Visual.NET Extensions – Tutorial

"vdxPickField – Foreign Key Validation with Picklist functionality"

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

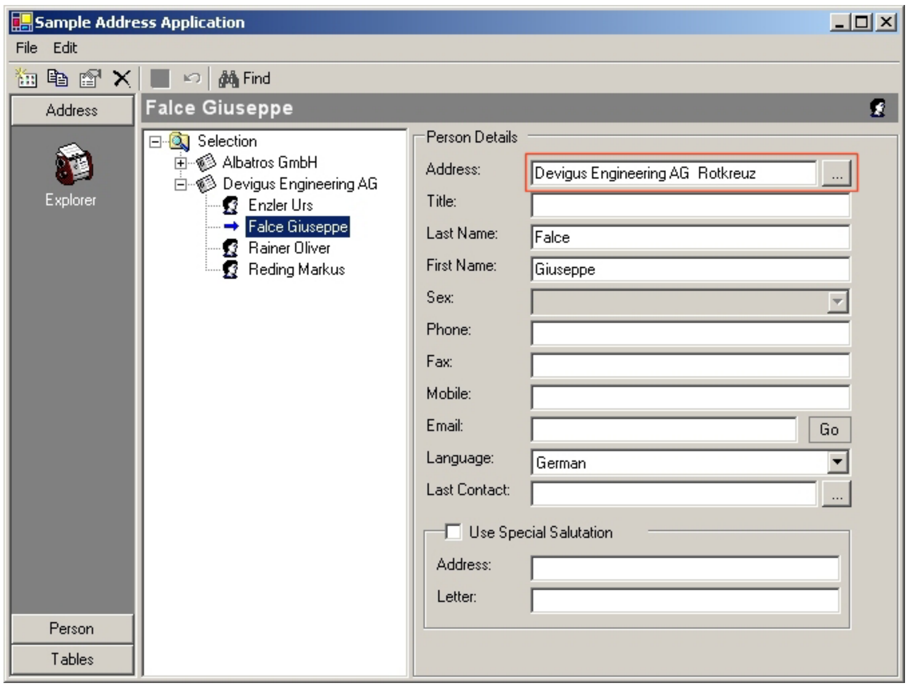
Devigus Engineering AG
Grundstrasse 3
CH-6343 Rotkreuz
Internet: http://www.devigus.com
Email: deag@devigus.com

Version: 1.1
Last Update: 11.11.2002

1 *vdxPickField – Foreign Key Validation with Picklist functionality.*

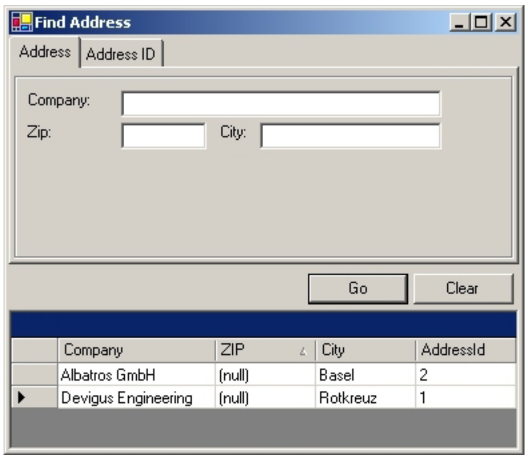
This tutorial describes the usage of the *vdxPickField*-control. The pickfield control allows foreign key validation and picklist functionality. In this tutorial, a sample pickfield scenario, where an address can be picked for a person, will be developed.

