Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Pivot-Tabellen automatisiert erstellen

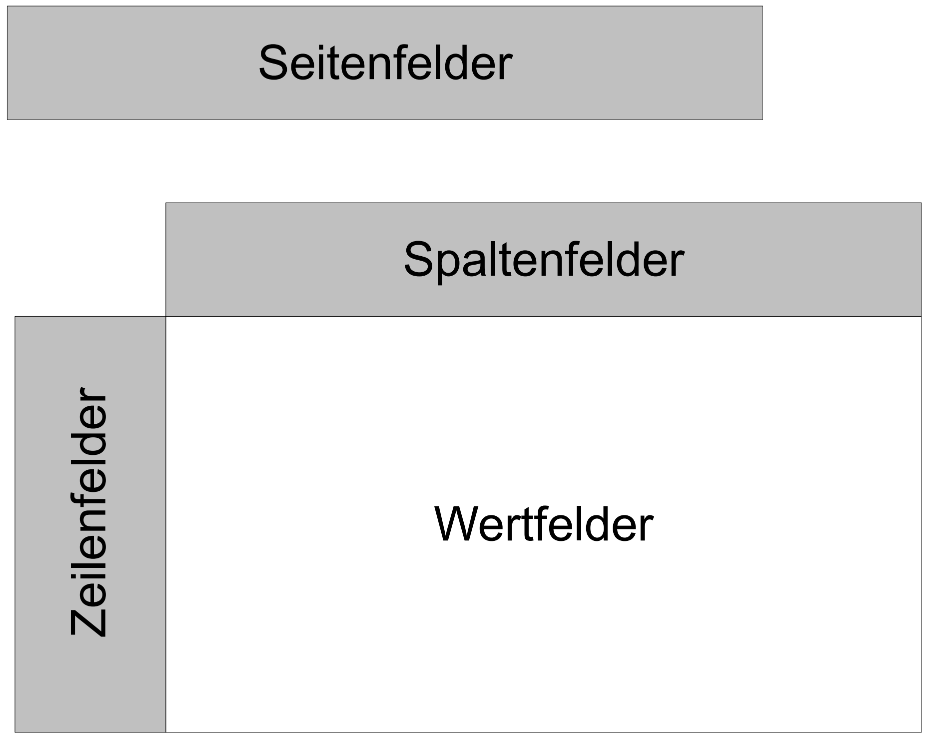
Pivot-Tabellen (Pivot Table)

- bieten eine interaktive Tabellenansicht.
- stellen verschiedene Sichten auf die Daten bereit.
- berechnen Gesamt- oder Teilergebnisse von großen Datenmengen. Für die Berechnung wird zum Beispiel bei Zahlen die Funktion « Summe() » genutzt.
- fassen Daten zusammen und analysiern diese.
- filtern, sortieren und gruppieren große Datenmengen.

… haben als Datenquelle eine ...

- Excel-Tabelle.
- als Tabelle definierten Zellbereich.
- externe Datenquelle wie zum Beispiel eine Access-Datenbank.

… haben folgenden Aufbau



Arbeitsschritte

- Die Datenquelle für eine Pivot-Tabelle ist markiert
- Das Menüband Einfügen ist geöffnet.
- Mit einem Klick auf das Symbol *PivotTable* im Bereich Tabellen das erste Dialogfeld geöffnet.
- In diesem Dialogfeld wird der markierte Zellbereich angezeigt und die Pivot-Tabelle wird auf einen neuen Datenblatt erstellt.
- Anschließend wird beispielhaft eine Pivot-Tabelle und der dazugehörige Aufgabenbereich angezeigt.
- Im Aufgabenbereich werden die Tierarten aus der Feldliste in die Zeilenbeschriftungen gezogen. Die Jahre 2006 und 2007 werden als Werte genutzt. Die Anzahl der Tiere pro Art werden in der Pivot-Tabelle angezeigt.

Arbeitsschritte automatisieren

- Voraussetzung: Die Entwicklertools sind eingeblendet.
- Die Arbeitsschritte werden einmalig mit Hilfe eines Makros aufgezeichnet.
- Nach Beendigung der Aufzeichnung werden die Arbeitsschritte automatisiert über ein Symbol oder Tastenkürzel gestartet.

Entwicklertools einblenden

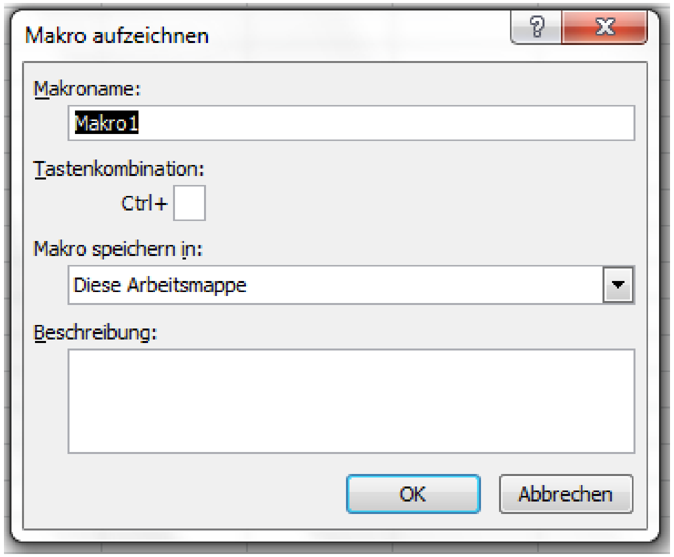
- *Datei – Optionen*.
- *Menüband anpassen*.
- In der rechten Liste Menüband anpassen werden alle Hauptregisterkarten angezeigt. Das Menüband Entwicklertools ist standardmäßig ausgeblendet (Kästchen ist leer).
- Durch einen Mausklick links von der Bezeichnung des Menübandes wird dieses aktiviert. In dem Kästchen wird ein Häkchen angezeigt.

