

Workshop 9: VBA-Programmierung mit MS Excel

1 Dialogarten in Excel	2
2 Eingebaute und eigene Dialogfelder	2
2.1 Eingebaute Dialogfelder	2
2.1.1 Überblick über die integrierten Dialogfelder	2
2.1.2 Dialogeigenschaften vorgeben und auswerten	4
2.2 Benutzerdefinierte Dialogfelder	5
2.2.1 Benutzerformular hinzufügen	5
2.2.2 Eigenschaften eines Benutzerformulars	6
2.2.3 Methoden eines Benutzerformulars.....	6
2.2.4 Ereignisse eines Benutzerformulars	6
2.3 Steuerelemente in Benutzerformular einfügen.....	6
2.3.1 Werkzeugsammlung.....	6
2.3.2 Steuerelemente platzieren	8
2.3.3 Steuerelements ausrichten und gruppieren.....	8
2.3.4 Eigenschaften von Steuerelementen festlegen	9
2.3.5 Reihenfolge der Steuerelemente festlegen	9
2.3.6 Ereignisprozeduren programmieren.....	9
3 Beispiel für die Erstellung eines Benutzerformulars.....	10
3.1 Steuerelemente.....	10
3.2 Ereignisprozeduren	11
3.2.1 Ereignisprozedur für die Arbeitsmappe.....	11
3.2.2 Ereignisprozeduren für das Formular	11
3.2.3 Ereignisprozeduren für die Steuerelemente des Formulars	12
3.3 Standardprozeduren	13

Workshop 9: VBA-Programmierung mit MS Excel

1 Dialogarten in Excel

Aus Anwendersicht gesehen lassen sich drei Arten von Dialogfenstern (kurz Dialoge) in Excel unterscheiden:

- Vordefinierte VBA-Dialoge: `MsgBox`, `InputBox`.
- Integrierte Excel-Dialoge: Alle in Excel integrierten Dialoge können in VBA-Programme eingebaut werden.
- Benutzerdefinierte Dialoge: Mit der integrierten Entwicklungsumgebung, also dem VBA-Editor, lassen sich für besondere Aufgaben maßgeschneiderte Dialoge erstellen.

Während des Entwurfs wird meist nicht von Dialogen, sondern von Formularen (in VBA `UserForm` genannt) gesprochen.

2 Eingebaute und eigene Dialogfelder




In Excel können die eingebauten Dialogfelder aufgerufen werden. Es besteht aber auch die attraktive Möglichkeit, eigene Dialogfelder zu programmieren. In diesem Workshop erfahren Sie wie

- eingebaute Dialogfelder aktiviert werden,
- eigene Dialogfelder programmiert werden und wie
- Steuerelemente eingesetzt werden.

2.1 Eingebaute Dialogfelder

2.1.1 Überblick über die integrierten Dialogfelder

Die Auflistung `Dialogs` enthält alle integrierten Dialoge von Excel. Die eingebaute Konstante `xlDialogPrint` repräsentiert beispielsweise den Druckendialog. Um einen Überblick über alle integrierten Dialoge zu erhalten, sind folgende Schritte erforderlich:

Schritt	Beschreibung	Symbole / Anzeige
1	Rufen Sie in Excel mit der Tastenkombination <code>Alt + F11</code> den Projekt-Explorer auf.	
2	Klicken Sie auf die nebenstehend abgebildete Schaltfläche oder drücken Sie die Funktionstaste <code>F2</code> , um den Objektkatalog von Excel zu öffnen.	 Objektkatalog
3	Geben Sie im obersten Kombinationsfeld <code>Excel</code> und im darunter stehenden Kombinationsfeld den Suchbegriff <code>xlBuiltInDialog</code> ein.	
4	Klicken Sie auf die Suchen-Schaltfläche	 Suchen

Workshop 9: VBA-Programmierung mit MS Excel

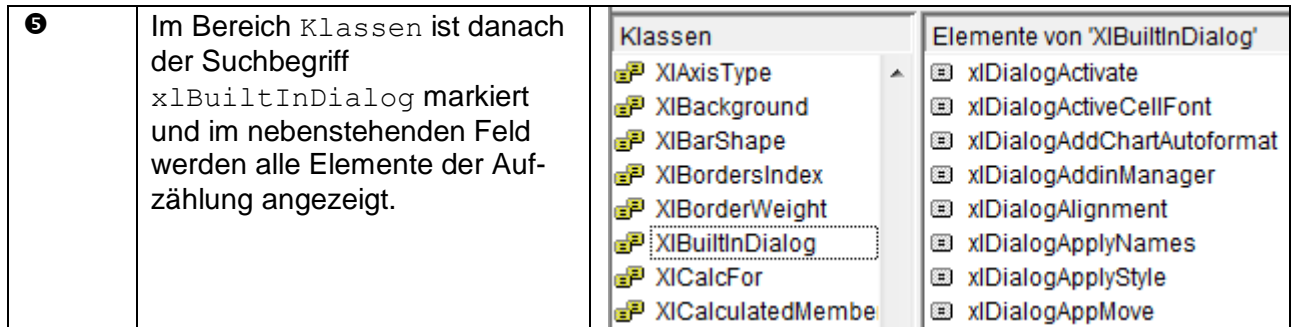


Tabelle 1: Schritte zur Anzeige der integrierten Dialogfenster im Objektkatalog von Excel

