## 2.1 Dialoge mit OpenOffice.org

Um einen Dialog zu entwickeln, stellt die OpenOffice-Entwicklungsumgebung einen Designer zur Verfügung, der es erlaubt, den Dialog grafisch zu konfigurieren. Ein Dialog muß also nicht »manuell« programmiert werden, sondern kann mit Drag&Drop entwickelt werden.

Alternativ gibt es natürlich die Möglichkeit, einen Dialog ohne Unterstützung des Designers zu entwickeln. Im ersten folgenden Beispiel zur Entwicklung von Dialogen beschreiten wir zur Einarbeitung den manuellen Weg. Das folgende Beispiel erstellt einen Dialog ohne Steuerelemente und ohne Designer-Unterstützung:

```
01: Sub ShowDialog
02:    Dim oDlgModel As Object
03:    Dim oWindow As Object
04:    Dim oDlg As Object
05:
06:    REM Initialisierung der Eigenschaften des Dialogs
07:    oDlgModel=CreateUnoService("com.sun.star.awt.UnoControlDialogModel")
08:    oDlgModel.PositionX = 50
09:    oDlgModel.PositionY = 50
10:    oDlgModel.Width  = 150
11:    oDlgModel.Height = 150
12:    oDlgModel.Title  = "Beispiel - Dialog ohne Designer"
13:
14:    '* Mittels des Modells den Dialog anzeigen
15:    oDlg = CreateUnoService("com.sun.star.awt.UnoControlDialog")
16:    oWindow = CreateUnoService("com.sun.star.awt.Toolkit")
17:    oDlg.setModel(oDlgModel)
18:    oDlg.createPeer(oWindow, null)
19:    oDlg.execute()
20: End Sub
```

Zuerst werden mit der *Dim*-Anweisung die drei Variablen *oDlgModel, oWindow* und *oDlg* vom Typ *Object* angelegt. *oDlgModel* legt den Bauplan (das Modell) des Dialogs fest. In Zeile 7 wird zuerst mit *CreateUnoService* die Vorlage für einen Dialog aus der Bibliothek gelesen. Der Parameter *com.sun.star.awt.UnoControlDialogModel* gibt an, wo in der Bibliothek das Modell zu laden ist. Die Zeilen 08 und 09 enthalten die Positionsangabe, wo der Dialog später eingeblendet wird. Ausgehend von der oberen linken Ecke des Dialogs wird der Dialog fünfzig Pixel waagerecht und fünfzig Pixel senkrecht vom Bildschirmrand angezeigt. Die Eigenschaften *Width* und *Height* (Zeilen 10 und 11) legen jeweils die Größe und Breite des Dialogs in Pixel fest. In Zeile 12 wird der Titel des Dialogs zugewiesen.

In der Zeile 15 wird abermals *CreateUnoService* ein Parameter übergeben, jetzt wird nicht mehr ein Modell, sondern der reale Dialog erzeugt. Die Fläche, auf der der Dialog dargestellt wird, wird in Zeile 16 zugewiesen.

*SetModel*, Zeile 17, des Objekts *oDlg* übergibt dem Dialog den Bauplan, der in der Variable *oDlgModel* steckt. In Zeile 18 sorgt der Aufruf von *CreatePeer* dafür, daß ein Fenster unterhalb des Hauptfensters erzeugt wird. Der erste Parameter von *CreatePeer* sorgt dafür, daß dem Dialog die zuvor angeforderte Zeichenfläche zugeordnet wird. Da im Beispiel das Hauptfenster der Desktop ist, wird als zweiter Parameter Null übergeben. Der Aufruf von *Execute* in Zeile 19 sorgt zum Schluß dafür, daß der Dialog angezeigt wird.

Nach dieser Einführung in die manuelle Erstellung eines Dialogs widmen wir uns dem Dialog-Designer. Dieser erleichtert die schnelle Entwicklung von Makros mit Oberflächen erheblich.