Arbeitsschritte aufzeichnen

- Das Menüband Entwicklertools ist eingeblendet.
- Durch einen Mausklick wird die Aktion *Makro aufzchn.* in der Gruppe Code gestartet.
- Es öffnet sich der Dialog Makro aufzeichnen. In dem Dialog werden die Grundeinstellungen der Aufzeichnung festgelegt. *OK* schließt den Dialog.
- Hinweis: Die Aufzeichnung endet nicht automatisch.

Dialog „Makro aufzeichnen“

- In dem obersten Textfeld wird ein eindeutiger Name für die Aufzeichnung eingegeben.
- Zum Starten des Makros kann ein Buchstabe eingegeben werden. Hinweis: Einige Kombinationen wie <CTRL>+<C> sind belegt.
- Mit Hilfe des Kombinationsfeldes wird der Speicherort des Makros festgelegt. Standardmäßig wird ein Makro in der aktuellen Arbeitsmappe gespeichert.
- In dem unteren Textfeld kann eine Beschreibung für das Makro ein-gegeben werden.



Aufzeichnung beenden

- Das Menüband Entwicklertools ist eingeblendet.
- Mit einem Mausklick auf die Aktion *Aufzeichnung beenden* wird die momentan laufende Aufzeichnung beendet.
- Hinweis: Alle Arbeitsschritte zwischen dem Beginn und dem Ende der Aufzeichnung sind in der Programmiersprache VBA gespeichert.

Visual Basic for Application (VBA) ...

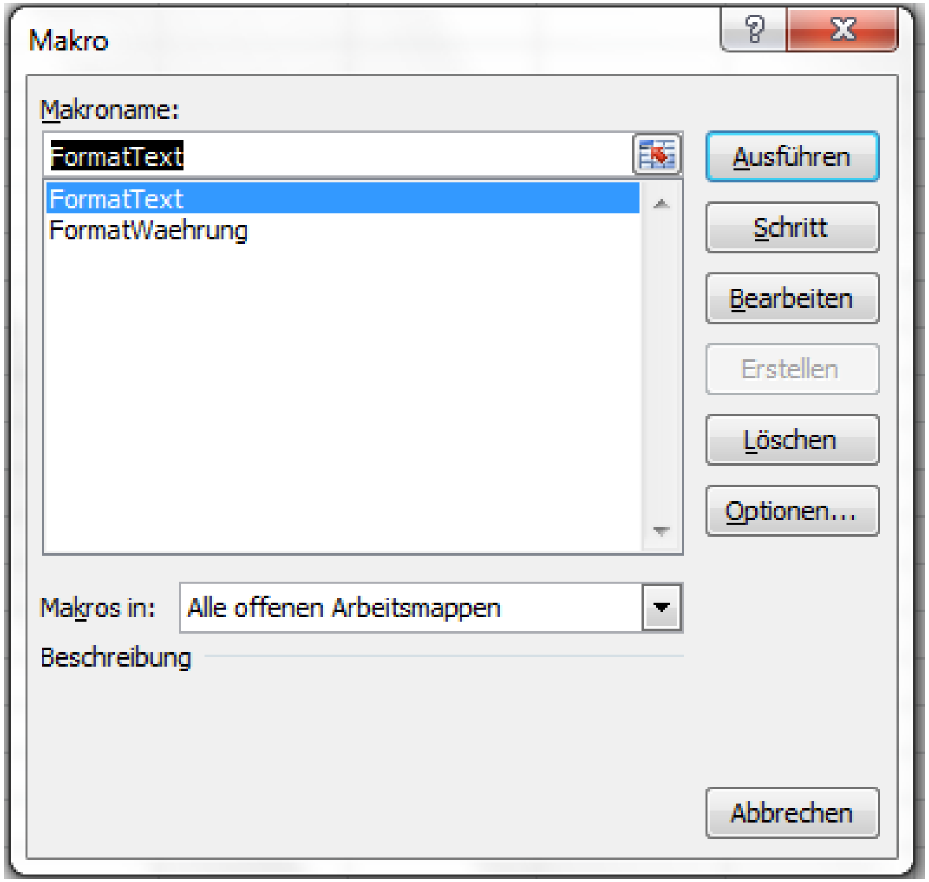
- automatisiert eine Folge von Arbeitsschritten.
- ist eine Programmiersprache, die in jeder Office-Anwendung eingebettet ist.
- erweitert die Funktionalität von Office-Anwendungen.
- passt eine Anwendung entsprechend der Wünsche des Benutzers an.

Start des Makros ...

- mit Hilfe des Symbols *Makros* im Bereich Code des Menübandes Entwicklertools.
- mit Hilfe von <CTRL> und der eingegebenen Taste.
- über ein Symbol in einem benutzerdefinierten Menüband.
- Nach dem Start werden die aufgezeichneten Arbeitsschritte abgearbeitet.

