# Implement OERA & achieve true productivity with GUI for .NET™ using SmartComponent Library

# Tutorial

A Getting Started Guide

Contributors: (none at this point)

Authors: Eva Gessner, dietrainerin, Mike Fechner & Marko Rüterbories, Consultingwerk

# Contents

# About this Tutorial

## Prerequisites

Make sure that you meet the following **Software requirements**:

- Windows Vista, Windows 7
- OpenEdge 10.2B SP 04
- SmartComponent Library Version 17433.
- Infragistic Ultra Controls Version 9.2.20092.

Before you start you should be familiar with:

- OpenEdge Architect
- Visual Designer
- Object Oriented Features of the Progress ABL
- GUI for .NET Programming

## Signs

ℹ This sign indicates special information.

⚠ This sign indicates very important information you should know.

👨 Best practice recommended by Consultingwerk.

👍 Congratulation! You've completed a task.

# Introduction

## About the SmartComponent Library

### Why it is so smart

*The SmartComponent Library is the framework for the OpenEdge GUI for .NET, the OpenEdge UltraControls (Infragistics NetAdvantage for .NET) and the OpenEdge Reference Architecture that increases the developers productivity while building data-centric applications. The SmartComponent library contains various prebuilt components and Controls that can be used to build the graphical client of an application. The SmartComponent Library is also equipped with an OpenEdge Reference Architecture (OERA) compliant backend that uses ProDatasets to read and update data in relation to a database (OpenEdge database or Progress data-server technology).*

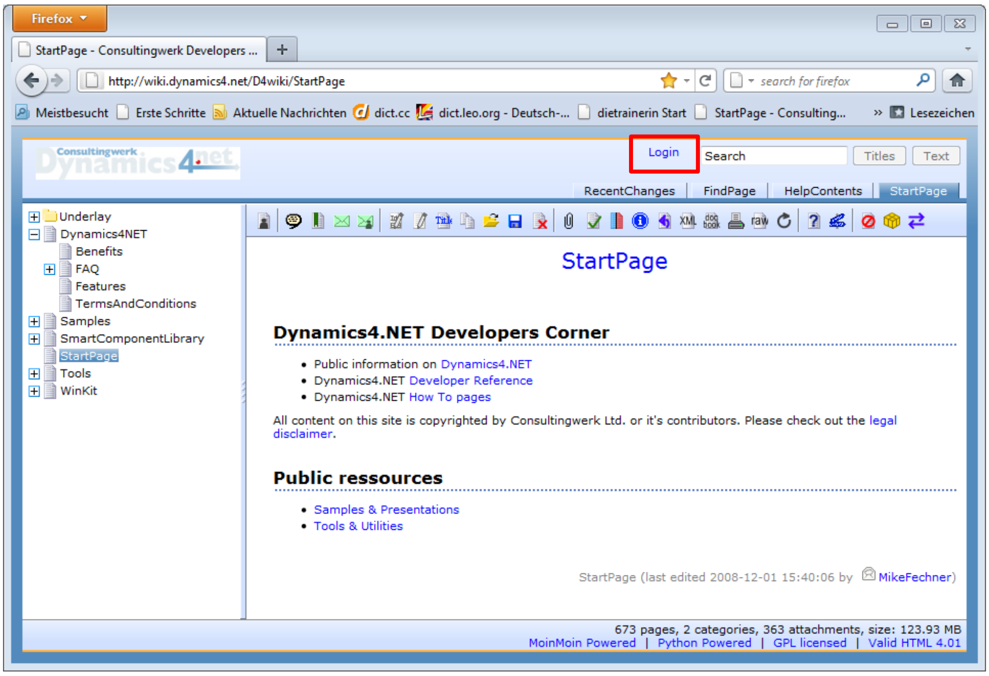
*This tutorial will introduce a developer the basic design principles of applications using the SmartComponent Library. The tutorial uses our sample / demo application – the CustomerExplorer – to explain the steps required to build basic screens. For additional information developers should consult the SmartComponent Library documentation on* http://wiki.dynamics4.net

# Setting up the SmartComponent Library environment

## Download the SmartComponent Library

Login to the *Consultingwerk Developers Corner* on *http://wiki.dynamics4.net*

As a licensed customer of the SmartComponent Library, you can request access by sending E-mail to support@consultingwerk.de. Your assigned username will be in the form of *FirstnameLastname* and you need to assign your own password on the wiki.



StartPage

Dynamics4.NET Developers Corner

- Public information on Dynamics4.NET
- Dynamics4.NET Developer Reference
- Dynamics4.NET How To pages

All content on this site is copyrighted by Consultingwerk Ltd. or it's contributors. Please check out the legal disclaimer.

Public ressources

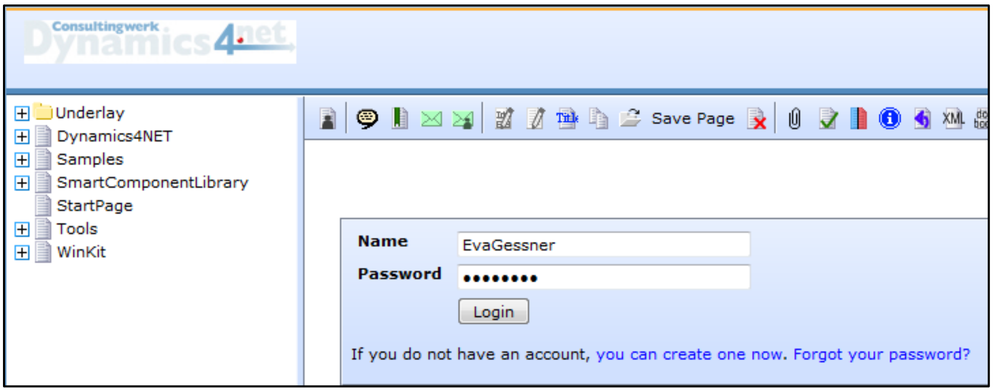
- Samples & Presentations
- Tools & Utilities

StartPage (last edited 2008-12-01 15:40:06 by MikeFechner)

673 pages, 2 categories, 363 attachments, size: 123.93 MB

MoinMoin Powered | Python Powered | GPL licensed | Valid HTML 4.01

Enter your **Name** and **Password** and click **Login**.



Name EvaGessner

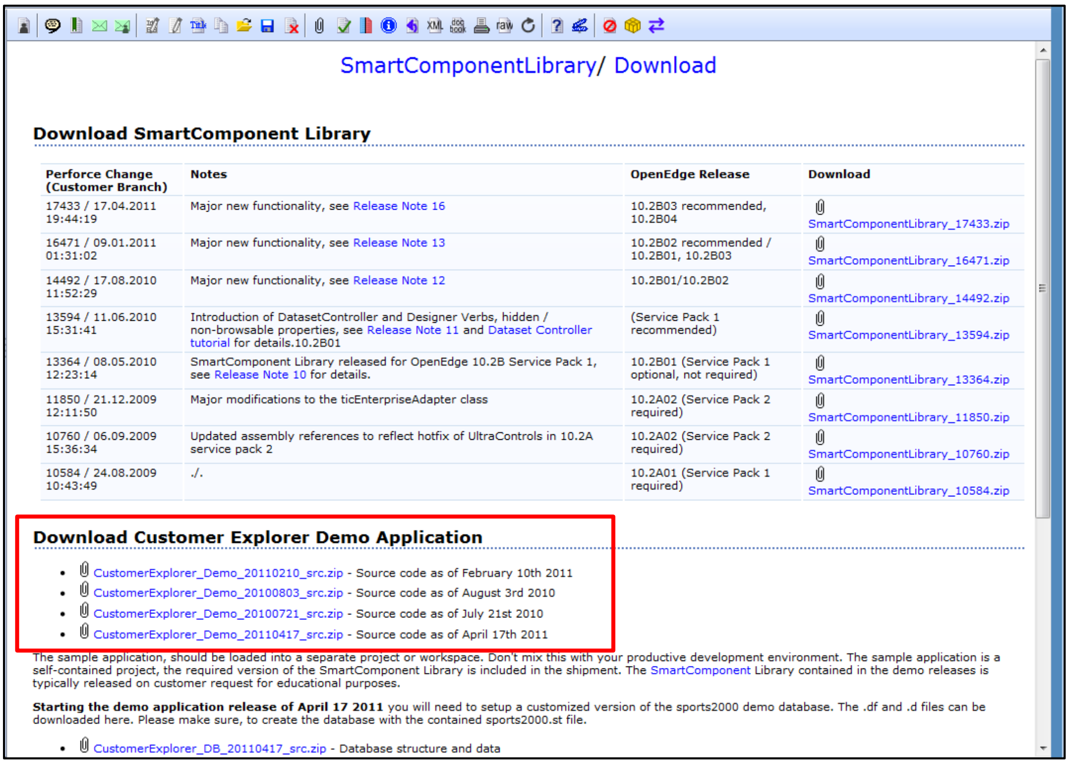
Password ••••••••

Login

If you do not have an account, you can create one now. Forgot your password?

Go to the **Download** section in the wiki at http://wiki.dynamics4.net/D4wiki/SmartComponentLibrary/Download and download the latest

*CustomerExplorer_Demo_yyyymmdd.zip* file. The file name ends with the release date of the file in the format yyyymmdd. Download the latest version.



SmartComponentLibrary/ Download

## Download SmartComponent Library

| Perforce Change (Customer Branch) | Notes | OpenEdge Release | Download |
|---|---|---|---|
| 17433 / 17.04.2011 19:44:19 | Major new functionality, see Release Note 16 | 10.2B03 recommended, 10.2B04 | SmartComponentLibrary_17433.zip |
| 16471 / 09.01.2011 01:31:02 | Major new functionality, see Release Note 13 | 10.2B02 recommended / 10.2B01, 10.2B03 | SmartComponentLibrary_16471.zip |
| 14492 / 17.08.2010 11:52:29 | Major new functionality, see Release Note 12 | 10.2B01/10.2B02 | SmartComponentLibrary_14492.zip |
| 13594 / 11.06.2010 15:31:41 | Introduction of DatasetController and Designer Verbs, hidden / non-browsable properties, see Release Note 11 and Dataset Controller tutorial for details.10.2B01 | (Service Pack 1 recommended) | SmartComponentLibrary_13594.zip |
| 13364 / 08.05.2010 12:23:14 | SmartComponent Library released for OpenEdge 10.2B Service Pack 1, see Release Note 10 for details. | 10.2B01 (Service Pack 1 optional, not required) | SmartComponentLibrary_13364.zip |
| 11850 / 21.12.2009 12:11:50 | Major modifications to the ticEnterpriseAdapter class | 10.2A02 (Service Pack 2 required) | SmartComponentLibrary_11850.zip |
| 10760 / 06.09.2009 15:36:34 | Updated assembly references to reflect hotfix of UltraControls in 10.2A service pack 2 | 10.2A02 (Service Pack 2 required) | SmartComponentLibrary_10760.zip |
| 10584 / 24.08.2009 10:43:49 | ./. | 10.2A01 (Service Pack 1 required) | SmartComponentLibrary_10584.zip |

## Download Customer Explorer Demo Application

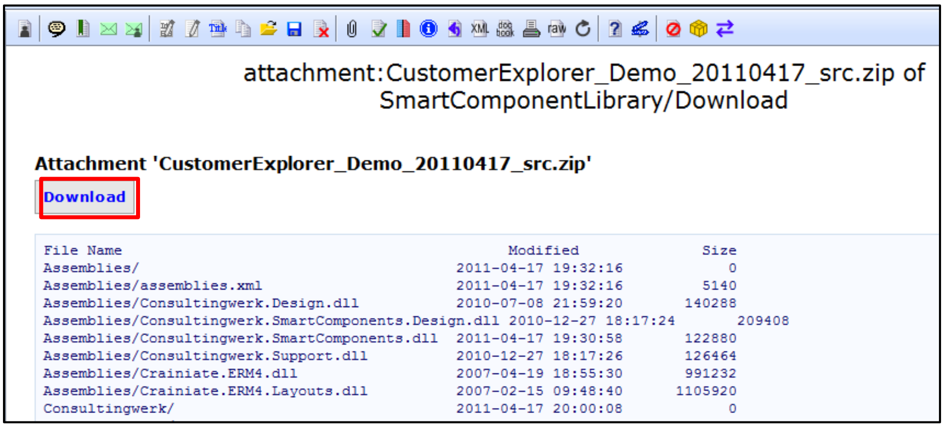
- CustomerExplorer_Demo_20110210_src.zip - Source code as of February 10th 2011
- CustomerExplorer_Demo_20100803_src.zip - Source code as of August 3rd 2010
- CustomerExplorer_Demo_20100721_src.zip - Source code as of July 21st 2010
- CustomerExplorer_Demo_20110417_src.zip - Source code as of April 17th 2011

The sample application, should be loaded into a separate project or workspace. Don't mix this with your productive development environment. The sample application is a self-contained project, the required version of the SmartComponent Library is included in the shipment. The SmartComponent Library contained in the demo releases is typically released on customer request for educational purposes.

**Starting the demo application release of April 17 2011** you will need to setup a customized version of the sports2000 demo database. The .df and .d files can be downloaded here. Please make sure, to create the database with the contained sports2000.st file.

- CustomerExplorer_DB_20110417_src.zip - Database structure and data

When you click on the selected file in the download pane a new page opens which shows all files contained in the zip-file. Click Button **Download** to start the download.



attachment:CustomerExplorer_Demo_20110417_src.zip of SmartComponentLibrary/Download

Attachment 'CustomerExplorer_Demo_20110417_src.zip'

```
File Name                                          Modified             Size
Assemblies/                                 2011-04-17 19:32:16            0
Assemblies/assemblies.xml                   2011-04-17 19:32:16         5140
Assemblies/Consultingwerk.Design.dll        2010-07-08 21:59:20       140288
Assemblies/Consultingwerk.SmartComponents.Design.dll 2010-12-27 18:17:24       209408
Assemblies/Consultingwerk.SmartComponents.dll  2011-04-17 19:30:58       122880
Assemblies/Consultingwerk.Support.dll       2010-12-27 18:17:26       126464
Assemblies/Crainiate.ERM4.dll               2007-04-19 18:55:30       991232
Assemblies/Crainiate.ERM4.Layouts.dll       2007-02-15 09:48:40      1105920
Consultingwerk/                             2011-04-17 20:00:08            0
```

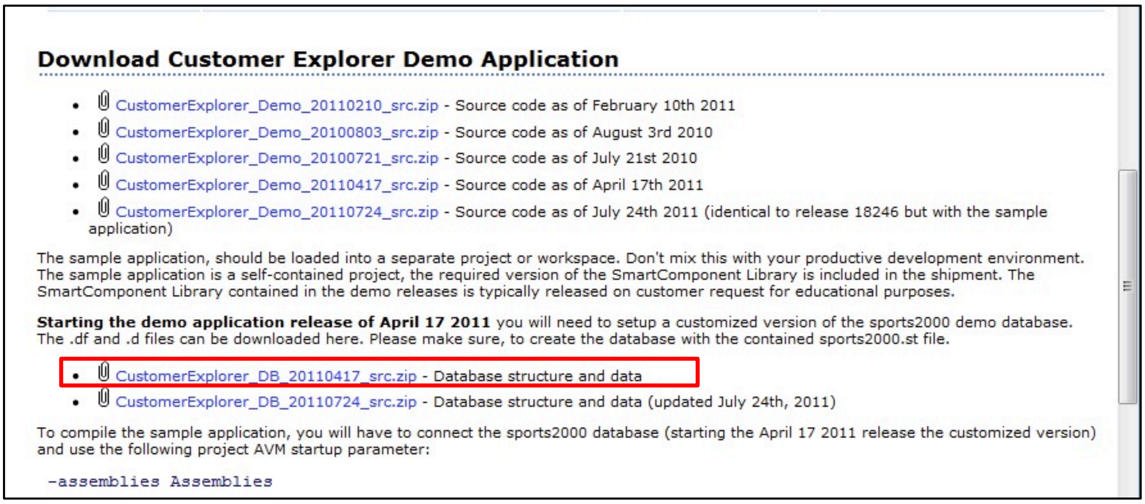
Save the file to a directory of your choice.

The *CustomerExplorer_Demo* file contains a full version of the *SmartComponent Library* plus a demo application based on a customized version of Progress' sports2000 database. If you want to learn more about the demo application go to chapter Start the Demo Application in the appendix of this tutorial.

The *CustomerExplorer-Demo* file is just for educational purpose and unsuitable as a starting point for real-world projects. For real-world environments you should use the *SmartComponentLibrary_yyymmdd.zip* files instead. For more information on how to set up a real-world environment refer to chapter Setting up the SmartComponent Library for real-world projects.

## Download database files

Repeat the above steps to download the database files for the *CustomerExplorer* demo. The demo application uses a customized version of the sports2000 database. Don't try to use the standard sports2000 database from the progress install directory. This database lacks some additional tables and fields and thus results in compilation errors. Instead download the latest file *CustomerExplorer_DB_yyyymmdd_src.zip*.



### Download Customer Explorer Demo Application

- CustomerExplorer_Demo_20110210_src.zip - Source code as of February 10th 2011
- CustomerExplorer_Demo_20100803_src.zip - Source code as of August 3rd 2010
- CustomerExplorer_Demo_20100721_src.zip - Source code as of July 21st 2010
- CustomerExplorer_Demo_20110417_src.zip - Source code as of April 17th 2011
- CustomerExplorer_Demo_20110724_src.zip - Source code as of July 24th 2011 (identical to release 18246 but with the sample application)

The sample application, should be loaded into a separate project or workspace. Don't mix this with your productive development environment. The sample application is a self-contained project, the required version of the SmartComponent Library is included in the shipment. The SmartComponent Library contained in the demo releases is typically released on customer request for educational purposes.

**Starting the demo application release of April 17 2011** you will need to setup a customized version of the sports2000 demo database. The .df and .d files can be downloaded here. Please make sure, to create the database with the contained sports2000.st file.

- CustomerExplorer_DB_20110417_src.zip - Database structure and data
- CustomerExplorer_DB_20110724_src.zip - Database structure and data (updated July 24th, 2011)

To compile the sample application, you will have to connect the sports2000 database (starting the April 17 2011 release the customized version) and use the following project AVM startup parameter:

```
-assemblies Assemblies
```

In the following chapters you use the customized sports2000 database to build up your sample application.

## Create the sports2000 database

Create a new sports2000 database based on the schema definition (.df file), the dumpfiles (.d) and the structure description file (.st) you downloaded earlier in the tutorial. Don't forget the binary files (.blb). They contain the images for the *CustomerExplorer* demo application.

If you need any assistance by setting up the database you'll find a detailed instruction in the chapter Create the sports2000 database in the appendix of this tutorial.

## Create a Workspace and Project directory

The sample application should be loaded into a separate project or workspace. Don't mix it with your productive development environment. The sample application is a self-contained project. The required version of the *SmartComponent Library* is included in the shipment.

In the **Windows Explorer** create a workspace and project directory.

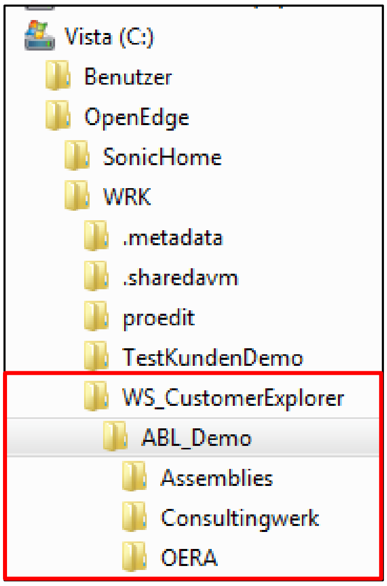
We have experienced that it is the safest to NOT use the space character as part of the workspace path. For instance we have run into issues with connecting to databases using relative path names when the workspace path does contain the space character.

Follow these naming conventions to make sure that the *CustomerExplorer* demo application will run.

- **Workspace**: WS_CustomerExplorer
- **Project**: ABL_Demo

These naming conventions are just for tutorial purposes. In real-world projects you can choose any names you want.

Unzip the file *CustomerExplorer_Demo_yyyymmdd_src.zip* to your workspace. The workspace structure for workspace *WS_CustomerExplorer* should look like this.

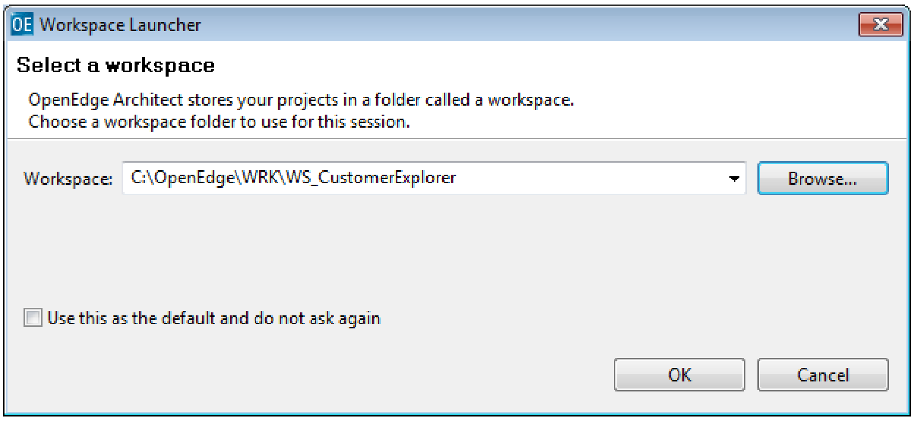


```
Vista (C:)
  Benutzer
  OpenEdge
    SonicHome
    WRK
      .metadata
      .sharedavm
      proedit
      TestKundenDemo
      WS_CustomerExplorer
        ABL_Demo
          Assemblies
          Consultingwerk
          OERA
```

## Import the project into the OpenEdge Architect

Open the **OpenEdge Architect** to import the project files.

In the **Workspace Launcher** select the appropriate workspace location.



Workspace Launcher

Select a workspace

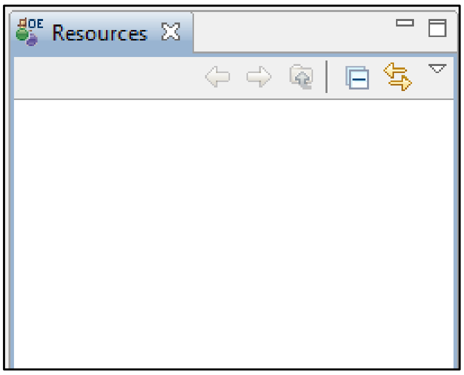
OpenEdge Architect stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\OpenEdge\WRK\WS_CustomerExplorer

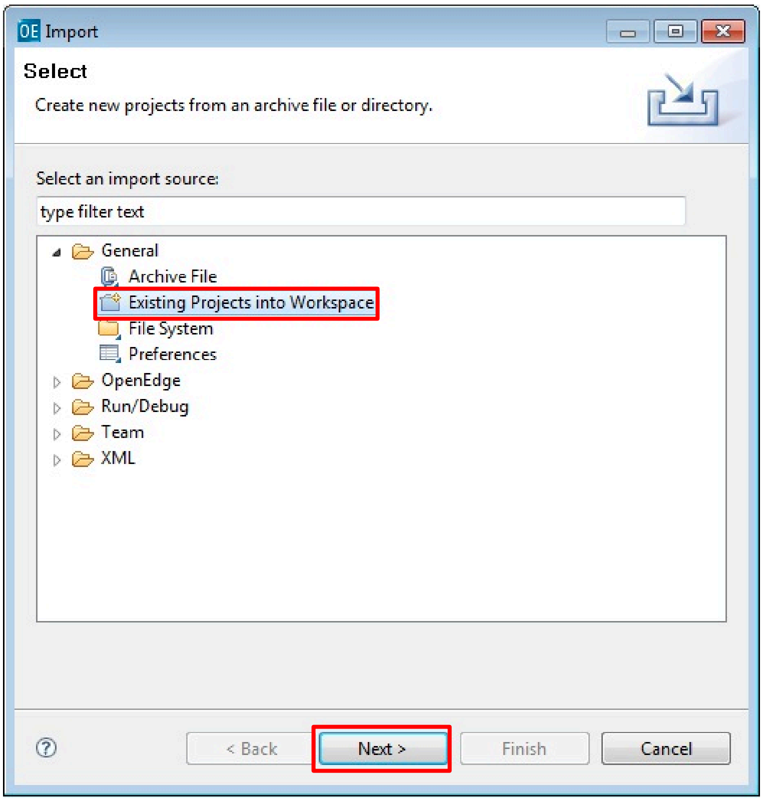
□ Use this as the default and do not ask again

The **OpenEdge Architect** opens.

The *Resources* View is empty so far.



- Select **File > Import** from the menu**.** The **Import** dialog opens.
- Select **General > Existing Projects into Workspace**.
- Click **Next**.



Import

Select

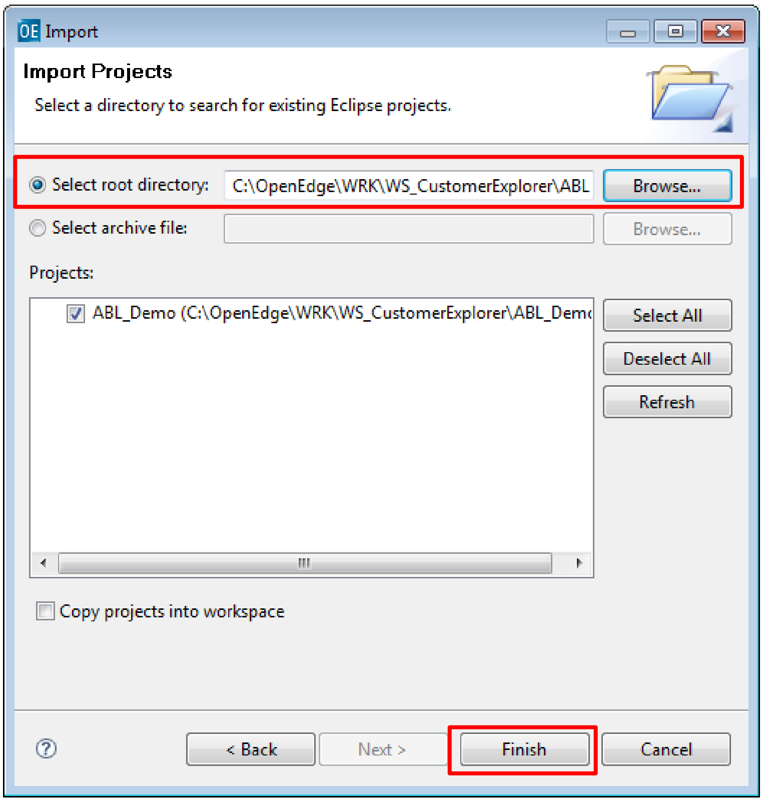
Create new projects from an archive file or directory.

Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- OpenEdge
- Run/Debug
- Team
- XML

- **Browse** to the directory that contains the project files and click **Finish**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\OpenEdge\WRK\WS_CustomerExplorer\ABL

○ Select archive file:

Projects:

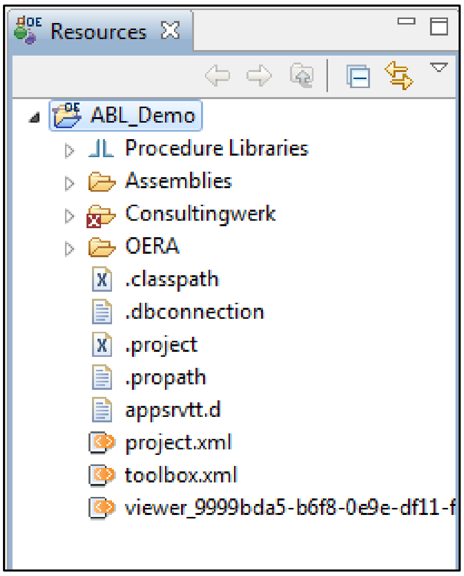
☑ ABL_Demo (C:\OpenEdge\WRK\WS_CustomerExplorer\ABL_Dem

□ Copy projects into workspace

## Project Structure

The *Resources* View should look like this.

If you don't see the folder Assemblies, Consultingwerk and OERA refresh the project (right click on the project folder > Refresh).



By default the deployment of the *SmartComponent Library* contains the folders *Assemblies*, *Consultingwerk* and *OERA*.

| Folder | Description |
|---|---|
| Assemblies | Contains the *assemblies.xml* file with references to all required Consultingwerk, Crainiate and Infragistics Assemblies (.dll files). |
| Consultingwerk | Contains the **SmartComponent Library** with all of its components you will discover during the tutorial. The central classes of the *SmartComponent Library* are stored in the folder **Consultingwerk/SmartComponents**.<br>For further information refer to the following release note:<br>http://wiki.dynamics4.net/D4wiki/SmartComponentLibrary/ReleaseNotes/ReleaseNote019 |
| OERA | Contains the procedural version of the Consultingwerk OERA implementation as well as the service interface procedures (required for AppServer calls). |

The *SmartComponent Library* is shipped with some .NET Assemblies (.dll files):

For further information refer to the following release notes:
http://wiki.dynamics4.net/D4wiki/SmartComponentLibrary/ReleaseNotes/ReleaseNote018

| Library .dll | Description |
|---|---|
| Consultingwerk.Design.dll | Design time Assemblies.<br>Provide functionality for the *SmartComponent Library*. |
| Consultingwerk.SmartComponents.Design.dll | |
| Consultingwerk.SmartComponents.dll | |
| Consultingwerk.Support.dll | |
| Crainiate.ERM4.dll | Provide functionality for the BusinessEntity Designer. |
| Crainiate.ERM4.Layouts.dll | |

The Consultingwerk Assemblies are required because the .NET integration into the .NET lacks some rarely used .NET framework features, especially required for the rich design time experience of the *SmartComponent Library*. The source code of these assemblies is made available to our clients.

With the expection of the *Consultingwerk.SmartComponents.Design.dll* you will need to make those Assemblies available to your end users as well.

The Crainiate assemblies are the foundation of the graphical Business Entity Designer that you'll use later in this tutorial. Crainiate is a 3rd party assembly vendor. Please be aware that Consultingwerk provides these assemblies for executing the *Business Entity Designer* only. You are not licensed to build and deploy you own applications based on the Crainiate assemblies unless you obtain the required license from Crainiate!

## OERA vs. OERA

If you take a closer look to the project directories you will recognize two OERA directories. *\OERA* contains the procedural part of the OERA implementation, originally released by Progress Software. *Consultingwerk\OERA* contains an OO OERA implementation developed by Consultingwerk Ltd. Both versions can coexist and do not impede each other.

For more information refer to chapter OERA vs. OERA in the appendix.

# Setting up Project Properties

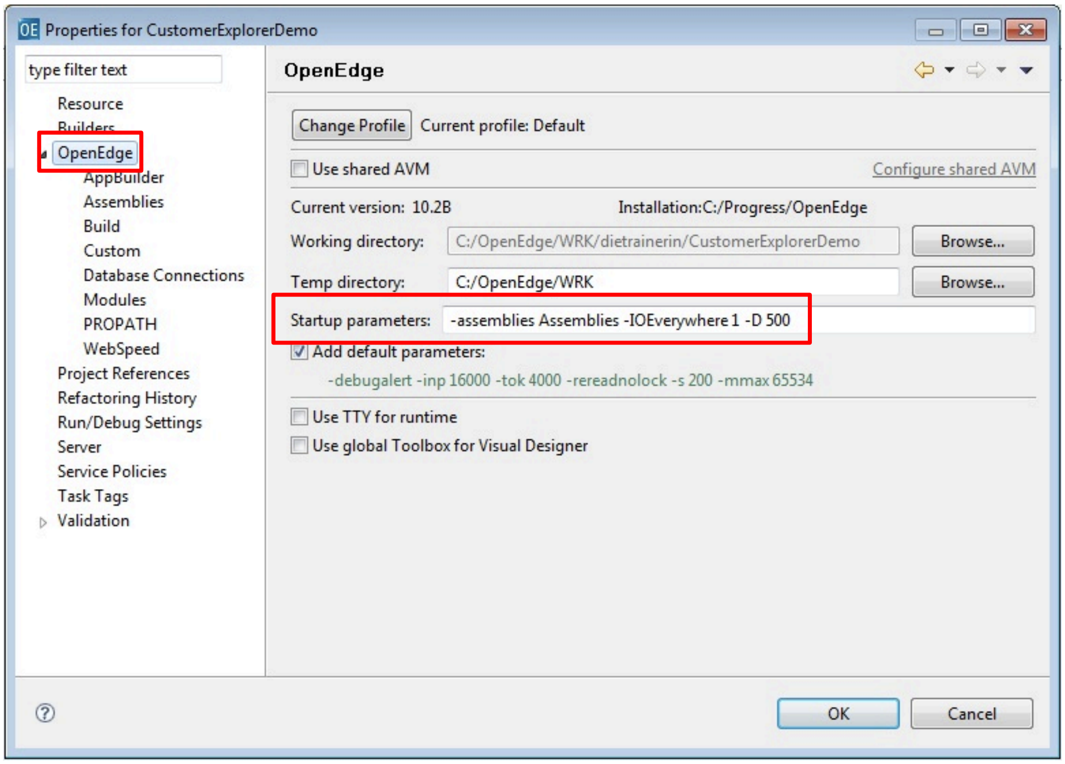
This section covers all project properties like startup parameters, PROPATH and database configurations that are required to compile and edit the files of the sample application.

## Startup Parameter

- Right click on the top-level node in the resources view representing the OpenEdge project you have just imported.
- Open the **Project Properties** dialog from the context menu and expand **OpenEdge** in the tree view on the left.

In the *Startup parameters* section specify the startup parameters below:

| Parameter | Value | Description |
|---|---|---|
| -assemblies | Assemblies | Shows the AVM that the assemblies.xml file (also referencing the required Infragistics Assemblies) and the SmartComponent Library assemblies are located in the Assemblies directory. |
| -IOEverywhere | 1 | When upgrading to 10.2B04 (or 10.2B02 at least) we recommend to use the new but not yet documented **-IOEverywhere 1** startup parameter. It removes the restrictions of using input-blocking statements in functions or non-void methods and activates a fix for a bug that is in the product since User Defined Functions have been introduced in Progress version 8.3 or 9.0.<br>Input-blocking statements typically used in a GUI for .NET application are the WAIT-FOR oForm:ShowDialog() or PROCESS EVENTS.<br>For further information refer to the following release note: http://wiki.dynamics4.net/D4wiki/SmartComponentLibrary/ReleaseNotes/ReleaseNote020. |
| -D | 500 | Use Directory Size (-D) to change the number of compiled procedure or class directory entries.<br>For further information refer to the OpenEdge documentation: OpenEdge Deployment: Startup Command and Parameter Reference. |



Current profile: Default

□ Use shared AVM

Current version: 10.2B

Installation:C:/Progress/OpenEdge

Working directory: C:/OpenEdge/WRK/dietrainerin/CustomerExplorerDemo

Temp directory: C:/OpenEdge/WRK

Startup parameters: -assemblies Assemblies -IOEverywhere 1 -D 500

☑ Add default parameters:

-debugalert -inp 16000 -tok 4000 -rereadnolock -s 200 -mmax 65534

□ Use TTY for runtime

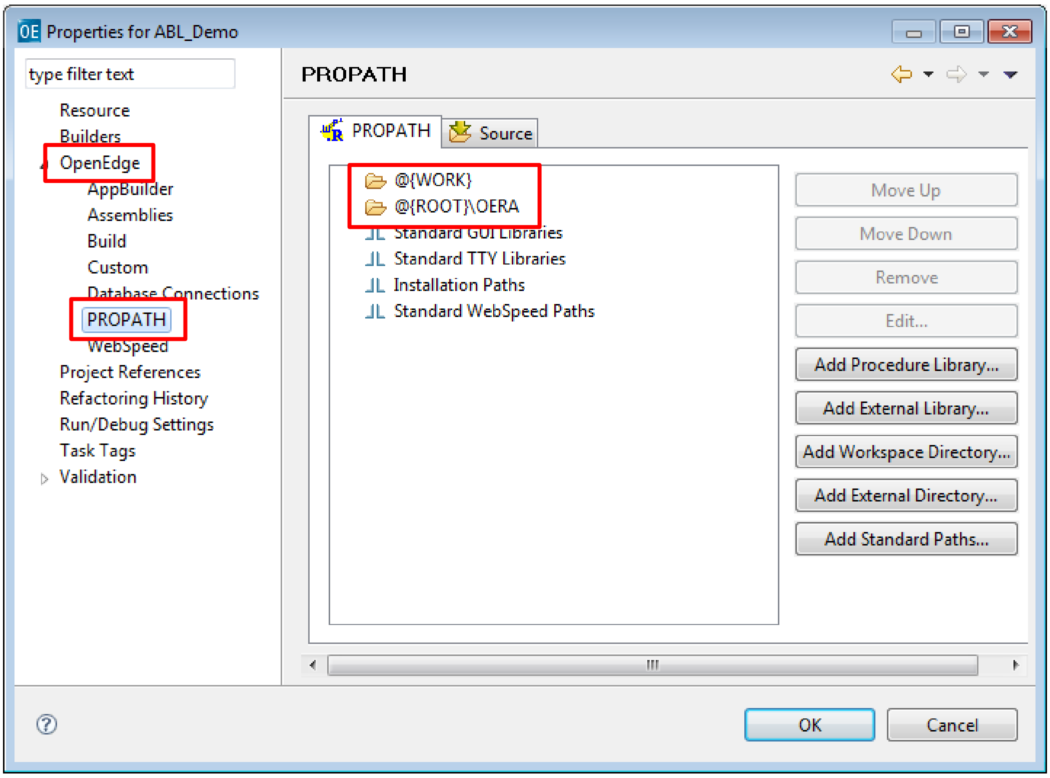
□ Use global Toolbox for Visual Designer

**Please note**: there is no delimiter but a blank between the parameters! And **don’t copy/paste hyphens** from the tutorial!

## Setting the PROPATH

The PROPATH was set automatically during the import of the existing project. The *CustomerExplorer_Demo_yyyymmdd_src.zip* file contains a **.propath** text file with all necessary PROPATH entries.

- To review the PROPATH, open the **Project > Properties**, expand the *OpenEdge* tree-view and select PROPATH.



@{WORK}
@{ROOT}\OERA
Standard GUI Libraries
Standard TTY Libraries
Installation Paths
Standard WebSpeed Paths

## Setting up connections to the customized sports2000 database

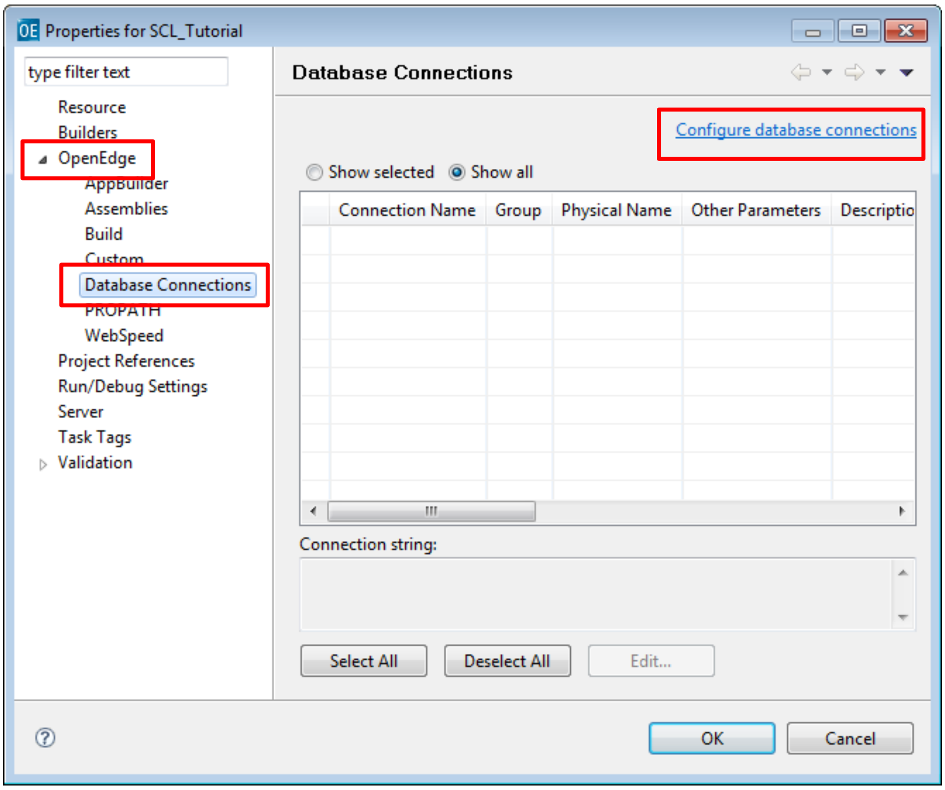
In this tutorial, you build an application which is based on a customized version of the sports2000 database. Don’t use a copy of the standard sports2000 database that you may have created earlier.

### Connect to the sports2000 database in the OpenEdge Architect

You now have to define a database connection for your own sample application as well as for the *CustomerExplorer* demo.

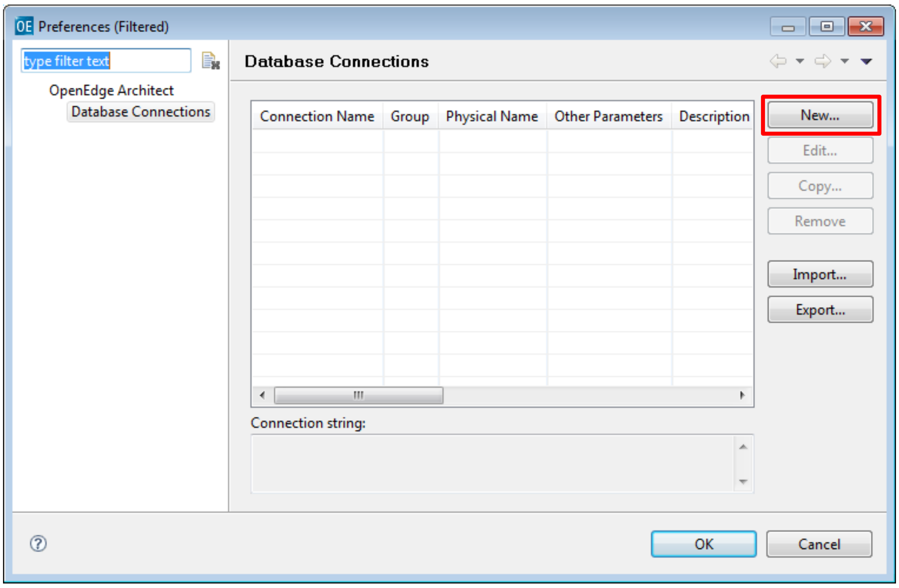
Database connection profiles contain all the information necessary to connect to a database. The information can include startup parameters, user name and password, host name, port number, and more. The database connection profile may also contain parameters for a JDBC/SQL connection to the database. OpenEdge Architect can use this database connection from plugins like the OpenEdge Database Navigator (Java based alternative to the Data Dictionary) or the Tools for Business Logic (T4BL), a graphical designer for ProDataSets with many functional limitations. For the purpose of this tutorial (and most of your actual ABL based work) you will not need to define the SQL connection.

- In the OpenEdge Architect menu open **Project > Properties > OpenEdge > Database Connections** and click the link **Configure database connections** in the upper right corner of the dialog.



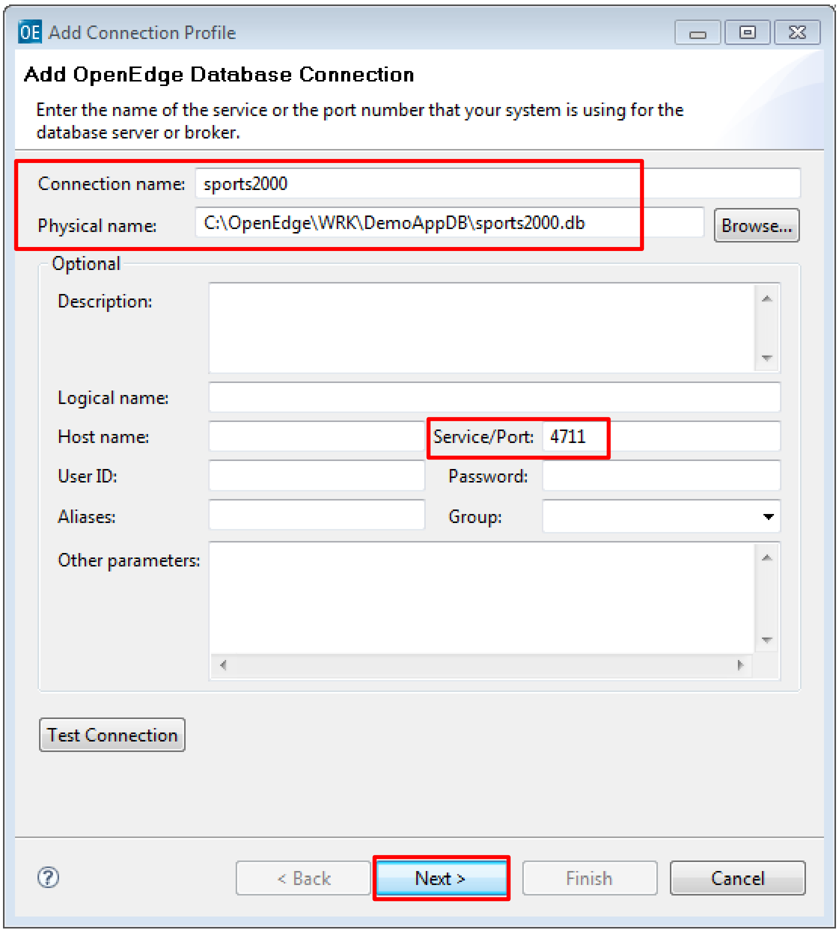
The **Preferences** dialog opens.

- Select **New** to create a new database connection.



The **Add Connection Profile** dialog opens:

- Type in the **Connection name** of the database: sports2000.
- Browse the *sports2000.db* file from your WRK directory where the copy of the customized sports2000 database resides.
- For training purposes only specify 4711 as the ***port number*** of the database.
- Click button **Next**



Add OpenEdge Database Connection

Enter the name of the service or the port number that your system is using for the database server or broker.

Connection name: sports2000

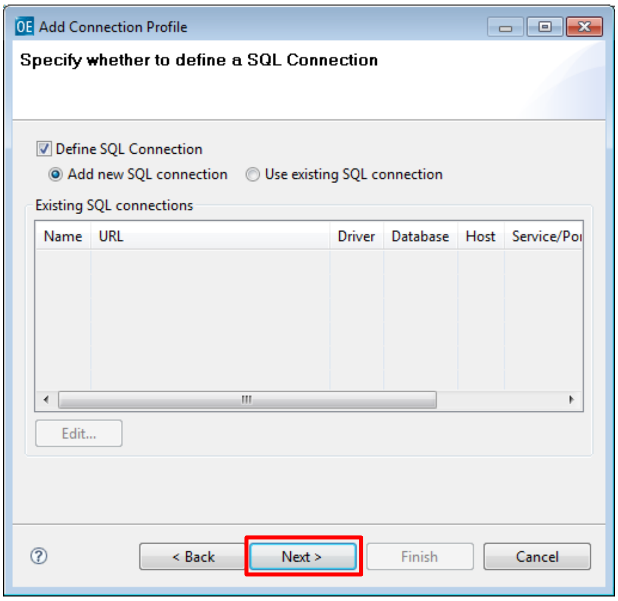
Physical name: C:\OpenEdge\WRK\DemoAppDB\sports2000.db

Service/Port: 4711

### *Optional: Specify whether to define a SQL connection.*

A SQL connection is necessary to display the tables and columns in the **DB Structure** View**.** You can create a new SQL connection or use an existing one.

- For tutorial purposes create a new one (default) and click **Next**.
- If you encounter any issues with the SQL connection on your system, don't worry! This is just optional.



Add Connection Profile

Specify whether to define a SQL Connection

☑ Define SQL Connection

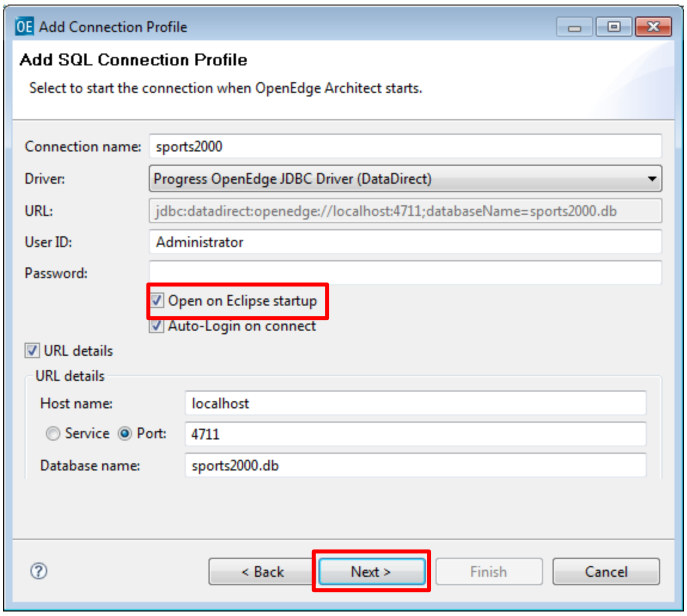
● Add new SQL connection ○ Use existing SQL connection

Existing SQL connections

| Name | URL | Driver | Database | Host | Service/Po |
|---|---|---|---|---|---|

The OpenEdge Architect copies the most important entries to the SQL Connection Profile.

- Validate the *Connection name* and *Port*, select the checkbox *Open on Eclipse startup* if you want to connect to the SQL profile when starting OpenEdge Architect.
- Click **Next**.



Add Connection Profile

Add SQL Connection Profile

Select to start the connection when OpenEdge Architect starts.

Connection name: sports2000

Driver: Progress OpenEdge JDBC Driver (DataDirect)

URL: jdbc:datadirect:openedge://localhost:4711;databaseName=sports2000.db

User ID: Administrator

Password:

☑ Open on Eclipse startup

☑ Auto-Login on connect

☑ URL details

URL details

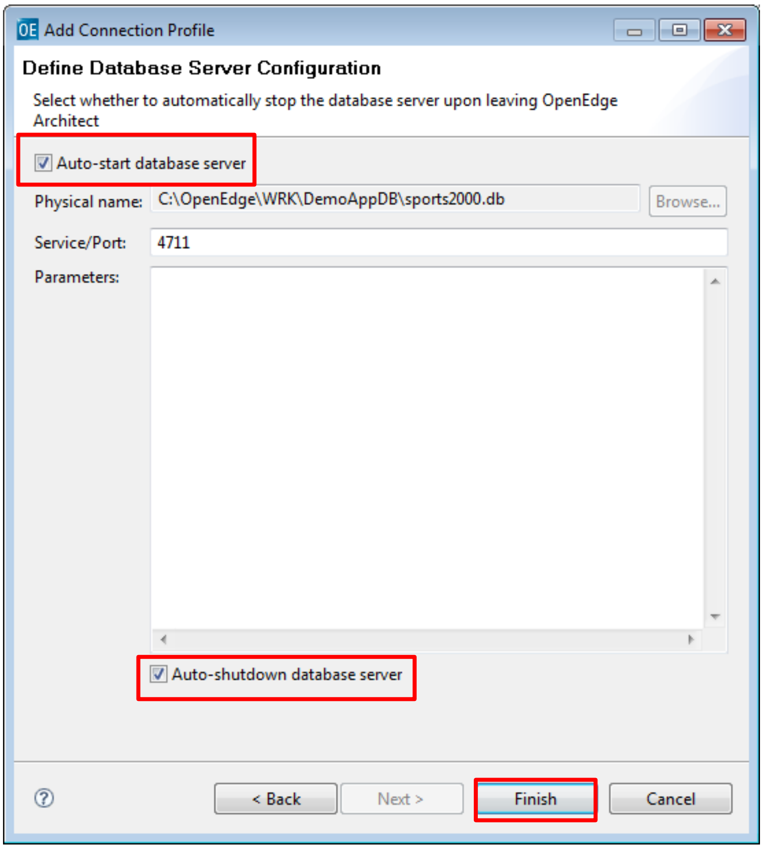
Host name: localhost

○ Service ● Port: 4711

Database name: sports2000.db

### *Define Database Server Configuration*

- Select whether to automatically start (default) and stop (non-default) the database server upon starting or leaving OpenEdge Architect.



Define Database Server Configuration

Select whether to automatically stop the database server upon leaving OpenEdge Architect

☑ Auto-start database server

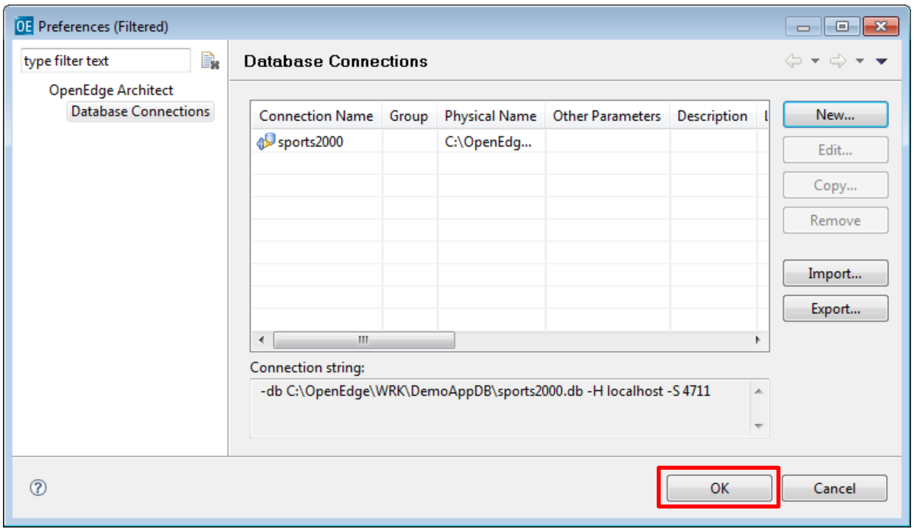
Physical name: C:\OpenEdge\WRK\DemoAppDB\sports2000.db

Service/Port: 4711

Parameters:

☑ Auto-shutdown database server

- Click **Finish**.



Database Connections

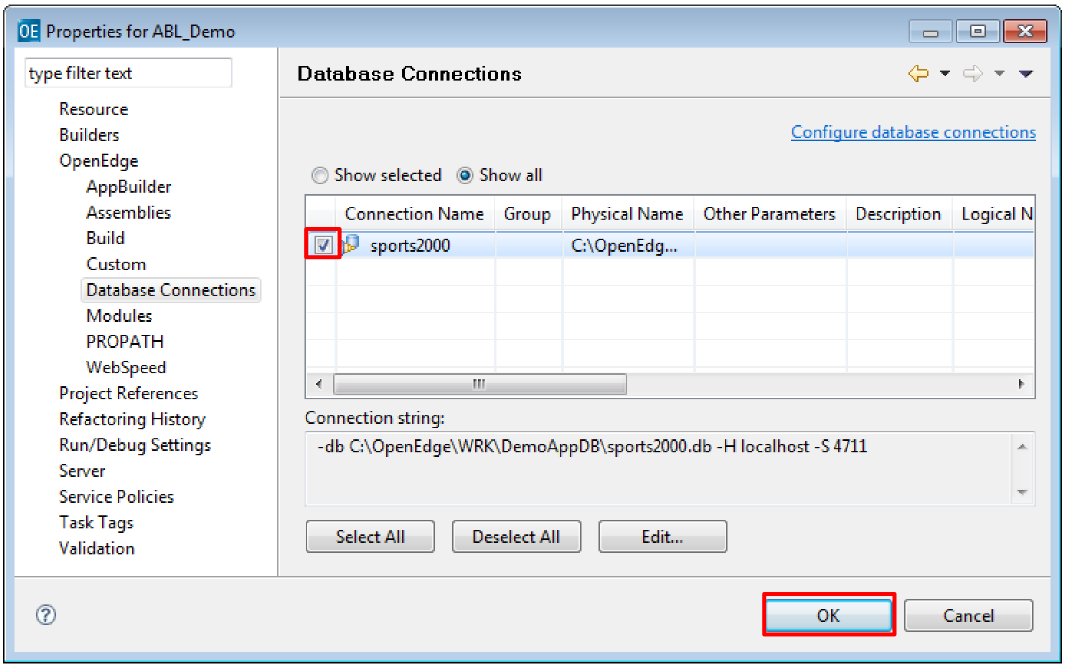
| Connection Name | Group | Physical Name | Other Parameters | Description | l |
|---|---|---|---|---|---|
| sports2000 | | C:\OpenEdg... | | | |

Connection string:

-db C:\OpenEdge\WRK\DemoAppDB\sports2000.db -H localhost -S 4711

- In order to connect to the sports2000 database select the checkbox on the left and click **OK**.

The OpenEdge runtime restarts.



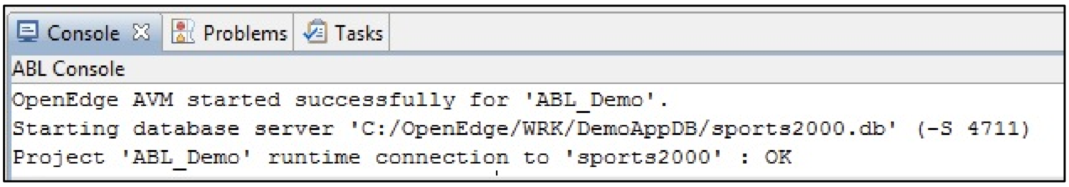
○ Show selected ● Show all

| | Connection Name | Group | Physical Name | Other Parameters | Description | Logical N |
|---|---|---|---|---|---|---|
| ☑ | sports2000 | | C:\OpenEdg... | | | |

Connection string:

```
-db C:\OpenEdge\WRK\DemoAppDB\sports2000.db -H localhost -S 4711
```

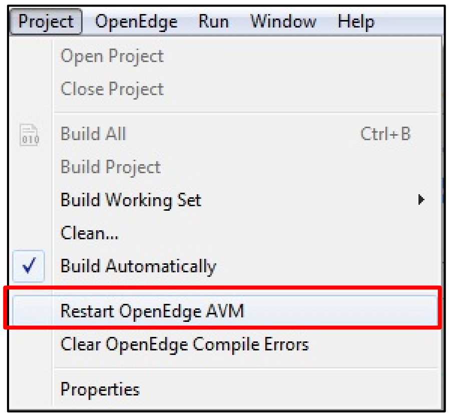
**Congratulation**! You have successfully finished the database configuration setup.



```
OpenEdge AVM started successfully for 'ABL_Demo'.
Starting database server 'C:/OpenEdge/WRK/DemoAppDB/sports2000.db' (-S 4711)
Project 'ABL_Demo' runtime connection to 'sports2000' : OK
```

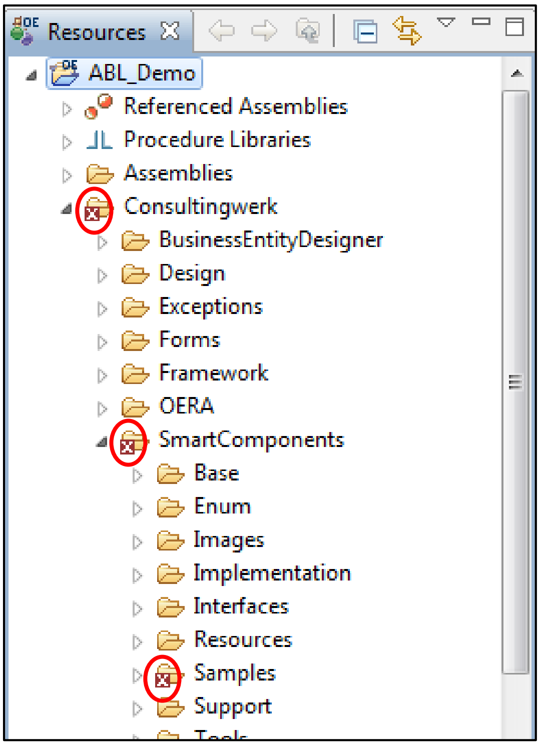
If the runtime does not start automatically you must restart the OpenEdge AVM manually.

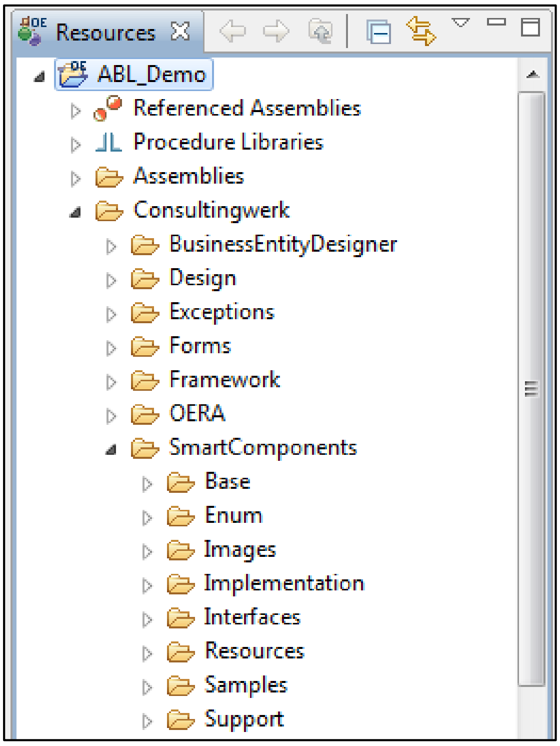
- To do so go to the **Project** menu and select **Restart OpenEdge AVM**.



## Compile the source code of the SmartComponent Library

You may still have several error flags, indicating that the database is missing or inconsistencies occurred between the various compiled classes. The red cross decorating a file or a folder indicates a compilation issue. OpenEdge Architect will not attempt to recompile automatically after you have provided modified startup parameters or additional database connections.



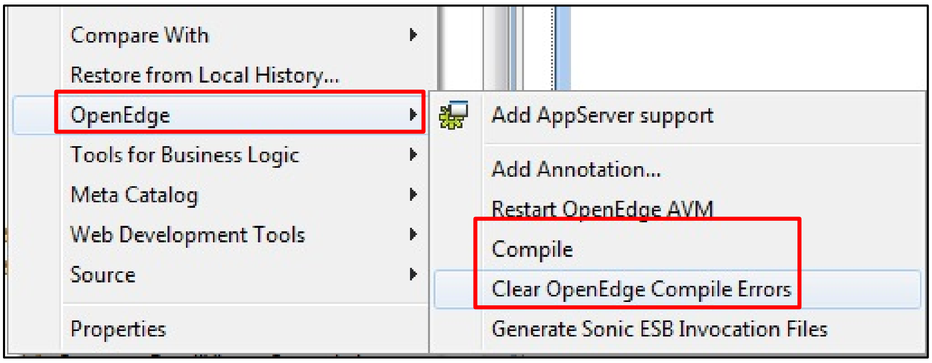


Before compilation

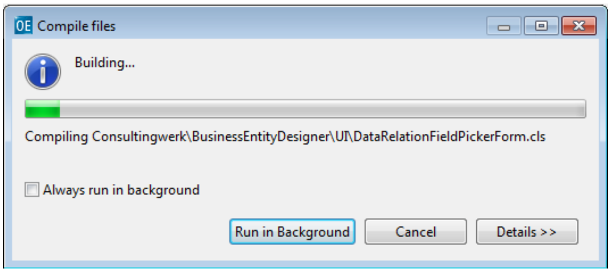
After Compilation

To clear the error flags follow these steps:

- **Clear OpenEdge Errors**: Right-click to the project folder and select the *OpenEdge > Clear OpenEdge Compile Errors*.
- **Compile the SmartComponent Library**: Right-click to the project folder and select *OpenEdge > Compile.*



OpenEdge Architect compiles the requested files.



Building...

Compiling Consultingwerk\BusinessEntityDesigner\UI\DataRelationFieldPickerForm.cls

□ Always run in background

The recommended approach to compile the SmartComponent Library is to delete existing R-code before compiling all files, because old R-code files can cause inconsistencies between compiled classes which typically results in the requirement to compile multiple times before all compilation issues are resolved.

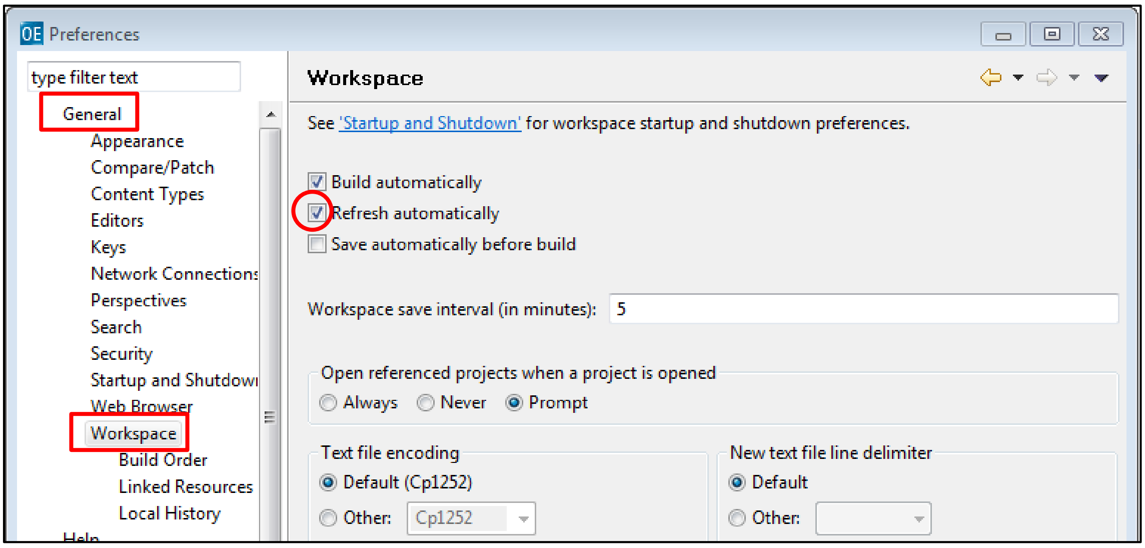
## Optional: Activate Auto-Refresh Option

It is often necessary to refresh the project folder, e.g. when you add files to the project folder in the file system. You need to refresh the project to make those files visible.