Um einen Dialog mit dem Designer zu erstellen, muß zuerst der Dialog *Makro* aufgerufen werden. Dies kann entweder direkt aus der Entwicklungsumgebung oder auch aus einem Text- oder Tabellendokument erfolgen. Im jeweiligen Hauptmenü ist das Menü *Extras/Makros/Makros verwalten/OpenOffice.org Basic* auszuwählen.

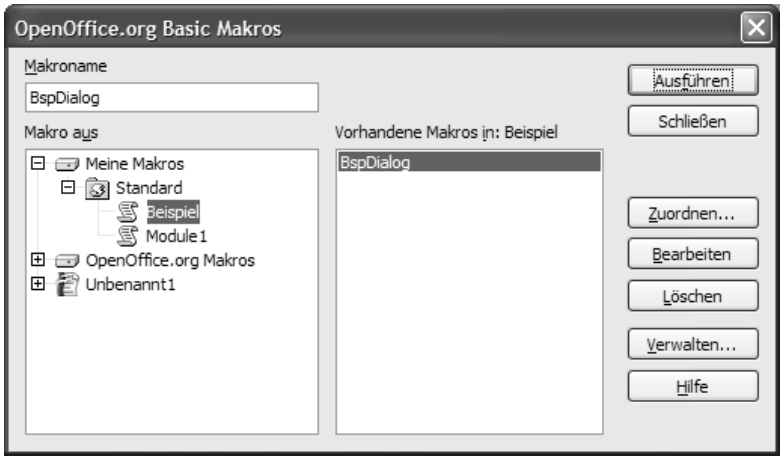


Abb. 2.1: Der Dialog »Makro«

Zuerst muß zur Entwicklung ein neuer Dialog angelegt werden. Nach dem Aufruf des Dialogs *Makros* ist dort die Schaltfläche mit der Bezeichnung *Verwalten…* zu betätigen. Über den sich dann öffnenden Dialog *Makros verwalten* läßt sich auf dem Register *Dialoge* entweder ein bereits vorhandener Dialog bearbeiten (Schaltfläche mit der Bezeichnung *Bearbeiten*) oder ein neuer erstellen (Schaltfläche mit der Bezeichnung *Neu…*).

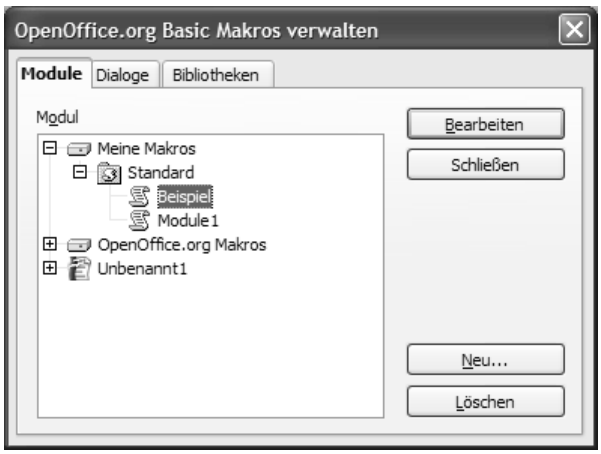


Abb. 2.2: Der Dialog »Makros verwalten«

Um einen ersten Dialog zu erzeugen, muß die Schaltfläche *Neu…* betätigt werden.

### 2.1.1 Der Dialog-Designer

Nach Betätigung der Schaltfläche öffnet sich ein Eingabefenster. Hier wird der Name des neuen Dialogs erfaßt. Darunter wird der Dialog im Quelltext angesprochen. Für das erste Beispiel wird *BspDialog* eingegeben. Nach der Eingabe des Namens und der Betätigung der Schaltfläche *OK* wird die Baumanzeige um einen neuen Eintrag erweitert. Die beiden noch offenen Dialoge können jetzt jeweils über die Schaltfläche *Schließen* verlassen werden. Um den Dialog weiter bearbeiten zu können, muß die jetzt vorhanden Registerkarte in der Entwicklungsumgebung, die den Namen des neuen Dialogs trägt, angeklickt werden. Nun wird in der Entwicklungsumgebung der Dialog-Designer geöffnet und der neue Dialog angezeigt, siehe Abb. 2.3.


