesem Beispiel wird das erste Element auf der Folie genutzt.
- Textframe bildet einen Textrahmen ab, in dem mit Hilfe der Eigenschaft TextRange ein Text gesetzt werden kann.

Elemente aus der Zwischenablage einfügen

pptPresentation.Application.ActiveWindow.View.Paste

- Mit Hilfe von Paste wird ein Element aus der Zwischenablage in eine Präsentation eingefügt.
- In diesem Beispiel wird das Element im aktiven Fenster (ActiveWindow) der Anwendung (Application) in der aktuellen Ansicht (View) eingefügt.