HARALD NAHRSTEDT

# Excel + VBA

## Ergänzungen

**Kapitel**

Einführung in VBA

### Einfache Dialoge in VBA

| | |
|---|---|
| Erstellt am | 15.10.2011 |
| Bearbeitet am | 28.03.2012 |
| Beschreibung | In VBA gibt es einfache Dialogfunktionen, die einen schnellen Prozeduraufbau erlauben. Anders als die Methode Dialogs, sind sie mit wenigen Parametern programmierbar. |

In VBA gibt es einfache Dialogfunktionen, die einen schnellen Prozeduraufbau erlauben. Anders als die Dialogs, sind sie mit wenigen Parametern programmierbar.

## 1 Die Inputbox-Funktion

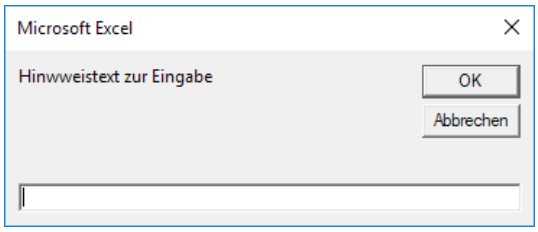
Nicht selten möchte man einer Variablen in einer Excel-Prozedur einen Wert zuweisen, der nicht fest vorgeben werden soll. Eine einfache Möglichkeit ist die Inputbox. Die Syntax lautet

InputBox (prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])

In eckigen Klammern stehende Parameter müssen dabei nicht gesetzt werden, so dass ein einfaches Beispiel die Form

```
Sub Test1()
   Dim vTest As Variant
   vTest = InputBox("Hinwweistext zur Eingabe")
End Sub
```

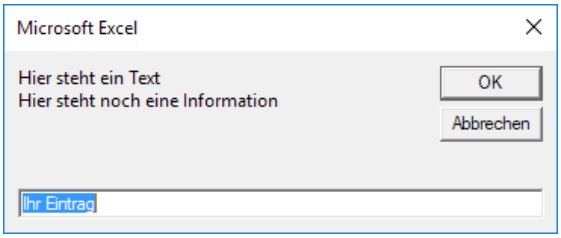
haben kann.




Fällt der Hinweistext etwas länger aus, kann auch ein Zeilenvorschub eingesetzt werden. Das einfache Beispiel, bei dem auch schon ein Vorgabetext steht

```
Sub Test2()
   Dim vTest As Variant
   vTest = InputBox("Hier steht ein Text" & vbLf & _
      "Hier steht noch eine Information", , "Ihr Eintrag")
End Sub
```

sieht dann wie folgt aus.



Über die Parameter xpos und ypos kann die Inputbox an eine gewünschte Stelle platziert werden, Die Werte xpos=1 und ypos=1 positionieren sie in die linke obere Ecke des Anwendungsfensters.

Die Parameter helpfile und context werden eher selten gebraucht. Sie setzen eine Hilfedatei (*.chm) voraus, die mit einem entsprechenden Hilfsprogramm erstellt werden muss.

**2 Die MsgBox-Methode**

Die Prozedur MsgBox (Messagebox) dient zur Ausgabe von Informationen. Sie kann auf zwei Arten genutzt werden. Die erste Möglichkeit ist eine einfache Ausgabe von Text. Zur Bestätigung, dass die Msgbox erkannt wurde, muss ein OK-Button betätigt werden.

Die Syntax der MsgBox für den ersten Fall lautet

```
MsgBox(Prompt,Buttons As vbMsgBoxStyle = vbOKOnly, _
   [Titel],[HelpFile],[Context] As vbMsgBoxResult)
```

Auch hier müssen die in eckigen Klammern gesetzten Parameter nicht gesetzt werden. Prompt steht auch hier für konstanten Text oder eine Variable, deren Inhalt gezeigt werden soll. Buttons legt die Anzahl und Form der Buttons fest. Siehe nachfolgende Tabelle. Ebenso steht eine Auswahl von Symbolen zur Verfügung, so dass auch mehrere Buttons kombiniert werden können. Die Verknüpfung erfolgt mit dem + Operator (nicht & Operator).

