Leibniz Universität IT Services

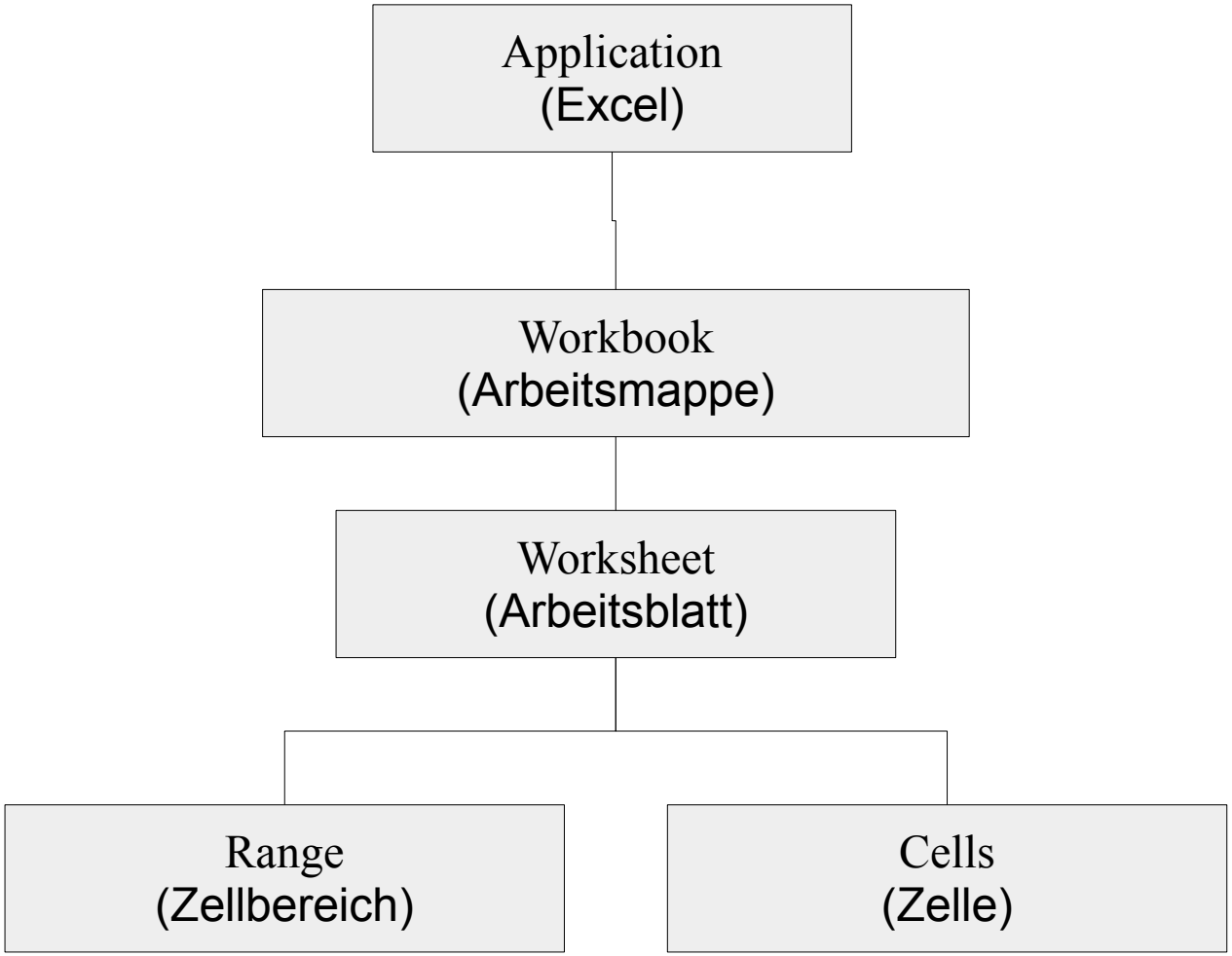
Leibniz Universität Hannover

V(isual) B(asic for) A(pplication)
Export von Daten nach Excel

… Microsoft Excel

- Tabellenkalkulationsprogramm von Microsoft Office - Paket.
- Darstellung von Daten in Tabellenform.
- Kalkulation von Daten aus vorhanden Daten.
- Berechnung von Daten mit Hilfe von Formeln.
- Weiterverarbeitung in Diagrammen und PivotTabellen.

