

Whidbey Hands-on-Labs

ADO.NET 2.0 - ObjectSpaces

Introduction

This hands-on lab introduces a new enhancement to ADO.NET called ObjectSpaces - the new Object/Relational Mapper included in Whidbey. In short, the Object/Relational Mapping system sits between the object-oriented model and the relational model and allows you to transform your objects into relational data and visa versa.

- The hands-on lab guides you in a number of steps through the basics of working with ObjectSpaces:
- Creating the Object Model Document
- Creating the Relational Model Document
- Creating the Mapping Document
- Creating and using the ObjectSpace

Time to complete the lab: 20 minutes

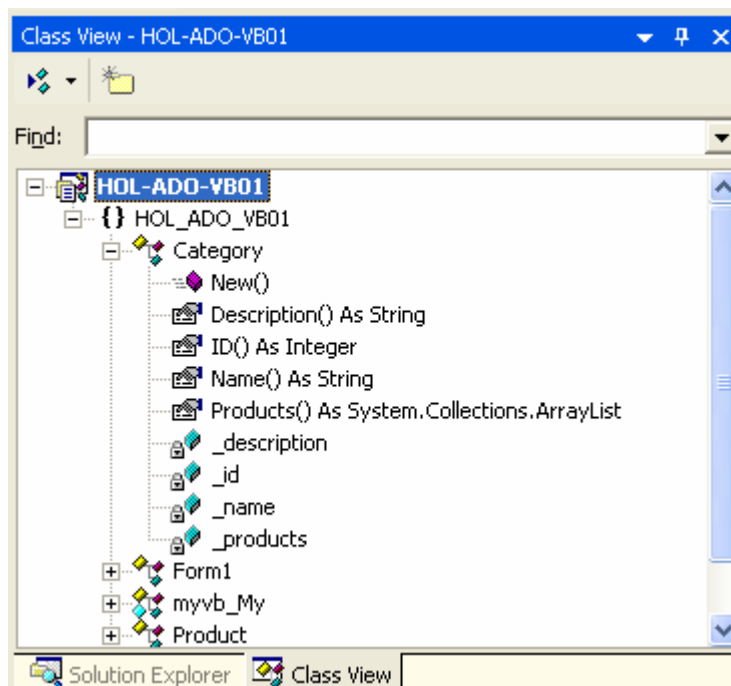
Before you start

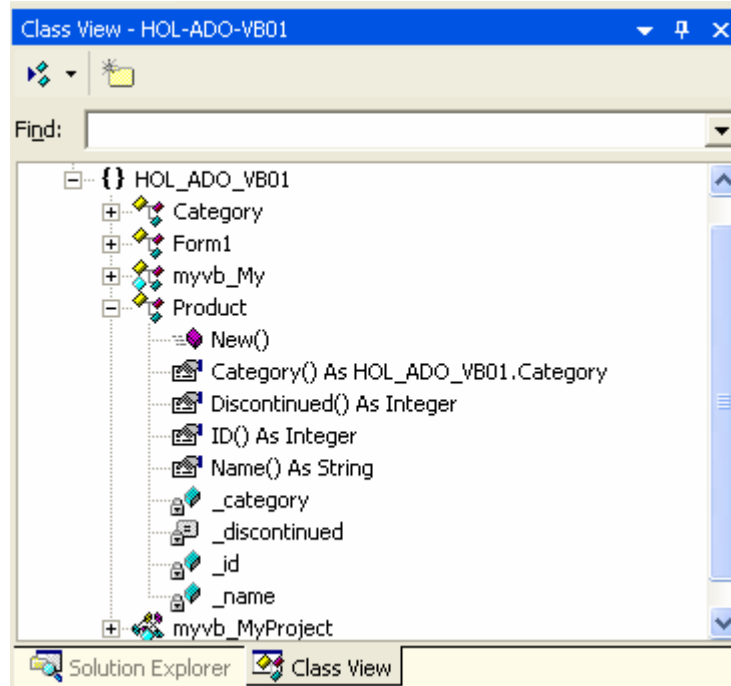
The starter project can be found on your desktop as the HOL-ADO-VB01 short-cut.

Exercise 1 – Preparing the OO Model

The starter project contains two classes, a category class having zero or more products.

- ▶ Open each of these classes and review their structure.



