Visual.NET Extensions – Referenz-Tutorial

“Erstellung des Middle-Tiers einer Client/Server-Applikation mit dem VDX Framework“

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 Erstellung des Middle-Tiers einer Client/Server-Applikation mit dem VDX Framework

In diesem Kapitel wird der *Middle Tier* einer exemplarischen Datenbank-Anwendung mit Hilfe der VDX-Klassenbibliothek erläutert.

1.1 Einleitung

Das Entitätsdiagramm (Abbildung 2) zeigt alle zur Verfügung stehenden Entitäten bzw. Tabellen der vdxSampleApp-Datenbank. Wobei dieses Tutorial sich auf einige Basis-Entitäten beschränkt.

- Country (Land)
- Lang (Sprache)
- AddrType (Industrie-Branche)
- Address (Firmeneintrag)
- Pers (Person)
- Ordering (Bestellung)
- OrderPos (Bestellposition)
- Item (Artikel)

Es wird unterschieden zwischen DataAccess-Fall(1), Entity-Fall(2) und Viewer-Fall(3).

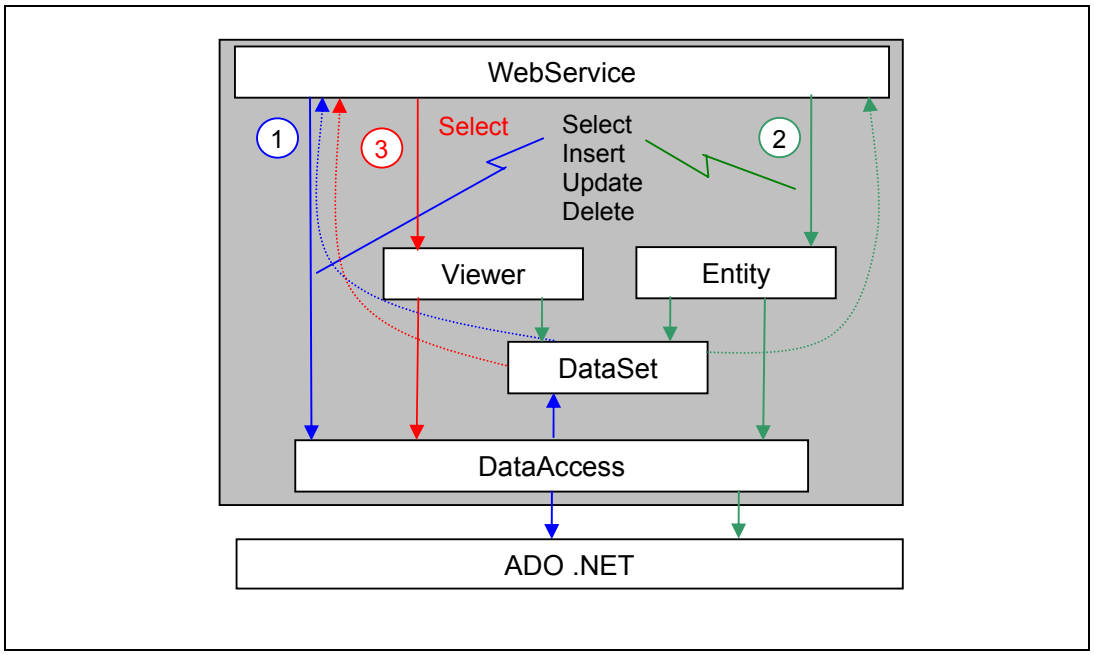


Abbildung 1 Kommunikation zwischen WebService und ADO.NET

DataAccess-Fall ①

Der einfachste Fall um Daten von einer Tabelle oder mehreren auszulesen, hinzuzufügen, zu verändern oder zu löschen. Der Rückgabewert des DataSets besteht nur aus einer Tabelle, wobei diese eine Zusammenfassung von mehreren Tabellen sein kann. Umgekehrt werden die Daten aus der Übergabetabelle auf die einzelnen physischen Tabellen abgefüllt bzw. aktualisiert.

Entity-Fall ②
Eignet sich besonders für die BusinessLogik von Entitäten[1], Objekt-Modelle bis zur Abstraktion von Entitäten. Hier wird das DataAccess Objekt vom EntityObjekt gesteuert. Vorteil, im EntityObjekt können Validierungen durchgeführt werden.

Viewer-Fall ③
Geeignet für Zusammensetzungen von Daten (verwalten von mehreren DataAccess-Objekte) bzw. Teilmengen. Z. B. arbeitet man auf Client Seite mit einem TreeView, welcher nicht alle Spalten einer Tabelle anzeigen soll, macht es Sinn mit einem Viewer zu arbeiten.

[1] Unter Entität wird die Abbildung eines Datenbankteils, i.d.R. eine Tabelle, verstanden.

1.2 Entitätsdiagramm

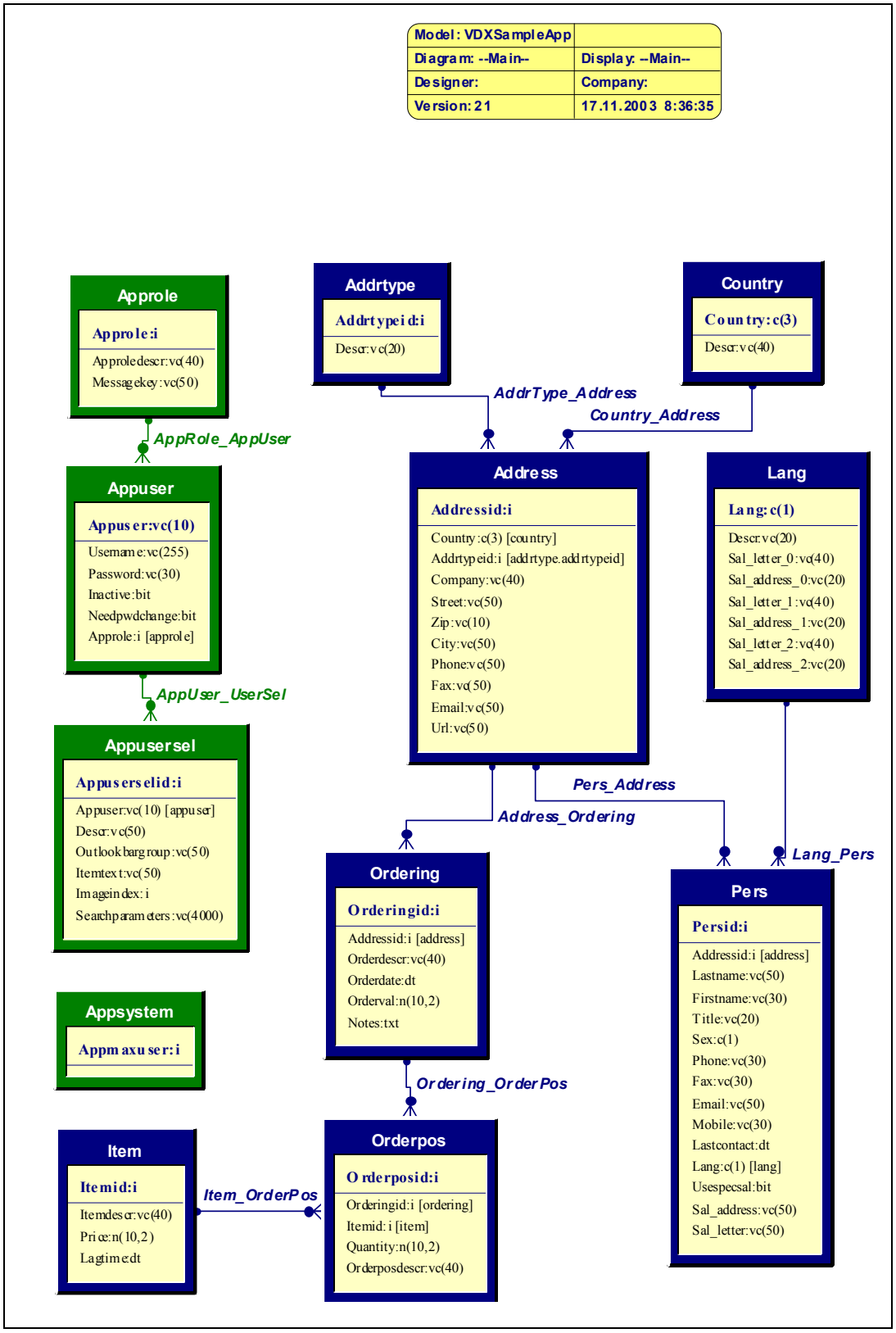


Abbildung 2 Entitätsdiagramm

1.3 Definition der Namespaces

Die Platzhalter stehen für die konfigurierten Namespaces im *ApplicationWizard*.

Platzhalter	Beispiel	Beschreibung
<DalProjectNamespace>	mySampleDal	Default-Namespace für DAL Projekt.
<SharedProjectNamespace>	mySampleShared	Default-Namespace für Shared Projekt.
<WebServiceName>	mySampleWebServices	Name des WebService-Projekts
<WebServiceNamespace>	mySampleWebServices	Default-Namespace für WebService Projekt.

Tabelle 1 Definition Namespaces

1.4 Zugriff auf Country-Tabelle erstellen

1.4.1 DataSet Country hinzufügen

Ein DataSet wird benötigt um Daten zwischen der Datenbank und dem Client zu versenden. DataSet Country wird verwendet um Länder von der Datenbank-Tabelle Country dem ExplorerCountry und DataDetailCountry zur Verfügung zu stellen und um Länder vom Client in die Datenbank-Tabelle zurück zu speichern, zu aktualisieren oder zu löschen.

1. Fügen Sie im Solution Explorer dem Ordner DataSets ein neues DataSet hinzu (via DataSets → Add → Add new Item...) und benennen Sie es *dsCountry*.

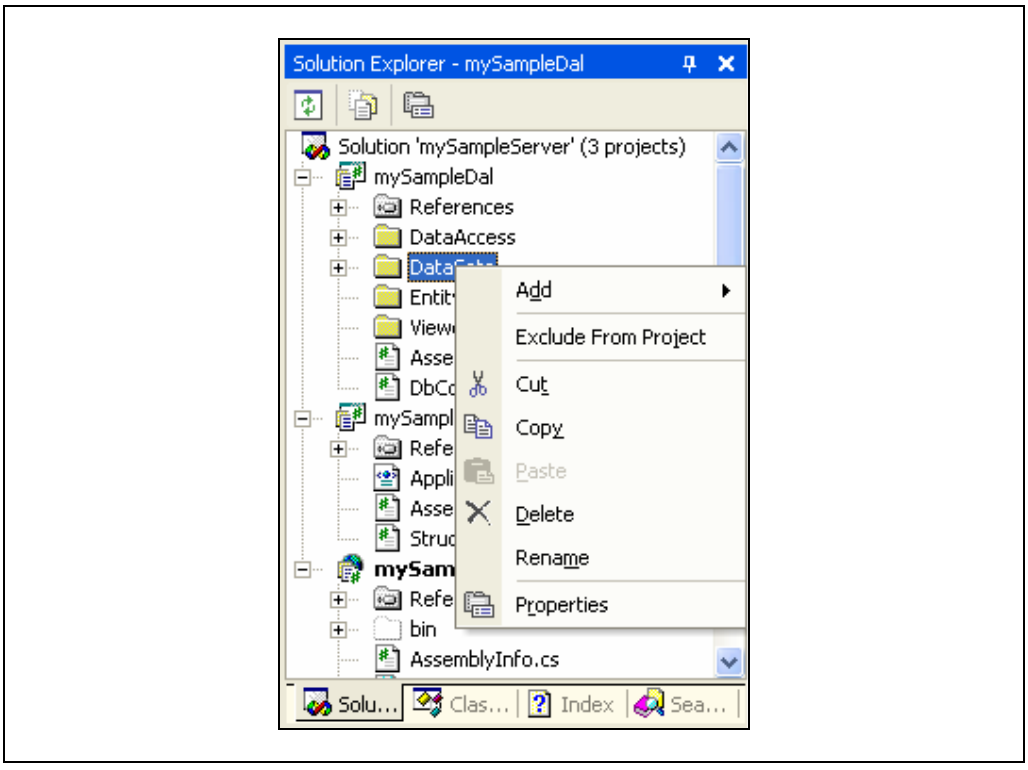


Abbildung 3 Solution Explorer

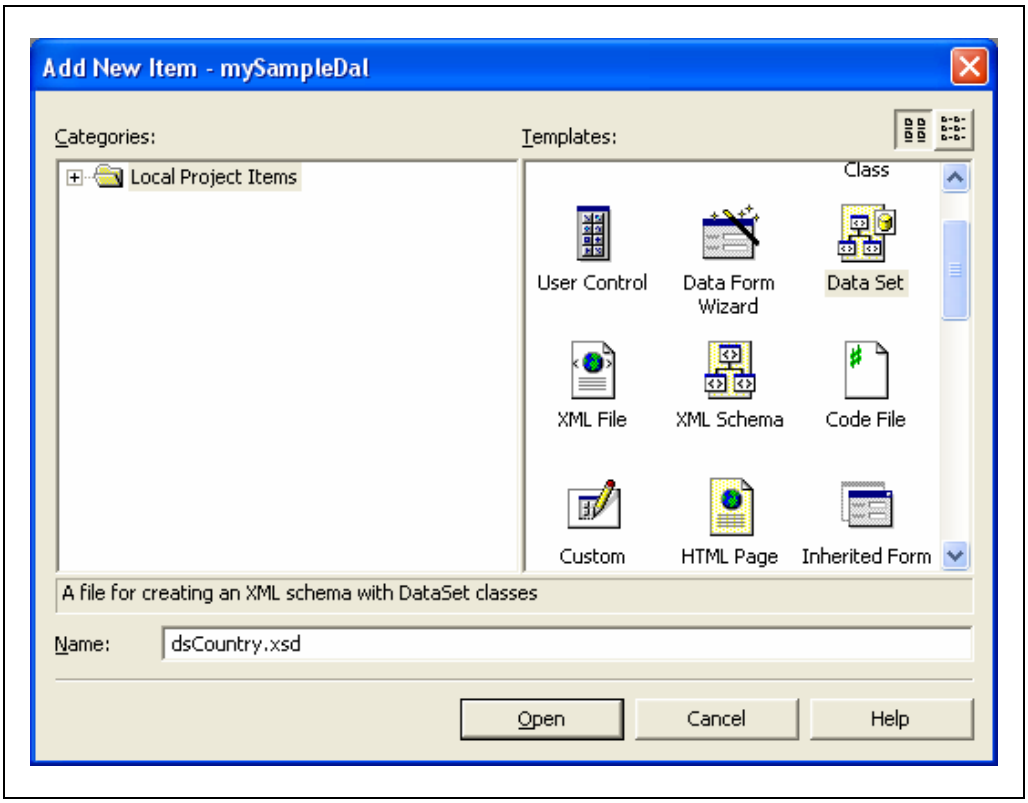


Abbildung 4 Add New Item Dialog

2. Öffnen Sie den Server Explorer und navigieren Sie zur Datenbank *VdxSampleApp* mit der Tabelle Country.