- Insgesamt stehen in Excel über 250 integrierte Dialogfelder zur Verfügung. Die genaue Zahl ist von der eingesetzten Excel-Version abhängig.
- Excel unterstützt nicht alle jeweils dazugehörigen Dialogfeldargumente, wie zum Beispiel die Konstante `xlDialogSaveCopyAs`. Das bedeutet, dass diese Argumente nicht verwendet werden können.
- Bei einigen dieser Konstanten wird kein Dialogfenster angezeigt, sondern die Aktion wird direkt ausgeführt, wie z.B. beim Autofilter ein/aus `xlDialogFilter`. Es wird also keine Dialogbox angezeigt.
- Viele der Dialogfelder beziehen sich auf ein bestimmtes Objekt. Dieses Objekt muss in der Arbeitsmappe vorhanden und teilweise selektiert sein, weil VBA sonst eine Fehlermeldung anzeigt.
- Die Methode `Show` wird verwendet, um die vordefinierten Dialogfenster anzuzeigen.
- Die `Dialogs`-Auflistung steht nicht als globales Objekt zur Verfügung, so dass immer `Application.Dialogs (xlDialog ...).Show` genutzt werden muss.
- Integrierte Dialogfelder können über verschiedene Schaltflächen geschlossen werden. Die jeweils verwendete Schaltfläche lässt sich durch Auswertung des Rückgabewertes der Methode `Show` bestimmen.

Eine Liste der in Excel integrierten Dialogfelder befindet sich in der Online-Hilfe unter dem Stichwort *Listen der integrierten Dialogfeldargumente*. Einen Ausschnitt davon zeigt Abb. 1.

Zahl und Art der Argumente für die Methode `Show` des jeweiligen Dialogfelds kann ebenfalls aus der gerade genannten Informationsquelle entnommen werden.

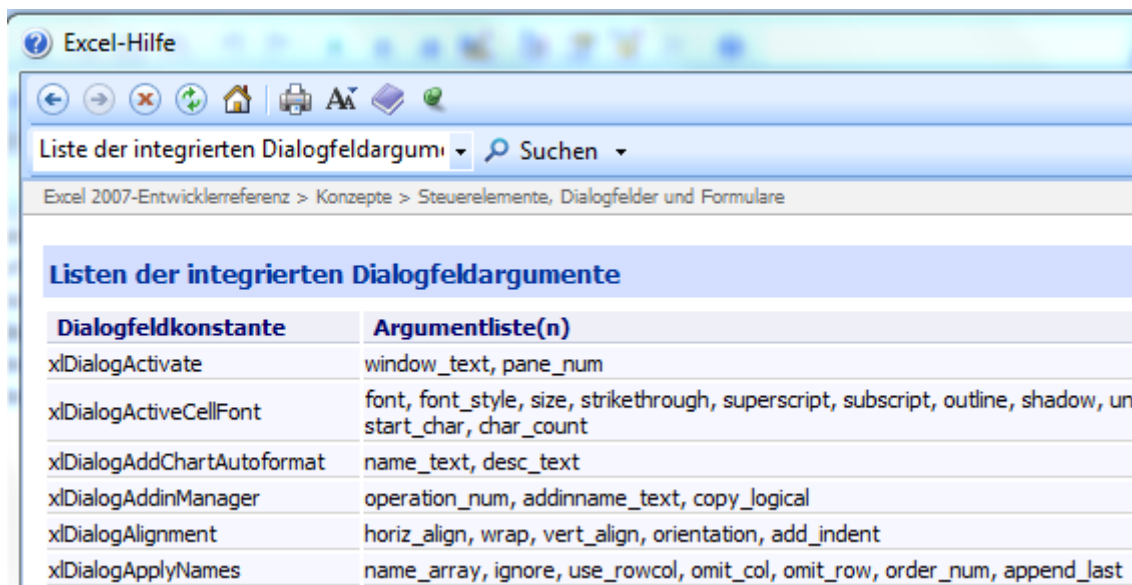


Abbildung 1: Auszug aus Liste der integrierten Dialogfeldargumente in der Excel-Online-Hilfe

Workshop 9: VBA-Programmierung mit MS Excel

An die Methode `Show` können bis zu 30 Parametern (`Arg1`, ..., `Arg30`) übergeben werden. Diese sind in der Online-Hilfe zu den integrierten Dialogfenstern dokumentiert (siehe Abb. 1).