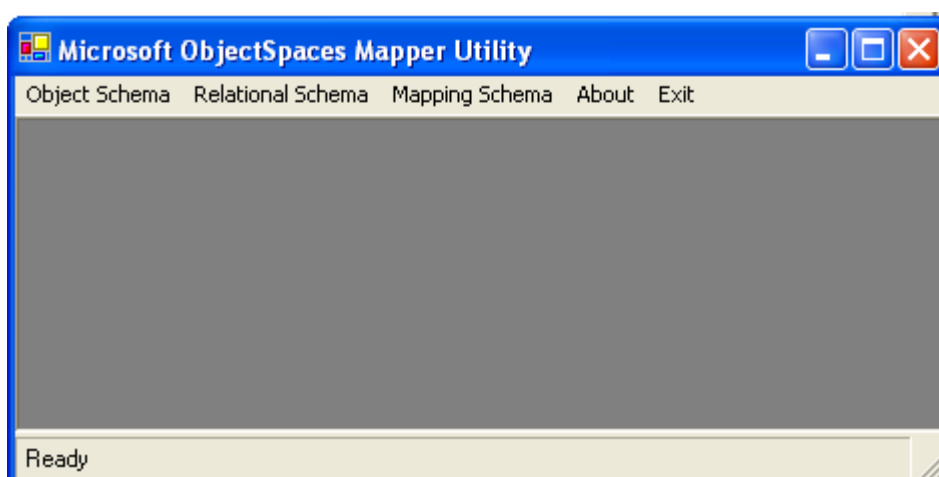


In order to map our data coming from a relational database to objects created from these classes (that is what ObjectSpaces is all about) you have to describe these classes in an XML file.

Your relational structure (that is the structure of the tables in e.g. SQL Server) also needs to be defined in an XML file.

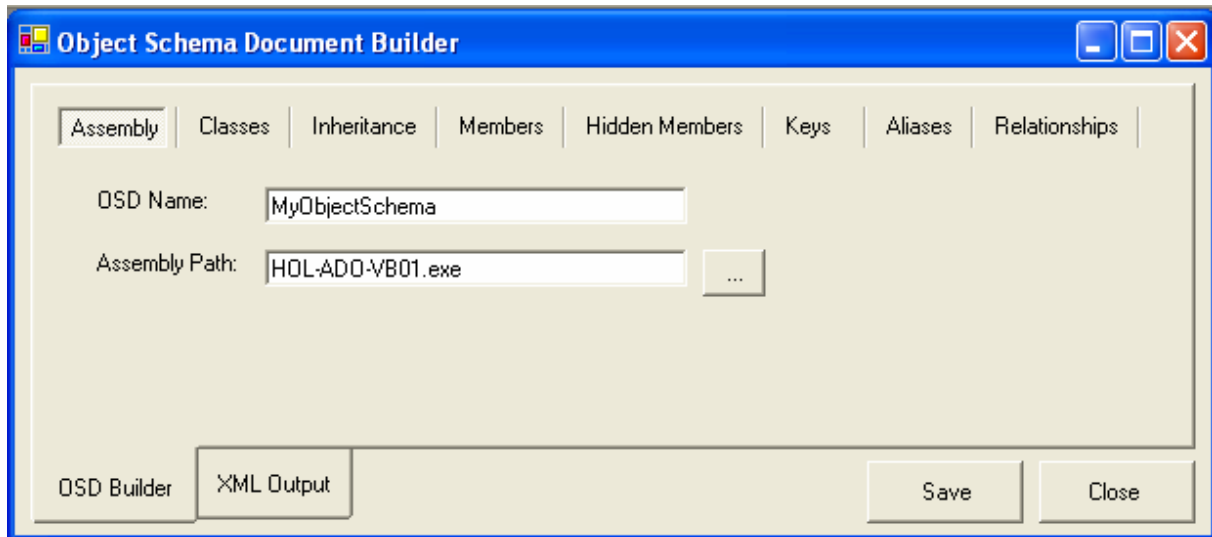
A third file, called the mapper file, will then store the mappings between the first XML file and the second one. You can create these files manually following a predefined schema or you can use a small utility downloadable from the GotDotNet site called the ObjectSpaces Mapper Utility.

- ▶ Start the "ObjectSpaces Mapper Utility"-tool by choosing to menu Start->All Programs->ObjectSpaces Mapper Utility->ObjectSpaces Mapper Utility.