Modell „Microsoft Excel“ (Ausschnitt)



… aus Microsoft Access aufrufen

- Voraussetzung: Excel ist auf dem Entwicklungsrechner vorhanden.
- Zuerst muss ein Verweis auf das Objektmodell angelegt.
- Excel wird geöffnet.
- Arbeitsmappen werden neu angelegt oder geöffnet.
- Daten werden aus einem Tabellenblatt gelesen oder in dieses geschrieben.

Einbindung des Objektmodells

- *Extras – Verweise im VBA-Editor.*
- Mit einem Klick auf das Kästchen wird die Bibliothek *Microsoft Excel 14.0 Object Library* eingebunden. Die Version bezieht sich auf Microsoft Excel 2010.
- Durch Aktivierung des Verweises wird das Objektmodell von Excel bekannt.

Öffnen von Microsoft Excel

```
Dim appExcel As Excel.Application

Set appExcel = New Excel.Application
```

Erläuterung

- Early Bindung.
- Voraussetzung: Der Verweis auf die Anwendung „Microsoft Excel“ muss vorhanden sein.
- Die Objekt-Variable kann nur auf Objekte von der Klasse „Excel“ verweisen. Mit Hilfe von New wird ein Verweis auf eine nicht geöffnete Microsoft Excel – Anwendung übergeben.
- Beim Kompilieren des VBA-Codes wird eine Syntaxprüfung durchgeführt.

Öffnen von Microsoft Excel

```
Dim appExcel As Object

Set appExcel = CreateObject("Excel.Application")
```

Erläuterung

- Late Binding.
- Die Objekt-Variable kann jeden beliebigen Typ von Klasse aufnehmen.
- Mit Hilfe von CreateObjekt() wird eine Instanz der entsprechenden Klasse erzeugt. Der Funktion wird der Name der gewünschten Klasse in runden Klammern übergeben.
- Es werden keine Verweise benötigt.

Einblenden der Anwendung

```
appExcel.Visible = True
```

- Die Methode .Visible blendet die Microsoft Office – Anwendung ein (True) oder aus (False).

Schließen von Microsoft Excel

```
appExcel.Quit
```

- Die Methode .Quit schließt die Microsoft Office - Anwendung.

Arbeitsmappe (Workbook)

- Datei mit der Endung „xls“ oder „xlsx“ ab Excel 2007.
- Container für alle Arbeitsblätter eines Microsoft Excel - Projekts.
- Die Arbeitsmappe enthält standardmäßig drei Arbeitsblätter.

Objekt-Variable vom Typ „Workbook“

```
Dim workbook As Excel.workbook
```

- Die Objekt-Variable kann Verweise auf eine Arbeitsmappe speichern.

Anlegen einer neuen Arbeitsmappe

```
Dim workbook As Excel.workbook

Set workbook = appExcel.Workbooks.Add
```

- Die Auflistung .Workbooks enthält alle, in einer Microsoft Excel – Anwendung geöffneten, Arbeitsmappen.
- Mit Hilfe der Methode .Add wird eine neue Arbeitsmappe erzeugt.

Schließen der Arbeitsmappe

```
Dim workbook As Excel.workbook

Set workbook = appExcel.Workbooks.Add
workbook.Close
```

- Mit Hilfe der Methode .Close wird eine Arbeitsmappe geschlossen.

Arbeitsblätter (Worksheet)

```
Dim worksheet As Excel.worksheet
```

- Tabellenblätter in Excel bestehen aus Zeilen und Spalten.

Aktivierung eines Arbeitsblattes

```
Set worksheet = workbook.Worksheets(1)
```

- Die Auflistung Worksheets enthält alle Arbeitsblätter in einer Arbeitsmappe.
- In den runden Klammern wird der Auflistung ein Index zur Identifizierung eines Arbeitsblattes übergeben. Die Arbeitsblätter werden von 1 bis n nummeriert.

Name eines Arbeitsblattes

```
Set worksheet = workbook.Worksheets(1)
worksheet.Name = "Kunde"
```

- Das Attribut .Name gibt Auskunft über den Namen eines Arbeitsblatts.

