

R | R | Z | N |

Regionales Rechenzentrum für Niedersachsen

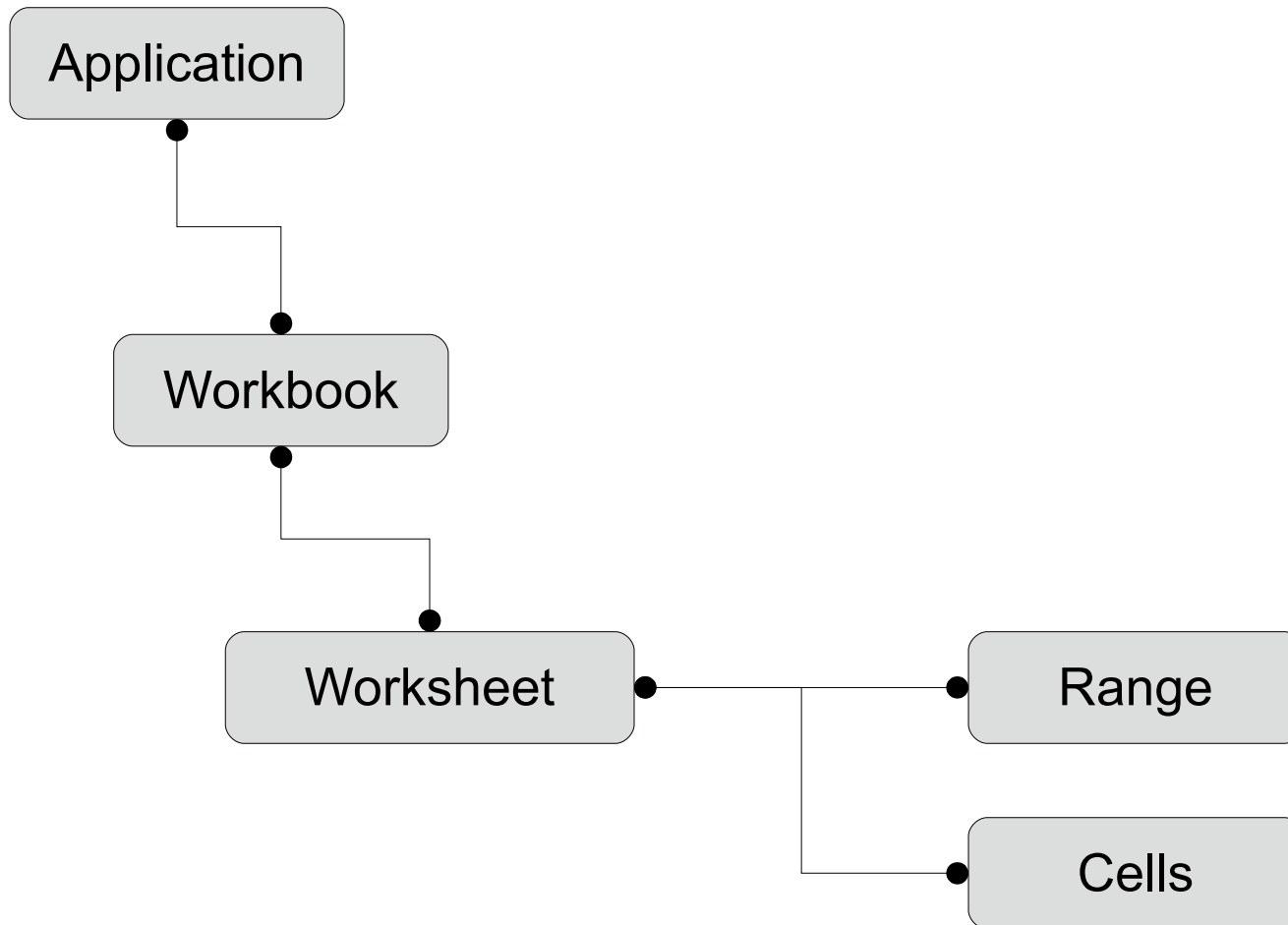


Access 2010 – Programmierung Import und Export nach Excel

Excel ...

