Visual.NET Extensions – Tutorial

“vdxExplorerTree – One To Many 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: 20.11.2003

Inhaltsverzeichnis

1 *vdxExplorerTree* – One To Many Datenmanipulation mit Treeview

In diesem Tutorial wird ein weiteres vdxExplorerTree-Szenario implementiert. Am Beispiel einer Bestellverwaltung soll ein 1:N-Szenario implementiert werden. Das 1:N-Szenario wird dadurch formuliert, dass *eine* Bestellung *mehrere* Artikel enthält (1 Bestellung : N Artikel).

In einem ersten Schritt gestaltet sich die Implementierung der Bestellverwaltung analog zur Adressverwaltung. Da Bestellungen aber jeweils mit mehreren Artikeln verbunden sind, bedarf es an geeigneter Stelle der entsprechenden Erweiterung. 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
<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:
   - wsItem
   - wsOrder

   Siehe dazu Kapitel 1.10 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 wsItemSoap wsItem
{
    get { return (wsItemSoap)GetWebService(typeof(wsItemSoap)); }
}

static public wsOrderHSoap wsOrderH
{
    get { return (wsOrderHSoap)GetWebService(typeof(wsOrderHSoap)); }
}
```

Codesegment 1 Properties von WebServiceProvider ergänzen

1.3 Voraussetzungen

Da die Klasse *DataDetailOrder* das im *vdxPickField*-Tutorial erstellte *DataPickFieldAddress*-Control verwendet, empfiehlt sich die vorherige Implementierung des *DataPickFieldAddress*-Controls.

Zudem sollte der Stammdatenfall *Item* bereits implementiert sein (siehe dazu auch das *vdxExplorerGrid*-Tutorial, Aufbau ist identisch), damit die Artikel bearbeitet werden können.

Alternativ können Sie auch das *DataPickFieldAddress*-Control und den *Item*-Stammdatenfall von der mitgelieferten *VDX Sample Application* übernehmen.

1.4 vdxExplorerTree-Szenario am Beispiel der Bestellverwaltung

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

1. Hinzufügen des Container-Controls für die Bestellverwaltung (Klasse *ExplorerOrder*)

2. Hinzufügen des Suchen-Controls für die Bestellungen (Klasse *SelectionOrder*)

3. Hinzufügen des Anzeige-Controls für die gefundenen Bestellungen (Klasse *SelectionGridOrder*)

4. Hinzufügen des Anzeige-Controls für die Bestelldetails (Klasse *DataDetailOrder*)

5. Programmierung des Suchen-Controls (Klasse *SelectionOrder*)

6. Programmierung des Personendetail-Controls (Klasse *DataDetailOrder*)

7. Programmierung des Container-Controls (Klasse *ExplorerOrder*)

8. Programmierung der *vdxActionItems*

1.5 Unterschiede zur Adressverwaltung

In gewissen Fällen unterscheidet sich die Implementation der Bestellverwaltung von der Adressverwaltung. Folgende Abschnitte beschreiben diese Unterschiede.

1.5.1 Erstellen der *SelectionOrder*-Klasse

Die Basisklasse für *SelectionOrder* ist *AppSelection* und nicht App*SelectionTab*. Da die Bestell-Suchmaske nur auf einer Seite implementiert wird, benötigt es keine Tab-Pages auf der Suchmaske.

1. Erstellen Sie den Ordner Order.

2. Fügen Sie auf der *SelectionOrder*-Klasse folgende Controls ein:

Eingefügtes Control	Text	Name
AppLabel	Order Description	lblOrderDescr
AppLabel	Order By	lblOrderBy
AppTextBox		txtOrderDescr
AppComboBox		cboOrderBy

Tabelle 2 Controls auf SelectionOrder

3. Select the Control *cboOrderBy* and the *Property-Sheet* to add following *Items*.

   - Order Description
   - Order ID
   - Order Date

Die folgende Abbildung zeigt einen Layout-Vorschlag:

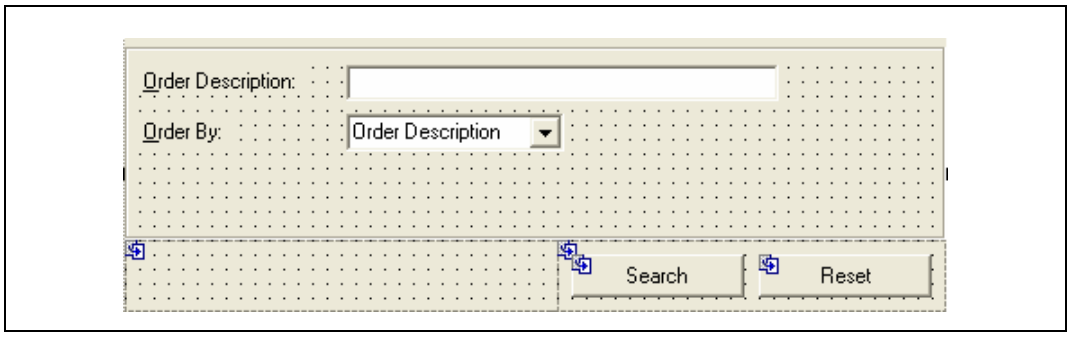

Abbildung 1 SelectionOrder in Design-Ansicht

1.5.2 DataSetReference *dsrOrderList* an *DataGrid* binden

Im Unterschied zur Adressverwaltung muss die DataSetReference *dsrOrderList* an das *dataGrid* der Klasse *SelectionGridOrder* gebunden werden. Öffnen Sie das *VDX-Register* der ToolBox und ziehen Sie ein *vdxDataSetReference* auf das *SelectionGridOrder*-Control.

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

Property	Selektion
Name	dsrOrderList
BindingContainer	SelectionGridOrder
RefDataSet	<ClientProjectNamespace>.<WebServiceName>.dsOrderList

Tabelle 3 Konfiguration des OrderList-DataSetReference von SelectionGridOrder

2. Das *dsrOrderList*-Objekt muss nun an das *dataGrid*-Control gebunden werden. Selektieren Sie dazu die Properties des *data*-Controls und setzen Sie die folgenden Werte:

Property	Selektion
DataSource	dsrOrderList
DataMember	Ordering

Tabelle 4 Konfiguration des SelectionGridOrder

3. Konfigurieren Sie die ColumnStyles wie folgt:

Column	Header Text	Width
Descr	Order Description	100
OrderDate	Order Date	60
OrderVal	Order Value	75

Tabelle 5 ColumnStyles des SelectionGridOrder

1.5.3 Erstellen des *DataPickFieldItem*-Controls

Damit der Benutzer einer Bestellposition den entsprechenden Artikel bzw. das entsprechende Item zuweisen kann, muss ein *DataPickFieldItem*-Control erstellt werden. Entweder übernehmen Sie das *DataPickFieldItem*-Control der mitgelieferten *VDX Sample Application* oder Sie erstellen das Control analog zum *vdxPickField*-Tutorial. Der Hauptunterschied besteht darin, dass das Code-Feld sichtbar ist und Methoden zusätzlich überschieben werden müssen.

Unterschiede zum *DataPickFieldAddress*.

1. *LoadData*-Methode in *PickerDialogItem* überschreiben.

