Runtime localization mit VFX

Herzlich Willkommen!

Runtime localization ist ein Projekt von www.my-vfx.de

Stefan Zehner *Stand: 01.03.2003*

Runtime localization mit VFX 1
Vorwort 4
Begriffe 5
Lokalisierung 5
Bestandteile 6
Gedanken 6
VFX-Tabellen 8
Include Dateien 10
Menüs 12
Active Desktop 13
Config.fpw 15
Reports 16
Reportübersetzung für den Anwender 18
Steuerelemente 20
VFX-Splash Screen 22
Automation 23
Die Sprache wechseln 23
Zusammenfassung 25
Weiteres... 26
Warum nicht...? 26
Was bringt die Zukunft? 26
Nachwort 27

Vorwort

Wäre es nicht wunderbar einem Anwender die Möglichkeit zu geben eine Anwendung in mehreren Sprachen ausführen zu lassen, ohne gleich ein Programmupdate, eine ganz andere Programmdatei oder Sprachdateien ausliefern zu müssen?

VFX bietet eine Komplettlösung für die Erstellung von Applikationen in verschiedenen Sprachen an. Diese ist allerdings darauf beschränkt, dass es keine Umstellmöglichkeit zur Laufzeit gibt und pro Sprache eine eigene Applikation kompiliert werden muss.

Aber VFX legt den Grundstein für echtes Multilanguageverhalten.

Zusatztools wie Steven Black's INTL-Toolbox werden überflüssig, wenn einem erst einmal klar geworden ist, wie's funktioniert.

Hier finden Sie alles, was Sie zur Lösung benötigen.

Eine Gemeinschaftsproduktion von:

- Uwe Habermann (Grundlagen, VFX-Behandlung),
- Stefan Zehner (Umsetzung für Powerpoint, Word und Beispielapplikation)

  und

- Michael Schmitz (Reports für mehrere Sprachen)

Begriffe

Was verstehen wir in diesem Fall eigentlich unter "Runtime" und "Designtime"?

Lokalisierung

Designtime

... ist die Sprachfestlegung während der Entwicklung der Anwendung.

Für jede Sprache muss eine separate EXE-Datei erzeugt werden.

Runtime

... ist die Sprachfestlegung während der Ausführung der Anwendung.

Es wird eine EXE-Datei erstellt, die dem Anwender die Möglichkeit bietet die Sprache auszuwählen

Bestandteile

Was machen wir hier eigentlich?

In diesem Teil erfahren Sie, welche Komponenten Ihrer Anwendung angepasst werden müssen, damit in nur einer Anwendung mit mehreren Sprachen gearbeitet werden kann.

Gedanken

Allgemeines

Was müssen wir rund um unsere Anwendung beachten?

Damit in nur einer Anwendung mehrere Sprachen nach Wahl angezeigt werden können, müssen wir außer an Teile unserer Anwendung auch an das Betriebssystem denken, auf welchem unsere Anwendung laufen soll.

So müssen wir z. B. die mit unserer Anwendung auszuliefernden DLL und/oder OCX Dateien für alle diese Sprachen bereithalten.

Wir müssen ebenso daran denken, dass Windowsmeldungen natürlich in der installierten Sprachen angezeigt werden.

Wir können sogar Sprachen darstellen, die eine andere Schrift als die Lateinische benutzen, wie etwa Griechisch oder Russisch. Allerdings muss das Betriebssystem das können (Win 2000/ XP), es müssen Tastaturtreiber und Schriften für den Zeichensatz vorliegen, und ab VFP8 in Verbindung mit XP können wir sogar Verschiedene Sprachen und auch Schriften innerhalb einer Anwendung auf nur einer Form, z. B. in verschiedenen Steuerelementen darstellen (vergl. Keynote von Ken Levy Devcon 2002).

Vorbereitung

Der Weg geht über die Designtime Lokalisierung

- Zunächst gehen Sie wie bei einer Designtime-Lokalisierung vor.
- Alle Texte innerhalb von Klassen, Formularen, Berichten und Programmdateien müssen durch Konstanten ersetzt werden.
- Bei Formularen kann hierzu der VFX-LangSetup-Builder verwendet werden:

  * Öffnen Sie das Formular im Editor.
  * Wählen Sie aus dem VFX-Menü den Eintrag „LangSetup-Builder“.
  * Aktivieren Sie bei Bedarf die Kontrollboxen für **ToolTipText** und **Statusbar.**
  * Aktivieren Sie die Kontrollbox für **Overwrite Code**, wenn Sie schon vorhandenen Code in der LangSetup-Methode Ihrem Formular überschreiben möchten.
  * Starten Sie den LangSetup-Builder.