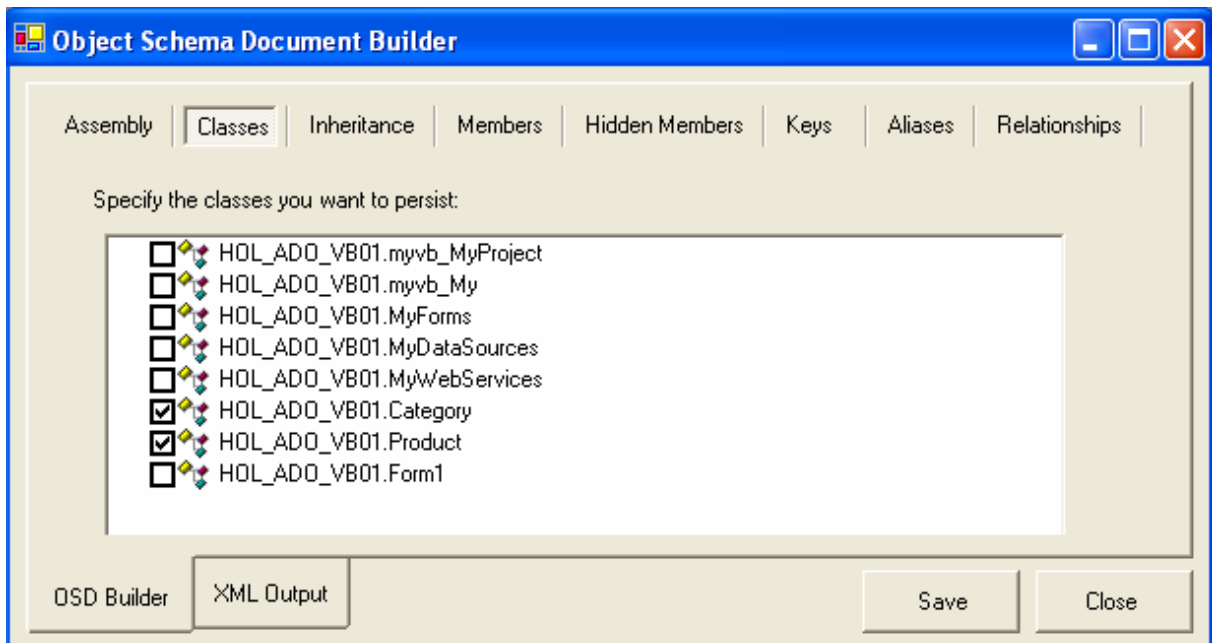


- ▶ Before you can use this tool, you will have to build your starter project. You can accomplish this by starting the project with F5.
- ▶ Go back to the utility and use the Object Schema->New menu.

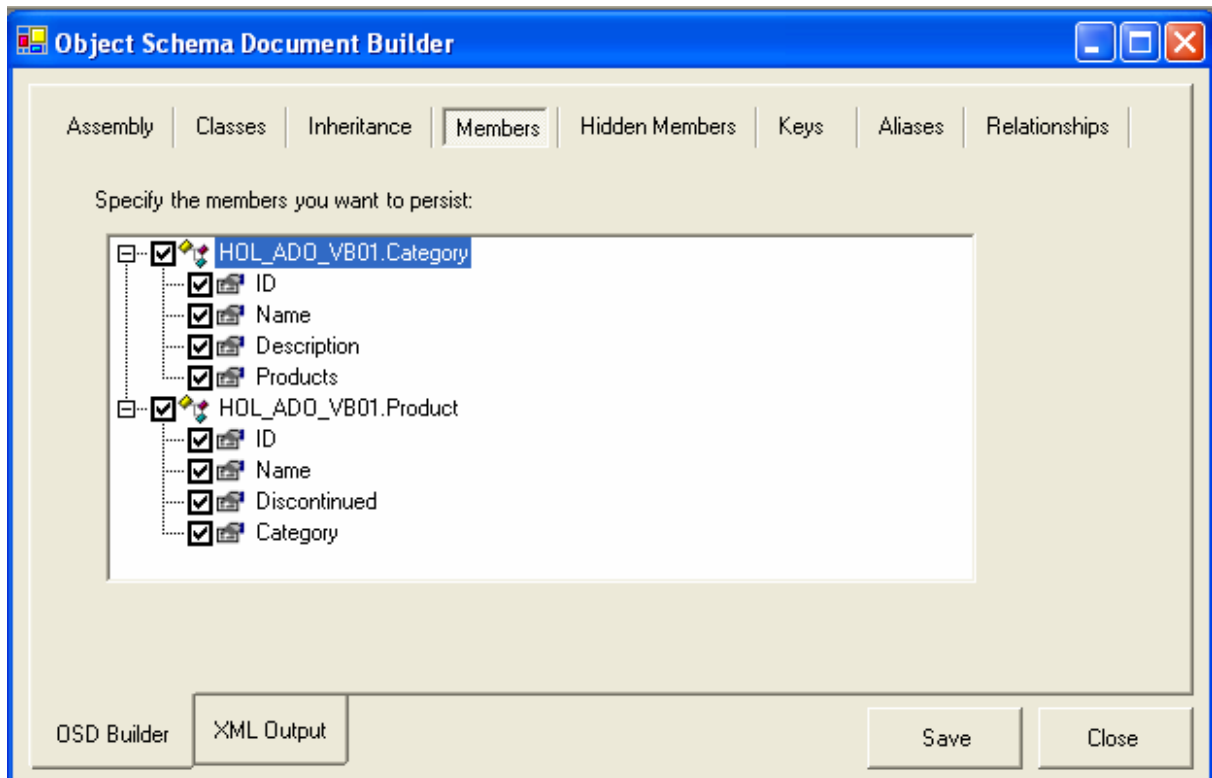
- ▶ Create a new Object Schema Document called products.osd.xml.
- ▶ Store the document in the C:\XML folder.
- ▶ Point to the assembly containing your Category and Product class.
The assembly can be found in the BIN folder of your starter project (C:\HOL-U2U\ADO.NET\HOL-ADO-VB01\BIN\HOL-ADO-VB01.EXE).



