Visual.NET Extensions – Tutorial

“vdxExplorerTree - Standard 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: 17.11.2003

Inhaltsverzeichnis

1 *vdxExplorerTree* - Standard Datenmanipulation mit Treeview

Dieses Tutorial beschreibt den Einsatz des *vdxExplorerTree*-Controls bei der Implementation eines Standard-Datenmanipulations-Szenarios. Ein solches Szenario kann z.B. bei der Verwaltung von Adressen inkl. Treeview Navigation angewendet werden.

Wie bei allen Tutorials, wird anhand der Beispiel-Applikation (Adressverwaltung) ein konkreter Anwendungsfall implementiert. Dieser beschreibt exemplarisch die Verwendung der *vdxExplorerTree*-Klasse. Die Abbildung 1 zeigt das Ergebnis dieses Tutorials mit dem *vdxExplorerTree*-Control, seiner Child-Controls *SelectionListAddress* und des *DataDetailAddress*:

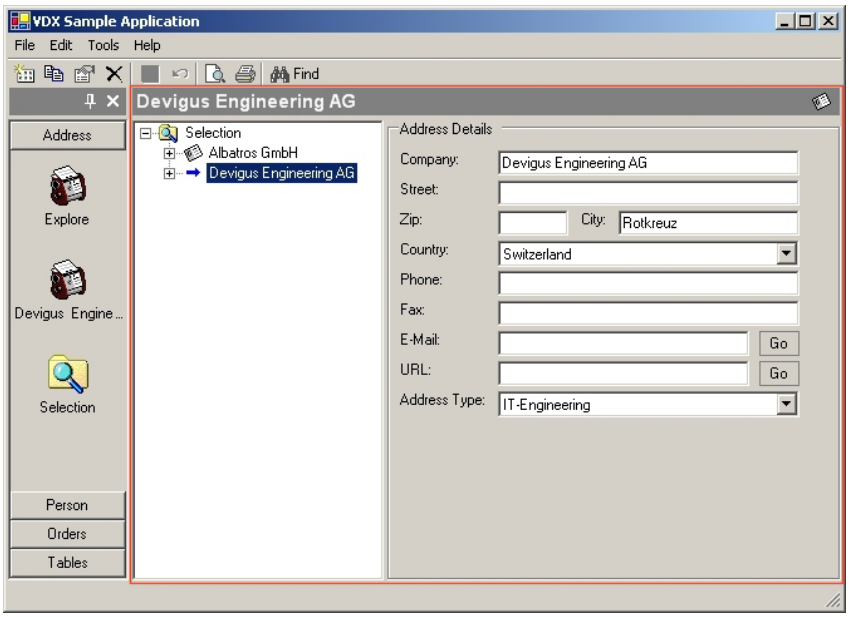


Abbildung 1 Sample Application: Adressendetail

1.1 Definition der Namespaces

Die folgende Tabelle zeigt die in diesem Tutorial verwendeten Bezeichnungen. Die Platzhalter stehen für die konfigurierten Projektnamen und Namespaces im ApplicationWizard.

Platzhalter	Beispiel	Beschreibung
<ServerProjectName>	mySampleServer	Name des Server-Projektes
<ClientProjectName>	mySampleClient	Name des Client-Projektes
<ClientProjectNamespace>	mySampleClient	Default-Namespace für Client-Projekt
<WebServiceName>	mySampleWebServices	Name des WebServices.

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:
   - wsCountry
   - wsAddrType
   - wsAddress

   Siehe dazu Kapitel 1.4, 1.5 und 1.6 des Server Tutorials *VDXTut_Server_de*.
3. Schliessen Sie alle Visual Studio Applikationen. Dadurch vermeiden Sie Probleme, welche bei der Initialisierung der vdxLanguage-Komponente auftreten könnten.
4. Öffnen Sie Client-Solution *mySampleClient.sln*.
5. Aktualisieren Sie die Web-Referenzen über das Kontext-Menü von *mySampleWebServices*.

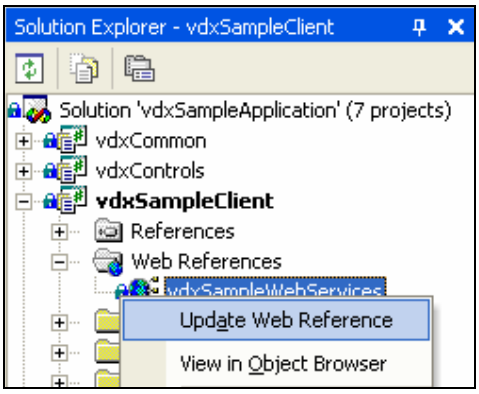


Abbildung 2 Kontext-Menü einer Web Referenz im Solution Explorer

6. Damit man *typensicher* auf die Web Services zugreifen kann, sollte die Klasse *WebServiceProvider* mit folgenden Properties ergänzt werden.

```
static public wsAddressSoap wsAddress
{
    get { return (wsAddressSoap)GetWebService(typeof(wsAddressSoap)); }
}

static public wsAddrTypeSoap wsAddrType
{
    get { return (wsAddrTypeSoap)GetWebService(typeof(wsAddrTypeSoap)); }
}

static public wsCountrySoap wsCountry
{
    get { return (wsCountrySoap)GetWebService(typeof(wsCountrySoap)); }
}
```

Codesegment 1 Properties von WebServiceProvider ergänzen

7. Compilieren Sie die Client-Solution erneut (Rebuild).

1.3 Erstellen der ExplorerAddress-Klasse (Container-Klasse)

In diesem Kapitel wird die Container-Klasse erstellt. In diesem Container werden die

- Adressensuche (*SelectionAddress*-Control),
- die gefundenen Adressen (*SelectionGridAddress*-Control) sowie die
- Adressendetails (*DataDetailAddress*-Control)

eingebettet. Zudem wird die Datenbezugs-Logik der eingebetteten Controls in dieser Klasse ausprogrammiert. Nur dort, wo es technisch nicht möglich oder sinnvoll ist, wird eine Methode direkt in die Control-Klasse implementiert.

Folgende Schritte zeigen das Erstellen der *ExplorerAddress*-Klasse:

1. Legen (vgl. Abbildung 3) Sie im Solution Explorer ein neues Verzeichnis *Address* an.

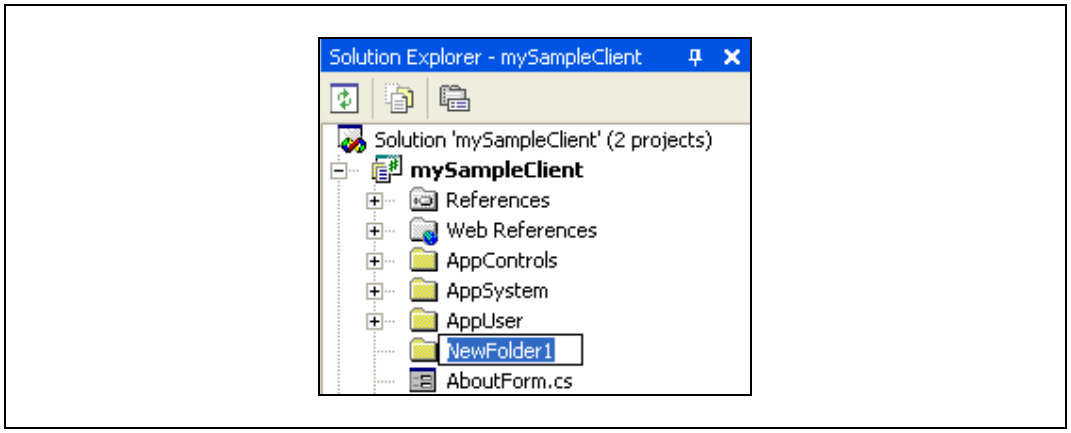


Abbildung 3 Neuer Ordner anlegen im Solution Explorer

Fügen Sie dem Ordner Address eine neue, abgeleitete *Control*-Klasse hinzu. Wählen Sie dazu den Befehl *Add Inherited Control...* aus dem *Project*-Menu aus.

