Active Desktop Benutzeroberfläche für VFX

- Erweiterung -

Stefan Zehner
Zehner Software & Beratung

Active Desktop Singleklick-Benutzeroberfläche

Visual Extend 7.0

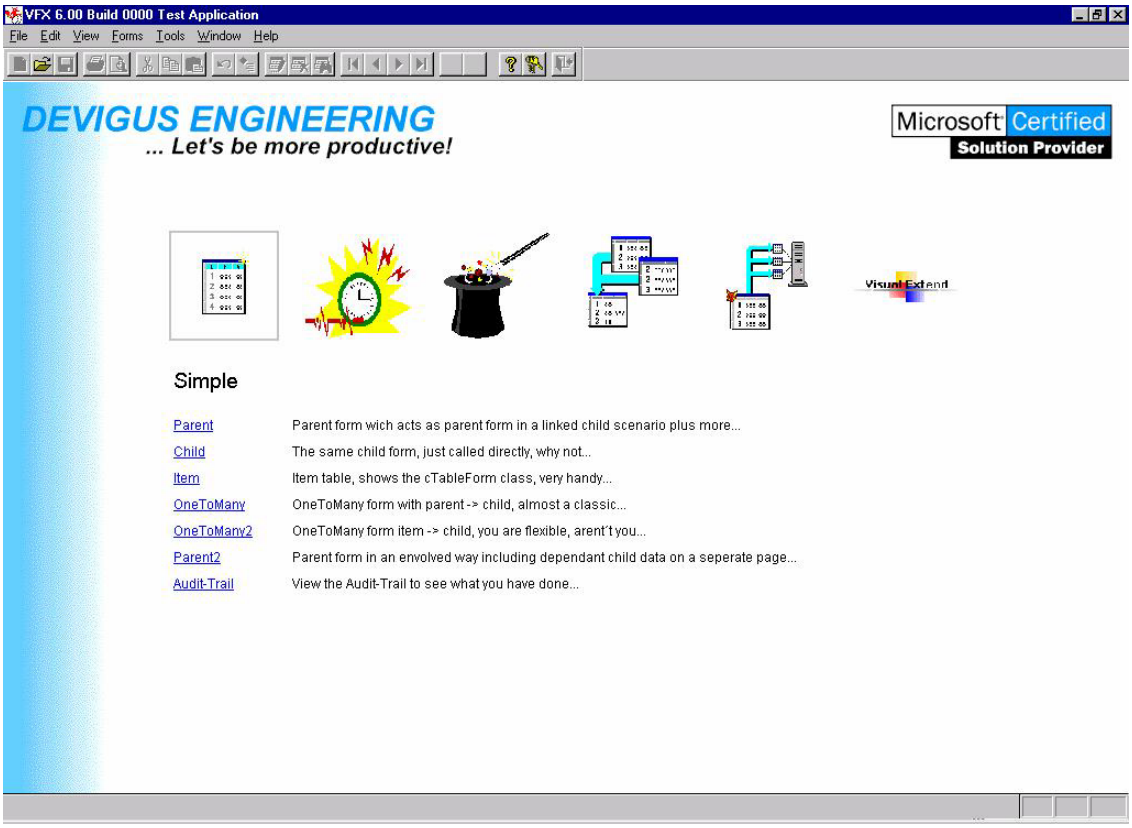
Visual Extend 7.0 (VFX 7.0) ist ein Anwendungs Entwicklungs Framework (RAD) für Softwareentwickler, die mit Microsoft Visual FoxPro Version 6.0 oder 7.0 arbeiten. Visual Extend beinhaltet Builders die den Entwickler in seiner täglichen Arbeit unterstützt und die Anwendungsentwicklung dramatisch beschleunigt, ohne die Möglichkeiten von Visual FoxPro einzuschränken. Mit Visual Extend wird Visual FoxPro ein richtiges “Rapid Application Development” (RAD) -werkzeug für die Erstellung von Desktop- und Client-Server Datenbankapplikationen.

Active Desktop Klasse

Die Active Desktop Klasse ermöglicht es dem Anwender jede beliebige Funktion einer Anwendung mit einem einzigen Klick zu starten. Diese Klasse „CNavCont“ können Sie in der Klassenbibliothek VFXTOOLS.VCX finden und erlaubt es Ihnen eine richtige Ein-Klick-Oberfläche zu entwickeln, wie Sie in der Windowswelt und dem Internet Standard ist.