- ist das Tabellenkalkulationsprogramm von Microsoft Office.
- wird genutzt, um numerische Daten in Tabellenform zu erfassen.
- kann Daten automatisch berechnen.
- bietet viele vordefinierte Formeln für die Berechnung.
- kann Daten mit Hilfe von Diagrammen darstellen.

Objektmodell



Daten nach Excel transferieren

```
Private Sub TransferToExcel()  
    Dim aktuellDatum As String  
    Dim datiname As String  
  
    aktuellDatum = CStr(Date)  
    aktuellDatum = Replace(aktuellDatum, ".", "_")  
  
    dateiname = CurrentProject.Path & "\bestellung" & aktuellDatum & ".xls"  
  
    DoCmd.TransferSpreadsheet TransferType:=acExport, _  
        SpreadsheetType:=acSpreadsheetTypeExcel9, _  
        TableName:="qryBestellung", _  
        FileName:=dateiname, HasFieldNames:=True  
  
End Sub
```

Erläuterung

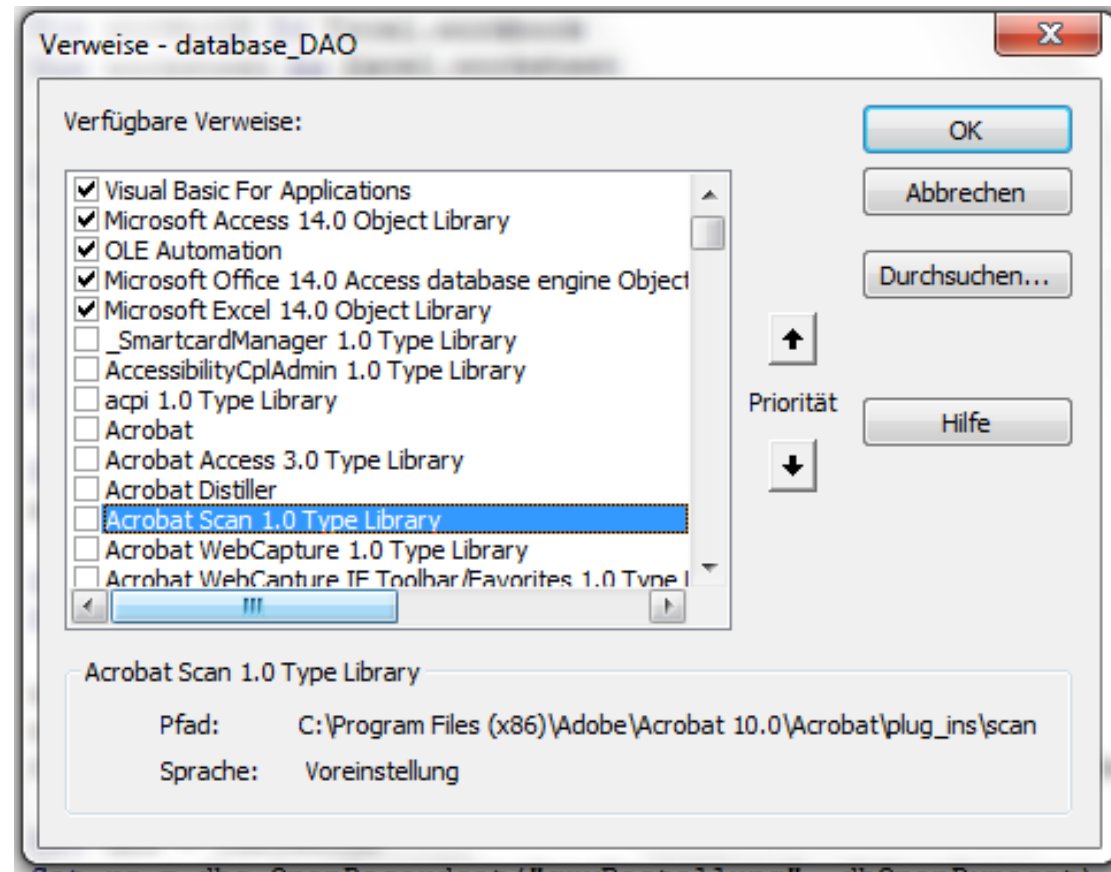
- « DoCmd.TransferSpreadsheet » transferiert ein Arbeitsblatt aus Access nach Excel.
- Folgende Angaben werden benötigt:
 - Werden die Daten exportiert, importiert oder verlinkt?
 - In welches Format werden die Daten transferiert?
 - Welche Daten werden transferiert?
 - In welche Datei werden die Daten transferiert?
 - Werden die Spaltenüberschriften der Tabelle übernommen?

