Weitere Tipps und Tricks zu VFX 9.5

Rainer Becker

Seit der Übernahme des Frameworks Visual Extend durch die dFPUG c/o ISYS GmbH erstelle ich mehr und mehr Anwendungen mit VFX. Mittlerweile sogar fast ausschließlich, da sich diese Vorgehensweise in vielerlei Hinsicht bewährt hat. Unter anderem durch schnellere Abwicklung und bessere Kalkulierbarkeit von Projekten sowie regelmäßigere Lieferung von Updates im Rahmen von Wartungsverträgen mit niedrigerem Aufwand.

Im Rahmen der praktischen Arbeit sammeln sich eine Vielzahl von kleinen Hilfsroutinen und nützlichen Funktionen. Im gedruckten Handbuch zu VFX 8.0 hatten wir dazu eine Tipps & Tricks-Sammlung von Stefan Zehner veröffentlicht. Das gedruckte Handbuch zu VFX 9.0 wurde aber wesentlich umfangreicher, so das für Anhänge leider kein Platz mehr war. In einem separaten Beitrag hatte ich deshalb eine separate Tipps & Tricks-Sammlung zu VFX 9.0 und früher veröffentlicht.

Nachfolgend nun anlässlich der neuen Version Visual Extend 9.5 eine Fortsetzung der Tipps- und Tricks-Sammlung, die allerdings nicht nur für Framework-Anwender interessant ist, sondern meist auch allgemein in Visual FoxPro-Anwendungen verwendet werden kann.

Arbeiten mit Terminalserver

In der Visual FoxPro Hilfe findet man eine kurze Erläuterung der SYS-Funktion SYS(602), die den Bildschirmaufbau unter Terminalservern beschleunigen kann. Dort wird darauf hingewiesen, dass man das Vorhandensein einer Terminalserver-Instanz auf dem aktuellen Rechner über OS(10) feststellen kann. Leider sagt einem diese Funktion nicht, ob die aktuelle Anwendung auch tatsächlich in einer Terminalserver-Session läuft. Deshalb empfehlen wir die Verwendung der Windows-Funktion GetSystemMetrics im Hauptprogramm (vfxmain.prg) wie folgt:

```
DECLARE INTEGER GetSystemMetrics IN WIN32API INTEGER nIndex
IF GetSystemMetrics(4096) != 0
	=SYS(602,1)
	*-- ggf. hier: SYS(3050,1 bzw 2, <reduzierter Speicherbedarf)
	*-- für weniger Speicherbedarf der Anwendung unter Terminalserver
ENDIF
```

Auf dem Terminalserver gibt es darüber hinaus noch eine weitere Problematik. Anwender können sich von einer Session abmelden, ohne die laufende Anwendung zu beenden. Diese läuft dann sprichwörtlich ewig weiter und verhindert exklusiven Zugriff auf Tabellen auch zu nachtschlafender Zeit. Da es natürlich unhöflich ist, so einen Task dann einfach zu beenden, sollte die Anwendung selbst dafür Sorge tragen - und zwar mit Hilfe eines Timers in der Anwendung, der diese z.B. nach einer Stunde Leerlauf zwangsbeendet.

Ein Timer ist leicht an das globale Programmobjekt addiert. Aber so einen Timer muss man natürlich immer wieder zurücksetzen, wenn der Anwender aktiv ist. Dafür bietet sich die VFX-Funktion oneventhookhandler an, die ja fortwährend durchlaufen wird. Damit mir aber nicht meine Entwicklungsumgebung beendet wird, wollte ich dabei den _VFP.StartMode abfragen. Das führt bei einem Aufruf am Anfang des Eventhookhandlers aber plötzlich zu einem OLE-Fehler bei diversen Anwendern! Deshalb die Empfehlung, das in ein seltener aufgerufenes Event zu verschieben und die Version-Funktion zu verwenden wie folgt:

```
CASE tcEvent=="ONRECORDMOVE"
IF TYPE("goprogram")="O" AND NOT ISNULL(goprogram) AND VERSION(2)=0
       *_vfp.StartMode != 0
       =goprogram.timeout()
ENDIF
```

Die entsprechende Methode muss man sich natürlich noch anlegen, die den Timer jeweils zurücksetzt...