Tipp Falls der Server Explorer noch nie verwendet wurde, muss dieser über *View* → *Server Explorer* (Abbildung 5) eingeblendet werden.

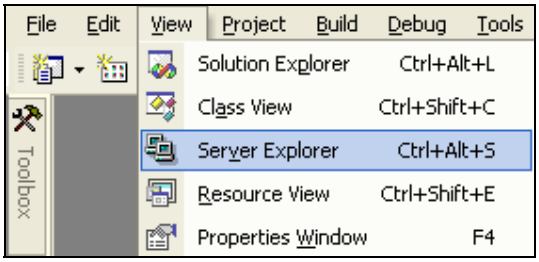


Abbildung 5 Menü View, Server Explorer

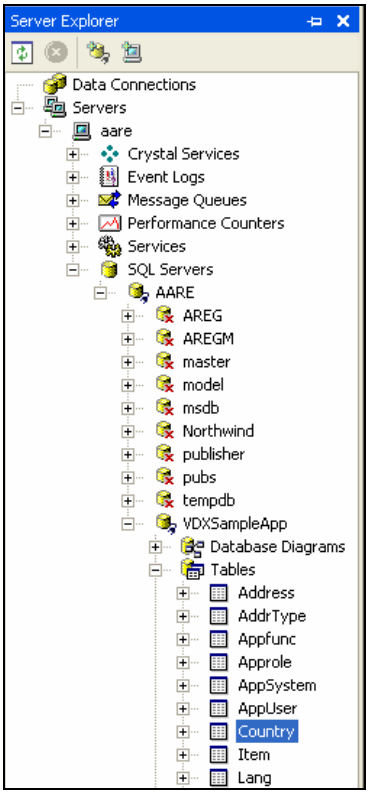


Abbildung 6 Server Explorer

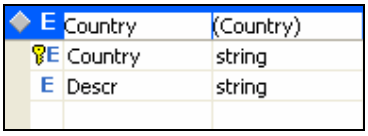


Abbildung 7 DataSet dsCountry

3. Ziehen Sie per Drag and Drop die Tabelle auf das neue DataSet.

4. Öffnen Sie den Property Browser, selektieren Sie die Spalte Country und setzen Sie den Default-Wert auf „new“.

Hinweis Alle DataSets, die im vdxServer-Framework verwendet werden, müssen konfiguriert werden. Falls ***Primär-Schlüssel*** ein ***numerischer*** Typ ist, dann muss ***AutoIncrementStep*** auf ***-1*** gesetzt werden.

1.4.2 DataAccess-Klasse Country hinzufügen

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

1. Fügen Sie im Ordner *DataAccess* eine neue Klasse hinzu (via *Project → Add Class...*) und benennen Sie die Klasse *daCountry*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 1 Using-Anweisung für daCountry

3. Das VDX-Server Framework bietet die zwei Klassen *SQLDataAccessDataSet* und *SQLDataAccessQuery* an um SQL-Anweisungen zu generieren.

Entscheidungskriterien	SQLDataAccess DataSet	SQLDataAccess Query
Es können aus der SELECT Anweisung, UPDATE, INSERT und DELETE Anweisungen generiert werden	X	
Es wird ein JOIN von mehreren Tabellen verwendet		X

Tabelle 2 Entscheidungskriterien für SQLDataAccess

Für die Country-Tabelle können alle Anweisungen aus der SELECT-Anweisung generiert werden. Somit verwenden wir die Klasse *SQLDataAccessDataSet*.

Fügen Sie der Klasse die Ableitung von *SQLDataAccessDataSet* hinzu und kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daCountry : SQLDataAccessDataset
   {
      public daCountry()
      {
         // Create select statement
         SelectCommandDynamicWhere select = new
                  SelectCommandDynamicWhere();
         select.Statement = "SELECT * FROM Country";
         select.ParameterMapping.Add
            (new ParameterMapping("Country", "Country"));
         select.ParameterMapping.Add
            (new ParameterMapping("Descr", "Descr"));
         SqlSelect = select;

         // Read connection string for the needed Database
         ConnectionString = DbConnection.GetConnectionString();
      }
   }
}
```

Codesegment 2 Implementation des daCountry-Konstruktors

4. Überschreiben Sie die *AdaptSqlParameters*-Methode.
   Dies muss gemacht werden damit die Konvertierung von C#-Typen auf SQL-Typen gewährleistet ist für jede Spalte in der Tabelle Country.

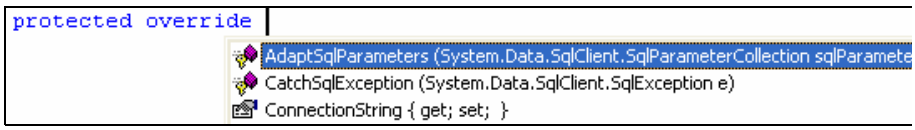
5. Selektieren Sie die *Class View*-Ansicht des Projektes (via Menu *View* → *Class View*).

6. Navigieren Sie zur *AdaptSqlParameters*-Methode; diese befindet sich in der Klasse *SQlDataAccess* (<DalProjectNamespace> → *daCountry* → Bases and Interfaces → *SQLDataAccessDataset* → Bases and Interfaces → *SQLDataAccess*).

7. Selektieren Sie den Befehl *Add* → *Override* aus dem Kontextmenu der *SQLDataAccess*-Klasse und klicken Sie auf die Methode *AdapSqlParameters*. Dies fügt eine leere Methode in die *daCountry*-Klasse ein, die diejenige aus der Basisklasse überschreibt.

Tipp

Überschreiben einer Methode funktioniert auch über Code-Completion, geben Sie **override** mit nachfolgendem Space ein und wählen Sie die Methode *AdaptSqlParameters* aus der Liste aus.



8. Fügen Sie in die *AdapSqlParameters*-Methode den folgenden Code ein.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("Country", SqlDbType.Char, 3);    // String>Char(3)
    TransformParameter("Descr", SqlDbType.VarChar, 40); // String>VarChar
}
```

Codesegment 3 Überschreiben der AdaptSqlParameters-Methode von daCountry

1.4.3 Web-Service hinzufügen und Web-Methoden auskodieren

1. Fügen Sie einen neuen Web Service hinzu (via <WebServiceName>→ Add → Add Web Service...) und benennen Sie diesen um nach *wsCountry.asmx*.

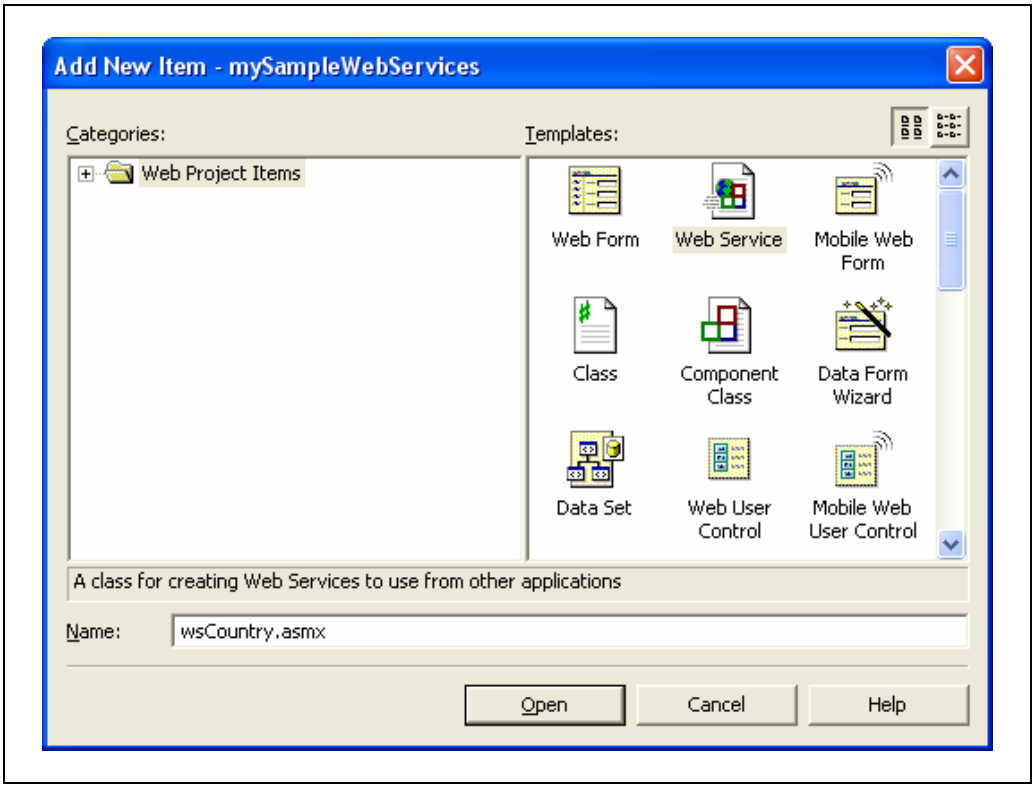


Abbildung 8 Add New Web Service Dialog

2. Um komfortabel auf die VDX-Klassen zugreifen zu können, fügen Sie die folgenden Namespace-Referenzen in den Code ein.

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.DataSets;
```

Codesegment 4 Using-Anweisung für wsCountry

Hinweis Der Platzhalter ***<DalProjectNamespace>*** muss ersetzt werden mit dem Namespace Ihres **D**ata**A**ccess**L**ayer-Projektes.

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsCountry.asmx")]
public class wsCountry : System.Web.Services.WebService
{
    ...
}
```

Codesegment 5 Klassenattribut von wsCountry

4. Fügen Sie die Methode *GetDataSetCountry* ohne Parameter hinzu.

```
[WebMethod]
public dsCountry GetDataSetCountry()
{
    dsCountry dsRet = new dsCountry();
    daCountry da = new daCountry();
    da.Table = dsRet.Country;
    da.Read();
    return dsRet;
}
```

Codesegment 6 Implementation der GetDataSetCountry-Methode von wsCountry

Mit dieser Methode können alle Einträge der Tabelle Country gelesen werden.

5. Fügen Sie die Methode *GetDataSetCountry* mit Parameter hinzu.

Um das Suchen nach einem bestimmten Land zu vereinfachen, werden die *vdxSearchParameters* verwendet. Wird auf Client Seite eine Suche nach dem Feld „Descr" (Description) abgesetzt, so wird das Property *VdxSearchName* auf „*SelectionCountryByDescr*" gesetzt und der Wert im *vdxSearchParameter* „Descr" gespeichert. Analog dazu wird für das Feld Country das Property *VdxSearchName* auf „*SelectionCountryById*" gesetzt.

VdxSearchName	Sp	Mögliche Filter	Filter	Wert
SelectionCountryByDescr	Descr	beliebig	StartsWith	Sch
SelectionCountryById	Country	Equal	Equal	CH

Tabelle 3 SearchNames für Country-WebService

Damit die SELECT-Anweisung korrekt zusammegestellt wird, muss jeweils der *vdxSearchParameter* in ein Parameter umgesetzt werden und dieser wiederum dem DataAccess-Objekt hinzugefügt werden.

```
[WebMethod(MessageName="GetDataSetCountryWithSearchParameters")]
public dsCountry GetDataSetCountry(object[] parameters)
{
    // DataAccess and DataSet initialization
    dsCountry dsRet = new dsCountry();
    daCountry da = new daCountry();

    try
    {
        // vdxSearchParameters deserialization
        vdxSearchParameters sp = new vdxSearchParameters(parameters);

        switch(sp.VdxSearchName)
        {
            case "SelectionCountryByDescr":
                string descr = Convert.ToString(sp["Descr"].Value);
                if (descr != null && descr.Length > 0)
                {
                    // Set search criteria on dataAccess object
                    da.AddParameter(new Parameter(
                           "Descr", Filter.CreateFilter(sp, "Descr")));
                }
                break;

            case "SelectionCountryById":
                string country = Convert.ToString(sp["Country"].Value);
                if (country != null && country.Length > 0)
                {
                    // Set search criteria on dataAccess object
                    da.AddParameter(new Parameter("Country", country));
                }
                break;
        }
        da.Table = dsRet.Country;
        da.Read();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not get dsCountry.", ex);
    }
    return dsRet;
}
```

Codesegment 7 Implementation der GetDataSetCountry-Methode mit Argument von wsCountry

Hinweis Erzeugen Sie eine Consts Datei mit allen nötigen *SearchNames* und *ParameterNames* im Project *mySampleShared*. Damit können Schreibfehler vermieden werden beim setzen des *SearchNames* und *ParameterNames* auf Server sowie auf Client Seite.

