Anhang 3

OOP Erweiterungen in VFP 7.0

Die Objektorientierung ist eines der mächtigsten Features von Visual FoxPro. VFP 7 enthält mehrere Verbesserungen in diesem Bereich, einschließlich neuer Eigenschaften und Methoden und auch einige neue Ereignisse.

Die großen Änderungen in VFP 3 waren die Umstellung von FoxPro von einer prozeduralen zu einer objektorientierten Sprache. Jede weitere Version fügte neue Klassen hinzu oder erweiterte die bereits bestehenden. VFP 7 enthält keine neuen Basisklassen, fügt den existierenden aber neue Features hinzu und verbessert so die Art, in der wir mit ihnen arbeiten.

Erstellen von Klassen

Die Erstellung von Klassen im Code (statt mit dem Klassendesigner) wird aus verschiedenen Gründen immer wichtiger.

Die Klasse Session (eingeführt mit VFP 6 Service Pack 3), die bei der Erstellung von Automation Servern extrem hilfreich ist, kann nicht im Klassendesigner abgeleitet werden. Zusätzlich können einige der neuen Features in VFP 7 wie die explizite Typangabe nur in Klassen eingesetzt werden, die mit Code erzeugt werden (also in nichtvisuellen Klassen). Lesen Sie dazu die Abschnitte „Using the session class in COM servers“ und „Storing VFP 7 COM classes“ in Kapitel 12 sowie „Building World-Class COM Servers in VFP 7“, um mehr über die Notwendigkeit der Klasse Session und die explizite Typangabe zu erfahren.

Wenn Sie in VFP 6 oder früheren Versionen eine Klassendefinition im Code hatten, mussten Sie sicherstellen, dass Sie mit SET PROCEDURE oder SET CLASSLIB auf die Klassenbibliothek für die Elternklasse verwiesen, bevor Sie die Klasse instanziierten. In VFP 7 ist das Schreiben von Klassendefinitionen einfacher, indem Sie der Anweisung DEFINE CLASS die optionale Klausel OF <classlib> hinzufügen. Um beispielsweise eine Klasse von _form aus den FoxPro Foundation Classes, abzuleiten können Sie die Klassendefinition wie folgt beginnen:

```
DEFINE CLASS fromMyForm AS _form OF HOME()+"FFC\_BASE.VCX"
```

DEFINE CLASS hat noch ein anderes Schlüsselwort: IMPLEMENTS. Vergleichen Sie Kapitel 13, „Implementing Interfaces“, für genauere Informationen über den Sinn und Gebrauch dieses Schlüsselworts.

Es gibt noch eine andere Änderung, die nur kodierte Klassenbibliotheken betrifft. Wenn Sie in früheren Versionen ein Objekt von einer PRG-basierten Klasse instanziierten und der Eigenschaft „Name“ nicht explizit im Abschnitt Properties der Klassendefinition ein Wert zugewiesen wurde, erhielt das Objekt den Namen der Klasse mit einer Nummer. Nummern werden sequentiell zugeordnet, jedes Mal erhöht, wenn Sie ein neues Objekt dieser Klasse instanziieren (also myObject1, myObject2, myObject3 usw.) Diese Lösung der Benennung der Objekte unterscheidet sich von der VCX-basierter Klassen und, noch wichtiger, die Entscheidung über die nächste verfügbare Nummer kann den Code entscheidend verlangsamen, wenn von einer einzelnen Klasse viele Objekte abgeleitet werden.

In VFP 7 hat sich dieses Verhalten geändert. Alle unbenannten Objekte werden dem Namen der Klasse zugeordnet, zu der sie gehören, statt einen eindeutigen Namen zu erhalten.

Die Klassen erforschen

Die Funktion AMEMBERS() gibt Ihnen die Möglichkeit, eine Klasse oder ein Objekt zu erforschen. In Abhängigkeit mit dem Wert, den Sie als dritten Parameter übergeben, wird ein Array mit einer Liste der Eigenschaften eines Objekts gefüllt, eine Liste aller Elemente eines Objekts (Eigenschaften, Methoden und eingebettete Objekte) erstellt oder nur eine Liste der eingebetteten Objekte. In VFP 6 und früher arbeitete AMEMBERS() nur mit den nativen Objekten von VFP.