Dateiabfrage abschalten

Wenn Visual FoxPro in einem SQL-Statement eine benötigte Tabelle nicht findet, erscheint ein Dateiauswahldialog, in welchem der Anwender wahllos irgendeine Tabelle auswählen oder über einen Button sogar versehentlich löschen kann. Von diesem Dialog fühlen sich nicht nur die meisten Anwender schlicht überfordert, sondern auch Administratoren halten so einen Dialog z.B. auf einem Terminalserver für eher unglücklich.

Zum Glück gibt es dafür die neue SET-Funktion SET TABLEPROMPT OFF. Statt dem Dialog erhält man eine handfeste Fehlermeldung und kann die Anwendung leidlich sauber beenden, statt in eine völlig unberechenbare Situation einer vom Anwender willkürlich ausgewählten Tabelle mit einer Vielzahl von möglichen Folgefehlern zu stolpern.

Im englischen Hilfstext wird es zwar nicht klar gesagt, aber SET TABLEPROMPT ist scoped, d.h. dieser Befehl muss in jeder Datasession neu gesetzt werden. Dies kann man in VFX vor 9.5 in der Funktion formsetup in dem Programm applfunc.prg tun. Ab VFX 9.5 gibt es dafür ganz einfach die Eigenschaft lTableprompt am Applikationsobjekt.

Beschleunigter Maskenaufbau

Manchmal dauert der Aufbau eines Formulars im Netzwerk einfach zu lange, weil man vielleicht sehr viele Tabellen öffnen und für diverse Grids positionieren muss. Dabei kommt einem dann der Screen-Refresh heftig in den Weg, da dieser beim Aufbau der Maske möglicherweise mehrfach aufgerufen wird. Marcia Akins und Andy Kramek haben mir dafür nachfolgende Funktion zur Verfügung gestellt, die ich gerne kurz vorstellen möchte.

Statt dem möglicherweise vielfach verschachtelten Aufruf der Lockscreen-Funktion von Visual FoxPro kann man die Windows-Funktion LockWindowUpdate verwenden, um ein Screen-Refresh ganz abzuschalten. Die Funktion muß man in seinem Hauptprogramm (vfxmain.prg) einmal wie folgt deklarieren:

```
DECLARE INTEGER LockWindowUpdate IN Win32API INTEGER nHandle
```

In das Init des jeweiligen Formulars kommt der folgende Aufruf:

```
Thisform.ReallyLockScreen( .T. )
```

Und in das Activate des jeweiligen Formulars kommen folgende Zeilen:

```
IF NOT EMPTY( DODEFAULT())
    Thisform.ReallyLockScreen( .F. )
ENDIF
```

Und jetzt müssen wir nur noch die Methode ReallyLockScreen in unserer Formularbasisklasse (cform in vfxobj.vcx oder cdataform in vfxform.vcx) anlegen und mit folgendem Inhalt füllen:

```
LPARAMETERS tlLock
       LOCAL lnHWnd
       *** Now set the Handle to lock according to the parameter
       lnHWnd = IIF( tlLock, ThisForm.HWnd, 0 )
       *** And call the function
       =LockWindowUpdate(lnHWnd)
RETURN
```

Das lässt sich natürlich in jedes Formular oder jede Formularbasisklasse einbauen und ist nicht VFX-spezifisch. Aber es funktioniert sehr gut, auch wenn es sich nur für wirklich aufwändige Formulare lohnt.

Kleiner Excel-Export-Button

Immer wieder kommt es vor, dass ein Anwender einem einmal eine Tabelle schicken soll und dabei mit den Endungen durcheinander kommt, oder dass ein Anwender Daten unbedingt in Excel haben möchte, aber mit der Bedienung der Anwendung nicht hinreichend klar kommt. Für diese Anwender bzw. diese Fälle platziere ich gerne auf die Schnelle einen kleinen Excel-Button auf dem Formular mit folgendem Code im Click-Event:

```
LPARAMETERS tcAlias
LOCAL lnRecno, lnSelect, lcAlias
lnSelect = SELECT()
lcAlias = EVL( tcAlias, thisform.cworkalias )
SELECT (lcAlias)
lnRecno = RECNO()
_vfp.datatoclip( lcAlias, , 3)
=ShellExecute(0,"Open","EXCEL.EXE","","",1)
WAIT WINDOW "Records copied to clipboard and Excel opened, insert data with
CTRL+V." NOWAIT
IF lnRecno > 0 AND lnRecno <= RECCOUNT()
       GOTO (lnRecno)
ELSE
       GO TOP
ENDIF
SELECT (lnSelect)
```