6. Fügen Sie die Methode *UpdateDataSet* hinzu. Damit können Länder erzeugt, verändert und gelöscht werden.

```
[WebMethod]
public dsCountry UpdateDataSet(dsCountry dsToUpdate)
{
    try
    {
        daCountry da = new daCountry();
        da.Table = dsToUpdate.Country;
        da.DataDrivenUpdate(); // INSERT, UPDATE or DELETE records
                               depending on the RecordState
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not update dsCountry.", ex);
    }
    return dsToUpdate;
}
```

Codesegment 8 Implementation der UpdateDataSet-Methode von wsCountry

1.5 Zugriff auf AddrType-Tabelle erstellen

1.5.1 DataSet AddrType hinzufügen

DataSet AddrType wird verwendet um Adresstypen von der Datenbank-Tabelle AddrType dem DataDetailAddress zur Verfügung zu stellen um eine Auswahl aller Adresstypen anzubieten.

Siehe Country Fall.

1. Fügen Sie dem Ordner DataSets ein neues DataSet hinzu und benennen Sie dieses *dsAddrType.xsd*.

2. Setzen Sie für Spalte *AddressID* das Property *AutoIncrementStep* auf -1.

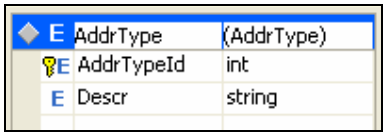


Abbildung 9 DataSet dsAddrType

Hinweis Alle DataSets, die im vdxServer-Framework verwendet werden, müssen konfiguriert werden. Falls ***Primär-Schlüssel*** ein ***numerischer*** Typ ist, dann sollte ***AutoIncrementStep*** auf ***-1*** gesetzt werden.

1.5.2 DataAccess-Klasse AddrType hinzufügen

Siehe Country Fall.

1. Fügen Sie ein neue Klasse hinzu und benennen Sie diese *daAddrType.cs*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 9 Using-Anweisung von daAddrType

3. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daAddrType : SQLDataAccessDataset
   {
      public daAddrType()
      {
         // Create select statement
         SelectCommandDynamicWhere select = new
               SelectCommandDynamicWhere();
         select.Statement = "SELECT * FROM AddrType";
         select.ParameterMapping.Add
             (new ParameterMapping("AddrTypeId", "AddrTypeId"));
         select.ParameterMapping.Add
             (new ParameterMapping("Descr", "Descr"));
         SqlSelect = select;

         // After an insert or update operation, the data will be read,
         // cause to guaranty the right AddressId in the entity object
         ReadAfterWrite = true;
         ReadAfterWriteWhere = "WHERE AddrTypeId = @@IDENTITY";

         // Read connection string for the needed Database
         this.ConnectionString = DbConnection.GetConnectionString();
      }
   }
}
```

Codesegment 10 Implementation des daAddrType-Konstruktors

Hinweis

Hier muss das Property ***ReadAfterWrite*** und ***ReadAfterWriteWhere*** gesetzt werden. Weil der Primärschlüssel vom DB-Server vergeben wird, muss dieser nach dem INSERT über ein SELECT automatisch wieder gelesen werden können.

Regel: Ist der Primärschlüssel vom **Typ String**, muss das Property ReadAfterWrite **nicht** gesetzt werden (Default-Wert ist **false**). Ist aber der Primärschlüssel vom **Typ Integer**, dann muss ein *ReadAfterWrite* ausgeführt werden um den neu vergebenen Primärschlüssel zu erhalten, d. h. Property *ReadAfterWrite* muss auf **true** gesetzt werden.

4. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("AddrTypeId", SqlDbType.Int);
    TransformParameter("Descr", SqlDbType.VarChar, 20);
}
```

Codesegment 11 Überschreiben der AdaptSqlParameters-Methode von daAddrType

1.5.3 Web-Service hinzufügen und Web-Methoden auskodieren

Siehe Country Fall.

1. Fügen Sie einen neuen Web Service hinzu und benennen Sie diesen *wsAddrType.asmx*.

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

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.DataSets;
```

Codesegment 12 Using-Anweisung von wsAddrType

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsAddrType.asmx")]
public class wsAddrType : System.Web.Services.WebService
{
    ...
}
```

Codesegment 13 Klassenattribut von wsAddrType

4. Fügen Sie die Methode *GetDataSetAddrType* ohne Parameters hinzu.

```
[WebMethod]
public dsAddrType GetDataSetAddrType()
{
    dsAddrType dsRet = new dsAddrType();
    daAddrType da = new daAddrType();
    da.Table = dsRet.AddrType;
    da.Read();
    return dsRet;
}
```

Codesegment 14 Implementation der GetDataSetAddrType-Methode von wsAddrType

5. Fügen Sie die Methode *GetDataSetAddrType* mit Parameters hinzu.

```
[WebMethod(MessageName="GetDataSetAddrTypeWithSearchParameters")]
public dsAddrType GetDataSetAddrType(object[] parameters)
{
    // DataAccess and DataSet initialization
    dsAddrType dsRet = new dsAddrType();
    daAddrType da = new daAddrType();

    try
    {
        // vdxSearchParameters deserialization
        vdxSearchParameters sp = new vdxSearchParameters(parameters);

        switch(sp.VdxSearchName)
        {
            case "SelectionAddrTypeByDescr":
                string descr = Convert.ToString(sp["Descr"].Value);
                if (descr != null && descr.Length > 0)
                {
                    da.AddParameter(new Parameter(
                        "Descr", Filter.CreateFilter(sp, "Descr")));
                }
                break;

            case "SelectionAddrTypeById":
            case "AddrType":
                int addrTypeId = Convert.ToInt32(sp["AddrTypeId"].Value);
                if(addrTypeId > 0)
                {
                    da.AddParameter(
                        new Parameter("AddrTypeId", addrTypeId));
                }
                break;
        }
        da.Table = dsRet.AddrType;
        da.Read();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not get dsAddrType.", ex);
    }

    return dsRet;
}
```

Codesegment 15 Implementation der GetDataSetAddrType-Methode mit Argument von wsAddrType

6. Fügen Sie die Methode *UpdateDataSet* hinzu.

```
[WebMethod]
public dsAddrType UpdateDataSet(dsAddrType dsToUpdate)
{
    try
    {
        daAddrType da = new daAddrType();
        da.Table = dsToUpdate.AddrType;
        da.DataDrivenUpdate();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not update dsAddrType.", ex);
    }
    return dsToUpdate;
}
```

Codesegment 16 Implementation der UpdateDataSet-Methode von wsAddrType

1.6 Zugriff auf Adress-Tabelle erstellen

1.6.1 DataSet Address hinzufügen

DataSet Address wird verwendet um Adressen von der Datenbank-Tabelle Address dem DataDetailAddress zur Verfügung zu stellen und um Adressen vom Client in die Datenbank-Tabelle zurück zu speichern, zu aktualisieren oder zu löschen.

Siehe Country Fall.

1. Fügen Sie ein neues DataSet hinzu und benennen Sie es *dsAddress.xsd*
2. Setzen Sie für Spalte *AddressID* die Properties *AutoIncrementStep* auf -1 und *ReadOnly* auf false.

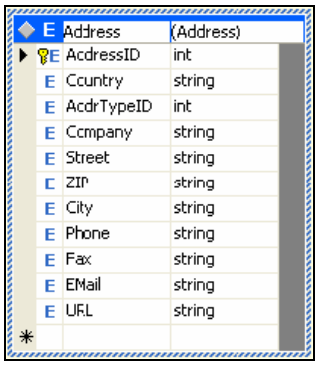


Abbildung 10 DataSet dsAddress

Hinweis Alle DataSets, die im vdxServer-Framework verwendet werden, müssen konfiguriert werden. Wird ein DataSet in einem *EntityObject* verwendet, muss das Property *ReadOnly* des *Primär-Schlüssels* auf *false* gesetzt werden.

1.6.2 DataSet AddressList hinzufügen

DataSet AddressList wird verwendet um Adressen von der Datenbank-Tabelle Address dem ExplorerAddress zur Verfügung zu stellen.

1. Fügen Sie ein neues DataSet hinzu und benennen Sie es *dsAddressList.xsd*.

2. Ziehen Sie die Tabellen Address und Pers auf das DataSet.

3. Löschen Sie alle Felder ausser Company, ZIP, City und AddressID aus der Tabelle Address.

4. Löschen Sie alle Felder ausser PersID, LastName, FirstName und AddressID aus der Tabelle Pers.

5. Selektieren Sie das Feld AddressID der Tabelle Pers und erstellen Sie eine Relation zur Tabelle Address.

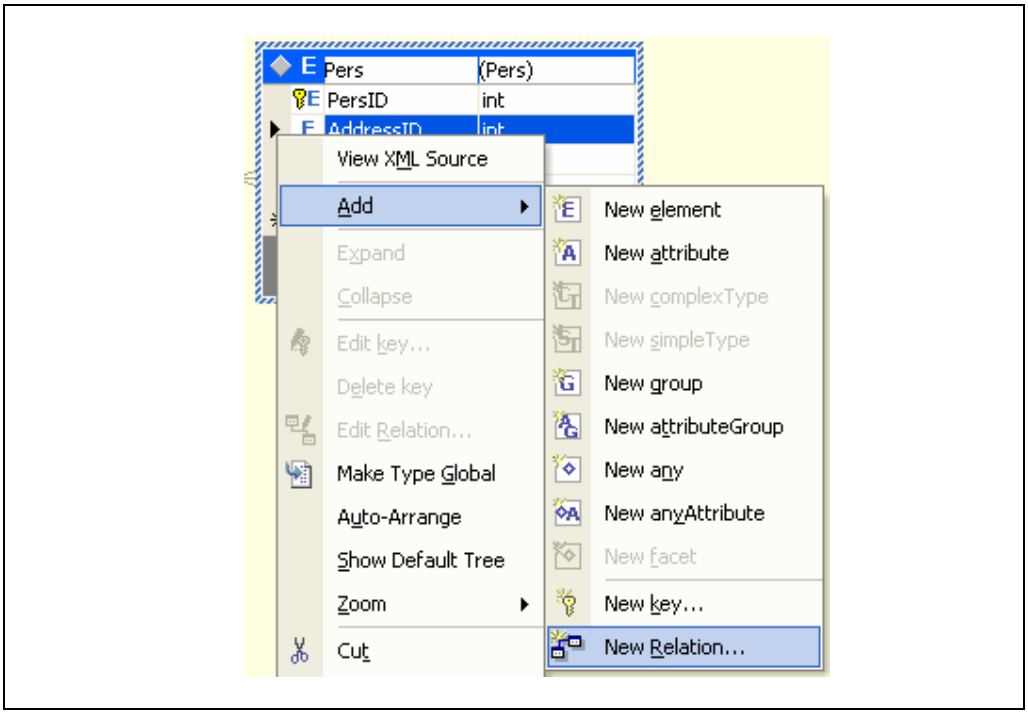


Abbildung 11 Add New Relation

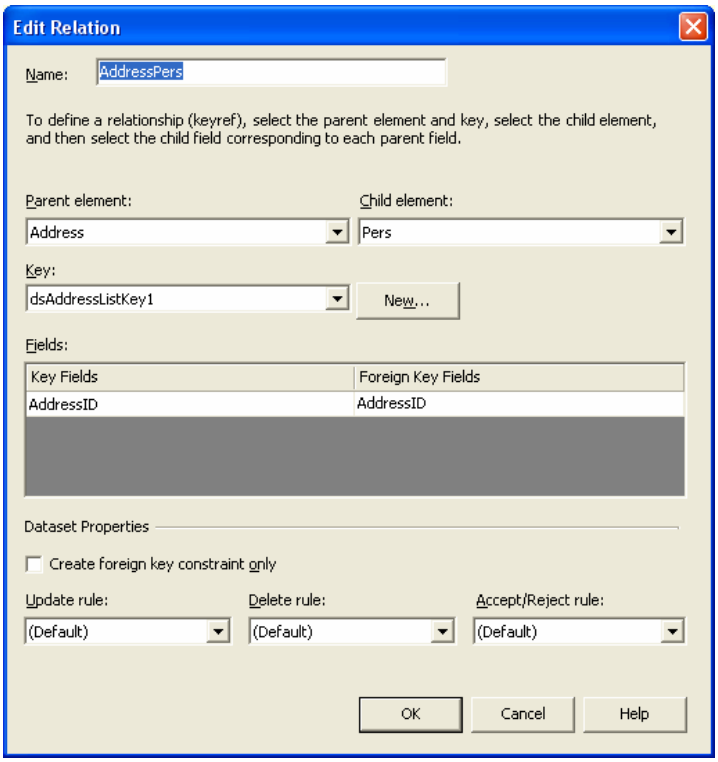


Abbildung 12 Relation Bearbeiten

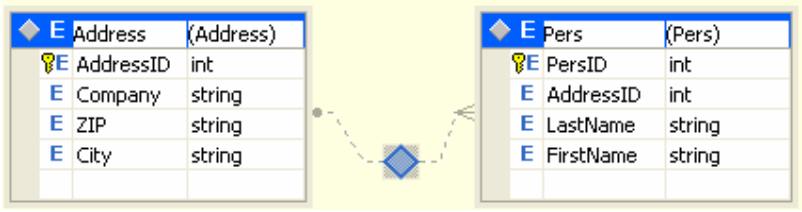


Abbildung 13 DataSet dsAddressList

1.6.3 DataAccess-Klasse Address hinzufügen

Siehe Country Fall.