Zur Nutzung dieser Klasse finden Sie weitere Informationen in der Dokumentation von VFX, sowie dem Worddokument zur Session V-ACTI, beziehbar über das dFPUG-Portal auf http://portal.dfpug.de.

Zur Erinnerung ein Screenshot eines Active Desktop, wie Sie es kennen:



Den Active Desktop erweitern

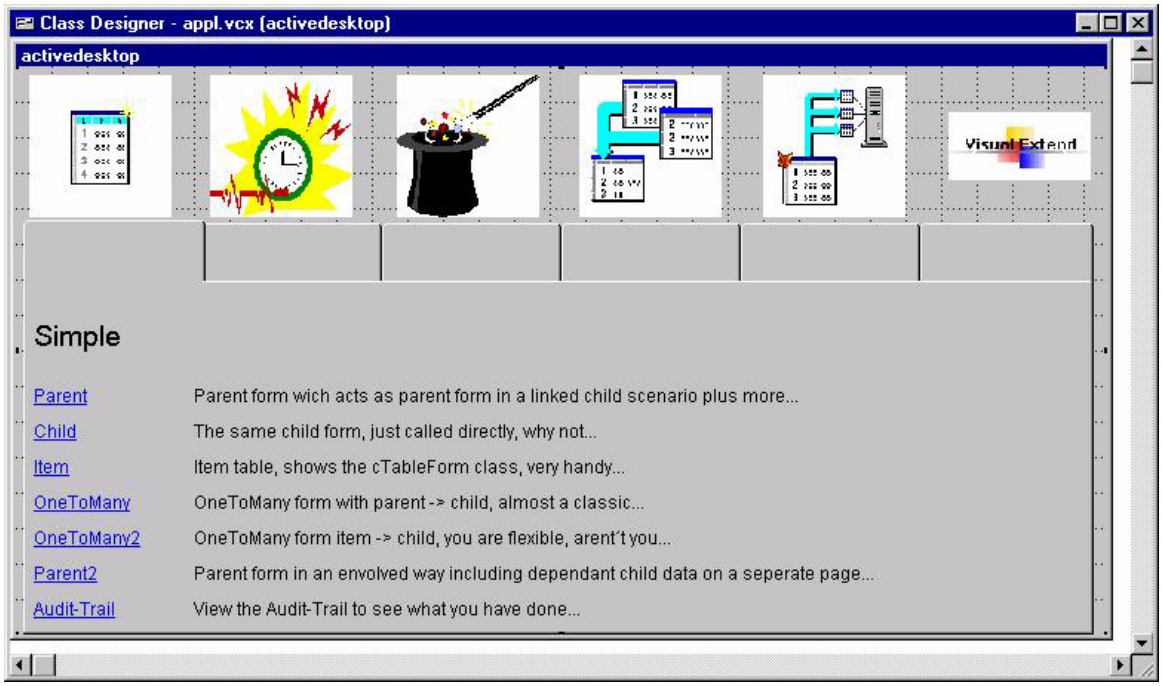
Trotz des schon recht großen vorhandenen Platzes zur Unterbringung von Links auf dem Desktop (sechs Registerkarten), kann es bei größeren Applikationen nötig sein noch weitere Registerkarten zuzufügen, um thematisch alle Funktionen Ihrer Applikation besser unterzubringen.

Wo die Erweiterungen vorgenommen werden

Alle Änderungen und Anpassungen passieren in der applikationseigenen Klassenbibliothek APPL.VCX. Hier ist eine vererbte Klasse CNavCont bei der Erstellung der Applikation angelegt worden. Ändern Sie niemals in der Klassenbibliothek VFXTOOLS.VCX, da diese bei einem Update von VFX überschrieben wird!

Der Aufbau der Klasse

Die Klasse besteht im Wesentlichen auf einem Formular mit Register, sechs Registerseiten, und Grafiken für die Steuerung, wie Sie aus der unteren Abbildung ersehen können.

