R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Auf Änderungen in einem Arbeitsblatt reagieren

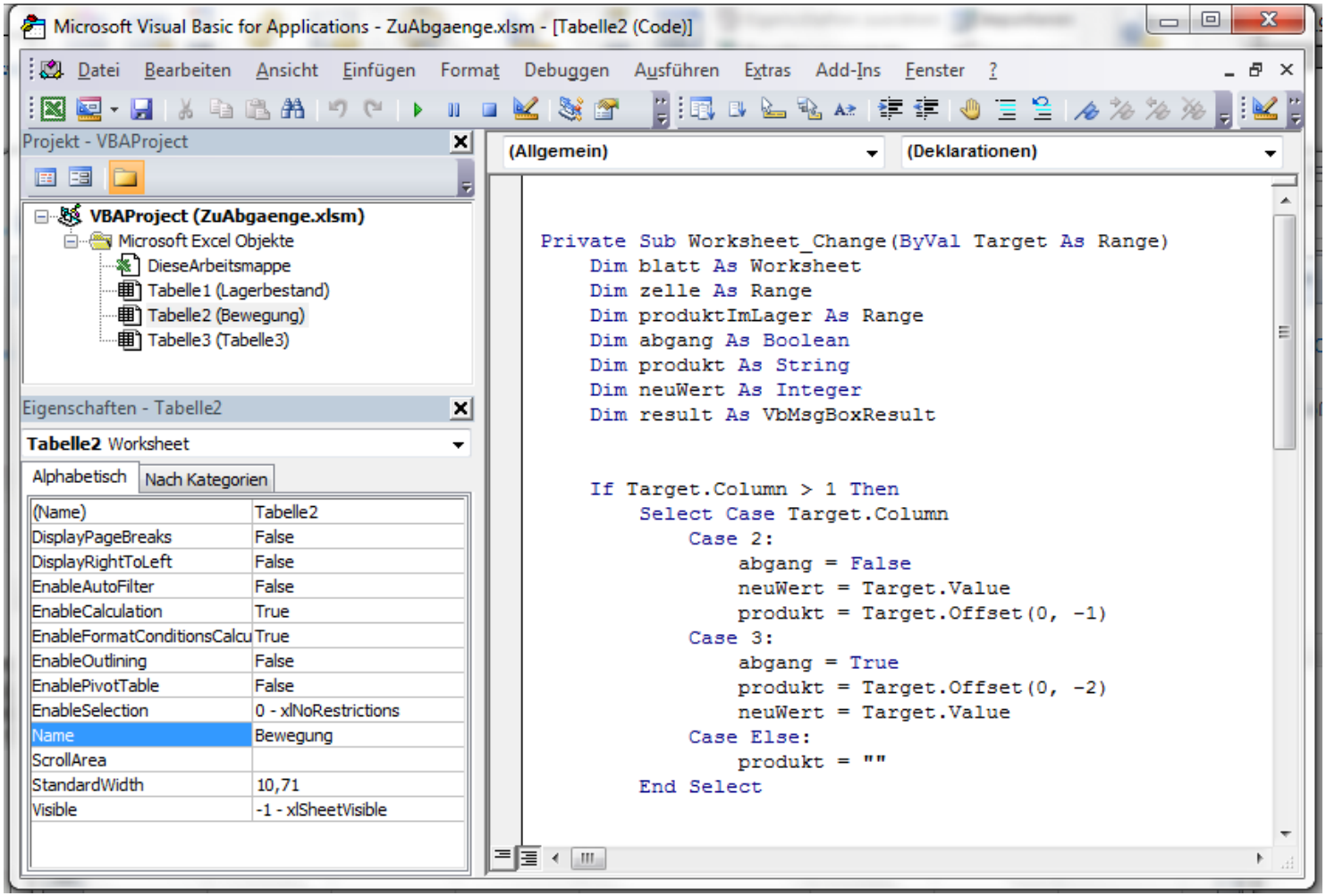
Aufgabe

- Der Benutzer trägt in dem Arbeitsblatt „Bewegung“ die Zu- und Abgänge eines Artikels ein.
- Sobald der Benutzer die Eintragung mit <RETURN> abgeschlossen hat, werden die Lagerbestände in dem Arbeitsblatt „Lagerbestand“ automatisiert angepasst.
- Hinweis: Diese Aufgabe ist nicht durch die Aufzeichnung eines Makros lösbar.

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 ...

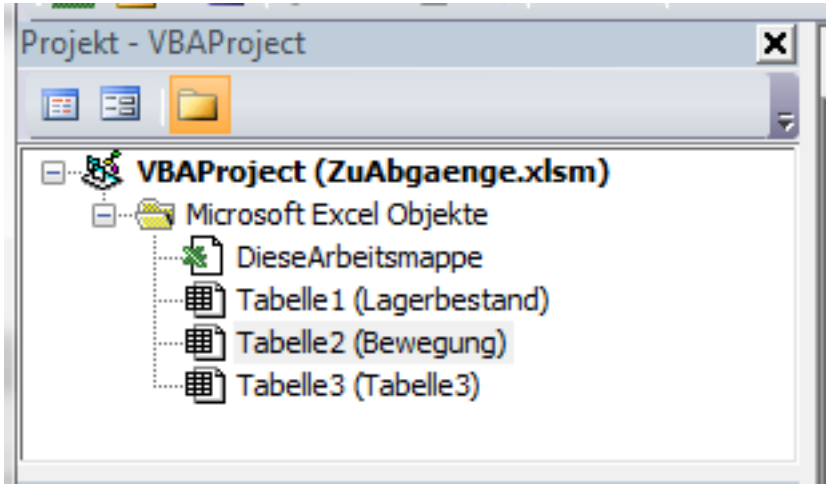
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle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

Modul-Typen ...