1. Fügen Sie eine neue Klasse hinzu und benennen Sie die Klasse *daAddress.cs*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 17 Using-Anweisung von daAddress

3. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
    public class daAddress : SQLDataAccessDataset
    {
        public daAddress()
        {

            // Create select statement
            SelectCommandDynamicWhere select =
                    new SelectCommandDynamicWhere();
            select.Statement = "SELECT * FROM Address";
            select.ParameterMapping.Add
                     (new ParameterMapping("AddressId", "AddressId"));
            SqlSelect = select;

            ReadAfterWriteWhere = "WHERE AddressID = @@IDENTITY";

            // Read connection string for the needed Database
            ConnectionString = DbConnection.GetConnectionString();
         }
    }
}
```

Codesegment 18 Implementation des daAddress-Konstruktors

Hinweis Hier muss das Property ***ReadAfterWriteWhere*** gesetzt werden. Weil der Primärschlüssel vom DB-Server vergeben wird, muss dieser nach dem INSERT über ein SELECT automatisch wieder gelesen werden können. ***ReadAfterWrite*** wird im *eoAddress* gesetzt, siehe eoAddress erstellen.

4. Überschreiben Sie *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("AddressId", SqlDbType.Int);
}
```

Codesegment 19 Überschreiben der AdaptSqlParameters-Methode von daAddress

1.6.4 DataAccess-Klassen AddressList und PersonList hinzufügen

Siehe Country Fall.

1. Fügen Sie eine neue Klasse hinzu und benennen Sie die Klasse *daAddressList.cs*.

2. Fügen Sie die folgenden Namespace-Referenzen in den Code ein.

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 20 Using-Anweisung von daAddressList

3. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
    public class daAddressList  : SQLDataAccessDataset
    {
        public daAddressList()
        {
            // Create select statement
            SelectCommandDynamicWhere select =
                    new SelectCommandDynamicWhere();
            select.Statement =
                    "SELECT AddressId, Company, ZIP, City FROM Address";
            select.ParameterMapping.Add
                    (new ParameterMapping("AddressId", "AddressId"));
            select.ParameterMapping.Add
                    (new ParameterMapping("Company", "Company"));
            select.ParameterMapping.Add
                    (new ParameterMapping("ZIP", "ZIP"));
            select.ParameterMapping.Add
                    (new ParameterMapping("City", "City"));
            SqlSelect = select;

            TableName = "Address";

            // Read connection string for the needed Database
            ConnectionString = DbConnection.GetConnectionString();
        }
    }
}
```

Codesegment 21 Implementation des daAddressList-Konstruktors

4. Fügen Sie der Klasse folgendes Property hinzu.

Es soll möglich sein auf drei Arten Adressendaten zu sortieren:

- -Company
- -ZIP
- -City

Um diese Arten zu gewährleisten, muss von der Web-Methode aus die Sortierungsreihenfolge gesetzt werden können.

```
public string OrderBy
{
    set
    {
        SelectCommandDynamicWhere select =
                (SelectCommandDynamicWhere)SqlSelect;
        if (value != null && value != string.Empty)
        {
            select.OrderBy = "ORDER BY " + value;
        }
        else
        {
            select.OrderBy = string.Empty;
        }
    }
}
```

Codesegment 22 Implementation des OrderBy-Property von daAddressList

5. Überschreiben Sie die *AdaptSqlParameters* -Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("AddressId", SqlDbType.Int);
    TransformParameter("Company", SqlDbType.VarChar, 40);
    TransformParameter("ZIP", SqlDbType.VarChar, 10);
    TransformParameter("City", SqlDbType.VarChar, 50);
}
```

Codesegment 23 Überschreiben der AdaptSqlParameters-Methode von daAddressList

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

1. Fügen Sie der Datei *daAddressList*.cs eine neue Klasse *daPersonList hinzu*.

2. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
    public class daAddressList : SQLDataAccessDataset
    ...

    public class daPersonList : SQLDataAccessDataset
    {
        public daPersonList()
        {
            // Create select statement
            SelectCommandDynamicWhere select =
              new SelectCommandDynamicWhere();
            select.Statement =
              "SELECT PersID,AddressID,LastName,FirstName FROM Pers";
            select.ParameterMapping.Add
              (new ParameterMapping("AddressID", " AddressID"));
            select.OrderBy = "ORDER BY LastName, FirstName";
            SqlSelect = select;

            TableName = "Pers";

            // Read connection string for the needed Database
            this.ConnectionString = DbConnection.GetConnectionString();
        }
    }
}
```

Codesegment 24 Implementation des daPersonList-Konstruktors

3. Überschreiben Sie *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("AddressID", SqlDbType.Int);
}
```

Codesegment 25 Überschreiben der AdaptSqlParameters-Methode von daPersonList

1.6.5 Entity-Klasse Address hinzufügen

Um serverseitige Validierungen durchzuführen, ist es angebracht, eine Entity-Klasse zu verwenden.

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

1. Fügen Sie im Ordner Entity eine neue Klasse hinzu (via *Project→ Add Class...*) und benennen Sie die Klasse *eoAddress*.

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

```
using System.Data;
using Deag.Vdx.Common.Exceptions.LocalExceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 26 Using-Anweisung von eoAddress

3. Fügen Sie der Klasse folgende Properties hinzu.

```
public int AddressId
{
    get { return (int)GetFromCacheDefault("AddressID", -1); }
    set { SetToCache("AddressID", value); }
}

// Overrides DsCache to return typed DataSet
protected new dsAddress DsCache
{
    get { return (dsAddress)base.DsCache; }
    set { base.DsCache = value; }
}
```

Codesegment 27 Implementation der Properties AddressId und DsCache von eoAddress

Hinweis Es ist sinnvoll ***DsCache*** zu überschreiben um später ***typensicher*** zugreifen zu können.

4. Fügen Sie die Ableitung von EntityObject der Klasse hinzu und kodieren Sie den Konstruktor wie folgt.

Es gibt drei Möglichkeiten um ein EntityObject VDX konform zu konfigurieren:

- Property Nilable (zu finden im DataSet in Design-Ansicht) auf true setzen auf jeder Constraint Column im DataSet.
- Property *EnforceCacheConstraints* (EntityBase) auf false setzen damit das Framework eine leere DataRow angelegen kann.
- Property *AddNewRow* (EntityBase) auf false setzen um das Anlegen der neuen DataRow zu unterbinden.

```
namespace <DalProjectNamespace>.Entity
{
    public class eoAddress : EntityObject
    {
        public eoAddress()
        {
            // init typed DataSet
            EnforceCacheConstraints = false;
            DataSetType = typeof(dsAddress);

            // init data access object
            daAddress da = new daAddress();
            da.Table = DsCache.Address;
            da.ReadAfterWrite = true;
            AddAccessObject(da);
        }
    }
}
```

Codesegment 28 Implementation des eoAddress-Konstruktors

5. Überschreiben Sie die *RefreshDataAccessObject*-Methode weil alle Suchparameter von *daAddress* neu geladen werden sollen.

```
protected override void RefreshDataAccessObject(…)
{
    accessObject.ClearParameters();
    accessObject.AddParameters(GetParametersFromCache());
}
```

Codesegment 29 Überschreiben der RefreshDataAccessObject-Methode von eoAddress

Überschreiben Sie die Validate-Methode. Die *ValidateExceptions* mit den Ids grösser 5000 sind Applikationsspezifische Fehlermeldungen, die mit dem ExceptionEditor erfasst werden müssen. Siehe Benutzerhandbuch ExceptionEditor.

```
public override void Validate(…)
{
    if (!IsInCache("AddrTypeID"))
    {
        RecordException(transaction,
               new vdxValidateException(5003), ref cancel);
    }
    if (!IsInCache("Country"))
    {
        RecordException(transaction,
               new vdxValidateException(5008), ref cancel);
    }
    if (!IsInCache("Company"))
    {
        RecordException(transaction,
               new vdxValidateException(5009), ref cancel);
    }
}
```

Codesegment 30 Überschreiben der Validate-Methode von eoAddress

1.6.6 DataView-Klasse AddressList hinzufügen

Anstatt im WebService direkt auf das DataAccess Objekt eine Suche auszuführen, können die Suchparameter über einen Viewer gesetzt werden.

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

1. Fügen Sie im Ordner Viewer eine neue Klasse hinzu (via *Project→ Add Class...*) und benennen Sie die Klasse *dvAddressList*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using Deag.Vdx.Server.DAL.SQLServer;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 31 Using-Anweisung von dvAddressList

3. Fügen Sie der Klasse folgende *private* Felder und *public* Konstanten hinzu.

```
private Parameter pAddressId = new Parameter("AddressID");
private Parameter pCompany = new Parameter("Company");
private Parameter pZip = new Parameter("ZIP");
private Parameter pCity = new Parameter("City");

private daAddressList daAddressList;

public const string AddressView = "AddressView";
public const string PersonView = "PersonView";
```

Codesegment 32 Definition der Felder von dvAddressList

Hinweis Das Argument im Konstruktor von der Klasse *Parameter* entspricht einem Spaltennamen in der Tabelle Address.

4. Fügen Sie der Klasse die Ableitung von Viewer hinzu und kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.Viewer
{
   public class dvAddressList : Deag.Vdx.Server.DAL.Viewer
   {
     public dvAddressList()
     {
          this.daAddressList = new daAddressList();
          AddView(AddressView);
          AddView(PersonView);
     }
   }
}
```

Codesegment 33 Implementation des dvAddressList-Konstruktors

5. Fügen Sie der Klasse folgende Properties für die entsprechenden Felder hinzu.

```
public Filter AddressId
{
    set { this.pAddressId.Filter = value; }
}
public Filter Company
{
    set { this.pCompany.Filter = value; }
}
public Filter Zip
{
    set { this.pZip.Filter = value; }
}
public Filter City
{
    set { this.pCity.Filter = value; }
}
public string AddressOrderBy
{
    set { this.daAddressList.OrderBy = value; }
}
```

Codesegment 34 Implementation der Properties von dvAddressList

6. Überschreiben Sie die *RefreshDataAccessObject*-Methode.

```
protected override void RefreshDataAccessObject(…)
{
    // Depending on the active view add its parameters
    accessObject.ClearParameters();
    switch (viewName)
    {
      case AddressView:
            accessObject.AddParameter(this.pAddressId);
            accessObject.AddParameter(this.pCompany);
            accessObject.AddParameter(this.pZip);
            accessObject.AddParameter(this.pCity);
            break;

      case PersonView:
            accessObject.AddParameter(this.pAddressId);
            break;
     }
}
```

Codesegment 35 Überschreiben der RefreshDataAccessObject-Methode von dvAddressList

7. Überschreiben Sie die InitView-Methode.

```
protected override void InitView(…)
{
    view.DataSetType = typeof(dsAddressList);
    switch (viewName)
    {
       case AddressView:
           view.AddAccessObject(this.daAddressList);
           break;
       case PersonView:
           // Address DataSet must not be filled
           view.ConstraintAction =
              EntityBase.ConstraintActionType.Deactivate;
           view.AddAccessObject(new daPersonList());
           break;
    }
}
```

Codesegment 36 Überschreiben der InitView-Methode von dvAddressList

Hinweis ConstraintActionType muss deaktiviert werden, damit beide Tabellen unabhängig voneinander gefüllt werden können.