You can activate the auto refresh option in the **Window > Preferences** from the menu.

- Expand the tree-view **General > Workspace** and check the **Refresh automatically** option.

The next time you make changes to the project files the workspace will refresh automatically.



Workspace

See 'Startup and Shutdown' for workspace startup and shutdown preferences.

☑ Build automatically
☑ Refresh automatically
□ Save automatically before build

Workspace save interval (in minutes): 5

Open referenced projects when a project is opened
○ Always ○ Never ● Prompt

Text file encoding
● Default (Cp1252)
○ Other: Cp1252

New text file line delimiter
● Default
○ Other:

For more information about *Architect Preferences* refer to the article OpenEdge Architect Preferences in the Consultingwerk Blog (www.blog.consultingwerk.de).

# Developing Business Entities

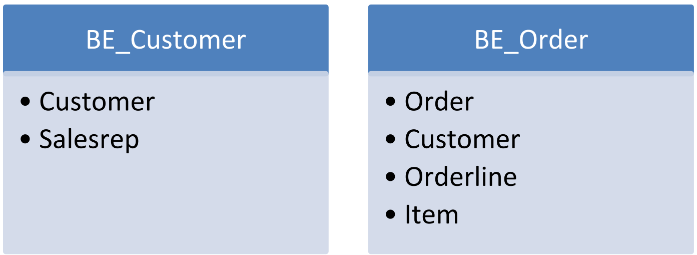
## Introduction

A **Business Entity** is a service object that provides access to data within an application. It defines a data structure and contains logic to perform actions related to this data. Business Entities used by the *SmartComponent Library* are built around a data structure that is defined using a ProDataset.

Business entities are more than simple data containers. They represent data from a business perspective, providing a logical view of data that may or may not relate directly to one or more database tables, xml files, text files or web services or provide abstraction from the physical form the data have in a system. Business Entities are business-centric, not data-centric, that means a Business Entity does **not** have any responsibility for accessing data sources or retrieving or updating data from or in a database. This is typically the responsibility of the **Data Access Object**.

Each Business Entity is normally paired with a Data Access object which is used as a delegate that manages and uses the actual data sources for reading and updating data in a database or a different system. The Data Access Object and Business Entities in the context of the *SmartComponent Library* are based on ProDataSets because of their capabilities to combine data from several data sources (e.g. tables) in a logical view with relations.

Before starting to build a new *.NET Form* using the *SmartComponent Library* let's take a closer look at the data the user interface will be based on. In the tutorial you will build a Form which presents the *Customer* and *Salesrep* data on the one hand and *Order, Customer*, *OrderLine*, *Item* data on the other hand. In other words you need two Business Entities, one for the logical structure between *Customer* and *SalesRep* and a second one for the view on *Order, Customer, OrderLine and Item*.



BE_Customer

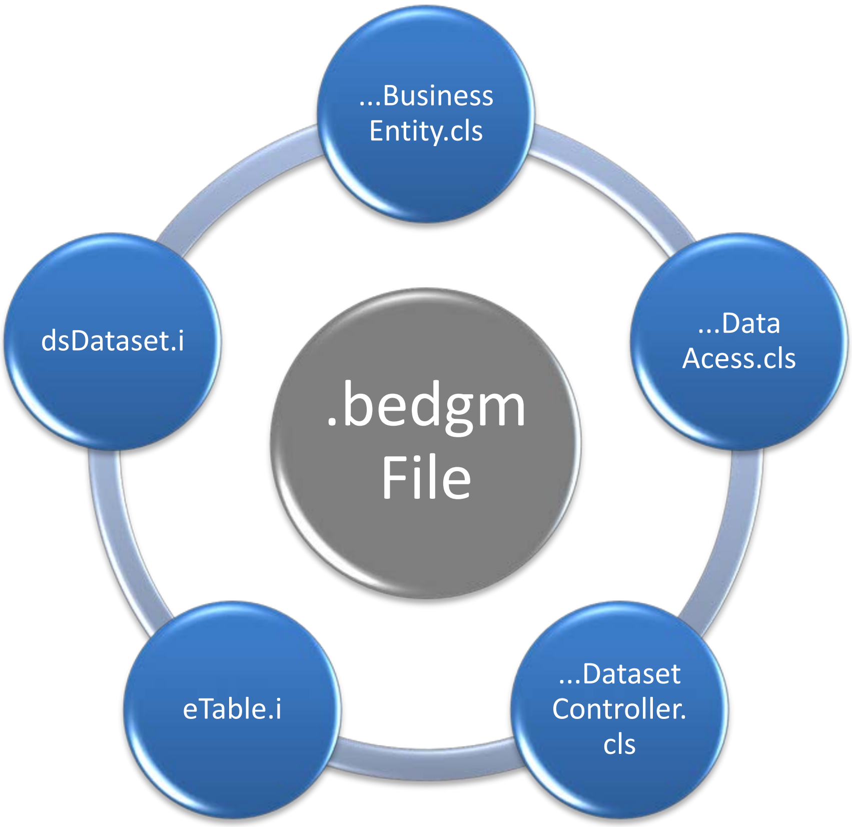
- Customer
- Salesrep

BE_Order

- Order
- Customer
- Orderline
- Item

Consultingwerk provides a graphical **Business Entity Designer** tool that supports developers when developing Business Entities, Data Access Objects and the related files for Temp-Table and ProDataSets in every aspect.

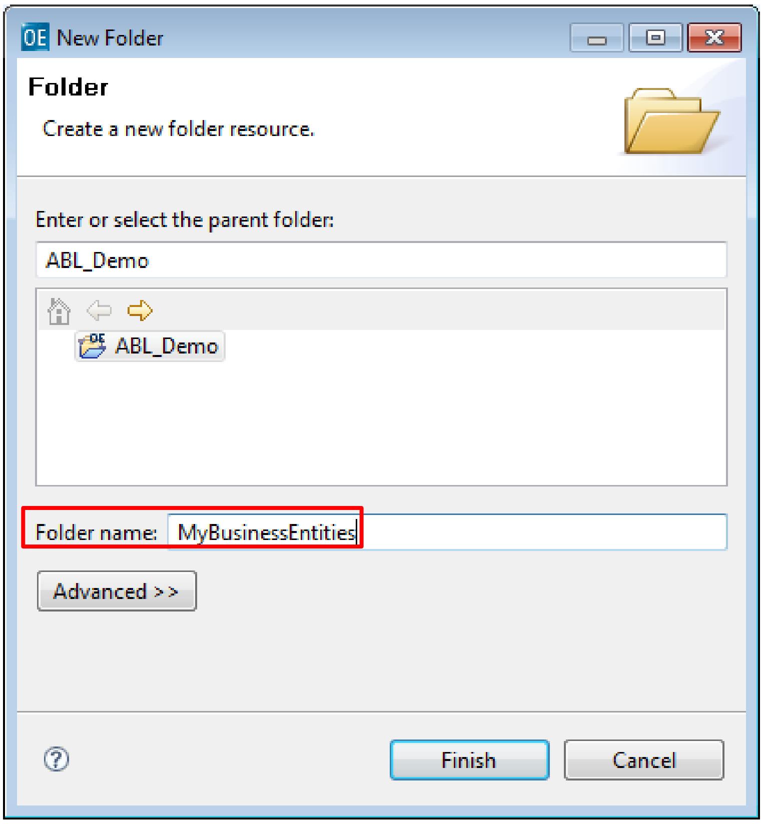
## Introducing the Business Entity Designer



The **Business Entity Designer** allows you to build graphical models that represent Business Entities as logical components. Using the graphical designer you create Business Entity components by designing the ProDataset typically by adding all necessary elements such as Temp-Table 's, relations, indexes and fields using drag and drop. As you build your components, the **Business Entity Designer** allows you to store all information about the structure and relationship in a **B**usiness **E**ntity **D**ia**g**ra**m** file with the extension .bedgm.

After defining the logical structure of a Business Entity, you generate and compile the corresponding ABL source code: Include files for each *Temp-Table* and *ProDataSet* definitions and the appropriate class files for the *Business Entity* itself, the *DataAccess* object and the *DatasetController* object.

- Before you start modeling your first Business Entity, create a new folder *MyBusinessEntities* in your project, where you'll store all entity files.



Folder

Create a new folder resource.

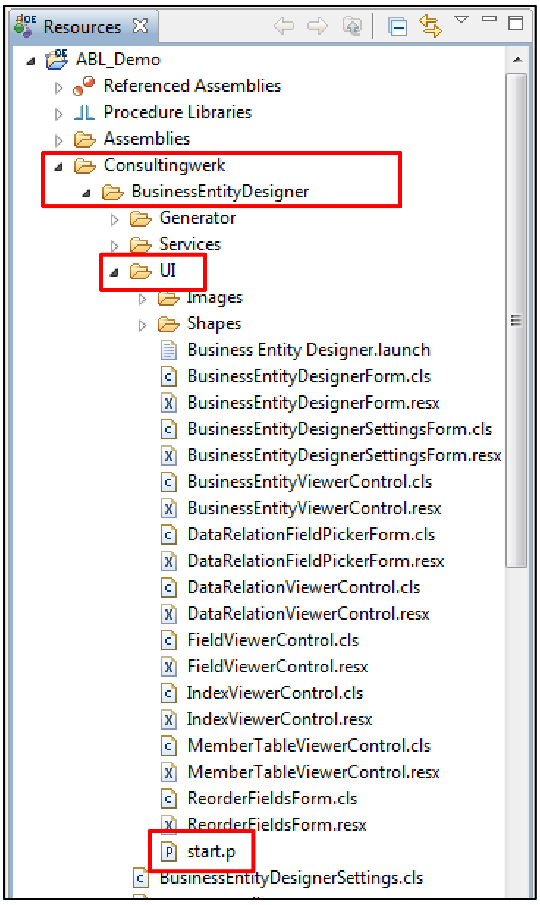
Enter or select the parent folder: ABL_Demo

Folder name: MyBusinessEntities

## Start the Business Entity Designer

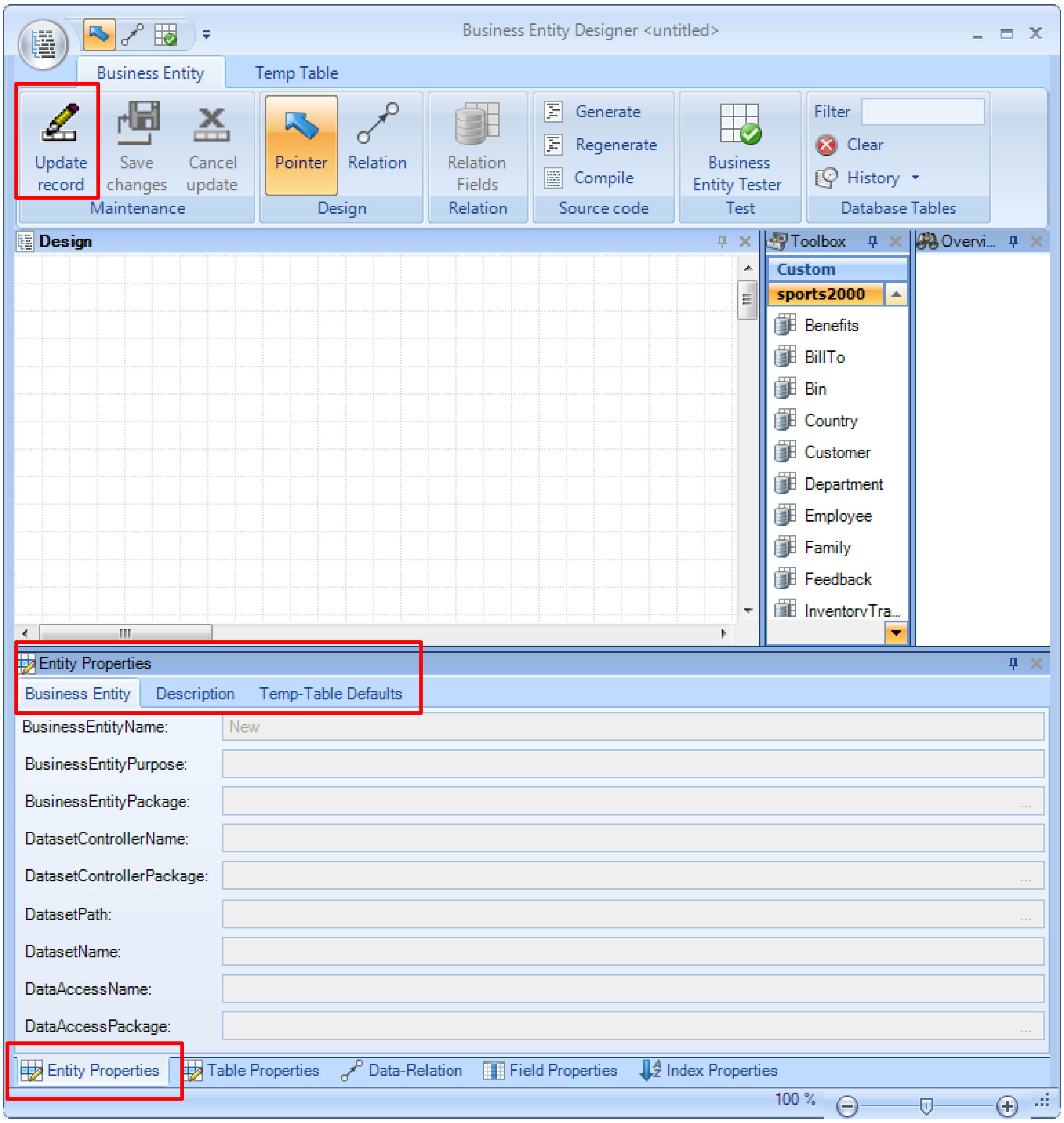
To start the **Business Entity Designer**, run the procedure *start.p* from directory *Consultingwerk\BusinessEntityDesigner\UI*.

You can customize the RUN configuration. For further information, please refer to chapter Changing Run Configurations in the appendix of the tutorial.



```
Resources
ABL_Demo
  Referenced Assemblies
  Procedure Libraries
  Assemblies
  Consultingwerk
    BusinessEntityDesigner
      Generator
      Services
      UI
        Images
        Shapes
        Business Entity Designer.launch
        BusinessEntityDesignerForm.cls
        BusinessEntityDesignerForm.resx
        BusinessEntityDesignerSettingsForm.cls
        BusinessEntityDesignerSettingsForm.resx
        BusinessEntityViewerControl.cls
        BusinessEntityViewerControl.resx
        DataRelationFieldPickerForm.cls
        DataRelationFieldPickerForm.resx
        DataRelationViewerControl.cls
        DataRelationViewerControl.resx
        FieldViewerControl.cls
        FieldViewerControl.resx
        IndexViewerControl.cls
        IndexViewerControl.resx
        MemberTableViewerControl.cls
        MemberTableViewerControl.resx
        ReorderFieldsForm.cls
        ReorderFieldsForm.resx
        start.p
      BusinessEntityDesignerSettings.cls
```

The **Business Entity Designer** opens.



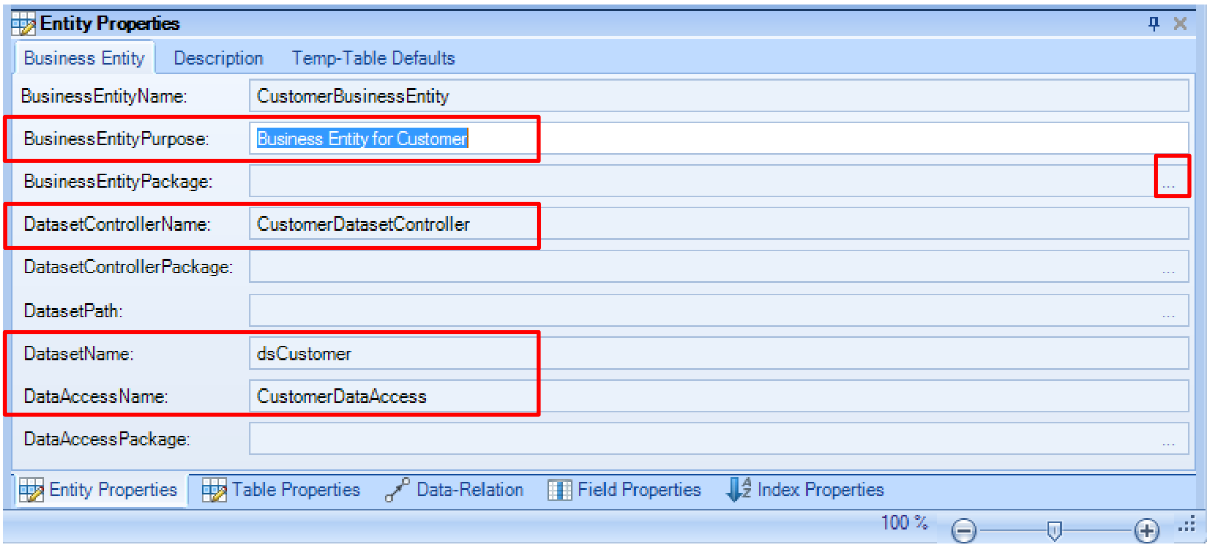
BusinessEntityName: New
BusinessEntityPurpose:
BusinessEntityPackage:
DatasetControllerName:
DatasetControllerPackage:
DatasetPath:
DatasetName:
DataAccessName:
DataAccessPackage:

At the bottom there is a viewer for all component properties you are going to create: **Entity Properties**, **Table Properties**, **Data-Relation**, **Field Properties** and **Index Properties**. By default all property views are read-only until you select an object on the design canvas and use the **Update record** button in the *Maintenance* group of the ribbon.

## Create a new Business Entity for Customer and Salesrep

- To enable the **Entity Properties** view to enter name and other information for your Business Entity, click the **Update record** button in the *Maintenance* group of the Ribbon.
- Type the name for the Business Entity: **CustomerBusinessEntity** and press the Tab Key or leave the field.

To enforce component naming standards*, Business Entity Designer* automatically determines the names for the ProDataSet and the class files following the naming conventions of Consultingwerk. You can use your own naming conventions if you prefer.



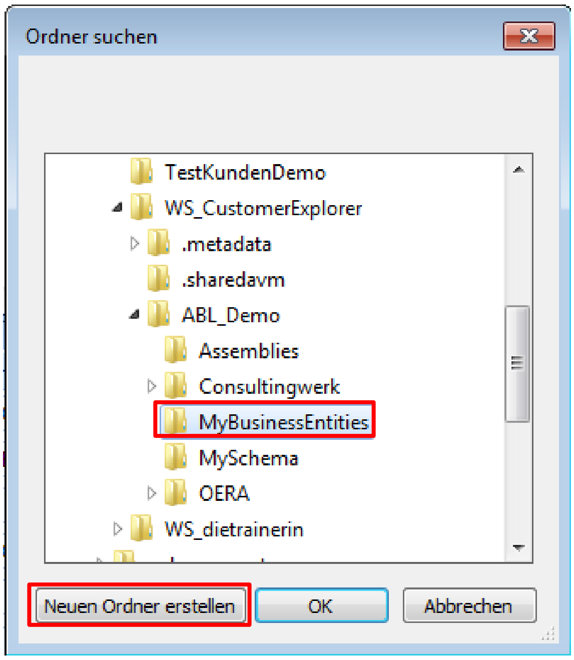
| | |
|---|---|
| BusinessEntityName: | CustomerBusinessEntity |
| BusinessEntityPurpose: | Business Entity for Customer |
| BusinessEntityPackage: | |
| DatasetControllerName: | CustomerDatasetController |
| DatasetControllerPackage: | |
| DatasetPath: | |
| DatasetName: | dsCustomer |
| DataAccessName: | CustomerDataAccess |
| DataAccessPackage: | |

After you've defined the names for the Business Entity and its main components, *ProDataSet*, *DatasetController* object and *DataAccess* object, you need to specify the directory to store the files in.

- Press the dotted button on the right side of the *BusinessEntityPackage* field and select the directory **MyBusinessEntities** you defined previously.

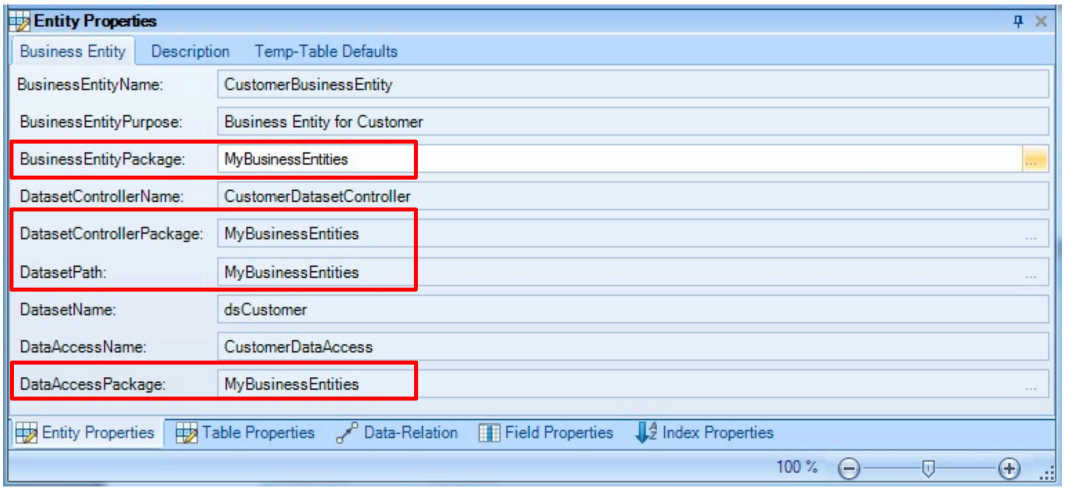
In the OpenEdge ABL the package of an object refers to the folder that contains this class. For our Business Entities the package does so define the directory relative to the working directory that typically contains the Business Entity class source and other related source code.

Note that the package name will replace the slash from a folder name with a "." (dot).



You can create a new directory for your project right here by clicking on the button **Neuen Ordner erstellen** (create new folder).

The **Business Entity Designer** automatically defaults the directory or package chosen for the Business Entity for *DatasetPath* and *DataAccessPackage*. By default all components are saved into the same directory. You can change the directory for every component if you prefer.



BusinessEntityName: CustomerBusinessEntity
BusinessEntityPurpose: Business Entity for Customer
BusinessEntityPackage: MyBusinessEntities
DatasetControllerName: CustomerDatasetController
DatasetControllerPackage: MyBusinessEntities
DatasetPath: MyBusinessEntities
DatasetName: dsCustomer
DataAccessName: CustomerDataAccess
DataAccessPackage: MyBusinessEntities

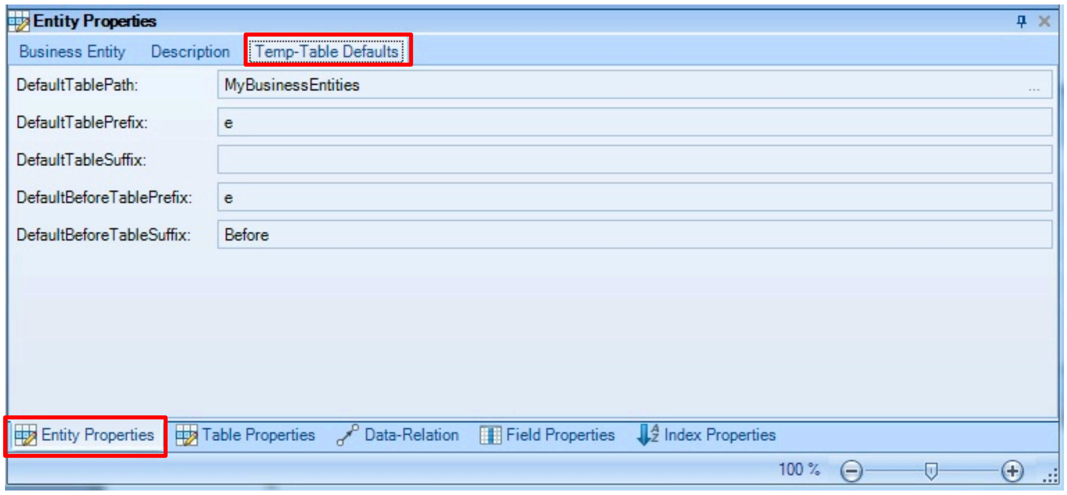
## Summary for CustomerBusinessEntity

| Field | Description |
|---|---|
| BusinessEntityName | CustomerBusinessEntity |
| BusinessEntityPurpose | Business Entity for Customer |
| BusinessEntityPackage | MyBusinessEntities |
| DatasetControllerName | CustomerDatasetController |
| DatasetControllerPackage | MyBusinessEntities |
| DatasetPath | MyBusinessEntities |
| DatasetName | dsCustomer |
| DataAccessName | CustomerDataAccess |
| DataAccessPackage | MyBusinessEntities |

## Optional: Change Temp-Table settings

To change the default settings for Temp-Table go to the *Temp-Table Defaults* tab in the Entity Properties viewer.

The **DefaultTablePath** is the same as for the ProDataSet and all other components of the Business Entity. Optionally you can change these settings for each individual Temp-Table of the Business Entity design.



DefaultTablePath: MyBusinessEntities

DefaultTablePrefix: e

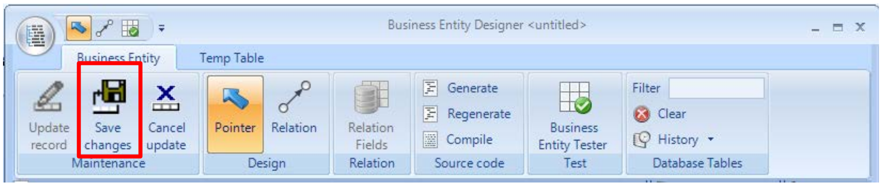
DefaultTableSuffix:

DefaultBeforeTablePrefix: e

DefaultBeforeTableSuffix: Before

## Save the Business Entity

- To save the modifications to the Business Entity *CustomerBusinessEntity* definition click on the **Save changes** button in the *Maintenance* section of the Ribbon.



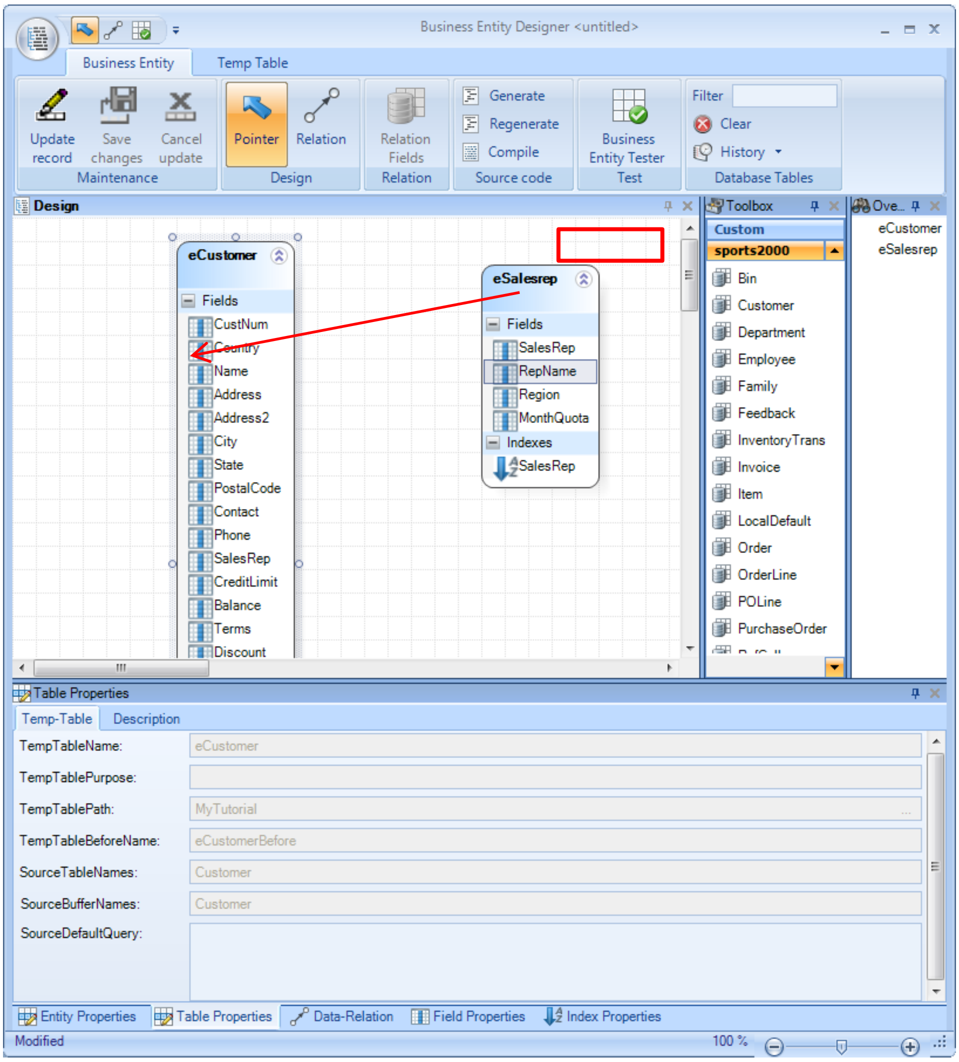
The *Entity Properties* fields become read only.

## Create the required Temp-Table

- Select the **Customer** table from the *Table Toolbox*, hold down the left mouse button and drag it in the *Design Canvas* of the **Business Entity Designer**.
- Release the mouse button.
- Repeat the steps to create a **Salesrep** table.

Please note that the Temp-Table s have the same structure like the database tables, but the name is prefixed with the default prefix of **e** (for entity).

The *Design Area* should look like this:



TempTableName: eCustomer

TempTablePurpose:

TempTablePath: MyTutorial

TempTableBeforeName: eCustomerBefore

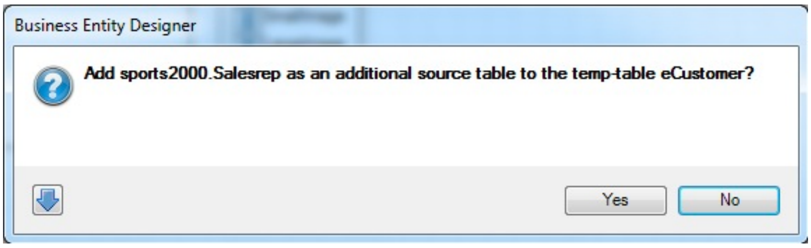
SourceTableNames: Customer

SourceBufferNames: Customer

SourceDefaultQuery:

While dragging the **Salesrep** table you will notice a different mouse pointer when dragging over the **eCustomer** Temp-Table or over the white area of the design canvas. At this moment, you should NOT drag the **Salesrep** table to the **eCustomer** table. Instead drag it to the white area of the design canvas.

If you accidentally drag the **Salesrep** table to the **eCustomer** table the following message appears:



Business Entity Designer

Add sports2000.Salesrep as an additional source table to the temp-table eCustomer?

- Select **No** and try again!

If you'd have chosen **Yes** on the above dialog the **Salesrep** table wouldn't have been added as a separate table. The fields of the **Salesrep** table would have been added to the **eCustomer** Temp-Table and the **Salesrep** table would have been added as a second source table to the **eCustomer** Temp-Table . This may be desired in some situations, but not in this case. For more considerations on this topic, please consult the chapter Business Entity Design Considerations.

## Temp-Table Names

The Temp-Table names are predefined by the *Business Entity Desginer*. Every Temp-Table is prefixed with **e** (member of an Entity) followed by the source table name. The default prefix of **e** can be customized in the options dialog of the *Business Entity Designer*.

You can change the settings for an individual Temp-Table in the *Table Properties* viewer of the *Business Entity Designer*.

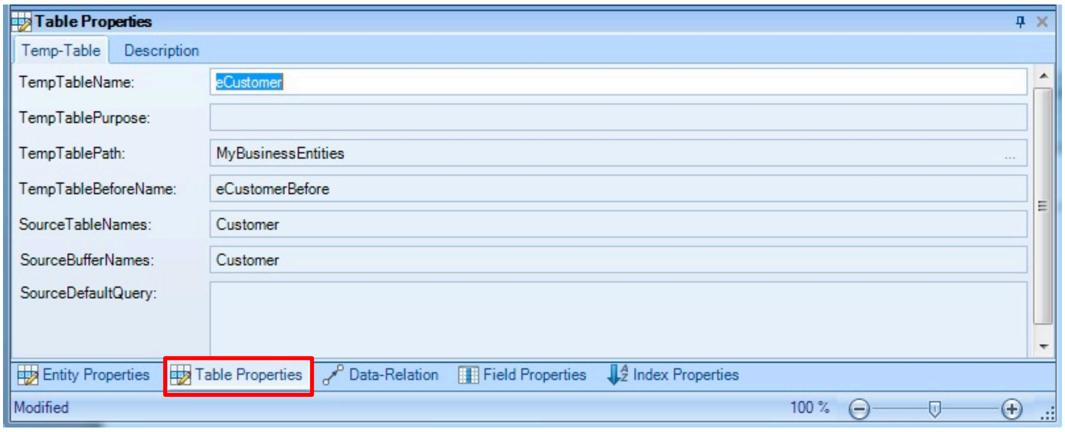


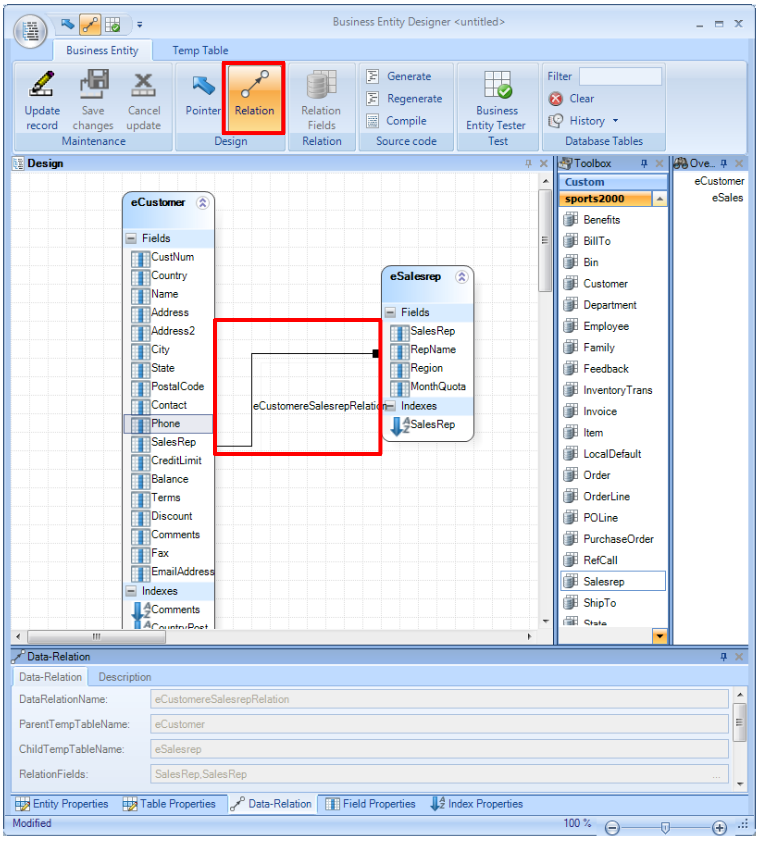
Table Properties

| | |
|---|---|
| TempTableName: | eCustomer |
| TempTablePurpose: | |
| TempTablePath: | MyBusinessEntities |
| TempTableBeforeName: | eCustomerBefore |
| SourceTableNames: | Customer |
| SourceBufferNames: | Customer |
| SourceDefaultQuery: | |

## Define a Relation

- To define the relation between two tables, press the **Relation** button in the *Design* section of the Ribbon.
- Click on the parent table (eCustomer) and hold down the left mouse button and drag the relation to the child table (eSalesrep).
- Release the mouse button.

The *Design Area* should look like this:



DataRelationName: eCustomereSalesrepRelation

ParentTempTableName: eCustomer

ChildTempTableName: eSalesrep

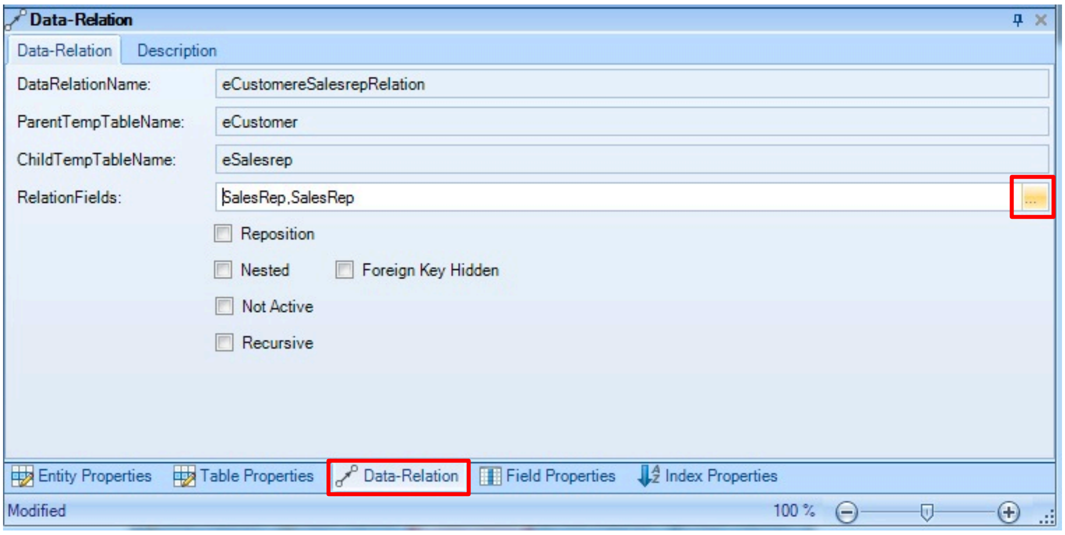
RelationFields: SalesRep,SalesRep

The name of the new relation is set automatically by the *Business Entity Designer* and describes the parent-child relationship of the tables: **eCustomerSalesrepRelation**. The first table name is the parent table, the second one the child table. You can customize how the *Business Entity Designer* builds the name of the relation in the **Options** dialog of the *Business Entity Designer*.

By default the *Business Entity Designer* searches for fields of the same name to build up a relation. In this case the relation is built by field *SalesRep* from the customer and salesrep table.

⚠ To check the relation and to change it, go to the *Data-Relation section* of the Designer. It is read-only. If you want to change field names or relation fields press button **Update record** in the

Ribbon. As already explained for entity or table properties you can change all predefined setting for the relation here.



DataRelationName: eCustomereSalesrepRelation

ParentTempTableName: eCustomer

ChildTempTableName: eSalesrep

RelationFields: SalesRep,SalesRep

□ Reposition

□ Nested □ Foreign Key Hidden

□ Not Active

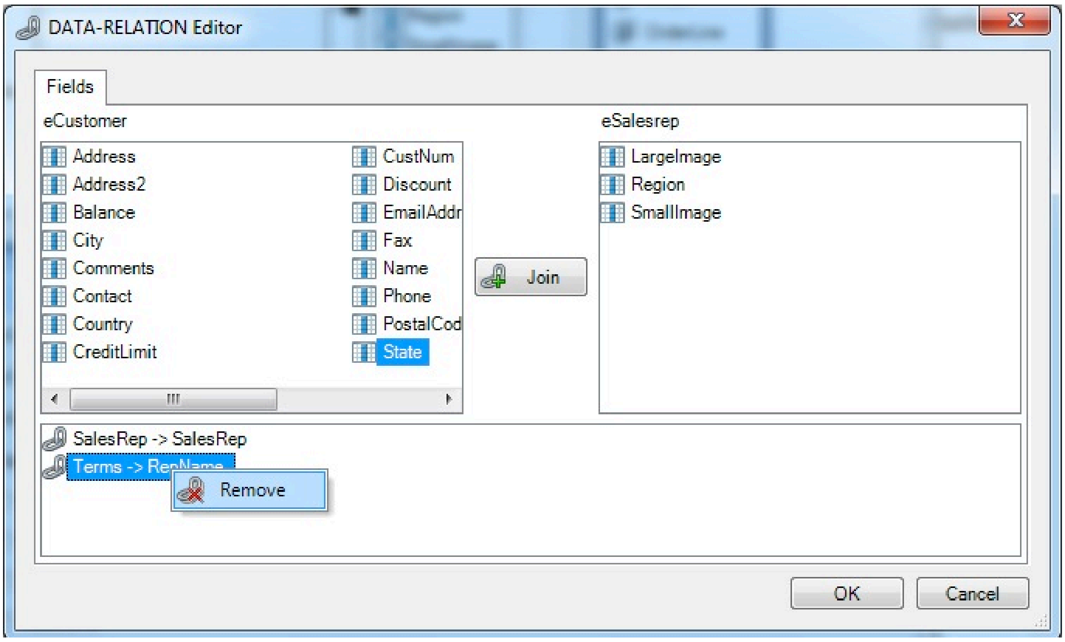
□ Recursive

If you want to change the relation fields, you can type the parent.child pair in the fill-in *RelationFields* in the Data-Relation viewer. Alternatively you can use the *Data-Relation Editor*.

- To open it, click the dotted button at the right end of the RelationFields fill-in.
- To define a field pair mark a field in both tables by holding the **Ctrl-key** and select **Join**.
- Or drag and drop the field from left to right or vice versa.

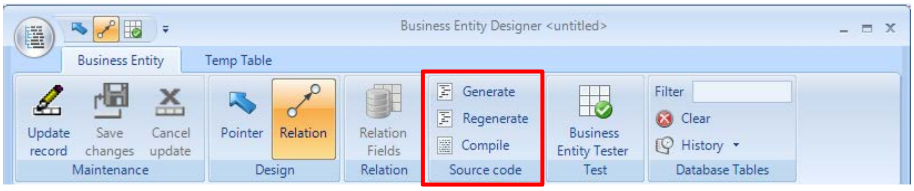
The new field pair appears in the lower section of the dialog.

- To remove a field pair right-mouse-click the field pair and select **Remove**.

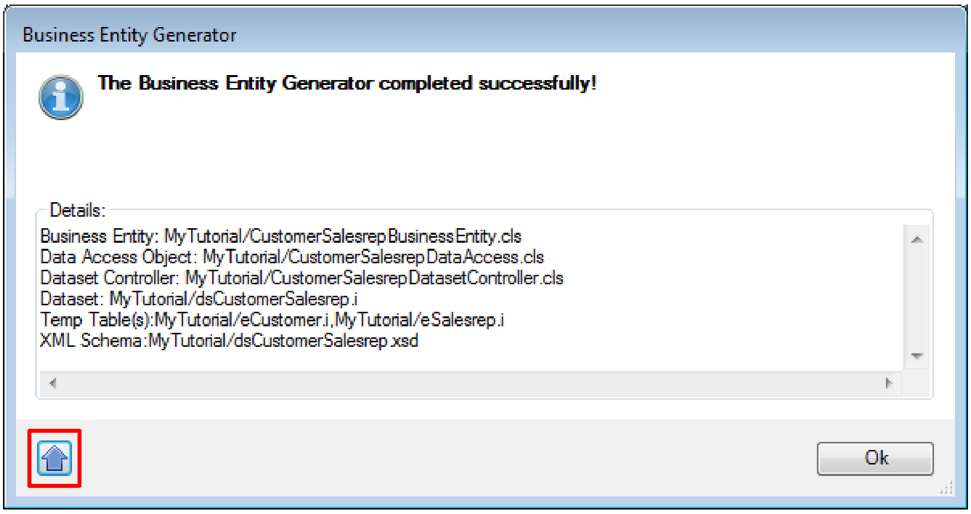


## Generate source code

- To generate and compile all required source code click the button **Generate** in the *Source Code* section of the Ribbon.



A message appears as a confirmation that the Business Entity has been successfully generated.. When you click the arrow button in the left lower corner you can verify what files have been generated for you.



**The Business Entity Generator completed successfully!**

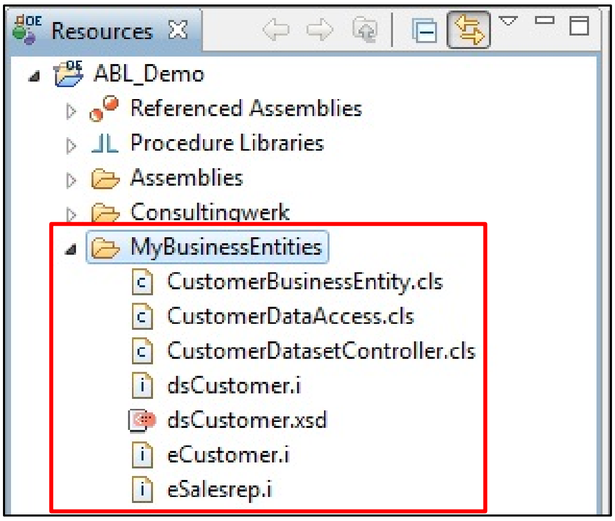
Details:

```
Business Entity: MyTutorial/CustomerSalesrepBusinessEntity.cls
Data Access Object: MyTutorial/CustomerSalesrepDataAccess.cls
Dataset Controller: MyTutorial/CustomerSalesrepDatasetController.cls
Dataset: MyTutorial/dsCustomerSalesrep.i
Temp Table(s):MyTutorial/eCustomer.i,MyTutorial/eSalesrep.i
XML Schema:MyTutorial/dsCustomerSalesrep.xsd
```

To verify that your Business Entity is performing correctly and data is filled into the ProDataSet of the Business Entity you should use the **Business Entity Tester** utility we are discussing later in the tutorial.

To see the generated files in your folder just **refresh** the project in OpenEdge Architect.

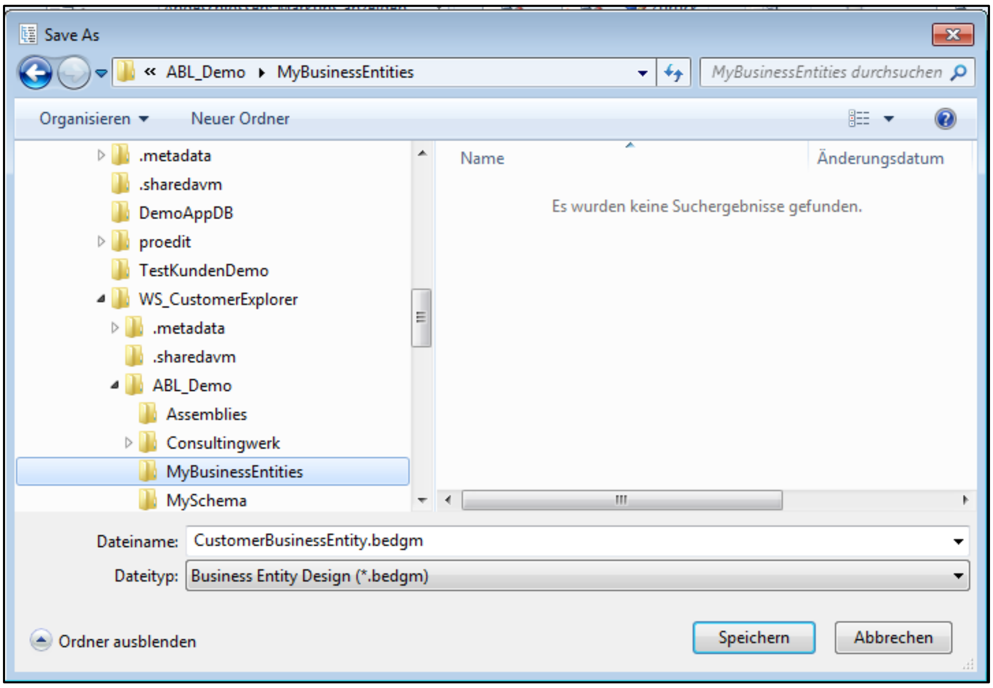
The *Resources* view should look like this:



To refresh a project, press F5 or right-click on the folder and select **Refresh** from the pop-up menu. You can also activate the auto-refresh option. For more information refer to chapter Activate Auto-Refresh Option.

## Save the Business Entity Diagram File

When you close the **Business Entity Designer** or choose **Save** from the application menu, you are asked to save the changes. By default, a file named *CustomerBusinessEntity.bedgm* file is created in the *MyBusinessEntities* directory (at the same location as the Business Entity source code would be generated). If you prefer, you can save the file to a directory of your choice.

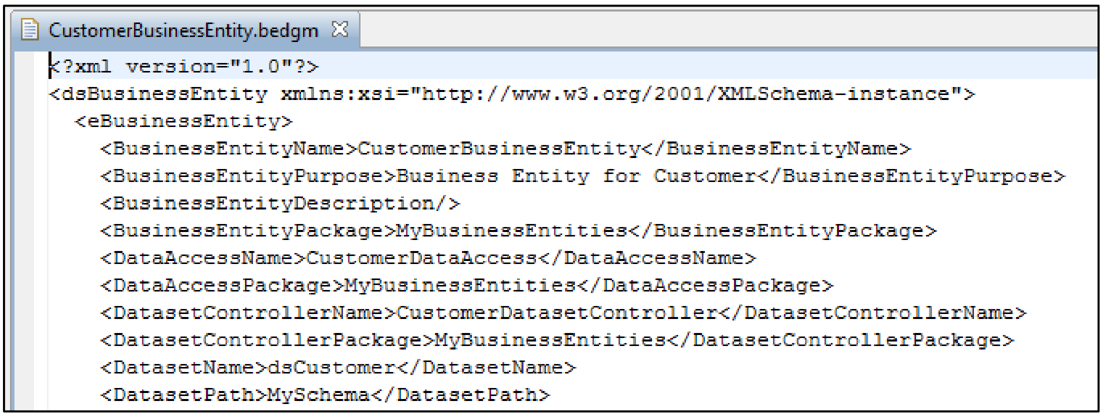


Es wurden keine Suchergebnisse gefunden.

Dateiname: CustomerBusinessEntity.bedgm

Dateityp: Business Entity Design (*.bedgm)

The CustomerBusinessEntity.bedgm file is an xml file where all properties and configuration you designed are saved. Actually the .bedgm file is a serialized version of the ProDataset used by the Business Entity Designer and code generator to handle the settings at runtime. This simplifies any kind of automation around in the context of the Business Entity Designer.

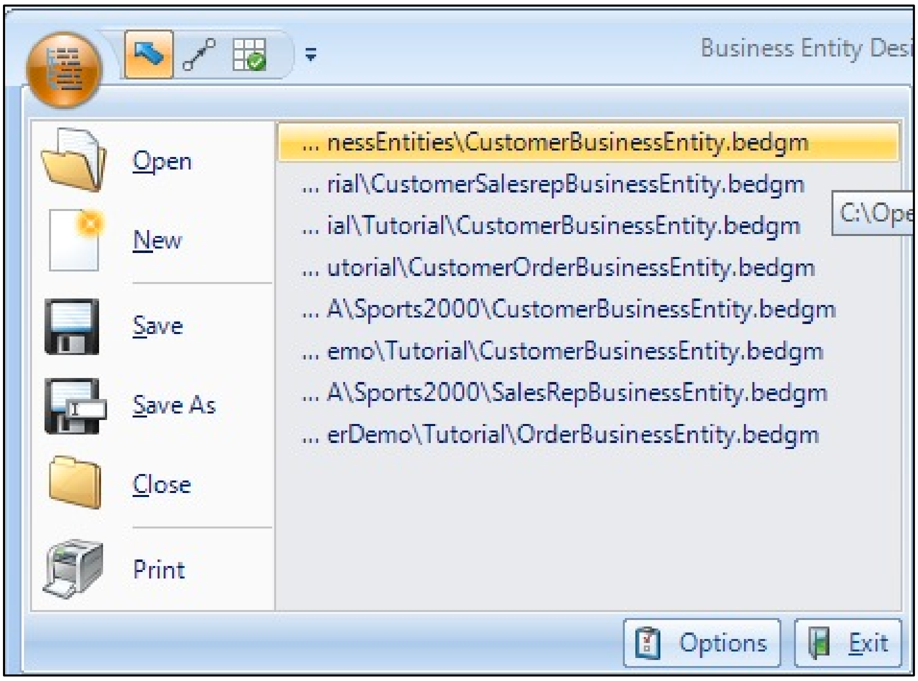


```
<?xml version="1.0"?>
<dsBusinessEntity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <eBusinessEntity>
    <BusinessEntityName>CustomerBusinessEntity</BusinessEntityName>
    <BusinessEntityPurpose>Business Entity for Customer</BusinessEntityPurpose>
    <BusinessEntityDescription/>
    <BusinessEntityPackage>MyBusinessEntities</BusinessEntityPackage>
    <DataAccessName>CustomerDataAccess</DataAccessName>
    <DataAccessPackage>MyBusinessEntities</DataAccessPackage>
    <DatasetControllerName>CustomerDatasetController</DatasetControllerName>
    <DatasetControllerPackage>MyBusinessEntities</DatasetControllerPackage>
    <DatasetName>dsCustomer</DatasetName>
    <DatasetPath>MySchema</DatasetPath>
```

## (Re)open a Business Entity Diagram File

To (re)open a *Business Entity Diagram* open the **Business Entity Designer**, select the application menu (orb) in the upper left corner and choose a *.bedgm* file from the last recently used list or click **Open** to select a file from the file system.

You can also drag-and-drop .bedgm files from the *Ressources View* of the OpenEdge Architect or the Windows Explorer to the *Design Area* of the **Business Entity Designer** to open the design.



Business Entity Desi
Open
New
Save
Save As
Close
Print
... nessEntities\CustomerBusinessEntity.bedgm
... rial\CustomerSalesrepBusinessEntity.bedgm
... ial\Tutorial\CustomerBusinessEntity.bedgm
... utorial\CustomerOrderBusinessEntity.bedgm
... A\Sports2000\CustomerBusinessEntity.bedgm
... emo\Tutorial\CustomerBusinessEntity.bedgm
... A\Sports2000\SalesRepBusinessEntity.bedgm
... erDemo\Tutorial\OrderBusinessEntity.bedgm
C:\Ope
Options
Exit

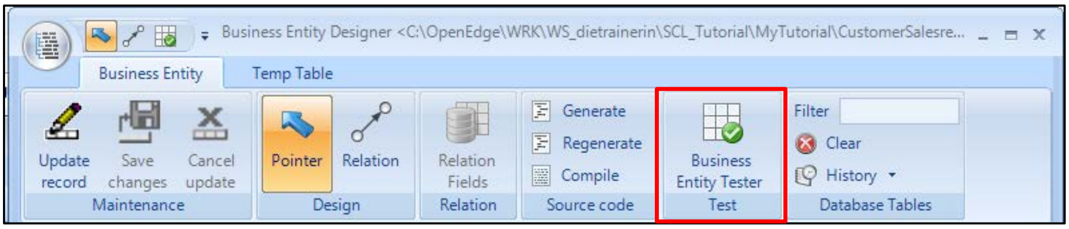
# Business Entity Tester

When implementing a new Business Entity you should verify that your Business Entity is performing correctly and data is populated in the ProDataSet of the Business Entity. The **Business Entity Tester** is a tool that can perform such tests and validate the whole process of retrieving data from the backend to the frontend, without having to build a custom user interface first for this purpose. The *Business Entity Tester* allows manual unit tests.

The **Business Entity Tester** uses the same components for reading data from the Business Entity as the user interfaces does, which we are going to build later in this tutorial (namely the *SmartBusinessEntityAdapter* and the *ServiceAdapter*).

## Starting the Business Entity Tester tool from Business Entity Designer

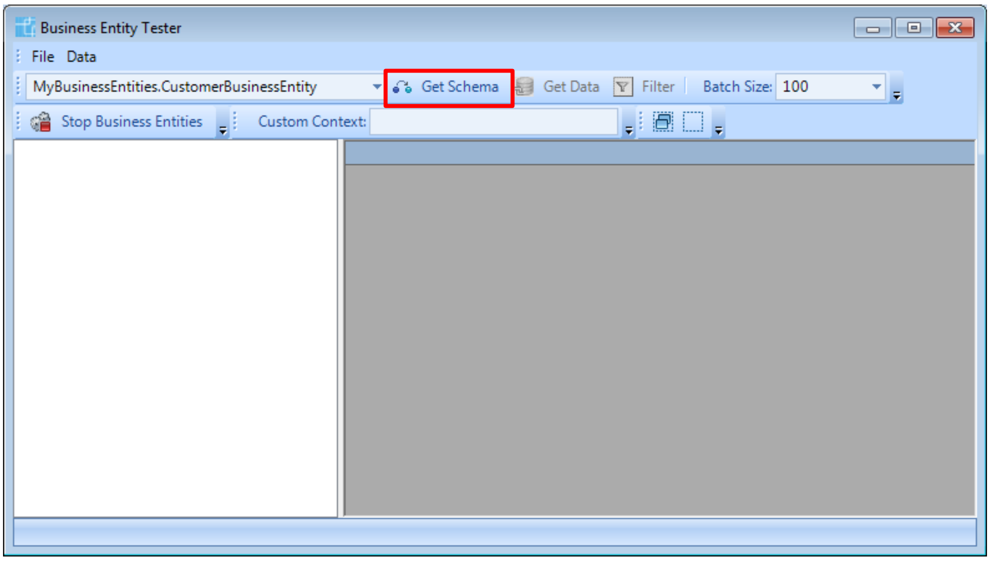
- To test your Business Entity, click the button **Business Entity Tester** in the *Test* group of the Ribbon.



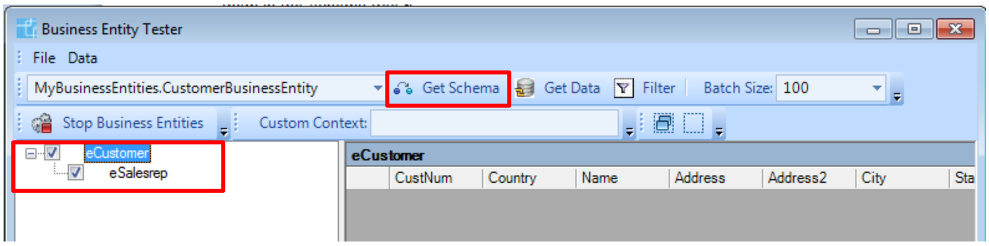
The **Business Entity Tester** tool opens.

If you want to test a Business Entity you have to provide the fully qualified class name of the Business Entity in the ComboBox (top left). When you start the **Business Entity Tester** from the **Business Entity Designer** the current Business Entity name is already filled.

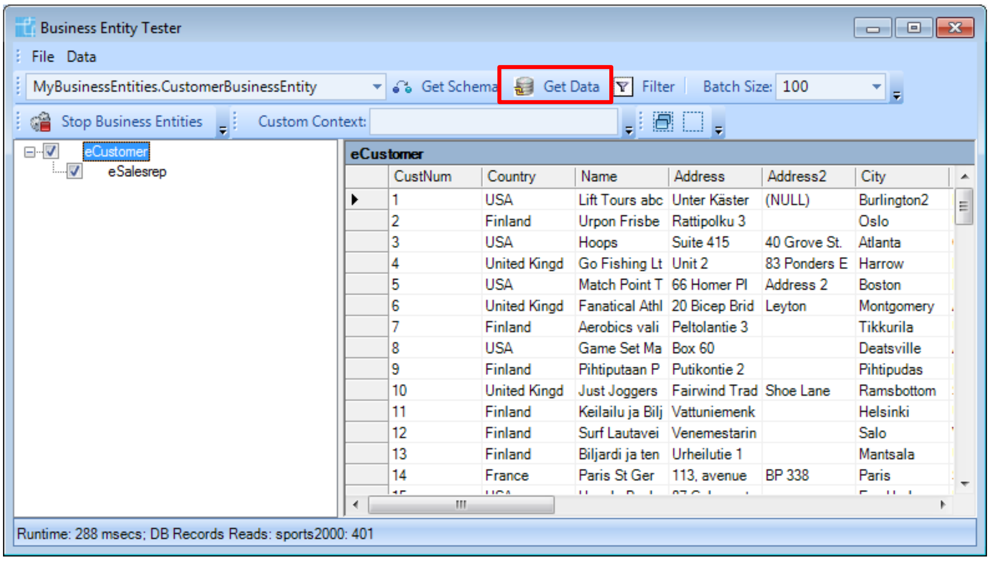
Alternatively you can start the **BusinessEntity Tester** from the OpenEdge Architect. To learn more about this option refer to chapter Starting the Business Entity Tester from OpenEdge Architect in the appendix of this tutorial.



- Select button **Get Schema** and the schema of the Business Entity will be shown in the tree-view on the left.
- Now you can select some tables to be requested from the backend by checking them in the schema tree.



- Choose button **Get Data**, which will request the data from the backend.



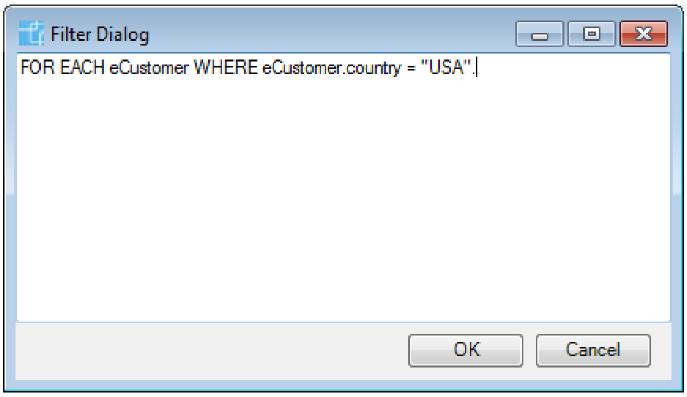
eCustomer

| CustNum | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours abc | Unter Käster | (NULL) | Burlington2 |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 4 | United Kingd | Go Fishing Lt | Unit 2 | 83 Ponders E | Harrow |
| 5 | USA | Match Point T | 66 Homer Pl | Address 2 | Boston |
| 6 | United Kingd | Fanatical Athl | 20 Bicep Brid | Leyton | Montgomery |
| 7 | Finland | Aerobics vali | Peltolantie 3 | | Tikkurila |
| 8 | USA | Game Set Ma | Box 60 | | Deatsville |
| 9 | Finland | Pihtiputaan P | Putikontie 2 | | Pihtipudas |
| 10 | United Kingd | Just Joggers | Fairwind Trad | Shoe Lane | Ramsbottom |
| 11 | Finland | Keilailu ja Bilj | Vattuniemenk | | Helsinki |
| 12 | Finland | Surf Lautavei | Venemestarin | | Salo |
| 13 | Finland | Biljardi ja ten | Urheilutie 1 | | Mantsala |
| 14 | France | Paris St Ger | 113, avenue | BP 338 | Paris |

Runtime: 288 msecs; DB Records Reads: sports2000: 401

It is also possible to filter the results passed back to the client on the backend.

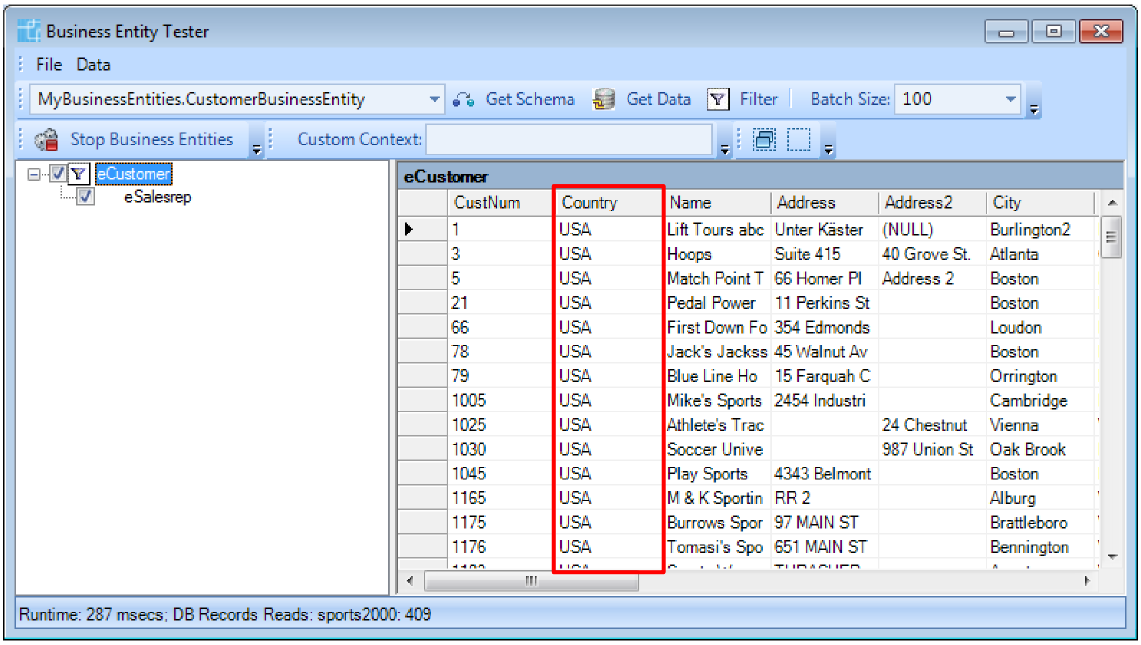
- To do so you have to choose button **Filter** for every table selected to be retrieved from the backend and provide a customized query.