1.1.2 Dialogeigenschaften vorgeben und auswerten

Im nachstehenden Code-Beispiel wird das Dialogfeld *Speichern unter* (`xlDialogSaveAs`) eingesetzt, um eine Excel-Datei unter einem Dateinamen zu speichern, der durch den Inhalt der Zelle A1 bestimmt ist. Die benutzerdefinierte Funktion `IstGespeichert` dient zur Auswertung des Rückgabewertes der Methode `Show`.

```
Sub SpeichernUnterDialog()  
    With Sheets(1)  
        .[A1] = "MeinDateiname"  
        ' benutzerdefinierte Funktion aufrufen  
        If IstGespeichert(.[A1]) Then  
            MsgBox Prompt:="Die Speichern-Schaltfläche wurde gedrückt!", _  
                Buttons:=vbExclamation, Title:="Speichern unter"  
        Else  
            MsgBox Prompt:="Die Abbrechen-Schaltfläche wurde gedrückt!", _  
                Buttons:=vbCritical, Title:="Speichern unter"  
        End If  
    End With  
End Sub  
  
Function IstGespeichert(strDocName As String) As Boolean  
    ' Rückgabewert der Methode Show  
    IstGespeichert = Application.Dialogs(xlDialogSaveAs).Show(Arg1:=strDocName)  
End Function
```

In der folgenden Prozedur wird das Dialogfeld *Schriftarten* (`xlDialogFont`) zur Formatierung von Zeichen angezeigt. Wird es über die Schaltfläche OK geschlossen, werden bestimmte Eigenschaften ausgewertet:

```
Sub DialogSchriftarten()  
    ' Argumente: Schriftart (hier: Arial) und Schriftgrad(hier 12)  
    If Application.Dialogs(xlDialogFont).Show(Arg1:"Arial", Arg2:"12") Then  
        ' Die folgenden Anweisungen werden nur dann ausgeführt,  
        ' wenn das Dialogfeld mit OK geschlossen wurde.  
        With Sheets(1)  
            .Cells(2, 2).Value = Selection.Font.Name  
            .Cells(3, 2).Value = Selection.Font.Size  
            With .Range(.Cells(2, 2), .Cells(3, 2))  
                If .Font.Name = "Arial" And .Font.Size = 12 Then  
                    MsgBox Prompt:"Schriftart und Schriftgrad würden gesetzt", _  
                        Buttons:=vbInformation, Title:"Dialog Schriftarten"  
                End If  
            End With  
        End With  
    End With  
End Sub
```

Welche Argumente auf welche Weise an die Methode `Show` übergeben werden, kann der Excel-Online-Hilfe zu integrierten Dialogfeldern entnommen werden.


Workshop 9: VBA-Programmierung mit MS Excel

2.2 Benutzerdefinierte Dialogfelder

Wird ein spezielles Dialogfenster benötigt, kann ein Benutzerformular (sogen. `UserForm`) erstellt werden. Je nach Bedarf wird dieses Formular mit Steuerelementen bestückt, die aus einer vorgegebenen *Werkzeugsammlung* ausgewählt werden. Durch das Programmieren von *Ereignisprozeduren* werden dem Benutzerformular die gewünschten Funktionalitäten hinzugefügt.

2.2.1 Benutzerformular hinzufügen

Die Entwicklungsumgebung von Excel wird durch die Tastenkombination `Alt + F11` aufgerufen. Zum Anlegen einer `UserForm` sind folgende Schritte notwendig:

Schritt	Beschreibung	Symbol
❶	Markieren Sie im Projekt-Explorer ein Projekt, dem ein benutzerdefiniertes Formular hinzugefügt werden soll.	
❷	Rufen Sie den Menüpunkt <code>EINFÜGEN – USERFORM</code> auf oder verwenden Sie die Schaltfläche <code>UserForm einfügen</code> .	

Ihnen wird ein leeres Benutzerformular (`UserForm`) angezeigt (siehe Abb. 2), das Sie nun mit den benötigten Steuerelementen bestücken können. Die Steuerelemente befinden sich in der sogenannten *Werkzeugsammlung*.