1.6.7 Web-Service hinzufügen und Web-Methoden auskodieren

Siehe Country Fall

1. Fügen Sie einen neuen Web Service hinzu und benennen Sie ihn *wsAddress.asmx*.

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

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.Entity;
using <DalProjectNamespace>.Viewer;
```

Codesegment 37 Using-Anweisung von wsAddress

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsAddress.asmx")]
public class wsAddress : System.Web.Services.WebService
{
    ...
}
```

Codesegment 38 Klassenattribut von wsAddress

4. Fügen Sie die Methode *GetDataSetAddress* mit Parameter hinzu.

```
[WebMethod]
public dsAddress GetDataSetAddress(object[] parameters)
{
    dsAddress dsRet = new dsAddress();

    try
    {
        eoAddress eo = new eoAddress();
        // vdxSearchParameters deserialization
        vdxSearchParameters sp = new vdxSearchParameters(parameters);
        int addressId = Convert.ToInt32(sp["AddressId"].Value);

        if (sp.VdxSearchName == "Address" && addressId > 0)
        {
            eo.AddressId = addressId;
            eo.Read();
            dsRet = (dsAddress)eo.Serialize();
        }
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not get dsAddress.", ex);
    }

    return dsRet;
}
```

Codesegment 39 Implementation der GetDataSetAddress-Methode von wsAddress

5. Fügen Sie die Methode *UpdateDataSet* hinzu.

```
[WebMethod]
public dsAddress UpdateDataSet(dsAddress dsToUpdate)
{
    dsAddress dsRet;
    Transaction trans = new Transaction();

    try
    {
       eoAddress eo = new eoAddress();
       eo.Deserialize(dsToUpdate);
       eo.DataDrivenUpdate();
       dsRet = (dsAddress)eo.Serialize();
       trans.Commit();
    }
    catch (vdxException ex)
    {
       trans.Rollback();
       throw ex;
    }
    catch (Exception ex)
    {
       trans.Rollback();
       throw new vdxException("Could not update dsAddress.", ex);
    }

    return dsRet;
}
```

Codesegment 40 Implementation der UpdateDataSet-Methode von wsAddress

6. Fügen Sie die Methode *GetDataSetAddressList* hinzu.

```
[WebMethod]
public dsAddressList GetDataSetAddressList(object[] parameters)
{
   // Viewer und DataSet initialization
   dvAddressList dv = new dvAddressList();
   dsAddressList dsRet = new dsAddressList();
   try
   {
      // vdxSearchParameters deserialization
      vdxSearchParameters sp = new vdxSearchParameters(parameters);
      switch(sp.VdxSearchName)
      {
         case "SelectionAddressById":
            int addressId = Convert.ToInt32(sp["AddressId"].Value);
            if (addressId > 0)
            {
               dv.AddressId = Filter.CreateFilter(sp, "AddressId");
            }
            dsRet = (dsAddressList)dv[dvAddressList.AddressView];

            break;

         case "SelectionAddressByDescr":
            dv.Company = Filter.CreateFilter(sp, "Company");
            dv.Zip = Filter.CreateFilter(sp, "Zip");
            dv.City = Filter.CreateFilter(sp, "City");
            if (sp["OrderBy"] != null)
            {
              dv.AddressOrderBy = Convert.ToString(sp["OrderBy"].Value);
            }
            dsRet = (dsAddressList)dv[dvAddressList.AddressView];

            break;

         case "PersTree":
            addressId = Convert.ToInt32(sp["AddressId"].Value);
            if (addressId > 0)
            {
               dv.AddressId = Filter.CreateFilter(sp, "AddressId");
            }
            dsRet = (dsAddressList)dv[dvAddressList.PersonView];

            break;
      }
   }
   catch (vdxException ex)
   {
      throw ex;
   }
   catch (Exception ex)
   {
      throw new vdxException("Could not get dsAddressList.", ex);
   }
   return dsRet;
}
```

Codesegment 41 Implementation der GetDataSetAddressList-Methode von wsAddress

1.7 Zugriff auf Language-Tabelle erstellen

1.7.1 DataSet Language hinzufügen

DataSet Language wird verwendet um Sprachen von der Datenbank-Tabelle Lang dem DataDetailPerson zur Verfügung zu stellen um eine Auswahl aller Sprachen anzubieten.

Siehe Country Fall.

1. Fügen Sie im Ordner DataSet ein neues DataSet hinzu und benennen Sie es *dsLang.xsd*.

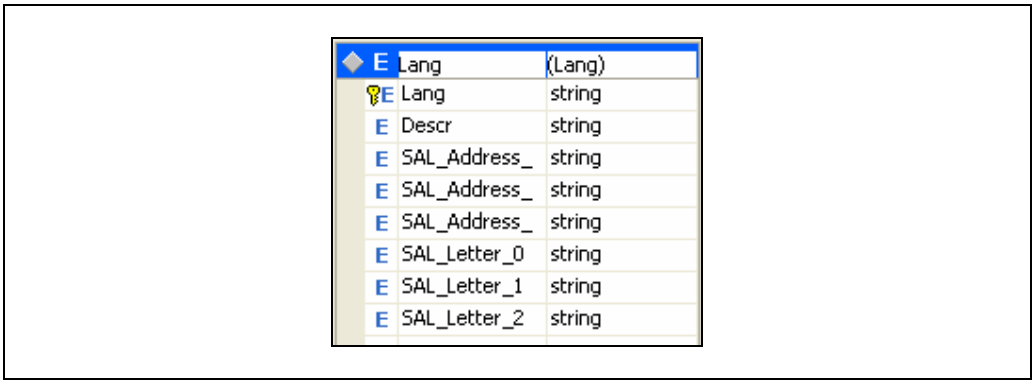


Abbildung 14 DataSet dsLang

1.7.2 DataAccess-Klasse Language hinzufügen

Siehe Country Fall.

1. Fügen Sie im Ordner DataAccess eine neue Klasse hinzu und benennen Sie es *daLang.cs*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 42 Using-Anweisung von daLang

3. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daLang : SQLDataAccessDataset
   {
      public daLang()
      {
         // Create select statement
         SelectCommandDynamicWhere select =
             new SelectCommandDynamicWhere();
         select.Statement = "SELECT * FROM Lang";
         select.ParameterMapping.Add
             (new ParameterMapping("Lang", "Lang"));
         select.ParameterMapping.Add
             (new ParameterMapping("Descr", "Descr"));
         SqlSelect = select;

         // Read connection string for the needed Database
         ConnectionString = DbConnection.GetConnectionString();
      }
   }
}
```

Codesegment 43 Implementation des daLang-Konstruktors

4. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("Lang", SqlDbType.Char, 1);
    TransformParameter("Descr", SqlDbType.VarChar, 20);
}
```

Codesegment 44 Überschreiben der AdaptSqlParameters-Methode von daLang

1.7.3 Web-Service hinzufügen und Web-Methoden auskodieren

Siehe Country Fall.

1. Fügen Sie einen neuen Web Service hinzu und benennen Sie ihn *wsLang*.asmx.

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

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.DataSets;
```

Codesegment 45 Using-Anweisung von wsLang

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsLang.asmx")]
public class wsLang : System.Web.Services.WebService
{
    ...
}
```

Codesegment 46 Klassenattribut von wsLang

4. Fügen Sie die Methode *GetDataSetLang* ohne Parameter hinzu.

```
[WebMethod]
public dsLang GetDataSetLang()
{
    dsLang dsRet = new dsLang();
    daLang da = new daLang();
    da.Table = dsRet.Lang;
    da.Read();
    return dsRet;
}
```

Codesegment 47 Implementation der GetDataSetLang-Methode von wsLang

5. Fügen Sie die Methode *GetDataSetLang* mit Parameter hinzu.

```
[WebMethod(MessageName="GetDataSetLangWithSearchParameters")]
public dsLang GetDataSetLang(object[] parameters)
{
    // DataAccess and DataSet initialization
    dsLang dsRet = new dsLang();
    daLang da = new daLang();

    try
    {
        // vdxSearchParameters deserialization
        vdxSearchParameters sp = new vdxSearchParameters(parameters);

        switch(sp.VdxSearchName)
        {
            case "SelectionLangByDescr":
                string descr = Convert.ToString(sp["Descr"].Value);
                if (descr != null && descr.Length > 0)
                {
                    da.AddParameter(new Parameter(
                        "Descr", Filter.CreateFilter(sp, "Descr")));
                }
                break;

            case "SelectionLangById":
                string lang = Convert.ToString(sp["Lang"].Value);
                if (lang != null && lang.Length > 0)
                {
                    da.AddParameter(new Parameter("Lang", lang));
                }
                break;
        }
        da.Table = dsRet.Lang;
        da.Read();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not get dsLang.", ex);
    }

    return dsRet;
}
```

Codesegment 48 Implementation der GetDataSetLang-Methode mit Argument von wsLang

6. Fügen Sie die Methode *UpdateDataSet* hinzu.

```
[WebMethod]
public dsLang UpdateDataSet(dsLang dsToUpdate)
{
    try
    {
        daLang da = new daLang();
        da.Table = dsToUpdate.Lang;
        da.DataDrivenUpdate();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not update dsLang.", ex);
    }
    return dsToUpdate;
}
```

Codesegment 49 Implementation der UpdateDataSet-Methode von wsLang

1.8 Zugriff auf Pers-Tabelle erstellen

1.8.1 DataSet Pers hinzufügen

DataSet Pers wird verwendet um Personen von der Datenbank-Tabelle Pers dem DataDetailPerson zur Verfügung zu stellen und um Personen vom Client in die Datenbank-Tabelle zurück zu speichern, zu aktualisieren oder zu löschen.

Siehe Country Fall.

1. Fügen Sie dem Ordner DataSet ein neues DataSet hinzu und benennen Sie es *dsPers.xsd*.
2. Setzen Sie für die Spalte *PersId* die Properties *AutoIncrementStep* auf -1 und *ReadOnly* auf false.
3. Fügen Sie ein neues string Feld *ADDRESSDESCR* dem DataSet hinzu.

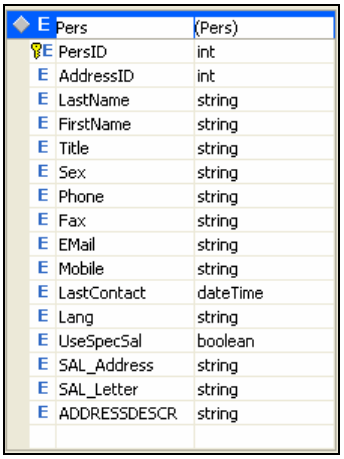


Abbildung 15 DataSet dsPers

1.8.2 DataSet PersList hinzufügen

1. Fügen Sie dem Ordner DataSet hinzu ein neues DataSet und benennen Sie es *dsPersList.xsd*.

2. Setzen Sie für die Spalte *PersId* die Property *AutoIncrementStep* -1.

3. Löschen Sie die Felder *Lang*, *LastContact*, *UseSpecSal*, *SAL_Address* und *SAL_Letter*.

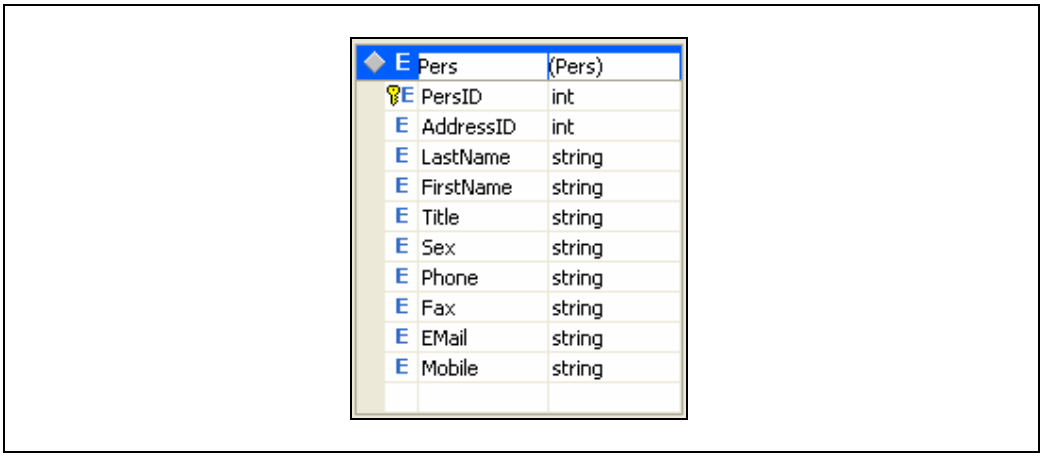


Abbildung 16 DataSet dsPersList

1.8.3 DataAccess-Klasse Pers hinzufügen

Siehe Country Fall.

1. Fügen Sie im Ordner DataAccess eine neue Klasse hinzu und benennen Sie diese *daPers.ds*.

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

```
using System.Data;
using System.Text;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 50 Using-Anweisung von daPers

3. Für diesen DataAccess-Fall muss von SQLDataAccessQuery abgeleitet werden, da die Information für das neue Feld *ADDRESSDESCR* von einer anderen Tabelle selektiert werden muss (INNER JOIN).

Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
  public class daPers : SQLDataAccessQuery
  {
    public daPers()
    {
      // Create select statement
      StringBuilder selcmd = new StringBuilder();
      selcmd.Append("SELECT PersID,Pers.AddressID,LastName,FirstName,Title,Sex, ");
      selcmd.Append("Pers.Phone, Pers.Fax, Pers.EMail, Mobile, LastContact, ");
      selcmd.Append("Lang, UseSpecSal, SAL_Address, SAL_Letter, ");
      selcmd.Append("(ISNULL(Address.Company, '')+ ' ' +ISNULL(Address.ZIP, '') +");
      selcmd.Append("' ' + ISNULL(Address.City, '')) AS ADDRESSDESCR");
      selcmd.Append("FROM Pers INNER JOIN Address");
      selcmd.Append("ON Pers.AddressID = Address.AddressID");

      SelectCommand select = new SelectCommand();
      Select.Query = selcmd.ToString() + "WHERE PersID=@PersID";
      SqlSelect = select;

      // Create insert statement
      InsertCommand insert = new InsertCommand();
      StringBuilder cmd = new StringBuilder();
      cmd.Append("INSERT INTO Pers (\n");
      cmd.Append("AddressID,LastName,FirstName,Title,Sex,Phone,Fax,Email, ");
      cmd.Append("Mobile,LastContact,Lang,UseSpecSal,SAL_Address,SAL_Letter)\n");
      cmd.Append("VALUES (\n");
      cmd.Append("@AddressID, @LastName, @FirstName, @Title, @Sex, @Phone," );
      cmd.Append("@Fax, @EMail, @Mobile, @LastContact, @Lang, @UseSpecSal, ");
      cmd.Append("@SAL_Address, @SAL_Letter); \n");
      // Return created ID
      cmd.Append(selcmd.ToString());
      cmd.Append("WHERE PersID=@@IDENTITY");
      insert.Query = cmd.ToString();
      SqlInsert = insert;

      // Create update statement
      UpdateCommand update = new UpdateCommand();
      cmd = new StringBuilder();
      cmd.Append("UPDATE Pers SET \n");
      cmd.Append("AddressID = @AddressID, LastName = @LastName, \n");
      cmd.Append("FirstName = @FirstName, Title = @Title, Sex = @Sex, \n");
      cmd.Append("Phone=@Phone, Fax=@Fax, EMail=@EMail, Mobile=@Mobile, \n");
      cmd.Append("LastContact=@LastContact,Lang=@Lang,UseSpecSal=@UseSpecSal,\n");
      cmd.Append("SAL_Address = @SAL_Address, SAL_Letter = @SAL_Letter ");
      cmd.Append("WHERE (PersID = @PersID); \n");
      update.Query = cmd.ToString();
      SqlUpdate = update;

      // Create delete statement
      DeleteCommand delete = new DeleteCommand();
      delete.Query =   "DELETE FROM Pers \nWHERE (PersID = @PersID)";
      SqlDelete = delete;

      // Read connection string for the needed Database
      ConnectionString = DbConnection.GetConnectionString();
    }
  }
}
```