2. Benennen Sie die Klasse *ExplorerAddress*; wir nennen sie *ExplorerAddress*, weil alle in ihr eingebetteten Controls dem *Exploring* bzw. dem Durchsuchen und Bearbeiten von Adressdaten dienen.

3. Klicken Sie auf *Open* um den Dialog zur Auswahl der Basisklasse zu öffnen.

4. Im nachfolgenden Dialog wählen sie dann die Klasse *AppExplorerTree* als Basis für *ExplorerAddress* aus. Die Abbildung 4 zeigt das Auswahl-Fenster:

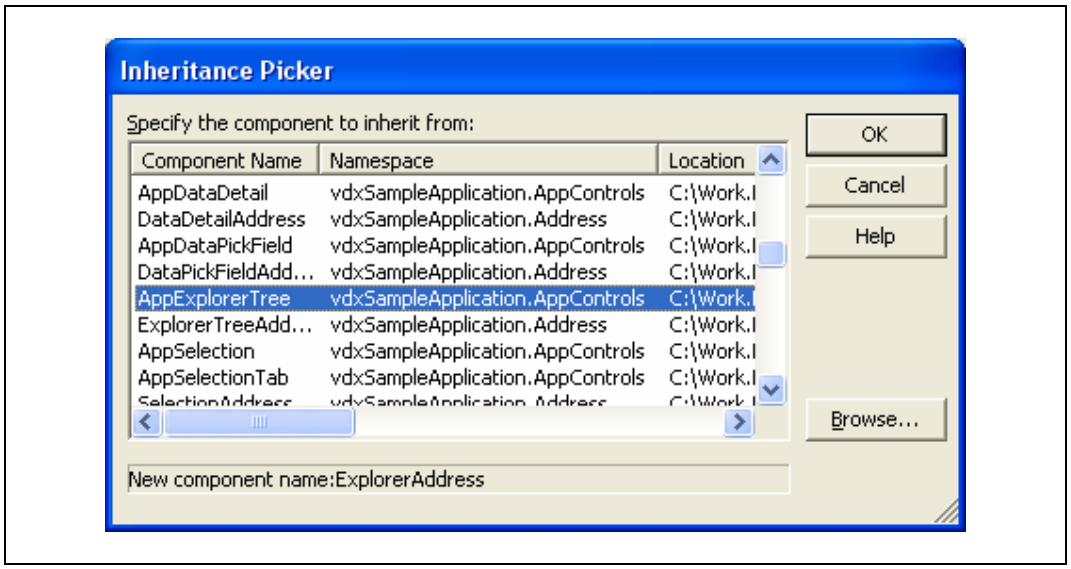


Abbildung 4 Auswahl der Basisklasse

5. Die neu erstellte Klasse (Abbildung 5) besteht durch die Ableitung bereits aus einem *TreeView* und einem Container für die entsprechenden Controls:

- *SelectionAddress*,
- *SelectionGridAddress* oder
- *DataDetailAddress*

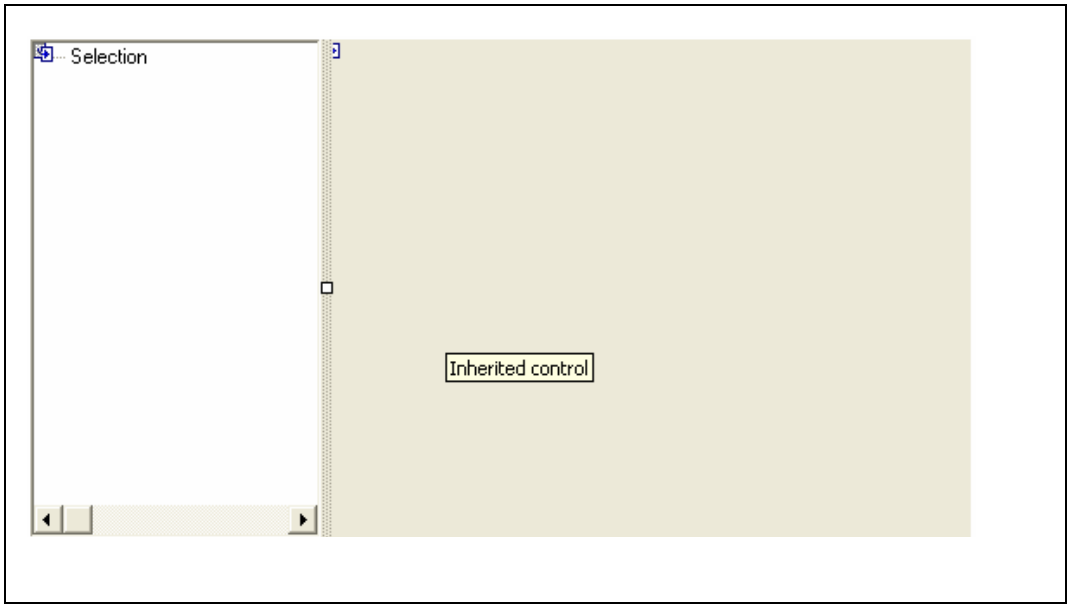


Abbildung 5 Design-Ansicht eines ExplorerTrees

6. Fügen Sie aus dem *Windows Form*-Register der ToolBox eine ImageList hinzu und benennen diese imageList. Anschliessend weisen Sie die erstellte imageList dem treeView-ImageList-Property zu.

7. Fügen Sie die folgenden *Images* zur Liste hinzu; die Icons finden Sie im VDX-Installationsverzeichnis unter *icons\treeView\small*. Die Tabelle 2 zeigt die Reihenfolge der Icons innerhalb der *Images Collection* auf:

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

Tabelle 2 ImageList des ExplorerAddress

8. Wechseln Sie, wie in Abbildung 6 gezeigt, in die Code-Ansicht.

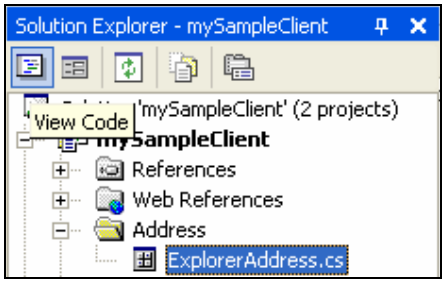


Abbildung 6 Wechseln zu Code-Ansicht

Tipp

Wechseln zur Code-Ansicht.

- F7 eingeben oder
- Button *View Code* in der Solution Browser ToolBar klicken oder
- über das Menu *View > Code*

Damit Sie komfortabel (Benutzung der Codecompletion!) auf die VDX-Klassen zugreifen können, fügen Sie die im Codesegment 1 angegebene VDX-Namespace-Referenzen ganz oben in den Code ein:

```
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 2 Using-Anweisung für ExplorerAddress

In dieser Klasse wird später die Hauptfunktionalität implementiert. In den folgenden Kapiteln werden zuvor die Klassen *SelectionAddress*, *SelectionGridAddress* und *DataDetailAddress* erstellt.

1.4 Erstellen der SelectionAddress-Klasse

In diesem Kapitel wird die Control-Klasse erstellt, in der die Benutzer die Suchbegriffe für Adressen eingeben können. Folgende Schritte zeigen das Erstellen dieser Control-Klasse:

1. Fügen Sie eine neues *Inherited User Control* hinzu. Benennen Sie die Klasse *SelectionAddress* und leiten Sie diese von der Klasse *AppSelectionTab* ab.

2. Kopieren Sie die Namespace-Referenzen in den Code.

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
```

Codesegment 3 Using-Anweisung für SelectionAddress

3. Wechseln Sie zur Design-Ansicht. Passen Sie die Grösse des Controls an damit die Buttons „Go“ und „Clear“ sichtbar werden.