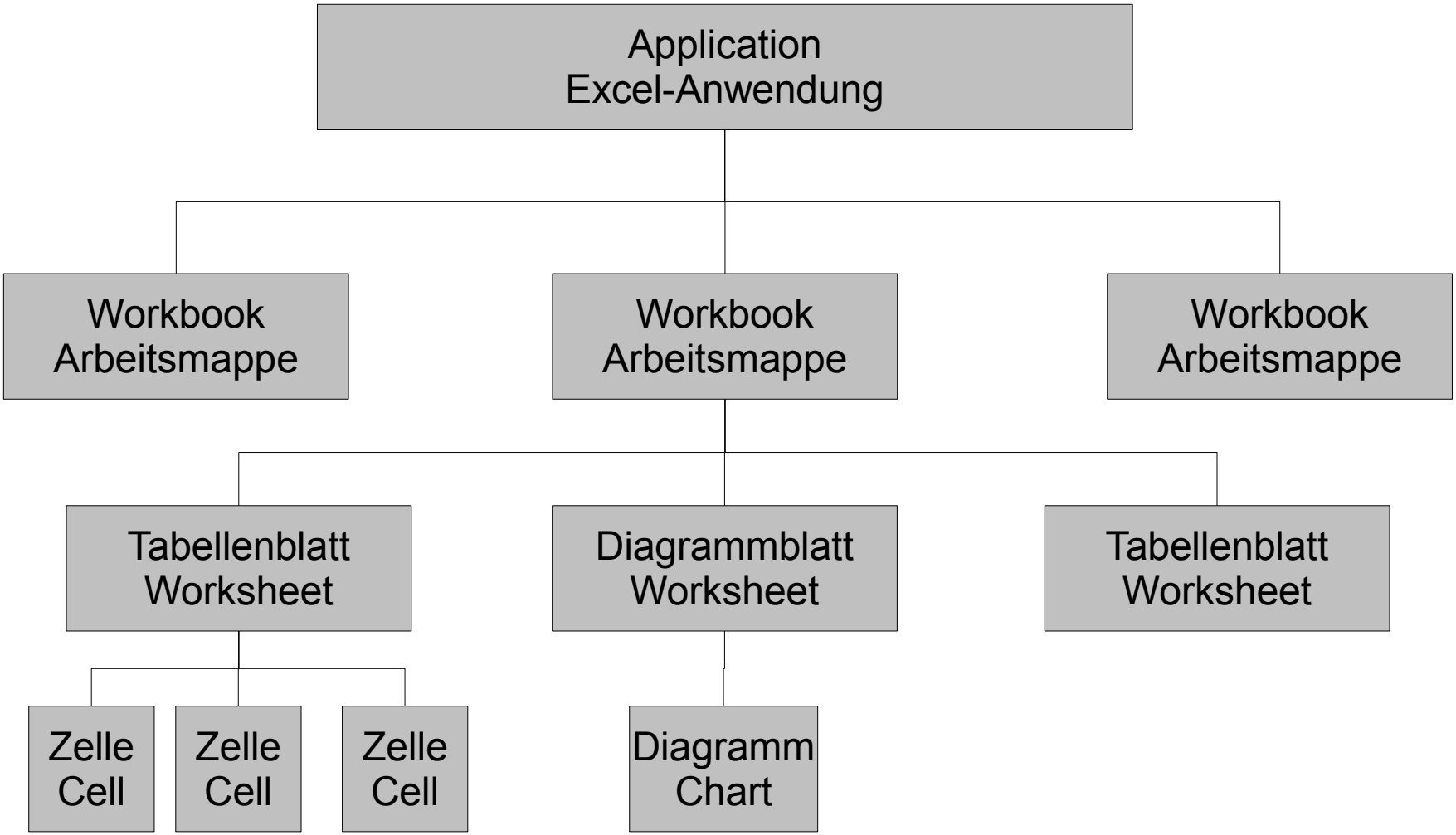
- werden mit Hilfe von Ordnern im Projekt-Explorer dargestellt.
- (Microsoft Excel Objekte) enthalten Module, die Code für die Arbeitsmappe oder die darin enthaltenen Arbeitsblätter enthalten
- (Module) enthalten aufgezeichnete Makros oder vom Entwickler selbst geschriebene Prozeduren.

Excel-Objekte im Projekt-Explorer

- 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 runden Klammer wird der Name des jeweiligen Tabellenblattes angezeigt.