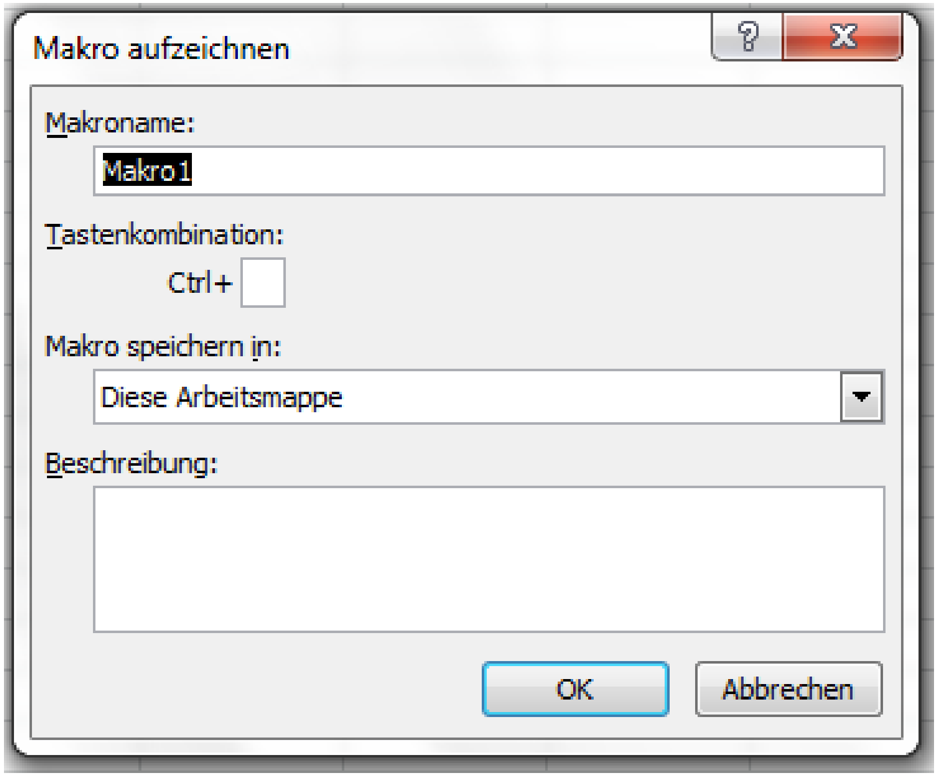
Entwicklertools - Makros

- Mit einem Mausklick wird ein Makro aus der Liste ausgewählt.
- Die Schaltfläche *Ausführen* startet das gewählte Makro.



… mit Hilfe einer Tastenkombination starten

- <CTRL>+<Taste> startet das Makro.



… über ein Menüband starten

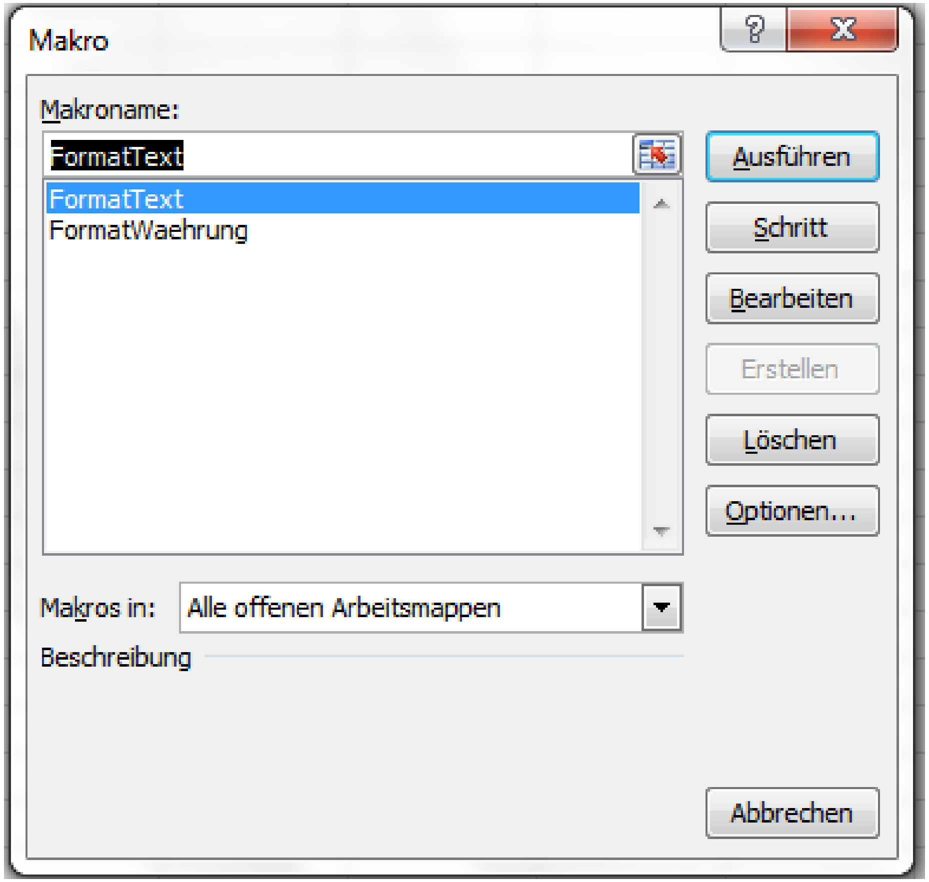
- Voraussetzungen: Das Makro ist als Icon in einem Menüband sichtbar.
- Mit einem Klick auf das passende Icon wird das Makro gestartet.

Arbeitsmappen mit einem Makro speichern

- *Datei – Speichern unter*.
- Als Dateityp wird der Eintrag Excel-Arbeitsmappe mit Makros (*.xlsm) genutzt.
- Für die Speicherung wird ein aussagekräftiger Name eingegeben.
- Ein Ordner wird als Speicherplatz ausgewählt.

Makros bearbeiten

- Das Menüband Entwicklertools ist aktiv.
- In der Gruppe Code wird auf das Symbol *Makros* geklickt.
- Die Schaltfläche *Bearbeiten* öffnet den VBA-Editor. Alle Schritte des Makros werden in VBA-Code dargestellt.
- Mit Hilfe der Schaltfläche *Optionen* werden die Voreinstellungen angezeigt und können verändert werden.



Aufgezeichneter Code

```
Sub PivotTable_Macro()
    Sheets.Add

    ActiveWorkbook.PivotCaches.Create(SourceType:=xlDatabase, SourceData:= _
        "AnzahlTiere!R6C1:R34C7", Version:=xlPivotTableVersion14).CreatePivotTable _
        TableDestination:="Tabelle1!R3C1", TableName:="PivotTable1", _
        DefaultVersion:=xlPivotTableVersion14

    Sheets("Tabelle1").Select
    Cells(3, 1).Select
```

Objekt-Sammlung „Sheets“ ...

- enthält alle Arbeitsblätter und Diagrammblätter einer Arbeitsmappe. Falls keine Arbeitsmappe angegeben ist, wird die aktive Arbeitsmappe genutzt.
- wird mit Hilfe der Methode .Add ein Tabellenblatt hinzugefügt.
- Sheets("[name]") identifiziert eindeutig ein Blatt in einer Arbeitsmappe. Der Name wird in der Arbeitsmappe auf dem Tabellenreiter angezeigt. Mit Hilfe von .Select wird ein Blatt ausgewählt. Die Methode .Activate aktiviert das Blatt.

