Tipps & Tricks mit Visual Extend (VFX)

Stefan Zehner

Tipps & Tricks mit VFX - Eine Sammlung von Nutzungsmöglichkeiten verschiedener Eigenschaften in VFX und/oder „blkankem“ VFP.

Übersicht

Die hier behandelten Themen werden folgend mit Stichwort zur Erklärung aufgeführt und unter eigener Überschrift beschrieben:

- **Refresh() nach cPickfield**
  Damit alle beeinflussten Controls sofort uo to date sind
- **„Picken“ im Grid**
  Wie binde ich Pickfield-Funktionalität in ein Grid ein?
- **Datenformate und cUpdateSourceFields**
  Keine Probleme mit Datenformatierung nach einem „Pick“
- **Memofelder und Grids**
  Wie zeige ich Memofelder im Grid an?
- **Positionierung im Grid**
  Wie kann ich bestimmen was oben oder unten steht?
- **nFormStaus im Childform**
  Anpassen der Formularreaktion von Parent/Child-Formularen

Refresh() nach cPickField

Eines der hilfreichsten Tools in VFX ist die cPickField-Klasse. Wenn Sie richtig angewendet wird, kann man sich eine Menge Arbeit sparen und macht ein Project 'more productive' .

Es gibt viele Dinge, die bei der Benutzung von cPickField bedacht werden müssen. Eines dieser Dinge ist die Nachbehandlung von Controls, die übernommene Daten anzeigen sollen:

Da VFX nach einem 'Pick' nicht weiß, ob es auf dem zugehörigen Formular Controls gibt die übernommene Daten anzeigen müssen, müssen Sie das selbst in die Hand nehmen. Die Daten sind übernommen, werden aber standardmäßig erst nach dem Speichern angezeigt. Zwei kleine Dinge müssen Sie dafür ändern:

1. Das Property *ldovalid* von unserem *cPickFIeld* muss auf *.T.* gesetzt werden, damit die vom PickField eigene *valid()* Methode aufgerufen wird. In dieser können Sie sich dann um die Controls kümmern. Im *Powerbuilder* für die *cPickFieldKlasse* können Sie ein entsprechendes Häkchen setzen, welches dann diese Eigenschaft automatisch setzt.
2. In dem *valid* vom *txtField* des *cPickFields* bauen Sie dann je einen *refresh()* pro Control ein, welches durch das „Picken“ beeinflusst wird.

VFX selbst macht keinen 'refresh()' aus gutem Grund: Nicht immer werden übernommene Daten angezeigt, und wo diese Daten gegebenenfalls angezeigt werden kann VFX auch nicht wissen. Das hätte zur Folge, dass VFX einen 'refresh()' immer auf die ganze Form machen müsste. - Und damit würde die Performance unserer Form ganz gehörig in den Keller gehen. Das wäre dann gar nicht mehr 'productive'.

„Picken“ im Grid

In einem Grid wird für die Suche und Übernahme von Daten aus anderen Tabellen nicht die *cPick-Field*-Klasse, sondern die *cPickTextBox*-Klasse benutzt.

Die Funktionalität ist natürlich ähnlich wie in *cPickField*, jedoch gibt es weder einen Knopf zum Aufruf der PickList noch ein Ergebnisfeld (*txtDesc*).

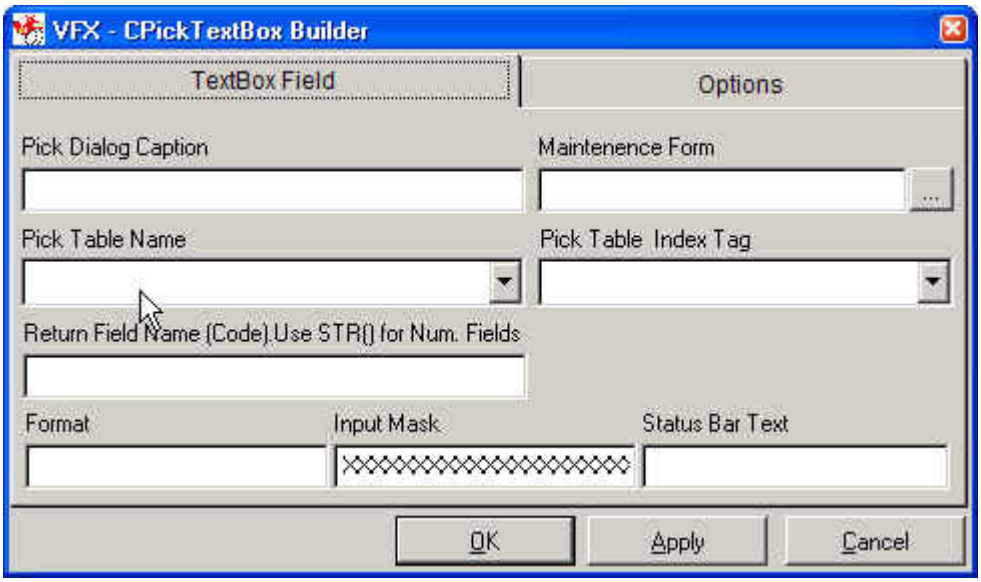
Um diese Klasse effektiv nutzen zu können, hat VFX auch hierfür einen *Powerbuilder*, der entsprechend dem Builder für die *cPickField*-Klasse funktioniert. Diesen können Sie allerdings nicht direkt über ein Kontextmenü aufrufen, da 1. Einstellungen für die Spalte vom Builder vorgenommen werden, und nicht für das Datencontrol in der Spalte, und 2. Ein Kontextmenü auf einem Grid aufgerufen sich immer auf das gesamte Grid und nicht auf die Spalte bezieht.