    Der LangSetup-Builder generiert nun Datensätze für den Bezeichnungscode in der vfxmsg.dbf und den Ausführungscode für die LangSetup-Methode.
  * Schließen Sie den LangSetup-Builder.

- Für jede Konstante ist mit dem VFX-Message-Editor ein Datensatz anzulegen:

  * Öffnen Sie den Eintrag **VFX-Message-Editor** aus dem **VFX-Menü.**
  * Auf der Seite **List** des Editors können Sie unter Aktivierung **Type=other** für alle Captions (inkl. TipToolText und Statusbar) und Texte die Übersetzungen in die jeweilige Sprache vornehmen.
  * Eigene Texte können Sie unter Aktivierung von **Type=Message** manuell hinzufügen. Diese Texte werden dann für Messageboxen, WaitWindows etc. herangezogen werden.

  Tipp: Eigene Meldungen die sich in der Anwendung wiederholen (z. B. Allgemeine Meldungen), nur Einmal anlegen und den vergebenen Konstantennamen so oft benutzen, wie gewünscht.

VFX-Tabellen

Das Herzstück der Übersetzungen. Alles läuft über Tabellen. VFX übergibt uns schon in der Installationsversion einen guten Grundstock. Nutzen wir ihn für uns....

VFXmsg Tabelle

Messages für Messages und alles Andere

In ihr finden alle Texte inklusive der Übersetzungen Platz, um Formulare, Steuerelemente und on-the-fly-Berichte zu übersetzen.

Die Tabelle VFXmsg.dbf muss in das Projekt eingeschlossen werden.

Während der Laufzeit wird dann die Tabelle ausgelesen.

Der folgende Codeblock wird in **vfxmain.prg** vor die Zeile

```
PUBLIC goenvironment,glclientsupport
```

eingefügt:

```
USE data\vfxmsg.dbf
SCAN
        lcpublicname = ALLTRIM(message_id)
        PUBLIC(lcpublicname)
        &lcpublicname = ALLTRIM(&id_language)
ENDSCAN
USE IN vfxmsg.dbf
```

Für jeden Datensatz wird hier eine Public Variable deklariert und diesen die Werte aus der Tabelle – entsprechend der ausgewählten Sprache – zugeordnet.

VFXfopen Tabelle

Wenn Sie in Ihrer Anwendung die Öffnen-Funktion von VFX benutzen, müssen Sie in dieser Tabelle ebenfalls die enthaltenen Texte übersetzen.

* Die VFXfopen.dbf für jede Sprache kopieren (Copy Stru... to ...) und mit verschienen Namen speichern (vfxfopenger.dbf,vfxfopenita,...).
* Wie in der Originaltabelle den Index **OBJECT** anlegen.
* Alle Sprachtabellen in das Projekt einbeziehen.

Der einfachste Weg unsere Übersetzungen zur Laufzeit dann in das Öffnen-Fenster zu bekommen, ist ganz einfach die VFXfopen.dbf zu leeren oder zu löschen und mit den übersetzten Inhalten aus den Sprach-VFXfopenXXX.dbf zu füllen. Der folgende Codeblock ist ebenfalls vor der Zeile

```
PUBLIC goenvironment,glclientsupport
```

einzufügen:

```
IF FILE(„data\vfxfopen.dbf“)
        DELETE FILE data\vfxfopen.dbf
        USE FILE data\vfxfopenXXX AGAIN SHARED
        SELECT data\vfxfopenXXX
        COPY TO data\vfxfopen.dbf
        USE data\vfxfopen EXCLUSIVE
        SELECT vfxfopen
        INDEX ON UPPER(objected)+STR(objectno) TAG OBJECT
ENDIF
```

Include Dateien

Include-Dateien, eigentlich zur Definition von Konstanten da, werden für übersetzte Texte unwichtig. Viel wichtiger ist, dass wir hier natürlich die "Verbindungen" zur Sprache rausnehmen müssen, damit diese nicht fest beim Kompilieren eingebunden werden. Im Folgenden die einzelnen Include-Dateien:

VFX.h

Die normalerweise enthaltenen Verweise auf weitere Include-Dateien müssen entfernt werden:

```
*#INCLUDE „INCLUDE\VFXTXT.H“
*#INCLUDE „INCLUDE\VFXMSG.H“
*#INCLUDE „INCLUDE\USERTXT.H“
*#INCLUDE „INCLUDE\USERMSG.H“
```

Damit die LangSetup-Methode in den Formularen auch ausgeführt wird, muss diese hier “eingeschaltet“ werden:

```
#DEFINE_LANG_SETUP .T.
```

Und nicht vergessen: Die VFX.h **muss** in das Formular eingebunden werden (**Menü Formular - Includedatei...)**

VFXdef.h

Die Definition der Sprachauswahl muss aus der VFXdef.h in das Hauptprogramm (vfxmain.prg) übernommen werden, weil die VFXdef.h die Sprachdefinition sonst bei der Kompilierung festlegt, und weil sie aus der VFX.h nun nicht mehr referenziert wird. Diese Aufgabe übernimmt jetzt eben das vfxmain.prg.

Aus

```
#IF ID_LANGUAGE =“ENG“
    #DEFINE ID_NUM_SEPARATOR = „,“
    #DEFINE ID_NUM_DECIMAL = „.“
#ENDIF
...
```

in der **VFXDEF.H** wird

```
DO CASE
    CASE...
    CASE ID_LANGUAGE = „ENG“
        ID_NUM_SEPARATOR =”,”
        ID_NUM_DECIMAL = “.”
    CASE...
    ...
ENDCASE
```

in **vfxmain.prg.**

Auch hier bitte wieder direkt vor der Zeile:

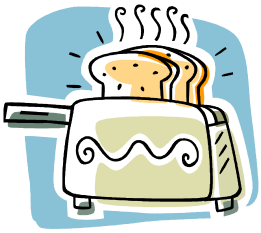
```
PUBLIC goenvironment,glclientsupport
```

Die Sprachfestlegungen aller Sprachen müssen ebenfalls noch entfernt werden:

```
*#DEFINE ID_LANGUAGE „ENG“
```

Menüs

"Mit unserem Programm können Sie in allen Sprachen speisen!"



Legen Sie für jede Sprache eigene Menüdateien mit den entsprechenden Übersetzungen des ursprünglichen Menüs an. Auch hier bietet es sich an vom **vfxmenu** bzw. **vfxmenu7** auszugehen. Für das englische Menü also **vfxmenueng**.

Schließen Sie alle Menüs mit in ihr Projekt ein. Nur dann ist es für die Anwendung möglich auch darauf zu referenzieren.

In **vfxmain.prg** übergeben Sie dann den Namen des Menüs als Parameter an das Anwendungsobjekt.

Die Zeile:

```
goprogram = CREATEOBJECT("CApplication", cap_application_title)
```

wird durch folgenden Codeblock ersetzt:

```
DO CASE
CASE id_language = "GER"
   goprogram = CREATEOBJECT("CApplication",cap_application_title,"ger.mpr")
CASE id_language = "ENG"
   goprogram = CREATEOBJECT("CApplication", cap_application_title,"eng.mpr")
CASE id_language = "FRE"
   goprogram = CREATEOBJECT("CApplication", cap_application_title,"frz.mpr")
CASE id_language = "ESP"
   goprogram = CREATEOBJECT("CApplication", cap_application_title,"esp.mpr")
CASE id_language = "ITA"
   goprogram = CREATEOBJECT("CApplication", cap_application_title,"ita.mpr")
ENDCASE
```

Active Desktop

Warum "Active"?

Let's get active!

Auch der VFX-Active Desktop kann entsprechend angepasst werden, schliesslich ist er im Prinzip nichts anderes als ein Formular, wenn auch als Klasse gespeichert. Lediglich eine LangSetup-Methode muss eingefügt werden. Auch für diese LangSetup-Methode können wir natürlich den LangSetup-Builder benutzen, wenn wir die Klasse als Formular bearbeiten.

Durchführung

Von der Klasse zum Formular...