In order to allow a person to be linked to an adress, a corresponding pickfield control must be added to the *DataDetailPerson* (see also *vdxExplorerList*-Tutorial). There, the edited person can be linked to a specific address. Following screen shows the pickfield, which will be implemented in this tutorial. The pickfield control is located at the top of the user control (highlighted with a red border in the screenshot below) and by clicking the three-dot *browse* button, the picklist can be called:



Address:	Devigus Engineering AG Rotkreuz
Title:	
Last Name:	Falce
First Name:	Giuseppe
Sex:	
Phone:	
Fax:	
Mobile:	
Email:	
Language:	German
Last Contact:	

This is the picklist dialog, where an address can be selected using a selection dialog and a list showing the fetched records from within the desired address can be selected by doubleclicking the desired row at the left border of the grid (arrow in below screenshot).:



Company	ZIP	City	AddressId
Albatros GmbH	(null)	Basel	2
Devigus Engineering	(null)	Rotkreuz	1

Since the classes *SelectionAddress* (above selection dialog) as well as *SelectionListAddress* (list showing matching addresses) already exists (these are exactly the same as for the regular address datamanagement!) , the implementation of the *vdxPickField*-control remains rather simple.

1.1 Creation of the PickAddress-class

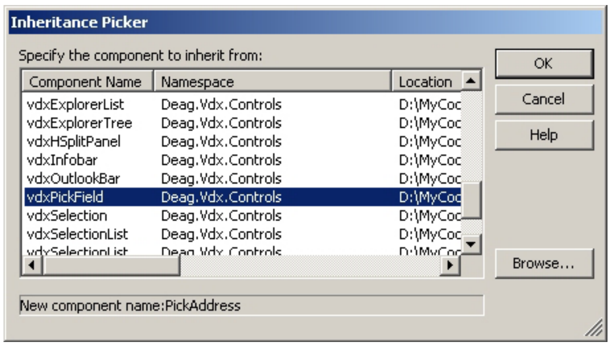
Following steps describe the creation of the *PickAddress*-class resp. the inheritance from the base class *vdxPickField*:

1. Add a new, derived *Control*-class to your project. Therefore select *Add Inherited Control...* from the *Project*-Menu.

2. Rename the class to *PickAddress*.

3. Click *Open* to access the dialog for the base class selection.

4. Click the *Browse*-button to select the corresponding *Assembly* (resp. dll-file, the Assembly is located in the VDX-Installationdirectory under *vdxControls\bin\release\*). Therein select the file *vdxControls.dll*. From the *component list to inherit from* select the class *vdxPickField* and click the *OK*-button. The following screen shows the *inheritance picker* in action:

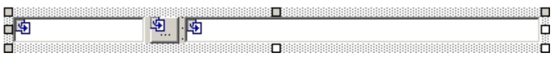


Specify the component to inherit from:

Component Name	Namespace	Location
vdxExplorerList	Deag.Vdx.Controls	D:\MyCoc
vdxExplorerTree	Deag.Vdx.Controls	D:\MyCoc
vdxHSplitPanel	Deag.Vdx.Controls	D:\MyCoc
vdxInfobar	Deag.Vdx.Controls	D:\MyCoc
vdxOutlookBar	Deag.Vdx.Controls	D:\MyCoc
vdxPickField	Deag.Vdx.Controls	D:\MyCoc
vdxSelection	Deag.Vdx.Controls	D:\MyCoc
vdxSelectionList	Deag.Vdx.Controls	D:\MyCoc
vdxSelectionList	Deag.Vdx.Controls	D:\MyCoc

New component name:PickAddress

5. The newly created class contains already a *Code*-field, a *Pickfield Selector* (the three-dot button) and a *Description*-field.



6. Insert the following VDX namespace references at the top of the code to allow easier access to the VDX classes.

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Common.Exceptions.LocalExceptions;
using Deag.Vdx.Controls;
using Deag.Vdx.Controls.EventArguments;
```

1.2 *Enhancing the SelectionAddress-class*

To allow the *PickAddress*-control to access the values of the Textboxes of the Adress-Selection dialog, these values must be made available in the class *SelectionAddress* as *Properties* (see also following chapter). This step is more elegant than making the textboxes public. Follow these steps:

1. Open the Code-View of the class *SelectionAddress* (the creation of this class is described in the tutorial *vdxExplorerTree – Standard datamanipulation with treeview*).

2. Add following two *Properties* to the class:

```
[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public string AddressId
{
   get{return this.txtAddressId.Text;}
   set{this.txtAddressId.Text = value;}
}

[DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
public string Company
{
   get{return this.txtCompany.Text;}
   set{this.txtCompany.Text = value;}
}
```

NOTE: Eventually, you must modify the name of your textboxes, if you named them differently in your *SelectionAddress*-class.

1.3 *Implementing the PickAddress-class*

In the following sections, all needed methods which must be overridden from the base class are described:

1.3.1 Overwriting the *VdxPickFormActivated*-method

Whenever the user clicks the *Browse*-button to open the Address-Pickfield dialog, the method *VdxPickFormActivated* of the baseclass will be called. To allow to pass *Code*- or *Description*-field values to the Adressen-Searchdialog (where the actual data validation occurrs) these values have to be passed to the selection dialog. To implement this functionality, follow these steps:

1. Select the *Class View* of the project (via Menu *View* → *Class View*).

2. Navigate to the *VdxPickFormActivated*-method; it is located in the class *vdxPickField* (*MyVdxSampleApplication* → *PickAddress* → *Bases and Interfaces* → *vdxPickField*).

3. Select *Add* → *Override* from the contextmenu of the *VdxPickFormActivated*-method. This adds an empty method to the *PickAddress*-class, overwriting the base class method.

4. Add following code to the method:

```
SelectionAddress selAddress = this.pickForm.VdxSelectionControl as
   SelectionAddress;

if (selAddress != null)
{
   selAddress.AddressId = code;
   selAddress.Company = "";
}
```

This way, an existing content from the *PickAdress*-control will be passed to the search dialog and written to the textboxes of the selection dialog.

1.3.2 Overwriting the *VdxLoadData*-method

When the user clicks on the *Go*-Button of the selection dialog, the corresponding data loading gets invoked. The data loading will be implemented – as in the class *ExplorerAddress* (see Tutorial "vdxExplorerTree - Standard datamanipulation with treeview") – in the method *VdxLoadData*. To implement this functionality, follow these steps:

1. Select the *Class View* of your project (via Menu *View* → *Class View*).

2. Navigate to the *VdxLoadData*-method; it is also located in the class *vdxPickField*.

3. Select *Add* → *Override* from the contextmenu of the *VdxLoadData*-method. This adds an empty method to the *PickAddress*-class, overwriting the one from the baseclass.

4. Remove the *return null*-statement and add following code to the method:

```
vdxSampleWebServices.BoAddressSoap boAddress = new vdxSampleWebServices.
   BoAddressSoap();

return boAddress.GetDataSetAddressList(vdxSearchParameters.
   GetParameters());
```

NOTE: The parameter passing uses a *vdxSearchParameters*-object with the name *SelectionAddress* or *SelectionAddressId*, depending on the type of selection the user invoked. A *vdxSearchParameters*-object contains one or several *vdxSearchParameter*-objects, whereas every single *vdxSearchParameter* object holds specific search criteria data. With the *Add*-method such an object will be added to the *vdxSearchParameters*-Object.

For more information regarding the *vdxSearchParameters* class consult the *vdxTechRef.chm* helpfile!

1.3.3 Overwriting the *VdxDataRowSelected*-method

When the pickfield dialog presents the selected records, the user can select the desired address by doubleclicking (on the left border within the grid where the record arrow is); this closes the address selection dialog and the address description will be written in the description-field of the *PickAddress*-control. To implement this functionality, follow these steps:

1. Select *Class View* in your project (via Menu *View* → *Class View*).

2. Navigate to the *VdxDataRowSelected*-method; it is also located in the class *vdxPickField*.

3. Select *Add* → *Override* from the contextmenu of the *VdxDataRowSelected*-method. This adds an empty method in the die *PickAddress*-class, overwriting the one from the baseclass.

4. Add following code to the method:

```
string strCompany = "";
string strZip = "";
string strCity = "";

if (!Convert.IsDBNull(e.DataRow["Company"]))
{
   strCompany = e.DataRow["Company"].ToString();
}

if (!Convert.IsDBNull(e.DataRow["ZIP"]))
{
   strZip = e.DataRow["ZIP"].ToString();
}

if (!Convert.IsDBNull(e.DataRow["City"]))
{
   strCity = e.DataRow["City"].ToString();
}

this.VdxPickCode = e.DataRow["AddressId"];
this.VdxPickDescription = strCompany + " " + strZip + " " + strCity;
base.VdxDataRowSelected(sender, e);
```

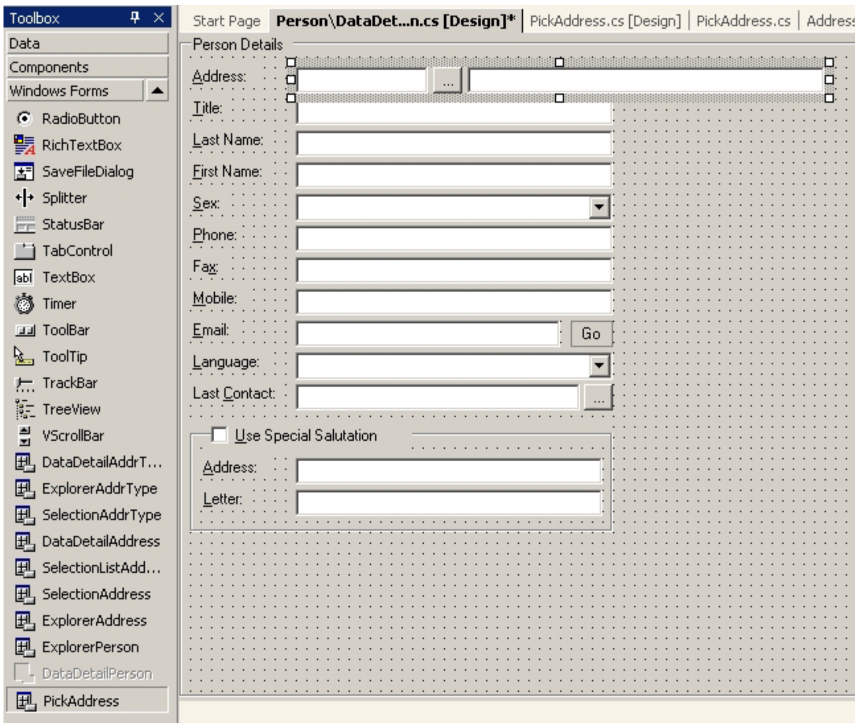
After reading the corresponding address data from the *vdxDataRowSelectedEventArgs*-parameter, the *Code*- and the *Description*-fields will be filled with the corresponding values.

1.4 Integration of the PickAddress-class

After the creation of the needed *Properties* and *Methods*, the integration of the *PickAddress*-control in the *DataDetailPerson*-class has to be implemented. Following steps describe this procedure:

1. Compile the project and open the Design-View of the class *DataDetailPerson*.

2. On your toolbox open the register *Windows Forms* an drag the *PickAddress*-control on the Design-Area of the *DataDetailPerson* class. Additionally place a lable control left to the pickfield holding the description *Address*. Following screen shows the Design-Area of the class *DataDetailPerson* with the new *PickAddress*-control:



NOTE: If the PickAddress-control is not visible in your *Windows Forms*-Register of the Toolbox, close the Solution. If you reopen the solution, the *PickAddress*-control should be visible on the Toolbox. Another solution is through *Customize Toolbox...* from the contextmenu of the Toolbox. Therein select as the Assembly the file *MyVdxSampleApplication.exe* from the *\bin\Debug*-Directory of your solution.

