Session V-FXDB

VFX - Einführung DBC-Anwendungen

Uwe Habermann, Venelina Jordanova

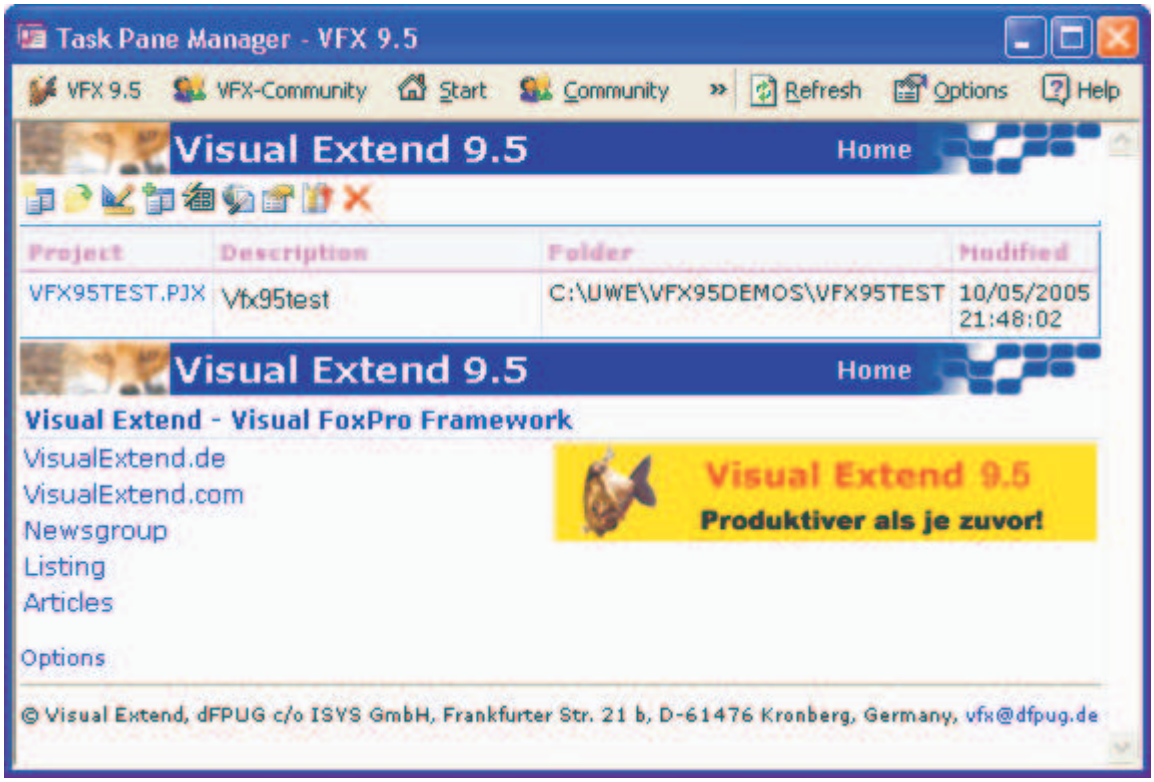
Einführung

Diese Session richtet sich an Interessenten und Neueinsteiger, die Anwendungen basierend auf FoxPro-Tabellen entwickeln wollen.

Wie viel Zeit wird benötigt um eine professionelle Anwendung mit VFP 9 und VFX 9.5 aufzubauen? Hier wird in einer Session eine vollständige, lauffähige Anwendung entwickelt. Dabei werden verschiedene Formulare basierend auf FoxPro-Tabellen generiert. Es werden einfache Formulare zur Datenbearbeitung erstellt, aber auch komplizierte 1:n-Formulare mit Auswahllisten werden mit den VFX Buildern ohne Programmierung erstellt. 1:n:m-Beziehungen werden durch hierarchisch verbundene Formulare dargestellt. Zahlreiche professionelle Features stehen den Endanwendern zur Verfügung.

VFX – Application Wizard

Der VFX - Application Wizard kann aus der VFP heraus aus der Visual Extend Task Pane über eine Schaltfläche in der Symbolleiste gestartet werden oder aus dem VFX-Menü über den Menüpunkt *Projekt, Application Wizard*.

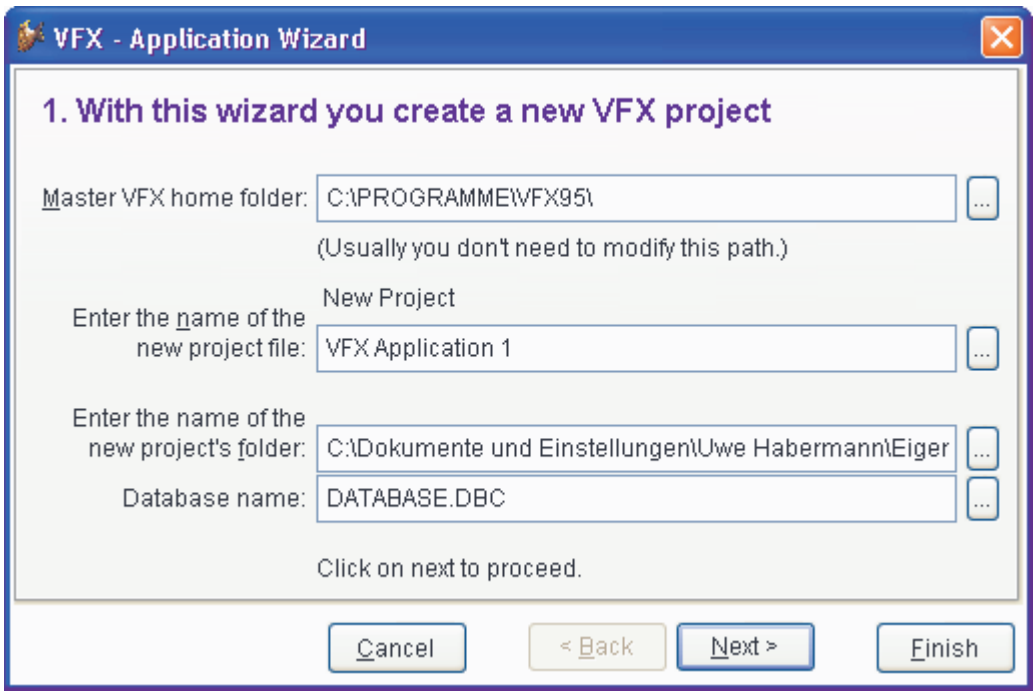


VFX Task Pane

Der VFX – Application Wizard dient dazu ein neues Projekt anzulegen. Der Wizard besteht aus mehreren Schritten. Im ersten Schritt sehen wir den Pfad zur VFX-Installation. Dieser Pfad wird aus der Datei VFX95.ini ausgelesen. Wir geben dem Projekt einen Namen und geben den Pfad an, in dem das Projekt gespeichert werden soll. Nach der Installation von VFX ist dieser Pfad standardmäßig

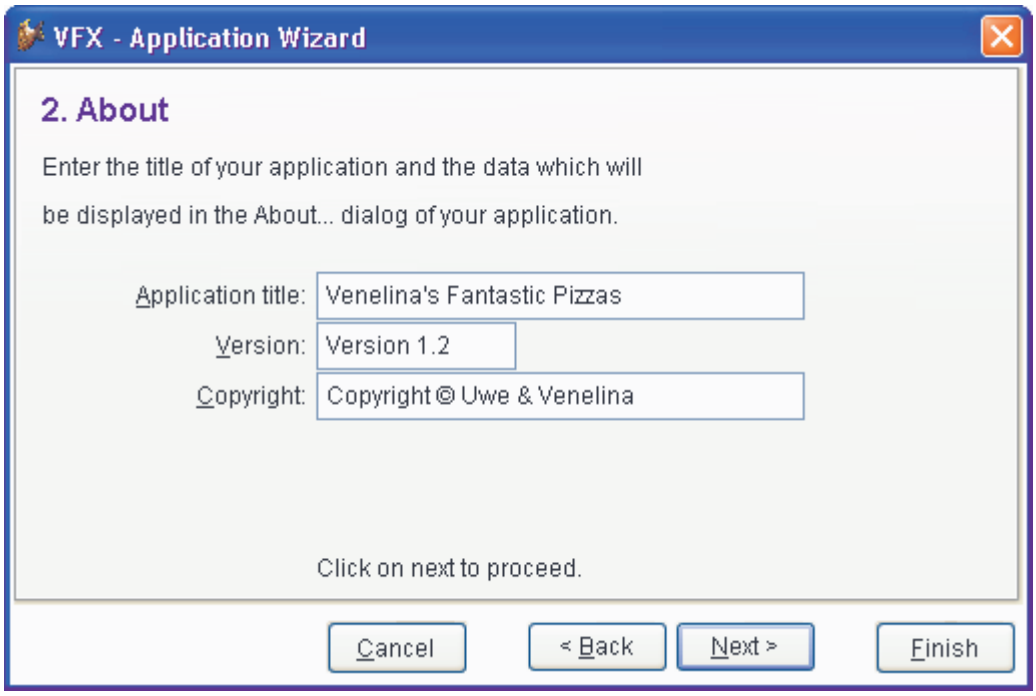
C:\Dokumente und Einstellungen\<Windows Anmeldename>\Eigene Dateien\VFX Projects

VFX ist zertifiziert als Certified for Windows XP. Um die Anforderungen für dieses Logo zu erfüllen, müssen neue Projekte in diesem Pfad gespeichert werden. Der Pfad kann jedoch beliebig geändert werden und VFX merkt sich diesen Pfad für weitere Projekte.



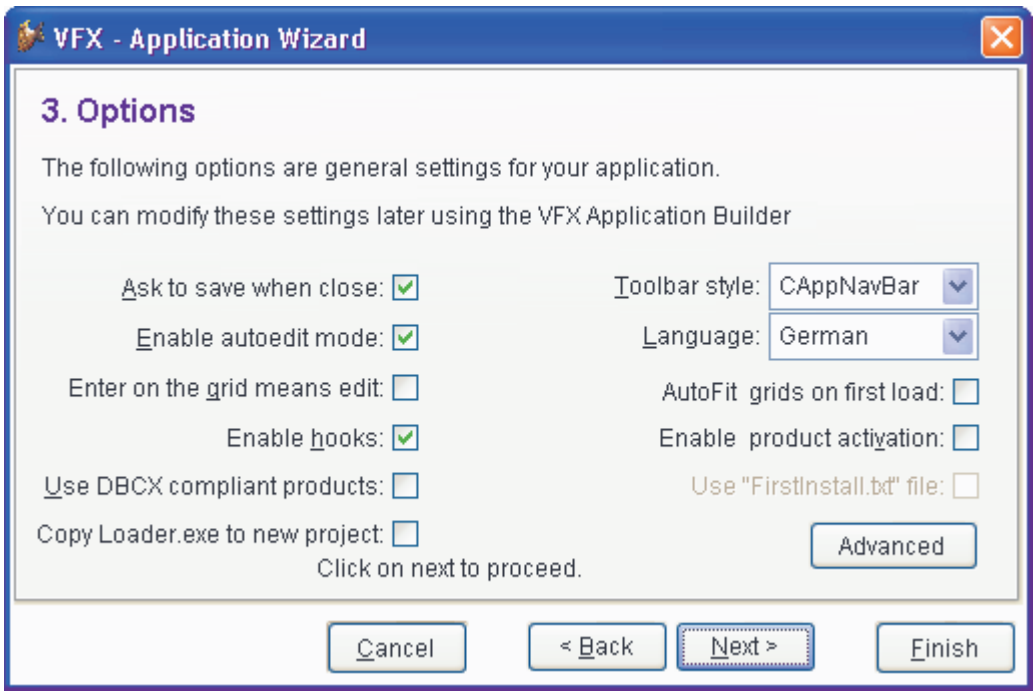
VFX – Application Wizard, Schritt 1

Auch der Name einer Datenbank kann eingegeben werden. Wenn eine Anwendung erstellt werden soll, die ausschließlich mit Remote Datenquellen arbeitet, kann man dieses Feld auch leer lassen.



VFX – Application Wizard, Schritt 2

Im nächsten Schritt des Application Wizard machen wir die Einstellungen für den Info-Dialog der Anwendung. Die fertige VFX-Anwendung wird einen Info-Dialog haben, wie wir ihn aus anderen Windows-Anwendungen kennen. Der Info-Dialog ist zur Laufzeit über den Menüpunkt *Hilfe, Info* erreichbar. Es müssen der Name der Anwendung, die Versionsnummer und ein Copyright-Vermerk eingegeben werden.



VFX – Application Wizard, Schritt 3

Auf der Seite Optionen können verschiedene Optionen zur Anwendung eingestellt werden. Diese Optionen bestimmen das Verhalten der Anwendung zur Laufzeit. Insbesondere kann hier die Sprache ausgewählt werden und es kann eingestellt werden, ob die in VFX integrierte Produktaktivierung verwendet werden soll.

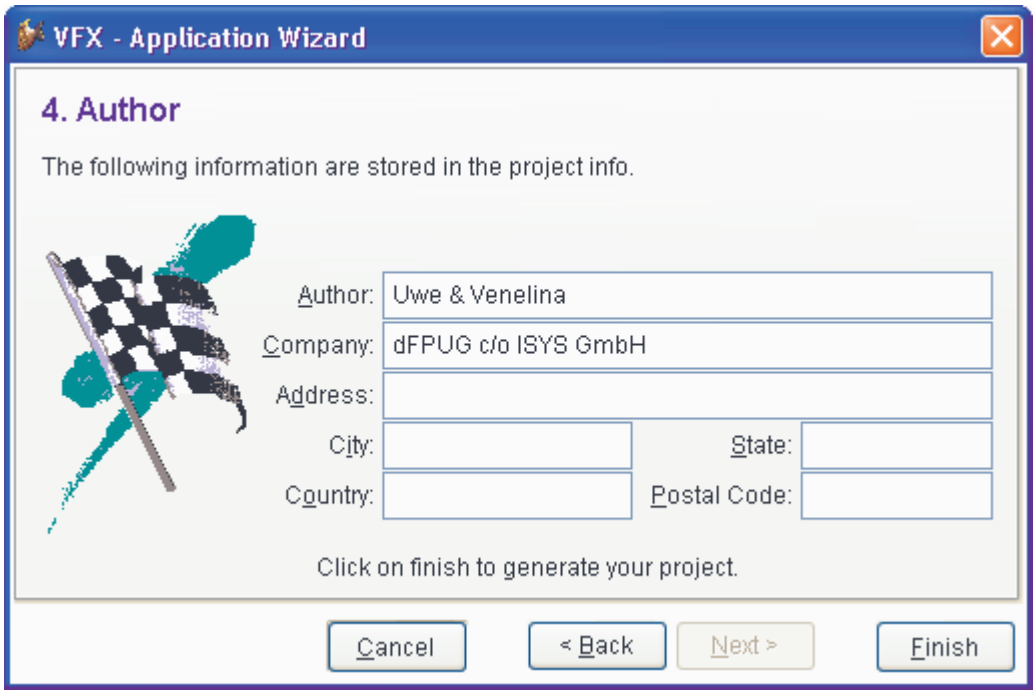
Die Optionen im Einzelnen:

- **Ask to save when close** – Wie soll sich die Anwendung verhalten, wenn ein Formular geöffnet ist, in dem Änderungen gemacht worden sind, und der Benutzer versucht das Formular zu schließen? Soll eine Frage erscheinen, ob die Änderungen gespeichert werden sollen oder soll beim Schließen des Formulars implizit gespeichert werden.
- **Enable autoedit mode** – Alle Steuerelemente auf Formularen sind standardmäßig enabled. Der Benutzer kann also mit der Maus oder mit der Tastatur jedes Steuerelement erreichen und sofort Änderungen an den Daten durchführen. Wenn diese Option nicht ausgewählt ist, sind standardmäßig alle Steuerelemente disabled und der Benutzer muss über die Tastenkombination *Strg+E* oder über den Menüpunkt *Bearbeiten, Bearbeite Datensatz* oder das Symbol *bearbeiten* in der Symbolleiste das Formular in den Bearbeitungsmodus umschalten. Dabei werden dann alle Steuerelemente auf enabled umgeschaltet und der Benutzer kann dann Änderungen machen.
- **Enter on the grid means edit** – Wenn diese Option ausgewählt ist und die Suchseite eines Formulars aktiv ist und der Benutzer die Eingabetaste drückt, wechselt das Formular zur ersten Bearbeitungsseite und schaltet in den Bearbeitungsmodus. Wenn diese Option nicht ausgewählt ist, wechselt das Formular beim Betätigen der Eingabetaste zur ersten Bearbeitungsseite. Das Formular bleibt jedoch im Ansichtsmodus.
- **Enable hooks** – Dies ist ein sehr leistungsfähiges Feature von VFX, mit dem wir die Möglichkeit haben, in den Ablauf von VFX-Methoden Einfluss zu nehmen. Hooks werden ausgeführt, während VFX den Code einer Methode wie *Init* oder *OnSave* ausführt. Mit Hooks können wir global, an zentraler Stelle Änderungen am Verhalten von VFX-Anwendungen vornehmen.
- **Use DBCX compliant products** – Diese Option braucht nur dann ausgewählt werden, wenn im Projekt ein DBCX-kompatibles Produkt, zum Beispiel der Stonefield Database Toolkit, eingebunden werden soll. VFX unterstützt die Einbindung von SDT. Wenn diese Option ausgewählt ist, müssen zusätzlich einige Dateien aus dem SDT in das Projekt eingebunden werden. Das Verhalten der Anwendung ändert sich insoweit, dass die Funktionen von SDT für die Datenbankwartung, die Datenbankreparatur sowie für die Aktualisierung der Datenbank beim Kunden verwendet werden.
- **Copy Loader.exe to new project** – Diese Option muss nur dann ausgewählt werden, wenn die Aktualisierung der Anwendung über das Internet verwendet werden soll und wenn das mit VFX gelieferte Loader-Projekt hierfür verändert werden soll. Wenn eine neue Version der Anwendung aus dem Internet herunter geladen wurde, muss die laufende Anwendung gegen die neue Version ausgetauscht werden. Das geschieht mit der Loader.exe. VFX liefert hierfür ein fertiges Projekt, das diese Funktion

übernimmt. Wer die Funktion der Loader-Anwendung erweitern möchte, kann das Loader-Projekt in einen Ordner seines Projekts kopieren und dort verändern. Änderungen im Loader-Projekt werden wohl nur in seltenen Fällen erforderlich sein.

