VFX 9.5 – Was ist neu?

Januar 2006

Uwe Habermann, Venelina Jordanova

Inhaltsverzeichnis

Neue Eigenschaften für Entwickler 4
Vererbungsarchitektur 4
Vfxobjbase.vcx 4
VFX-Formularklassen 4
Datenzugriff 4
Einstellungen für die Datenumgebung 4
Das neue Objekt goPath 5
Erweiterungen in der Klasse cBaseDataAccess 5
Builder und Wizards 6
VFX – Task Pane 6
VFX – Update Project Wizard 7
VFX – Application Builder 7
VFX – Upsizing Wizard 8
VFX – CursorAdapter Wizard 14
VFX – Data Environment Builder 14
VFX – Form Builder 16
VFX – Parent/Child Builder 18
VFX – Business Graph Builder 19
VFX – TextBox Builder 20
VFX – Toolbar Builder 21
Lokalisierung 21
VFX - Language Management Builder 21
Ausführen der Methode LangSetup 23
VFX - LangSetup Builder 23
Project Hook 23
Produktaktivierung 23
Definieren der Aktivierungsregeln 24
Aktivierungsschlüssel erstellen 25
VFX – Kundenverwaltung 26
Web Service für die Registrierung 28
Aktualisierung von VFX 29
Die Klasse cUpdate 29
Die Klasse cUpdateEngine 30
AFX Unterstützung 31
Hilfe 32
Benutzerhandbuch und Dokumentation der Neuheiten 32
Visual Extend Online 32
Senden Sie uns eine E-Mail! 33
So erreichen Sie uns 33
Support-Anfragen an das Forum richten 33
Info 35
Neue Eigenschaften des Anwendungsobjekts 35
Weitere Verbesserungen für Entwickler 36
Neue Eigenschaften für Endbenutzer 37
Erforderliche Rechte zur Ausführung 37
Neue Icons 37
Datenzugriff 38
Der Dialog Datenzugriff bearbeiten 38
Neue Eigenschaften in Onetomany-Formularen 40

OnChildRequery ... 40
Seriendokumenterstellung ... 40
Die Klasse cMailMerge ... 45
Berichte ... 47
Berichte bearbeiten ... 47
ReportOutput und ReportPreview ... 47
PDF-ReportListener ... 48
Erweiterter Druckdialog ... 48
Die Klasse cPrintDialog ... 49
Die Klasse cPrintEngine ... 50
XP-Öffnen-Dialog ... 52
Datenexport ... 52
Suchdialog ... 53
Anpassen-Dialog ... 56
Die Klasse CArchive ... 57
Die neue Klasse CTextSkype ... 57
Behandlung von Laufzeitfehlern ... 57
Aktualisierung der Anwendung ... 58
Aktualisierung der Datenbank beim Kunden ... 58
Datenbankreparatur ... 58
Terminalserver-Unterstützung ... 59
Weitere Verbesserungen für Endbenutzer ... 60
Anhang I - VFX AFX Wizard ... 61
Anhang II - Transact-SQL ... 76

Neue Eigenschaften für Entwickler

Vererbungsarchitektur

Vfxobjbase.vcx

Die Vererbungsarchitektur der VFX-Klassen wurde erweitert. In bisherigen VFX-Versionen konnte nur mithilfe von Hooks in die Funktion und in das Layout von VFX-Basisklassen eingegriffen werden. Wenn ein Entwickler zum Beispiel in seiner gesamten Anwendung eine bestimmte Schriftart verwenden wollte, konnte dies nur über Hooks im *Init* Ereignis erreicht werden. Dies hatte den Nachteil, dass die Anwendung in den VFP-Designern stets in der VFX-Standardschriftart Arial angezeigt wurde und nur zur Laufzeit die über einen Hook eingestellte Schriftart angezeigt wurde.

In VFX 9.5 ist nun eine zusätzliche Vererbungsschicht vorhanden. Die in bisherigen VFX-Versionen vorhandenen VFX-Basisklassen aus der Klassenbibliothek *Vfxobj.vcx* befinden sich nun in der Klassenbibliothek *Vfxobjbase.vcx*. An die bisherigen Namen der Klassen wurde *base* als Suffix angehängt.

Zu jeder Klasse aus dieser Klassenbibliothek gibt es eine 1:1-Ableitung in der Klassenbibliothek *Vfxobj.vcx*. Die Klassenbibliothek *Vfxobj.vcx* steht nunmehr dem Entwickler für eigene Anpassungen und Erweiterungen zur Verfügung. Hier ist es beispielsweise möglich die Schriftart einer Klasse zu ändern. Diese Einstellung wirkt sich dann auf alle Steuerelemente basierend auf dieser Klasse in der gesamten Anwendung aus.

Bei einer Aktualisierung des Projekts mit dem VFX – Update Project Wizard wird die Klassenbibliothek *Vfxobj.vcx* nicht aktualisiert.

VFX-Formularklassen

Die VFX-Formulare verwenden jetzt keine Datenumgebung mehr. Das Design der VFX-Formularklassen wurde so erneuert, dass alle benötigten Datenquellen programmatisch geöffnet werden. Alle VFX-Formulare sind 1:1-Ableitungen aus den entsprechenden Formularklassen. Künftige Änderungen in VFX erfolgen also nur noch in den Formularklassen. Eine Änderung der VFX-Formulare ist nicht mehr erforderlich. Dadurch hat der Entwickler die Möglichkeit die VFX-Formulare nach eigenem Bedarf anzupassen, ohne dass bei einer Aktualisierung von VFX Änderungen verloren gehen bzw. noch mal gemacht werden müssen.

Datenzugriff

VFX-Anwendungen sowie alle Entwicklerwerkzeuge unterstützen SQL Server 2000 und die MSDE sowie SQL Server 2005 und SQL Server 2005 Express.

Einstellungen für die Datenumgebung

Viele SET-Einstellungen werden schon vor dem Öffnen von Cursorn in der Datenumgebung benötigt. Zusätzlich zu der Methode *SetDataEnvironment()* des Objekts *goEnvironment* können jetzt SET-Einstellungen für CursorAdapter-Objekte gemacht werden. Standardmäßig gelten jetzt für alle CursorAdapter-Objekte die gleichen SET-Einstellungen, die auch für Formulare gelten.

Die neue Methode *OnSetEnv* aus der Cursoradapter-Klasse *cBaseDataAccess* überprüft, ob das Formular, das den CursorAdapter instanziiert, eine Methode *OnSetEnv* besitzt. Wenn diese Methode existiert und wenn diese Methode nicht bereits von einem anderen CursorAdapter aufgerufen wurde, wird sie ausgeführt. Wenn das Formular keine *OnSetEnv*-Methode hat, wird überprüft, ob das Objekt

goEnvironment existiert. Wenn dieses Objekt existiert, wird dessen Methode *SetDataEnvironment* aufgerufen. In allen anderen Fällen werden fest codierte SET-Einstellungen ausgeführt.

In der Klasse *cAppDataAccess* in der Klassenbibliothek *Appl.vcx* können eigene SET-Einstellungen hinzugefügt werden.

Das neue Objekt goPath

Zur Laufzeit einer Anwendung wird das neue Objekt *goPath* instanziiert. Dieses Objekt hat Eigenschaften, deren Werte auf die aktuell verwendeten Pfade zeigen:

CDataDir – Pfad zur aktuell verwendeten Datenbank.

ClientName – Name der aktuellen Datenbank. Dies ist der für den Benutzer im Mandantenauswahldialog sichtbare Name. Dies ist nicht unbedingt der physikalische Name der Datenbank.

VfxPath – Pfad zu den VFX Systemtabellen.

ReportPath – Pfad zu den Berichtsdateien.

UpdatePath – Pfad zu dem Ordner mit einer aktualisierten Datenbankstruktur.

ImportPath – Diese Eigenschaft wird von VFX nicht verwendet und steht zur freien Verfügung.

ExportPath – Diese Eigenschaft wird von VFX nicht verwendet und steht zur freien Verfügung.

Wenn die Tabelle *Vfxpath.dbf* zur Mandantenauswahl verwendet wird, wird jedes Feld dem Objekt goPath als Eigenschaft hinzugefügt.

Wenn die Datei *Config.vfx* zur Mandantenauswahl verwendet wird, wird jedes Feld dem Objekt goPath als Eigenschaft hinzugefügt.

So kann zur Laufzeit auf einfachem Weg auf die aktuellen Pfadeinstellungen zugegriffen werden.

Erweiterungen in der Klasse cBaseDataAccess

Neue Eigenschaften

cFieldsToWriteNULLWhenEmpty – Dieser Eigenschaft kann eine Liste von Feldern zugewiesen werden, deren Werte mit *NULL* ersetzt werden, bevor Daten in die Datenbank geschrieben werden.

cForeignKeyName – Diese Eigenschaft wird nur bei Child-CursorAdaptern auf OneToMany-Formularen verwendet. Hier wird der Name des Feldes im Parent-Arbeitsbereich gespeichert, das den Primärschlüssel enthält.

cForeignKeyValue – Diese Eigenschaft wird nur bei Child-CursorAdaptern auf OneToMany-Formularen verwendet. Mit diesem Ausdruck wird der Wert des Fremdschlüssels des neu gespeicherten Parent-Datensatzes ermittelt. Normalerweise ist dies das Feld mit dem Primärschlüssel aus dem Parent-Arbeitsbreich. Das Ergebnis der Evaluierung dieses Ausdrucks wird in dem Feld gespeichert, das in der Eigenschaft *cForeignKeyName* angegeben ist.

cWhereClause – Hier kann eine Where-Klausel angegeben werden. Zur Laufzeit wird dieser Wert an den Wert aus der Eigenschaft *SelectCmd* angefügt. Anschließend wird die *CursorFill*-Methode aufgerufen. Da dieser Wert auf Formularebene eingegeben werden kann, bleibt die CursorAdapter-Klasse unverändert. Änderungen an der CursorAdapter-Klasse, insbesondere wegen Strukturänderungen der Basistabellen, können unabhängig von der Where-Klausel durchgeführt werden.

lWriteNULLWhenEmptyForDBC – Wenn der Wert dieser Eigenschaft mit *.T.* eingestellt wird, werden leere Felder der Typen *Date* und *Datetime* mit dem Zustand *NULL* in der Datenbank gespeichert. Hierdurch können leere Datumswerte in einer Datenbank gespeichert werden, die später in eine SQL Datenbank portiert werden soll. Der Standardwert dieser Eigenschaft ist *.F.* Diese Eigenschaft wird nur verwendet wenn der *DataSourceType* eines CursorAdapter *Native* ist.

Neue Methode

OnSetEnv- Diese Methode wird während der Initialisierung aufgerufen.

Die neue Methode *OnSetEnv* überprüft, ob das Formular, das den CursorAdapter instanziiert, eine Methode *OnSetEnv* besitzt. Wenn diese Methode existiert und wenn diese Methode nicht bereits von einem anderen CursorAdapter aufgerufen wurde, wird sie ausgeführt. Wenn das Formular keine *OnSetEnv*-Methode hat, wird überprüft, ob das Objekt goEnvironment existiert. Wenn dieses Objekt existiert, wird dessen Methode SetDataEnvironment aufgerufen. In allen anderen Fällen werden fest codierte SET-Einstellungen ausgeführt.

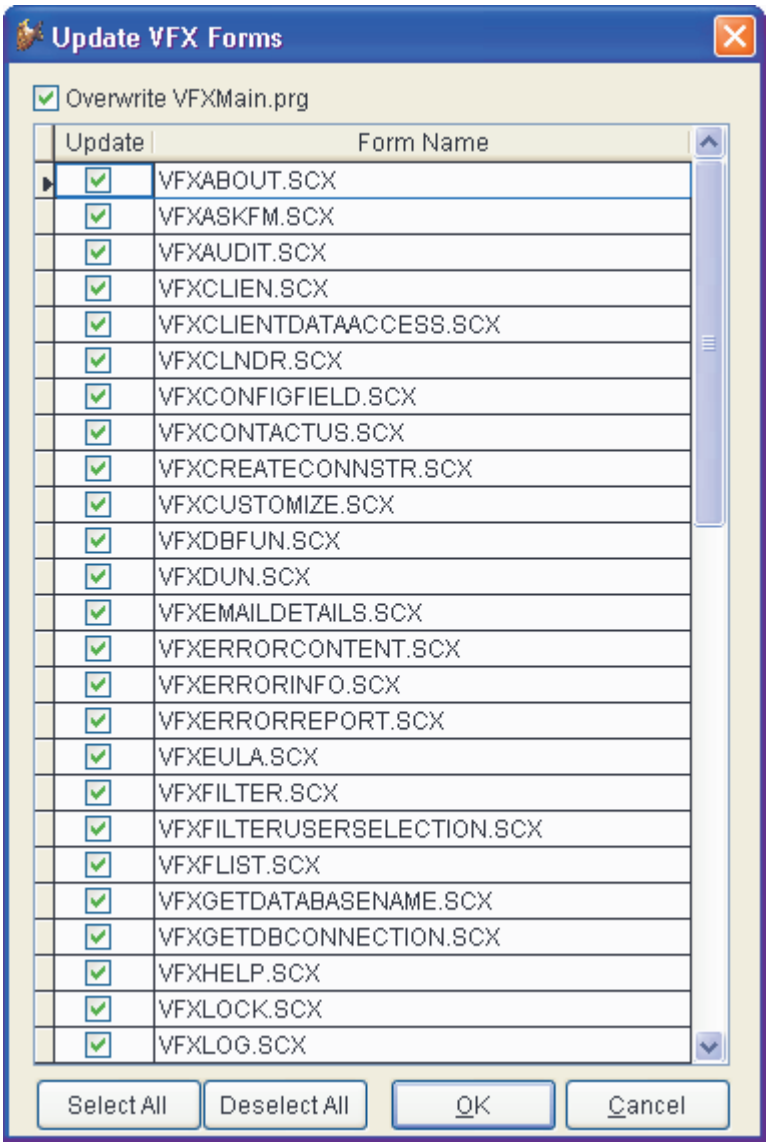
Builder und Wizards

VFX – Task Pane

In der VFX – Task Pane befindet sich in der Symbolleiste eine neue Schaltfläche. Hierüber kann der VFX – Update Project Wizard gestartet werden.

VFX – Update Project Wizard

Vor der Aktualisierung wird eine Archivdatei vom Projektordner mit allen Unterordnern angelegt. Der Dateiname besteht aus dem Projektnamen, dem aktuellen Datum im ANSI-Format sowie der VFX-Version, die das Projekt vor der Aktualisierung hat.



In einem anschließend erscheinenden Dialog kann eingestellt werden, welche der VFX-Formulare bei der Aktualisierung überschrieben werden sollen. Entwickler, die in VFX-Formularen Änderungen gemacht haben, dürfen diese Formulare nicht überschreiben. Die Einstellungen in diesem Dialog werden für spätere Aktualisierungen gespeichert und bleiben so erhalten.

VFX – Application Builder

Alle neuen Eigenschaften der Klassen *cFoxAppl* und *cAppUpdateEngine* können im VFX – Application Builder bearbeitet werden.

Zu jeder Eigenschaft wird im Builder ein Tooltip mit dem Namen der bearbeiteten Klasse und Eigenschaft im Format *Klasse.Eigenschaft* angezeigt.

Der VFX – Application Builder wurde um eine Suchfunktion erweitert. Damit ist es möglich nach jedem Text, der in einer Bezeichnung im Builder vorkommt, zu suchen.

VFX – Upsizing Wizard

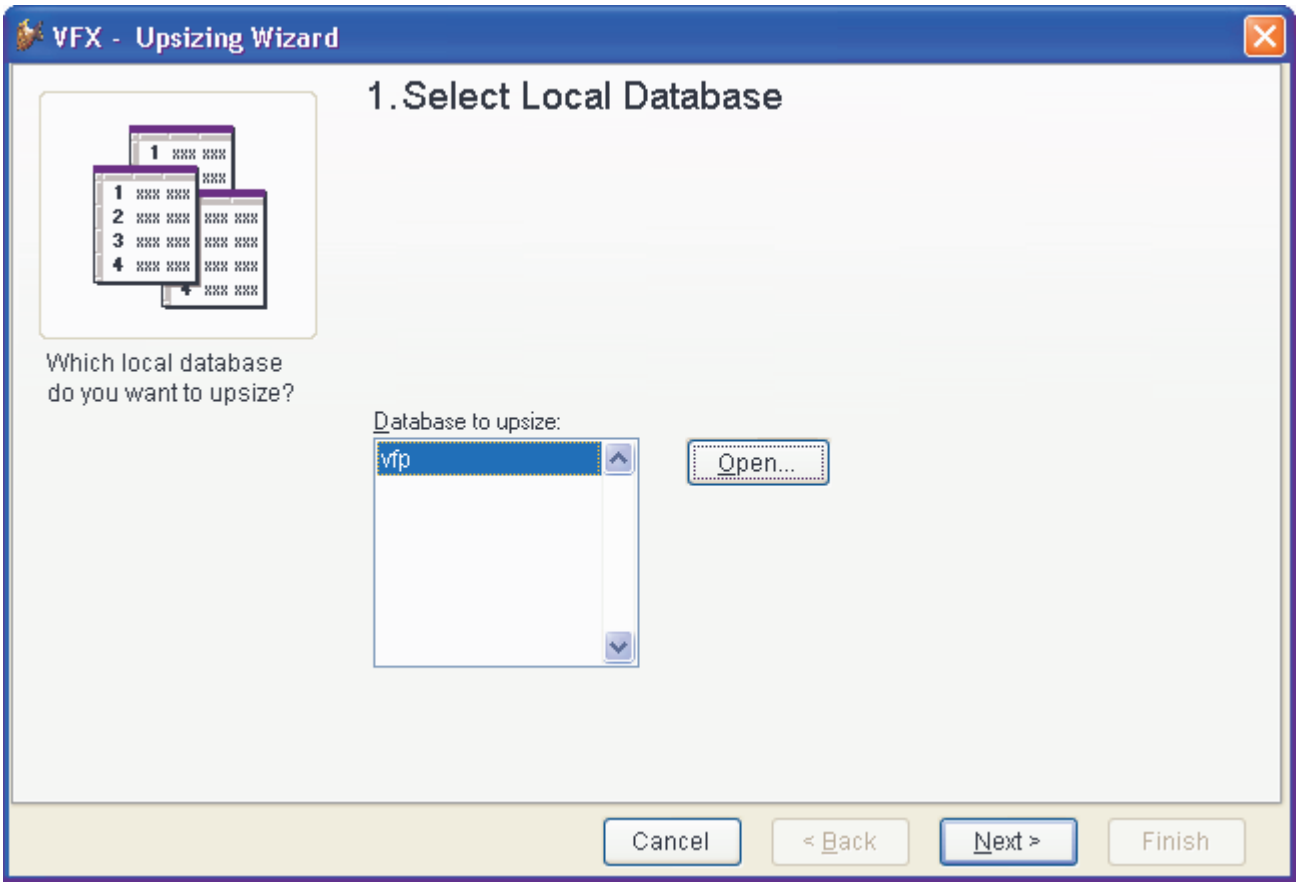
Der VFX – Upsizing Wizard entspricht in seinem Funktionsumfang etwa dem VFP – Upsizing Wizard. Mit dem VFX – Upsizing Wizard können jedoch auch ID-Werte korrekt in eine Remote Datenbank portiert werden. Die Portierung von Daten ist ohne Codepage-Konflikte möglich. Auch Feldnamen, die im SQL Server geschützte Befehlsworte sind, können portiert werden. Standardmäßig sind in allen Feldern Nullwerte erlaubt. Im Gegensatz zum VFP – Upsizing Wizard wird standardmäßig eine neue Datenbank angelegt und nicht vorgeschlagen eine vorhandene Datenbank zu überschreiben. Und schließlich erfordert der VFX – Upsizing Wizard keine DSN-Verbindung, sondern erlaubt beliebige Verbindungswege zur Remote Datenbank.

Der VFX – Upsizing Wizard ermöglicht es eine vorhandene VFP-Datenbank auf einen SQL Server zu portieren. Dabei wird die Struktur so genau wie möglich auf dem SQL Server abgebildet. Die Daten werden in die neue SQL Server-Datenbank übernommen. Auch die Portierung von Ansichten wird unterstützt.

Der VFX – Upsizing Wizard führt den Entwickler durch sechs Schritte:

1. Auswahl der lokalen Datenbank

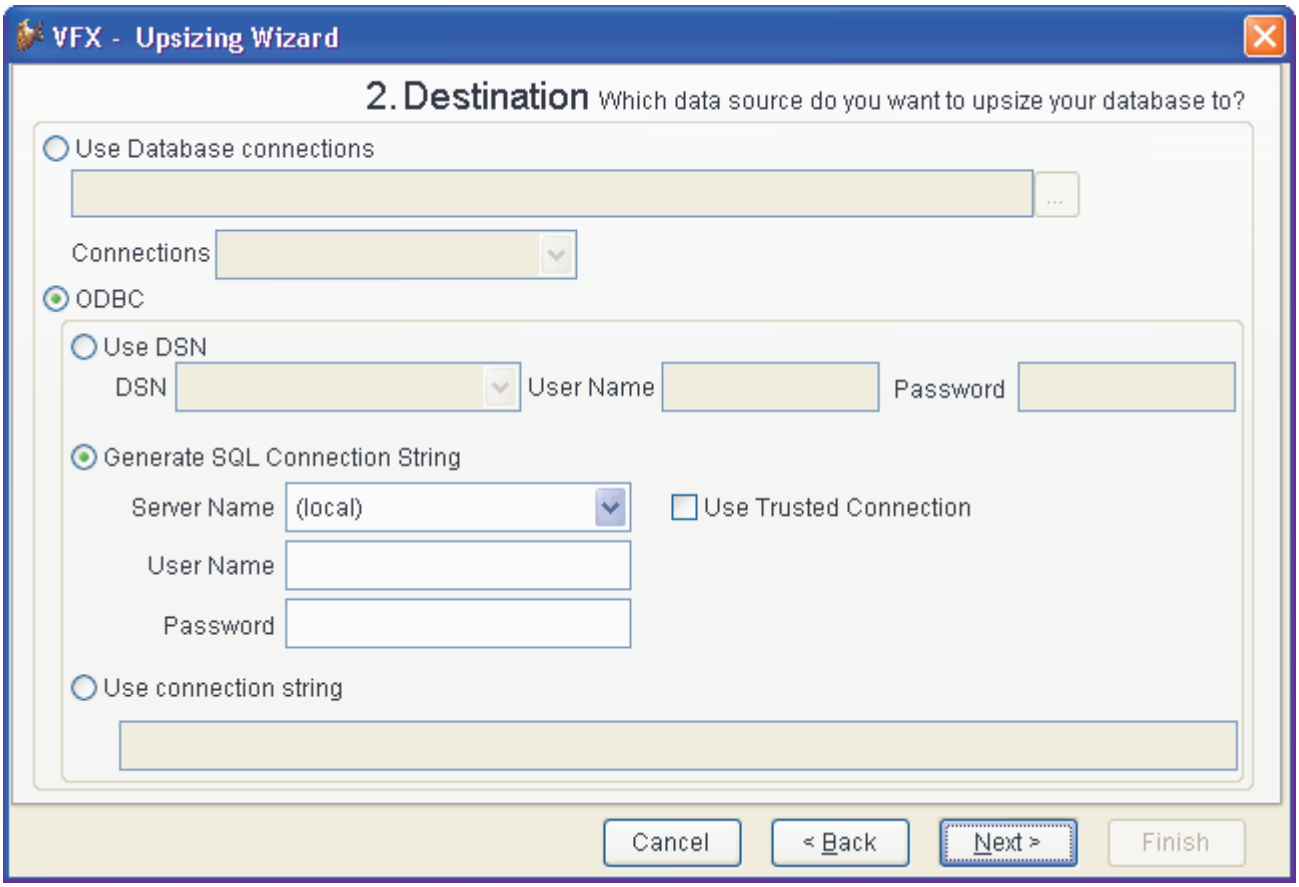
Im ersten Schritt wird die lokale Datenbank ausgewählt, die auf den SQL Server portiert werden soll.



Es wird eine Liste der zurzeit geöffneten Datenbanken angezeigt. Wenn die zu portierende Datenbank nicht geöffnet ist, kann sie hier über die Schaltfläche *Open* geöffnet werden.

2. Ziel

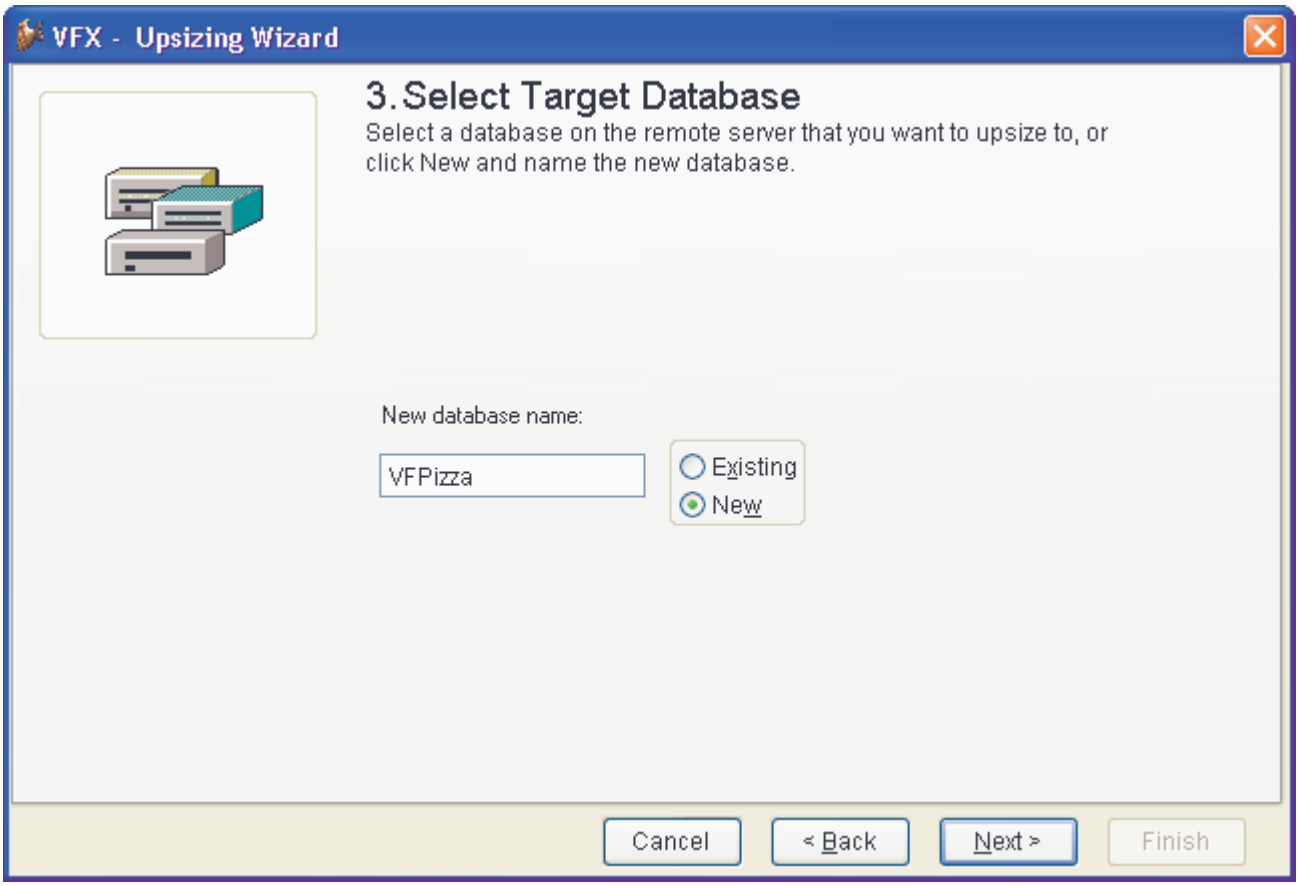
In diesem Schritt wird die Verbindung zum SQL Server angegeben. Es kann eine vorhandene Verbindung aus einem DBC verwendet werden. Es kann aber auch eine vorhandene DSN-Verbindung oder eine vorhandene Verbindungszeichenfolge verwendet werden.



Mit dem Wizard für Verbindungszeichenfolgen kann ebenfalls eine Verbindung hergestellt werden. Hierbei wird standardmäßig der lokal installierte SQL Server vorgeschlagen. Wenn der eingegebene Benutzername mit dem Kennwort nicht zu einer erfolgreichen Anmeldung führt, wird automatisch versucht eine vertrauenswürdig Verbindung mit den Windows-Anmeldedaten herzustellen. In der Regel sind auf dieser Seite des Assistenten also keine Eingaben erforderlich.

3. Eingabe des Datenbanknamens

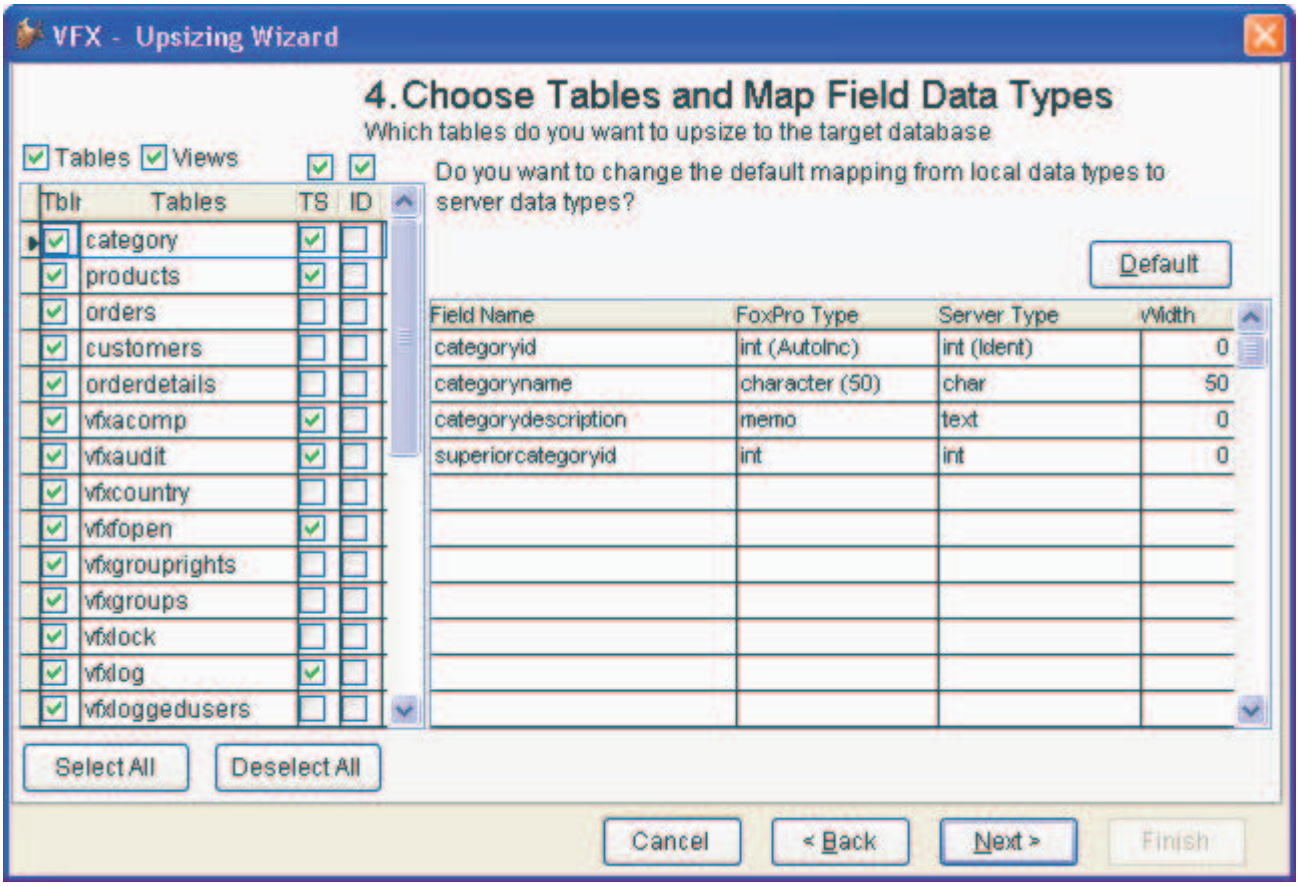
Wenn die Option *New* ausgewählt ist, kann der Name der neuen Datenbank eingegeben werden. Der Name muss ein gültiger Name für eine remote Datenbank sein.



Wenn die Option *Existing* ausgewählt wird, wird eine Liste der auf dem SQL Server vorhandenen Datenbanken angezeigt. Wählen Sie die Datenbank aus, die vom VFX – Upsizing Wizard aktualisiert werden soll.

4. Auswahl der Tabellen und Zuordnung der Datentypen

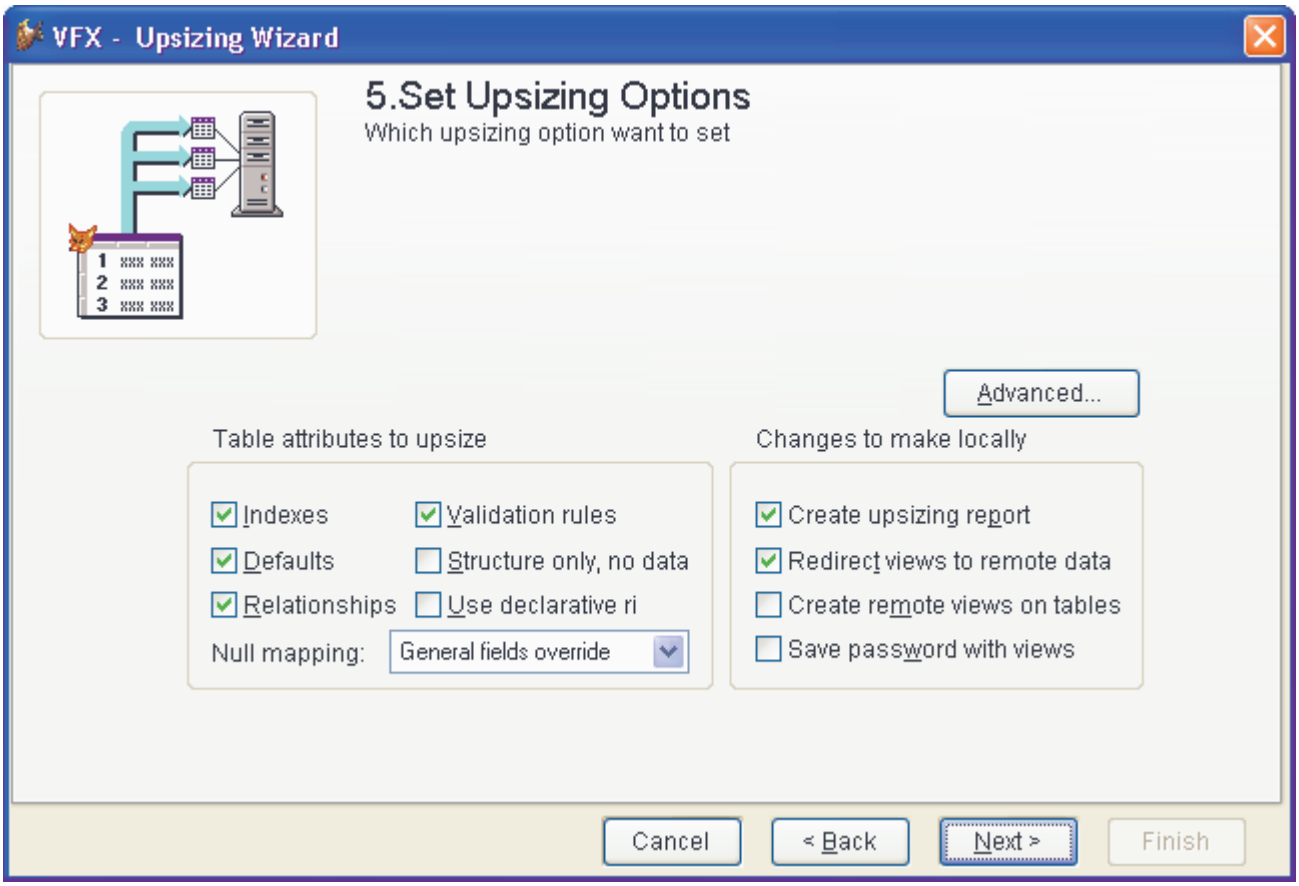
Hier kann ausgewählt werden welche Tabellen aus der VFP-Datenbank auf den SQL Server portiert werden sollen. Standardmäßig sind alle Tabellen zur Portierung markiert. Jeder Tabelle kann ein Timestamp-Feld sowie ein ID-Feld hinzugefügt werden. Der VFX – Upsizing Wizard fügt Tabellen mit Memo-Feldern standardmäßig ein Timestamp-Feld hinzu. Tabellen, die kein Primärschlüsselfeld enthalten, wird automatisch ein ID-Feld hinzugefügt. Diese Einstellungen können bei Bedarf je Tabelle geändert werden.