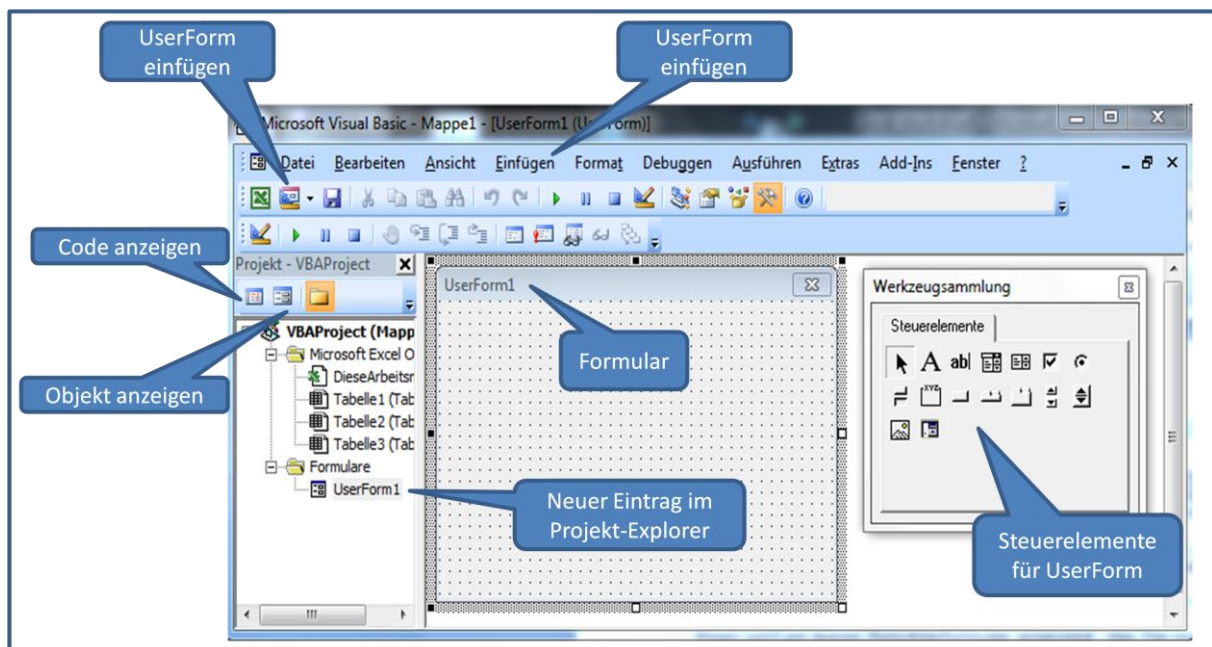


Abbildung 2: Benutzerformular (`UserForm`) einfügen

Die Darstellung des Formulars ist standardmäßig gerastert, um die Positionierung der Steuerelemente zu vereinfachen. Die Größe des Formulars bestimmen Sie entweder über das zugehörige Eigenschaftsfenster oder indem Sie den Rand des Formulars mit der Maus verschieben.

Um zum zugehörigen Codefenster zu gelangen, klicken Sie auf die Schaltfläche `CODE ANZEIGEN` oder betätigen Sie die Funktionstaste `F7`. Zurück zur Entwurfsansicht des Formulars gelangen Sie mit der Schaltfläche `OBJEKT ANZEIGEN` oder durch die Tastenkombination `↑ + F7`.

Workshop 9: VBA-Programmierung mit MS Excel

2.2.2 Eigenschaften eines Benutzerformulars

Mit der Schaltfläche EIGENSCHAFTEN oder der Funktionstaste F4 wird das Eigenschaftsfenster aktiviert. Hier können Sie den Namen des Formulars, die Größe, die Position und einiges mehr ändern (siehe Tab. 2).

Eigenschaft	Kurzbeschreibung
(Name)	Name des Formulars. Damit wird es bei der Programmierung angesprochen.
BackColor	Hintergrundfarbe des Formulars
Caption	Bezeichnung des Formulars in der Titelzeile
Height, Width	Höhe und Breite des Formulars
StartUpPosition	Position des Formulars bei der Anzeige

Tabelle 2: Wichtige Eigenschaften eines Benutzerformulars

2.2.3 Methoden eines Benutzerformulars

Das Anzeigen eines Formulars erfolgt mit der Methode `Show`. Um es zu schließen, wird entweder die Methode `Hide` (*Ausblenden*) oder `Unload` (*Entfernen*) angewandt. Beim *Entfernen* werden die Werte aller Variablen gelöscht, die in den Prozeduren des Formulars verwendet werden. Dagegen bleiben beim *Ausblenden* des Formulars alle Einstellungen im Speicher erhalten.

2.2.4 Ereignisse eines Benutzerformulars

Jedes Formular besitzt Ereignisse, für die Sie mit entsprechende Ereignisprozeduren codieren müssen. Die wichtigsten Ereignisse eines Benutzerformulars sind in Tab. 3 zusammengestellt:

Ereignis	Kurzbeschreibung
<code>Initialize</code>	Tritt ein, wenn das Formular geladen wird, also vor dem ersten Anzeigen.
<code>Click</code>	Tritt ein, wenn ein Klick mit der linken Maustaste auf das Formular erfolgt.
<code>KeyPress</code>	Tritt ein, wenn eine Taste betätigt wird, die ein darstellbares Zeichen darstellt.
<code>KeyDown / KeyUp</code>	Tritt ein, wenn eine beliebige Taste gedrückt oder losgelassen wird. Dieses Ereignis tritt auch bei Tastenkombinationen ein.
<code>MouseDown, MouseUp, MouseMove</code>	Tritt ein, wenn die Maus betätigt (<code>MouseDown</code>), losgelassen (<code>MouseUp</code>) oder über das Formlar bewegt (<code>MouseMove</code>) wird.
<code>QueryClose</code>	Tritt ein, wenn das Formular geschlossen werden soll. Es kann jedoch verhindert werden, z. B. wenn die Eingabedaten nicht korrekt sind.
<code>Terminate</code>	Tritt ein, wenn das Formular geschlossen wird. Mit Hilfe der Anweisung <code>Unload</code> oder durch das Klicken auf das Kreuz in der Titelleiste wird dieses Ereignis ausgelöst.