Einmal draufklicken und im bereits gestarteten Excel die Daten einfügen, fertig. Den Text für das WAIT WINDOWS muss man noch anpassen oder dafür die weiter unten erwähnte Funktion GETSOUND verwenden. Die Funktion ShellExecute muss natürlich im Hauptprogramm definiert worden sein. Falls dies nicht der Fall sein sollte, folgende Zeilen einkopieren für die sehr praktische Funktion:

```
DECLARE INTEGER ShellExecute IN SHELL32.DLL ;
       INTEGER nWinHandle, STRING cOperation, STRING cFileName, ;
       STRING cParameters, STRING cDirectory, INTEGER nShowWindow
```

Damit der Button auch schön aussieht und sich im Resizer richtig verhält, stellen wir schnell noch ein paar Eigenschaften wie folgt ein:

```
Caption = Excel
Comment = <NORESIZE>
Height = 23
Picture = bitmap\xls.bmp
PictureMargin = 2
PicturePosition = 1
PictureSpacing = 0
Width = 60
```

In der Refresh-Methode müssen wir nur noch dafür sorgen, dass der Button auch nur in sinnvollen Situationen enabled ist (also nicht beim Bearbeiten oder wenn gar keine Daten vorhanden sind):

```
=DODEFAULT()
WITH THISFORM
       THIS.Enabled = .nFormStatus=0 AND ;
NOT .lEmpty AND NOT EMPTY( .cworkalias )
ENDWITH
```

Fertig ist der kleine Excel-Export-Button. Einfach auf das Formular kleben, am besten oben rechts neben die integrierte Toolbar, falls vorhanden. Und sofern man nicht die Haupttabelle exportieren möchte, sondern eine andere Tabelle kann man in das Click-Event des Buttons auf der Form einfach DODEFAULT("<Aliasname>") schreiben...

Sounds in Anwendungen

Windows macht immer so hübsche Töne bei bestimmten Ereignissen. Die wollte ich in meiner Anwendung auch unbedingt haben. Also ein Unterverzeichnis Sounds angelegt und die entsprechenden WAV-Dateien von Windows flugs kopiert. Und dann die nachfolgende Funktion GETSOUND in applfunc.prg eingebaut:

```
LPARAMETERS tcCase, tcMessage, tcTitle
LOCAL lcReturn, lcSetBell
tcCase = EVL( tcCase, "" )    && gewünschter Sound
tcMessage = EVL( tcMessage, "")  && optional: anzuzeigender Text
tcTitle = EVL( tcTitle, "")      && optional: Titel für Messagebo
lcSetBell = SET("BELL")
DO CASE
       CASE ".wav" $ tcCase
          * wav-filename already provided...
          lcReturn = tcCase
       CASE tcCase = "login"
          lcReturn = "Windows XP-Anmeldesound.wav"
       CASE tcCase = "logout"
          lcReturn = "Windows XP-Abmeldesound.wav"
       CASE tcCase = "warning"
          lcReturn = "Windows XP-Ping.wav"
       CASE tcCase = "error"
          lcReturn = "Windows XP-Fehler.wav"
       CASE tcCase = "critical"
          lcReturn = "Windows XP-kritischer Fehler.wav"
       CASE tcCase = "message"
          lcReturn = "Windows XP-Hinweis.wav"
       OTHERWISE
          lcReturn = "Windows XP-Hinweis.wav"
ENDCASE
```