Zur selektierten Tabelle werden die Struktur in der VFP-Datenbank sowie die Struktur in der zu erstellenden SQL Server-Datenbank angezeigt. In der Regel wird für jeden Feldtyp eine sinnvolle Portierung vorgenommen. Bei Bedarf kann die Zuordnung des Datentyps hier geändert werden.

5. Portierungsoptionen

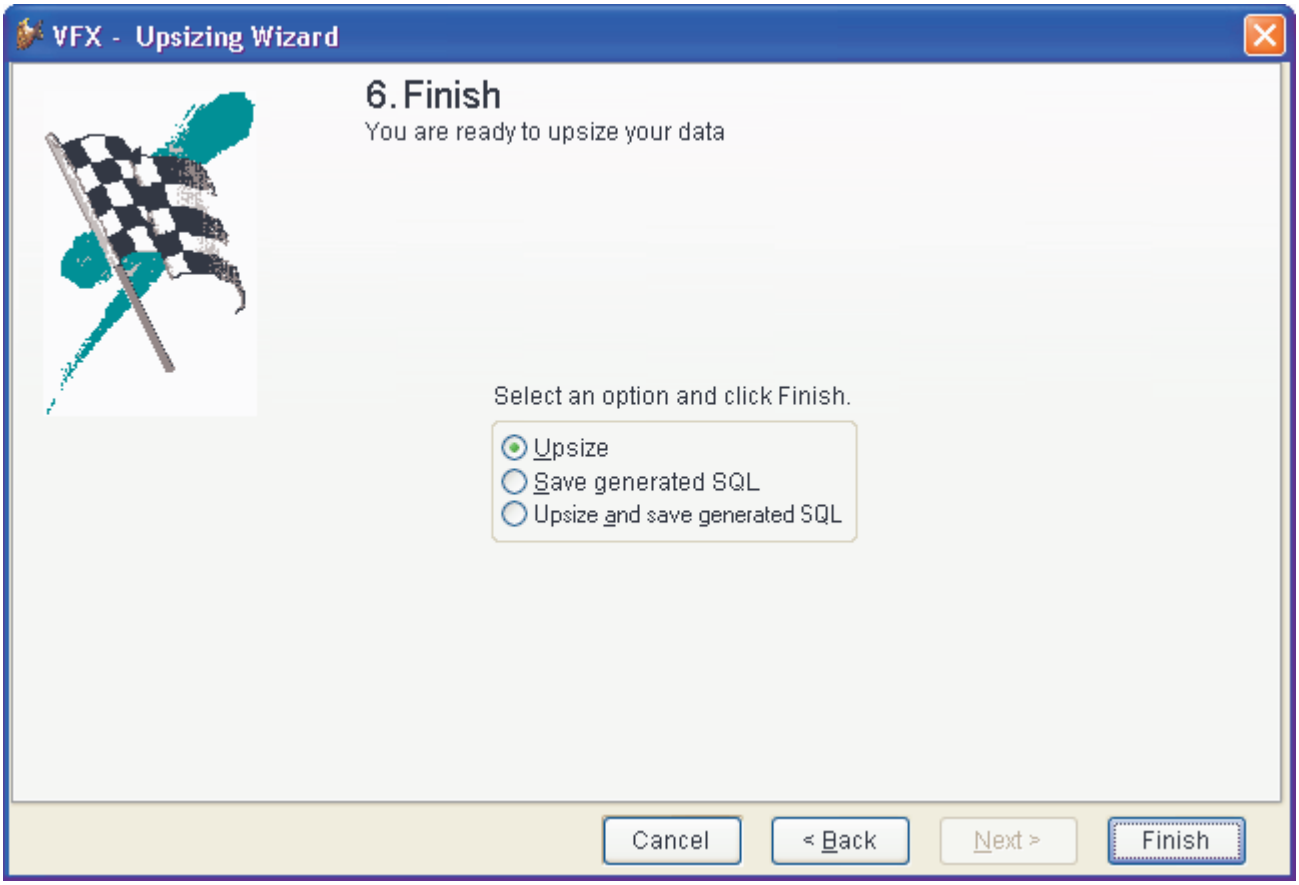
Standardmäßig werden die Strukturen von Tabellen sowie die Daten portiert. Es können auch Indexschlüssel, Standardwerte, Beziehungen (RI Constraints) und Validierungsregeln portiert werden. In der Combobox *Null mapping* kann eingestellt werden, ob Nullwerte erlaubt sind. Diese Option hilft sicherzustellen, dass Einfüge- und Aktualisierungsvorgänge erfolgreich durchgeführt werden können.



In diesem Schritt kann insbesondere auch eingestellt werden, ob ein Bericht über die Portierung erstellt werden soll. Der Bericht wird in ein neues VFP-Projekt eingefügt. Aus dem Bericht sind Probleme bei der Portierung ersichtlich.

6. Fertig

In diesem Schritt kann eingestellt werden, wie die Portierung durchgeführt werden soll.



Es kann eine der Optionen gewählt werden:

Upsize– Führt die Portierung wie oben beschrieben durch.

Save generated SQL– Generiert SQL Befehle, die für die Portierung erforderlich sind. Durch Ausführung dieser Befehle kann die eigentliche Portierung zu einem späteren Zeitpunkt durchgeführt werden.

Upsize and save generated SQL– Führt die Portierung wie oben beschrieben durch und generiert zusätzlich SQL Befehle, um die Portierung zu einem späteren Zeitpunkt wiederholen zu können.

Es ist eine gute Idee vor der Portierung eine Datensicherung durchzuführen. Während der Portierung werden Tabellen und lokale Ansichten aus der VFP-Datenbank umbenannt, um Tabellen und remote Ansichten mit den gleichen Namen in der SQL Server-Datenbank erstellen zu können.

Der VFX – Upsizing Wizard erlaubt es Felder vom Typ *Date* und *Datetime* mit leeren Werten in eine SQL Datenbank zu portieren. Wenn diese Felder in der SQL Datenbank den Zustand *NULL* erlauben, wird *NULL* in die SQL Datenbank geschrieben. Wenn der Zustand *NULL* nicht erlaubt ist, wird *01.01.1900* in die SQL Datenbank geschrieben.

VFX – CursorAdapter Wizard

CursorAdapter-Klassen können jetzt auch basierend auf Ansichten, gespeichert in einem DBC oder einer SQL Server Datenbank, erstellt werden.

Der Wizard kann jetzt auch auf einzelne Tabellen oder Ansichten angewendet werden.

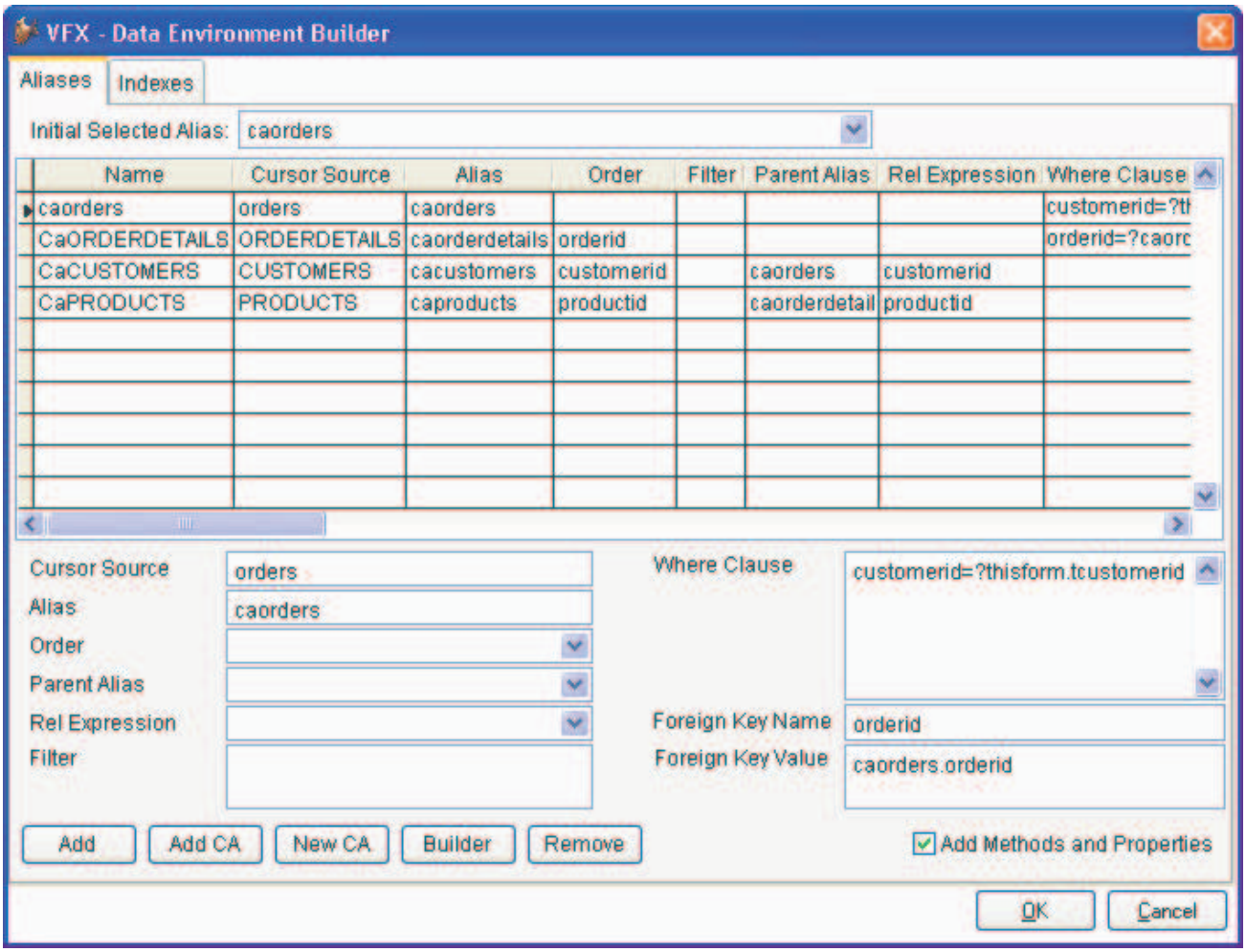
Für jedes Feld aus jeder Tabelle kann eine Typkonvertierung durchgeführt werden. Es stehen alle VFP-Feldtypen zur Verfügung.

Der VFX – ConnectionString Wizard steht jetzt auch im VFX – CursorAdapter Wizard zur Verfügung.

Der VFX – CursorAdapter Wizard fügt der Datei *Config.vfx* automatisch einen Verbindungseintrag zur ausgewählten Remote Datenbank hinzu.

VFX – Data Environment Builder

Der VFX – Data Environment Builder kann jetzt als eigenständiger Builder auf *Dataenvironment*-Klassen eingesetzt werden. Bereits in VFX 9.0 wurden *Dataenvironment*-Klassen für Formulare unterstützt. In VFX 9.5 kann nun der VFX – Data Environment Builder zur Bearbeitung dieser eigenständigen Klassen eingesetzt werden.



Unterhalb des Grid können jetzt alle Einstellungen zum aktuellen Cursor zusätzlich in Textboxen, Editboxen und Comboboxen bearbeitet werden. Längere Zeichenketten, wie zum Beispiel für einen Filterausdruck oder eine Where-Klausel können so einfacher bearbeitet werden.

Im VFX – Data Environment Builder gibt es einige neue Einstellmöglichkeiten:

- *Where Clause* – Dieser Wert wird nur bei Verwendung von CursorAdaptern berücksichtigt. In dieser Spalte kann eine Where-Klausel eingetragen werden. Der eingegebene Wert wird in der Eigenschaft *cWhereClause* des CursorAdapters gespeichert. Zur Laufzeit wird diese Where-Klausel automatisch dem Select-Befehl aus der Eigenschaft *SelectCMD* eines CursorAdapters hinzugefügt, bevor der CursorAdapter mit Daten gefüllt wird. Dies hat den Vorteil, dass die Eigenschaft *SelectCMD* nicht auf Formularebene verändert werden braucht und der Wert in jedem Fall aus der basierenden CursorAdapter-Klasse vererbt wird. Bei einer Veränderung der Struktur der zugrunde liegenden Tabellen kann die CursorAdapter-Klasse mit dem VFX – CursorAdapter Wizard aktualisiert werden. Änderungen an Formularen sind nicht mehr erforderlich.

Wenn mit CursorAdaptern gearbeitet wird und Primärschlüssel von der Datenbank vergeben werden, müssen in einem 1:n Szenario die vergebenen Primärschlüssel für den Parent-Datensatz als Fremdschlüssel in den Child-Datensätzen gespeichert werden. Dazu ist es erforderlich, dass zuerst der Parent-Datensatz gespeichert wird und nach dem Speichervorgang der Primärschlüssel aus der Datenbank gelesen wird. Im CursorAdapter für die Child-Daten muss der Fremdschlüssel spezifiziert werden. VFX speichert den Fremdschlüssel automatisch in allen Datensätzen eines Child-CursorAdapters. Zu diesem Zweck gab es bereits in VFX 9.0 in der VFX-CursorAdapter-Klasse die Eigenschaften *cForeignKeyName* und *cForeignKeyValue*. In VFX 9.5 werden die Werte dieser Eigenschaften automatisch ermittelt und vorbelegt und können im VFX – Data Environment Builder bearbeitet werden.

In VFP-Datenbanken kann zur Erzeugung von Primärschlüsseln der Datentyp *Integer (Autoinc)* und bei SQL Server-Datenbanken der Datentyp *Integer Identity* verwendet werden.

- *Foreign Key Name* – Hier wird der Name des Feldes angegeben, in dem der Fremdschlüssel gespeichert werden soll. Der Name des Feldes wird in der Eigenschaft *cForeignKeyName* gespeichert.

- *Foreign Key Value* – Hier wird der Name des Feldes aus dem Parent-CursorAdapter angegeben, das den neuen Primärschlüssel nach dem Speichern enthält. Der hier eingegebene Wert wird in der Eigenschaft *cForeignKeyValue* gespeichert. Hier kann auch ein Ausdruck eingegeben werden. Dieser Ausdruck wird evaluiert und dem in der Eigenschaft *cForeignKeyName* eingetragenen Feld zugewiesen.

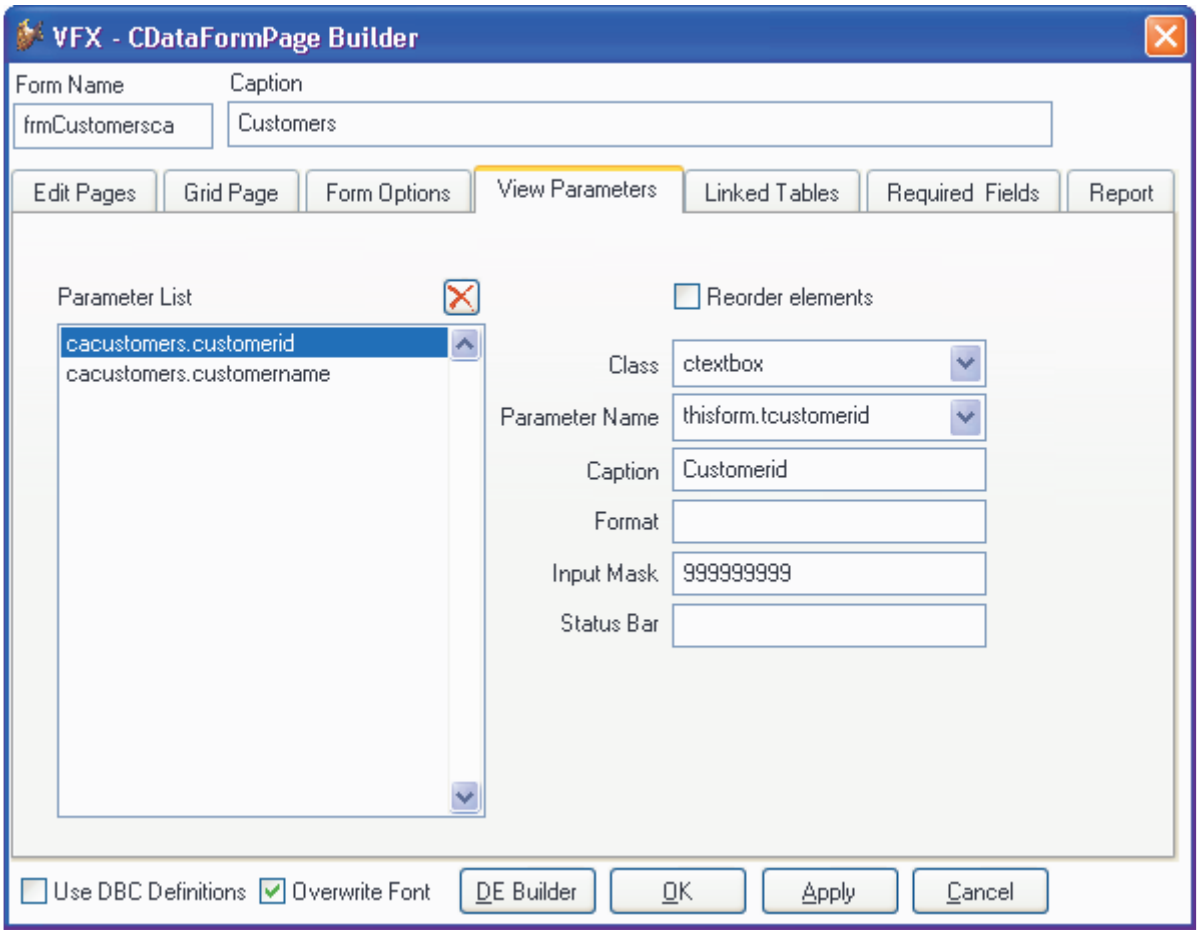
Wenn CursorAdapter verwendet werden, die nicht auf der VFX-Klasse *cAppDataAccess* basieren, aber die Eigenschaften des VFX – Data Environment Builder trotzdem genutzt werden sollen, kann das Kontrollkästchen *Add Methods and Properties* markiert werden. Hierdurch werden dem CursorAdapter die benötigten Eigenschaften automatisch hinzugefügt.

VFX – Form Builder

Wenn man im VFP – Formular-Designer den Builder startet, wird direkt der VFX – Form Builder gestartet. Der VFX – Data Environment Builder wahlweise durch einen Klick auf die Schaltfläche *DE Builder* oder direkt aus der Datenumgebung des VFP – Formular-Designers gestartet werden.

Neu in VFX 9.5 kann in allen Formular Buildern die basierende Klasse für Steuerelemente zur Eingabe von Ansichtsparametern ausgewählt werden.

Die in der Eigenschaft *cWhereClause* verwendeten Parameter eines CursorAdapters können aus der Combobox *Parameter Name* ausgewählt werden. Zusätzlich stehen in dieser Combobox

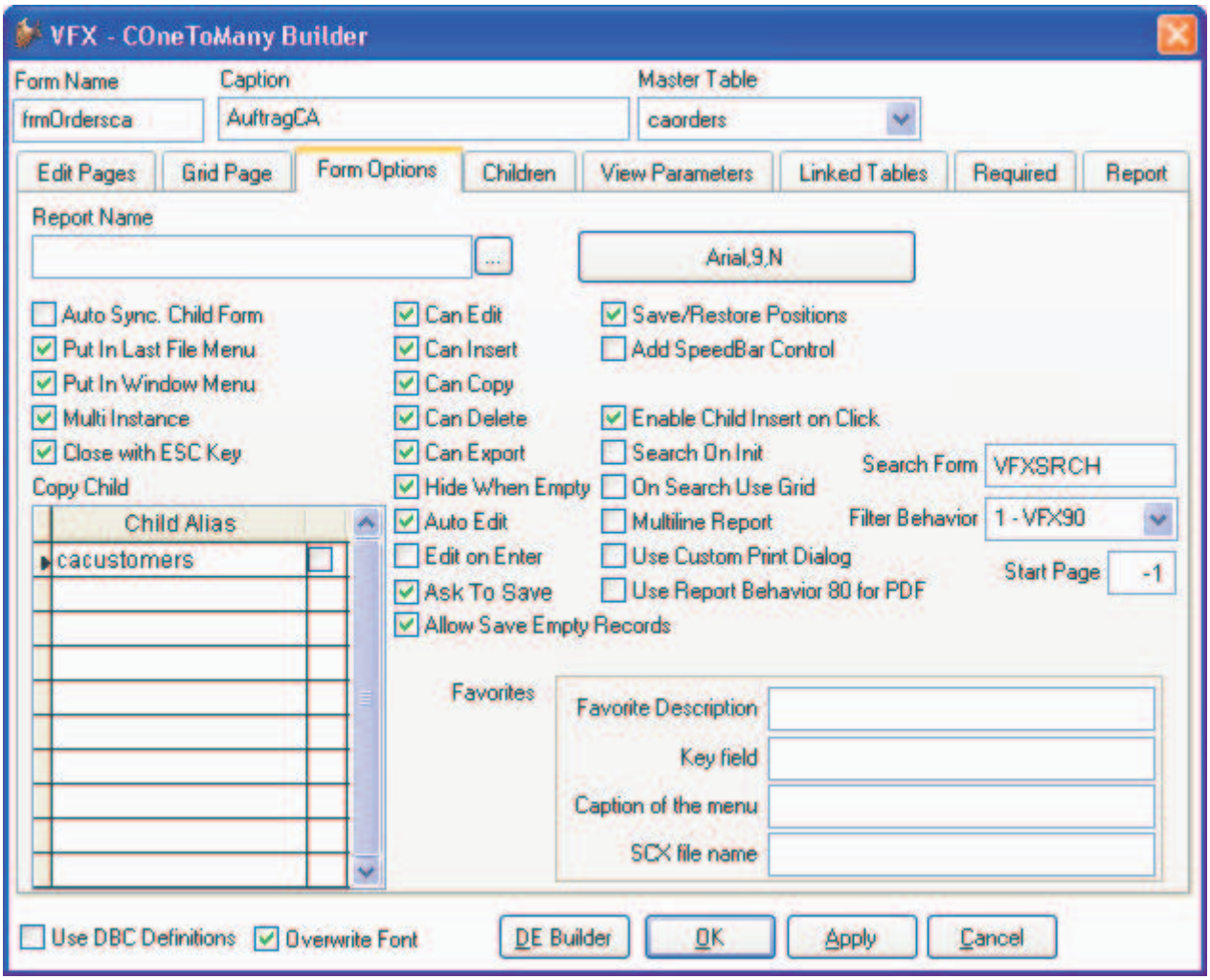


Die Einstellungen in den VFX – Form Buildern *Is Child Form* und *Has linked child form* werden nicht mehr benötigt und sind redundant. In VFX 9.5 kann jedes Formular ohne besondere Einstellung als Parent- und als Child-Formular dienen.

Wenn Formulare basierend auf CursorAdaptern als Datenquellen erstellt werden, lesen die VFX – Form Builder jetzt die Eigenschaften aus den zugrunde liegenden Tabellen aus, wenn die Datenbanken zur Verfügung stehen.

Die VFX – Form Builder erstellen jetzt bei Seitenrahmen lokalisierte Überschriften für alle Seiten.

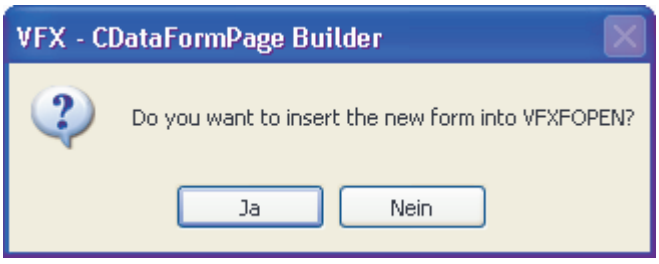
Eine Vielzahl neuer Eigenschaften kann auf der Seite *Form Options* der Form Builder eingestellt werden.



Mit dem Kontrollkästchen *Allow Save Empty Records* kann eingestellt werden, ob neue, leere Datensätze beim Speichern automatisch und ohne Rückfrage verworfen werden sollen. Entsprechend der Markierung wird der Wert der Formulareigenschaft *lAllowSaveEmptyRecords* gesetzt. Der Standardwert ist *.T.*, leere Datensätze dürfen gespeichert werden.

Alle Einstellungen, um einen Datensatz dem Favoriten-Menü hinzuzufügen, können jetzt in Textboxen auf der Seite *Form Options* gemacht werden.

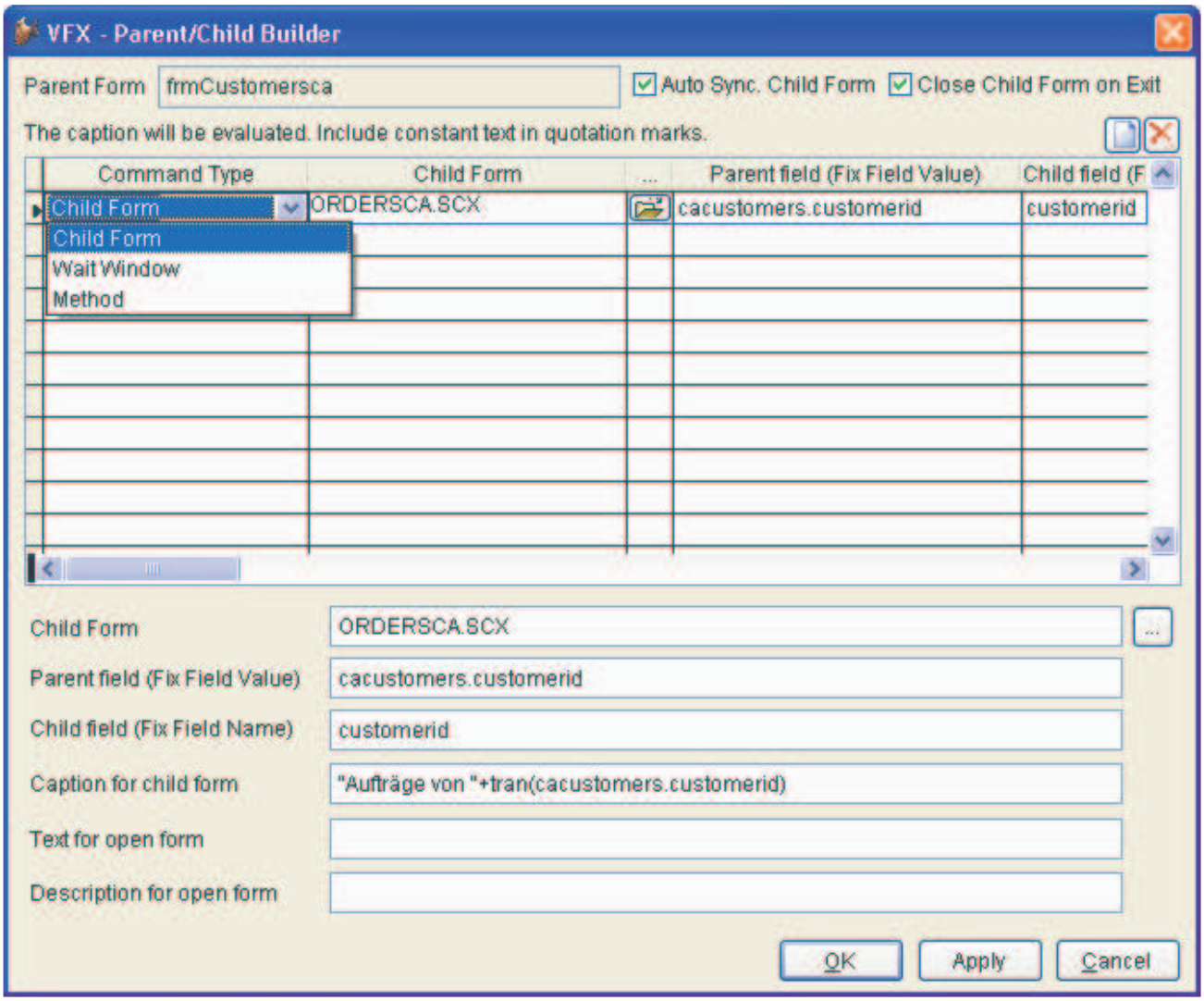
Beim erstmaligen Generieren eines neuen Formulars wird gefragt, ob das neue Formular in die Tabelle *Vfxfopen* eingetragen werden soll. Wenn diese Frage mit *Ja* beantwortet wird, kann das neue Formular über den Öffnen-Dialog gestartet werden.



VFX – Parent/Child Builder

In früheren VFX-Versionen war es möglich in der *OnMore*-Methode Child-Formulare oder Methoden des Parent-Formulars aufzurufen oder auch ein *Wait Window* anzuzeigen. Alle diese drei Aufrufmöglichkeiten können in VFX 9.5 auch im VFX – Parent/Child Builder verwaltet werden. Dafür kann im Grid in der Spalte *Command Type* eine der drei Aufrufmöglichkeiten ausgewählt werden.

Zusätzlich zur Bearbeitungsmöglichkeit im Grid können jetzt alle Werte wahlweise auch in Textboxen unterhalb des Grids eingegeben werden.



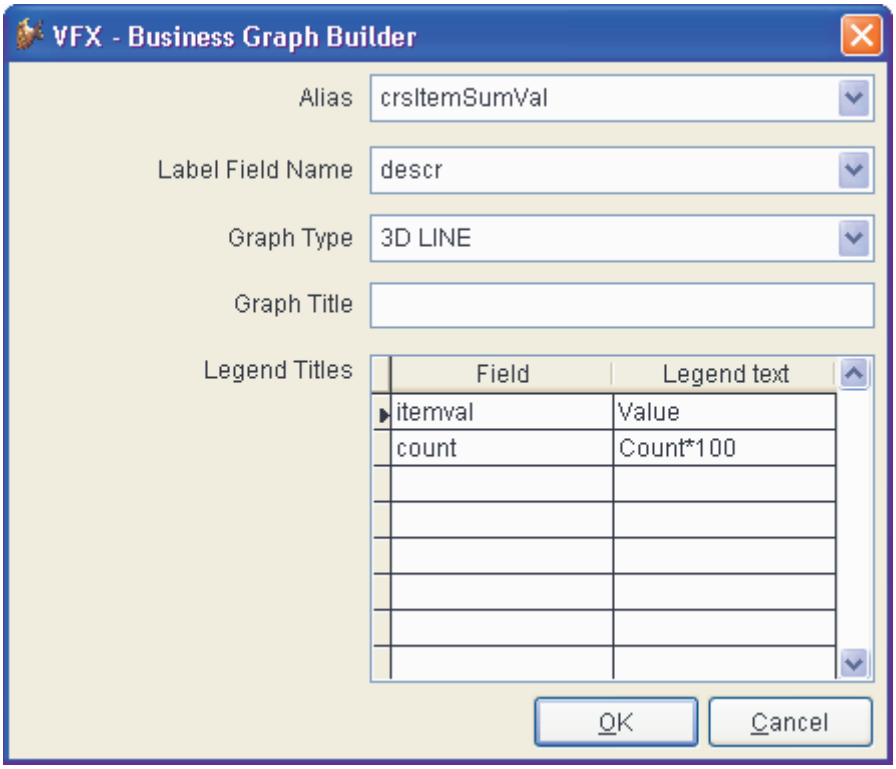
Wenn das Kontrollkästchen *Auto Sync. Child Form* markiert ist, werden beim Bewegen des Satzzeigers im Parent-Formular automatisch alle Child-Formulare synchronisiert.

Mit dem Kontrollkästchen *Close Child Form on Exit* kann eingestellt werden, ob beim Schließen des Parent-Formulars automatisch auch alle geöffneten Child-Formulare geschlossen werden sollen.

Bei der Auswahl eines Child-Formulars versucht der VFX – Parent/Child Builder möglichst die Informationen aus dem Parent- und dem Child-Formular zu ermitteln, um die folgenden Spalten im Grid richtig auszufüllen. In VFX 9.5 werden dabei auch Parent/Child-Szenarien basierend auf CursorAdaptern unterstützt. Als *Parent Field* wird dabei stets das Feld mit dem Primärschlüssel aus dem Parent-Formular vorgeschlagen.

VFX – Business Graph Builder

Mit dem neuen VFX – Business Graph Builder können alle Eigenschaften von *cBusinessGraph* Objekten eingestellt werden.



Eine Geschäftsgrafik wird basierend auf einem Arbeitsbereich erstellt. Dieser Arbeitsbereich kann aus der Combobox im Builder aus den Datenquellen der Datenumgebung ausgewählt oder manuell eingegeben werden. Der Arbeitsbereich wird in der Eigenschaft *cAliasName* gespeichert.

In der Combobox *Label Field Name* werden alle Felder des ausgewählten Arbeitsbereichs aufgeführt. Hier kann ein Feld ausgewählt werden, das als Beschriftung für die Datenreihen verwendet wird. Der ausgewählte Feldname wird in der Eigenschaft *cLabelField* gespeichert.

Der Darstellungstyp wird in der Combobox *Graph Type* ausgewählt. Es kann aus den Typen 3D BAR, 2D BAR, 3D LINE, 2D LINE, 3D AREA, 2D AREA, 3D STEP, 2D STEP, 3D COMBINATION, 2D PIE und 2D XY gewählt werden. Der ausgewählte Wert wird in der Eigenschaft *nGraphType* gespeichert.

In der Textbox *Graph Title* kann eine Überschrift für die Geschäftsgrafik eingegeben werden. Die Überschrift wird in der Eigenschaft *cGraphTitle* gespeichert.