VFP 7 erweitert AMEMBERS() auf unterschiedliche Weise. Zunächst einmal arbeitet die Funktion jetzt mit COM-Objekten wie auch mit nativen Objekten. Außerdem liefert sie zusätzliche Informationen über native Objekte. Das Übergeben des Werts 3 als dritten Parameter weist die Funktion an, ein vierspaltiges Array zurückzuliefern – die Inhalte der Spalten sehen Sie in Tabelle 1. Dieser Wert ist auch der Schlüssel zum Erforschen von COM-Objekten.

Tabelle 1. Was befindet sich in einem Objekt? Der Aufruf von AMEMBERS() mit einer 3 als drittem Parameter gibt ein vierspaltiges Array zurück.

Spalte	Inhalt
1	Name der Eigenschaft, des Ereignisses oder der Methode.
2	Typ des Eintrags. Für VFP-Objekte sind die möglichen Werte „Property“, „Event“ und „Method“. Bei COM-Objekten können die Werte „PropertyPut“, „PropertyGet“, „PropertyPutRef“ oder „Method“ sein.
3	Bei Eigenschaften von VFP-Objekten leer. Bei Methoden von VFP-Objekten die Parameterliste. Bei Eigenschaften und Methoden von COM-Objekten die Signatur des Elements auf der Grundlage der Parameterliste und der Rückgabewert.
4	Der Hilfestring des Eintrags.

Um eine Liste der Elemente des FileSystemObject des Windows Scripting Host können Sie den folgenden Code benutzen:

```
oWSH = CREATEOBJECT("Scripting.FileSystemObject")
nMemberCount = AMEMBERS( aWSHMembers, oWSH, 3 )
```

Hier ein Teil des Listings des erstellten Array. (Das Array hat 27 Zeilen).

Hier ein Teil des Listing des Array. Das Array hat 27 Zeilen.

```
(1,1) C  "BuildPath"
(1,2) C  "Method"
(1,3) C  "(Path as String, Name as String) as String"
(1,4) C  "Generate a path from an existing path and a name"
(2,1) C  "CopyFile"
(2,2) C  "Method"
(2,3) C  "(Source as String, Destination as String,
             [OverWriteFiles as Logical=.T.])"
(2,4) C  "Copy a file"
(3,1) C  "CopyFolder"
(3,2) C  "Method"
(3,3) C  "(Source as String, Destination as String,
             [OverWriteFiles as Logical=.T.])"
(3,4) C  "Copy a folder"
(4,1) C  "CreateFolder"
(4,2) C  "Method"
(4,3) C  "(Path as String) as IFolder"
(4,4) C  "Create a folder"
(5,1) C  "CreateTextFile"
(5,2) C  "Method"
(5,3) C  "(FileName as String, [Overwrite as Logical=.T.],
             [Unicode as Logical=.F.]) as ITextStream"
(5,4) C  "Create a file as a TextStream"
```

AMEMBERS() hat noch das vierte Parameter cFlags erhalten, mit dem Sie bei nativen Objekten angeben können, welche Elemente zurückgegeben werden sollen. In Tabelle 2 sehen Sie die möglichen Werte. Die Werte innerhalb der Filtergruppen sind exklusiv. Wenn Sie mehrere Werte für cFlag aneinanderhängen, werden diese mit OR verbunden, so dass das Parameter „HP" alle verborgenen und geschützten Eigenschaften, Ereignisse und Methoden auflistet. Die Übergabe von „GU" enthält alle Elemente, die entweder PUBLIC oder benutzerdefiniert sind. Achten Sie darauf, dass das Flag „C" keine Änderungen an Eigenschaften zurückgibt, die Arrays enthalten.

Tabelle 2. Auswahl von Elementen – AMEMBERS() neuer vierter Parameter gibt Ihnen die Möglichkeit, die zurückgegebene Liste der Elemente zu filtern.