Excel-Objektmodell



Excel-Objekte in VBA

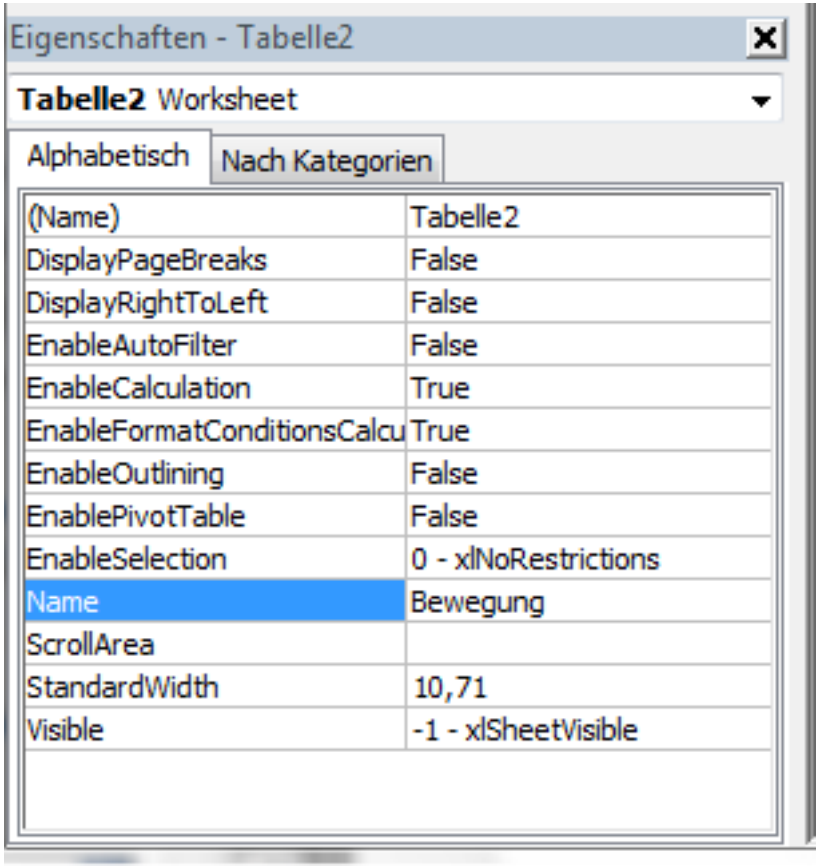
- Die Arbeitsmappe (Workbook) besteht aus mindestens drei Arbeitsblättern (Worksheet).
- Arbeitsblätter sind Tabellenblätter. Tabellenblätter bestehen aus Zellen (Cell). In diese Zellen können Diagramme (ChartObject) eingebettet werden.
- Diagrammblätter (Chart) in einer Arbeitsmappe stellen Diagramme im A4-Format dar.

Eigenschaften (Attribute) ...

- beschreiben ein Objekt eindeutig.
- verändern das Layout eines Objekts.
- werden teilweise im Eigenschaftenfenster des VBA-Editors angezeigt.
- können in den meisten Fällen mit Hilfe von VBA verändert werden.
- werden durch ein Punkt mit dem dazugehörigen Objekt verbunden.

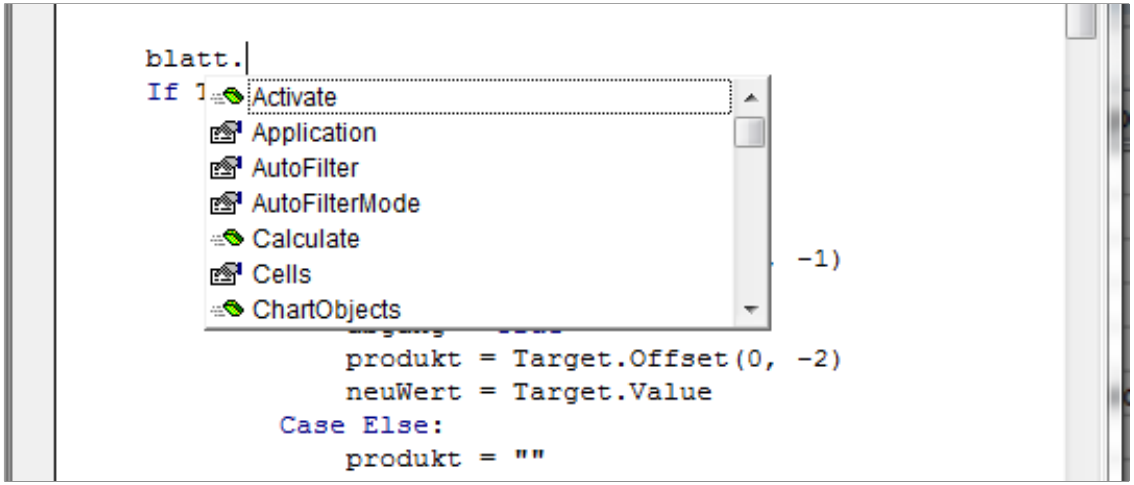
… im Eigenschaftenfenster

- Das Eigenschaftenfenster kann über das Menü *Ansicht* ein- oder ausgeblendet werden.
- Im oberen Kombinationsfeld wird der Name des Objekts angezeigt.
- Die Eigenschaften dieses Objekts werden darunter zeilenweise angezeigt.



… mit Hilfe von Microsoft Intellisense angeben

- Sobald der Punkt eingegeben ist, werden alle Eigenschaften des dazugehörigen Objekts in einer Liste angezeigt.
- Eigenschaften werden mit dem Finger, der auf den geschrieben Text zeigt, gekennzeichnet.
- Mit einem Doppelklick wird eine Eigenschaft aus der Liste in das Codefenster übernommen.