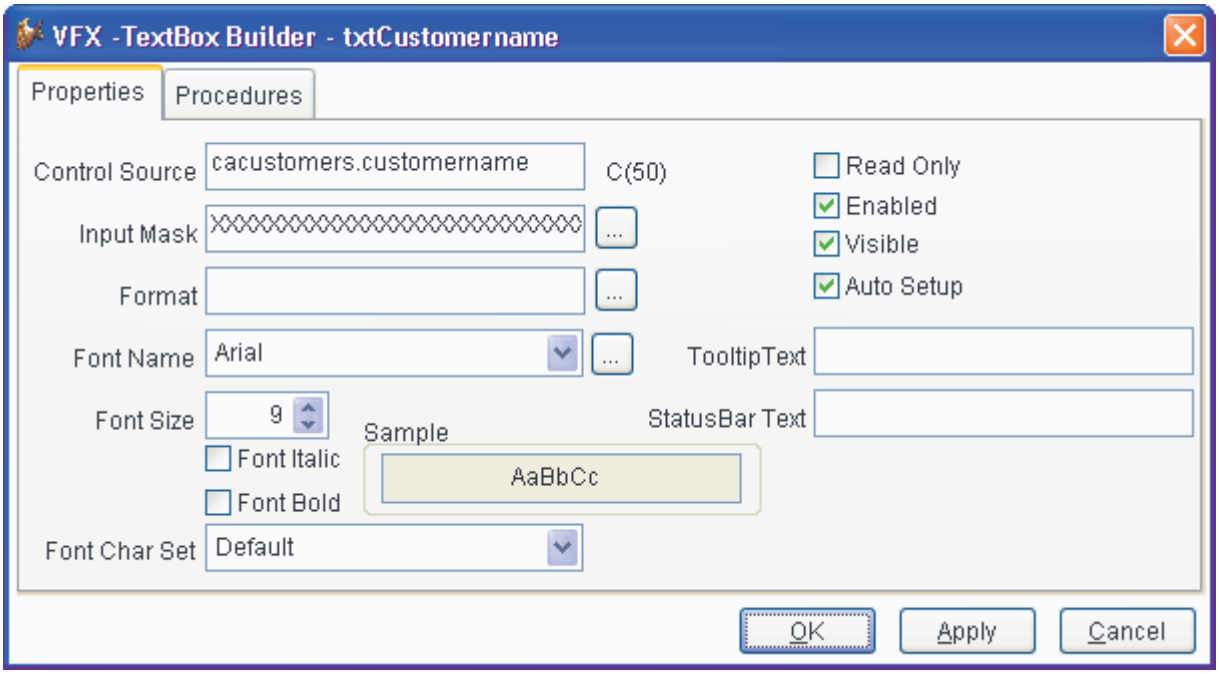
Im Grid wird für jedes Feld aus dem ausgewählten Arbeitsbereich eine Zeile angezeigt, ausgenommen ist das Feld, das als Beschriftung für die Datenreihen dient. Die Daten aus allen diesen Feldern werden in der Geschäftsgrafik angezeigt. In der ersten Spalte des Grids werden die Namen der Felder angezeigt. In der zweiten Spalte sollte für jedes Feld eine Bezeichnung angegeben werden. Diese Bezeichnungen werden in der Legende der Geschäftsgrafik angezeigt. Die Legendenbezeichnungen werden in einer Komma-Separierten Liste entsprechend der Reihenfolge der Felder in der Eigenschaft *cLegendTitles* gespeichert.

VFX – TextBox Builder

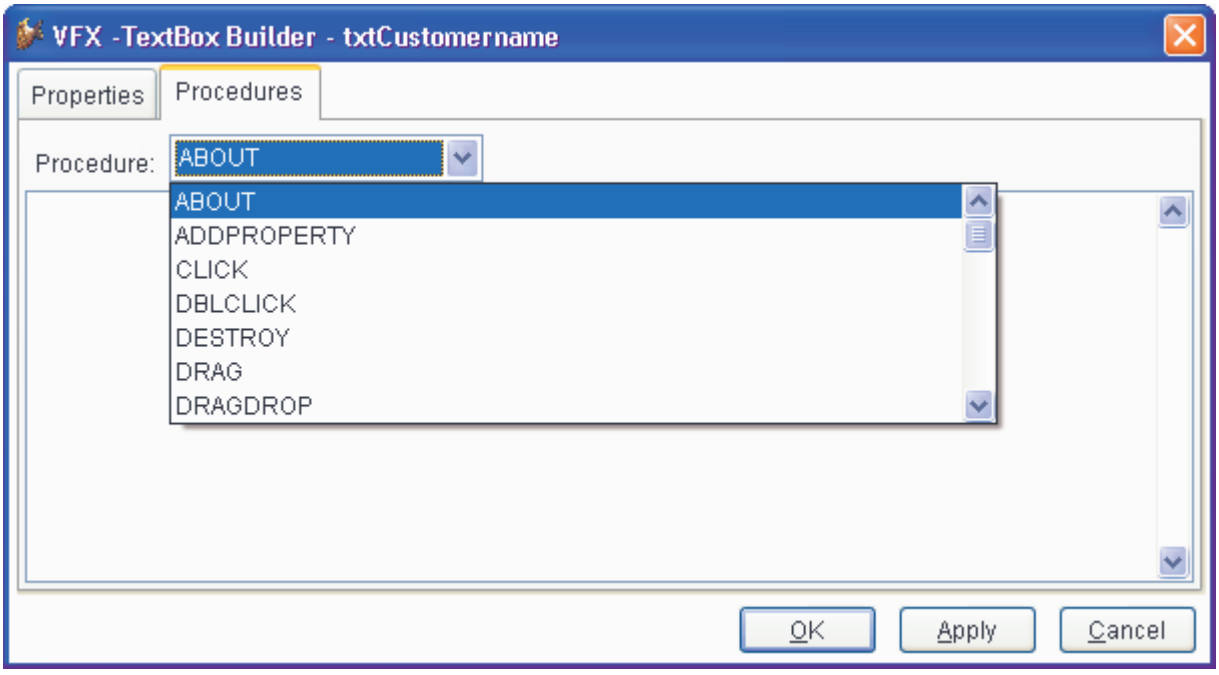
Im neuen VFX – TextBox Builder können die wichtigsten Eigenschaften von allen Steuerelementen basierend auf der VFP Basisklasse Textbox eingestellt werden. Auch die Bearbeitung sämtlicher Methoden ist hier möglich.

Der VFX – TextBox Builder kann aus dem VFX Menü über den Menüpunkt *VFX Power Builders* gestartet werden, wenn eine Textbox das ausgewählte Steuerelement im VFP Formular-Designer ist. Wahlweise kann der Builder auch aus dem Kontextmenü der Textbox gestartet werden.

Auf der Seite *Properties* können insbesondere die Schriftarteinstellungen gemacht werden. Im *Sample* Feld wird eine Vorschau auf die eingestellten Werte gegeben.



Auf der Seite *Procedures* kann der Code aller Methoden und Ereignisse bearbeitet werden.

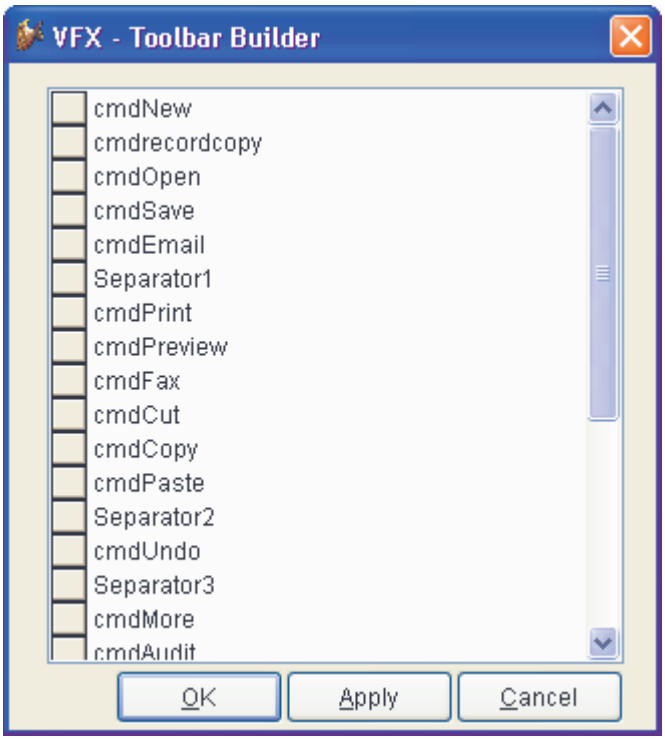


VFX – Toolbar Builder

Ein weiterer neuer Builder in VFX ist der Builder für Symbolleisten.

Der VFX – Toolbar Builder erleichtert es Steuerelemente und Separatoren innerhalb von Symbolleisten in eine neue Reihenfolge zu bringen. Dazu können die Elemente in einer Listbox nach oben oder unten bewegt werden.

Der VFX – Toolbar Builder kann im VFP Klassen-Designer mit der Auswahl Builder aus dem Kontextmenü gestartet werden, wenn eine Symbolleisten-Klasse geöffnet ist.



Die Anordnung der Elemente in der Listbox entspricht der Reihenfolge der Elemente in der Symbolleiste. Um die Reihenfolge zu ändern, können die Einträge mit den Mover-Bars verschoben werden.

Lokalisierung

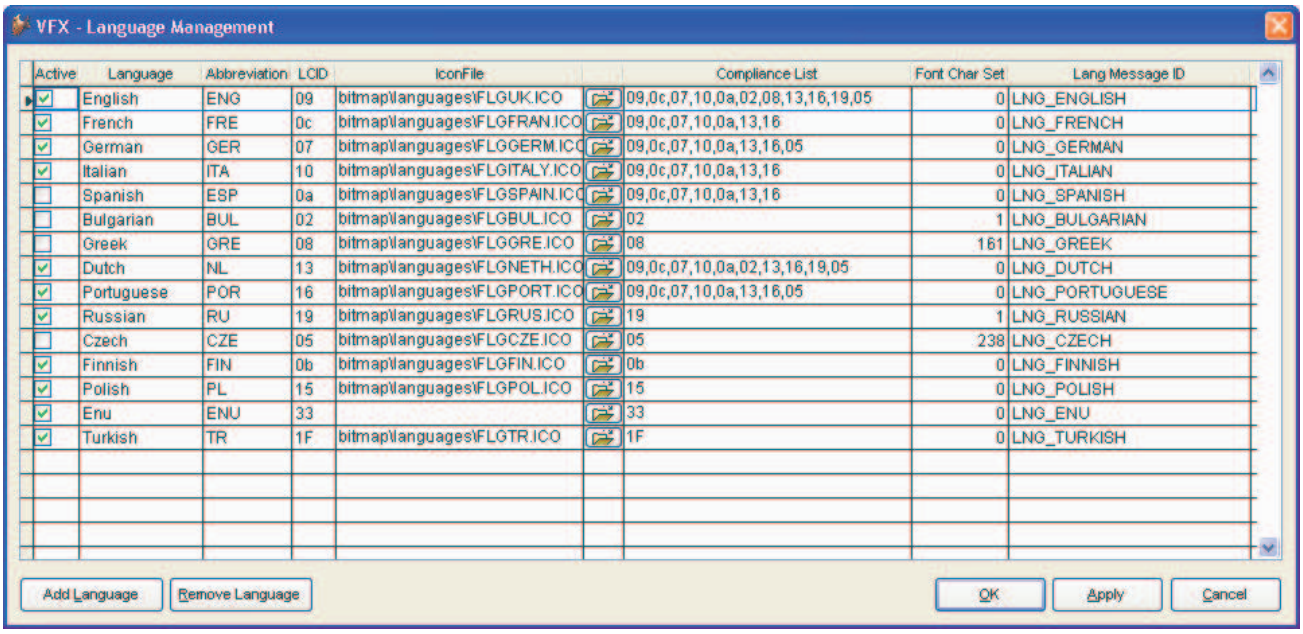
Entsprechend der gewählten Sprache werden jetzt auch die Einstellungen für das Datumsformat und das Zeitformat angepasst. Um diese Lokalisierung zu aktivieren, muss im VFX – Application Builder das *Date format* auf *VFX LOCALIZED* eingestellt werden. Wahlweise kann manuell in der Klasse *cFoxAppl* der Eigenschaft *cDateFormat* der Wert *VFX LOCALIZED* zugewiesen werden.

VFX - Language Management Builder

Dieser Builder verwaltet die verfügbaren Sprachen für eine Anwendung, die mit Lokalisierung zur Laufzeit arbeitet. Die Informationen über die zur Verfügung stehenden Sprachen sind in *VFXLanguage.dbf* gespeichert.

Bei der Generierung einer neuen Anwendung werden alle Sprachen, die mit VFX geliefert werden, in das neue Projekt kopiert. Hier können Sprachen hinzugefügt, aber auch gelöscht werden. Die mit VFX gelieferten Sprachen können jedoch nicht gelöscht werden, sondern nur als nicht aktiv gekennzeichnet werden. Als nicht aktiv gekennzeichnete Sprachen stehen zur Laufzeit der Anwendung

nicht in den Comboboxen zur Sprachauswahl im Anmeldedialog und in der Standard-Symbolleiste zur Verfügung.



In der Spalte *Language* wird die Bezeichnung einer Sprache eingetragen, so wie sie in der Sprachauswahl-Combobox zur Laufzeit angezeigt werden soll.

In der Spalte *Abbreviation* wird der Name des Feldes in der Tabelle *Vfxmsg.dbf* eingetragen. Aus diesem Feld werden zur Laufzeit die Texte der gewählten Sprache gelesen.

Die Spalte *LCID* enthält den *Locale Identifier* der Sprache. Dies ist ein in Windows definierter Wert und wird für die Regionaleinstellungen verwendet.

In der Spalte *IconFile* kann der Name zu einer Icondatei ausgewählt werden. Das Icon sollte eine Flagge zur Veranschaulichung einer Sprache anzeigen. Das Icon wird in der Sprachauswahl-Combobox zur Laufzeit angezeigt.

Die Spalte *Compliance List* enthält eine durch Komma separierte Liste *Locale Identifiers* von Sprachen, die zur aktuellen Sprache kompatibel sind. Diese Liste enthält Werte von Regionaleinstellungen, die geeignet sind, die aktuelle Sprache korrekt anzuzeigen.

Die Spalte *Lang Message Id* enthält den Wert des Feldes *MessageID* aus der Tabelle *Vfxmsg.dbf,* die die lokalisierten Texte der aktuellen Sprache für die Sprachauswahl-Combobox enthält.

Die oben beschriebenen Einstellungen können für mit VFX gelieferte Sprachen nicht geändert werden. Diese Einstellungen können nur für neu hinzugefügte Sprachen bearbeitet werden. Eine neue Sprache kann über die Schaltfläche *Add Language* hinzugefügt werden. Eine so hinzugefügte Sprache wird in der Sprachauswahl-Combobox angezeigt. Für eine neu angelegte Sprache wird automatisch ein Feld in der Tabelle *Vfxmsg.dbf* mit Bezeichnung aus der Spalte *Abbreviation* angelegt.

Um eine neu hinzugefügte Sprache verwenden zu können, müssen alle Texte aus der Tabelle *Vfxmsg.dbf* in die neue Sprache übersetzt werden.

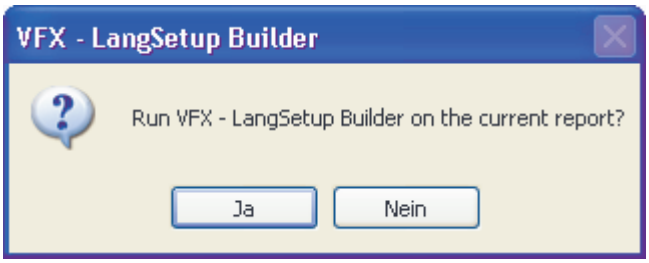
Ausführen der Methode LangSetup

Bei Aufruf einer *LangSetup*-Methode auf Formularebene werden automatisch alle auf dem Formular befindlichen Objekte nach Vorhandensein einer *LangSetup*-Methode durchsucht. Container-Objekte werden rekursiv durchsucht. Die *LangSetup*-Methode wird so in allen Objekten ausgeführt.

VFX - LangSetup Builder

Der VFX – LangSetup Builder unterstützt in VFX 9.5 nicht nur Formulare, sondern auch Berichte sowie den Öffnen-Dialog.

Um den VFX – LangSetup Builder auf einer Berichtsdatei anzuwenden, muss die Berichtsdatei zunächst im VFP Berichts-Designer geöffnet werden. Jetzt kann der VFX – LangSetup Builder aus dem VFX-Menü gestartet werden. Es erscheint folgende Messagebox:



Wenn mit „ja“ geantwortet wird, fügt der VFX – LangSetup Builder für jede Bezeichnung in dem Bericht einen Datensatz an die Tabelle *Vfxmsg.dbf* an. Die Bezeichnungen im Bericht werden durch „^“ gefolgt von der *Message_Id* des neuen Datensatzes in *Vfxmsg.dbf* ersetzt. Zur Laufzeit ersetzt ein ReportListener die Bezeichnungen im Bericht durch die lokalisierten Texte aus der Tabelle *Vfxmsg.dbf*.

Wenn nur ein Projekt, aber kein Formular oder Bericht im Designer geöffnet ist und der VFX – LangSetup Builder gestartet wird, wird die Tabelle *Vfxfopen.dbf* lokalisiert.

Project Hook

Bereits in VFX 9.0 wurden zur Laufzeit Objekte angelegt für alle Felder aus *Vfxsys.dbf* und aus dem aktuellen Datensatz aus *Vfxusr.dbf*. Aus Gründen der Kompatibilität mit früheren VFX-Versionen gibt es die Datei *Vfxglobal.h*, die Ersetzungen für alle Variablennamen enthält, die in früheren VFX-Versionen verwendet wurden. In VFX 9.5 wird die Datei *Vfxglobal.h* automatisch bei jeder Neuerstellung des Projekts oder beim Erstellen einer App- oder Exe-Datei neu angelegt. Dabei werden alle Felder, also auch selbst hinzugefügte, aus den Dateien *Vfxsys.dbf* und *Vfxusr.dbf* berücksichtigt.

Produktaktivierung

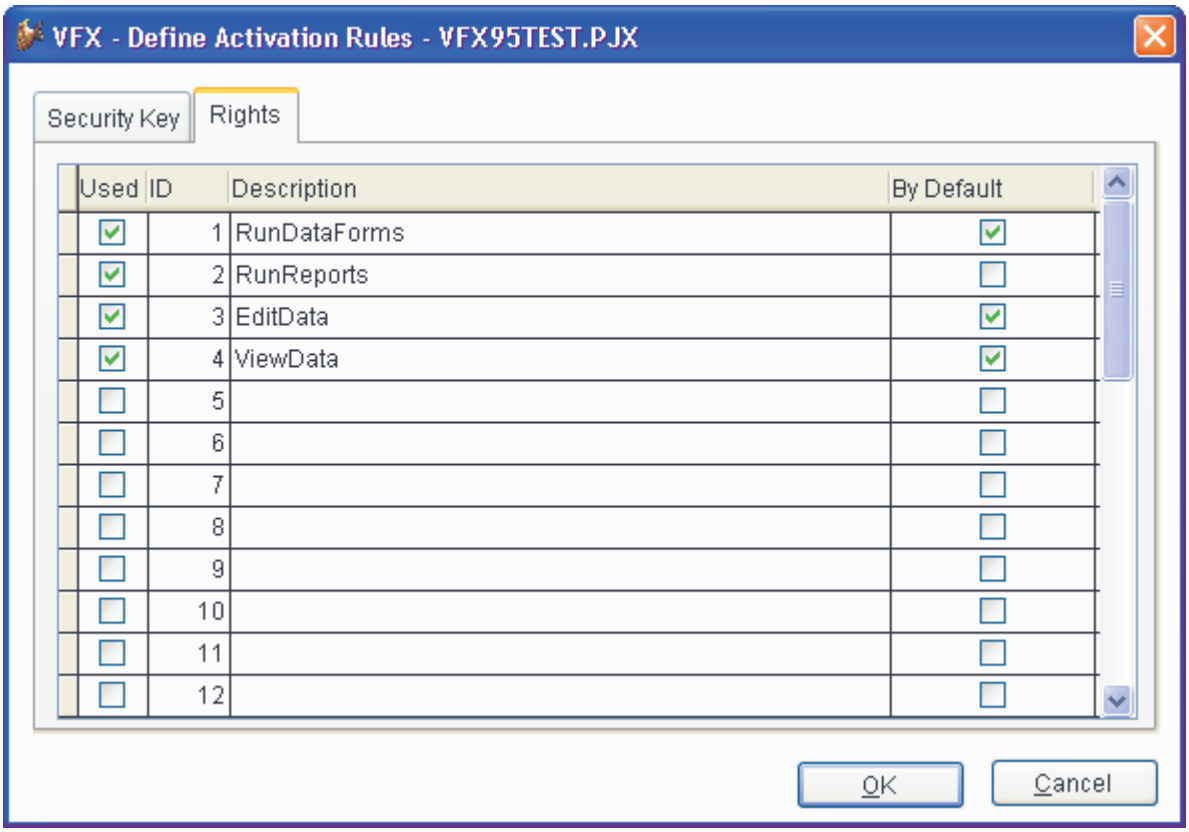
VFX 9.5-Anwendungen können vor unberechtigter Nutzung durch eine Produktaktivierung geschützt werden. Die Produktaktivierung funktioniert so, wie es im Benutzerhandbuch beschrieben ist. Hier werden nur die Erweiterungen in VFX 9.5 erläutert.

Für die in VFX integrierte Produktaktivierung steht jetzt ein Web Service zur Verfügung. Ähnlich wie bei der Aktivierung von VFX können sich Benutzer jetzt über einen Web Service einen Aktivierungsschlüssel für eine Anwendung holen.

Zur Verwaltung der Aktivierungsschlüssel und Kundendaten steht die neue Anwendung VFX – Kundenverwaltung zur Verfügung.

Definieren der Aktivierungsregeln

Zunächst müssen jedoch in der Anwendung die Aktivierungsregeln festgelegt werden. Dies geschieht im Dialog *VFX – Define Activation Rules*. Die Definition der Regeln geschieht genau so, wie in VFX 9.0. Die Aktivierungsregeln werden in der Klassenbibliothek *Appl.vcx* in der Klasse *cVfxActivation* in der Eigenschaft *cActPattern* verschlüsselt gespeichert.



Um ein Recht zu aktivieren muss zunächst die Checkbox in der ersten Grid-Spalte markiert werden. In der Spalte *Description* muss dem Recht ein Name gegeben werden. Zur Laufzeit wird dem Objekt *SecurityRights* eine Eigenschaft mit dem gewählten Namen hinzugefügt.

In der Spalte *By Default* kann eingestellt werden, ob dieses Recht standardmäßig aktiviert werden soll. Die Standardwerte gelten für neu angelegte Benutzer, können in der Kundenverwaltung je Benutzer geändert werden.

Die Registrierungsnummer ist ein numerischer Wert mit 10 Stellen Länge. Der Benutzer muss die Registrierungsnummer dem Entwickler mitteilen oder per E-Mail senden. Der Entwickler erfasst in der VFX – Kundenverwaltung einen neuen Datensatz für diesen Benutzer und gibt hier die Registrierungsnummer ein.

Aktivierungsschlüssel erstellen

Der Aktivierungsschlüssel enthält die Berechtigungen, für die einzelnen Module der Anwendung. Aktivierungsschlüssel für VFX 9.5-Anwendungen werden mit der VFX – Kundenverwaltung erstellt. Die VFX – Kundenverwaltung ist ein eigenständiges Projekt und wird mit VFX geliefert. Aus dem Projekt VFX – Kundenverwaltung kann eine Exe-Datei erstellt werden. Die Verwaltung der Kundendaten und die Generierung von Aktivierungsschlüsseln kann so auf einem PC unabhängig vom Entwicklungsrechner durchgeführt werden.

Damit die VFX – Kundenverwaltung Aktivierungsschlüssel erstellen kann, müssen ihr die Aktivierungsregeln bekannt sein. Die Aktivierungsregeln sind aber in der eigentlichen Anwendung in der Klassenbibliothek *Appl.vcx* gespeichert.

Bei der Generierung eines Aktivierungsschlüssels benutzt die VFX – Kundenverwaltung die Registrierungs-Dll. Das Projekt zur Erstellung der Registrierungs-Dll befindet sich unterhalb des Projektordners der Anwendung und wird vom VFX – Application Wizard in jedes neue Projekt kopiert. Der Name des Registrierungsprojekts ist: *„Register“ + <Name der Anwendung>*

Bei der Erstellung der Registrierungs-Dll werden über einen Projekt Hook die Aktivierungsregeln aus der Klassenbibliothek *Appl.vcx* der Anwendung gelesen und in der Klasse *cRegDll* in der Eigenschaft *cActPatternName* gespeichert. Die Registrierungs-Dll enthält also die Aktivierungsregeln. Dadurch ist die VFX – Kundenverwaltung unabhängig von einer bestimmten Anwendung. Die VFX – Kundenverwaltung kann Registrierungs-Dlls für verschiedene Anwendungen benutzen. Die Registrierungs-Dll kann aus dem VFX-Menü über den Menüpunkt *Activation, Build register DLL* erstellt werden.

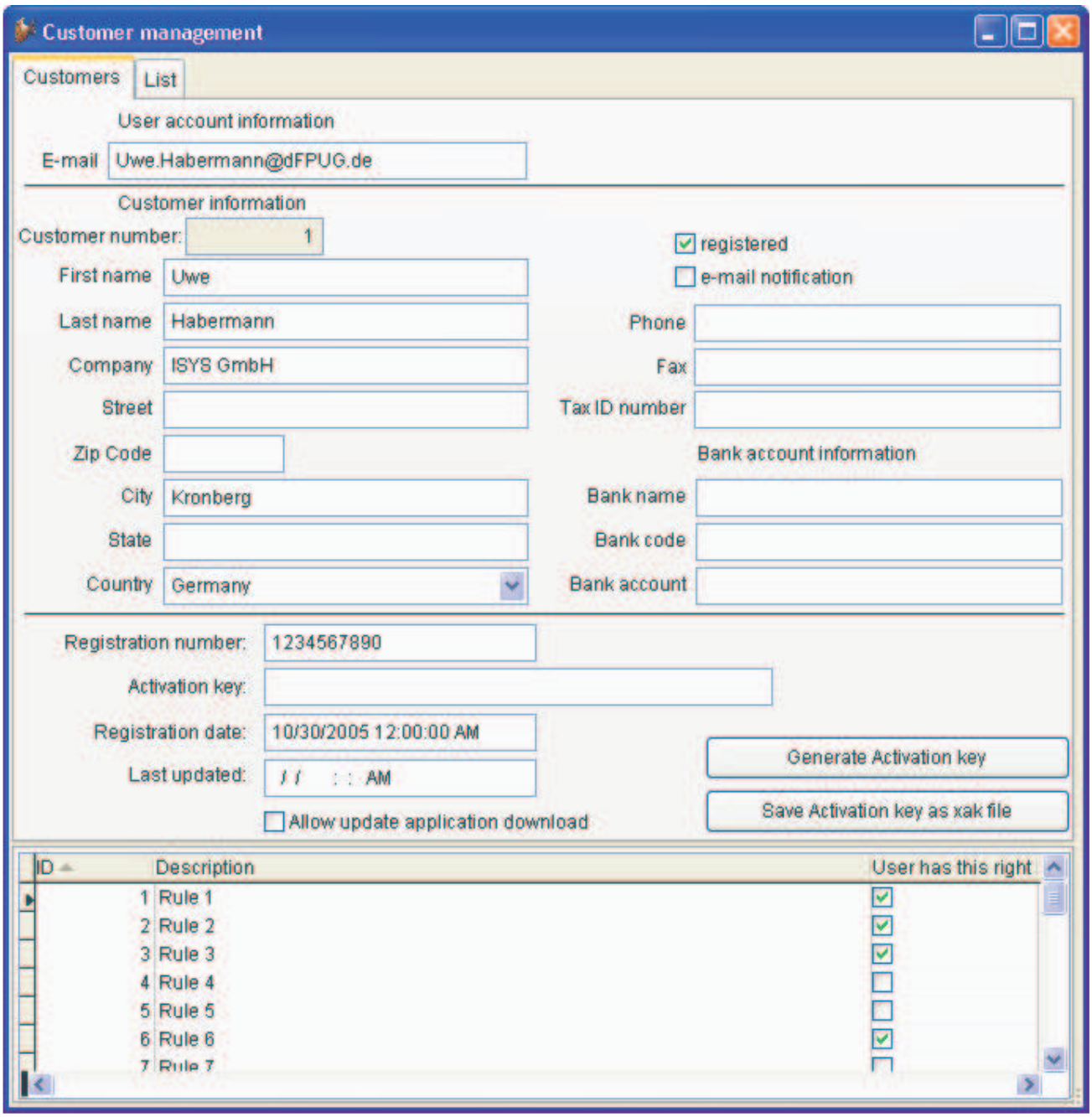
In der Registrierungs-Dll befindet sich die Methode *generateactkey*. Die Parameter dieser Methode sind die Registrierungsnummer, eine Zeichenkette mit den zu erteilenden Rechten sowie der Pfad zur Datei *VFXGenActKey.app*. Der Rückgabewert ist der generierte Aktivierungsschlüssel.

Die Registrierungs-Dll ruft eine Funktion der Anwendung *VFXGenActKey.app* auf. *VFXGenActKey.app* wird mit VFX geliefert und befindet sich im Projektordner der VFX – Kundenverwaltung. *VFXGenActKey.app* enthält den Algorithmus, der anhand der Aktivierungsregeln einen Aktivierungsschlüssel erstellt. Der Quell-Code von *VFXGenActKey.app* wird nicht mit VFX geliefert. Entwickler, die den Quell-Code zur Verfügung hätten, könnten Aktivierungsschlüssel für VFX-Anwendungen anderer Entwickler erstellen.

Die VFX – Kundenverwaltung greift auf die Kundendatenbank über die Datei *Config.vfx* zu. Die Kundendatenbank kann sich in einer VFP-Datenbank oder in einer SQL Server-Datenbank befinden. Der Datenzugriff erfolgt genauso, wie bei anderen VFX-Anwendungen auch. Für die VFX – Kundenverwaltung wurde der Datei *Config.vfx* eine Spalte hinzugefügt. Die Spalte *RegDllName* enthält den Namen der zu verwendenden Registrierungs-Dll.

VFX – Kundenverwaltung

Diese VFX-Anwendung enthält zwei Formulare: Kundenverwaltung und Versionsverwaltung.



Im Formular Kundenverwaltung können die Kundendaten bearbeitet werden. Zu jedem Kunden werden die Registrierungsnummer und die vergebenen Rechte gespeichert. Die Rechte können bei Bedarf verändert werden und es kann aus diesem Formular ein neuer Aktivierungsschlüssel generiert werden.

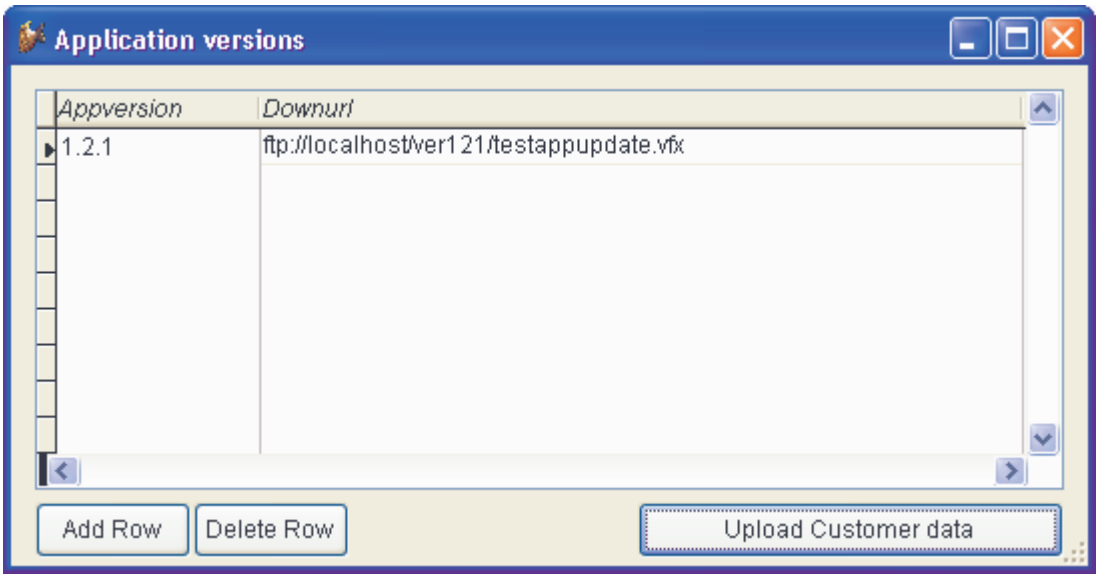
Über die Schaltfläche *Generate Activation Key* kann ein Aktivierungsschlüssel entsprechend der eingegebenen Registrierungsnummer und Benutzerrechte erstellt werden. Der generierte Aktivierungsschlüssel wird in der Kundenverwaltung gespeichert.

Über die Schaltfläche *Save Activation key as xak file* kann der angezeigte Aktivierungsschlüssel in einer Datei mit dem Namen *<Projektname>.xak* im aktuellen Ordner gespeichert werden. Diese

Datei kann an den Kunde geschickt und im Ordner der Anwendung gespeichert werden. Beim Start der Anwendung wird der Aktivierungsschlüssel automatisch aus der Datei gelesen und die Anwendung wird damit aktiviert.

Aus der Kundenverwaltung können E-Mails mit der XAK-Datei als Anhang versendet werden. Diese E-Mails werden versendet, wenn der Registrierungstyp (*cvfxActivation.nRegWay*) der Anwendung ungleich 10 ist. Beim Registrierungstyp 10 wird erwartet, dass sich der Anwender den Aktivierungsschlüssel über einen Web Service holt. In diesem Fall bekommt der Anwender eine E-Mail mit einer Anleitung zum Erhalt des Aktivierungsschlüssels. Der Betreff und Text der generierten E-Mails können im Formular *Manage E-Mail Texts* bearbeitet werden. Der E-Mailtext wird mit der *Textmerge* Funktion von VFP verarbeitet und kann so beliebige Felder aus der Kundenverwaltung enthalten. Auf diesem Weg können persönliche E-Mails gestaltet werden.

Mit dem Formular Versionsverwaltung werden die Versionen und Download-Links der Anwendung verwaltet. Neue Versionen der Anwendung können auf einem Internet Server bereitgestellt werden. Wenn ein Kunde seine Anwendung aktualisieren will, lädt die Anwendung zunächst die Datei *UpdateCustomers.vfx* herunter. In dieser Datei befinden sich die Registrierungsnummern, der zur Aktualisierung berechtigten Kunden. Wenn die Aktualisierungsberechtigung besteht, wird die Datei *Updateversions.vfx* heruntergeladen. In dieser Datei befinden sich die Download-Links zu den verfügbaren Anwendungsversionen. Die Download-Links zu den Dateien *UpdateCustomers.vfx* und *UpdateVersions.vfx* sind in der Anwendung gespeichert und können mit dem VFX – Application Builder eingestellt werden.