Zeichen des Parameters	Filtergruppe	Bedeutung
P	Sichtbarkeit	Geschütze Eigenschaften, Methoden oder Ereignisse
H	Sichtbarkeit	Versteckte Eigenschaften, Methoden oder Ereignisse
G	Sichtbarkeit	Öffentliche Eigenschaften, Methoden oder Ereignisse
N	Herkunft	Systemeigene (interne) Eigenschaften, Ereignisse oder Methoden
U	Herkunft	Benutzerdefinierte (externe) Eigenschaften, Methoden oder Ereignisse
I	Vererbung	Benutzerdefinierte (externe) Eigenschaften, Methoden oder Ereignisse
B	Vererbung	Basiseigenschaften, -methoden oder –ereignisse
C	Änderung	Geänderte Eigenschaften
R	Schreibschutz	Schreibgeschützte Eigenschaften

Es gibt noch zwei spezielle Werte. Eines davon ist das „+", das im Parameter cFlags anzeigt, dass der Filter mit AND, nicht mit OR gebildet werden soll. Der Parameter „GU+" liefert nur Elemente, die sowohl als PUBLIC gekennzeichnet als auch benutzerdefiniert sind.

Der zweite spezielle Wert ist „#", mit dem das Ergebnisarray eine zusätzliche Spalte erhält. In dieser Spalte werden die Flags des Elements mit den Werten aus Tabelle 2 angezeigt.

Dieser Code erstellt eine Instanz der Klasse _MoverLists aus den FoxPro Foundation Classes und listet anschließend die geschützten Elemente der Klasse zusammen mit deren Flags auf:

```
oObject = NEWOBJECT( "_MoverLists",HOME()+"FFC\_CONTROLS" )
AMEMBERS( aMemberList, oObject, 3, "P#" )
LIST MEMORY LIKE aMemberList
AMEMBERLIST   Global  A
(1,1) C  "ADDTOPROJECT"
(1,2) C  "Method"
(1,3) C  ""
(1,4) C  "Dummy code for adding files to project."
(1,5) C  "CPUI"
(2,1) C  "NINSTANCES_ACCESS"
(2,2) C  "Method"
(2,3) C  ""
(2,4) C  "Access method for nInstances property."
(2,5) C  "CPUI"
(3,1) C  "NINSTANCES_ASSIGN"
(3,2) C  "Method"
(3,3) C  "vNewVal"
(3,4) C  "Assign method for nInstances property."
(3,5) C  "CPUI"
(4,1) C  "NOBJECTREFCOUNT_ACCESS"
(4,2) C  "Method"
(4,3) C  ""
(4,4) C  "Access method for nObjectRefCount property."
(4,5) C  "CPUI"
(5,1) C  "NOBJECTREFCOUNT_ASSIGN"
(5,2) C  "Method"
(5,3) C  "m.vNewVal"
(5,4) C  "Assign method for nObjectRefCount property."
(5,5) C  "CPUI"
```

Dieses Beispiel enthält alle Elemente der Klasse, aber auch alle Flags (hier nur ein Teil der Ausgabe).

```
AMEMBERS(aMemberList, oObject, 3, "GPH#")
LIST MEMORY LIKE aMemberList

AMEMBERLIST   Global  A
(1,1) C  "ACTIVECONTROL"
(1,2) C  "Property"
(1,3) C  ""

(1,4) C  "Verweist auf das aktive Steuerelement auf einem Objekt."
(1,5) C  "GRNI"
(2,1) C  "ADDOBJECT"
(2,2) C  "Method"
(2,3) C  "cName, cClass"

(2,4) C  "Fügt einem Container-Objekt während der " + ;
         "Laufzeit ein Objekt hinzu."
(2,5) C  "GRNI"
(3,1) C  "ADDPROPERTY"
```

```
(3,2) C  "Method"
(3,3) C  "cPropertyName,eNewValue"
(3,4) C  "Fügt einem Objekt eine neue Eigenschaft hinzu."
(3,5) C  "GRNI"
(4,1) C  "ADDTOPROJECT"
(4,2) C  "Method"
(4,3) C  ""
(4,4) C  "Dummy code for adding files to project."
(4,5) C  "CPUI"
(5,1) C  "AOBJECTREFS"
(5,2) C  "Property"
(5,3) C  ""
(5,4) C  "Array of object references properties."
(5,5) C  "GUI"
(6,1) C  "BACKCOLOR"
(6,2) C  "Property"
(6,3) C  ""
(6,4) C  "Gibt die für Text und Grafik eines Objekts " + ;
         "zu verwendende Hintergrundfarbe an."
(6,5) C  "GNI"
```

Im Downloadbereich von www.hentzenwerke.com finden Sie das Formular ShowAMembers.SCX, mit dessen Hilfe Sie mit AMEMBERS() experimentieren können.

