Visual.NET Extensions – Tutorial

“vdxExplorerTree - Erweiterte Datenmanipulation mit Treeview“

Das umfangreiche Applikationsentwicklungs-Framework für die einfache Entwicklung von Microsoft Visual Studio.NET Datenbank-Applikationen!

Devigus Engineering AG
Grundstrasse 3
CH-6343 Rotkreuz
Internet: http://www.devigus.com
Email: deag@devigus.com

Version: 2.0
Letztes Update: 18.11.2003

Inhaltsverzeichnis

1 vdxExplorerTree - Erweiterte Datenmanipulation mit Treeview

In diesem Tutorial wird ein weiteres *vdxExplorerTree*-Szenario implementiert. Zusätzlich sollen aber die zu verwaltenden Entitäten – in diesem Fall Personen – mit der Adressverwaltung verknüpft werden. Bitte verwechseln Sie dieses Szenario nicht mit dem OneToMany (Eins-zu-n) Szenario, wo die Parent und die Child Tabelle in demselben Szenario bearbeitet wird. Im vorliegenden Fall wird lediglich eine Personenverwaltung (child) mit der Addressverwaltung (parent) über den Treeview verbunden.

Die Implementierung der Personenverwaltung gestaltet sich grundasätzlich analog zur Adressverwaltung. Da Personen aber bestimmten Adressen zugeordnet werden können, sind an bestimmten Stellen Erweiterungen vorzunehmen. Dieses Tutorial geht insbesondere auf diese Erweiterungen ein.

1.1 Definition der Namespaces

Die Platzhalter stehen für die konfigurierten Projektnamen und Namespaces im ApplicationWizard.

Platzhalter	Beispiel	Beschreibung
<ServerProjectName>	mySampleServer	Name und Default-Namespace für Server-Projekt
<ClientProjectName>	mySampleClient	Name und Default-Namespace für Client-Projekt
<WebServiceName>	mySampleWebServices	Name und Default-Namespace für WebService.

Tabelle 1 Definition der Namespaces

1.2 Erstellung der Web Services

1. Öffnen Sie die Server-Solution *mySampleServer.sln*.
2. Folgende Web Services müssen zuerst erstellt werden um den Zugriff auf die Datenbank-Tabellen zu ermöglichen:
   - wsLang
   - wsPers

   Siehe dazu Kapitel 1.7 und 1.8 des Server Tutorials *VDXTut_Server_de*.
3. Öffnen Sie Client-Solution *mySampleClient.sln*.
4. Aktualisieren Sie die Web-Referenzen über das Kontext-Menü von *mySampleWebServices*.
5. Damit man *typensicher* auf die Web Services zugreifen kann, sollte die Klasse *WebServiceProvider* mit folgenden Properties ergänzt werden.

```
static public wsLangSoap wsLang
{
    get { return (wsLangSoap)GetWebService(typeof(wsLangSoap)); }
}

static public wsPersSoap wsPers
{
    get { return (wsPersSoap)GetWebService(typeof(wsPersSoap)); }
}
```

Codesegment 1 Properties von WebServiceProvider ergänzen

1.3 vdxExplorerTree-Szenario am Beispiel der Personenverwaltung

In einem ersten Schritt implementieren Sie die Personenverwaltung analog zur Adressenverwaltung. Folgende Schritte beschreiben den grundsätzlichen Ablauf der Implementierung. Lesen Sie bei Bedarf im *vdxExplorerTree – Standard* - Tutorial und in der mitgelieferten Beispiel-Applikation nach, um Details zur entsprechenden Implementierung zu erfahren:

1. Hinzufügen des Suchen-Controls für die Personen (Klasse *SelectionPerson*)
2. Hinzufügen des Anzeige-Controls für die gefundenen Personen (Klasse *SelectionGridPerson*)
3. Hinzufügen des Anzeige-Controls für die Personendetails (Klasse *DataDetailPerson*)
4. Hinzufügen des Container-Controls für die Personenverwaltung (Klasse *ExplorerPerson*)
5. Programmierung des Suchen-Controls (Klasse *SelectionPerson*)
6. Programmierung des Personendetail-Controls (Klasse *DataDetailPerson*)
7. Programmierung des Container-Controls (Klasse *ExplorerPerson*)
8. Programmierung der *vdxActionItems*

Fahren Sie erst weiter wenn Sie diese Schritte erledigt sind!

