Visual BASIC 4.0 - Ein Kinderspiel

Ist Visual BASIC eine kinderleichte Programmiersprache oder eine Programmiersprache für Kinder? Ist VB möglicherweise eine ernst zu nehmende Programmiersprache? Oder ist VB ein professionelles Entwicklerwerkzeug für Client-Server-Lösungen, Workgroup-Lösungen und ähnliches?

Eduard Fleck

Visual BASIC 4.0 kann all das sein, kinderleicht, eine Programmiersprache für Kinder und ein professionelles Entwicklerwerkzeug. Jedoch nicht gleichzeitig. Um all den Funktionsumfang von VB zu nützen, wie zum Beispiel das Arbeiten und Erstellen von Objektklassen, die Erstellung von OLE-Servern, das Entwickeln von Lösungen mit Komponenten im Dienstemodell, die Projektverwaltung und vieles mehr, ist doch einiges an Einarbeitungszeit erforderlich. Diese ist aber mit der vorliegenden Version VB 4.0 zu kurz wie nie zuvor!

Exemplarisch für die Möglichkeiten von VB 4.0 möchte ich die OLE-Automatisation anführen, mit deren Hilfe Objekte mit ihren Eigenschaften und Methoden den verschiedensten Applikationen zur Verfügung gestellt werden können. Jedes OLE-Automatisierungsobjekt ist ein Baustein, den Sie im Code verwenden können, um Daten und Funktionen anderer Anwendungen so zusammenzustellen und offenzulegen, wie es in Ihrer Anwendung sinnvoll ist. Sie können beispielsweise eine Anwendung erstellen, die Microsoft Excel für Berechnungen verwendet und in der Berichte in Microsoft Word-Dokumenten erstellt werden.

Im folgenden Beispiel möchte ich das Zusammenspiel eines OLE-Servers mit einer Client-Applikation erläutern:

Anmeldedialog-Beispielanwendung

In diesem Beispiel wird mit Hilfe eines prozeßinternen OLE-Servers (DLL) ein Standardelement der Benutzeroberfläche angezeigt, in diesem Fall ein Dialogfeld zum Anmelden an eine Datenbank.

Auf das von diesem OLE-Server zur Verfügung gestellte SQLLoginDialog-Objekt kann von jedem OLE-Automatisierungs-Client zugegriffen werden. Im Visual BASIC-Projekt DLGTEST.VBP wird dieses Objekt z.B. verwendet. Ebenso kann von Microsoft Excel aus auf das SQLLoginDialog-Objekt zugegriffen werden.

Dieses Beispiel ist im Ordner \samples\oleserv von Visual BASIC 4.0 enthalten.

Erstellen des OLE-Servers:

Zu Beginn muß die Projektoption OLE-DLL auf „Einschränkungen verwenden“ eingestellt werden. Beim prozeßinternen OLE-Server handelt es sich um eine DLL, die im Prozeß einer anderen Anwendung ausgeführt wird. Ein außerprozeßlicher OLE-Server ist dagegen eine ausführbare EXE-Datei, die als unabhängiger Prozeß geführt wird. Die Kommunikation zwischen der OLE-Client-Anwendung und dem OLE-Server wird als Interprozeßkommunikation bezeichnet.

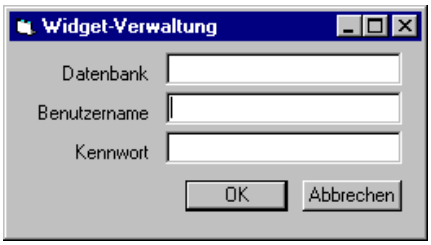
Der Projektname ergibt gemeinsam mit den vom OLE-Server bereitgestellten Klassennamen den eindeutigen Klassennamen (z.B.: SdtLoginDialogs.SQLLoginDialog für ein Objekt der LoginDialog - Klasse)

Prozeßinterne OLE-Server müssen immer eine „Sub Main“ Prozedur haben. Diese wird in ein Standardmodul eingetragen.

```
Sub Main()
    ' Initialisierungscode für die Objektanwendung
    ' würde hier stehen. Der Anmeldedialog erfordert keine
    ' Initialisierung, aber eine Sub Main ist erforderlich, damit die
    ' Anwendung beim Starten keine Form anzeigen muß.
End Sub
```

Anmeldeform:

Im Folgenden wird eine Form für den Anmeldedialog angelegt.



Eigenschaft DialogCanceled

Nun wird zum Form eine Eigenschaft hinzugefügt, die schreibgeschützt sein soll, da sie ausschließlich im Formmodul verwendet wird. Dies wird damit erreicht, daß keine der „Property Get“ Prozedur entsprechende „Property Let“ Prozedur definiert wird.

```
' Schreibgeschützte Eigenschaft, die anzeigt, ob der
' Dialog mit "Abbrechen" geschlossen wurde.
Property Get DialogCanceled() As Boolean
    DialogCanceled = blnDialogCanceled
End Property
```

Die Eigenschaft *DialogCanceled* liefert den Wert der als Private im Formmodul deklarierten Variable *blnDialogCanceled*. Auf diese Variable kann von außerhalb des Objektes nicht zugegriffen werden.