Tabelle 3: Wichtige Ereignisse eines Benutzerformulars

2.3 Steuerelemente in Benutzerformular einfügen

2.3.1 Werkzeugsammlung

Nach dem Erzeugen eines Formulars und der Festlegung seiner Eigenschaften können dem Formular die benötigten Steuerelemente hinzugefügt werden. Diese befinden sich in der so-

Workshop 9: VBA-Programmierung mit MS Excel

gen. *Werkzeugsammlung* (siehe Abb. 2). Mit dem Menüpunkt ANSICHT – WERZEUGSAMMLUNG oder dem Symbol



kann auf die Werkzeugsammlung zugegriffen werden. In Tab. 4 sind die standardmäßig verfügbaren Steuerelemente aufgeführt.

Symbol	Bezeichnung	Kurzbeschreibung
	Kombinationsfeld	Stellt eine Mischung aus Eingabefeld und Listenfeld dar.
	Kontrollkästchen	Kontrollkästchen erlauben die Auswahl zweier Zustände wie True/False, ja/nein, an/aus.
	Befehlsschaltfläche	Dienen zum Ausführen von Programmen.
	Rahmen	Dienen zur Gruppierung von Optionsfeldern. Sonst sind sie ein rein gestalterisches Element.
	Anzeige	Damit lassen sich Bilder auf einem Dialogfeld platzieren.
	Bezeichnungsfeld	Dient zum Anzeigen von Texten. Der Text ist vom Benutzer nicht änderbar.
	Listenfeld	Es erlaubt die Auswahl eines oder mehrerer Einträge. Es benötigt mehr Platz als ein funktionsgleiches Kombinationsfeld.
	Multiseiten	Damit lassen sich Informationen auf verschiedenen Registerblättern hintereinander darstellen.
	Optionsfeld	Es ermöglicht die Auswahl zweier Zustände. Es kann in einen Rahmen eingefügt werden und bildet dann eine Optionsgruppe. Daraus kann nur eine einzige Option gewählt werden.
	Bildlaufleiste	Horizontale oder vertikale Rollbalken, um innerhalb einer Liste nach oben oder unten scrollen zu können.
	Drehfeld	Enthält Pfeile zum Erhöhen oder Verringern von angezeigten Werten.
	Register	Registergruppe mit standardmäßig zwei Registern. Es ist veraltet und wird eigentlich nicht mehr verwendet.
	Umschaltfeld	Ermöglicht die Einrichtung beschrifteter Schalter. Damit wird auch ein Zustand ein- oder ausgeschaltet.
	Textfeld	Dient zur Anzeige oder Erfassung beliebiger Texte.
	Eingabefeld f. Bereiche	Ermöglicht die Referenzierung eines Zellenbereichs auf einem Tabellenblatt.

Tabelle 4: Kurzbeschreibung der Steuerelemente der Werkzeugsammlung

Das Element (*Objekte auswählen*) in der Werkzeugsammlung dient zum Markieren und Bearbeiten von Steuerelementen. Damit wird *kein* Steuerelement erstellt.

Präfix	Steuerelement	Bezeichnung	Symbol
cbo	ComboBox	Kombinationsfeld	
chk	CheckBox	Kontrollkästchen	
cmd	CommandButton	Befehlsschaltfläche	
fra	Frame	Rahmen	
img	Image	(Bild-)Anzeige	
lbl	Label	Bezeichnungsfeld	
lst	ListBox	Listenfeld	
mpg	MultiPage	Multiseiten (Tabulatorfeld)	

Workshop 9: VBA-Programmierung mit MS Excel

opt	OptionButton	Optionsfeld	
scr	ScrollBar	Bildlaufleiste (Rollbalken)	
spn	SpinButton	Drehfeld	
tab	TabStrip	Register (Tabulatorstreifen) <i>Achtung: veraltet!</i>	
tgl	ToggleButton	Umschaltfeld	
txt	TextBox	Textfeld	
ref	RefEdit	Eingabefeld f. Bereiche	

Tabelle 5: Namenskonventionen für Steuerelemente in der Werkzeugsammlung

Präfix	Alternativer Präfix	Bezeichnung	Standardereignis	Standardeigenschaft
mpg	mup	MultiPage	Change	Value
scr	vsb / hsb	ScrollBar	Change	Value
spn	spb	SpinButton	Change	Value
tab	tbs	TabStrip	Change	Value
txt		TextBox	Change	Value
ref		RefEdit	Change	Value
fra	frm	Frame	Click	Caption
lbl	lab	Label	Click	Caption
img		Image	Click	Picture
chk		CheckBox	Click	Value
cmd	btn	CommandButton	Click	Value
lst		ListBox	Click	Value
opt		OptionButton	Click	Value
tgl		ToggleButton	Click	Value
cbo	cbx	ComboBox	Click	Value (Text)

Tabelle. 6: Standardereignis und Standardeigenschaft von Steuerelementen

2.3.2 Steuerelemente platzieren

Um ein Steuerelement auf dem Benutzerformular abzulegen, markieren Sie es in der Werkzeugsammlung und ziehen Sie im Formular bei gedrückter Maustaste ein Rechteck auf, das so groß ist, wie das Steuerelement angezeigt werden soll.