4. Die Basisklasse – die bereits ein *Tab*-Control enthält – ermöglicht es Ihnen auf einfache Weise eine differenzierte Suchmaske zu gestalten. Selektieren Sie dazu das *TabPages*-Property von *tabControl* und fügen Sie zwei *TabPages* hinzu. Benennen Sie die erste *TabPage* nach *tbpAddress* und das Zweite nach *tbpAddressId*. Zudem besitzt das abgeleitete Control bereits zwei Buttons. Der *Go*-Button setzt die Suchabfrage ab und der *Clear*-Button löscht alle Inhalte der Textboxen auf diesem Control. Folgende Tabelle enthält das *Text*- und *Name*-Property der zwei neu erstellten *TabPages*:

Eingefügte TabPage	Text	Name
TabPage1	Address	tbpAddress
TabPage2	Address ID	tbpAddressId

Tabelle 3 TabPages des SelectionAddress-Controls

5. Klicken Sie auf die *Address-TabPage* und fügen dort jeweils drei *Label*- und *TextBox*-Controls aus dem *App Controls*-Register und eine ComboBox aus dem *Windows Forms*-Register der Toolbox ein. Setzen Sie die *Text*- und *Name*-Properties der Controls auf folgende Werte:

Eingefügtes Control	Text	Name
AppLabel1	Company	lblCompany
AppLabel2	ZIP	lblZip
AppLabel3	City	lblCity
AppLabel4	Order By	lblOrderBy
AppTextBox1		txtCompany
AppTextBox2		txtZip
AppTextBox3		txtCity
AppComboBox1		cboOrderBy

Tabelle 4 Controls der Address-TabPage

6. Ordnen Sie die eingefügten Controls an.

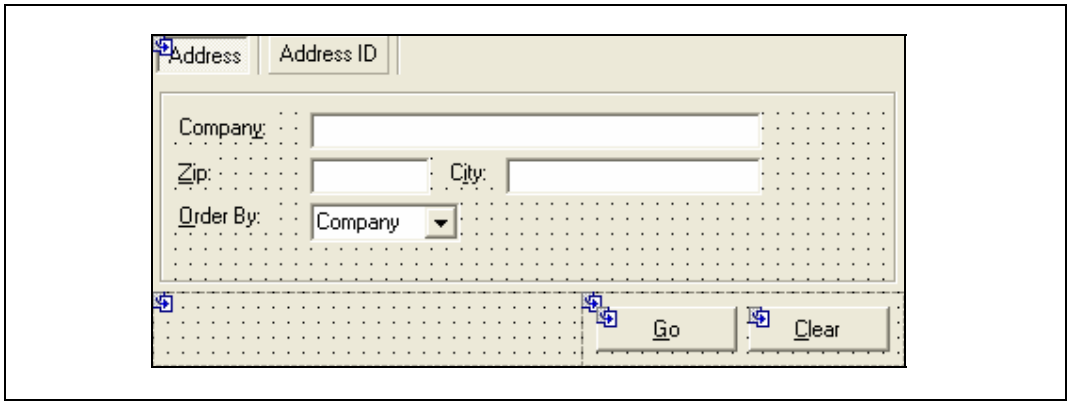


Abbildung 7 Layout-Vorschlag für SelectionAddress-Address-Tab

7. Selektieren Sie das Control *cboOrderBy* und öffnen Sie das *Property-Sheet* um folgende *Items* hinzuzufügen.
   - Company
   - Country
   - Zip
   - City

8. Klicken Sie nun auf die *Address ID*-TabPage und fügen Sie dort jeweils ein *Label* und ein *TextBox*-Control ein. Setzen Sie das *Text*- und *Name*-Property der beiden Controls auf folgende Werte:

Einzufügendes Control	Text	Name
AppLabel1	Address ID	lblAddressId
AppTextBox1		txtAddressId

Tabelle 5 Controls der AddressId-TabPage

9. Ordnen Sie die beiden Controls wie folgt an:

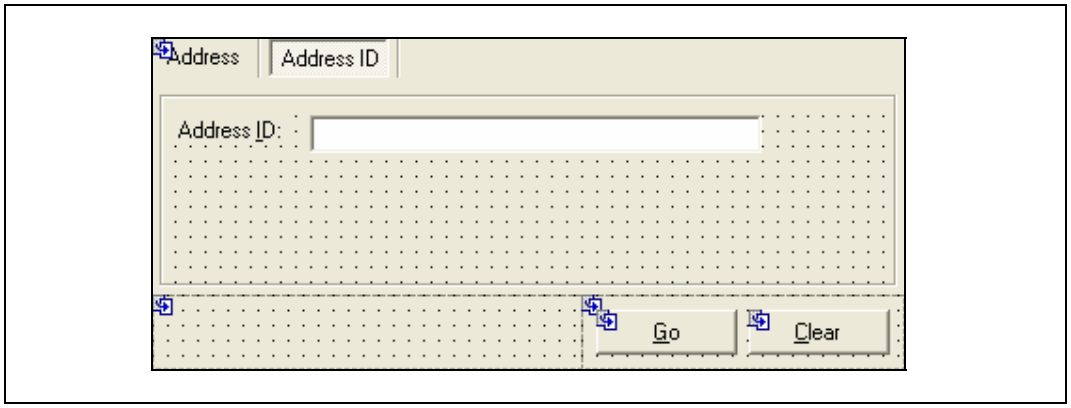


Abbildung 8 Layout-Vorschlag für SelectionAddress-ID-Tab

1.5 Erstellen der SelectionGridAddress-Klasse

Als Nächstes wird die Control-Klasse erstellt, in welcher zur Laufzeit die gefundenen Adressen in einer Liste präsentiert werden. Diese Liste enthält dieselben Adressen wie die Tree-Ansicht:

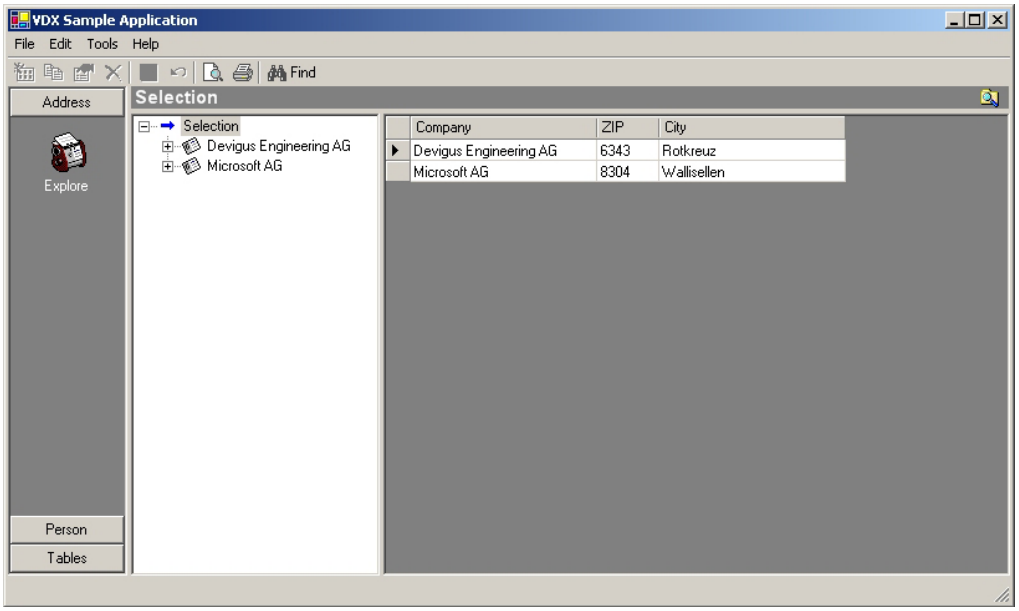


Abbildung 9 Treeview mit Listenansicht

Da diese Liste kein *System.Windows.Forms.UserControl* ist, sondern eine *System.Windows.Component*, kann nicht der *Inheritant Control Picker* verwendet werden. Visual Studio bietet zwei Arten an um eine Klasse zu vererben. Nachfolgend werden beide Möglichkeiten ausführlich beschrieben.