Durch einen Mausklick auf die Schaltfläche *Upload Customer Data* werden die Dateien *UpdateCustomers.vfx* und *UpdateVersions.vfx* erstellt.

Damit diese Dateien auf den Internet Server hochgeladen werden können, müssen die Anmeldeinformationen im Projekt der Registrierungs-Dll in der Klassenbibliothek *Regdll.vcx* in der Klasse *cRegDll* gespeichert werden.

Eigenschaften

cFtpUrl – Domainname eines FTP-Servers.

cFtpDir – Verzeichnis auf dem FTP-Server, in dem die Dateien *UpdateCustomers.vfx* und *UpdateVersions.vfx* gespeichert werden sollen.

cPort – Zu verwendender Übertragungsport. Standardmäßig wird für FTP-Uploads der Port 21 verwendet.

cUserName – Benutzername für den FTP-Zugang.

cPassword – Kennwort für den FTP-Zugang.

Web Service für die Registrierung

Mit der VFX – Kundenverwaltung wird das Projekt *RegistrationWebService.pjx* im Ordner *RegistrationWebService* unterhalb der VFX – Kundenverwaltung geliefert. Aus diesem Projekt kann ein COM Server erstellt werden, der auf einem Internet Information Server als Web Service installiert werden kann. An diesen Web Service können Anwendungen die Registrierungsdaten von Benutzern senden. Der Web Service kann an die Anwendung einen Aktivierungsschlüssel senden.

Der Web Service benutzt eine *Config.vfx* für den Datenzugriff. Hier muss auf die gleiche Datenbank gezeigt werden, die auch von der VFX – Kundenverwaltung verwendete wird. Im einfachsten Fall kann der Web Service im gleichen Ordner wie die VFX – Kundenverwaltung installiert werden und so die gleiche *Config.vfx* benutzen, wie die Kundenverwaltung.

Wenn der Web Service auf einem entfernten Internet Server laufen soll, so ist die Kundendatenbank mit dem VFX – Upsizing Wizard auf einen SQL Server zu portieren. Auf die so erzeugte SQL Server Datenbank können sowohl der Web Service, als auch die VFX – Kundenverwaltung über das Internet zugreifen.

Damit eine Anwendung über den Web Service aktiviert werden kann, müssen in der Anwendung ein paar Einstellungen im Formular *VfxRegister.scx* gemacht werden:

cWSDL – Enthält die URL der WSDL Datei. Diese Datei wird bei der Registrierung des Web Service mit dem SOAP Toolkit auf dem Internet Server generiert.

cServiceName – Enthält den Namen des Web Service.

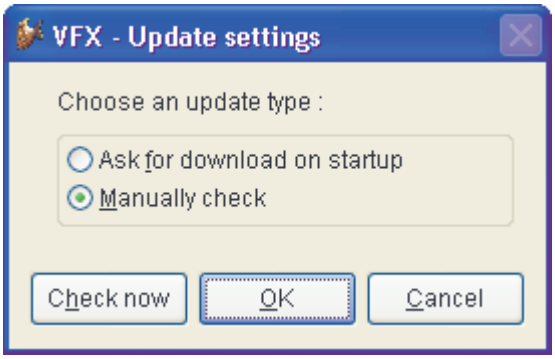
cServiceMethodName – Enthält den Namen der verwendeten Web Service Methode. Standardmäßig ist dies die Methode *GenerateActKey*.

Wenn sich ein Kunde über den Web Service registriert, werden die Registrierungsdaten an den Web Service im XML Format übertragen. Der Web Service sucht in der Kundendatenbank nach einem Datensatz mit der gleichen E-Mailadresse sowie dem gleichen Anwendungsnamen und der gleichen Version. Wenn dieser Datensatz gefunden wird, wird der dort eingetragene Aktivierungsschlüssel an den Kunden übertragen. Die Anwendung wird dabei automatisch aktiviert. Wenn kein solcher Datensatz gefunden wird, werden die Registrierungsinformationen in der Kundendatenbank gespeichert.

Der Web Service für die Registrierung enthält die Methode *ReceiveErrorInfo*. Diese Methode kann Fehlermeldungen von Endanwendungen empfangen. Damit haben Anwender die Möglichkeit die Dateien, die lokal in der Fehlerprotokolltabelle *Vfxlog* gespeichert werden, an den Web Service zu senden.

Aktualisierung von VFX

Es ist sinnvoll VFX regelmäßig zu aktualisieren, damit immer der aktuelle Stand zur Verfügung steht. VFX 9.5 kann automatisch auf Aktualisierungen prüfen. Dies kann im Dialog *Update Settings* eingestellt werden. Wenn die Option *Ask for download on startup* gewählt ist, überprüft VFX bei jedem ersten Start an jedem Tag, ob ein aktualisierter Build zur Verfügung steht. Falls ja, wird gefragt, ob der neue Build heruntergeladen und installiert werden soll. Die Überprüfung wird nicht durchgeführt, wenn keine Verbindung mit dem Internet besteht. Über die Schaltfläche *Check for updates now* kann die Überprüfung nach aktualisierten Builds jederzeit manuell gestartet werden.



Die Klasse cUpdate

Diese Klasse stellt den Dialog zur Bearbeitung der Aktualisierungseinstellungen bereit. Über eine Schaltfläche kann sofort geprüft werden ob Aktualisierungen zur Verfügung stehen. Die Klasse wird sowohl von VFX intern als auch in Endanwendungen verwendet.

tlUpdateApp - Initialisierungsparameter.

.T. – Aktualisierung für Endanwendungen.

.F. – Aktualisierung für VFX.

Methode

WriteData()

Speichert die gewählte Methode zur Aktualisierung in der Tabelle *Vfxsys*.

Die Klasse cUpdateEngine

Mit dieser Klasse wird die Aktualisierung der Anwendung beim Kunden durchgeführt. Abhängig von den Einstellungen im Dialog für die Aktualisierungseinstellungen wird eine Datei heruntergeladen, die die aktuell verfügbare Versionsnummer enthält. Um eine neue Programmversion herunterzuladen wird ein Objekt der Klasse *cDownload* instanziiert.

Eigenschaften

cIniURL – URL der Datei, die die aktuelle Versionsnummer enthält.

cVersion – Versionsnummer der aktuell laufenden Anwendung.

dLastChecked – Datum der letzten Überprüfung auf Aktualisierungen. Dieses Datum wird aus der Tabelle *Vfxsys* gelesen.

nManualCheck – Wie wurde die Überprüfung auf Aktualisierungen gestartet?

0 – Automatische Überprüfung auf Aktualisierungen bei Anwendungsstart.
1 – Manuelle Überprüfung auf Aktualisierungen aus dem Menü.

nUpdateType – Aktualisierungstyp. Dieser Wert wird aus der Tabelle *Vfxsys* gelesen.

1 – Automatisches herunterladen und installieren, wenn Aktualisierungen vorhanden sind. Die Überprüfung wird beim ersten Anwendungsstart täglich durchgeführt.
2 – Automatisches herunterladen von Aktualisierungen mit anschließender Frage, ob die Aktualisierung installiert werden soll. Die Überprüfung wird beim ersten Anwendungsstart täglich durchgeführt.
3 – Überprüfung nach verfügbaren Aktualisierungen täglich beim ersten Anwendungsstart. Bei vorhandener aktualisierter Programmversion wird der Benutzer gefragt, ob er die Aktualisierung herunterladen und installieren will.
4 – Manuelle Überprüfung. Die Überprüfung auf Aktualisierungen kann aus dem Menü aufgerufen werden.

Methoden

CheckUpdType()

Ermittelt die Versionsnummer der laufenden Anwendung sowie den Aktualisierungstyp und das Datum der letzten Überprüfung auf Aktualisierung. Die ermittelten Werte werden in den Eigenschaften *cVersion*, *nUpdateType* und *dLastChecked* gespeichert. Wenn alle Werte erfolgreich ermittelt werden können, wird .T. zurückgegeben. Bei Auftreten eines Fehlers, wird .F. zurückgegeben.

DoUpdate(tnUpdateType)

tnUpdateType = nUpdateType
Aufruf der Methode *UpdateApp*(*tnUpdateType)* (in *Vfxfunc.prg*).

NewVersion()

Herunterladen der Datei aus *cIniURL*. Rückgabewert:
0 – Es ist keine neue Version verfügbar.
1 – Eine neue Version ist verfügbar.
2 – Es ist ein Fehler aufgetreten.

SetNewCheckDate(tdDate)

tdDate – Datum, an dem die nächste Überprüfung auf Aktualisierungen stattfindet. Das als Parameter übergebene Datum wird in der Tabelle *Vfxsys* im Feld *lastcheckd* gespeichert.

StartUpdate(nManualCheck)

nManualCheck – Einstellen des Aktualisierungstyps.
0/.F. – Automatische Aktualisierung.
1/.T. – Manuelle Aktualisierung.
Dies ist die wichtigste Methode dieser Klasse. Hiermit wird die Aktualisierung gestartet. Bevor diese Methode aufgerufen wird, muss der Eigenschaft *cIniURL* die URL mit der Versionsdatei zugewiesen werden. Aus dieser Methode werden die Methoden *CheckUpdType()* und *NewVersion()* aufgerufen. Die Methode *SetNewCheckDate(tdDate)* wird aufgerufen, um das Datum der letzten Überprüfung zu speichern. Wenn eine neue Version verfügbar ist, wird die Methode *DoUpdate(tnUpdateType)* aufgerufen. Bei manueller Überprüfung auf Aktualisierungen wird die Methode *DoUpdate(tnUpdateType)* ohne vorherige Prüfung aufgerufen.

In der Klassenbibliothek *Appl.vcx* befindet sich die Klasse *cAppUpdateEngine*. Dies ist eine 1:1-Ableitung der Klasse *cUpdateEngine*. In der Klasse *cAppUpdateEngine* sollten Entwickler eigene Einstellungen machen. Die Eigenschaft *cIniURL* kann auch im VFX – Application Builder eingestellt werden.

AFX Unterstützung

In der Klasse cFoxAppl befinden sich zwei neue Methoden, die die Generierung von Web-Anwendungen basierend auf VFX-Projekten vereinfachen. Es sind die Methoden *VFXMessageBox* und *VFXWaitWindow*. Diese Methoden akzeptieren die gleichen Parameter wie die entsprechende VFP-Funktion *Messagebox* bzw. der Befehl *Wait Window*.

Methoden

VFXMessageBox()
LPARAMETERS *emessagetext, ndialogboxtype, ctitlebartext, ntimeout*

emessagetext – Anzuzeigender Text in der Messagebox. Wenn dieser Wert fehlt oder von falschem Typ ist, wird eine leere Zeichenkette angezeigt.
ndialogboxtype – Typ der Messagebox. Hierüber können Schaltflächen und das Icon eingestellt werden. Der Standardwert ist 0.
ctitlebartext – Titel der Messagebox. Wenn dieser Wert fehlt oder von falschem Typ ist, wird eine leere Zeichenkette angezeigt.
ntimeout – Zeitspanne zur Anzeige der Messagebox. Wenn dieser Wert fehlt oder von falschem Typ ist, wird gewartet, bis der Benutzer eine Schaltfläche betätigt.

Diese Methode führt die VFP-Funktion MESSAGEBOX() aus. Wenn die Anwendung als Web-Anwendung läuft, werden Web-Seiten angezeigt. Wenn die Anwendung ohne sichtbare Ausgaben läuft, wird der Wert der Standardschaltfläche der Messagebox zurückgegeben.

VFXWaitWindow()
LPARAMETERS *tcMessageText, tnRow, tnColumn, tlNowait, tlClear, tlNoclear, tnTimeout*

tcMessageText – Anzuzeigender Text. Der Standardwert ist eine leere Zeichenkette.

tnRow – Zeilennummer des Wait Window. Nur in Zusammenhang mit *tnColumn* verwendbar.

tnColumn – Spaltennummer des Wait Window. Nur in Zusammenhang mit *tnRow* verwendbar.

tlNowait - .T. um die Programmausführung nach Anzeige des Wait Window fortzusetzen, .F. um die Programmausführung anzuhalten bis der Benutzer eine Taste oder Maustaste drückt. Der Standardwert ist .F.

tlClear - .T. um aktuell angezeigte Wait Windows zu löschen. Der Standardwert ist .F.

tlNoclear - .T. und das Wait Window angezeigt zu lassen, bis der Befehl WAIT CLEAR ausgeführt wird. Der Standardwert ist .F.

tnTimeout – Anzeigedauer des Wait Window. Wenn 0 angegeben wird, wird das Wait Window nicht gelöscht, bis der Befehl WAIT CLEAR ausgeführt wird. Der Standardwert ist 0.

Diese Methode führt den VFP-Befehl WAIT WINDOW aus. Wenn die Anwendung ohne sichtbare Ausgaben läuft, erfolgt keine Anzeige.

Hilfe

In der neuen Hilfe-Sektion des VFX-Menüs kann das Benutzerhandbuch geöffnet werden und es werden nützliche Informationen rund um VFX sowie Links zu Online Ressourcen angeboten.

Benutzerhandbuch und Dokumentation der Neuheiten

Über den Menüpunkt *VFX Help, User Manuals and What's New* können das Benutzerhandbuch und die Neuheitendokumentation in deutscher und englischer Sprache geöffnet werden. Alle diese Dokumente liegen als PDF-Dateien vor. Zum Öffnen ist der Adobe Reader erforderlich.

Über den Menüpunkt *Update Notes* kann ein Dokument geöffnet werden, das alle Änderungen in der VFX 9.5-Template-Anwendung seit dem Erscheinen beschreibt.

Wenn eins der anzuzeigenden Dokumente nicht installiert ist, wird das entsprechende Dokument automatisch heruntergeladen.

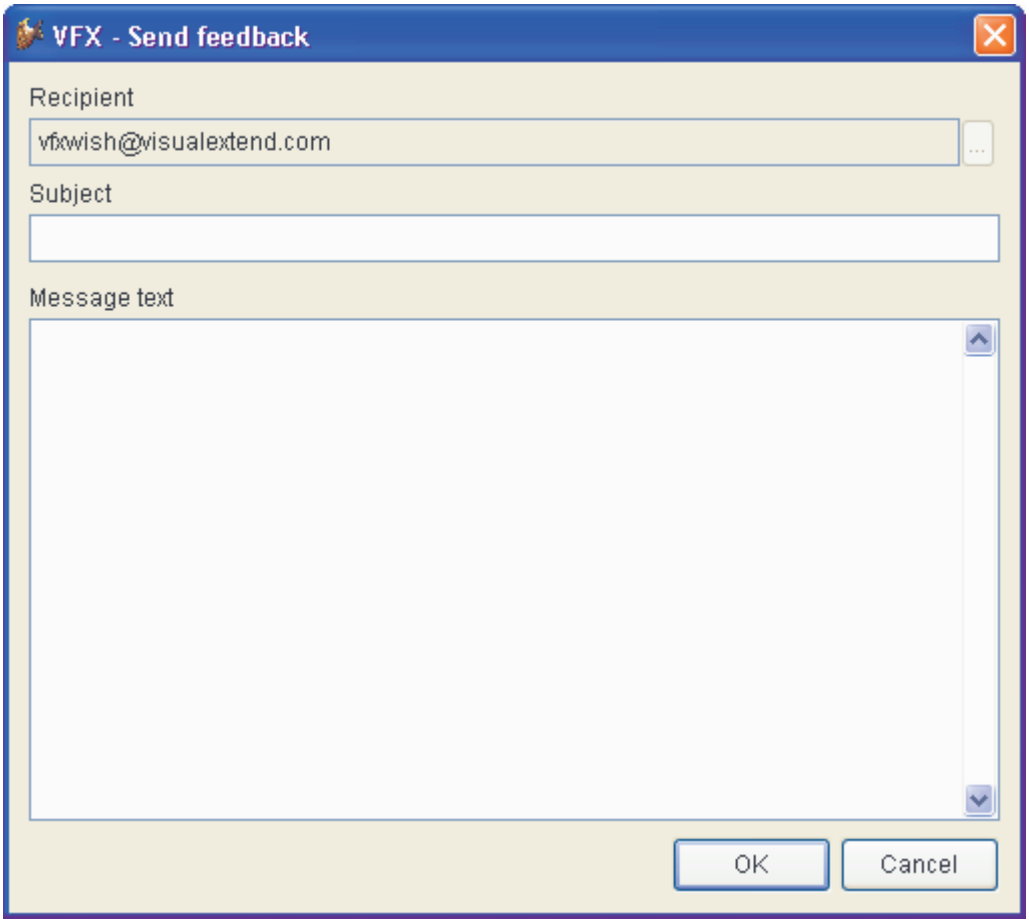
Visual Extend Online

Über den Menüpunkt *VFX Help, Visual Extend Online* erreichen Sie direkt die folgenden Seiten der VFX Website:

- VFX Startseite (zurzeit in vier Sprachen verfügbar - deutsch, englisch, französisch und spanisch)
- VFX Directory (Startseite des Visual Extend Portals)
- Forum (lesen und schreiben Sie im Online-Forum)
- Newsletters (abonnieren Sie aktuelle Informationen über VFX)
- DevCons (besuchen Sie die Entwicklerkonferenz!)
- Online shop (kaufen Sie VFX, Bücher, VFP und anderes)

Senden Sie uns eine E-Mail!

Machen Sie Vorschläge, sagen Sie uns Ihre Meinung oder äußern Sie Erweiterungswünsche. Über den Menüpunkt *VFX Help, Send user feedback* können Sie dem VFX-Team eine E-Mail senden.

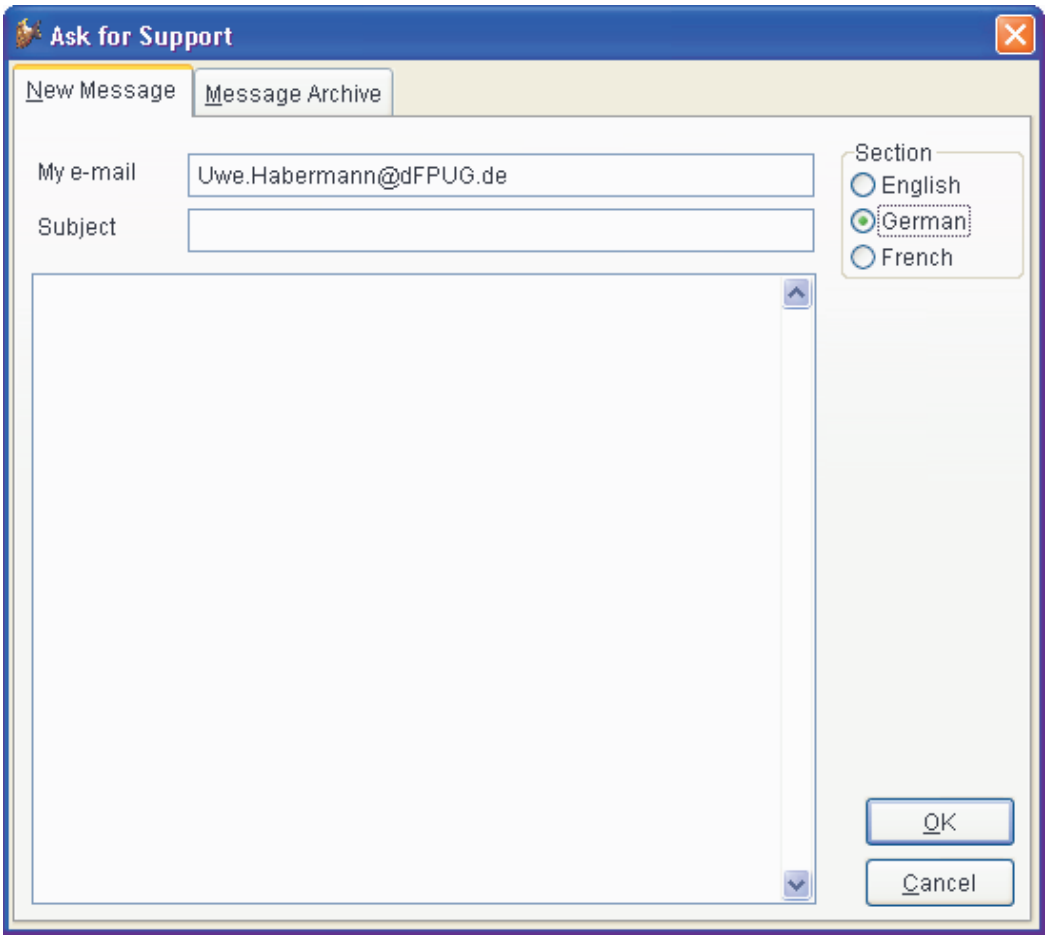


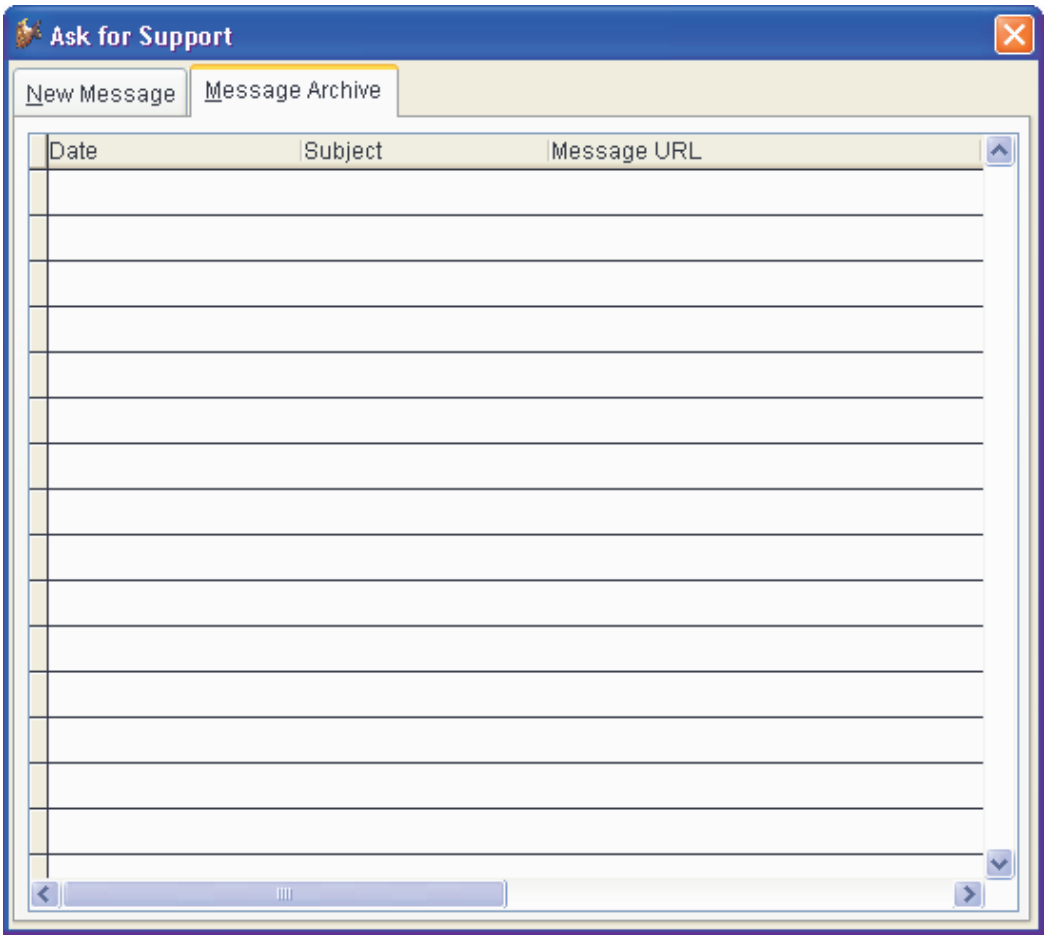
So erreichen Sie uns

Über den Menüpunkt *VFX Help, How to reach us* wird Ihnen angezeigt, wie Sie uns erreichen können.

Support-Anfragen an das Forum richten

Es gibt zwei Möglichkeiten um direkt aus VFX eine Support-Anfrage im Forum zu publizieren. Zum einen kann aus dem VFX-Menü über den Menüpunkt *VFX Help, Visual Extend Online, Forum* der Internet Browser gestartet werden. Es öffnet sich die Startseite des Forums. Hier können Nachrichten online gelesen und auch neue Nachrichten erstellt werden. Zum anderen kann über den Menüpunkt *VFX Help, Ask for Support* ein Dialog geöffnet werden, in dem offline eine Forumsnachricht verfasst werden kann. Zum Versenden einer Nachricht ist eine Internetverbindung erforderlich. Die auf diesem Weg verfassten Nachrichten bleiben gespeichert und können später auf der Seite *Message Archive* angesehen werden.


Ask for Support
New Message
Message Archive
My e-mail
Uwe.Habermann@dFPUG.de
Subject
Section
English
German
French
OK
Cancel


Ask for Support
New Message
Message Archive
Date
Subject
Message URL

Info

Informationen über die installierte Visual Extend Version sowie Registrierungsinformationen erhalten Sie über den Menüpunkt *VFX Help, About Visual Extend*.

Neue Eigenschaften des Anwendungsobjekts

lHideRegistrationFiles – Wenn der Wert dieser Eigenschaft mit *.T.* eingestellt wird, werden die Dateien *VFX.ini* und *FirstInstall.txt* mit dem Attribut *hidden* gekennzeichnet. *.T.* ist der Standardwert.

lInstallClickyes – Mit dieser Eigenschaft kann eingestellt werden, ob die Anwendung *ClickYes* automatisch installiert werden soll, wenn E-Mails mit Outlook versendet werden sollen. Die Anwendung *ClickYes* unterdrückt die Sicherheitsfragen von Outlook. Der Standardwert dieser Eigenschaft ist *.T.*

lTableprompt – Wenn der Wert dieser Eigenschaft auf .T. gestellt wird, gilt die VFP-Einstellung *SET TABLEPROMPT ON*. Wenn ein Befehl ausgeführt wird, der eine geöffnete Tabelle oder Ansicht voraussetzt, erscheint bei Bedarf automatisch ein Datei-Öffnen-Dialog. Wenn der Wert dieser Eigenschaft auf .T. und es ist keine Tabelle oder Ansicht im aktuellen Arbeitsbereich geöffnet, kommt es zu einem Laufzeitfehler.

lVarcharmapping – Mit dieser Eigenschaft kann eingestellt werden, ob Zeichentypen mit variabler Feldlänge in Zeichentypen mit fester Feldlänge konvertiert werden sollen. Wenn der Wert *.T.* eingestellt wird, werden Datentypen mit variabler Zeichenfeldlänge von remote Datenquellen im VFP Feldtyp *Varchar* abgebildet. Der Standardwert ist *.F.* Dies entspricht der VFP Einstellung *SET VARCHARMAPPING OFF*.

nAllowSaveEmptyRecords – Verhindert das Speichern von leeren Datensätzen in allen Formularen, wenn der Wert *2* eingestellt wird. Wenn der Wert *1* eingestellt wird, können Benutzer leere Datensätze speichern. Beim Wert *0* kann individuell je Formular mit der Eigenschaft *lAllowSaveEmptyRecords* eingestellt werden, ob leere Datensätze gespeichert werden dürfen. Der Standardwert ist *0*.

nComboBoxListEntriesLimit – Der Wert dieser Eigenschaft gibt die Anzahl der Einträge einer Liste in einer aufgeklappten Combobox an. Der Standardwert ist *15*.

nReprocess – Dieser Wert gibt die Anzahl der Wiederholungsversuche für erfolglose Satz- oder Dateisperren an. Der Standardwert ist *0*.

nUseReportbehavior80forPDFOutput – Wenn Berichtsausgaben in eine PDF-Datei mit der Einstellung *SET REPORTBEHAVIOR 90* erfolgen, ist es nicht möglich aus der PDF-Datei Texte zu kopieren. Mit dieser Eigenschaft wird es ermöglicht PDF-Ausgaben mit der Einstellung *SET REPORTBEHAVIOR 80* zu erstellen, sodass Anwender Text kopieren können. Wenn der Wert dieser Eigenschaft *0* ist, kann je Formular mit der Eigenschaft *lUseReportbehavior80forPDFOutput* das Verhalten eingestellt werden. Wenn der Wert *1* ist, wird in allen Formularen *SET REPORTBEHAVIOR 80* bei der Erstellung von PDF-Ausgaben verwendet. Wenn der Wert *2* ist, wird die aktuelle Einstellung von *SET REPORTBEHAVIOR* verwendet.

Weitere Verbesserungen für Entwickler

Das Formular für die Aktivierung von VFX ist jetzt lokalisiert.

Im VFX – Messagebox Builder und im VFX – Message Editor können jetzt die Texte aller Sprachen bearbeitet werden, unabhängig von den Unicode-Einstellungen des Betriebssystems.

In den VFX – Form Buildern steht jetzt eine größere Auswahl von VFX-Klassen zur Verfügung.

Die VFX – Form Builder verwenden als Standardsteuerelement für Felder vom Typ *Date* die Klasse *cPickDate*. Dadurch haben Anwender die Möglichkeit ein Datum aus einem Kalender auszuwählen.

Im VFX – Class Switcher kann jetzt aus einem Auswahldialog eine beliebige Klassenbibliothek und Klasse aus dem aktuellen Projekt gewählt. Die Anzeige der Klassen und Klassenbibliothek ist alphabetisch. Wenn versucht wird eine nicht geeignete Klasse zuzuweisen, erscheint ein Warnhinweis und der Vorgang wird nicht fortgesetzt.

Im VFX-Menü gibt es unter *Project* jetzt eine Möglichkeit vom aktuellen Projekt eine Archivdatei anzulegen. Dateiname ist der Projektname gefolgt von einem Zeitstempel.

Das Debug-Menü für die Entwicklungsumgebung kann jetzt im VFX – Application Builder eingeschaltet werden. Wahlweise kann manuell die Eigenschaft *lDebugMode* der Klasse *cFoxAppl* in der Klassenbibliothek *Appl.vcx* auf .T. eingestellt werden.

In *Vfxfunc.prg* wurde die neue Funktion *GetNewGUID* hinzugefügt, die einen global und für immer eindeutigen ID-Wert zurückgibt. Die GUID wird mithilfe der API-Funktion *CoCreateGuid* ermittelt. Die Länge eines GUID ist 36 Zeichen. GUIDs können insbesondere dann als ID in Tabellen verwendet werden, wenn ein Datenabgleich mit anderen Tabellen vorgesehen ist.

Bei der Aktualisierung von Child-Daten in OneToMany-Formularen werden jetzt auch die Felder mit dem Benutzernamen (*ins_usr, edt_usr*) und dem Timestamp (*ins_date, edt_date*) von VFX automatisch mit Daten gefüllt.

In der Klasse *cFormbase* gibt es im Ereignis *Destroy* einen neuen Hook. Damit wird eine Eingriffsmöglichkeit für eigenen Code beim Schließen eines Formulars gegeben.

Neue Eigenschaften für Endbenutzer

Erforderliche Rechte zur Ausführung

VFX 9.5 Anwendungen können für Windows XP zertifiziert werden. Zur Ausführung einer VFX 9.5-Anwendung sind Windows-Standard-Benutzerrechte ausreichend. Entsprechend den Windows-Design-Richtlinien, können die ausführbaren Programmdateien (Exe-Datei, VFX.fll) unter *C:\Programme* von einem Installationsprogramm installiert werden. Alle anderen verwendeten Dateien können in anderen Ordnern installiert werden. Standardmäßig werden alle von einer Anwendung selbst erstellten Dateien im Ordner *C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\<Firmenname>\<Anwendungsname>* gespeichert.

Zum Finden der Dateien Vfxpath.dbf oder Config.vfx wird die folgende Suchstrategie verwendet:

- Installationsordner der Anwendung (Exe-Datei) (aus Kompatibilitätsgründen zu früheren Versionen)
- *C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\<Firmenname>\<Anwendungsname>*
- *C:\Dokumente und Einstellungen\<Aktueller Windows-Anmelde-name>\Anwendungsdaten\<Firmenname>\<Anwendungsname>*

Wenn eine dieser Dateien neu erstellt werden muss, wird zunächst versucht diese Datei im Ordner *C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\<Firmenname>\<Anwendungsname>* zu speichern. Wenn der aktuelle Benutzer in diesem Ordner keine Schreibrechte hat, wird die Datei im Ordner *C:\Dokumente und Einstellungen\<Aktueller Windows-Anmeldename>\Anwendungsdaten\<Firmenname>\<Anwendungsname>* gespeichert.

Wenn mit einer remote Datenbank gearbeitet wird, wird die von VFP verwendete freie Tabelle zur Speicherung von *Autocomplete*-Werten *VFXAComp.dbf* nach der gleichen Strategie wie oben beschrieben gesucht und gespeichert.

Neue Icons

Viele neue Icons wurden für Endbenutzer erstellt und geben den Anwendungen ein deutlich verbessertes Erscheinungsbild. Auch in die Builder von VFX wurden zahlreiche neue Icons integriert und verbessern die intuitive Bedienung für Entwickler.

Datenzugriff

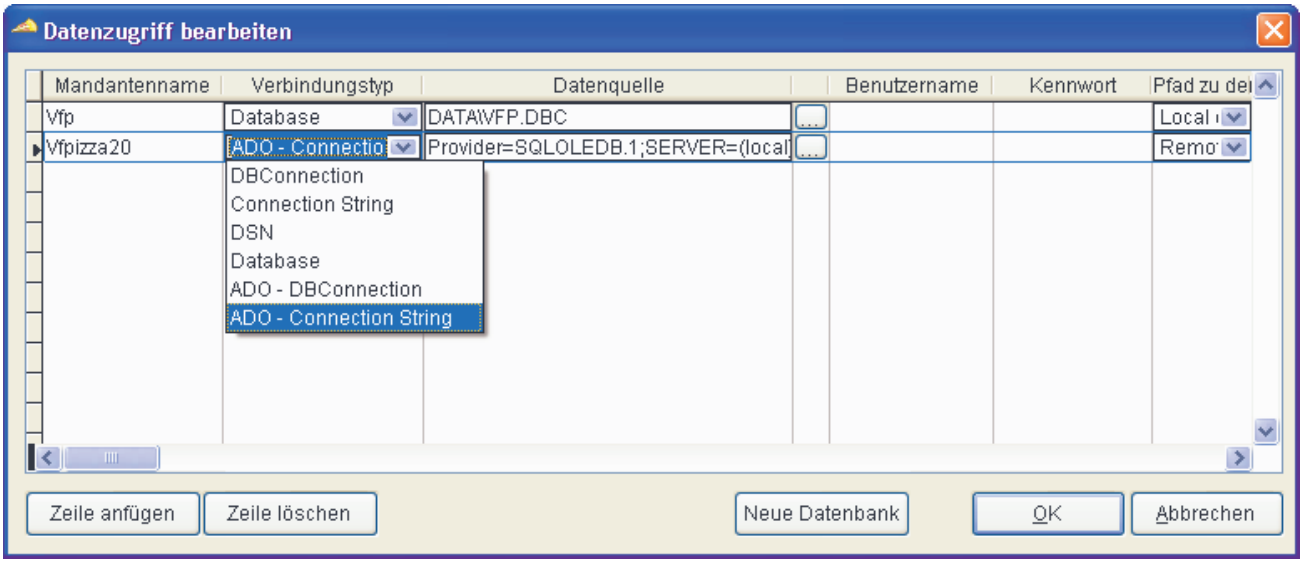
Der Dialog Datenzugriff bearbeiten

Zusätzlich zu den VFX 9.0 bekannten Möglichkeiten des Datenzugriffs kann in VFX 9.5 ein OLE-DB Provider zum Zugriff auf die Daten verwendet werden. Die Funktionalität der Klasse *cConnectionMgr* wurde erweitert um OLE-DB Verbindungen zu ermöglichen.

