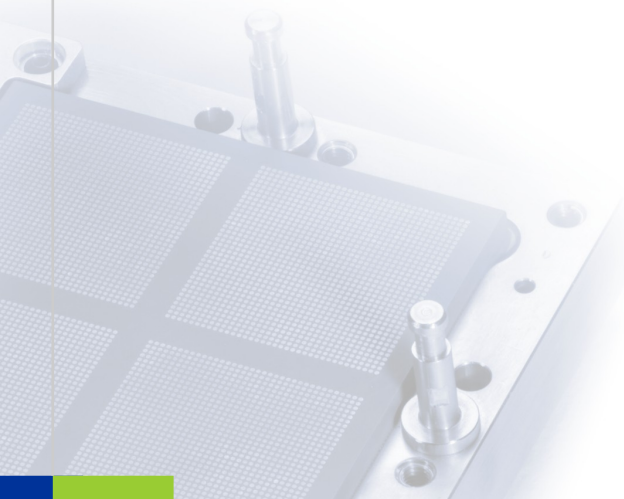
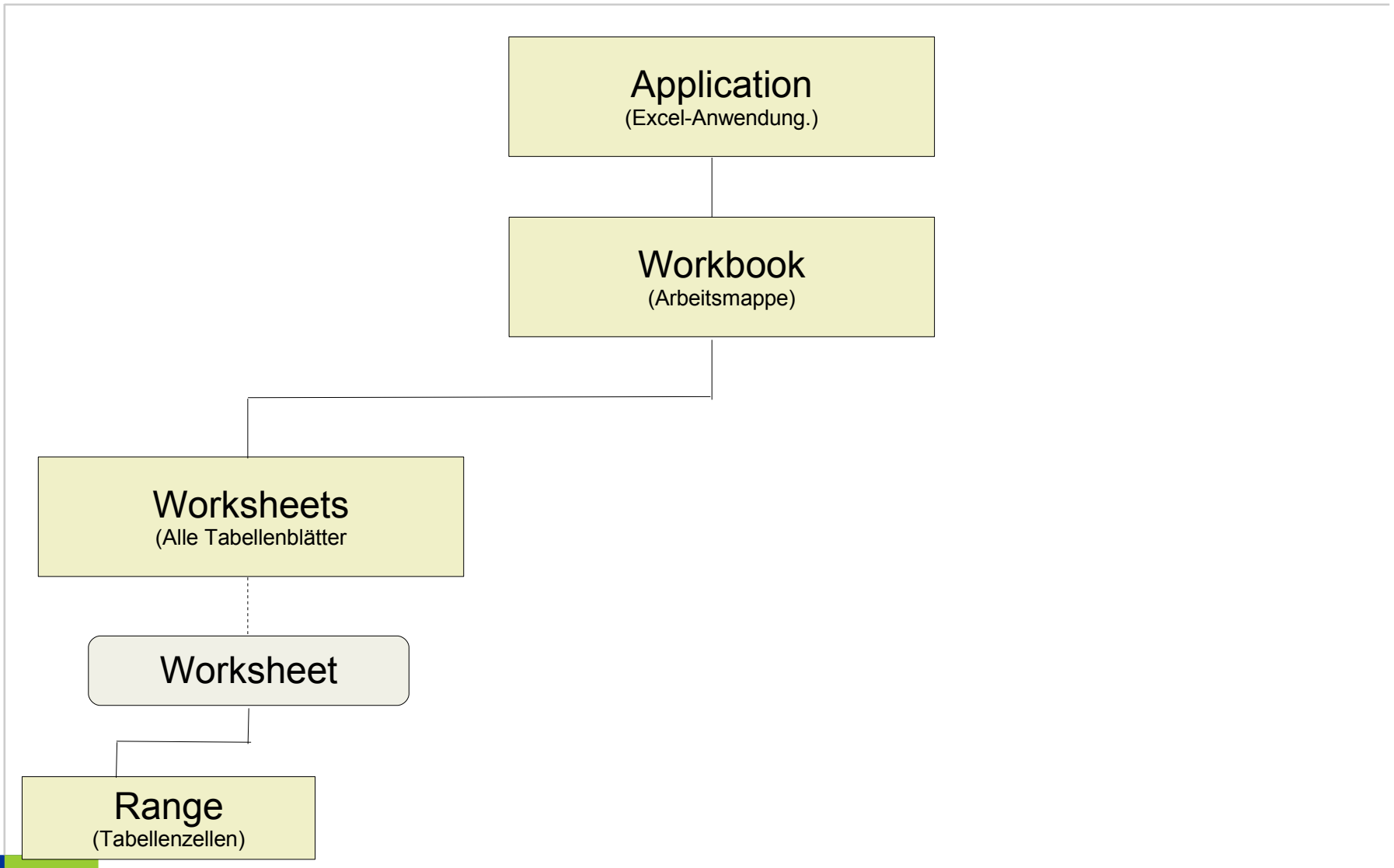