A) Klasse vererben über die Class View

1. Wechseln Sie zur Class View (Klassenansicht), selektieren Sie das *mySampleClient-Projekt* und fügen Sie eine neue Klasse hinzu.

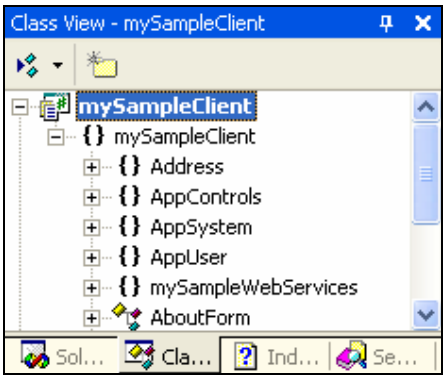


Abbildung 10 Class View

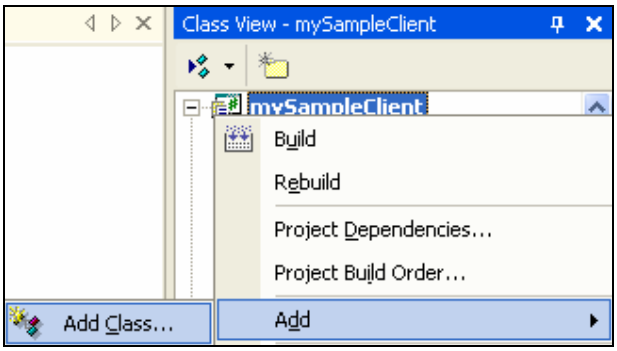


Abbildung 11 Klasse Hinzufügen über Class View

2. Bennenen Sie die Klasse SelectionGridAddress und setzen Sie den Namespace auf *mySampleClient.Address*.

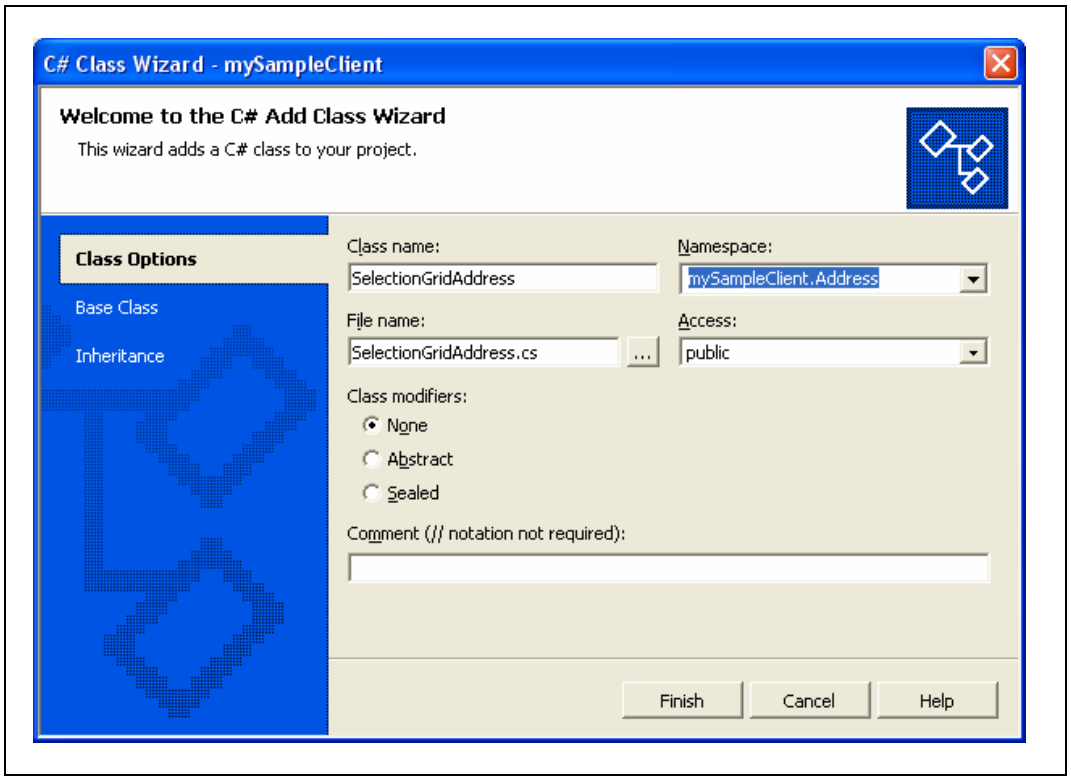


Abbildung 12 Class Wizard, Optionen

3. Leiten Sie die Klasse von AppSelectionGrid ab, die sich im Namespace *mySampleClient.AppControls* befindet.

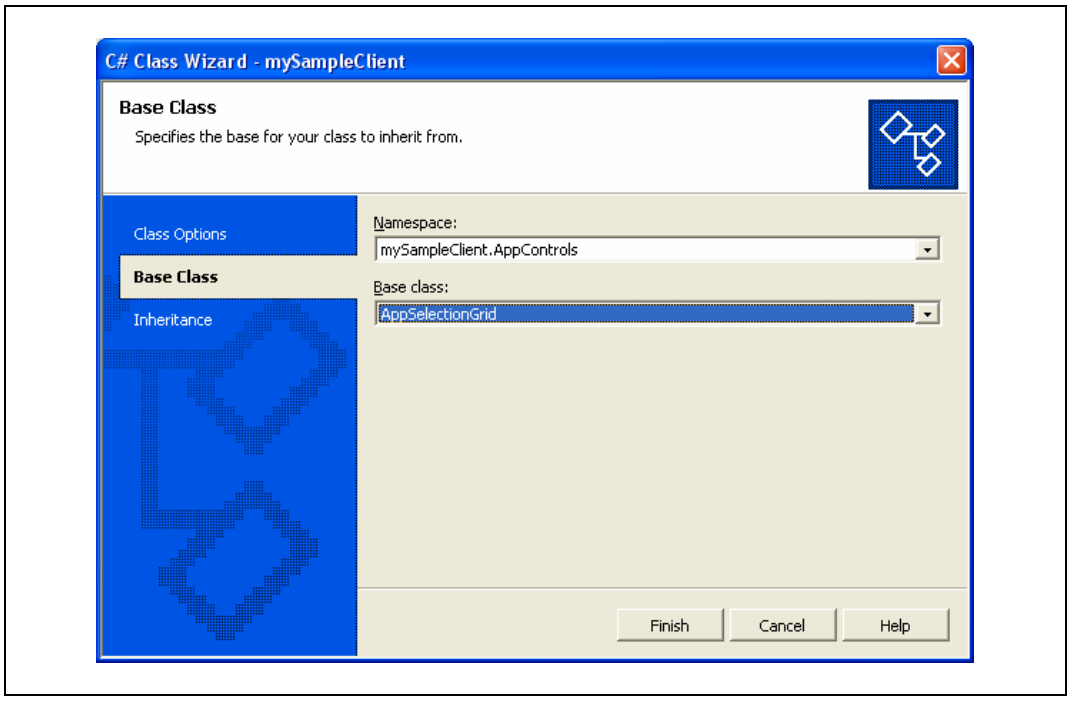


Abbildung 13 Class Wizard, Basisklasse

B) Klasse vererben über Solution Explorer und Code

1. Selektieren Sie den Ordner Address im Solution Explorer und fügen Sie eine neue Klasse über das Kontext-Menu hinzu.

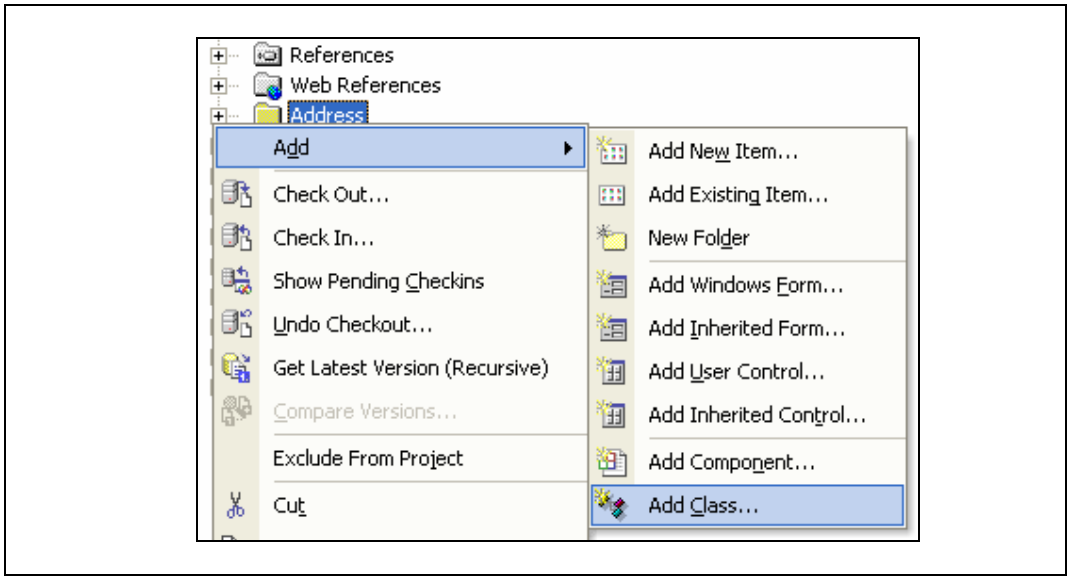


Abbildung 14 Neue Klasse über Solution Explorer erstellen

2. Benennen Sie die Klasse als SelectionGridAddress

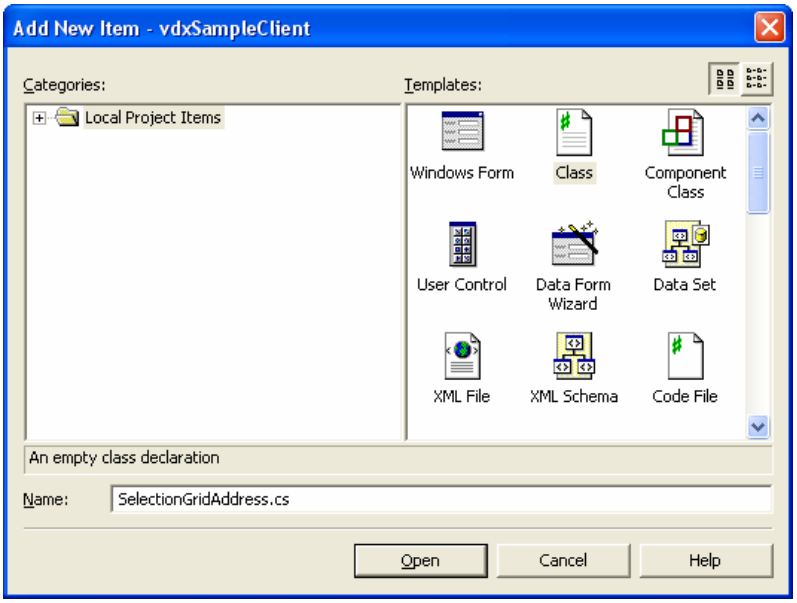


Abbildung 15 Klassennamen vergeben

3. Leiten Sie von *AppControls.AppSelectionGrid* ab.

```
namespace <ClientProjectNamespace>.Address
{
    ...
    public class SelectionGridAddress : AppControls.AppSelectionGrid
    {
        ...
    }
}
```

Codesegment 4 SelectionGridAddress Klassen-Definition

1.5.1 DataSetReference *dsrAddressList* an *DataGrid* binden

Damit die selektierten Adressen in der Liste angezeigt werden, müssen wir ein *dsrAddressList*-DataSetReference der *SelectionGridAddress*-Klasse hinzufügen und an das *DataGrid* binden. Diese *DataSetReference* referenziert das DataSet dsAddressList, welches Bestandteil der Server-Anwendung ist:

1. Öffnen Sie das *SelectionGridAddress*-Control in der Designer-Ansicht.

