Visual.NET Extensions – Tutorial

“vdxExplorerTree - Extended data-manipulation with treeview“

The extensive application development framework for the simple development of Microsoft Visual Studio.NET database applications!

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

Version: 1.1
Last Update: 11.11.2002

1 vdxExplorerTree - Extended data-manipulation with treeview

In this tutorial, another *vdxExplorerTree*-Scenario will be implemented. The extension compared to the first *vdxExplorerTree* standard tutorial consists of the additional integration of a child table which is individually managed – in this tutorial these are persons – and linked with the addresses. Do not confound this scenario with the OneToMany scenario, where the parent and child tables are managed within the same data manipulation (this is described in a separate tutorial). Here, essentially, we have a regular person data-management scenario which is integrated through a treeview navigation with the address data-manipulation.

In the first step, the implementation of the person data-management is analog to the address data-management.. Since persons can be linked to addresses (AddressID is a foreign key in the person table), we need to extend our scenario at the appropriate locations. This tutorial shows especially these extensions.

1.1 vdxExplorerTree-scenario for the person

In the first step, we implement the person data-management analog to the address scenario. Following steps describe the principal procedure; if you need additional support, have a look at the tutorial "*vdxExplorerTree - Standard data-manipulation with treeview"* and the VDX Sample Application:

1. Add a Container-Control for the person data-management (class *ExplorerPerson*)
2. Add the Search-Control for the selection dialog (class *SelectionPerson*)
3. Add the Control for the list of found persons (class *SelectionListPerson*)
4. Add the Datadetail-Control (class *DataDetailPerson*)
5. Program the Search-Control (class *SelectionPerson*)
6. Program the Datadetail-Control (class *DataDetailPerson*)
7. Program the Container-Control (class *ExplorerPerson*)
8. Program the *vdxActionItems*

1.2 Differences compared to the address scenario

Regarding some details, the implementation of the person scenario differs from the one of the addresses. In the following steps, these differences are described...

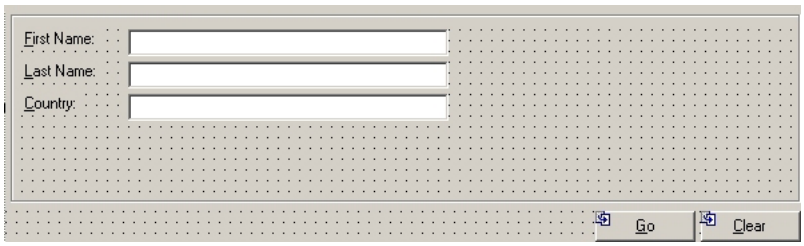
1.2.1 Creation of the *SelectionPerson*-class

In our scenario, the base class for *SelectionPerson* ist *vdxSelection* and not *vdxSelectionTab*. Since the person-Searchdialog is implemented on a single page, we don't need any tabpages.

Add following controls on the *SelectionPerson*-class:

Added Control	Text	Name
vdxLabel	First Name	lblFirstName
vdxLabel	Last Name	lblLastName
vdxLabel	Country	lblCountry
vdxTextBox		txtFirstName
vdxTextBox		txtLastName
vdxTextBox		txtCountry

The layout of the searchdialog looks similar to this:



1.2.2 Bind DataSet *dsPersList* to *vdxDataGrid*

Obviously the DataSet *dsPersList* must be bound to the *vdxDataGrid* class *SelectionListPerson*. Make sure to rename the DataSet from *dsPersList1* to *dsPersList*. Following table contains the corresponding properties of the class *SelectionListPerson*:

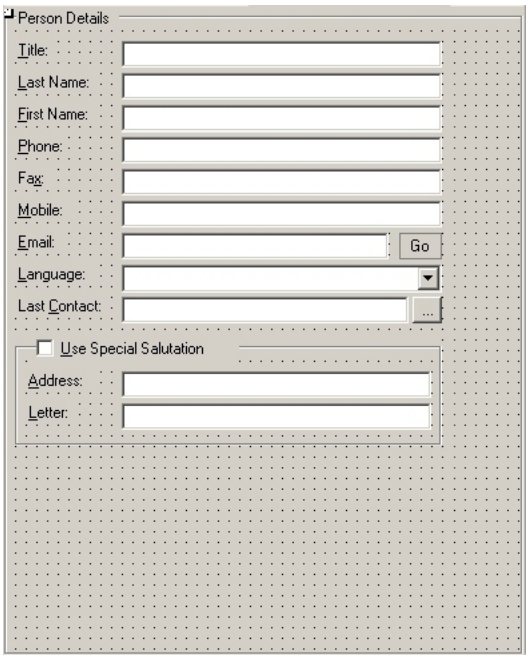
Property	Selection
DataSource	DsPersList
DataMember	Pers

1.2.3 Creation of the *DataDetailPerson*-class

Add the following controls to the *DataDetailPerson*-class:

Control to be added	Text	Name
VdxLabel	Title	lblTitle
VdxLabel	Last Name	lblLastName
VdxLabel	First Name	lblFirstName
VdxLabel	Sex	lblSex
VdxLabel	Phone	lblPhone
VdxLabel	Fax	lblFax
VdxLabel	Mobile	lblMobile
VdxLabel	E-Mail	lblEmail
VdxLabel	Language	lblLanguage
VdxLabel	Last Contact	lblLastContact
VdxLabel	Address	lblAddress
VdxLabel	Letter	lblLetter
VdxButton	Go	btnGoEmail
VdxTextBox		txtTitle
VdxTextBox		txtLastName
VdxTextBox		txtFirstName
VdxTextBox		txtPhone
VdxTextBox		txtFax
VdxTextBox		txtMobile
VdxTextBox		txtEmail
VdxTextBox		txtAddress
VdxTextBox		txtLetter
VdxDateTimePicker		dateTimePicker
VdxComboBox		cboLanguage
VdxCheckBox	Use Special Salutation	chkUseSpecSal

Layout the controls as follows:



1.2.4 DataSet *dsPers* and *dsLang* binding

The *dsPers*-DataSet must be bound to the corresponding controls on the *DataDetailPerson* class. Following table describes the bindings:

Control	Property DataBindings.Text
txtTitle	dsPers – Pers.Title
txtLastName	dsPers – Pers.LastName
txtFirstName	dsPers – Pers.FirstName
txtPhone	dsPers – Pers.Phone
txtFax	dsPers – Pers.Fax
txtEmail	dsPers – Pers.EMail
txtAddress	dsPers – Pers.SAL_Address
txtLetter	dsPers – Pers.SAL_Letter
chkUseSpecSal	dsPers – Pers.UseSpecSal
dateTimePicker	dsPers – Pers.LastContact

The *dsLang*-DataSet must be bound to the *cboLanguage*-ComboBox control:

Property	Control cboLanguage
SelectedValue	dsPers – Pers.Lang
DataSource	dsLang.Lang
Display Member	Descr
Value Member	Lang

To fill the *Language*-ComboBox with the available languages, add following code in the constructor of the *DataDetailPerson*-class; add it after the *InitializeComponent*-call as follows:

```
vdxSampleWebServices.BoLangSoap boLang = new vdxSampleWebServices.BoLangSoap();
dsLang.Merge(boLang.GetDataSetLang());
```

1.2.5 Configuratioin of the *vdxDataStatusManager*

Set the properties of the *vdxDataStatusManager*-object to the following:

Property	Selection
VdxBindingContainer	DataDetailPerson
VdxDataSet	dsPers
VdxBusinessObjectClassName	[MyVdxSample…]BoPersSoap
VdxMainDataTable	dsPers.Pers
VdxUpdateDataSetMethodName	UpdateDataSet