FOR EACH eCustomer WHERE eCustomer.country = "USA".

The Business Entity query strings are expressed against the Temp-Table definitions. The *DataAccess* object contains logic to translate this query string into the actual database query string. Business Entities may abstract the database schema from the client (the *ProDataSet* schema). Field names may be changed to provide meaningful names and tables may be normalized or denormalized. In these cases the client should not know anything about the database schema and thus specifying the query strings against the Temp-Table s provides better decoupling between the user interface (consumer) and the backend (provider). However the query string will be translated into a database query string by the *DataAccess* object so that this object is the only object that needs to know about the actual mapping between the Temp-Table s and the database tables.

- Press button **Get Data** again. The result should now look like this:



MyBusinessEntities.CustomerBusinessEntity

Batch Size: 100

Custom Context:

eCustomer

| CustNum | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours abc | Unter Käster | (NULL) | Burlington2 |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 5 | USA | Match Point T | 66 Homer Pl | Address 2 | Boston |
| 21 | USA | Pedal Power | 11 Perkins St | | Boston |
| 66 | USA | First Down Fo | 354 Edmonds | | Loudon |
| 78 | USA | Jack's Jackss | 45 Walnut Av | | Boston |
| 79 | USA | Blue Line Ho | 15 Farquah C | | Orrington |
| 1005 | USA | Mike's Sports | 2454 Industri | | Cambridge |
| 1025 | USA | Athlete's Trac | | 24 Chestnut | Vienna |
| 1030 | USA | Soccer Unive | | 987 Union St | Oak Brook |
| 1045 | USA | Play Sports | 4343 Belmont | | Boston |
| 1165 | USA | M & K Sportin | RR 2 | | Alburg |
| 1175 | USA | Burrows Spor | 97 MAIN ST | | Brattleboro |
| 1176 | USA | Tomasi's Spo | 651 MAIN ST | | Bennington |

Runtime: 287 msecs; DB Records Reads: sports2000: 409

As you can see, only customers from the “USA” are available in the Grid.

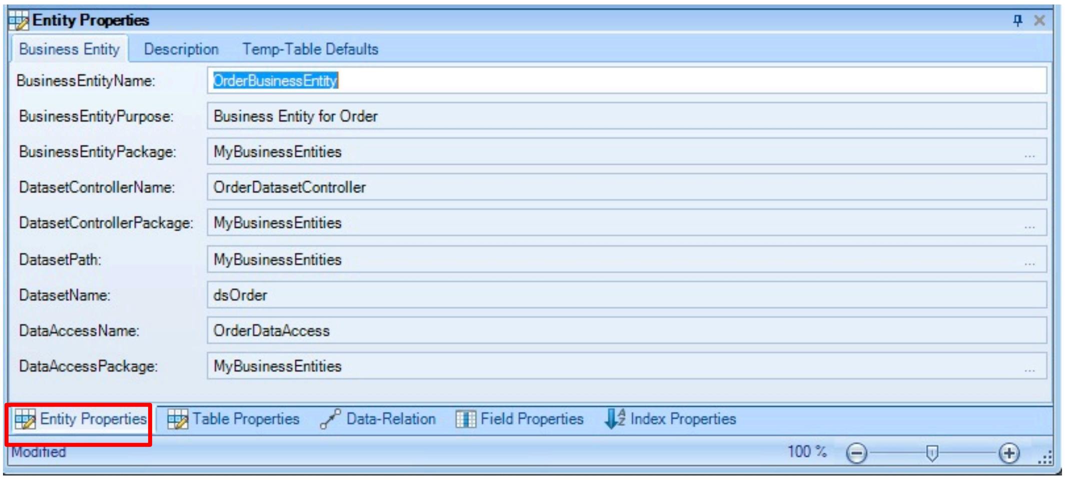
The applied filter can be removed:

- by using the context menu entry **Remove Filter** on the noted in the tree
- specifying an empty filter using the filter dialog
- by deselecting the table.

Test your Business Entity *CustomerBusinessEntity* and close the **Business Entity Tester**.

## Create a new Business Entity for Order-Customer-OrderLine-Item

In order to show dependent data in the sample application for order header and order details for customer records you need a second Business Entity. Create a new Business Entity **OrderBusinessEntity** for *Order*, *Customer*, *OrderLine* and *Item* following the naming conventions below.



BusinessEntityName: OrderBusinessEntity

BusinessEntityPurpose: Business Entity for Order

BusinessEntityPackage: MyBusinessEntities

DatasetControllerName: OrderDatasetController

DatasetControllerPackage: MyBusinessEntities

DatasetPath: MyBusinessEntities

DatasetName: dsOrder

DataAccessName: OrderDataAccess

DataAccessPackage: MyBusinessEntities

### Summary for OrderBusinessEntity

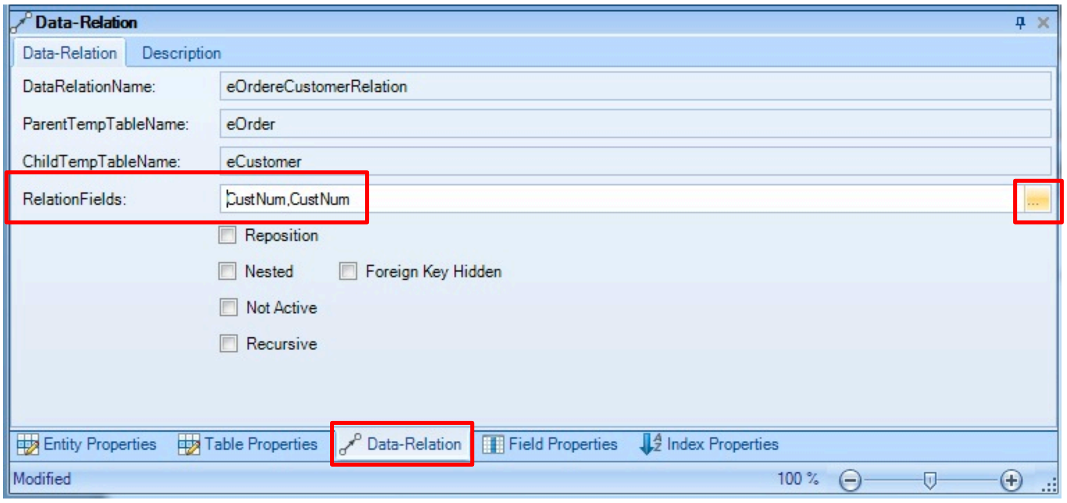
| Field | Description |
|---|---|
| BusinessEntityName | OrderBusinessEntity |
| BusinessEntityPurpose | Business Entity for Order |
| BusinessEntityPackage | MyBusinessEntities |
| DatasetControllerName | OrderDatasetController |
| DatasetControllerPackage | MyBusinessEntities |
| DatasetPath | MyBusinessEntities |
| DatasetName | dsOrder |
| DataAccessName | OrderDataAccess |
| DataAccessPackage | MyBusinessEntities |

Create the following **Temp-Table s** and the appropriate **Relations**.

| Table | Relation | Relation-Parent | Relation-Child |
|---|---|---|---|
| Order | eOrder/ecustomer | eOrder.custnum | eCustomer.custnum |
| Customer | | | |
| OrderLine | eOrder/eOrderLine | eOrder.linenum | eOrderLine.linenum |
| Item | eOrderLine/eItem | eOrderLine.itemnum | eItem.itemnum |

You should always verify the relations the *Business Entity Designer* creates for you!

- Change the *eOrdereCustomerRelation* fields to **Custnum,Custnum**.



Data-Relation

Data-Relation | Description

DataRelationName: eOrdereCustomerRelation

ParentTempTableName: eOrder

ChildTempTableName: eCustomer

RelationFields: CustNum,CustNum

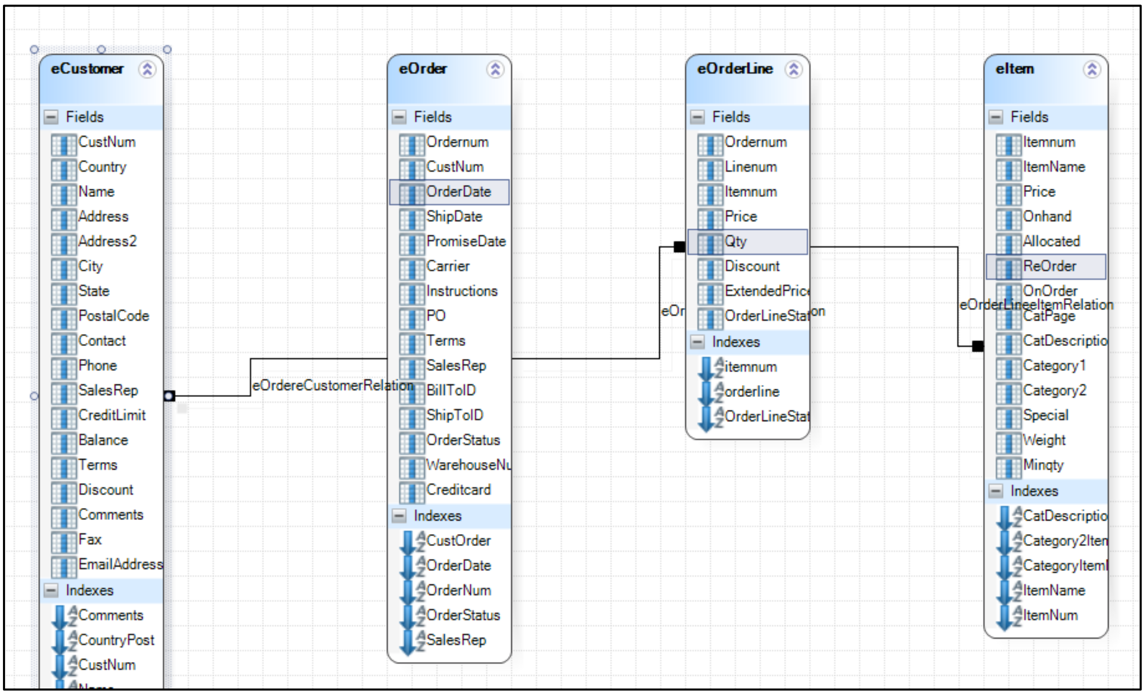
□ Reposition

□ Nested □ Foreign Key Hidden

□ Not Active

□ Recursive

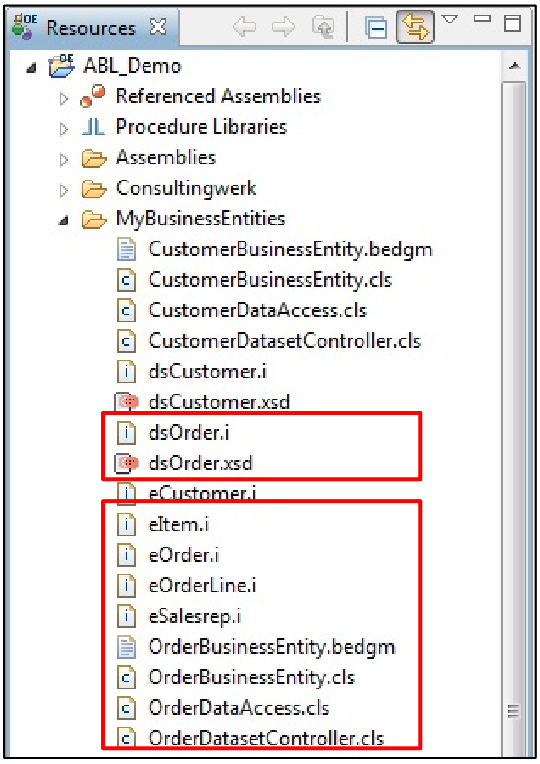
The *design canvas* of the *Business Entity Designer* should look like this.



- **Generate** and **Compile** the new Business Entity and **Refresh** your project folder in OpenEdge Architect.

The files have been created!

The *Resources* view should look like this:



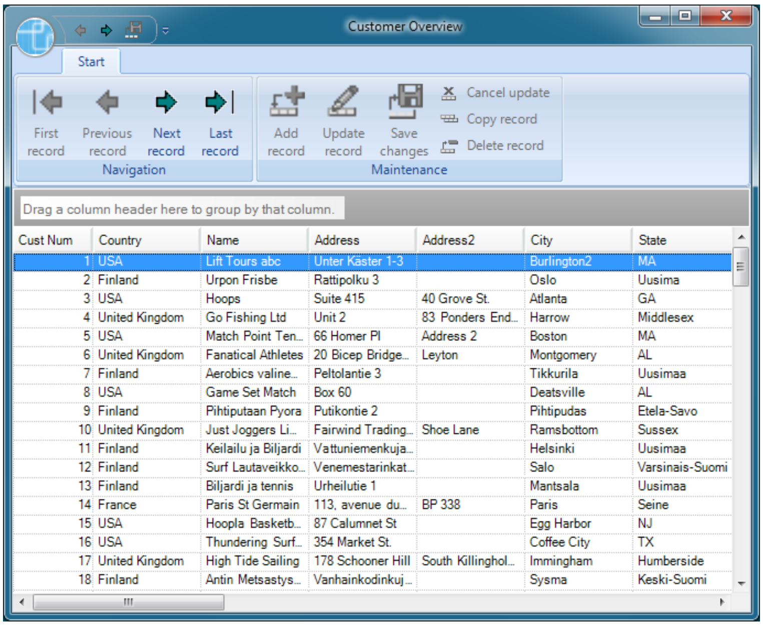
- **Test** your Business Entity using the Business Entity Tester.

# Building the sample application

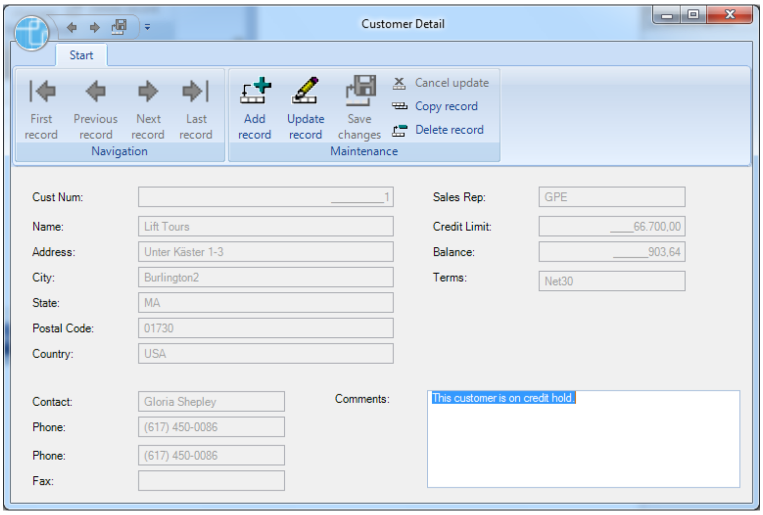
## Customer Overview and Customer Detail

After you have done all preliminary work by designing the backend so far, you are ready to build the user interface now where most of the functionality of the OERA implementation will be used, e.g. ProDataSets, BindingSources, DataSetAdapater.

At the end of this chapter your application contains two forms. The first one, **Customer Overview** shows all customer records in a browser Control. From a ribbon you can navigate but not update. The second Form, **Customer Detail** shows detail information about the selected customer record in a viewer Control. This Form will have update functionality.



| Cust Num | Country | Name | Address | Address2 | City | State |
|---|---|---|---|---|---|---|
| 1 | USA | Lift Tours abc | Unter Käster 1-3 | | Burlington2 | MA |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo | Uusima |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta | GA |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow | Middlesex |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston | MA |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery | AL |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila | Uusimaa |
| 8 | USA | Game Set Match | Box 60 | | Deatsville | AL |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas | Etela-Savo |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom | Sussex |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki | Uusimaa |
| 12 | Finland | Surf Lautaveikko... | Venemestarinkat... | | Salo | Varsinais-Suomi |
| 13 | Finland | Biljardi ja tennis | Urheilutie 1 | | Mantsala | Uusimaa |
| 14 | France | Paris St Germain | 113, avenue du... | BP 338 | Paris | Seine |
| 15 | USA | Hoopla Basketb... | 87 Calumnet St | | Egg Harbor | NJ |
| 16 | USA | Thundering Surf... | 354 Market St. | | Coffee City | TX |
| 17 | United Kingdom | High Tide Sailing | 178 Schooner Hill | South Killinghol... | Immingham | Humberside |
| 18 | Finland | Antin Metsastys... | Vanhainkodinkuj... | | Sysma | Keski-Suomi |



Cust Num: 1
Name: Lift Tours
Address: Unter Käster 1-3
City: Burlington2
State: MA
Postal Code: 01730
Country: USA

Sales Rep: GPE
Credit Limit: 66.700,00
Balance: 903,64
Terms: Net30

Contact: Gloria Shepley
Phone: (617) 450-0086
Phone: (617) 450-0086
Fax:

Comments: This customer is on credit hold.

Follow this roadmap to build the sample application step by step:

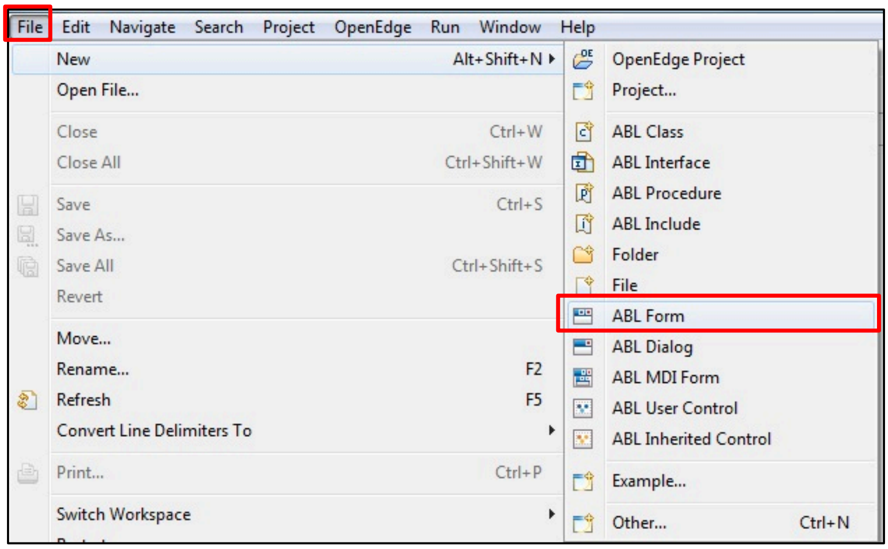
| Step | To Do | Function |
|---|---|---|
| 1 | Create a new **Form** *CustomerForm* | Container object |
| 2 | Add a Smart**ToolbarControlle**r | Toolbar for navigation purposes |
| 3 | Add a SmartBusiness**EntityAdapater** | DataSource object |
| 4 | Add a SmartBusinessEntity**BindingSource** | Binds the DataSource object to the visualization object. |
| 5 | Add a SmartData**Browser** | Visualization of records in a browse Control |
| 6 | Specify **Links** | Communication between Controls to display or update data |
| 7 | Create a new **Form** *Customer Detail* | New container object for dependent data like customer detail, order and orderlines |
| 8 | Insert a **Tab Control** in a Form | Create a new Form *Customer Detail* and insert a Tab Control |
| 9 | Create an **User Control** *CustomerDetailViewer* | Visualization of detail information in a viewer |
| 10 | Insert an **User Control** on a tab page | Insert the User Control *CustomerDetailViewer* to the first tab page |
| 11 | Specify **Links** for Form *Customer Detail* | Communication between to forms |
| 12 | Open Form *Customer Detail* from *Customer Overview* | Define a **double-click event** for Form *Customer Overview* that opens the *Customer Detail* Form |

This next step is optional but helps you organize your files. Separating the user interface will help you in the future when deploying you application distributed on on AppServer and a client (optional).

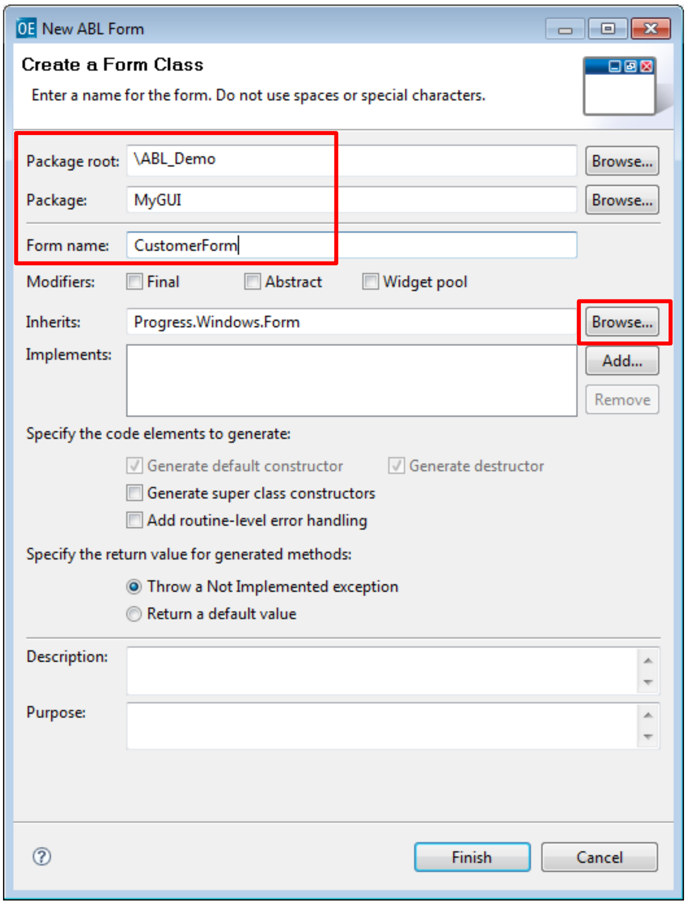
- Create a new folder **MyGUI** to store all GUI related files.
- Open the *Visual Designer Perspective* by selecting **Window > Open Perspective > OpenEdge Visual Designer** from the menu.

## 1. Create a new Form *Customer Overwiew*

- Create a new ABL Form *CustomerForm.cls* from the menu **File > New > ABL Form**.



The **New ABL Form** dialog opens.



Create a Form Class

Enter a name for the form. Do not use spaces or special characters.

Package root: \ABL_Demo

Package: MyGUI

Form name: CustomerForm

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Progress.Windows.Form

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

□ Add routine-level error handling

Specify the return value for generated methods:

● Throw a Not Implemented exception

○ Return a default value

Description:

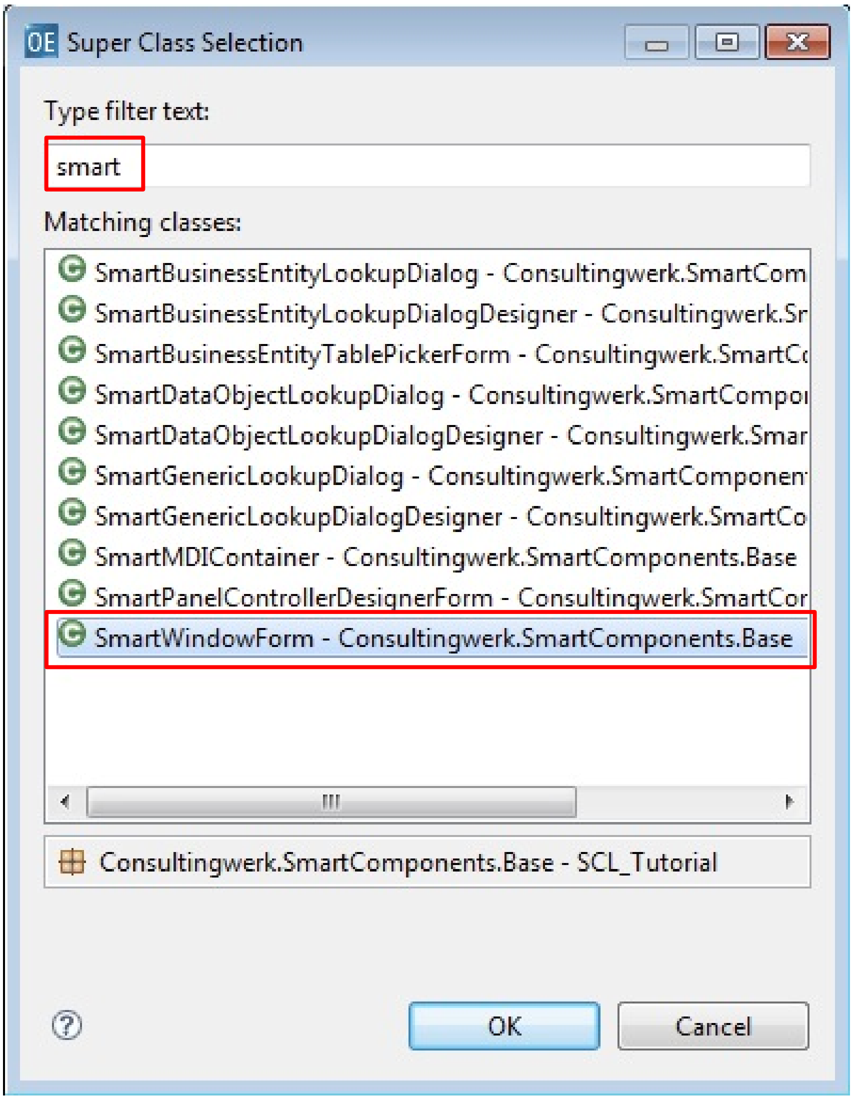
Purpose:

By default the new Form inherits from *Progress.Windows.Form*. You should change this to a different foundation class provided as part of the *SmartComponent Library*.

- Click the **Browse** button to change the Inherits class to Consultingwerk.SmartComponents.Base.SmartWindowFom.

The **Super Class Selection** dialog opens.

- To preselect all smart classes, type in *smart* in the **filter text** fill-in.
- Select class *SmartWindowForm – Consultingwerk.Smart.Components.Base.SmartWindowForm* and click **OK**.



OE Super Class Selection

Type filter text:

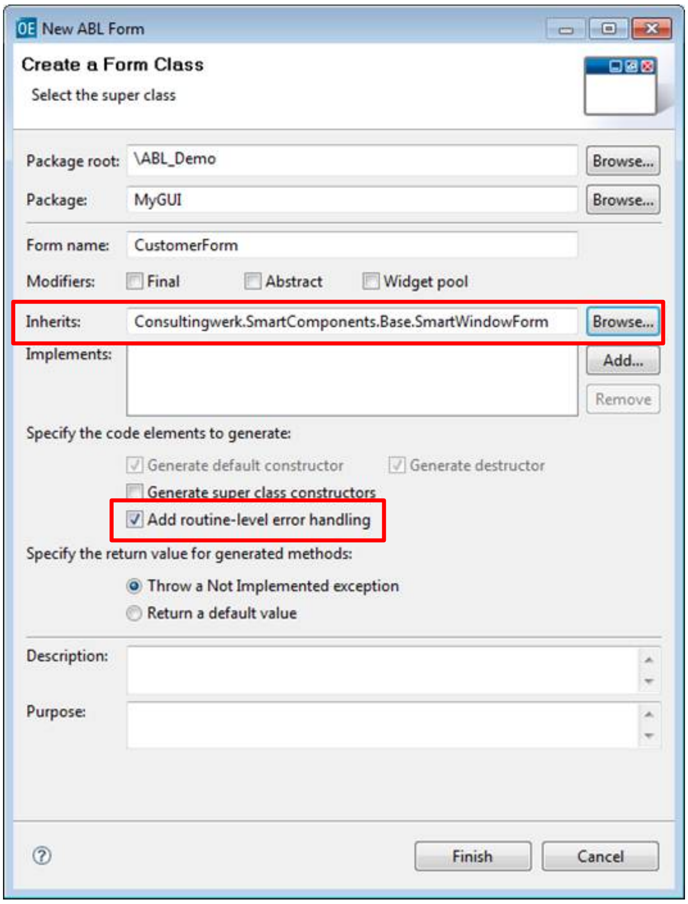
smart

Matching classes:

SmartBusinessEntityLookupDialog - Consultingwerk.SmartCom
SmartBusinessEntityLookupDialogDesigner - Consultingwerk.Sr
SmartBusinessEntityTablePickerForm - Consultingwerk.SmartC
SmartDataObjectLookupDialog - Consultingwerk.SmartCompo
SmartDataObjectLookupDialogDesigner - Consultingwerk.Smar
SmartGenericLookupDialog - Consultingwerk.SmartComponen
SmartGenericLookupDialogDesigner - Consultingwerk.SmartCo
SmartMDIContainer - Consultingwerk.SmartComponents.Base
SmartPanelControllerDesignerForm - Consultingwerk.SmartCor
SmartWindowForm - Consultingwerk.SmartComponents.Base

Consultingwerk.SmartComponents.Base - SCL_Tutorial

The result should look like this.



New ABL Form

Create a Form Class

Select the super class

Package root: \ABL_Demo

Package: MyGUI

Form name: CustomerForm

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Consultingwerk.SmartComponents.Base.SmartWindowForm

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

☑ Add routine-level error handling

Specify the return value for generated methods:

● Throw a Not Implemented exception

○ Return a default value

Description:

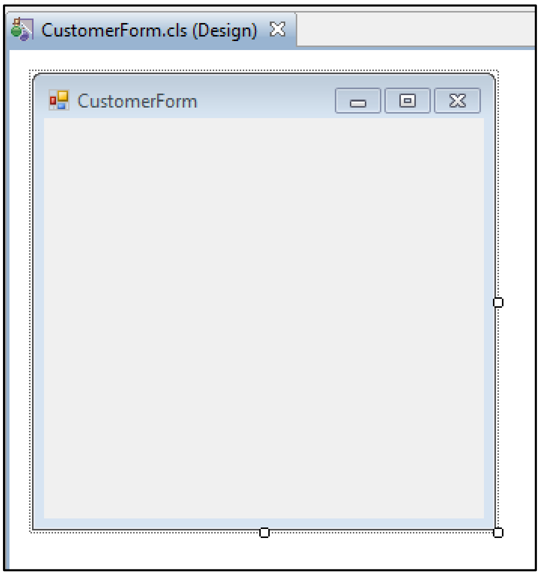
Purpose:

The *SmartWindowForm* used as a base Form provides standard functionality like optionally storing and restoring the Window position and size in the registry (or a different storage) or prompting the user to save changes when the window is getting closed with pending changes.

We highly recommend the ROUTINE-LEVEL error handling for the *SmartComponent Library* (and all new OO code).

- To do so check **Add routine-level error handling**.
- Click **Finish** to close the **New ABL Form** dialog.

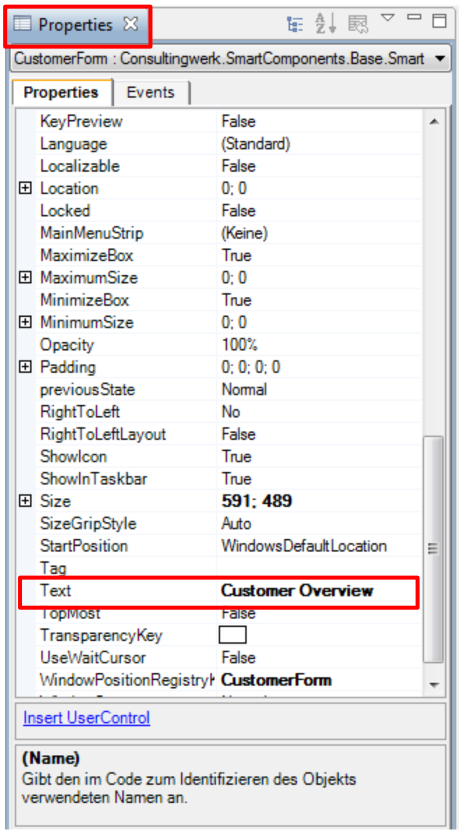
The new Form has been created and you'll be presented the Visual Designer design surface hosting the empty Form object.



CustomerForm.cls (Design)
CustomerForm

In the next step you’re going to change the title of the Form.

- To change the title use the *Properties* view and change the **Text** property from *CustomerForm* to **Customer Overview** or anything meaningful to the end user.



| | |
|---|---|
| KeyPreview | False |
| Language | (Standard) |
| Localizable | False |
| ⊞ Location | 0; 0 |
| Locked | False |
| MainMenuStrip | (Keine) |
| MaximizeBox | True |
| ⊞ MaximumSize | 0; 0 |
| MinimizeBox | True |
| ⊞ MinimumSize | 0; 0 |
| Opacity | 100% |
| ⊞ Padding | 0; 0; 0; 0 |
| previousState | Normal |
| RightToLeft | No |
| RightToLeftLayout | False |
| ShowIcon | True |
| ShowInTaskbar | True |
| ⊞ Size | **591; 489** |
| SizeGripStyle | Auto |
| StartPosition | WindowsDefaultLocation |
| Tag | |
| Text | **Customer Overview** |
| TopMost | False |
| TransparencyKey | □ |
| UseWaitCursor | False |
| WindowPositionRegistryK | **CustomerForm** |

**(Name)**
Gibt den im Code zum Identifizieren des Objekts verwendeten Namen an.

You can set properties by typing the new property value next to the name of the property in the property grid. In some cases you can use *Designer Verbs* (hyperlinks) to modify one or more properties. The *SmartComponent Library* provides for you a couple of these Designer Verbs which we’ll introduce on the following pages. Other useful *Designer Verbs* are provided by Microsoft or 3rd party vendors like Infragistics.

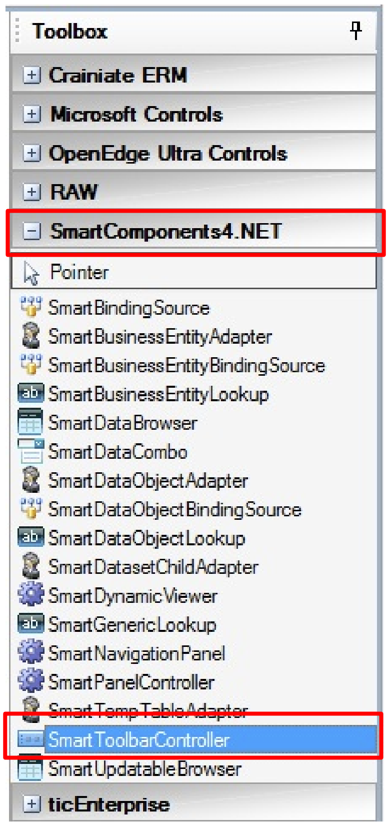
## 2. Add a SmartToolbarController

All Controls you need for the UI are available in the *Toolbox* and are managed in groups. The Control group you are going to use primarily is the **SmartComponent4.NET** group.

To add a Control to the Form, double-click the Control in the *Toolbox*. Some Controls have visual presence on the Form like buttons or fill-ins. There are also Controls that don't have any visualization and appear only in the non-visual Control tray in the lower section of the designer. For precise distinction we will reference those objects as *Components* and will use the term *Controls* only for visual elements. For more information on the distinction between *Controls* and *Components*, please reference the Microsoft Developer Network (MSDN) for more information.

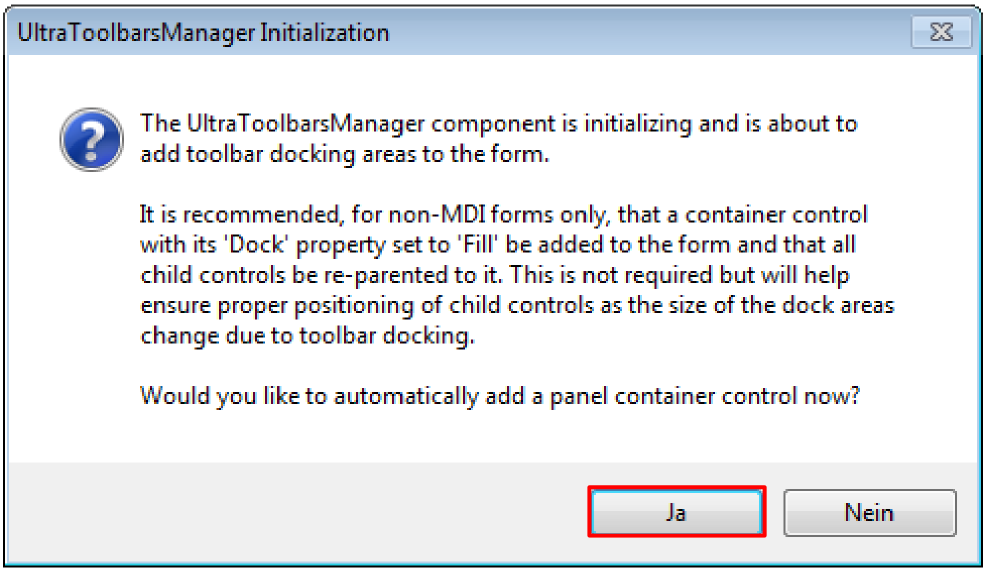
The **SmartToolbarController** offers standard tools for navigation and update functionality. It is based upon the *UltraToolbarsManager* from *Infragistics* and you can choose between a classical look and feel with toolbar and menu and the new *Ribbon* style known from Office 2007/2010 or Windows 7 applications like Paint or Notepad .

- To add a **SmartToolbarController** to the Form double-click the item in the *Toolbar*.



When an *UltraToolbarsManager* component is added to a Form, the class design time functionality (component designer) will check whether the *Docking Areas* for the toolbar and the *FillPanel* for the actual contents of the Form already exists or not. If they do not exist the Controls will be created for you and the toolbar can be docked on different borders of the Form. So the following message is expected to appear almost whenever you add an *UltraToolbarsManager* or *SmartToolbarController* to a Form. You will only choose **No** to the below message when adding the Toolbar Component to the MDI Container of your application.

- Select ***Yes*** to continue.



UltraToolbarsManager Initialization

The UltraToolbarsManager component is initializing and is about to add toolbar docking areas to the form.

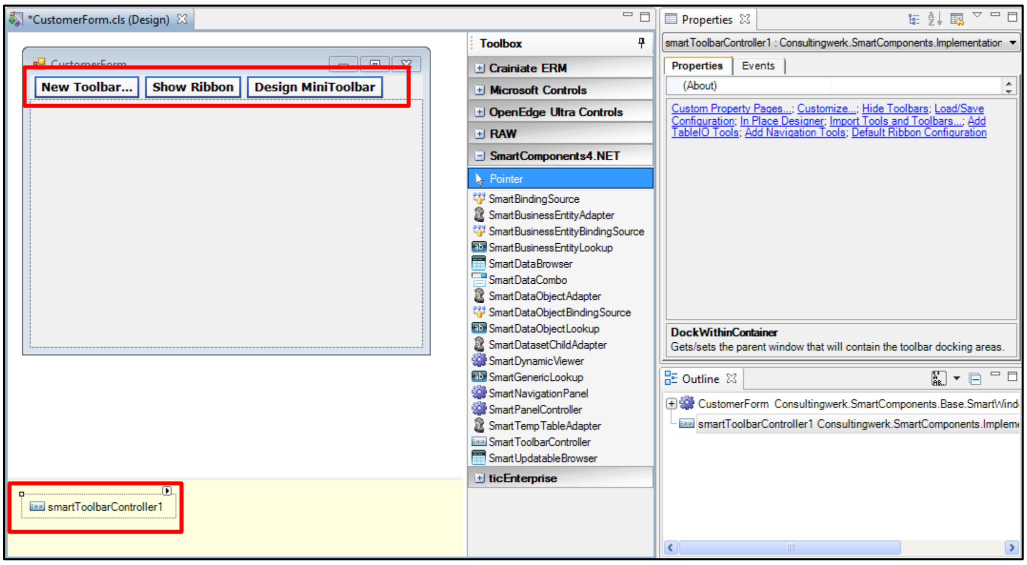
It is recommended, for non-MDI forms only, that a container control with its 'Dock' property set to 'Fill' be added to the form and that all child controls be re-parented to it. This is not required but will help ensure proper positioning of child controls as the size of the dock areas change due to toolbar docking.

Would you like to automatically add a panel container control now?

It is recommended to add the *SmartToolbarController* always as the first Control to a new Form. It is based on Infragistics *UltraToolbarsManager* component and the *UltraToolbarsManager* design time functionality will always add a number of default Controls to a Form to support the anchoring of toolbars and proper resizing (e.g. when the Ribbon get's minimized or maximized). One of the additional Controls is a *FillPanel* that should be used as the parent for every (visual) Control on the Form.

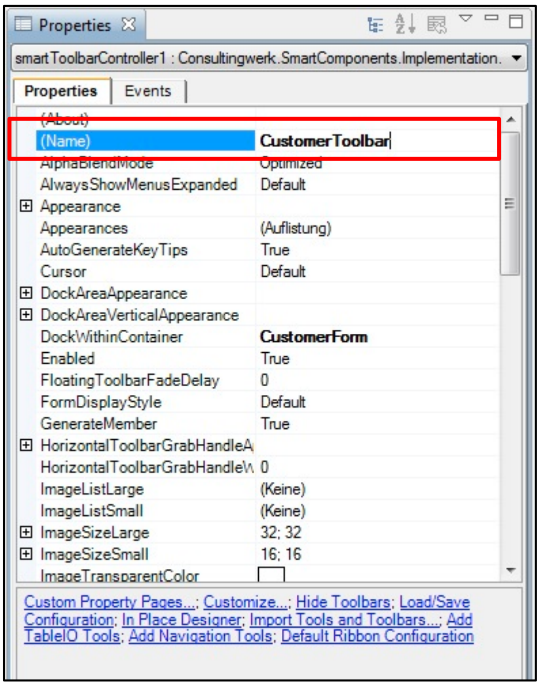
For more information, please refer to the *UltraToolbarsManager documentation* from Infragistics.

The new toolbar Component *smartToolbarController1* has been added to your Form and will appear in the non-visual Component tray in the lower section of the designer (typically with the yellow background).


*CustomerForm.cls (Design)
CustomerForm
New Toolbar...
Show Ribbon
Design MiniToolbar
smartToolbarController1
Toolbox
Crainiate ERM
Microsoft Controls
OpenEdge Ultra Controls
RAW
SmartComponents4.NET
Pointer
SmartBindingSource
SmartBusinessEntityAdapter
SmartBusinessEntityBindingSource
SmartBusinessEntityLookup
SmartDataBrowser
SmartDataCombo
SmartDataObjectAdapter
SmartDataObjectBindingSource
SmartDataObjectLookup
SmartDatasetChildAdapter
SmartDynamicViewer
SmartGenericLookup
SmartNavigationPanel
SmartPanelController
SmartTempTableAdapter
SmartToolbarController
SmartUpdatableBrowser
ticEnterprise
Properties
smartToolbarController1 : Consultingwerk.SmartComponents.Implementatior
Properties
Events
(About)
Custom Property Pages...; Customize...; Hide Toolbars; Load/Save Configuration; In Place Designer; Import Tools and Toolbars...; Add TableIO Tools; Add Navigation Tools; Default Ribbon Configuration
Outline
CustomerForm Consultingwerk.SmartComponents.Base.SmartWind
smartToolbarController1 Consultingwerk.SmartComponents.Impleme

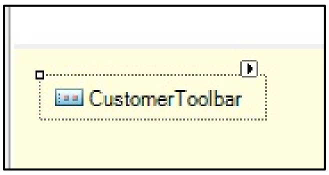
**DockWithinContainer**
Gets/sets the parent window that will contain the toolbar docking areas.

- For more readability you may change the name of the Control from smartToolbarControler1 to **CustomerToolbar** using the *Name* property.


Properties
smartToolbarController1 : Consultingwerk.SmartComponents.Implementation.
Properties
Events
Custom Property Pages...; Customize...; Hide Toolbars; Load/Save Configuration; In Place Designer; Import Tools and Toolbars...; Add TableIO Tools; Add Navigation Tools; Default Ribbon Configuration

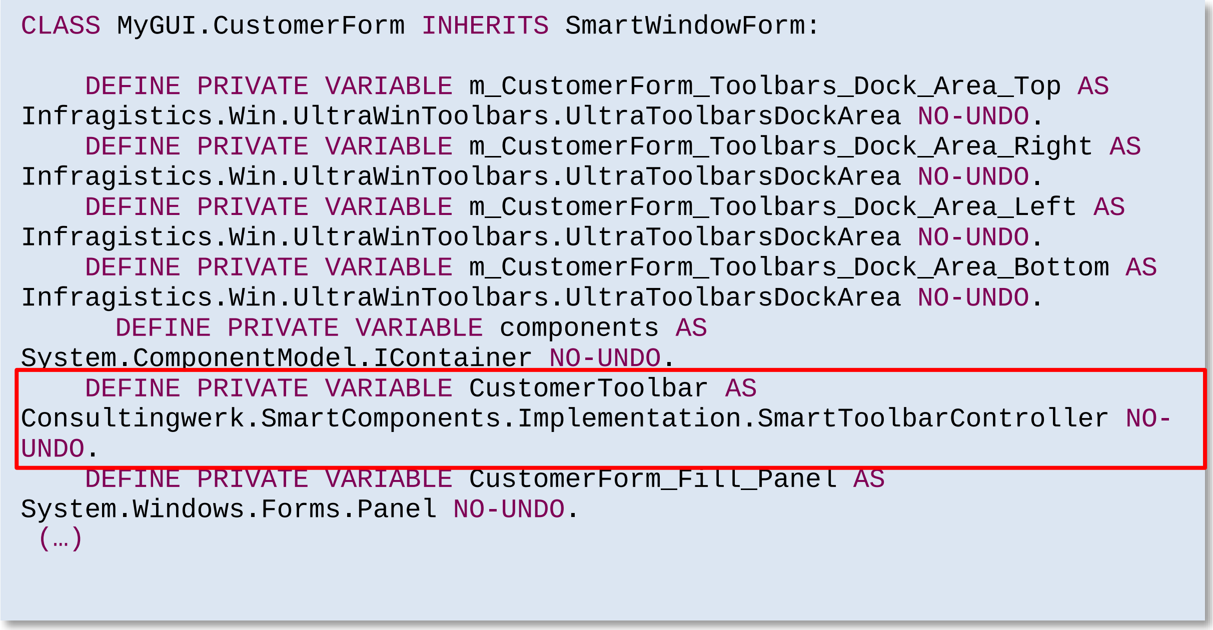
| | |
|---|---|
| (About) | |
| (Name) | CustomerToolbar |
| AlphaBlendMode | Optimized |
| AlwaysShowMenusExpanded | Default |
| ⊞ Appearance | |
| Appearances | (Auflistung) |
| AutoGenerateKeyTips | True |
| Cursor | Default |
| ⊞ DockAreaAppearance | |
| ⊞ DockAreaVerticalAppearance | |
| DockWithinContainer | **CustomerForm** |
| Enabled | True |
| FloatingToolbarFadeDelay | 0 |
| FormDisplayStyle | Default |
| GenerateMember | True |
| ⊞ HorizontalToolbarGrabHandleA | |
| HorizontalToolbarGrabHandleW | 0 |
| ImageListLarge | (Keine) |
| ImageListSmall | (Keine) |
| ⊞ ImageSizeLarge | 32; 32 |
| ⊞ ImageSizeSmall | 16; 16 |
| ImageTransparentColor | □ |

The result should look like this:



When a Control is added to a Form it creates code for the object instance of the Control or Component class and provides a unique name in the Form which is also used as the name of the reference variable that can be used to manipulate the Control or Component in the program code at runtime.

- To open the code in the ABL Editor, select **Open With > OpenEdge ABL Editor** from the context menu of the *CustomerForm*.*cls* file in the resouces view
- or press **F9** or “View Source” or the Visual Designers context menu.



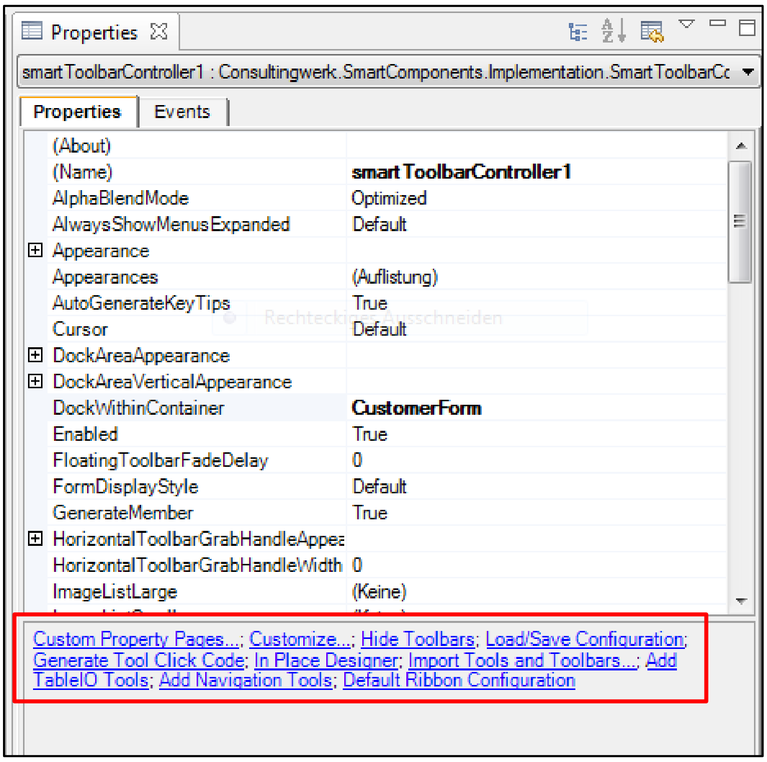
```
CLASS MyGUI.CustomerForm INHERITS SmartWindowForm:

    DEFINE PRIVATE VARIABLE m_CustomerForm_Toolbars_Dock_Area_Top AS
Infragistics.Win.UltraWinToolbars.UltraToolbarsDockArea NO-UNDO.
    DEFINE PRIVATE VARIABLE m_CustomerForm_Toolbars_Dock_Area_Right AS
Infragistics.Win.UltraWinToolbars.UltraToolbarsDockArea NO-UNDO.
    DEFINE PRIVATE VARIABLE m_CustomerForm_Toolbars_Dock_Area_Left AS
Infragistics.Win.UltraWinToolbars.UltraToolbarsDockArea NO-UNDO.
    DEFINE PRIVATE VARIABLE m_CustomerForm_Toolbars_Dock_Area_Bottom AS
Infragistics.Win.UltraWinToolbars.UltraToolbarsDockArea NO-UNDO.
      DEFINE PRIVATE VARIABLE components AS
System.ComponentModel.IContainer NO-UNDO.
    DEFINE PRIVATE VARIABLE CustomerToolbar AS
Consultingwerk.SmartComponents.Implementation.SmartToolbarController NO-
UNDO.
    DEFINE PRIVATE VARIABLE CustomerForm_Fill_Panel AS
System.Windows.Forms.Panel NO-UNDO.
 (…)
```

## Set configurations for the SmartToolbarController

You will now modify the appearance and the functionality of *CustomerToolbar*.

- Use the **Properties** view to set properties for *CustomerToolbar*.
- Select the designer verb Default Ribbon Configuration to change the appearance of the *SmartToolbarController* from a Menu to a Ribbon.

**Designer Verb's** are accessible from the context menu of a Component or Control or are shown as hyperlinks under the property grid.



smartToolbarController1 : Consultingwerk.SmartComponents.Implementation.SmartToolbarCc

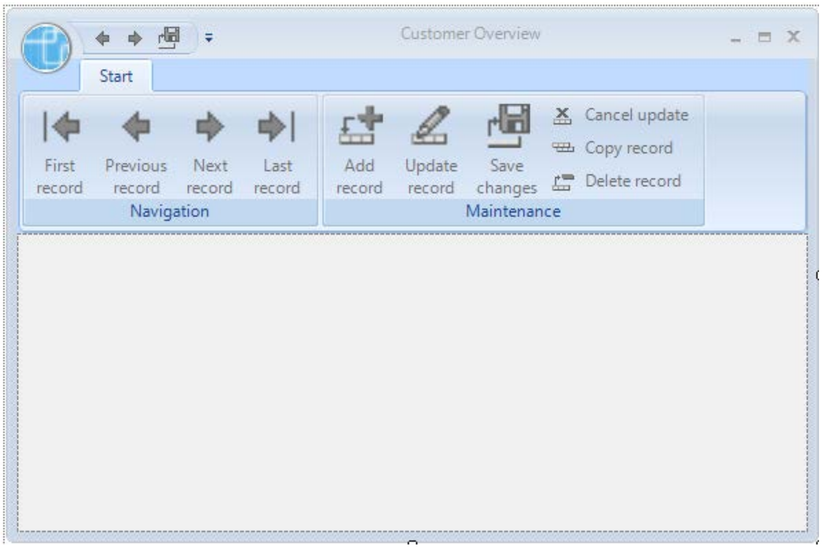
| | |
|---|---|
| (About) | |
| (Name) | **smartToolbarController1** |
| AlphaBlendMode | Optimized |
| AlwaysShowMenusExpanded | Default |
| ⊞ Appearance | |
| Appearances | (Auflistung) |
| AutoGenerateKeyTips | True |
| Cursor | Default |
| ⊞ DockAreaAppearance | |
| ⊞ DockAreaVerticalAppearance | |
| DockWithinContainer | **CustomerForm** |
| Enabled | True |
| FloatingToolbarFadeDelay | 0 |
| FormDisplayStyle | Default |
| GenerateMember | True |
| ⊞ HorizontalToolbarGrabHandleAppea | |
| HorizontalToolbarGrabHandleWidth | 0 |
| ImageListLarge | (Keine) |

Custom Property Pages...; Customize...; Hide Toolbars; Load/Save Configuration; Generate Tool Click Code; In Place Designer; Import Tools and Toolbars...; Add TableIO Tools; Add Navigation Tools; Default Ribbon Configuration

The *Default Ribbon Configuration* is a standardized Ribbon design provided by the *SmartComponent Library*. This design can be customized e.g. by dragging around the buttons or adding additional Ribbon tabs, groups or tools.

For more information about customizing the Ribbon refer to chapter Ribbon Configuration in the appendix of the tutorial.

The result should look like this:

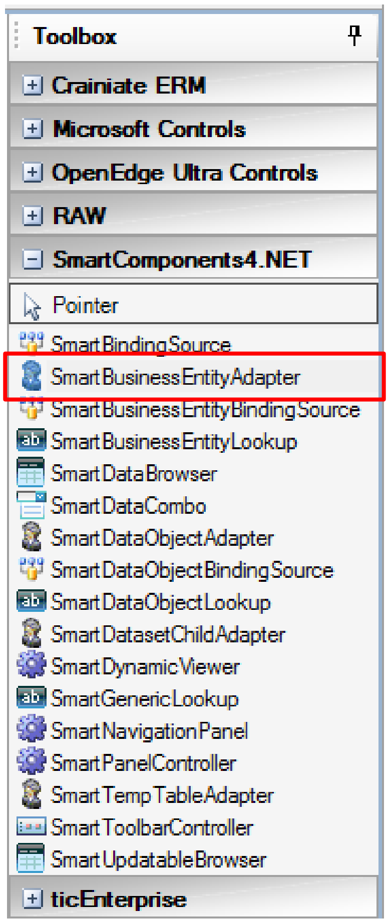

Customer Overview
Start
First record
Previous record
Next record
Last record
Navigation
Add record
Update record
Save changes
Cancel update
Copy record
Delete record
Maintenance

## 3. Add a SmartBusinessEntity Adapter

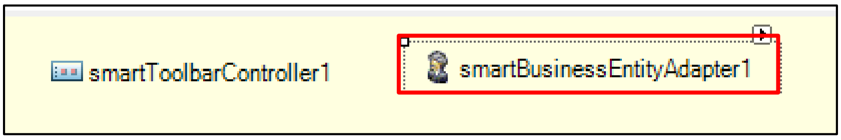
As you learned so far Business Entities are business-centric, not data-centric and do **not** have any responsibility for accessing data sources or for retrieving or updating data from a database. They are normally paired with Data Access objects (…DataAccess.cls) that manage the connection to the actual data sources. The DataAccess class was generated with the Business Entity. However the layered architecture of the SmartComponent Library and the OpenEdge Reference Architecture, makes the Business Entity the actual data provider for the front end. The fact that the Business Entity will typically delegate this functionality to the Data Access Object is encapsulated at the backend.

To use the *CustomerBusinessEntity* you created earlier in the tutorial for data access in the *CustomerForm*, you need a **SmartBusinessEntity Adapter** Component which transparently provides all the functionality to have access to the Business Entity on the backend.

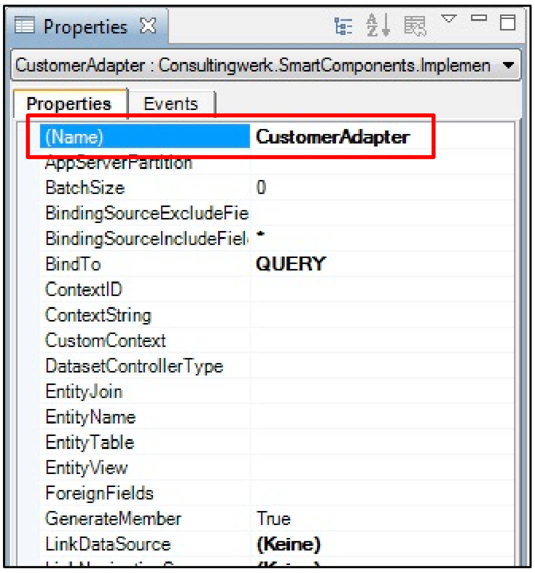
- To add a **SmartBusinessEntityAdapter** to the Form double-click on the Component in the *Toolbox*.



The new Component *smartBusinessEntityAdapter1* has been added to the non-visual section of the design canvas.



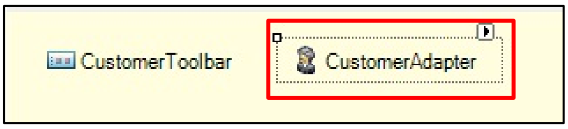
- Change Property **Name** to *CustomerAdapter*.



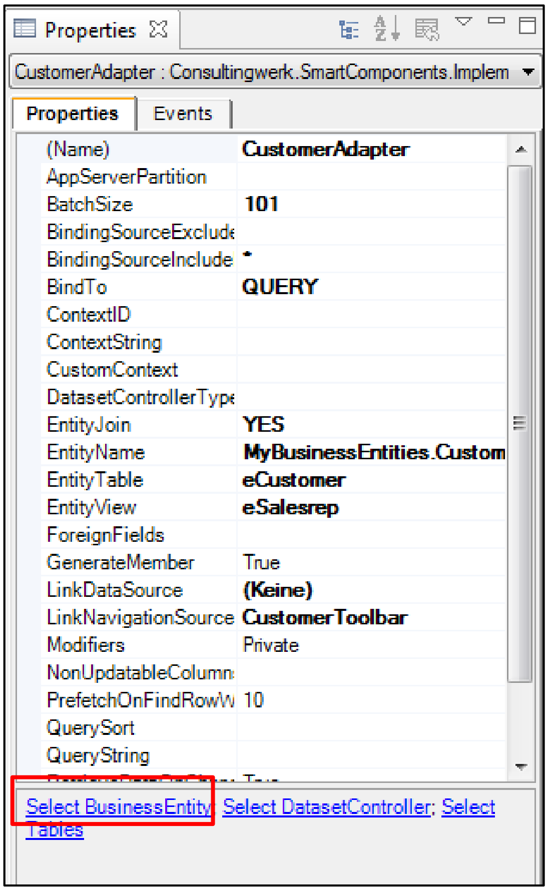
CustomerAdapter : Consultingwerk.SmartComponents.Implemen

| | |
|---|---|
| (Name) | CustomerAdapter |
| AppServerPartition | |
| BatchSize | 0 |
| BindingSourceExcludeFie | |
| BindingSourceIncludeFiel | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControllerType | |
| EntityJoin | |
| EntityName | |
| EntityTable | |
| EntityView | |
| ForeignFields | |
| GenerateMember | True |
| LinkDataSource | (Keine) |

The result should look like this:



To select the Business Entity this adapter will work with, you may hand type the Business Entities class name to Property **EntityName** in the property sheet or use the Designer Verb **Select BusinessEntity** The Designer Verb will open a dialog that lists all Business Entities found in your current project and provides a simplified way of entering the *EntityName* property in the property grid.

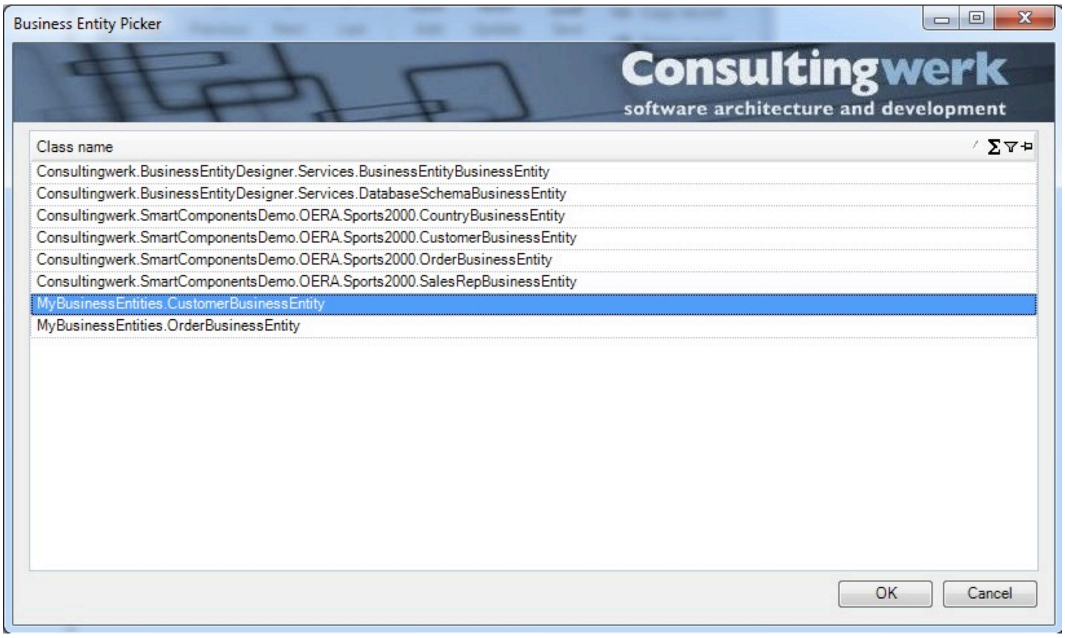


| | |
|---|---|
| (Name) | CustomerAdapter |
| AppServerPartition | |
| BatchSize | 101 |
| BindingSourceExclude | |
| BindingSourceInclude | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControllerType | |
| EntityJoin | YES |
| EntityName | MyBusinessEntities.Custom |
| EntityTable | eCustomer |
| EntityView | eSalesrep |
| ForeignFields | |
| GenerateMember | True |
| LinkDataSource | (Keine) |
| LinkNavigationSource | CustomerToolbar |
| Modifiers | Private |
| NonUpdatableColumn | |
| PrefetchOnFindRowW | 10 |
| QuerySort | |
| QueryString | |

Select BusinessEntity; Select DatasetController; Select Tables

The **BusinessEntity Picker** dialog opens.

- Choose the Business Entity *CustomerBusinessEntity* from the *MyBusinessEntities* package and click **OK**.



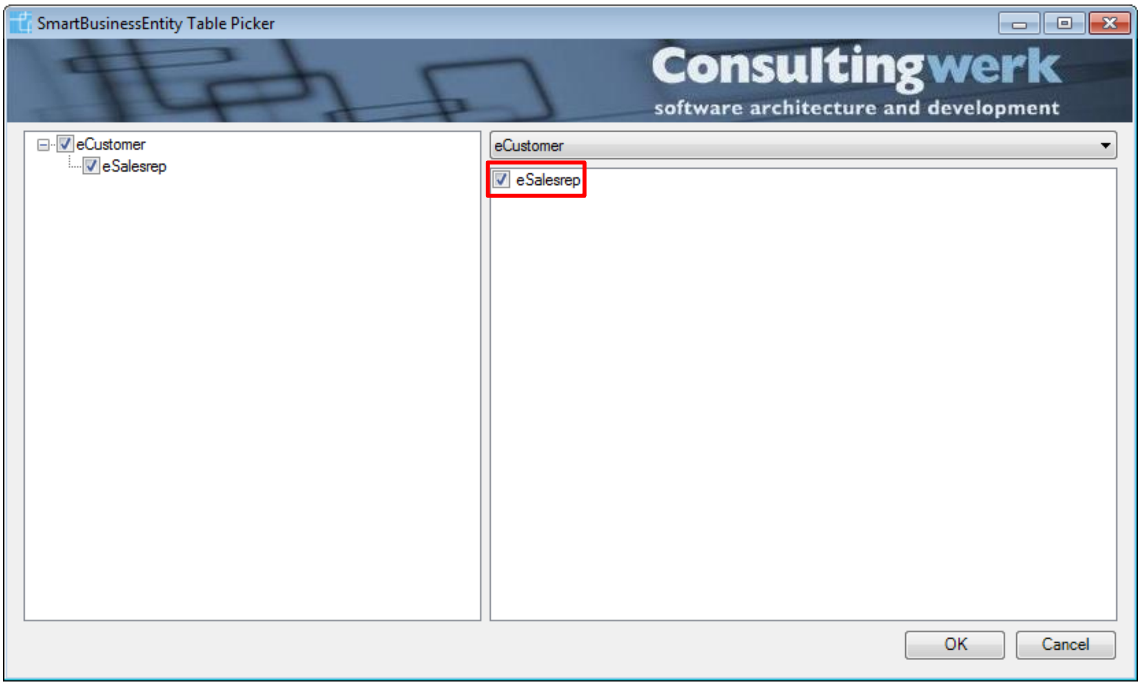
Consultingwerk.BusinessEntityDesigner.Services.BusinessEntityBusinessEntity
Consultingwerk.BusinessEntityDesigner.Services.DatabaseSchemaBusinessEntity
Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CountryBusinessEntity
Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CustomerBusinessEntity
Consultingwerk.SmartComponentsDemo.OERA.Sports2000.OrderBusinessEntity
Consultingwerk.SmartComponentsDemo.OERA.Sports2000.SalesRepBusinessEntity
MyBusinessEntities.CustomerBusinessEntity
MyBusinessEntities.OrderBusinessEntity

Next you are going to select Temp-Table s from the Business Entities ProDataSet schema. These Temp-Tables will be read from the Business Entity and navigated using the *SmartBusinessEntityAdapter* Component. You can select one or more Temp-Tables using a dialog or enter the first Temp-Table name in the *EntityTable* property and the additional Temp-Table names in the *EntityView* property.

