

Bedingte Formatierung

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 /365

Erstellungsdatum: 15.10.2011

Überarbeitung: 01.12.2023

Beschreibung:

Zellen und Zellbereiche lassen sich durch Formate gestalten. Dazu liefert Excel einige vorgegebene Formate. Aber Formate lassen sich auch mit Bedingungen koppeln, so dass sie immer nur bei erfüllter Bedingung zur Anwendung kommen. Wie sich solche bedingten Formate per VBA erstellen lassen, ist Gegenstand dieses Kapitels.

Anwendungs-Datei: AE-009_BedingteFormatierung.xlsm

1 Bereiche vorbereiten

Eine bedingte Formatierung wird immer auf den Bereich angewendet, der zuvor markiert ist. Das ändert sich auch unter VBA nicht. Erstellen wir daher eine einfache Zahlenliste wie nachfolgend dargestellt (Bild 1).

	A
1	102
2	45
3	32
4	76
5	45
6	66
7	92
8	84
9	12
10	9
11	39
12	43
13	59

Bild 1. Datenliste

Für den Bereich A1:A13 erstellen wir in einem Modul ein Range-Objekt und anschließend selektieren wir dieses Objekt.

Codeliste 1. Die Prozedur SetFormatConditions markiert den Datenbereich

```
Sub SetFormatConditions()  
    Dim Bereich1 As Range  
  
    Set Bereich1 = Range("A1:A13")  
    Bereich1.Select  
End Sub
```

Jeder Bereich besitzt die Unterobjektliste *FormatConditions*. In ihr lassen sich also bedingte Formate instanziiieren und auch wieder entfernen. Das Löschen aller vorhandenen bedingten Formate kann über die Methode Delete erfolgen.

```
Selection.FormatConditions.Delete
```

Die Anweisung sollte immer am Anfang einer Prozedur stehen, damit eventuell noch vorhandene Formate gelöscht werden.

2 Die Methode Add

Immer wenn wir es mit Objektlisten zu tun haben und ein weiteres Objekt instanziiieren wollen, kommt die Methode *Add* ins Spiel. Sie hat in diesem Fall die allgemeine Syntax

```
Add(Type As XlFormatConditionType, _  
    [Operator], _  
    [Formula1], _  
    [Formula2]) As FormatCondition
```

Es gibt zwei Formen von *FormatConditionType*'s. Je nachdem, ob man die Bedingung vom Inhalt der Zellen abhängig macht, oder über eine Funktionskonstruktion bestimmt.

```
xlCellValue (Bedingung für Zelleninhalte)
xlExpression (Bedingung laut Formel)
```

Um den Zelleninhalt mit einem Wert vergleichen zu können, benötigt man einen Operator. Es ist einer der nachfolgend aufgeführten.

```
xlBetween
xlEqual
xlGreater
xlGreaterEqual
xlLess
xlLessEqual
xlNotBetween
xlNotEqual
```

Damit ist auch schon das erste sinnvolle Beispiel möglich. Darin werden die Werte im Bereich1 gesucht, die gleich oder größer sind als 50, und mit lichtem Gelb (ColorIndex=36) als Hintergrund gekennzeichnet.

Codeliste 2. Die Prozedur Beispiel1 markiert den Datenbereich nach Regeln

```
Sub Beispiel1()
    Dim Bereich1 As Range

    Set Bereich1 = Range("A1:A13")
    Bereich1.Select
    With Selection
        .FormatConditions.Add _
            Type:=xlCellValue, _
            Operator:=xlGreaterEqual, _
            Formula1:="50"
        .FormatConditions(1).Interior.ColorIndex = 36
    End With
End Sub
```

Da wir hier nur einen Wert (nämlich 50) haben, wird auch nur der Parameter Formula1 benötigt. Anders sieht das im nächsten Beispiel aus. Da werden alle Werte zwischen 50 und 90 markiert.

Codeliste 3. Die Prozedur Beispiel2 markiert den Datenbereich nach Regeln

```
Sub Beispiel2()
    Dim Bereich1 As Range

    Set Bereich1 = Range("A1:A13")
    Bereich1.Select
    With Selection
        .FormatConditions.Add _
            Type:=xlCellValue, _
            Operator:=xlBetween, _
            Formula1:="50", _
            Formula1:="90"
        .FormatConditions(1).Interior.ColorIndex = 36
    End With
End Sub
```

Das Beispiel1 lässt sich auch über den Formeltyp definieren.