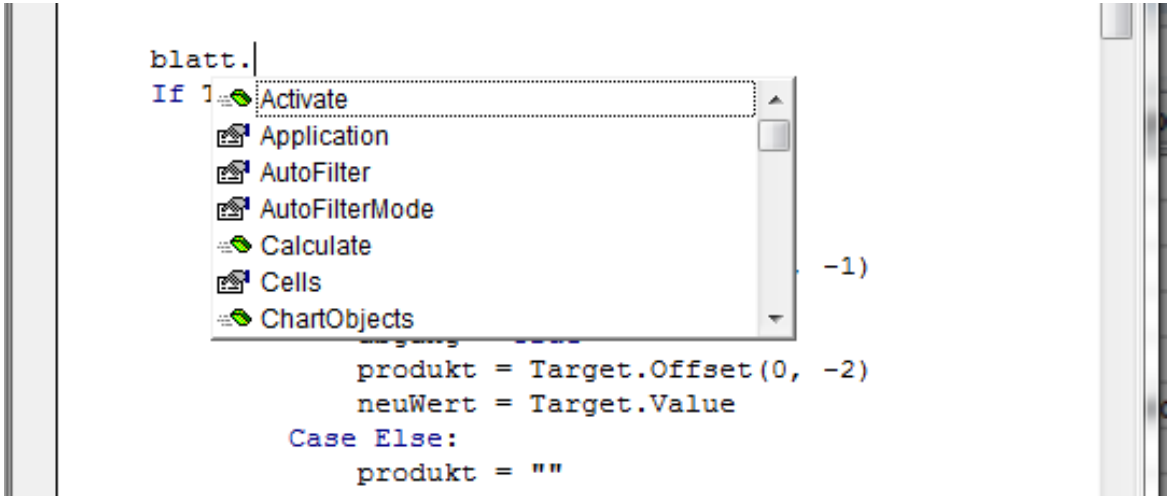


Methoden ...

- sind vordefinierte Sammlungen von Arbeitsschritten für die Veränderung von Attributwerten.
- beeinflussen das Layout oder Verhalten eines Objekts.
- können von außen aufgerufen werden.
- bekommen manchmal Parameter übergeben, die den Ablauf des Codes beeinflussen.
- geben manchmal einen Statuswert an den Aufrufer zurück.
- werden mit dem dazugehörigen Objekt durch einen Punkt verbunden.

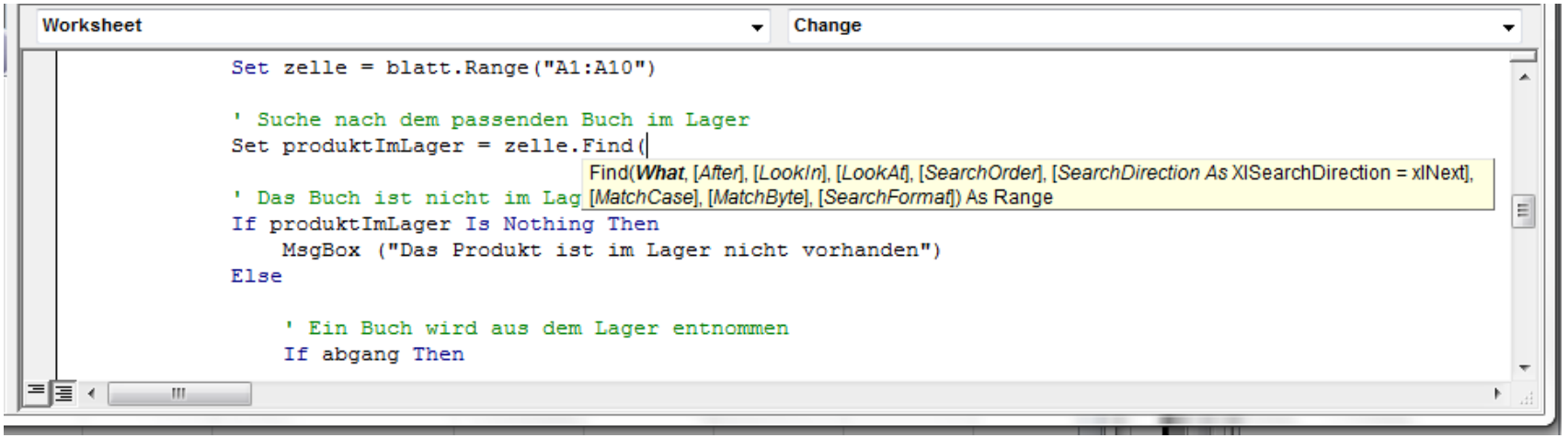
.. mit Hilfe von Microsoft Intellisense auswählen

- Sobald der Punkt eingegeben ist, werden alle Methoden des dazugehörigen Objekts in einer Liste angezeigt.
- Methoden werden mit dem grünen Radiergummi gekennzeichnet.
- Mit einem Doppelklick wird eine Methode aus der Liste in das Codefenster übernommen.



Hilfe zur Methode

- Nach dem Methodennamen wird die runde Klammer auf angegeben.
- Anschließend wird in einem gelben Erklärfenster der Rückgabewert sowie die Eingabeparameter angezeigt.



