# Inhalt

# 1. Problemstellung / Zielsetzung

Setzt man in einer VFX-Applikation CursorAdapters (CA) in Zusammenhang mit demSQL-Server ein, dann besteht keine Möglichkeit, lokal gewisse Defaults zu definieren (wie dies beispielsweise bei Remote Views noch möglich war). Korrekte Defaults sind jedoch Bedingung, dass gewisse Controls (z.B. Optiongroups) von Beginn weg eine Korrekte Anzeige aufweisen.

Nun werden ja Defaults bereits auf Datenbankebene definiert. Die einzige Herausforderung besteht nun darin, diese Werte bei Bedarf auf dem SQL-Server zu ermitteln und unseren VFX-Formularen (oder Funktionen) zur Verfügung zu stellen. Das ganze ist so generisch zu halten, dass alle Datenformulare die Defaultwerte automatisch beziehen.

# 2. Designüberlegungen

## 2.1. SQL-Server

Auf dem SQL-Server 2005 können die entsprechenden Information aus der View `information_schema.columns` abgerufen werden. Diese View ist jedoch versionsabhängig. Unsere Abfrage wird daher auf dem Server in einer Stored Procedure (SP) gekapselt.

## 2.2. VFX

Default-Werte werden in unseren Applikation in den meisten Fällen in Formularen benötigt, die auf dem cDataForm basieren (Methoden onInsert oder onPostInsert). Daher werden wir den Automatismus an dieser Stelle einbauen.

Es darf zudem angenommen werden, dass während der „Lebensdauer" eines Dataformulares die Default-Werte mehrmals benötigt werden. Diese jedesmal vom Server zu holen, wäre daher nicht sehr performant, bzw. vergrössert nur unnötig den Netzwerk Verkehr.

# 3. Lösungsweg

## 3.1. SQL-Server

Auf dem SQL-Server stellen wir eine Stored Procedure, die gemäss übergebenem Tabellennamen die Defaults ermittelt:

```
/****** Object:  StoredProcedure [dbo].[GetDefaultValues]    Script Date:
06/08/2008 16:40:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:          Fritz Maurhofer
-- Create date:     02.11.2007
-- Description:     Ermittelt die Attribute einer Tabelle
--                  mit Default-Werten
-- =============================================
CREATE PROCEDURE [dbo].[GetDefaultValues]
       -- Add the parameters for the stored procedure here
       @tcTableName VARCHAR(200)
WITH EXECUTE AS OWNER
AS
BEGIN
       -- SET NOCOUNT ON added to prevent extra result sets from
       -- interfering with SELECT statements.
       SET NOCOUNT ON;

       SELECT Table_name, Column_name, Column_default, IS_Nullable, Data_type
      FROM information_schema.columns
      where table_name = @tcTableName AND column_default is not null
END
```

Wichtig ist hier noch die Zeile:
`WITH EXECUTE AS OWNER`

Da ja normalerweise der Anwender einer Applikation nicht SA Rechte hat (bzw. diese auch nicht benötigt), hat er oder sie normalerweise keinen Zugriff auf Systeminformationen. Damit wird die SP im Sicherheits Kontext des Tabellenowners ausgeführt.

Retourniert wird ein Resultat Cursor.