Im Dialog *Manage Config.vfx* kann zwischen drei OLE-DB Verbindungsmöglichkeiten gewählt werden:

ADO – DBConnection: Das ist eine Verbindung, die in einem Datenbank-Container gespeichert ist.
ADO – Connection String: Eine Verbindungszeichenfolge für einen OLE-DB Provider.

Diese beiden OLE-DB Verbindungstypen entsprechen etwa den ODBC-Verbindungstypen DBConnection und Connection String.



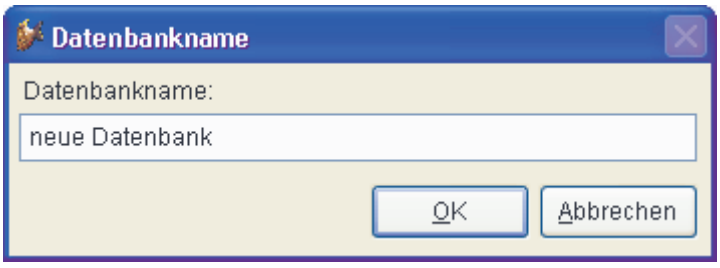
Die VFX-Tabellen können jetzt wahlweise auch in einer eigenen Remote Datenbank gespeichert werden und stehen so auf Wunsch Datenbank-Übergreifend zur Verfügung. Alle Verbindungstypen stehen auch für die Datenbank mit den VFX-Tabellen zur Verfügung.

Anlegen einer neuen Datenbank beim Kunden

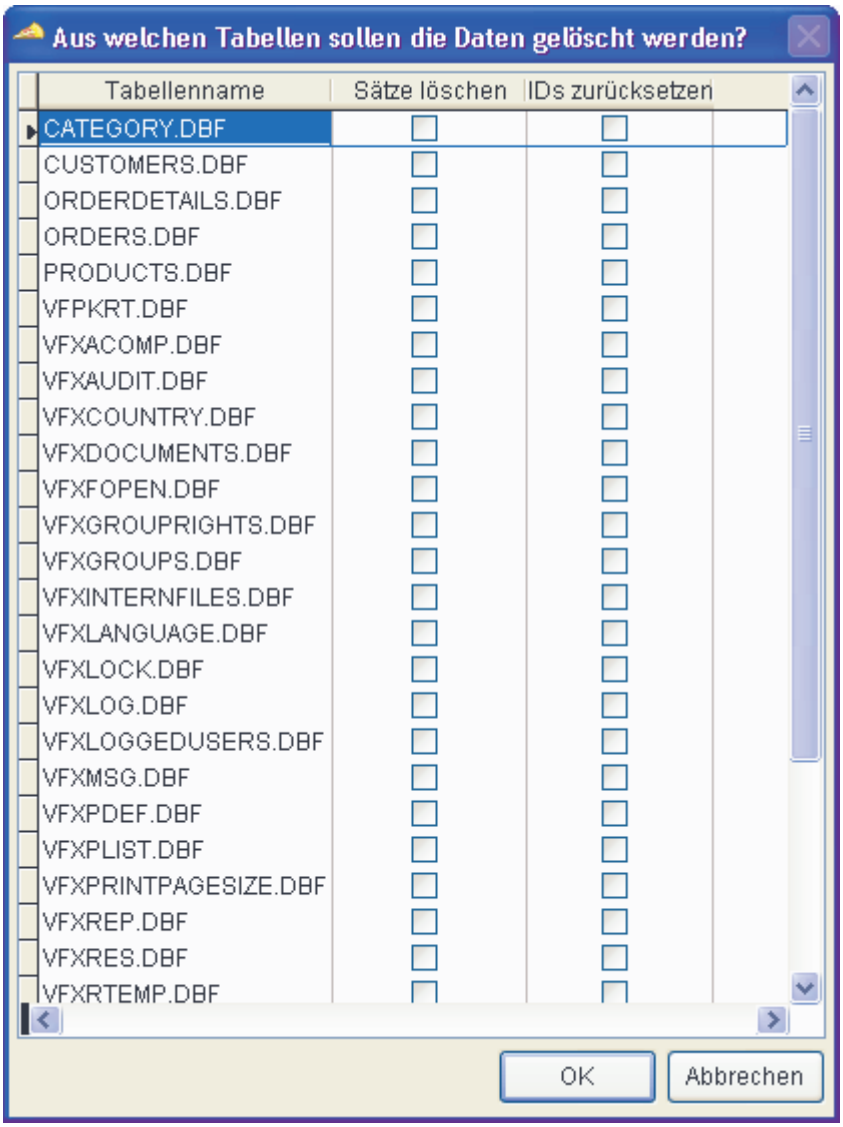
In VFX 9.5-Anwendungen kann auf einfachem Weg eine neue Datenbank angelegt werden. Zur Laufzeit ist im Dialog *Datenzugriff bearbeiten* die Schaltfläche *Neue Datenbank* sichtbar. Die neue Datenbank wird mit der Struktur der aktuell selektierten Datenbank angelegt.

Wenn die aktuelle Datenbank eine VFP-Datenbank ist, erscheint ein Ordnerauswahldialog und der Benutzer kann einen neuen Ordner für die neue Datenbank anlegen.

Wenn die aktuelle Datenbank eine Remote Datenbank ist, kann der Benutzer der neuen Datenbank in einem Dialog einen Namen geben.



Im nächsten Dialog wird gefragt, aus welchen Tabellen die Daten in der neu angelegten Datenbank gelöscht werden sollen. Auf Wunsch können auch die ID-Werte für einzelne Tabellen zurückgesetzt werden.



Neue Eigenschaften in Onetomany-Formularen

Die Daten aus Child-Grids auf OneToMany-Formularen können jetzt per Drag & Drop in andere Anwendungen kopiert werden. Diese Option kann im VFX – COneToMany Builder und im VFX – CChildgrid Builder ein- bzw. ausgeschaltet werden.

In VFX – OneToMany Builder und im VFX – CchildGrid Builder kann für jede Spalte eines Child Grid eingestellt werden, ob diese Spalte beim OLE Drag & Drop mit kopiert wird.

Zu jeder Spalte in einem Child-Grid auf einem OneToMany-Formular kann eine Summe gebildet werden. Diese Option kann im VFX – COneToMany Builder und im VFX – CChildgrid Builder ein- bzw. ausgeschaltet werden. Wenn eine Summe gebildet werden soll, wird am unteren Formularrand ein Label mit der Bezeichnung der Spalte sowie eine Textbox mit der Summe hinzugefügt.

Das Datum sowie ggf. die Zeit und der Benutzername werden jetzt auch bei Child-Datensätze protokolliert, wenn die entsprechenden Felder in der Child-Tabelle vorhanden sind.

OnChildRequery

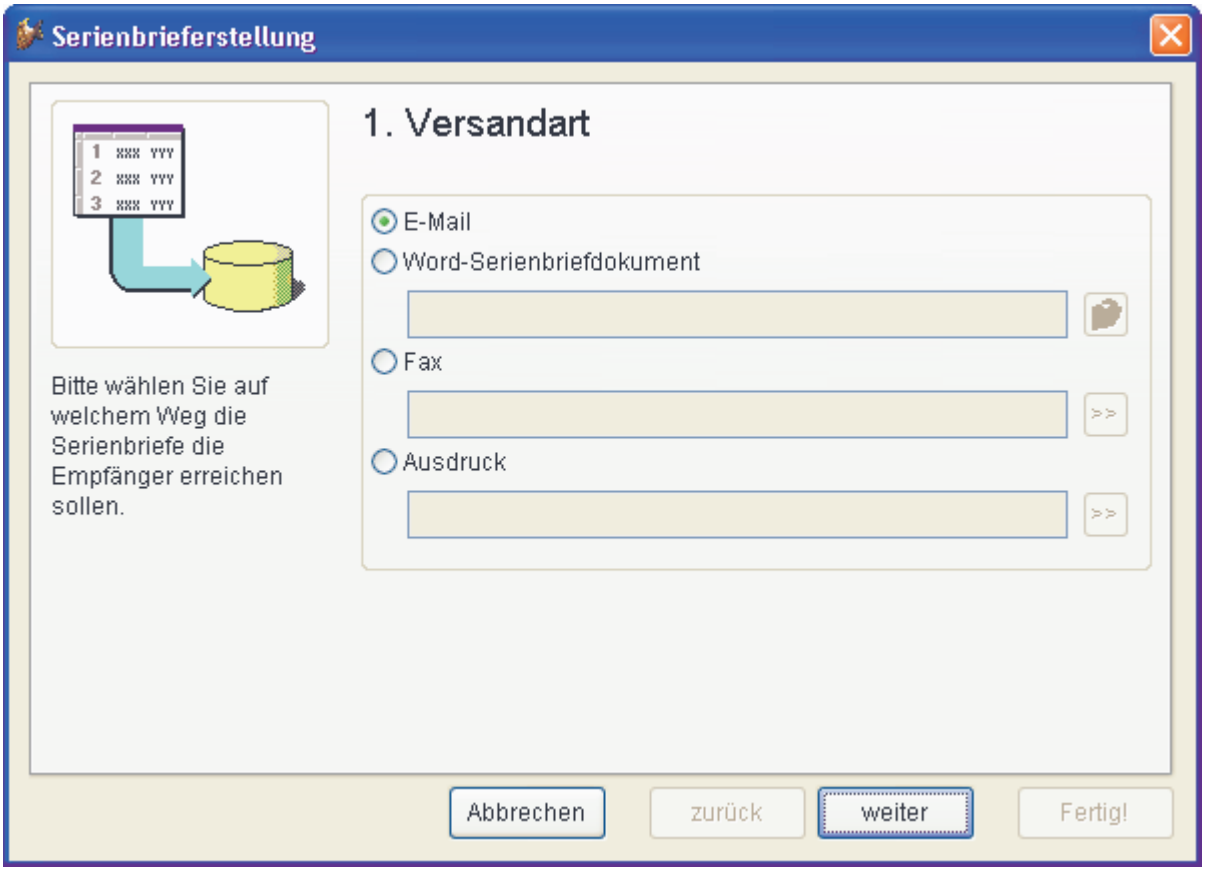
In früheren VFX-Versionen musste Code in die Methode *OnChildRequery* von OneToMany-Formularen eingetragen werden, wenn die Child-Daten auf Ansichten oder CursorAdaptern basierten.

In VFX 9.5 ist dies nicht mehr erforderlich. Beim Bewegen des Satzzeigers im Parent-Teil des Formulars werden automatisch alle Child-Arbeitsbereiche überprüft. Wenn ein Child-Arbeitsbereich auf einer Ansicht oder einem CursorAdapter basiert, werden die Daten aktualisiert. Bei Ansichten wird dazu *REFRESH()* aufgerufen. Bei CursorAdaptern wird die Methode *CursorRefresh()* ausgeführt.

Seriendokumenterstellung

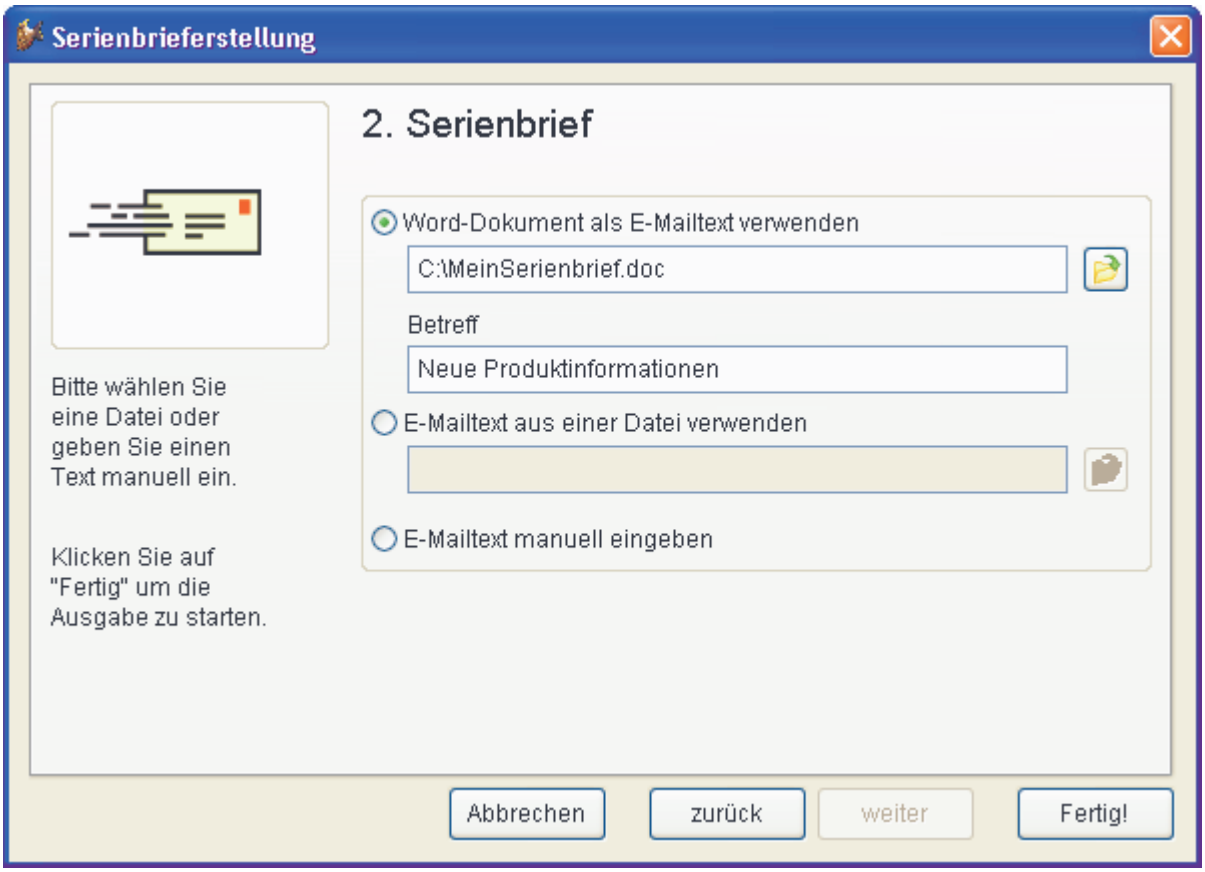
Mit dem Assistenten zur Seriendokumenterstellung kann dem Benutzer die Möglichkeit gegeben werden Seriendokumente basierend auf den Daten der Anwendung zu erstellen. Als Text für die generierten Dokumente kann ein Word-Serienbriefdokument oder eine Textdatei verwendet werden oder es kann manuell im Assistenten ein Text eingegeben werden. Das Ergebnis der Seriendokumentausgabe kann wahlweise als Word-Dokument gespeichert werden, gedruckt werden, als Fax gesendet werden oder als E-Mail gesendet werden. Der Benutzer wird durch den Assistenten in wenigen intuitiven Schritten geführt.

Im ersten Schritt wählt der Benutzer die Versandart.

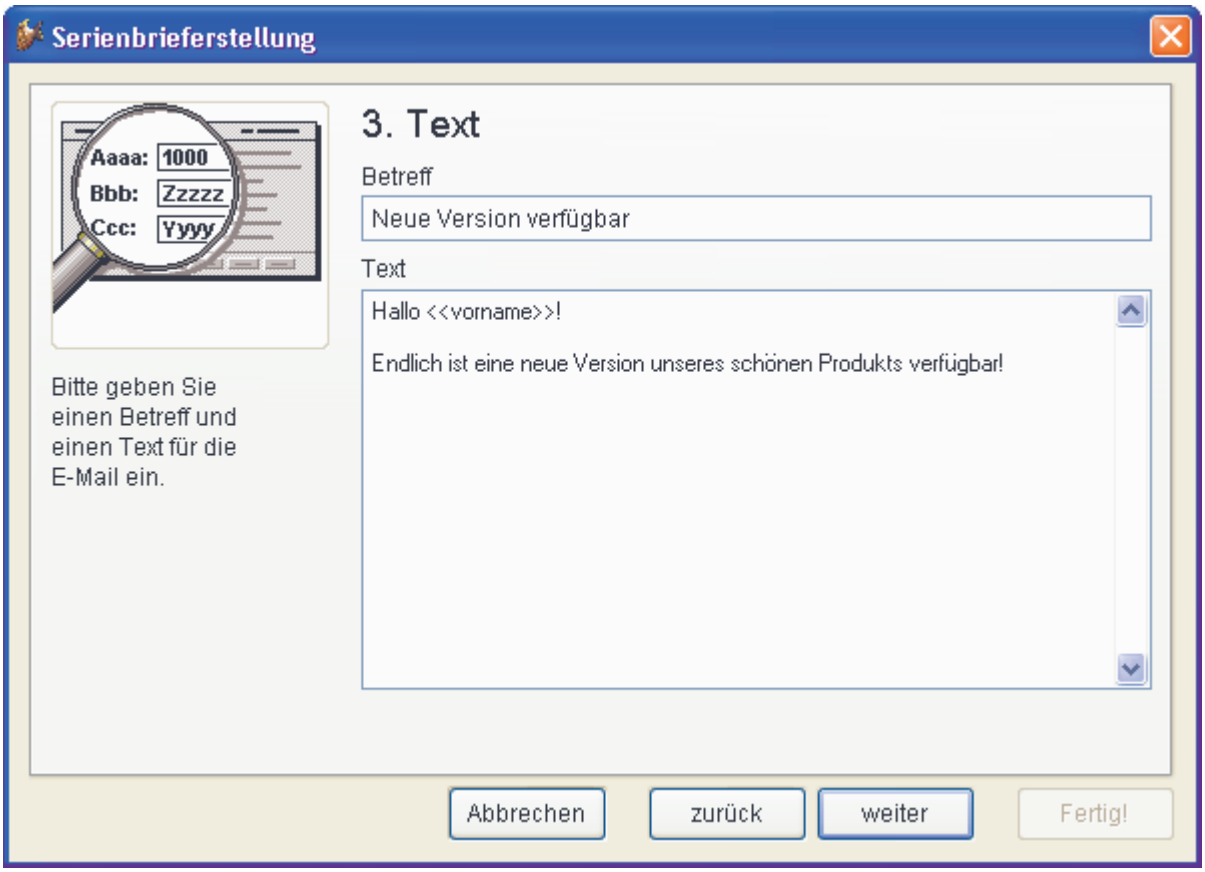


Im zweiten Schritt wird der Text für die zu erstellenden Dokumente ausgewählt. Wenn im ersten Schritt Word-Serienbriefdokument, Fax oder Ausdruck gewählt wurde, kann der Benutzer in disem Schritt den Daei- und Pfadnamen des zu erstellenden Dokuments eingeben.

Wenn im ersten Schritt E-Mail ausgewählt wurde, kann der Benutzer im zweiten Schritt zwischen drei möglichen Textquellen wählen.



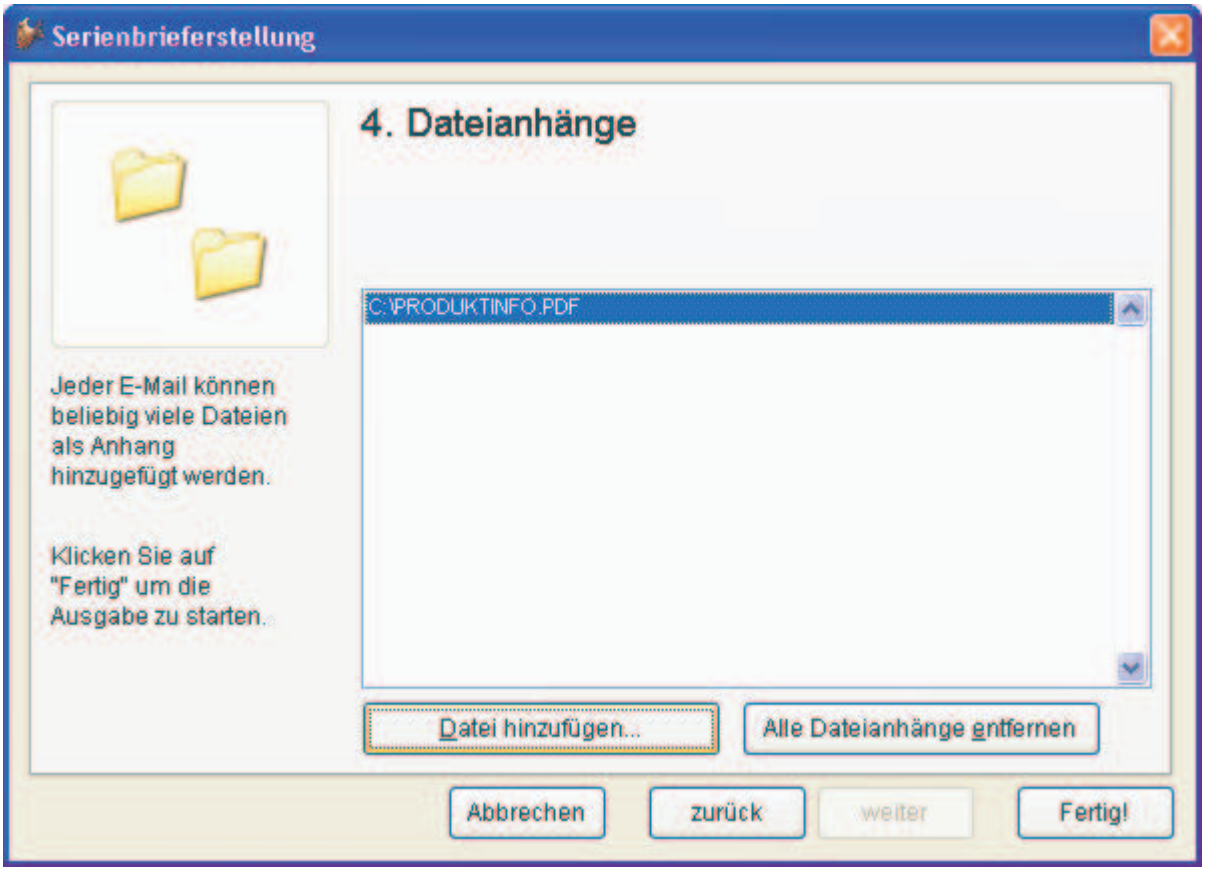
Wenn im zweiten Schritt ausgewählt wurde, dass ein Text manuell eingegeben werden soll, kann der Text im dritten Schritt erfasst werden.



Wenn im zweiten Schritt ein Textdokument ausgewählt wurde, kann dieser Text im dritten Schritt bei Bedarf geändert werden.

In Serienbriefdokumenten kann, ähnlich wie in Word, ein variabler Text verwendet werden. Dieser variable Text muss in speziellen Zeichen eingeschlossen sein. Standardmäßig sind diese Zeichen doppelte spitze Klammern, also „<<“ und „<<“. Die Begrenzungszeichen können vom Entwickler in den Eigenschaften *cLeftDelim* und *cRightDelim* der Klasse *cMailMerge* eingestellt werden.

Wenn E-Mails versendet werden sollen, können im vierten Schritt Anhänge hinzugefügt werden.



Durch einen Mausklick auf die Schaltfläche *Fertig* werden die Seriendokumente erstellt. Im letzten Schritt wird dem Benutzer die Anzahl der erstellten Seriendokumente angezeigt.

Die Klasse cMailMerge

Diese Formularklasse ist in der Klassenbibliothek *Vfxform.vcx* gespeichert.

Mit dieser Klasse können Endanwender anspruchsvolle Seriendokumente erstellen. Folgende Optionen stehen zur Verfügung:

1. E-Mail
   Erstellen von Serien-E-Mails. Der E-Mailtext kann aus einem Word-Dokument oder einer Textdatei stammen oder auch manuell in einer Editbox eingegeben werden. Wenn eine Text-E-Mail erstellt wird, können zusätzlich beliebig viele Dateien als Anhang versendet werden.
2. Word-Dokument
   Erstellen einer Word-Serienbriefausgabe basierend auf einem Word-Serienbriefdokument. Die Word-Serienbriefausgabe kann dann in Word beliebig weiterbearbeitet werden.
3. Fax
   Versenden von Serienfaxen basierend auf einem Word-Serienbriefdokument.
4. Drucken
   Drucken von Serienbriefen basierend auf einem Word-Serienbriefdokument.

Zur Seriendokumenterstellung muss ein Cursor vorhanden sein, der die erforderlichen Felder für jede mögliche Benutzerauswahl enthält.

Eigenschaften

cDataSource Enthält den Namen der Datenquelle für die Seriendokumenterstellung. Diese Datenquelle wird von Word oder vom Assistenten direkt verwendet. Alle variablen Felder müssen in dieser Datenquelle enthalten sein.

cMailAddressFieldName Enthält den Namen des Feldes für die E-Mailadresse. Dieser Feldname muss in dem in *cDataSource* angegebenen Cursor enthalten sein. Diese Eigenschaft wird nur beim Versand von E-Mails verwendet.

cCCFieldName Enthält den Feldnamen für eine CC E-Mailadresse. Dieser Feldname muss in dem in *cDataSource* angegebenen Cursor enthalten sein. Diese Eigenschaft wird nur beim Versand von E-Mails verwendet.

cBCCFieldName Enthält den Feldnamen für eine BCC E-Mailadresse. Dieser Feldname muss in dem in *cDataSource* angegebenen Cursor enthalten sein. Diese Eigenschaft wird nur beim Versand von E-Mails verwendet.

cFaxNumberFieldName Enthält den Namen des Feldes mit der Faxnummer. Dieser Feldname muss in dem in *cDataSource* angegebenen Cursor enthalten sein. Diese Eigenschaft wird nur beim Faxversand verwendet.

cLeftDelim Linkes Begrenzungszeichen für die Ersetzung von Text. Standardwert ist „<<“. Die hier angegebene Zeichenkette ist die linke Begrenzung eines variablen Wertes.

cRightDelim Rechtes Begrenzungszeichen für die Ersetzung von Text. Standardwert ist „>>“. Die hier angegebene Zeichenkette ist die rechte Begrenzung eines variablen Wertes.

cMergeText Nur zur internen Verwendung. Hier ist der Serientext gespeichert.

nEmailsSent Nur zur internen Verwendung. Zähler für die Anzahl der zu erstellenden Dokumente. Die Anzahl wird auf der letzten Seite des Wizard angezeigt.

nPreviousPageNum Nur zur internen Verwendung. Enthält die Nummer der vorhergehenden Seite im Wizard.

Methoden

LoadFileContent Lädt die im zweiten Schritt angegebene Textdatei. Der Text kann im dritten Schritt in einer Editbox bearbeitet werden.

SendMails Erstellen der Seriendokumente entsprechend der gewählten Optionen. Von hier wird eine der Methoden *SendThroughMapi* oder *SendThroughOleWord* aufgerufen.

SendThroughMapi Erstellt Serien-E-Mails unter Verwendung der VFX-Klasse *cEmail*.

SendThroughOleWord Diese Methode erstellt Serienbriefe per OLE-Automatisierung von Word. Auf diesem Weg kann die Serienbriefausgabe in ein Word-Dokument gespeichert, per Fax gesendet, gedruckt oder per E-Mail versendet werden.

Berichte

In VFX 9.5-Anwendungen ist die Standardeinstellung für Berichtsausgaben *Set Reportbehavior 90*. Damit können alle neuen Eigenschaften des Berichts-Designers und der Berichts-Engine von VFP 9 genutzt werden. Mit der Eigenschaft *nReportBehavior* des Anwendungsobjekts kann auf Wunsch das Reportbehavior auf 80 eingestellt werden. Diese Eigenschaft kann auch im VFX – Application Builder eingestellt werden.

Alle Berichte, die auf Grids basieren und zur Laufzeit erzeugt werden, können jetzt mit mehrzeiligen Detailbändern ausgegeben werden. Mit der Eigenschaft *nMultiLineReport* des Anwendungsobjekts kann das Verhalten eingestellt werden. Ein Wert von 0 bedeutet, dass mit der Eigenschaft *lMultiLineReport* auf Formularebene mehrzeilige Berichte aktiviert werden können. Bei einem Wert von 1 werden mehrzeilige Berichte in allen Formularen aktiviert. Mit dem Wert 2 wird das Verhalten von früheren VFX-Versionen eingestellt. Berichte werden einzeilig, bis zum rechten Papierrand bedruckt.

Berichte bearbeiten

Endbenutzer haben die Möglichkeit Berichtsdateien selbst zu bearbeiten. Dafür muss sich die Datei *Vfxmodifyreport.exe* im gleichen Ordner wie die Anwendung befinden. Außerdem muss der aktuelle Benutzer das Recht zur Berichtsbearbeitung haben. Dieses Recht kann in der Benutzerverwaltung sowie für Benutzergruppen für jeden Benutzer individuell von Administratoren mit der Benutzerstufe 1 eingestellt werden.

Der Start der Anwendung *Vfxmodifyreport.exe* aus dem Windows-Explorer ist nicht möglich. Eine unberechtigte Benutzung dieser Anwendung wird so verhindert.

Die Anwendung zur Bearbeitung der Berichtsdateien befindet sich aus Sicherheitsgründen in einer eigenen ausführbaren Datei. Diese Datei erhält beim Start als Parameter die aktuell eingestellte Sprache der Anwendung und startet somit lokalisiert. Zu beachten ist, dass einige der Dialoge von der Laufzeitumgebung von VFP stammen und damit in jedem Fall in der Sprache der VFP Laufzeitumgebung angezeigt werden.

Entwickler, die diese Anwendung programmatisch starten wollen, können als zweiten Parameter den Namen einer Berichtsdatei übergeben. Beim Start aus dem Menü *Extras* der Anwendung erscheint ein Öffnen-Dialog zum Öffnen einer Berichtsdatei.

Berichte werden in jedem Fall mit der Klausel *PROTECTED* des *MODIFY REPORT* Befehls bearbeitet. Dadurch können alle Schutzeinstellungen von Reportbehavior 90 genutzt werden.

ReportOutput und ReportPreview

Der Quellcode der VFP-Berichtsanwendungen *ReportOutput.app* und *ReportPreview.app* ist in VFX integriert worden. Selbstverständlich wurde der Code so angepasst, dass sowohl die Lokalisierung zur Entwicklungszeit als auch zur Laufzeit unterstützt werden.

Die Anwendung ReportOutput enthält alle verfügbaren ReportListener. Zusätzlich gibt es einen neuen ReportListener zur Erstellung von PDF-Dateien.

In der Seitenansicht von Berichten steht ein Rechtsklickmenü zur Verfügung. Hierüber kann der Benutzer die Berichtsausgabe drucken, in eine Datei speichern oder als E-Mail versenden.

Für das Drucken von Berichten steht ein erweiterter Druckdialog zur Verfügung. In diesem Dialog können die zu druckenden Seiten gewählt werden. Es ist auch möglich mehrere Seiten eines Berichts auf einer Seite zu drucken. Die Sortierfolge der Seiten kann eingestellt werden.

PDF-ReportListener

Dieser neue ReportListener wurde zusätzlich zu den bereits in VFX 9.0 vorhandenen ReportListenern hinzugefügt. Mit diesem ReportListener ist die Berichtsausgabe in PDF-Dateien möglich. Die Installation von Ghostscript ist nicht erforderlich.

Der PDF-ReportListener verwendet Dateien mit einer beträchtlichen Größe. Diese Dateien sind daher nicht in VFX-Anwendungen eingeschlossen. Die benötigten Dateien befinden sich im Projekt *PDFOutput.pjx*. Die aus diesem Projekt erstellte App-Datei kann im Exe-Ordner der Anwendung installiert werden und wird dann automatisch verwendet. Wahlweise kann die Datei *PDFOutput.app* in einem Zip-Archiv bei Bedarf automatisch aus dem Internet heruntergeladen werden. Der Download-Link befindet sich in der Tabelle *Vfxsys.dbf* im Feld *Install_GS*. Standardmäßig wird diese Datei von der Visual Extend-Webseite heruntergeladen.

```
D: http://files.visualextend.com/files95/PDFOutput.Zip
```

Hinweis: In früheren Versionen von VFX befand sich im Feld *Vfxsys.Install_GS* das Installationsskript für GhostScript. Wenn die Download Option von *PDFOutput.app* genutzt werden soll, ist in dieses Feld manuell der Download-Link einzutragen.

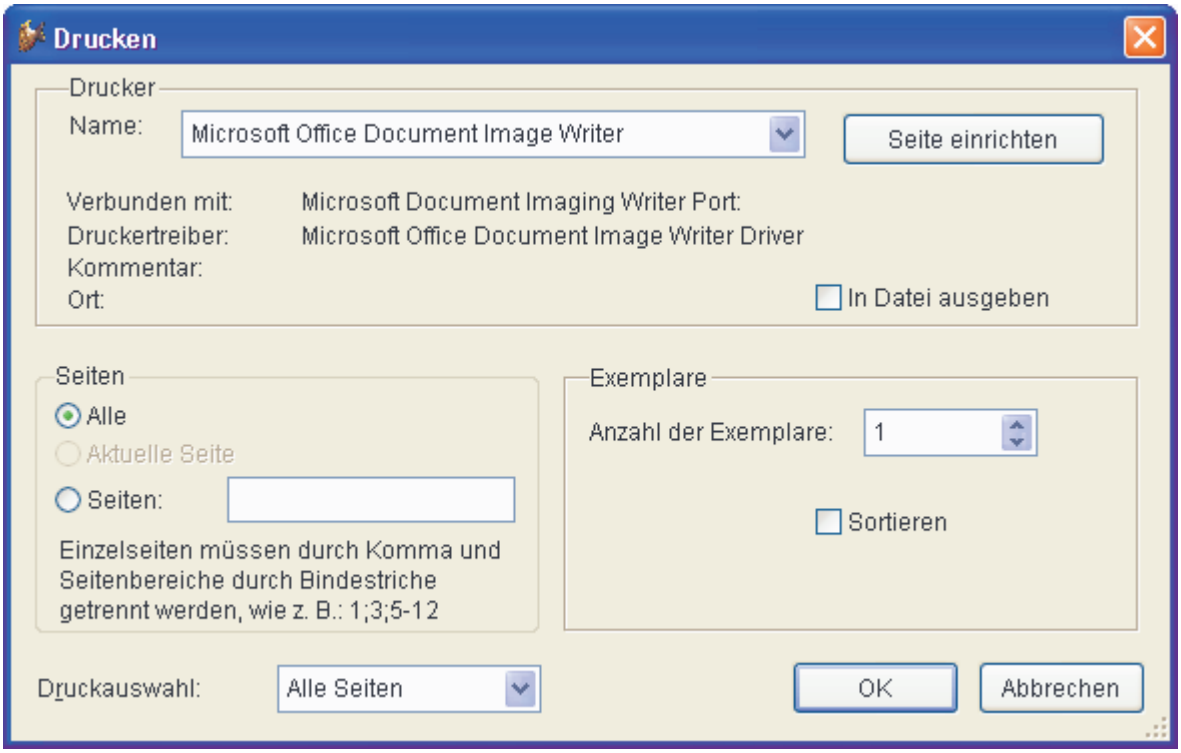
Die Installation von GhostScript ist für VFX 9.5-Anwendungen nicht erforderlich.