1.2.6 Overwriting the *VdxFillSearchParameters*-method

Add the following code to the VdxFillSearchParameters method:

```
Deag.Vdx.Common.vdxSearchParameters sp;
sp = new vdxSearchParameters("SelectionPersByDescr");

sp.Add(new vdxSearchParameter("FirstName",this.txtFirstName.Text,
   vdxSearchType.Equal));

sp.Add(new vdxSearchParameter("LastName",this.txtLastName.Text,
   vdxSearchType.Equal));

sp.Add(new vdxSearchParameter("Country",this.txtCountry.Text,
   vdxSearchType.Equal));

sp.Add(new vdxSearchParameter("OrderBy","LastName",vdxSearchType.Equal));

return sp;
```

1.2.7 Configuring the ExplorerAddress-control

Following table describes the three most important properties:

Property	Selection
VdxSelectionControlClassName	MyVdxSampleApplication.SelectionPerson
VdxSelectionListControlClassName	MyVdxSampleApplication.SelectionListPerson
VdxMainDataDetailControlClassName	MyVdxSampleApplication.DataDetailPerson

1.2.8 Overwriting the *VdxLoadData*-method

Add following code to the *VdxLoadData* method:

```
DataSet dsRet = null;
DataSet ds = null;
vdxSampleWebServices.BoPersSoap boPers = new vdxSampleWebServices.BoPersSoap();

switch(dataLoadContext)
{
   case vdxDataLoadContext.Selection:
      ds = boPers.GetDataSetPersList(vdxSearchParameters.GetParameters());
      if (ds.Tables["Pers"].Rows.Count > 0)
      {
         dsRet = ds;
      }
      else
```

```
      {
         MessageBox.Show("No Data found. Please try again!",
            "VDX Sample Application",MessageBoxButtons.OK,
            MessageBoxIcon.Information);
      }
      break;

   case vdxDataLoadContext.SubSelection:
      break;

   case vdxDataLoadContext.Detail:
      ds = boPers.GetDataSetPers(vdxSearchParameters.GetParameters());
      if (ds.Tables["Pers"].Rows.Count > 0)
      {
         dsRet = ds;
      }
      break;
}

return dsRet;
```

1.2.9 Overwriting the *VdxFillData*-method

Add following code to the *VdxFillData* method:

```
switch(dataLoadContext)
{
   case vdxDataLoadContext.Selection:
      nodeToFill.Nodes.Clear();
      nodeToFill.IsDummy = false;
      break;

   case vdxDataLoadContext.Detail:
      break;

   case vdxDataLoadContext.SubSelection:
      nodeToFill.Nodes.Clear();
      nodeToFill.IsDummy = false;
      break;

   default:
      break;
}

if (dataLoadContext == vdxDataLoadContext.Selection | dataLoadContext ==
vdxDataLoadContext.SubSelection)
{
   foreach (DataRow rw in dataSet.Tables["Pers"].Rows)
   {
      string lastName = "";
      string firstName = "";
      if (!Convert.IsDBNull(rw["LastName"]))
      {
         lastName = rw["LastName"].ToString();
      }
      if (!Convert.IsDBNull(rw["FirstName"]))
      {
         firstName = rw["FirstName"].ToString();
      }

      vdxSearchParameters vdxSearchParameters = new vdxSearchParameters(
         "Person");

      vdxSearchParameters.Add(new vdxSearchParameter("PersId",
         ((int)rw["PersId"]),Deag.Vdx.Common.Enums.vdxSearchType.Equal));
```

```
      nodeToFill.Nodes.Add(new vdxTreeNode(lastName + " " + firstName,3,0,false,
         vdxSearchParameters,"MyVdxSampleApplication.DataDetailPerson"));
   }
}
```