Neue und erweiterte PEMs

Die Klassen von VFP 7 enthalten viele Änderungen an den Eigenschaften, Ereignissen und Methoden (PEMs). Es gibt brandneue PEMs, PEMs die zusätzlichen Objekten hinzugefügt wurden und bei einigen Eigenschaften auch neue Werte. Dieser Abschnitt behandelt zunächst die Änderungen, die mehrere Klassen betreffen und später solche, die nur in einer Klasse vorkommen.

Änderungen an mehreren Klassen

Viele der Änderungen an den PEMs betreffen mehr als eine Klasse.

Die Ereignisse MouseEnter und MouseLeave

Das Bereitstellen neuer Ereignisse ist für Visual FoxPro nicht üblich, aber in VFP 7 verfügen alle sichtbaren Kontrollelemente über die neuen Ereignisse MouseEnter und MouseLeave. Diese Ereignisse geben Ihnen die Möglichkeit Aktionen auszuführen, wenn die Maus über das Kontrollelement gezogen wird. Die neuen Ereignisse erwarten die gleichen Parameter wie MouseMove: ist eine Maustaste gedrückt, ist dabei auch die Strg-, Shift- oder Alt-Taste betätigt sowie die aktuelle Position der Maus. Sie können mit Hilfe dieser Methoden auf der Basis der Mausposition unterschiedliche Optionen aktivieren

oder deaktivieren. Außerdem lässt sich damit bequem die neue Eigenschaft VisualEffect der Schaltflächen manipulieren, die im Abschnitt „Änderungen an Kontrollelementen“ weiter hinten in diesem Kapitel beschrieben wird.

Die Collection Objects

Jede Containerklasse in VFP bietet über eine Collection, die nach dem Typ der Elemente benannt ist, die Möglichkeit des Zugriffs auf ihre Elemente. So hat beispielsweise PageFrame eine Collection Pages und Grid die Collection Columns. Einige Container bieten aber noch einen generischeren Zugriff auf deren Elemente – die Collection Objects. Objects ist eine COM-Collection, die auch die Eigenschaften Count und Item enthält.

In VFP 7 verfügen alle Containerklassen über eine Collection Objects, auch diejenigen, die sie in VFP 6 noch nicht hatten (CommandGroup, DataEnvironment, PageFrame und OptionGroup).

Die Eigenschaft SpecialEffect

Viele Kontrollelemente haben für die Eigenschaft Spezialeffekt die neue Einstellung „Hot Tracking“ (2), mit denen sie flach angezeigt werden, es sei denn, der Mauscursor befindet sich über ihnen. In diesem Fall wird das Element entweder erhöht oder gedrückt angezeigt. Bei Kontrollkästchen und Optionsfeldern funktioniert das Hot Tracking nur, wenn als Style Graphical eingestellt ist. Wie auch die Änderungen an den Menüs, die in Kapitel 3, „New and Better Tools“, vereinfacht auch dies die Entwicklung von Anwendungen, die der neuen flachen Gestaltung der Benutzeroberfläche folgen.

Im Downloadbereich von www.hentzenwerke.com finden Sie das Formular UIChanges.SCX, das die Ereignisse MouseEnter und MouseLeave sowie die Hot Tracking-Einstellung für SpecialEffect und die neue Eigenschaft VisualEffect der Schaltflächen demonstriert.

Die Methode WriteMethod()

Die Methode WriteMethod() ermöglicht es Ihnen, programmatisch einer Methode Code hinzuzufügen. In VFP 7 hat sie einen neuen dritten Parameter, mit dem Sie „on the fly“ Methoden erstellen können. Wenn die im ersten Parameter angegebene Methode nicht existiert und der Parameter lCreateMethod auf True steht, wird die Methode erstellt. Dieses Vorgehen funktioniert nur zu Designzeit und das Formular oder die Klasse muss nach dem Hinzufügen der Methode gespeichert werden. Diese Einschränkung ist sinnvoll, da WriteMethod() für den Einsatz in Buildern gedacht ist.

Änderungen an Formularen und Toolbars

Einige Änderungen betreffen Formulare und/oder Toolbars. Diese Änderungen helfen beim Erstellen von Anwendungen, die dem Windowsstandard entsprechen.