Objekt-Sammlung „Cells“ ...

- enthält alle Zellen auf einem Arbeitsblatt. Falls kein Arbeitsblatt angegeben ist, bezieht sich die Zellangabe auf das aktive Arbeitsblatt.
- Cells([spalte], [zeile]) identifiziert eindeutig eine Zelle auf einem Arbeitsblatt.
- Mit Hilfe von .Select wird eine Zelle ausgewählt.

Zwischenspeicher anfordern

```
ActiveWorkbook.PivotCaches.Create(SourceType:=xlDatabase, SourceData:= _
                        "AnzahlTiere!R6C1:R34C7", Version:=xlPivotTableVersion14)
```

- ActiveWorkbook bezieht sich auf die aktive Arbeitsmappe.
- In dieser Arbeitsmappe wird Zwischenspeicher (PivotCaches) für eine Pivot-Tabelle angefordert (Create).
- Eltern-Objekte und deren Kind-Objekte werden wie Methoden mit Objekten durch ein Punkt verbunden.
- Der Methode .Create werden in der runden Klammern Eingabeparameter als benannte Argumente übergeben:
  - Von welchen Typ ist die Datenquelle?
  - Wo befindet sich die Datenquelle?
  - In welcher Excel-Version wird die Pivot-Tabelle erzeugt?

Benannte Argumente

SourceData:= "AnzahlTiere!R6C1:R34C7

- 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 ein String (Zeichenkette) genutzt. Strings werden immer durch Anführungszeichen begrenzt.

Pivot-Tablle erstellen

```
.CreatePivotTable TableDestination:="Tabelle1!R3C1", TableName:="PivotTable1", _
        DefaultVersion:=xlPivotTableVersion14
```

- In welcher Zelle beginnt die Tabelle?
- Name der Pivot-Tabelle.
- Standardversion der Pivot-Tabelle

Aufgezeichneter Code

```
 With ActiveSheet.PivotTables("PivotTable1").PivotFields("Tierarten")
        .Orientation = xlRowField
        .Position = 1
    End With

    ActiveSheet.PivotTables("PivotTable1").AddDataField ActiveSheet.PivotTables( _
        "PivotTable1").PivotFields("2006"), "Anzahl von 2006", xlCount
    ActiveSheet.PivotTables("PivotTable1").AddDataField ActiveSheet.PivotTables( _
        "PivotTable1").PivotFields("2007"), "Anzahl von 2007", xlCount
End Sub
```

Objekthierarchie

- Workbook (Arbeitsmappe)
  - Worksheet (Tabellen- / Arbeitsblatt)
    - PivotTables (Pivot-Tabellen)
      - PivotFields (Pivot-Tabellenfelder)
        - PivotItems (Pivot-Tabellenfelddaten)
  - Worksheet (Tabellen- / Arbeitsblatt)
    - Cells (Zellen)

Objektbezeichnungen in dem Code-Beispiel

```
With ActiveSheet.PivotTables("PivotTable1").PivotFields("Tierarten")
    .Orientation = xlRowField
    .Position = 1
End With
```

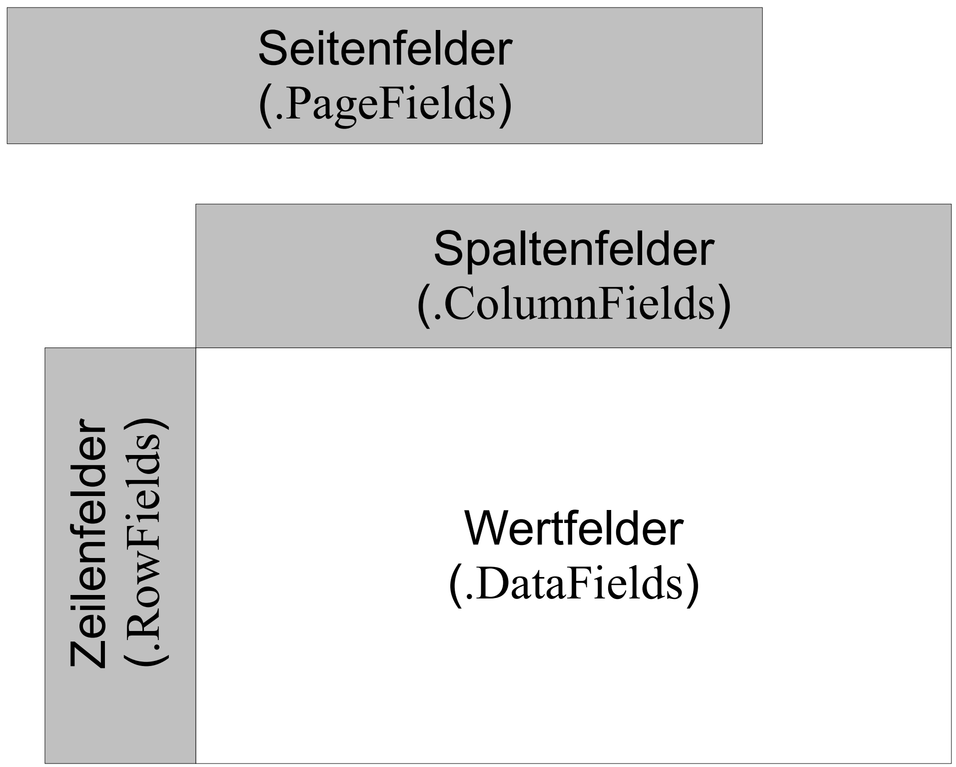
- ActiveSheet ist ein Synonym das aktuell aktive Tabellenblatt.
- PivotTables("PivotTable1") bezeichnet eindeutig eine Pivot-Tabelle.
- PivotFields("Tierarten") identifiziert ein Tabellenfeld in eine Pivot-Tabelle.