Nachfolgend der Weg zum Aufrufen des *VFX-PowerBuilder*:

- Wählen Sie die Spalte des Grids aus, in der die PickTextBox eingebaut ist.
  Wichtig: Die Spalte, nicht die PickTextBox selbst!
- Jetzt aus dem Menü 'VFX' den Eintrag *VFX Powerbuilders* aufrufen.

VFX erkennt selbständig, ob das für uns passende Objekt den Fokus hat und ruft den entsprechenden Builder auf. Sollten Sie auf einem anderen Objekt stehen, wird automatisch der zugehörige VFX-Powerbuilder aufgerufen. Gibt es keinen VFX-Powerbuilder, wird der interne VFP-Builder gestartet, sofern vorhanden.

In diesem Fall muss der VFX-Powerbuilder folgendermaßen aussehen:



In diesem Builder können Sie nun alle PickField-Einstellungen wie gewohnt vornehmen.

Datenformate und cUpdateSourceFields

In einem *cPickField* besteht die Möglichkeit weitere Datenfelder automatisch aus der Source-Tabelle zu befüllen. Dazu werden die Eigenschaften *cUpdSourceFields* und *cUpdTargetFields* benutzt, die auch im Builder schon gefüllt werden können.

Hierbei reicht es im Normalfall aus, jeweils eine Liste mit Tabellenfeldnamen getrennt mit Semikola einzugeben. Lediglich auf die Reihenfolge muss geachtet werden.

In Ausnahmefällen, muss allerdings noch Hand angelegt werden, wenn Datentypen nicht übereinstimmen. In der Sourcetabelle ist ein Datenfeld z.B. als Character definiert und in der Targettabelle soll dessen Inhalt in ein Feld gespeichert werden, welches als Numeric definiert ist.
Die 'Befüllung' der Targettabelle wird in der Methode *updatetargetfields* automatisch vorgenommen. Allerdings funktioniert dieses nicht, wenn die Datenformate nicht übereinstimmen, Entweder gibt es dann eine VFP-Fehlermeldung bez. der Datenformate oder, wenn man wie in diesem Beispiel, einfach ein val() um das Feld in *cUpdSourceFields* bastelt, keine Fehlermeldung, aber es wird auch kein Wert übernommen.

Wenn man sich diese Funktion einmal ansieht, werden auch keine Überprüfungen mit Type() vorgenommen. Um trotzdem den Wert übernehmen zu können, müssen wir im Ereignis *updatetargetfields* unseres *cPickFIeld* folgendes tun:

```
[Schnipp...]

=DoDefault()        && Ausführung des Klassencodes
Replace <TaregTable>.<Tagefield> with val(<SourceTable>.<SourceField>) in <TargetTable>

[Schnapp...]
```

Natürlich müssen wir dieses Feld aus *cUpdTagerFields* und *cUpdSourceFields* rausnehmen!

Memo-Felder und Grids

Wir benutzen oft, ja fast in jedem Formular die Klasse cGrid, um Auflistungen anzuzeigen. Hier, genauso wie in Standard-VFX-Formularen oder in der cPickList. In allen Grids ist es mit einem kleinen Trick möglich, Inhalte aus Memofeldern anzuzeigen:

Bei einem cGrid in einem Formular, z. B. die List-Seite, manipulieren Sie die Eigenschaft *ControlSource* z. B. wie folgt:

```
left(mytable.mymemofiled,20)
```

Das gleiche können Sie auch in der Eigenschaft *cfieldlist* einer *cPIckTextBox* oder eines *cPickField* machen. Einfach den 'normalen' Feldausdruck mit einem z. B. 'Left()' erweitern.