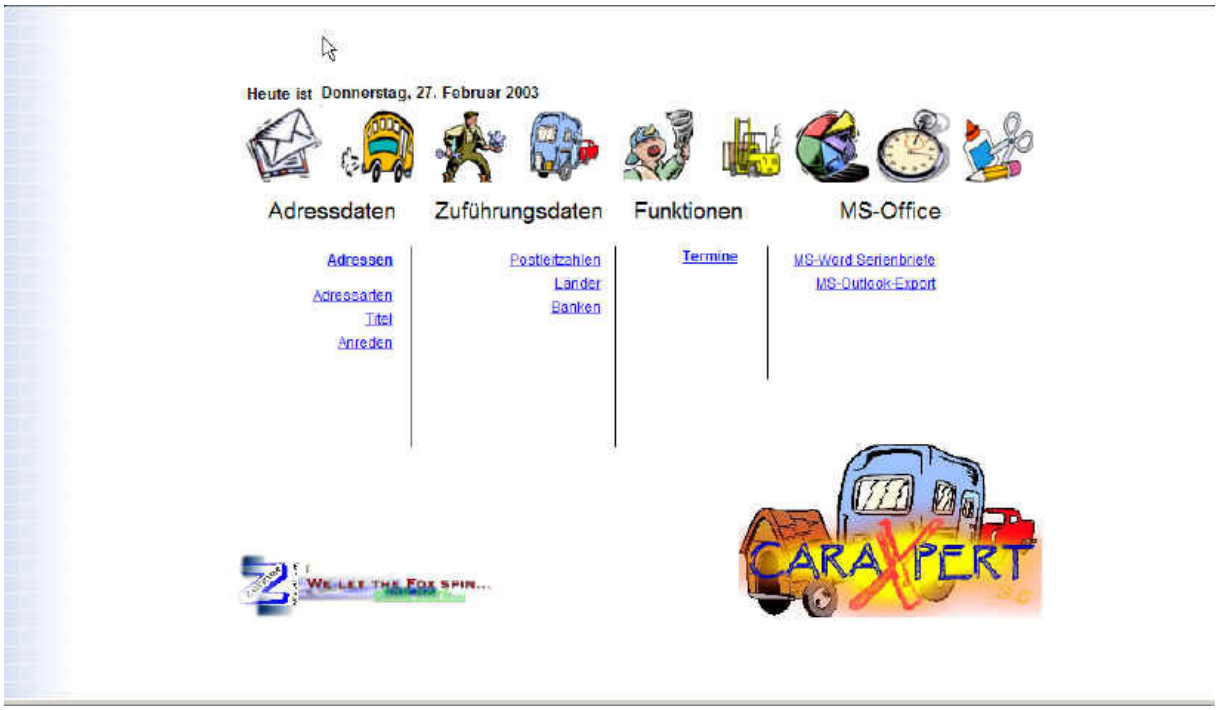


Während der Laufzeit werden die Registerreiter ausgeschaltet, damit man Sie nicht sehen kann. Die weitere Funktionalität des Registers, wie auch der Grafiken ergibt sich aus der Beschreibung der Änderungen.

Das Ziel

Unten sehen Sie ein Beispiel eines erweiterten Desktops. Einmal im Klassendesigner und einmal zur Laufzeit als Desktop. Wie Sie sehen hat dieser Desktop nunmehr 9 Registerkarten. Entsprechend haben Sie also drei weitere Unterteilungsmöglichkeiten für die Funktionen Ihrer Applikation. Ebenso benötigen Sie für eine Erweiterung die entsprechende Anzahl als Grafiken.

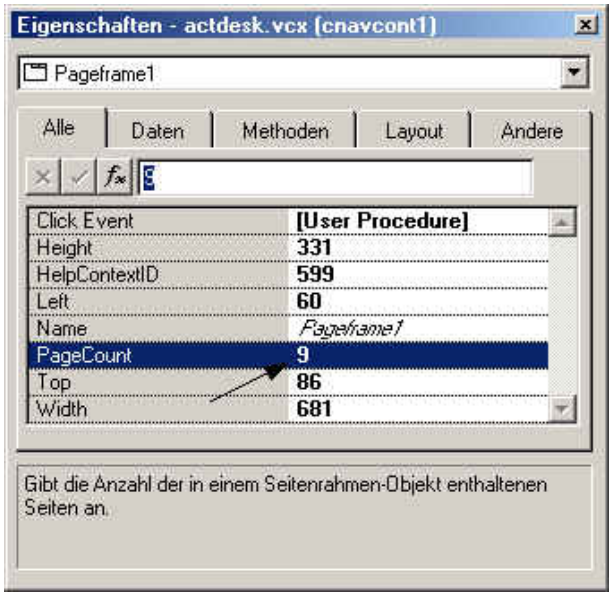
Auf wie viele Registerkarten Sie den Desktop schließlich erweitern ist gänzlich Ihnen überlassen, bedenken Sie aber bitte die evtl. Bildschirmauflösung des Anwenders und die mit der steigenden Anzahl an Grafiken für die Registerkarten geänderten Größenverhältnisse.