Tabellenfelder einen Bereich zuweisen

```
With ActiveSheet.PivotTables("PivotTable1").PivotFields("Tierarten")
    .Orientation = xlRowField
    .Position = 1
End With
```

- Die Methode .Orientation legt die Ausrichtung der Tabellenfelder fest.
    - xlRowField ordnet die Daten in Zeilen an.
    - xlColumnField ordnet die Daten in Spalten an.
    - xlDataField ordnet die Daten den berechnenden Bereich zu.
- Die Methode .Position bestimmt die Reihenfolge der Tabellenfelder.

Felder in einem bestimmten Bereich



Tabellenfelder berechnen

```
ActiveSheet.PivotTables("PivotTable1").AddDataField
        ActiveSheet.PivotTables("PivotTable1").PivotFields("2006"),
       "Anzahl von 2006", xlCount
```

- Mit Hilfe der Methode .AddDataField wird ein berechnender Datenbereich erzeugt.
- Der Methode werden folgende Parameter übergeben:
  - Datenquelle für die Funktion.
  - Name des Bereichs.
  - Genutzte Funktion. In diesem Beispiel wird die Anzahl berechnet. (siehe in der Hilfe: XlConsolidationFunction-Enumeration)

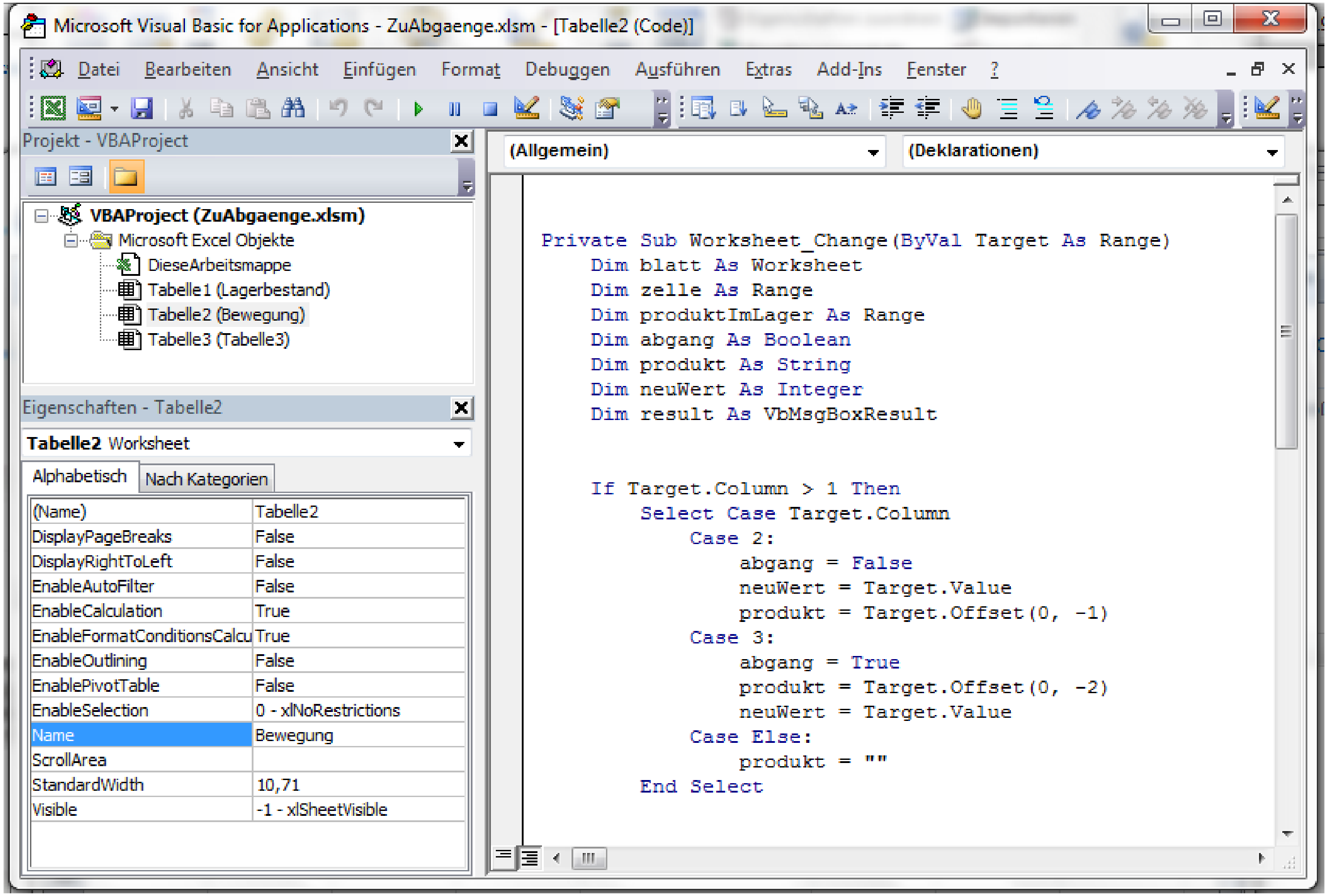
Nachteile des Codes

- Die Bereitstellung des Zwischenspeichers erzeugt immer den Laufzeitfehler 1004.
- Die Datenquelle ist fest verdrahtet. Die Datenquelle kann über ein Dialogfeld abgefragt werden. (siehe „excel_SparklinesDialog.pdf“)

Eigenen Code im VBA-Editor erzeugen

- Das Menüband Entwicklertools ist aktiv.
- Mit Hilfe des Symbols *Visual Basic* 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 ...