Erweiterter Druckdialog

In VFX 9.5-Anwendungen kann ein erweiterter Druckdialog verwendet werden, der es den Benutzern erlaubt die Druckausgaben genauer einzustellen. Der neue Druckdialog basiert auf der Klasse *cPrintDialog* und benutzt die neue Klasse *cPrintEngine*.

Die Klasse cPrintDialog

Die Klasse *cPrintDialog* ist in der Klassenbibliothek *Vfxform.vcx* gespeichert. Diese Klasse zeigt den erweiterten Druckdialog an.



In diesem Dialog kann der Drucker ausgewählt werden, die Seiteneinstellungen können verändert werden, die Ausgabe kann in eine Datei umgelenkt werden, die Anzahl der Exemplare kann eingestellt werden und die zu druckenden Seiten können ausgewählt oder eingegeben werden.

Eigenschaften

cPageRange – In dieser Eigenschaft steht die Auswahl der zu druckenden Seiten. Dieser Wert wird nur berücksichtigt, wenn ein manuell eingegebener Seitenbereich gedruckt werden soll (*nPagesSelectionType* = 3).

nAllOddEven - Auswahl zu druckender Seiten.
1 – Alle
2 – Ungerade Seiten
3 – Gerade Seiten

nCollate – Sortierfolge der Seiten.
0 – Exemplare werden nacheinander gedruckt.
1 – Zu allen Exemplaren wird zunächst die erste Seite gedruckt, dann werden alle zweiten Seiten gedruckt usw.

nNumberOfCopies – Anzahl zu druckender Exemplare.

nPagesSelectionType - Seitenauswahl
1 – Alle Seiten
2 – Aktuelle Seite
3 – Seitenbereich

nPrintToFile – Numerischer Wert mit dem Ausgabeziel.
0 – Drucken
1 – Ausgabe in eine Datei

oUnderlyingObject – Referenz auf ein Objekt der Klasse *cPrintEngine*.

Printers – Array mit den Informationen aller installierten Druckertreiber.

Die Klasse cPrintEngine

Diese Klasse dient zum drucken von Berichten. Als Parameter wird eine Referenz auf ein ReportListener-Objekt übergeben. Diese Klasse hat Methoden um einen Druckauftrag zu starten, mit Seiten zu füllen und zu schließen. Abhängig vom Wert der Eigenschaft *goProgram.nCustomPrintDialog* kann ein erweiterter Druckdialog angezeigt werden.

Dieser Dialog ermöglicht den Drucker, die Druckereigenschaften, die Anzahl zu druckender Exemplare, eine Auswahl von Seiten und die Sortierung der Seiten zu wählen. Der erweiterte Druckdialog ist eine Ableitung von der Klasse *cPrintDialog*.

Eigenschaften

aPagesToPrint – In diesem Array werden die zum Drucken ausgewählten Seiten gespeichert. Das erste Element jeder Zeile enthält die Nummer der Startseite. Das zweite Element jeder Zeile enthält die Nummer der Endseite. Eine einzeln zu druckende Seite wird in beide Elemente einer Zeile eingetragen. Wenn alle Seiten gedruckt werden sollen, enthält dieses Array nur eine Zeile.

cPageRange - In dieser Eigenschaft steht die Auswahl der zu druckenden Seiten. Dieser Wert wird nur berücksichtigt, wenn ein manuell eingegebener Seitenbereich gedruckt werden soll (*nPagesSelectionType* = 3).

cPrinterDriver - Name des aktuell ausgewählten Druckertreibers.

cPrinterName - Name des aktuell ausgewählten Druckers.

cPrintFileName - Name der Datei, wenn die Ausgabe in eine Datei umgelenkt werden soll.

lCustomPrintDialog – Soll der erweiterte Druckdialog angezeigt werden?
.T. – Anzeige des erweiterten Druckdialogs.
.F. – Anzeige des Standarddruckdialogs.

nAOE – Auswahl der zu druckenden Seiten.
1 – Alle Seiten
2 – Nur ungerade Seiten
3 – Nur gerade Seiten

nHeight – Höhe des bedruckbaren Bereichs des aktuellen Druckers.

nLeft – Linker Rand des bedruckbaren Bereichs des aktuellen Druckers.

nPagesSelectionType – Seitenauswahl.
1 – Alle Seiten
2 – Aktuelle Seite
3 – Seitenbereich

nPrinterHandle – Handle des verwendeten Druckertreibers.

nPrintToFile – Numerischer Wert mit dem Ausgabeziel.
1 – Berichtsausgabe in eine Datei.
0 – Drucken.

nShowDialog – Nur zur internen Verwendung. Gibt an, ob der Druckdialog angezeigt wurde.

nTop – Oberer Rand des bedruckbaren Bereichs des aktuellen Druckers.

nWidth – Breite des bedruckbaren Bereichs des aktuellen Druckers.

oRepListener – Referenz zum aufrufenden ReportListener.

Methoden

EndPrinterJob()
Beendet einen Druckauftrag. Bei erfolgreicher Ausführung wird *.T.* zurückgegeben. Im Fehlerfall wird *.F.* zurückgegeben.

GetPageProp()
Ermittelt die Eigenschaften Breite, Höhe, oberer Rand und linker Rand des bedruckbaren Bereichs des ausgewählten Druckers und speichert die Werte in den Eigenschaften *nWidth, nHeight, nTop, nLeft*. Bei erfolgreicher Ausführung wird *.T.* zurückgegeben. Im Fehlerfall wird *.F.* zurückgegeben.

GetPagesToPrint()
Füllt das Array *aPagesToPrint* entsprechend den Werten der Eigenschaften *nPagesSelectionType* und *cPageRange*. Wenn keine gültigen Seiten ermittelt werden können, wird das Array verworfen und dem Benutzer wird eine Meldung angezeigt. Bei erfolgreicher Ausführung wird *.T.* zurückgegeben. Im Fehlerfall wird *.F.* zurückgegeben.

OpenPrintDialog()
Zeigt den erweiterten Druckdialog an. Wenn der Benutzer auf die Schaltfläche *OK* klickt, werden die Methoden *GetPagesToPrint()* und *StartPrinterJob()* aufgerufen. Bei erfolgreicher Ausführung wird *.T.* zurückgegeben. Im Fehlerfall wird *.F.* zurückgegeben.

Output_Page(nPageNo)
nPageNo – Gibt die Nummer der zu druckenden Seite an. Es wird die ReportListener-Methode *OutputPage()* verwendet um die Seite an den Druckertreiber zu senden.

Print_Pages()

Liest das Array *aPagesToPrint* und ruft für jede Seite die Methode *Output_Page(nPageNo)* auf. Abschließend wird die Methode *EndPrinterJob()* aufgerufen. Bei erfolgreicher Ausführung wird *.T.* zurückgegeben. Im Fehlerfall wird *.F.* zurückgegeben.

StartPrinterJob()

Stellt den im erweiterten Druckdialog selektierten Drucker als aktuellen Drucker ein und startet den Druckauftrag. Rückgabewert ist ein Handle zu dem Druckauftrag.

XP-Öffnen-Dialog

Die Funktionalität des XP-Öffnen Dialogs wurde so erweitert, dass darüber nicht nur Formulare geöffnet werden können, sondern wahlweise auch ein Befehl ausgeführt werden kann. Um Befehle auszuführen wird das Feld *Form* in der Tabelle *Vfxfopen* leer gelassen. Der auszuführende Befehl wird in das Feld *Parameter* eingetragen. Hierüber lassen sich insbesondere Prozeduren, Funktionen und Methoden aufrufen.

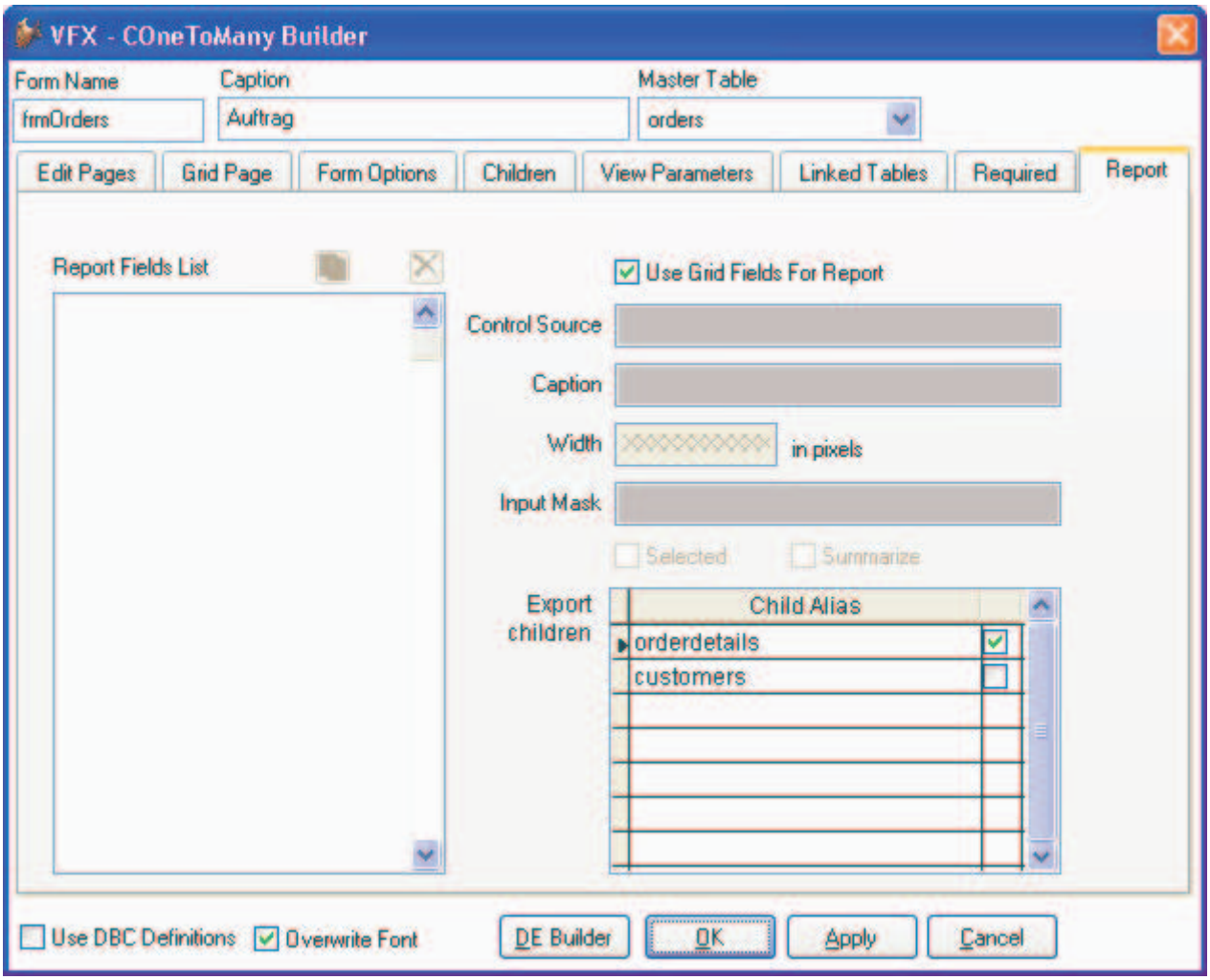
In der Tabelle *Vfxfopen.dbf* gibt es jetzt ein Feld *Iconfile*. In diesem Feld kann der Dateiname zu einem Icon zu dem aktuellen Formular gespeichert werden. Dieses Icon wird im XP-Öffnen-Dialog angezeigt.

Favoriten erscheinen jetzt auch im XP-Öffnen-Dialog in eigenen Gruppen je Formular. Jeder Benutzer kann im Anpassen-Dialog individuell für sich einstellen, ob Favoriten im XP-Öffnen-Dialog angezeigt werden sollen.

Datenexport

Im Menü für Endanwender gibt es im Menü unter *Datei* den neuen Menüpunkt *Export als*. Darunter gibt es die Auswahlmöglichkeiten *CSV*, *Excel*, *XML* und *DBF*. Diese Menüpunkte sind aktiviert, wenn ein Formular mit Daten geöffnet und aktiv ist. Die Auswahl einer dieser Optionen öffnet einen *Speichern unter*-Dialog. Nach Eingabe eines Dateinamens werden die Daten aus dem *Initialselectedalias* des Formulars in einer Datei mit dem gewählten Dateiformat gespeichert. Die aktuelle Sortierung sowie ein eventuell gesetzter Filter werden berücksichtigt. Es werden alle Felder exportiert.

Wenn als Exportformat *XML* gewählt wird, können aus Onetomany-Formularen wahlweise auch die Child-Daten exportiert werden. Diese Möglichkeit besteht nur dann, wenn die Parent- und Child-Daten über eine Relation in einer Beziehung stehen. Alle an der Relation beteiligten Child-Tabellen können im *XML*-Format exportiert werden. Im VFX – COneToMany Builder kann auf der Seite Report eingestellt werden, welche Child-Tabellen mit exportiert werden sollen.



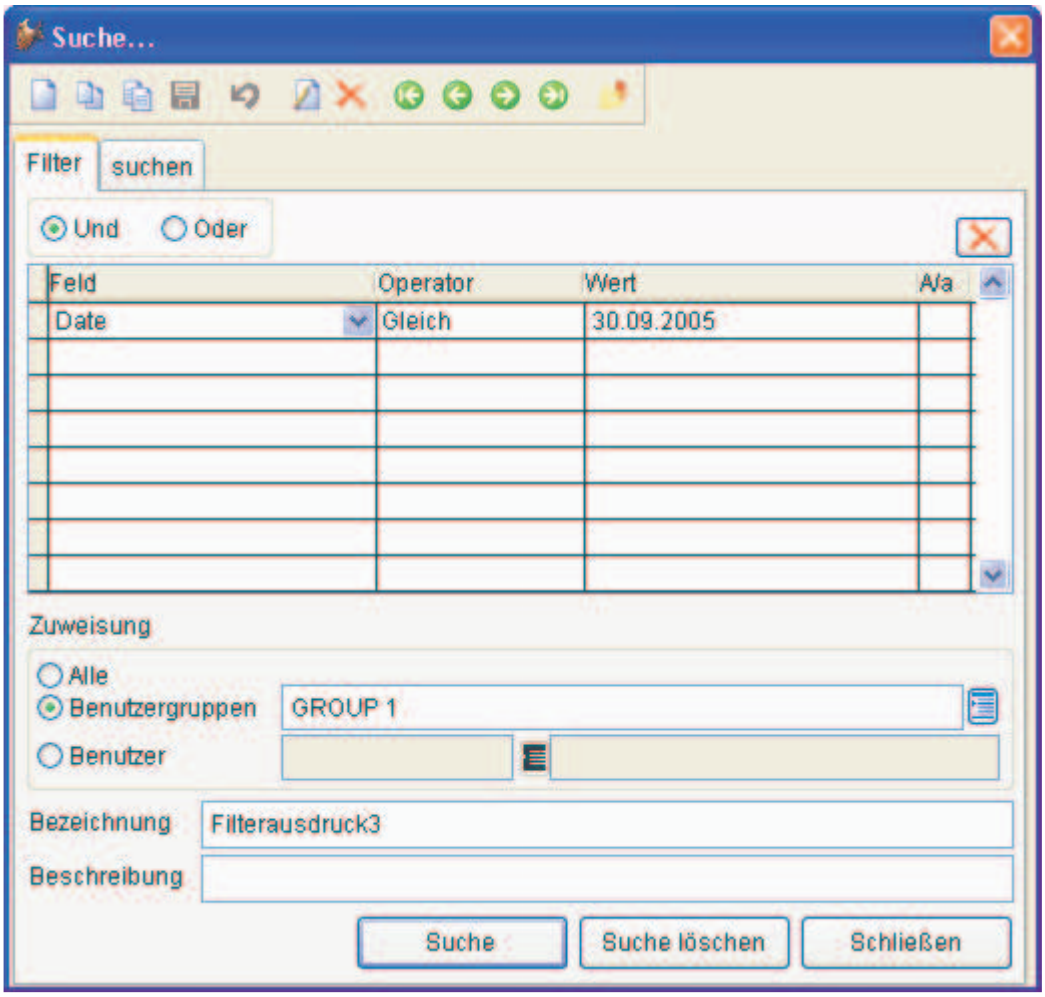
Wenn Child-Daten für den *XML*-Export markiert wurden, wird der Benutzer zur Laufzeit gefragt, ob die Child-Daten mit exportiert werden sollen.

Suchdialog

Filtereinstellungen können jetzt je Formular und je Benutzer oder Benutzergruppe gespeichert werden. Der Suchdialog wurde um Steuerelemente zur Verwaltung der Filterdefinitionen erweitert. Filterdefinitionen können für andere Benutzer kopiert werden.

Über die Eigenschaft *nFilterBehavior* in der Klasse *cFoxAppl* kann eingestellt werden, ob die neuen Eigenschaften des Suchdialogs von VFX 9.5 genutzt werden sollen oder ob der Suchdialog die gleichen Funktionen wie in VFX 9.0 haben soll. Es ist auch möglich den gewünschten Suchdialog für jedes Formular einzustellen. Hierfür ist *cFoxAppl.nFilterBehavior*=0 einzustellen. Die Formulareigenschaft *nFilterBehavior* kann auf 1 eingestellt werden, um das zu VFX 9.0 kompatible Verhalten einzustellen. Mit dem Wert 2 wird der neue Suchdialog aktiviert.

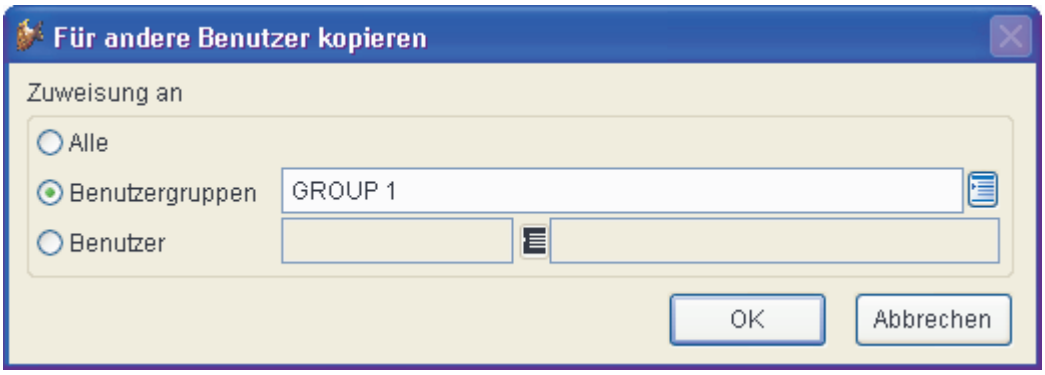
Im neuen Suchdialog stehen die gleichen Filteroptionen wie in VFX 9.0 zur Verfügung. Zusätzlich können Benutzer die Einstellungen speichern und anderen Benutzern oder Benutzergruppen zur Verfügung stellen.



Eine Filtereinstellung kann allen Benutzern, einer Benutzergruppe oder einem Benutzer zugänglich gemacht werden. Jeder Filtereinstellung kann ein Name und eine Beschreibung gegeben werden. Die Filtereinstellungen werden zum aufrufenden Formular gespeichert und können später wieder verwendet werden.

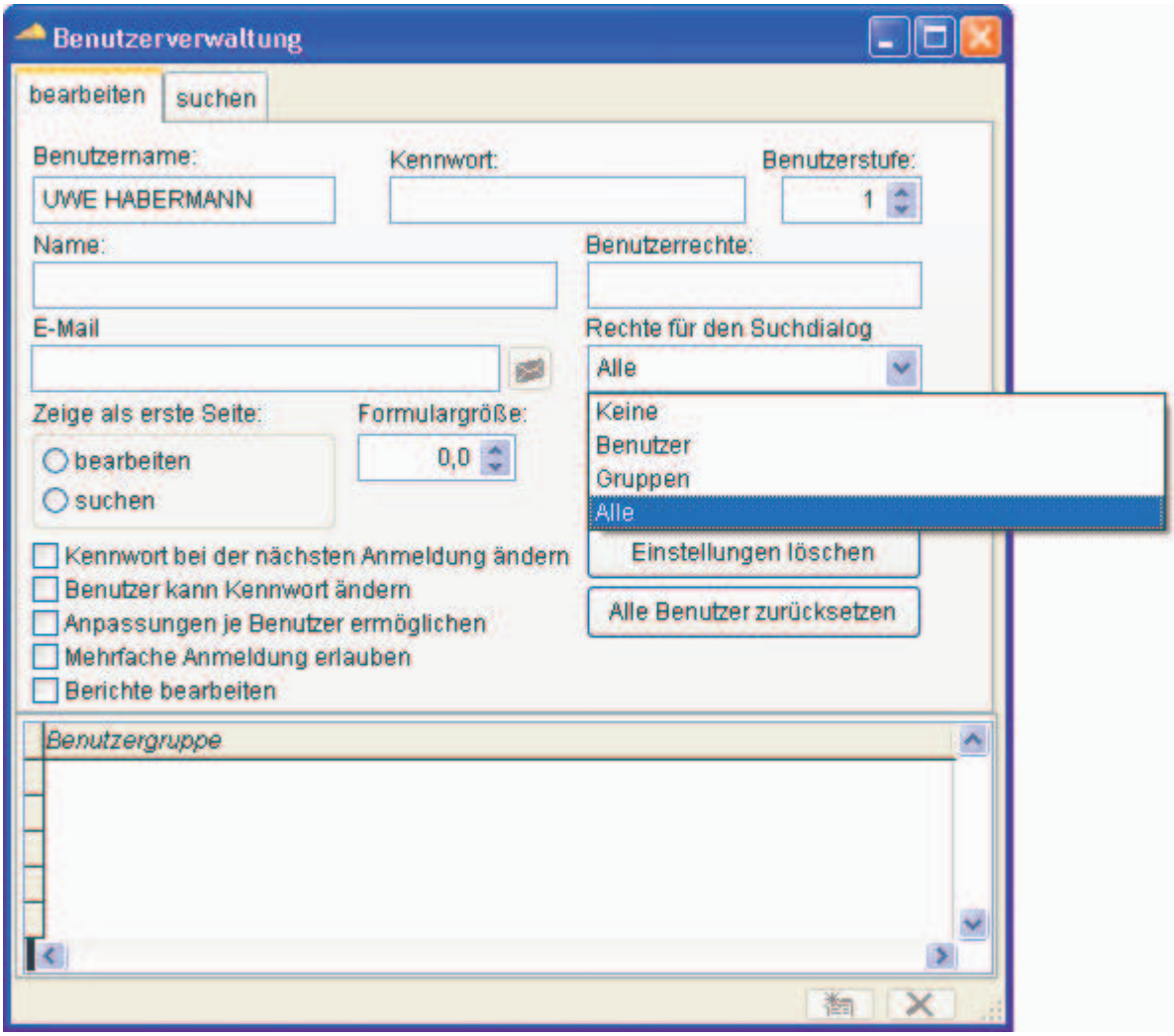
Ein Benutzer kann seine eigenen Filtereinstellungen sehen sowie die Filtereinstellungen, die für alle Benutzer oder für Benutzergruppen freigegeben sind, in denen der Benutzer Mitglied ist. Für jedes Formular kann im Rechtedialog bzw. in der Verwaltung der Benutzergruppen eingestellt werden, welche Benutzerstufe erforderlich ist, um Filtereinstellungen bearbeiten zu können.

Benutzer können Filtereinstellungen neu anlegen, kopieren, für andere Benutzer kopieren, bearbeiten und löschen. Wenn eine Filtereinstellung für andere Benutzer kopiert werden soll, erscheint ein Dialog zur Auswahl des Benutzers bzw. der Benutzergruppe.



Wenn ein eingegebener Filter zu einer leeren Ergebnismenge führt, wird dies dem Benutzer in einer Messagebox angezeigt und der Suchdialog bleibt geöffnet. Wenn alle Filtereinstellungen gelöscht werden, bleibt der Suchdialog ebenfalls geöffnet. Einzelne Zeilen im Suchdialog können jetzt über eine Schaltfläche gelöscht werden.

Die Rechte für den erweiterten Suchdialog können in der Benutzerverwaltung und in der Verwaltung der Benutzergruppen eingestellt werden.



Für jeden Benutzer bzw. für jede Benutzergruppe kann eingestellt werden:

- Keine – Der Benutzer kann Filterbedingungen nicht speichern. Er kann aber Filterbedingungen verwenden, die von anderen Benutzern gespeichert wurden. Dies ist die Standardeinstellung.

- Benutzer – Der Benutzer kann Filterbedingungen zu seiner eigenen Verwendung speichern. Er kann seine Filterbedingungen nicht anderen Benutzern zur Verfügung stellen.

- Gruppen – Der Benutzer kann Filterbedingungen für sich und für Benutzergruppen speichern.

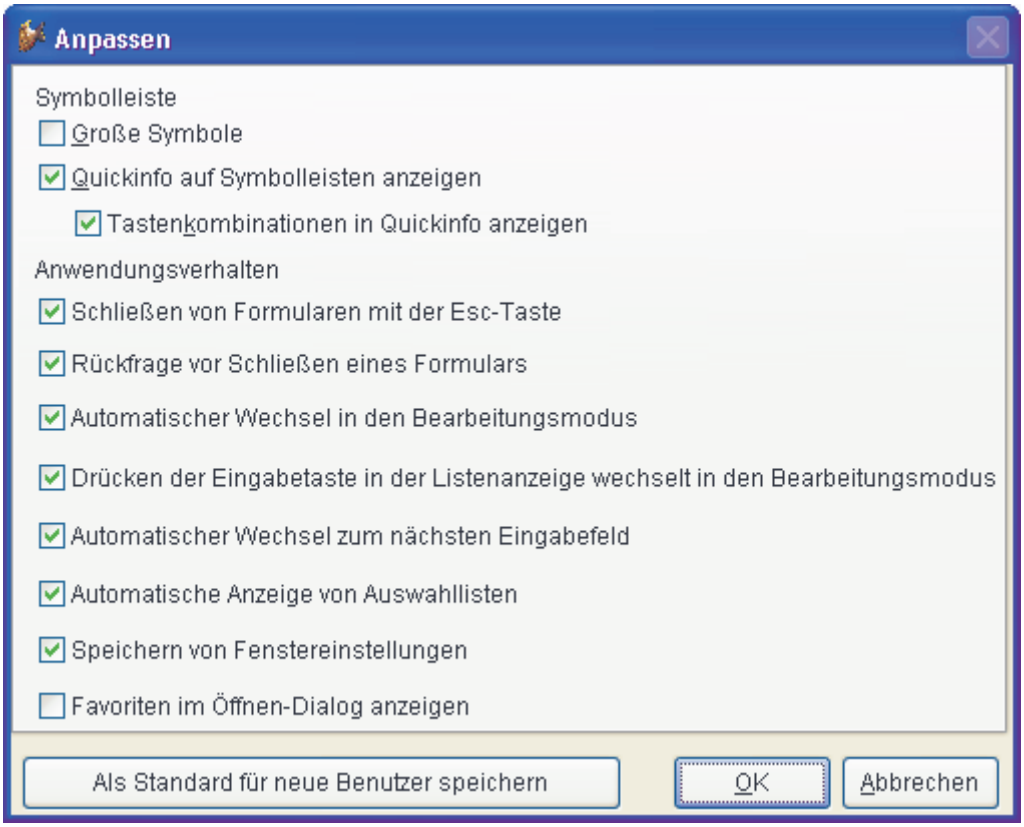
- Alle – Der Benutzer kann Filtereintellungen für jeden speichern.

Wenn ein Benutzer Mitglied in mehreren Gruppen ist, gilt das höchste Recht.

Anpassen-Dialog

Viele Eigenschaften der Anwendung kann sich jeder Benutzer individuell selbst anpassen. Die Anpassbarkeit dieser Einstellungen kann über die Eigenschaft *lAllowUserCustomization* des Anwendungsobjekts für die Anwendung gesteuert werden. Wenn der Wert dieser Eigenschaft .F. ist, ist der Anpassen-Dialog in der Anwendung nicht sichtbar. Wenn der Wert dieser Eigenschaft .T. ist, kann der Administrator für jeden Benutzer individuell erlauben den Anpassen-Dialog zu verwenden.

Der Anpassen-Dialog kann aus dem *Optionen*-Menü aufgerufen werden.



Der Anpassen-Dialog kann nur von Benutzern angezeigt werden, die das Recht *Anpassungen je Benutzer ermöglichen* haben. Dieses Recht kann nur von Administratoren vergeben werden, wenn die Eigenschaft *goProgram.lAllowUserCustomization* auf .T. eingestellt ist.

Über den Anpassen-Dialog kann jeder Benutzer seine individuellen Einstellungen zu Symbolleisten und zum Anwendungsverhalten machen.

Die Schaltfläche *Als Standard für neue Benutzer speichern* ist nur für Administratoren sichtbar. Wenn ein Administrator auf diese Schaltfläche klickt, werden die aktuell sichtbaren Einstellungen als Standardwerte für neu anzulegende Benutzer gespeichert. Wenn sich ein neuer Benutzer erstmalig anmeldet, gelten diese Standardwerte.

Mit der Schaltfläche *OK* werden die aktuell sichtbaren Einstellungen für den angemeldeten Benutzer übernommen. Mit der Schaltfläche *Abbrechen* werden die Einstellungen verworfen.

Die Klasse CArchive

Dieser Klasse wurde die neue Eigenschaft *lQuietMode* hinzugefügt. Über diese Eigenschaft kann gesteuert werden, ob alle Anzeigen während eines Archivierungsvorgangs unterdrückt werden sollen.

lQuietMode – Gibt an, ob die Archivierung ohne Meldungen ausgeführt werden soll. Diese Eigenschaft muss eingestellt werden, bevor eine der Methoden *CreateArchive()* oder *ExtractFromArchive()* ausgeführt wird.

.T. – Meldungen unterdrücken (unterdrückt auch die Fortschrittsanzeige)
.F. – Meldungen anzeigen

Die neue Klasse CTextSkype

Die neue Klasse *CTextSkype* befindet sich in der Klassenbibliothek *Vfxctrl.vcx*. Diese Klasse besteht aus einem Container mit einer Textbox und einer Schaltfläche.

Wenn der Benutzer zur Laufzeit auf die Schaltfläche klickt, wird der in der Textbox befindliche Wert als Skype-Name an das Programm Skype übergeben. Mit Skype ist es möglich Telefongespräche über das Internet zu führen und Sofortnachrichten zu senden. Mehr Informationen zu Skype finden Sie im Internet auf http://www.skype.com.

Behandlung von Laufzeitfehlern

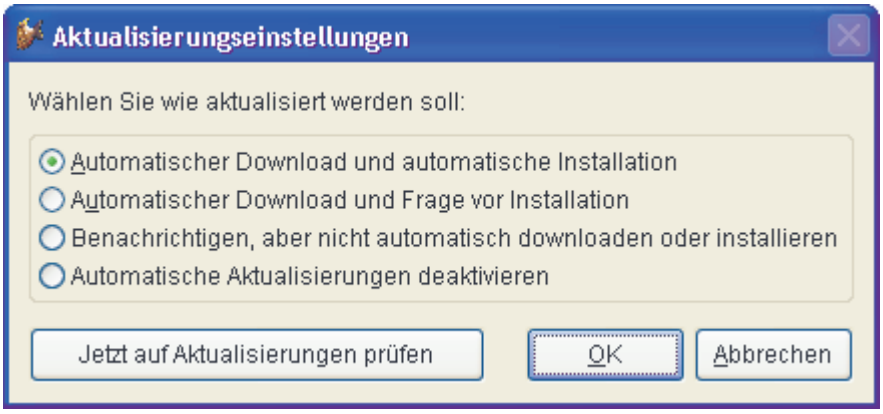
Wenn ein Laufzeitfehler auftritt, prüft die Funktion *OnError* ob eine Verbindung zur Remote Datenbank besteht. Wenn möglich, werden die Fehlerinformationen in der Tabelle *Vfxlog* in der Remote Datenbank gespeichert. Wenn eine Verbindung zur Remote Datenbank nicht möglich ist, werden die Fehlerinformationen in der lokalen Tabelle *Vfxlog.dbf* gespeichert.

Wenn eine VFP-Datenbank verwendet wird, werden die Fehlerinformationen immer in der lokalen Tabelle *Vfxlog.dbf* gespeichert.

Wenn ein Fehler beim Speichern eines Datensatzes auftritt, wird jetzt zusätzlich der Name des Arbeitsbereichs, in dem der Fehler aufgetreten ist, im Fehlerprotokoll gespeichert. Dies hilft bei der Lokalisierung des Problems.

Aktualisierung der Anwendung

Mit VFX erstellte Anwendungen können über das Internet aktualisiert werden. Wenn die Eigenschaft *lAllowUpdates* des Anwendungsobjekts auf *.T.* eingestellt ist, ist die Aktualisierungsfunktion in der Anwendung aktiviert und die Menüpunkte *Aktualisierung der Anwendung* und *Aktualisierungseinstellungen* stehen zur Verfügung.



Im Dialog *Aktualisierungseinstellungen* können Benutzer zwischen vier Optionen wählen. Wenn automatische Aktualisierungen nicht deaktiviert sind, prüft die Anwendung täglich bei jedem ersten Start ob Aktualisierungen vorliegen.

Die Aktualisierungsfunktionen sind in den Klassen *cUpdate* und *cUpdateEngine* implementiert.

Aktualisierung der Datenbank beim Kunden

Die Aktualisierung der Datenbank beim Kunden wurde erweitert. Das neue Verfahren ist kompatibel zu bisherigen Versionen von VFX. Die Aktualisierung kann also weiterhin dadurch erfolgen, dass eine neue Datenbankstruktur im *Update*-Ordner an den Kunden ausgeliefert wird.

Um das neue Verfahren aus VFX 9.5 zu nutzen, muss die Exe-Datei mir einer Versionsnummer versehen werden. Es empfiehlt sich die Versionsnummer automatisch bei jedem Build von VFP erhöhen zu lassen. Bei jedem Erstellen einer Exe-Datei werden über einen Project Hook für alle Datenbanktypen aus der Datei *Config.vfx* im Projektordner Metadaten erstellt. Die Metadaten werden in die Exe-Datei eingeschlossen und stehen so beim Kunden zur Verfügung.

Wenn eine Exe-Datei gestartet wird, wird die Versionsnummer mit der Nummer verglichen, die im Feld *AppVersion* in der Tabelle *Vfxsys* gespeichert ist. Wenn die Version der aktuellen Exe-Datei größer als die gespeicherte Versionsnummer ist, wird die Aktualisierung der Datenbankstruktur gestartet. Die Aktualisierung wird für jede Datenbank durchgeführt, die in *Config.vfx* oder in *Vfxpath.dbf* eingetragen ist. Es werden dabei VFP-Datenbanken und Remote Datenbanken aktualisiert. Anschließend wird die neue Versionsnummer im Feld *AppVersion* in der Tabelle *Vfxsys* gespeichert.