## 3.2. VFX

### 3.2.1. fmGetSQLDefaults

Kernstück ist hier die Funktion fmGetSQLDefaults. Damit wird gemäss aktueller Connection die Information abgeholt und aufbereitet:

```
** PROGRAMMNAME.........: FMGETSQLDEFAULTS.PRG
** DATENBANKSYSTEM......: Visual FoxPro 09.00.0000.5411 for Windows
** SYSTEM...............: Bibliotheksfunktion
*) BESCHREIBUNG.........: Ermittelt die Attribute mit Default-Values
*)                        gemäss übergebenem Tabellen Name
** PARAMETERS PASSED....: <ExpC1> Name der Tabelle
**                        <ExpN>  Connectionhandle zur DB
** PARAMETERS RETURNED..: <ExpC>  Replace Statement (VFP-Syntax)
** BEMERKUNGEN..........:
**
** AUTHOR...............: Fritz Maurhofer
** ERSTELLT AM..........: 02.11.2007
** LETZTE AENDERUNG.....:
**
*********************************************************************
* COPYRIGHT: Maurhofer Informatik AG  - Softwareentwicklung         *
*            Gossauerstrasse 14, CH-8340 Hinwil Tel. 043 843 04 44 *
*            http://www.maurhofer-informatik.ch                     *
*********************************************************************
LPARAMETERS tcTableName, tnConnHandle
LOCAL lnCurrSele, lcRetVal, lcSqlExec, llNoDispError

#include "INCLUDE\VFX.H"

lnCurrSele = SELECT()
lcRetVal   = ""
lcSQLExec  = ""
lcMsg      = ""
IF VERSION(2) > 0
   llNoDispError = .F.
ELSE
   * in der Runtime unterdrücken
   llNoDispError = .T.
ENDIF

IF NOT EMPTY(tcTableName) AND tnConnHandle > 0
   lcSQLExec = "execute dbo.GetDefaultValues '" + ALLTRIM(tcTableName) + "' "
   IF vfxSQLEXEC(tnConnHandle, lcSQLExec, "DefaRes", llNoDispError) > 0
      IF USED("DefaRes")
         IF RECCOUNT("DefaRes") > 0
            lcRetVal = ComposeReplace()
         ENDIF
         USE IN DefaRes
      ENDIF
   ENDIF
ENDIF

SELECT (lnCurrSele)

RETURN lcRetVal
```

Die Funktion ComposeReplace setzt nun einen Foxpro ausführbaren Replace Befehl zusammen. Zu beachten ist, dass das Resultat (also die Defaults) im Unicode Format vorliegt und daher für unsere Zwecke noch umgewandelt werden muss:

```
    lcValueRaw = STRCONV(column_default,6)

*--
FUNCTION ComposeReplace
* setzt den Replace Befehl für Default-Werte
* gemäss SQL-Backend zusammen
* (gemäss aktuellem Resultatcursor)
LOCAL lcRetVal, lcDelim, lcValue, lcDataType, lcValueRaw

lcRetVal  = ""
lcDelim   = "REPLACE "
lcValue   = ""

SCAN
    lcDataType = LOWER(ALLTRIM(Data_type))
    lcValueRaw = STRCONV(column_default,6)
    IF ATC("NULL",lcValueRaw) > 0
       lcValue = ".NULL."
    ELSE
       DO case
       CASE ATC("space(",lcValueRaw) > 0
            * Funktion SPACE()
            IF LEFT(lcValueRaw,1) = "("
               * Anzahl Spaces ermitteln entfernen
               lcValue = "SPACE(" + STREXTRACT(lcValueRaw,"((","))") + ")"
            ELSE
               lcValue = lcValueRaw
            ENDIF
       CASE INLIST(lcDataType,"numeric")
            IF LEFT(lcValueRaw,2) = "(("
               * Umklammerung mit zwei Klammern weg
               lcValue = STREXTRACT(lcValueRaw,"((","))")
            ELSE
               * Annahme: dann hat es noch eine
               lcValue = STREXTRACT(lcValueRaw,"(",")")
            ENDIF
            IF EMPTY(lcValue)
               * falls die obigen Annahmen nicht stimmen
               * dann ist jetzt lcValue leer und wir nehmen
               * den Originalwert (ohne ASCI(0))
               lcValue = lcValueRaw
            ENDIF
       CASE INLIST(lcDataType,"smallint","tinyint","int","bigint")
            lcValue = TRANSFORM(EVALUATE(lcValueRaw))
       CASE INLIST(lcDatatype,"varchar","char","text")
            lcValue = EVALUATE(lcValueRaw)
       CASE INLIST(lcDatatype,"datetime")
            DO case
            CASE LEFT(lcValueRaw,2) = "(("
                 lcValue = STREXTRACT(lcValueRaw,"((","))")
            CASE LEFT(lcValueRaw,1) = "("
                 lcValue = SUBSTR(lcValueRaw,2,LEN(ALLTRIM(lcValueRaw))-2)
            OTHERWISE
                 lcValue = lcValueRaw
            ENDCASE
       CASE INLIST(lcDatatype,"bit")
            lcValue = TRANSFORM(EVALUATE(lcValueRaw) = 1)
       ENDCASE
    ENDIF
```

```
    * nun noch auf Funktionsaufrufe abtesten
    DO CASE
    CASE ALLTRIM(UPPER(lcValue)) = "GETDATE()"
         lcValue = "DATETIME()"
    ENDCASE
    IF NOT EMPTY(lcValue)
      lcRetVal = lcRetVal + lcDelim + ;
                 ALLTRIM(Column_name) + " WITH " + lcValue
      lcDelim  = ", "
    ENDIF
ENDSCAN

RETURN lcRetVal
```