- **Toolbar style** – VFX kommt mit zwei verschiedenen Layouts für die Standard-Symbolleiste der Anwendung. Die Klasse *CAppNavBar* ist eine Symbolleiste, die auch die Navigation innerhalb eines Formulars ermöglicht. Darin gibt es also Schaltflächen um an den Anfang einer Datei, auf den vorhergehenden Datensatz, auf den nächsten Datensatz oder auf den letzten Datensatz einer Datei zu springen. Die Klasse *CAppToolbar* enthält diese Navigationsschaltflächen nicht.
- **AutoFit grids on first load** – Wenn diese Option ausgewählt ist, wird das AutoFit-Feature von VFP Grids beim erstmaligen Laden von Grids verwendet. Dabei ist darauf zu achten, dass das AutoFit bei der Initialisierung von Grids ausgeführt wird. Dabei werden für das AutoFit die Daten verwendet, die in diesem Moment sichtbar sind. Wenn ein Formular auf Ansichten oder Cursoradaptern als Datenquelle basiert und diese erst zur Laufzeit des Formulars mit Daten gefüllt werden, sind beim Laden des Formulars noch keine Daten sichtbar. In diesem Fall wird das AutoFit also nur für die Überschriften im Grid durchgeführt.
- **Enable product activation** – Hier kann die in VFX integrierte Produktaktivierung für diese Anwendung eingeschaltet werden.
- **Advanced** – Hier kann der VFX – Application Builder gestartet werden. Hier können noch sehr viel mehr Einstellungen gemacht werden, die das Verhalten der Anwendung beeinflussen. Alle Einstellungen im Application Wizard können später über den VFX - Application Builder geändert werden.

Im letzten Schritt des Application Wizard werden die Informationen zum Projekt gespeichert. Diese Informationen werden in der Projektdatei gespeichert. Diese Angaben haben keinen Einfluss auf die Funktionalität der Anwendung.



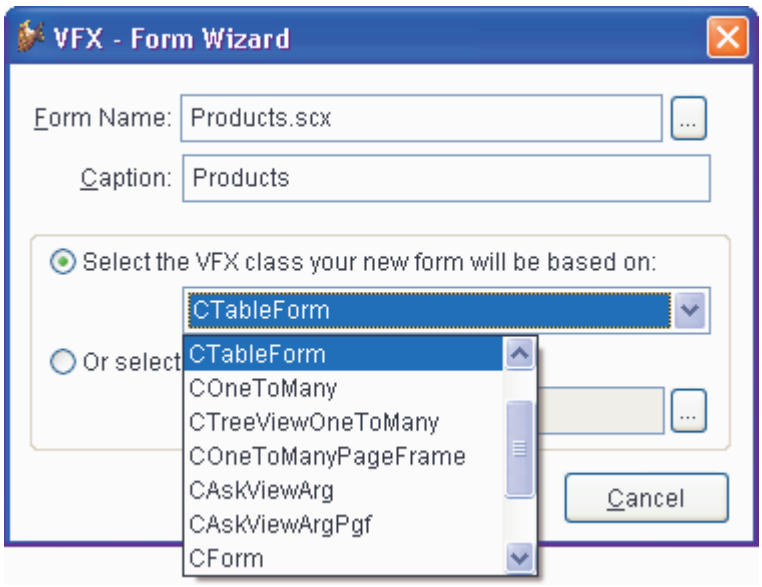
VFX – Application Wizard, Schritt 4

Damit sind alle Eingaben im Application Wizard gemacht und das neue Projekt kann generiert werden. Auf der letzten Dialogseite wird nach dem Klick auf *Finish* das VFX-Musterprojekt in den gewählten Projektordner kopiert und es werden die gewünschten Einstellungen gemacht. Anschließend werden alle Dateien des Projekts kompiliert.

Wir haben damit den Rahmen für unsere Anwendung geschaffen und müssen uns jetzt dem Datenbank-Design zuwenden. Wenn eine VFP-Datenbank verwendet werden soll, kann die Datenbank mit dem VFP-Datenbank-Designer erstellt und bearbeitet werden. Insbesondere beim Einsatz von Remote Datenbanken bietet sich aber der Einsatz von Zusatzprodukten, wie xCase, SDT oder auch der SQL Server Enterprise Manager an.

VFX - CTableForm Builder

Nun wollen wir unser erstes Formular mit VFX erstellen. Dafür verwenden wir den VFX – CTableForm Builder. Benutzer von früheren VFX-Versionen wissen, dass man ein VFX-Formular mit dem VFX – Form Wizard anlegen kann. Im nächsten Schritt musste man manuell mit dem VFP Formular-Designer die Datenumgebung des Formulars einrichten. Anschließend musste mit dem VFX – Form Builder das Formular mit Steuerelementen gefüllt und fertig gestellt werden. Diese ganzen Arbeitsschritte sind in neueren VFX-Versionen in einem Builder vereint.



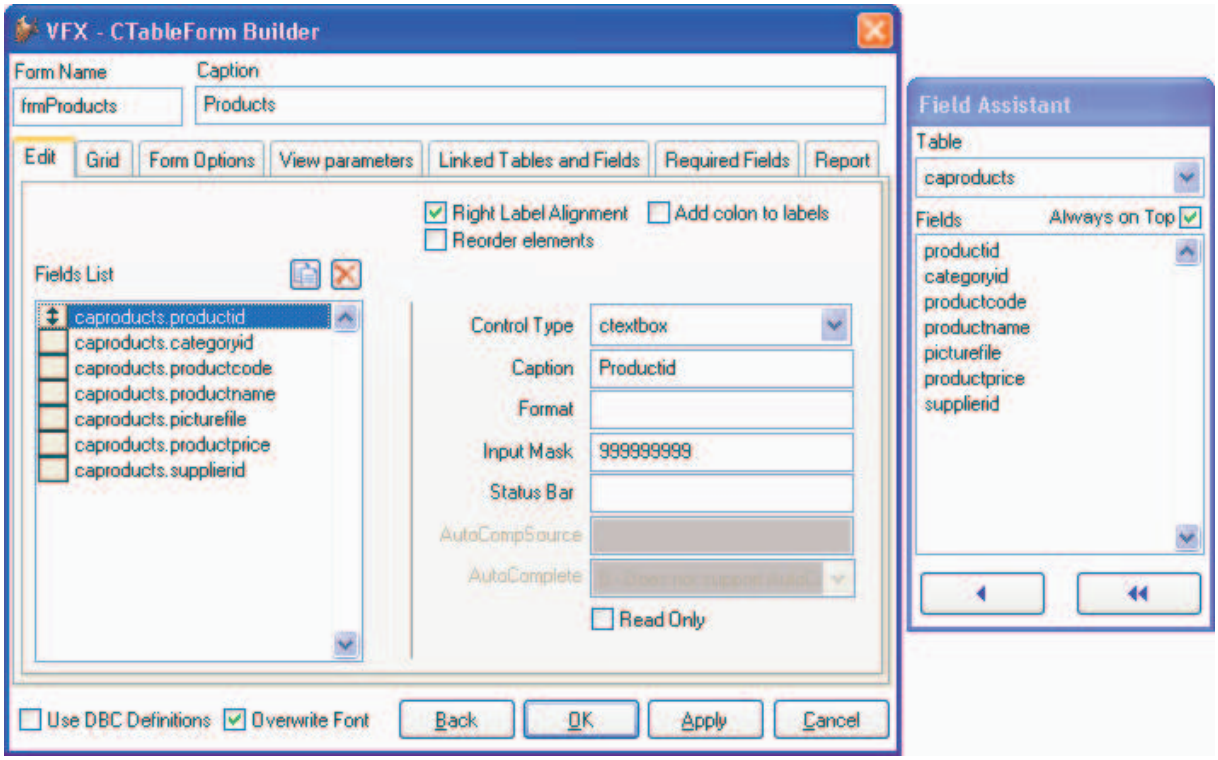
VFX – Form Wizard

Nach wie vor starten wir aus dem VFX-Menü den Form Wizard. Dort wird dem Formular ein Name gegeben und es wird die für dieses Formular zu verwendende Klasse ausgewählt. Fortgeschrittene Entwickler können eigene Ableitungen von den VFX-Formularklassen bilden und die Funktionalität erweitern oder verändern. Auch eigene abgeleitete Formularklassen werden von den VFX Buildern unterstützt. Das Formular wird automatisch im Ordner *Form* unterhalb des aktuellen Projekts gespeichert.

Im nächsten Schritt erscheint der VFX – Data Environment Builder, mit dem die Datenumgebung für das Formular erstellt wird. Dem Formular können beliebige Datenquellen, also Tabellen, Ansichten und Cursoradapter hinzugefügt werden. Es können temporäre Indexdateien für CursorAdapter erstellt werden und es können Relationen definiert werden. Alle Einstellungen im VFX – Data Environment Builder sind kompatibel zu den Einstellungen in der Datenumgebung des Formular-Designers von VFP. Die wesentlichen Vorteile des VFX – Data Environment Builder werden in der Session V-FXCS gezeigt, in der CursorAdapter als Datenquelle für Formulare verwendet werden.

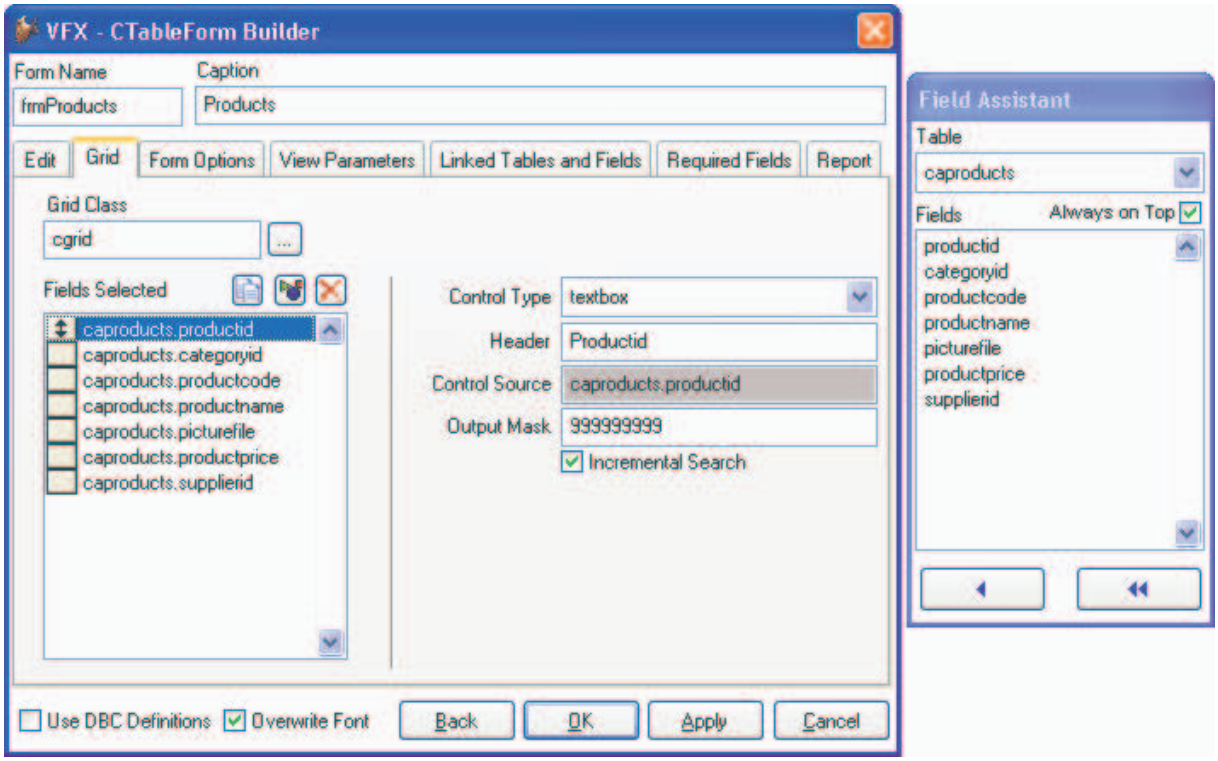
Um ein einfaches Formular basierend auf der Klasse *CTableForm* erstellen zu können, reicht eine Tabelle als Datenquelle aus. Im VFX – Data Environment Builder können auch Filter eingestellt werden und es kann ein Aliasname festgelegt werden. Nach einem Klick auf *Next* erscheint der Form Builder, der auf vielen Seiten Optionen und Einstellmöglichkeiten zu den einzelnen Formularen bietet.

Die Klasse *CTableForm* enthält ein Grid, in dem der Benutzer nach Daten suchen kann, sowie einen freien Platz, in dem der Form Builder Steuerelemente zur Bearbeitung von Daten platziert.