Abb. 2.3: Das neue Dialogfenster

Auf diesem Dialogfenster werden später Steuerelemente plaziert. Zunächst sollte der Dialog allerdings konfiguriert werden. Hierzu ist einer der Randbereiche des Dialogs mit der Maus anzuklicken. Wurde der Dialog erfolgreich selektiert, erscheinen innerhalb des Designers um den Dialog herum angeordnet grüne Markierungen. Fährt man mit dem Mauscursor über eine solche Markierung, verändert er sich. Er zeigt dem Anwender damit an, daß nun die Größe des Dialogfensters bei gedrückter linker Maustaste verändert werden kann.

### 2.1.2 Eigenschaften des Dialogs anpassen

Um das Beispiel zu vervollständigen, soll der Dialog noch eine Überschrift erhalten, auch *Titel* genannt. Sie wird links oben im blau unterlegten Balken des Fensters angezeigt. Sie wird am besten über das *Eigenschaften*-Fenster eingestellt. Um die Eigenschaften des Dialogs konfigurieren zu können, muß das Eigenschaften-Fenster in der Entwicklungsumgebung angezeigt werden. Es wird über ein Symbol in der Werkzeugleiste geöffnet.

Die Werkzeugleiste wird durch das entsprechende Symbol aktiviert. Zu finden ist es in der Makroleiste.

Abb. 2.4: Werkzeugleiste aktivieren

Die Werkzeugleiste enthält diverse Symbole. Die ersten drei sind allerdings keine Steuerelemente, sondern haben andere Aufgaben. So befindet sich hier die Aktivierung des Eigenschaften-Fensters und das Symbol zum Starten des Testmodus für den aktuellen Dialog. Die weiteren Symbole sind Steuerelemente, die in Verbindung mit einem Dialog benutzt werden. Beispielsweise ist dort ein Symbol zu finden, das eine Schaltfläche, einen *CommandButton*, anzeigt. Diese Schaltfläche kann auf dem Dialog abgelegt werden. Um das Beispiel fortzuführen, sollte als nächstes das Eigenschaften-Fenster geöffnet werden.


Abb. 2.5: Die Werkzeugleiste mit den Steuerelementen

Ist die Werkzeugleiste einmal geöffnet, sieht man das Symbol zum Öffnen des Eigenschaften-Fensters im oberen Bereich. Alternativ dazu kann es durch Aktivierung des Kontextmenüs (Mausklick rechts auf dem Dialog) angezeigt werden.

Abb. 2.6: Das Symbol für das Eigenschaften-Fenster

Das Eigenschaften-Fenster ermöglicht den Zugriff auf die Eigenschaften, wie beispielsweise Titel, Höhe und Breite des Dialogs. Diese zuerst sichtbare Registerkarte heißt *Allgemein*. Über eine weitere Registerkarte wird auf die Ereignisse, die einem Dialog zugeordnet werden können, zugegriffen, dazu später mehr. Für das Beispiel wird nun die Eigenschaft *Titel* bearbeitet. Dazu wird der Mauscursor im Textfeld direkt neben der Beschriftung *Titel* positioniert und der Text *Beispiel-Dialog* eingetragen. Abb. 2.7 zeigt zum Vergleich die entsprechend konfigurierte Eigenschaft.