Mit Excel aus Access arbeiten

- 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.

Excel in Access einbinden

- *Extras – Verweise.*
- Mit einem Klick auf das Kästchen wird die Bibliothek Microsoft Excel 14.0 Object Library eingebunden.
- Durch Aktivierung des Verweises wird das Objektmodell von Excel bekannt.



Application ...

- ist die Wurzel aller Objekte in Excel.
- beschreibt die Anwendung „Excel“ selber.
- hat Eigenschaften, die Informationen zur Anwendung bereithalten.
- hat Methoden, die die Anwendung verändern.
- Die Anweisung « Dim appExcel As Excel.Application » erstellt eine Objektvariable vom Datentyp „Excel-Anwendung“.

Application „Excel“ öffnen und schließen

```
Dim appExcel As Excel.Application
```

```
Set appExcel = Excel.Application
```

```
appExcel.Visible = True
```

```
appExcel.Quit
```

Andere Möglichkeit

```
Dim appExcel As Object
```

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

```
appExcel.Visible = True
```

```
appExcel.Quit
```

Anwendung starten

- Mit Hilfe von « `Set appExcel = Excel.Application` » wird ein Verweis auf die Excel-Application erstellt.
- Die Anweisung « `Set appExcel = CreateObject("Excel.Application")` » erzeugt eine Instanz einer Excel-Application.
- Beide Zuweisungen starten eine Excel-Anwendung. Die Anwendung ist aber nicht sichtbar.

Anwendung einblenden

- Die Anweisung « `appExcel.Visible = True` » blendet die Anwendung ein.

Anwendung schließen

- Die Methode « `appExcel.Quit` » schließt die Anwendung.
- Auf Nachfrage können nicht gespeicherte Arbeitsmappen gespeichert werden.

Arbeitsmappe (Workbook) ...

- ist eine Datei mit der Endung „xls“ oder „xlsx“ ab Excel 2007.
- ist ein Container für alle Arbeitsblätter eines Projekts.
- hat standardmäßig drei Arbeitsblätter.
- Die Anweisung « Dim workbook As Excel.Workbook » erstellt eine Objektvariable vom Datentyp „Excel-Arbeitsmappe“.

Neue Arbeitsmappe hinzufügen und schließen

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

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

```
workbook.Close
```

Erläuterung

- « Workbooks » ist eine Collection (Aufzählung) der geöffneten Anwendung, die alle Arbeitsmappen enthält.
- Mit Hilfe der Methode « appExcel.Workbooks.Add » wird eine neue Arbeitsmappe in der geöffneten Anwendung angelegt.
- Die Methode « workbook.Close » schließt die momentan geöffnete Arbeitsmappe.

Vorhandene Arbeitsmappe öffnen

```
Dim workbook As Excel.Workbook  
Dim dateiName As String  
  
dateiName = CurrentProject.Path & "\kundeNew.xlsx"  
Set workbook = appExcel.Workbooks.Open(dateiName)  
  
workbook.Close
```

Erläuterung

- Mit Hilfe der Methode « `appExcel.Open(dateiname)` » wird eine vorhandene Excel-Arbeitsmappe geöffnet.
- Der Methode wird der Name der zu öffnenden Arbeitsmappe als Argument übergeben.
- Mit Hilfe des Arguments « `ReadOnly:=True` » wird die Arbeitsmappe schreibgeschützt.
- Weitere Argumente werden in der Hilfe erläutert.

Arbeitsblätter (Worksheet) ...

- enthalten die Daten einer Arbeitsmappe.
- sind Tabellen in Excel.
- bestehen aus Spalten, die eine bestimmte Information speichern.
- bestehen aus Zeilen, die ein bestimmtes Objekt wie zum Beispiel „Kunde“ beschreiben.
- Die Anweisung « Dim worksheet As Excel.Worksheet » erstellt eine Objektvariable vom Datentyp „Excel-Arbeitsblatt“.