Codeliste 4. Die Prozedur *SetFormatConditions* setzt Format-Konditionen

```
Sub SetFormatConditions()  
    Dim Bereich1 As Range  
  
    Set Bereich1 = Range("A1:A13")  
    Bereich1.Select  
    With Selection  
        .FormatConditions.Add _  
            Type:=xlExpression, _  
            Formula1:="=A1>=50"  
        .FormatConditions(1).Interior.ColorIndex = 36  
    End With  
End Sub
```

Diesmal nimmt der Parameter *Formula1* die als Bedingung definierte Formel auf.

3 Die Liste bedingter Formate

In der Version 2003 ließen sich nur bis zu drei bedingten Formaten für einen Zellbereich festlegen. Ab der Version 2007 können es auch mehr sein. Bereits in den Beispielen haben wir den Listenindex benutzt, um die Hintergrundfarbe einzustellen.

```
.FormatConditions(1)
```

Mit jedem Add wird also ein weiteres Listenelement instanziiert. Das nachfolgende Beispiel erzeugt drei bedingte Formatierungen für den Bereich. Die erste Bedingung sucht wieder alle Zahlen, die größer oder gleich 50 sind. Die zweite Bedingung sucht alle Zahlen, die kleiner oder gleich 20 sind. Die dritte Bedingung markiert die Zahl 32.

Die erste Bedingung formatiert den Hintergrund wieder in lichtem Gelb. Die zweite Bedingung erstellt einen blauen Rahmen um die Zelle, mit unterschiedlichen Linientypen. Die dritte Bedingung verändert Farbe und Typ der Schrift.

Codeliste 5. Die Prozedur *SetFormatConditions2* setzt Format-Konditionen

```
Sub SetFormatConditions2()  
    Dim Bereich1 As Range  
  
    Set Bereich1 = Range("A1:A13")  
    Bereich1.Select  
    With Selection  
        '1. Bedingung  
        .FormatConditions.Add _  
            Type:=xlExpression, _  
            Formula1:="=A1>=50"  
        '2. Bedingung  
        .FormatConditions.Add _  
            Type:=xlExpression, _  
            Formula1:="=A1<20"  
        '3. Bedingung  
        .FormatConditions.Add _  
            Type:=xlExpression, _  
            Formula1:="=A1=32"  
        'Zellformate setzen  
        'Hintergrundfarbe  
        .FormatConditions(1).Interior.ColorIndex = 36  
        'Zellrahmen zur bedingten Formatierung 2
```

```

With .FormatConditions(2).Borders(xlLeft) 'Linie links
.LineStyle = xlContinuous
.Weight = xlHairline
.ColorIndex = 5
End With
With .FormatConditions(2).Borders(xlRight) 'Linie rechts
.LineStyle = xlContinuous
.Weight = xlHairline
.ColorIndex = 5
End With
With .FormatConditions(2).Borders(xlTop) 'Linie oben
.LineStyle = xlContinuous
.Weight = xlHairline
.ColorIndex = 5
End With
With .FormatConditions(2).Borders(xlBottom) 'Linie unten
.LineStyle = xlContinuous
.Weight = xlThin
.ColorIndex = 5
End With
'Schrift
With .FormatConditions(3).Font
.Bold = True
.Italic = True
.ColorIndex = 3
End With
End With
End Sub

```

	A	B	C	D
1		Einzelkosten	%-Anteil	
2	Entwicklung	=MITTELWERT('Kalkulation 1:Kalkulation 3'!B2)		
3	Material	6.433,33 €	16,6%	
4	Fertigung	7.166,67 €	18,5%	
5	Verwaltung	3.400,00 €	8,8%	
6	Marketing	5.833,33 €	15,1%	
7	Vertrieb	8.200,00 €	21,2%	

Bild 3. Mittelwertfunktion mit 3D-Bezug

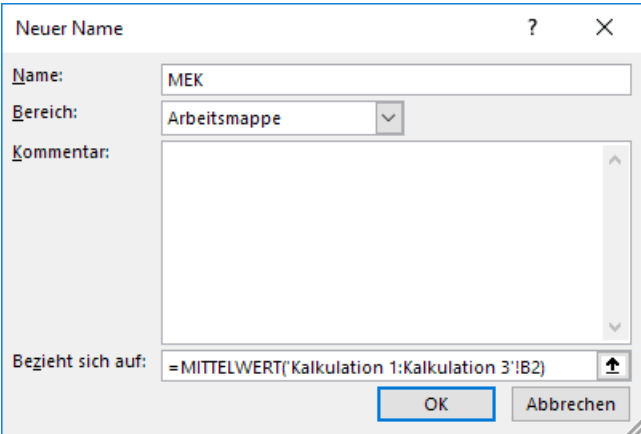
- Die so bestimmte Formel wird anschließend durch Ziehen auf die Zellen B3:B7 übertragen (Bild 4).