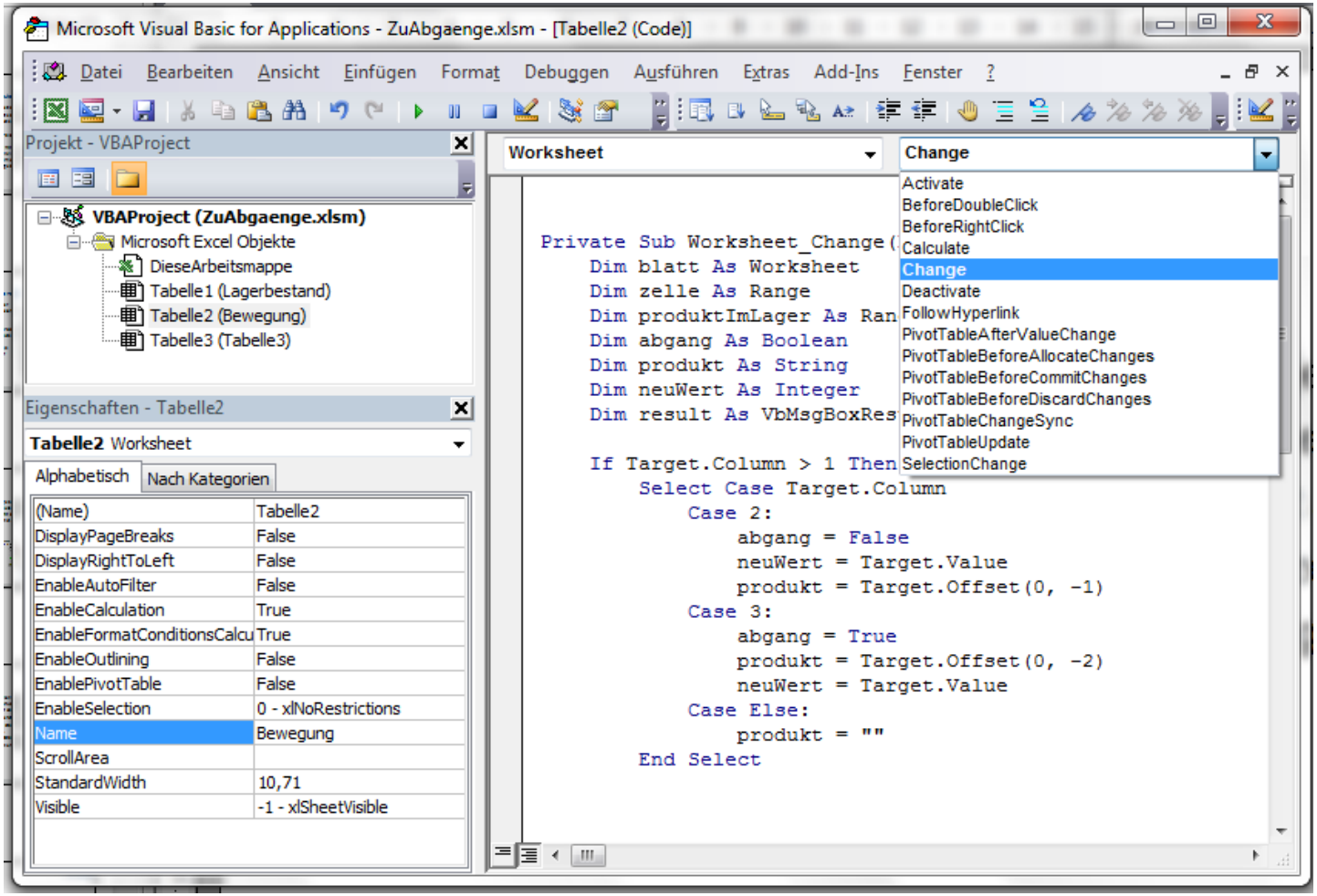
Ereignisse ...

- sind Aktionen, die in Verbindung mit einem Objekt steht.
- werden durch die Tastatur oder die Maus ausgelöst
- werden durch Veränderungen an einem Objekt ausgelöst.
- werden im Codefenster des VBA-Editors angezeigt.
- Auf Ereignisse kann mit Hilfe sogenannter Ereignisprozeduren reagiert werden.

… im VBA-Editor

- Das gewünschte Excel-Objekt wird im Projekt-Explorer ausgewählt.
- Der Code zu dem Objekt wird angezeigt.
- In dem Kombinationsfeld Objekte wird der passende Name des Objekts ausgewählt.
- In dem Kombinationsfeld Prozeduren werden alle dazugehörigen Ereignisse angezeigt.
- Mit Hilfe eines Mausklicks wird ein Ereignis ausgewählt. Falls das Ereignis nicht vorhanden ist, wird das Grundgerüst für eine Ereignisprozedur erstellt.

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
Worksheet
Change
Activate
BeforeDoubleClick
BeforeRightClick
Calculate
Change
Deactivate
FollowHyperlink
PivotTableAfterValueChange
PivotTableBeforeAllocateChanges
PivotTableBeforeCommitChanges
PivotTableBeforeDiscardChanges
PivotTableChangeSync
PivotTableUpdate
SelectionChange
Private Sub Worksheet_Change(
Dim blatt As Worksheet
Dim zelle As Range
Dim produktImLager As Ran
Dim abgang As Boolean
Dim produkt As String
Dim neuWert As Integer
Dim result As VbMsgBoxRes
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

Auflistungen von Objekten

- Zum Beispiel: Die Auflistung
  - Worksheets enthält alle Arbeitsblätter einer Arbeitsmappe.
  - Cells ist eine Auflistung aller Zellen eines Arbeitsblattes.
- Auflistungen werden in VBA als Collections bezeichnet.
- Collections sind Sammlungen von bestimmten Objektarten.
- Der Name einer Collection endet immer mit einem „s“.