2. Öffnen Sie das *VDX-Register* der Toolbox und ziehen Sie ein *vdxDataSetReference* auf das *SelectionGridAddress*-Control.

Hinweis Wieso wird kein DataSet auf das Control gezogen? Weil in diesem Falle alle DataSets der WebService-Referenz initialisiert werden. Im Gegensatz dazu initialisiert die *DataSetReference* nur das benötigte DataSet

3. Öffnen Sie das Property-Sheet und konfigurieren Sie das *vdxDataSetReference*-Objekt wie folgt:

Property	Selektion
Name	dsrAddressList
BindingContainer	SelectionGridAddress
RefDataSet	<ClientProjectNamespace>.<WebServiceName>.dsAddressList

Tabelle 6 Konfiguration des AddressList-DataSetReference von SelectionGridAddress

4. Das *dsrAddressList*-Objekt muss nun an das *DataGrid*-Control gebunden werden. Selektieren Sie dazu die Properties des *SelectionGridAddress*-Controls und setzen die folgenden Werte:

Property	Selektion
DataSource	dsrAddressList
DataMember	Address

Tabelle 7 Konfiguration des SelectionGridAddress-DataGrids

5. Compilieren Sie die Client-Solution erneut.

6. Erstellen Sie die ColumnStyles über *Generate ColumnStyles* des Properties-Fensters.

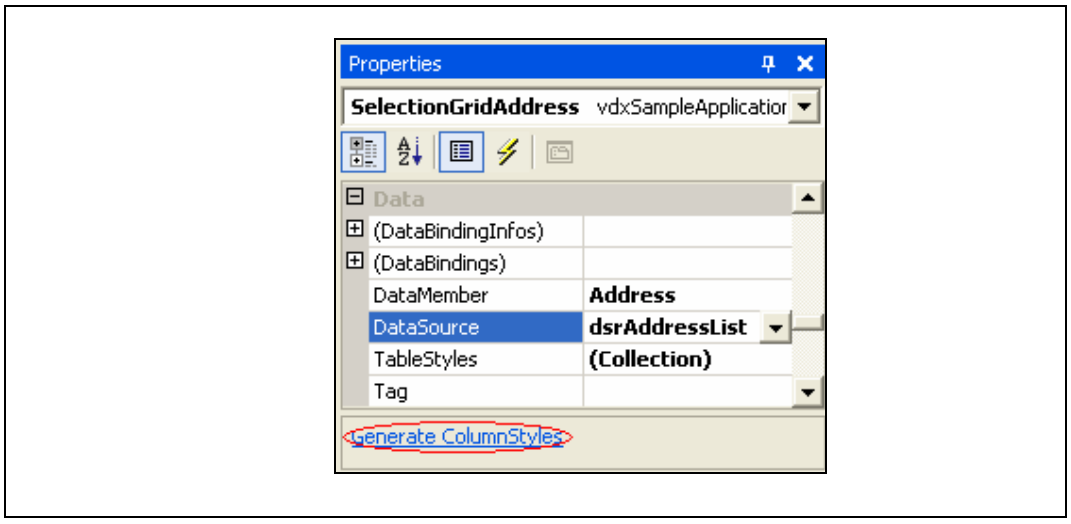


Abbildung 16 ColumnStyles für DataGrid generieren

7. Öffnen Sie den TableStyle Editor und entfernen Sie alle ColumnStyles ausser dcsCompany, dcsZIP und dcsCity. Und setzen folgende Werte:

Column	Header Text	Width
Company	Company	90
ZIP	ZIP	30
City	City	90

Tabelle 8 Konfiguration der ColumnStyles von SelectionGridAddress

8. Kompilieren Sie das Projekt, um den bisher implementierten Code auf Fehler zu überprüfen.

1.6 Erstellen der DataDetailAddress-Klasse

Dieses Control präsentiert die Details einer entsprechenden Adresse. Die Anzeige kann dabei auf zwei verschiedene Arten aktiviert werden:

- Selektion einer Adresse in der Tree-Ansicht oder
- Doppel-Klick auf eine Adresse innerhalb der Adressliste der Klasse *SelectionGridAddress*.