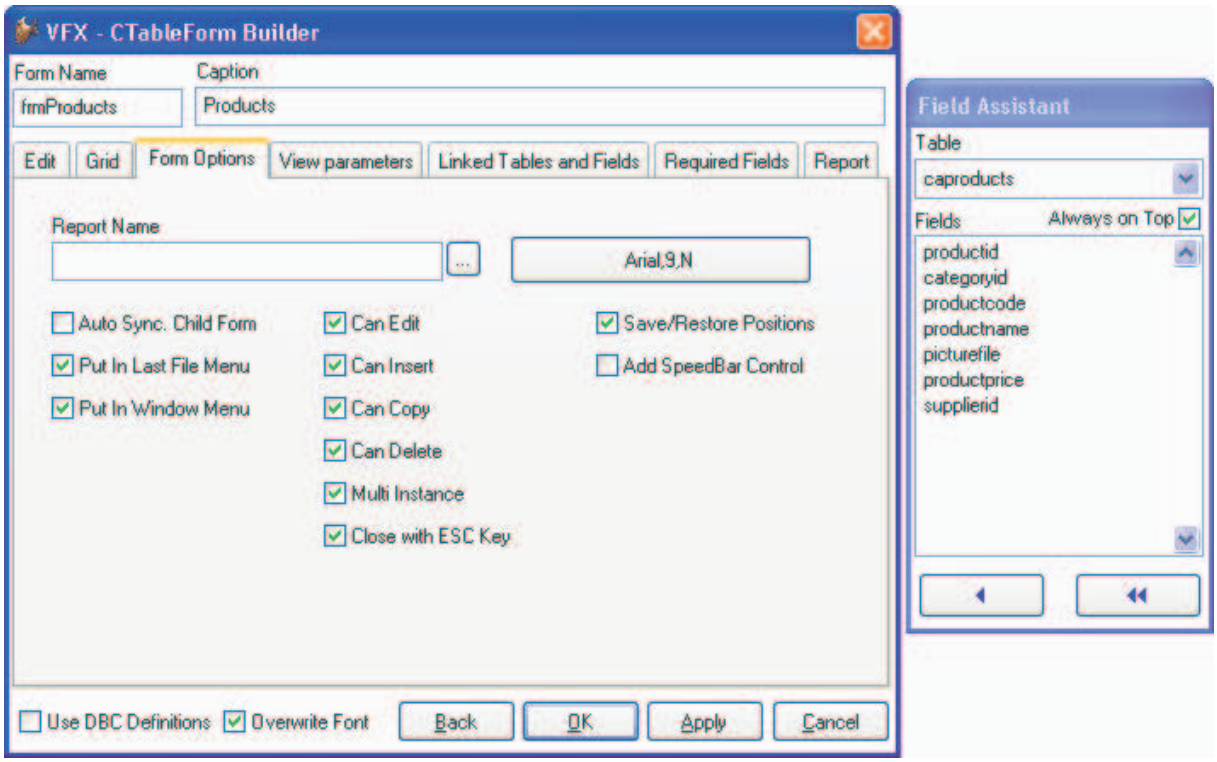
VFX – CTableForm Builder, Edit Page

- **Edit**– Hier können dem Formular Steuerelemente zur Dateneingabe für Benutzer hinzugefügt werden. Im Field Assistant stehen alle Datenquellen aus der Datenumgebung zur Verfügung. Die Felder, für die Steuerelemente erstellt werden sollen, können im Field Assistant ausgewählt werden. Dabei werden für jedes Steuerelement Eigenschaften, wie zum Beispiel Caption oder Format aus dem Datenbank-Container gelesen und die Eigenschaften des generierten Steuerelements übernommen. Für jedes Steuerelement können die wichtigsten Eigenschaften im Form Builder eingestellt werden. In der Regel brauchen aber keine Eigenschaften verändert werden und es kann alles bei den Standardeinstellungen von VFX bleiben.



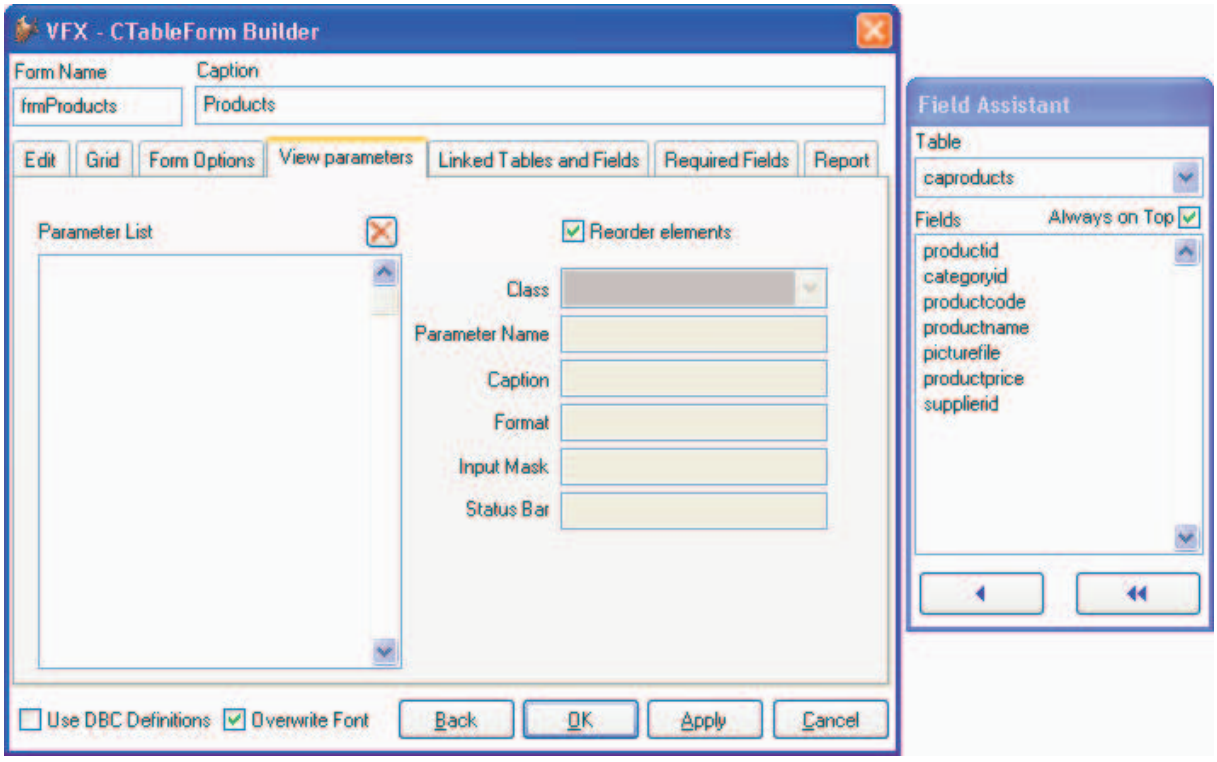
VFX – CTableForm Builder, Grid Page

- **Grid** – Hier kann das Such-Grid erstellt werden. Genau wie auf der Seite *Edit* können Felder aus dem Field Assistant ausgewählt werden. Für jedes ausgewählte Feld wird eine Spalte im Grid angelegt. Für jede Grid-Spalte stellt VFX standardmäßig eine inkrementelle Suche zur Verfügung. Für Spalten, in denen der Benutzer nicht inkrementell suchen soll, kann diese Option im Builder ausgeschaltet werden.



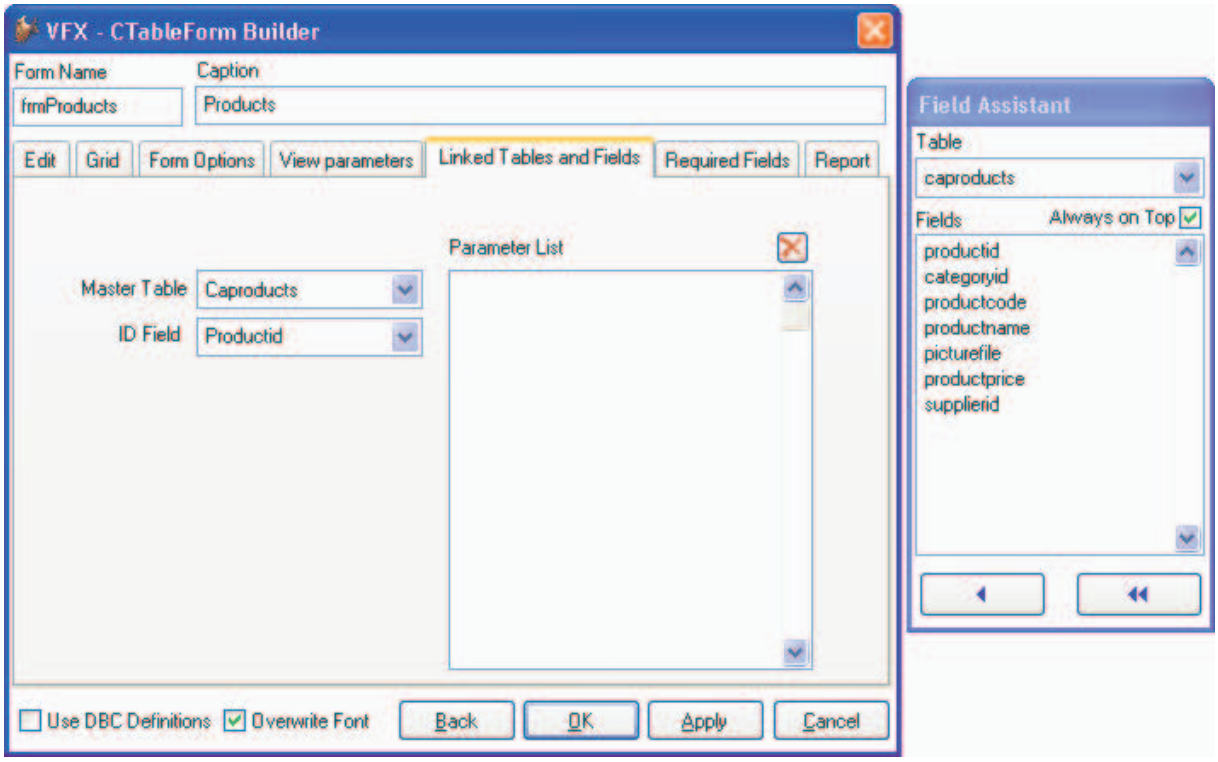
VFX – CTableForm Builder, Form Options Page

- **Form Options** – Hier können Einstellungen für Child-Formulare gemacht werden und es können die Bearbeitungsmöglichkeiten für den Benutzer festgelegt werden.



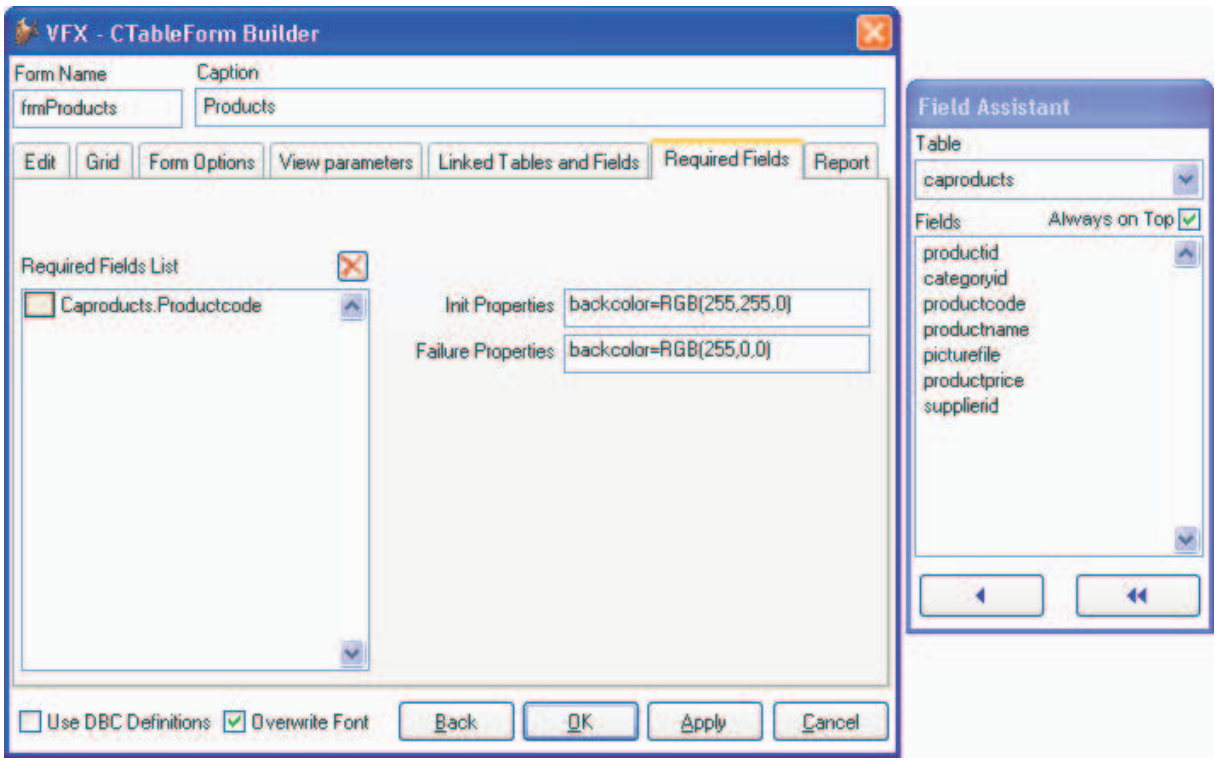
VFX – CTableForm Builder, View Parameters Page

- **View Parameters** – Hier können dem Formular vom Builder automatisch Steuerelemente hinzugefügt werden, die zur Laufzeit zur Eingabe von Parametern für Ansichten oder Cursoradapter dienen. Wenn in den Ansichten oder Cursoradaptern aus der Datenumgebung Where-Klauseln verwendet werden und dort Werte für Parameter eingegeben werden müssen, können wir die erforderlichen Steuerelemente vom Builder dem Formular hinzufügen lassen. Auch hier können Felder aus dem Field Assistant ausgewählt werden. Auch zu diesen Feldern können viele Eigenschaften direkt im Builder eingestellt werden. Zu diesem Thema wird in der Session V-FXCS mehr gezeigt.



VFX – CTableForm Builder, Linked Tables and Fields Page

- **Linked Tables and Fields** – VFX bietet eine automatische Verwaltung von 1:1-Beziehungen Tabellen an. Es kann Situationen geben, in denen solche 1:1-Beziehungen sinnvoll sind, zum Beispiel wenn in einer Tabelle mehr als 255 Felder verwaltet werden sollen. VFP unterstützt nicht mehr Felder in einer Tabelle, sodass dann mit einer 1:1-Beziehung gearbeitet werden muss. Aber es gibt auch andere sinnvolle Anwendungsfälle für 1:1-Beziehungen. Voraussetzung für die Verwendung in VFX ist, dass die Haupttabelle einen Primärschlüssel hat, und dass alle an der Beziehung beteiligten Tabellen ein entsprechendes Schlüsselfeld haben. Genau diese Einstellung wird hier im Form Builder gemacht. Bei der Anlage von neuen Datensätzen zur Laufzeit legt die VFX-Anwendung automatisch auch in den in Beziehung stehenden Tabellen neue Datensätze an. Wenn Datensätze in der Haupttabelle gelöscht werden, werden automatisch auch die Datensätze in den in Beziehung stehenden Tabellen gelöscht.

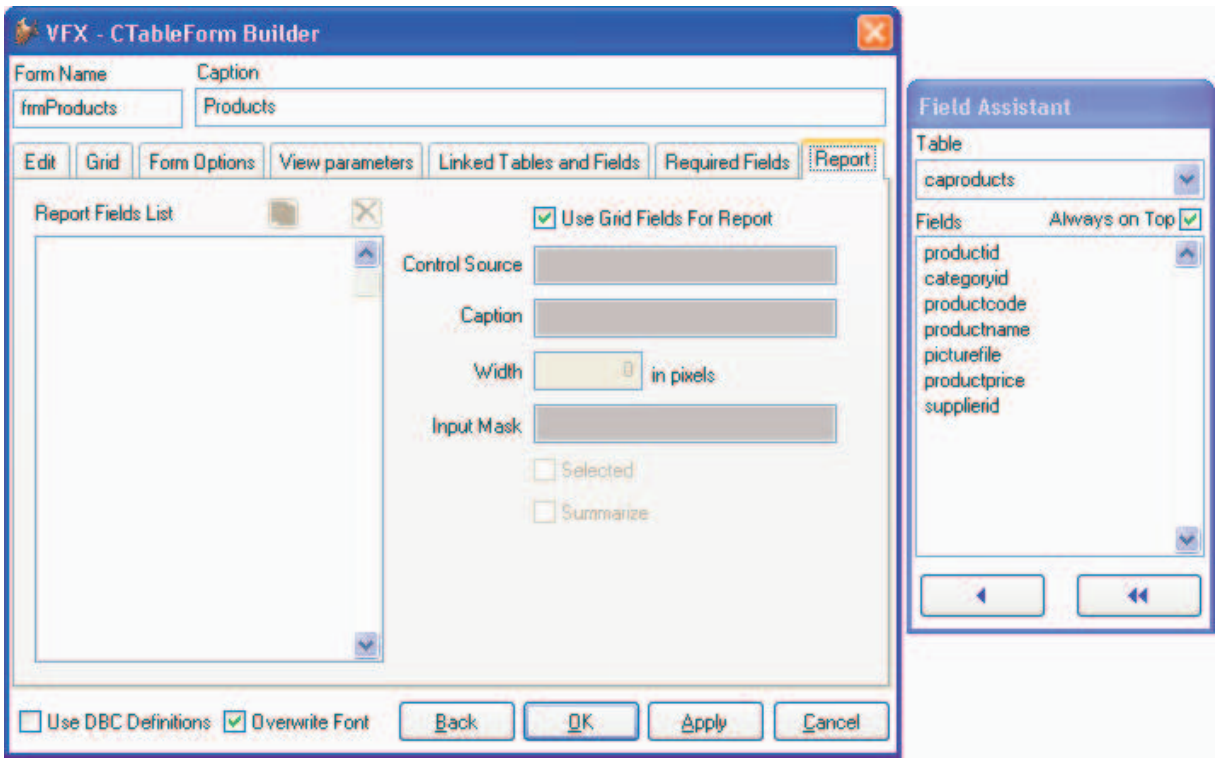


VFX – CTableForm Builder, Required Fields Page

- **Required Fields** – Hier können Pflichteingabefelder festgelegt werden, also Felder, in denen der Benutzer eine Eingabe machen muss, bevor die Daten gespeichert werden können. Solange mindestens ein Pflichteingabefeld leer ist, können die Daten nicht gespeichert werden. Die Pflichteingabefelder können durch verschiedene Eigenschaften gekennzeichnet werden, um dem Benutzer zu visualisieren, dass hier Werte eingegeben werden müssen. Wenn der Benutzer zur Laufzeit diese Felder leer lässt und versucht zu speichern, können die leeren Pflichteingabefelder besonders gekennzeichnet werden, um dem Benutzer anzuzeigen, dass er hier eine Eingabe vergessen hat. Es können mehrere Eigenschaften in einer durch Semikolon separierten Liste eingegeben werden.
  Beispiel:

  Init Properties `backcolor=RGB(255,255,0)  && gelbe Hintergrundfarbe`

  Failure Properties `backcolor=RGB(255,0,0)  && rote Hintergrundfarbe`

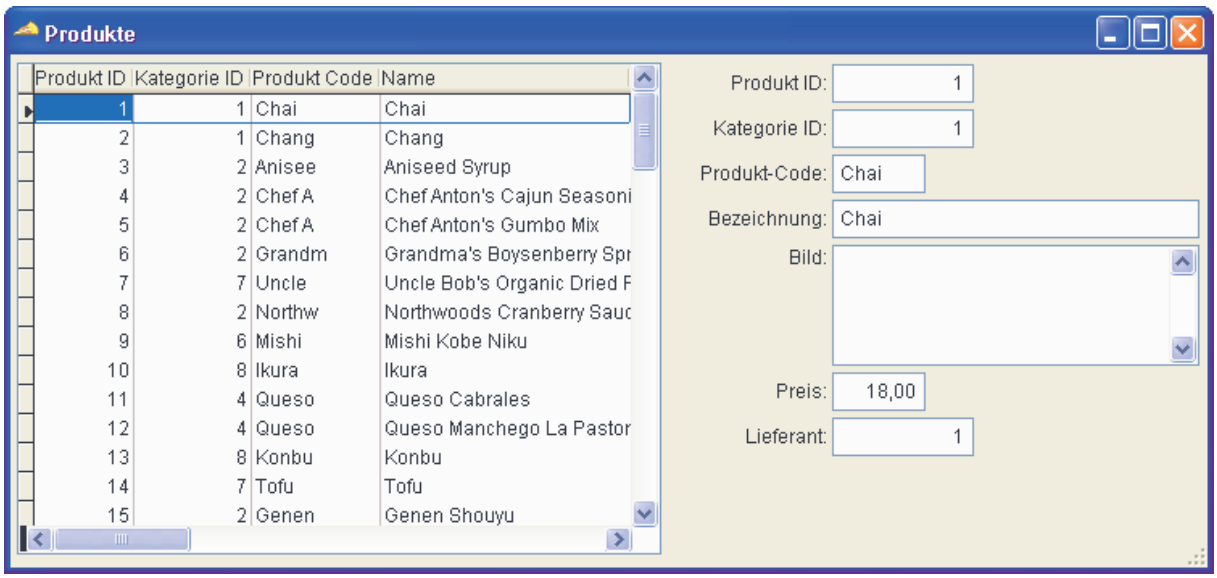


VFX – CTableForm Builder, Report Page

- **Report** – Auf der letzten Seite des Builders kann eingestellt werden, wie Berichte zur Laufzeit erstellt werden sollen. Standardmäßig kann VFX für jedes Formular zur Laufzeit einen Bericht erstellen, der auf den Spalten des Such-Grids basiert. Wenn das nicht gewünscht ist, können hier die Felder ausgewählt werden, die auf dem Bericht verwendet werden sollen. Wenn hier individuelle Spalten eingestellt werden, können zu den einzelnen Spalten noch einige Eigenschaften mit Werten vorbelegt werden. Zum Beispiel kann die Breite der Spalten eingestellt werden oder bei numerischen Feldern kann eine Summierung hinzugefügt werden. All diese Einstellungen sind nur Voreinstellungen, die vom Benutzer zur Laufzeit geändert werden können.
  Die im Bericht angezeigten Daten entsprechen genau der Anzeige im Grid. Filter- und Sortiereinstellungen werden berücksichtigt.

  Wenn nur einzelne Felder für die Erstellung von Berichten verwendet werden, trägt der Form Builder Code in der Methode *CreateReportFieldList* des Formulars ein. Hier werden die Spalten definiert, die im Bericht erscheinen sollen. Bei Bedarf kann dieser Code auch manuell angepasst werden.

Durch einen Klick auf die Schaltfläche *OK* wird das Formular generiert. In einer Messagebox wird angezeigt, dass das neue Formular in die Tabelle *Vfxfopen.dbf* eingetragen wurde. In der fertigen Anwendung wird ein Öffnen-Dialog angezeigt, der auf der Tabelle *Vfxfopen.dbf* basiert. Der Form Builder fügt dieser Tabelle automatisch einen Datensatz hinzu, der es zur Laufzeit ermöglicht das Formular zu starten.



Das Formular Produkte zur Laufzeit

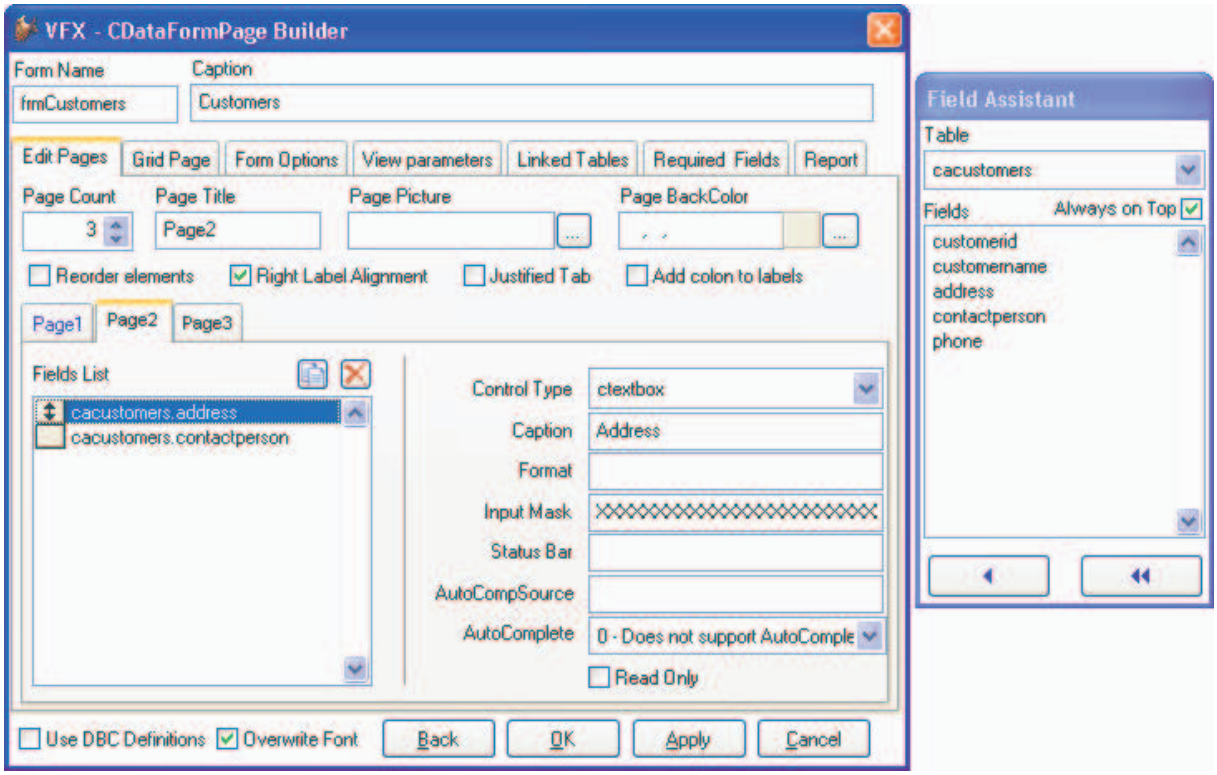
Das generierte Formular kann jetzt gestartet werden. Dazu starten wir aus dem Projekt-Manager *Vfxmain.prg*. Es erscheint der Begrüßungsbildschirm, gefolgt vom Anmeldedialog. In der nun laufenden Anwendung kann durch einen einfachen Mausklick auf einen Eintrag im Öffnen-Dialog ein Formular gestartet werden.

So lassen sich ohne Programmierung mit dem VFX – CTableForm Builder Formulare mit vielen guten Eigenschaften generieren. Pflichteingabefelder werden, wie im Builder eingestellt, zur Laufzeit gekennzeichnet.

Alle VFX – Form Builder sind reentrant. Die Builder können für ein Formular beliebig oft aufgerufen werden, zum Beispiel um Steuerelemente hinzuzufügen oder zu entfernen oder um Einstellungen zu verändern. Auf diesem Weg kann dem Formular auch nachträglich eine Speedbar hinzugefügt werden. Eine Speedbar ist eine Gruppe von Schaltflächen auf einem Formular, die ähnliche Funktionen, wie die Standard-Symbolleiste zur Verfügung stellen. Beim Hinzufügen einer Speedbar wird die Größe des Formulars automatisch so angepasst, dass die Speedbar ihren Platz am oberen Formularrand findet.

VFX – CDataFormPage Builder

Wir wollen jetzt ein weiteres Formular aufbauen, diesmal basierend auf der Klassen *CDataFormPage*. Der VFX - CDataFormPage Builder bietet so ähnliche Funktionen, wie der VFX – CTableForm Builder. Der Aufruf erfolgt ebenfalls aus dem VFX-Menü über den Menüpunkt Form, Form Wizard. Nachdem wir dem neu zu erstellenden Formular einen Namen gegeben haben, können wir gleich auf *Next* klicken. Die Klasse *CDataFormPage* ist der Standardwert in der Combobox für die Formularklasse. Anschließend wird auch hier der VFX – Data Environment Builder gestartet. Wir fügen dem Formular eine Tabelle als Datenquelle hinzu. Im nächsten Schritt erscheint der eigentliche VFX – CDataFormPage Builder.



VFX – CDataFormPage Builder mit mehreren Bearbeitungsseiten

- **Edit** – Im Gegensatz zum VFX – CTableForm Builder, der die Erstellung von Steuerelementen neben einem Grid erlaubt, können hier auf einem Seitenrahmen beliebig viele Seiten hinzugefügt werden und auf jeder Seite können beliebig viele Steuerelementen verschiedenen Typs hinzugefügt werden. Wenn die Anzahl der Seiten im Builder erhöht wird, erhöht sich auch die Anzahl der Seiten im dem darunter angezeigten Seitenrahmen, dem die Steuerelemente hinzugefügt werden können. Jeder Seite auf dem Seitenrahmen kann im Builder eine Bezeichnung gegeben werden.
- **Grid** – Wie auf Formularen basierend auf der Klasse *CTableForm* kann auch hier dem Formular ein Such-Grid hinzugefügt werden. Standardmäßig ist in allen Spalten eine inkrementelle Suche möglich. Bei Bedarf erstellt VFX benötigte Indexdateien temporär zur Laufzeit.

Alle anderen Einstellungen im VFX – CDataFormPage Builder sind genauso zu machen, wie im VFX – CTableForm Builder.

Das erstellte Formular kann direkt aus dem Projekt-Manager gestartet und getestet werden.

VFX – COneToMany Builder

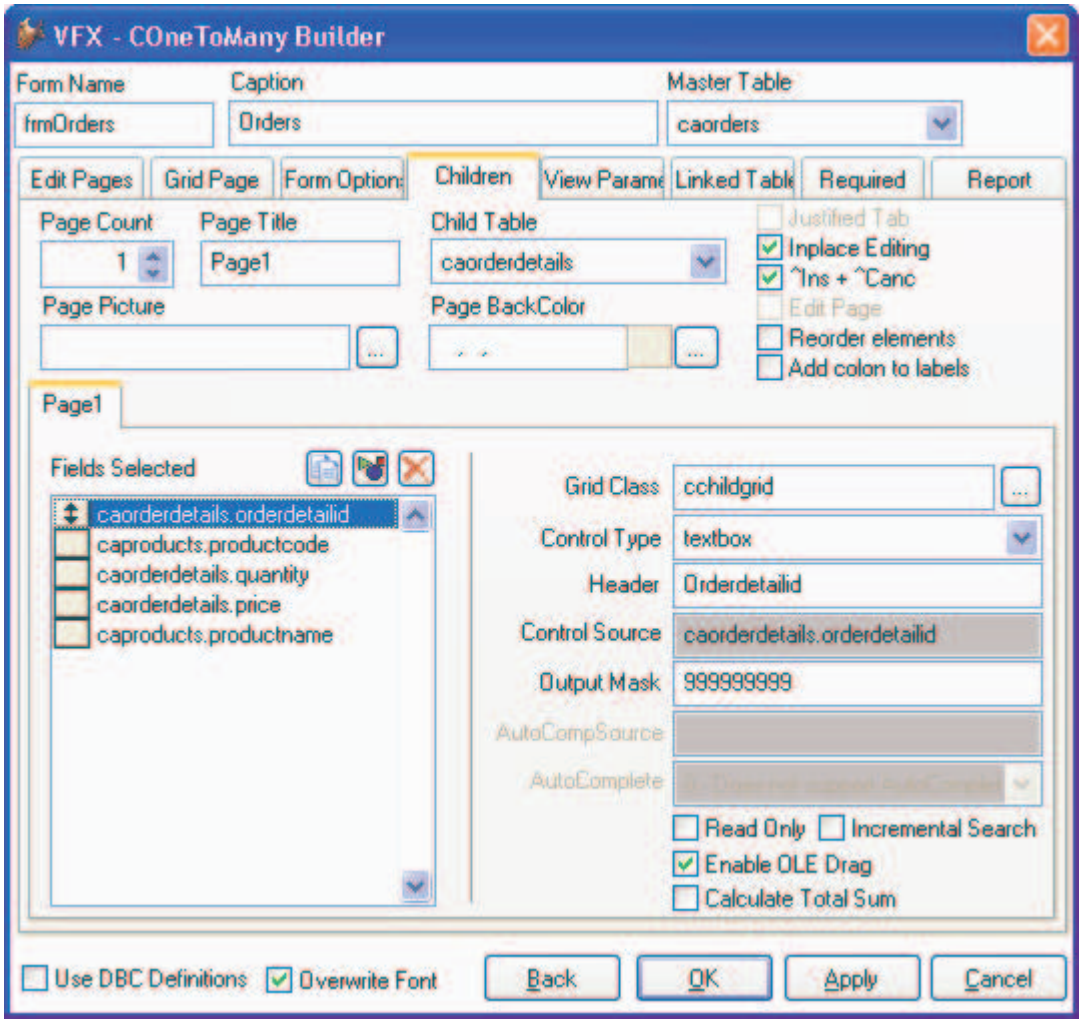
Nachdem wir zwei relativ einfache Formulare erstellt haben, wollen uns nun einem etwas komplizierteren Formular zuwenden. Die zuerst erstellten Formulare haben nur auf jeweils einer einzigen Tabelle basiert. Jetzt wollen wir eine 1:n-Beziehung in einem Formular mit zwei Tabellen darstellen. Die Formularklasse *COneToMany* sieht im oberen Teil genauso aus, wie die Klasse *CDataFormPage*, die wir schon kennen. Unterhalb des Seitenrahmens befindet sich ein Child-Teil, der ebenfalls einen Seitenrahmen enthält. Auf den einzelnen Seiten dieses Seitenrahmens werden normalerweise Grids hinzugefügt, in denen der Benutzer zur Laufzeit Daten bearbeiten kann. VFX unterstützt auf OneToMany-Formularen mehrere Children. Man kann auf mehreren Seiten des Child-Seitenrahmens verschiedene Children bearbeiten oder auch verschiedene Felder des gleichen Childs bearbeiten. In den Child-Seiten können die Child-Daten wahlweise in einem Grid oder in beliebigen anderen Steuerelementen, zum Beispiel Textbox, Editbox, Spinner usw. bearbeitet werden.