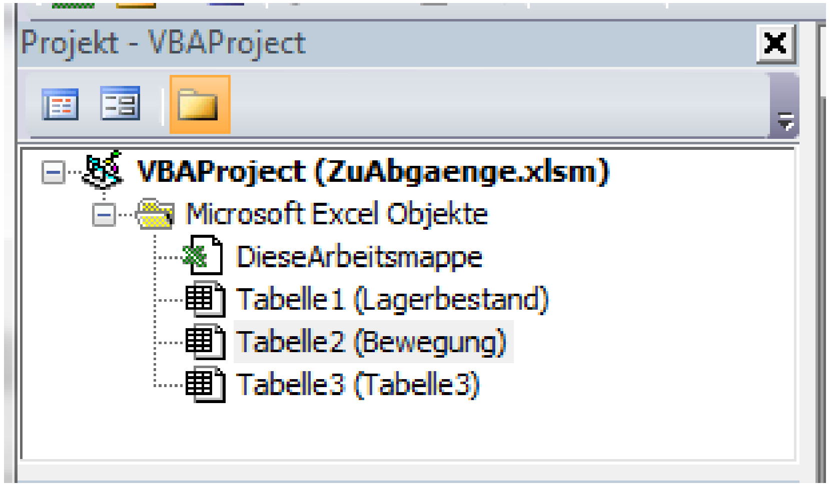
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle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

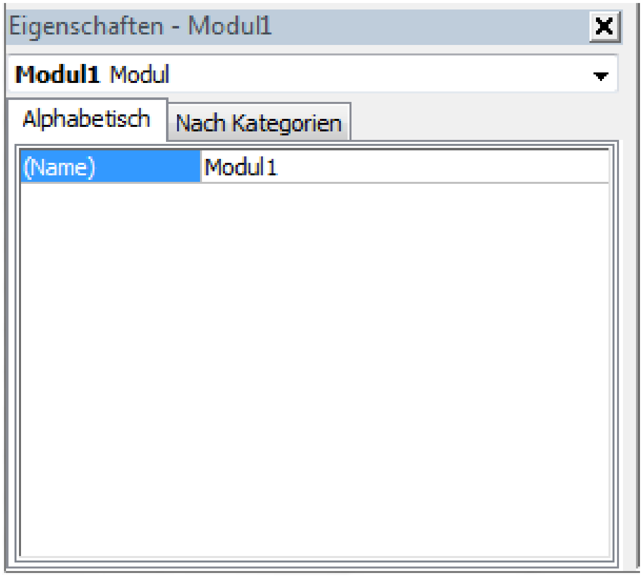
- In der Titelleiste wird die Schließen-Schaltfläche angezeigt.
- Darunter befindet sich die Symbolleiste mit den Icons
  - „Zeige zum gewählten Element den Code an".
  - „Zeige das passende Objekt zum Code an".
  - „Sortierung mit Hilfe von Ordnern".
- Unterhalb der Symbolleiste werden die Module alphabetisch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

Eigenes Modul 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

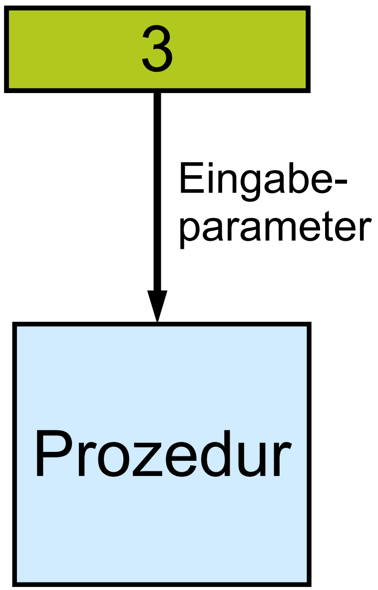
```
Sub NewPivotTable()


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.

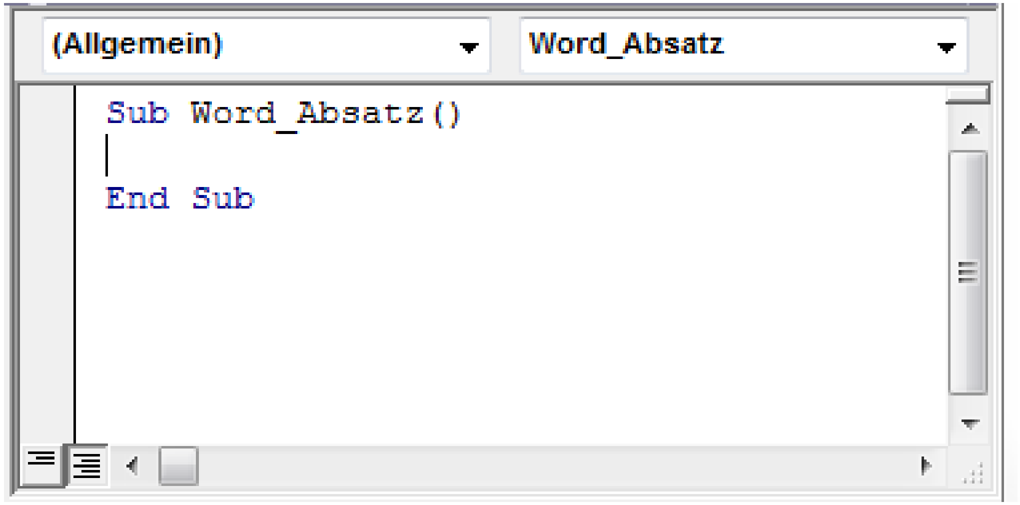
Arbeitsweise einer Prozedur

- 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 NewPivotTable()


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.

Variablen

```
Sub NewPivotTable()
  Dim ptName 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 ptName As String

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

… zuweisen

ptName = "AnzahlTiere"

- Mit Hilfe des Gleichheitszeichens wird einer Variable ein Wert zugewiesen. Der Wert kann durch ein Ausdruck berechnet werden.
- In diesem Beispiel wird der Variablen ptName ein Text zugewiesen. Ein Text wird immer durch Anführungszeichen begrenzt.

Verweise auf Objekte in VBA

```
Sub NewPivotTable()

   Dim ptCache As PivotCache
   Dim ptTable As PivotTable
   Dim xlsSheet As Worksheet

End Sub
```

Objektvariablen ...