Datenbankreparatur

Die Datenbankreparaturfunktion von VFX 9.0 wurde erheblich verbessert. Um eine Datenbank reparieren zu können muss die Struktur der korrekten Datenbank bekannt sein. Zu diesem Zweck führt der Project Hook, der in allen VFX-Projekten aktiv ist, bei jedem Erstellen einer Anwendung das Programm *Gendbc.prg* aus. *Gendbc.prg* wird mit VFP geliefert. Dieses Programm liest einen Datenbank-Container aus und generiert dabei eine Programmdatei. Die spätere Ausführung dieser Pro-

grammdatei führt dazu, dass eine neue, leere Datenbank mit der gleichen Struktur hergestellt wird. Diese Datenbankstruktur kann für die Reparatur einer beschädigten Datenbank verwendet werden.

Die von *Gendbc.prg* erzeugte Programmdatei wird automatisch in das Projekt eingeschlossen und steht somit zur Ausführung in einer Exe-Datei zur Verfügung. Wenn eine Datenbank gespeicherte Prozeduren enthält, werden diese von *Gendbc.prg* in ein Memofeld einer Tabelle kopiert. Diese generierte Tabelle wird ebenfalls automatisch in das Projekt eingeschlossen.

Ein defekter DBC kann in den meisten Fällen sofort beim Start der Exe-Datei erkannt werden und wird dann ohne Fragen an den Benutzer sicher repariert.

Zur Reparatur aller weiteren Beschädigungen muss manuell aus dem Menü Extras, Datenbankwartung aufgerufen werden.

Diese Probleme lassen sich über den Dialog Datenbankwartung beheben:

- Beschädigte Tabellenköpfe, insbesondere auch defekte Satzzähler, werden in jedem Fall ohne Datenverlust wiederhergestellt.

- Defekte Datensätze, zum Beispiel mehrfach vorkommende Primärschlüssel, können sicher erkannt werden und die doppelten Datensätze werden automatisch gelöscht. Anschließend kann die Tabelle wieder geöffnet und verwendet werden. Es ist sicher, dass nur zerstörte Datensätze verloren gehen.

- Fehlende DBF-Dateien können wiederhergestellt werden. Die Daten der betreffenden Tabelle sind dann natürlich verloren.

- Fehlende CDX-Dateien werden in jedem Fall ohne Datenverlust wiederhergestellt.

- Beschädigte CDX-Dateien können in vielen Fällen erkannt und ohne Datenverlust wiederhergestellt werden.

- Defekte Memofelder können in fast allen Fällen repariert werden. Die Memos des betreffenden Datensatzes gehen dabei verloren.

- Fehlende Memo-Dateien können wiederhergestellt werden. Dabei gehen natürlich alle Memos der betreffenden Tabelle verloren.

Tabellen können repariert werden, während der DBC von anderen Benutzern geöffnet ist. Zur Reparatur ist in jedem Fall nur der exklusive Zugriff auf die zu reparierenden Dateien erforderlich. Ausgenommen, wenn im Dialog „gesamte Datenbank“ gewählt wird. Hierzu ist der exklusive Zugriff auf alle Dateien inklusiv DBC erforderlich.

Terminalserver-Unterstützung

Wenn eine Anwendung in einer Terminalserver-Sitzung läuft, wird automatisch die Bitmap-Anzeige mit dem VFP-Befehl *SYS(602)* optimiert.

Zur weiteren Optimierung der Darstellung von Symbolleisten in Anwendungen, die in Terminalserver-Sitzungen laufen sollen, empfehlen wir der *Visible* Eigenschaft von Steuerelementen nur dann Werte zuzuweisen, wenn dies unbedingt erforderlich ist. Dadurch werden unnötige ausgeführte *REFRESH()*Ereignisse vermieden und die Symbolleiste wird flackerfrei angezeigt.

Beispiel-Code für das *REFRESH()* Ereignis in *cAppToolBar* oder *cAppNavBar*:

```
* Nicht empfehlenswert:
DODEFAULT()
This.cmdNew.Visible = .F.

* Empfehlenswert:
DODEFAULT()
IF This.cmdNew.Visible
   This.cmdNew.Visible = .F.
ENDIF
```

Weitere Verbesserungen für Endbenutzer

Alle benutzerspezifischen Einstellungen, wie Formulargröße, Position auf dem Bildschirm und Grid-Einstellungen können jetzt wahlweise entsprechend der verwendeten Bildschirmauslösung gespeichert und geladen werden. Dafür ist die Eigenschaft *lSaveFormLayoutResolutionDependent* in der Klasse *cFoxAppl* in der Klassenbibliothek *Appl.vcx* auf *.T.* zu stellen. Der Standardwert ist *.F.* Diese Einstellung kann auch mit dem VFX – Application Builder gemacht werden.

Die Archivierungsfunktion aus dem Menü erstellt jetzt Dateinamen, die aus dem Ordnernamen, dem Datenbanknamen sowie dem aktuellen Datum im ANSI-Format bestehen.

In Formularen mit einem Treeview-Steuerelement erfolgt die Navigation mit den Schaltflächen vor, zurück, Anfang und Ende in der Symbolleiste, jetzt entsprechend der logischen Anzeigefolge im Treeview-Steuerelement.

In VFX 9.0 konnten die Daten aus allen Grids per Drag & Drop in andere Anwendungen gezogen werden. In VFX 9.5 ist dieses Verhalten global und je Grid einstellbar. Wenn die Eigenschaft *nOLEDragGrid* der Klasse *cFoxAppl* in der Klassenbibliothek *Appl.vcx* auf *1* eingestellt wird, können die Daten aus allen Grids der Anwendung per Drag & Drop in andere Anwendungen gezogen werden. Dies war das Standardverhalten in VFX 9.0. Wenn der Wert dieser Eigenschaft auf *0* eingestellt wird, kann dies für jedes Grid individuell mit der Eigenschaft *lOLEDragGrid* eingestellt werden. *1* ist der Standardwert. Wenn der Wert der Eigenschaft *nOLEDragGrid* auf *2* eingestellt wird, ist OLE Drag & Drop in allen Grids der Anwendung ausgeschaltet.

Bereits in VFX 9.0 konnte mit der Eigenschaft *lKeepLoggedUsers* des Anwendungsobjekts eingestellt werden, ob Benutzeranmeldungen protokolliert werden sollen. Für eine Anwendung konnte mit dem VFX – Application Builder eingestellt werden, ob sich Benutzer mehrmals gleichzeitig anmelden dürfen. In VFX 9.5 kann der Administrator für jeden Benutzer individuell einstellen, ob eine mehrfache Anmeldung erlaubt ist. Zu diesem Zweck gibt es in der Benutzerverwaltung das Kontrollkästchen *Mehrfache Anmeldung erlauben*.

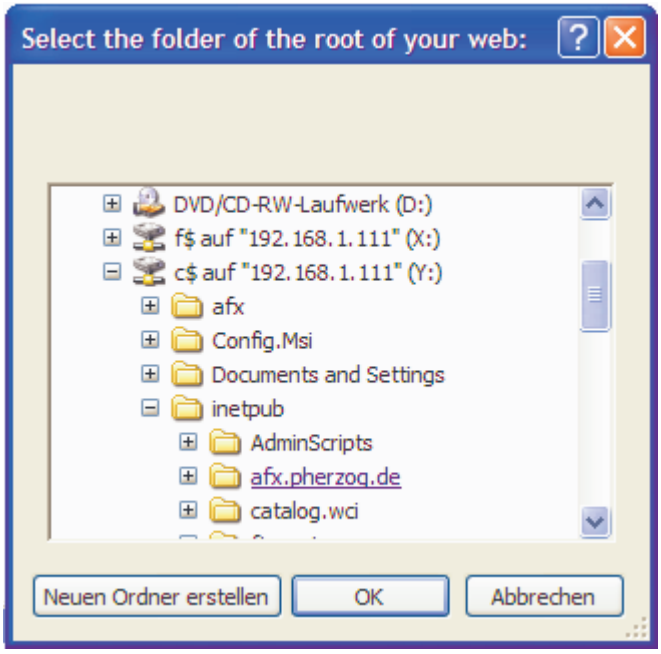
Das Menü *Hilfe* in Anwendungen wurde erweitert. Über den Eintrag *Besuchen Sie unsere Website* kann der Benutzer die Website besuchen, deren URL in der Eigenschaft *cCompanyWebSiteURL* des Anwendungsobjekts hinterlegt ist. Über den Menüpunkt *So erreichen Sie uns* können dem Benutzer Kontaktinformationen angezeigt werden. Als Kontaktinformation dient eine HTML Datei, die in der Tabelle *Vfxinternfiles.dbf* im Datensatz mit *type="contactus"* gespeichert ist.

Anhang I - VFX AFX Wizard

von Peter Herzog

Mit dem VFXAFXWizard können sie VFXMasken, welche mit dem Form Wizard erzeugt wurden, in Internetfähige DHMTL Masken umwandeln.

Sobald Sie den Wizard das erste Mal starten, werden Sie aufgefordert das Ausgabeverzeichnis für die erzeugten Dateien anzugeben.

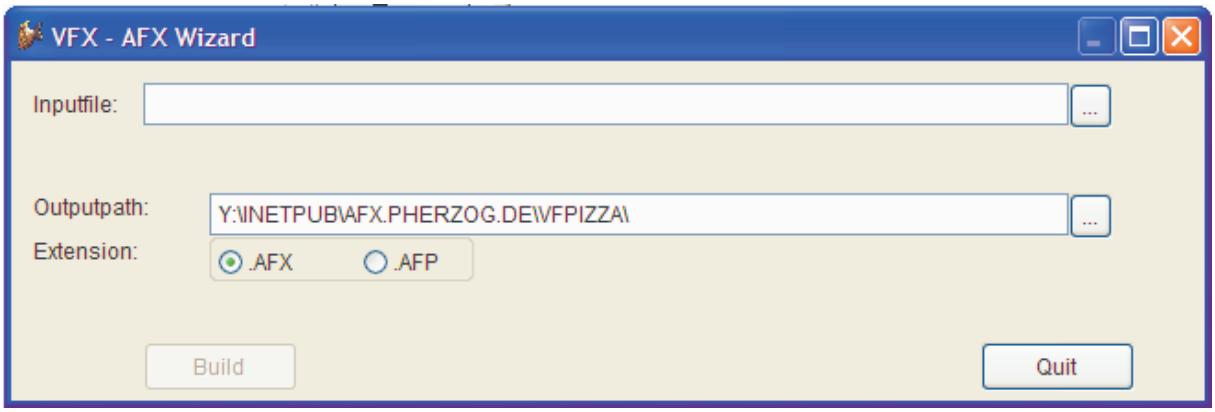


Sollten Sie bereits im internen Netzwerk einen Internetserver besitzen, so können Sie bereits hier den Pfad angeben, in dem Ihre Daten abgespeichert werden sollen.
(HKLM\SOFTWARE\Microsoft\InetStp)
Der Wizard sucht sich in der Registry den Pfad eines eventuell lokal installierten IIS und schlägt diesen Pfad bereits vor.

Nun werden alle notwendigen Dateien, wie Bilder, Stylesheets, vorgefertigte HTML-Seiten unter
`C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\dFPUG\Visual Extend\9.5\Wizfiles`
angelegt.

Und es wird die Metadatentabelle VFXAFXMETA.DBF unter
`C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\dFPUG\Visual Extend\9.5`
erzeugt.

Nun erscheint die Maske des Wizards.



Der vorher ausgewählte Pfad ist als Outputpath: voreingestellt. Jede Änderung wird in der VFXAFXMETA.DBF gespeichert.

Es kann hier gewählt werden, ob .AFX als Extension verwendet werden soll, oder .AFP
Der erzeugte Code ist identisch, da beide Script Engines gleichermaßen den Code abarbeiten können.

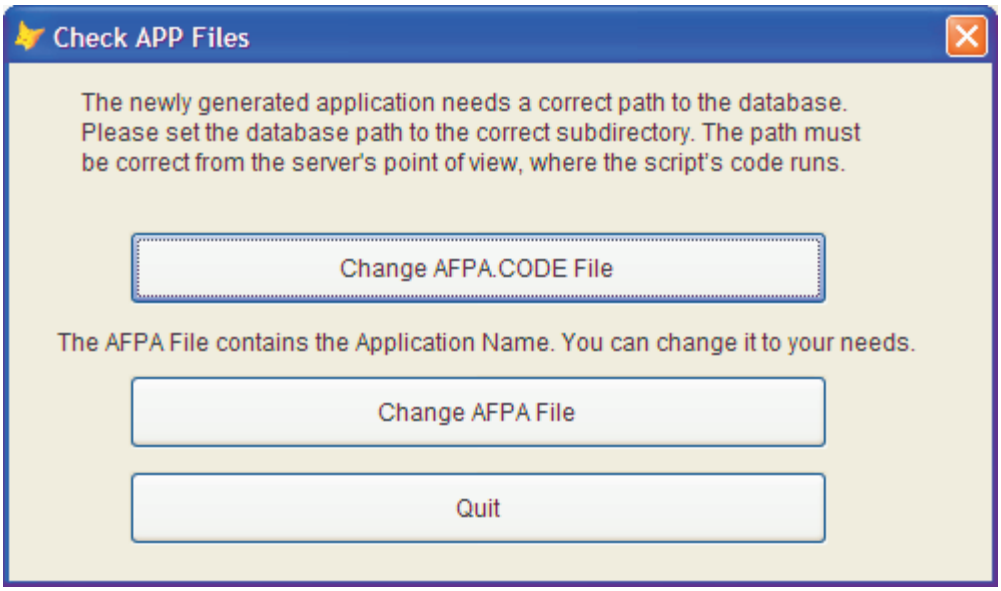
Sobald Sie eine Maske aus Ihrem Projekt ausgewählt haben und durch einen Klick auf „Build“ den Wizard starten, wird im Hintergrund die komplette Umgebung geladen, Dateien geöffnet und eventuelle SQL-Server anfragen ausgeführt.

Sie müssen sich an der laufenden Applikation auch anmelden, damit die Form mit den notwendigen Rechten geöffnet werden kann.

Nun wird die laufende Form Objekt für Objekt analysiert und mit Hilfe der Meta-Tabelle eine lauffähige HTML-Seite erzeugt.

Bei jedem Aufruf wird überprüft ob unterhalb des Ausgabepfades die notwendigen Zusatzdateien vorhanden sind. Sollte eine Datei fehlen, so wird sie aus
`C:\Dokumente und Einstellungen\All Users\Anwendungsdaten\dFPUG\Visual Extend\9.5\Wizfiles`
geholt.

Nachdem alle Dateien erzeugt wurden und auch die .AFPA bzw. .AFPA.CODE erzeugt wurden, werden Sie aufgefordert, den richtigen Pfad und den Applikationsnamen aus Sicht des Servers einzustellen.



Achten Sie auf folgenden Codeteil:

```
with goprogram
   .oConnMgr = Createobject("cConnectionMgr")
   .SetupDataAccessProps(.f.,1)
   .cdatadir="c:\vfx90traders\data\"
   .cmaindatabase="tastrade.DBC "
   .cvfxdir=justpath(File.cLocation)
   .lautologin = .f.
   .ldebugmode = .f.
   .cdatasourcetype = "Native"
Endwith
```

Hier muss der Pfad unter .cdatadir und die Datenbank unter .cmaindatabase eingestellt werden. Unter Umständen müssen Sie die notwendigen Dateien auch noch auf den Server kopieren.

Abschließend wird in der vfxfopen.dbf bei der generierten Maske das Feld inetlevel auf 1 gesetzt. Erst dadurch ist es möglich in xpopendir.af(x/p) die Maske als Link aufzurufen.

Wichtiger Hinweis

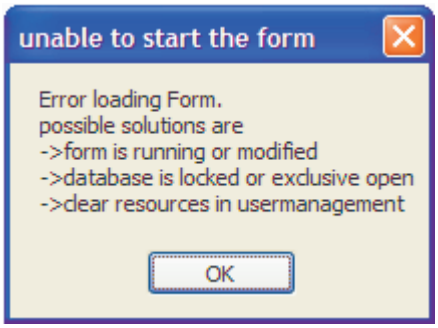
Achten Sie immer darauf, dass der Pfad **aus Sicht des Servers** einzustellen ist. Dies bedeutet, das z.B.: das Rootverzeichnis des Webservers meistens auf dem Laufwerk C liegt. Sollten Sie ein Laufwerk auf dem Server gemapped haben, so ist dies ein anderer Laufwerksbuchstabe. Da die AFX oder AFP aber den Server als „Arbeitsplatz“ sehen, ist das Laufwerk C für sie die Hauptpartition. Achten Sie außerdem darauf, dass die Daten möglichst nicht unterhalb der HTML Seiten liegen, da sie ansonsten unter Umständen aus dem Internet heraus lesbar sind.

Wird ein Server neu installiert, so liegt das Rootverzeichnis unter c:\inetpub\wwwroot. Und wird dem Server nun die Domaine www.meinedomain.de zugewiesen, so wird per default die Datei c:\inetpub\wwwroot\default.htm gelesen. Wenn nun die Daten unterhalb von c:\inetpub\wwwroot gespeichert werden, sind diese unter Umständen direkt aus dem Internet heraus ansprechbar und werden, wenn nicht extra gesichert, sogar über das Internet ladbar.

Sie sollten daher die Daten immer außerhalb dieses Verzeichnisses speichern. Nun aber müssen sie aber auch darauf achten, dass die AFX oder AFP auf die Daten zugreifen können.
Beachten Sie dabei die Zugriffsrechte.

Mögliche Probleme beim Erzeugen einer Internetform:

Unter Umständen erhalten Sie folgende Fehlermeldung:

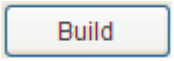


Der häufigste Fall ist, dass die Form gerade zum Bearbeiten geöffnet ist.

Eine weitere Fehlerquelle, dass man die Daten unter Umständen in einer 2ten VFP9 Umgebung exklusiv geöffnet hat.

Und unter Umständen kann es vorkommen, dass sie zuerst die Applikation starten müssen und unter Menüpunkt Extras/Benutzerverwaltung „Einstellungen löschen“ klicken müssen.

Wie arbeitet der VFX AFX Wizard

Sobald im Wizard auf 

geklickt wird, wird die vorher ausgewählte Maske mit der kompletten Applikation und Ihren Daten geöffnet.

Diese „lebende“ Form wird nun analysiert und die Daten werden in einem internen Cursor gespeichert.

Auch für jedes Grid wird ein extra Cursor intern angelegt.

Sobald die Daten gesammelt wurden, wird die Applikation bzw. die Form wieder geschlossen.

Nun greift der Wizard auf die VFXAFXMETA.DBF zurück und verarbeitet jeden Datensatz des zuvor angelegten Cursors.

Ausschlaggebend ist der Klassenname, welcher herangezogen wird, um in der vfxafxmeta nach dem richtigen Datensatz zu suchen.

Wird der Klassenname gefunden, und ist der hinterlegte Code ***nicht*** leer, wird der Inhalt von cmemo mittels Textmerge Befehl verarbeitet und im HTML-Code eingefügt.

Wenn der Klassenname nicht gefunden wird, wird er automatisch in der vfxafxmeta.dbf angelegt und der Datensatz der Basisklasse wird gesucht. Nun wird dieser anstelle des Klassennamens verwendet.

Sie können diese Art der „Vererbung“ aber auch absichtlich unterbrechen, was z.B. im *ctoolbarbutton* gemacht wird. Für die Speedbar, bzw. die normale Toolbar in den VFX-Formularen wird eine extra Klasse verwendet und somit auch ein extra Code, welcher im HTML eingefügt wird. Jeder Toolbarbutton hat den Klassennamen „*ctoolbarbutton*“. Da in einer Form, welche mit der Speedbar ausgestattet ist, auch der Toolbarbutton vorkommt und dieser auf der Basisklasse „textbox“ basiert, würde im erzeugten HTML eine Reihe von Buttons erscheinen, weil

ctoolbarbutton nicht verwendet wird und die Basisklasse „textbox“ anstelle dessen im HTML eingefügt wird.

Dies wird aber explizit verhindert, indem in cmemo von „ctoolbarbutton“ der Text
<!-- disabled-->
eingefügt ist. Sie sehen, dass es sich um einen HTML Kommentar handelt. Dieser wird nun anstelle des Codes für die Basisklasse „textbox“ eingefügt und es sind keine Buttons mehr im HTML zu sehen.

HTML funktioniert immer mit einem Öffnendem und einem Schließendem „TAG“. Also jedes muss mit einem wieder geschlossen werden.
Deshalb gibt es für jeden Klassen-Datensatz und Basisklassen-Datensatz ein Pendant zum schließen. Beispiel textbox -> textbox_end
Sinn macht es bei Pageframe, Page, cdataformpage also allen Containerobjekten. Jeder Container beinhaltet mehrere andere Objekte und muss deshalb am Ende wieder geschlossen werden. Beispiel Pageframe_end, welches nur ein beinhaltet.

Die Variablen mit den Daten der Form

Wie bereits beschrieben, wird intern ein Cursor gehalten, welcher alle notwendigen Daten des originalen Formulars beinhaltet.

Diese Daten sind direkt als Variablen ansprechbar und müssen in doppelten Spitzen Klammern eingefügt werden.

Beispiel Optiongroup:

```
<div id="div_<<cname>>_<<nlfd>>" style="position: absolute; border-style:solid
;border-width:0px ;
left:<<nleft>>; top:<<ntop>> ; height: <<nheight>> ; width: <<nwidth>> ;z-
index:<<nlevel>>">
```

cname, nlfd, nleft, ntop, nheight, nwidth und nlevel sind Variablen, aus dem erwähnten Cursor und werden direkt als Werte im erzeugten HTML eingefügt.

Am Beispiel der textbox sehen Sie das auch komplexe Ausdrücke eingefügt werden können:

```
size="<<int(IIF(LEN(cinputmask)>0,LEN(cinputmask),nwidth/FONTMETRIC(6,cfont,nf
ontsize)))>>"
```

Unter Umständen ist es jedoch sinnvoller kompletten VFP Code auszuführen. Dafür gibt es in der vfxafxmeta.dbf ein Flag mit dem Namen lcode. Ist dies auf .T. wird der Inhalt von cmemo temporär compiliert und ausgeführt.

Ein Beispiel dafür finden Sie unter Pagescript indem der Javascriptcode erzeugt wird.

```
local lcs
lcs=[<script>]+chr(13)+chr(10)
lcs=lcs+[function activate]+alltrim(conload)+[()] +chr(13)+chr(10)
lcs=lcs+[{]+chr(13)+chr(10)
for i=1 to nmaxcount
   if i=nlfd
lcs=lcs+[Page]+trans(nlevel)+[_]+trans(i)+[.style.visibility="visible";]+chr(1
3)+chr(10)
lcs=lcs+[changetab(Tabspan]+trans(nlevel)+[_]+trans(i)+[,"vfxafximage/tab-
active.png");]+chr(13)+chr(10)
lcs=lcs+[Tabspan]+trans(nlevel)+[_]+trans(i)+[.onmouseout='changetab(Tabspan]+
trans(nlevel)+[_]+trans(i)+[,"/vfxafximage/tab-active.png");']+chr(13)+chr(10)
   else
lcs=lcs+[Page]+trans(nlevel)+[_]+trans(i)+[.style.visibility="hidden";]+chr(13
)+chr(10)
lcs=lcs+[changetab(Tabspan]+trans(nlevel)+[_]+trans(i)+[,"vfxafximage/tab.png"
);]+chr(13)+chr(10)
lcs=lcs+[Tabspan]+trans(nlevel)+[_]+trans(i)+[.onmouseout='changetab(Tabspan]+
trans(nlevel)+[_]+trans(i)+[,"/vfxafximage/tab.png");']+chr(13)+chr(10)
   endif
endfor
lcs=lcs+[};]+chr(13)+chr(10)
lcs=lcs+[</script>]+chr(13)+chr(10)
return lcs
```

Sollte ein Fehler auftreten, wenn solch ein Code ausgeführt wird, wird der Fehler als HTML-Kommentar im HTML gespeichert.

Wenn Sie also Quellcode selbst verändern oder erstellen, so überprüfen Sie immer das erzeugte Ergebnis.

Die Laufzeittabellen

Der Cursorname der VFXAFXMETA.DBF zur Laufzeit lautet htmlbuildx

Ckey c(40)	Der Klassenname
Cdesc c(80)	Beschreibung
Cmemo M	Inhalt als HTML oder als Code
Lparam l	.T. wenn Steuerdaten für den Wizard
Lcode l	Bei .T. wird es als VFP-Code compiliert
Nvers n(5,2)	Versionsnummer bei 99.99 schreibgeschützt

In der Tabelle sind auch noch Steuerdaten für den Wizard selbst abgelegt. Diese Steuerdaten sind mit lparam = .T. gekennzeichnet und lauten:

Extension	.AFX oder .AFP wird direkt in der Wizardmaske gesetzt
Outputpath	Ausgabepfad der erzeugten Dateien
Postfix	Namenserweiterung nach dem Dateinamen
Postfixexec	Namenserweiterung des Ausführenden Codes “_EXEC”
Postfixproc	Namenserweiterung der Proceduredatei “_PROC”
Prefix	Vorangestellte Zeichenkette, vor jeder Datei “VFX_”

Der Cursorname der Form zur Laufzeit lautet htmltemp
Diese Tabelle sollte nur innerhalb eines lcode=.T. verwendet werden. Alle notwendigen Felder werden zusätzlich in Public Variablen hinterlegt. Siehe weiter unten.

level i	Der Level für DHTML Ebenen
name c(220)	Der Name der Klasse, wie er vom Wizard erzeugt wird. Dieser Name beinhaltet immer auch alle Parentnamen.
namesort c(220)	Ein Sortierfeld, nach dem dann abgearbeitet wird.
baseclass c(20)	Die Basisklasse jeden Objektes
class c(20)	Der Klassenname jeden Objektes
parent m	Der Parent jedes Objektes
caption c(100)	
left i	
top i	
width i	
height i	
font c(30)	
fontsize i	
forecolor i	
backcolor i	
alignment i	
value m	Wird unter Umständen anders verwendet. Cursor: value beinhaltet den Alias Cursoradapter: value beinhaltet den Alias Grid: value beinhaltet controlsource der Column[x]
csource m	Controlsource Grid: csource beinhaltet recordsource Textbox: ist thisform als text vorhanden, wird es in g_thisform_ gewandelt. Notwendig für Viewparameter
backstyle i	
lfd i	Page: lfd enthält Pageorder Onload wird ebenfalls damit gefüllt
maxcount i	Page: beinhaltet Pagecount
inputmask m	
tabs l	Pageframe.tabs
visible l	
onload c(40)	Scriptcode: "Page_"+TRANSFORM(nlevel)+"_"+TRANSFORM(lfd)
tablen i	20+(5*LEN(ALLTRIM(caption)))
tableft i	Addierte Tablen
speedbar l	.T. wenn speedbar in der Form verwendet
RowSource m	
RowSourceType i	
ColumnCount i	
BoundColumn i	
tabindex i	
pageframeindex i	Fortlaufende Nummer der vorhandenen Pageframes
valid M	Inhalt aus afx_valid
gotfocus M	Inhalt aus afx_gotfocus
lostfocus M	Inhalt aus afx_lostfocus
keypress M	Inhalt aus afx_keypress
click M	Inhalt aus afx_click
dblclick M	Inhalt aus afx_dblclick
tooltiptext M	
statusbartext M	
user M	Noch nicht verwendet
cviewparameter c(100)	

Es werden Public Variablen zur Verfügung gestellt, welche die Daten des jeweiligen Datensatzes beinhalten. Diese sollten verwendet werden, anstelle des Feldes aus dem Cursor.
Es wird als Beispiel bei cforecolor der originalwert automatisch in das HTML-pendant umgewandelt.

ckey	ALLTRIM(htmlbuildx.ckey)
cdesc	ALLTRIM(htmlbuildx.cdesc)
nlevel	htmltemp.level
cname	ALLTRIM(htmltemp.name)
cbaseclass	ALLTRIM(htmltemp.baseclass)
ccaption	ALLTRIM(htmltemp.caption)
nleft	htmltemp.left
ntop	htmltemp.top
nwidth	htmltemp.width
nheight	htmltemp.height
cfont	ALLTRIM(htmltemp.font)
nfontsize	htmltemp.fontsize
nforecolor	htmltemp.forecolor
nbackcolor	htmltemp.backcolor
cforecolor	ALLTRIM(ohtmlbuilder.htmlcolor(htmltemp.forecolor))
cbackcolor	ALLTRIM(ohtmlbuilder.htmlcolor(htmltemp.backcolor))
nalignment	htmltemp.alignment
cvalue	ALLTRIM(htmltemp.value)
ccontrolsource	ALLTRIM(htmltemp.csource)
Cvalid	htmltemp.valid
Cgotfocus	htmltemp.gotfocus
Clostfocus	htmltemp.lostfocus
Cclick	htmltemp.click
Cdblclick	htmltemp.dblclick
Ckeypress	htmltemp.keypress
cRowSource	htmltemp.rowsource
nRowSourceType	htmltemp.rowsourcetype
nColumnCount	htmltemp.columncount
nBoundColumn	htmltemp.boundcolumn
cstatusbartext	htmltemp.statusbartext
Ctooltiptext	htmltemp.tooltiptext
nbackstyle	htmltemp.backstyle
nlfd	htmltemp.lfd
nmaxcount	htmltemp.maxcount
cinputmask	htmltemp.inputmask
ltabs	htmltemp.tabs
lvisible	htmltemp.visible
conload	htmltemp.onload
ntablen	htmltemp.tablen
ntableft	htmltemp.tableft
lspeedbar	htmltemp.speedbar
nRowSource	htmltemp.rowsource
nRowSourceType	htmltemp.rowsourcetype
ntabindex	htmltemp.tabindex
npageframeindex	htmltemp.pageframeindex

Der Cursorname eines Grids zur Laufzeit lautet htmltempgrid<n> n ist die fortlaufende Nummer.

feldnr i	Fortlaufende Nummer
caption c(30)	Überschrift für die Gridcolumn
width i	Breite
csource c(50)	Controlsource
crecordsource c(50)	Recordsource

Es werden außerdem noch folgende Felder als Variablen angelegt, wodurch der Cursor des Grids nicht unbedingtverwendet werden muss.

ngridfeldnr	feldnr
cgridcaption	caption
ngridwidth	width
cgridcsource	ALLTRIM(csource)
Cgridrecordsource	ALLTRIM(crecordsource)
Cgridshortcsource	STRTRAN(ALLTRIM(lower(csource)),ALLTRIM(LOWER(crecordsource))+".","")

Der Cursorname des goprogram-Objektes der Applikation lautet goprogdata

cmaindatabase c(100)	Name der Hauptdatenbank
cdatasourcetype c(100)	Datenquellentyp (Native, ODBC usw.)
clangid c(100)	Sprachid
cmaintitle c(100)	Haupttitel der Applikation

Es werden vor dem Build-Lauf noch globale Variablen gefüllt, welche ebenfalls verwendet werden können. Diese Variablen sind teilweise direkt als Properties des Wizards ausgelegt.
Es gibt auch noch Methoden, welche direkt verwendet werden können:

Methoden

ohtmlbuilder.evalthis(tcwhat)	Evaluiert jeden Begriff oder jeden Wert. Handelt es sich um eine Zeichenkette, wird sie als ckey in htmlbuildx (vfxafxmeta.dbf) gesucht und der Inhalt von cmemo wird wiederum evaluiert. Dies wird sogar rekursiv vorgenommen.
ohtmlbuilder.htmlcolor(tncolor)	Umwandlung einer RGB-Zahl nach deren Internetentsprechung

Properties

ohtmlbuilder.cappname	VFX-Projekt Verzeichnis
ohtmlbuilder.cappdir	Pfad zur VFXAFXMeta.dbf
ohtmlbuilder.cappfullname	Pfad zum VFX-Projekt
Outputpath	Ausgabepfad
Extension	Gewählte Erweiterung .AFX oder .AFP
Filename	Dateiname des Formulars

Der Aufbau der erzeugten Dateien

Sobald eine Form mit dem VFX AFX Wizard umgewandelt wird, werden automatisch die folgenden Verzeichnisse und Dateien angelegt. Achtung. Abhängig der Auswahl ob AFX oder AFP erzeugt werden soll, werden dementsprechend die Dateiendung und die Links angepasst.