Elemente in einer Collection identifizieren

- Zum Beispiel: `Worksheets("Tabelle1"), Cells(1).`
- Die Elemente in einer Auflistung werden eindeutig durch die Angabe eines Indizes identifiziert.
- Als Index kann eine Ganzzahl oder der Name des Objekts genutzt werden.

Ganzzahl als Index nutzen

- Zum Beispiel: `Worksheets("Tabelle1"), Cells(1).`
- Das erste Element hat den Index eins, das zweite Element den Index zwei und so weiter.
- In diesem Beispiel wird die erste Zelle auf einem Tabellenblatt angesprochen.
- Der ganzzahlige Index ist von der Anzahl der Elemente in der Auflistung abhängig. Sobald ein Objekt nicht mehr zur Verfügung steht, wird das Objekt aus der Auflistung gelöscht und die Auflistung neu durchnummeriert. Alle Objekte nach dem gelöschten Objekt bekommen einen neuen Index.

Namen als Index nutzen

- Zum Beispiel: `Worksheets("Tabelle1"), Cells(1).`
- Als Index wird der eindeutige Name des Elements genutzt.
- Der Name wird durch Anführungsstriche begrenzt. Der Name ist ein beliebiger Text in VBA.
- Diese Art von Index ist nicht von der Position des Elements in der Auflistung abhängig.

Arbeitsmappe (Workbook) ...

- ist ein Container für alle Arbeitsblätter in Excel.
- besteht standardmäßig aus drei Arbeitsblättern.
- fasst den Inhalt einer Excel-Datei zusammen.
- wird mit der Dateiendung "xls", „xlsx“ (ab Excel 2007) oder ".xlsm" (Excel-Dateien mit Makro ab Excel 2007) gespeichert.

… in VBA

```
Dim arbeitsmappe As Workbook

' Arbeitsmappe, die mit VBA verbunden ist
Set arbeitsmappe = ThisWorkbook

' Die aktive Arbeitsmappe
Set arbeitsmappe = ActiveWorkbook
```

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 arbeitsmappe vom Datentyp Workbook (Arbeitsmappe).

… deklarieren

Dim arbeitsmappe As Workbook

Zugriff objektvariablename As Objekttyp

- Jede Objektvariable hat einen eindeutigen Namen. In diesem Beispiel arbeitsmappe.
- Objektvariablen sind von einem bestimmten Typ „Objekt“ (Workbook; Arbeitsmappe).
- 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 arbeitsmappe = ThisWorkbook
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 `arbeitsmappe` auf die zu dem Code gehörende Arbeitsmappe in der geöffneten Excel-Anwendung.

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 arbeitsblatt = ThisWorkbook.ActiveSheet

' Verweis auf das Arbeitsblatt TB1
Set arbeitsblatt = ThisWorkbook.Worksheets("TB1")
```

Ereignisse des Arbeitsblatts anzeigen

- Im Kombinationsfeld Objekte des Codefensters ist der Eintrag Worksheet markiert.
- Im Kombinationsfeld Prozeduren werden alle, zu dem Arbeitsblatt gehörende Ereignisse angezeigt.
- Mit einem Klick auf ein Ereignis wird ...
  - das Grundgerüst der Prozedur, falls nicht vorhanden, angelegt.
  - der Code zu dem Ereignis aufgerufen.

Beispiele

- Acitivate und Deactivate werden durch das Aktivieren und Deaktivieren des Arbeitsblatts ausgelöst.
- Nach der Neuberechnung des Arbeitsblattes wird Calculate ausgelöst.
- Wenn sich der Inhalt einer Zelle verändert, wird Change ausgelöst.
- Wenn Zellen neu markiert werden, wird SelectionChange ausgelöst.

Ereignisprozeduren ...

- reagieren auf die Auslösung eines Ereignisses.
- beeinflussen die Veränderung eines Objekts in Abhängigkeit eines bestimmten Ereignisses.
- sind immer mit einem bestimmten Objekt verbunden.
- haben als Argument häufig einen Hinweis auf den Auslöser des Ereignisses.
- geben keinen Statuswert an den Aufrufer zurück.

Aufbau einer Ereignisprozedur

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Anweisung
End Sub
```

- Die Prozedur beginnt mit den Schlüsselwort Sub und endet mit End Sub.
- Der Name der Ereignisprozedur setzt sich aus der Bezeichnung des auslösenden Objekts (Worksheet) und dem Ereignis (Change) zusammen. Die Bezeichnungen werden durch ein Unterstrich getrennt.

Zugriff auf die Prozedur ...

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Anweisung
End Sub
```

- wird im Kopf festgelegt.
- erfolgt privat (Private). Auf diese Prozedur kann nur das dazugehörige Objekt zugreifen. Die Prozedur kann nur durch eine bestimmte Arbeitsmappe gestartet werden.

Argumente der Prozedur

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Anweisung
End Sub
```

- Für das Ereignis Change wird der Zellbezug der veränderten Zelle (Target As Range) übergeben. Der Zellbezug wird als Wert übergeben (ByVal).
- Ereignisprozeduren haben Argumente in Abhängigkeit des ausgelösten Ereignisses.

Cells ...

Cells(1)				
		Cells(4,3)		

- enthält alle Zellen eines Arbeitsblattes.
- beschreibt mit Hilfe des Zeilen- und Spaltenindex eindeutig eine Zelle.

... in VBA

```
Dim arbeitsmappe As Workbook
Dim arbeitsblatt As Worksheet
Dim zelle As Range
Set arbeitsmappe = ThisWorkbook
Set arbeitsblatt = arbeitsmappe.ActiveSheet

' Eine bestimmte Zelle
Set zelle = arbeitsblatt.Cells(1)

' Die aktive Zelle
Set zelle = Application.ActiveCell
```

Range ...