VBA (Visual Basic for Application)

Zugriff auf Excel



- Klicken Sie auf das Menü Extras – Verweise im VBA-Editor.
- Der Verweis Microsoft *DAO 3.6 Object Library* ist aktiviert.
- Der Verweis Microsoft *Microsoft Excel 11.0 Libray* ist aktiviert.



```
Sub newQuery()  
  Dim sql As String  
  Dim qry As QueryDef  
  Dim pfad As String  
  
  sql = "SELECT * FROM tab_lieferant WHERE ort LIKE 'Mölln'"  
  
  Set qry = CurrentDb.CreateQueryDef("Lieferant", sql)  
  
  pfad = Application.CurrentProject.Path & "\lieferant.xls"  
End Sub
```

```
Sub exportTransfer()  
  Dim sql As String  
  Dim qry As QueryDef  
  Dim pfad As String  
  
  sql = "SELECT * FROM tab_lieferant WHERE ort LIKE 'Mölln'"  
  
  Set qry = CurrentDb.CreateQueryDef("Lieferant", sql)  
  
  pfad = Application.CurrentProject.Path & "\lieferant.xls"  
End Sub
```

- *Application.CurrentProject* beschreibt die aktuell geöffnete Datenbank.
- *Path* gibt den Speicherort zurück.

```
Sub exportTransfer()  
  Dim sql As String  
  Dim qry As QueryDef  
  Dim pfad As String  
  
  Err.Clear  
  sql = "SELECT * FROM tab_lieferant WHERE ort LIKE 'Mölln'"  
  
  On Error GoTo errorMeldung  
  Set qry = CurrentDb.CreateQueryDef("Lieferant", sql)  
  
errorMeldung:  
  If (Err.Number = 3012) Or (Err.Number = 0) Then  
    pfad = Application.CurrentProject.Path & "\lieferant.xls"  
  
    DoCmd.TransferSpreadsheet acExport, acSpreadsheetTypeExcel8,  
      "Lieferant", pfad, True  
  
  End If  
End Sub
```

```
Sub exportTransfer()  
  Dim sql As String  
  Dim qry As QueryDef  
  Dim pfad As String  
  
  Err.Clear  
  sql = "SELECT * FROM tab_lieferant WHERE ort LIKE 'Mölln'"  
  
  On Error GoTo errorMeldung  
  Set qry = CurrentDb.CreateQueryDef("Lieferant", sql)  
  
errorMeldung:  
  If (Err.Number = 3012) Then  
    pfad = Application.CurrentProject.Path & "Lieferant.xls"  
  
    DoCmd.TransferSpreadsheet acExport, acSpreadsheetTypeExcel8,  
      "Lieferant", pfad, True  
  
  End If  
End Sub
```