Das Verzeichnis LIB beinhaltet alle Libraries für die Internetapplikation

```
Lib
Lib \ afx.dll
Lib \ vfx.fll
Lib \ vfxafx.vcx
Lib \ vfxafx.vct
```

Im Verzeichnis Include sind die Headerdateien untergebracht. Es werden alle VFX Headerdateien mitgeliefert, obwohl nicht alle benutzt werden.

```
Lib \ Include
Lib \ Include \ FOXPRO.H
Lib \ Include \ FOXPRO_REPORTING.H
Lib \ Include \ REPORTLISTENERS.H
Lib \ Include \ REPORTLISTENERS_LOCS.H
Lib \ Include \ USERDEF.H
Lib \ Include \ USERMSG.H
Lib \ Include \ USERTXT.H
Lib \ Include \ VFX.H
Lib \ Include \ VFXDEF.H
Lib \ Include \ VFXGLOBAL.H
Lib \ Include \ VFXMSG.H
Lib \ Include \ VFXOFFCE.H
Lib \ Include \ VFXTOOLBOX.H
Lib \ Include \ VFXTXT.H
Lib \ Include \ _FRXCURSOR.H
```

Im Verzeichnis Program liegt die vfxfunc.prg aus dem VFX95 Projekt.

```
Program
Program \ vfxfunc.prg
Program \ vfcfunc.fxp
```

Die Images findet man unter

```
Vfxafximage
```

Das Grid ist mit einem Javascript Bestandteil ausgestattet. Dieser ist hier untergebracht

```
Vfxafxjs \ grid.js
```

Die Stylesheets

```
Vfxafxstyle \ basic.css
Vfxafxstyle \ grid.css
```

Wird mit Cursoradapter gearbeitet wird die config.vfx in die Datei config.afx umgewandelt. Zum Bearbeiten dieser config.afx können sie das mitgelieferte Programm afxconfig.exe verwenden. Diese Datei wird angelegt, wenn Sie nicht vorhanden ist. Sie wird nicht bei jeder Maske neu erzeugt.

```
config.afx
```

Die Applikations Datei. Diese Datei wird angelegt, wenn Sie nicht vorhanden ist. Sie wird nicht bei jeder Maske neu erzeugt.

```
<app>.AFPA
```

Die Applikations-Code Datei, welche in allen Formularen hinein kompiliert wird. Wichtig sind der Pfad und der Name. Diese Datei wird angelegt, wenn Sie nicht vorhanden ist. Sie wird nicht bei jeder Maske neu erzeugt.

```
<app>.AFPA.CODE
```

Die Include-Datei der Applikation. Hier ist die Klassendefinition des Cursoradapters enthalten. Diese Datei wird angelegt, wenn Sie nicht vorhanden ist. Sie wird nicht bei jeder Maske neu erzeugt.

```
<app>.AFPI
```

Die Loginmaske und die dazu gehörige Login Validierung. Diese Dateien werden aus dem Wizfiles Verzeichnis kopiert.

```
LOGIN.AFP
LOGINVALID.AFP
LOGINVALID.AFP.CODE
```

Die Oberfläche und das Menü für die Applikation. Es wird direkt eine vfxfopen.dbf ausgelesen. Das Feld Inetlevel wird dabei berücksichtigt.

```
XPOPEN.AFP
XPOPEN.AFP.CODE
XPOPENBOTTOM.HTM
XPOPENDIR.AFP
XPOPENDIR.AFP.CODE
XPOPENMAIN.HTM
XPOPENTOP.HTM
```

Die eigentliche Form besteht aus mehren Dateien:

Das eigentliche Formular. Die einzelnen Seiten der Pageframe werden mittels Javascript umgeschalten. Die Grids werden in einem IFrame dargestellt.

```
vfx_<form>.AFP
```

Der Codeteil des Formulars mit Dateiöffnungsroutinen und Filtersetzungen

```
vfx_<form>.AFP.CODE
```

Eventuelle Definitionen von Cursoradaptern werden hier abgelegt

```
vfx_<form>.AFPI
```

Der Execute Teil des Formulares. Sobald ein Button geklickt wird, wird hier auf die einzelnen Aktionen reagiert. Unter Umständen wird auf andere Seiten weiter verzweigt, wie z.B. beim Filter

```
vfx_<form>_EXEC.AFP
```

Der Codeteil des Execute Teils mit Dateiöffnungsroutinen und Filtersetzungen

```
vfx_<form>_EXEC.AFP.CODE
```

Der Filterdialog

```
vfx_<form>_filter.AFP
```

Dateiöffnungsroutinen für den Filterdialog

```
vfx_<form>_filter.AFP.CODE
```

Der Execute Teil des Filters mit Weiterleitung zur original Maske

```
vfx_<form>_filter_exec.AFP
```

Die Gridmaske

```
vfx_<form>_grid<lfd>.AFP
```

Dateiöffnungsroutinen für die Gridmaske

```
vfx_<form>_grid<lfd>.AFP.CODE
```

Die Procedure Datei, in der die AJAX-Codeteile abgearbeitet werden

```
vfx_<form>_PROC.AFP
```

Dateiöffnungsroutinen für die Procedure Datei

```
vfx_<form>_PROC.AFP.CODE
```

AJAX

Ist die Abkürzung für **Asynchrones Java and XML.**
Es bedeutet nichts anderes, als das Asynchron mit Hilfe von Java und XML Daten übertragen werden.

In AFX wird der Code automatisch erzeugt, sobald eine der folgenden Methoden in einer Klasse gefunden werden:

AFX_GotFocus
AFX_LostFocus
AFX_KeyPress
AFX_Valid

In der HTML-Maske werden daraufhin diese Klassen in Javascript-Code umgewandelt.

“AFX_Valid” wird zu “onChange”
“AFX_Keypress” wird zu “onKeypress“
“AFX_Gotfocus” wird zu “onFocus“
“AFX_Lostfocus” wird zu “onBlur”

Die dazugehörigen Scriptteile sind in der VFXAFXMeta.dbf unter den Namen „KeypressCode“, „GotfocusCode“, „LostfocusCode“ und „ValidCode“ zu finden.

Es wird z.B. beim ValidCode folgendes Script eingefügt:

```
<script type="text/javascript">
function id_<<cname>>_Valid(Feld) {
   var DataToSend = "controlfield=id_<<cname>>_Valid&"
   DataToSend = DataToSend + "value="+Feld+"&"
   DataToSend = DataToSend + "recno=<%?recno()%>&"
   DataToSend = DataToSend + "alias=<%?alias()%>&"
   DataToSend = DataToSend + "controlsource=<<ccontrolsource>>"
   var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP")
   xmlhttp.Open("POST","<<filename>>_PROC<<extension>>",false)
   xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded")
   xmlhttp.send(DataToSend)
   var xmldoc = new ActiveXObject("Microsoft.XMLDOM")
   xmldoc.async="false"
   xmldoc.loadXML(xmlhttp.responsetext)
   cfooter.innerText
=xmldoc.getElementsByTagName("statustext").item(0).text
   if(xmldoc.getElementsByTagName("error").item(0).text== "1")
   {
      if (xmldoc.getElementsByTagName("message").item(0).text != "")
      {
         alert(xmldoc.getElementsByTagName("message").item(0).text)
      }
      document.<<filename>>.id_<<cname>>.focus()
   }else
   {
      if (xmldoc.getElementsByTagName("message").item(0).text != "")
      {
         alert(xmldoc.getElementsByTagName("message").item(0).text)
      }
```

```
      document.<<filename>>.id_<<cname>>.value =
xmldoc.getElementsByTagName("value").item(0).text
   }

}
</script>
```

Es wird hier automatisch für jedes Objekt, welches eine der AFX-Methoden in sich trägt ein Aufruf der „PROC-Datei“ vorbereitet.

Das bedeutet, dass bei jedem Event, welcher im Javascript erkannt wird und für den ein Scriptteil vorhanden ist, die PROC-Datei abgearbeitet wird. Und zwar nicht am Client sondern durch den Aufruf von

```
xmlhttp.Open("POST","<<filename>>_PROC<<extension>>",false)
xmlhttp.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded")
xmlhttp.send(DataToSend)
```

direkt am Server. Es wird also beim Keypress-Event bei jedem Tastendruck eine AFX oder AFP-Seite abgearbeitet.

Diese AFX / AFP Seite bekommt Controlfield, value, recno(), alias() und controlsource mitgeliefert und man kann nun darauf reagieren.

Innerhalb der PROC-Datei wird z.B.

```
lreturn = id_<form><Feld_ID>_Lostfocus(calias, nrecno, ccontrolsource,
ccontrolfield, @cvalue, @cstatustext)
```

eingetragen. In der dazugehörigen PROC.AFP.CODE Datei wird die dazugehörige procedure hinterlegt. Beispiel:

```
PROCEDURE id_frm<form><Feld_ID>_Lostfocus
LPARAMETERS calias, nrecno, ccontrolsource, ccontrolfield, cvalue, cstatustext
```

Die LPARAMETER Anweisung wird automatisch eingefügt. Wenn in der AFX_Lostfocus Methode des Objektes ein Code eingetragen war, wird dieser automatisch hier eingefügt.

Die Variablen cValue und cStatustext werden per Referenz übergeben, so können Sie sowohl den Wert verändern, als auch den Statustext, welcher auf jeder Seite automatisch eingetragen wird.

Anwendungsbeispiele dafür wären z.B. eine Überprüfung der cValue im Valid mit Rückgabe einer Fehlermeldung, welche als Alert() im Browser angezeigt wird.

Die Art der Rückgabe bzw. eine eventuelle Fehlermeldung sind von verschiedenen Kriterien abhängig. Es gibt eine globale Fehlermeldungsvariable goform.cpendingmessage welche als Javascript Alert( ) angezeigt wird. Alle Daten werden im XML-Format übertragen.

Es wird nach folgender Logik vorgegangen:

```
IF TYPE( lreturn ) = “L“
   IF lreturn = .t.
               IF TYPE("goform.cpendingmessage")="C" AND ;
                     EMPTY(goform.cpendingmessage)=.f.

          Rückgabe ohne Fehlerkennzeichen (ERROR=0)
          Message = goform.cpendingmessage
       ELSE
          Rückgabe ohne Fehlerkennzeichen (ERROR=0)
                     Message = “”
       ENDIF
   ELSE
               IF TYPE("goform.cpendingmessage")="C" AND ;
                     EMPTY(goform.cpendingmessage)=.f.

          Rückgabe mit Fehlerkennzeichen (ERROR=1)
                     Message = goform.cpendingmessage
       ELSE
          Rückgabe mit Fehlerkennzeichen (ERROR=1)
                     Message = “”
       ENDIF
   ENDIF
ELSE
   Rückgabe ohne Fehlerkennzeichen (ERROR=0)
   Message = lreturn
ENDIF
```

Im Javascript wird mit

```
var xmldoc = new ActiveXObject("Microsoft.XMLDOM")
   xmldoc.async="false"
xmldoc.loadXML(xmlhttp.responsetext)
```

das von der PROC-datei erzeugte XML intern aufbereitet.

Zuerst wird der Statuszeilentext mit

```
cfooter.innerText
      =xmldoc.getElementsByTagName("statustext").item(0).text
```

zugewiesen.

Dann wird aufgrund des Fehlerkennzeichens entschieden ob es sich um einen Fehler handelt und der Text wird mit Alert( ) ausgegeben. Danach wird auf das Feld zurück fokusiert.

Ist kein Fehler aufgetreten, wird geprüft ob eine Meldung ausgegeben werden soll und die Value des Feldes wird ersetzt.

```
if(xmldoc.getElementsByTagName("error").item(0).text== "1")
{
   if (xmldoc.getElementsByTagName("message").item(0).text != "")
   {
   alert(xmldoc.getElementsByTagName("message").item(0).text)
   }
   document.frm<form><Feld_ID>.focus()
}
else
{
   if (xmldoc.getElementsByTagName("message").item(0).text != "")
   {
   alert(xmldoc.getElementsByTagName("message").item(0).text)
   }
   document.vfx frm<form><Feld_ID>.value =
            xmldoc.getElementsByTagName("value").item(0).text
}
```

Anhang II - Transact-SQL

von Igor Nikiforov

Die folgenden User-Defined Transact-SQL Zeichenfolgenfunktionen wurden freundlicherweise von Igor Nikiforov zur Verfügung gestellt und werden mit VFX geliefert.

AT()

Gibt die numerische Anfangsposition zurück, an der ein Zeichenausdruck zum ersten Mal in einem anderen Zeichenausdruck vorkommt, und zwar vom äußersten linken Zeichen aus gerechnet.

Syntax

AT(@cSearchExpression, @cExpressionSearched [, @nOccurrence])

Parameter

@cSearchExpression - Gibt den Zeichenausdruck an, nach dem AT() in @cExpressionSearched sucht.

@cExpressionSearched - Gibt den Zeichenausdruck an, in dem mit @cSearchExpression gesucht wird. Sowohl @cSearchExpression als auch @cExpressionSearched können von beliebiger Größe sein.

@nOccurrence - Gibt an, nach dem wie vielten Vorkommen (ersten, zweiten, dritten usw.) von @cSearchExpression in @cExpressionSearched gesucht werden soll. Standardmäßig sucht AT() nach dem ersten Vorkommen von @cSearchExpression (@nOccurrence = 1). Durch Angabe von @nOccurrence können Sie weitere Vorkommen von @cSearchExpression in @cExpressionSearched suchen. Wenn @nOccurrence größer ist als die Anzahl der Vorkommen von @cSearchExpression in @cExpressionSearched, gibt AT() den Wert 0 zurück.

Rückgabewert

Smallint

Hinweise

AT() sucht im zweiten Zeichenausdruck nach dem ersten Vorkommen des ersten Zeichenausdrucks. Ist die Suche erfolgreich, gibt AT() eine ganze Zahl zurück, die die Position des ersten Zeichens des gefundenen Zeichenausdrucks angibt. Ist die Suche nicht erfolgreich, gibt AT() den Wert 0 zurück.

Die mit AT() ausgeführte Suche berücksichtigt Groß- und Kleinschreibung. Wenn Sie einen Suchvorgang ausführen möchten, bei dem die Groß-/Kleinschreibung nicht berücksichtigt wird, verwenden Sie die ATC()-Funktion.

Ähnlich zu der bekannten Oracle-Funktion INSTR.

Siehe auch RAT().

Beispiel

```
declare @gcString nvarchar(4000), @gcFindString nvarchar(4000)
select @gcString = N'Johann Wolfgang von Goethe (1749-1832)', @gcFindString = 'von'
select dbo.AT(@gcFindString, @gcString, default) -- Anzeige 17
set @gcFindString = 'VON'
select dbo.AT(@gcFindString, @gcString, default) -- Anzeige 0, case-sensitive
```

ATC()

Gibt die numerische Anfangsposition des ersten Auftretens eines Zeichenausdrucks innerhalb eines anderen Zeichenausdrucks zurück, ohne die Groß-/Kleinschreibung dieser beiden Ausdrücke zu berücksichtigen.

Syntax

ATC(@cSearchExpression, @cExpressionSearched [, @nOccurrence])

Parameter

@cSearchExpression - Gibt den Zeichenausdruck an, nach dem ATC() in @cExpressionSearched sucht. Der Ausdruck kann von beliebiger Größe sein.

@cExpressionSearched - Gibt den Zeichenausdruck an, in dem mit @cSearchExpression gesucht wird. Der Ausdruck kann von beliebiger Größe sein.

@nOccurrence - Gibt an, nach dem wie vielten Vorkommen (ersten, zweiten, dritten usw.) von @cSearchExpression in @cExpressionSearched gesucht werden soll. Standardmäßig sucht ATC() nach dem ersten Vorkommen von @cSearchExpression (@nOccurrence = 1). Durch Angabe von @nOccurrence können Sie weitere Vorkommen von @cSearchExpression in @cExpressionSearched suchen.

Rückgabewert

Smallint

Hinweise

ATC() sucht im zweiten Zeichenausdruck nach dem ersten Zeichenausdruck, ohne dabei für die beiden Ausdrücke die Groß-/Kleinschreibung (Groß- oder Kleinbuchstaben) zu berücksichtigen. Soll bei einem Suchvorgang die Groß-/Kleinschreibung berücksichtigt werden, verwenden Sie die AT()-Funktion.

ATC() gibt eine ganze Zahl zurück, die die Position angibt, an der das erste Zeichen des gesuchten Zeichenausdrucks gefunden wurde. Wird der jeweilige Zeichenausdruck nicht gefunden, gibt ATC() den Wert 0 zurück.

Siehe auch AT(), RAT().

Beispiel

```
declare @gcString nvarchar(4000), @gcFindString nvarchar(4000)
select @gcString = N'Johann Wolfgang von Goethe (1749-1832)', @gcFindString = 'VON'
select dbo.ATC(@gcFindString, @gcString, default) -- Anzeige 17, case-insensitive
```

RAT()

Gibt für eine Zeichenfolge die numerische Position zurück, ab der der Ausdruck das letzte Mal (äußerst rechts) in einer anderen Zeichenfolge vorkommt.

Syntax

RAT(@cSearchExpression, @cExpressionSearched [, @nOccurrence])

Parameter

@cSearchExpression - Gibt den Zeichenausdruck an, nach dem RAT() in @cExpressionSearched sucht. Der Ausdruck kann von beliebiger Größe sein.

@cExpressionSearched - Gibt den Zeichenausdruck an, den RAT() durchsucht. Der Ausdruck kann von beliebiger Größe sein.

@nOccurrence - Gibt an, nach welchem Vorkommen (von links nach rechts) von @cSearchExpression RAT() in @cExpressionSearched sucht. Standardmäßig sucht RAT() nach dem letzten Vorkommen von @cSearchExpression (@nOccurrence = 1). Wenn @nOccurrence gleich 2 ist, sucht RAT() nach dem vorletzten Vorkommen usw.

Rückgabewert

Smallint

Hinweise

RAT(), die Umkehrfunktion zu AT(), durchsucht den Zeichenausdruck in @cExpressionSearched von rechts nach links nach dem letzten Auftreten der in @cSearchExpression angegebenen Zeichenfolge.

RAT() gibt eine ganze Zahl zurück, die die Position des ersten Zeichens von @cSearchExpression in @cExpressionSearched angibt. RAT() gibt 0 zurück, wenn @cSearchExpression nicht in @cExpressionSearched gefunden wird oder wenn @nOccurrence größer ist als die Anzahl des Auftretens von @cSearchExpression in @cExpressionSearched.

Die mit RAT() ausgeführte Suche berücksichtigt Groß- und Kleinschreibung.

Siehe auch AT(), ATC().

Beispiel

```
declare @gcString nvarchar(4000), @gcFindString nvarchar(4000)
select @gcString = N'"Alles Vergängliche / Ist nur ein Gleichnis // Das Unzulängliche, // Hier wirds Ereignis; // Das Unbeschreibliche, // Hier ist es getan; // Das Ewig- Weibliche // Zieht uns hinan. " - Faust II, Vers 12104ff, Chorus mysticus ', @gcFindString = 'Das'
select dbo.RAT(@gcFindString, @gcString, 2) -- Anzeige 94, case-sensitive
```

OCCURS(), OCCURS2()

Gibt den Wert zurück, wie oft ein Zeichenausdruck in einem anderen Zeichenausdruck vorkommt.

Syntax

OCCURS(@cSearchExpression, @cExpressionSearched)
OCCURS2(@cSearchExpression, @cExpressionSearched)

Parameter

@cSearchExpression - Gibt einen Zeichenausdruck an, den OCCURS() in @cExpressionSearched sucht.

@cExpressionSearched - Gibt den Zeichenausdruck an, in dem OCCURS() nach @cSearchExpression sucht.

Rückgabewert

Smallint

Hinweise

OCCURS() gibt 0 (Null) zurück, wenn @cSearchExpression nicht in @cExpressionSearched gefunden wird.

OCCURS(): einschließlich Deckungen.
select dbo.OCCURS('ABCA', 'ABCABCABCA') -- Anzeige 3
1 Auftreten 'ABCA .. BCABCA'
2 Auftreten 'ABC...ABCA...BCA'
3 Auftreten 'ABCABC...ABCA'
OCCURS2(): ausschließlich der Deckungen.
select dbo.OCCURS2('ABCA', 'ABCABCABCA') -- Anzeige 2
1 Auftreten 'ABCA .. BCABCA'
2 Auftreten 'ABCABC... ABCA'

Siehe auch AT(), RAT(), OCCURS2()

Beispiel 1

```
declare @gcString nvarchar(4000)
select @gcString = "'Blut ist ein ganz besondrer Saft." - Faust I, Vers 1740, Mephistopheles '
select dbo.OCCURS('a', @gcString ) -- Anzeige 3
select dbo.OCCURS('b', @gcString ) -- Anzeige 1
```

Beispiel 2

Zählt das Auftreten verschiedener Buchstaben aus der Zeichenkette @gcCaracters in der Zeichenkette @gcString.

```
declare @gcString nvarchar(4000), @gcCaracters nvarchar(256), @i smallint, @counter smallint
select @i = 1, @counter = 0
select @gcString = N'Den Teufel spürt das Völkchen nie, und wenn er sie beim Kragen hätte.',
@gcCaracters = N'abccaü'
while @i <= datalength(@gcCaracters)/2
begin
if charindex(substring(@gcCaracters,@i,1), left(@gcCaracters, @i - 1)) = 0
select @counter = @counter + dbo.OCCURS2(substring(@gcCaracters,@i,1), @gcString)
select @i = @i + 1
end
select @counter -- Anzeige 5
```

PADL(), PADR(), PADC()

Gibt aus einem Ausdruck eine Zeichenfolge zurück, die links, rechts oder auf beiden Seiten bis zu einer angegebenen Länge mit Leerzeichen oder Zeichen aufgefüllt ist.

Syntax

PADL(@eExpression, @nResultSize [, @cPadCharacter])
PADR(@eExpression, @nResultSize [, @cPadCharacter])
PADC(@eExpression, @nResultSize [, @cPadCharacter])

Parameter

@eExpression - Gibt den aufzufüllenden Ausdruck an. Bei diesem Ausdruck kann es sich um jeden Ausdruckstyp mit Ausnahme eines logischen Ausdrucks bzw. einer Währung, eines Objekt- oder eines Bildfeldes handeln.

@nResultSize - Gibt die Gesamtzahl der Zeichen im Ausdruck nach dem Auffüllen an.

@cPadCharacter - Gibt den Wert an, der zum Auffüllen verwendet werden soll. Dieser Wert wird so oft wiederholt, bis der Ausdruck auf die angegebene Anzahl an Zeichen aufgefüllt ist. Wenn Sie @cPadCharacter nicht angeben, werden zum Auffüllen Leerzeichen (ASCII-Zeichen 32) verwendet.

Rückgabewert

Nvarchar(4000)

Hinweise

Mit PADL() wird ein Ausdruck links, mit PADR() rechts und mit PADC() auf beiden Seiten aufgefüllt.

Beispiel

```
declare @gcString nvarchar(4000)
select @gcString = 'Mephistopheles'
select dbo.PADL(@gcString, 40, default) -- Anzeige '                    Mephistopheles'
select dbo.PADL(@gcString, 40, '+=+') --
Anzeigen'+=++=++=++=++=++=++=++=++=Mephistopheles'
select dbo.PADR(@gcString, 40, '=!!=') -- Anzeige 'Mephistopheles=!!!==!!!==!!!==!!!==!!!=='
select dbo.PADC(@gcString, 40, '=*=') – Anzeige
'=*==*==*==*==Mephistopheles=*==*==*==*=='
```

CHRTRAN()

Jedes Zeichen in einem Zeichenausdruck, das einem Zeichen in einem zweiten Zeichenausdruck entspricht, wird durch das entsprechende Zeichen eines dritten Zeichenausdrucks ersetzt.

Syntax

CHRTRAN(cSearchedExpression, @cSearchExpression, @cReplacementExpression)

Parameter

cSearchedExpression - Gibt den Ausdruck an, in dem CHRTRAN() Zeichen ersetzt.

@cSearchExpression - Gibt den Ausdruck mit den Zeichen an, nach denen CHRTRAN() in cSearchedExpression sucht.

@cReplacementExpression - Gibt den Ausdruck mit den Ersetzungszeichen an.

Rückgabewert

Nvarchar(4000)

Hinweise

Wird ein Zeichen aus @cSearchExpression in cSearchedExpression gefunden, wird es in cSearchedExpression durch das Zeichen in @cReplacementExpression ersetzt, dessen Position in @cReplacementExpression seiner Position in @cSearchExpression entspricht. Hat @cReplacementExpression weniger Zeichen als @cSearchExpression, werden die übrigen Zeichen

aus @cSearchExpression in cSearchedExpression gelöscht. Im umgekehrten Fall werden die überschüssigen Zeichen in @cReplacementExpression ignoriert.

CHRTRAN() übersetzt mit Hilfe der Übersetzungsausdrücke @cSearchExpression und @cReplacementExpression den Zeichenausdruck cSearchedExpression und gibt die sich ergebende Zeichenfolge zurück.

Siehe auch STRFILTER()

Beispiel

```
select dbo.CHRTRAN('ABCDEF', 'ACE', 'XYZ') -- Anzeige 'XBYDZF'
select dbo.CHRTRAN('ABCDEF', 'ACE', 'XYZQRST') -- Anzeige 'XBYDZF'
```

STRTRAN()

Durchsucht einen Zeichenausdruck nach dem Auftreten eines zweiten Zeichenausdrucks und ersetzt diesen jeweils durch einen dritten Zeichenausdruck.

Syntax

```
STRTRAN(@cSearched, @cExpressionSought [, @cReplacement]
 [, @nStartOccurrence] [, @nNumberOfOccurrences] [, @nFlags])
```

Parameter

@cSearched - Gibt den Zeichenausdruck an, der durchsucht wird.

@cExpressionSought - Gibt den Zeichenausdruck an, nach dem in @cSearched gesucht wird. Bei der Suche wird die Groß- und Kleinschreibung berücksichtigt.

@cReplacement - Gibt den Zeichenausdruck an, der cSearchFor bei jedem Auftreten in @cSearched ersetzt. Wenn Sie @cReplacement nicht angeben, wird @cExpressionSought bei jedem Auftreten durch eine leere Zeichenfolge ersetzt.

@nStartOccurrence - Gibt an, bei welchem Auftreten von @cExpressionSought die Ersetzung beginnen soll. Wenn Sie beispielsweise für @nStartOccurrence den Wert 4 angeben, beginnt das Ersetzen beim vierten Auftreten von @cExpressionSought in @cSearched. Die ersten drei aufgetretenen Ausdrücke werden nicht geändert. Ohne Angabe von @nStartOccurrence beginnt das Ersetzen standardmäßig beim ersten Auftreten von @cExpressionSought.

@nNumberOfOccurrences - Gibt an, wie oft @cExpressionSought ersetzt werden soll. Wenn Sie @nNumberOfOccurrences nicht angeben, wird @cExpressionSought bei jedem Auftreten ersetzt, beginnend mit dem in @nStartOccurrence angegebenen Auftreten.

@nFlags - Gibt an, ob bei der Suche die Groß-/Kleinschreibung berücksichtigt werden soll, und zwar entsprechend den Werten in der folgenden Liste: Wert für @nFlags.

- 0 (Standardwert) - Beim Suchen wird die Groß-/Kleinschreibung berücksichtigt, das Ersetzen findet mit dem exakten @cReplacement-Text statt.
- 1 - Beim Suchen wird die Groß-/Kleinschreibung nicht berücksichtigt, das Ersetzen findet mit dem exakten @cReplacement-Text statt.
- 2 - Beim Suchen wird die Groß-/Kleinschreibung berücksichtigt. Die Groß-/Kleinschreibung beim Parameter @cReplacement wird an die Groß-/Kleinschreibung beim Parameter @cExpressionSought angepasst, der ersetzt wird.
- 3 - Beim Suchen wird die Groß-/Kleinschreibung nicht berücksichtigt. Die Groß-/Kleinschreibung beim Parameter @cReplacement wird an die Groß-/Kleinschreibung beim Parameter @cExpressionSought angepasst, der ersetzt wird.

Rückgabewert

Nvarchar(4000)

Hinweise

Sie können angeben, wo die Ersetzung beginnen und wie oft diese durchgeführt werden soll. STRTRAN() gibt die Ergebniszeichenfolge zurück. Geben Sie den Wert -1 für optionale Parameter ein, die übersprungen werden sollen. Gleiches gilt, wenn Sie nur die Einstellung für @nFlags angeben wollen.

Siehe auch replace(), CHRTRAN()

Beispiel

```
select dbo.STRTRAN('ABCDEF', 'ABC', 'XYZ',-1,-1,0) -- Anzeige XYZDEF
select dbo.STRTRAN('ABCDEF', 'ABC', default,-1,-1,0) -- Anzeige DEF
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', default,2,-1,0) -- Anzeige
ABCDEFGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', default,2,-1,1) -- Anzeige
ABCDEFGHJQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 1, 1) -- Anzeige
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 3, 1) -- Anzeige
ABCDEFXYZGHJXYZQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 1, 2) -- Anzeige
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'XYZ', 2, 3, 2) -- Anzeige
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'xyZ', 2, 1, 2) -- Anzeige
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFABCGHJabcQWE', 'ABC', 'xYz', 2, 3, 2) -- Anzeige
ABCDEFXYZGHJabcQWE
select dbo.STRTRAN('ABCDEFAbcCGHJAbcQWE', 'Aab', 'xyZ', 2, 1, 2) -- Anzeige
ABCDEFAbcCGHJAbcQWE
select dbo.STRTRAN('abcDEFabcGHJabcQWE', 'abc', 'xYz', 2, 3, 2) -- Anzeige
abcDEFxyzGHJxyzQWE
select dbo.STRTRAN('ABCDEFAbcCGHJAbcQWE', 'Aab', 'xyZ', 2, 1, 3) -- Anzeige
ABCDEFAbcCGHJAbcQWE
select dbo.STRTRAN('ABCDEFAbcGHJabcQWE', 'abc', 'xYz', 1, 3, 3) -- Anzeige
XYZDEFXyzGHJxyzQWE
```

STRFILTER()

Entfernt alle Buchstaben aus einer Zeichenkette, ausgenommen den spezifizierten Zeichen.

Syntax

STRFILTER(@cExpressionSearched, @cSearchExpression)

Rückgabewert

Nvarchar(4000)

Parameter

@cExpressionSearched - Spezifiziert die Zeichenfolge, die durchsucht werden soll.
@cSearchExpression - Spezifiziert die Buchstaben, die in @cExpressionSearched erhalten bleiben sollen.

Hinweise

STRFILTER() entfernt alle Buchstaben von @cExpressionSearched, die nicht in @cSearchExpression enthalten sind.

Siehe auch CHRTRAN().

Beispiel

select dbo.STRFILTER('asdfghh5hh1jk6f3b7mn8m3m0m6','0123456789') -- Anzeige 516378306
select dbo.STRFILTER('ABCDABCDABCD', 'AB') -- Anzeige ABABAB

GETWORDCOUNT()

Zählt die Anzahl der Wörter in einer Zeichenfolge.

Syntax

GETWORDCOUNT(@cString[, @cDelimiters])

Parameter

@cString - Gibt die Zeichenfolge an, deren Wörter gezählt werden sollen.
@cDelimiters - Gibt ein oder mehrere Zeichen an, durch die Zeichengruppen in @cString getrennt werden sollen. Die Standardtrennzeichen sind Leerzeichen, Tabulator- und Wagenrücklaufzeichen. Beachten Sie, dass GETWORDCOUNT() jedes der Zeichen in @cDelimiters als Trennzeichen verwendet und nicht die ganze Zeichenkette als einzelnes Trennzeichen.

Rückgabewert

Smallint

Hinweise

GETWORDCOUNT() geht standardmäßig davon aus, dass Wörter durch Leerzeichen oder Tabstopps getrennt werden. Wenn Sie als Trennzeichen andere Zeichen angeben, ignoriert diese Funktion Leerzeichen und Tabstopps und verwendet nur die angegebenen Zeichen.

Siehe auch GETWORDNUM(), GETALLWORDS()

Beispiel

declare @cString nvarchar(4000)
set @cString = N'Werd ich zum Augenblicke sagen: Verweile doch! Du bist so schön! Dann magst du mich in Fesseln schlagen, dann will ich gern zugrunde gehn!'
-- Wenn Sie als Zielzeichenfolge für GETWORDCOUNT() @cString verwenden, erhalten Sie folgende Ergebnisse:
select dbo.GETWORDCOUNT(@cString, default) -- Anzeige 34 Wörter, getrennt duch Leerzeichen.
select dbo.GETWORDCOUNT(@cString, ',') -- Anzeige 2 Zeichnfolgen abgegrenzt mit ','.

GETWORDNUM()

Gibt ein angegebenes Wort aus einer Zeichenfolge zurück.

Syntax

GETWORDNUM(@cString, @nIndex[, @cDelimiters])

Parameter

@cString - Gibt die Zeichenfolge zurück, die ausgewertet werden soll.
@nIndex - Gibt die Indexposition des zurückzugebenden Worts an. Wenn beispielsweise @nIndex auf 3 gesetzt ist, gibt GETWORDNUM() das dritte Wort zurück (wenn @cString drei oder mehr Wörter enthält).
@cDelimiters - Gibt ein oder mehrere optionale Zeichen an, die verwendet werden, um die Wörter in @cString zu trennen. Die Standardtrennzeichen sind Leerzeichen, Tabulator- und Wagenrücklaufzeichen. Beachten Sie, dass GETWORDCOUNT() jedes der Zeichen in @cDelimiters als Trennzeichen verwendet und nicht die ganze Zeichenkette als einzelnes Trennzeichen.

Rückgabewert

Nvarchar(4000)

Hinweise

Gibt das Wort an der Position zurück, die von @nIndex in der Zielzeichenfolge @cString angegeben wurde. Wenn @cString weniger Wörter als die in @nIndex angegebene Anzahl enthält, gibt GETWORDNUM() eine leere Zeichenfolge zurück.

Siehe auch GETWORDCOUNT(), GETALLWORDS()

Beispiel

```
declare @cString nvarchar(4000)
set @cString = N'Wer immer strebend sich bemüht, Den können wir erlösen.'
select dbo.GETWORDNUM(@cString, 7, default) -- Anzeige 'können'
```

GETALLWORDS()

Fügt die Wörter aus einer Zeichenkette in eine Tabelle ein.

Syntax

GETALLWORDS(@cString[, @cDelimiters])

Parameter

@cString nvarchar(4000) - Spezifiziert die Zeichenkette, deren Wörter in die Tabelle @GETALLWORDS eingesetzt werden.
@cDelimiters - Gibt ein oder mehrere Zeichen an, durch die Zeichengruppen in @cString getrennt werden sollen. Die Standardtrennzeichen sind Leerzeichen, Tabulator- und Wagenrücklaufzeichen. Beachten Sie, dass GETWORDCOUNT() jedes der Zeichen in @cDelimiters als Trennzeichen verwendet und nicht die ganze Zeichenkette als einzelnes Trennzeichen.

Rückgabewert

Tabelle @GETALLWORDS (WORDNUM smallint, WORD nvarchar(4000), STARTOFWORD smallint, LENGTHOFWORD smallint)

Hinweise

GETWORDCOUNT() geht standardmäßig davon aus, dass Wörter durch Leerzeichen oder Tabstopps getrennt werden. Wenn Sie als Trennzeichen andere Zeichen angeben, ignoriert diese Funktion Leerzeichen und Tabstopps und verwendet nur die angegebenen Zeichen.

Siehe auch GETWORDNUM(), GETWORDCOUNT().

Beispiel

```
declare @cString nvarchar(4000)
set @cString = N'Wo fass ich dich, unendliche Natur? Euch Brüste, wo? Ihr Quellen alles Lebens'
select * from dbo.GETALLWORDS(@cString, default)
select * from dbo.GETALLWORDS(@cString, ' ,.?')
```

PROPER()

Gibt für einen Zeichenausdruck eine Zeichenfolge zurück, deren Wörter kleingeschrieben sind, aber jeweils mit einem Großbuchstaben beginnen.

Syntax

```
PROPER(@cExpression)
```

Parameter

@cExpression - Gibt den Zeichenausdruck an, von dem PROPER() eine Zeichenfolge zurückgibt, deren Wörter kleingeschrieben sind, aber jeweils mit einem Großbuchstaben beginnen.

Rückgabewert

Nvarchar(4000)

Hinweise

Siehe auch lower(), upper().

Beispiel

```
select dbo.PROPER(N'JOHANN CARL FRIEDRICH GAUß') -- Anzeige 'Johann Carl Friedrich Gauß'
```

ARABTOROMAN()

Wandelt einen numerischen Ausdruck (von 1 bis 3999) in römische Ziffern um.

Syntax

```
ARABTOROMAN(@tNum)
```

Parameter

@tNum Zahl

Rückgabewert

Varchar(15)

Beispiel

```
select dbo.ARABTOROMAN(1777) -- Anzeige MDCCLXXVII
```

ROMANTOARAB()

Wandelt römische Ziffern (von I bis MMMCMXCIX) in einen numerischen Ausdruck um.

Syntax

ROMANTOARAB(@tcRomanNumber)

Parameter

@tcRomanNumber - varchar(15) römische Ziffern.

Rückgabewert

Smallint

Beispiel

select dbo.ROMANTOARAB('MDCCCLV') -- Anzeige 1855

Mehr als 5000 Entwickler haben bereits meine Funktionen gedownloadet. Ich hoffe, dass sie auch für Sie nützlich sind.

Um mehr Informationen über die Zeichenketten UDFs in Transact-SQL zu erhalten, besuchen Sie bitte:

http://www.universalthread.com/wconnect/wc.dll?LevelExtreme~2,54,33,27115