Folgende Schritte zeigen das Erstellen der *DataDetailAddress*-Klasse:

1. Fügen Sie dem Ordner Address eine neue Klasse, hinzu. Wählen Sie dazu den Befehl *Add Inherited Control...* aus dem Kontext-Menu des Address Ordners aus.

2. Benennen Sie die Klasse in *DataDetailAddress* um.

3. Klicken Sie auf *Open* um den Dialog zur Selektion der Basisklasse zu öffnen.

4. Aus der folgenden Komponenten-Liste wählen Sie die Klasse *AppDataDetail* aus.

5. Kopieren Sie folgende Namespace-Referenzen in den Code.

```
using System.Diagnostics;
using System.Windows.Forms;
using <ClientProjectNamespace>.<WebServiceName>;
```

Codesegment 5 Using-Anweisung für DataDetailAddress

6. Selektieren Sie die Designer-Ansicht der *DataDetailAddress*-Klasse und ziehen Sie folgende Controls darauf; diese befinden sich im *App Controls*-Register der Toolbox:

Einzufügendes Control	Text	Name
AppLabel	Company:	lblCompany
AppLabel	Street:	lblStreet
AppLabel	ZIP:	lblZip
AppLabel	City:	lblCity
AppLabel	Country:	lblCountry
AppLabel	Phone:	lblPhone
AppLabel	Fax:	lblFax
AppLabel	E-Mail:	lblEmail
AppLabel	URL:	lblUrl
AppLabel	Address Type:	lblAddrType
AppButton	Go	btnGoEmail
AppButton	Go	btnGoUrl
AppTextBox		txtCompany
AppTextBox		txtStreet
AppTextBox		txtZip
AppTextBox		txtCity
AppTextBox		txtPhone
AppTextBox		txtFax
AppTextBox		txtEmail
AppTextBox		txtUrl
AppComboBox		cboCountry
AppComboBox		cboAddrType

Tabelle 9 Controls auf DataDetailAddress

7. Ordnen Sie die Controls folgendermassen an:

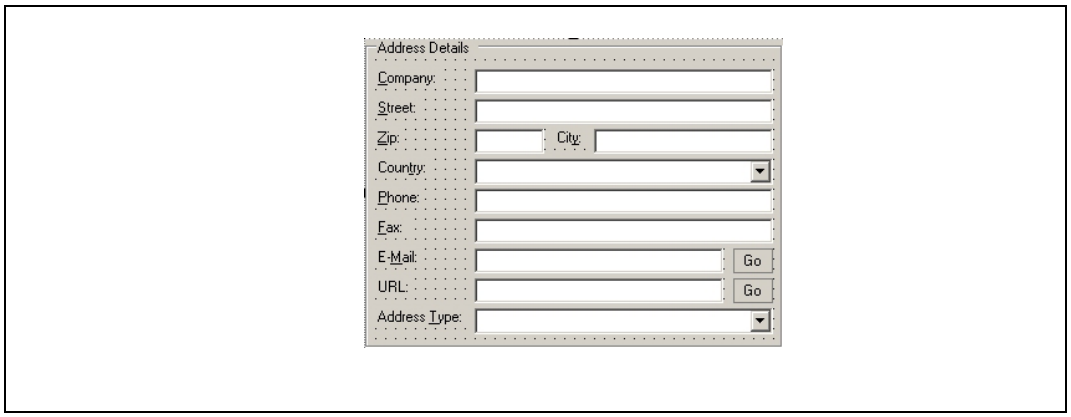


Abbildung 17 DataDetailAddress in Designer-Ansicht

1.6.1 Konfiguration des *dataStatusManager*

Da im *DataDetailAddress*-Control Daten geändert und gespeichert werden, müssen wir das d*ataStatusManager*–Control entsprechend konfigurieren:

1. Selektieren Sie *dataStatusManager*-Control und setzen Sie die Properties auf die folgenden Werte damit das DataSet vom DataStatusManager verwaltet werden kann.

Property	Selektion
DataSetType	<ClientProjectNamespace>.<WebServiceName>.dsAddress
MainDataTableName	Address
SelectedDataTableName	Address

Tabelle 10 Konfiguration des DataStatusManagers von DataDetailAddress

Zusätzlich können auf dem *dataStatusManager* Status-Texte gesetzt werden. Die Fragestellungen, ob gespeichert oder gelöscht werden soll, ist ebenfalls zu definierten.

1.6.2 DataStatusManager, *dsrAddrType* und *dsrCountry* an Controls binden

Damit die Details der in der Tree-Ansicht selektierten Adresse im entsprechenden *DataDetailAddress*-Control angezeigt werden, müssen zwei DataSetReferences der *DataDetailAddress*-Klasse hinzugefügt werden.

1. Damit die *ComboBoxen* nebst dem anzuzeigenden Wert auch eine Auswahl an alternativen Werten anzeigen, fügen Sie zwei *DataSetReferences* ein. Ziehen Sie dafür zwei vdxDataSetReferences auf das *DataDetailAddress* -Control.

2. Konfigurieren Sie die *vdxDataSetReference*-Objekte wie folgt:

Property	Selektion
Name	dsrAddrType
BindingContainer	DataDetailAddress
RefDataSet	<ClientProjectNamespace>.<WebServiceName>.dsAddrType

Tabelle 11 Konfiguration des AddrType-DataSetReference von DataDetailAdress

Property	Selektion
Name	dsrCountry
BindingContainer	DataDetailAddress
RefDataSet	<ClientProjectNamespace>.<WebServiceName>.dsAddrCountry

Tabelle 12 Konfiguration des Country-DataSetReference von DataDetailAdress

3. Der *dataStatusManager* muss nun an die entsprechenden *TextBox*-Controls gebunden werden. Folgende Tabelle beschreibt die entsprechenden *TextBox*-BindingInfos:

Control	Property DataBindingInfos.Text
txtCompany	dataStatusManager - Address.Company
txtStreet	dataStatusManager - Address.Street
txtZip	dataStatusManager - Address.ZIP
txtCity	dataStatusManager - Address.City
txtPhone	dataStatusManager - Address.Phone
txtFax	dataStatusManager - Address.Fax
txtEmail	dataStatusManager - Address.Email
txtUrl	dataStatusManager - Address.URL

Tabelle 13 Controls an DataStatusManager binden

Hinweis DataBindingInfos entspricht nicht den .NET DataBindings. DataBindingInfos setzt erst zur Runtime die DataBinding auf den Controls.

4. Analog dazu müssen die beiden *ComboBox*-DataBindingInfos definiert werden. Diese sind in der folgenden Tabelle beschrieben:

Property	Control cboCountry	Control cboAddrType
SelectedValue[1)]	dataStatusManager - Address.Country	dataStatusManager - Address.AddrTypeID
DataSource	dsrCountry	dsrAddrType
Display Member	Country.Descr	AddrType.Descr
Value Member	Country.Country	AddrType.AddrTypeId

[1]) Property Window: (DataBindingInfos) SelectedValue

Tabelle 14 Country-ComboBox an DSM und DSR binden

Hinweis Falls die Selektion eines bestimmten Properties keine Auswahl enthält, dann kompilieren Sie jeweils die Solution.

5. Kompilieren Sie nun die Applikation, um zu testen, ob Fehler vorhanden sind. In den nächsten Kapiteln werden Sie die erstellten Controls miteinander verknüpfen und den entsprechenden Code implementieren.

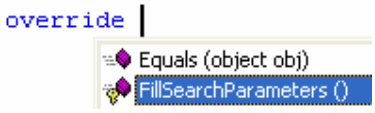
1.7 Programmierung des SelectionAddress-Controls

1.7.1 Überschreiben der *FillSearchParameters*-Methode

Damit die vom Benutzer eingegebenen Suchbegriffe an den entsprechenden *WebService* übermitteln werden, müssen die Begriffe in der Methode *FillSearchParameters* in ein *vdxSearchParameters*-Objekt eingepackt werden. Dazu führen Sie folgende Schritte aus:

1. Überschreiben Sie die *FillSearchParameters*-Methode.