Die Änderungen

Nachfolgend erhalten Sie eine detaillierte Beschreibung über die Änderungen, die Sie vornehmen müssen:

1. Ändern Sie auf **thisform.pgpgframe.pagecount** auf die Anzahl Registerkarten, die Sie wünschen:



2. Stellen Sie die gewünschten zusätzlichen Grafiken im Ordner **bitmap** Ihrer Anwendung bereit und fügen Sie diese Grafiken Ihrem Projket hinzu.
3. Erzeugen Sie neue Objekte aus der Klasse **cbuttonimage**, ebenfalls in der Klassenbibliothek VFXTOOLS.VCX vorhanden, in entsprechender Anzahl, je nachdem, wie viele Registerseiten bzw. Grafiken hinzukommen, positionieren und ändern Sie die Größe der Grafikobjekte (**width** und **height**)

4. Machen Sie die Code-Anpassungen in den Grafikobjekten, dem Register nebst Karten und der Klasse selbst, wie folgend beschrieben:

Code ändern

Sämtlicher Code, der geändert wird, wird wie o. b. nie in der Parentklasse geändert, sondern in der eigenen abgeleiteten Klasse. Da in der Ableitung natürlich kein Code hinterlegt ist, er wird ja vererbt, können Sie mit DODEFAULT() und zusätzlichem Code arbeiten, wenn keine *wirklichen* Änderungen des Codes vorgenommen werden müssen. Um persönlich einen besseren Überblick zu behalten, was wo passiert, habe ich mich in diesem Beispiel dazu entschlossen den gesamten Code in meine Ableitungsfunktionen der Klasse zu kopieren und dort direkt anzupassen. Dann rufe ich natürlich auch kein DODEFAULT() auf.

Änderungen in cButtonImage

Durch Einfügen eines neuen ButtonImages auf die Klasse erhält das neueButtonimage automatisch den Namen „cButtonImage“ gefolgt von einer laufenden Nummer, für den ersten Button also die „7“. Sie müssen sich die Nummern bzw. Namen dieser Objekte merken, weil Sie in den Funktionen angesprochen werden. Die Methode *oninteractivechange* ist schon eingerichtet, es steht aber kein vererbbarer Code zur Verfügung. Diesen müssen Sie sich aus einem anderen ButtonImage kopieren und ändern, oder neu schreiben

OnInterActiveChange des buttons

Hierbei ist zu beachten, dass der Code die Verbindung des Image zu der entsprechenden Registerkarte darstellt. Welche Ziffer also für die Abfrage und die Zuweisung benutzt werden muss. Ist es wie in diesem Falle die 7. Registerkarte, lautet der Code wie folgt:

```
IF THIS.PARENT.pageframe1.ACTIVEPAGE <> 7
    THIS.PARENT.pageframe1.ACTIVEPAGE = 7
    THIS.PARENT.onrefresh()
    THIS.BORDERWIDTH=4
ENDIF
```

MouseMoveEvent des Buttons

Diese Methode hat den Code aus CButtonImage geerbt und braucht auch nicht geändert werden. Hier wird lediglich die Methode *OnInterActiveChange* des ButtonImages aufgerufen.

Änderungen in ThisForm

Durch Einfügen eines neuen ButtonImages auf die Klasse erhält das neueButtonimage automatisch den Namen „cButtonImage“ gefolgt von einer laufenden Nummer, für den ersten Button also die „7“. Sie müssen sich die Nummern bzw. Namen dieser Objekte merken, weil Sie in den Funktionen angesprochen werden. Die Methode *oninteractivechange* ist schon eingerichtet, es steht aber kein vererbbarer Code zur Verfügung. Diesen müssen Sie sich aus einem anderen ButtonImage kopieren und ändern, oder neu schreiben

Centerme und OnEventHook

Diese Methoden sind aus *CNavCont* geerbt und müssen nicht geändert werden.