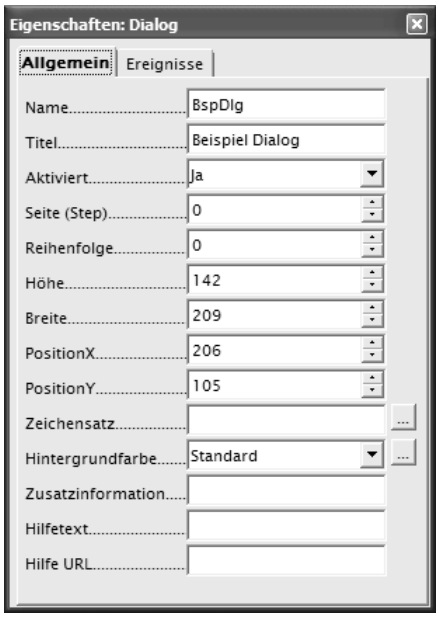


Abb. 2.7: Das Eigenschaften-Fenster zur Konfiguration des Dialogs

Nachdem der Titel-Text über die gleichnamige Eigenschaft im Eigenschaften-Fenster eingetragen wurde, ist er direkt im Dialog-Designer sichtbar. Das Eigenschaften-Fenster kann nun geschlossen werden, was über das Schließen-Symbol rechts oben geschieht. Jetzt ist ein guter Zeitpunkt, um den Dialog zu testen. Dies erfolgt über den Testmodus.

Auch der Testmodus für ein Formular wird über die Werkzeugleiste aktiviert. Hierzu ist nur das entsprechende Symbol einmalig anzuklicken. Nun wird der Dialog außerhalb der Entwicklungsumgebung angezeigt. Er kann verschoben, vergrößert und verkleinert werden. Ist man mit dem Ergebnis zufrieden, ist die Zeit gekommen, den Dialog direkt über ein Makro anzusteuern. Hierzu muß etwas Quelltext eingegeben werden. Dazu öffnet man über das Menü *Extras/Makros/Makros verwalten/OpenOffice.org Basic* den Makro-Dialog.

Abb. 2.8: Das Symbol um den Testmodus für Formular einzuschalten

Dort wird durch Markierung des Ordners *Standard* und anschließender Betätigung der Schaltfläche mit der Bezeichnung *Neu* ein neues Makro angelegt. Das Makro erhält automatisch den Namen *Macro1*. Die Entwicklungsumgebung öffnet sich nun und der Rumpf für das neue Makro ist bereits angelegt. Seine Bezeichnung läßt sich anpassen, im Beispiel wurde der Name *BspDialog* gewählt. Im Makro muß nun Code hinterlegt werden, der bewirkt, daß der Dialog geladen und angezeigt wird, sobald es gestartet wurde.

Der komplette Quelltext sieht wie folgt aus:

```
01: Sub BspDialog
02:   Dim dlg, dlgBeschreibung As Object
03:    dlgBeschreibung = DialogLibraries.Standard.BspDlg
```

```
04:    dlg = CreateUnoDialog(dlgBeschreibung)
05:    dlg.Execute()
06: End Sub
```

In Zeile 1 wird zuerst eine neue Variable vom Typ *Object* mit der Bezeichnung *dlg* angelegt. Diese Variable zeigt später den Dialog an. Ebenso eine zweite Variable *dlgBeschreibung*, auch vom Typ *object*. Ihr wird in Zeile 2 ein Wert zugewiesen. Dabei handelt es sich um den Ort, an dem die Dialogbeschreibung (die Beschreibung wurde durch den Dialog-Designer erzeugt) gespeichert wurde. Der erste Teil der Anweisung *DialogLibraries* in Zeile 3 sorgt dafür, daß auf die Bibliotheken von OpenOffice.org zugegriffen wird. Die nächste Angabe ist die Bezeichnung eines Moduls, in dem die Dialogbeschreibung gespeichert wurde. In unserem Fall ist dies die Bibliothek *Standard* und die entsprechende Beschreibung heißt natürlich genauso wie der Dialog *BspDlg*.

Die Anweisung in Zeile 4 sorgt dafür, daß die Dialogbeschreibung geladen und der Variablen *dlg* zugewiesen wird. Das macht die Systemfunktion *CreateUnoDialog*. Als Parameter erwartet sie den Namen der Variablen mit der Beschreibung des Dialogs. Mit der letzten Anweisung wird der Dialog angezeigt (Zeile 5). Die entsprechende Funktion trägt den Namen *execute*. Die Anzeige eines Dialogs beendet die Funktion *EndExecute*.