Tipp Überschreiben einer Methode mit Code-Completion. Geben Sie **override** mit nachfolgendem Space ein und wählen Sie die Methode *FillSearchParameters* aus der Liste aus.



2. Ersetzen Sie den vorhandenen Code mit dem folgenden Code:

```
vdxSearchParameters sp;

if (this.tabControl.SelectedTab.Equals(this.tbpAddress))
{
   sp = new vdxSearchParameters("SelectionAddressByDescr");
   sp.Add("Company",this.txtCompany.Text, vdxSearchType.StartWith);
   sp.Add("Zip",this.txtZip.Text, vdxSearchType.StartWith);
   sp.Add("City",this.txtCity.Text, vdxSearchType.StartWith));
   sp.Add("OrderBy", this.cboOrderBy.Text); // Default vdxSearchType.Equal
}
else
{
   int addressId = 0;
   if (this.txtAddressId.Text != string.Empty)
   {
      addressId = Convert.ToInt32(this.txtAddressId.Text);
   }

   sp = new vdxSearchParameters("SelectionAddressId");
   sp.Add("AddressId", addressId);         // Default vdxSearchType.Equal
}

return sp;
```

Codesegment 6 Überschreiben der FillSearchParameters-Methode von SelectionAddress

Diese Methode wird aufgerufen, sobald der Benutzer auf den *Go*-Button des Suchen-Controls klickt. Die Parameterübergabe läuft dabei jeweils über ein *vdxSearchParameters*-Objekt mit dem Namen *SelectionAddressByDescr* oder *SelectionAddressById*, abhängig aus welchem Tab der Suchmaske der Benutzer die Suche startet. Ein *vdxSearchParameters*-Objekt enthält jeweils ein bis mehrere *vdxSearchParameter*-Objekte, wobei jedes einen spezifischen Suchbegriff enthält. Mit der *Add*-Methode wird ein solches Objekt dem *vdxSearchParameters*-Objekt hinzugefügt.

Der Add-Aufruf des *vdxSearchParameters*-Objektes verlangt folgende Angaben:

- Namen für den Suchbegriff
- Suchbegriff
- Suchart, falls nicht *vdxSearchType.Equal*.

Folgende Sucharten stehen zur Verfügung:

- *vdxSearchType.Equal*: Wird angegeben, falls der zu suchende Datenbankwert exakt mit dem eingegebenen Suchbegriff übereinstimmen soll (SQL: field = value).
- *vdxSearchType.StartWith*: Wird angegeben, falls der zu suchende Datenbankwert mit dem eingegebenen Suchbegriff beginnen soll (SQL: field LIKE value%).
- *vdxSearchType.Contains*: Wird angegeben, falls der zu suchende Datenbankwert den eingegebenen Suchbegriff enthalten soll (SQL: field LIKE %value%).
- *vdxSearchType.BetweenStart*: Wird angegeben, falls der zu suchende Datenbankwert ab dem eingegebenen Suchbegriff starten soll (SQL: field BETWEEN value AND anotherValue). Muss in Kombination mit der folgenden Suchart kombiniert werden.

- *vdxSearchType.BetweenEnd*: Wird angegeben, falls der zu suchende Datenbankwert bis zum eingegebenen Suchbegriff suchen soll (SQL: field BETWEEN anotherValue AND value). Muss in Kombination mit der vorstehenden Suchart kombiniert werden.
- *vdxSearchType.SmallerAs*: Wird angegeben, falls der zu suchende Datenbankwert bis zum eingegebenen Suchbegriff suchen soll (SQL: field < value).
- *vdxSearchType.LargerAs*: Wird angegeben, falls der zu suchende Datenbankwert ab dem eingegebenen Suchbegriff starten soll (SQL: field > value).
- *vdxSearchType.SmallerEqualAs*: Wird angegeben, falls der zu suchende Datenbankwert bis und mit dem eingegebenen Suchbegriff suchen soll (SQL: field <= value).
- *vdxSearchType.LargerEqualAs*: Wird angegeben, falls der zu suchende Datenbankwert gleich oder grösser dem eingegebenen Suchbegriff sein soll (SQL: field >= value).

1.8 Programmierung des DataDetailAddress-Controls

Damit der Benutzer die eingegebene URL (URL-Textbox) überprüfen kann, muss die entsprechende Website geöffnet werden können. Dazu müssen Sie den Klick-Event des *Go*-Buttons implementieren:

1. Öffnen Sie das *DataDetailAddress*-Control in der Design-Ansicht.

2. Doppelklicken Sie auf das *btnGoUrl*-Objekt.

3. Fügen Sie den folgenden Code in die Event-Listener-Methode *btnGoUrl_Click* ein:

```
if (txtUrl.Text != string.Empty)
{
   Process process = new Process();
   try
   {
      process.StartInfo.FileName = txtUrl.Text;
      process.Start();
   }
   catch (System.Exception ex)
   {
      MessageBox.Show(ex.Message);
   }
}
```

Codesegment 7 Implementation des URL-Button-Klick Listeners

4. Analog dazu implementieren Sie den Klick auf den *Go*-Button der Email-*TextBox* (*btnGoEmail_Click*):

```
if (txtEmail.Text != string.Empty)
{
   Process process = new Process();
   try
   {
      process.StartInfo.FileName = "mailto:" + txtEmail.Text;
      process.Start();
   }
   catch (System.Exception ex)
   {
      MessageBox.Show(ex.Message);
   }
}
```

Codesegment 8 Implementation des MailTo-Button-Klick Listeners

5. Damit die *Country*- und *Address Type-ComboBoxen* mit den entsprechenden Ländern bzw. Adresstypen gefüllt werden, ist folgender Code in den Konstruktor der *DataDetailAddress*-Klasse einzufügen. Die DataSets können über die entsprechenden WebSevices abgerufen werden. Fügen Sie den Code nach dem *InitializeComponent*-Aufruf ein:

```
this.dsrCountry.FillData(
      WebServiceProvider.wsCountry.GetDataSetCountry());
this.dsrAddrType.FillData(
      WebServiceProvider.wsAddrType.GetDataSetAddrType());

this.DataStatusManager.Updater = WebServiceProvider.wsAddress;
```

Codesegment 9 Implementation des DataDetailAddress-Konstruktors

Hinweis Das Property *UpdateMethodName* des *DataStatusManagers* muss nicht gesetzt werden, da dieses in der Basisklasse *AppDataDetail* implementiert ist. Die WebService Update-Methode heisst überall *UpdateDataSet*. Nur falls dies abweichen sollte, muss das Property explizit auf dem DataDetail gesetzt werden.

6. Überschreiben Sie die *LoadData*-Methode

```
return WebServiceProvider.wsAddress.GetDataSetAddress(
      searchParameters.GetParameters());
```

Codesegment 10 Überschreiben der LoadData-Methode von DataDetailAddress

7. Überschreiben Sie die *GetInfoBarText*-Methode

```
dsAddress ds = this.DataStatusManager.DataSet as dsAddress;
if (ds != null && ds.Address.Count == 1)
{
    dsAddress.AddressRow row = (dsAddress.AddressRow)ds.Address.Rows[0];
    return row.Company;
}
return string.Empty;
```

Codesegment 11 Überschreiben der GetInfoBarText-Methode von DataDetailAddress

8. Kompilieren Sie das Projekt, um den bisher implementierten Code auf Fehler zu überprüfen.

Hinweis Falls trotz Kompilierung nicht das gewünschte Ergebnis erreicht wird, schliessen sie alle Projektfiles und öffnen Sie diese erneut.

1.9 Konfiguration des ExplorerAddress-Controls

Damit die Container-Klasse *ExplorerAddress* auf die eingebetteten Controls

- *SelectionAddress*,
- *SelectionGridAddress* und
- *DataDetailAddress*

zugreifen und diese miteinander verknüpfen kann, sind folgende Schritte auszuführen:

1. Öffnen Sie die Design-Ansicht der *ExplorerAddress*-Klasse und selektieren Sie das *SearchControlType* Property im Property-Sheet.