- To select the necessary Temp-Table (s), select the designer verb **Select Table**.

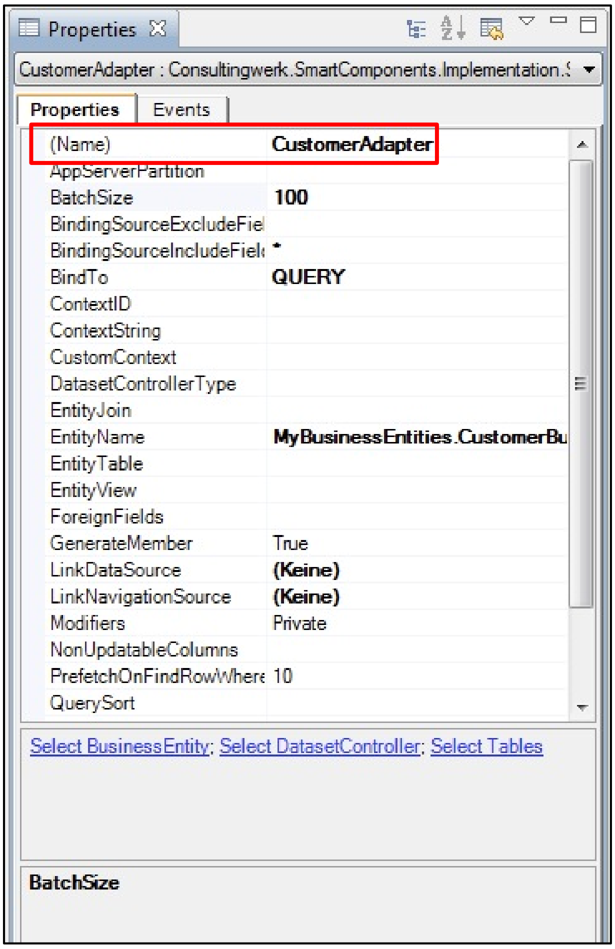
The **SmartBusinessEntity TablePicker** dialog opens.

- Check the *eCustomer* and *eSalesrep* table on the left side.
- Check *eSalesrep* on the right side; indicating that you wish to join both tables with each other and navigate the joined query result on the user interface.
- Click **OK**.


SmartBusinessEntity Table Picker
Consultingwerk
software architecture and development
☑ eCustomer
☑ eSalesrep
eCustomer
☑ eSalesrep
OK
Cancel

It is recommended to enable batching for your sample application to optimize the usage of system resources. Otherwise the Data Access Object would read all available data from the data source at one stroke which might not be an optimal use of resources and cause delays when starting the Form.

When you enable batching on your *SmartBusinessEntityAdapter* instance the data will be read in small chunks. As the user navigates further in the available data the *SmartBusinessEntityAdapter* will automatically request further chunks of data. This behavior is similar to the data handling in the classical progress browse widget. To enable batching for the *customerAdapter* set the property **BatchSize** to **100**, which is a reasonable setting.



CustomerAdapter : Consultingwerk.SmartComponents.Implementation.S

| | |
|---|---|
| (Name) | CustomerAdapter |
| AppServerPartition | |
| BatchSize | 100 |
| BindingSourceExcludeFiel | |
| BindingSourceIncludeFiel | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControllerType | |
| EntityJoin | |
| EntityName | MyBusinessEntities.CustomerBu |
| EntityTable | |
| EntityView | |
| ForeignFields | |
| GenerateMember | True |
| LinkDataSource | (Keine) |
| LinkNavigationSource | (Keine) |
| Modifiers | Private |
| NonUpdatableColumns | |
| PrefetchOnFindRowWhere | 10 |
| QuerySort | |

Select BusinessEntity; Select DatasetController; Select Tables

BatchSize

**Congratulation!** The Business Entity *CustomerBusinessEntity* is connected to your *CustomerForm*. Your user interface will now have access to the data from the database.

## 4. Add a SmartBusinessEntityBindingSource

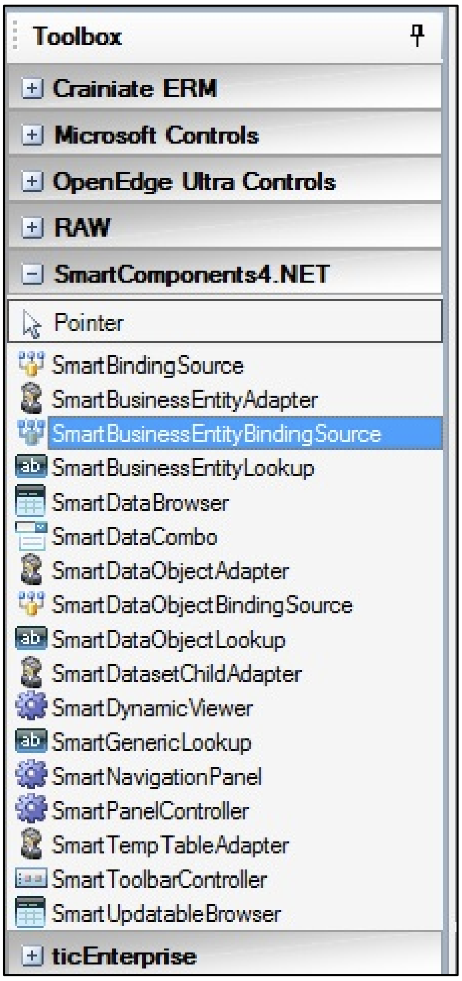
In the next chapter you are going to add a *SmartDataBrowser* Control to your UI that shows customer data in list format and allows navigation using keyboard or mouse actions. You will need the schema information of the Business Entities Dataset Temp-Table at design time schema to be able to format and order the columns of the browser Control. The *Data Adapter* you have just created in the previous chapter is **not** designed to provide schema at design time.

Consultingwerk provides the *SmartBusinessEntityBindingSource* Component for you as part of the OERA client infrastructure. It is an enhanced version of the *ProBindingSource* Component from Progress Software that makes data from an OpenEdge database such as a ProDataset, a Query or a Buffer available for binding to .NET Controls. The main purpose of the *SmartBusinessEntityBindingSource* right now is to provide schema information at design time. At runtime it will bind the Controls to the data that the *SmartBusinessEntityAdapter* has read from the backend; **it does not read or display data from the database directly**!

- To add a **SmartBusinessEntityBindingSource** to the Form double-click on the Component in the *Toolbox*.

Be careful not to select *SmartBindingSource* Control instead, which is the basic class for the specialized *SmartComponet Library BindingSources* like *BusinessEntityBindingSource*. However the *SmartBindingSource* class cannot be used to perform the following task.

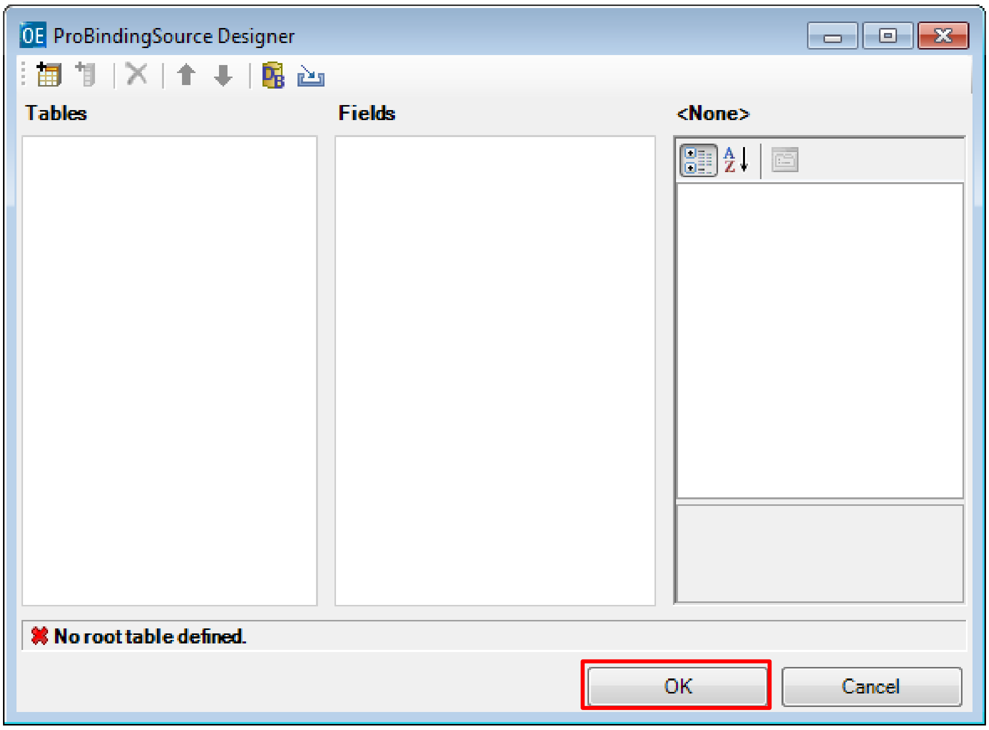
The **SmartBindingSource** can be used when the Form should read it's data directly from the database and not from an OERA backend.



After adding any of the *SmartBindingSource* Components to the design canvas the **ProBindingSource Designer** will open. It is empty because there is no schema defined so far.

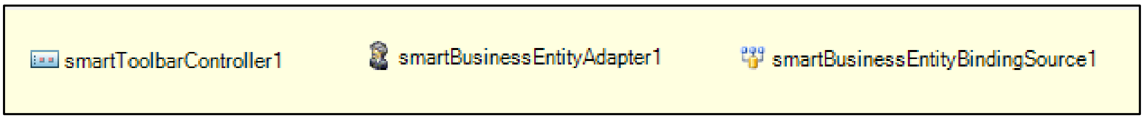
- Close the dialog.

Within the *SmartComponent Library* we will typically use more advanced versions of defining the schema information of the *BindingSource*. We will come back to this later in the tutorial.



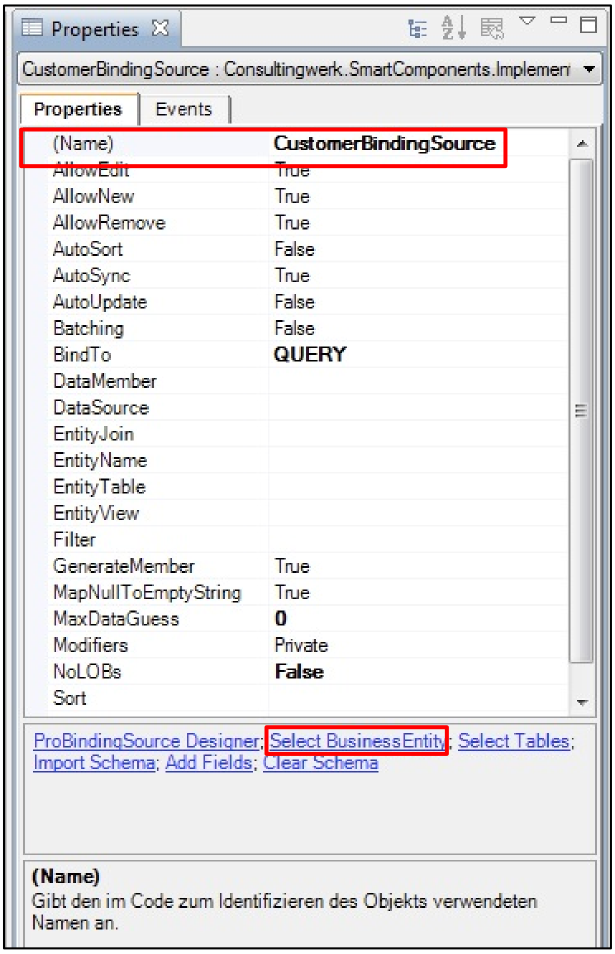
✖ No root table defined.

A new Component *smartBusinessEntityBindingSource1* has been added to the non-visual section of the designer.


smartToolbarController1
smartBusinessEntityAdapter1
smartBusinessEntityBindingSource1

With the new *smartBusinessEntityBindingSource1* Component selected:

- Change the *Name* property to **CustomerBindingSource**
- Select the designer verb **Select BusinessEntity**



CustomerBindingSource : Consultingwerk.SmartComponents.Implemen

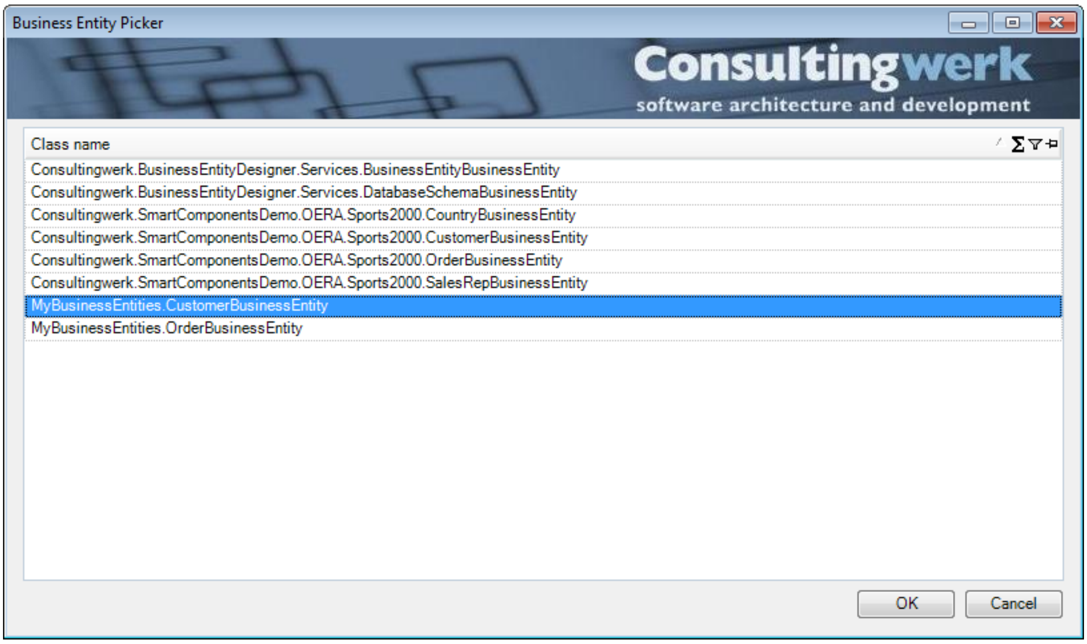
| | |
|---|---|
| (Name) | **CustomerBindingSource** |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |
| BindTo | **QUERY** |
| DataMember | |
| DataSource | |
| EntityJoin | |
| EntityName | |
| EntityTable | |
| EntityView | |
| Filter | |
| GenerateMember | True |
| MapNullToEmptyString | True |
| MaxDataGuess | **0** |
| Modifiers | Private |
| NoLOBs | **False** |
| Sort | |

ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

**(Name)**
Gibt den im Code zum Identifizieren des Objekts verwendeten Namen an.

The **Business Entity Picker** dialog opens.

- Select the appropriate BusinessEntity you used for the SmartBusinessEntityAdapter – CustomerBusinessEntity from the MyBusinessEntities package and click **OK**.

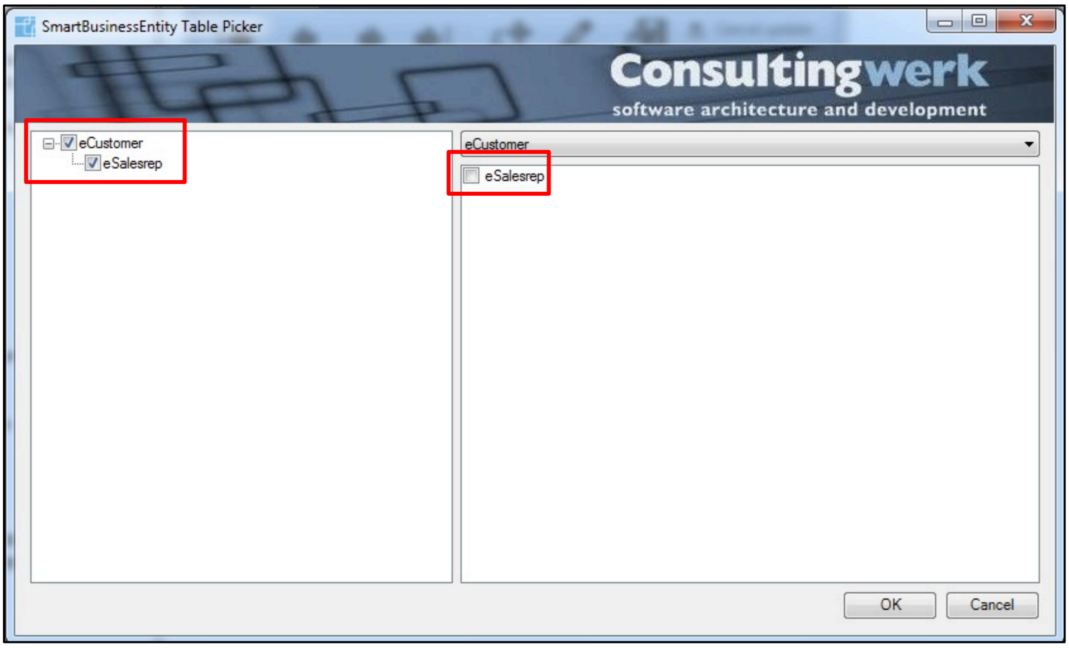


| Class name |
|---|
| Consultingwerk.BusinessEntityDesigner.Services.BusinessEntityBusinessEntity |
| Consultingwerk.BusinessEntityDesigner.Services.DatabaseSchemaBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CountryBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CustomerBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.OrderBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.SalesRepBusinessEntity |
| MyBusinessEntities.CustomerBusinessEntity |
| MyBusinessEntities.OrderBusinessEntity |

- Select the designer verb **Select Table**.

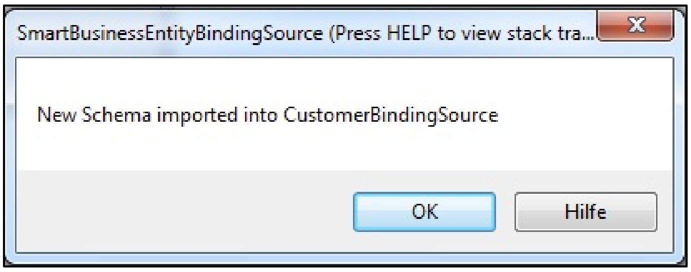
The *SmartBusinessEntity TablePicker* dialog opens.

- Select the *eCustomer* and *eSalesrep* tables on the left and then *eSalesRep* on the right and choose **OK**.


SmartBusinessEntity Table Picker
Consultingwerk
software architecture and development
☑ eCustomer
☑ eSalesrep
eCustomer
□ eSalesrep
OK
Cancel

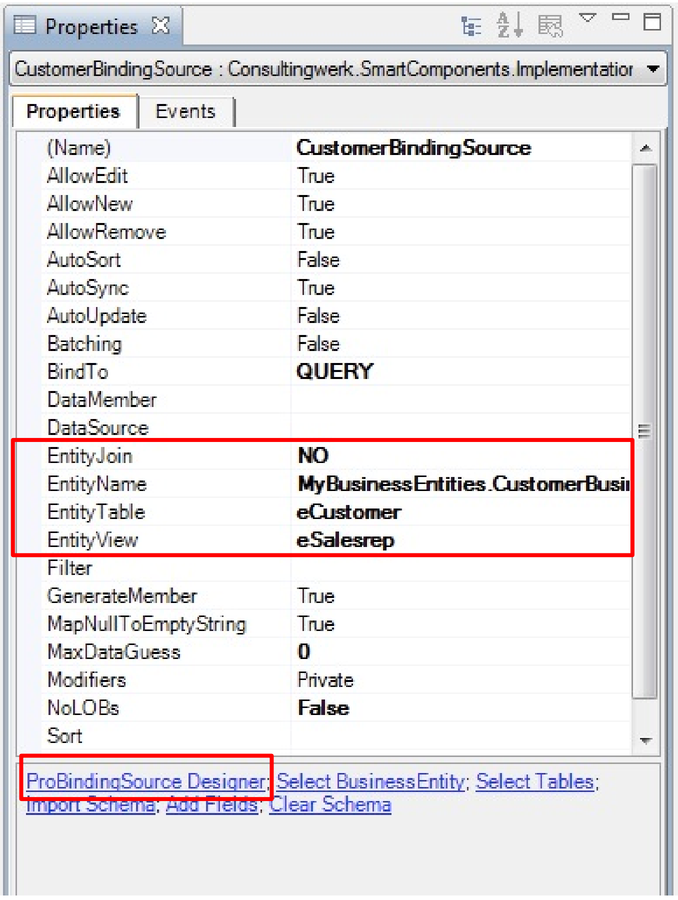
- Select the designer verb **Import schema**.

A message appears to inform you that new schema has been imported into *CustomerBindingSource.* The schema has been requested from the *BusinessEntity* and has been imported into the ProBindingSource schema information.


SmartBusinessEntityBindingSource (Press HELP to view stack tra...

New Schema imported into CustomerBindingSource

The **Properties** view of *CustomerBindingSource* should look like this:



Properties

CustomerBindingSource : Consultingwerk.SmartComponents.Implementatior

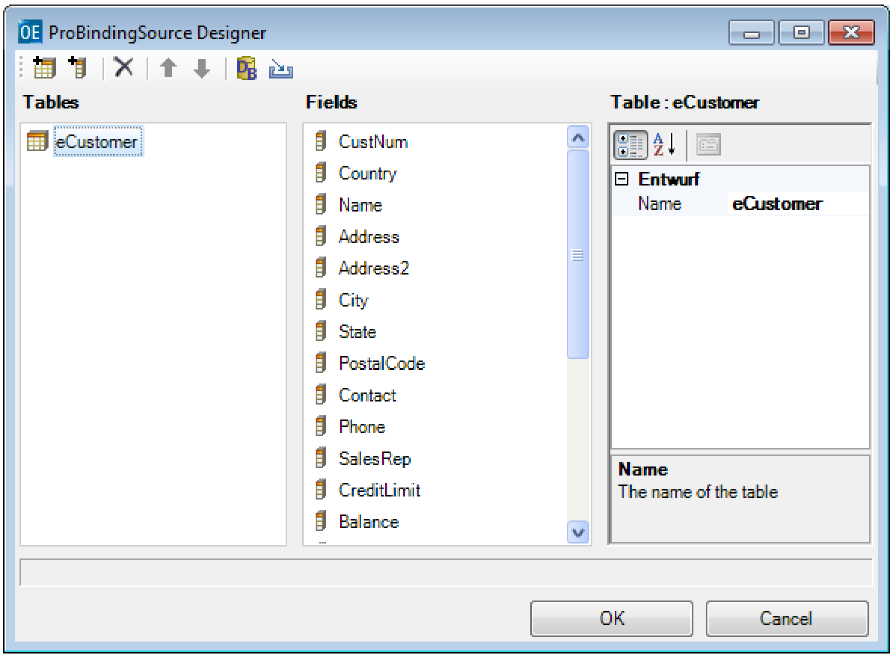
Properties | Events

| | |
|---|---|
| (Name) | CustomerBindingSource |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |
| BindTo | QUERY |
| DataMember | |
| DataSource | |
| EntityJoin | NO |
| EntityName | MyBusinessEntities.CustomerBusi |
| EntityTable | eCustomer |
| EntityView | eSalesrep |
| Filter | |
| GenerateMember | True |
| MapNullToEmptyString | True |
| MaxDataGuess | 0 |
| Modifiers | Private |
| NoLOBs | False |
| Sort | |

ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

- Select the designer verb **ProBindingSource Designer**.

The **ProBindingSource Designer** you closed earlier in this tutorial opens and shows the schema for the *eCustomer* Temp-Table of your *CustomerBusinessEntitiy*. When you scroll further down in the list of fields you will see that the fields from the *eSalesRep* table (like MonthQuota) will appear there as well.



Tables
- eCustomer

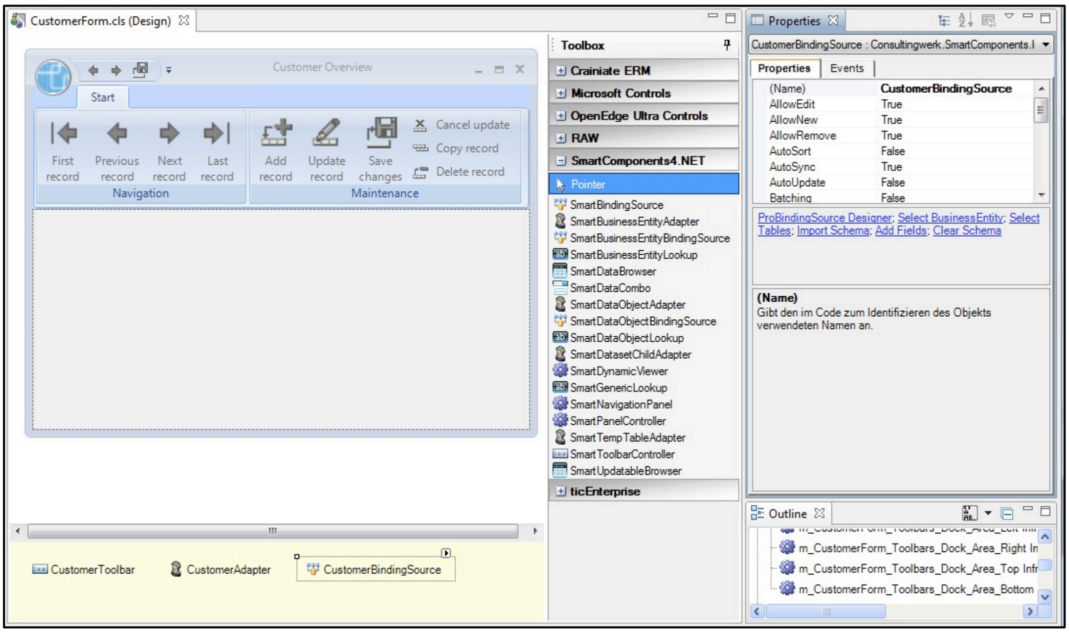
Fields
- CustNum
- Country
- Name
- Address
- Address2
- City
- State
- PostalCode
- Contact
- Phone
- SalesRep
- CreditLimit
- Balance

Table : eCustomer

| | |
|---|---|
| Entwurf | |
| Name | eCustomer |

**Name**
The name of the table

**Congratulation**! You successfully added all non-visual Components to your Form. Your Design Area should look like this:



CustomerBindingSource : Consultingwerk.SmartComponents.I

| | |
|---|---|
| (Name) | **CustomerBindingSource** |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |

ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

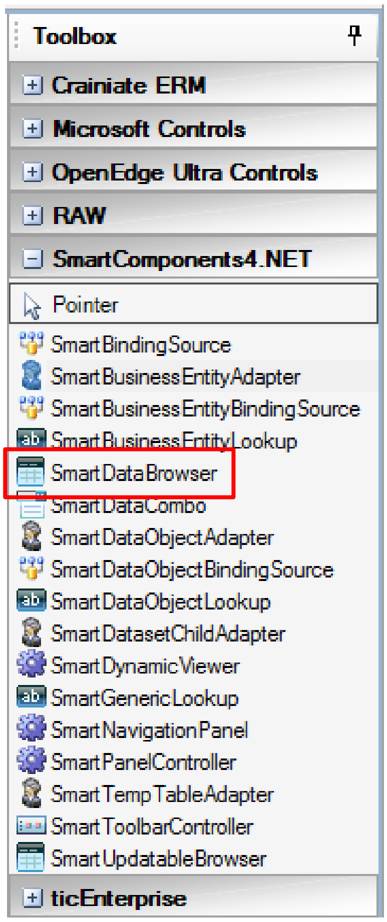
**(Name)**
Gibt den im Code zum Identifizieren des Objekts verwendeten Namen an.

- m_CustomerForm_Toolbars_Dock_Area_Right In
- m_CustomerForm_Toolbars_Dock_Area_Top Infr
- m_CustomerForm_Toolbars_Dock_Area_Bottom

## 5. Add a SmartDataBrowser

You are now ready to add a **SmartDataBrowser** Control to your *CustomerForm* to show data. Note, that the *SmartComponent Library* does alternatively provide an updatable browser which can be used to update data in the grid.

- To add a **SmartDataBrowser** to the Form double-click on the Control in the *Toolbox*.

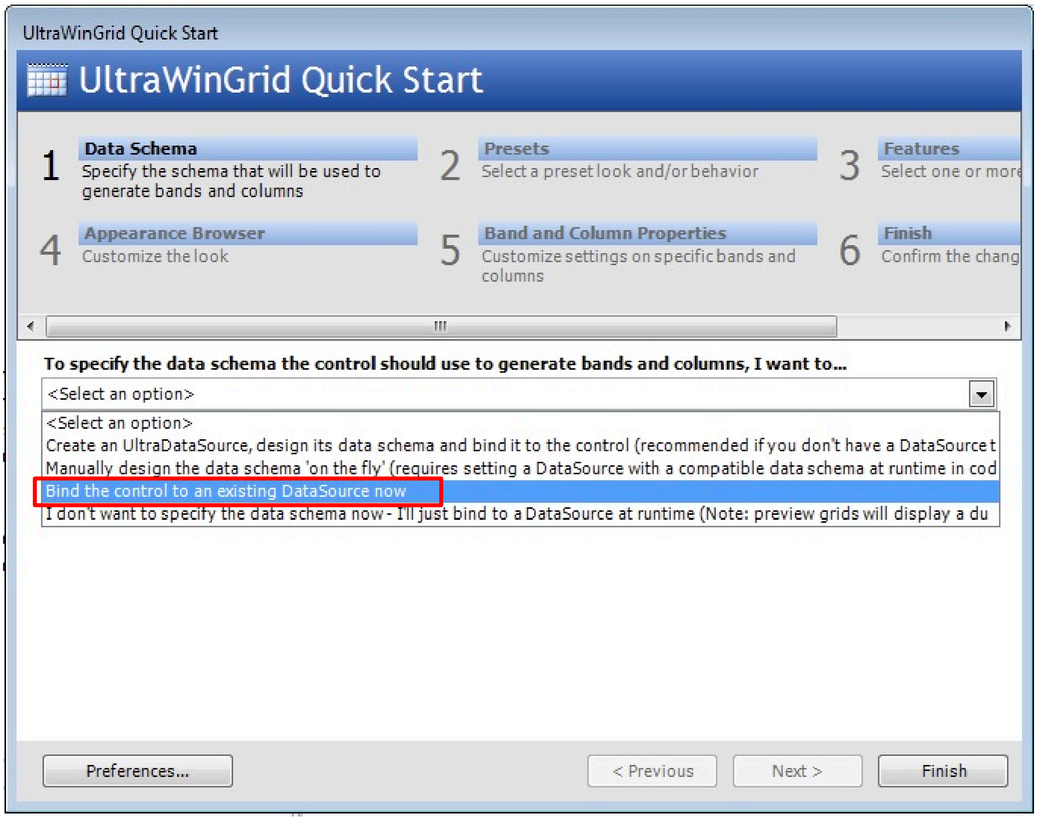


The **UltraWinGrid Quick Start** dialog opens that will guide you to the rest of the setup for a *SmartDataBrowser* Control.

The SmartDataBrowser is an enhanced version (through inheritance) of the UltraGrid Control from Infragistics. Most of the design time functionality comes from Infragistics, e.g. the Quick Start dialog. The *SmartComponent Library* is designed to offer the same options as the underlying Infragistics Controls do – both at runtime and design time.

To set up all necessary settings do the following:

- First specify the **Data Schema**.
- Choose **Bind the Control to an existing DataSource now** from the pull down menu.

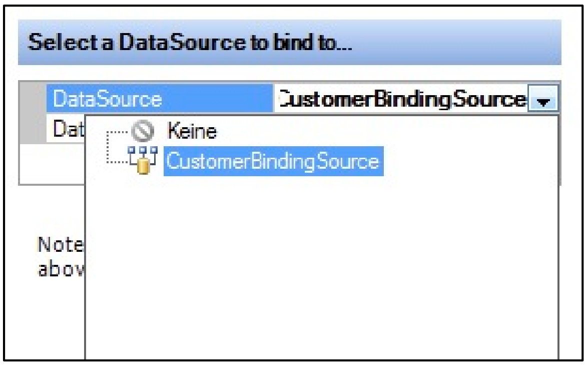


To specify the data schema the control should use to generate bands and columns, I want to...

<Select an option>

<Select an option>
Create an UltraDataSource, design its data schema and bind it to the control (recommended if you don't have a DataSource t
Manually design the data schema 'on the fly' (requires setting a DataSource with a compatible data schema at runtime in cod
Bind the control to an existing DataSource now
I don't want to specify the data schema now - I'll just bind to a DataSource at runtime (Note: preview grids will display a du

- Click **Next**.
- From the drop down list select the **DataSource** Component *CustomerBindingSource* you added earlier to the Form.

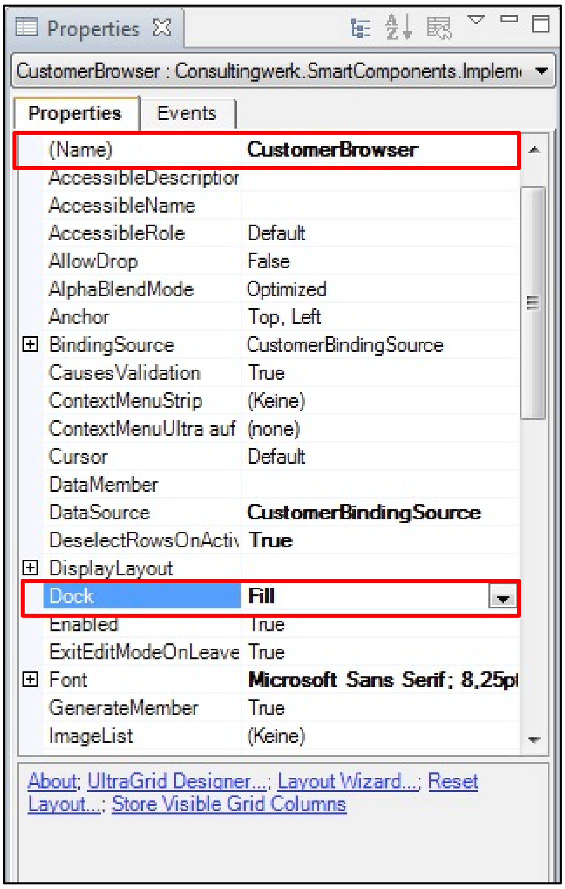


- You don’t need to make any changes to the browser Control so far.
- Click **Finish** to exit the dialog.

For more information about the browser setting refer to Online Help from Infragistics: http://www.infragistics.com/support/documentation.aspx#OnlineDocumentation

- Rename the Control to **CustomerBrowser**.
- To automatically maximize the **SmartDataBrowser** in the Form use the **Properties** view and set the property **Dock** to **Fill**.

From now on the .NET Framework will always resize the Control to the maximum available size, no matter what will cause the Form to resize at runtime. Even maximizing or minimizing the Ribbon will cause the *SmartDataBrowser* to resize properly.

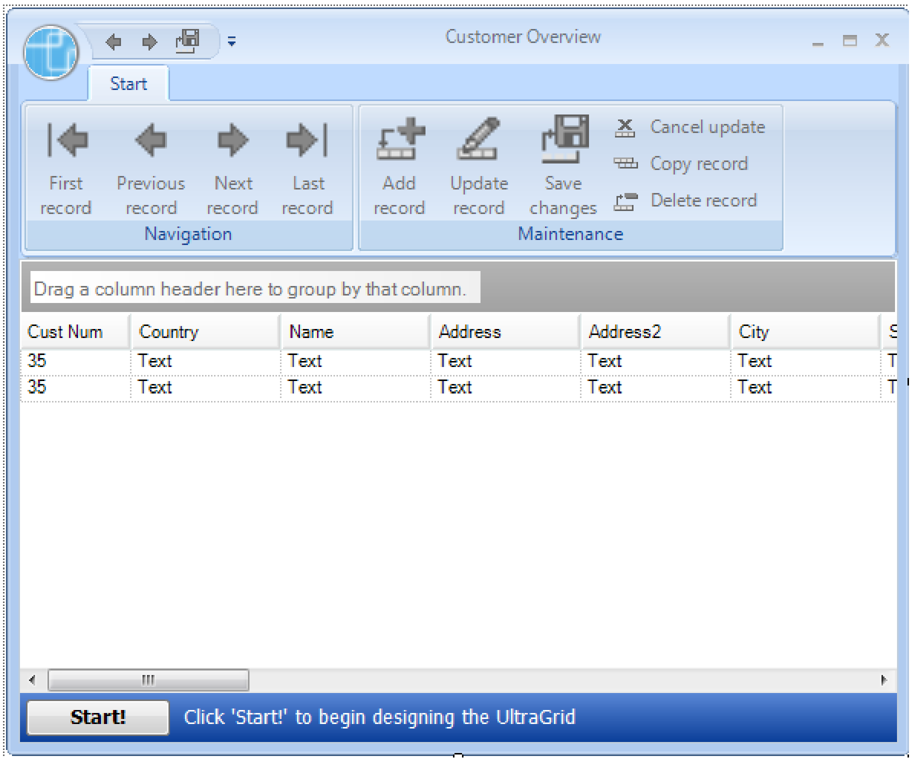


CustomerBrowser : Consultingwerk.SmartComponents.Impleme

| | |
|---|---|
| (Name) | **CustomerBrowser** |
| AccessibleDescriptior | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| AlphaBlendMode | Optimized |
| Anchor | Top, Left |
| ⊞ BindingSource | CustomerBindingSource |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| ContextMenuUltra auf | (none) |
| Cursor | Default |
| DataMember | |
| DataSource | **CustomerBindingSource** |
| DeselectRowsOnActiv | **True** |
| ⊞ DisplayLayout | |
| Dock | **Fill** |
| Enabled | True |
| ExitEditModeOnLeave | True |
| ⊞ Font | **Microsoft Sans Serif; 8,25pt** |
| GenerateMember | True |
| ImageList | (Keine) |

About; UltraGrid Designer...; Layout Wizard...; Reset Layout...; Store Visible Grid Columns

Your Form should look like this:



| Cust Num | Country | Name | Address | Address2 | City | S |
|---|---|---|---|---|---|---|
| 35 | Text | Text | Text | Text | Text | T |
| 35 | Text | Text | Text | Text | Text | T |

Click 'Start!' to begin designing the UltraGrid

## 6. Define Links

If you run the Form *Customer Overview* you will see the *SmartDataBrowser* Control and the Ribbon but you won't see any data in the browse Control, because, as already mentioned, the *BindingSource* Control does not read or display data and we have not yet connected the browser to the *SmartBusinessEntityAdapter* that will actually read the data.

The complete read and update functionality is provided to you in several methods in the class hierarchy of the *SmartComponent Library* Controls and Components. All you have to do is to invoke the appropriate methods and connect the Controls or Components. You do this by defining links between Controls.

A **Link** is a connection between two Controls or Components, like a SmartDataBrowser and its Data Source, and defines how the Controls interact with each other and the kind of interaction in the form of messages (method calls) that are passed between them. In a linked relationship one Control or Component is always the source and the other one is the target for information.

Typically the link is defined using the property sheet of the target because the target will need to know the source more than the source needs to know the target. Also typically a link target will have exactly one single source for any link, but a source (like a *SmartBusinessEntityAdapter* as a data source) can interact with multiple link targets at the same time.

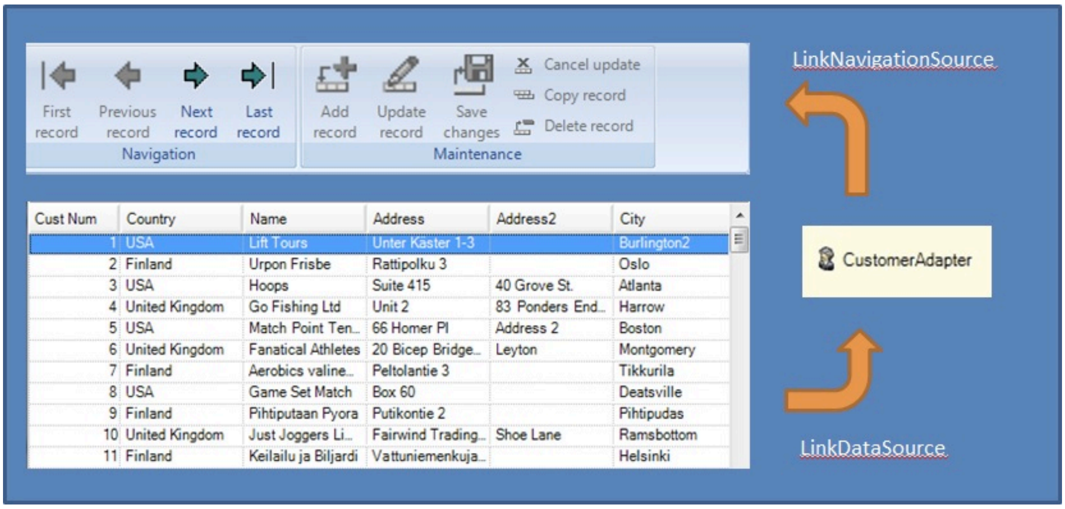
The link types you can specify when placing Controls or Components into a Form:

| Link type | Description | Target Control > Source Control |
|---|---|---|
| Data | Enables dataflow from the data source object to the visualization objects and modification (update) of the data in reverse order | SmartDataBrowser > SmartBusinessEntityAdapter<br>SmartDataViewer > SmartBusinessEntityAdapter |
| Navigation | Enables the *SmartToolbarController* to instruct a *SmartBusinessEntityAdapter* to move to the next, previous, first or last record | SmartBusinessEntityAdapter > SmartToolbarController |
| TableIO | Enables the *SmartToolbarController* to Control a visual Control to modify, create or delete data. | SmartUpdatableBrowser > SmartToolbarController<br>SmartViewerControl > SmartToolbarController |
| GroupAssign | Provides a way to update a record that is displayed in more than one SmartDataViewer. All data of a defined SmartGroupAssign group will be saved simultaneously. All group members must refer to the same Temp-Table (s) of the same *SmartBusinessEntityAdapter*. A *SmartGroupAssign* Link is typically used when data from a single record is presented on two pages of a tab folder. | SmartViewerControl (SmartGroupAssignTarget) -><br>SmartViewerControl (SmartGroupAssignSource) |

To display data in the browser you need to define a **Data-Link** between the *SmartDataBrowser* Control and the *CustomerAdapter* Component.

To navigate through the records using the toolbar buttons you need to define a **Navigation Link** between the *CustomerAdapter* and *CustomerToolbar*.

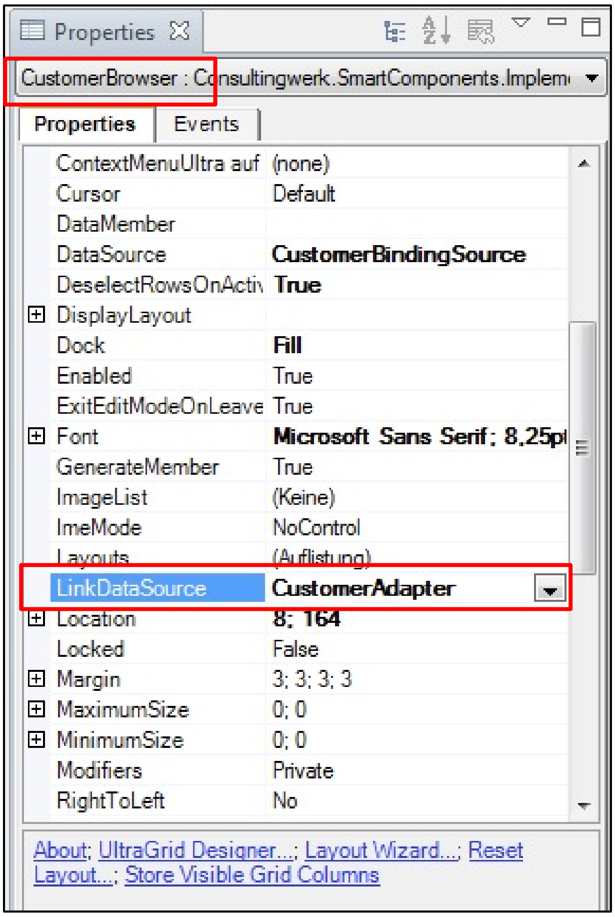
In the *SmartComponent Library* links are defined using properties. The property names provided by the *SmartComponent Library* define the appropriate links and likewise qualify link type and direction. In other words the property **LinkDataSource** means: “link the selected Control to its data source”. That makes it very easy for you to set all links properly.



| Cust Num | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours | Unter Kaster 1-3 | | Burlington2 |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila |
| 8 | USA | Game Set Match | Box 60 | | Deatsville |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki |

## Define a Data Link to display data in the SmartDataBrowser

- To provide data for the *SmartDataBrowser*, set the property **LinkDataSource** of the *CustomerBrowser* Control to *CustomerAdapter*.



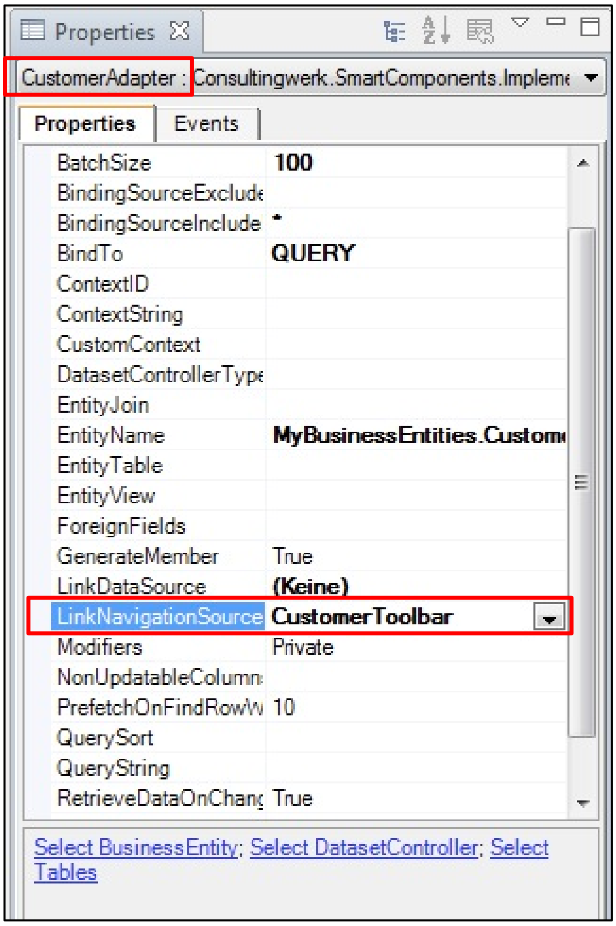
CustomerBrowser : Consultingwerk.SmartComponents.Implem

| | |
|---|---|
| ContextMenuUltra auf | (none) |
| Cursor | Default |
| DataMember | |
| DataSource | **CustomerBindingSource** |
| DeselectRowsOnActiv | **True** |
| ⊞ DisplayLayout | |
| Dock | **Fill** |
| Enabled | True |
| ExitEditModeOnLeave | True |
| ⊞ Font | **Microsoft Sans Serif; 8,25pt** |
| GenerateMember | True |
| ImageList | (Keine) |
| ImeMode | NoControl |
| Layouts | (Auflistung) |
| LinkDataSource | **CustomerAdapter** |
| ⊞ Location | **8; 164** |
| Locked | False |
| ⊞ Margin | 3; 3; 3; 3 |
| ⊞ MaximumSize | 0; 0 |
| ⊞ MinimumSize | 0; 0 |
| Modifiers | Private |
| RightToLeft | No |

About; UltraGrid Designer...; Layout Wizard...; Reset Layout...; Store Visible Grid Columns

## Define a Navigation Link to navigate data using the buttons in the toolbar

- To navigate the data, set the property **LinkNavigationSource** of the *CustomerAdapter* Control to **CustomerToolbar**.



CustomerAdapter : Consultingwerk.SmartComponents.Impleme

| | |
|---|---|
| BatchSize | 100 |
| BindingSourceExclude | |
| BindingSourceInclude | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControllerType | |
| EntityJoin | |
| EntityName | MyBusinessEntities.Custome |
| EntityTable | |
| EntityView | |
| ForeignFields | |
| GenerateMember | True |
| LinkDataSource | (Keine) |
| LinkNavigationSource | CustomerToolbar |
| Modifiers | Private |
| NonUpdatableColumn | |
| PrefetchOnFindRowW | 10 |
| QuerySort | |
| QueryString | |
| RetrieveDataOnChang | True |

Select BusinessEntity; Select DatasetController; Select Tables

## Write the required initialization Code

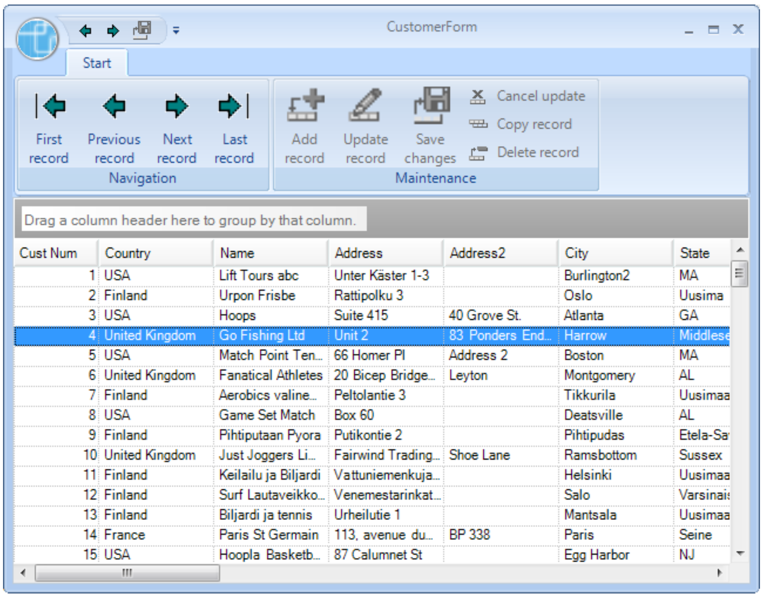
To instruct the *SmartBusinessEntityAdapter* to read data from the backend you need to invoke the *RetrieveData* method of the *CustomerAdapter* instance. This can be done in the constructor of the Form by adding the following code after the call to the *InitializeComponent ()* method. Alternatively you can invoke the RetrieveData method in the OnLoad method override of the Form (http://msdn.microsoft.com/en-us/library/system.windows.forms.form.onload.aspx).

⚠ Prior to the call to *InitializeComponent* () in the Constructor of the Form, the instance of the *CustomerAdapter* has not yet been created and the call to *RetrieveData* () would fail!

```
CONSTRUCTOR PUBLIC CustomerForm (  ):

  SUPER().
  InitializeComponent().

  THIS-OBJECT:CustomerAdapter:RetrieveData () .

CATCH e AS Progress.Lang.Error:
UNDO, THROW e.
END CATCH.

END CONSTRUCTOR.
```

- Save and test your Form.

The result should look like this!



Drag a column header here to group by that column.

| Cust Num | Country | Name | Address | Address2 | City | State |
|---|---|---|---|---|---|---|
| 1 | USA | Lift Tours abc | Unter Käster 1-3 | | Burlington2 | MA |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo | Uusima |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta | GA |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow | Middlese |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston | MA |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery | AL |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila | Uusimaa |
| 8 | USA | Game Set Match | Box 60 | | Deatsville | AL |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas | Etela-Sa |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom | Sussex |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki | Uusimaa |
| 12 | Finland | Surf Lautaveikko... | Venemestarinkat... | | Salo | Varsinais |
| 13 | Finland | Biljardi ja tennis | Urheilutie 1 | | Mantsala | Uusimaa |
| 14 | France | Paris St Germain | 113, avenue du... | BP 338 | Paris | Seine |
| 15 | USA | Hoopla Basketb... | 87 Calumnet St | | Egg Harbor | NJ |

**Congratulation**! You finished your first Form using the *SmartComponent Library*.

# 7. Create a new Form for Customer Details

To show dependent data like customer details, order or orderline information you build a new Form with a tab Control (alternatively you can also present the detail data in the same Form, however using two Forms is a very common approach). The tab Control manages Controls on multiple pages in a Container.

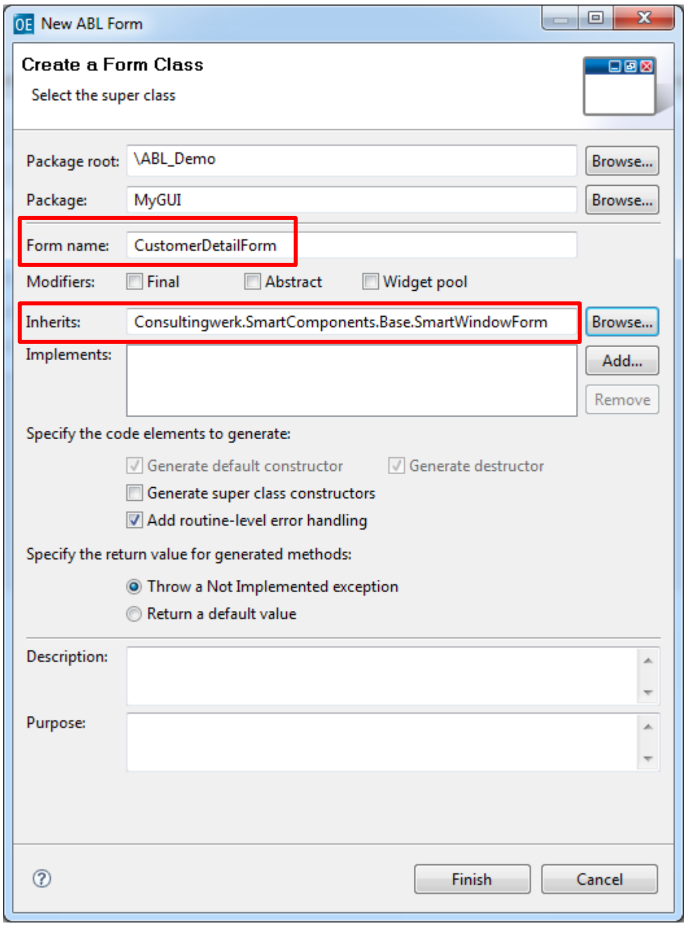
- Create a new Form **CustomerDetailForm** in your *MyGUI* directory.

**Characteristics of the new Form CustomerDetailForm:**

- Change the super class reference to *Consultingwerk.SmartComponents.Base.SmartWindowForm.*
- Add a *SmartToolbarController*, load the *Default Ribbon configuration* and rename it to *CustomerDetailToolbar*.

You already know how to create a new Form. If you need any assistance refer to chapter 1. Create a new Form Customer Overwiew.

The screen of the **New ABL Form** wizard should look like this:



Create a Form Class

Select the super class

Package root: \ABL_Demo

Package: MyGUI

Form name: CustomerDetailForm

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Consultingwerk.SmartComponents.Base.SmartWindowForm

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

☑ Add routine-level error handling

Specify the return value for generated methods:

● Throw a Not Implemented exception

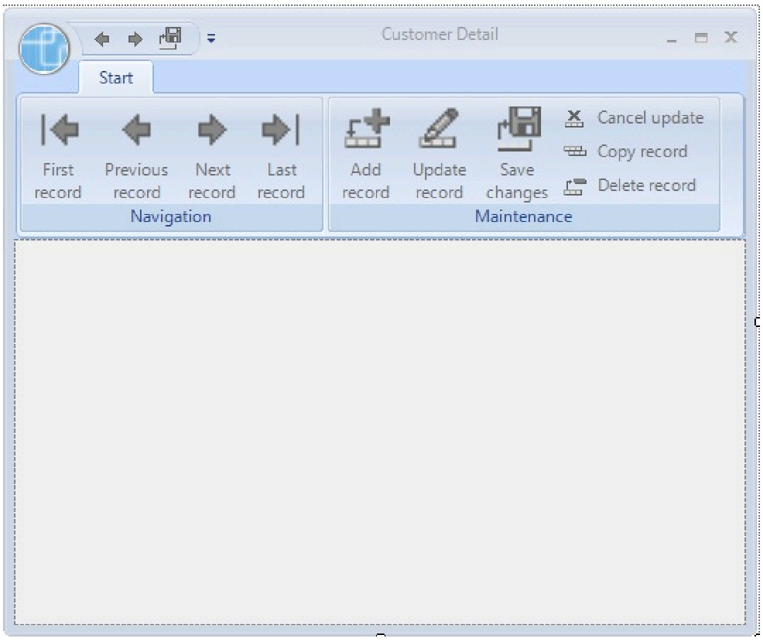
○ Return a default value

Description:

Purpose:

- Click **Finish**.
- Change the property *Text* of the Form to **Customer Detail** to provide a meaningful title of the Form.

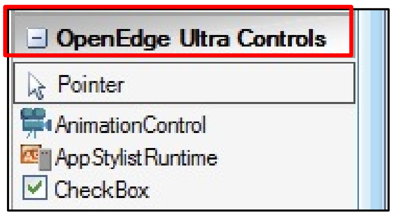
The result should look like this:

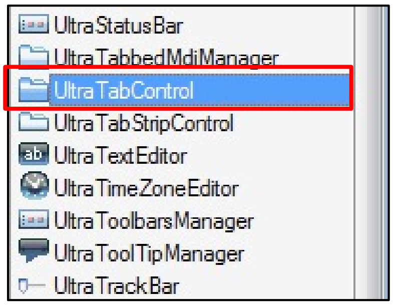


## 8. Insert a Tab Control to a Form

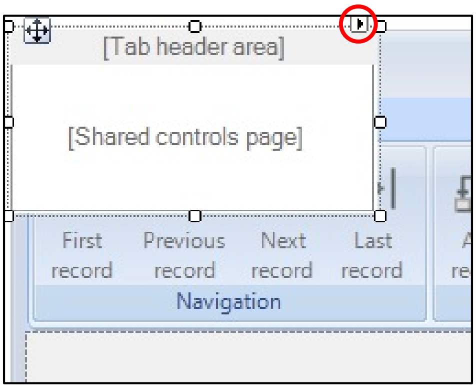
- To insert a Tab Control, go to the **OpenEdge Ultra Controls** group of the Toolbar and select **UltraTabControl**.

⚠ Don't add an *UltraTabStripControl* to the Form. For more information about the actual difference between the *UltraTabControl* and the *UltraTabStrip* Control please refer to the Infragistics Documentation.



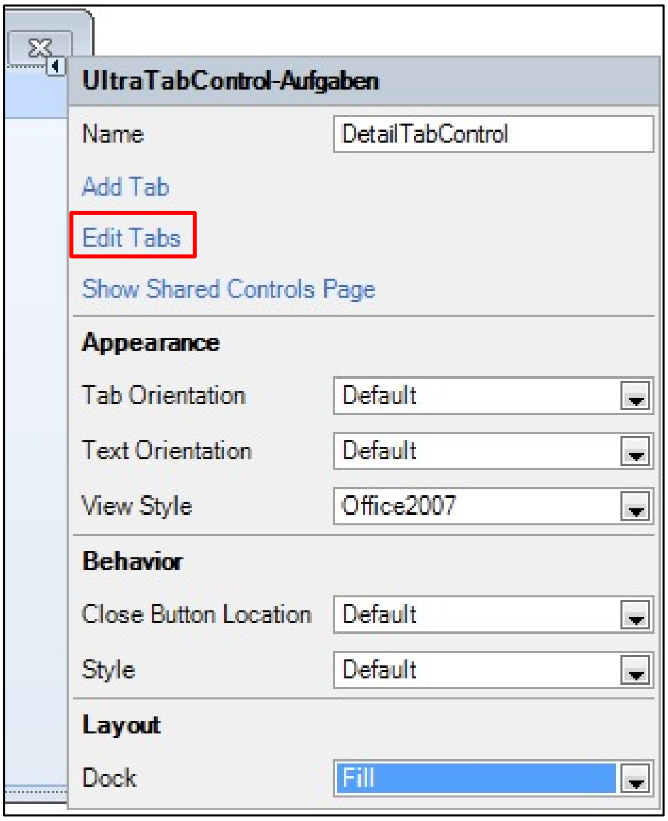


The result should look like this:



- To add a new tab or to edit the Control, open the *SmartTag* for the Control and select **Edit Tab**.

The SmartTag is the tiny arrow pointing to the right on the top right corner of the UltraTabControl. The SmartTag can typically be used to Control the most important properties of a Control in the Visual Designer.



UltraTabControl-Aufgaben

Name: DetailTabControl

**Appearance**

Tab Orientation: Default

Text Orientation: Default

View Style: Office2007

**Behavior**

Close Button Location: Default

Style: Default

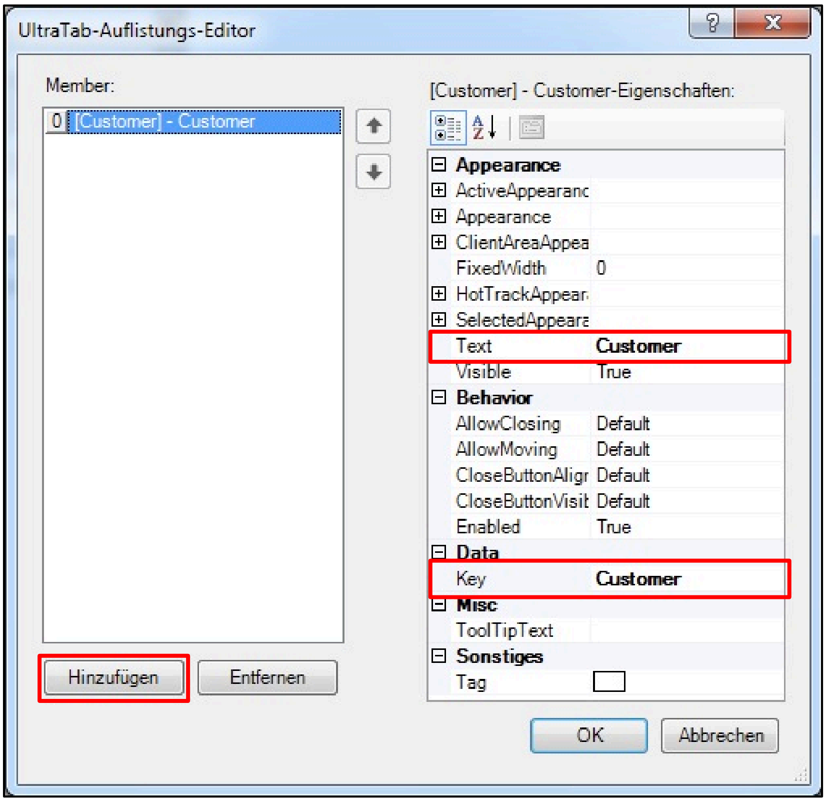
**Layout**

Dock: Fill

The **UltraTab Editor** opens.

- Select button **Add** (Hinzufügen in the screenshot taken on a German installation of Windows) to add a new tab and change the properties **Text** and **Key** to Customer, where **Text** is just the label for the tab and **Key** is the internal unique identifier of the tab page.

You can search for a tab by that Key and use it to identify which tab is currently chosen. We come back to this later in this tutorial.