Um sich den Dialog makrogesteuert anzeigen zu lassen, ist in der Makroleiste das Symbol zum Ausführen eines Makros zu betätigen. Nach dem Start des Makros wird der Dialog wie zuvor im Testmodus angezeigt und er kann verschoben werden, auch die Größe kann geändert werden.

Abb. 2.9: Das Symbol zum Ausführen eines Makros

Viel kann der erste Dialog noch nicht. Das Beispiel soll noch etwas ausgebaut werden, es soll ein Ereignis verarbeitet werden können. Sobald ein Benutzer den Mauscursor auf den Dialog bewegt und die linke Maustaste betätigt, soll ein Meldungsfenster angezeigt werden. Um diese Aktion zu verarbeiten, damit der Programmcode darauf reagieren kann, muß der Dialog mit einem Ereignis verknüpft werden.

### 2.1.3 Ereignisverarbeitung

Bevor mit der Verknüpfung von Dialog- und Ereigniscode begonnen wird, ein Wort zur Verfahrensweise. Man entwickelt zuerst eine Prozedur, in der die Aktionen ausgeführt werden, wenn das Ereignis eintritt. Um beim Beispiel zu bleiben, wird zuerst eine neue Prozedur *BspDialog_Click* geschrieben. In dieser Prozedur wird ein Meldungsfenster mit dem Text *Mausklick wurde ausgeführt* aufgerufen. Sobald diese Prozedur erstellt ist, kann sie mit dem entsprechenden Ereignis des Dialogs verknüpft werden. Es folgt nun der Quelltext für die Prozedur *BspDialog_Click*:

```
Sub BspDialog_Click
  MsgBox "Mausklick wurde ausgeführt!"
End Sub
```

Die bereits bekannte Messagebox (Meldungsfenster) wird hier wieder zur Ausgabe von Text genutzt. Die Prozedur kann direkt unterhalb des Codes zur Anzeige des Dialogs eingegeben werden.

Sobald die Prozedur fertiggestellt ist, muß in den Dialog-Designer gewechselt werden. Der Dialog ist zu markieren, so daß die grünen Markierungspunkte zu sehen

sind. Dann wird das Eigenschaften-Fenster aufgerufen. Dort wird die zweite Registerkarte *Eigenschaften* angeklickt. Alle Ereignisse, auf die der Dialog reagieren kann, sind hier zu sehen. Das für das Beispiel passende Ereignis, *Beim Auslösen*, ist das erste in der Liste. Am rechten Rand in derselben Zeile befindet sich eine Schaltfläche mit drei Punkten. Sie öffnet den Dialog *Makro zuweisen*.

Dort wird die Verknüpfung der entwickelten Prozedur mit dem Ereignis des Dialogs vorgenommen, siehe Abb. 2.10 und Abb. 2.11.