```
return WebServiceProvider.wsItem.GetDataSetItem(
       searchParameters.GetParameters());
```

Codesegment 2 Überschreiben der LoadData-Methode in PickerDialogItem

2. GetPickerInfo-Methode in PickerDialogItem überschreiben.

```
dsItem._ItemRow itemRow = navigatorInfo.SelectedRow as dsItem._ItemRow;
if (itemRow != null)
{
    return new vdxPickerInfo(itemRow.ItemDescr, null, itemRow.ItemID);
}
return null;
```

Codesegment 3 Überschreiben der GetPickerInfo-Methode in PickerDialogItem

3. DataPickFieldItem konfigurieren

Property	Selektion
NavigatorClassName	<ClientProjectNamespace>.Item.SelectionGridItem
SearchControlClassName	<ClientProjectNamespace>.Item.SelectionItem

Tabelle 6 Konfiguration des pickFieldItem-Controls

Damit nach der Eingabe der ItemID im Code-Feld die Artikelbeschreibung automatisch gefunden wird, müssen drei Methoden überschrieben werden.

4. FillSearchParameters-Methode in DataPickFieldItem überschreiben (Suchparameter)

```
int itemID = Convert.ToInt32(this.PickCode);
vdxSearchParameters sp = new vdxSearchParameters("SelectionItemById");
sp.Add(new vdxSearchParameter("ItemID", itemID, vdxSearchType.Equal));
return sp;
```

Codesegment 4 Überschreiben der FillSearchParameters-Methode in DataPickFieldItem

5. LoadData-Methode in DataPickFieldItem überschreiben (Suche starten)