3. Select the *PickAddress*-control and set the *VdxCodeHide*-property to *true*. This hides the code field (we just want to see the description of the address, altough we loos the possibility that the user snters an address-id directly on in the code filed, without calling the pickfield-dialog).

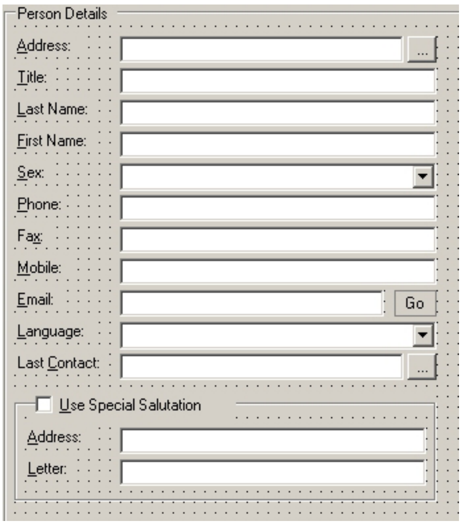
4. Set the *VdxSelectionControlClassName*-property to the value *SelectionAddress* resp. *MyVdxSampleApplication.SelectionAddress*.

5. Set the *VdxSelectionListControlClassName*-property to the value *SelectionListAddress* resp. *MyVdxSampleApplication.SelectionListAddress*.

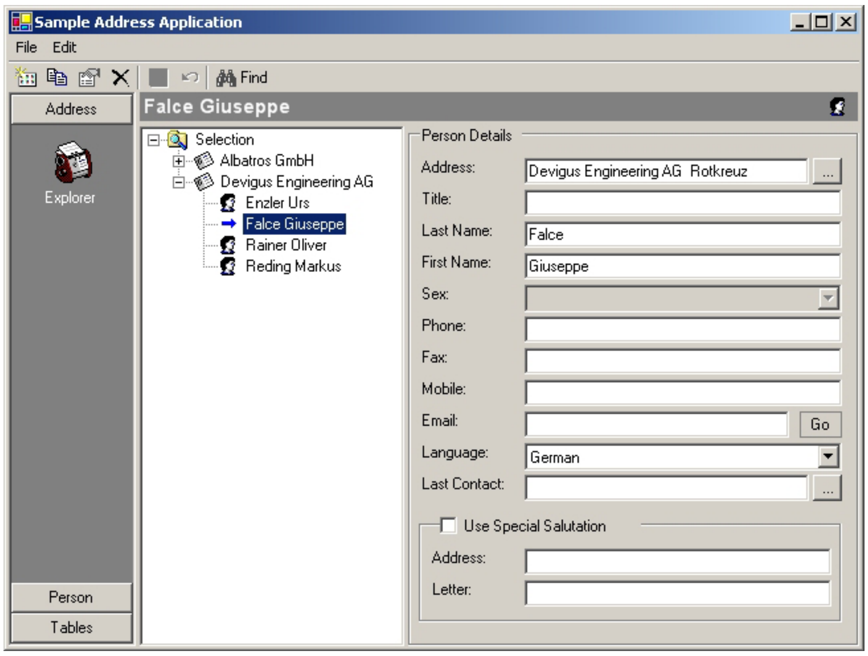
6. To allow the *PickAddress*-control to work with the corresponding data, it must be linked with the DataSet *dsPers*. Set the following *DataBinding*-Properties to the corresponding values:

Property	Selektion
(DataBindings).VdxPickCode	dsPers - Pers.Addressid
(DataBindings).VdxPickDescription	dsPers - Pers.ADDRESSDESCR

The Design-View of the class *DataDetailPerson* should now be similar to the following form.



Compile and start your application. Now, you can link a person to a specific address. In the context of the *vdxExplorerTree – Extended Embedded Case*-Tutorial (Person Datamanagement linked with Address Datamanagement), the persons will be shown as subnodes of the corresponding address node:


Sample Address Application

Falce Giuseppe

- Selection
  - Albatros GmbH
  - Devigus Engineering AG
    - Enzler Urs
    - Falce Giuseppe
    - Rainer Oliver
    - Reding Markus

Person Details

Address:	Devigus Engineering AG Rotkreuz
Title:	
Last Name:	Falce
First Name:	Giuseppe
Sex:	
Phone:	
Fax:	
Mobile:	
Email:	
Language:	German
Last Contact:	

Use Special Salutation

Address:	
Letter:	