Hier ein paar einfache Beispiele.

```
MsgBox "Dieser Text wird angezeigt!"
MsgBox Variable
MsgBox "Dieser Text ist wichtig", vbOKOnly + vbCritical
MsgBox "Ausgabetext", vbCritical, "Ausgabetitel"
```

**3 Die Reaktion auf eine MsgBox abfragen**

Aus der Syntax der MsgBox-Methode wird ersichtlich, dass es sich hierbei um eine Funktion handelt. (As vbMsgBoxResult), die einen Wert vom Typ vbMsgBoxResult liefert. Dieser Wert kann abgefragt und zur weiteren Steuerung der Prozedur verwendet werden. So wie im nachfolgenden Beispiel:

```
Sub Test3()
   Dim Result As Variant
   Result = MsgBox("Bitte drücken Sie die OK-Taste!", _
      vbOKCancel, "Abfrage")
   If Result = vbOK Then
      MsgBox "Richtig! " & _
      "Sie haben die OK-Taste gedrückt!", vbInformation
   Else
      MsgBox "Falsch! " & _
      "Das war die Abbrechen-Taste.", vbCritical
```

```
   End If
End Sub
```

Die nachfolgenden Tabellen zeigen noch einmal die verschiedenen Buttons und Symbole, so wie die Reaktionswerte für eine Abfrage. Außer den VB Konstanten können auch die dazugehörigen Binärwerte genutzt werden. So kann statt vbOKCancel + vbCritical auch der Wert: 1 + 16 = 17 gesetzt werden.

Tabelle 1: Buttons

| *Wert* | *Konstantenname* | *Beschreibung* |
|---|---|---|
| 0 | vbOKOnly | OK |
| 1 | vbOKCancel | OK - Abbrechen |
| 2 | vbAbortRetry | Wiederholen - Ignorieren |
| 3 | vbYesNoCancel | Ja - Nein - Abbrechen |
| 4 | vbYesNo | Ja – Nein |
| 5 | vbRetryCancel | Wiederholen - Abbrechen |

Tabelle 2: Results

| *Wert* | Konstantenname | *Reaktion* |
|---|---|---|
| 1 | vbOK | Klick auf OK |
| 2 | vbCancel | Klick auf Abbrechen |
| 3 | vbAbort | Klick auf Abbrechen |
| 4 | vbRetry | Klick auf Wiederholen |
| 5 | vbIgnore | Klick auf Ignorieren |
| 6 | vbYes | Klick auf Ja |
| *7* | *vbNo* | *Klick auf Nein* |

Tabelle 3: Symbole

| *Wert* | *Konstantenname* | *Beschreibung* |
|---|---|---|
| 16 | vbCritical | Kritische Meldung |
| 32 | vbQuestion | Fragezeichen |
| 48 | vbExclamation | Warnung |
| 64 | vbInformation | Information |

**4 Den Focus auf einen anderen Button legen**

Beim Aufruf der MsgBox bekommt standardmäßig die erste Schaltfläche von links den Focus. Soll eine andere Schaltfläche den Focus bekommen, kann dies durch Addition von VB Konstanten erfolgen. Eine Möglichkeit zeigt das nachfolgende Beispiel.

```
Sub Beispiel_Focus()
   Dim Result As Variant
   Result = MsgBox("Bitte wählen Sie!", _
      vbYesNoCancel + vbQuestion + vbDefaultButton2, _
      "ATTENTION!!!")
   If Result = vbNo Then
      MsgBox "Nein Taste gedrückt!", vbCritical
   Else
      If Result = vbYes Then
         MsgBox "Ja Taste gedrückt!", vbInformation
      Else
         MsgBox "Cancel gedrückt!", vbQuestion
      End If
   End If
End Sub
```

Entsprechend bekommt mit vbDefaultButton3 das dritte Button von links den Focus.

**5 Die MsgBox verschieben**

Die MsgBox gehört wohl mit zu den am häufigsten verwendeten Dialogfenstern. Leider wird die MsgBox aber immer in der Bildschirmmitte angezeigt. Soll die MsgBox an einer beliebigen Position auf dem Desktop platziert werden, müssen wir wieder einmal in die API-Trickkiste greifen.