```
dsItem ds = WebServiceProvider.wsItem.GetDataSetItem(
       searchParameters.GetParameters());

if (ds != null && ds._Item.Count == 0)
{
    MessageBox.Show(FindForm(), "No item found.");
}
return ds;
```

Codesegment 5 Überschreiben der LoadData-Methode in DataPickFieldItem

6. FillData-Methode in DataPickFieldItem überschreiben (Resultat anzeigen)

```
dsItem ds = dataset as dsItem;
if (ds != null)
{
    dsItem._ItemRow row = ds._Item.Rows[0] as dsItem._ItemRow;

    if (row != null)
    {
        SetPickValues(row.ItemID.ToString(), row.ItemDescr);
    }
}
```

Codesegment 6 Überschreiben der FillData-Methode in DataPickFieldItem

1.5.4 Erstellen der *DataDetailOrder*-Klasse

1. Fügen Sie der *DataDetailOrder*-Klasse in der Designer-Sicht folgende Controls hinzu:

Einzufügendes Control	Text	Name
AppLabel	Address	lblAddress
AppLabel	Description	lblOrderDescr
AppLabel	Date	lblDate
AppLabel	Value	lblOrderVal
AppLabel	Notes	lblNotes
AppLabel	Item	lblItem
AppLabel	Quantity	lblQuantity
AppLabel	Additional Descr.	lblAddDescr
AppLabel	Order Pos.	lblOrderPos
AppTextBox		txtOrderPosDescr
AppTextBox		txtOrderVal
AppTextBox		txtNotes [multiline = true]
AppTextBox		txtQuantity
AppTextBox		txtOrderPosDescr
AppButton	Add	btnAdd
AppButton	Delete	btnDelete
AppDataGrid		dtgOrderPos
AppDatePickField		pickDateTime
Address.DataPickFieldAddress		DataPickFieldAddress
Item.DataPickFieldItem		DataPickFieldItem

Tabelle 7 Controls auf DataDetailOrder

2. Ordnen Sie die Controls wie folgt an:

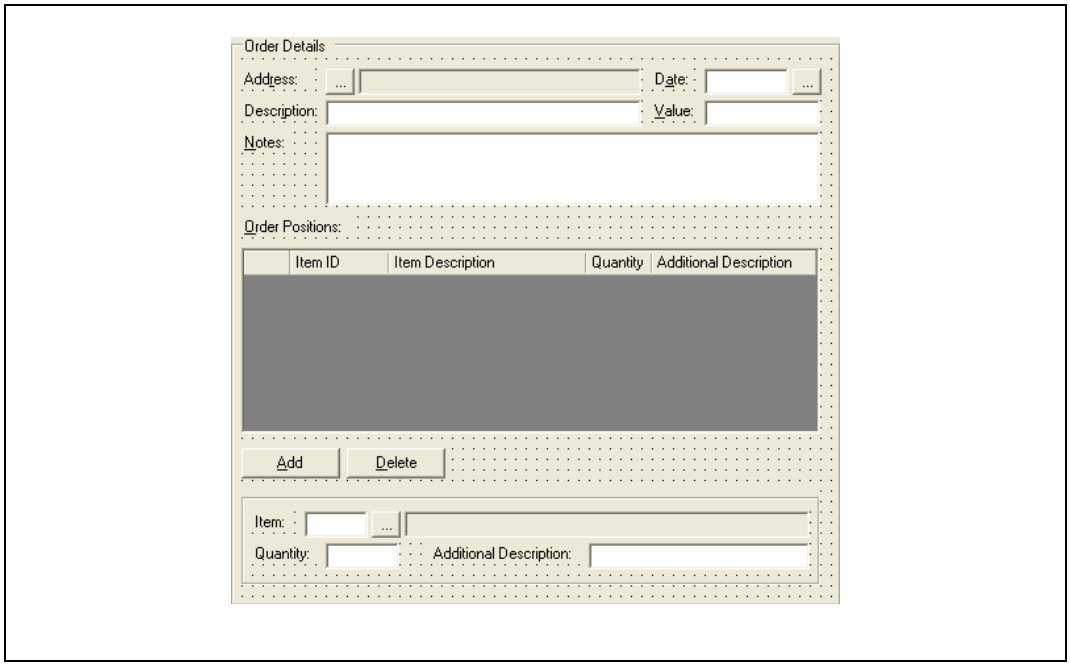

Abbildung 2 DataDetailOrder in Design-Ansicht

1.5.5 Konfiguration des *dataStatusManager*

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

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

Property	Selektion
DataSetType	<ClientProjectNamespace>.<WebServiceName>.dsOrderDetail
MainDataTableName	Ordering
SelectedDataTableName	Ordering

Tabelle 8 Konfiguration des DataStatusManagers von DataDetailAddress

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

1.5.6 DataStatusManager an Controls binden

DataStatusManager an *dtgOrderPos*-Control binden. Folgende Tabelle enthält die Auswahl der entsprechenden Properties:

Property	Selektion
DataSource	dataStatusManager
DataMember	OrderPos

Tabelle 9 DataBinding auf OrderPos-DataGrid

Konfigurieren Sie die ColumnStyles wie folgt:

Column	Header Text	Width
ItemID	Item ID	75
ItemDescr	Item Description	150
Quantity	Quantity	50
OrderPosDescr	Additional Description	125

Tabelle 10 ColumnStyles des dtgOrderPos-Controls

Der *dataStatusManager* muss zudem noch an die restlichen Controls gebunden werden. Folgende Tabelle beschreibt die entsprechenden BindingInfos:

Control	Property DataBindingInfos.Text
txtOrderDescr	dataStatusManager.Ordering.OrderDescr
txtOrderVal	dataStatusManager.Ordering.OrderVal
txtNotes	dataStatusManager.Ordering.Notes
txtQuantity	dataStatusManager.OrderPos.Quantity
txtOrderPosDescr	dataStatusManager.OrderPos.OrderPosDescr

Tabelle 11 DataBindingInfos auf TextBoxen

Analog dazu muss der *dataStatusManager* an das *pickAddress*-Control gebunden werden:

Property	Selektion
PickForeignKey	dataStatusManager.Ordering.AddressID
PickDescription	dataStatusManager.Ordering.ADDRESSDESCR

Tabelle 12 DataBindingInfos auf Address-PickField

Das Binding für das *pickItem*-Control hat folgende Werte:

Property	Selektion
PickCode	dataStatusManager.OrderPos.ItemID
PickDescription	dataStatusManager.OrderPos.ItemDescr

Tabelle 13 DataBindingInfos auf Item-PickField

Zuletzt muss noch das Binding für das *pickDateTime*-Control gesetzt werden:

Property	Selektion
Date	dataStatusManager.Ordering.Date

Tabelle 14 DataBindingInfos auf OrderDate-PickField

1.5.7 Überschreiben der *FillSearchParameters*-Methode

Fügen Sie den folgenden Code in diese Methode ein:

```
string orderBy = "OrderDescr";
if (this.cboOrderBy.Text == "Order Description") orderBy = "OrderDescr";
if (this.cboOrderBy.Text == "Order ID") orderBy = "OrderID";
if (this.cboOrderBy.Text == "Order Date") orderBy = "OrderDate";

vdxSearchParameters sp = new vdxSearchParameters("SelectionOrderByDescr");
sp.Add("Descr", this.txtOrderDescr.Text);
sp.Add("OrderBy", orderBy, vdxSearchType.Other));

return sp;
```

Codesegment 7 Überschreiben der FillSearchParameters-Methode in SelectionOrder

1.5.8 Konfiguration des *ExplorerOrder*-Controls

Fügen Sie dem *ExplorerOrder*-Control eine neue ImageList hinzu und benennen diese *imageList*.
Fügen Sie der *imageList* folgende Icons hinzu.

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

Tabelle 15 Konfiguration ImageList auf ExplorerOrder

Folgende Tabelle fasst die Einstellungen der vier wichtigsten Properties zusammen:

Property	Selektion
SearchControlType	<ClientProjectNamespace>.Order.SelectionOrder
SelectionGridType	<ClientProjectNamespace>.Order.SelectionGridOrder
DisplayMode	Selection
ImageList (auf treeView)	imageList

Tabelle 16 Konfiguration von ExplorerOrder

Kopieren Sie die folgenden Namespace-Referenzen in den Code.

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

Codesegment 8 Using-Anweisung von ExplorerOrder

1.5.9 Überschreiben der *LoadData*-Methode im ExplorerOrder-Control

Fügen Sie den folgenden Code in die Methode ein.