- Bearbeiten Sie die Klasse „activedesktop“ aus der Bibliothek „appl“ wie gewohnt im Klassendesigner von VFP.
- Erzeugen Sie im Formulardesigner eine neue leere **VFP**-Form.
- Ziehen Sie die Klasse „activedesktop“ auf dieses Formular.
- Wählen Sie im Eigenschaftsfenster „Form1“, den automatisch durch VFP generierten Namen des Formulars.
- Benutzen Sie für dieses Formular den LangSetup-Builder wie gewohnt.
- Kopieren Sie den erzeugten Code in die Windows Zwischenablage.
- Schließen Sie das Formular ohne zu speichern.

... und wieder zurück

- Öffnen Sie die Klasse „activedesktop“ erneut im Klassendesigner.
- Erstellen Sie eine Methode „langsetup“ (Menü Klasse -> neue Methode) und kopieren Sie den Code aus der Zwischenablage hinein.
- „ThisForm“ wird zu „This“.
- „PageFrame1“ ist die oberste Hierarchie, dass heißt, alles was vor „PageFrame1“ in den Objektpfaden steht kann gelöscht werden.
- Löschen Sie bitte ebenfalls die Caption der Formularüberschrift.
- Fügen Sie in der Init-Methode der Klasse **this.langsetup()** ein (DoDefault() nicht vergessen!)
- Übersetzen Sie alle Texte, die der LangSetup-Builder in der langSetup-Methode eingetragen hat und geben Sie diese in die vfxmsg.dbf ein (Wie wär’s wieder mit dem VFX-Message-Editor?)
- Löschen Sie den Datensatz für die Formularularüberschrift, die bei der ersten Benutzung des LangSetup-Builders automatisch angelegt wurde

Hier ein Codebeispiel für eine so entstandene LangSetup-Methode:

```
#ifdef _lang_setup
  *this.Caption = CAP_FRMFORM1
  …
  this.pageframe1.page1.label1.Caption = CAP_LBLMEINACTIVEDESKTOP
  this.pageframe1.page1.label2.Caption = CAP_LBLMEINELIEBLINGSPROJEKTE
  this.pageframe1.page1.clinklabel1.Caption = CAP_LBLPROJEKTE
  this.pageframe1.page2.label1.Caption = CAP_LBLPAGE2
  this.pageframe1.page2.label2.Caption = CAP_LBLEXAMPLE
  this.pageframe1.page3.label1.Caption = CAP_LBLPAGE3
  …
  return .T.
#endif
```

…und noch die Include:

Wie für das Formular muss auch für die Klasse die Includedatei VFX.h zugefügt werden (Menü Klasse →; Include-Datei...)

Alles rekompilieren, - und schon haben wir eine multilinguale Formularklasse. Wenn Ihre Formulare also auf speziellen Klassen aufgebaut sind, ist das eine schöne Möglichkeit mit einem Schlag für alle Formulare die Grundelemente multilingual zu gestalten!

Nachträgliche Änderungen durch neue Steuerelemente etc. können selbstverständlich einfach in der Methode langsetup() und der Tabelle vfxmsg.dbf, nachbearbeitet werden.

Config.fpw

Variablen

Variable Variablen!

VFP muss alle Public-Variablen die deklariert werden, verwalten. Standardmäßig verwaltet VFP bis zu 1024 Public-Variablen. Diese Anzahl an zur Verfügung stehenden Variablen kann ein wenig eng werden, wenn wir Texte vieler Formulare und freie Texte zur Verfügung stellen möchten.

Die Standardgröße der vfxmsg.dbf hat 285 Datensätze, also reichlich Platz für generierte 285 Variablen in den 1024 durch VFP verwalteten. Aber durch unsere Habgier nach Variablen sollten wir diesen „Puffer“ ruhig etwas höher setzten.

Was?

Dann woll'n wir mal zählen...

MVCOUNT stellt die maximale Anzahl an Variablen ein, die Visual FoxPro verwalten kann. Der Bereich geht von 128 bis 65.000; Standardeinstellung ist 1.024.

Wie?

oder Wo?

Diese Einstellung nimmt man am besten in der **config.fpw** vor, die man vorzugsweise im Projektverzeichnis erstellt, damit nicht eine eventuelle config.fpw aus dem VFP-Ordner herangezogen wird, bzw. unsere Einstellung nicht andere Projekte berühren.

```
MVCOUNT=2048
```

Reports

VFX-Reports

Müssen wir denn unbedingt drucken?

JA, unsere Kunden wollen es so. Und sie sollen auch. Wir kriegen das schon alles übersetzt:

Sie heißen "automatisch" - weil's eben "automatisch" ist!