- verweisen auf ein bestimmtes Objekt in Excel.
- enthalten eine Referenz auf die Excel-Anwendung selbst oder auf Elemente in einer Arbeitsmappe.
- werden für Objekte definiert, die häufig in einem Programm genutzt werden.
- In dem vorherigen Beispiel ist die Variable xlsSheet vom Datentyp Worksheet (Arbeitsblatt).

… deklarieren

```
Dim xlsSheet As Worksheet
```

Zugriff objektvariablename As Objekttyp

- Jede Objektvariable hat einen eindeutigen Namen. In diesem Beispiel xlsSheet.
- Objektvariablen sind von einem bestimmten Typ „Objekt“ (Worksheet; Arbeitsblatt).
- Auf diese Variable kann nur zwischen Sub und End Sub zugegriffen werden. Es ist eine private Variable der Ereignisprozedur.

Verweis auf ein Objekt speichern

```
Set xlsSheet = ActiveWorkbook.Sheets.Add(After:=ActiveSheet)
Set objektvariablename = Objektverweis
```

- Die Zuweisung muss mit dem Schlüsselwort Set eingeleitet werden.
- Mit Hilfe des Gleichheitszeichens wird der Variablen ein Verweis auf ein bestimmtes Objekt zugewiesen.
- In diesem Beispiel verweist die Variable xlsSheet auf ein neu erstelltes Arbeitsblatt in aktiven Arbeitsmappe.

Hierarchie von Objekten

```
Set blatt = ActiveWorkbook.Sheets
```

- Eine Arbeitsmappe besteht aus mindestens ein Arbeitsblatt.
- Diese Hierarchie spiegelt sich auch in dem Verweis wieder.
- Das übergeordnete Objekt wird mit dem untergeordneten Objekt durch einen Punkt verbunden.

Blätter einer Arbeitsmappe ...

- werden in der Auflistung Sheets gesammelt. Über diese Auflistung kann einer Arbeitsmappe ein neues Blatt hinzugefügt werden.
- können Arbeitsblätter (Worksheet) sein. Arbeitsblätter bestehen aus Zeilen und Spalten.
- können Diagrammblätter (Chart) sein. Das Diagramm wird als DIN A4-Blatt angezeigt. Diagrammblätter werden für Präsentationen oder Ausdrucke genutzt.

Arbeitsblätter in VBA

```
Dim arbeitsblatt As Worksheet

' Das aktive Blatt
Set xlsSheet = ThisWorkbook.ActiveSheet

' Verweis auf das Arbeitsblatt TB1
SetxlsShee = ThisWorkbook.Worksheets("TB1")

' Verweis auf das Arbeitsblatt TB1 in der Auflistung
' aller Blätter in der aktiven Arbeitsmappe
Set xlsSheet = ActiveWorkbook.Sheets("TB1")
```

… erstellen

```
Set xlsSheet = ActiveWorkbook.Sheets.Add(After:=ActiveSheet)


xlsSheet.Name = "PivotTable_VBA"
xlsSheet.Activate
```

- Mit Hilfe der Methode .Add wird der Liste alle Blätter (.Sheets) ein neues Blatt hinzugefügt.
- In den runden Klammern wird der Methode die Einfügeposition übergeben. In diesem Beispiel wird das neue Blatt nach dem aktiven Arbeitsblatt eingefügt.

Eigenschaften und Methoden

```
Set xlsSheet = ActiveWorkbook.Sheets.Add(After:=ActiveSheet)


xlsSheet.Name = "PivotTable_VBA"
xlsSheet.Activate
```

- Die Eigenschaften .Name wird ein Text als Bezeichnung für das Arbeitsblatt zugewiesen. Falls der Name in der Auflistung aller Arbeitsblätter vorhanden ist, wird eine Fehlermeldung eingeblendet.
- Mit Hilfe der Methode .Activate wird ein Tabellenblatt aktiviert.

Verweise für Pivot-Tabellen

```
Sub NewPivotTable()
  ' Objektvariable für den Zwischenspeicher
  Dim ptCache As PivotCache

  ' Verweis auf eine Pivot-Tabelle
  Dim ptTable As PivotTable

End Sub
```

Zwischenspeicher anfordern

```
Set ptCache = ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, _
                    SourceData:="AnzahlTiere!R6C1:R34C7")
```

- Das Objekt PivotCaches ist ein Synonym für den Zwischenspeicher der Pivot-Tabellendaten.
- In diesem Beispiel wird der Zwischenspeicher in Bezug auf die aktive Arbeitsmappe mit Hilfe der Methode .Add angefordert.
- In diesem Beispiel wird eine Excel-Tabelle / Datenbank (xlDatabase) als Quelle genutzt. Mit Hilfe des Arguments SourceData wird die Quelle exakt angegeben.

Pivot-Tabelle erstellen

```
Set ptTable = ptCache.CreatePivotTable(TableDestination:=xlsSheet.Range("C1"), _
                                       TableName:=ptName)
```

- Mit Hilfe der Methode .CreatePivotTable wird eine Pivot-Tabelle erstellt.
- Der Methode wird die Anfangsposition der Tabelle (TableDestination) und eine Bezeichnung (TableName) übergeben.

Zeilenbeschriftungen setzen

```
With ptTable.PivotFields("Tierarten")
        .Orientation = xlRowField
        .Position = 1
End With
```

- In der Quelle für die Pivot-Tabelle ist eine Spaltenüberschrift „Tierarten“ vorhanden. Diese Überschrift wird in der PivotTable-Feldliste angezeigt.
- Dieses Feld wird als Zeilenbeschriftung (.Orientation = xlRowField) genutzt.
- Mit Hilfe der Eigenschaft .Position wird die Reihenfolge festgelegt.

Zeilenbeschriftungen setzen

```
With Objekt
        .Methode/Eigenschaft = Wert

End With
```

- Mit Hilfe der Anweisung With und End With werden Anweisungen zu einem Objekt zusammengefasst.
- Dem Schlüsselwort With folgt ein Objekt. Für dieses Objekt werden die nachfolgenden Anweisungen gesammelt.
- Elemente, die mit einem Punkt beginnen, beziehen sich immer auf das im Kopf angegebene Objekt.