UltraTab-Auflistungs-Editor

Member:

0 [Customer] - Customer

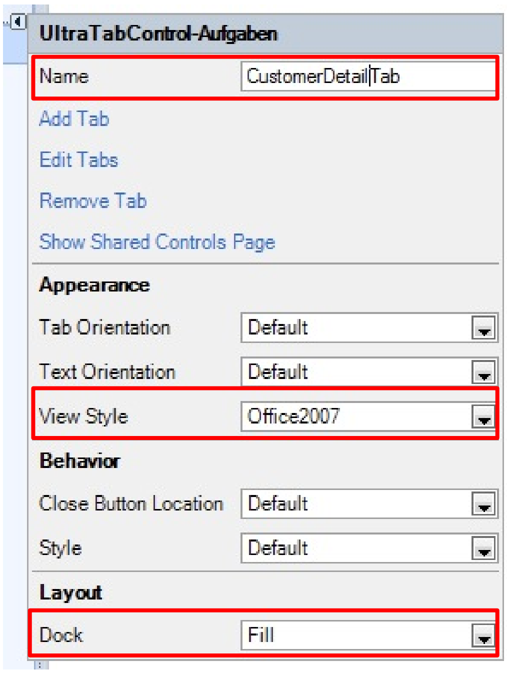
[Customer] - Customer-Eigenschaften:

| | |
|---|---|
| **Appearance** | |
| ActiveAppearanc | |
| Appearance | |
| ClientAreaAppea | |
| FixedWidth | 0 |
| HotTrackAppear | |
| SelectedAppeara | |
| Text | **Customer** |
| Visible | True |
| **Behavior** | |
| AllowClosing | Default |
| AllowMoving | Default |
| CloseButtonAligr | Default |
| CloseButtonVisit | Default |
| Enabled | True |
| **Data** | |
| Key | **Customer** |
| **Misc** | |
| ToolTipText | |
| **Sonstiges** | |
| Tag | □ |

- Click **OK**.

Change the following properties:

- **Name**: CustomerDetailTab
- **View Style**: Office 2007
- **Dock**: Fill



UltraTabControl-Aufgaben

Name: CustomerDetailTab

**Appearance**

Tab Orientation: Default

Text Orientation: Default

View Style: Office2007

**Behavior**

Close Button Location: Default

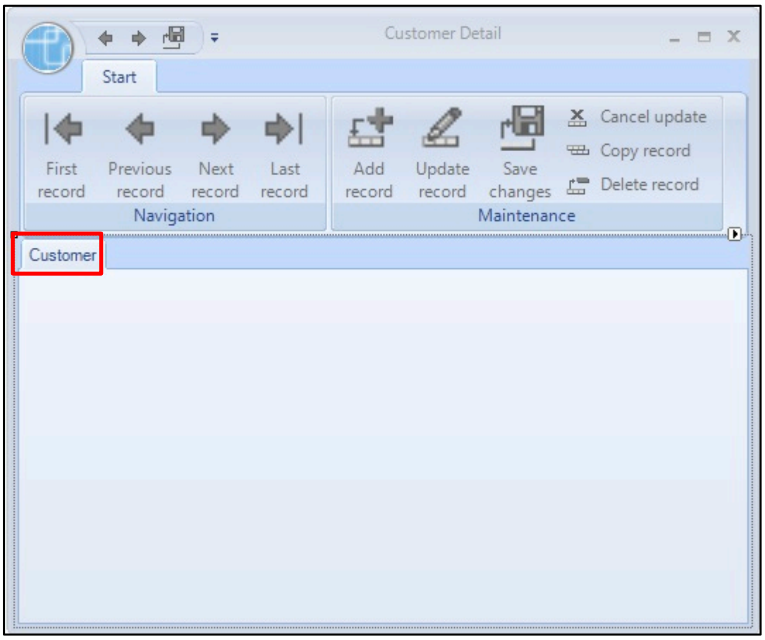
Style: Default

**Layout**

Dock: Fill

So far you have one tab defined in your tab Control.

The result should look like this.



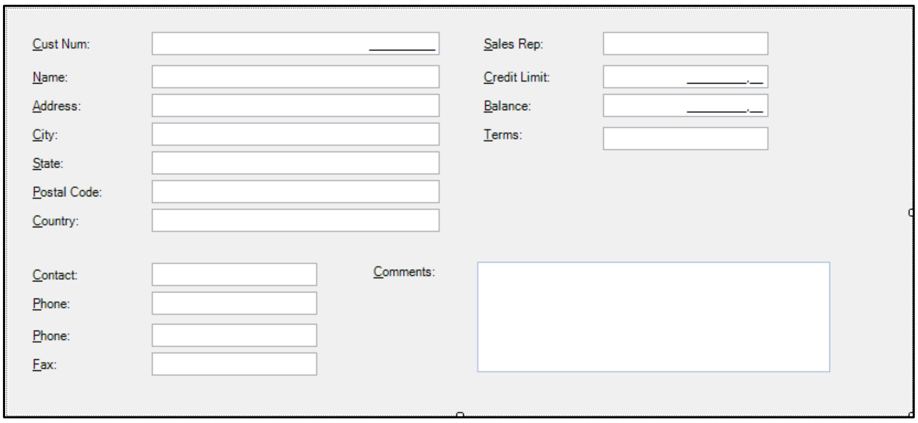
In the next chapter you will learn how to create a viewer Control to display customer detail data and how to insert the viewer Control to the Customer tab page. Further in this tutorial you will add two additional tabs for order and orderline information.

## 9. Create a SmartViewerControl *CustomerDetailViewer*

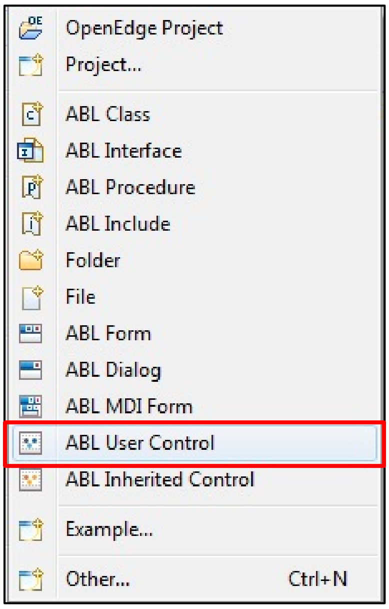
In the previous chapter you have assembled the *Customer Overview* Form that lists all customer records in a browse Control. In this lesson you will learn how to create a viewer Control that shows detail information for a single customer and allows the user to modify the record or create new records.

In the *SmartComponent Library* a viewer Control is based in the *UserControl* type. A **User Control** is a special type of inherited Control that allows you to combine Controls into a common container that can be reused. In this lesson you combine Fill-Ins from the customer table.

The end result of the *CustomerDetailViewer* will look like this:

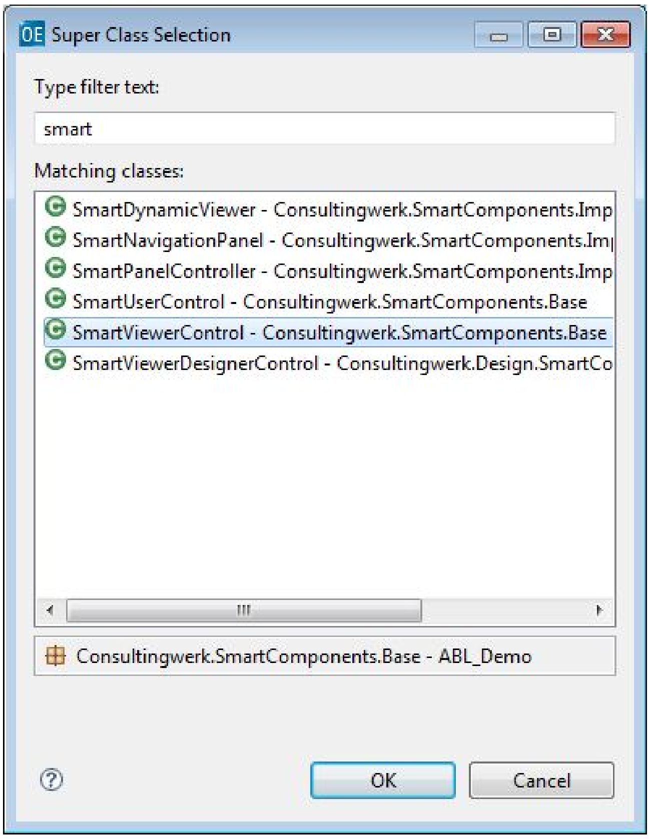


- To create a new ABL User Control, select **File > New > ABL User Control** from the menu.



Make sure that the Control is saved to your *MyGUI* folder and inherits from class *Consultingwerk.SmartComponents.Base.**SmartViewerControl*** to leverage the behavior that is provided with this base type.

- To change the super class in the *Inherits* field of the wizard, click **Browse** and type in *smart* in the *Super Class Selection* dialog. Select class **SmartViewerControl** and click **OK**.



Super Class Selection

Type filter text:

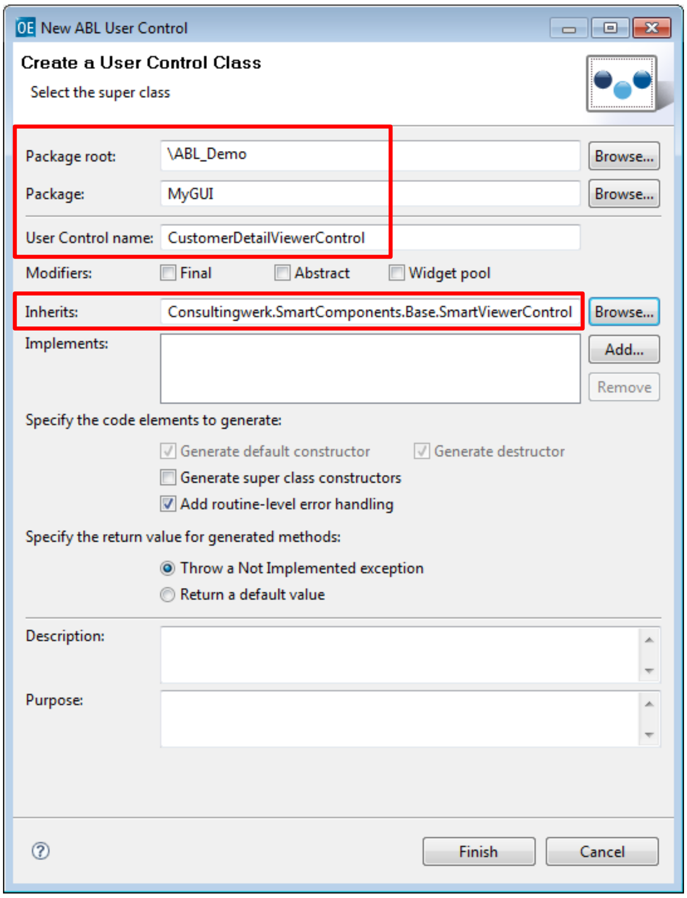
smart

Matching classes:

SmartDynamicViewer - Consultingwerk.SmartComponents.Imp
SmartNavigationPanel - Consultingwerk.SmartComponents.Im|
SmartPanelController - Consultingwerk.SmartComponents.Imp
SmartUserControl - Consultingwerk.SmartComponents.Base
SmartViewerControl - Consultingwerk.SmartComponents.Base
SmartViewerDesignerControl - Consultingwerk.Design.SmartCo

Consultingwerk.SmartComponents.Base - ABL_Demo

The result screen of the **New ABL User Control** dialog should look like this:



New ABL User Control

Create a User Control Class

Select the super class

Package root: \ABL_Demo

Package: MyGUI

User Control name: CustomerDetailViewerControl

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Consultingwerk.SmartComponents.Base.SmartViewerControl

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

☑ Add routine-level error handling

Specify the return value for generated methods:

● Throw a Not Implemented exception

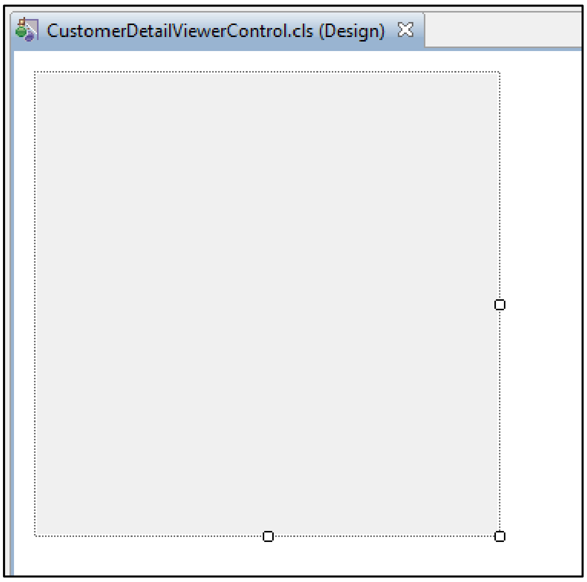
○ Return a default value

Description:

Purpose:

- Click **Finish**.

The User Control Form is opened in the Visual Designer.



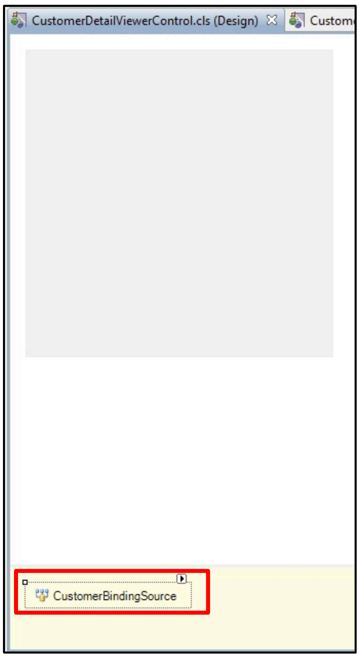
## Duplicate the Binding Source

As mentioned earlier in the tutorial a .NET Control cannot directly connect to a Business Entity or an Entity Adapter. Therefore you need a *BindingSource* Control that binds the data to visualization Controls like a browser or a viewer. The *BindingSource* you need for the *Customer Detail* Form is typically identical to the one you created earlier for the *Customer Overview* Form. You can repeat the steps described in chapter 4. Add a SmartBusinessEntityBindingSource or copy-paste the BindingSource Component from the Form design.

**To copy the *SmartBusinessEntityBindingSource*:**

- Open the file *CustomerForm.cls*, select *CustomerBindingSource* and copy it.
- Open the file *CustomerDetailViewerControl.cls* and paste it.

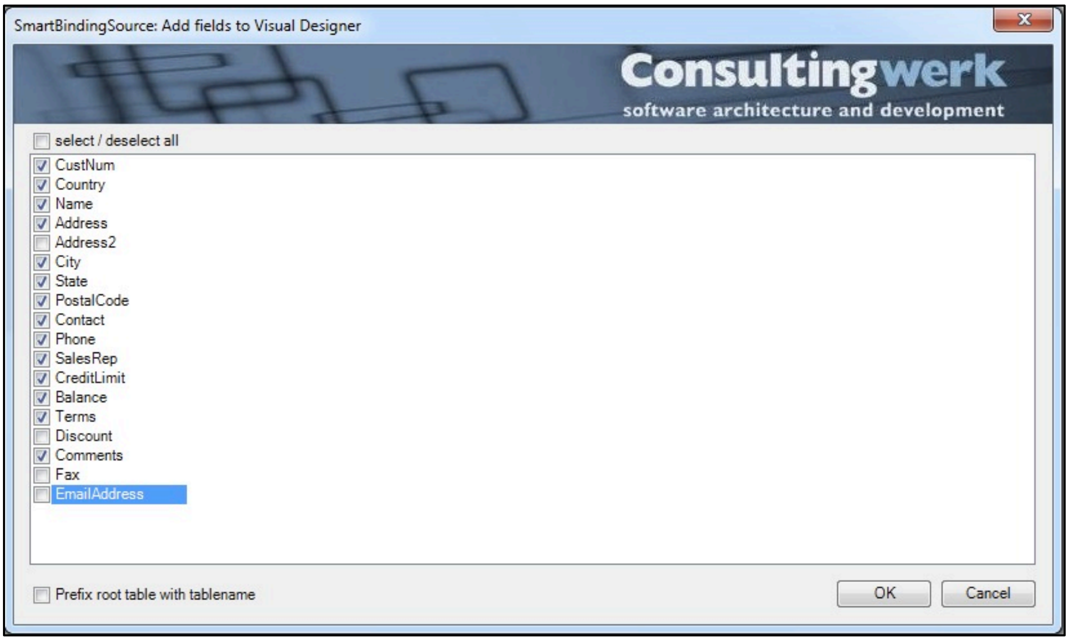
The result should look like this:



## Add Fields to the User Control

After you copied the *SmartBindingSource* Control to the User Control you have access to the schema, because the schema has been copied along with all other properties of the Component Instance and you can directly add the fields you need for the viewer.

- To add fields, use the **Properties** view and select the designer verb **Add Fields** for *CustomerBindingSource*.
- The **Add fields to Visual Designer** dialog opens.
- Select the fields of your choice and click **OK**.



SmartBindingSource: Add fields to Visual Designer

☐ select / deselect all

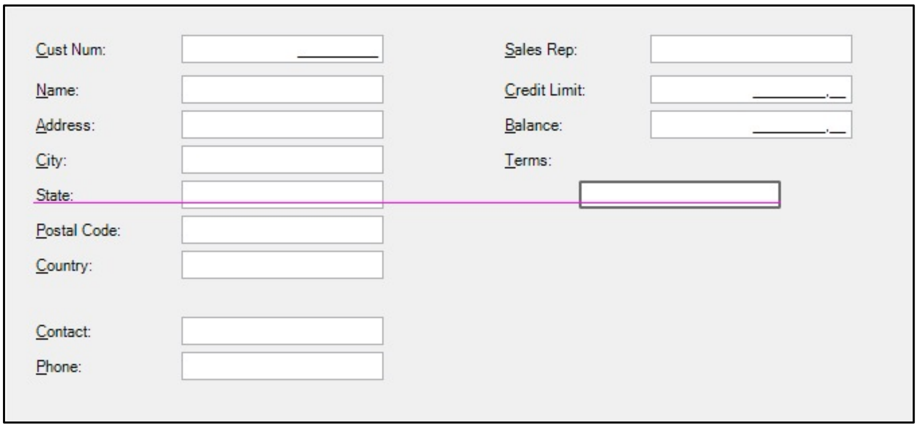
☑ CustNum
☑ Country
☑ Name
☑ Address
☐ Address2
☑ City
☑ State
☑ PostalCode
☑ Contact
☑ Phone
☑ SalesRep
☑ CreditLimit
☑ Balance
☑ Terms
☐ Discount
☑ Comments
☐ Fax
☐ EmailAddress

☐ Prefix root table with tablename

The ***Add Fields*** functionality of the *SmartBusinessEntityBindingSource* and all other *SmartBindingSource* Components is a very convenient method of adding fields for a viewer. Based on the data type of each field added to the Viewer design, standardized properties for the Controls will be used.

The functionality of the *SmartViewerControl* does not rely on this method of adding Controls to the SmartViewer. You may add any .NET Control from your Visual Designer Toolbox and specify Data Bindings for any property required. Please consult the documentation of your Control's vendor for details about the Data Binding capabilities.

To change the order of the fields you can drag and drop them to the desired location. Colored helplines (Snaplines) help you to align the fields properly.



Cust Num:

Name:

Address:

City:

State:

Postal Code:

Country:

Contact:

Phone:

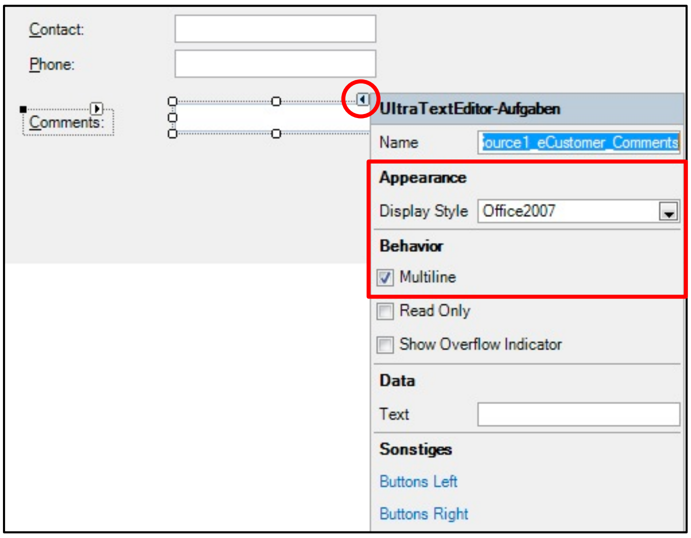
Sales Rep:

Credit Limit:

Balance:

Terms:

- Use the **SmartTag** of the **comments** field to change the layout to *Multiline* (Editor) and the appearance to *Office 2007* style.
- Increase the new field.



Contact:

Phone:

Comments:

UltraTextEditor-Aufgaben

Name ource1_eCustomer_Comments

**Appearance**

Display Style Office2007

**Behavior**

☑ Multiline

☐ Read Only

☐ Show Overflow Indicator

**Data**

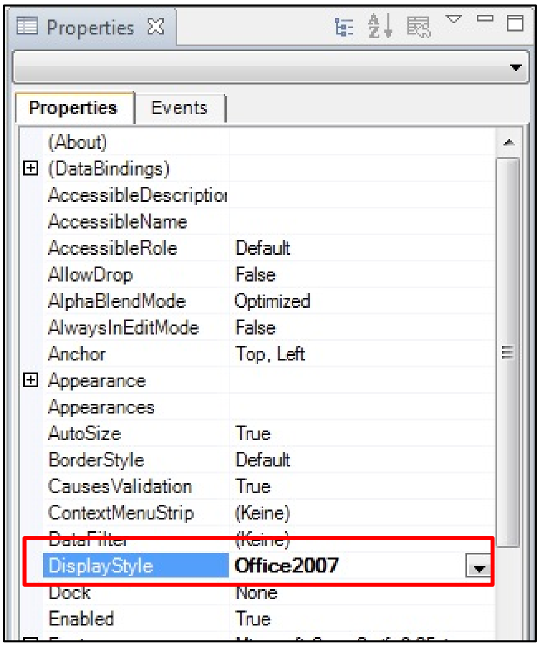
Text

**Sonstiges**

Buttons Left

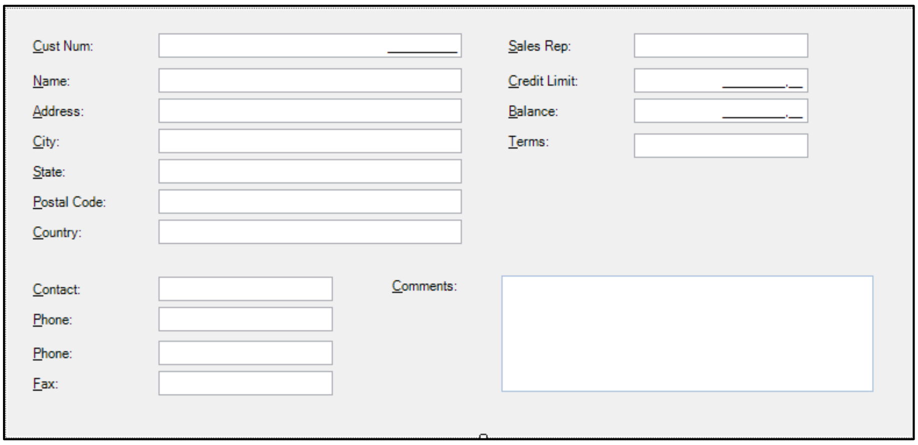
Buttons Right

- Mark all remaining *UltraTextEditor* Controls (Fill-Ins) and change the property *DisplayStyle* to **Office 2007** as well.



| | |
|---|---|
| (About) | |
| ⊞ (DataBindings) | |
| AccessibleDescriptio | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| AlphaBlendMode | Optimized |
| AlwaysInEditMode | False |
| Anchor | Top, Left |
| ⊞ Appearance | |
| Appearances | |
| AutoSize | True |
| BorderStyle | Default |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| DataFilter | (Keine) |
| DisplayStyle | **Office2007** |
| Dock | None |
| Enabled | True |

The final result should look like this:



**Congratulation**! You successfully created a *SmartViewerControl*.

The User Control itself cannot be executed. To execute it you must place it on a Form. We will show you how to do this in the next chapter.

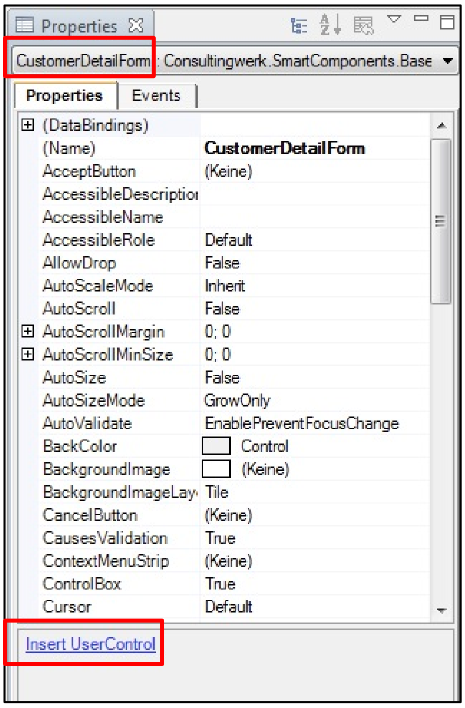
# 10. Insert a User Control in a Form

You already have the Customer Detail Form with a tab Control and you have created a *SmartViewer* Control to display customer detail data. In this chapter you will bring those two together. You'll add the User Control in the Tab Control.

**To insert the User Control in the Form follow these steps:**

- Open the file **CustomerDetailForm.cls**
- Select the Form Control itself by clicking on the Window Title bar
- In the Properties view select the designer verb **Insert UserControl** for Form *CustomerDetailForm*.

The designer verb **Insert UserControl** is a special feature of *SmartWindowForm* derived classes. Normally you must add the user Control to the toolbar to first before adding it to a Form.



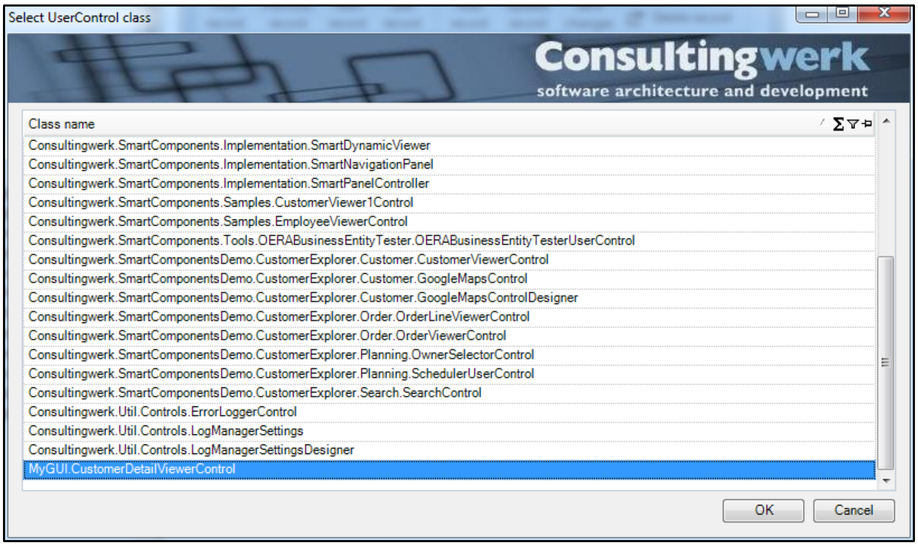
CustomerDetailForm : Consultingwerk.SmartComponents.Base

| | |
|---|---|
| ⊞ (DataBindings) | |
| (Name) | **CustomerDetailForm** |
| AcceptButton | (Keine) |
| AccessibleDescriptio | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| AutoScaleMode | Inherit |
| AutoScroll | False |
| ⊞ AutoScrollMargin | 0; 0 |
| ⊞ AutoScrollMinSize | 0; 0 |
| AutoSize | False |
| AutoSizeMode | GrowOnly |
| AutoValidate | EnablePreventFocusChange |
| BackColor | □ Control |
| BackgroundImage | □ (Keine) |
| BackgroundImageLay | Tile |
| CancelButton | (Keine) |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| ControlBox | True |
| Cursor | Default |

Insert UserControl

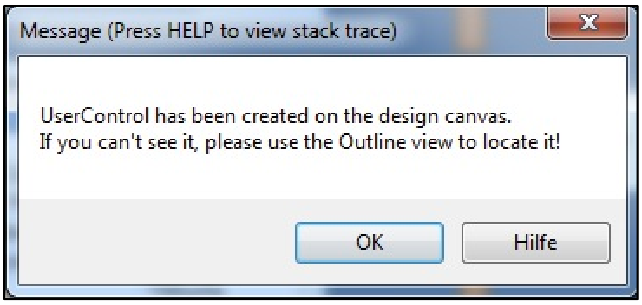
The *Select UserControl class* dialog opens.

- Select *CustomerDetailViewerControl* you created earlier in the tutorial and click **OK**.



Consultingwerk.SmartComponents.Implementation.SmartDynamicViewer
Consultingwerk.SmartComponents.Implementation.SmartNavigationPanel
Consultingwerk.SmartComponents.Implementation.SmartPanelController
Consultingwerk.SmartComponents.Samples.CustomerViewer1Control
Consultingwerk.SmartComponents.Samples.EmployeeViewerControl
Consultingwerk.SmartComponents.Tools.OERABusinessEntityTester.OERABusinessEntityTesterUserControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Customer.CustomerViewerControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Customer.GoogleMapsControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Customer.GoogleMapsControlDesigner
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Order.OrderLineViewerControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Order.OrderViewerControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Planning.OwnerSelectorControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Planning.SchedulerUserControl
Consultingwerk.SmartComponentsDemo.CustomerExplorer.Search.SearchControl
Consultingwerk.Util.Controls.ErrorLoggerControl
Consultingwerk.Util.Controls.LogManagerSettings
Consultingwerk.Util.Controls.LogManagerSettingsDesigner
MyGUI.CustomerDetailViewerControl

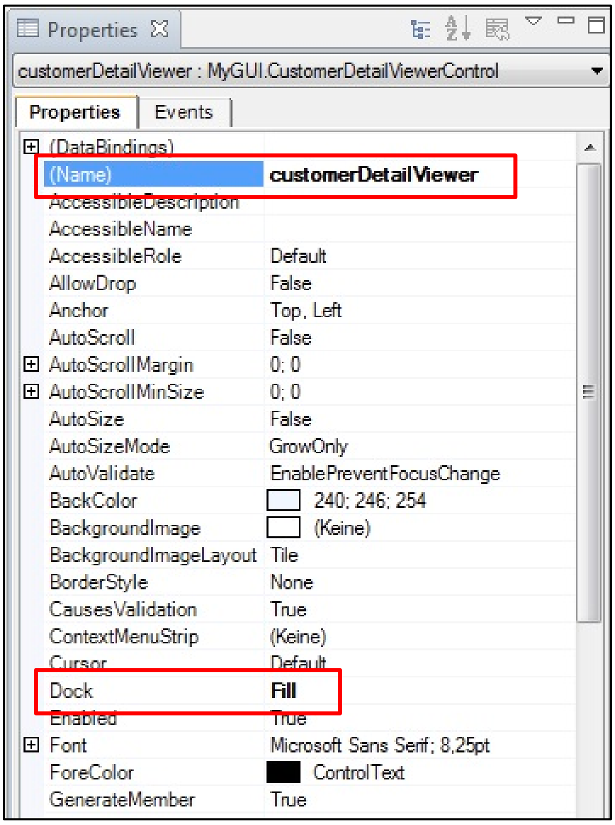
The wizard functionality cannot Control the actual location of the new *UserControl* instance on the Form. In rare cases the *UserControl* is created in the background of some other Control(s) so that you may not see it. A message reminds you to refer to the Outline view if this will happen.



UserControl has been created on the design canvas.
If you can't see it, please use the Outline view to locate it!

Change the following properties for *CustomerDetailViewer*:

- Property **Name**: CustomerDetailViewer
- Property **Dock**: Fill



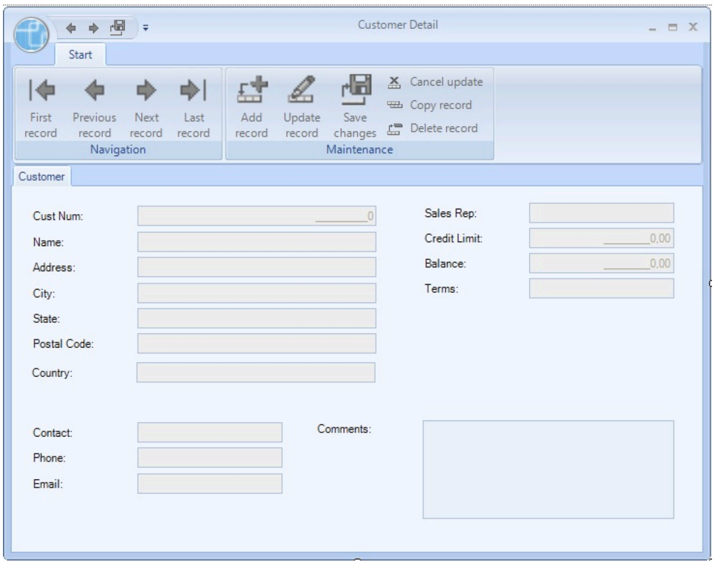
Properties

customerDetailViewer : MyGUI.CustomerDetailViewerControl

Properties | Events

| | |
|---|---|
| (DataBindings) | |
| (Name) | customerDetailViewer |
| AccessibleDescription | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| Anchor | Top, Left |
| AutoScroll | False |
| AutoScrollMargin | 0; 0 |
| AutoScrollMinSize | 0; 0 |
| AutoSize | False |
| AutoSizeMode | GrowOnly |
| AutoValidate | EnablePreventFocusChange |
| BackColor | 240; 246; 254 |
| BackgroundImage | (Keine) |
| BackgroundImageLayout | Tile |
| BorderStyle | None |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| Cursor | Default |
| Dock | Fill |
| Enabled | True |
| Font | Microsoft Sans Serif; 8,25pt |
| ForeColor | ControlText |
| GenerateMember | True |

The result screen should look like this:

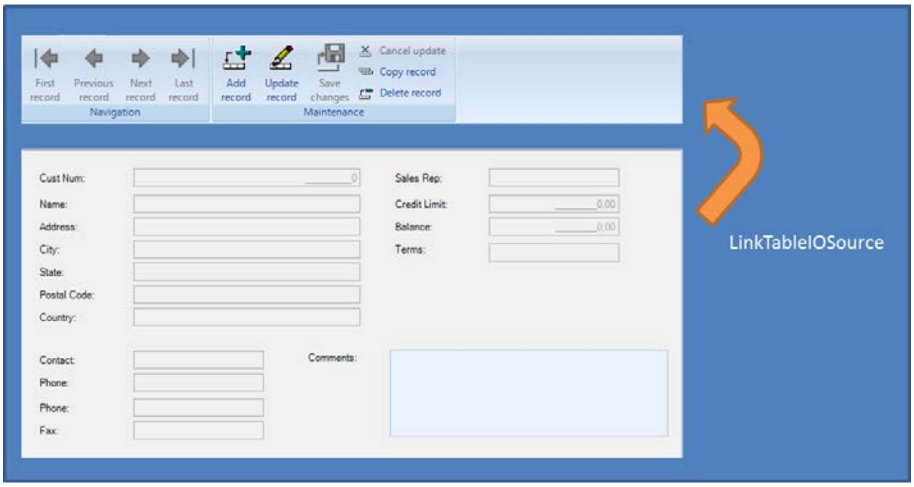


# 11. Specify Links for Form Customer Detail

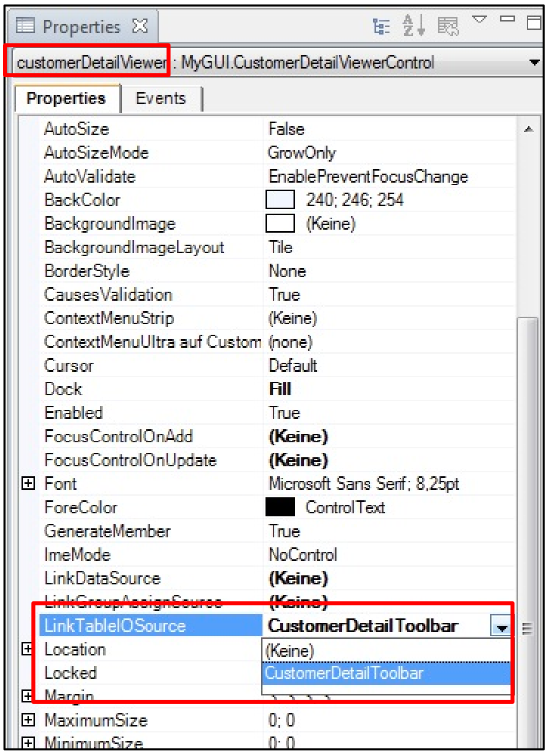
In order to show detail data for customer records in the *SmartViewerControl* you need to specify a **Data Link** to *CustomerAdapter* in the Form *Customer Overview*. Because the Controls and Components in each of the Forms are typically encapsulated, you have to define a property as a method that allows setting a link at runtime.

For update purposes you need a **Table-IO Link** from *CustomerDetailToolbar* in the *Customer Detail* Form to the *customerDetailViewer* in Form *Customer Detail.*

## Define a Table I/O Link



- To specify the **Table-IO** Link, use the *Properties* view and set the property **LinkTabkeIOSource** for *customerDetailViewer* to *CustomerDetailToolbar*.



Properties

customerDetailViewer : MyGUI.CustomerDetailViewerControl

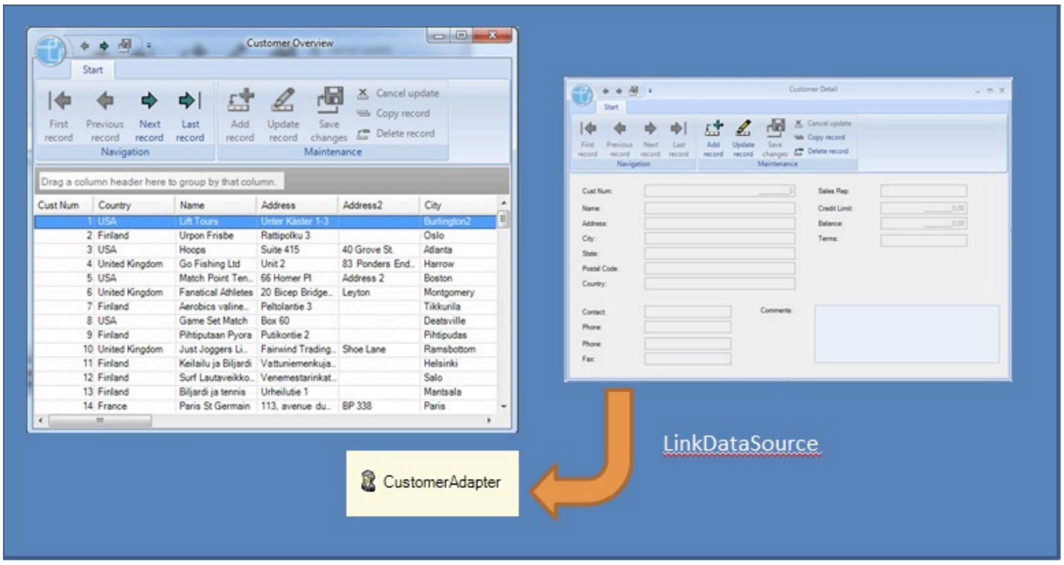
Properties | Events

| | |
|---|---|
| AutoSize | False |
| AutoSizeMode | GrowOnly |
| AutoValidate | EnablePreventFocusChange |
| BackColor | 240; 246; 254 |
| BackgroundImage | (Keine) |
| BackgroundImageLayout | Tile |
| BorderStyle | None |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| ContextMenuUltra auf Custom | (none) |
| Cursor | Default |
| Dock | **Fill** |
| Enabled | True |
| FocusControlOnAdd | **(Keine)** |
| FocusControlOnUpdate | **(Keine)** |
| ⊞ Font | Microsoft Sans Serif; 8,25pt |
| ForeColor | ControlText |
| GenerateMember | True |
| ImeMode | NoControl |
| LinkDataSource | **(Keine)** |
| LinkGroupAssignSource | **(Keine)** |
| LinkTableIOSource | **CustomerDetail Toolbar** |
| ⊞ Location | (Keine) |
| Locked | CustomerDetailToolbar |
| ⊞ Margin | |
| ⊞ MaximumSize | 0; 0 |
| ⊞ MinimumSize | 0; 0 |

## Define a Data Link to the *CustomerAdapter* in Form *Customer Overview*

In order to show detail data for customer records in the *SmartViewerControl* you need to specify a **Data Link** between the *CustomerDetailViewer* Control in Form *Customer Detail* and the *CustomerAdapter* component in the Form *Customer Overview.*

To define a Data Link between two forms you must define your own property.



| Cust Num | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours | Unter Käster 1-3 | | Burlington2 |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila |
| 8 | USA | Game Set Match | Box 60 | | Deatsville |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki |
| 12 | Finland | Surf Lautaveikko... | Venemestarinkat... | | Salo |
| 13 | Finland | Biljardi ja tennis | Urheilutie 1 | | Mantsala |
| 14 | France | Paris St Germain | 113, avenue du... | BP 338 | Paris |

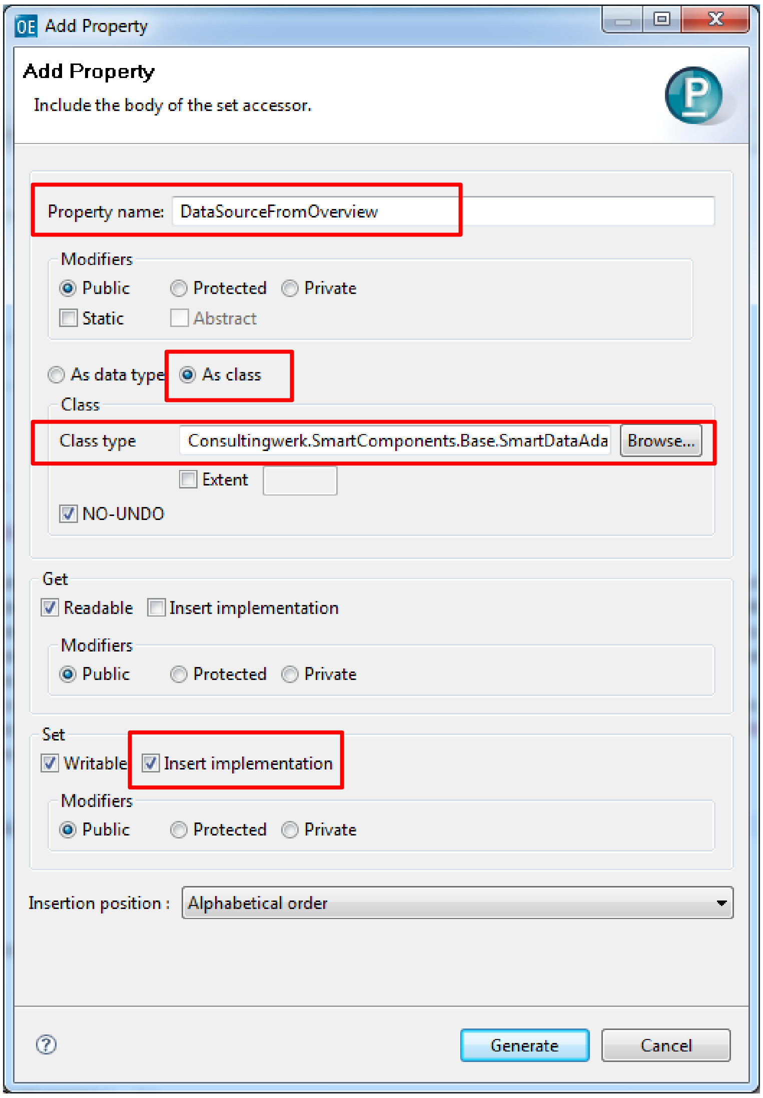
To add a new property, open the file *CustomerDetailForm.cls* and switch to the ABL Editor if it is not already open. You may switch from the Design view to the source code using the shortcut F9.

Right-click on the code editor and select **Source > Add Property**.

The **Add Property** dialog opens.

- Name the new property **DataSourceFromOverview**.
- Select **As class** and click **Browse**.
- In the **Type Selection** dialog type in *smart* as filter text and choose class *Consultingwerk.SmartComponents.Base.SmartDataAdapter.*
- Click **OK**.
- Check **Insert implementation** for the **Set** method.
- Leave all other settings as default.

The result should look like this:



Add Property

Include the body of the set accessor.

Property name: DataSourceFromOverview

Modifiers

● Public ○ Protected ○ Private

□ Static □ Abstract

○ As data type ● As class

Class

Class type Consultingwerk.SmartComponents.Base.SmartDataAda Browse...

□ Extent

☑ NO-UNDO

Get

☑ Readable □ Insert implementation

Modifiers

● Public ○ Protected ○ Private

Set

☑ Writable ☑ Insert implementation

Modifiers

● Public ○ Protected ○ Private

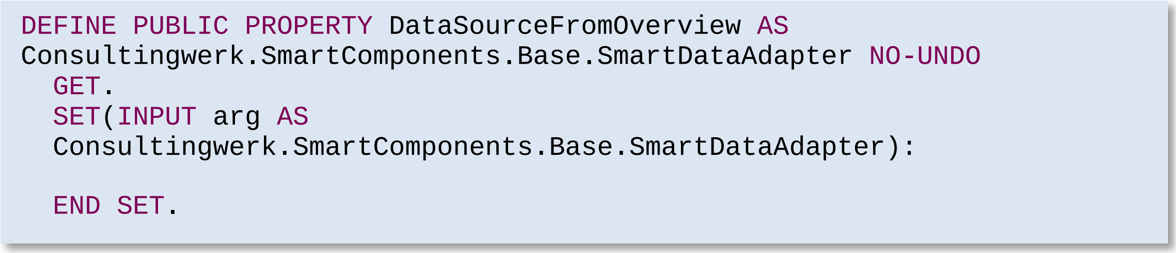
Insertion position : Alphabetical order

- Click Button **Generate**.

The new property is generated.

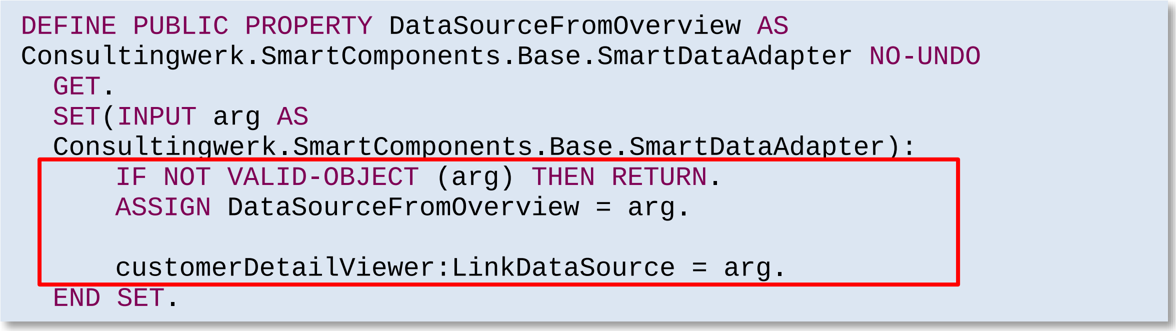
The Property SET implementation (part of the property definition) needs to be defined with an INPUT Parameter *arg* of the same type as the Property is defined.

Why is that? When a value is assigned to a Property, e.g. *PropertyName = "Test",* an internal SET method is called: SetPropertyName (INPUT arg AS CHAR). The input parameter (arg) must be assigned to the PropertyName variable. If not, the Property does not have the correct value.



```
DEFINE PUBLIC PROPERTY DataSourceFromOverview AS
Consultingwerk.SmartComponents.Base.SmartDataAdapter NO-UNDO
  GET.
  SET(INPUT arg AS
  Consultingwerk.SmartComponents.Base.SmartDataAdapter):

  END SET.
```

- Add the following code to the Property SET implementation to define a **Data Link** between the *CustomerDetailViewer* (data target) in the *Customer Detail* Form and the *SmartDataAdapter* (data source), which is the *CustomerAdapater*, in the *Customer Overview* Form.



```
DEFINE PUBLIC PROPERTY DataSourceFromOverview AS
Consultingwerk.SmartComponents.Base.SmartDataAdapter NO-UNDO
  GET.
  SET(INPUT arg AS
  Consultingwerk.SmartComponents.Base.SmartDataAdapter):
      IF NOT VALID-OBJECT (arg) THEN RETURN.
      ASSIGN DataSourceFromOverview = arg.

      customerDetailViewer:LinkDataSource = arg.
  END SET.
```

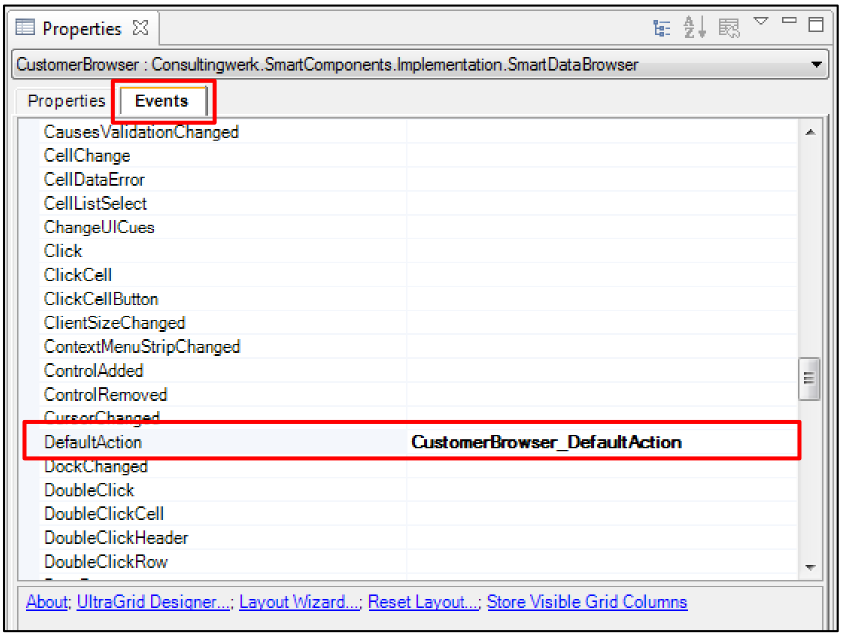
# 12. Open the Form *Customer Detail* from *Customer Overview*

In this chapter you will add code to the *Customer Overview* Form to open the *Customer Detail* Form by double-clicking a record in the browse Control and dynamically adding the link by setting a value to the property you have created in the previous chapter.

**DoubleClick** is the default action for the *SmartDataBrowser* Control. It is defined in class *Consultingwerk.SmartComponents.Implementation.**SmartDataBrowser***.

- To subscribe the *CustomerBrowser* Control to the event select the **Events** tab in the *Properties* view and double-click the event **DefaultAction** as shown below.

The event handler method **CustomerBrowser_DefaultAction** is automatically defined.


Properties
Properties
Events
About; UltraGrid Designer...; Layout Wizard...; Reset Layout...; Store Visible Grid Columns

CustomerBrowser : Consultingwerk.SmartComponents.Implementation.SmartDataBrowser

| | |
|---|---|
| CausesValidationChanged | |
| CellChange | |
| CellDataError | |
| CellListSelect | |
| ChangeUICues | |
| Click | |
| ClickCell | |
| ClickCellButton | |
| ClientSizeChanged | |
| ContextMenuStripChanged | |
| ControlAdded | |
| ControlRemoved | |
| CursorChanged | |
| DefaultAction | CustomerBrowser_DefaultAction |
| DockChanged | |
| DoubleClick | |
| DoubleClickCell | |
| DoubleClickHeader | |
| DoubleClickRow | |

## Add all necessary code for the double-click event

Add the following code in the variable section of the class file to define a reference variable:

```
(…)
DEFINE VARIABLE oCustomerDetailForm AS MyGUI.CustomerDetailForm NO-
UNDO.
(…)
```

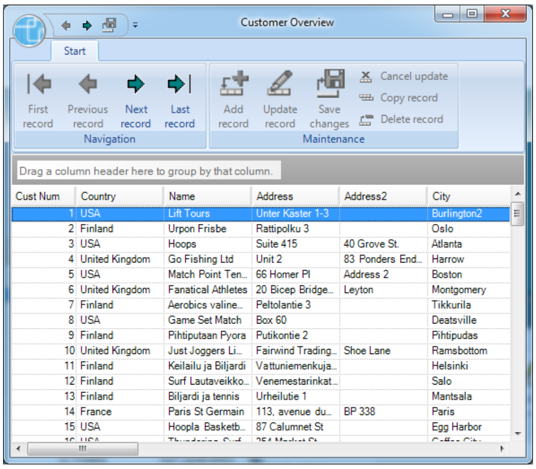
Add the following code to the *CustomerBrowser_DefaultAction* method.

- The code instantiates the class *CustomerDetailForm,*
- assigns *CustomerAdapter* as the Data Source for Form *Customer Detail*,
- shows the Form *Customer Detail*.

```
METHOD PRIVATE VOID CustomerBrowser_DefaultAction( INPUT sender AS
System.Object, INPUT e AS System.EventArgs ):

IF NOT VALID-OBJECT (oCustomerDetailForm )THEN DO:
  oCustomerDetailForm = NEW  MyGUI.CustomerDetailForm().
  oCustomerDetailForm:DataSourceFromOverview = CustomerAdapter.
  oCustomerDetailForm:show().
END.

ELSE
  oCustomerDetailForm:BringToFront().

END METHOD.
```

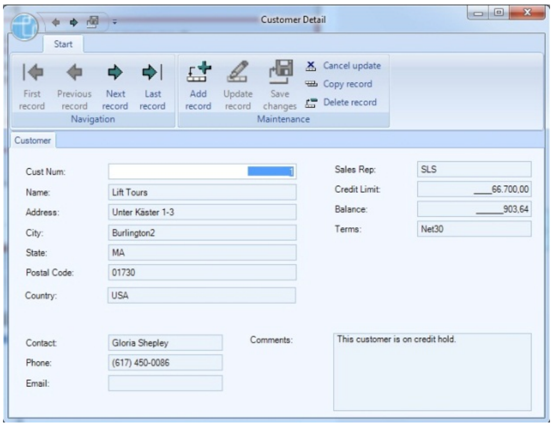
- Save and test your application!
- Double-click a customer record.

The Customer Detail dialog should open where you can update the selected record, delete it or create a new one.



Drag a column header here to group by that column.

| Cust Num | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours | Unter Käster 1-3 | | Burlington2 |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila |
| 8 | USA | Game Set Match | Box 60 | | Deatsville |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki |
| 12 | Finland | Surf Lautaveikko... | Venemestarinkat... | | Salo |
| 13 | Finland | Biljardi ja tennis | Urheilutie 1 | | Mantsala |
| 14 | France | Paris St Germain | 113, avenue du... | BP 338 | Paris |
| 15 | USA | Hoopla Basketb... | 87 Calumnet St | | Egg Harbor |



Cust Num: 1
Name: Lift Tours
Address: Unter Käster 1-3
City: Burlington2
State: MA
Postal Code: 01730
Country: USA
Sales Rep: SLS
Credit Limit: 66.700,00
Balance: 903,64
Terms: Net30
Contact: Gloria Shepley
Phone: (617) 450-0086
Email:
Comments: This customer is on credit hold.

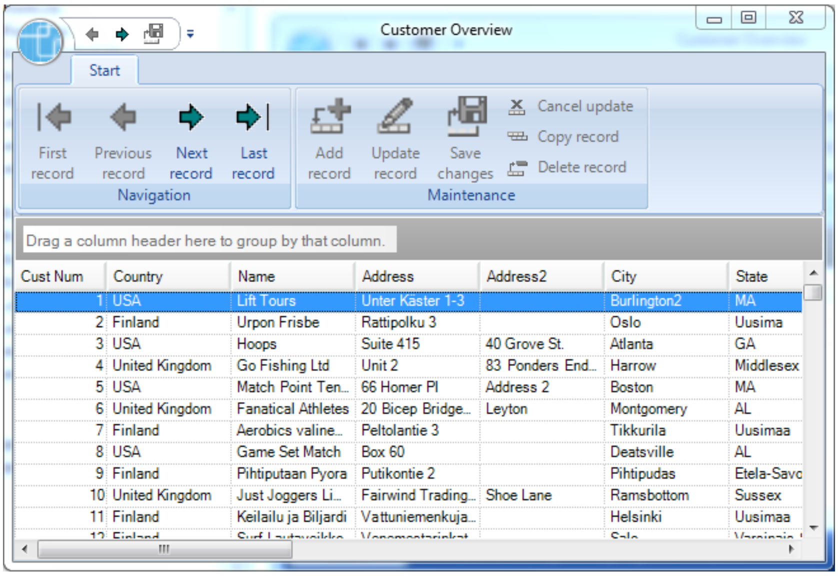
**Congratulation**! You finished the customer part of your sample application! You're going to build the order part in the next chapters.

## II. Order and Order Lines

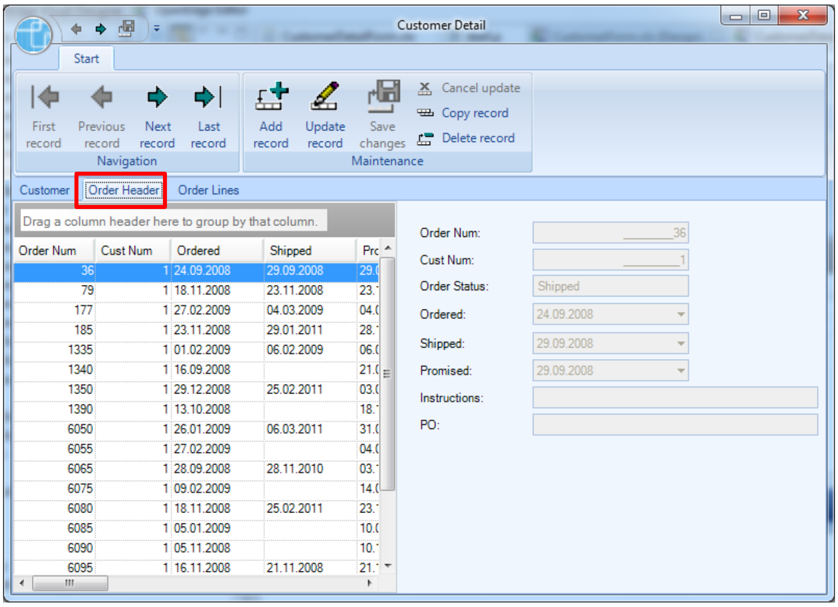
In this chapter you will extend the *Customer Detail* Form to show further information about order header and order lines. The **Order Header** page lists all orders of the selected customer in a browser and in a viewer we will provide update functionality to add new or modify existing order records.

You already know how to create browser Controls, viewer Controls and how to define the appropriate links. So the explanations in this chapter are reduced to the essentials in list form with some screenshots to assist you.

The final result should look like this:



| Cust Num | Country | Name | Address | Address2 | City | State |
|---|---|---|---|---|---|---|
| 1 | USA | Lift Tours | Unter Käster 1-3 | | Burlington2 | MA |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo | Uusima |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta | GA |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow | Middlesex |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston | MA |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery | AL |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila | Uusimaa |
| 8 | USA | Game Set Match | Box 60 | | Deatsville | AL |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas | Etela-Savo |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom | Sussex |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki | Uusimaa |



| Order Num | Cust Num | Ordered | Shipped | Pro |
|---|---|---|---|---|
| 36 | 1 | 24.09.2008 | 29.09.2008 | 29.0 |
| 79 | 1 | 18.11.2008 | 23.11.2008 | 23.1 |
| 177 | 1 | 27.02.2009 | 04.03.2009 | 04.0 |
| 185 | 1 | 23.11.2008 | 29.01.2011 | 28.1 |
| 1335 | 1 | 01.02.2009 | 06.02.2009 | 06.0 |
| 1340 | 1 | 16.09.2008 | | 21.0 |
| 1350 | 1 | 29.12.2008 | 25.02.2011 | 03.0 |
| 1390 | 1 | 13.10.2008 | | 18.1 |
| 6050 | 1 | 26.01.2009 | 06.03.2011 | 31.0 |
| 6055 | 1 | 27.02.2009 | | 04.0 |
| 6065 | 1 | 28.09.2008 | 28.11.2010 | 03.1 |
| 6075 | 1 | 09.02.2009 | | 14.0 |
| 6080 | 1 | 18.11.2008 | 25.02.2011 | 23.1 |
| 6085 | 1 | 05.01.2009 | | 10.0 |
| 6090 | 1 | 05.11.2008 | | 10.1 |
| 6095 | 1 | 16.11.2008 | 21.11.2008 | 21.1 |

Order Num: 36

Cust Num: 1

Order Status: Shipped

Ordered: 24.09.2008

Shipped: 29.09.2008

Promised: 29.09.2008

Instructions:

PO:

Follow this roadmap to build up the second part of the sample application step by step:

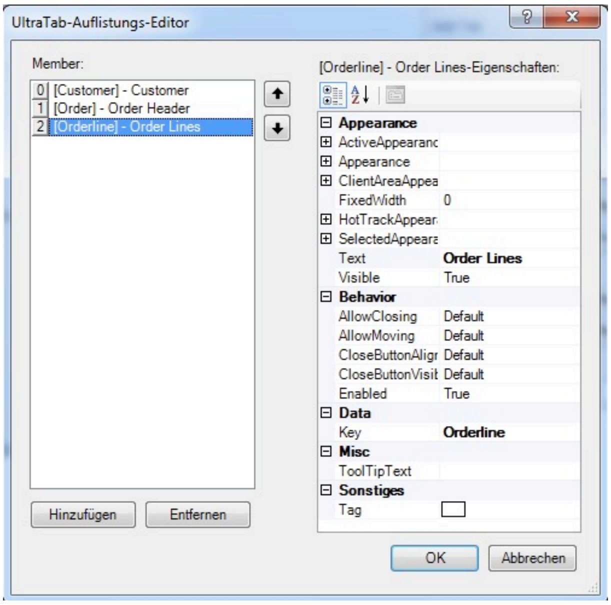
| Step | To Do | Function |
|---|---|---|
| 1 | Create a **new Tab** for order and order line | Modify the properties of the tab Control |
| 2 | Add the OrderAdapter and OrderBindingSource components | Add the DataSource object for Order and bind the DataSource object to the visualization object. |
| 3 | Add a SmartData**Browser** for Order information | Visualization of Order Header information in a browse Control |
| 4 | Customize **Code** for **Order Browser** and specify **Links** | Communication between Controls to display or update data |
| 5 | Create an **User Control** *OrderDetailViewer* | Visualization of detail Order information in a viewer |
| 6 | Add **Navigation** Functionality | Define Links |
| 7 | Add the **OrderLineAdapter** and **OrderLineSource** components | Add the DataSource object for Order Line and bind the DataSource object to the visualization object. |
| 8 | Add a SmartData**Browser** for Order Line information | Visualization of Order Line information in a browse Control |
| 9 | Create an **User Control** *OrderLineDetailViewer* | Visualization of detail Order Line information in a viewer |
| 10 | Add **Navigation** Functionality for Order Lines | Define Links |

## 1. **Create new Tabs for the Tab Control**

At first you must extend the tab Control to show additional tabs for the order header and order line information. To do so, open the Form *CustomerDetailForm.cls* and select the Control **CustomerDetailTab** you have built earlier in this tutorial.

Use the SmartTag of the Control and click **Edit Tabs**. The **UltraTab Listeditor** opens.

- Add a tab for **Order Header**
  - **Text**: Order Header
  - **Key**: Order
- Add a tab for **Order Lines**
  - **Text**: Order Lines
  - **Key**: Orderline
- Click **OK**



UltraTab-Auflistungs-Editor

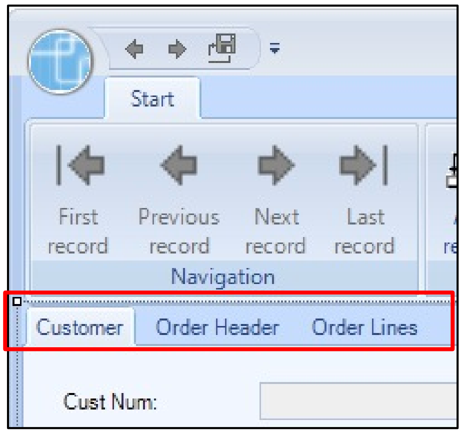
Member:

0 [Customer] - Customer<br>
1 [Order] - Order Header<br>
2 [Orderline] - Order Lines

[Orderline] - Order Lines-Eigenschaften:

| | |
|---|---|
| **Appearance** | |
| ActiveAppearanc | |
| Appearance | |
| ClientAreaAppea | |
| FixedWidth | 0 |
| HotTrackAppear | |
| SelectedAppeara | |
| Text | **Order Lines** |
| Visible | True |
| **Behavior** | |
| AllowClosing | Default |
| AllowMoving | Default |
| CloseButtonAligr | Default |
| CloseButtonVisit | Default |
| Enabled | True |
| **Data** | |
| Key | **Orderline** |
| **Misc** | |
| ToolTipText | |
| **Sonstiges** | |
| Tag | |

As a result you have two additional tabs and it should look like this:

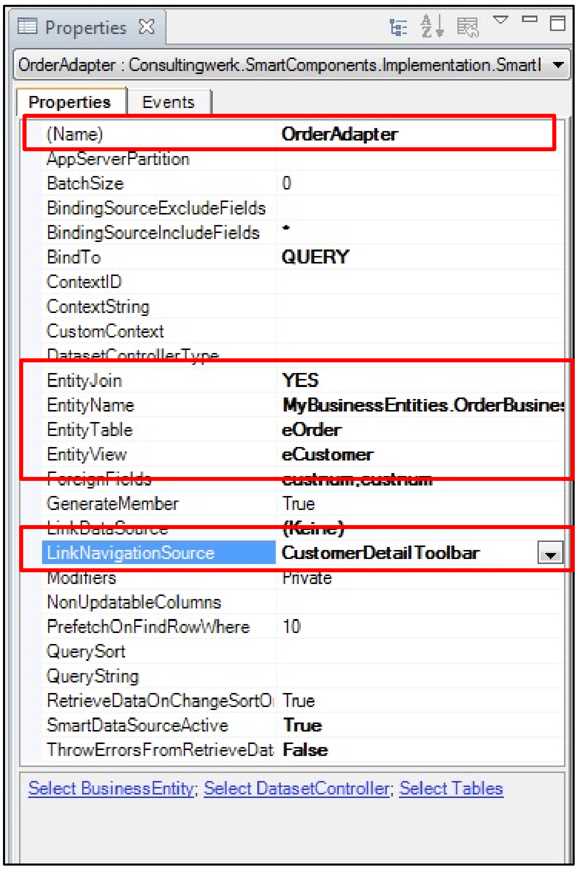


## 2. **Add the OrderAdapter and OrderBindingSource components**

In order to display data in a browser or viewer Control you need access to the *OrderBusinessEntity* you created earlier in the tutorial. To provide access to the data, you need to add a **SmartBusinessEntity Adapter** and for the data binding at runtime and design time you need to add a **SmartBusinessEntity Binding Source** component.