Die automatischen Reports benötigen keine Änderung, da diese auf den VFX-Grids beruhen und dadurch schon mit Captions zw. Deren Übersetzungen versorgt sind.

Reportauswahl: Wie mach ich es?

Doppelte Arbeit bei wenig "Gehirnwringen":

Sie könnten für jede Sprache einen Report anlegen. Nachteil ist sicherlich der Aufwand der für die Pflege entsteht.

Vor dem Start des Reports wird dann entschieden, welcher der Reports ausgeführt wird (id_language).

Report-Coded

Wir sind auf dem richtigen Weg...

Sie erstellen einen Report für alle Sprachen. Damit alle Captions in der richtigen Sprache dargestellt werden, definieren Sie die Captions als Textboxen. In den Textboxen entscheiden Sie anhand der id_language, welchen String Sie direkt aus dem Report anzeigen.

Schon besser!

Report-Tabelle

Jetzt wird's richtig variabel:

Sie erstellen einen Report für alle Sprachen. Damit alle Captions in der richtigen Sprache dargestellt werden, definieren Sie die Captions als Textboxen. In den Textboxen werden die Inhalte aus Sprachtabellen dargestellt.

Diese können bei bedarf auch recht schnell angepasst werden.

- Erstellen Sie einen Report im VFP-Report Designer.
- Erstellen Sie eine (freie) Tabelle, in der für diesen Report alle Captions (Spalten) in jeder Sprache (Zeilen) abgelegt werden. Zur Identifikation der Sprache definieren Sie zusätzlich ein Feld, in der Sie die gleiche ID speichern, die von VFX benutzt wird.
- Fügen Sie diese Tabelle der Datenumgebung des Reports zu und setzen Sie einen Filter auf die ID, z. B.: "Tabellenfeld"=id_language.
- Löschen Sie die Captions Ihres Reports und ersetzen Sie diese durch Textboxen, denen Sie die Felder Ihrer Sprachtabelle zuordnen.

Da oftmals mehrere Reports ähnlich sind, bzw. viele gleiche Daten ausgeben, können Sie natürlich auch eine Tabelle für mehrere Reports erstellen.

Reportübersetzung für den Anwender

Was heißt das?

Internationale Kunden bedienen: Professionell in der jeweiligen Landessprache.

Die Globalisierung lässt grüßen. Aber auch, wer sich 'nur' im europäischen Ausland bewegt, möchte seine Kunden in deren Landessprache anreden. Das schafft Vertrauen und stärkt die Geschäftsbeziehungen.

Wir lassen unsere Kunden arbeiten ;-)

Als kleine Erweiterung der Reportgenerierung, mit wenigen Eingriffen des Entwicklers, können Sie für Ihre Kunden eine Erweiterung erstellen, die es dem Anwender ermöglicht einen Report in jeder gewünschten Sprache zu erstellen. **Sie brauchen nicht zu wissen, welche Sprachen benutzt werden sollen**, alles läuft über Tabellen, die der Anwender selbst pflegt, aber keine Änderungen an Reports vornehmen muss!

Wie wird’s gemacht?

- Erstellen Sie eine Tabelle, in der u.a. die Überschriften hinterlegt werden.

Structure of table : **FORMULARTEXTE** (Fields: 8)

Field Name	Type	Width	Dec	Caption
CFELDNAME	C	2	0	Feldnummer
CFELDSPRACHE	C	2	0	Sprache
CFELDKURZ	C	40	0	Formulartext kurz
MFELDLANG	M	4	0	Formulartext lang
CINS_USR	C	10	0	angelegt von
DINS_DATE	D	8	0	angelegt am
CEDT_USR	C	10	0	geändert von
DEDT_DATE	D	8	0	geändert von
Record Length		84		

Indices of table : **FORMULARTEXTE**

Tag Name	Primary	Key
FELDSPR	Yes	CFELDNAME+CFELDSPRACHE

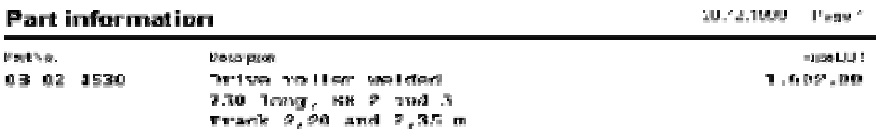
- Und eine Funktion (txt_konstante()) die in der Tabelle nach der Überschrift sucht. Falls keine Überschrift gefunden wird, wird immer die deutsche genommen. Zu jeder Feldnummer kann es auch eine Langform geben, die immer genommen wird, wenn sie da ist. So werden nicht nur Überschriften übersetzt, sondern der Anwender kann auch längere Texte in der gewünschten Sprache drucken. (z.B. "Keine Warenannahme in der Zeit vom 22.12.2001 bis 2.1.2002").