Wollen Sie nachträglich die Größe eines Steuerelements ändern, ziehen Sie das markierte Element mittels des Markierungsrahmens auf die gewünschte Größe. Um die Position eines Elements zu verändern, klicken Sie in das Element und ziehen Sie es bei gedrückter Maustaste an die neue Position.

Alternativ können Sie Größe und Position eines Steuerelements exakt im Eigenschaftsfenster des jeweiligen Steuerelements einstellen.

2.3.3 Steuerelements ausrichten und gruppieren

Wollen Sie mehrere Steuerelements ausrichten oder gruppieren nutzen Sie dazu die Funktionen der Symbolleiste UserForm. Über den Menüpunkt ANSICHT SYMBOLEISTEN USERFOM wird sie aufgerufen:

Workshop 9: VBA-Programmierung mit MS Excel

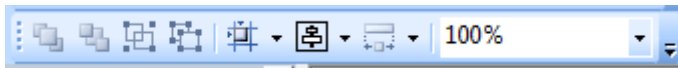


Abbildung 3: Die Symbolleiste UserForm

Weitere Möglichkeiten zum Ausrichten von Steuerelementen und weitere Funktionen zur Anpassung der Größe von Steuerelementen bietet das Menü FORMAT.

2.3.4 Eigenschaften von Steuerelementen festlegen

Für Steuerelemente lassen sich sehr viele Eigenschaften einstellen. Markieren Sie das betreffende Steuerelement und blenden Sie das zugehörige Eigenschaftsfenster ein, z. B. mit der Funktionstaste F4. Verändern Sie die Eigenschaften nach ihren Vorstellungen. Die Änderungen werden sofort sichtbar.

Im Folgenden werden einige Eigenschaften beschrieben, die für alle Steuerelemente gelten:

Eigenschaft	Bedeutung
BackColor	bestimmt die Farbe des Hintergrunds eines Steuerelements
ControltipText	definiert den optionalen Hilfetext zu einem Steuerelement
Enabled	legt fest, ob das Steuerelement ausgewählt werden kann oder nicht
Height	gibt die Höhe eines Steuerelements an
Left	legt die Position des linken Rands eines Steuerelements fest
Name	legt den Namen fest, der im Quellcode verwendet wird, um auf das Steuerelement zuzugreifen
TabIndex	definiert die Reihenfolge der Auswahl mittels der Tab-Taste.
Tag	ist vorgesehen für optionale Zusatzinformationen
Top	legt die Position des oberen Rands eines Steuerelements fest
Visible	legt fest, ob das Steuerelement sichtbar oder unsichtbar ist
Width	definiert die Breite eines Steuerelements

Tabelle 7: Allgemeingültige Eigenschaften von Steuerelementen

2.3.5 Reihenfolge der Steuerelemente festlegen

Die Reihenfolge, in der die Steuerelemente im Formular nacheinander angesprochen werden, wird durch die Eigenschaft `TabIndex` festgelegt.

Die Aktivierungsreihenfolge der Steuerelemente kann im Dialogfenster AKTIVIERUNGSREIHENFOLGE festgelegt werden.

Die gewünschte Reihenfolge der Steuerelemente kann auch im Eigenschaftsfenster über die bereits erwähnte Eigenschaft `TabIndex` bestimmt werden. Ausgehend vom ersten Element wird diese mit 0 beginnend benummert.

2.3.6 Ereignisprozeduren programmieren

Erst die Programmierung von Ereignisprozeduren erzeugt die gewünschte Funktionalität eines benutzerdefiniertes Dialogfelds. Ereignisprozeduren werden für ein bestimmtes Steuerelement und ein dazugehöriges Ereignis benötigt.

- Markieren Sie das betreffende Steuerelement
- Öffnen Sie das Code-Fenster
- Wählen Sie im linken Listenfeld ❶ einen Eintrag aus.
- Wählen Sie anschließend im rechten Listenfeld das Ereignis ❷ aus, auf das reagiert werden soll