Add a **SmartBusinessEntityAdapter** from the *Toolbox* and configure it as shown below:

- Change Property **Name** to *OrderAdapter*.
- Use the Designer Verb **Select Business Entity** and select *OrderBusinessEntity*.
- Use the Designer Verb **Select Tables** and select *eOrder* and *eCustomer*.
- Make sure that the property **ForeignFields** is set to *custnum*, *custnum*.
- Define a **Navigation Link** from the *CustomerDetailToolbar* to the *OrderAdapter*.



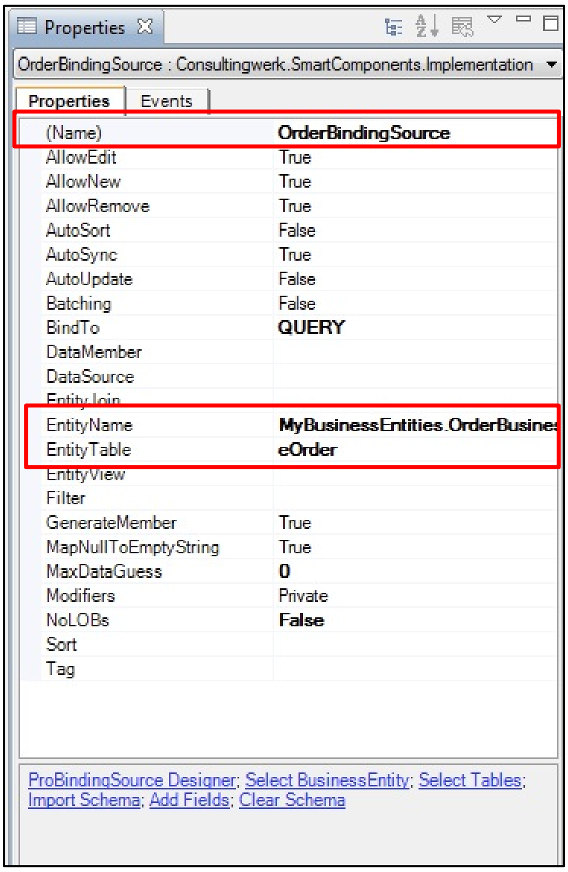
OrderAdapter : Consultingwerk.SmartComponents.Implementation.SmartI

| | |
|---|---|
| (Name) | OrderAdapter |
| AppServerPartition | |
| BatchSize | 0 |
| BindingSourceExcludeFields | |
| BindingSourceIncludeFields | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControllerType | |
| EntityJoin | YES |
| EntityName | MyBusinessEntities.OrderBusine |
| EntityTable | eOrder |
| EntityView | eCustomer |
| ForeignFields | custnum,custnum |
| GenerateMember | True |
| LinkDataSource | (Keine) |
| LinkNavigationSource | CustomerDetailToolbar |
| Modifiers | Private |
| NonUpdatableColumns | |
| PrefetchOnFindRowWhere | 10 |
| QuerySort | |
| QueryString | |
| RetrieveDataOnChangeSortO | True |
| SmartDataSourceActive | True |
| ThrowErrorsFromRetrieveDat | False |

Select BusinessEntity; Select DatasetController; Select Tables

Add a **SmartBusinessEntityBindingSource** from the *Toolbox* and configure it as shown below:

- Close the *ProBindingSource Designer*.
- Change Property **Name** to *OrderBindingSource***.**
- Use the Designer Verb **Select Business Entity** to select *OrderBusinessEntity*.
- Use the Designer Verb **Select Tables** to select *eOrder* only.
- Click the Designer Verb **Import Schema**.



OrderBindingSource : Consultingwerk.SmartComponents.Implementation

| | |
|---|---|
| (Name) | **OrderBindingSource** |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |
| BindTo | **QUERY** |
| DataMember | |
| DataSource | |
| EntityJoin | |
| EntityName | **MyBusinessEntities.OrderBusines** |
| EntityTable | **eOrder** |
| EntityView | |
| Filter | |
| GenerateMember | True |
| MapNullToEmptyString | True |
| MaxDataGuess | **0** |
| Modifiers | Private |
| NoLOBs | **False** |
| Sort | |
| Tag | |

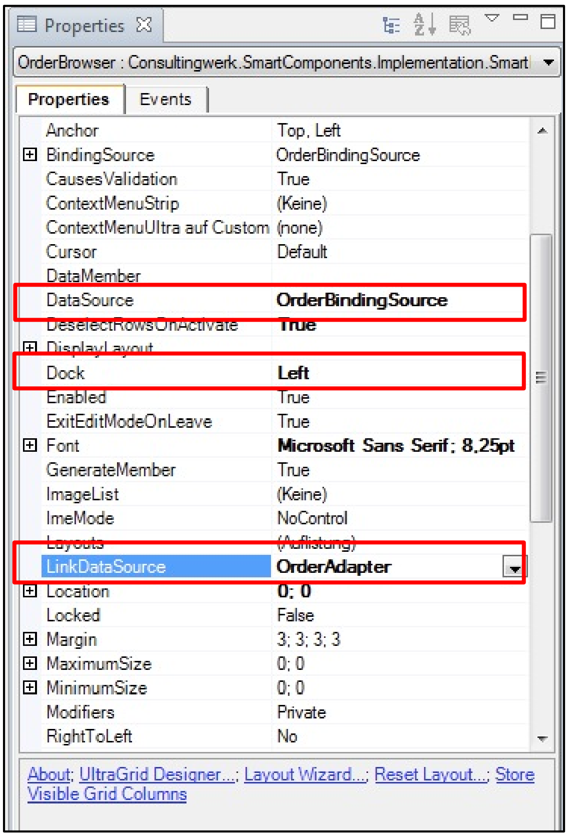
ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

# 3. **Add a SmartDataBrowser for Order information**

You are now ready to add a **SmartDataBrowser** visualization Control to the Order Header tab of *CustomerDetailForm* to show data.

Add a new *SmartDataBrowser* Control from the **Toolbox** and configure it as shown below:

- Switch to Tab *Order Header*. You can select the Control from the properties drop down list or simply click on it.
- Add a new *SmartDataBrowser* Control. The *UltraWinGrid QuickStart dialog* opens.
- Select *Bind the Control to an existing DataSource now*.
- Select *OrderBindingSource* from the drop down list.
- Click **Finish.**
- Change Property **Name** to *OrderBrowser*.
- Change Property **Dock** to *Left*.
- Define a **Data Link** from the *OrderAdapter* to the *OrderBrowser*.

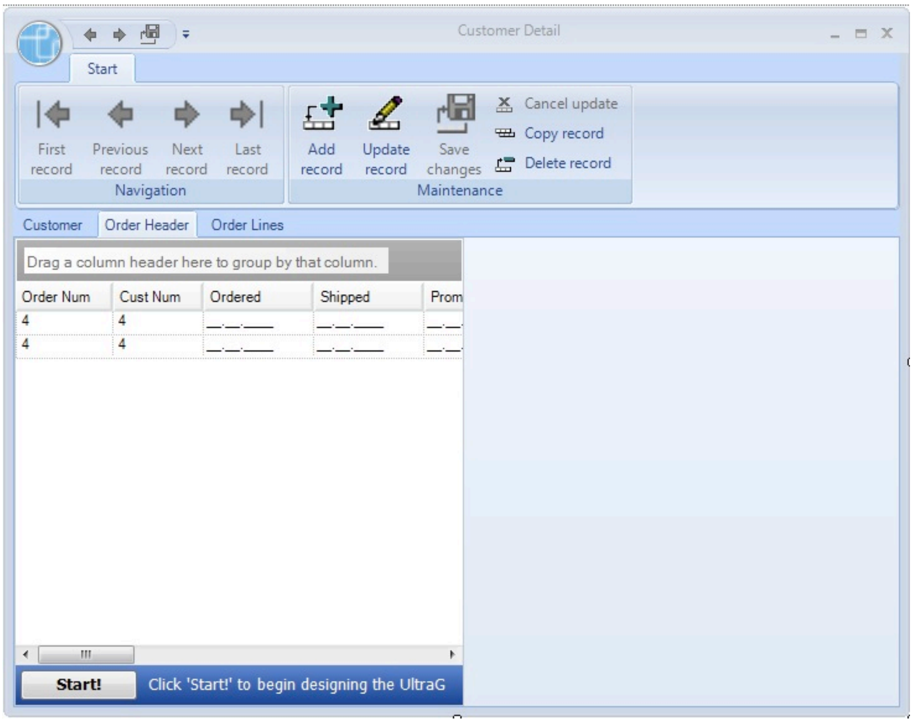


OrderBrowser : Consultingwerk.SmartComponents.Implementation.Smart

| | |
|---|---|
| Anchor | Top, Left |
| ⊞ BindingSource | OrderBindingSource |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| ContextMenuUltra auf Custom | (none) |
| Cursor | Default |
| DataMember | |
| DataSource | **OrderBindingSource** |
| DeselectRowsOnActivate | **True** |
| ⊞ DisplayLayout | |
| Dock | **Left** |
| Enabled | True |
| ExitEditModeOnLeave | True |
| ⊞ Font | **Microsoft Sans Serif; 8,25pt** |
| GenerateMember | True |
| ImageList | (Keine) |
| ImeMode | NoControl |
| Layouts | (Auflistung) |
| LinkDataSource | **OrderAdapter** |
| ⊞ Location | **0; 0** |
| Locked | False |
| ⊞ Margin | 3; 3; 3; 3 |
| ⊞ MaximumSize | 0; 0 |
| ⊞ MinimumSize | 0; 0 |
| Modifiers | Private |
| RightToLeft | No |

About; UltraGrid Designer...; Layout Wizard...; Reset Layout...; Store Visible Grid Columns

The result should look like this:



| Order Num | Cust Num | Ordered | Shipped | Prom |
|---|---|---|---|---|
| 4 | 4 | __.__.____ | __.__.____ | __.__ |
| 4 | 4 | __.__.____ | __.__.____ | __.__ |

When you save and test your application so far you will recognize, that the *OrderBrowser* Control doesn’t show any data for the selected customer. You need some code to link the *SmartBusinessEntityAdapter* for the *OrderBusinessEntity* to the Business Entity for customers of the *CustomerOverview* Form you created earlier in this tutorial.

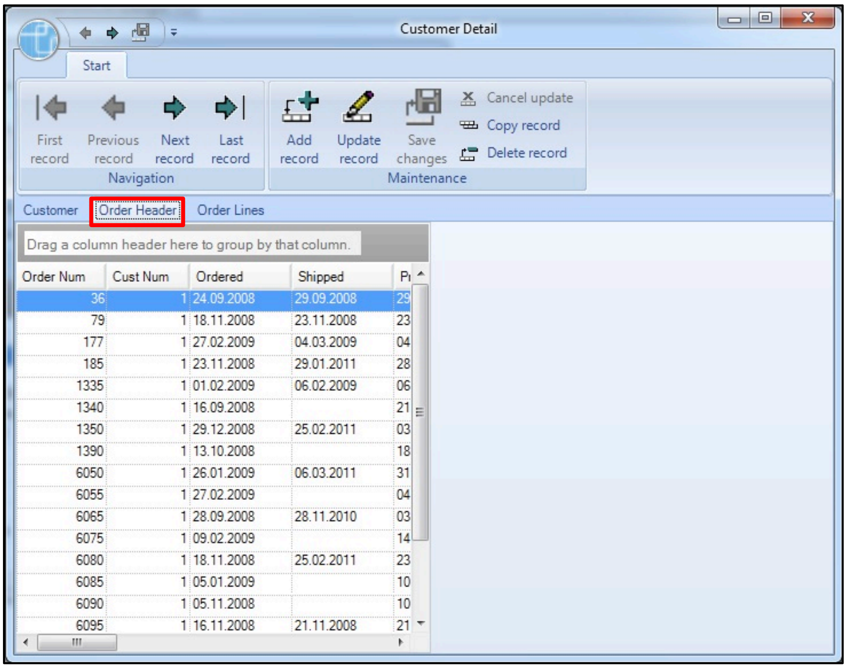
## 4. **Customize code for Order Browser**

To show related order data for selected customer records you must register the *OrderAdapter* as a data source for the *CustomerDetail* Form. In the Form *Customer Detail,* you have already defined a property *DataSourceFromOverview* that provides a Data Link between the *CustomerDetailViewer* and the *CustomerAdapter* in the Form *Customer Overview* (pass through link).

- Open the file *CustomerDetailForm.cls* in source view.
- Locate the source code of the Property **DataSourceFromOverview**
- Add the code below.
- Save and test you application.

```
DEFINE PUBLIC PROPERTY DataSourceFromOverview AS
Consultingwerk.SmartComponents.Base.SmartDataAdapter NO-UNDO
  GET.
  SET(INPUT arg AS
  Consultingwerk.SmartComponents.Base.SmartDataAdapter):
      IF NOT VALID-OBJECT (arg) THEN RETURN.
      ASSIGN DataSourceFromOverview = arg.

      customerDetailViewer:LinkDataSource = arg.

/* Order */
  THIS-OBJECT:OrderAdapter:SmartDataSource = arg.

  END SET.
```

At runtime the **Customer Detail** Form on the *Order Header Tab* should look like this:



| Order Num | Cust Num | Ordered | Shipped | Pı |
|---|---|---|---|---|
| 36 | 1 | 24.09.2008 | 29.09.2008 | 29 |
| 79 | 1 | 18.11.2008 | 23.11.2008 | 23 |
| 177 | 1 | 27.02.2009 | 04.03.2009 | 04 |
| 185 | 1 | 23.11.2008 | 29.01.2011 | 28 |
| 1335 | 1 | 01.02.2009 | 06.02.2009 | 06 |
| 1340 | 1 | 16.09.2008 | | 21 |
| 1350 | 1 | 29.12.2008 | 25.02.2011 | 03 |
| 1390 | 1 | 13.10.2008 | | 18 |
| 6050 | 1 | 26.01.2009 | 06.03.2011 | 31 |
| 6055 | 1 | 27.02.2009 | | 04 |
| 6065 | 1 | 28.09.2008 | 28.11.2010 | 03 |
| 6075 | 1 | 09.02.2009 | | 14 |
| 6080 | 1 | 18.11.2008 | 25.02.2011 | 23 |
| 6085 | 1 | 05.01.2009 | | 10 |
| 6090 | 1 | 05.11.2008 | | 10 |
| 6095 | 1 | 16.11.2008 | 21.11.2008 | 21 |

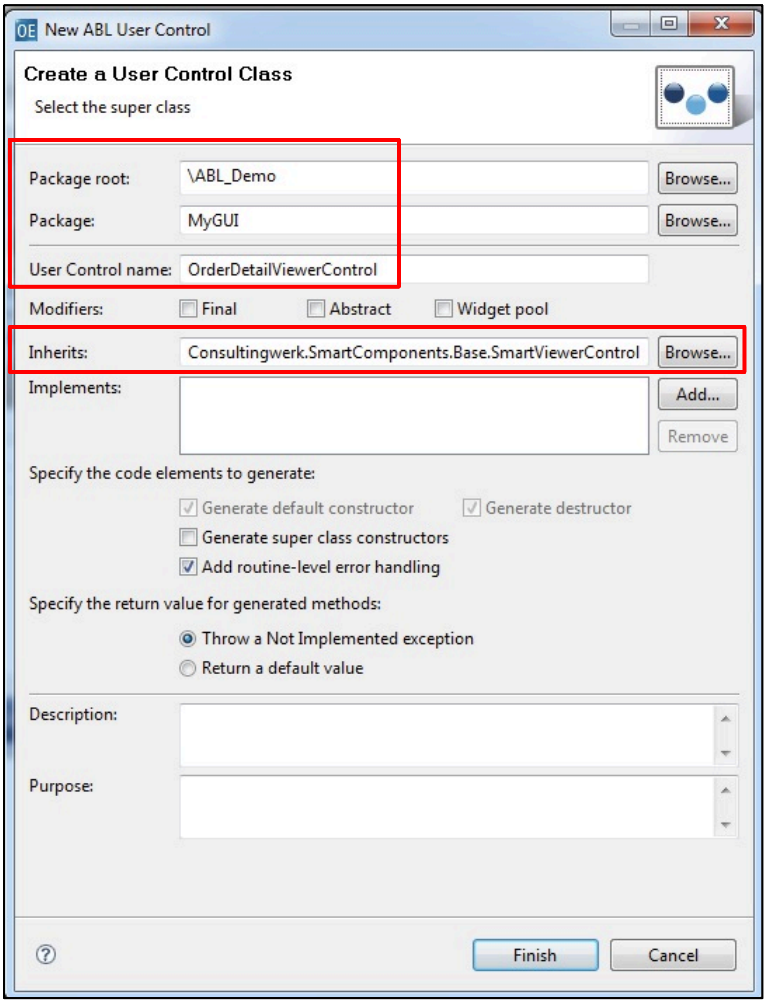
## 5. Create a User Control OrderDetailViewer

In this chapter you create a new User Control to show detailed order information and to provide update functionality for order records in a Viewer Control.

For more information about how to create User Controls refer to chapter 9. Create a User Control CustomerDetailViewer.

Create a new ABL User Control and use the following settings in the New ABL User Control wizard as shown below:

- **Package root**: your project folder
- **Package**: your GUI folder
- **User Control name**: *OrderDetailViewerControl*
- **Inherits**: *SmartViewerControl*



New ABL User Control

Create a User Control Class

Select the super class

Package root: \ABL_Demo

Package: MyGUI

User Control name: OrderDetailViewerControl

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Consultingwerk.SmartComponents.Base.SmartViewerControl

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

☑ Add routine-level error handling

Specify the return value for generated methods:

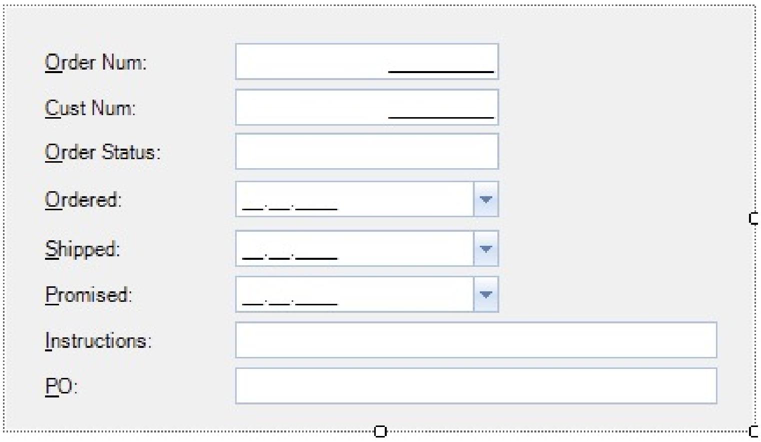
● Throw a Not Implemented exception

○ Return a default value

Description:

Purpose:

- Copy the *OrderBindingSource* from Form *Customer Detail* and paste it to your new User Control. This will simplify your task as you do not have to select the Business Entity and tables again.
- With the *OrderBindingSource* component selected click the Designer Verb **Add Fields** and select the fields shown below and order them in a similar fashion.



Order Num:

Cust Num:

Order Status:

Ordered: __.__.____

Shipped: __.__.____

Promised: __.__.____

Instructions:

PO:

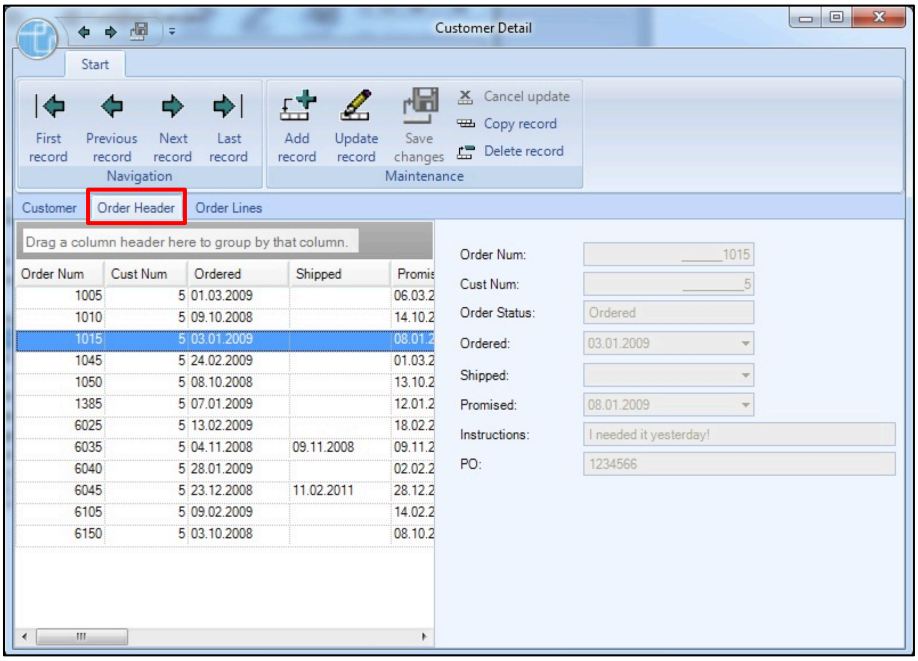
- Select all Fill-Ins and change Property **DisplayStyle** to *Office2007* and save your viewer Control.

## Insert the User Control into the Form

To insert the *OrderDetail* viewer to the *Order Header* tab do the following:

- Open the file *CustomerDetailForm.cls*.
- Switch to the Tab *Order Header* and make sure that the Form is selected.
- Select the Designer Verb **Insert UserControl**.
  - The *Select UserControl class dialog* opens.
  - From the list select *OrderDetailViewerControl*.
- Click **OK**.
- A message appears to inform you that the UserControl might have been created in the background.
  - If you cannot see the Control on the design surface, use the Outline View to locate it and bring it to the front.
- Click **OK**.
- Drag the UserControl to the right side of the Form.
- Change Property **Dock** to *Fill***.**
- Set Property **LinkDataSource** to *OrderAdapter*.
- Set Property **LinkTableIOSource** to *CustomerDetailToolbar*.
- Save and test you application.

At runtime the *Order Header* tab of the **Customer Detail** Form should look like this:



Drag a column header here to group by that column.

| Order Num | Cust Num | Ordered | Shipped | Promis |
|---|---|---|---|---|
| 1005 | 5 | 01.03.2009 | | 06.03.2 |
| 1010 | 5 | 09.10.2008 | | 14.10.2 |
| 1015 | 5 | 03.01.2009 | | 08.01.2 |
| 1045 | 5 | 24.02.2009 | | 01.03.2 |
| 1050 | 5 | 08.10.2008 | | 13.10.2 |
| 1385 | 5 | 07.01.2009 | | 12.01.2 |
| 6025 | 5 | 13.02.2009 | | 18.02.2 |
| 6035 | 5 | 04.11.2008 | 09.11.2008 | 09.11.2 |
| 6040 | 5 | 28.01.2009 | | 02.02.2 |
| 6045 | 5 | 23.12.2008 | 11.02.2011 | 28.12.2 |
| 6105 | 5 | 09.02.2009 | | 14.02.2 |
| 6150 | 5 | 03.10.2008 | | 08.10.2 |

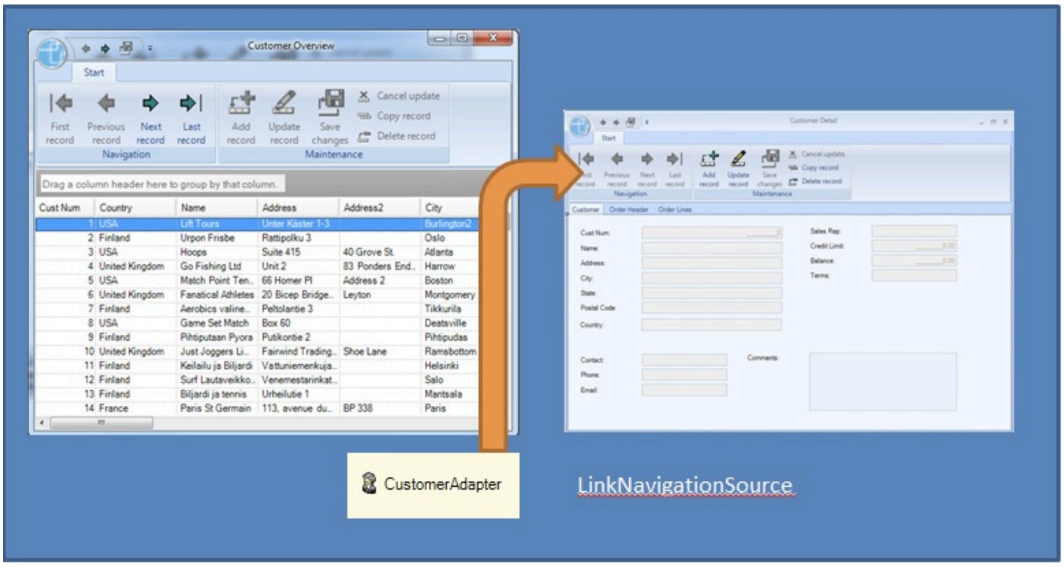
Order Num: 1015
Cust Num: 5
Order Status: Ordered
Ordered: 03.01.2009
Shipped:
Promised: 08.01.2009
Instructions: I needed it yesterday!
PO: 1234566

# 6. Add Navigation Functionality

So far the Toolbar *CustomerDetailToolbar* in Form *Customer Detail* is ready for update purposes like add, delete or update but not yet for any kind of navigation. When you have finished this chapter you will be able to navigate through the records depending on which tab you selected.

## Navigating the Customer records

We start with the customer records. First of all you need a Navigation Link between the *CustomerAdapter* component as the Navigation-Target and the *CustomerDetailToolbar* Control as the Navigation-Source. As you already know you must provide inter-form-communication manually because the encapsulation prohibits any predefined link option between objects contained on two different forms.

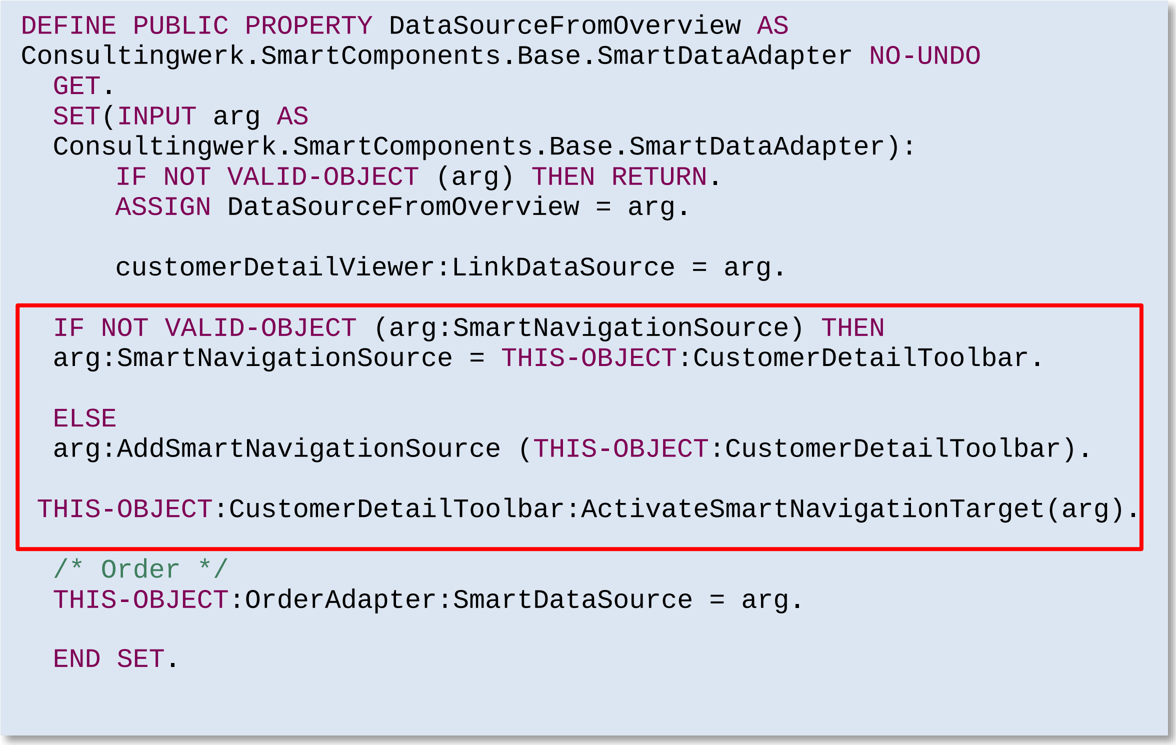


| Cust Num | Country | Name | Address | Address2 | City |
|---|---|---|---|---|---|
| 1 | USA | Lift Tours | Unter Käster 1-3 | | Burlington2 |
| 2 | Finland | Urpon Frisbe | Rattipolku 3 | | Oslo |
| 3 | USA | Hoops | Suite 415 | 40 Grove St. | Atlanta |
| 4 | United Kingdom | Go Fishing Ltd | Unit 2 | 83 Ponders End... | Harrow |
| 5 | USA | Match Point Ten... | 66 Homer Pl | Address 2 | Boston |
| 6 | United Kingdom | Fanatical Athletes | 20 Bicep Bridge... | Leyton | Montgomery |
| 7 | Finland | Aerobics valine... | Peltolantie 3 | | Tikkurila |
| 8 | USA | Game Set Match | Box 60 | | Deatsville |
| 9 | Finland | Pihtiputaan Pyora | Putikontie 2 | | Pihtipudas |
| 10 | United Kingdom | Just Joggers Li... | Fairwind Trading... | Shoe Lane | Ramsbottom |
| 11 | Finland | Keilailu ja Biljardi | Vattuniemenkuja... | | Helsinki |
| 12 | Finland | Surf Lautaveikko... | Venemestarinkat... | | Salo |
| 13 | Finland | Biljardi ja tennis | Urheilutie 1 | | Mantsala |
| 14 | France | Paris St Germain | 113, avenue du... | BP 338 | Paris |

- Open the file *CustomerDetailForm.cls* in source code view.
- Go to the Property **DataSourceFromOverview** and add the code shown inside the red box.

This code checks whether the passed in *SmartDataAdapter* has a valid *SmartNavigationSource*. If not the *SmartNavigationSource* is set to *CustomerDetailToolbar*.

If there is already a valid *SmartNavigationSource* we can register another one by calling the method **AddSmartNavigationSource ()** of the *SmartDataAdapter* and pass it to the *SmartToolbarController* instance.



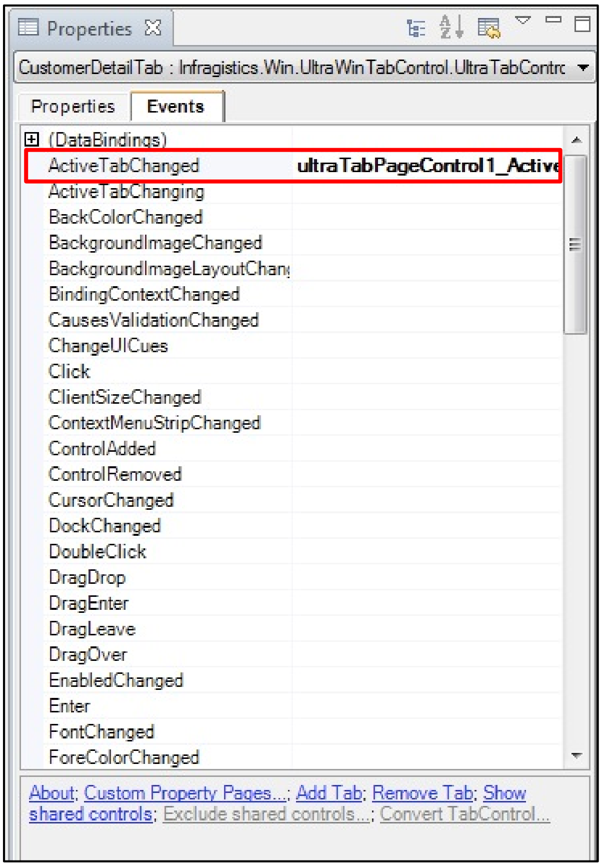
```
DEFINE PUBLIC PROPERTY DataSourceFromOverview AS
Consultingwerk.SmartComponents.Base.SmartDataAdapter NO-UNDO
  GET.
  SET(INPUT arg AS
  Consultingwerk.SmartComponents.Base.SmartDataAdapter):
      IF NOT VALID-OBJECT (arg) THEN RETURN.
      ASSIGN DataSourceFromOverview = arg.

      customerDetailViewer:LinkDataSource = arg.

  IF NOT VALID-OBJECT (arg:SmartNavigationSource) THEN
  arg:SmartNavigationSource = THIS-OBJECT:CustomerDetailToolbar.

  ELSE
  arg:AddSmartNavigationSource (THIS-OBJECT:CustomerDetailToolbar).

 THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget(arg).
  /* Order */
  THIS-OBJECT:OrderAdapter:SmartDataSource = arg.

  END SET.
```

- Save and test you application.

## Navigating the Order records

So far you implemented navigation functionality for the *Customer* tab, so that it actually navigates the record of the *Customer Overview* Form. When you switch to the *Order Header* tab you'll find out that it navigates through the customer records as well and not through the depending order records. What you will have to do is to write an event handler method, that checks which tab is active and then activates all the links from the toolbar component for the tab page and disables links you do not need in that context.

- To add a suitable event to the Form, double-click the event **ActiveTabChanged** for the Control *CustomerDetailTab*.



CustomerDetailTab : Infragistics.Win.UltraWinTabControl.UltraTabContrc

|  |  |
|---|---|
| ⊞ (DataBindings) |  |
| ActiveTabChanged | ultraTabPageControl1_Active |
| ActiveTabChanging |  |
| BackColorChanged |  |
| BackgroundImageChanged |  |
| BackgroundImageLayoutChang |  |
| BindingContextChanged |  |
| CausesValidationChanged |  |
| ChangeUICues |  |
| Click |  |
| ClientSizeChanged |  |
| ContextMenuStripChanged |  |
| ControlAdded |  |
| ControlRemoved |  |
| CursorChanged |  |
| DockChanged |  |
| DoubleClick |  |
| DragDrop |  |
| DragEnter |  |
| DragLeave |  |
| DragOver |  |
| EnabledChanged |  |
| Enter |  |
| FontChanged |  |
| ForeColorChanged |  |

About; Custom Property Pages...; Add Tab; Remove Tab; Show shared controls; Exclude shared controls...; Convert TabControl...

This generates an Event Handler Method **CustomerDetailTab_ActiveTabChanged**.

Open the file *CustomerDetailForm.cls* in source code view.

Got to the method **CustomerDetailTab_ActiveTabChanged** and write the code below:

```
METHOD PRIVATE VOID CustomerDetailTab_ActiveTabChanged
 ( INPUT sender AS System.Object, INPUT e AS
  Infragistics.Win.UltraWinTabControl.ActiveTabChangedEventArgs ):

CASE e:Tab:Key:
  WHEN "Customer":U THEN DO:
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget
    (DataSourceFromOverview).
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartTableIOTarget
    (THIS-OBJECT:customerDetailViewer).

   OrderAdapter:SmartDataSourceActive = FALSE.
  /*  OrderLineAdapter:SmartDataSourceActive = FALSE.*/
  END.

  WHEN "Order":U THEN DO:
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget
    (THIS-OBJECT:OrderAdapter).
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartTableIOTarget
     (THIS-OBJECT:orderDetailViewer).

   OrderAdapter:SmartDataSourceActive = TRUE.
   /*  OrderLineAdapter:SmartDataSourceActive = FALSE.*/
  END.

END CASE.

END METHOD.
```

The code does the following:

- checks which tab is selected after the user has changed the current tab page,
- activates the necessary *Navigation* and *TableIO* link from the toolbar,
- deactivates the *DataAdapters* which are not needed because the data of those is not currently shown on the active tab.

To identify the Tab you use the Tab’s Key property you defined earlier in this tutorial as this provides a unique name.

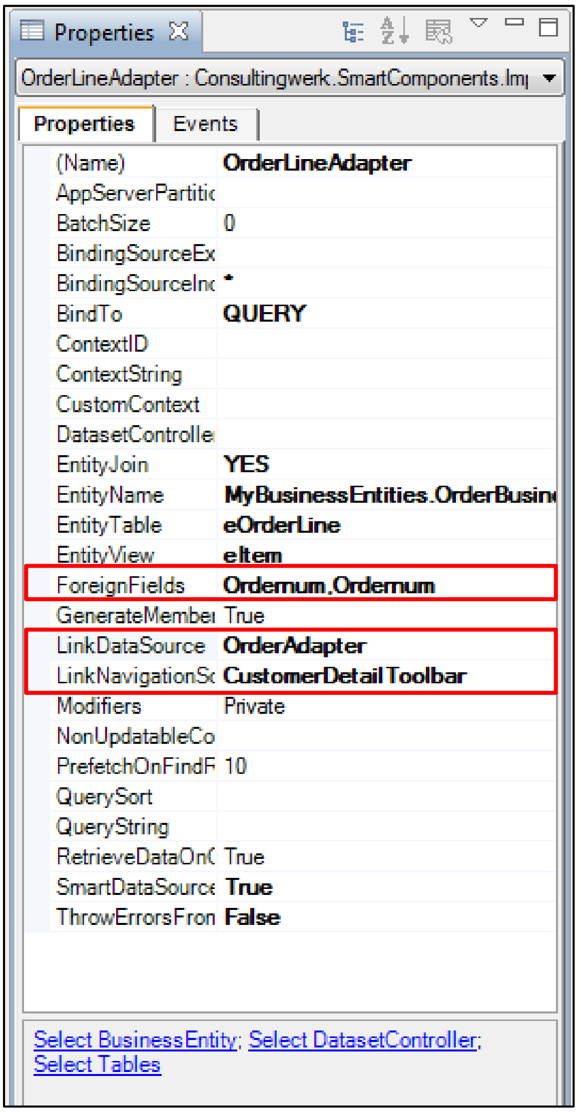
You can add the code for the *OrderLineAdapter* if you want to as a comment; right now it may cause compilation errors, but you will need it soon.

# 7. Add the OrderLineAdapter and OrderLineBindingSource

In order to display data in a browser or viewer Control you need access to the Orderline records from the ***Order****BusinessEntity* you created earlier in the tutorial. To provide this access to the data, you need a **SmartBusinessEntity Adapter** and for runtime schema you must add a **SmartBusinessEntity Binding Source**.

Add a **SmartBusinessEntityAdapter** from the *Toolbox* and configure it as shown below:

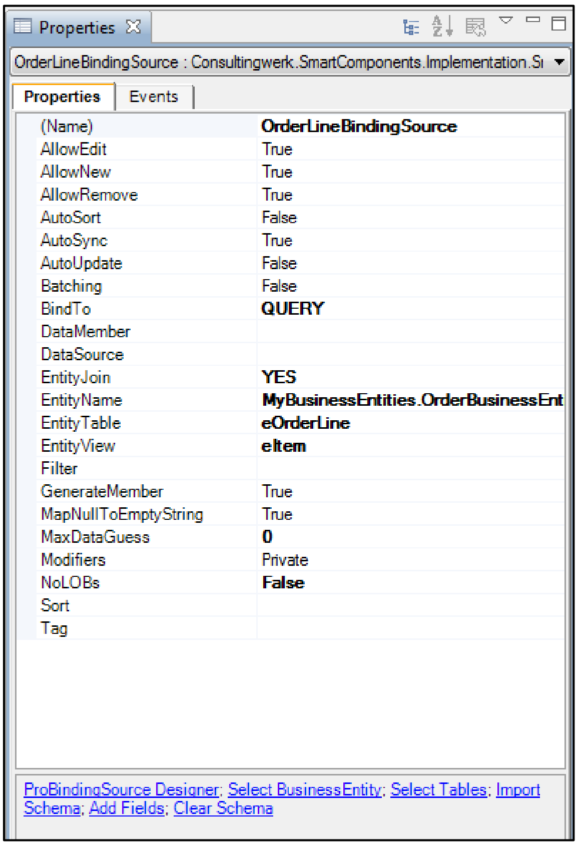
- Change Property **Name** to *OrderLineAdapter*.
- Use the Designer Verb **Select Business Entity** and select *OrderBusinessEntity*.
- Use the Designer Verb **Select Tables** and select *eOrderline* and *eItem*.
- **Join** *eOrderline and eItem*.
- Make sure that the **Foreign Fields** are set to *ordernum*, *ordernum*.
- Define a **DataLink** from the *OrderAdapter* to the *OrderLineAdapter*.
- Define a **Navigation Link** from *CustomerDetailToolbar* to *OrderLineAdapter.*



| | |
|---|---|
| (Name) | OrderLineAdapter |
| AppServerPartitic | |
| BatchSize | 0 |
| BindingSourceEx | |
| BindingSourceInc | * |
| BindTo | QUERY |
| ContextID | |
| ContextString | |
| CustomContext | |
| DatasetControlle | |
| EntityJoin | YES |
| EntityName | MyBusinessEntities.OrderBusin |
| EntityTable | eOrderLine |
| EntityView | eItem |
| ForeignFields | Ordernum,Ordernum |
| GenerateMembei | True |
| LinkDataSource | OrderAdapter |
| LinkNavigationSc | CustomerDetailToolbar |
| Modifiers | Private |
| NonUpdatableCo | |
| PrefetchOnFindF | 10 |
| QuerySort | |
| QueryString | |
| RetrieveDataOn( | True |
| SmartDataSourc | True |
| ThrowErrorsFron | False |

Add a **SmartBusinessEntityBindingSource** from the Toolbox and configure it as shown below:

- Close the *ProBindingSource* Designer.
- Change Property **Name** to *OrderLineBindingSource*.
- Use the Designer Verb **Select Business Entity** to select *OrderBusinessEntity*.
- Use the Designer Verb **Select Tables** to select *eOrderLine* and *eItem*.
- Click the Designer Verb **Import Schema**.



Properties

OrderLineBindingSource : Consultingwerk.SmartComponents.Implementation.Si

Properties | Events

| | |
|---|---|
| (Name) | OrderLineBindingSource |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |
| BindTo | QUERY |
| DataMember | |
| DataSource | |
| EntityJoin | YES |
| EntityName | MyBusinessEntities.OrderBusinessEnt |
| EntityTable | eOrderLine |
| EntityView | eItem |
| Filter | |
| GenerateMember | True |
| MapNullToEmptyString | True |
| MaxDataGuess | 0 |
| Modifiers | Private |
| NoLOBs | False |
| Sort | |
| Tag | |

ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

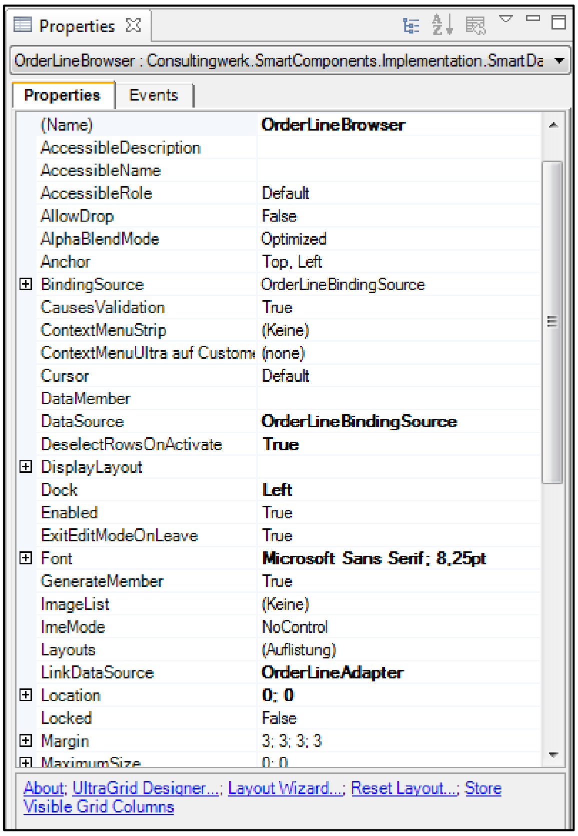
## 8. Add a SmartDataBrowser for Order Line information

You are now ready to add a **SmartDataBrowser** visualization Control to the Order Line tab of *CustomerDetailForm* to show data.

Add a new *SmartDataBrowser* Control from the Toolbox and configure it as shown below:

- Switch to Tab *Order Line*. You can select the Control from the properties drop down list or simply click it.
- Add a new *SmartDataBrowser* Control. The **UltraWinGrid QuickStart** dialog opens.
- Select *Bind the Control to an existing DataSource now*.
- Select *OrderLineBindingSource* from the drop down list.
- Click **Finish.**

- Change Property **Name** to *OrderLineBrowser*.
- Change Property **Dock** to *Left*.
- Define a **Data Link** from the *OrderLineAdapter* to the *OrderLineBrowser*.

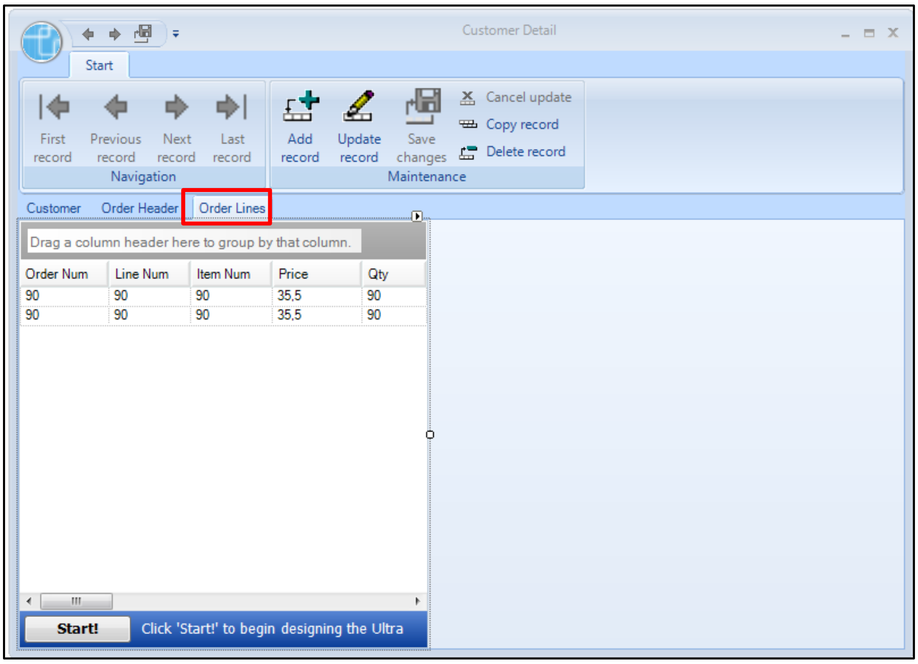


Properties
OrderLineBrowser : Consultingwerk.SmartComponents.Implementation.SmartDa
Properties
Events

| | |
|---|---|
| (Name) | **OrderLineBrowser** |
| AccessibleDescription | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| AlphaBlendMode | Optimized |
| Anchor | Top, Left |
| ⊞ BindingSource | OrderLineBindingSource |
| CausesValidation | True |
| ContextMenuStrip | (Keine) |
| ContextMenuUltra auf Custom | (none) |
| Cursor | Default |
| DataMember | |
| DataSource | **OrderLineBindingSource** |
| DeselectRowsOnActivate | **True** |
| ⊞ DisplayLayout | |
| Dock | **Left** |
| Enabled | True |
| ExitEditModeOnLeave | True |
| ⊞ Font | **Microsoft Sans Serif; 8,25pt** |
| GenerateMember | True |
| ImageList | (Keine) |
| ImeMode | NoControl |
| Layouts | (Auflistung) |
| LinkDataSource | **OrderLineAdapter** |
| ⊞ Location | **0; 0** |
| Locked | False |
| ⊞ Margin | 3; 3; 3; 3 |
| ⊞ MaximumSize | 0; 0 |

About; UltraGrid Designer...; Layout Wizard...; Reset Layout...; Store Visible Grid Columns

The result should look like this:



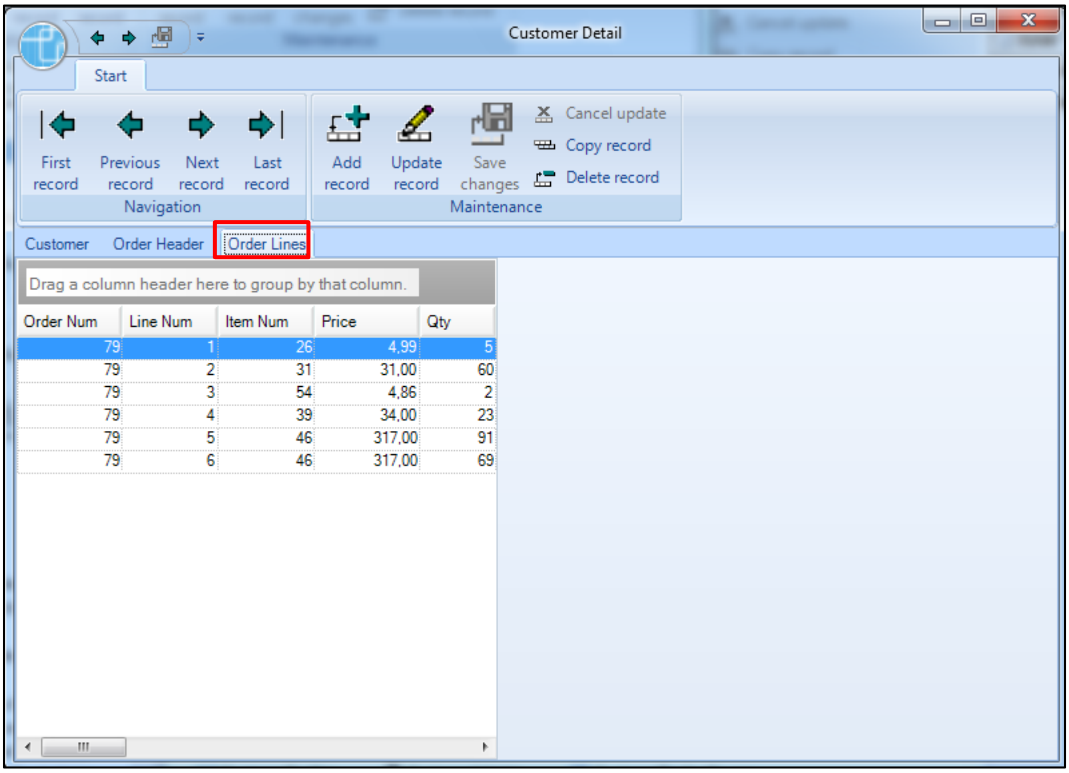
Drag a column header here to group by that column.

| Order Num | Line Num | Item Num | Price | Qty |
|---|---|---|---|---|
| 90 | 90 | 90 | 35,5 | 90 |
| 90 | 90 | 90 | 35,5 | 90 |

You don’t need any coding here because you defined a DataLink between *OrderLineAdapter* and *OrderAdapter* in chapter 7. Add the OrderLineAdapter and OrderLineBindingSource.

The code we added in the former task was required for linking Controls and Components that are placed within different Forms. The *OrderAdapter* and the *OrderLineAdapter* are located in the same Form and thus you can use the *LinkDataSource* property on the target *SmartBusinessEntityAdapter* to add the link.

At runtime the Order Lines Tab for Customer 1, order 70 of **Customer Detail** Form should look like this:



| Order Num | Line Num | Item Num | Price | Qty |
|---|---|---|---|---|
| 79 | 1 | 26 | 4,99 | 5 |
| 79 | 2 | 31 | 31,00 | 60 |
| 79 | 3 | 54 | 4,86 | 2 |
| 79 | 4 | 39 | 34,00 | 23 |
| 79 | 5 | 46 | 317,00 | 91 |
| 79 | 6 | 46 | 317,00 | 69 |

Note, that the actual data displayed may differ depending on the current data in your database.

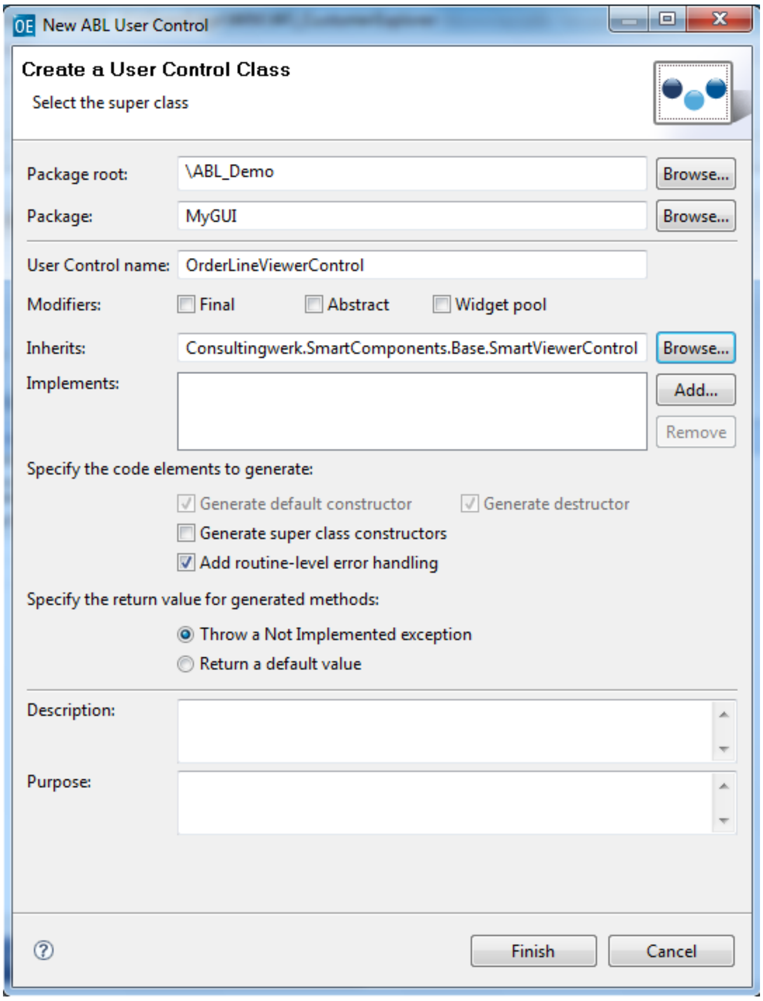
# 9. Create a User Control OrderLineDetailViewer

In this chapter you will create a new User Control to show detailed order line information and to provide update functionality for order line records in a Viewer Control.

For more information about how to create a *SmartViewerControl* User Controls refer to chapter 9. Create a User Control CustomerDetailViewer.

Create a new ABL User Control and configure it as shown below:

- **Package root**: your project folder
- **Package**: your folder for the GUI objects
- **User Control name**: *OrderLineDetailViewerControl*
- **Inherits**: *SmartViewerControl (Consultingwerk.SmartComponents.Base.SmartViewerControl)*
- **Add routine-level error-handling:** Checked



New ABL User Control

Create a User Control Class

Select the super class

Package root: \ABL_Demo

Package: MyGUI

User Control name: OrderLineViewerControl

Modifiers: □ Final □ Abstract □ Widget pool

Inherits: Consultingwerk.SmartComponents.Base.SmartViewerControl

Implements:

Specify the code elements to generate:

☑ Generate default constructor ☑ Generate destructor

□ Generate super class constructors

☑ Add routine-level error handling

Specify the return value for generated methods:

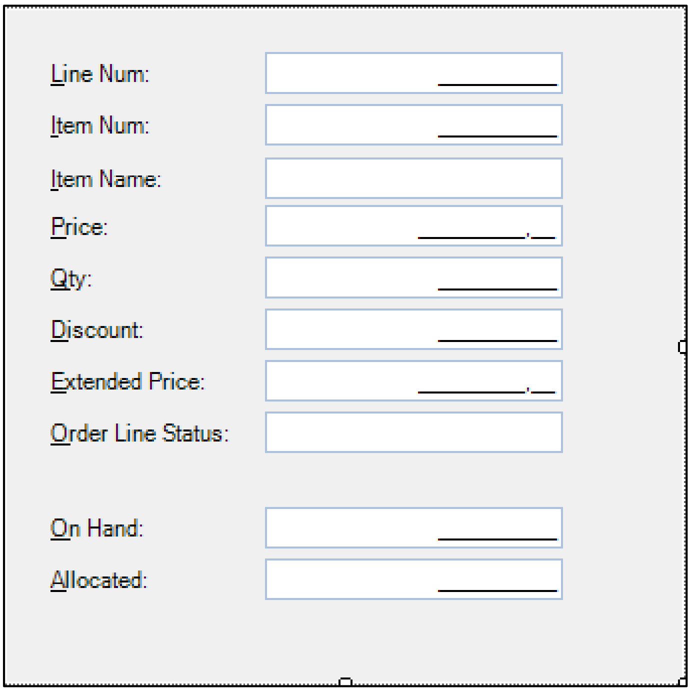
● Throw a Not Implemented exception

○ Return a default value

Description:

Purpose:

- Copy the *OrderLineBindingSource* from the *Customer Detail* Form and paste it to your new User Control (using the system clipboard).
- With the *OrderLineBindingSource* component selected click Designer Verb **Add Fields** and build the viewer as shown below.



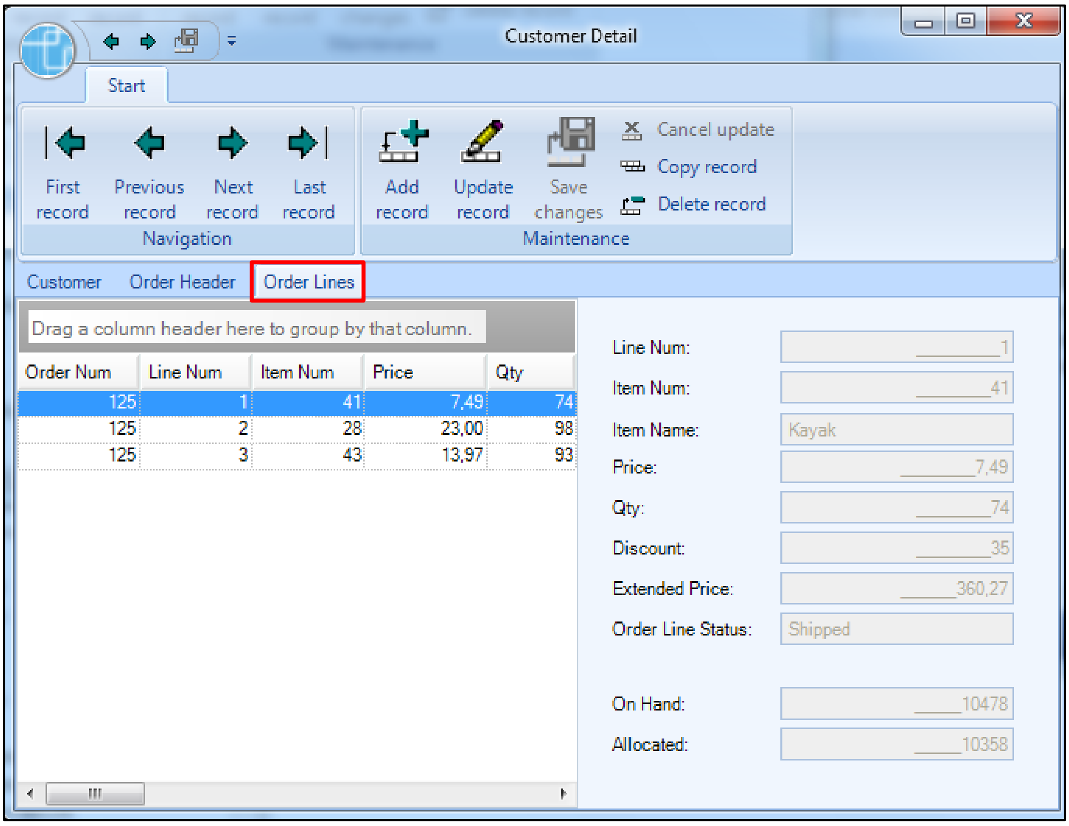
- Select all Fill-Ins and change property **DisplayStyle** to *Office2007*.

## Insert the User Control to the Form

To insert the *OrderLineDetail* viewer to the *Order Lines* tab please process as follows:

- Open the file *CustomerDetailForm.cls*.
- Switch to the Tab *Order Lines* and make sure that the Form is selected.
- Select the Designer Verb **Insert UserControl**.
  - The *Select UserControl class dialog* opens.
  - From the list select *OrderLineViewerControl*.
- Click **OK**.
- A message appears to inform you that the UserControl might have been placed in the background or behind other Controls.
  - If you cannot see the Control, use the Outline View to bring it to the front.
- Click **OK**.
- Drag the UserControl to the right side of the Form.
- Change Property **Dock** to *Fill***.**
- Set Property **LinkDataSource** to *OrderLineAdapter*.
- Set Property **LinkTableIOSource** to *CustomerDetailToolbar*.
- Save and test you application.

At runtime the *Customer Detail* Form for order line data should look like this:



| Order Num | Line Num | Item Num | Price | Qty |
|---|---|---|---|---|
| 125 | 1 | 41 | 7,49 | 74 |
| 125 | 2 | 28 | 23,00 | 98 |
| 125 | 3 | 43 | 13,97 | 93 |

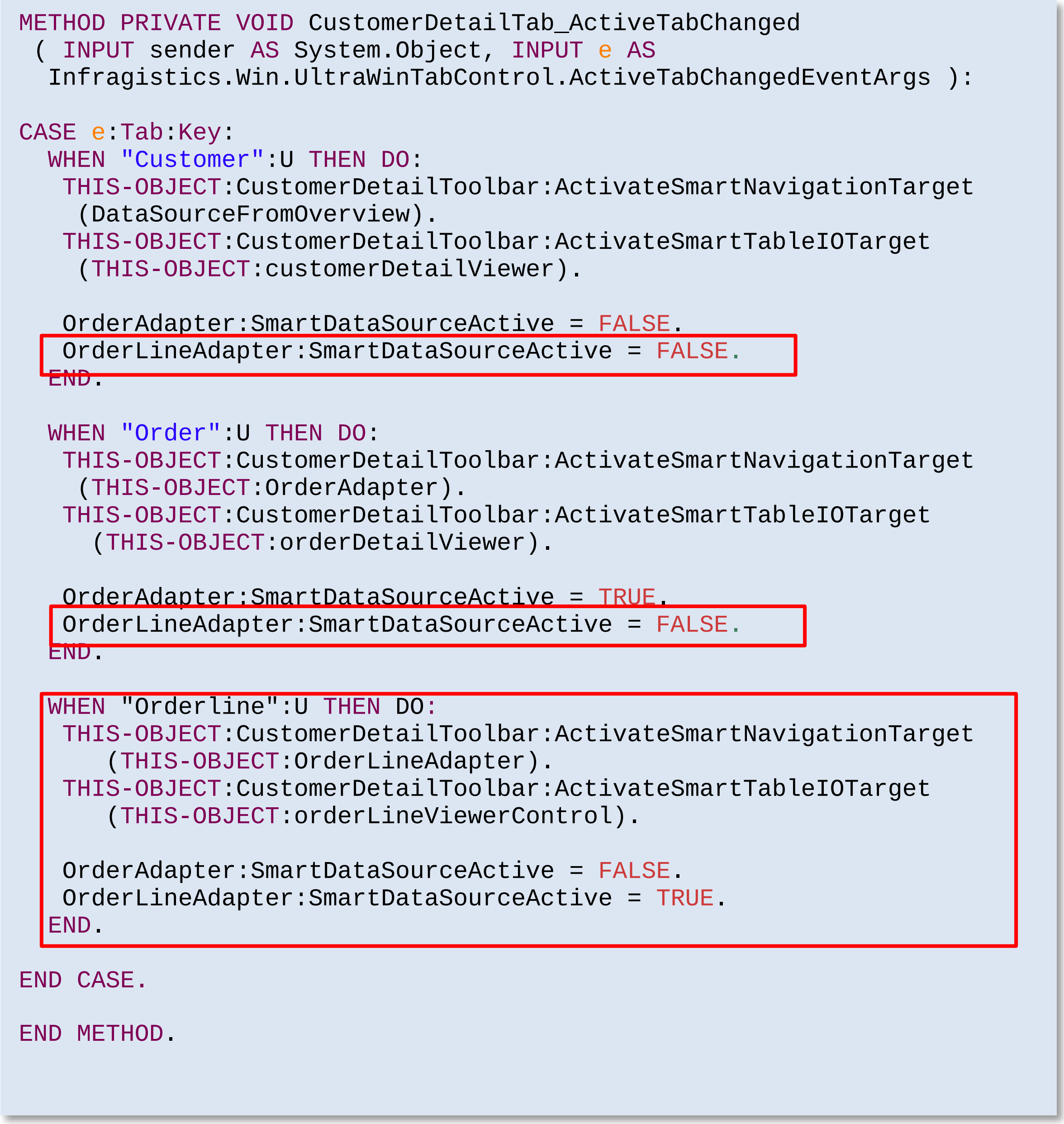
| | |
|---|---|
| Line Num: | 1 |
| Item Num: | 41 |
| Item Name: | Kayak |
| Price: | 7,49 |
| Qty: | 74 |
| Discount: | 35 |
| Extended Price: | 360,27 |
| Order Line Status: | Shipped |
| On Hand: | 10478 |
| Allocated: | 10358 |

## 10. Add Navigation Functionality

To complete the navigation function for the order lines based on the current selected tab page (as you have already implemented for customer and order) simply add some code to the event handler method *CustomerDetailTab_ActiveTabChanged* in *CustomerDetailForm.cls*.