Die Eigenschaft hWnd

Jedes Fenster hat in Windows ein „Handle“, mit dem es identifiziert wird. In VFP 7 kann auf das Fensterhandle von Formularen und Toolbars über die neue Eigenschaft hWnd direkt zugegriffen werden. In älteren Versionen mussten Sie einige Funktionsaufrufe durchführen, um an diese Information zu kommen. Abbildung 1 zeigt ein Formular, das seine hWnd an eine API-Funktion übergibt und sich selbst in einen Kreis verwandelt. Den Code für dieses Formular (wie er aus dem Klassenkatalog exportiert wurde) finden Sie in Listing 1.

Das Formular in Abbildung 1 finden Sie unter dem Namen MkCircle.SCX im Downloadbereich von www.hentzenwerke.com.

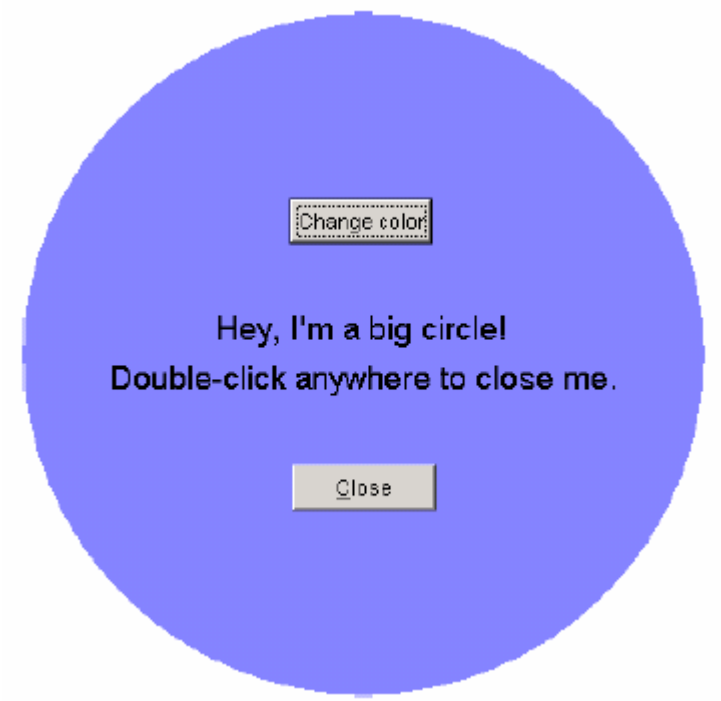

Abbildung 1. Das Handle des Formulars. Dieses Formular übergibt sein Window-Handle an die API-Funktion SetWindowRgn, um seine Form in einen Kreis umzuwandeln. Geschlossen wird das Formular durch einen Doppelklick auf eine beliebige Stelle.

Listing 1. Benutzen des Handles eines Formulars – dieser Code erstellt das Formular aus Abbildung 1.

```
PUBLIC ofrmcircle

ofrmcircle=NEWOBJECT("frmcircle")
ofrmcircle.Show
RETURN

DEFINE CLASS frmcircle AS Form

      Height = 400
      Width = 400
      DoCreate = .T.
      AutoCenter = .T.
      BorderStyle = 0
      Caption = "Form"
      Movable = .F.
      TitleBar = 0
      WindowType = 1
      BackColor = RGB(128,128,255)
      Name = "frmCircle"

      ADD OBJECT lblcircle AS label WITH ;
         AutoSize = .T., ;
         FontSize = 14, ;
         BackStyle = 0, ;
         Caption = "Hey, I'm a big circle!" ;
         Height = 25, ;
         Left = 113, ;
         Top = 173, ;
         Width = 173, ;
         Name = "lblCircle"

      ADD OBJECT lbldblclick AS label WITH ;
         AutoSize = .T., ;
         FontSize = 14, ;
         BackStyle = 0, ;
         Caption = "Double-click anywhere to close me" ;
         Height = 25, ;
         Left = 50, ;
         Top = 202, ;
         Width = 300, ;
         Name = "lblDblClick"

      ADD OBJECT cmdcolor AS commandbutton WITH ;
         Top = 108, ;
         Width = 156, ;
         Height = 27, ;
         Width = 84, ;
         Caption = "Change color" ;
         Name = "cmdColor"

      ADD OBJECT cmdclose AS commandbutton WITH ;
         Top = 264, ;
         Width = 158, ;
         Height = 27, ;
```

```
      Width = 84, ;
      Caption = "\<Close" ;
      Name = "cmdClose"


   PROCEDURE DblClick
      ThisForm.Release()
   ENDPROC

   PROCEDURE Init
      LOCAL nhWnd, nWidth, nHeight, nRegion

      DECLARE INTEGER DreateEllipticRgn IN gdi32 ;
         INTEGER X1 , INTEGER Y1 , INTEGER X2 , INTEGER Y2
      DECLARE INTEGER SetWindwRgn IN user32 ;
         INTEGER HWND, INTEGER hRgn , INTEGER bRedraw

      nhWnd = This.HWnd
      nWidth = This.Width / 1 &&& change ratio
      nHeight = This.Height / 1 &&& change ratio
      * Call API to convert an otherwise regular
      * form into a circular one.
      nRegion = CreateEllipticRgn(0, 0, nWidth, nHeight)
      SetWindowRgn(nhWnd, nRegion, 1)
   ENDPROC

   PROCEDURE lblcircle.DblClick
      ThisForm.DblClick()
   ENDPROC

   PROCEDURE lbldblclick.DblClick
      ThisForm.DblClick()
   ENDPROC

   PROCEDURE cmdcolor.Click
      nColor = GETCOLOR(ThisForm.BackColor)
      IF nColor <> -1
         ThisForm.BackColor=nColor
      ENDIF
   ENDPROC

   PROCEDURE cmdclose.Click
      ThisForm.Release()
   ENDPROC

ENDDEFINE
```