Range("A1")				
	Range("B3:C4")			

- beschreibt eine Zelle, Zeile, Spalte oder Zellbereich.
- nimmt auf einen Zellbereich Bezug.

... in VBA

```
Dim arbeitsmappe As Workbook
Dim arbeitsblatt As Worksheet
Dim zellbereich As Range
Set arbeitsmappe = ThisWorkbook
Set arbeitsblatt = arbeitsmappe.ActiveSheet

Set zellbereich = arbeitsblatt.Range("A1")

Set zellbereich = arbeitsblatt.Range("B1:D6")
Set zellbereich = arbeitsblatt.Range(arbeitsblatt.Cells(1, 2), arbeitsblatt.Cells(6, 4))

Set zellbereich = arbeitsblatt.UsedRange
```

Erläuterung

- .Range("A1")bezieht sich auf die angegebene Zelle „A1“. Der Range-Bereich umfasst eine Zelle.
- .Range("A1:D6") bezieht sich auf den zusammenhängenden Zellbereich „A1:D6“. Der Range-Bereich umfasst alle Zellen („A1“, „A2“, ..., „B1“, „B2“, ...) des angegebenen Zellbereichs.
- .Range(blatt.Cells(1, 2), blatt.Cells(6, 4))). Die linke, obere Zelle sowie die untere, rechte Zelle des Bereichs werden durch Zellangaben definiert.
- .UsedRange bezieht sich auf den verwendeten Bereich in einer Arbeitsmappe.

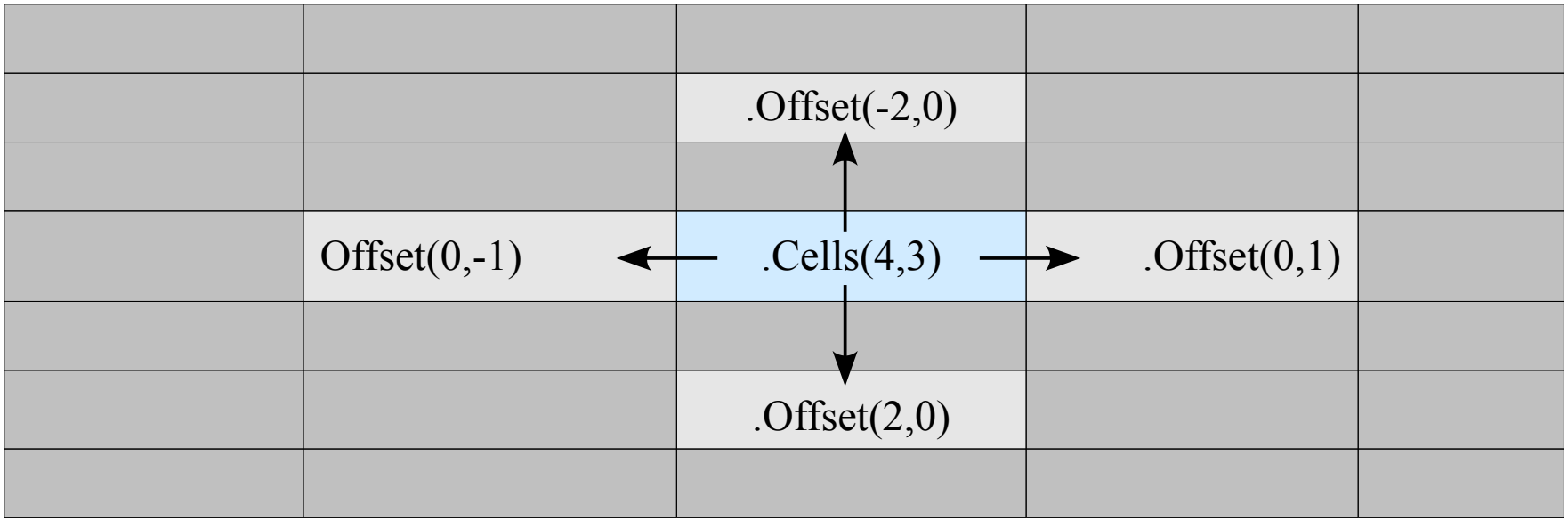
Zellbezug in Abhängigkeit der aktiven Zelle

```
Dim arbeitsmappe As Workbook
Dim arbeitsblatt As Worksheet
Dim zelle As Range
Dim zellbereich As Range

Set zelle = arbeitsblatt.Cells(4, 3)

Set zellbereich = zelle.Offset(0, 1)
```

Erläuterung

- Die Methode .Offset legt einen Zellbereich in Abhängigkeit einer anderen Zelle fest.
- Der Methode werden folgende Argumente übergeben:
  - Relative Verschiebung der Zeile.
  - Relative Verschiebung der Spalte.

Beispiel



Eine Verschiebung um ...

- Null. Die Zeilen- oder Spaltennummer wird übernommen.
- einen positiven Wert. Die Zeilennummer wird um den angegebenen Wert nach unten verschoben. Die Spaltennummer wird um den angegebenen Wert addiert und dementsprechend nach rechts verschoben.
- einen negativen Wert. Die Zeilennummer wird um den angegebenen Wert nach oben verschoben. Die Spaltennummer wird um den angegebenen Wert subtrahiert und dementsprechend nach links verschoben.