MouseMove

Diese Methode ist aus *CNavCont* geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist:

```
LPARAMETERS nbutton, nshift, nxcoord, nycoord

THIS.centerme()

THIS.cbuttonimage1.REFRESH()
THIS.cbuttonimage2.REFRESH()
THIS.cbuttonimage3.REFRESH()
THIS.cbuttonimage4.REFRESH()
THIS.cbuttonimage5.REFRESH()
THIS.cbuttonimage6.REFRESH()
THIS.cbuttonimage7.REFRESH()    &&      <- NEU
```

OnRefresh

Diese Methode ist aus *CNavCont* geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist:

```
THIS.cbuttonimage1.BORDERWIDTH=0
THIS.cbuttonimage2.BORDERWIDTH=0
THIS.cbuttonimage3.BORDERWIDTH=0
THIS.cbuttonimage4.BORDERWIDTH=0
THIS.cbuttonimage5.BORDERWIDTH=0
THIS.cbuttonimage6.BORDERWIDTH=0
THIS.cbuttonimage7.BORDERWIDTH=0    && <- NEU
```

Änderungen in Page*1* bis Page*n*

Änderungen, bzw. zusätzlicher Code aus jeder Registerkarte betreffen zwei Methoden:

MouseMoveEvent

Diese Methode ist aus *CNavCont* geerbt, muss aber angepasst werden, da hier nur der Code für 6 Registerkarten hinterlegt ist. Zu beachten ist, dass diese Methoden für die 6 vorhandenen Seiten verändert werden müssen und für jede neue Seite ganz neu gefüllt werden müssen, allerdings immer mit dem gleichen Code:

```
LPARAMETERS nbutton, nshift, nxcoord, nycoord

WITH THIS.PARENT.PARENT
    .cbuttonimage1.REFRESH()
    .cbuttonimage2.REFRESH()
    .cbuttonimage3.REFRESH()
    .cbuttonimage4.REFRESH()
    .cbuttonimage5.REFRESH()
    .cbuttonimage6.REFRESH()
    .cbuttonimage7.REFRESH()    && <- NEU
    .centerme()
ENDWITH
```

Click

Diese Methode ist aus *CNavCont* geerbet, und muss für geerbte „Pages“ nicht geändert werden. Für neue „Pages“ muss hier entsprechend den anderen Registerkarten neuer Code rein. Bezogen auf die neue Registerkarte:

THIS.PARENT.PARENT.cbuttonimage7.CLICK()

Zusammenfassung

Sie sehen, mit ein wenig zusätzlichem Code und minimalem Zeitaufwand können Sie den Active Desktop von VFX erweitern und anpassen. Durch weitere Grafische Elemente, wie Sie sie in meinem Beispieldesktop sehen können, können Sie weitere auflockernde, erklärende und/oder funktionelle Objekte auf den Desktop bringen. Texte können Sie einfach auf die Klassenobjekte verteilen und bei Grafiken, die nicht in *cButtonImage* eingebettet sind, muss dann lediglich die Prozedur *DrawBackground* in der Programmdatei *vfxmain.prg* angepasst werden. Hier nochmals der Verweis zum Worddokument zur Session V-ACTI.

Wie immer garantiere ich für nichts! Sicherlich gibt es in Details andere Möglichkeiten des Active Desktop von VFX zu ändern, scheuen Sie sich nicht *Ihre* Lösung niederzuschreiben. Sie können auch gerne Hinweise, weitere Verbessungsvorschläge und Anregungen zusenden. Das Gleiche gilt natürlich auch für Fehler, die Ich hier natürlich absichtlich eingebaut habe <g>. Besuchen Sie die Internetseite www.my-vfx.de für weitere Informationen und Kontaktmöglichkeiten.
Probieren Sie diese Anpassungen erst an einem Beispielprojekt aus, bis Sie mit den Vererbungen, Codes und Verhaltensweisen des Active Desktop vertraut sind. Nehmen Sie erst dann Änderungen für Ihr richtiges Projekt vor. Wenn Sie mit dem Active Desktop vertraut sind, dann ist der Arbeitsaufwand es bei einem Anderen Projekt zu wiederholen sehr gering und rechtfertigt dann nicht einmal mehr 7 Seiten Dokumentation.