Codesegment 51 Implementation des daPers-Konstruktors

4. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("PersID", SqlDbType.Int);
}
```

Codesegment 52 Überschreiben der AdaptSqlParameters-Methode von daPers

1.8.4 DataAccess-Klasse PersList hinzufügen

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

1. Fügen Sie im Ordner DataAccess eine neue Klasse hinzu und benennen Sie diese *daPersList.ds*.

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

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 53 Using-Anweisung von daPersList

3. Um Personen auch nach dem Land zu suchen, müssen die Tabellen Pers und Address selektiert werden. Hier kann die Klasse *SQLDataAccessDataset* verwendet werden, weil nur die SELECT-Anweisung benötigt wird. Kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daPersList : SQLDataAccessDataset
   {
      public daPersList()
      {
         // Create select statement
         SelectCommandDynamicWhere select =
            new SelectCommandDynamicWhere();
         // Attention:The column Phone is on both tables Pers and Address
         select.Statement =
            "SELECT LastName, FirstName, Title, Pers.Phone, PersId " +
            "FROM Pers INNER JOIN Address " +
            "ON Pers.AddressID = Address.AddressID";
         select.ParameterMapping.Add
            (new ParameterMapping("LastName", "LastName"));
         select.ParameterMapping.Add
            (new ParameterMapping("FirstName", "FirstName"));
         select.ParameterMapping.Add
            (new ParameterMapping("Country", "Address.Country"));
         SqlSelect = select;

         // Read connection string for the needed Database
         ConnectionString = DbConnection.GetConnectionString();
      }
   }
}
```

Codesegment 54 Implementation des daPersList-Konstruktors

4. Fügen Sie der Klasse folgende Properties für die entsprechenden Felder hinzu.

```
public string OrderBy
{
    set
    {
        ((SelectCommandDynamicWhere)SqlSelect).OrderBy =
            "ORDER BY " + value;
    }
}
```

Codesegment 55 Implementation des OrderBy-Properties von daPersList

5. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("PersID", SqlDbType.Int);
    TransformParameter("LastName", SqlDbType.VarChar, 50);
    TransformParameter("FirstName", SqlDbType.VarChar, 30);
    TransformParameter("Country", SqlDbType.VarChar, 3);
}
```

Codesegment 56 Überschreiben der AdaptSqlParameters-Methode von daPersList

1.8.5 Entity-Klasse Pers hinzufügen

Siehe Address Fall.

1. Fügen Sie im Ordner Entity eine neue Klasse hinzu und benennen Sie diese *eoPers.cs*.

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

```
using System.Text;
using System.Data;
using Deag.Vdx.Common.Exceptions.LocalExceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 57 Using-Anweisung von eoPers

3. Fügen Sie der Klasse folgende Properties hinzu.

```
public int PersId
{
    get { return (int)GetFromCacheDefault("PersID", -1); }
    set { SetToCache("PersID", value); }
}
public string FirstName
{
   get { return (string)GetFromCacheDefault("FirstName", string.Empty); }
   set { SetToCache("FirstName", value); }
}
public string LastName
{
   get { return (string)GetFromCacheDefault ("LastName", string.Empty); }
   set { SetToCache("LastName", value); }
}
protected new dsPers DsCache
{
    get { return (dsPers)base.DsCache; }
    set { base.DsCache = value; }
}
```

Codesegment 58 Implementation der Properties von eoPers

4. Fügen Sie der Klasse die Ableitung von EntityObject hinzu und kodieren Sie den Konstruktor wie folgt.

```
namespace <DalProjectNamespace>.Entity
{
    public class eoPers : EntityObject
    {
        public eoPers()
        {
            // init entity object
            EnforceCacheConstraints = false;
            DataSetType = typeof(dsPers);

            // init data access object
            daPers da = new daPers();
            da.Table = DsCache.Pers;
            AddAccessObject(da);
        }
    }
}
```

Codesegment 59 Implementation des eoPers-Konstruktors

5. Überschreiben Sie die *RefreshDataAccessObject*-Methode.

```
protected override void RefreshDataAccessObject(…)
{
    accessObject.ClearParameters();
    accessObject.AddParameters(GetParametersFromCache());
}
```

Codesegment 60 Überschreiben der RefreshDataAccessObject-Methode von eoPers

6. Überschreiben Sie die *Validate*-Methode. Die *ValidateExceptions* mit den Ids grösser 5000 sind Applikationsspezifische Fehlermeldungen, die mit dem *ExceptionEditor* erfasst werden müssen.

```
public override void Validate(…)
{
    if (!IsInCache("AddressID"))
    {
        RecordException(transaction,
               new vdxValidateException(5002), ref cancel);
    }
    if (!IsInCache("FirstName") || FirstName.Length == 0)
    {
        RecordException(transaction,
               new vdxValidateException(5004), ref cancel);
    }
    if (!IsInCache("LastName") || LastName.Length == 0)
    {
        RecordException(transaction,
               new vdxValidateException(5005), ref cancel);
    }
    if (!IsInCache("Sex"))
    {
        RecordException(transaction,
               new vdxValidateException(5006), ref cancel);

    }
    if (!IsInCache("Lang"))
    {
        RecordException(transaction,
               new vdxValidateException(5007), ref cancel);
    }
}
```

Codesegment 61 Überschreiben der Validate-Methode von eoPers

1.8.6 Web-Service hinzufügen und Web-Methoden auskodieren

Siehe Address Fall.

1. Fügen Sie einen neuen Web Service hinzu und benennen Sie diesen *wsPers.asmx*.

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

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.Entity;
```

Codesegment 62 Using-Anweisung von wsPers

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsPers.asmx")]
public class wsPers : System.Web.Services.WebService
{
    ...
}
```

Codesegment 63 Klassenattribut von wsPers

4. Fügen Sie die Methode *GetDataSetPers* mit Parameter hinzu.

```
[WebMethod]
public dsPers GetDataSetPers(object[] parameters)
{
   dsPers dsRet = new dsPers();

   try
   {
      eoPers eo = new eoPers();
      // vdxSearchParameters deserialization
      vdxSearchParameters sp = new vdxSearchParameters(parameters);
      int persId = Convert.ToInt32(sp["PersId"].Value);

      if (persId > 0)
      {
         eo.PersId = persId;
         eo.Read();
         dsRet = (dsPers)eo.Serialize();
      }
   }
   catch (vdxException ex)
   {
      throw ex;
   }
   catch (Exception ex)
   {
      throw new vdxException("Could not get dsPers.", ex);
   }

   return dsRet;
}
```

Codesegment 64 Implementation der GetDataSetPers-Methode mit Argument von wsPers

5. Fügen Sie die Methode *UpdateDataSet* hinzu.

```
[WebMethod]
public dsPers UpdateDataSet(dsPers dsToUpdate)
{
    dsPers dsRet;
    Transaction trans = new Transaction();

    try
    {
       eoPers eo = new eoPers();
       eo.Deserialize(dsToUpdate);
       eo.DataDrivenUpdate();
       dsRet = (dsPers)eo.Serialize();
       trans.Commit();
    }
    catch (vdxException ex)
    {
       trans.Rollback();
       throw ex;
    }
    catch (Exception ex)
    {
       trans.Rollback();
       throw new vdxException("Could not update dsPers.", ex);
    }

    return dsRet;
}
```

Codesegment 65 Implementation der UpdateDataSet-Methode von wsPers

6. Fügen Sie die Methode *GetDataSetPersList* hinzu.

```
[WebMethod]
public dsPersList GetDataSetPersList(object[] parameters)
{
    // DataAccess and DataSet initialization
    daPersList da = new daPersList();
    dsPersList dsRet = new dsPersList();
    try
    {
       // vdxSearchParameters deserialization
       vdxSearchParameters sp = new vdxSearchParameters(parameters);

       switch(sp.VdxSearchName)
       {
           case "SelectionPersByDescr":
              da.AddParameter(new Parameter(
                  "Country", Filter.CreateFilter(sp, "Country")));
              da.AddParameter(new Parameter(
                  "FirstName", Filter.CreateFilter(sp, "FirstName")));
              da.AddParameter(new Parameter(
                  "LastName", Filter.CreateFilter(sp, "LastName")));
              da.OrderBy = Convert.ToString(sp["OrderBy"].Value);
              da.Table = dsRet.Pers;
              dsRet.EnforceConstraints = false;
              da.Read();
              break;
       }
    }
    catch (vdxException ex)
    {
       throw ex;
    }
    catch (Exception ex)
    {
       throw new vdxException("Could not get dsPersList.", ex);
    }

    return dsRet;
}
```

Codesegment 66 Implementation der GetDataSetPersList-Methode von wsPers

1.9 Zugriff auf Order-Tabellen erstellen

1.9.1 DataSet dsItem hinzufügen

DataSet Item wird verwendet um Artikel von der Datenbank-Tabelle Item dem *DataPickFieldItem* und *PickerDialogItem* zur Verfügung zu stellen um eine Auswahl aller Artikeln anzubieten.

Siehe Pers Fall.

1. Fügen Sie im Ordner DataSet ein neues DataSet hinzu und benennen Sie es *dsItem.xsd*.

2. Setzen Sie für die Spalte *ItemID* das Property *AutoIncrementStep* auf -1.

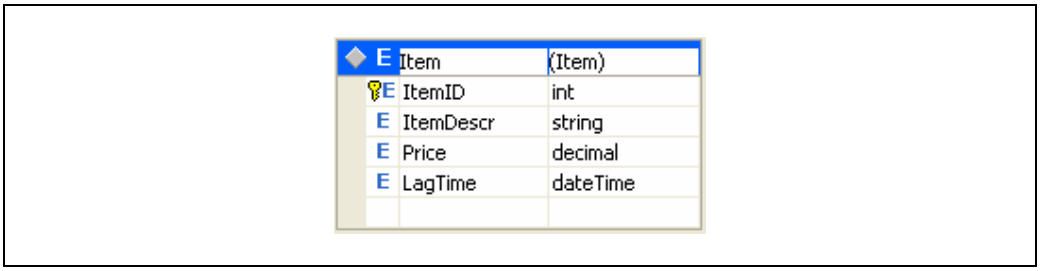


Abbildung 17 DataSet dsItem

1.9.2 DataSet dsOrder hinzufügen

DataSet Order wird verwendet als *DataSetType* im EntityObject Order (Siehe eoOrder erstellen). Das DataSet wird nicht zum Client kommuniziert!

1. Fügen Sie im Ordner DataSet ein neues DataSet hinzu und benennen Sie es *dsOrder.xsd*.

2. Setzen Sie für die Spalte *OrderID* die Properties *AutoIncrementStep* auf -1 und *ReadOnly* auf false.

3. Fügen Sie dem DataSet ein neues string Feld *ADDRESSDESCR* hinzu.

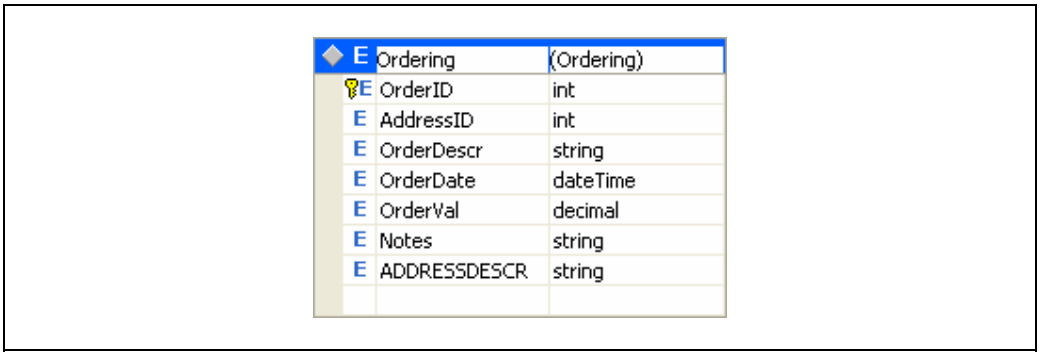


Abbildung 18 DataSet dsOrder

1.9.3 DataSet dsOrderPos hinzufügen

DataSet OrderPos wird verwendet als *DataSetType* im EntityObject OrderPos (Siehe eoOrderPos und elOrderPos erstellen). Das DataSet wird nicht zum Client kommuniziert!