Eigenschaften des Objekts „Range“ ...

- beziehen sich immer auf die aktive Zelle im angegebenen Zellbereich. In einem Zellbereich sind mehrere Zellen markiert, aber nur eine Zelle ist aktiv.
- zellbereich.Address gibt den Zellbezug zurück.
- zellbereich.Column gibt die Spaltennummer zurück.
- zellbereich.Row gibt die Zeilennummer zurück.
- zellbereich.Value enthält den aktuellen Wert.

Methoden eines Ranges

- zellbereich.Select markiert den gewählten Zellbereich.
- zellbereich.Activate aktiviert eine Zelle.
- zellbereich.Clear löscht den Inhalt eines Zellbereichs.
- zellbereich.Find sucht nach bestimmten Inhalten in dem angegebenen Zellbereich.

Beispiel: Suche nach ...

```
Dim arbeitsblatt As Worksheet
Dim zelle As Range
Dim gefundenIn As Range

Set arbeitsblatt = ThisWorkbook.Worksheets("La")
Set zelle = arbeitsblatt.Range("A1:A10")

Set gefundenIn = zelle.Find(What:="Buch A", LookIn:=xlValues)
```

Methode .Find() ...

- sucht in einem bestimmten Zellbereich.
- entspricht dem Dialog „Suchen“.
- gibt die Zelladresse zurück, in der das gesuchte Wort gefunden wurde.
- Die Suche kann mit Hilfe von .FindNext() fortgesetzt werden.

Argumente der Methode in dem Beispiel

- What. Suchwort. Nach welchen Begriff wird gesucht?
- LookIn beschreibt den Typ der Information. In diesem Beispiel wird der Inhalt der Zellen in dem angegebenen Bereich mit dem Suchwort verglichen. Weitere Möglichkeiten sind xlFormulas oder xlComments.
- Die Methode hat noch viele weitere optionale Argumente, die in der Hilfe erläutert werden. Aus diesem Grund werden in diesem Beispiel die Parameter als benannte Argumente übergeben.

… anhand des Suchen-Dialogs

Benannte Argumente

LookIn:=xlValues

- Ein Argument wird an eine Methode übergeben. Die Übergabeparameter werden für einen korrekten Ablauf benötigt.
- Der Zuweisungsoperator := wird aus dem Doppelpunkt und dem Gleichheitszeichen zusammengesetzt. Der Operator weist einem Argument ein Wert zu.
- In diesem Beispiel wird dem Argument LookIn der Methode Range().Find die Art der zu suchenden Information als Wert übergeben. Der Wert wird mit Hilfe einer vordefinierten Konstanten beschrieben.

Bedingungen ...

- sind Ausdrücke, die wahr oder falsch sind.
- vergleichen Werte.
- nutzen Vergleichsoperatoren.
- überprüfen das boolsche Ergebnis von Funktionen, die mit „Is“ beginnen.
- können verknüpft werden.
- werden häufig in runde Klammern gesetzt. Die runden Klammern dienen der besseren Lesbarkeit.

Vergleichsoperatoren

Operator	Erläuterung	Beispiel
=	ist gleich	3 = 4 ≈ *falsch*
<>	ungleich	3 <> 4 ≈ *richtig*
<	kleiner als	3 < 4 ≈ *richtig*
<=	kleiner gleich	3 <= 4 ≈ *richtig*
>	größer	3 > 4 ≈ *falsch*
<=	größer gleich	3 >= 4 ≈ *falsch*

… werden in Anweisungen zur Auswahl genutzt

```
If Target.Column > 1 Then

   If Target.Column = 2 Then
      ' Anweisungen
   ElseIf Target.Column = 3 Then
      ' Anweisungen
   Else
      ' Anweisungen
   End If

End If
```

Erläuterung

```
Wenn Bedingung = Wahr Dann

  Wenn Bedingung = Wahr Dann
    ' Anweisungen
  Andernfalls Wenn Bedingung = Wahr Dann
    ' Anweisungen
  Andernfalls
    ' Anweisungen
  Ende

Ende
```

Ist die Objektvariable leer?

```
Dim zelle As Range

zelle = Nothing

If zelle Is Nothing Then
   ' Anweisungen
End If
```

Fallunterscheidung

```
If Target.Column > 1 Then

   Select Case Target.Column
         Case 2:
            ' Anweisungen
         Case 3:
            ' Anweisungen
         Case Else:
            ' Anweisungen
   End Select

End If
```

Erläuterung

```
If Target.Column > 1 Then

   Wähle Fälle aus [Variable/Objekt]
         Fall [Wert]:
            ' Anweisungen
         Fall [Wert]:
            ' Anweisungen
         Standardfall:
            ' Anweisungen
   Ende

End If
```

Ausgabe am Bildschirm

MsgBox("Das Produkt ist nicht im Lager vorhanden.")

result = MsgBox(prompt [, buttons] [, title] [, helpfile, context])

- Der Funktion können folgende Argumente übergeben werden:
  - Der anzuzeigende Text prompt. Der Text informiert den Benutzer über Fehler etc.
  - Welche Schaltflächen (buttons) werden im Meldungsfenster angezeigt?
  - Ein Text (title) für die Titelleiste des Fensters.
- Rückgabewert der Funktion:
  - Es wird ein Integer-Wert zurück gegeben. Mit Hilfe der Ganzzahl wird die gedrückte Schaltfläche codiert.