The code you need for order line is highlighted below:

### Code for Order Lines



```
METHOD PRIVATE VOID CustomerDetailTab_ActiveTabChanged
 ( INPUT sender AS System.Object, INPUT e AS
  Infragistics.Win.UltraWinTabControl.ActiveTabChangedEventArgs ):

CASE e:Tab:Key:
  WHEN "Customer":U THEN DO:
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget
    (DataSourceFromOverview).
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartTableIOTarget
    (THIS-OBJECT:customerDetailViewer).

   OrderAdapter:SmartDataSourceActive = FALSE.
   OrderLineAdapter:SmartDataSourceActive = FALSE.
  END.

  WHEN "Order":U THEN DO:
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget
    (THIS-OBJECT:OrderAdapter).
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartTableIOTarget
     (THIS-OBJECT:orderDetailViewer).

   OrderAdapter:SmartDataSourceActive = TRUE.
   OrderLineAdapter:SmartDataSourceActive = FALSE.
  END.

  WHEN "Orderline":U THEN DO:
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartNavigationTarget
      (THIS-OBJECT:OrderLineAdapter).
   THIS-OBJECT:CustomerDetailToolbar:ActivateSmartTableIOTarget
      (THIS-OBJECT:orderLineViewerControl).

   OrderAdapter:SmartDataSourceActive = FALSE.
   OrderLineAdapter:SmartDataSourceActive = TRUE.
  END.

END CASE.

END METHOD.
```

**Congratulation**! You successfully added the Order and OrderLine information to your sample application.

# Add a Lookup for Salesrep

Normalized relational database schemas (and thus typically many Business Entities or ProDatasets) are using inherited, foreign keys to link data in two tables. Typically the linking table only contains the value of a primary unique key (in the sports2000 sample database for instance the Customer.SalesRep or OrderLine.ItemNum fields) of the linked table instead of duplicating values (like the SalesRep.RepName or Item.ItemName fields). The unique key values used for linking both tables can actually be a key meaningful to the user (like in the sports2000 database the actual customer number, salesrep code or item number) or can be a pure technical key, for instance generated using a GUID (globally unique identifier).

The *SmartBusinessEntityLookup* Control can help the user to update values in the linking table using search and lookup functionality on the linked table. The user can search records by entering a (partial) key value and the lookup Control will locate the linked record using a *SmartBusinessEntityAdapter* accessing a Business Entity on the back end and can display values (descriptive fields) of the located record. Alternatively the user can use a standardized (or customizable) lookup dialog using a browser Control and a simple filter mechanism.

For viewing the current key value and updating the key value when you are creating or modifying records the *SmartBusinessEntityLookup* Control uses standard .NET data binding on the Value or Text properties. The lookup also supports the usage of non-meaningful keys (e.g. GUID’s). If you are using non-meaningful keys you will need to use data binding to the *LookupKeyValue* property.

## Create a new Business Entity for SalesRep data

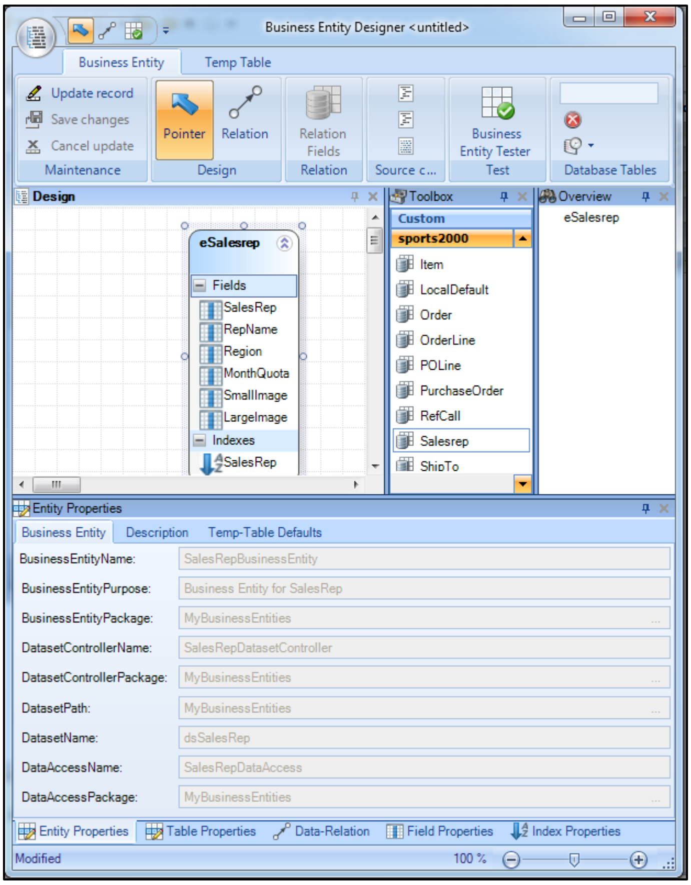
You already know how to create Business Entities. Open the **Business Entity Designer** and create a new Business Entity *SalesRepBusinessEntity* containing only one temp table: *eSalesRep* (=SalesRep in the sports2000 database).

If you need any assistance please refer to the chapter Developing Business Entities in this tutorial.

### Summary for SalesRepBusinessEntity

| Field | Description |
|---|---|
| BusinessEntityName | SalesRepBusinessEntity |
| BusinessEntityPurpose | Business Entity for SalesRep |
| BusinessEntityPackage | MyBusinessEntities |
| DatasetControllerName | SalesRepDatasetController |
| DatasetControllerPackage | MyBusinessEntities |
| DatasetPath | MyBusinessEntities |
| DatasetName | dsSalesRep |
| DataAccessName | SalesRepDataAccess |
| DataAccessPackage | MyBusinessEntities |

The result should look like this:



BusinessEntityName: SalesRepBusinessEntity

BusinessEntityPurpose: Business Entity for SalesRep

BusinessEntityPackage: MyBusinessEntities

DatasetControllerName: SalesRepDatasetController

DatasetControllerPackage: MyBusinessEntities

DatasetPath: MyBusinessEntities

DatasetName: dsSalesRep

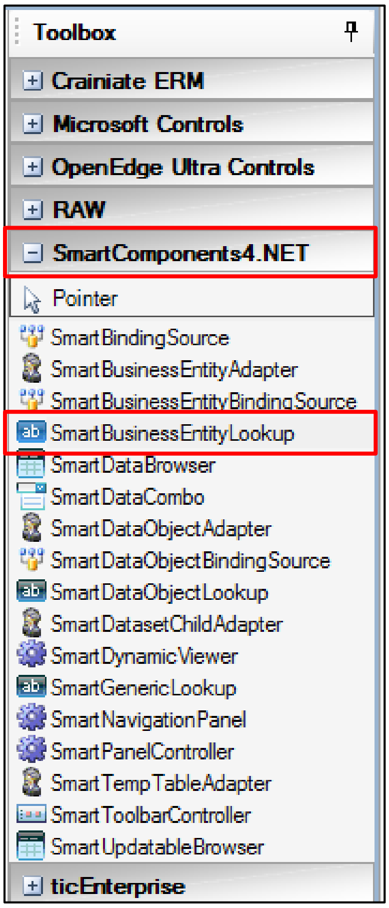
DataAccessName: SalesRepDataAccess

DataAccessPackage: MyBusinessEntities

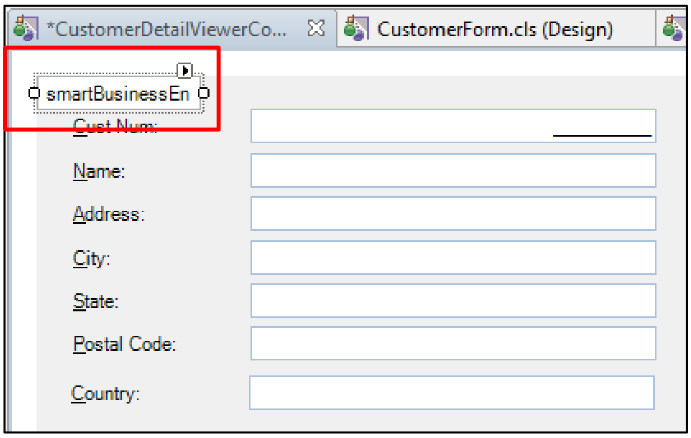
Please generate and compile the code for the Business Entity.

## Add a *SmartBusinessEntityLookup* Control to a Form

- Open the file CustomerDetailViewerControl.cls.
- From the *SmartComponents4.NET* group in the **Toolbox** add a *SmartBusinessEntityLookup* Control to the *OrderLineDetailViewer* Control by double-clicking the Control in the **Toolbox**.

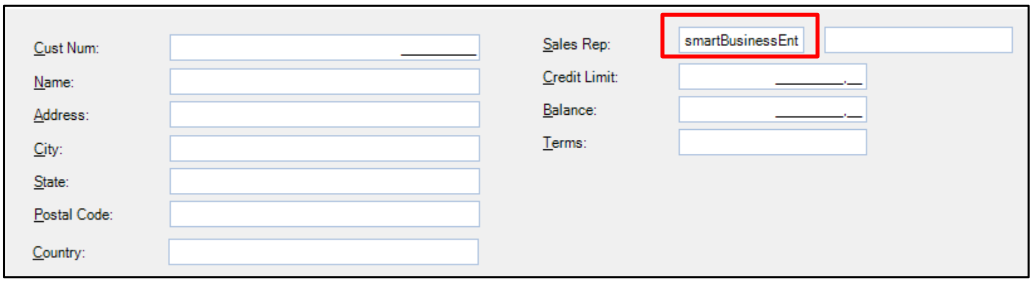


The new Lookup Control appears in the upper left corner of the *CustomerDetailViewerControl*.

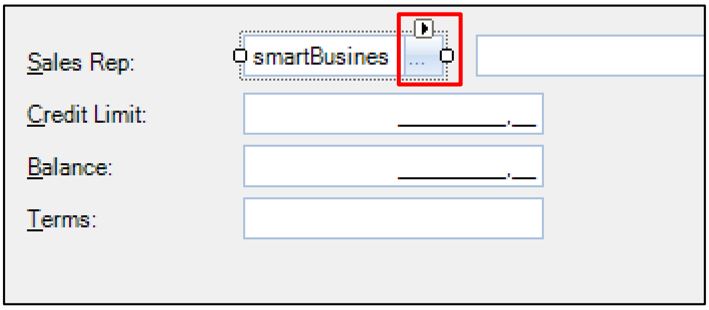


The *SalesRep* Control that you have previously created using the *Add Fields* wizard is no longer required. You can delete it from the Viewer design by pressing the *DEL* key after you have selected the Control. The *SmartBusinessEntityLookup* Control is a good replacement for any key value field.

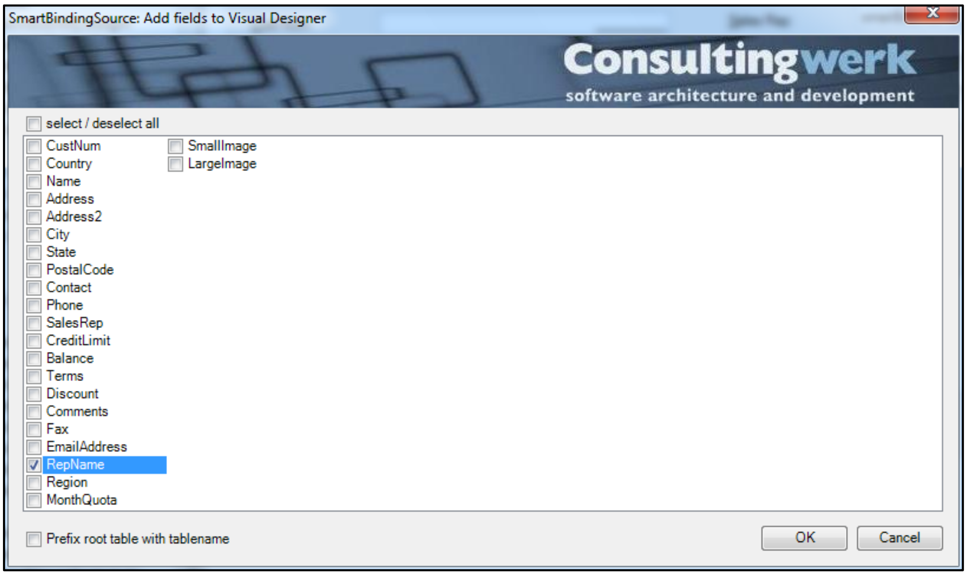
- Replace the *SalesRep* Control with the *Lookup Control* as shown below:



- Change Property **DisplayStyle** to *Office2007*.
- Select the Designer Verb **AddLookupButton**. This will add the lookup button to the Lookup (the button is shown at the right end of the lookup).



- Delete the field **SalesRep** and add the Field **RepName** instead.
- To do this, select the component *CustomerBindingSource* and click the *Designer Verb* **AddFields**. The **Add fields to Visual Designer** dialog opens.
- Select the field *RepName* and click **OK**.



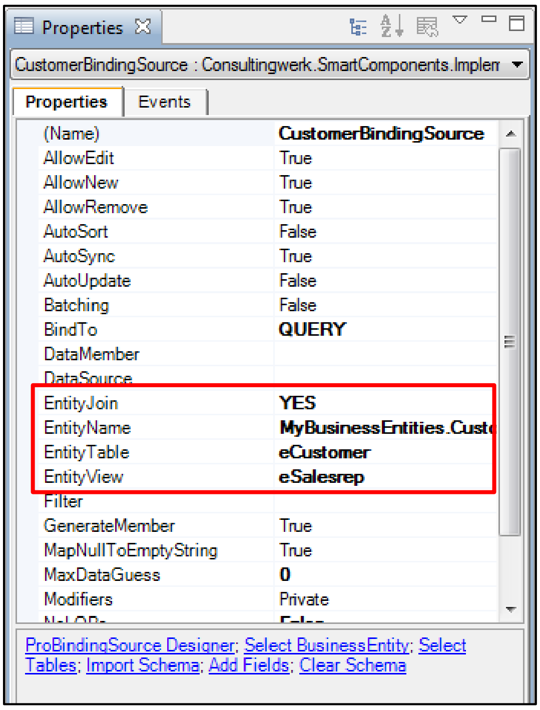
SmartBindingSource: Add fields to Visual Designer

□ select / deselect all

□ CustNum
□ Country
□ Name
□ Address
□ Address2
□ City
□ State
□ PostalCode
□ Contact
□ Phone
□ SalesRep
□ CreditLimit
□ Balance
□ Terms
□ Discount
□ Comments
□ Fax
□ EmailAddress
☑ RepName
□ Region
□ MonthQuota
□ SmallImage
□ LargeImage

□ Prefix root table with tablename

If you cannot see the field *RepName*, make sure that you joined both tables: *eCustomer* and *eSalesrep*. To test it, select the *customerBindingSource* component, check whether Property **Entity Join** is set to *YES* and if necessary import the schema again. If you must change the setting open the **Select Tables** dialog from the Designer Verb area.



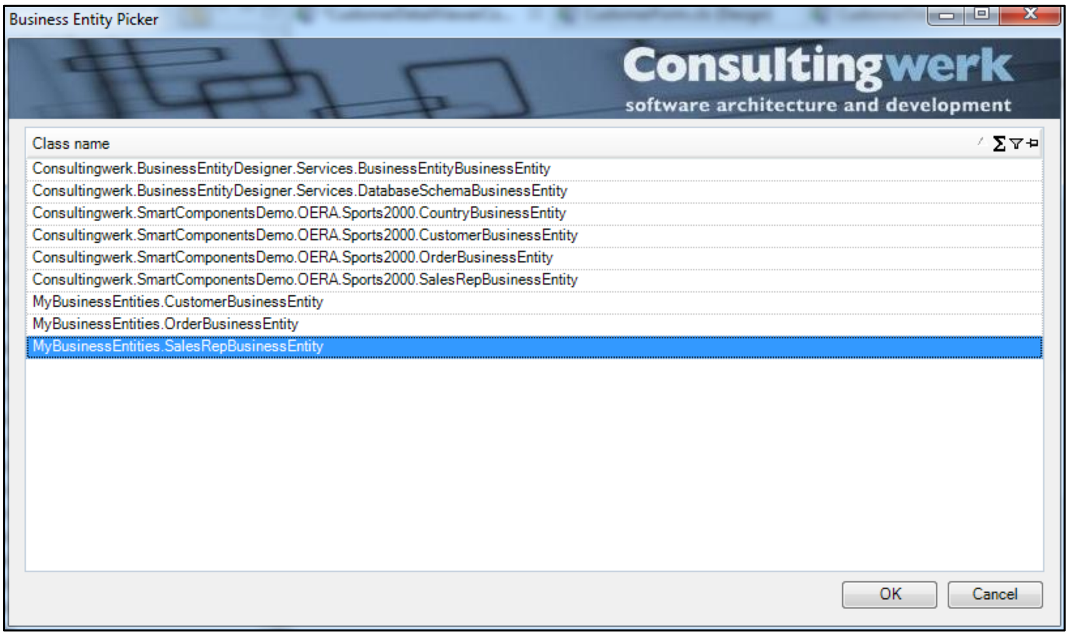
CustomerBindingSource : Consultingwerk.SmartComponents.Implem

| | |
|---|---|
| (Name) | CustomerBindingSource |
| AllowEdit | True |
| AllowNew | True |
| AllowRemove | True |
| AutoSort | False |
| AutoSync | True |
| AutoUpdate | False |
| Batching | False |
| BindTo | QUERY |
| DataMember | |
| DataSource | |
| EntityJoin | YES |
| EntityName | MyBusinessEntities.Custo |
| EntityTable | eCustomer |
| EntityView | eSalesrep |
| Filter | |
| GenerateMember | True |
| MapNullToEmptyString | True |
| MaxDataGuess | 0 |
| Modifiers | Private |

ProBindingSource Designer; Select BusinessEntity; Select Tables; Import Schema; Add Fields; Clear Schema

## Select the SalesRepBusinessEntity and the tables to be used by the Lookup

- To connect the new Lookup Control to the *SalesRepBusinessEntity* you have created earlier, mark the Lookup Control and select the Designer Verb **SelectBusinessEntity**.
- From the **Business Entity Picker** select *SalesRepBusinessEntity* and click **OK**.



| Class name |
|---|
| Consultingwerk.BusinessEntityDesigner.Services.BusinessEntityBusinessEntity |
| Consultingwerk.BusinessEntityDesigner.Services.DatabaseSchemaBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CountryBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.CustomerBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.OrderBusinessEntity |
| Consultingwerk.SmartComponentsDemo.OERA.Sports2000.SalesRepBusinessEntity |
| MyBusinessEntities.CustomerBusinessEntity |
| MyBusinessEntities.OrderBusinessEntity |
| MyBusinessEntities.SalesRepBusinessEntity |

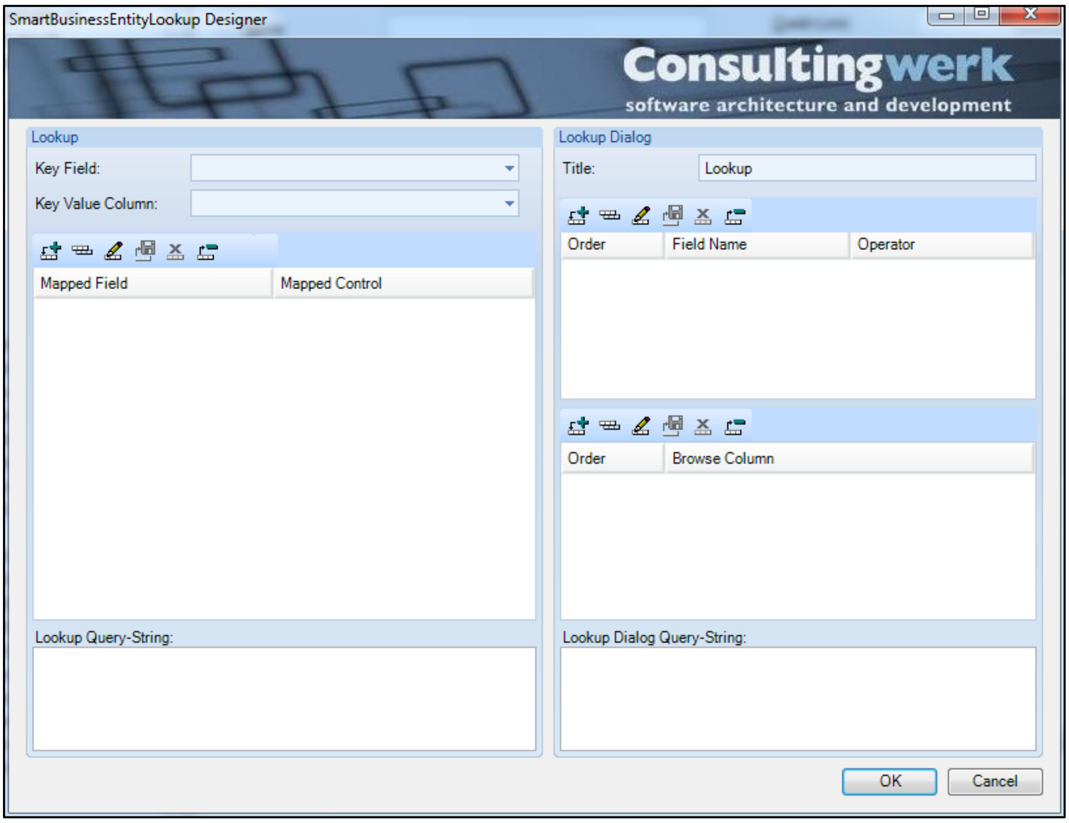
- Select the Designer Verb **SelectTables** by right-clicking the lookup on the design surface.
- Check the e*Salesrep* table in the tree view on the left.
- Click **OK**.

## SmartBusinessEntityLookup Designer

In this chapter you are going to make all necessary settings so that the Lookup will work as required. Consultingwerk provides a graphical **Lookup Designer** dialog that supports developers when defining Lookup settings. Alternatively you can set all properties in the property sheet. The Lookup Designer Dialog offers more support when assigning the properties of the lookup instance.

- To open the Lookup Designer select the Designer Verb **LookupDialog** on the *SmartBusinessEntityLookup* instance.

The **SmartBusinessEntityLookup Designer** opens.



The tool is separated in two different areas. On the left side you can define all settings concerning the lookup Control in the *SmartViewerControl*.

On the right side you define the options for the Lookup dialog, particularly the Lookup Browser and the filter options. The Lookup Dialog will open, when the user presses the Lookup Button of the lookup Control.

For the Lookup Control itself assign the following settings:

| Lookup | Values | Description |
|---|---|---|
| **Lookup Keys** | | |
| Key Field | SalesRep | The **Key Field** defines the column from the lookup table (eSalesRep in this sample) who's value will be assigned to the lookup Control itself (the Value property) when the user selects a record by entering a (partial) key in the lookup Control itself or selects a record from the lookup dialog. |
| Key Value Column | SalesRep | The **Key Value Column** defines the column from the lookup table (eSalesRep in this sample) whose value will be assigned to the *LookupKeyValue* property of the lookup Control when the user selects a record by entering a (partial) key in the lookup Control itself or selects a record from the lookup dialog. The *LookupKeyValue* property and thus the Key Value Column is only required when using non-meaningful keys in your database design and/or Business Entity design. |
| **Mapping** | | **Field/Control mapping** allows the lookup Control to assign values (*Text* property) to additional Controls in the Viewer. This is useful to return additional meaningful values (like the RepName) from the lookup whenever the user selects a record. The mapped fields are updated in a browser using two combo-box columns. |
| Mapped Field | RepName | The **Mapped Fields** define the fields from the lookup's Business Entity that will be used to assign values to the mapped controls. |

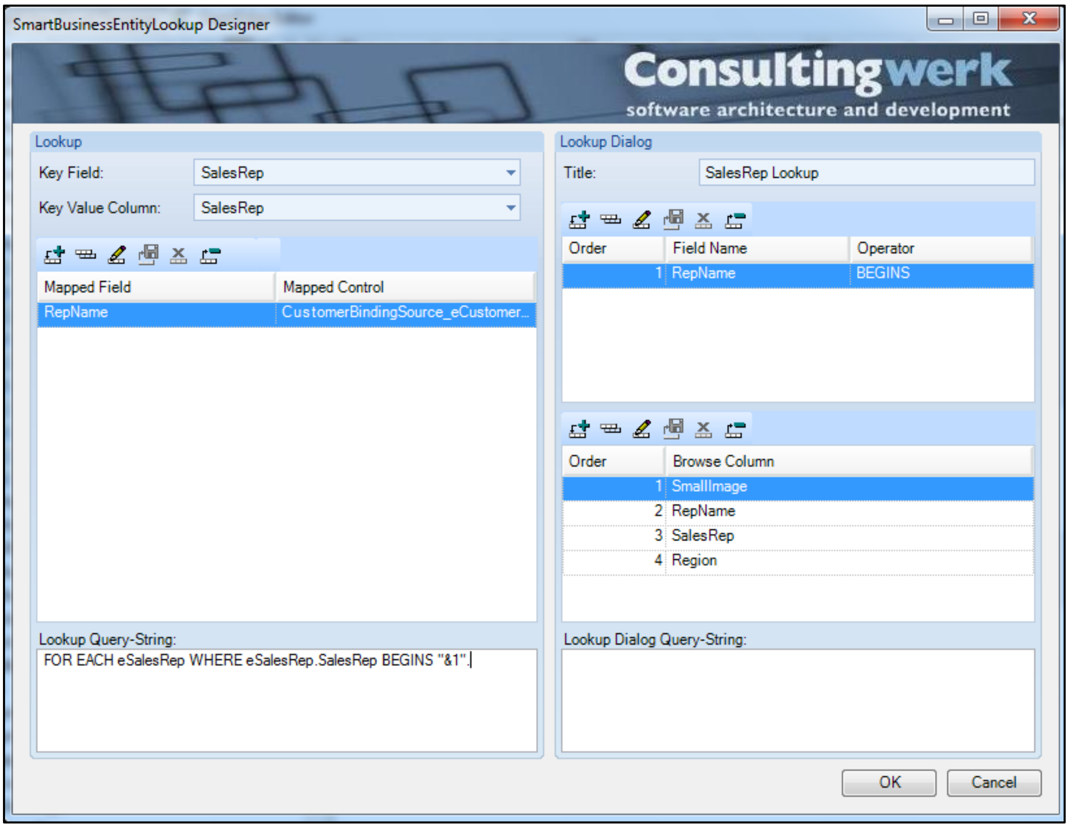
|  |  |  |
|---|---|---|
| Mapped Control | CustomerBindingSource _eCustomer_RepName | The mapped controls are the controls receiving the values from the lookup business entity. |
| **Lookup Query String** | ▪ FOR EACH SalesRep WHERE eSalesRep BEGINS "&1". | The lookup query string defines the query used to search for values in the lookups Business Entity when the user either leaves the lookup Control or does not type any keys for 0.75 seconds. The current value of the lookup Control (potentially a partial key) is inserted to the query string at the location of the &1 token. The token should be put in quotes no matter of the data-type of the key field. |

For the Lookup dialog assign the following settings on the right side:

| Lookup Dialog | Values | Description |
|---|---|---|
| **Title** | SalesRep Lookup | The title for the lookup dialog, like “Lookup SalesRep” |
| **Filter** |  | Filters can be used by the user to search records in tables with a lot of records. The lookup dialog supports any number of filters you need (including zero and one). To keep things simple for the user, a filter is always applied to a single field only. |
| Order | 1 | The order the filter options should appear in the lookup dialog. |
| Fieldname | RepName | The field from the lookup Business Entity to filter on. |
| Operator | BEGINS | The operator to filter on. Valid operators are value ABL query operators. |
| **Browser** |  |  |
| Order | 1-4 | The order of the columns in the browser |
| Browse Column | 1. Small Image | Select the columns for the lookup |

| | | |
|---|---|---|
| | 2. RepName<br>3. SalesRep<br>4. Region | browser in this order |
| **Lookup-Dialog Query String** | | The query string to be used by the lookup dialog. Leave empty for returning all values (filters specified as the source default query in the Data Access Object will apply always). |

The result should look like this:



SmartBusinessEntityLookup Designer

Lookup

Key Field: SalesRep

Key Value Column: SalesRep

| Mapped Field | Mapped Control |
|---|---|
| RepName | CustomerBindingSource_eCustomer... |

Lookup Query-String:

```
FOR EACH eSalesRep WHERE eSalesRep.SalesRep BEGINS "&1".
```

Lookup Dialog

Title: SalesRep Lookup

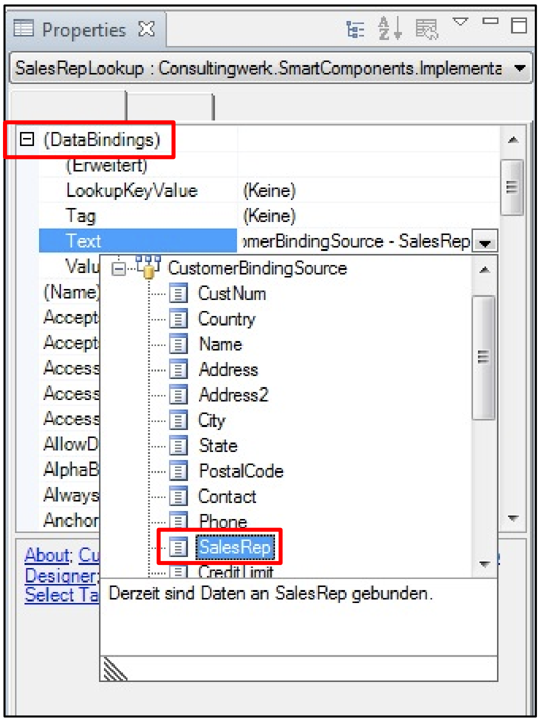
| Order | Field Name | Operator |
|---|---|---|
| 1 | RepName | BEGINS |

| Order | Browse Column |
|---|---|
| 1 | SmallImage |
| 2 | RepName |
| 3 | SalesRep |
| 4 | Region |

Lookup Dialog Query-String:

- With the *SalesRepLookup* Control selected, expand the Property Group **DataBindings** and change the binding of the Property **Value** to *CustomerBindingSource – SalesRep*.

Otherwise the Control does not have any data binding and cannot display the actual SalesRep for a selected record.



SalesRepLookup : Consultingwerk.SmartComponents.Implementa

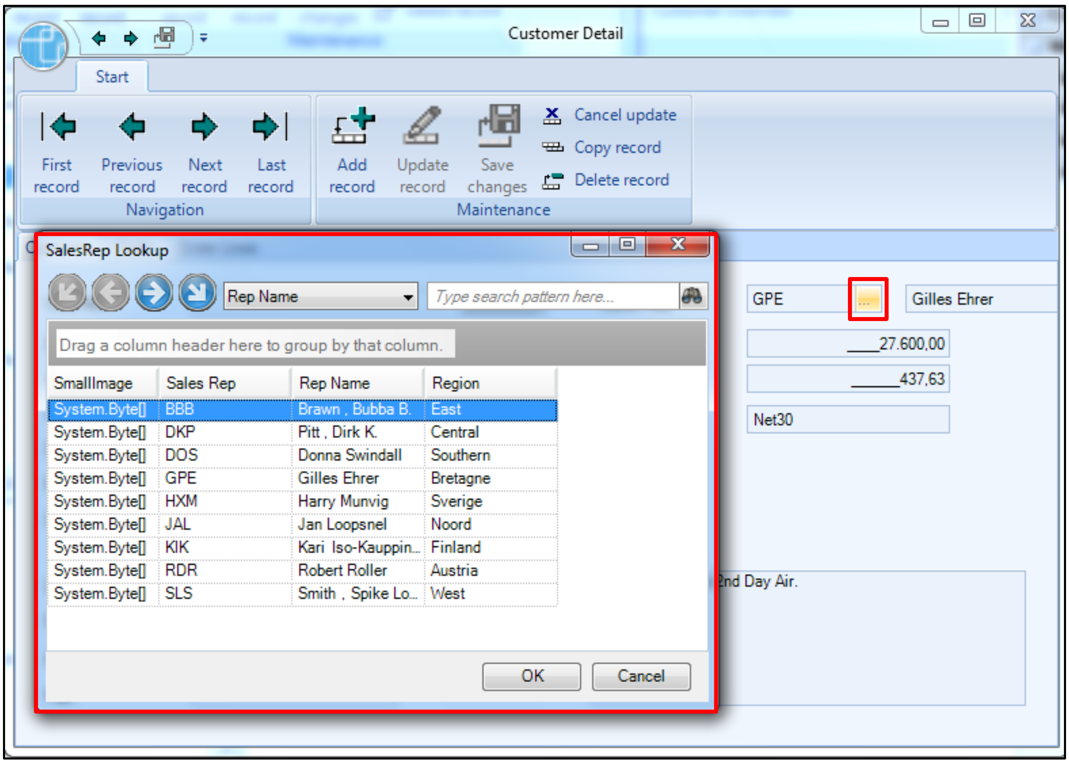
| | |
|---|---|
| (DataBindings) | |
| (Erweitert) | |
| LookupKeyValue | (Keine) |
| Tag | (Keine) |
| Text | merBindingSource - SalesRep |

CustomerBindingSource
- CustNum
- Country
- Name
- Address
- Address2
- City
- State
- PostalCode
- Contact
- Phone
- SalesRep
- CreditLimit

Derzeit sind Daten an SalesRep gebunden.

- Save and test your application.
- Open the *Customer Detail* dialog and activate the update function in the toolbar.
- Click on the *Lookup* button.

The **SalesRep Lookup** Dialog opens. It should look like this:



Rep Name ▾ | Type search pattern here...

Drag a column header here to group by that column.

| SmallImage | Sales Rep | Rep Name | Region |
|---|---|---|---|
| System.Byte[] | BBB | Brawn , Bubba B. | East |
| System.Byte[] | DKP | Pitt , Dirk K. | Central |
| System.Byte[] | DOS | Donna Swindall | Southern |
| System.Byte[] | GPE | Gilles Ehrer | Bretagne |
| System.Byte[] | HXM | Harry Munvig | Sverige |
| System.Byte[] | JAL | Jan Loopsnel | Noord |
| System.Byte[] | KIK | Kari Iso-Kauppin... | Finland |
| System.Byte[] | RDR | Robert Roller | Austria |
| System.Byte[] | SLS | Smith , Spike Lo... | West |

GPE | ... | Gilles Ehrer

27.600,00

437,63

Net30

nd Day Air.

You see all the Browse Columns you defined in the Lookup Designer and the filter option for RepName.

## Add some more functionality

There is some more work to do to complete the Lookup functionality:

- Disable the field **RepName** in the *CustomerDetailViewer* Control
  - Enable the field **CustNum**, when adding a new record only.
- Show the Images in the Lookup Browser.

## Disable the field **RepName** in the *CustomerDetailViewer* Control

Because you select the SalesRep with your new Lookup Control when adding or updating customer records the field *RepName* does not need to be active. To disable it permanently do the following:

- Select the field **RepName**.
- Set Property **Enabled** to *False*.

## Show the Images in the Lookup Browser

Last not least you want to bring the small images to the Lookup Browser.

What you need is an event handler for the *InitializeLookupBrowserLayout* event of the *LookupControl*. In this Method you set the Label of the Column to “*Image*” and the Style of the GridColumn to “`ColumnStyle:Image`” which enables the Browser to show images in that Column. The browse of the lookup dialog is based on the Infragistics UltraGrid. The UltraGrid will handle the content of BLOB fields in the Business Entity as Images when you set the ColumnStyle to “*Image”*.

- Add the highlighted code to the constructor of *customerDetailViewerControl.cls* :

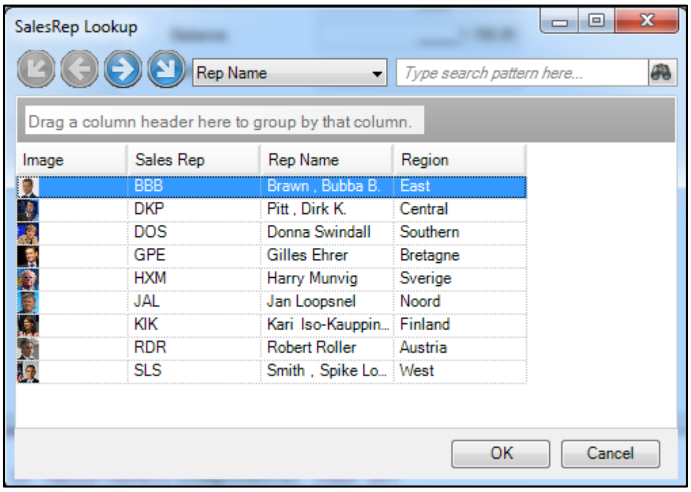
```
CONSTRUCTOR PUBLIC CustomerDetailViewerControl (  ):

  SUPER().
  InitializeComponent().

  SalesRepLookup:InitializeLookupBrowserLayout:Subscribe
   (InitializeLookupBrowserLayoutHandler).


  SetControlEnabled
    (THIS-OBJECT: customerBindingSource_eCustomer_Custnum,
    Consultingwerk.SmartComponents.Enum.ControlEnabledEnum:Add).

  CATCH e AS Progress.Lang.Error:
    UNDO, THROW e.
  END CATCH.

END CONSTRUCTOR.
```

- Add a new method as an event handler for the *InitializeLookupBrowserLayout* Event
- Write the following code:

```
METHOD PRIVATE VOID InitializeLookupBrowserLayoutHandler(Sender AS
System.Object, e AS Consultingwerk.SmartComponents.Base.
InitializeLookupBrowserLayoutEventArgs):

e:Grid:DisplayLayout:Bands[0]:Columns["SmallImage"]:Header:Caption =
"Image".

e:Grid:DisplayLayout:Bands[0]:Columns["SmallImage"]:Style =
Infragistics.Win.UltraWinGrid.ColumnStyle:Image.

END METHOD.
```

- Save and test your application.

The result should look like this:


SalesRep Lookup
Rep Name
Type search pattern here...
Drag a column header here to group by that column.
OK
Cancel

| Image | Sales Rep | Rep Name | Region |
|---|---|---|---|
| | BBB | Brawn , Bubba B. | East |
| | DKP | Pitt , Dirk K. | Central |
| | DOS | Donna Swindall | Southern |
| | GPE | Gilles Ehrer | Bretagne |
| | HXM | Harry Munvig | Sverige |
| | JAL | Jan Loopsnel | Noord |
| | KIK | Kari Iso-Kauppin... | Finland |
| | RDR | Robert Roller | Austria |
| | SLS | Smith , Spike Lo... | West |

**Congratulation**! You successfully added a Lookup to your sample application. The construction part of our tutorial is finished now.

# Code Review

## Overview

In this section we will take a closer look at the source code that has been generated for you by the **Business Entity Designer**. By default, ***Business Entity Designer*** generates separate include files for all Temp-Table s and the ProDataSet contained in a Business Entity component diagram.

By default the file names follow these conventions:

- ***e**TableName.i* for *the* Temp-Table includes
- ***ds**EntityName.i* for the ProDataSet includes

The advantage of generating includes files for each single component is that you can reuse the components in several Business Entities or ProDataSets.

Each Temp-Table contains two optional preprocessor directives:

- {&ACCESS}
- {&REFERENCE-ONLY}.

For more information please refer to the chapter Preprocessors in use.

Additionally there is a ***before-table*** *eTableNameBefore* defined for every Temp-Table .You need to use the before-table when you plan to make changes to the records of the Temp-Table in a ProDataSet. They are companion Temp-Tables to the after-table which is the Temp-Table you are typically working with and which holds the data. The contents of the before-table are managed automatically by the OpenEdge AVM when the TRACKING-CHANGES attribute is set to true which will be done by the SmartComponent Library on the frontend. A before-table holds versions of the records before modification (create, update, delete) were made to the returned records. When data is saved to the database the SAVE-ROW-CHANGES method uses the before-table for optimistic locking techniques. The Data Access object is dependent on the before-table for knowing which records have been changed by the user.

## Temp-Table: eCustomer.i

```
DEFINE {&ACCESS} TEMP-TABLE eCustomer NO-UNDO {&REFERENCE-ONLY} BEFORE-
TABLE eCustomerBefore

FIELD CustNum AS INTEGER FORMAT ">>>>9":U INIT "0":U LABEL "Cust Num"
FIELD Country AS CHARACTER FORMAT "x(20)":U INIT "USA":U LABEL "Country"

INDEX Comments AS WORD-INDEX Comments ASCENDING
INDEX CountryPost Country ASCENDING PostalCode ASCENDING
(…)
.
```

- Definition of the Temp-Table *eCustomer* and all appropriate fields and indexes (excerpt)

## Temp-Table: eSalesrep.i

```
DEFINE {&ACCESS} TEMP-TABLE eSalesrep NO-UNDO {&REFERENCE-ONLY} BEFORE-
TABLE eSalesrepBefore

FIELD SalesRep AS CHARACTER FORMAT "x(4)":U LABEL "Sales Rep"
FIELD RepName AS CHARACTER FORMAT "x(30)":U LABEL "Rep Name"
FIELD Region AS CHARACTER FORMAT "x(8)":U LABEL "Region"
FIELD MonthQuota AS INTEGER FORMAT "->,>>>,>>9":U INIT "0":U LABEL
"Month Quota"

INDEX SalesRep AS UNIQUE PRIMARY SalesRep ASCENDING.
```

- Definition of the Temp-Table *eSalesrep* and all appropriate fields and indexes

## ProDataSet: dsCustomer.i

```
&SCOPED-DEFINE ACCESS {&ACCESS}
&SCOPED-DEFINE REFERENCE-ONLY {&REFERENCE-ONLY}

{ MySchema/eCustomer.i }
{ MySchema/eSalesrep.i }


DEFINE {&ACCESS} DATASET dsCustomer {&REFERENCE-ONLY} FOR eCustomer,
eSalesrep
    DATA-RELATION eCustomerRelation FOR eCustomer, eSalesrep
        RELATION-FIELDS (SalesRep,SalesRep).
```

- Definition of the preprocessor directives
- References to the Temp-Table include files
- Definition of the ProDataSet **dsCustomer**, Data-Relation **eCustomerRelation** and Relation-Fields (*SalesRep,SalesRep*)

ℹ The ProDataset include file passes the values of the *ACCESS* and *REFERENCE-ONLY* preprocessor directives to the contained Temp-Table include files. To avoid complicated passing as include file parameters here the values are replicated as SCOPED-DEFINEs

## Business Entity: CustomerBusinessEntity.cls

```
CLASS MyBusinessEntities.CustomerBusinessEntity INHERITS BusinessEntity:

{ MySchema/dsCustomer.i }

CONSTRUCTOR PUBLIC CustomerBusinessEntity ():
  SUPER (DATASET dsCustomer:HANDLE).
  THIS-OBJECT:DataAccessName = "MyBusinessEntities.CustomerDataAccess":U
END CONSTRUCTOR.
(…)
```

- Definition of the class **CustomerBusinessEntity**
- Reference to the ProDataSet include file *dsCustomer*.i
- Definition of the DataAccess Object

By default the OERA implementation starts a Business Entity the first time when it is used during runtime or design time. The *CustomerBusinessEntity* class inherits from a *BusinessEntity* base (parent) class and is the owner and creator of the dataset *dsCustomer*.

Unless you modify the Service Interface procedures located in the OERA/support folder or the *Consultingwerk.OERA.ServiceInterface* class the Business Entities will remain in the memory of the client or AppServer process until it terminates. The Service Interface has methods to shut down Business Entities. The strategies for using this functionality highly depends on your application.

The *Constructor* passes the dataset handle to the *BusinessEntity* base class where it is validated and verified to make sure that the Business Entity always uses a ProDataset.

Subsequently the name of the DataAccess Object **CustomerDataAccess** is defined. This Data Access object is used whenever the Business Entity needs to read and write data to or from the database.

There are two void messages defined in the *CustomerBusinessEntity.cls* file, **ReceiveData()** and **ValidateDatat()**. Both methods are defined as abstract in the base class, so the must be created here in the class file of the Business Entity.

The **ReceiveData()** method provides a hook to modify data in the ProDataSet **after** all read and update operations. It is invoked during the *FetchData()* and *SaveChanges()* process.

```
METHOD OVERRIDE PUBLIC VOID ReceiveData ():

/*--------------------------------------------------------------------

Purpose: Provides a hook to modify data in the ProDataset after Read and
update operations (i.e. population of aggregated values)

Notes: Invoked during FetchData () and SaveChanges ()

--------------------------------------------------------------------*

END METHOD.
```

The **ValidateData()** method provides a hook for high level data validation **before** all update operations happen in the Data Access Object. It is invoked during the *SaveChanges()* process. In case of an ERROR in the ProDataSet the update operation is cancelled before writing the data back to the database.

```
METHOD OVERRIDE PUBLIC VOID ValidateData ():

/*---------------------------------------------------------------------

Purpose: Provides a hook for high level data validation before Update
operations

Notes: Invoked during SaveChanges (). When the ERROR flag of the
ProDataset is set, the Update operation will be cancelled before writing
back the data to the database using the DataAccess object

---------------------------------------------------------------------*/

END METHOD.
```

## Data Access Object: CustomerDataAccess.cls

The *Data Access Object* is used by the Business Entity whenever data needs to be read from or written to the database. According to the OERA model the Data Access Object is the only object that is aware of the existence of database tables.

⚠ This is the ideal approach. However in the real world it might be o.k. to violate this concept when you are aware of the consequences.

```
CLASS MyBusinessEntities.CustomerDataAccess INHERITS DataAccess:

{ MySchema/dsCustomer.i &ACCESS="PRIVATE" &REFERENCE-ONLY="REFERENCE-
ONLY"}

DEFINE PRIVATE DATA-SOURCE src_Customer FOR Customer .
DEFINE PRIVATE DATA-SOURCE src_Salesrep FOR Salesrep .
(…)

CONSTRUCTOR PUBLIC CustomerDataAccess (phDataset AS HANDLE):
  SUPER (INPUT phDataset).
  BindDataset (DATASET-HANDLE phDataset BIND).
END CONSTRUCTOR.
(…)

METHOD OVERRIDE PROTECTED VOID AttachDataSources ():
  Consultingwerk.Util.DatasetHelper:SetTrackingChanges (DATASET
  dsCustomer:HANDLE, FALSE) .

  BUFFER eCustomer:ATTACH-DATA-SOURCE (DATA-SOURCE src_Customer:HANDLE,
  "CustNum,Customer.CustNum,Country,Customer.Country,
  Name,Customer.Name…”).
  BUFFER eSalesrep:ATTACH-DATA-SOURCE (DATA-SOURCE src_Salesrep:HANDLE,
  "SalesRep,Salesrep.SalesRep,RepName,Salesrep.RepName,
  Region,Salesrep.Region,MonthQuota,Salesrep.MonthQuota") .
END METHOD.
(…)

METHOD PRIVATE VOID BindDataset (DATASET dsCustomer BIND):
        /* NOOP */
END METHOD.

(…)

METHOD OVERRIDE PROTECTED VOID DetachDataSources ():
  Consultingwerk.Util.DatasetHelper:SetTrackingChanges
  (DATASETdsCustomer:HANDLE, FALSE) .

  BUFFER eCustomer:DETACH-DATA-SOURCE () .
  BUFFER eSalesrep:DETACH-DATA-SOURCE () .
END METHOD.
```

The *CustomerDataAccess* class is derived from the *Consultingwerk.OERA.DataAccess* class and manages database access for FILL and SAVE processes of the ProDataSet. For that reason all necessary data-sources are defined, attached and detached in this class: *src_Customer* and *src_Salesrep*.

Please note the appropriate methods *AttachDataSources()* and *DetachDataSources()*!

The reference to the dataset include file *dsCustomer*.i is set with ACCESS-Mode = PRIVATE and REFERENCE-ONLY flag, because the owner of the dataset is the *CustomerBusinessEntity* class.

The call of the *BindDataSet()* method binds the dataset to the ProDataSet instance of the *BusinessEntity* class, allowing static access to all of its Temp-Table s.

This will result in static access to the same ProDataset instance both in the Business Entity class (CustomerBusinessEntity) and the Data Access class (CustomerDataAccess). The *BindDataSet()* method does not require any code. It's executed only once when the Data Access object is started. For details on passing ProDatasets with the BIND option we recommend reading the relevant chapters in the ABL documentation.

The **DefineReadEvents()** method is a placeholder method to define callback events for a ProDataSet. It is defined as abstract in the base class and is called during the read process to activate the callback procedures.

```
METHOD OVERRIDE PROTECTED VOID DefineReadEvents ():

/*------------------------------------------------------------------

Purpose: TO-DO: Subscribe to ProDataset Event Handlers using SET-
CALLBACK as needed

Notes: Overrides ABSTRACT method in Consultingwerk.OERA.DataAccess,
Invoked in FetchData

------------------------------------------------------------------*/
```

The **SourceColumn()** method is called during the read process and translates the Temp-Table field names of the query string returned by the client to the database field names. As a developer you only interfere here when you use different field names in the ProDataSet and the database or when fields in a joined Data-Source are ambiguous.

```
METHOD OVERRIDE PUBLIC CHARACTER SourceColumn (pcTable AS CHARACTER,
pcColumn AS CHARACTER):

/*----------------------------------------------------------------------

Purpose: Returns the database field name matching a Temp-Table  field
name contained in a consumers query string (query string vs. Temp-Table
definition)

Notes: Call-back used by Consultingwerk.OERA.Query.DSQueryString (part
of DataAccess:FetchData () query preparation

TO-DO: Provide code for alternative mapping

----------------------------------------------------------------------*/
```

The **SourceDefaultQuery()** method returns the base query string used to retrieve data for the Temp-Table s. As a developer you can interfere, to set a filter by company in the DataAccess object.

```
METHOD OVERRIDE PUBLIC CHARACTER SourceDefaultQuery (pcTable AS
CHARACTER):

/*----------------------------------------------------------------------

Purpose: Returns the base query string used to retrieve data for the
temp tables. This query string will be appended by the query provided by
the consumer (FetchDataRequest object).

Notes: Call-back used by Consultingwerk.OERA.Query.DSQueryString (part
of DataAccess:FetchData () query preparation

TO-DO: Provide code to return the DATA-SOURCE root query string

----------------------------------------------------------------------*/
```

## Dataset Controller Object: CustomerSalesrepDatasetController.cls

```
CLASS MyBusinessEntities.CustomerDatasetController IMPLEMENTS
IDatasetController:

{ MySchema/dsCustomer.i }
(…)

CONSTRUCTOR PUBLIC CustomerDatasetController ():
  SUPER ().
  THIS-OBJECT:DatasetHandle = DATASET dsCustomer:HANDLE.
END CONSTRUCTOR.
```

The DatasetController defines a static ProDataset with the same structure (using the same include files that define the schema) as the Business Entity dataset (procedural or object-oriented). Because of the static schema definition and a centrally accessible location the DatasetController simplifies writing (client side) logic working on the data managed by the SmartBusinessEntityAdapter (or derived types).

This development strategy is closely related to common MVC / MVP design patterns. It simplifies the separation of the actual UI logic from client side business logic and may also be used in Business Logic on the AppServer, referenced by the actual Business Entity (as long as developers does not access functionality like .NET classes that are not available on the AppServer).

### Using a DatasetController with SmartBusinessEntityAdapter and SmartDatasetChildAdapter components

There are two ways to associate a DatasetController with a SmartBusinessEntityAdapter:

The standard way to associate a DatasetController with a SmartBusinessEntityAdapter is to use the property sheet of the SmartBusinessEntityAdapter instance on a Form. The SmartBusinessEntityAdapter has a property called DatasetControllerType where a developer should enter the class name of the DatasetController for the used Business Entity. The class name entered in the property sheet will be validated immediately.

An alternative way of associating a DatasetController with a SmartBusinessEntityAdapter would be to assign a reference of an instance of a DatasetController to the property named DatasetController of the SmartBusinessEntityAdapter instance. This needs to be done before the first time RetrieveData() or similar methods have been called that might require that the SmartBusinessEntityAdapter is required to get a dataset instance.

The property DatasetController can in any case be used to get a reference to the current DatasetController instance associated with a SmartBusinessEntityAdapter. When using the DatasetControllerType property in the property grid the SmartBusinessEntityAdapter will dynamically create an instance of that type and assign the reference to the DatasetController property.

In any case the SmartBusinessEntityAdapter will “introduce” himself to the DatasetController using the RegisterConsumer method of the IDatasetController interface. This method needs to be implemented in the DatasetController and might be used to register to events of the SmartBusinessEntityAdapter class.

A SmartDatasetChildAdapter will - like a SmartBusinessEntityAdapter - register with the DatasetController when it’s direct or indirect SmartDataSource is a SmartBusinessEntityAdapter that is working together with a DatasetController. The SmartDatasetChildAdapter will work on buffers on the static defined ProDataset provided by the DatasetController. A SmartDatasetChildAdapter is not able to independently create an instance of a DatasetController. From an architectural point of view there is no need for that. The SmartDatasetChildAdapter only creates another view on the data managed by the SmartBusinessEntityAdapter. The SmartDatasetChildAdapter is navigating on the same data as the SmartBusinessEntityAdapter and will always use the same ProDataset instance as its direct or indirect SmartDataSource that is a SmartBusinessEntityAdapter.

The method DeregisterConsumer of the interface IDatasetController will be called by both types of the SmartDataAdapters when they are being destroyed.

Further information on the DatasetController can be found at:
http://wiki.dynamics4.net/D4wiki/SmartComponentLibrary/DevelopersReference/DatasetController

## Preprocessor in use

Each Temp-Table and ProDataSet include file typically contain two preprocessor directives {&ACCESS} and {&REFERENCE-ONLY}.

**{&ACCESS}:**

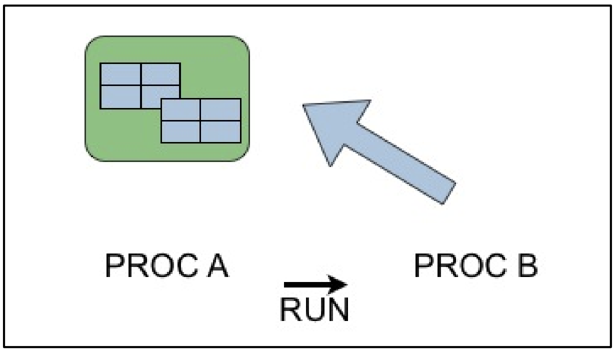
Sets the access mode for the Temp-Table (PRIVATE or PROTECED) if needed. When it's empty it is ignored, meaning that the ABL will default to PRIVATE. In this way you can use the same include files for object oriented and procedural applications.

**{&REFERENCE-ONLY}:**

When you pass a ProDataSet by reference, the called routine's ProDataSet definition is bound to the calling routine's ProDataSet.

When a Temp-Table or ProDataset is defined REFERENCE-ONLY the compiler will be able to validate access to fields and tables during compile time and allow static access. However the object (or procedure) does not create it's own instance of the data-structure. To work with the data you need to receive a Temp-Table or ProDataset from another object or procedure either BY-REFERENCE (for the duration of the call to a single method or internal method) or BIND (until the target object or persistent procedure is destroyed).

The primary advantage of defining Temp-Tables or ProDatasets in objects receiving a data-structure is the reduction of resource consumption. A Temp-Table or ProDataset that is defined without the REFERENCE-ONLY option will consume more memory and eventually also cause the client process and/or the AppServer to write to the dbi file (known as Temp-Table Dataset). Even though Progress Software has announced some optimization in OpenEdge 11 this issue is known in the community as the "too many Temp-Table issue (TMTT)".



# Using Images for Browser Columns and Combo-Boxes

TBD: Intro

## Einer Browserspalte (UltraGrid) zuweisen

```
THIS-OBJECT:smartDataBrowser1:DisplayLayout:Bands[0]:Columns["Flags"]:ValueList
=Consultingwerk.SmartComponentsDemo.CustomerExplorer.ValueLists:CustomerFlags.

THIS-
OBJECT:smartDataBrowser1:DisplayLayout:Bands[0]:Columns["OrderStatus"]:ValueList
=
            Consultingwerk.SmartComponentsDemo.CustomerExplorer.ValueLists:Order
Status .

THIS-
OBJECT:smartDataBrowser3:DisplayLayout:Bands[0]:Columns["OrderLineStatus"]:Value
List =
Consultingwerk.SmartComponentsDemo.CustomerExplorer.ValueLists:OrderLineStatus .
```

## Einer ComboBox zuweisen:

```
THIS-OBJECT:ultraComboEditor1:ValueList =CAST
(Consultingwerk.SmartComponentsDemo.CustomerExplorer.ValueLists:OrderSta
tus:Clone(),Infragistics.Win.ValueList)
```

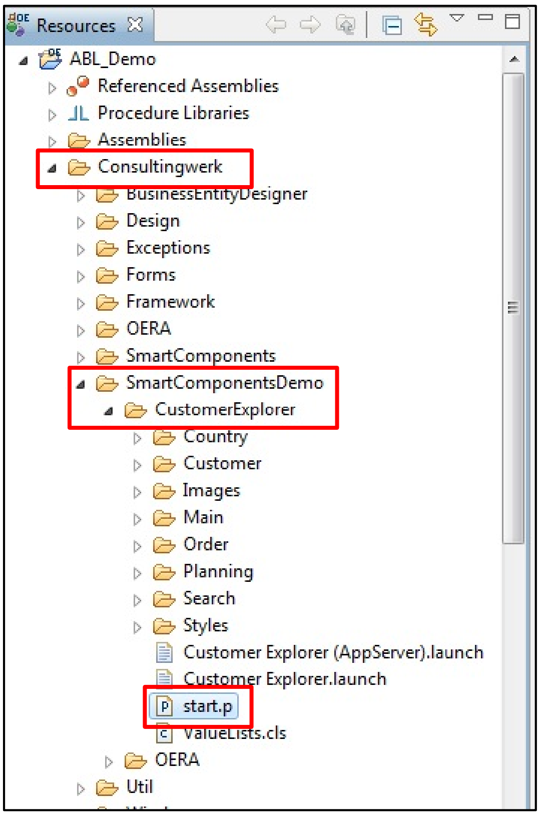
Hierbei wird der ComboBox eine Kopie zugewiesen – sonst gibt es offenbar Fehlermeldungen wenn die selbe ValueList gleichzeitig in Grid und Combo verwendet wird. Prinzipiell würde es auch nicht schaden, beim Grid ebenfalls eine Kopie (Clone() Methode) zu verwenden!

Die ValueLists Klasse enthält statische Referenzen auf ValueLists, die am Client vorhanden sind – könnten auch einmalig dynamisch aus der DB generiert werden.

# Appendix

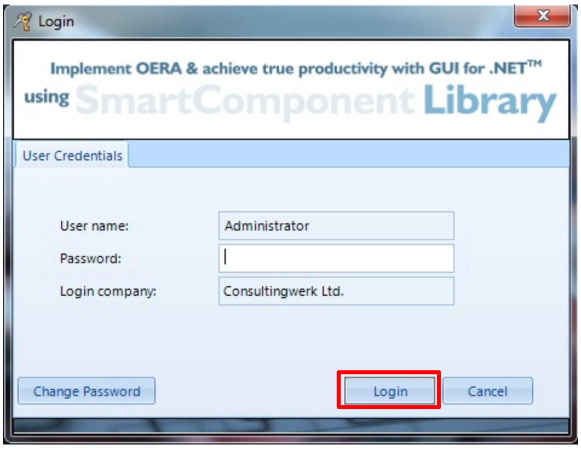
## Start the Demo Application

The CustomerExplorer_Demo deployment comes together with a demo application for test and educational purposes called the *CustomerExplorer* demo. The demo files are stored in the *Consultingwerk/SmartComponentDemo/CustomerExplorer* folder for the frontend code and the *Consultingwerk/SmartComponentsDemo/OERA* for the backend code. To start the demo application, expand your project folder as shown below and open the **start.p** procedure.



A login window appears.

- The username is prefilled with your Windows user name; leave the password blank (it is not validated) and click **Login**.

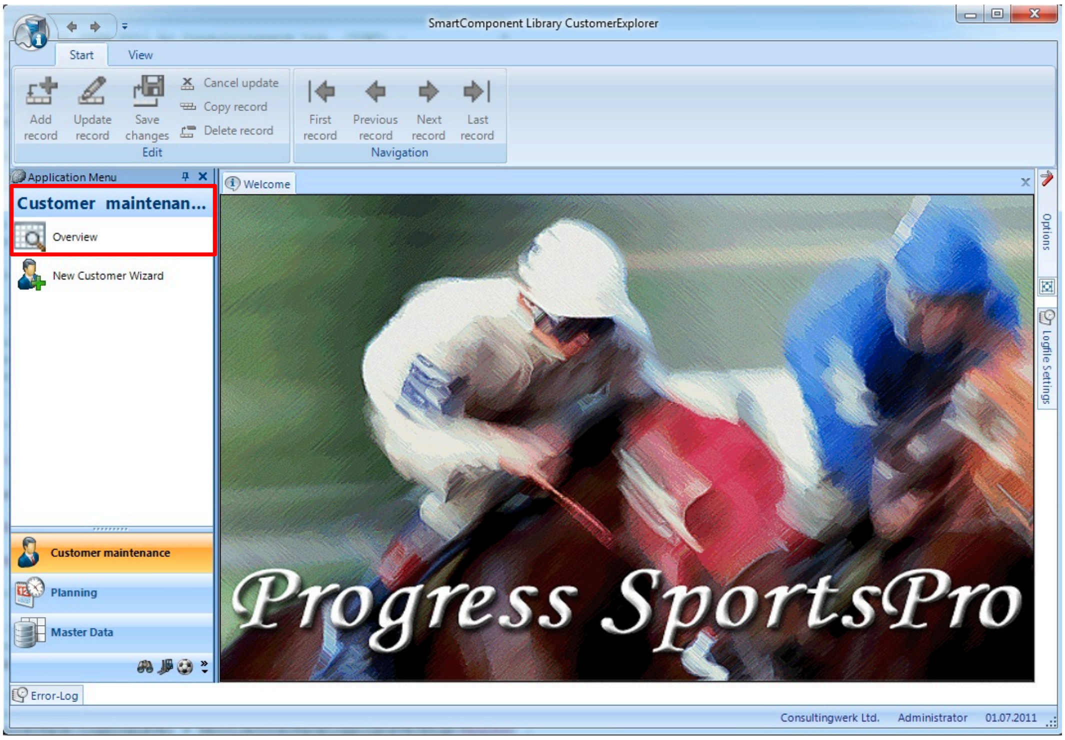

Login
Implement OERA & achieve true productivity with GUI for .NET™
using SmartComponent Library
User Credentials
Change Password
Login
Cancel

User name: Administrator

Password:

Login company: Consultingwerk Ltd.