Workshop 9: VBA-Programmierung mit MS Excel

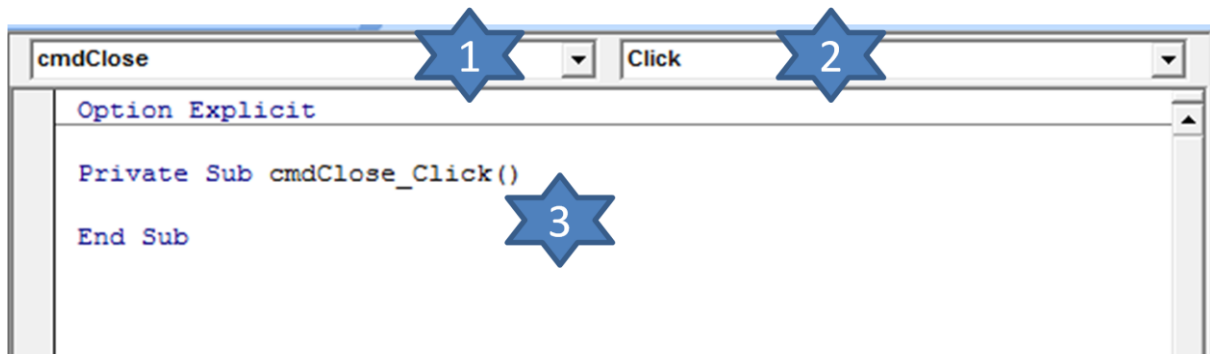


Abbildung 4: Ereignisbehandlung festlegen

Sofort nach der Wahl eines Ereignisses ② wird automatisch ein leerer Prozedurrumpf ③ erzeugt, in dem Sie die Codezeilen einfügen, die beim Eintritt des Ereignisses ausgeführt werden sollen.

Der Name der Ereignisprozedur (hier: `cmdClose_Click`) setzt sich zusammen aus dem Namen des Steuerelements (`cmdClose`) und dem Namen des Ereignisses (`Click`), verbunden durch einen Unterstrich.

3 Beispiel für die Erstellung eines Benutzerformulars

3.1 Steuerelemente

Im Folgenden wird ein einfacher Datumsrechner entwickelt, der zu einem vorgegebenen Kalenderdatum eine bestimmte Anzahl von Kalender- oder Arbeitstagen addiert oder subtrahiert. Unterstellt wird eine 5-Tage-Woche. Gesetzliche Wochenfeiertage werden der Einfachheit nicht berücksichtigt. Abb. 5 veranschaulicht die Formulargestaltung und die Benennung der darauf platzierten Steuerelemente.

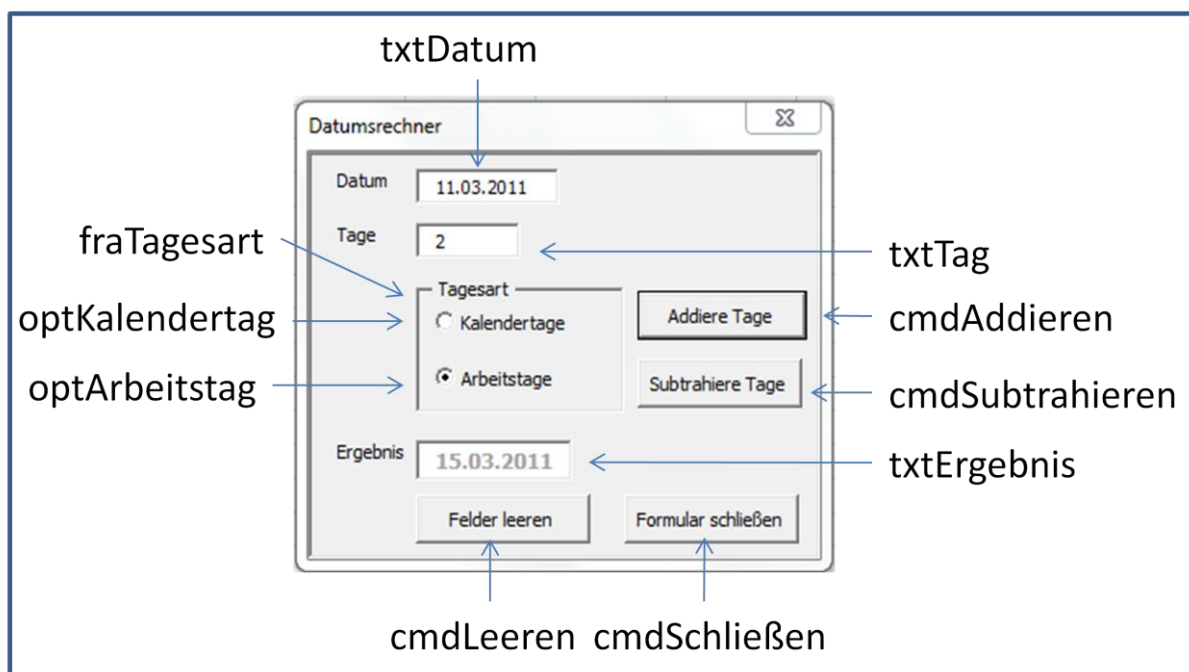


Abbildung 5: Formular für Datumsrechner

Workshop 9: VBA-Programmierung mit MS Excel

Das Formular `frmDatumsRechner` beinhaltet genau 13 Steuerelemente:

- 3 Textfelder
- 3 zugehörige Bezeichnungsfelder
- 4 Befehlsschaltflächen
- 2 Optionsfelder
- 1 Rahmen zur Bildung der Optionsgruppe

Mit folgender Standardprozedur lassen sich die Namen aller Steuerelemente des Formulars in das Direktfenster schreiben:

```
Private Sub SteuerelementeListen()  
    Dim ctl As Control  
    For Each ctl In frmDatumsrechner.Controls  
        Debug.Print ctl.Name & Space(15 - Len(ctl.Name)) & _  
                    Space(3) & TypeName(ctl)  
    Next ctl  
End Sub
```