1.2.10 Overwrite the *VdxUpdateNode*-method

Add following code to the *VdxUpdateNode*-method:

```
switch (nodeToUpdate.TreeNodeType)
{
   case "MyVdxSampleApplication.DataDetailPerson":
      string lastName = "";
      string firstName = "";
      if (!Convert.IsDBNull(rw["LastName"]))
      {
         lastName = rw["LastName"].ToString();
      }
      if (!Convert.IsDBNull(rw["FirstName"]))
      {
         firstName = rw["FirstName"].ToString();
      }
      nodeToUpdate.Text= lastName + " " + firstName;
      nodeToUpdate.ImageIndex = 3;
      nodeToUpdate.VdxSearchParameters = new vdxSearchParameters("Person");
      nodeToUpdate.VdxSearchParameters.Add(new vdxSearchParameter("PersId",
      ((int)rw["PersId"]), Deag.Vdx.Common.Enums.vdxSearchType.Equal));
      break;

   default:
      break;
}
```

1.2.11 Overwrite the *VdxUpdateSelectionList*-method

Add following code:

```
targetValues = base.VdxDataRowToSelectionListValueArray(rw);
base.VdxUpdateSelectionList(targetValues, rw);
```

1.2.12 Programming the *vdxActionItem*

Add the following code to the *aiAddressExplore_vdxActionInvoked*-Event-Listener-method:

```
this.vdxMultiContainer.VdxCurrentControlType = typeof(ExplorerPerson);
((vdxOutlookDetail)this.vdxMultiContainer.VdxCurrentControl).Show();
```

1.3 Linking with the adress data-management

After completion of the person scenario, in the following, the additional steps are described to link the person scenario to the address scenario...

1. Open the Code-View of the class *ExplorerAddress*. Complete the method *VdxLoadData* with the following code (the grey code has already ben implemented in the standard address data-manipulation):

   ```
   DataSet dsRet = null, ds = null;
   ```

```
vdxSampleWebServices.BoAddressSoap boAddress;
vdxSampleWebServices.BoPersSoap boPers;

switch(dataLoadContext)
{
   // Lade die Adress-Treedaten.
   case vdxDataLoadContext.Selection:
      boAddress = new vdxSampleWebServices.BoAddressSoap();
      ds = boAddress.GetDataSetAddressList(vdxSearchParameters.
         GetParameters());

      if (ds.Tables["Address"].Rows.Count > 0)
      {
         dsRet = ds;
      }
      else
      {
         MessageBox.Show("No Data found. Please try again!",
            "VDX Sample Application",MessageBoxButtons.OK,
            MessageBoxIcon.Information);
      }
      break;

   // Lade die Person-Subnodes.
   case vdxDataLoadContext.SubSelection:
      boAddress = new vdxSampleWebServices.BoAddressSoap();
      ds = boAddress.GetDataSetAddressList(vdxSearchParameters.
         GetParameters("PersTree"));

      if (ds.Tables["Pers"].Rows.Count > 0)
      {
         dsRet = ds;
      }
      break;

   // Lade die Datendetails der entsprechenden Adresse oder Person.
   case vdxDataLoadContext.Detail:
      switch (vdxSearchParameters.VdxSearchName)
      {
         case "Address":
            boAddress = new vdxSampleWebServices.BoAddressSoap();
            ds = boAddress.GetDataSetAddress(vdxSearchParameters.
               GetParameters());

            if (ds.Tables["Address"].Rows.Count > 0)
            {
               dsRet = ds;
            }
            break;

         case "Person":
            boPers = new vdxSampleWebServices.BoPersSoap();
            ds = boPers.GetDataSetPers(vdxSearchParameters.
               GetParameters());

            if (ds.Tables["Pers"].Rows.Count > 0)
            {
               dsRet = ds;
            }
            break;
      }
      break;
}

return dsRet;
```

In the first *case*-block when expanding the adress-tree-node, the corresponding persons will be loaded. The second *case*-block loads the detaildata of a specific person, when clicking on a person node within the address-treeview.