1. Fügen Sie dem Ordner DataSet ein neues DataSet hinzu und benennen Sie es *dsOrderPos.xsd*.

2. Setzen Sie für die Spalte *OrderPosID* die Properties *AutoIncrementStep* auf -1 und *ReadOnly* auf false.

3. Fügen Sie dem DataSet ein neues string Feld *ItemDescr* hinzu.

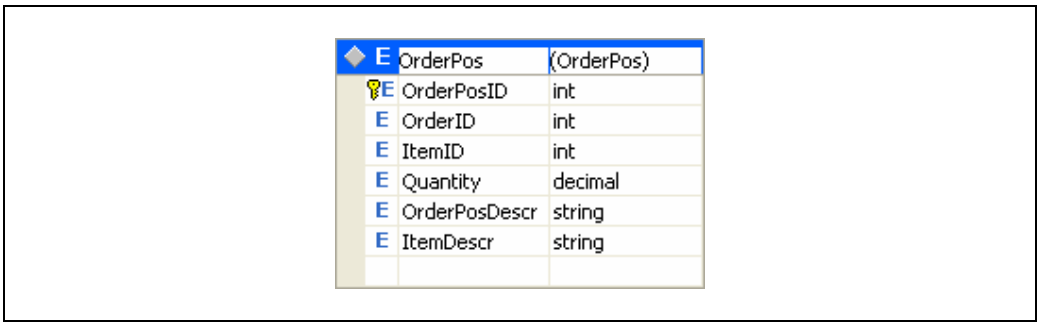


Abbildung 19 DataSet dsOrderPos

1.9.4 DataSet dsOrderDetail hinzufügen

DataSet OrderDetail wird verwendet um eine Bestellung von der Datenbank-Tabelle *Order* und deren Artikel von der Tabelle *Item* dem DataDetailOrder zur Verfügung zu stellen und um Bestellungen/Artikel vom Client in die Datenbank-Tabelle zurück zu speichern, zu aktualisieren oder zu löschen.

1. Fügen Sie ein neues DataSet hinzu und benennen Sie es *dsOrderDetail.xsd*.

2. Ziehen Sie die Tabellen *Ordering* und *OrderPos* auf das DataSet.

3. Fügen Sie in der Tabelle Ordering das Feld *ADDRESSDESCR* hinzu.

4. Fügen Sie in der Tabelle *OrderPos* das Feld *ItemDescr* hinzu.

5. Selektieren Sie das Feld *OrderID* der Tabelle *OrderPos* und erstellen Sie eine Relation zur Tabelle Ordering.

6. Setzen Sie das Property *AutoIncrementStep* auf -1 für die Spalte *OrderPosID* der Tabelle *OrderPos* und für die Spalte *OrderID* der Tabelle Ordering.

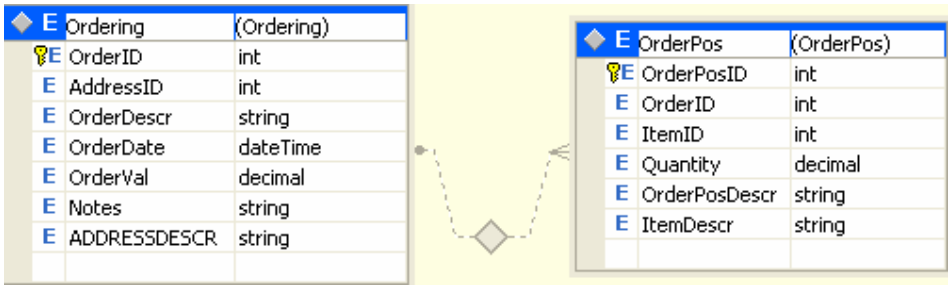


Abbildung 20 DataSet dsOrderDetail

1.9.5 DataSet dsOrderList hinzufügen

DataSet OrderList wird verwendet um Bestellungen von der Datenbank-Tabelle Order dem ExplorerOrder zur Verfügung zu stellen.

1. Fügen Sie ein neues DataSet hinzu und benennen Sie es *dsOrderList.xsd*.
2. Ziehen Sie die Tabellen Ordering auf das DataSet.
3. Löschen Sie aus dem DataSet die Felder *AddressID* und *Notes*.
4. Setzen Sie für die Spalte *OrderID* das Property *AutoIncrementStep* auf -1.

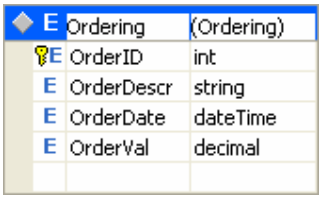


Abbildung 21 DataSet dsOrderList

1.9.6 DataAccess-Klasse daItem hinzufügen

Siehe Country Fall.

1. Namespace

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 67 Using-Anweisung von daItem

2. Ableiten von *SQLDataAccessDataset* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daItem : SQLDataAccessDataset
   {
      public daItem()
      {
         // Create select statement
         SelectCommandDynamicWhere select =
            new SelectCommandDynamicWhere();
         select.Statement = "SELECT * FROM Item";
         select.ParameterMapping.Add(
            new ParameterMapping("ItemID", "ItemID"));
         select.ParameterMapping.Add(
            new ParameterMapping("ItemDescr", "ItemDescr"));
         SqlSelect = select;
         // After an insert or update operation, the data will be read,
         // cause to guaranty the right AddressId in the entity object
         ReadAfterWrite = true;
         ReadAfterWriteWhere = "WHERE ItemID = @@IDENTITY";
         // Read connection string for the needed Database
         this.ConnectionString = DbConnection.GetConnectionString();
      }
   }
}
```

Codesegment 68 Implementation des daItem-Konstruktors

3. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("ItemID", SqlDbType.Int);
    TransformParameter("ItemDescr", SqlDbType.VarChar);
}
```

Codesegment 69 Überschreiben der AdaptSqlParameters-Method von daItem

1.9.7 DataAccess-Klasse daOrder hinzufügen

Siehe Pers Fall.

1. Namespace

```
using System.Data;
using System.Text;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 70 Using-Anweisung von daOrder

2. Ableiten von *SQLDataAccessQuery* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daOrder : SQLDataAccessQuery
   {
      public daOrder()
      {
         // Create select statement
         SelectCommand select = new SelectCommand();
         StringBuilder cmd = new StringBuilder();
         cmd.Append("SELECT\n");
         cmd.Append("OrderID,Order.AddressID,OrderDescr,OrderDate,OrderVal,Notes,\n");
         cmd.Append("ISNULL(Address.Company, '') + ' ' + ISNULL(Address.ZIP, '') + ");
         cmd.Append("' ' + ISNULL(Address.City, '') AS ADDRESSDESCR \n");
         cmd.Append("FROM Order INNER JOIN Address ");
         cmd.Append("ON Order.AddressID = Address.AddressID \n");
         cmd.Append("WHERE OrderID=@OrderID");
         select.Query = cmd.ToString();
         SqlSelect = select;
         // Create insert statement
         InsertCommand insert = new InsertCommand();
         cmd = new StringBuilder();
         cmd.Append("INSERT INTO Order (\n");
         cmd.Append("AddressID, OrderDescr, OrderDate, OrderVal, Notes) \n");
         cmd.Append("VALUES (\n");
         cmd.Append("@AddressID, @OrderDescr,@OrderDate,@OrderVal,@Notes); \n");
         // Return created ID
         cmd.Append("SELECT\n");                  cmd.Append("OrderID,Order.AddressID,
         OrderDescr, OrderDate, OrderVal, Notes,\n");
         cmd.Append("ISNULL(Address.Company, '') + ' ' + ISNULL(Address.ZIP, '') + ");
         cmd.Append("' ' + ISNULL(Address.City, '') AS ADDRESSDESCR \n");
         cmd.Append("FROM Order INNER JOIN Address ");
         cmd.Append("ON Order.AddressID = Address.AddressID\n");
         cmd.Append("WHERE OrderID=@@IDENTITY");
         insert.Query = cmd.ToString();
         SqlInsert = insert;
         // Create update statement
         UpdateCommand update = new UpdateCommand();
         cmd = new StringBuilder();
         cmd.Append("UPDATE Order SET \n");
         cmd.Append("AddressID = @AddressID, OrderDescr = @OrderDescr, \n");
         cmd.Append("OrderDate = @OrderDate, OrderVal = @OrderVal, Notes = @Notes \n");
         cmd.Append("WHERE (OrderID = @OrderID); \n");
         update.Query = cmd.ToString();
         SqlUpdate = update;
         // Create delete statement
         DeleteCommand delete = new DeleteCommand();
         delete.Query =
            "DELETE FROM Order \nWHERE (OrderID = @OrderID)";
         SqlDelete = delete;
         // Read connection string for the needed Database
         this.ConnectionString = DbConnection.GetConnectionString();
       }
   }
}
```

Codesegment 71 Implementation des daOrder-Konstruktors

3. *AdaptSqlParameters*-Methode überschreiben.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("OrderID", SqlDbType.Int);
    TransformParameter("OrderDescr", SqlDbType.VarChar);
```

Codesegment 72 Überschreiben der AdaptSqlParameters-Method von daOrder

1.9.8 DataAccess-Klasse daOrderPos hinzufügen

Siehe Pers Fall.

1. Namespace

```
using System.Data;
using System.Text;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 73 Using-Anweisung von daOrderPos

2. Ableiten von *SQLDataAccessQuery* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daOrderPos : SQLDataAccessQuery
   {
      public daOrderPos()
      {
         // Create select statement
         SelectCommand select = new SelectCommand();
         StringBuilder cmd = new StringBuilder();
         cmd.Append("SELECT\n");
         cmd.Append("OrderPosID,OrderID,OrderPos.ItemID, Quantity,OrderPosDescr,\n");
         cmd.Append("Item.ItemDescr, \n");
         cmd.Append("FROM OrderPos LEFT OUTER JOIN Item ");
         cmd.Append("ON OrderPos.ItemID = Item.ItemID\n");
         cmd.Append("WHERE (OrderPos.OrderID = @OrderID)");
         select.Query = cmd.ToString();
         SqlSelect = select;

         // Create insert statement
         InsertCommand insert = new InsertCommand();
         cmd = new StringBuilder();
         cmd.Append("INSERT INTO OrderPos (\n");
         cmd.Append("OrderID, ItemID, Quantity, OrderPosDescr) \n");
         cmd.Append("VALUES (\n");
         cmd.Append("@OrderID, @ItemID, @Quantity, @OrderPosDescr); \n");
         // Return created ID
         cmd.Append("SELECT\n");
         cmd.Append("OrderPosID,OrderID,OrderPos.ItemID, Quantity,OrderPosDescr,\n");
         cmd.Append("Item.ItemDescr \n");
         cmd.Append("FROM OrderPos LEFT OUTER JOIN Item ");
         cmd.Append("ON OrderPos.ItemID = Item.ItemID\n");
         cmd.Append("WHERE (OrderPosID = @@IDENTITY)");
         insert.Query = cmd.ToString();
         SqlInsert = insert;

         // Create update statement
         UpdateCommand update = new UpdateCommand();
         cmd = new StringBuilder();
         cmd.Append("UPDATE OrderPos SET \n");
         cmd.Append("OrderID = @OrderID, ItemID = @ItemID, \n");
         cmd.Append("OrderPosDescr = @OrderPosDescr \n");
         cmd.Append("WHERE (OrderPosID = @OrderPosID); \n");
         update.Query = cmd.ToString();
         SqlUpdate = update;

         // Create delete statement
         DeleteCommand delete = new DeleteCommand();
         delete.Query =
            "DELETE FROM OrderPos \nWHERE (OrderPosID = @OrderPosID)";
         SqlDelete = delete;

         // Read connection string for the needed Database
         this.ConnectionString = DbConnection.GetConnectionString();
       }
   }
}
```

Codesegment 74 Implementation des daOrderPos-Konstruktors

3. *AdaptSqlParameters*-Methode überschreiben.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("OrderPosID ", SqlDbType.Int);
}
```

Codesegment 75 Überschreiben der AdaptSqlParameters-Method von daOrderPos

1.9.9 DataAccess-Klasse daOrderList hinzufügen

Siehe Country Fall.

1. Namespace

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Commands;
using Deag.Vdx.Server.DAL.SQLServer;
```

Codesegment 76 Using-Anweisung von daOrderList

2. Ableiten von *SQLDataAccessDataset* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.DataAccess
{
   public class daOrderList : SQLDataAccessDataset
   {
      public daOrderList()
      {
         // Create select statement
         SelectCommandDynamicWhere select =
            new SelectCommandDynamicWhere();
         select.Statement =
            "SELECT OrderID,OrderDescr,OrderDate,OrderVal FROM Order ";
         select.ParameterMapping.Add(
            new ParameterMapping("OrderDescr", "OrderDescr"));
         select.ParameterMapping.Add(
            new ParameterMapping("OrderID", "OrderID"));

         SqlSelect = select;

         // Read connection string for the needed Database
         this.ConnectionString = DbConnection.GetConnectionString();
       }
   }
}
```

Codesegment 77 Implementation des daOrderList-Konstruktors

3. Fügen Sie der Klasse folgende Properties für die entsprechenden Felder hinzu.

```
public string OrderBy
{
     set
     {
          ((SelectCommandDynamicWhere)SqlSelect).OrderBy =
               "ORDER BY " + value;
     }
}
```

Codesegment 78 Implementation des OrderBy-Property von daOrderList

4. Überschreiben Sie die *AdaptSqlParameters*-Methode.

```
protected override void AdaptSqlParameters(…)
{
    TransformParameter("OrderID", SqlDbType.Int);
    TransformParameter("OrderDescr", SqlDbType.VarChar);
}
```

Codesegment 79 Überschreiben der AdaptSqlParameters-Method von daOrderList

1.9.10 Entity-Klasse eoOrderPos hinzufügen

Siehe Address Fall.