1.4 Unterschiede zur Adressverwaltung

In bestimmten Fällen unterscheidet sich die Implementation der Personenverwaltung von der Adressverwaltung. Folgende Abschnitte beschreiben diese Unterschiede.

1.4.1 Erstellen der *SelectionPerson*-Klasse

1. Als erstes legen Sie das Verzeichnis Person an.
2. Die Basisklasse für *SelectionPerson* ist *AppSelection* und nicht *AppSelectionTab*. Da die Personen-Suchmaske nur auf einer Seite implementiert wird, benötigt es keine Tab-Pages auf der Suchmaske. Fügen Sie auf der *SelectionPerson*-Klasse folgende Controls ein:

Eingefügtes Control	Text	Name
AppLabel	First Name	lblFirstName
AppLabel	Last Name	lblLastName
AppLabel	Country	lblCountry
AppLabel	Order By	lblOrderBy
AppTextBox		txtFirstName
AppTextBox		txtLastName
AppTextBox		txtCountry
AppComboBox		cboOrderBy

Tabelle 2 Controls auf SelectionPerson

3. Selektieren Sie das Control *cboOrderBy* und öffnen Sie das *Property-Sheet* um folgende *Items* hinzuzufügen.
   - LastName
   - FirstName
   - Country

   Ein Layout-Vorschlag zeigt die folgende Abbildung:

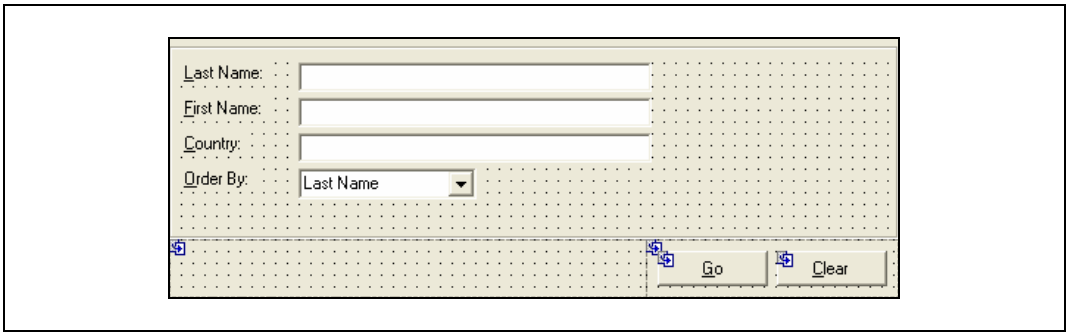

Abbildung 1 SelectionPerson in Design-Ansicht

1.4.2 Überschreiben der *FillSearchParameters*-Methode

Fügen Sie den folgenden Code in die Methode ein:

```
vdxSearchParameters sp = new vdxSearchParameters("SelectionPersByDescr");
sp.Add("FirstName", this.txtFirstName.Text, vdxSearchType.StartsWith);
sp.Add("LastName", this.txtLastName.Text, vdxSearchType.StartsWith);
sp.Add("Country", this.txtCountry.Text, vdxSearchType.StartsWith));
sp.Add("OrderBy", "LastName"); // Sort order by last name column
return sp;
```

Codesegment 2 Überschreiben der FillSearchParameters-Methode von SelectionPerson

1.4.3 DataSetReference *dsrPersList* an *DataGrid* binden

Im Unterschied zur Adressenverwaltung muss die DataSetReference *dsrPersList* an die Klasse *SelectionGridPerson* gebunden werden. Öffnen Sie das *VDX-Register* der ToolBox und ziehen Sie ein *vdxDataSetReference* auf das *SelectionGridPerson*-Control. Benennen Sie es von *vdxDataSetReference1* nach *dsrPersList* um und konfigurieren es wie folgt.