- Erstellen Sie ein Formular, in der der Anwender die Texte übersetzt:

Was in dem Feld "Feldnummer" zulässig ist, wird über eine andere Tabelle festgelegt, die vom Entwickler gepflegt und immer ausgeliefert wird.

- Im Report steht überall, wo in einer Fremdsprache gedruckt werden soll: **txt_konstante("04",lcsprache)**.

Eigentlich ganz einfach. Und so sieht's aus:



Steuerelemente

Grids, Combos, Listen und andere

Label, Checkbox und Option

Prinzipiell ist der Ansatz bei jedem Steuerelement gleich. Ob dabei der anzuzeigende Text eine Caption oder ein Value ist, ist dabei eigentlich egal. Die Bildschirmausgaben müssen halt angepasst werden.

Wenn in diesem Zusammenhang die Includedateien bearbeitet wurden und programmatisch die Auswahl der Sprache vorgenommen wurde, dann läst sich der Rest unter Verwendung der VFX-Generatoren bzw Builder automatisieren.

Vorbereitet von VFX sind dadurch prinzipiell alle Steuerelemente, die eine ‚Caption' besitzen. also **Labels, Checkboxen und Optionsgruppen**.

Grids

Ebenfalls abgedeckt dürfte das Thema **Grids** sein, da wir ja als brave VFX-Entwickler immer ein Grid nehmen, welches aus einer VFX-Klasse kommt. Diese sind ebenfalls abgedeckt und funktionieren sozusagen automatisch über den VFX-LangSetup-Builder

Combos und Listen

An **Comboboxen und Listen** müssen wir dagegen noch Hand anlegen:

Diese Steuerelemente, die Sie in gewohnter Weise mit Tabellen- bzw. festen Werten (über den Generator) bestücken, werden nun auf Arrays umgestellt:

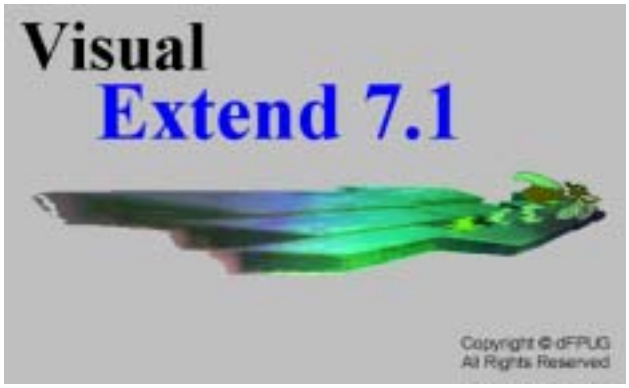
- RowSourceType auf „5“ stellen (Array)
- Fügen Sie der Datenumgebung des Formulares eine Tabelle mit den Einträgen und deren Übersetzungen zu (z.B.: vfxComboLanguage.dbf)
- Legen Sie ein Array mit den gewünschten Dimensionen als Formularvariable an (aComboLanguage)
- Im INIT des Formulars füllen Sie dann das Array mit den Tabellenwerten der eingestellten Sprache:

```
SELECT vfxcombolanguage
GO TOP
LOCATE FOR UPPER(ALLTRIM(langid)=UPPER(ALLTRIM(id_language)
…
STORE vfxcombolanguage.ger TO ThisForm.aCombolanguage[1]
…
STORE vfxcombolanguage.usr TO ThisForm.aCombolanguage[6]
…
```

- Und noch ein REQUERY() auf das Steuerelement, damit der Inhalt angezeigt wird.
- Bei Bedarf können Sie noch den Defaultvalue einstellen (...cCombobox.Displayvalue=aComboLanguage[4])

VFX-Splash Screen

Splish - Splash - und nass gemacht!