Für ein 1:n-Formular werden zwei Datenquellen benötigt. Im Beispiel wird ein Auftragsformular erstellt, das im Parent-Teil die Tabelle *Orders.dbf* und im Child-Teil die Tabelle *Orderdetails.dbf* enthält.

Auch der VFX – COneToMany Builder wird aus dem VFX-Menü mit dem Menüeintrag Form, Form Wizard gestartet. Nachdem wir dem Formular den Namen *Auftrag* gegeben haben, wählen wir aus der Combobox die Formularklasse *COneToMany* aus. Im nächsten Schritt erscheint der VFX – Data Environment Builder. Schon in diesem Moment ist das generierte Formular im Hintergrund im VFP Formular-Designer zu sehen.

Im VFX – Data Environment Builder werden die Tabellen *Orders.dbf* für die Auftragsköpfe und *Orderdetails.dbf* für die Auftragspositionen hinzugefügt. Dabei wird automatisch die in der Datenbank definierte Relation zwischen den Tabellen in den Data Environment Builder übernommen. Die Relation zeigt vom Primärschlüsselfeld der Parent-Tabelle zum dazugehörigen Indexschlüssel in der Child-Tabelle.

Im nächsten Schritt erscheint der VFX – COneToMany Builder. Dabei haben wir weitgehend die gleichen Einstellungen zur Verfügung, die wir schon vom VFX – CDataFormPage Builder her kennen. Zusätzlich haben wir jedoch die Möglichkeit die Bearbeitungsmöglichkeiten der Child-Daten einzustellen.



VFX – COneToMany Builder, Children

Zunächst werden genau wie im VFX – CDataFormPage Builder die Bearbeitungsseiten und die Grid-Seite des Formulars aufgebaut. Anschließend aktivieren wir die Seite *Children* im Builder.

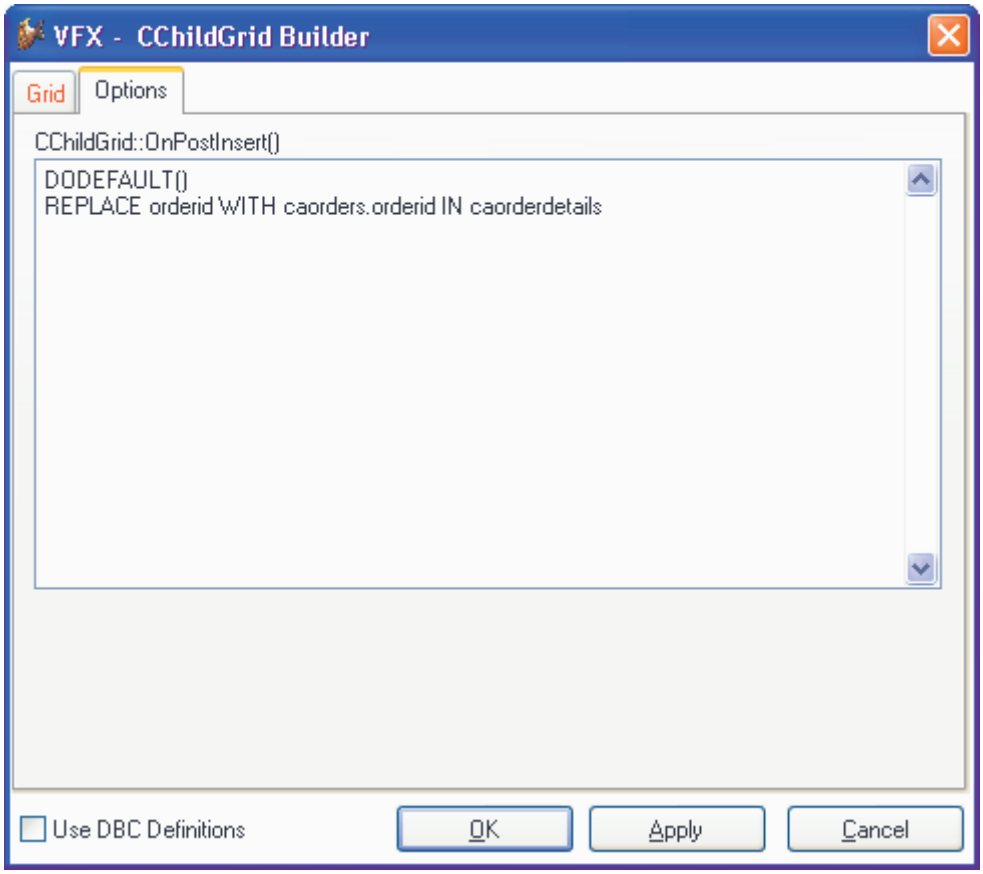
- **Children** – Hier wird der untere Teil des Formulars, der Child-Teil aufgebaut. Zunächst muss in der Combobox *Child Table* ausgewählt werden, welcher Cursor aus der Datenumgebung die Datenquelle für die Child-Daten sein soll. In unserem Beispiel ist dies die Tabelle *Orderdetails.dbf*. Sobald die Tabelle in der Combobox ausgewählt ist, wird diese Tabelle auch im Field Assistant mit allen Feldern angezeigt. Genau wie bei anderen Bearbeitungsseiten können hier Felder ausgewählt werden.

Die weiteren Seiten im VFX – COneToMany Builder sind genauso aufgebaut, wie wir es bereits vom CDataFormPage Builder kennen. Bei einem Klick auf *OK* im Builder wird das Formular generiert und automatisch auch in der Tabelle *Vfxfopen.dbf* eingetragen und kann damit zur Laufzeit über den Öffnen-Dialog gestartet werden. Das Formular ist jetzt im VFP Formular-Designer zu sehen. Bei Bedarf kann das Formular hier noch manuell nachbearbeitet werden. Nach dem Speichern kann das Formular getestet werden.

Wenn in einem OneToMany-Formular neue Child-Datensätze erfasst werden sollen, muss in jedem Child-Datensatz die ID des Parent-Datensatzes gespeichert werden. Wenn ein OneToMany-Formular im VFP Formular-Designer geöffnet wird und das Child-Grid selektiert wird, kann mit einem Rechtsklick auf das Grid das Kontextmenü geöffnet werden. Aus dem Kontextmenü kann über den Menüpunkt Generator der VFX –

CChildGrid Builder gestartet werden. Hier können alle Einstellungen für das Child-Grid geändert werden, die wir im VFX – COneToMany Builder gemacht haben. Wenn nur das Grid bearbeitet werden soll, muss also nicht der Form Builder gestartet werden. Auf der Seite *Options* kann der Code der Methode *OnPostInsert* des Child-Grids bearbeitet werden. Hierfür versucht VFX Code zu generieren, so weit dies möglich. Mit diesem Code wird in neu hinzugefügten Child-Datensätzen die ID des dazu gehörenden Parent-Datensatzes gespeichert. In unserem Beispiel wird im Feld *OrderId* in der Tabelle *Orderdetails.dbf* die *OrderID* aus *Orders.dbf* gespeichert.

```
REPLACE orderid WITH orders.orderid IN orderdetails
```



VFX – CChildGrid Builder, Options Page

Damit wird die Beziehung vom Child-Datensatz zum Parent-Datensatz hergestellt. Immer, wenn ein neuer Child-Datensatz hinzugefügt wird, wird der Code der Methode *OnPostInsert* des Child-Grid ausgeführt.

VFX – CTreeviewForm Builder

Wir erstellen nun mit Visual Extend ein Formular, das ein Treeview-Steuerelement zur Suche von Daten beinhaltet. Wir alle kennen das Treeview-Steuerelement aus dem Windows-Explorer. Dort dient es dazu Laufwerke und Ordner auf unserem PC oder aus der Netzwerkumgebung darzustellen. Das Treeview-Steuerelement ermöglicht eine hierarchische Darstellung von Daten in einer Baumstruktur. Die Grundlage für ein Treeview-Steuerelement ist in jedem Fall ein Cursor. Es kann sich hierbei um eine Tabelle, eine Ansicht oder auch um einen Cursoradapter handeln. Wichtig ist, dass dieser Cursor eindeutige Schlüsselwerte enthält, denn jeder Eintrag im Treeview muss eine eindeutige Node-ID bekommen.

Das Treeview-Steuerelement ist ein ActiveX-Steuerelement. Bei der Installation beim Kunden muss sichergestellt werden, dass dieses ActiveX-Steuerelement mit installiert wird. Das Treeview-Steuerelement ist in der Datei *MSComctl.ocx (v6.0 SP6)* enthalten.

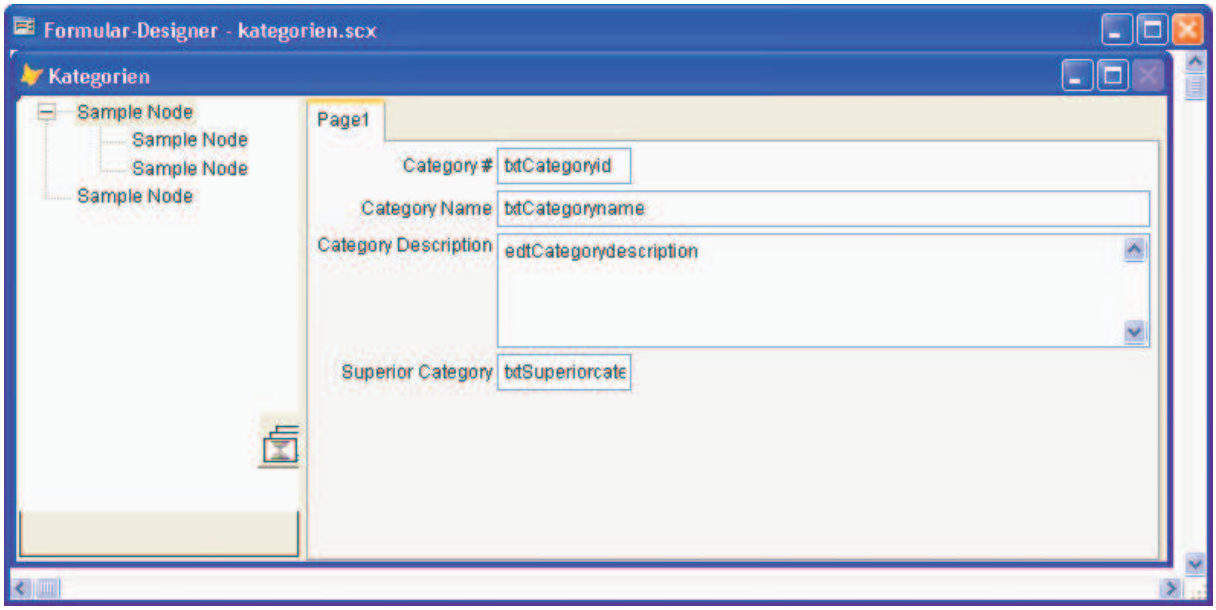
In VFX gibt es für Formulare mit einem Treeview-Steuerelement eine vorbereitete Formularklasse. Dies ist die Klasse *CTreeviewForm* und natürlich gibt es dafür auch einen passenden Form Builder. Für das Treeview-Steuerelement können im Builder die gleichen Layout-Einstellungen gemacht werden, wie dies auch im

Microsoft Builder möglich ist. Die wichtigste Einstellung dabei ist die Einzugsbreite der Knoten, die standardmäßig ziemlich breit eingestellt ist.

Die übrigen Einstellungen im VFX – CTreeviewForm Builder sind sehr ähnlich dem VFX – CDataFormPage Builder. Der wesentliche Unterschied ist die zweite Seite des Builders. Wo im VFX – CDataFormPage Builder eine Seite zur Bearbeitung von Grids zu finden ist, finden wir hier eine Seite auf der alle Einstellungen für das Treeview-Steuerelement gemacht werden können.

Auch dieser Builder wird aus dem VFX-Menü über den Form Wizard gestartet. Durch die Auswahl der Formularklasse „weiß“ VFX welcher Form Builder anschließend verwendet werden muss.

Als Beispiel erstellen wir ein Formular basierend auf der Tabelle *Kategorien.dbf*. Als Formularklasse wird die Klasse *CTreeviewForm* ausgewählt. Im Data Environment Builder wird die Tabelle *Kategorien.dbf* ausgewählt. Im nächsten Schritt startet dann der VFX – CTreeviewForm Builder.

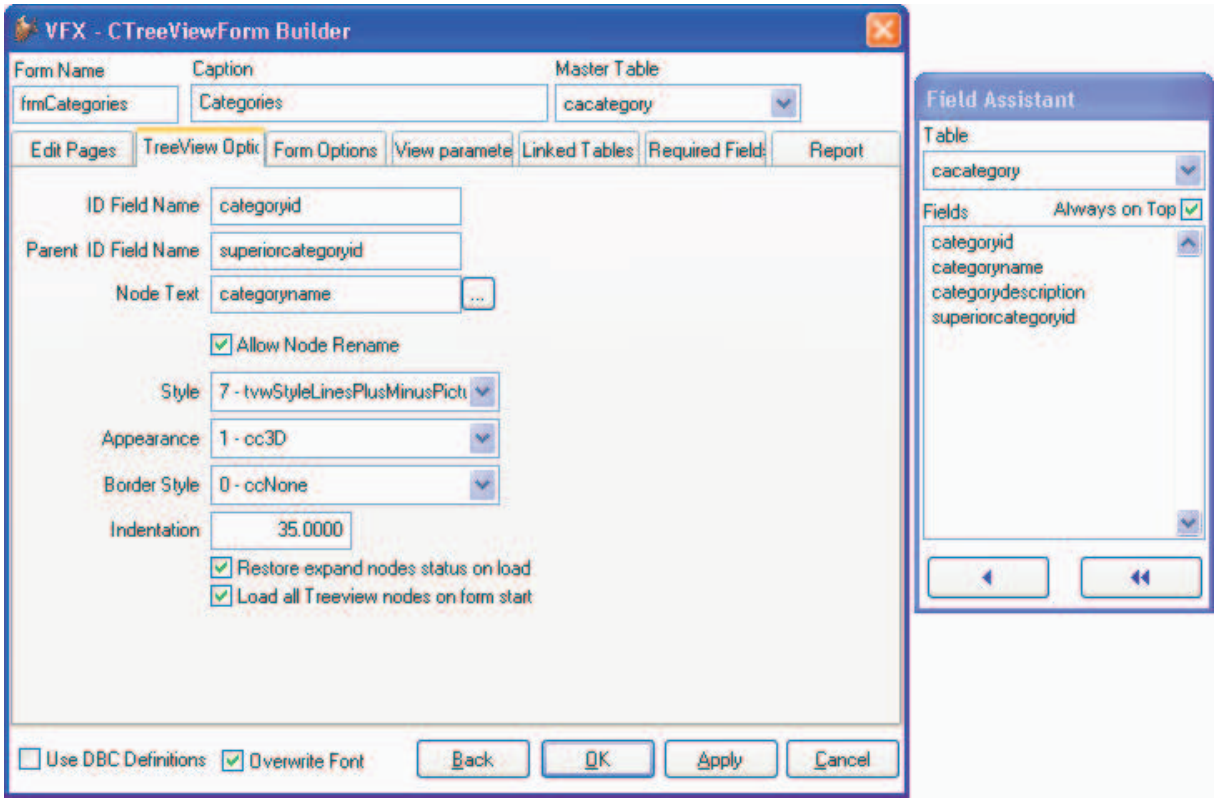


Formular mit Treeview im VFP Formular-Designer

In diesem Moment ist das Formular bereits im Hintergrund im VFP Formular-Designer zu sehen. Wir sehen im linken Teil des Formulars das Treeview-Steuerelement und im rechten Teil des Formulars befindet sich ein Seitenrahmen zur Bearbeitung von Daten, so ähnlich wie auf der Klasse *CDataFormPage*. Das Such-Grid gibt es in diesem Seitenrahmen jedoch nicht. Stattdessen werden die Daten zur Laufzeit im Treeview-Steuerelement gesucht.