Auch das Hauptfenster von VFP, auf das über die Systemvariable _VFP zugegriffen wird, sowie dessen Clientbereich, _Screen, verfügen über die Eigenschaft hWnd. Beide haben jeweils einen anderen Wert dafür. Für Informationen über weitere Unterschiede zwischen _VFP und _Screen in VFP 7 vergleichen Sie Kapitel 8, „Resource Management“.

Die Eigenschaft ShowInTaskBar

Formulare verfügen über die neue Eigenschaft ShowInTaskBar, mit der festgelegt wird, ob ein Formularobjekt der obersten Ebene in der Taskleiste von Windows angezeigt wird. Als Vorgabewert steht die Eigenschaft auf True und alle Formulare werden unmittelbar in der Taskleiste angezeigt. Stellen Sie die Eigenschaft auf False, erscheint das Formular nicht in der Taskleiste und beim Minimieren wird ein Symbol auf dem Desktop angezeigt.

Die Eigenschaft Style

Die Eigenschaft Style wurden den Zwischenraum-Objekten der Toolbars hinzugefügt, also den Objekten, mit denen die einzelnen Schaltflächen der Toolbar voneinander getrennt werden. Als Vorgabewert ist Style auf 0-Normal gesetzt. Sie können Style aber auch auf 1-Vertikal einstellen. Diese neue Einstellung ermöglicht Ihnen die Erstellung von Toolbars, die zwischen den einzelnen Gruppen der Schaltflächen eine vertikale Linie anzeigen. Beachten Sie, dass diese Linien nur zur Laufzeit sichtbar sind, nicht während des Designs.

Abbildung 2 zeigt Ihnen eine Standard-Toolbar, in der Style auf 0 gesetzt ist. Abbildung 3 zeigt die gleiche Toolbar mit Style auf 1.

Abbildung 2. Nicht sichtbare Zwischenraum-Objekte - Die Eigenschaft Style für die Zwischenräume ist auf 0-Normal gesetzt.

Abbildung 3. Sichtbare Zwischenräume - in dieser Version der Toolbar ist die Eigenschaft Style der Zwischenräume auf 1 - vertikale Linie gesetzt, so dass sie zur Laufzeit sichtbar werden.

Die Toolbar-Klasse aus Abbildung 2 und 3 finden Sie in der Klassenbibliothek Chapter6.VCX im Downloadbereich von www.hentzenwerke.com.

Um die Toolbarklasse aus dem Beispiel zu testen, instanziieren Sie sie in eine Variable und setzen Visible auf True. Wenn Sie dem Objekt False oder nichts übergeben, erhalten Sie die Toolbar in Abbildung 2 (nicht sichtbar).

```
oToolbar = NEWOBJECT("tbrStandard", "chapter6.vcx")
oToolbar.Visible = .T.
```

Um die Version mit den vertikalen Linien zu sehen, übergeben Sie der Methode Init() True:

```
oToolbar = NEWOBJECT("tbrStandard", "chapter6.vcx" "", .T.)
oToolbar.Visible = .T.
```

Änderungen an Kontrollelementen

Zum Schluss noch einige Änderungen, die individuelle Kontrollelemente betreffen. Wie anderes auch wurden diese Änderungen bereits seit längerem von den Entwicklern eingefordert.