1. Namespace

```
using System.Data;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions.LocalExceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 80 Using-Anweisung von eoOrderPos

2. Properties

```
public int OrderPosId
{
   get { return (int)GetFromCacheDefault("OrderPosID", -1);}
   set { SetToCache("OrderPosID", value); }
}

public int OrderId
{
    get { return (int)GetFromCacheDefault("OrderID", -1); }
    set { SetToCache("OrderID", value); }
}

protected new dsOrderPos DsCache
{
    get { return (dsOrderPos)base.DsCache; }
    set { base.DsCache = value; }
}
```

Codesegment 81 Properties von eoOrderPos

3. Ableiten von *EntityObject* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.Entity
{
   public class eoOrderPos : EntityObject
   {
      public eoOrderPos()
      {
         // init entity object
         AutoAddRow = false;
         DataSetType = typeof(dsOrderPos);

         // init data access object
         daOrderPos da = new daOrderPos();
         da.Table = DsCache.OrderPos;
         AddAccessObject(da);
      }
   }
}
```

Codesegment 82 Implementation des eoOrderPos-Konstruktors

4. *RefreshDataAccessObject*-Methode überschreiben

```
protected override void RefreshDataAccessObject(…)
{
    accessObject.ClearParameters();
    accessObject.AddParameters(GetParametersFromCache());
}
```

Codesegment 83 Überschreiben der RefreshDataAccessObject-Method von eoOrderPos

5. Validate-Methode überschreiben.

```
public override void Validate(…)
{
    if (!IsInCache("Quantity"))
    {
        RecordException(transaction,
               new vdxValidateException(5011), ref cancel);
    }
    if (!IsInCache("OrderPosDescr"))
    {
        RecordException(transaction,
               new vdxValidateException(5012), ref cancel);
    }
}
```

Codesegment 84 Überschreiben der Validate-Method von eoOrderPos

1.9.11 EntityList-Klasse elOrderPos hinzufügen

1. Namespace

```
using System.Data;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 85 Using-Anweisung von elOrderPos

2. Ableiten von *EntityList* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.Entity
{
   public class elOrderPos : EntityList
   {
      private int orderID;

      public elOrderPos()
      {
          // init entity List
          DataSetType = typeof(dsOrderPos);
          this.ItemType = typeof(eoOrderPos);
          this.UseCollection = true;
      }
   }
}
```

Codesegment 86 Implementation des elOrderPos-Konstruktors

3. Properties

```
public int OrderId
{
    get { return this.orderID; }
    set
    {
        this.orderID = value;

        SetValue("OrderID", "OrderId", value);
    }
}
```

Codesegment 87 Property OrderId von elOrderPos

4. *RefreshDataAccessObject*-Methode überschreiben

```
protected override void RefreshDataAccessObject(…)
{
    accessObject.ClearParameters();
    accessObject.AddParameters(GetParametersFromCache());
}
```

Codesegment 88 Überschreiben der RefreshDataAccessObject-Methode von elOrderPos

1.9.12 Entity-Klasse eoOrder hinzufügen

Siehe Address Fall.

1. Namespace

```
using System.Data;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions.LocalExceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.Entity;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
```

Codesegment 89 Using-Anweisung von eoOrder

2. Properties

```
public int OrderId
{
    get { return (int)GetFromCacheDefault("OrderID", -1);}
    set { SetToCache("OrderID", value); }
}

// Gets or sets an entityList with all order positions.
public elOrderPos OrderPos
{
    get
    {
        if (orderPos == null)
        {
            orderPos = new elOrderPos();
            orderPos.Parent = this;
            orderPos.OrderId = OrderId;
        }
        return orderPos;
    }
    set
    {
        orderPos = value;
        orderPos.Parent = this;
        orderPos.OrderId = OrderId;
    }
}

protected new dsOrder DsCache
{
     get { return (dsOrder)base.DsCache; }
     set { base.DsCache = value; }
}
```

Codesegment 90 Properties von eoOrder

3. Ableiten von *EntityObject* und Konstruktor wie folgt kodieren.

```
namespace <DalProjectNamespace>.Entity
{
    public class eoOrder : EntityObject
    {
        private elOrderPos orderPos;

        public eoOrder()
        {
            // init entity object
            AutoAddRow = false;
            DataSetType = typeof(dsOrder);

            // init data access object
            daOrder da = new daOrder();
            da.Table = DsCache.Ordering;
            AddAccessObject(da);
        }
    }
}
```

Codesegment 91 Implementation des eoOrder-Konstruktors

4. *RefreshDataAccessObject*-Methode überschreiben

```
protected override void RefreshDataAccessObject(…)
{
    accessObject.ClearParameters();
    accessObject.AddParameters(GetParametersFromCache());
}
```

Codesegment 92 Überschreiben der RefreshDataAccessObject-Methode von eoOrder

5. *Validate*-Methode überschreiben

```
public override void Validate(…)
{
   if (!IsInCache("AddressID"))
   {
      RecordException(transaction,
            new vdxValidateException(5002), ref cancel);
   }
   if (!IsInCache("OrderDate"))
   {
      RecordException(transaction,
            new vdxValidateException(5010), ref cancel);
   }
}
```

Codesegment 93 Überschreiben der Validate-Methode von eoOrder

6. *Insert*-Methode überschreiben

```
public override void Insert(Transaction transaction)
{
   base.Insert(transaction);
   orderPos.OrderId = (int)GetFromCache("OrderID", true);
   orderPos.DataDrivenUpdate(transaction);
}
```

Codesegment 94 Überschreiben der Insert-Methode von eoOrder

7. *Update*-Methode überschreiben

```
public override void Update(Transaction transaction)
{
   base.Update(transaction);
   orderPos.DataDrivenUpdate(transaction);
}
```

Codesegment 95 Überschreiben der Update-Methode von eoOrder

8. *Delete*-Methode überschreiben

```
public override void Delete(Transaction transaction)
{
    orderPos.Delete(transaction);
    base.Delete(transaction);
}
```

Codesegment 96 Überschreiben der Delete-Methode von eoOrder

9. *DataDrivenUpdate*-Methode überschreiben

```
public override void DataDrivenUpdate(Transaction transaction)
{
    if (this.Status == EntityStatus.Unspecified ||
        this.Status == EntityStatus.Unchanged)
    {
       orderPos.DataDrivenUpdate(transaction);
    }
    else
    {
       base.DataDrivenUpdate(transaction);
    }
}
```

Codesegment 97 Überschreiben der DataDrivenUpdate-Methode von eoOrder

1.9.13 Web-Service Item hinzufügen und Web-Methoden auskodieren

Siehe Country Fall.

1. Fügen Sie einen neuen Web Service hinzu und benennen Sie diesen *wsItem.asmx*.

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

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.DataSets;
```

Codesegment 98 Using-Anweisung von wsItem

3. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsItem.asmx")]
public class wsItem : System.Web.Services.WebService
{
    ...
}
```

Codesegment 99 Klassenattribut von wsItem

4. Fügen Sie die Methode *GetDataSetItem* mit Parameters hinzu.

```
[WebMethod]
public dsItem GetDataSetItem(object[] parameters)
{
    // DataAccess and DataSet initialization
    daItem da = new daItem();
    dsItem dsRet = new dsItem();

    try
    {
        // vdxSearchParameters deserialization
        vdxSearchParameters sp = new vdxSearchParameters(parameters);

        switch(sp.VdxSearchName)
        {
            case "SelectionItemByDescr":
                string descr = Convert.ToString(sp["ItemDescr"].Value);
                if (descr != null && descr.Length > 0)
                {
                    da.AddParameter
                    (
                        new Parameter
                        (
                            "ItemDescr",
                            Filter.CreateFilter(sp, "ItemDescr")
                        )
                    );
                }
                break;

            case "SelectionItemById":
                int itemId = Convert.ToInt32(sp["ItemID"].Value);
                if (itemId > 0)
                {
                    da.AddParameter(new Parameter("ItemID", itemId));
                }
                break;
        }
        da.Table = dsRet._Item;
        da.Read();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not get dsItem.", ex);
    }

    return dsRet;
}
```

Codesegment 100 Implementation der GetDataSetItem-Methode von wsItem

5. Fügen Sie die Methode *UpdateDataSet* hinzu.

```
[WebMethod]
public dsItem UpdateDataSet(dsItem dsToUpdate)
{
    try
    {
        daItem da = new daItem();
        da.Table = dsToUpdate._Item;
        da.DataDrivenUpdate();
    }
    catch (vdxException ex)
    {
        throw ex;
    }
    catch (Exception ex)
    {
        throw new vdxException("Could not update dsItem.", ex);
    }
    return dsToUpdate;
}
```

Codesegment 101 Implementation der UpdateDataSet-Methode von wsItem

1.9.14 Web-Service wsOrder hinzufügen

Siehe Address Fall.

1. Namespace

```
using Deag.Vdx.Common;
using Deag.Vdx.Common.Enums;
using Deag.Vdx.Common.Exceptions;
using Deag.Vdx.Server.DAL;
using Deag.Vdx.Server.DAL.Filters;
using <DalProjectNamespace>.DataSets;
using <DalProjectNamespace>.DataAccess;
using <DalProjectNamespace>.Entity;
```

Codesegment 102 Using-Anweisung von wsOrder

2. Fügen Sie der Klasse folgendes Attribut hinzu.

```
[WebService(Namespace="wsOrder.asmx")]
public class wsOrder : System.Web.Services.WebService
{
    ...
}
```

Codesegment 103 Klassenattribut von wsOrder

3. Methode *GetDataSetOrderDetail* hinzufügen.

```
[WebMethod]
public dsOrderDetail GetDataSetOrderDetail(object[] parameters)
{
    // DataAccess and DataSet initialization
    daOrder daH = new daOrder();
    daOrderPos daPos = new daOrderPos();
    dsOrderDetail dsRet = new dsOrderDetail();

    // vdxSearchParameters deserialization
    vdxSearchParameters sp = new vdxSearchParameters(parameters);

    try
    {
        int orderId = (int)sp["OrderID"].Value;
        if (orderId > 0)
        {
            daH.AddParameter
            (
                new Parameter
                (
                    "OrderID",
                    Filter.CreateFilter(sp, "OrderID")
                )
            );
            daPos.AddParameter
            (
                new Parameter
                (
                    "OrderID",
                    Filter.CreateFilter(sp, "OrderID")
                )
            );
            daH.Table = dsRet.Ordering;
            daH.Read();
            daPos.Table = dsRet.OrderPos;
            daPos.Read();
        }
    }
    catch(vdxException ex)
    {
        throw ex;
    }
    catch(Exception ex)
    {
        throw new vdxException("Could not get dsOrderDetail.", ex);
    }

    return dsRet;
}
```

Codesegment 104 Implementation der GetDataSetOrderDetail-Methode von wsOrder

4. Methode *UpdateDataSet* hinzufügen.

```
[WebMethod]
public dsOrderDetail UpdateDataSet(dsOrderDetail dsToUpdate)
{
    dsOrderDetail dsRet = new dsOrderDetail();
    Transaction trans = new Transaction();

    try
    {
       eoOrder eo = new eoOrder();
       eo.Deserialize(dsToUpdate.Ordering);
       eo.OrderPos.Deserialize(dsToUpdate.OrderPos);
       eo.DataDrivenUpdate(trans);
       dsRet.Merge((dsOrder)eo.Serialize());
       dsRet.Merge((dsOrderPos)eo.OrderPos.Serialize());
       trans.Commit();
    }
    catch (vdxException ex)
    {
       trans.Rollback();
       throw ex;
    }
    catch (Exception ex)
    {
       trans.Rollback();
       throw new vdxException("Could not update dsOrderDetail.", ex);
    }

    return dsRet;
}
```

Codesegment 105 Implementation der UpdateDataSet-Methode von wsOrder

5. Methode *GetDataSetOrderList* hinzufügen.

```
[WebMethod]
public dsOrderList GetDataSetOrderList(object[] parameters)
{
    // DataAccess and DataSet initialization
    daOrderList da = new daOrderList();
    dsOrderList dsRet = new dsOrderList();

    // vdxSearchParameters deserialization
    vdxSearchParameters sp = new vdxSearchParameters(parameters);

    try
    {
        switch(sp.VdxSearchName)
        {
            case "SelectionOrderByDescr":
                string descr = Convert.ToString(sp["Descr"].Value);
                string orderBy = Convert.ToString(sp["OrderBy"].Value);
                if (descr != null && descr.Length > 0)
                {
                    da.AddParameter
                    (
                        new Parameter
                        (
                            "OrderDescr",
                            Filter.CreateFilter(sp, "OrderDescr")
                        )
                    );
                }
                da.Table = dsRet.Ordering;
                da.Read();
                break;
        }
    }
    catch(vdxException ex)
    {
        throw ex;
    }
    catch(Exception ex)
    {
        throw new vdxException("Could not get dsOrderList.", ex);
    }
    return dsRet;
}
```

Codesegment 106 Implementation der GetDataSetOrderList-Methode von wsOrder

1.10 Bildverzeichnis

1.11 Tabellenverzeichnis

1.12 Codesegment-Verzeichnis