Mit Hilfe dieses Makro-Befehls können Daten aus Access in eine Excel-Tabelle exportiert oder importiert werden. Mit Hilfe von True werden die Tabellennamen als Spaltenüberschriften genutzt.

```
Sub exportOutPut()  
  Dim sql As String  
  Dim qry As QueryDef  
  Dim pfad As String  
  
  sql = "SELECT * FROM tab_lieferant WHERE ort LIKE 'Mölln'"  
  
  Set qry = CurrentDb.CreateQueryDef("Lieferant", sql)  
  
  pfad = Application.CurrentProject.Path & "\lieferant.xls"  
  
  DoCmd.OutputTo acOutputQuery, "Lieferant", acFormatXLS, pfad  
  
End Sub
```

Mit Hilfe dieses Makro-Befehls können Daten in andere Formate ausgegeben werden. Hier wird die Abfrage "Lieferant" in einem Excel-Format ausgegeben.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application
```

```
pfad = Application.CurrentProject.Path  
xlsDatei = "lieferant.xls"  
pfad = pfad & "\" & xlsDatei
```

```
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True
```

Mit Hilfe von *CreateObjekt* wird eine Office-Anwendung gestartet. In diesem Beispiel wird eine Excel-Anwendung gestartet. Mit Hilfe von VBA kann auf die verschiedenen Elemente von Excel zugegriffen werden.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application
```

```
pfad = Application.CurrentProject.Path  
xlsDatei = "lieferant.xls"  
pfad = pfad & "\" & xlsDatei
```

```
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True
```

Die Eigenschaft *Sichtbar* der Anwendung wird gesetzt. Die Anwendung wird am Bildschirm angezeigt.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application
```

```
pfad = Application.CurrentProject.Path  
xlsDatei = "lieferant.xls"  
pfad = pfad & "\" & xlsDatei
```

```
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True
```

```
xlsApp.Quit
```

Die Anwendung wird geschlossen.

- ... ist eine Auflistung aller geöffneten Arbeitsmappen.
- Jede Arbeitsmappe enthält n Tabellenblätter
- *Workbook* beschreibt eine Arbeitsmappe.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook
```

```
pfad = Application.CurrentProject.Path  
xlsDatei = "lieferant.xls"  
pfad = pfad & "\" & xlsDatei
```

```
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True
```

```
Set xlsBook = xlsApp.Workbooks.Add  
xlsBook.SaveAs pfad
```

Workbooks ist eine Auflistung aller geöffneten Arbeitsmappen. Dieser Liste wird mit Hilfe von *Add* eine neue Arbeitsmappe hinzugefügt.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook
```

```
pfad = Application.CurrentProject.Path  
xlsDatei = "lieferant.xls"  
pfad = pfad & "\" & xlsDatei
```

```
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True
```

```
Set xlsBook = xlsApp.Workbooks.Add  
xlsBook.SaveAs pfad
```

Die neue Arbeitsmappe wird gespeichert.

```
Dim pfad As String  
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook  
  
pfad = Application.CurrentProject.Path  
xlsDatei = "klima.xls"  
pfad = pfad & "\" & xlsDatei  
  
Set xlsApp = CreateObject("Excel.Application")  
xlsApp.Visible = True  
  
If Dir(pfad) = "" Then  
    MsgBox("Die Datei ist nicht vorhanden")  
    Exit Sub  
End If  
  
Set xlsBook = xlsApp.Workbooks.Open(pfad)  
  
xlsBook.Close
```

Falls die angegebene Datei vorhanden ist, wird diese mit Hilfe von *Open* geöffnet. *.Close* schließt die Arbeitsmappe.

- ... ist eine Auflistung aller Tabellenblätter.
- Worksheet ist ein bestimmtes Tabellenblatt.
- Jedes Tabellenblatt enthält Informationen zu einem bestimmten Informationen in Zeilen und Spalten.
- Eine Arbeitsmappe enthält mindestens ein Tabellenblatt und maximal 255.