Property	Selektion
Name	dsrPersList
BindingContainer	SelectionGridPerson
RefDataSet	<ClientProjectNamespace>.<WebServiceName>.dsPersList

Tabelle 3 Konfiguration des PersList-DataSetReference

Folgende Tabelle enthält die Auswahl der entsprechenden Eigenschaften/Properties des *SelectionGridPerson*-Controls:

Property	Selektion
DataSource	dsrPersList
DataMember	Pers

Tabelle 4 Konfiguration von SelectionGridPerson

Konfigurieren Sie die ColumnStyles wie folgt:

Column	Header Text	Width
LastName	Last Name	75
FirstName	First Name	75
Title	Title	30
Phone	Phone	75

Tabelle 5 ColumnStyles von SelectionGridPerson

1.4.4 Erstellen der *DataDetailPerson*-Klasse

1. Fügen Sie auf der *DataDetailPerson*-Klasse folgende Controls ein:

Einzufügendes Control	Text	Name
AppCheckGroupBox	Use Special Salutation	cgbUseSpecSal
AppLabel	Title	lblTitle
AppLabel	Last Name	lblLastName
AppLabel	First Name	lblFirstName
AppLabel	Sex	lblSex
AppLabel	Phone	lblPhone
AppLabel	Fax	lblFax
AppLabel	Mobile	lblMobile
AppLabel	E-Mail	lblEmail
AppLabel	Language	lblLanguage
AppLabel	Last Contact	lblLastContact
AppLabel	Address	lblAddress
AppLabel	Letter	lblLetter
AppButton	Go	btnGoEmail
AppTextBox		txtTitle
AppTextBox		txtLastName
AppTextBox		txtFirstName
AppTextBox		txtPhone
AppTextBox		txtFax
AppTextBox		txtMobile
AppTextBox		txtEmail
AppTextBox		txtLetter
DataPickFieldAddress[1]		pickFieldAddress
AppDateTimePickField		dateTimePickField
AppComboBox		cboLanguage

Tabelle 6 Controls auf DataDetailPerson

[1] DataPickFieldAddress in Design-Ansicht öffnen, dann erscheint das Control im *My User Controls*-Register der Toolbox.

2. Ordnen Sie die Controls folgendermassen an:

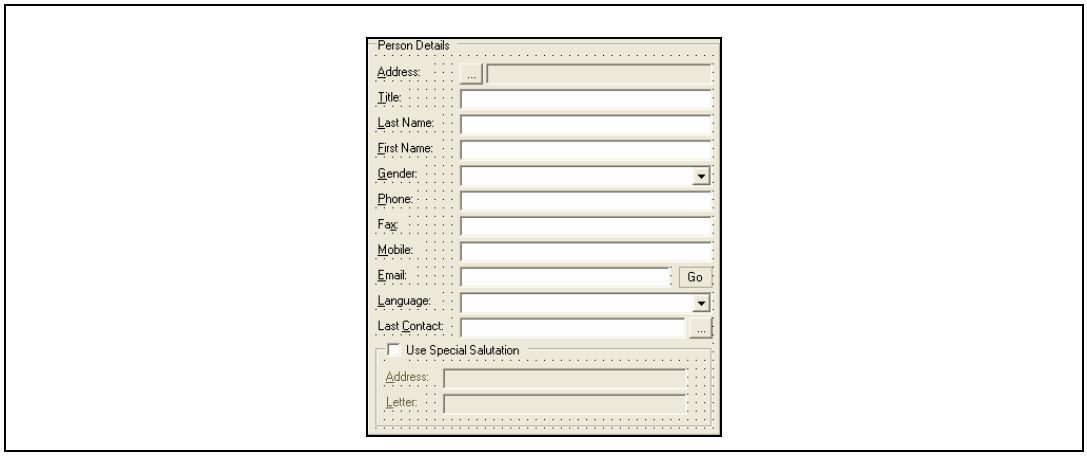

Abbildung 2 DataDetailPerson in Design-Ansicht

Hinweis

Falls das Control nicht im *My User Controls*-Register der Toolbox enthalten ist, dann schliessen Sie die Solution. Beim nächsten Öffnen der Solution sollte das *DataPickFieldAddress*-Control angezeigt werden.

Eine weitere Lösung besteht im Hinzufügen des Controls über *Customize Toolbox...* aus dem Kontextmenu der Toolbox. Wählen Sie als Assembly die Datei *mySampleClient.exe* aus dem *\bin\Debug*-Verzeichnis ihrer Solution aus

1.4.5 Konfiguration des *dataStatusManager*

Setzen Sie die Eigenschaften/Properties des *dataStatusManager*-Objektes auf die folgenden Werte:

Property	Selektion
DataSetType	<ClientProjectNamespace>.<WebServiceName>.dsPers
MainDataTable	Pers
SelectedDataTable	Pers

Tabelle 7 Konfiguration des DataStatusManagers von DataDetailPerson

Zusätzlich können auf dem *dataStatusManager* Status-Texte gesetzt werden und Fragestellungen, ob gespeichert oder gelöscht werden soll, definiert werden.