Die Variable *blnDialogCanceled* wird durch die Schaltfläche *Abbrechen* in der Ereignisprozedur *cmdCancel_Click* auf True gestellt:

```
Private Sub cmdCancel_Click()
    ' Wert der schreibgeschützten Eigenschaft festlegen, die dem Aufrufer
    ' anzeigt, ob der Dialog mit "Abbrechen" geschlossen wurde.
    blnDialogCanceled = True
    Me.Hide
End Sub
```

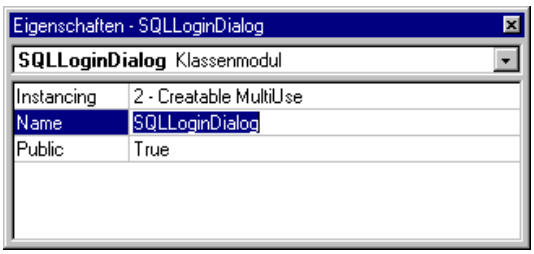
Mit der Schaltfläche OK wird das Form unsichtbar gemacht und nicht entladen, um weiterhin auf deren Inhalte zugreifen zu können.

```
Private Sub cmdOK_Click()
    ' Form nicht entladen, sondern nur ausblenden, damit der Inhalt
    ' des Textfeldes abgerufen werden kann.
    Me.Hide
End Sub
```

Erzeugen eines Klassenmoduls:

Der vorliegende OLE-Server soll eine Methode bereitstellen, die das Anmeldedialogfenster anzeigt und die Anmeldeinformationen an die aufrufende Anwendung zurück liefert.

1. Zuerst muß ein Klassenmodul dem Projekt zugefügt werden.
2. Dieses erhält den Namen *SQLLoginDialog*, der Klassenname für Erstellung neuer Anmelde-Objekte.
3. Weiters wird die Public-Eigenschaft auf *True* gesetzt, damit andere Anwendungen diese Klasse verwenden können.
4. Ebenso muß die Instancing-Eigenschaft gesetzt werden.



Die Instancing - Eigenschaft hat nur in Verbindung mit der Public-Eigenschaft eine Wirkung, welche die Klasse offenlegt.

Drei Möglichkeiten stehen zur Verfügung:

1. Not Creatable
2. Creatable MultiUse
3. Creatable SingleUse

Mit Creatable MultiUse ist es möglich, beliebig viele Objekte zu erstellen, unabhängig von der Anzahl der anfordernden Anwendungen.

Methode Show:

Nun wird die Methode implementiert, die den Anmeldedialog aufruft.

```
' Show ist die einzige für SQLLoginDialog offengelegte Methode.
Public Function Show(ByVal Caption As String, _
        ByVal Username As String, ByVal DBName As String) As String
    Dim frmNew As New frmSQLLoginDialog

    ' Die Form wird beim ersten Verweis auf ihre Eigenschaften erzeugt.
    With frmNew
        .Caption = Caption
        .txtDatabase.Text = DBName
        .txtUsername.Text = Username
        ' Nach dem Setzen der Eigenschaften Dialog anzeigen.
        .Show vbModal
        ' Wenn der Benutzer den Dialog abgebrochen hat,
        ' leere Zeichenfolge zurückgeben.
        If .DialogCanceled Then
            Show = ""
        Else
           ' Vollständige Anmeldezeichenfolge zusammensetzen
           ' und zurückgeben.
            Show = "ODBC;UID=" & .txtUsername.Text & _
                ";PWD=" & .txtPassword.Text & ";DATABASE=" & _
                .txtDatabase.Text & ";LoginTimeOut=30"
        End If
    End With
    Unload frmNew
End Function
```

Dieser Methode werden vom aufrufenden Programm drei Parameter übergeben:

1. Dialogüberschrift
2. Standard-Benutzername
3. Standard-Datenbankname

Die Funktion wird als Public deklariert und liefert die vom Benutzer eingegebenen Informationen zurück, andernfalls wird eine leere Zeichenfolge zurückgegeben.