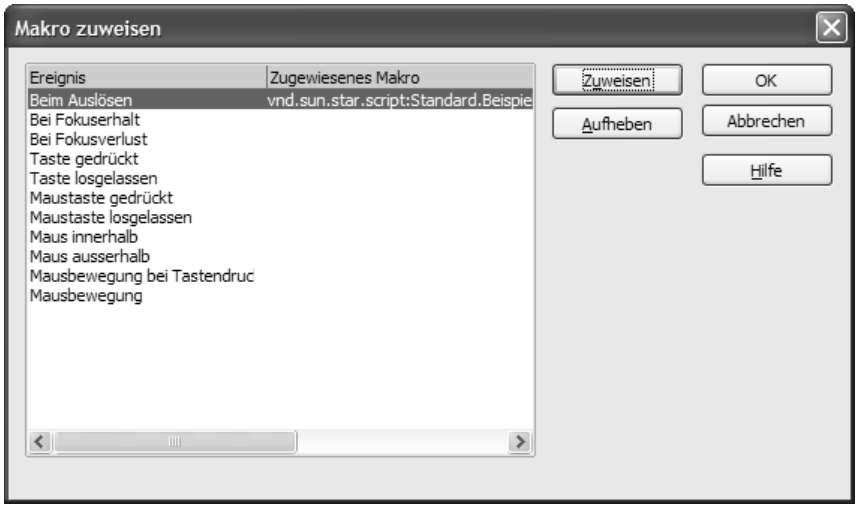


Abb. 2.10: Der Dialog *Makro zuweisen* zur Verknüpfung von Makro und Ereignis


Abb. 2.11: Der Makro-Selektor

Zur Verknüpfung von Prozedur und Ereignis sind jetzt mehrere Aktionen durchzuführen. Zuerst muß die Schaltfläche *Zuweisen* betätigt werden. Als nächstes öffnet sich der Dialog *Makro-Selektor*. In der Box *Bibliothek* muß der Punkt *My Macros* und dort das Modul *Standard* erweitert werden (Rechtsklick auf das Plus-Symbol). In der Bibliothek *Standard* wird das gewünschte Modul mit dem Code markiert. In der rechten Box *Name des Makros* werden alle Prozeduren und Funktionen aus dem gewählten Modul angezeigt. Für das Beispiel wird in der Box *BspDialog_Click* selektiert. Anschließend ist die Schaltfläche OK zu betätigen. Der Dialog wird automatisch geschlossen. In der Spalte *Zugewiesenes Makro* erscheint jetzt ein entsprechender Eintrag mit der Bezeichnung der Prozedur. Dies waren bereits alle notwendigen Arbeitsschritte zur Verknüpfung von Prozedur und Ereignis.

Eine bereits bestehende Zuweisung wird durch Auswahl der entsprechenden Zeile und anschließende Betätigung der Schaltfläche *Aufheben* aufgehoben. *Abbrechen* schließt den Dialog, Änderungen werden nicht übernommen. Die Änderungen werden mit OK übernommen und gespeichert.

Die Prozedur *BspDialog_Click* im Modul *Standard* und das entsprechende Ereignis des Dialogs sind jetzt verknüpft. Zum Test wird wieder in den Makro-Editor gewechselt und das Makro *BspDialog* gestartet. Das Ergebnis ist in Abb. 2.12 zu sehen.


Abb. 2.12: Der Dialog mit aktiviertem Ereignis

### *Mausereignis manuell*

Im Abschnitt 8.2 wurde beschrieben, wie ein Dialog manuell ohne Designer entwikkelt wird. Der Verkettung der Ereignisprozedur mit dem CommandButton diente ja der Dialog *Makro zuweisen*. Man kann den CommandButton auch manuell mit einem Ereignis verknüpfen. Das ist immer dann sinnvoll, wenn ein Ereignis mit mehreren Steuerementen, beispielsweise Schaltflächen, verknüpft werden muß. Das folgende Beispiel zeigt einen Dialog mit zwei Schaltflächen. Nach einem Mausklick auf eine der beiden Schaltflächen soll eine Meldung erscheinen, welche der beiden Schaltflächen betätigt wurde. Im Dialog gibt es aber nur eine Ereignisprozedur, weshalb diese mit beiden Schaltflächen verknüpft werden muß und im Ereignis festgestellt werden kann, welche der beiden Schaltflächen betätigt wurde:

```
01: Sub BspButtonEvent
02:   Dim dlg As Object
03:   Dim myBtn1 As Object
04:   Dim myBtn2 As Object
05:   Dim MouseClick As Object
06:   dlg = LoadDialog("Standard", "Dialog1")
07:   myBtn1 = dlg.getControl("CommandButton1")
08:   myBtn2 = dlg.getControl("CommandButton2")
09:   MouseClick = CreateUnoListener(" MouseClick_", _
                                     "com.sun.star.awt.XActionListener")
10:   myBtn1.addActionListener(MouseClick)
11:   myBtn2.addActionListener(MouseClick)
12:   dlg.Execute()
13: End Sub
14:
15: Sub MouseClick_actionPerformed(Event As Object)
16:  Dim ausgabe As String
17:  If Event.Source.Model.Name = "CommandButton1" Then
18:    ausgabe = "CommandButton Nr.1 wurde betätigt!"
19:  Else
20:    ausgabe = "CommandButton Nr.2 wurde betätigt!"
21:  End If
22:  MsgBox ausgabe
23: End Sub
```