	A	B	C
1		Einzelkosten	%-Anteil
2	Entwicklung	7.700,00 €	19,9%
3	Material	6.433,33 €	16,6%
4	Fertigung	7.166,67 €	18,5%
5	Verwaltung	3.400,00 €	8,8%
6	Marketing	5.833,33 €	15,1%
7	Vertrieb	8.200,00 €	21,2%
8			
9	Gesamtkosten	38.733,33 €	

Bild 4. Übertragung der Formel

2 Erstellung eines Bereichsnamens für einen 3D-Bezug

Der zuvor bestimmten Funktion können wir auch einen Bereichsnamen zuordnen. Dazu wird im Menüband unter *Formeln* in der Gruppe *Definierte Namen* die Methode *Namen definieren* aufgerufen und mit der Schaltfläche *Neu* ein Fenster *Neuer Name* zur Eingabe geöffnet (Bild 5).



The screenshot shows the 'Neuer Name' dialog box with the following fields:

- Name: MEK
- Bereich: Arbeitsmappe
- Bezieht sich auf: =MITTELWERT('Kalkulation 1:Kalkulation 3'!B2)

Bild 5. Bereichsnamen erstellen

Als Namen verwenden wir in diesem Fall *MEK* zur Abkürzung von *Mittelwert Einzelkosten*. Im Feld *Bezieht sich auf* wird die Formel für die erste Zelle des Bereichs der Einzelkosten eingetragen. Die Fenster zur Namensbearbeitung werden geschlossen und die Zelle B2 erhält die Formel =MEK (Bild 6).

	A	B	C
1		Einzelkosten	%-Anteil
2	Entwicklung	=MEK	19,9%
3	Material	6.433,33 €	16,6%
4	Fertigung	7.166,67 €	18,5%
5	Verwaltung	3.400,00 €	8,8%
6	Marketing	5.833,33 €	15,1%
7	Vertrieb	8.200,00 €	21,2%

Bild 6. Formel mit Bereichsnamen

Anschließend wird diese auf die restlichen Einzelkosten übertragen (Bild 7).

	A	B	C
1		Einzelkosten	%-Anteil
2	Entwicklung	7.700,00 €	19,9%
3	Material	6.433,33 €	16,6%
4	Fertigung	7.166,67 €	18,5%
5	Verwaltung	3.400,00 €	8,8%
6	Marketing	5.833,33 €	15,1%
7	Vertrieb	8.200,00 €	21,2%
8			
9	Gesamtkosten	38.733,33 €	

Bild 7. Übertragung der Formel

3 Ein 3D-Bezug in VBA

Die Mittelwertfunktion hat in VBA den Namen *AVERAGE* und die Übertragung der Formel wird mit der *AutoFill*-Methode ausgeführt.

Codeliste 1. Die Prozedur DreiDBezug in Tabelle4

```
Sub DreiDBezug ()
    Range("B2").Formula =
        "=AVERAGE('Kalkulation 1:Kalkulation 3'!B2)"
    Range("B2").Select
    Selection.AutoFill Destination:=Range("B2:B7"),
        Type:=xlFillDefault
End Sub
```

4 Bereichsnamens für einen 3D-Bezug in VBA

Bereichsnamen sind Unterobjekte eines Workbooks und werden mit der *Add-Methode* erstellt.

Codeliste 2. Die Prozedur DreiDBezugsName in Tabelle4

```
Sub DreiDBezugsName ()
    ActiveWorkbook.Names.Add Name:="MEK", RefersToR1C1:=
        "=AVERAGE('Kalkulation 1:Kalkulation 3'!RC)"
    Range("B2").Select
    ActiveCell.FormulaR1C1 = "=MEK"
    Range("B2").Select
    Selection.AutoFill Destination:=Range("B2:B7"),
        Type:=xlFillDefault
End Sub
```