Der obige Case ist natürlich um die verwendeten Funktionen zu ergänzen

### 3.2.2. vfxHook

Um Roundtrips zum Server zu vermeinden, werden die Defaults im Init ermittelt und in einer Formular Eigenschaft zwischengelagert:

```
CASE UPPER(tcEvent) = "CDATAFORM.INIT"


    *-- Beginn Gültigkeitsbereich lcAlias = Tabelle
    IF UPPER(LEFT(toForm.cWorkAlias,2)) = "CA"
       * CursorAdapter
       lcAlias = SUBSTR(toForm.cWorkAlias,3)
    ELSE
       lcAlias = toForm.cWorkAlias
    ENDIF

    *- Code für SQL-Defaults bestimmen
    IF PEMSTATUS(toForm,"cSQLDefault",5)
       IF NOT EMPTY(lcAlias)
          toForm.cSQLDefault = fmGetSQLDefaults(lcAlias, ;
                                                goProgram.oConnMgr.getConnection())
       ENDIF
    ENDIF
```

Beim bestimmen des Alias-Namens gehen wir von der Namenskonvention aus, die vom Cursor Adapter Wizard vorgegeben wird: „CA“ + Namen der Tabelle.

### 3.2.3. cDataform

Folgende Ergänzungen sind hier notwendig:

| Eigenschaften | |
|---|---|
| cSQLDefault | Hier wird der vom Hook gem. 3.2.2 ermittelte Code geparkt. |
| **Methoden** | |
| onPostInsert | Hier wird nun der Code gem. cSQLDefault ausgeführt:<br><br>`LOCAL loError, lcMsg`<br>`lcMsg = ""`<br><br>`* zuerst SQL-Defaults`<br>`IF NOT EMPTY(this.cSQLDefault)`<br>`  TRY`<br>`  EXECSCRIPT(this.cSQLDefault)`<br><br>`  CATCH TO loError`<br>`  lcMsg = this.cSQLDefault + ID_CRLF +;`<br>`  fmFormCatchErr(loError, 2)`<br><br>`  ENDTRY`<br>`  IF NOT EMPTY(lcMsg)`<br>`  goProgram.vfxmessagebox(lcMsg,0+48,;`<br>`  "Error Setting SQL-Server Defaults")`<br>`  ENDIF`<br><br>`ENDIF`<br><br>`* dann den Rest`<br>`RETURN DODEFAULT()` |

Die Ausgabe einer Meldung im Fehlerfall ist mehr für den Entwickler gedacht, da der Code generisch erstellt wird und nach dem debuggen…
<SET fromme Wünsche ON>
…grundsätzlich nicht mehr zu Fehlern führen sollte….
<SET fromme Wünsche OFF>.

# 4. Dateiverzeichnis

Zu diesem Thema werden folgende Dateien mitgeliefert:

vfxHook_snippet

| | |
|---|---|
| Lib | Ergänzte Klassen (ersetzen, bzw. ergänzen die Origniale in vfxForm, resp. vfxObj): |
| | vfxForm_trans cDataform |
| Programm | vfxHook_snippet.prg |
| SQL | Script für die GetDefaultValues Stored Procedure |
| Text | Dieses Dokument (UebernahmeSQLDefaults.pdf) |