```
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook  
Dim xlsSheet As Excel.Worksheet
```

```
Set xlsSheet = xlsBook.Worksheets.Add
```

```
ActiveSheet.Name = "Lieferant"
```

Mit Hilfe von *Add* des Worksheets wird ein neues Tabellenblatt erstellt.

Worksheets.Add After:=Worksheets(2) legt das neue Tabellenblatt hinter dem zweiten Tabellenblatt an.

Worksheets.Add Before:=Worksheets(2), Count:=3 legt drei neue Tabellenblätter vor dem zweiten Tabellenblatt an.

```
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook  
Dim xlsSheet As Excel.Worksheet
```

```
Set xlsSheet = xlsBook.Worksheets.Add
```

```
ActiveSheet.Name = "Lieferant"
```

ActiveSheet bezeichnet das im Vordergrund liegende Tabellenblatt.
Mit Hilfe der Eigenschaft *.Name* wird dem Tabellenblatt eine Bezeichnung übergeben.

```
Dim xlsDatei As String  
Dim xlsApp As Excel.Application  
Dim xlsBook As Excel.Workbook  
Dim xlsSheet As Excel.Worksheet  
  
Set xlsSheet = xlsBook.Worksheets("KlimaHannover")  
  
xlsSheet.Select
```

Mit Hilfe des Namens kann ein bestimmtes Tabellenblatt in einer Arbeitsmappe angesprochen werden.
Die Methode *.Select* wählt ein bestimmtes Tabellenblatt aus und aktiviert dieses.

- *ActiveSheet.Columns(1)* spricht die erste Spalte im aktiven Tabellenblatt an.
- *ActiveSheet.Columns("A")* nutzt den Spaltennamen.
- *ActiveSheet.Rows(1)* spricht die erste Zeile im aktiven Tabellenblatt an.

- *ActiveSheet.Columns(strName).Insert* fügt eine Spalte ein. Die nachfolgenden Spalten werden um eine Position nach rechts verschoben.
- *ActiveSheet.Rows(position).Insert* fügt eine Zeile ein. Alle anderen Zeilen werden nach unten verschoben.
- *ActiveSheet.Columns("C:E").Insert* fügt einen Bereich von Spalten ein. Der Anfang und das Ende werden durch ein Doppelpunkt getrennt. Hier werden zum Beispiel drei Spalten nach der Spalte B eingefügt.
- *ActiveSheet.Columns("C:E, S:U").Insert* fügt einen nicht zusammenhängenden Bereich ein. Die einzelnen Bereiche werden durch Kommata getrennt.

- *ActiveSheet.Columns(strName).Delete* löscht eine Spalte ein. Die nachfolgenden Spalten werden um eine Position nach links verschoben.
- *ActiveSheet.Rows(position).Delete* löscht eine Zeile ein. Alle anderen Zeilen werden nach nach verschoben.
- *ActiveSheet.Rows("3:6").Delete* löscht einen Bereich von Zeilen ein. Der Anfang und das Ende werden durch ein Doppelpunkt getrennt. Hier werden zum Beispiel die Zeilen 3, 4, 5 und 6 gelöscht.
- *ActiveSheet.Columns("C:E, S:U").Delete* löscht einen nicht zusammenhängenden Bereich ein. Die einzelnen Bereiche werden durch Kommata getrennt.

```
Set dbs = CurrentDb
```

```
Set rs = dbs.OpenRecordset("SELECT * FROM tab_lieferant")
```

```
xlsSheet.Cells(1, 1).CopyFromRecordset rs
```

- `.Cells` beschreibt immer durch die Angabe der Zeile und der Spalte eine bestimmte Zelle in einem Tabellenblatt
- `.Cells(Zeile, Spalte).Value` gibt den Wert der Zelle zurück.
- `.Cells(Zeile, Spalte).CopyFromRecordset` kopiert ab einer bestimmten Zelle den Inhalt eines Recordsets in eine Excel-Tabelle.