```
protected override LoadData()
{
    dsOrderList ds = WebServiceProvider.wsOrder.GetDataSetOrderList(
        SearchParameters.GetParameters());
    if (ds.Ordering.Count == 0)
    {
        MessageBox.Show("No order found. Please try again!",
          "Order Search", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    return ds;
}
```

Codesegment 9 Überschreiben der LoadData-Methode in ExplorerOrder

1.5.10 Überschreiben der *GetRootNode*-Methode im ExplorerOrder-Control

Fügen Sie den folgenden Code in die Methode ein.

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

Codesegment 10 Überschreiben der GetRootNode-Methode in ExplorerOrder

1.5.11 Überschreiben der *GetTreeNode*-Methode im ExplorerOrder-Control

Fügen Sie den folgenden Code in die Methode ein.

```
StringBuilder sb = new StringBuilder();
dsOrderList.OrderingRow row = ((vdxNavigatorInfoItemRow)navInfoItem).DataRow as
       dsOrderList.OrderingRow;

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

if (row != null)
{
    if (!row.IsOrderDescrNull())
    {
       sb.Append(row.OrderDescr);
    }
    else
    {
       sb.Append("New Order");
    }
}
return new vdxTreeNode(sb.ToString(), 2, 1, false);
```

Codesegment 11 Überschreiben der GetTreeNode-Methode in ExplorerOrder

1.5.12 Überschreiben der *GetLoaderInfo*-Methode im ExplorerOrder-Control

Fügen Sie den folgenden Code in die Methode ein.

```
vdxSearchParameters sp = null;
dsOrderList.OrderingRow row =
       navigatorInfo.SelectedRow as dsOrderList.OrderingRow;

if (row != null)
{
    sp = new vdxSearchParameters("Order");
    sp.Add("OrderId", row.OrderID);
}
return new vdxLoaderInfo(sp, typeof(DataDetailOrder));
```

Codesegment 12 Überschreiben der GetLoaderInfo-Methode in ExplorerOder

1.5.13 Programmierung DataDetailOrder

Um Bestellungen zu ändern, löschen und speichern zu können, muss das Property-*Updater* vom *DataStatusManager* auf den *Order*-WebService gesetzt werden. Damit Artikel der Bestellung hinzugefügt oder gelöscht werden können, muss das Event *ReadBindingsCompleted* abonniert werden um die *TableManagers* zu initialisieren. Fügen Sie den Code nach dem *InitializeComponent*-Aufruf ein:

```
// This call is required by the Windows.Forms Form Designer.
InitializeComponent();
this.DataStatusManager.Updater = WebServiceProvider.wsOrder;
this.DataStatusManager.ReadBindingsCompleted +=
   new System.EventHandler(this.DataStatusManager_ReadBindingsCompleted);
```

Codesegment 13 Implementation des DataDetailOrder-Konstruktors

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

Hinzufügen der TableManager Variablen.

```
private vdxTableManager tmOrder    = null;
private vdxTableManager tmOrderPos = null;
```

Codesegment 14 Definition der TableManager-Variablen

Implementation der *DataStatusManager_ReadBindingsCompleted*-Methode

```
//store Tablemanagers in local variables
this.tmOrder = this.DataStatusManager.TableManagers["Ordering"];
this.tmOrderPos = this.DataStatusManager.TableManagers["OrderPos"];
this.tmOrderPos.CurrencyManager.PositionChanged +=
       new EventHandler(CurrencyManager_PositionChanged);
this.DataStatusManager.Filled += new EventHandler(DataStatusManager_Filled);
```

Codesegment 15 Implementation der ReadBindingsCompleted-Listeners von DataStatusManager

Damit die *OrderPos*-Controls deaktiviert werden, falls keine Position erfasst ist, muss die Methode *EnableControls* implementiert werden.