Die Funktion EVL stellt einem netterweise leere Strings bereit, wenn der Parameter gar nicht übergeben wurde. Die Sounds sind in diesem Beispiel natürlich hardcodiert, was man auch generisch über eine Steuertabelle lösen könnte. Alternativ könnte man natürlich die entsprechende Sound-Belegung des aktuellen Windows-Systems abfragen. Aber für eine erste einfache Implementation ist der obige Code völlig ausreichend. Und wahlweise kann man ein anderes .wav-File übergeben, das wir natürlich noch auf Vorhandensein prüfen müssen:

```
IF NOT EMPTY( lcReturn ) AND NOT FILE( lcReturn )
       lcReturn = "sounds\" + lcReturn
       IF NOT FILE( lcReturn )
          lcReturn = ""
       ENDIF
ENDIF
```

Und dann brauchen wir es nur noch abspielen. Dabei kann man über eine globale Varialbe gs_usesound den Sound auch abschalten. Einfach ein Feld usesound in der vfxsys.dbf anlegen...

```
IF NOT EMPTY( lcReturn )
       SET BELL ON
       SET BELL TO (lcReturn)
       IF TYPE("m.gs_usesound")="U" OR m.gs_usesound
          ?? CHR(7)
       ENDIF
       SET BELL TO
       IF lcSetBell = "OFF"
          SET BELL OFF
       ENDIF
ENDIF
```

Sofern ein Messagetext übergeben wurde, wird der noch als WAIT WINDOW angezeigt. Gibt es zusätzlich einen Titel für eine Messagebox, wird natürlich stattdessen eine Messagebox angezeigt wie folgt:

```
IF NOT EMPTY( tcMessage )
       tcMessage = LEFT( tcMessage, 250)
       IF EMPTY( tcTitle )
          WAIT WIND tcMessage NOWAIT
       ELSE
          =MESSAGEBOX( tcMessage, 16, tcTitle )
       ENDIF
ENDIF
```

Der eigentliche Aufwand ist nunmehr natürlich, die diversen Nachrichten-Anzeigen in seinem Programm herauszusuchen und um den entsprechenden Sound-Parameter zu erweitern...

SET-Einstellungen und SYS-Funktionen

Es gibt eine Vielzahl von SET-Einstellungen und SYS-Funktionen in Visual FoxPro und was wäre eine neue Version ohne weitere SETs und SYSs! Beim Update übersieht man dabei gerne die eine oder andere neue Möglichkeit, insbesondere wenn man beim Update eine oder gar mehrere Versionen von Visual FoxPro überspringt. Deshalb hier ein paar Hinweise auf von VFX nicht explizit ohnehin gesetzte Einstellungen und SYS-Aufrufe, die Sie in Ihrem Hauptprogramm (vfxmain.prg) nachtragen können:

In Visual FoxPro 8.0 funktionierte SET TABLEVALIDATE sehr gut. Aber mit Visual FoxPro haben einige Entwickler von fehlerhaften Fehlermeldungen berichtet. Setzen Sie ggf. SET TABLEVALIDATE TO 0, sofern Sie diese Funktionalität nicht explizit benötigen...

Manche Comboboxen enthalten arg viele Einträge und werden Bildschirmfüllen geöffnet. Mit Visual FoxPro 9.0 ist eine einfache Limitierung der Anzahl der angezeigten Einträge in einer Combobox mit der Funktion SYS(2910, <Anzahl>) möglich. Ich setze das in meinen Anwendungen z.B. auf 15.

Nicht neu ist SYS(2450,1), damit erst die Anwendung und dann der Pfad nach Komponenten durchsucht wird. Ebenfalls nicht neu ist SET OLEOBJECT OFF, damit bei der Suche nach Objekten nicht erst die ganze Registry durchsucht wird.

Statt mühselig die verschiedene Themes-Eigenschaften umzuschalten, kann man auch einfach =SYS(2700,0) aufrufen, um den XP Themes-Support in der Anwendung pauschal abzuschalten...

Eine Checksumme für einen Datensatz kann man ganz einfach mit SYS(2017) für vor Änderungen zu schützende Tabellen (z.B. Tabellen mit Passwörtern wie VFXUSR) ermitteln.

Und noch ein letzter Tipp: Wussten Sie schon, dass man in AutoComplete-Auswahllisten durch das Drücken der Entfernen-Taste bei geöffneter Auswahlliste unerwünschte Einträge einfach wieder entfernen kann?