1.4.6 DataStatusManager und DataSetReference *dsrLang* an Controls binden

Der *dataStatusManager* muss an die entsprechenden Controls der Klasse *DataDetailPerson* gebunden werden. Folgende Tabelle beschreibt die entsprechenden DataBindingInfos:

Control	Property DataBindingInfos.Text
txtTitle	dataStatusManager.Pers.Title
txtLastName	dataStatusManager.Pers.LastName
txtFirstName	dataStatusManager.Pers.FirstName
txtPhone	dataStatusManager.Pers.Phone
txtFax	dataStatusManager.Pers.Fax
txtEmail	dataStatusManager.Pers.EMail
txtLetter	dataStatusManager.Pers.SAL_Letter
cgbUseSpecSal	dataStatusManager.Pers.UseSpecSal (Property DataBindingInfos.Checked)
dateTimePicker	dataStatusManager.Pers.LastContact (Property DataBindingInfos.Date)

Tabelle 8 DataStatusManager an Controls binden

Damit das *pickFieldAddress*-Control die entsprechenden Adressdaten einer Person anzeigen kann, muss es mit dem DataStatusManager verknüpft werden. Setzen Sie dazu die folgenden *DataBindingInfos*-Properties mit den entsprechenden Werten:

Property	Selektion
(DataBindingInfos).PickForeignKey	dataStatusManager.Pers.Addressid
(DataBindingInfos).PickDescription	dataStatusManager.Pers.ADDRESSDESCR

Tabelle 9 Konfiguration des *pickFieldAddress* auf DataDetailPerson

Analog dazu muss die *dsrLang*-DataSetReference an das *cboLanguage*-Control gebunden werden:

Property	Control cboLanguage
SelectedValue	dataStatusManager – Pers.Lang
DataSource	dsrLang
Display Member	Lang.Descr
Value Member	Lang.Lang

Tabelle 10 DSM und DSR an Language-ComboBox binden

Damit die *Language*-ComboBox mit den entsprechenden Sprachen gefüllt wird, fügen Sie den folgenden Code in den Konstruktor der *DataDetailPerson*-Klasse ein; fügen Sie ihn nach dem *InitializeComponent*-Aufruf ein:

```
dsrLang.FillData(WebServiceProvider.wsLang.GetDataSetLang());
this.DataStatusManager.Updater = WebServiceProvider.wsPers;
```

Codesegment 3 Implementation des DataDetailPerson-Konstruktors

1.4.7 Überschreiben der *LoadData*-Methode im *DataDetailPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
return WebServiceProvider.wsPers.GetDataSetPers(
       searchParameters.GetParameters());
```

Codesegment 4 Überschreiben der LoadData-Methode von DataDetailPerson

1.4.8 Überschreiben der *GetInfoBarText*-Methode im *DataDetailPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
dsPers ds = DataStatusManager.DataSet as dsPers;
if (ds != null && ds.Pers.Count == 1)
{
    dsPers.PersRow row = (dsPers.PersRow)ds.Pers.Rows[0];
    return string.Format("{0} {1}", row.LastName, row.FirstName);
}
return string.Empty;
```

Codesegment 5 Überschreiben der GetInfoBarText-Methode von DataDetailPerson

1.4.9 Konfiguration des *ExplorerPerson*-Controls

1. Fügen Sie auf dem Explorer eine neue ImageList hinzu und benennen Sie diese imageList.

2. Fügen Sie der imageList folgende Icons hinzu.

Collection Index	Image
0	Selection.ico
1	Arrow.ico
2	Person.ico

Tabelle 11 Konfiguration der ImageList von ExplorerPerson

Folgende Tabelle fasst die Einstellungen der vier wichtigsten Eigenschaften/Properties zusammen:

Property	Selektion
SearchControlType	<ClientProjectNamespace>.<WebServiceName>.SelectionPerson
SelectionGridType	<ClientProjectNamespace>.<WebServiceName>.SelectionGridPerson
DisplayMode	Selection
ImageList (treeView)	imageList