Schaltflächen

Schaltflächen besitzen die neue Eigenschaft VisualEffect, mit der Sie die Schaltfläche zur Laufzeit gedrückt oder erhaben anzeigen können. Während des Designs ist VisualEffect schreibgeschützt. VisualEffect kann drei Werte beinhalten: 0 - Keine, 1 – Herausstehend und 2 - Gedrückt.

Wenn Sie VisualEffect in MouseEnter auf 1 – Keine stellen und in MouseLeave den Normalzustand von 0 – keine wieder herstellen, erhalten Sie den gleichen Effekt, als wenn Sie SpecialEffect auf die neue Einstellung 2-Hot Tracking stellen würden.

Die Schaltfläche Close auf dem Formular UIChanges.SCX, das Sie im Downloadbereich auf www.hentzenwerke.com finden, demonstriert die Eigenschaft VisualEffect.

Grids

Seit dem Erscheinen von Visual FoxPro 3.0 haben sich die Entwickler gefragt, weshalb die Ereignisse BeforeRowColChange und AfterRowColChange nicht in die getrennten Ereignisse BeforeRowChange, BeforeColChange, AfterRowChange und AfterColChange aufgeteilt wurden. So weit geht auch VFP 7 nicht. Die neue Eigenschaft RowColChange macht es aber einfach herauszufinden, was die beiden Ereignisse ausgelöst hat: die Zeile (1), die Spalte (2), beides (3) oder nichts von beiden (0). Den Wert 0 erhalten Sie, wenn das Grid zum ersten Mal angezeigt oder neu gezeichnet wird. Ihr Code kann diesen Wert prüfen und entsprechend darauf reagieren.

Die neue Eigenschaft HighlightRowLineWidth gibt die Breite der horizontalen Rahmenlinie an. Diese Einstellung wird nur genutzt, wenn HighlightRow auf True (dem Vorgabewert) steht.

Header

Dem Headerobjekt der Grids wurde die neue Eigenschaft WordWrap hinzugefügt, so dass die Spalten auch mehrzeilige Überschriften erhalten können. Dies ist wahrscheinlich die auffälligste Neuerung im Hinblick auf die Header.

Änderungen an den ProjectHooks

Als die ProjectHooks mit VFP 6 eingeführt wurden, fanden die Entwickler sofort Einsatzmöglichkeiten dafür. Aber es fehlten noch einige Features. VFP 7 stopft diese Löcher mit drei neuen Ereignissen: QueryNewFile, Activate und Deactivate.

QueryNewFile wird ausgelöst, wenn Sie beginnen, einem Projekt eine neue Datei hinzuzufügen, wenn Sie also im Projektmanager auf die Schaltfläche „Neu“ klicken. Bislang wurde kein Ereignis ausgelöst, wenn Sie einem Projekt eine neue Datei einfügen.

Die Ereignisse Activate und Deactivate verhalten sich im Objekt ProjectHook wie die in anderen Klassen – sie werden ausgelöst, wenn das Objekt aktiv wird bzw. deaktiviert wird. Im Fall der ProjectHooks ist dies der Fall, wenn das Projekt, zu dem der ProjectHook gehört, aktiviert oder deaktiviert wird. Auf diese Weise haben Sie eine neue Möglichkeit, die Umgebung von VFP zu modifizieren, wenn Sie zwischen mehreren Projekten hin- und herschalten, z. B. den VFP-Path, Feldzuordnungen oder andere projektspezifische Einstellungen.

Zusammenfassung

VFPs Änderungen am OOP-Teil der Sprache erleichtern die Erstellung von Klassen und die Arbeit mit ihnen. Außerdem erhalten Formulare und Kontrollelemente mehr vom Verhaltens, das Entwickler und Anwender wünschen und erwarten. Außerdem enthält VFP 7 die Werkzeuge, die für die Erstellung von Benutzeroberflächen im Stil von Windows 2000 erforderlich sind.