Zur Demonstration der Funktionsweise fügen wir der Edit-Seite alle Felder der Tabelle Kategorien hinzu. Dazu gehören insbesondere auch die Felder *CategoryId* und *SuperiorCategoryId*.

Mit dem Feld *SuperiorCategoryId* wird auf den nächst höher gelegenen Knoten im Treeview verwiesen. Eine Baumstruktur ist immer so aufgebaut, dass jedes Element auf ein darüber liegendes Element verweisen muss. Wenn die *SuperiorCategoryId* den Wert 0 enthält, wird der Eintrag als Root Node angezeigt. Dadurch, dass diese Felder mit in die Edit-Seite aufgenommen werden, kann das Verhalten zur Laufzeit gut beobachtet werden. In einem Formular für die Praxis sollten die ID-Felder natürlich nicht auf dem Formular sichtbar sein.



VFX – CTreeviewForm Builder

Auf der Seite *Treeview Options* werden die Einstellungen für das Treeview-ActiveX-Steuerelement gemacht. Zunächst wird ein Klick in das Feld *ID Field Name* gemacht. Anschließend wird mit einem Doppelklick das Feld *categoryid* im Field Assistant ausgewählt. Dabei wird *categoryid* in *ID Field Name* übernommen. Die Auswahl im Field Assistant wirkt also auf das Eingabefeld im Builder, das gerade aktuell den Fokus hat. So kann auch auf dieser Seite des Builders der Field Assistant ganz einfach benutzt werden. Jetzt klicken wir in das Feld *Parent ID Field Name* und aus dem Field Assistant wird wiederum mit einem Doppelklick *superiorcategoryid* ausgewählt. Damit haben wir bereits die Hierarchiestruktur des Treeview aufgebaut.

Als nächstes muss der Node Text gewählt werden. Dies ist der Text, der bei den einzelnen Einträgen im Treeview angezeigt wird. Hier kann ein beliebiger Ausdruck eingetragen werden, also auch eine Zusammensetzung aus mehreren Feldern und auch Funktionen oder Methodenaufrufe können verwendet werden. Auch feststehende Texte sind möglich. In unserem Beispiel verwenden wir *categoryname*, also ein einfacher Feldname. Wenn ein einfacher Feldname verwendet wird, kann dieser Name zur Laufzeit direkt im Treeview-Steuerelement bearbeitet werden. Dazu finden wir im Rechtsklick-Menü den Menüpunkt *umbenennen*. Dabei wechselt das Formular in den Bearbeitungsmodus.

Die einzige weitere Einstellung, die noch geändert werden sollte ist die Indentation. Standardmäßig ist dieser Wert ziemlich breit eingestellt. Meine Empfehlung ist eine Einzugsbreite von 15 oder 20 zu verwenden.

Im unteren Teil der Seite befinden sich noch zwei Checkboxen. *Restore expand nodes status on load* speichert den aktuellen Status (geöffnet oder geschlossen) eines Knotens in der Tabelle *Vfxres.dbf*. Wenn der gleiche Benutzer dieses Formular erneut aufruft, werden alle Knoten wieder so angezeigt, wie es beim Schließen des Formulars war.

Load all Treeview nodes on form start ist das Standardverhalten. Das heißt, dass das gesamte Treeview-Steuerelement beim Start des Formulars mit Daten gefüllt wird. Wenn das Treeview auf einer Datenquelle mit sehr vielen Daten basiert, kann das zu einem recht langen Ladevorgang des Formulars führen. Um dies zu optimieren können wir das Häkchen in dieser Checkbox entfernen. Es werden dann nur die Daten in das Treeview geladen, die beim Start des Formulars auch sichtbar angezeigt werden. Beim Öffnen eines Knoten werden die erforderlichen Daten dann dynamisch nachgeladen.

Mit einem Klick auf die Schaltfläche *OK* wird auch dieses Formular vom VFX Form Builder generiert. Auch dieses Formular wird in den Öffnen-Dialog eingetragen. Schon im VFP Formular-Designer ist zu sehen, dass das Formular vom Layout her gut aussieht.

Zum Test starten wir *Vfxmain.prg* und öffnen das Formular Kategorien im Öffnen-Dialog. Das Treeview-Steuerelement ist mit den erwarteten Daten gefüllt. Mit einem Mausklick auf + können Knoten geöffnet werden. Hierarchisch verschachtelt können weitere Knoten geöffnet werden. Der aktuelle Zustand der Knoten (geöffnet oder geschlossen) wird von der Anwendung gespeichert. Wir schließen also das Formular und sehen beim erneuten Öffnen, das alle Knoten wieder so hergestellt werden, wie wir es beim Schließen des Formulars gesehen haben.

Wie wir gesehen haben, sind dank der Builder-Unterstützung von VFX Formulare mit einem Treeview-Steuerelement sehr schnell zum Laufen zu bekommen.

VFX – CPickField Builder

Jetzt wollen wir uns mit Auswahllisten beschäftigen. Auch hierfür bietet VFX einen leistungsfähigen Builder. Der VFX – CPickField Builder bearbeitet die Eigenschaften der Klasse *CPickField*. *CPickField* ist ein Container, in dem zwei Textboxen und eine Schaltfläche enthalten sind. In die linke Textbox kann der Benutzer zur Laufzeit Werte eingeben. Daneben befindet sich eine Schaltfläche zum Aufruf einer Auswahlliste und rechts davon ist noch eine Textbox in der zusätzliche Werte aus der Auswahltabelle angezeigt werden können. Diese Container-Klasse bietet sehr viele Einstellmöglichkeiten, die alle mit dem VFX – CPickField Builder bearbeitet werden können.

Ein Auswahlfeld zur Laufzeit

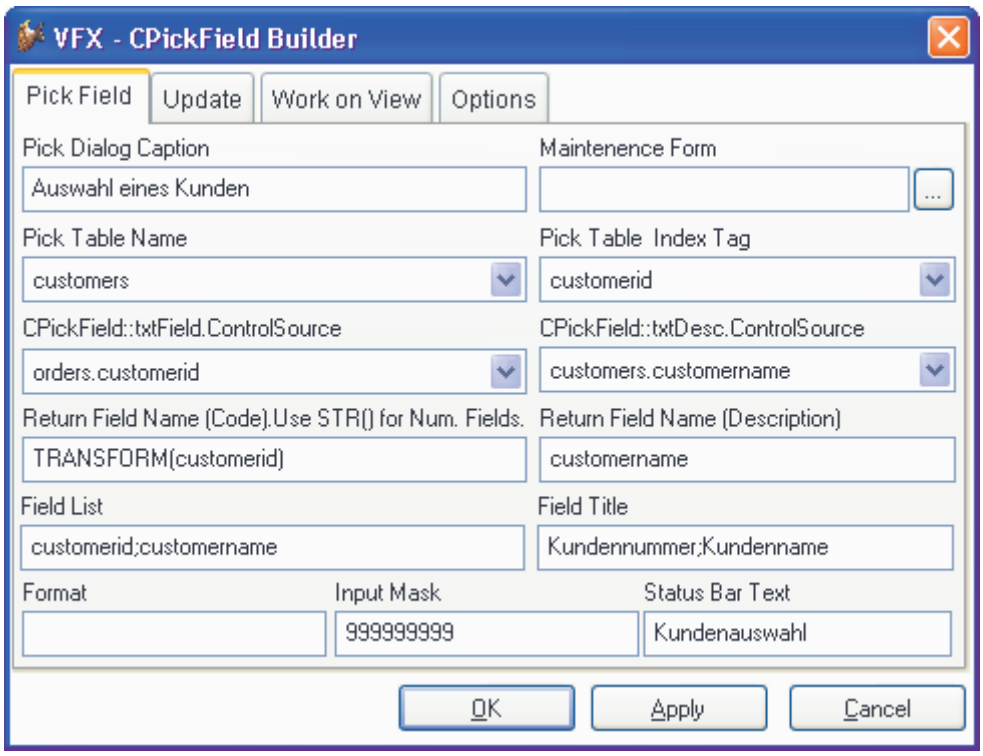
Steuerelemente aus der Klasse *CPickField* können mit den Form Buildern Formularen hinzugefügt werden. Auch die nachträgliche Änderung der Klasse eines Steuerelements ist möglich. So kann zum Beispiel aus einer *CTextbox* ein *CPickField* gemacht werden. Die Möglichkeit eine Klasse zu ändern besteht nicht nur in den VFX Form Buildern, sondern kann auch direkt aus dem VFX-Menü mit dem VFX – Class Switcher gemacht werden.

Wir wollen dem Auftragsformular im Parent-Teil für die Auftragsdaten ein Auswahlfeld hinzufügen, in dem der Benutzer einen Kunden eintragen kann. Dazu öffnen wir das Auftragsformular im VFP Formular-Designer und starten aus dem Kontextmenü den VFX – COneToMany Builder.

Im VFX – Data Environment Builder fügen wir die Tabelle *Customers.dbf* hinzu. Dies wird die Datenquelle für die Auswahlliste sein. Von der Tabelle *Orders.dbf* mit den Auftragsköpfen brauchen wir eine Relation zu der Tabelle *Customers.dbf*. Immer, wenn der Datensatzzeiger in der Tabelle *Orders.dbf* bewegt wird, soll der Datensatzzeiger in der Tabelle *Customers.dbf* automatisch nachgeführt werden. Dazu stellen wir in der Tabelle *Customers.dbf* den Indexschlüssel *customerid* ein. Der Parent-Alias wird die Tabelle *Orders.dbf*. Der Relationsausdruck wird *customerid*. Dadurch wird von einem Auftragskopf auf einen Datensatz in der Tabelle *Customers.dbf* gezeigt. Im Auswahlfeld soll zur Laufzeit in der rechten Textbox der Kundenname aus der Tabelle *Customers.dbf* angezeigt werden. Damit sind alle in der Datenumgebung erforderlichen Einstellungen fertig.

Mit einem Mausklick auf die Schaltfläche *Next* kommen wir zum VFX – COneToMany Builder. Im Parent-Teil des Formulars fügen wir auf der Seite *Edit Pages* ein weiteres Steuerelement mit *Orders.customerid* als Controlsource hinzu. Als Klasse für dieses Steuerelement wählen wir die Klasse *CPickField* aus. Damit sind wir auch im VFX – COneToMany Builder mit allen Einstellungen fertig. Mit einem Klick auf *OK* wird das zusätzliche Steuerelement dem Formular hinzugefügt.

Am neu hinzugefügten *CPickField* Steuerelement müssen noch ein paar Einstellungen gemacht werden. Dazu markieren wir dieses Steuerelement im VFP Formular-Designer und wählen aus dem Rechtsklickmenü *Builder*. Es startet der für diese Klasse zuständige Builder, der VFX – CPickField Builder. In diesem Builder werden alle Einstellungen gemacht.



VFX – CPickField Builder, Pick Field Page

- **Pick Field** – Hier finden wir Einstellungen, die für das Auswahlfeld ganz allgemein sind oder die beide Textboxen betreffen.
  Pick Dialog Caption – Dies ist die Überschrift der Auswahlliste, aus der der Benutzer zur Laufzeit Werte auswählen kann.

  Pick Table Name – Dies ist die Datenquelle der Auswahlliste.

  CPickField: txtField.ControlSource – Dies ist ein Feld aus der Bearbeitungstabelle. In dieses Feld macht der Benutzer manuelle Eingaben.

  Return Field Name (Code) – Wenn der Benutzer die Auswahlliste zur Selektion eines Wertes benutzt, ist in diesem Feld der Rückgabewert enthalten. Dieser Rückgabewert wird nach der Rückkehr aus der Auswahlliste in *txtField* geschrieben. Der Typ dieses Wertes muss Zeichen sein. Andere Feldtypen können mit TRANSFORM() in einen Zeichentyp umgewandelt werden. Wichtig ist hier keinen Aliasnamen mitzugeben, denn dieser Wert wird aus der Auswahlliste geholt. Die Datenquelle für die Auswahlliste wird immer mit einem temporären Aliasnamen geöffnet.

  Maintenance Form – Hier kann der Name eines Formulars angegeben werden, in dem der Benutzer zur Laufzeit die Daten aus der Auswahltabelle bearbeiten kann. Dadurch erhält der Benutzer die Möglichkeit Werte in der Auswahltabelle zu ändern und der Auswahltabelle neue Datensätze hinzuzufügen. Dieses Feld kann auch leer bleiben.

  Pick Table Index Tag – Dies ist ein Indexschlüssel aus der Auswahltabelle. Mit diesem Indexschlüssel wird die Benutzereingabe in *txtField* validiert. Wenn der Benutzer einen Wert von Hand eingibt, muss dieser Wert in der Auswahltabelle gesucht werden.

  CPickField: txtDesc.ControlSource – Hier wird ein Feld aus der Auswahltabelle ausgewählt. Hier ist der richtige Datensatz durch die in der Datenumgebung gesetzte Relation aktuell.

  Return Field Name (Description) – Auch dieser Wert wird von der Auswahlliste zurückgegeben. Hier ist also ein Feldname aus der Auswahltabelle ohne Aliasnamen anzugeben. In der Regel ist dies der gleiche Feldname, der auch in *txtDesc.ControlSource* angegeben wird. Auch dieser Wert muss vom Typ Zeichen sein. Ggf. kann der Wert mit der Funktion TRANSFORM() in einen Zeichentyp umgewandelt werden.

  Bis hier haben wir alle Felder im Builder ausgefüllt, die erforderlich sind um das Auswahlfeld funktionsfähig zu machen. Alle weiteren Angaben sind optional.

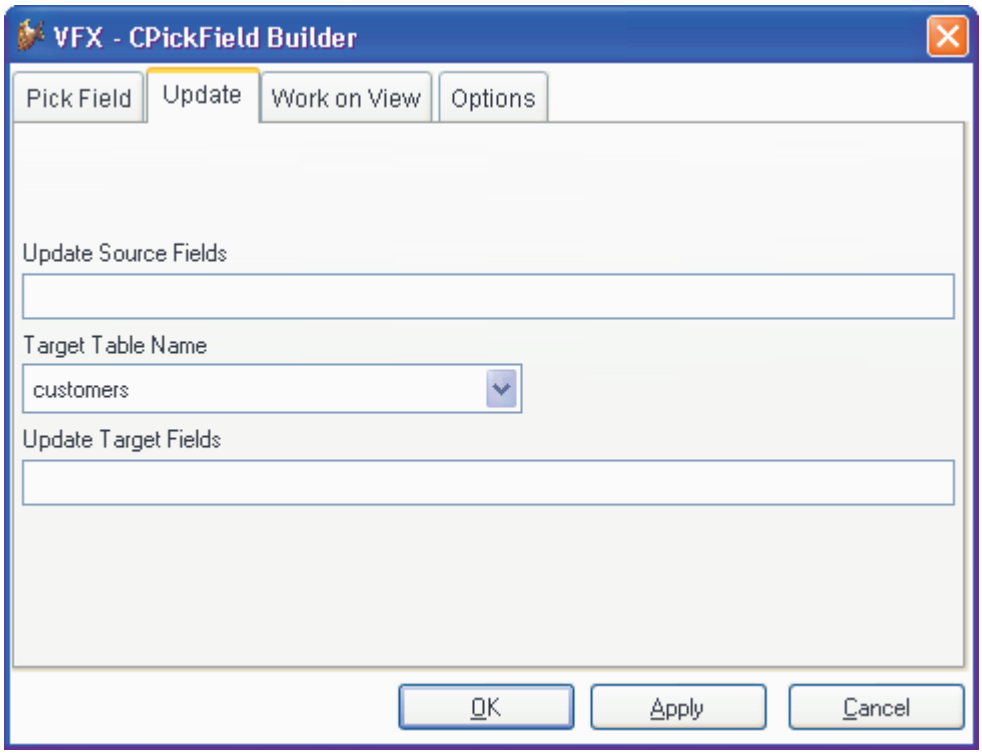
Field List – Standardmäßig wird im Fenster der Auswahlliste die gesamte Auswahltabelle angezeigt. Es erscheinen alle Felder aus der Tabelle in der physikalischen Reihenfolge der Tabellenstruktur. In vielen Fällen wird dies nicht gewünscht sein, sondern man möchte die anzuzeigenden Spalten selbst definieren. So können zum Beispiel Schlüsselfelder unterdrückt werden. Hier kann eine Liste von Feldnamen angegeben werden, aus denen die Auswahlliste aufgebaut wird. Die Elemente der Liste sind mit Semikolon zu trennen.