Tabelle 12 Konfiguration von ExplorerPerson

3. Kopieren Sie die folgenden Namespace-Referenzen in den Code.

```
using System.Data;
using System.Text;
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Controls.Base;
using Deag.Vdx.Controls.Core;
using <ClientProjectNamespace>.<WebServiceName>;
```

Codesegment 6 Using-Anweisung für ExplorerPerson

1.4.10 Überschreiben der *LoadData*-Methode ohne Argument im *ExplorerPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
dsPersList ds = WebServiceProvider.wsPers.GetDataSetPersList(
       this.SearchParameters.GetParameters());
if (ds.Pers.Count == 0)
{
   MessageBox.Show("No person found. Please try again!",
       "Person Search", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
return ds;
```

Codesegment 7 Überschreiben der LoadData-Methode in ExplorerPerson

1.4.11 Überschreiben der *GetLoaderInfo*-Methode im *ExplorerPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
vdxSearchParameters sp = null;
dsPersList.PersRow row = navigatorInfo.SelectedRow as dsPersList.PersRow;

if (row != null)
{
   sp = new vdxSearchParameters("Person");
   sp.Add("PersId", (int)row.PersID);
}
return new vdxLoaderInfo(searchParameters, typeof(DataDetailPerson));
```

Codesegment 8 Überschreiben der GetLoaderInfo-Methode in ExplorerPerson

1.4.12 Überschreiben der *GetRootNode*-Methode im *ExplorerPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
return new vdxTreeNode("Found Persons", 0, 1);
```

Codesegment 9 Überschreiben der GetRootNode-Methode in ExplorerPerson

1.4.13 Überschreiben der *GetTreeNode*-Methode im *ExplorerPerson*-Control

Fügen Sie den folgenden Code in diese Methode ein:

```
StringBuilder sb = new StringBuilder();
dsPersList.PersRow row = ((vdxNavigatorInfoItemRow)navInfoItem).DataRow as
       dsPersList.PersRow;

if (isCopied)
    sb.Append("Copy of ");

if (row != null)
{
    if (!(row.IsLastNameNull() && row.IsFirstNameNull()))
    {
        sb.Append(row.LastName);
        sb.Append(" ");
        sb.Append(row.FirstName);
    }
    else
    {
        sb.Append("New person");
    }
}

return new vdxTreeNode(sb.ToString(), 2, 1, false);
```

Codesegment 10 Überschreiben der GetTreeNode-Methode in ExplorerPerson

1.4.14 Programmierung des *vdxActionItem*

Fügen Sie den folgenden Code in die *aiExplorePerson_ActionInvoked*-Event-Listener-Methode ein:

```
Show(typeof(Person.ExplorerPerson));
```

Codesegment 11 Implementation des ActionItem-ExplorePerson-Invoked-Listeners

1.5 Verknüpfung mit der Adressverwaltung

Nach Fertigstellung der isolierten Personenverwaltung werden nun die zusätzlichen Ergänzungen gezeigt, welche zur Verknüpfung mit der Adressverwaltung notwendig sind.

1. Öffnen Sie die Klasse ExplorerAddress in Code-Ansicht.
2. Fügen Sie ein neues Feld subNodeSearchParameters hinzu.

```
private vdxSearchParameters subNodeSearchParameters;
```

Codesegment 12 Variable subNodeSearchParameters in ExplorerAddress

3. Ergänzen Sie die *LoadData*-Methode ohne Argument.

```
dsAddressList ds = WebServiceProvider.wsAddress.GetDataSetAddressList(
   this.SearchParameters.GetParameters());
if (ds.Address.Count == 0)
{
   MessageBox.Show("No address found. Please try again!",
      "Address search", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

this.subNodeSearchParameters = searchParameters;

return ds;
```

Codesegment 13 LoadData-Methode ohne Argument in ExplorerAddress ergänzen

4. Überschreiben Sie die *LoadData(vdxSearchParameters)*-Methode.