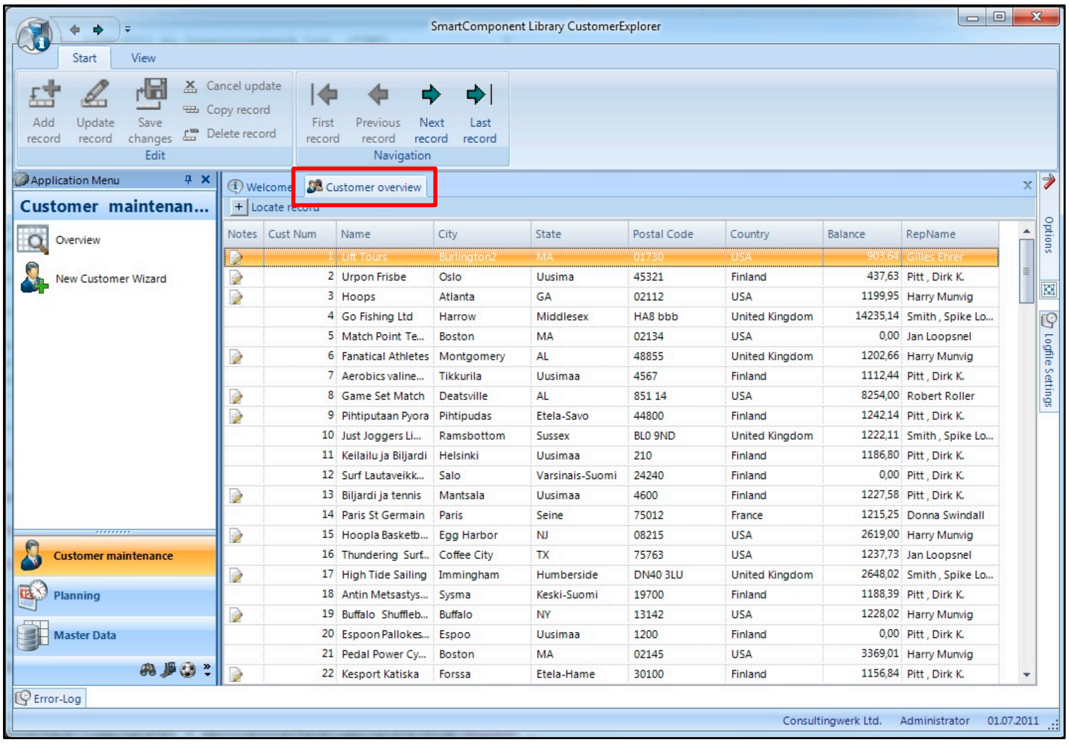
The CustomerExplorer demo opens and shows the main menu.



Consultingwerk Ltd. Administrator 01.07.2011

- Select *Customer maintenance* > *Overview*.

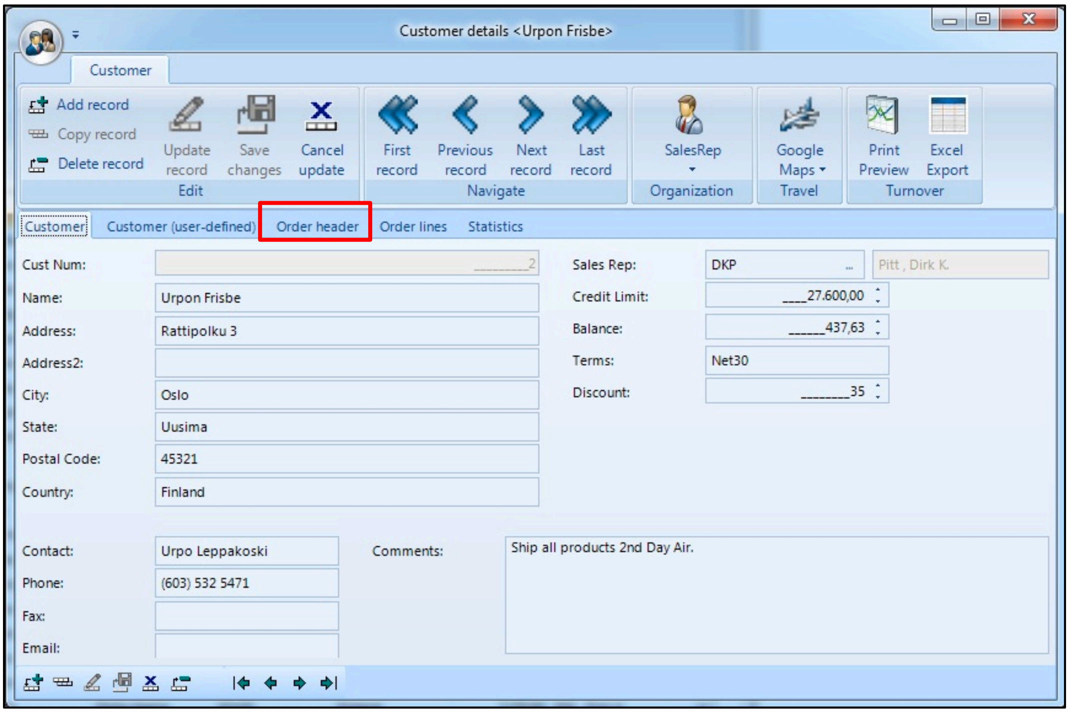
The *Customer Overview* appears.



| Notes | Cust Num | Name | City | State | Postal Code | Country | Balance | RepName |
|---|---|---|---|---|---|---|---|---|
| ✎ | 1 | Lift Tours | Burlington2 | MA | 01730 | USA | 903,64 | Gilles Ehrer |
| ✎ | 2 | Urpon Frisbe | Oslo | Uusima | 45321 | Finland | 437,63 | Pitt , Dirk K. |
| ✎ | 3 | Hoops | Atlanta | GA | 02112 | USA | 1199,95 | Harry Munvig |
|  | 4 | Go Fishing Ltd | Harrow | Middlesex | HA8 bbb | United Kingdom | 14235,14 | Smith , Spike Lo... |
|  | 5 | Match Point Te... | Boston | MA | 02134 | USA | 0,00 | Jan Loopsnel |
| ✎ | 6 | Fanatical Athletes | Montgomery | AL | 48855 | United Kingdom | 1202,66 | Harry Munvig |
|  | 7 | Aerobics valine... | Tikkurila | Uusimaa | 4567 | Finland | 1112,44 | Pitt , Dirk K. |
| ✎ | 8 | Game Set Match | Deatsville | AL | 851 14 | USA | 8254,00 | Robert Roller |
| ✎ | 9 | Pihtiputaan Pyora | Pihtipudas | Etela-Savo | 44800 | Finland | 1242,14 | Pitt , Dirk K. |
|  | 10 | Just Joggers Li... | Ramsbottom | Sussex | BL0 9ND | United Kingdom | 1222,11 | Smith , Spike Lo... |
|  | 11 | Keilailu ja Biljardi | Helsinki | Uusimaa | 210 | Finland | 1186,80 | Pitt , Dirk K. |
|  | 12 | Surf Lautaveikk... | Salo | Varsinais-Suomi | 24240 | Finland | 0,00 | Pitt , Dirk K. |
| ✎ | 13 | Biljardi ja tennis | Mantsala | Uusimaa | 4600 | Finland | 1227,58 | Pitt , Dirk K. |
|  | 14 | Paris St Germain | Paris | Seine | 75012 | France | 1215,25 | Donna Swindall |
| ✎ | 15 | Hoopla Basketb... | Egg Harbor | NJ | 08215 | USA | 2619,00 | Harry Munvig |
|  | 16 | Thundering Surf... | Coffee City | TX | 75763 | USA | 1237,73 | Jan Loopsnel |
| ✎ | 17 | High Tide Sailing | Immingham | Humberside | DN40 3LU | United Kingdom | 2648,02 | Smith , Spike Lo... |
|  | 18 | Antin Metsastys... | Sysma | Keski-Suomi | 19700 | Finland | 1188,39 | Pitt , Dirk K. |
| ✎ | 19 | Buffalo Shuffleb... | Buffalo | NY | 13142 | USA | 1228,02 | Harry Munvig |
|  | 20 | Espoon Pallokes... | Espoo | Uusimaa | 1200 | Finland | 0,00 | Pitt , Dirk K. |
|  | 21 | Pedal Power Cy... | Boston | MA | 02145 | USA | 3369,01 | Harry Munvig |
| ✎ | 22 | Kesport Katiska | Forssa | Etela-Hame | 30100 | Finland | 1156,84 | Pitt , Dirk K. |

- Double-click a customer record.

The *Customer Detail* dialog opens.



Cust Num: 2

Name: Urpon Frisbe

Address: Rattipolku 3

Address2:

City: Oslo

State: Uusima

Postal Code: 45321

Country: Finland

Contact: Urpo Leppakoski

Phone: (603) 532 5471

Fax:

Email:

Sales Rep: DKP Pitt , Dirk K.

Credit Limit: 27.600,00

Balance: 437,63

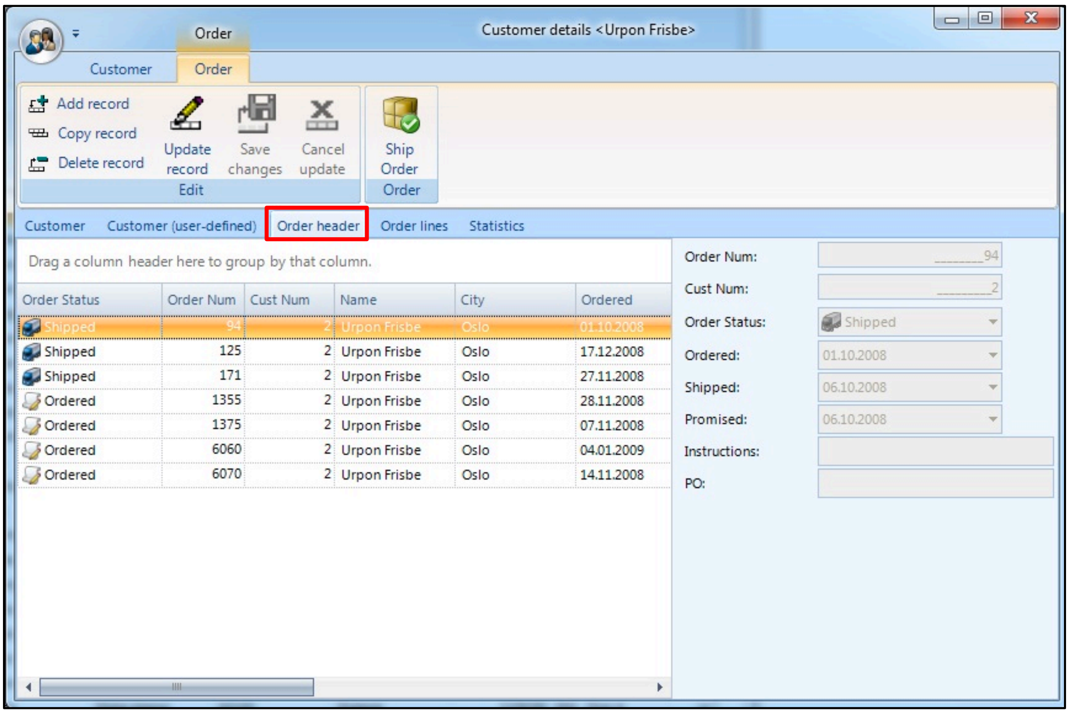
Terms: Net30

Discount: 35

Comments: Ship all products 2nd Day Air.

- To see related order data for the customer record switch to the **Order header** tab.

The result screen should look like this.



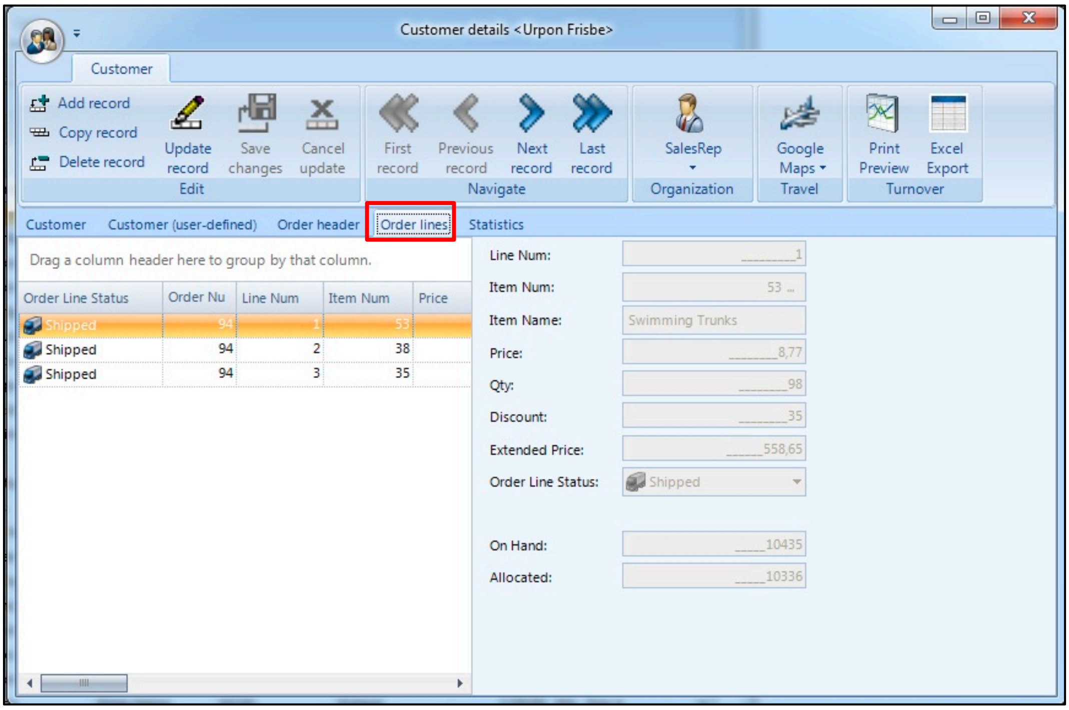
| Order Status | Order Num | Cust Num | Name | City | Ordered |
|---|---|---|---|---|---|
| Shipped | 94 | 2 | Urpon Frisbe | Oslo | 01.10.2008 |
| Shipped | 125 | 2 | Urpon Frisbe | Oslo | 17.12.2008 |
| Shipped | 171 | 2 | Urpon Frisbe | Oslo | 27.11.2008 |
| Ordered | 1355 | 2 | Urpon Frisbe | Oslo | 28.11.2008 |
| Ordered | 1375 | 2 | Urpon Frisbe | Oslo | 07.11.2008 |
| Ordered | 6060 | 2 | Urpon Frisbe | Oslo | 04.01.2009 |
| Ordered | 6070 | 2 | Urpon Frisbe | Oslo | 14.11.2008 |

| | |
|---|---|
| Order Num: | 94 |
| Cust Num: | 2 |
| Order Status: | Shipped |
| Ordered: | 01.10.2008 |
| Shipped: | 06.10.2008 |
| Promised: | 06.10.2008 |
| Instructions: | |
| PO: | |

For every order record you can show order line information as well.

- To see the order lines switch to tab **Order lines**.

The result should look like this:



| Order Line Status | Order Nu | Line Num | Item Num | Price |
|---|---|---|---|---|
| Shipped | 94 | 1 | 53 | |
| Shipped | 94 | 2 | 38 | |
| Shipped | 94 | 3 | 35 | |

| | |
|---|---|
| Line Num: | 1 |
| Item Num: | 53 |
| Item Name: | Swimming Trunks |
| Price: | 8,77 |
| Qty: | 98 |
| Discount: | 35 |
| Extended Price: | 558,65 |
| Order Line Status: | Shipped |
| On Hand: | 10435 |
| Allocated: | 10336 |

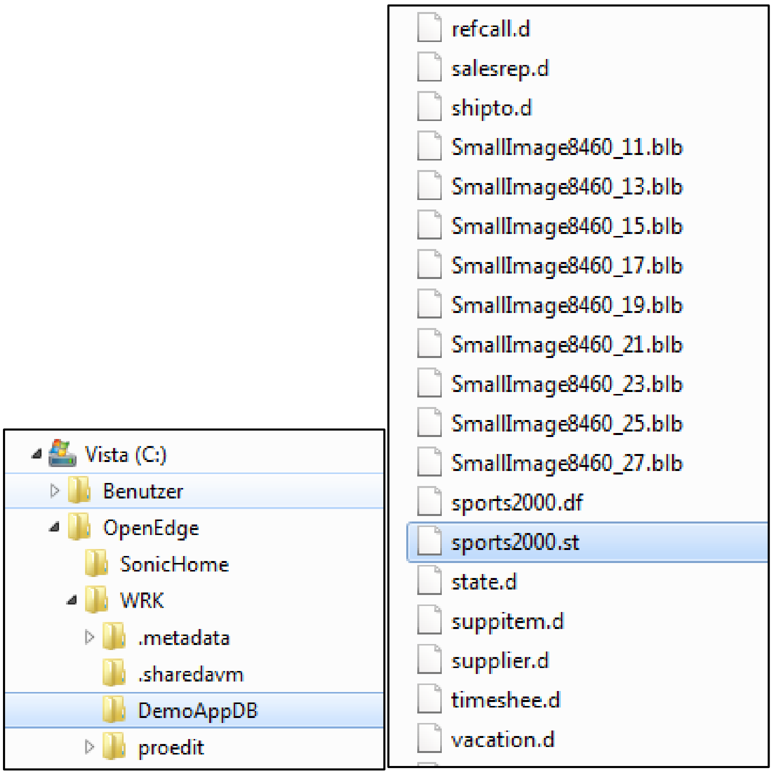
There is more to discover in the CustomerExplorer demo. Feel free!

## Create the sports2000 database

The CustomerExplorer sample application and the labs in this tutorial are based on an enhanced version of Progress Software’s sports2000 database. This dataset is enhanced with some additional fields and tables, e.g. for storing images of the salesreps. You will need to setup and connect this database for running the CustomerExplorer application and the labs in this tutorial.

If not done so far download the file *CustomerExplorer_DB_yyyymmdd_src.zip* from the download section of the Consultingwerk Developers Corner and copy it to a directory of your choice.

Create a folder **DemoAppDB** in your *OpenEdge\WRK* directory and unzip the file.



- Vista (C:)
  - Benutzer
  - OpenEdge
    - SonicHome
    - WRK
      - .metadata
      - .sharedavm
      - DemoAppDB
      - proedit

refcall.d
salesrep.d
shipto.d
SmallImage8460_11.blb
SmallImage8460_13.blb
SmallImage8460_15.blb
SmallImage8460_17.blb
SmallImage8460_19.blb
SmallImage8460_21.blb
SmallImage8460_23.blb
SmallImage8460_25.blb
SmallImage8460_27.blb
sports2000.df
sports2000.st
state.d
suppitem.d
supplier.d
timeshee.d
vacation.d

The structure description file (.st) is required to create a new void database, the data definition file (.df) is needed to load the logical schema and the dumpfiles (.d) contain the data.

Note that there are some binary files as well (.blb), that contain the images used in the demo application.

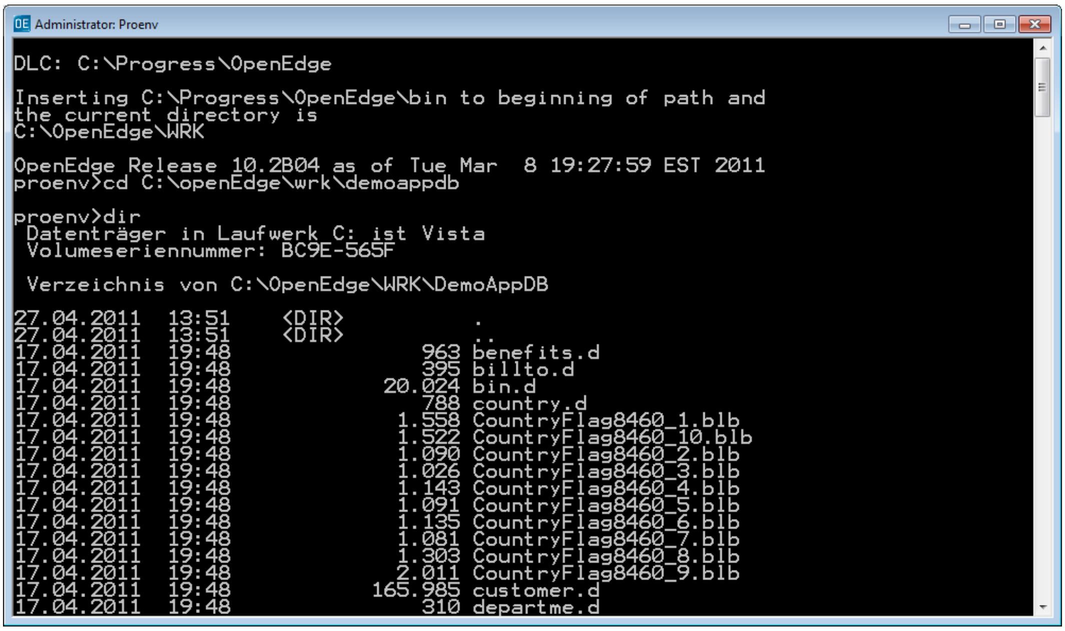
## Create a void empty Database using the sports2000.st file

To create the sports2000 database you use the **proenv** shell from Progress Software. It's a command line shell that sets up the path to the OpenEdge RDBMS executables and by default switches to the working directory */wrk* (or whatever was chosen during the installation of OpenEdge on your computer). It enables you to invoke many database administration utilities and commands.

- To open **proenv** choose **Start > All Programs > OpenEdge > proenv**.

⚠ When using **proenv**, you should be mindful of its working directory because it is where your database files are stored by default. You can change the *proenv working directory* to any directory of your choice. To do so, use the CD command.

- Change directory to **C:\OpenEdge\WRK\DemoAppDB**.

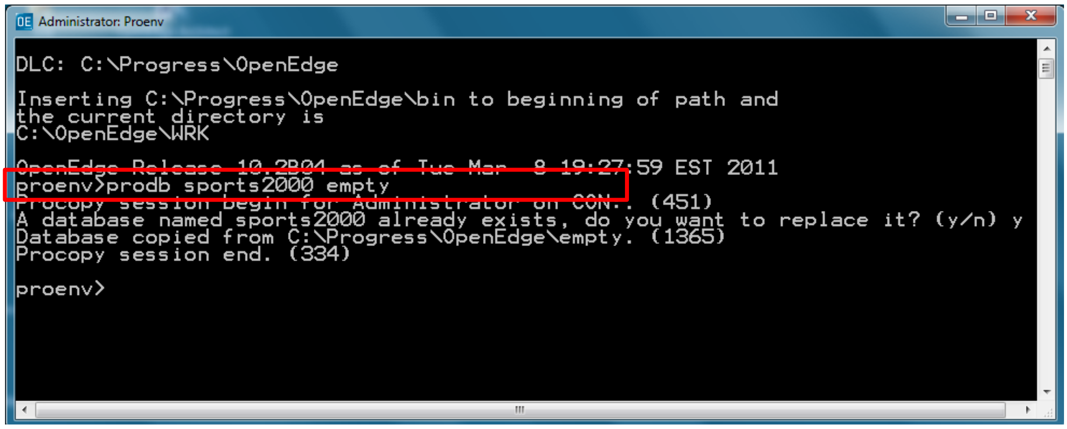


```
DLC: C:\Progress\OpenEdge

Inserting C:\Progress\OpenEdge\bin to beginning of path and
the current directory is
C:\OpenEdge\WRK

OpenEdge Release 10.2B04 as of Tue Mar  8 19:27:59 EST 2011
proenv>cd C:\openEdge\wrk\demoappdb

proenv>dir
 Datenträger in Laufwerk C: ist Vista
 Volumeseriennummer: BC9E-565F

 Verzeichnis von C:\OpenEdge\WRK\DemoAppDB

27.04.2011  13:51    <DIR>          .
27.04.2011  13:51    <DIR>          ..
17.04.2011  19:48               963 benefits.d
17.04.2011  19:48               395 billto.d
17.04.2011  19:48            20.024 bin.d
17.04.2011  19:48               788 country.d
17.04.2011  19:48             1.558 CountryFlag8460_1.blb
17.04.2011  19:48             1.522 CountryFlag8460_10.blb
17.04.2011  19:48             1.090 CountryFlag8460_2.blb
17.04.2011  19:48             1.026 CountryFlag8460_3.blb
17.04.2011  19:48             1.143 CountryFlag8460_4.blb
17.04.2011  19:48             1.091 CountryFlag8460_5.blb
17.04.2011  19:48             1.135 CountryFlag8460_6.blb
17.04.2011  19:48             1.081 CountryFlag8460_7.blb
17.04.2011  19:48             1.303 CountryFlag8460_8.blb
17.04.2011  19:48             2.011 CountryFlag8460_9.blb
17.04.2011  19:48           165.985 customer.d
17.04.2011  19:48               310 departme.d
```

- Create a new, empty sports2000 database using the PRODB command.

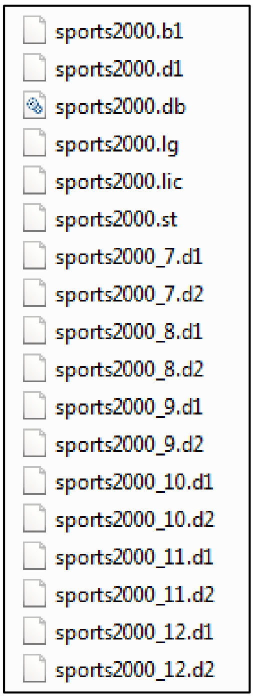
The PRODB uses the sports2000.st file automatically when it is saved in the same directory where the database resides

```
prodb sports2000 empty
```



```
DLC: C:\Progress\OpenEdge

Inserting C:\Progress\OpenEdge\bin to beginning of path and
the current directory is
C:\OpenEdge\WRK

OpenEdge Release 10.2B04 as of Tue Mar  8 19:27:59 EST 2011
proenv>prodb sports2000 empty
Procopy session begin for Administrator on CON:. (451)
A database named sports2000 already exists, do you want to replace it? (y/n) y
Database copied from C:\Progress\OpenEdge\empty. (1365)
Procopy session end. (334)

proenv>
```

The following files have been added to your *WRK\DemoAppDB* directory.



```
sports2000.b1
sports2000.d1
sports2000.db
sports2000.lg
sports2000.lic
sports2000.st
sports2000_7.d1
sports2000_7.d2
sports2000_8.d1
sports2000_8.d2
sports2000_9.d1
sports2000_9.d2
sports2000_10.d1
sports2000_10.d2
sports2000_11.d1
sports2000_11.d2
sports2000_12.d1
sports2000_12.d2
```

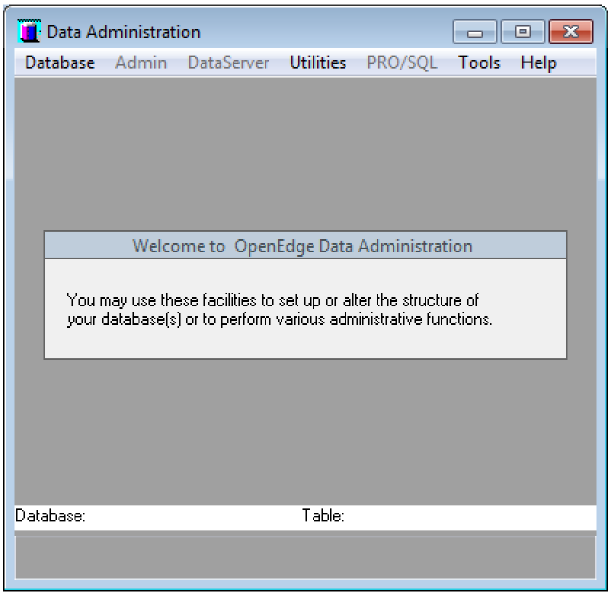
## Loading data definitions

Once you have created the target database, you can load the dump files into it. Load the data definition file *sports2000.df* into the database first.

You can use the *Data Administration* tool for load operations.

- To open Data Administration choose **Start > All Programs > OpenEdge > Data Administration**.

The *Data Administration* tool opens.



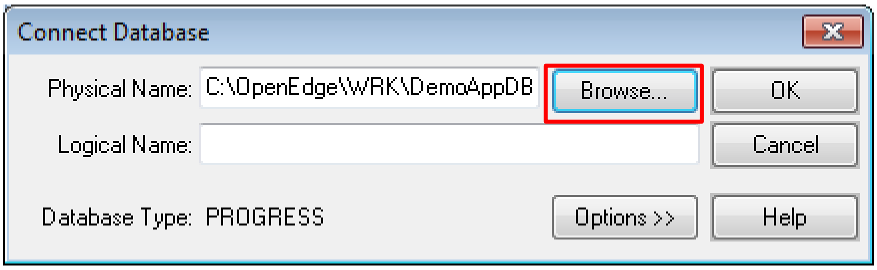
Welcome to OpenEdge Data Administration

You may use these facilities to set up or alter the structure of your database(s) or to perform various administrative functions.

Database:

Table:

- From the menu choose **Database > connect** and connect to your *sports2000.db* database by selecting the sports2000.db file and click **OK**.

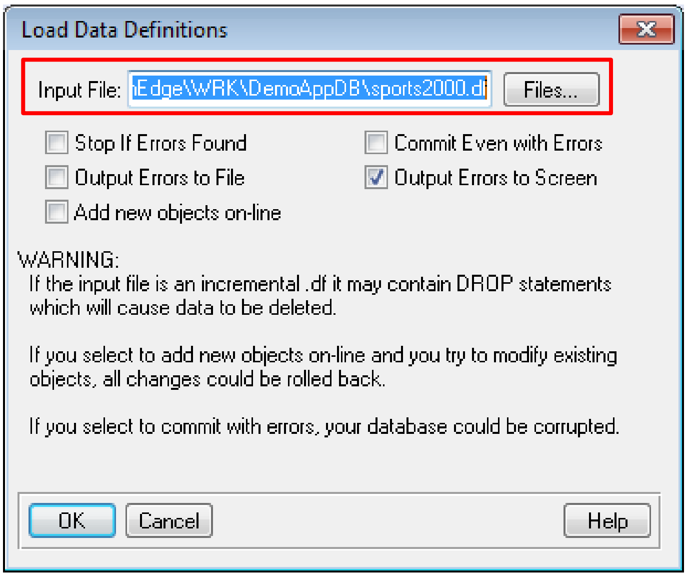


Physical Name: C:\OpenEdge\WRK\DemoAppDB

Logical Name:

Database Type: PROGRESS

- From the menu choose **Admin > Load Data and Definition > Data Definitions (.df file)**.
- The **Load Data Definitions** dialog box opens and shows *WRK\DemoAppDB\sports2000.df* as the default input file because it is stored in the same directory where the database resides.



Load Data Definitions

Input File: \Edge\WRK\DemoAppDB\sports2000.df

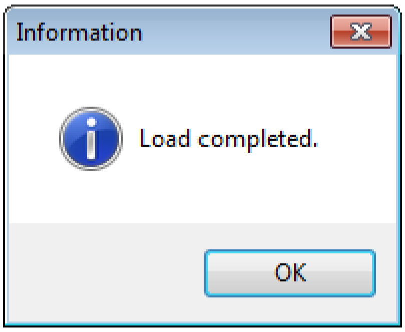
□ Stop If Errors Found
□ Commit Even with Errors
□ Output Errors to File
☑ Output Errors to Screen
□ Add new objects on-line

WARNING:
If the input file is an incremental .df it may contain DROP statements which will cause data to be deleted.

If you select to add new objects on-line and you try to modify existing objects, all changes could be rolled back.

If you select to commit with errors, your database could be corrupted.

- Click **OK** to load the data definitions of sports2000.df.
- Click **OK** on the **Information** box to return to the main *Data Administration* interface.



Information

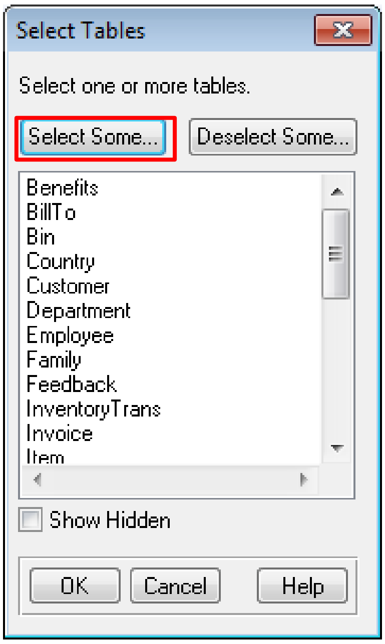
Load completed.

## Loading table contents

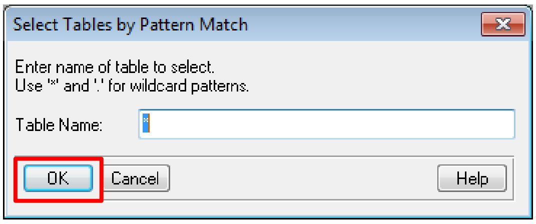
After you have loaded the data definitions of all tables into the target database, you can load the table contents.

- Choose **Admin > Load Data and Definitions > Table Contents (.d files).**

The **Select Tables** dialog box opens, showing an alphabetical list of all the tables in the database.



Select one or more tables.



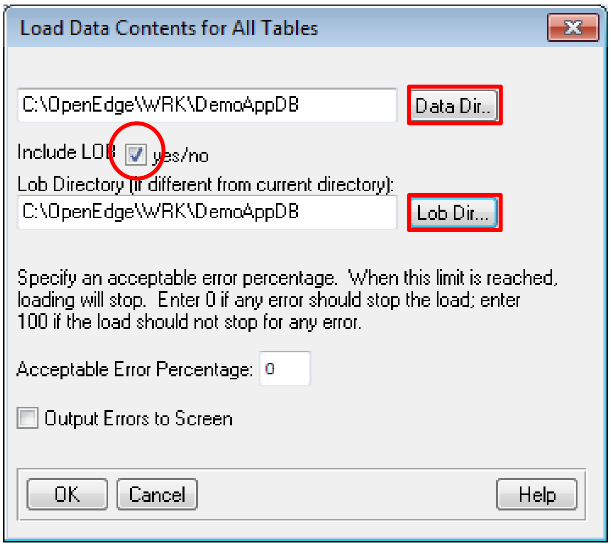
Enter name of table to select.
Use '*' and '.' for wildcard patterns.

Table Name: *

- Click button **Select Some** to select all tables and click **OK**.
- Click **OK** to close the **Select Tables** dialog box.

The **Load Data Contents for All Tables** dialog box opens. By default, the input directory is empty.

- Click **Data Dir**.. to change the input directory to your *WRK\DemoAppDB* directory.
- Click **Lob Dir**.. to change the input directory to your *WRK\DemoAppDB* directory.



Load Data Contents for All Tables

C:\OpenEdge\WRK\DemoAppDB

Include LOB ☑ yes/no

Lob Directory (if different from current directory):

C:\OpenEdge\WRK\DemoAppDB

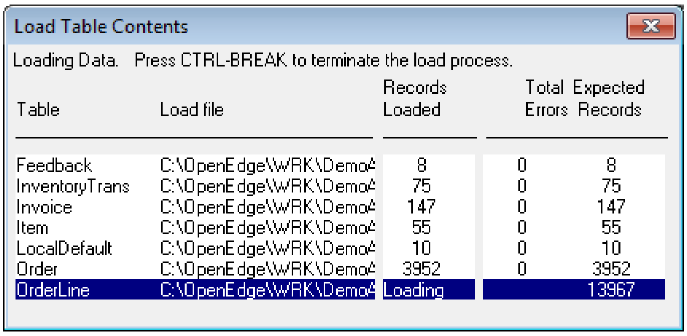
Specify an acceptable error percentage. When this limit is reached, loading will stop. Enter 0 if any error should stop the load; enter 100 if the load should not stop for any error.

Acceptable Error Percentage: 0

□ Output Errors to Screen

There are several Binary Large Objects (BLOBs) in the *WRK\DemoAppDB* directory that are important for the CustomerExplorer demo application. Make sure that **Include LOB** toggle box is checked.

- Click **OK** to load the data.

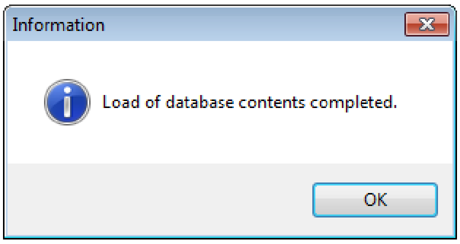


Load Table Contents

Loading Data. Press CTRL-BREAK to terminate the load process.

| Table | Load file | Records Loaded | Total Errors | Expected Records |
|---|---|---|---|---|
| Feedback | C:\OpenEdge\WRK\DemoA | 8 | 0 | 8 |
| InventoryTrans | C:\OpenEdge\WRK\DemoA | 75 | 0 | 75 |
| Invoice | C:\OpenEdge\WRK\DemoA | 147 | 0 | 147 |
| Item | C:\OpenEdge\WRK\DemoA | 55 | 0 | 55 |
| LocalDefault | C:\OpenEdge\WRK\DemoA | 10 | 0 | 10 |
| Order | C:\OpenEdge\WRK\DemoA | 3952 | 0 | 3952 |
| OrderLine | C:\OpenEdge\WRK\DemoA | Loading |  | 13967 |

- When the load is completed click **OK** to return to the main interface.



Information

Load of database contents completed.

## Loading sequence definitions and values

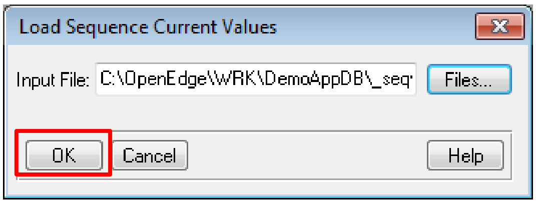
After you have loaded the table contents into the target database, you can load the current sequence definition and values for the database.

The sequence definitions have been loaded with the data definition so you just need to load the sequence current values.

- To load the sequence values choose **Admin > Load Data and Definitions > Sequence Current Values** from the menu.

The **Load Sequence Current Values** dialog box opens showing a *_seqvals.d* as the default input file.

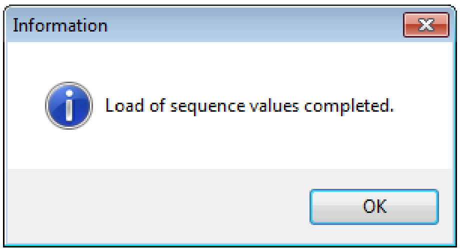
- Change to your *WRK\DemoAppDB* directory and select the **_seqvals.d** file.



Load Sequence Current Values

Input File: C:\OpenEdge\WRK\DemoAppDB_seq

- Click **OK** to load the current sequence values.



Information

Load of sequence values completed.

- Click **OK** to return to the main interface.
- Choose **Database > Exit** to close the *Data Administration* tool.

Congratulation! You successfully finished the creation of the sports2000 database.

## OERA vs. OERA

The SmartComponent Library deployment contains two folders named *OERA*. Even though this may be confusing at first sight, this is a design decision made that we do not intend to change.

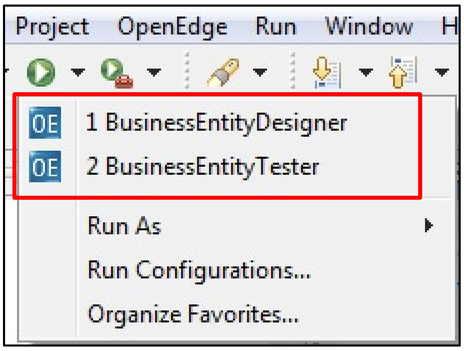
1) The first OERA folder is the folder Consultingwerk\OERA. This folder contains the class based OpenEdge Reference Architecture compliant backend implementation. This folder contains the classes contained in the Consultingwerk.OERA package and this folder must remain in this location.
2) The second OERA folder is the folder OERA under the deployment root folder. This folder contains the service interface support procedures. These are the procedures invoked from (GUI) clients for accessing the backend through AppServer calls. The location of this folder is variable and customers may move this to another folder (even to the Consultingwerk\OERA folder as there are no ambiguous files in the two folders). One reason to move this folder to a different location may be security related. Through the use of the SESSION:EXPORT method (see the SESSION system handle in the OpenEdge documentation) customers can restrict the access from clients to the AppServer only to specific procedures or folders. Thus it may be desired to merge the contents of this folder with another folder already exposed to clients using that way. The OERASI setting in the Consultingwerk/products.i file can be changed to point to the correct location of the service interface support procedures:

```
/* default path to OERA Service Interface */
&GLOBAL-DEFINE OERASI OERA/support
```

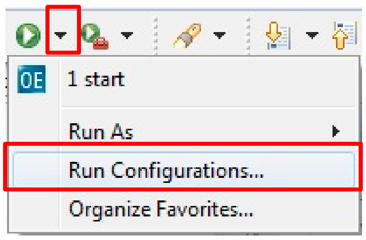
## Changing Run Configurations

You can change the **Run configuration** of the OpenEdge Architect to list the start procedures for the *SmartComponent Library* tools in the **Run as**… drop down menu, e.g. BusinessEntityDesigner and BusinessEntityTester.

The result should look like this:

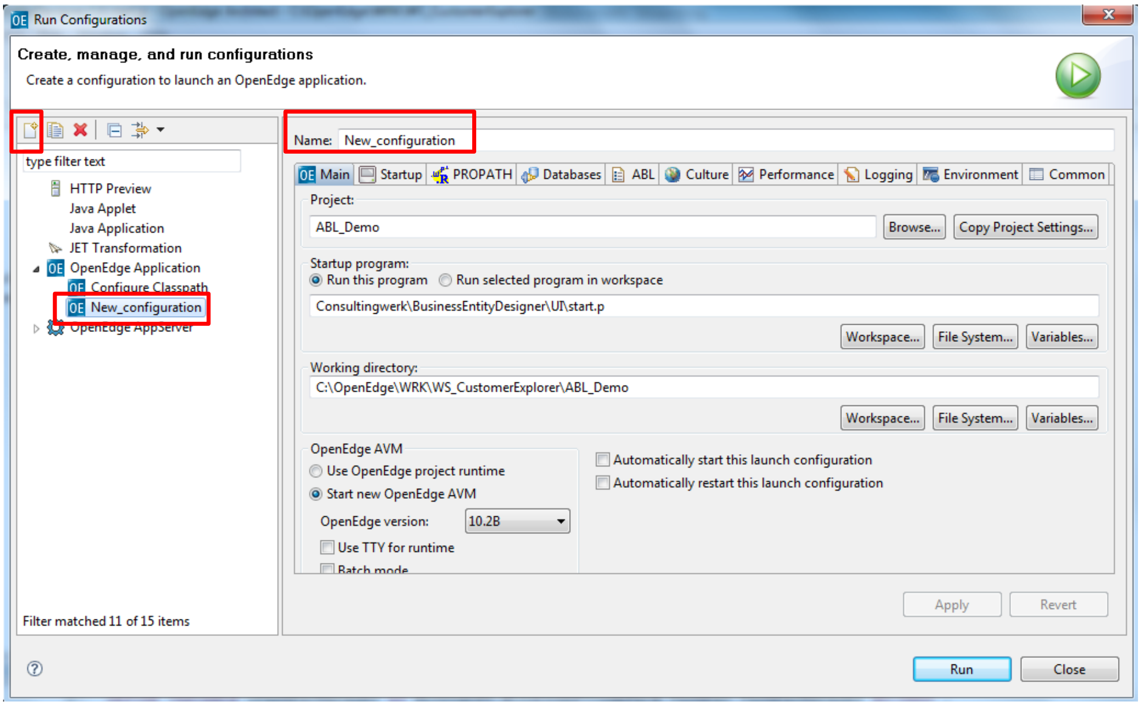


- With the target start.p procedure (*Consultingwerk\BusinessEntityDesigner\UI*) selected in the OpenEdge Architect editor pull down the **Run as…** menu and select **Run Configurations…**



The **Run Configurations** dialog opens.

- Press the New Button in the upper left corner to create a new Run Configuration



**Create, manage, and run configurations**

Create a configuration to launch an OpenEdge application.

type filter text

Filter matched 11 of 15 items

Name: New_configuration

Project: ABL_Demo

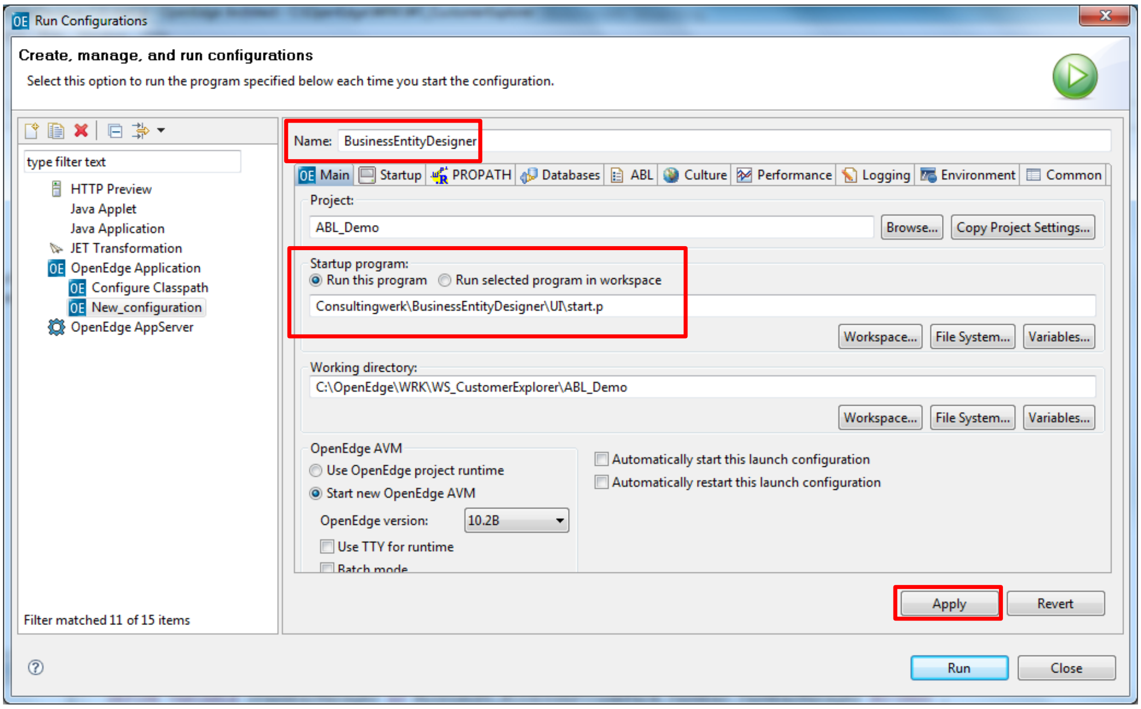
Startup program:
● Run this program ○ Run selected program in workspace
Consultingwerk\BusinessEntityDesigner\UI\start.p

Working directory:
C:\OpenEdge\WRK\WS_CustomerExplorer\ABL_Demo

OpenEdge AVM
○ Use OpenEdge project runtime
● Start new OpenEdge AVM
OpenEdge version: 10.2B
□ Use TTY for runtime
□ Batch mode

□ Automatically start this launch configuration
□ Automatically restart this launch configuration

- Type in the name of the procedure you want to define as a startup procedure, in this case BusinessEntityDesigner (no blanks)
- Check the **Startup program**
- Click Button **Apply**



**Create, manage, and run configurations**

Select this option to run the program specified below each time you start the configuration.

type filter text

Filter matched 11 of 15 items

Name: BusinessEntityDesigner

Project: ABL_Demo

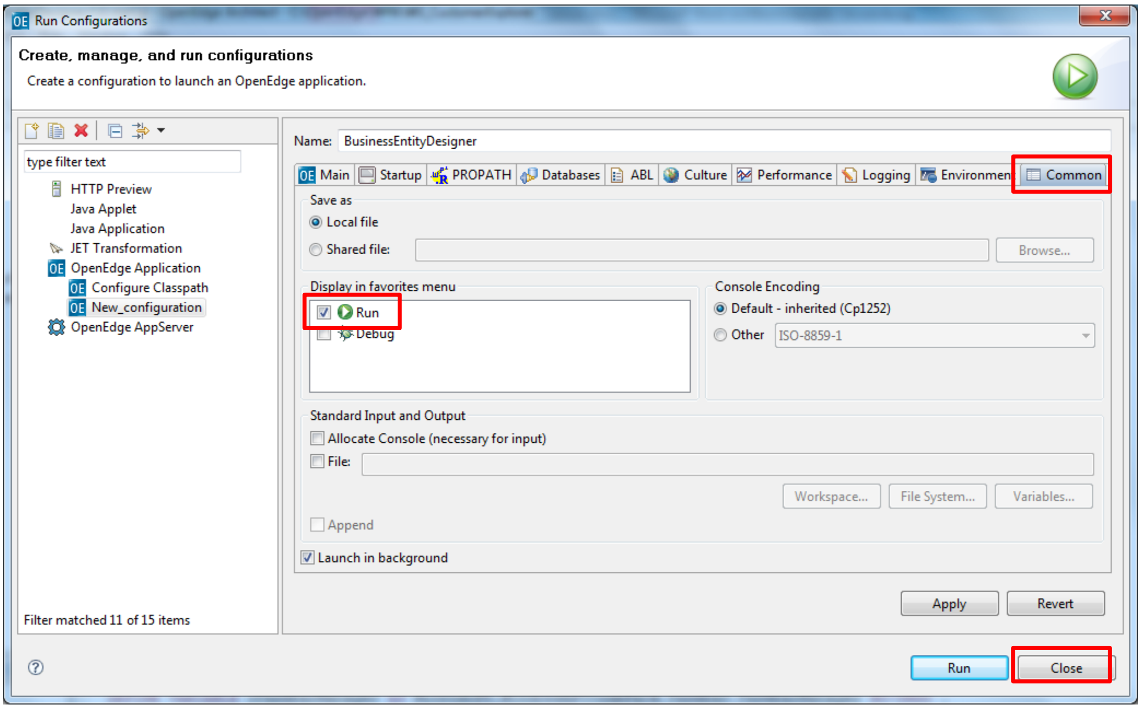
Startup program:
● Run this program ○ Run selected program in workspace
Consultingwerk\BusinessEntityDesigner\UI\start.p

Working directory:
C:\OpenEdge\WRK\WS_CustomerExplorer\ABL_Demo

OpenEdge AVM
○ Use OpenEdge project runtime
● Start new OpenEdge AVM
OpenEdge version: 10.2B
□ Use TTY for runtime
□ Batch mode

□ Automatically start this launch configuration
□ Automatically restart this launch configuration

- Change to folder **Common** and select the **Run** checkbox to *Display in favorites menu*.



Run Configurations

Create, manage, and run configurations

Create a configuration to launch an OpenEdge application.

Name: BusinessEntityDesigner

Save as
● Local file
○ Shared file:

Display in favorites menu
☑ Run
□ Debug

Console Encoding
● Default - inherited (Cp1252)
○ Other ISO-8859-1

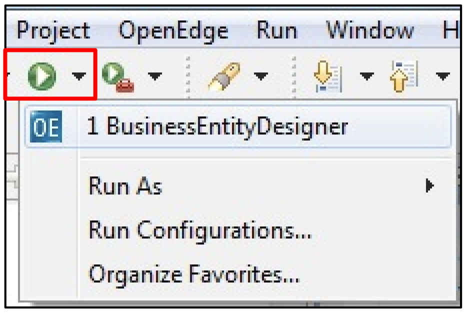
Standard Input and Output
□ Allocate Console (necessary for input)
□ File:
□ Append

☑ Launch in background

Filter matched 11 of 15 items

- Select **Close** to close the dialog.

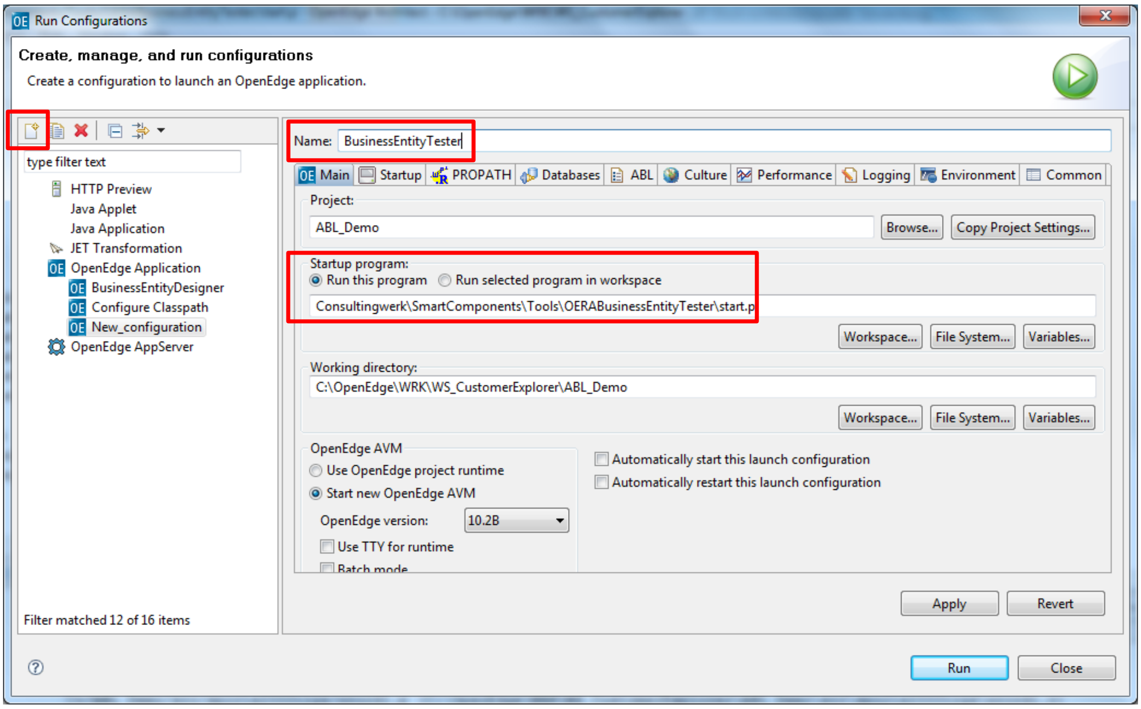
The new procedure is now shown in the start menu of the *Run as…* menu of the OpenEdge Architect.



- Repeat the steps to define the startup procedure for the **Business Entity Tester** for the Run menu as well.

The startup procedure for the **Business Entity Tester** you find here:

*C:\OpenEdge\WRK\WS_CustomerExplorer\ABL_Demo\Consultingwerk\SmartComponents\Tools\OERABusinessEntityTester*



Run Configurations

**Create, manage, and run configurations**

Create a configuration to launch an OpenEdge application.

Name: BusinessEntityTester

Project: ABL_Demo

Startup program:
● Run this program ○ Run selected program in workspace

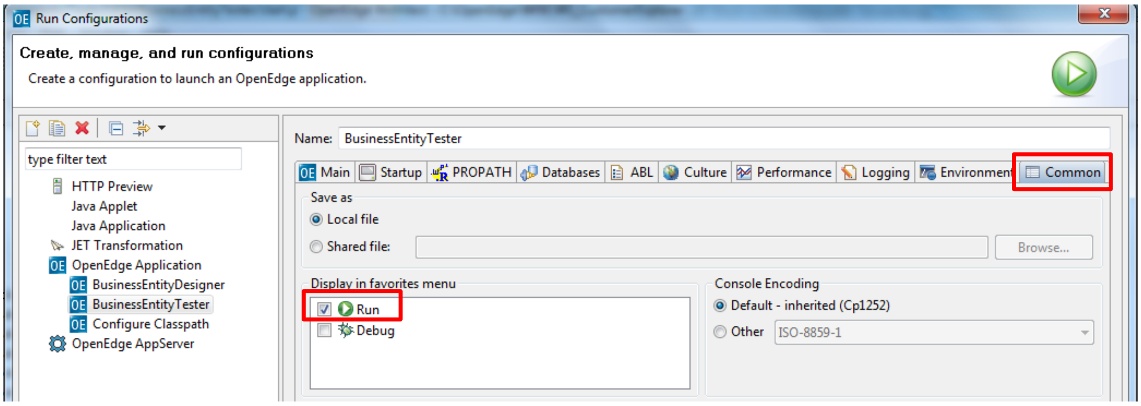
Consultingwerk\SmartComponents\Tools\OERABusinessEntityTester\start.p

Working directory: C:\OpenEdge\WRK\WS_CustomerExplorer\ABL_Demo

OpenEdge AVM
○ Use OpenEdge project runtime
● Start new OpenEdge AVM
OpenEdge version: 10.2B
□ Use TTY for runtime
□ Batch mode

□ Automatically start this launch configuration
□ Automatically restart this launch configuration

Filter matched 12 of 16 items



Run Configurations

**Create, manage, and run configurations**

Create a configuration to launch an OpenEdge application.

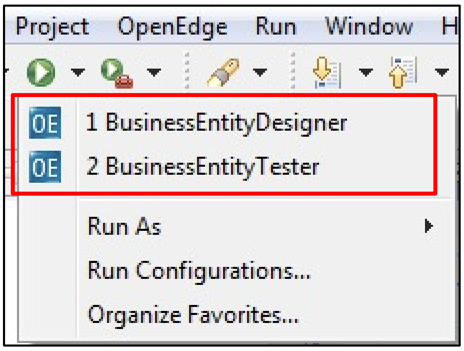
Name: BusinessEntityTester

Save as
● Local file
○ Shared file:

Display in favorites menu
☑ Run
□ Debug

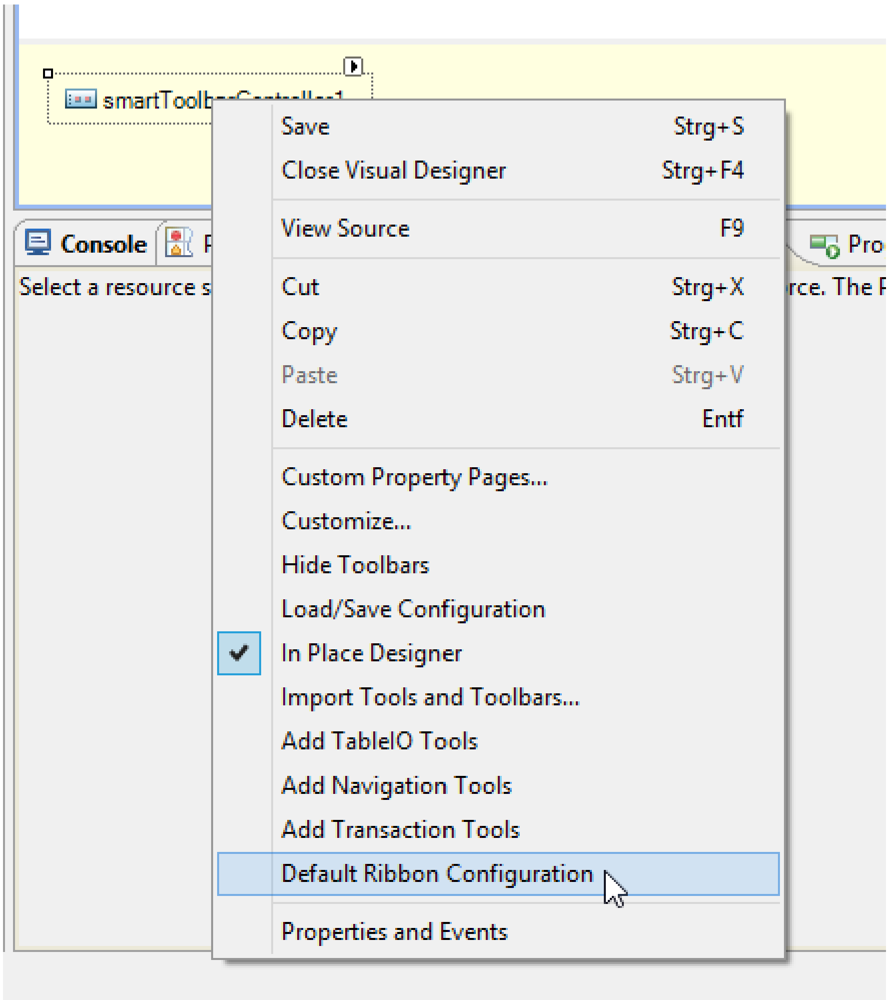
Console Encoding
● Default - inherited (Cp1252)
○ Other ISO-8859-1

The result should look like this:



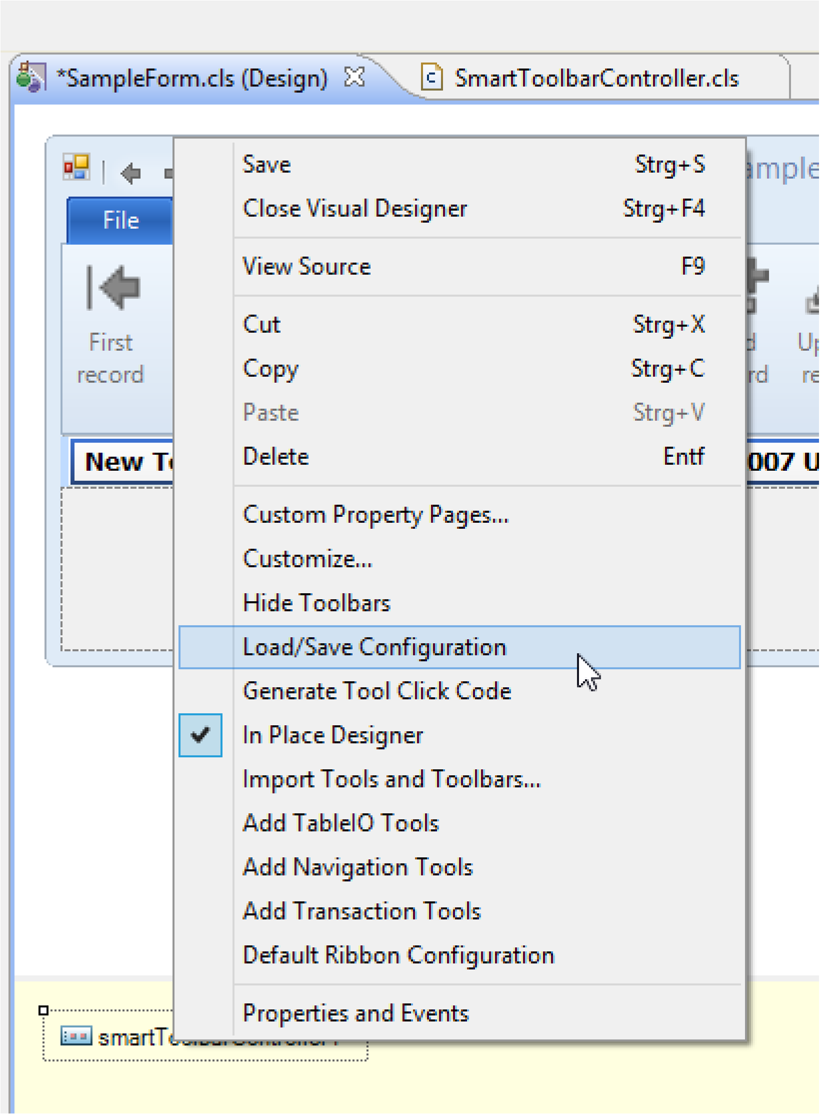
# Customizing the Default Ribbon Configuration

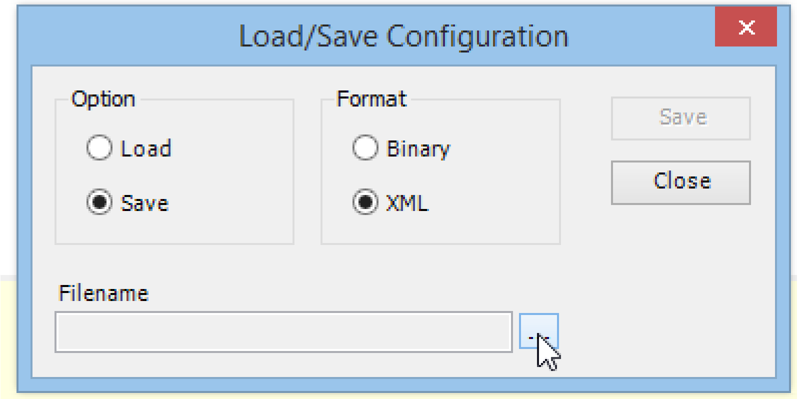
The SmartToolbarController provides a designer verb called “Default Ribbon Configuration”. This tool loads a standard ribbon layout containing the TableIO, Navigation and Transaction tools into the design time instance of the SmartToolbarController. Developers can use this layout as a foundation for Ribbons in screens they are working on.



A developer team may demand to modify this Ribbon design to match their own standards. Developer teams may also demand to add different default Ribbon layouts to the designer context menu (for master data screens, for transactional screens, …).

The  Default Ribbon Configuration functionality loads  a Ribbon layout stored in an Infragistics default XML file. The recommended approach to customize the Ribbon is to manipulate the Ribbon in a sample/mockup screen and  save the layout to an XML file from the Visual Designer:





Option: ○ Load ● Save

Format: ○ Binary ● XML

Filename:

Select a file/folder that will be accessible by all developers in your team.

To use your design from the Visual Designer context menu as the new default Ribbon, you should override the method “LoadDefaultRibbonConfiguration” of the SmartToolbarController class in your own Toolbar Controller custom base class (inheriting from SmartToolbarController).

Add code similar to the code from the original method and adjust the file name and path of your Ribbon XML design:

```
/*------------------------------------------------------------------------------
    Purpose: Loads a default Ribbon Design
    Notes:
------------------------------------------------------------------------------*/
METHOD PROTECTED VOID LoadDefaultRibbonConfiguration ():

    IF NOT THIS-OBJECT:DesignTime THEN
        RETURN .

    FILE-INFO:FILE-NAME =
"Consultingwerk/SmartComponents/Resources/RibbonTemplate.xml":U .

    IF FILE-INFO:FULL-PATHNAME > "":U THEN
        THIS-OBJECT:LoadFromXml(FILE-INFO:FULL-PATHNAME) .
    ELSE
        Consultingwerk.Util.MessageFormHelper:ShowMessage
            ('The file
"Consultingwerk/SmartComponents/Resources/RibbonTemplate.xml" cannot be found.',
             "SmartToolbarController",
             "Make sure it is available on your propath!",
             MessageFormImages:ImageWarning) .

END METHOD.
```

You can also add additional tools to the context menu providing quick access to different ribbon designs. Please refer to the SetDesignerProperties and OnVerbClicked methods in the SmartToolbarController for samples.