Werte hinzufügen

```
ptTable.AddDataField ptTable.PivotFields("2006"), "Anzahl Tiere 2006", xlCount
```

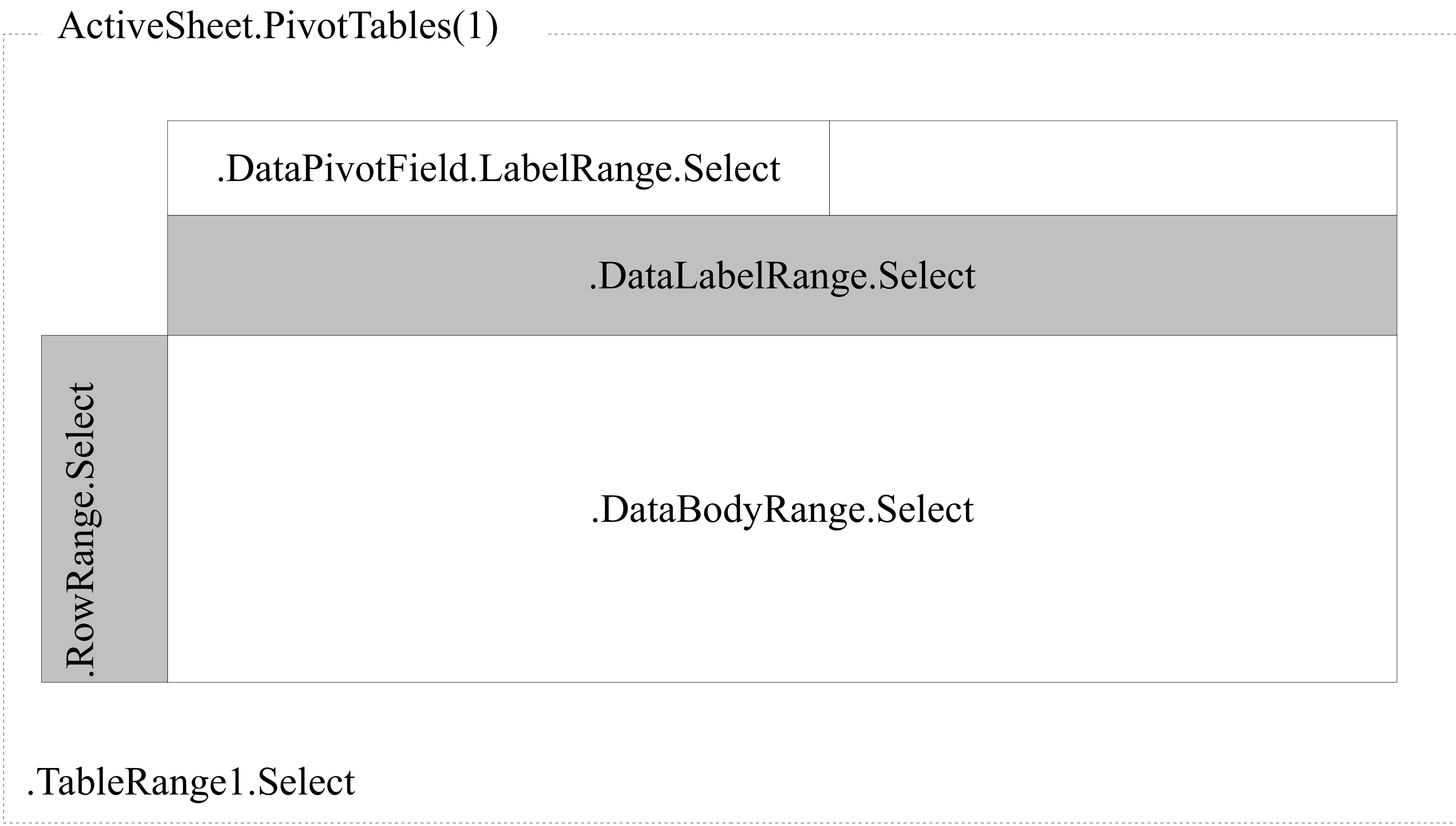
- Mit Hilfe der Methode .AddDataField wird ein berechnender Datenbereich erzeugt.
- Der Methode werden folgende Parameter übergeben:
  - Aus welchen Daten soll der Wert berechnet werden?
  - Überschrift für den Bereich. Name des Wertes.
  - Wie soll der Wert berechnet werden? In diesem Beispiel wird die Anzahl der Tiere berechnet. (siehe in der Hilfe: XlConsolidationFunction-Enumeration)

Werte als Spaltenüberschriften nutzen

```
With ptTable.DataPivotField
    .Orientation = xlColumnField
    .PivotItems(2).Position = 1
    .PivotItems(1).Position = 2
End With
```

- Das Objekt .DataPivotField erstellt alle Datenfelder (alle berechneten Werte) dar.
- Die Namen werden als Spaltenüberschriften (.Orientation = xlColumnField) genutzt.
- Das Objekt .PivotItems(1) identifiziert mit Hilfe der Ganzzahl eindeutig ein Datenfeld. Mit Hilfe der Eigenschaft .Position wird die Spaltenreihenfolge der Datenfelder festgelegt.

Elemente der Pivot-Tabelle auswählen



… formatieren

```
With ptTable.DataPivotField

    With .LabelRange
        .Value = "Jahresangaben"                ' Beschriftung der Datenfelder als „Überschrift“
    End With

    With .DataRange                             ' Spaltenüberschriften der Datenfelder
        .Interior.ColorIndex = 34               ' Hintergrundfarbe der Zellen
        .Font.Italic = True                     ' Kursiv-Schrift nutzen
        .Font.Name = "Arial"                    ' Schriftart setzen
        .Font.Color = vbBlack                   ' Schriftfarbe setzen
        Columns.AutoFit                         ' Automatische Spaltengrößen-Anpassung

    End With
End With
```