```
dsAddressList ds = wsAddress.GetDataSetAddressList(
    searchParameters.GetParameters());
if (ds.Pers.Count == 0)
{
    MessageBox.Show("No person found. Please try again!",
       "Person search", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

this.subNodeSearchParameters = searchParameters;

return ds;
```

Codesegment 14 Überschreiben der LoadData(vdxSearchParameters)-Methode in ExplorerAddress

Diese Methode wird immer dann aufgerufen, wenn ein Datenbezug für den Sub-Level ausgeführt werden muss, anders formuliert: Das Expandieren eines Adressen-Tree-Knotens führt zum Laden der enstprechenden Personen-Tree-Knoten.

5. Überschreiben Sie die *DesiredTable()*-Methode. Falls das geladene DataSet mehrere Tabellen enthält, muss über diese Methode anhand der gespeicherten *subNodeSearchParameters* aus der *LoadData(vdxSearchParameters)*-Methode die gewünschte Tabelle gesetzt werden.

```
if (this.subNodeSearchParameters != null)
{
    switch (this.subNodeSearchParameters.VdxSearchName)
    {
       case "Address":
          return dataSet.Tables["Address"];

       case "PersTree":
          return dataSet.Tables["Pers"];
    }
}
return base.DesiredTable(dataset);
```

Codesegment 15 Überschreiben der DesiredTable-Methode in ExplorerAddress

6. Überschreiben Sie die GenerateSubNodeSearchParameters-Methode

```
vdxSearchParameters sp = null;
dsAddressList.AddressRow row =
  ((vdxNavigatorInfoItemRow)navInfoItem).DataRow asdsAddressList.AddressRow;

if (row != null)
{
    sp = new vdxSearchParameters("PersTree");
    sp.Add("AddressId", (int)row.AddressID);
}

return sp;
```

Codesegment 16 Überschreiben der GenerateSubNodeSearchParameters-Methode

7. Ergänzen Sie die *GetTreeNode*-Methode.

```
StringBuilder sb = new StringBuilder();
int imageIndex = 0;
int selectedImageIndex = 1;
bool hasChildren = false;

if (isCopied)
    sb.Append("Copy of ");

DataRow row = ((vdxNavigatorInfoItemRow)navInfoItem).DataRow;
switch (row.Table.TableName)
{
   case "Address":
       if (!Convert.IsDBNull(row["Company"]))
       {
          sb.Append(row["Company"]);
       }
       else
       {
          sb.Append("New address");
       }
       imageIndex = 2;
       hasChildren = true;
       break;

   case "Pers":
       if (!(Convert.IsDBNull(row["LastName"]) &&
          Convert.IsDBNull(row["FirstName"])))
       {
          sb.Append(row["LastName"]);
          sb.Append(" ");
          sb.Append(row["FirstName"]);
       }
       else
       {
          sb.Append("New person");
       }
       imageIndex = 3;
       break;
}
return new vdxTreeNode(sb.ToString(), imageIndex, selectedImageIndex,
       hasChildren);
```

Codesegment 17 GetTreeNode-Methode in ExplorerAddress ergänzen

Dieser zusätzliche *case*-Block erzeugt – falls vorhanden – die TreeNodes mit den enstprechenden Personen-Daten, d.h. Anzeige der entsprechenden Sub-Tree-Knoten, die aufgrund des Öffnens eines bestimmten Tree-Knotens gefunden wurden.

8. Ergänzen Sie die *GetLoaderInfo*-Methode.

```
vdxSearchParameters sp = null;
Type  dataDetailType = typeof(DataDetailAddress);
DataRow row = navigatorInfo.SelectedRow;

if (row != null)
{
   switch (row.Table.TableName)
   {
      case "Address":
         int addressId = (int)row["AddressID"];
         sp = new vdxSearchParameters("Address");
         sp.Add("AddressId", addressId);
         break;

      case "Pers":

         int persId = (int)row["PersID"];
         sp = new vdxSearchParameters("Person");
         sp.Add("PersId", persId);
         dataDetailType = typeof(Person.DataDetailPerson);
         break;
   }
}
return new vdxLoaderInfo(sp, dataDetailType);
```

Codesegment 18 GetLoaderInfo-Methode in ExplorerAddress ergänzen

Damit ist die Personenverwaltung mit der Adressverwaltung verknüpft. Im Tutorial "*vdxExplorerTree* – One To Many Datenmanipulation" wird 1:N-Szenario implementiert .

1.6 Bildverzeichnis

1.7 Tabellenverzeichnis

1.8 Codesegment-Verzeichnis