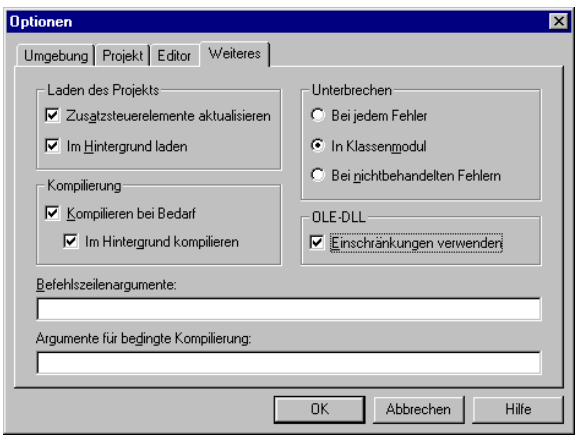
Zu Beginn der Funktion wird mit dem New-Operator eine Form-Objektvariable angelegt. Dieses Formobjekt wird beim ersten Verweis darauf erzeugt.

Am Ende der Funktion wird das Objekt mit Unload wieder zerstört.

Klassenmodule haben außerdem zwei Ereignisse, Initialize und Terminate, mit deren Hilfe Code beim Erzeugen und entfernen eines Objekts ausgeführt werden kann.

Ausführung als prozeßinterner Server

Um nun einen prozeßinternen OLE-Server (DLL) zu erstellen, muß in den Optionen im Register *Weiteres* das Kontrollkästchen "Einschränkungen verwenden" im Feld "OLE-DLL" aktiviert werden.



Anschließend wird im Menü *Datei* der Befehl *OLE-DLL erstellen* ausgewählt. Als Dateiname wird LOGINDLG.DLL eingetragen.

Zuletzt muß der OLE-Server in der Windows Registrierung eingetragen werden. Dies erfolgt bei der Compilierung der DLL automatisch.

Erstellen der Client-Applikation:

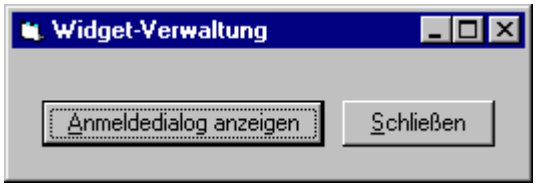
Der obige OLE-Server kann nun leicht getestet werden, in dem zum Beispiel zum Form-Load Ereignis die Showmethode des SQLLoginDialog-Objekts aufgerufen wird:

Erzeugen einer neuen Instanz

```
Sub Form_Load()
    Dim dlg As StdLoginDialogs.SQLLoginDialog
    Dim strResult As String

    ' Im Gegensatz zur Syntax 'Dim dlg As New ...'
    ' aus der Click-Ereignisprozedur der Befehlsschaltfläche
    ' wird hier das SQLLoginDialog-Objekt mit Hilfe des
    ' New-Operators und der Set-Anweisung erstellt.
    Set dlg = New StdLoginDialogs.SQLLoginDialog
    ' In diesem Fall existiert das Objekt bereits, wenn
    ' die folgende Codezeile ausgeführt wird.
    strResult = dlg.Show(Me.Caption, "JoeUser", "CorpWidgetBase")
    MsgBox "Ihre Anmeldezeichenfolge ist: " & strResult

    ' Objekt zerstören.
    Set dlg = Nothing

End Sub
```

Mit dem New-Operator in der Zeile `Set dlg = New StdLoginDialogs.SQLLoginDialog` wird eine neue Instanz des SQLLoginDialog-Objekts erzeugt. Da das Dialogform als modales Form geöffnet wird, wird die Form_Load Prozedur angehalten, bis der Anmeldevorgang abgeschlossen ist. Das Ergebnis wird in einem Meldungsfenster angezeigt, das Anmeldeobjekt zerstört und das Form der Client-Anwendung angezeigt.

Mit der Schaltfläche *Anmeldedialog anzeigen* kann wieder die Anmeldeinformationen mit dem Anmeldeobjekt abgefragt werden.



In diesem Fall wurde eine etwas andere Form gewählt, das Amelde-Objekt zu erzeugen:

```
Private Sub cmdLogin_Click()
    Dim dlg As New StdLoginDialogs.SQLLoginDialog
    Dim strResult As String

    ' Da die Variable 'dlg' mit dem Schlüsselwort
    ' New deklariert wurde, wird das SQLLoginDialog-Objekt erstellt,
    ' wenn die folgende Codezeile ausgeführt wird.
    strResult = dlg.Show(Me.Caption, "JoeUser", "CorpWidgetBase")
    MsgBox "Ihre Anmeldezeichenfolge ist: " & strResult

    ' Objekt zerstören.
    Set dlg = Nothing

End Sub
```

Hier wird eine neue Instanz des Objekts *SQLLoginDialog* erzeugt, wenn zum ersten Mal auf das Objekt Bezug genommen wird.

Am Ende der Prozedur wird das Objekt wieder mit einer Set- Anweisung zerstört.

Zuletzt muß ich noch erwähnen, daß die solcherart erstellten OLE-Server nur als 32 Bit - Komponente in Windows95 oder WinNT benützt werden können. □