2. Wählen Sie aus dem entsprechenden Drop-Down-Menu die Klasse *SelectionAddress* aus.

3. Analog dazu wählen Sie für das *SelectionGridType*-Property den Eintrag *SelectionGridAddress* aus.

4. Damit der Explorer die Suchmaske als erstes anzeigt, setzen Sie den DisplayMode auf Selection. Folgende Tabelle fasst die drei Einstellungen nochmals zusammen:

Property	Selektion
SearchControlType	<ClientProjectNamespace>.Address.SelectionAddress
SelectionGridType	<ClientProjectNamespace>.Address.SelectionGridAddress
DisplayMode	Selection

Tabelle 15 Konfiguration des ExplorerAddress

Durch diese Selektionen sind die Adressen-Controls mit seinem Container – der *ExplorerAddress*-Klasse – verknüpft.

1.10 Programmierung des ExplorerAddress-Controls

Die Programmlogik der eingebetteten Controls wird zentral in der Klasse *ExplorerAddress* implementiert. Folgende Abschnitte beschreiben die Implementation dieser Funktionalität.

1.10.1 Überschreiben der *LoadData*-Methode

Sobald der Benutzer auf den *Go*-Button der Suchmaske klickt, soll der entsprechende Datenbezug ausgelöst werden. Der Datenbezug wird in der Methode *LoadData* eingebaut. Um diese Funktionalität zu implementieren, führen Sie die nachfolgenden Schritte aus:

1. Überschreiben Sie *LoadData*-Methode ohne Parameter

```
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);
}

return ds;
```

Codesegment 12 Überschreiben der LoadData-Methode von ExplorerAddress

Diese Methode wird immer dann aufgerufen, sobald ein Datenbezug für den Top-Level ausgeführt werden muss. Ein Top-Level Datenbezug wird grundsätzlich beim Klick auf den *Go*-Button der Suchmaske (bzw. *SelectionAddress*) ausgeführt.

1.10.2 Überschreiben der *GetRootNode*-Methode

Bestimmen Sie mit dieser Methode ob Sie einen RootNode haben wollen oder wie der RootNode aussehen soll.

1. Überschreiben Sie die *GetRootNode*-Methode

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

Codesegment 13 Überschreiben der GetRootNode-Methode

Hinweis Falls kein RootNode erwünscht, dann muss null retourniert werden.

1.10.3 Überschreiben der *GetTreeNode*-Methode

Das Präsentieren der zuvor geladenen Daten wird in der *GetTreeNode*-Methode implementiert.

1. Überschreiben Sie die Methode GetTreeNode wie folgt:

```
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;
}
return new vdxTreeNode(sb.ToString(), imageIndex, selectedImageIndex,
       hasChildren);
```

Codesegment 14 Überschreiben der GetTreeNode-Methode von ExplorerAddress

Diese Methode wird im Anschluss an den vorhergehenden Datenbezug aufgerufen, sofern der Bezug ein entsprechendes *DataSet* retournierte. Für jede *DataRow* in der *Address*-Tabelle wird das *Company-Feld* mit dem entsprechenden Tree-Knoten (Tree-Node) verknüpft.

Folgende Tabelle beschreibt die Parameter des *vdxTreeNode*-Konstruktoraufrufs:

Parameter und Typ	Beschreibung
1. string	Die Bezeichnung des entsprechenden Tree-Knotens
2. int	Der *imageListTreeView*-Index für nicht selektierte Tree-Knoten
3. int	Der *imageListTreeView*-Index für selektierte Tree-Knoten
4. bool	True, falls der Knoten Sub-Knoten haben kann. Es wird ein + Zeichen vor dem Knoten dargestellt.

Tabelle 16 Parameter des vdxTreeNode-Konstruktor

Damit ist die Füllung der Tree-Ansicht implementiert...

1.10.4 Überschreiben der *GetLoaderInfo*-Methode

Um das DataDetail zu laden muss eine *vdxLoaderInfo* entsprechend der *vdxNavigatorInfo* erstellt werden.

1. Überschreiben Sie die Methode *GetLoaderInfo*

```
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;
    }
}
return new vdxLoaderInfo(sp, dataDetailType);
```

Codesegment 15 Überschreiben der GetLoaderInfo-Methode von ExplorerAddress

Die Parameter des *vdxLoaderInfo*-Objektes bestimmen dabei in welchem Kontext die Datenpräsentation stattfindet und mit welchen Daten bzw. welchem *DataSet* die Daten angezeigt werden sollen.

1.11 Programmierung des vdxActionItem

Damit die Adressenverwaltung durch einen Klick auf das entsprechende Outlookbar-Icon geladen wird, muss entweder der entsprechende Klick-Event implementiert oder das Icon mit dem *actionManager*-Objekt verknüpft werden. Folgende Schritte beschreiben die Verknüpfung mit dem *actionManager*-Objekt:

1. Selektieren Sie die Properties des *actionManager*-Objekts; dieses befindet sich auf dem *mySampleClientMainForm* (das *actionManager*-Objekt wurde bereits im Rahmen vom Setup-Tutorial erstellt).

2. Öffnen Sie den *Collection Editor* des *ActionItems*-Properties.

3. Klicken Sie auf *Add* und benennen Sie das neue vdx*ActionItem* in *aiExploreAddress* um.

4. Selektieren Sie die Properties des *aiExploreAddress*-Objektes. Beachten Sie dabei, dass im *Collection Editor* nicht direkt auf die entsprechenden Events gewechselt werden kann; daher muss der *Collection Editor* geschlossen und das *ActionItem*-Objekt durch Auswahl in der ComboBox des *Properties Windows* selektiert werden.

5. Selektieren Sie die Events-Ansicht (Schaltfläche mit gelbem Blitz) und Doppel-Klicken auf den *ActionInvoked*-Event des *aiExploreAddress*-Objektes.

6. Dies fügt die *aiExploreAddress_ActionInvoked*-Event-Listener-Methode in die *MainForm*-Klasse ein. Fügen Sie den folgenden Code in diese Methode ein:

```
Show(typeof(Address.ExplorerAddress));
```

Codesegment 16 Implementation des ActionItem-ExploreAddress-Listeners

Dieser Code lädt beim initialen Klick auf das Adress-Icon der Outlookbar die entsprechende Suchmaske über die Klasse *SelectionAddress*.

7. Um das Action-Event mit dem entsprechenden Control zu verknüpfen, fügen Sie in der Methode AddInvokers diese Zeile ein.

```
this.aiExploreAddress.AddInvoker(this.obiExploreAddress);
```

Codesegment 17 Registratrion des OutllookBarIcon-ExploreAddress-Actions

Damit haben wir den Adressenteil unserer Beispiel-Applikation implementiert. Kompilieren Sie das Projekt und starten Sie die Applikation um den bisher implementierten Code auszuführen. Sie können an dieser Stelle auf den *Explore*-Button in der Outlookbar-Kategorie *Address* klicken; dies startet die Adressverwaltung, in der Sie über die Suchbegriffe die gewünschten Adressen anzeigen lassen können. Diese können Sie dann ändern und sichern. Zudem können Sie neue Adressen erstellen oder bestehende kopieren und entsprechend anpassen.

Im *vdxExplorerList*-Tutorial wird exemplarisch anhand der Beispiel-Applikation ein Referenzdaten-Szenario implementiert. Dieses dient der Verwaltung von Referenzdaten bzw. von Tabellen auf dem Server, deren Inhalt durch andere Tabellen referenziert wird. Ein Beispiel ist die Verwaltung der Länder-Tabelle, deren Inhalt einer Adresse zur Auswahl des entsprechenden Landes dient.

1.12 Bildverzeichnis

1.13 Tabellenverzeichnis

1.14 Codesegment-Verzeichnis