Zuerst werden vier Variablen vom Typ *Object* deklariert. Eine nimmt den Dialog auf, zwei die Schaltfächen *CommandButton1* und *CommandButton2*, die letzte wird für die Verknüpfung mit dem Ereignis benötigt. Als nächstes wird die Beschreibung des Dialogs geladen und der lokalen Variablen *dlg* zugewiesen. In Zeile 07 und 08 werden die beiden Schaltflächen den Variablen zugewiesen. Die Zeile 09 ist der erste Schritt zur manuellen Verknüpfung von CommandButton und Ereignis. Die Beschreibung eines Ereignisses wird aus der Bibliothek geladen und dem Objekt *MouseClick* zugewiesen. Das macht die Funktion *CreateUnoService*. Zwei Parameter vom Typ String werden übergeben, der erste enthält den ersten Teil des Namens der Ereignisprozedur *MouseClick_*, der zweite gibt an, in welcher Bibliothek die Beschreibung eines Mausklick-Ereignisses geladen wird.

An dieser Stelle kurz zur Zeile 15. Hier beginnt die Ereignisprozedur *MouseClick_actionPerformed*. Sie enthält den Quellcode, der ausgeführt werden soll, sobald der Mauszeiger über eine Schaltfläche bewegt und die linke Maustaste betätigt wird. Der Name der Prozedur beziehungsweise ein Teil davon wurde in Zeile 09 als zweiter Parameter dem Objekt *MouseClick* zugewiesen. Wieso nur ein Teil? Der Begriff *actionPerformed* ist fest vorgegeben. Wenn eine Ereignisprozedur manuell mit einem Steuerelement zur Laufzeit verknüpft wird, muß der Name *actionPerformed* angehängt werden.

Kehren wir zurück zur Zeile 10. Dort wird das Objekt *MouseClick* – dahinter verbirgt sich die Ereignisprozedur – mit dem jeweiligen CommandButton verknüpft. Dies geschieht durch Übergabe des Parameters *MausClick* an die Methode *addActionListener*. *AddActionListener* ist das Bindeglied zwischen Ereignisprozedur und CommandButton. Ab Zeile 15 bis 23 ist die Ereignisprozedur deklariert. Darin wird durch *Name* abgefragt, welcher Button betätigt wurde. Deshalb, weil der betätigte CommandButton als Parameter *Event As Object* übergeben wird.

### *Tastaturereignisse*

Neben den Mausereignissen wie *Maustaste gedrückt* und *Mausbewegung* gibt es noch die Tastatur-Ereignisse, beispielsweise *Taste gedrückt* und *Taste losgelassen*. Auch sie werden über den entsprechenden Dialog, wie im Beispiel zum Mausklick, mit einer Ereignisprozedur verknüpft.

Im folgenden Beispiel wurde eine Verknüpfung mit dem Ereignis *Taste gedrückt* und der Prozedur *KeyPressed* vorgenommen. In der Ereignisprozedur *KeyPressed* wird darauf reagiert, wenn eine Taste gedrückt wurde. Die Prozedur prüft, welche Taste gedrückt wurde, und teilt dies dem Benutzer über ein Meldungsfenster mit.

```
Dim dlg, dlgBeschreibung As Object

Sub BspKeyEvents
  BasicLibraries.LoadLibrary("Tools")
  dlg = LoadDialog("Standard", "BspDlg")
  dlg.Execute()
End Sub

Sub KeyPressed(Event As Object)
  Dim Taste As String
  Select Case Event.KeyCode
    Case com.sun.star.awt.Key.DELETE
      Taste = "Entf.-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.ESCAPE
      Taste = "ESC-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.DOWN
      Taste = "Pfeilrunter-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.UP
      Taste = "Pfeilrauf-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.LEFT
      Taste = "Pfeillinks-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.RIGHT
      Taste = "Pfeilrechts-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.TAB
      Taste = "Tab-Taste wurde gedrückt"
    Case com.sun.star.awt.Key.RETURN
      Taste = "Enter-Taste wurde gedrückt"
    Case Else
      Taste = "Taste: " & Event.KeyChar & " wurde gedrückt"
  End Select
  MsgBox Taste
End Sub
```