```
private void EnableControls()
{
    bool enableControls = (this.tmOrderPos.CurrencyManager.Count > 0);
    this.btnDelete.Enabled = enableControls;
    this.lblItem.Enabled    = enableControls;
    this.pickItem.Enabled   = enableControls;
    this.lblQuantity.Enabled = enableControls;
    this.txtQuantity.Enabled = enableControls;
    this.txtOrderPosDescr.Enabled = enableControls;
    this.lblOrderPosDescr.Enabled = enableControls;
}
```

Codesegment 16 OrderPos-Controls aktivieren/Deaktivieren

Implementation der *CurrencyManager_PositionChanged*-Methode

```
private void CurrencyManager_PositionChanged(object sender, EventArgs e)
{
    EnableControls();
}
```

Codesegment 17 Implementation des PositionChanged-Listeners von OrderPos-CurrencyManager

Implementation der *DataStatusManager_Filled*-Methode

```
private void DataStatusManager_Filled(object sender, EventArgs e)
{
    EnableControls();
}
```

Codesegment 18 Implementation der Filled-Listeners von DataStatusManager

1.5.14 Überschreiben der *LoadData*-Methode im *DataDetailOrder*-Control

Fügen Sie den folgenden Code in die Methode ein.

```
return WebServiceProvider.wsOrder.GetDataSetOrderDetail(
       searchParameters.GetParameters());
```

Codesegment 19 Überschreiben der LoadData-Methode in DataDetailOrder

1.5.15 Überschreiben der *GetInfoBarText*-Methode im *DataDetailOrder*-Control

Fügen Sie den folgenden Code in die Methode ein.

```
dsOrderDetail ds = DataStatusManager.DataSet as dsOrderDetail;
if (ds != null && ds.Ordering.Count == 1)
{
    dsOrderDetail.OrderingRow row = (dsOrder.OrderingRow )ds.Ordering.Rows[0];
    return (string)row.OrderDescr;
}
return string.Empty;
```

Codesegment 20 Überschreiben der GetInfoBarText-Methode in DataDetailOrder

1.5.16 Programmierung des *vdxActionItem*

Öffnen Sie das *mySampleClientMainForm* in Design-Ansicht und fügen Sie ein neues ActionItem dem *actionManager*-Control hinzu. Benennen Sie das ActionItem in aiExploreOrder um und abonnieren Sie den ActionInvoked-Event.

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

```
this.dataMultiContainer.Show(typeof(Order.ExplorerOrder));
```

Codesegment 21 Implementation des ActionItem-ExploreOrder-Invoked-Listeners

1.6 Erweiterung

Damit der User via *Add*- bzw. *Delete*-Button neue Items zur Bestellung hinzufügen bzw. löschen kann, müssen zwei EventListener-Methoden implementiert werden.

1.6.1 Erstellen der *btnAdd_Click*-EventListener-Methode

Öffnen Sie die Design-Ansicht der *DataDetailOrder*-Klasse und doppelklicken Sie auf den *Add*-Button. Dies fügt die Click-EventListener-Methode *btnAdd_Click* in die Klasse *DataDetailOrder* ein. Fügen Sie nun den folgenden Code in diese Methode ein:

```
private void btnAdd_Click(object sender, System.EventArgs e)
{
    if (this.tmOrder.CurrentDataRow != null)
    {
       this.tmOrder.CurrencyManager.EndCurrentEdit();
       this.tmOrderPos.CurrencyManager.EndCurrentEdit();
       this.tmOrderPos.Add();
       // Fremdschüssel setzen
       this.tmOrderPos.CurrentDataRow["OrderID"] =
              this.tmOrder.CurrentDataRow["OrderID"];
    }
}
```

Codesegment 22 Implementation des Click-Listeners vom Add-Button

Bei einem Klick auf den *Add*-Button wird die *Add*-Methode des *OrderPos-vdxTableManagers* aufgerufen. Dieser *vdxTableManager* verwaltet die Items, die zu einer Bestellung gehören.

1.6.2 Erstellen der *btnDelete_Click*-EventListener-Methode

Öffnen Sie die Design-Ansicht der *DataDetailOrder*-Klasse und doppelklicken Sie auf den *Delete*-Button. Dies fügt die Click-EventListener-Methode *btnDelete_Click* in die Klasse *DataDetailOrder* ein. Fügen Sie nun den folgenden Code in diese Methode ein:

```
private void btnDelete_Click(object sender, System.EventArgs e)
{
    this.tmOrderPos.Delete();
}
```

Codesegment 23 Implementation des Click-Listeners vom Delete-Button

Bei einem Klick auf den *Delete*-Button wird die *Delete*-Methode des *OrderPos-vdxTableManagers* aufgerufen. Dieser *vdxTableManager* verwaltet die Items, die zu einer Bestellung gehören.

Kompilieren und starten Sie die Anwendung um den implementierten Code zu testen.

1.7 Bildverzeichnis

1.8 Tabellenverzeichnis

1.9 Codesegment-Verzeichnis