Sicherlich gibt es viele Ansätze eine SplashScreen zu erstellen und eigentlich ist in unserem Falle ebenfalls ein Formular angesagt. In VFX ist die Splash Screen ein BMP und in einer Grafik kann man bekanntlich keine Texte austauschen. Für die Benutzung eines Formulars als Splash Screens eignet sich als Anschauungsobjekt Das VFP-Beispiel Nordwind.

Zurück zu VFX: In unserem Fall erstellen wir für jede Sprache ein Intro-Logo, behalten aber auch ein allgemeingültiges für alle Fälle. Alle Bitmaps kommen in den Ordner "bitmap" und haben wieder verschiedene Namen, z. B. "IntroGER.bmp".

In VFX ist es jetzt ganz einfach dieses Bitmap entsprechend der Sprache aufzurufen. Wir ändern dafür lediglich eine Zeile in vfxmain.prg:

```
Alt:
        cintrobitmap        = introform_loc
Neu:
cintrobitmap=       IIF(id_language="GER","bitmap\introger.bmp",;
                    IIF(id_language="ENG","bitmap\introeng.bmp",;
                    IIF(id_language="FRE","bitmap\introfra.bmp",;
                    IIF(id_language="ITA","bitmap\introita.bmp",;
                    IIF(id_language="ESP","bitmap\introesp.bmp",;
            IIF(id_language="USR","bitmap\introkis.bmp","bitmap\intro.bmp"))))))
```

Tipp: Wenn Sie genau hinschauen finden Sie in der 1. Zeile die Konstante "introform_loc". Diese wird/wude in "vfx.h" definiert, aber nicht benutzt.

Ebenfalls besteht die Möglichkeit eine Formularklasse oder ein Formular zu verwenden. Lediglich der Aufruf in **vfxmain.prg** muss angepasst werden. Und wie Sie Formularklassen und Formulare handhaben, haben Sie jetzt schon gelernt.

Automation

Die Sprache wechseln

Was passiert wann

Beim Start der Anwendung wird die gewünschte (eingestellte) Sprache geladen.

Während die Anwendung läuft wird die Sprache dann über das Menü oder ein Steuerelement einer Form ausgewählt (Combobox, Liste, Optionen)

Wie passiert es

Bleibt zum Schluss noch zu klären wo und wie die gerade eingestellte Sprache gespeichert wird, damit sie beim Start der Anwendung auch geladen werden kann. Dazu gibt es folgende Überlegungen:

- Erweitern Sie die vfxmsg.dbf um ein Feld, in der Sie die ausgewählte Sprache hinterlegen (id_language).

  oder

- Lesen Sie die Windows Registrierung aus und starten Sie die Anwendung angepasst an das Betriebssystem, dann brauchen Sie keine Sprache zu speichern, die Anwendungssprache ist aber festgelegt auf die Windowseinstellung.

  oder

- Erzeugen Sie eine eigene INI-Datei und speichern Sie dort die ausgewählte Sprache bzw. lesen Sie sie aus, um die eingestellte Sprache zu laden.

Letzteres ist mein Favorit, jetzt können wir auch noch etwas über das Behandeln von INI-Dateien lernen:

Sprachauswahl über INI

Die INI-Datei enthält die Sprachabkürzung (id_language) derAnwendung:

```
...
[LANGUAGE]
APPLANG=”ITA”
...
```

Lesen Sie die INI-Datei aus, damit die Sprache geladen werden kann:

Mit Hilfe einer Klasse, die aus Funktionen von ***Dave Rooney*** erstellt wurde, kann die ini-Datei gelesen und in ihr geschrieben werden. Diese Klasse mit Namen **iniclass** wird in die Bibiothek **appl** eingebunden, die von VFX beim Erstellen des Projektes erzeugt wurde. Hier der Code-Snippet zum Lesen aus vfxmain.prg:

```
SET CLASSLIB TO lib\appl
oIniClass=CREATEOBJECT('iniclass')

IF FILE("myini.ini")
   id_language=oIniClass.xProfileString( "myini.ini", "LANGUAGE", "APPLANG")
ENDIF
```

Dieser Code-Block muss vor dieser Zeile eingefügt werden:

PUBLIC goenvironment,glclientsupport

Natürlich können Sie das auch selbst erledigen…

Wenn eine andere Sprache während der Laufzeit ausgewählt wird, speichern Sie die Auswahl in Ihrer INI-DateI:

```
lok=oIniClass.xSetProfile( "myini.ini", "LANGUAGE", "APPLANG", '"ENG"')
```

So ist dann auch die in der Newsgroup häufig gestellte Frage, wie man INI-Dateien bequem nutzen kann, geklärt.