Field Title – Hier wird eine Liste mit Spaltenüberschriften, passend zu den oben angegebenen Feldnamen eingetragen. Auch hier sind die Listenelemente mit Semikolon zu trennen.

Format – Wert für die Eigenschaft *Format* des *txtField*.

Input Mask – Wert für die Eigenschaft *InputMask* des *txtField*.

Status Bar Text – Wert für die Eigenschaft *StatusBarText* des *txtField*.

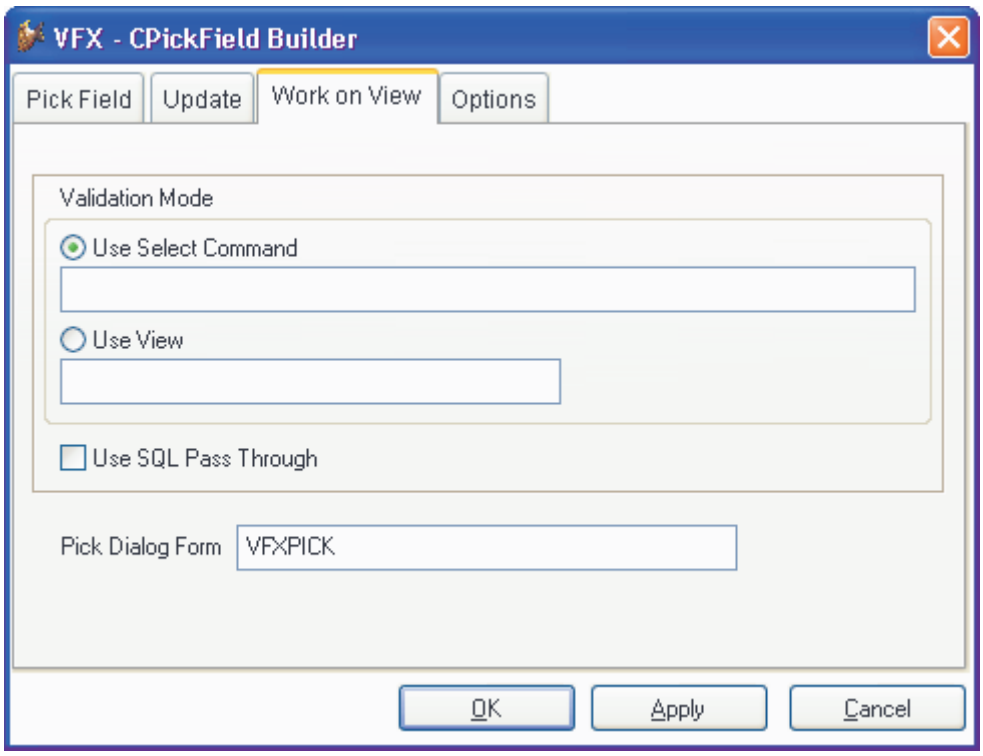


VFX – CPickField Builder, Update Page

- **Update** – Auf dieser Seite können weitere Werte aus der Auswahltabelle geholt und in der Zieltabelle gespeichert werden.
  Update Source Fields – Hier kann eine durch Semikolon separierte Liste von Feldnamen angegeben werden, deren Werte zusätzlich aus der Auswahltabelle geholt werden sollen.

  Target Table Name – Dies ist Bearbeitungstabelle des Formulars, in das die Werte gespeichert werden.

  Update Target Fields – Hier kann eine durch Semikolon separierte Liste von Feldnamen aus der Bearbeitungstabelle angegeben werden. Hier werden die Werte aus der Auswahltabelle gespeichert. Diese Liste von Feldnamen muss mit der Liste von Feldern aus *Update Source Fields* korrespondieren.



VFX – CPickField Builder, Work on View Page

- **Work on View** – Hier werden Einstellungen gemacht, die bei der Arbeit mit Ansichten oder CursorAdaptern erforderlich sind. Es können verschiedene Optionen zur Validierung der Benutzereingabe gewählt werden. VFX erkennt automatisch, ob mit Ansichten oder CursorAdaptern gearbeitet wird. Wenn mit Tabellen gearbeitet wird, werden die Eintragungen auf dieser Seite nicht berücksichtigt.

  Wenn wir für die Auswahlliste eine Ansicht verwenden, die eine große Anzahl von Datensätzen enthält, wäre es nicht performant diese Ansicht für die Validierung der Benutzereingabe zu verwenden. Für die Validierung braucht eigentlich nur maximal ein Datensatz aus der Datenbank geholt werden. Die Benutzereingabe soll einfach und schnell auf Gültigkeit überprüft werden.

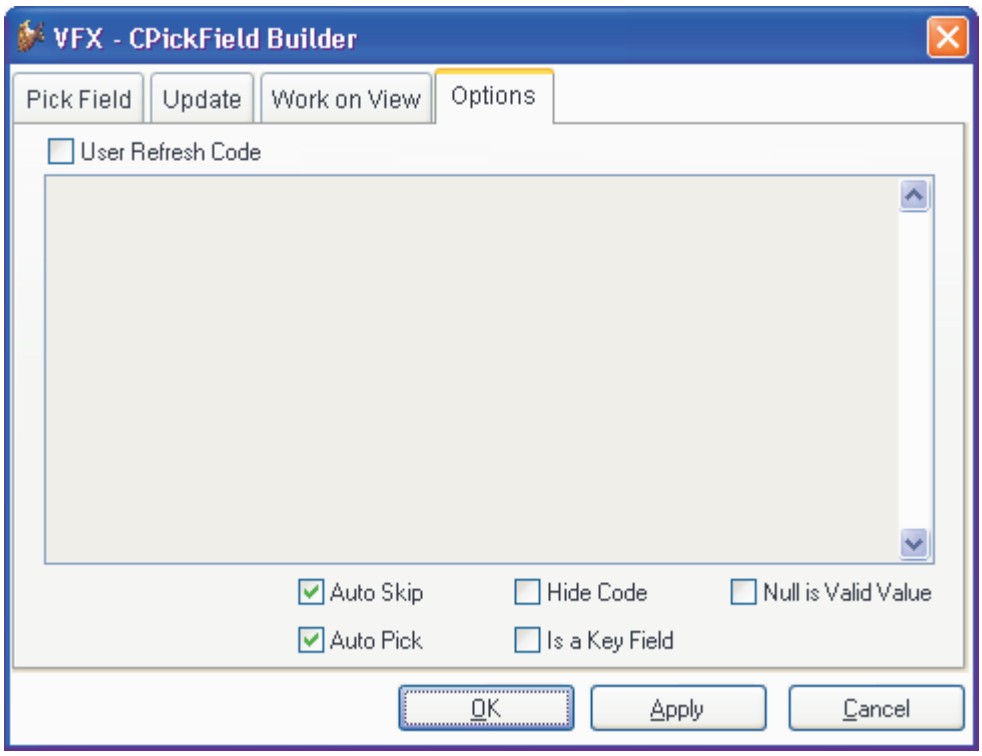
  Zunächst wird der Validierungsmodus ausgewählt. Die Validierung der Benutzereingabe kann durch einen SELECT-Befehl oder durch eine Ansicht erfolgen. Eine Ansicht enthält einen SELECT-Befehl, der genauso aufgebaut ist, wie der SELECT-Befehl, der im Feld *Use Select Command* eingegeben werden kann. Meine Empfehlung ist einen SELECT-Befehl zu verwenden. Dann wird für die Validierung der Benutzereingabe gar keine zusätzliche Ansicht benötigt und nichts muss unnötig im Datenbank-Container gespeichert werden. Die zusätzlichen Features, die eine Ansicht bietet, werden hier sowieso nicht gebraucht.

  Der SELECT-Befehl, der zur Validierung der Benutzereingabe dient, muss eine Where-Klausel mit genau einem Parameter enthalten. Der Name dieses Parameters muss eine Variable sein. Der Name dieser Variablen spielt keine Rolle. VFX analysiert den SELECT-Befehl zur Laufzeit und ermittelt den Namen des Parameters. Dieser Variablen wird dann der Wert der Benutzereingabe zugewiesen und anschließend wird der SELECT-Befehl ausgeführt. Ist das Ergebnis kein Datensatz, war die Validierung nicht erfolgreich. Wenn ein Datensatz zurückgeliefert wird, war die Benutzereingabe gültig.

  Use SQL Pass Through – Wenn diese Option ausgewählt ist, versucht VFX den SELECT-Befehl per SQL Pass Through an eine Remote Datenbank zu übergeben. Hierdurch wird die Ausführung des Befehls noch ein bisschen verbessert.

  Ein Beispiel für die Validierung eines Pickfields basierend auf einer Ansicht finden Sie in VFX95Test, im Formular *Xvchild.scx* und dort im Auswahlfeld *cntParentid*.

  Pick Dialog Class – Hier kann der Name des Formulars angegeben werden, dass zur Anzeige des Auswahldialogs verwendet wird. In der Regel braucht am Standardwert nichts geändert werden.



VFX – CPickField Builder, Options Page

- **Options** – Hier können noch einige besondere Einstellungen zum Auswahlfeld gemacht werden.
  User Refresh Code – Die hier eingegebene Code wird in der Methode *OnUserRefresh* gespeichert. Wenn dieser Code verwendet werden soll, muss der Wert der Eigenschaft *lUserRefresh* auf *.T.* eingestellt werden. In machen Fällen mag es erforderlich sein in der rechten Textbox des Auswahlfeldes etwas anzuzeigen, das sich nicht aus einer Relation gewinnen lässt. Hier kann beliebiger Code eingegeben werden, der den Wert des *Description*-Feldes anzeigt. Ein Beispiel hierfür befindet sich in VFX95Test, im Formular *Child.scx* und dort im Auswahlfeld *cntItemid*.

  Auto Skip – Wenn der Benutzer einen Wert aus der Auswahlliste auswählt und die Eingabetaste drückt, wird der Fokus sofort auf das nächste Steuerelement in der Aktivierfolge gesetzt. Wenn diese Option nicht ausgewählt ist, bleibt der Fokus nach einer Auswahl aus der Auswahlliste auf dem Auswahlfeld stehen und der Benutzer kann seine Eingabe wiederholen oder mit der Eingabetaste zum nächsten Steuerelement gelangen.

  Auto Pick – Wenn diese Option ausgewählt ist und der Benutzer eine ungültige Eingabe macht, wird automatisch die Auswahlliste angezeigt. Wenn diese Option nicht ausgewählt ist, erscheint bei falschen Eingaben eine Messagebox und der Benutzer wird gefragt, ob er die Auswahlliste anzeigen oder seine Eingabe wiederholen möchte.

  Hide Code – Mit dieser besonderen Einstellung werden zur Laufzeit die Steuerelemente im Auswahlfeld-Container anders angeordnet. Das Eingabefeld verschwindet ganz. Der Benutzer hat nicht die Möglichkeit einen Wert einzugeben, sondern kann nur aus der Auswahlliste Werte auswählen. Die Schaltfläche zum Aufruf der Auswahlliste wird auf die rechte Seite von *txtDescr* verschoben.

  Is a Key Field – Wenn diese Option ausgewählt ist, kann der Benutzer nur dann Werte eingeben oder auswählen, wenn sich das Formular im Neuaufnahmemodus befindet. Bei einer späteren Bearbeitung der Daten bleibt das Auswahlfeld schreibgeschützt.

  Null is valid Value – Wenn diese Option gewählt ist, kann der Benutzer das Auswahlfeld ohne die Eingabe eines Wertes überspringen. Wenn diese Option nicht ausgewählt ist, muss der Benutzer einen gültigen Wert eingeben oder auswählen.

Nach einem Klick auf die Schaltfläche *OK* können im Eigenschaftsfenster alle Einstellungen, die der Builder gemacht hat, betrachtet werden.

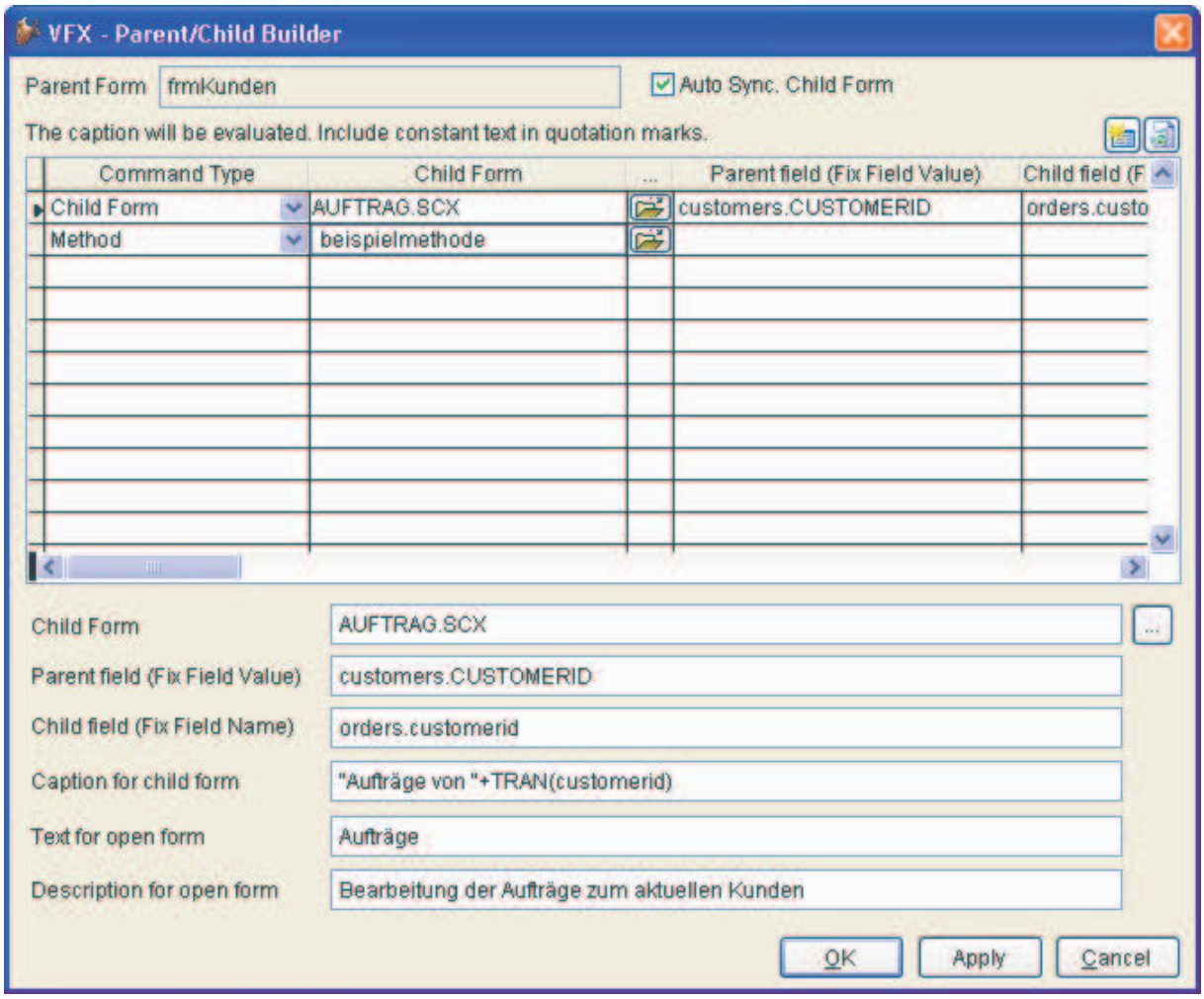
Zur Laufzeit hat das Formular der Auswahlliste alle guten Eigenschaften, die wir von anderen VFX-Formularen kennen. Im Grid kann in allen Spalten inkrementell gesucht werden. Alle Formulareinstellungen werden für den aktuell angemeldeten Benutzer gespeichert.

VFX – Parent/Child Builder

Der Parent/Child Builder dient dazu Beziehungen zwischen zwei oder mehr Formularen herzustellen. Alle dafür erforderlichen Einstellungen können in diesem Builder gemacht werden. Zur Laufzeit kann ein Child-Formular mit der Funktionstaste *F6* aufgerufen werden oder auch über ein Symbol in der Symbolleiste. Wenn mehr als ein Child-Formular zur Verfügung steht, öffnet sich automatisch ein Dialog, aus dem der Benutzer zwischen den verschiedenen Children auswählen kann. Es ist möglich zu einem Parent-Formular mehrere Children aufzurufen. Ein Child-Formular kann wiederum andere Child-Formulare aufrufen. Dadurch ist eine beliebige hierarchische Verschachtelung von Formularen möglich.