Speicherung eines Arbeitsblattes

```
worksheet.SaveAs "Kundeadresse.xlsx"
```

- Die Methode .SaveAs speichert ein Arbeitsblatt unter dem angegebenen Namen.
- Die Methode .Save speichert Änderungen in einem vorhandenen Arbeitsblatt.

Kopieren einer Tabelle von Access nach Excel

```
Dim dbs As DAO.Database
Dim rs As DAO.Recordset
Dim workbook As Excel.workbook
Dim worksheet As Excel.worksheet

Set workbook = appExcel.Workbooks.Add
Set worksheet = workbook.Worksheets(1)

Set dbs = Application.CurrentDb
Set rs = dbs.OpenRecordset("Kunden", dbOpenTable)

If Not (rs.BOF And rs.EOF) Then
   worksheet.Range("A1").CopyFromRecordset rs
End If

rs.Close
```

Erläuterung der if-Anweisung

```
Set rs = dbs.OpenRecordset("Kunden", dbOpenTable)

If Not (rs.BOF And rs.EOF) Then

End If
```

- Wenn die Marker BOF „Beginn des Recordsets“ und EOF „Ende des Recordsets“ nicht gleich sind …
- Wenn die Tabelle nicht leer ist ...

Kopieren der Datensätze

```
worksheet.Range("A1").CopyFromRecordset rs
```

- Dem Zellbereich .Range wird der Name einer Zelle in runden Klammern übergeben. In diesem Beispiel wird auf die Zelle in der Spalte A in der ersten Zeile in dem angegeben Arbeitsblatt verwiesen.
- Beginnend mit dieser Zelle werden die Daten aus dem Recordset mit Hilfe der Methode .CopyFromRecordset kopiert.

Arbeitsschritte der Methode „CopyFromRecordset“

- Kopiere die zu exportierenden Datensätze in die Zwischenablage.
- Markiere die Zelle A1 auf dem ersten Arbeitsblatt in der geöffneten Arbeitsmappe.
- Füge die in der Zwischenablage vorhandenen Datensätze ab der markierten Zelle in das aktive Arbeitsblatt ein.

Setzen von „Kopfzeilen“

```
Spalte = 1

For Each element In rs.Fields
    worksheet.Cells(1, spalte).Value = element.Name
    spalte = spalte + 1
Next
```

- Für jedes Feld in dem Recordset wird eine Spaltenüberschrift gesetzt.
- Die Auflistung .Cells enthält jede Zelle in einem Arbeitsblatt. Die Auflistung ist mehrdimensional.

Index einer Zelle

```
Spalte = 1

For Each element In rs.Fields
    worksheet.Cells(1, spalte).Value = element.Name
    spalte = spalte + 1
Next
```

- Die Auflistung .Cells enthält jede Zelle in einem Arbeitsblatt.
- Eine Zelle wird durch die Zeile und die Spalte eindeutig identifiziert.
- Die Zeile und die Spalte werden in den runden Klammern durch ein Kommata getrennt.

Inhalt einer Zelle

```
Spalte = 1

For Each element In rs.Fields
    worksheet.Cells(1, spalte).Value = element.Name
    spalte = spalte + 1
Next
```

- Die Eigenschaft .Value speichert den Inhalt einer Zelle.

Formatierung einer Zelle

```
worksheet.Cells(zeile, spalte).NumberFormat = "#,##0.00  €"
```

- Die Eigenschaft .NumberFormat formatiert Zahlen in einer Zelle entsprechend eines Musters.
- Das Muster enthält Formatierungszeichen wie zum Beispiel das Hash-Zeichen oder die Null. Das Muster kann aber auch Literale wie das Euro-Zeichen enthalten.

Formatierungszeichen

```
worksheet.Cells(zeile, spalte).NumberFormat = "#,##0.00  €"
```

- An der Position des Hash-Zeichens kann eine Zahl vorhanden sein, muss aber nicht.
- An der Position des Formatierungszeichen „0“ muss eine Zahl vorhanden sein, andernfalls wird 0 ausgegeben.
- Das Komma symbolisiert das länderspezifische Tausender-Trennzeichen.
- Der Punkt symbolisiert das Dezimaltrennzeichen.