Arbeitsblätter öffnen

```
Dim appExcel As Excel.Application
Dim workbook As Excel.workbook
Dim worksheet As Excel.worksheet
Dim dateiname As String

Set appExcel = Excel.Application
appExcel.Visible = True

Set workbook = appExcel.Workbooks.Open(dateiname)
Set worksheet = workbook.Worksheets(1)
```

Erläuterung

- Die Collection « Worksheets » enthält alle Arbeitsblätter einer geöffneten Arbeitsmappe.
- Die Anweisung « Set worksheet = workbook.Worksheets(1) » aktiviert ein Arbeitsblatt.
- Als Index kann ...
 - eine Ganzzahl genutzt werden. 1 bezeichnet das erste Arbeitsblatt in der Auflistung.
 - der Name des Arbeitsblattes genutzt werden.

Recordset kopieren

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

```
With rs  
    If (.BOF And .EOF) Then  
        MsgBox "Keine Kunden vorhanden."  
    Else  
        worksheet.Range("A1").CopyFromRecordset rs  
    End If  
End With
```

```
rs.Close  
dbs.Close
```

Zellbereich definieren

- Der Zellbereich « `worksheet.Range` » wird definiert.
- Mit Hilfe von « `worksheet.Range(A1)` » wird die Zelle in der ersten Spalte, ersten Zeile markiert.
- Die Anweisung « `worksheet.Range(A1:E10)` » nutzt den Zellbereich von A1 bis E10. Die Angabe links vom Doppelpunkt definiert die linke obere Ecke des Bereichs. Die Angabe rechts vom Doppelpunkt definiert die rechte untere Ecke des Bereichs.

Daten in ein Zellbereich kopieren

- Die Anweisung « `worksheet.Range("A1").CopyFromRecordset rs` » kopiert die Daten des angegebenen Recordset ab der Zelle A1 in das aktive Arbeitsblatt.
- Die Daten aus dem Recordset werden, beginnend mit der linken oberen Ecke eines Bereichs, in das aktive Arbeitsblatt kopiert.

Kopfzeilen des Recordset nutzen

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

Name eines Feldes

- Der Anweisung « rs.Name » gibt den Namen einer Spalte einer Tabelle zurück.
- Die Anweisung gibt den Namen eines Datenfeldes eines Recordset zurück.

Zellen in einem Arbeitsblatt

- Die Zellen in einem Arbeitsblatt zeigen die Daten an. In Zellen können Daten geschrieben werden.
- Der Anweisung « `worksheet.Cells(1, spalte)` » beschreibt eine Zelle auf dem aktiven Arbeitsblatt.
- Die Zelle wird eindeutig durch die Angabe der Zeile und der Spalte identifiziert.

Zellen lesen oder in Zellen schreiben

- Die Anweisung
« `worksheet.Cells(zeile, 8) = rs.Fields("Gesamt")` » schreibt den Inhalt der Spalte „Gesamt“ des aktuellen Datensatzes in eine Zelle in der 8. Spalte.
- Die Anweisung « `.Fields("Anrede") = worksheet.Cells(zeile, 1)` » schreibt den Inhalt der erste Spalte einer Zeile in das Datenfeld „Anrede“ des aktuellen Datensatzes.

Zahlenformat einer Zelle

- Die Anweisung
« `worksheet.Cells(zeile, 7).NumberFormat= "#,##0.00 €"` » legt
das Zahlenformat der Zelle fest.
- In diesem Beispiel symbolisiert ...
 - die Null eine Zahl, die vorhanden sein muss.
 - das Hash-Zeichen eine Zahl, die vorhanden sein kann.
 - das Komma das länderspezifische Tausender-Zeichen.
 - der Punkt das länderspezifische Dezimal-Trennzeichen.
- Weitere Möglichkeiten werden in der Hilfe angezeigt.

Arbeitsblätter speichern

```
worksheet.SaveAs (dateiname)
```

```
workbook.Close
```

```
appExcel.Quit
```

Erläuterung

- Die Anweisung « `worksheet.SaveAs(dateiname)` » speichert die Änderungen an dem Arbeitsblatt in einer neuen Datei.
- Der Name der Datei wird als Argument übergeben.
- Falls die Datei vorhanden ist, wird diese auf Nachfrage überschrieben.