Die Information, welche Taste gedrückt wurde, erhält die Prozedur aus dem übergebenen Parameter vom Typ *Object*, der mit dem Namen *Event* angesprochen wird. In diesem Parameter »steckt« quasi der Auslöser des Ereignisses. Über das Auslesen der Eigenschaft *KeyCode* (enthält den Tastaturcode) kann nun festgestellt werden, welche Taste als Parameter »übergeben« wurde. Entsprechende Konstanten ermöglichen einen Vergleich.

Dialoge können die Eigenschaften *Name, Titel, Aktiviert, Seite, Reihenfolge, Höhe, Breite, PositionX, PositionY, Schrift, Hintergrundfarbe, Zusatzinformation, Hilfetext*

und *Hilfe-URL* aus Tabelle 2.1 (ab Seite 116 bis Seite 122) besitzen. Es stehen die Ereignisse *Bei Fokuserhalt, Bei Fokusverlust, Taste gedrückt, Taste losgelassen, Maus innerhalb, Mausbewegung bei Tastendruck, Mausbewegung, Maustaste gedrückt, Maustaste losgelassen* und *Maus ausserhalb* aus Tabelle 2.2 auf Seite 123 zur Verfügung.

Alle Ereignisse werden wie im Beispiel beschrieben verwendet. Das heißt, zuerst wird die entsprechende Prozedur erstellt und im Anschluß mit dem entsprechenden Ereignis des Dialogs verknüpft.

In den folgenden Abschnitten lernen Sie die Steuerelemente (Controls) kennen, die StarBasic für Dialoge bereithält. Mit ihnen programmiert man Dialoge, die die Eingaben eines Benutzers entgegennehmen und verarbeiten.

## 2.2 Die Steuerelemente von OpenOffice.org

Steuerelemente bilden neben den Dialogen das Rückgrat von OpenOffice.org-Makros. Nur mit Steuerelementen kann ein Dialog in einem Makro mit einer sinnvollen Funktion versehen werden. Die einzelnen Steuerelemente wie beispielsweise Schaltflächen und Textfelder sind das Thema des folgenden Abschnitts.

| *Kategorie* | *Englisch* | *Deutsch* |
|---|---|---|
| *Schalter* | | |
| | CommandButton | Schaltfläche |
| | Checkbox | Markierfeld |
| | RadioButton | Optionsfeld |
| *Eingabefelder* | | |
| | Textfield | Textfeld |
| | DateField | Datumsfeld |
| | Timefield | Zeitfeld |
| | NumericField | Zahlenfeld |
| | CurrencyField | Währungsfeld |
| | FormatedField | Formatiertes Feld |
| | PatternField | Maskenfeld |
| *Auswahllisten* | | |
| | Listbox | Listenfeld |
| | Combobox | Kombinationsfeld |
| *Grafische Elemente* | | |
| | ImageControl | Grafisches Kontrollfeld |
| | Groupbox | Gruppierungsrahmen |
| | Fixedline | Horizontale und vertikale Line |
| *Sonstige Steuerelemente* | | |
| | HScrollBar, VScrollBar | Horizontale und vertikale Bildlaufleiste |
| | ProgressBar | Fortschrittsbalken |
| | Label | Beschriftungsfeld |
| | FileControl | Dateiauswahl |

Tabelle 2.3: Die Steuerelemente kategorisiert