Die Prozedur Aufruf_MsgBox liest die x-Position aus der Zelle A1 und die y-Position aus der Zelle B1, bevor sie die alternative Funktion msg startet, die eine Hook-Prozedur erstellt, dessen Fenster frei gewählt werden kann, und in der eine MsgBox gestartet wird. So die etwas einfache Beschreibung des Vorgangs.

ACHTUNG! Hook-Prozeduren können bei unsachgemäßer Handhabung das System verändern. Der Einsatz dargestellter Prozeduren geschieht auf eigene Gefahr.

```
Private Declare Function FindWindow Lib "user32" Alias _
      "FindWindowA" (ByVal lpClassName As String, ByVal _
      lpWindowName As String) As Long
Public Declare Function UnhookWindowsHookEx Lib "user32" _
      (ByVal hHook As Long) As Long
Public Declare Function GetWindowLong Lib "user32" _
      Alias "GetWindowLongA" _
      (ByVal hwnd As Long, ByVal nIndex As Long) As Long
Public Declare Function GetCurrentThreadId _
      Lib "kernel32" () As Long
Public Declare Function SetWindowsHookEx Lib "user32" Alias _
```

```
      "SetWindowsHookExA" (ByVal idHook As Long, _
      ByVal lpfn As Long, ByVal hmod As Long, _
      ByVal dwThreadId As Long) As Long
Public Declare Function SetWindowPos Lib "user32" _
      (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, _
      ByVal x As Long, ByVal y As Long, ByVal cx As Long, _
      ByVal cy As Long, ByVal wFlags As Long) As Long
Public Declare Function GetWindowRect Lib "user32" _
      (ByVal hwnd As Long, lpRect As RECT) As Long

Public Const GWL_HINSTANCE = (-6)
Public Const SWP_NOSIZE = &H1
Public Const SWP_NOZORDER = &H4
Public Const SWP_NOACTIVATE = &H10
Public Const HCBT_ACTIVATE = 5
Public Const WH_CBT = 5

Public hHook As Long
Public MsgBoxPosX As Integer
Public MsgBoxPosY As Integer

Type RECT
   Left As Long
   Top As Long
   Right As Long
   Bottom As Long
End Type

Function WinProc(ByVal lMsg As Long, ByVal wParam As Long, ByVal
lParam As Long) As Long
   If lMsg = HCBT_ACTIVATE Then
      SetWindowPos wParam, 0, MsgBoxPosX, MsgBoxPosY, _
      0, 0, SWP_NOSIZE Or SWP_NOZORDER Or SWP_NOACTIVATE
      UnhookWindowsHookEx hHook
   End If
   WinProc = False
End Function

Function msg(msgText As String, msgButton As Long, _
   msgTitel As String, lXPos As Long, lYPos As Long)
   Dim hInst As Long
   Dim XLInst As Long
   Dim Thread As Long
   MsgBoxPosX = lXPos
   MsgBoxPosY = lYPos
   XLInst = FindWindow("xlmain", vbNullString)
   hInst = GetWindowLong(XLInst, GWL_HINSTANCE)
   Thread = GetCurrentThreadId()
   hHook = SetWindowsHookEx(WH_CBT, AddressOf WinProc, _
      hInst, Thread)
   MsgBox msgText, msgButton, msgTitel, 0, 0
End Function

Sub Aufruf_MsgBox()
   Dim Dummy      As Byte
```

```
    Dim sHilfe      As String
    Dim lXPos       As Long
    Dim lYPos       As Long

    'linke obere Ecke aus A1 und B1
    lXPos = Val(ActiveSheet.Cells(1, 1))
    lYPos = Val(ActiveSheet.Cells(1, 2))

    sHilfe = "Position:" & vbLf & _
       "x =" & Str(lXPos) & vbLf & _
       "y =" & Str(lYPos)
    Dummy = msg(sHilfe, 64, "Achtung!", lXPos, lYPos)
End Sub
```