Positionierung im Grid

Jedes mal, wenn ein Sortierungsbefehl in einem *cGrid* gefeuert wird, bleibt der zuletzt ausgewählte Datensatz weiterhin ausgewählt. Man sieht es an der Markierung der aktuellen Gridline.

Um den Datensatzzeiger automatisch nach einer Neusortierung an eine andere Stelle des Grids zu stellen, muss man in jedem 'Header' des Grids eine Änderung vornehmen:

[Schnipp…]

```
this.Parent.Parent.OnSetOrder(this.Parent)
GO top (zum Beispiel)
this.Parent.Parent.Refresh()
```

[Schnapp…]

Natürlich wäre es einfacher die KLasse cGrid abzuleiten., wenn man nicht schon in allen Formularen das original cGrid einsetzen würde.

nFormstatus im Childform

In vielen Fällen ist es sinnvoll korrespondierende Daten zu Daten in einem Formular in einem zweiten Formular anzuzeigen. Normalerweise passiert das mit der *onmore()* Methode und einigen Eigenschaften der aufrufenden Formulares und der Childformulares.

Hier wollen wir aber Daten aus dem gleichen Datensatz in einem Child-Formular anzeigen, welches aber im EDIT-Modus geöffnet wird, wenn auch aufrufende Formul gerade im EDIT-Modus ist. Folgende Dinge sind zu tun:

1. Erstellen der Formulare (Parent und Child) wie gewöhnlich mit Aktivierung der entsprechenden Propertie-Boxen im VFX-Builder

2. Der Aufruf des Childformulares geschieht wie gewohnt mittels der *onmore*()-Methode

3. Im Childformular wird eine Eigenschaft erstellt, z. B.: '*nParentFormstate*'

4. In der *Load()*-Methode des Child-Formulares wird diese Eigenschaft dann mit dem Namen des Parent-Formulares gefüllt:

   ```
   [Schnipp…]
       this.nPrentFormstate=_screen.activeform.nformstatus
       RETURN DODEFAULT()
   [Schnapp…]
   ```

5. Am Ende der Init()-Methode im Child-Formular wird dann der Status des Parent-Formulares abgefragt und der Status im Child-Formular entsprechend gesetzt:

   ```
   [Schnipp…]
       IF this.nParentFormstate=1
           this.onedit()
       ENDIF
   [Schnapp…]
   ```

Somit wird das Child-Formular nun im Edit-Modus geöffnet, wenn das Parent-Formular auch im Edit-Modus ist.

Für Fälle in denen Sie Generell den Status des aufrufenden Formulares übernehmen möchten codieren Sie anstatt des einfachen IF eine Casestruktur.

Beachten Sie bitte, dass der Formularstatus INSERT (2) nicht benutzt werden kann, wenn Sie im Childformular Daten des aufrufenden Forumlares benutzen möchten. In diesem Fall könnten Sie z. B. folgendermaßen vorgehen:

- Speichern Sie die Daten im aufrufenden Formular mit der Methode *onsave()*
- Legen Sie programmatisch einen Datensatz in der Tabelle für das Childformular an
- Wechseln Sie im aufrufenden Formular in den EDIT-Status (1)
- Rufen Sie dann das Childformular auf

Das ist sicherlich nur eine von mehreren Möglichkeiten die mir spontan einfallen, ich möchte Sie aber jetzt nicht alle im Detail wiedergeben, dazu wäre ein eigenes Themenpapier erforderlich.

Selbsterstellte Cursor

Normalerweise benutzen wir Tabellen oder Views in VFX-Formularen. Was aber nun, wenn wir Daten benutzen, die andernorts ihren Ursprung haben?

Wir erstellen uns einen *Cursor*. Damit der *Cursor* aber auch genauso funktioniert wie eine in der Datenumgebung geöffnete Tabelle, reicht es nicht aus eine Verbindung zur *ControlSource*-Eigenschaft herzustellen. Im *load*' des Formulars müssen wir dazu den *Cursor* als *InitialSelected Alis* in der Datenumgebung anmelden - ausser den Cursor zu erstellen natürlich:

[Schnipp…]

```
CREATE CURSOR mycursor ...
this.dataenvironment.initialselectedalias="mycursor"
RETURN DODEFAULT()
```

[Schnapp…]

Jetzt können wir den CURSOR wie eine Tabelle oder einen View benutzen.