2. Next, we complete the *vdxFillData* method with following code:

```
if (dataLoadContext == vdxDataLoadContext.Selection | dataLoadContext ==
vdxDataLoadContext.SubSelection)
{
   nodeToFill.Nodes.Clear();
   nodeToFill.IsDummy = false;

   foreach(DataTable dataTable in dataSet.Tables)
   {
      switch (dataTable.TableName)
      {
         case "Address":
            foreach (DataRow rw in dataTable.Rows)
            {
               string company = "";
               if (!Convert.IsDBNull(rw["Company"]))
               {
                  company = (string)rw["Company"];
               }
               vdxSearchParameters sp = new
                  vdxSearchParameters("Address");

               sp.Add(new vdxSearchParameter("AddressId",
                  ((int)rw["AddressId"]),vdxSearchType.Equal));

               nodeToFill.Nodes.Add(new vdxTreeNode(company,2,0,true,sp,
                  "MyVdxSampleApplication.DataDetailAddress"));
            }
            break;

         case "Pers":
            foreach (DataRow rw in dataTable.Rows)
            {
               string lastName = "", firstName = "";
               if (!Convert.IsDBNull(rw["LastName"]))
               {
                  lastName = (string)rw["LastName"];
               }
               if (!Convert.IsDBNull(rw["FirstName"]))
               {
                  firstName = (string)rw["FirstName"];
               }
               vdxSearchParameters sp = new
                  vdxSearchParameters("Person");

               sp.Add(new vdxSearchParameter("PersId",
                  ((int)rw["PersId"]),vdxSearchType.Equal));

               nodeToFill.Nodes.Add(new vdxTreeNode(lastName + " " +
                  firstName,3,0,false,sp,
                  "MyVdxSampleApplication.DataDetailPerson"));
            }
            break;
      }
   }
}
```

This additional *case*-block fills the treeview with the corresponding person-data.

3. To update a person-tree-node when updating datail data of the corresponding person, we add following code to the *vdxUpdateNode* method:

```
switch (nodeToUpdate.TreeNodeType)
{
   case "MyVdxSampleApplication.DataDetailAddress":
      string company = "";
      if (!Convert.IsDBNull(dataRowFromDataDetailControl["Company"]))
      {
         company = (string)dataRowFromDataDetailControl["Company"];
      }
      nodeToUpdate.Text = company;

      // Falls die Drag'n Drop-Funktionalität von Tree-Knoten auf die
      // Outlookbar aktiviert ist, dann sollten jeweils die Suchparameter
      // nach einem Knoten-Update neu gesetzt werden. Damit können auch
      // neu eingefügte Tree-Knoten per Drag'n Drop auf die Outlookbar
      // gezogen werden.
      nodeToUpdate.VdxSearchParameters = new
         vdxSearchParameters("Address");
      nodeToUpdate.VdxSearchParameters.Add(new vdxSearchParameter(
         "AddressId",((int)dataRowFromDataDetailControl["AddressId"]),
         vdxSearchType.Equal));
      break;

   case "MyVdxSampleApplication.DataDetailPerson":
      string lastName = "", firstName = "";
      if (!Convert.IsDBNull(dataRowFromDataDetailControl["LastName"]))
      {
         lastName = dataRowFromDataDetailControl["LastName"].ToString();
      }
      if (!Convert.IsDBNull(dataRowFromDataDetailControl["FirstName"]))
      {
         firstName = dataRowFromDataDetailControl["FirstName"].ToString();
      }
      nodeToUpdate.Text = lastName + " " + firstName;
      nodeToUpdate.VdxSearchParameters = new
         vdxSearchParameters("Person");
      nodeToUpdate.VdxSearchParameters.Add(new vdxSearchParameter(
         "PersId",((int)dataRowFromDataDetailControl["PersId"]),
         vdxSearchType.Equal));
      break;

   default:
      break;
}
```

This completes the integration of the person data-manipulation with the the address scenario. In the Tutorial "vdxPickField – Foreign Key Validation with Picklist functionality" an additional control will be added to allow the selection of an address to a person; this completes the person data-management. The persons now are true "Subnodes" of the corresponding addresses.