3.2 Ereignisprozeduren

3.2.1 Ereignisprozedur für die Arbeitsmappe

Das benutzerdefinierte Dialogfeld `frmDatumsrechner` soll automatisch gestartet werden mit folgender Ereignisprozedur auf der Ebene der aktuellen Arbeitsmappe:

```
Private Sub Workbook_Open()  
    frmDatumsrechner.Show  
End Sub
```

3.2.2 Ereignisprozeduren für das Formular

Das Formular (UserForm) enthält zwei Ereignisprozeduren: Eine für das Ereignis `Initialize` und eine weitere für das Ereignis `QueryClose`.

```
Private Sub UserForm_Initialize()  
    Me!txtDatum = Date  
    Me!optKalendertage = True  
    Me!cmdAddieren.Enabled = False  
    Me!txtErgebnis.Enabled = False  
    Me!cmdSubtrahieren.Enabled = False  
    With Me!txtErgebnis  
        .Enabled = False  
        .ForeColor = vbBlue  
        With .Font  
            .Bold = True  
            .Size = 10  
        End With  
    End With  
    Me!txtTage.SetFocus  
End Sub
```

Workshop 9: VBA-Programmierung mit MS Excel

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    ' Verhindert das Schließen des Formulars
    ' durch das Kreuz in der Titelleiste
    Const conMsg As String = "Dieses Fenster kann nur über die Befehlsschaltfläche 'Schließen' geschlossen werden!"
    If CloseMode = vbFormControlMenu Then
        MsgBox Prompt:=conMsg, Buttons:=vbExclamation, Title:="Formular schließen"
        Cancel = True
    End If
End Sub
```

3.2.3 Ereignisprozeduren für die Steuerelemente des Formulars

Für alle vier Befehlsschaltflächen werden Ereignisprozeduren benötigt. Außerdem wird für das Textfeld `txtTage` eine Ereignisprozedur für das `Change`-Ereignis bereitgestellt.

```
Private Sub cmdAddieren_Click()
    If Me!optKalendertage Then
        Me!txtErgebnis = DateAdd("d", Me!txtTage, Me!txtDatum)
    Else
        Me!txtErgebnis = ArbeitstageZaehlen(Me!txtTage, Me!txtDatum)
    End If
End Sub
```

```
Private Sub cmdSubtrahieren_Click()
    If Me!optKalendertage Then
        Me!txtErgebnis = DateAdd("d", Me!txtTage * -1, Me!txtDatum)
    Else
        Me!txtErgebnis = ArbeitstageZaehlen(Me!txtTage * -1, Me!txtDatum)
    End If
End Sub
```

```
Private Sub cmdLeeren_Click()
    Dim ctl As Control
    For Each ctl In Me.Controls
        If TypeName(ctl) = "TextBox" Then
            ctl.Value = vbNullString
        End If
    Next ctl
    Me!optKalendertage = True
    Me!cmdAddieren.Enabled = False
    Me!cmdSubtrahieren.Enabled = False
End Sub
```

Workshop 9: VBA-Programmierung mit MS Excel

```
Private Sub cmdSchließen_Click()  
    Me.Hide  
    Unload Me  
End Sub
```

```
Private Sub txtTage_Change()  
    With Me!txtTage  
        If .Value > 0 Then  
            If IsDate(Me!txtDatum) Then  
                Me!cmdAddieren.Enabled = True  
                Me!cmdSubtrahieren.Enabled = True  
            Else  
                Me!txtDatum.SetFocus  
            End If  
        Else  
            .Value = vbNullString  
            .SetFocus  
        End If  
    End With  
End Sub
```

3.3 Standardprozeduren

Für die Addition bzw. Subtraktion von Arbeitstagen werden zwei Public-Funktionen eingesetzt:

```
Public Function IstWochenende(ByVal dtmDatum As Variant) As Boolean  
    ' Bestimmt, ob ein Datum auf ein Wochenende fällt  
    Const conSonntag As Integer = 7  
    Const conSamstag As Integer = 6  
    Select Case Weekday(dtmDatum, vbMonday)  
        Case conSamstag, conSonntag  
            IstWochenende = True  
        Case Else  
            IstWochenende = False  
    End Select  
End Function  
  
Public Function ArbeitstageZaehlen(ByVal intTage As Integer, _  
    ByVal dtmDatum As Date) As Date  
    ' Zählt Arbeitstage  
    ' Benötigt die Funktion IstWochenende  
    Dim intZaehler As Integer  
    Do While intZaehler <= Abs(intTage)  
        If Not IstWochenende(dtmDatum) Then  
            intZaehler = intZaehler + 1  
        End If  
        dtmDatum = dtmDatum + (1 * Sgn(intTage))  
    Loop  
    ArbeitstageZaehlen = dtmDatum - (1 * Sgn(intTage))  
End Function
```