Zusammenfassung

Sie haben gesehen, das VFX uns die globalen Einsatzmöglichkeiten unserer Software nicht nur ermöglicht, sondern auch dabei unterstützt mehr als den "Standard" zu erreichen.

Vielen mag sicherlich jetzt durch den Kopf gehen, das es sehr viel Arbeit bedeutet seine Anwendung auf die Art und Weise multilingual zu machen. Und sicherlich gibt es noch Dinge an die wir alle, die hier mitgearbeitet haben, noch nicht gedacht haben.

Aber wenn Sie sich mit dieser Materie vertraut gemacht haben und erkennen, was VFX schon vorgearbeitet hat, dann werden Sie auch sehen, dass beim täglichen Arbeitsaufwand nicht viel Zeit investiert werden muss.

Viele Dinge müssen nur einmalig pro Projekt gemacht werden, andere können direkt bei der Erstellung von neuen Formularen mit erledigt werden und selbst wenn Sie eine fertige Anwendung umstellen möchten, werden viele und quälende Arbeiten schon von VFX im Hintergrund erledigt.

Versuchen Sie sich mal zu überlegen, was VFX mit den Konstanten noch alles anfangen muss, bevor sie dann richtig im Formular angezeigt werden… Sehen Sie?

Downloads zu diesem Projekt erhalten Sie unter www.my-vfx.de.

Weiteres...

Warum nicht...?

Können andere etwas, was wir nicht können? - Aber ich bin sicher wir bekommen noch raus wie es geht.

Und vom Prinzip - machen andere es auch nicht anders.

Was bringt die Zukunft?

Weitere Hilfesysteme?

Tja, wenn ich das nur wüsste was die Zukunft bringt? Vielleicht gibt es ja jemanden unter Ihnen, der auch Lust hat der VFX-Gemeinde etwas beizusteuern. Ich bin sicher, auch noch so unwesentlich erscheinende Dinge rechtfertigen unsere Aufmerksamkeit. Deshalb können wir auch weitere Hilfesysteme zu verschiedensten Bereichen veröffentlichen, die bestimmt immer jemanden erreichen, der genau nach der Lösung gesucht hat.

Schauen Sie mal weiter, es gibt noch mehr...

www.my-vfx.de

Was ist das nu wieder? Sie können es im Internet sehen, Sie können es ohne vorher gesehen zu haben bedienen, denn Sie kennen ja VFX, aber was das wichtigste ist, hier können wir sammeln! Alles was das Herz begehrt - vorzugsweise natürlich alles zu VFX (und VFP).

Weitere Hilfedateien, Slides, Codesnippets, Ressourcen, Tipps und Tricks aus dem VFX-Forum und Lustiges.

Es gibt auch die VFX-Family, da können Sie sich Eintragen - kostenlos mit Bild und Links zu Ihren Websites und E-Mail Adressen.

Wer seine Entwicklungen vorstellen möchte kann das im Ressourceguid machen, natürlich auch kostenlos - naja, fast. Der Preis: ein paar Tipps und Tricks fürs Websitearchiv und mindestens ein Aufsatz in den Rubriken Lösungen und/oder Teillösungen. - Also eigentlich doch nichts.

Schauen Sie doch mal rein - ganz unverbindlich auf:

www.my-vfx.de

Nachwort

An dieser Stelle dank an alle, die mich bei der Erarbeitung dieser Idee unterstützt haben:

- Steven Black — für die Idee an sich - aus technischer Sicht
- Arturo Devigus — für die Idee von VFX
- Uwe Habermann — für seine zahlreichen Hilfestellungen zur Anwendung von VFX
- Michael Schmitz — für die Idee die Kunden doch ihre Reports selber machen zu lassen
- Dave Rooney — für die *iniclass*
- Meiner Familie — für die Extrazeit, die ich im Büro verbringen durfte <g>
- Unserem AuPair — für die Übersetzung der Slides zu diesem Thema (Victoria Wood, Oregon USA)

und...

- Meinem Sohn Joshua — für die Sprachtabellenübersetzung nach Kishuaheli