In unserem Beispiel soll das Formular *Kunden.scx* als Parent-Formular verwendet werden. Das Formular *Auftrag.scx* soll das Child-Formular werden. Zur Laufzeit soll es möglich sein, einen Kunden auszuwählen und aus dem Kundenformular soll dann das Auftragsformular aufgerufen werden können, das dann nur die Aufträge des aktuellen Kunden enthält.

Um das zu realisieren wird das Kundenformular im VFP Formular-Designer geöffnet. Aus dem VFX-Menü wird nun der VFX – Parent/Child Builder gestartet.



VFX – Parent/child Builder

Im Builder wird zunächst eine neue Zeile hinzugefügt. Aus der Combobox *Command Type* können verschiedene Aktionen aus dem Parent-Formular gestartet werden. Für unser Beispiel wählen wir hier *Child Form.*

Das Child-Formular, das aus dem Kundenformular heraus aufgerufen werden soll, ist das Auftragsformular. Das Child-Formular kann über die Schaltfläche in der dritten Grid-Spalte ausgewählt werden. Bei der Auswahl öffnet VFX das Child-Formular kurz im Formular-Designer um ein paar Einstellungen aus diesem Formular auszulesen. VFX versucht so viele Werte wie möglich automatisch in die weiteren Spalten des Parent/Child Builders einzutragen, sodass hier nur wenige Eintragungen manuell gemacht werden müssen.

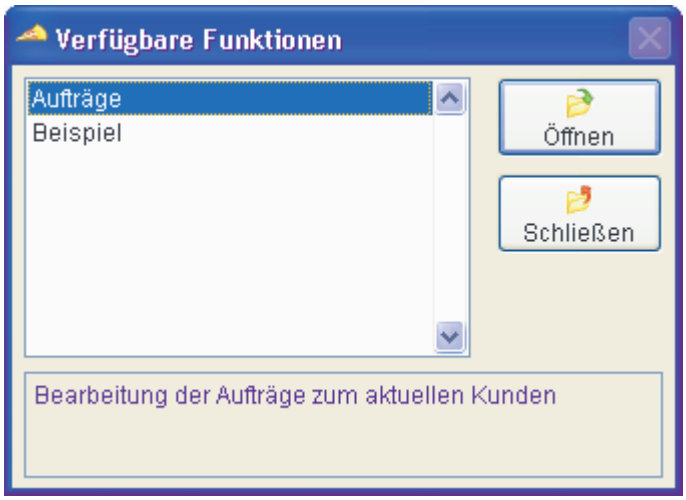
Der Builder hat erkannt, dass das Schlüsselfeld im Kundenformular *customers.customerid* ist. Im Auftragsformular muss es in der Tabelle *Orders.dbf* mit den Auftragsdaten ebenfalls ein Feld geben, dass die

customerid enthält. Über die *customerid* wird zur Laufzeit im Auftragsformular ein Filter gesetzt. Der Wert *customers.customerid*, der aus dem Kundenformular an das Auftragformular übergeben werden muss, ist hier im Builder schon richtig vorbelegt.

In der Spalte *Child field (Fix Field Name)* muss der Name des Feldes angegeben werden, das im Child-Formular für den Filter verwendet wird. In unserem Beispiel ist das *orders.customerid*.

In der nächsten Spalte *Caption for child form* kann eine Überschrift für das Child-Formular angegeben werden. Wenn das Auftragformular aus dem Kundenformular heraus gestartet wird, soll dem Formular eine Überschrift gegeben werden, aus der ersichtlich ist, dass hier nur die Aufträge zu einem bestimmten Kunden angezeigt werden. Hier kann ein beliebiger Ausdruck eingegeben werden, der zum Beispiel die Kundennummer und den Kundennamen enthält.

Wenn nur ein Eintrag im Parent/Child Builder gemacht wird, brauchen keine weiteren Eintragungen gemacht. Wenn mehrere Child-Formulare gestartet werden sollen, zeigt VFX zur Laufzeit automatisch einen Dialog, aus dem der Anwender das gewünschte Formular auswählen kann.



OnMore-Dialog

Text for open Form ist die Bezeichnung des Child-Formulars. Dieser Eintrag ist in der Listbox des *OnMore*-Dialogs zu sehen.

Description for open form ist die Beschreibung des Child-Formulars. Dieser Eintrag ist in der Editbox am unteren Rand des *OnMore*-Dialogs zu sehen.

Alle Spalten können wahlweise im Grid oder in den darunter liegenden Textboxen editiert werden.

Ganz oben auf dem Parent/Child Builder finden wir noch eine Checkbox *Auto Sync. Child Form*. Wenn diese Option ausgewählt ist, wird die Anzeige des Child-Formulars automatisch synchronisiert, wenn im Parent-Formular der Satzzeiger bewegt wird. In der Regel ist dies das gewünschte Verhalten und die Standardeinstellung kann unverändert bleiben.

Mit einem Mausklick auf *OK* werden alle erforderlichen Änderungen im Formular gemacht.

Zur Laufzeit sehen wir, dass während das Parent-Formular geöffnet und aktiv ist, ist der Symbolleiste die Schaltfläche *Weitere Funktionen* aktiv ist. Auch der entsprechende Menüeintrag im Menü *Bearbeiten* ist jetzt enabled. Hier sehen wir auch den dazugehörigen Hotkey *F6*. Wenn der Benutzer die Funktionstaste *F6* drückt oder den Menüeintrag auswählt oder auf die Schaltfläche in der Symbolleiste klickt, wird das Child-Formular geöffnet. Wir sehen sofort, dass es die gewünschte Überschrift hat.

Im Child-Formular werden nur noch die Aufträge des im Kundenformular aktuellen Kunden angezeigt. Wenn im Kundenformular der Satzzeiger bewegt wird, wird die Ansicht im Auftragsformular automatisch aktualisiert, sodass immer die Aufträge zum aktuellen Kunden angezeigt werden.

Auf dem beschriebenen Weg kann ganz ohne Programmierung oder durch manuelle Einstellung von Eigenschaften, nur durch die Builder-Unterstützung von VFX eine sehr leistungsfähige Beziehung zwischen zwei Formularen hergestellt werden.

Der Parent/Child Builder kann aber noch mehr. Neben der Möglichkeit Child-Formulare zu starten, können mit dem Parent/Child Builder auch Einstellungen gemacht werden, die andere Aktionen aus dem Parent-Formular ausführen lassen.

In unserem Beispiel öffnen wir das Formular *Kunden.scx* erneut im VFP Formular-Designer und starten aus dem VFX-Menü den VFX – Parent/Child Builder. Hier fügen wir noch eine zusätzliche Zeile hinzu. Außer der Möglichkeit Child-Formulare zu starten, kann in der Combobox *Command Type* ausgewählt werden, dass eine Methode ausgeführt werden soll. Dir dritte mögliche Option *Wait Window* dient eigentlich nur zu Testzwecken. Der Name der Methode wird in der Spalte *Method Name* eingetragen. Dies muss eine Methode auf dem aktuellen Formular, also dem Parent-Formular sein. Wenn es sich hierbei um eine neue, noch nicht existierende Methode handelt, so muss diese im VFP Formular-Designer manuell angelegt werden. Als nächstes machen wir noch die Eintragungen für den Öffnen-Dialog in den Spalten *Text for open form* und *Description for open form*. Damit sind auch die Einstellungen für den *OnMore*-Dialog gemacht und mit einem Klick auf *OK* werden alle Einstellungen in das Formular übernommen.

In die manuell neu erstellte Methode tragen wir als Beispiel-Code ein:

```
MESSAGEBOX("Dies ist eine Beispielmethode.")
```

Zur Laufzeit klicken wir aus dem Kundenformular wieder auf die Schaltfläche *Weitere Funktionen*. Diesmal erkennt VFX, dass es zu diesem Formular mehr als nur eine weitere Funktion gibt. Es wird der *OnMore*-Dialog angezeigt. Der Benutzer muss hier jetzt auswählen, was er gerne machen möchte. Hier sehen wir auch zum ersten Mal die Eintragungen, die wir im Parent/Child Builder in den Spalten *Text for open form* und *Description for open form* gemacht haben. Wir können hier das Child-Formular öffnen, wie wir es bereits gesehen haben. Dann aktivieren wir wieder das Kundenformular und klicken erneut auf *Weitere Funktionen*. Jetzt klicken wir auf *Beispiel* um die *Beispielmethode* aufzurufen. Und auch das funktioniert wie erwartet.

Auf diese Art kann der Parent/Child Builder benutzt werden, um aus einem Formular beliebige Funktionen aufzurufen. In einer Methode kann beliebiger VFP-Code ausgeführt werden.

Welche Einstellungen macht der VFX – Parent/Child Builder?

Wir öffnen das Kundenformular erneut im VFP Formular-Designer und betrachten das Eigenschaftsfenster. Dort sehen wir, dass sich auf dem Formular zwei Objekte befinden. Ein Relations-Manager *oRelationMgr* und ein Child-Manager *oChildManager*. Der VFX – Parent/Child Builder wirkt im Wesentlichen auf dem Child-Manager-Objekt. Wenn wir alle Eigenschaften ohne Standardwert anzeigen lassen sehen wir, dass nur in einer Methode Code eingetragen ist. Das ist die Methode *DefineChildForms*. Hier hat der Parent/Child Builder zwei Zeilen Code eingetragen.

Damit die Schaltfläche Weitere Funktionen in der Symbolleiste aktiviert wird und der entsprechende Menüeintrag enabled wird, ist der Eigenschaft *lMore* des Formulars der Wert *.T.* zugewiesen.

Symbolleisten zu Formularen

Der *OnMore*-Dialog ist noch nicht so überzeugend benutzerfreundlich. Wir wollen es unseren Benutzern etwas komfortabler machen und dem Formular eine Symbolleiste hinzufügen. Eine Symbolleiste, die nur dann aktiv ist, wenn das dazugehörige Formular aktiv ist. Über diese Symbolleiste sollen genau die Funktionen aufgerufen werden, die im VFX – Parent/Child Builder definiert wurden.

Dafür wird zunächst eine neue Symbolleiste angelegt, die auf der Klasse *CToolbar* aus der Klassenbibliothek *Vfxobj.vcx* basiert. Diese Symbolleiste wird in der Klassenbibliothek *Appl.vcx* gespeichert, in der alle eigenen Klassen zu einer VFX-Anwendung gespeichert werden sollten. Die Symbolleiste bekommt in unserem Beispiel den Namen *symKunden*. Es ist die Symbolleiste für das Kundenformular.

Dieser Symbolleiste müssen Schaltflächen hinzugefügt werden. Die Schaltflächen sollten auf der VFX-Klasse für Schaltflächen in Symbolleisten *CToolbarButton* aus der Klassenbibliothek *Vfxctrl.vcx* basieren. Da wir zwei *OnMore*-Aufrufe im Kundenformular definiert haben, fügen wir hier gleich zwei Schaltflächen hinzu. In das *Click*-Ereignis dieser Schaltflächen muss Code eingetragen werden, der die *OnMore*-Funktionen aufruft.

Die Schaltflächen bekommen eine Caption. Die erste Schaltfläche bekommt als Caption *A* und als Tooltiptext *Aufträge des aktuellen Kunden*. Die zweite Schaltfläche soll zum Aufruf der Beispielmethode dienen. Hier geben wir als Caption *B* ein und als Tooltiptext *Beispielmethode*. Auf der Symbolleiste befindet sich auch eine Caption. Hier tragen wir *Kunden* ein. Diese Caption wird sichtbar, sobald der Benutzer diese Symbolleiste entdockt. Außerdem ist diese Symbolleiste im Dialog zum Öffnen von Symbolleisten sichtbar.

Symbolleiste zum Formular Kunden

Jetzt müssen die Schaltflächen in der Symbolleiste noch funktionieren. Dafür fügen wir Code in das *Click*-Ereignis der Schaltfläche *A* ein:

```
_screen.activeform.onmore(1)
```

Hierdurch wird im aktiven Formular die *OnMore*-Methode aufgerufen und es wird dieser Methode der Parameter *1* mitgegeben. Diese Symbolleiste ist nur dann sichtbar, wenn das Kundenformular aktiv ist. Daher kann hier bedenkenlos auf *_screen.activeform* referenziert werden. Die Methode *OnMore* wird hier genauso aufgerufen, als würde der Benutzer auf *Weitere Funktionen* in der Symbolleiste klicken oder die Funktionstaste *F6* drücken.

Der *OnMore*-Methode kann ein Parameter mitgegeben werden. Wenn die Zahl *1* übergeben wird, wird das erste Formular aus dem Parent/Child Builder gestartet. Wenn eine *2* übergeben wird, wird entsprechend das zweite Formular gestartet, bzw. in unserem Beispiel die Beispielmethode aufgerufen.

In der Schaltfläche *B* wird im *Click*-Ereignis entsprechend *2* als Parameter übergeben.

Damit ist die Symbolleiste fertig und funktionsfähig.

Dem Formular muss nun die neue Symbolleiste zugewiesen werden. Dafür wird das Formular *Kunden.scx* im VFP Formular-Designer geöffnet. Im Eigenschaftsfenster wird bei der Eigenschaft *cToolbarClass* der Name der Symbolleistenklasse *symKunden* eingetragen.

Zu jedem Formular wird in der Tabelle mit den Benutzereinstellungen *Vfxres.dbf* gespeichert, ob die zu dem Formular gehörende Symbolleiste beim Schließen des Formulars geöffnet war. Beim Erneuten Öffnen des Formulars wird die Symbolleiste nur dann geöffnet, wenn sie beim Schließen des Formulars sichtbar war. Auch der Dock-Zustand der Symbolleiste bzw. die Position der Symbolleiste auf dem Bildschirm wird in der Tabelle *Vfxres.dbf* gespeichert.

In der Eigenschaft *lShowToolbar* des Formulars kann eingestellt werden, dass eine Symbolleiste beim erstmaligen Öffnen des Formulars ebenfalls angezeigt werden soll.

Wenn eine Symbolleiste zu einem Formular zur Laufzeit nicht angezeigt wird, kann der Benutzer aus dem Menü *Ansicht, Symbolleisten* wählen. Es erscheint der Dialog zum Öffnen von Symbolleisten. In diesem Dialog wird die Caption der Symbolleiste als Name angezeigt.

Menüs zu Formularen

In manchen Fällen mag es sinnvoll sein einem Formular anstelle einer Symbolleiste ein Menü hinzuzufügen, aus dem die Benutzer Formular-spezifische Aktionen starten können. Natürlich wird auch dies von VFX unterstützt.

Dafür wird im VFP Projekt-Manager ein neues Menü angelegt. Es startet der VFX – Menu Designer und das neue Menü kann mit Inhalt gefüllt werden.

Das Menü bekommt den Namen Kunden. Wir fügen zwei Menüeinträge hinzu. Der erste Eintrag heißt Aufträge und der zweite Eintrag bekommt den Namen Beispielmethode. Nun muss noch eingegeben werden was passieren soll, wenn der Benutzer einen dieser Einträge auswählt. Dafür wird über die Schaltfläche *ActionOnSelect* ein Code-Fenster geöffnet. Hier tragen wir, genau wie bei den Schaltflächen der Symbolleiste, ein:

```
_screen.activeform.onmore(1)
```

Bei dem Menü muss nun noch eingestellt werden, dass es das bereits sichtbare Menü nicht ersetzen soll, sondern diesem Menü hinzugefügt werden soll. Dafür klicken wir im VFX – Menu Designer auf die Schaltfläche *Options* und wählen hier als *Location Append*. Dadurch wird dieses Menü rechts an das vorhandene Menü angehängt.

Das Menü wird unter dem Namen *Kunden.vmx* gespeichert.

Jetzt muss beim Kundenformular noch eingestellt werden, dass es dieses Menü verwenden soll. Dafür Öffnen wir das Formular *Kunden.scx* im VFP Formular-Designer und suchen im Eigenschaftsfenster die Eigenschaft *cMenuName*. Hier wird der Name des Menüs eingetragen, also *Kunden*. Mehr Einstellungen sind nicht erforderlich um ein Formular mit einem Menü zu verbinden.

Das Ergebnis können wir zur Laufzeit betrachten. Das Menü *Kunden* ist tatsächlich nur dann sichtbar, wenn das Kundenformular aktiv ist.