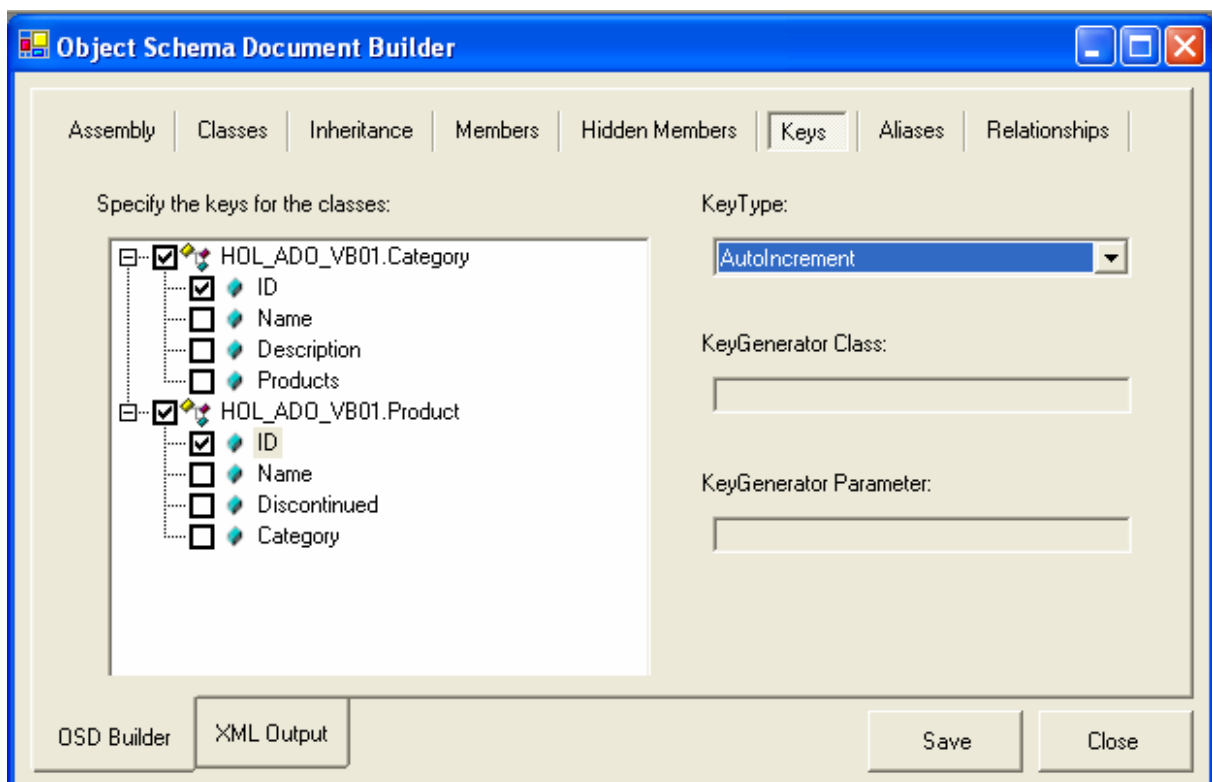
- ▶ Click on the Classes tab to select the classes in the assembly that will be mapped to the relational model.
- ▶ Make sure that both the Category and Product classes have been checked. Unselect the other classes.



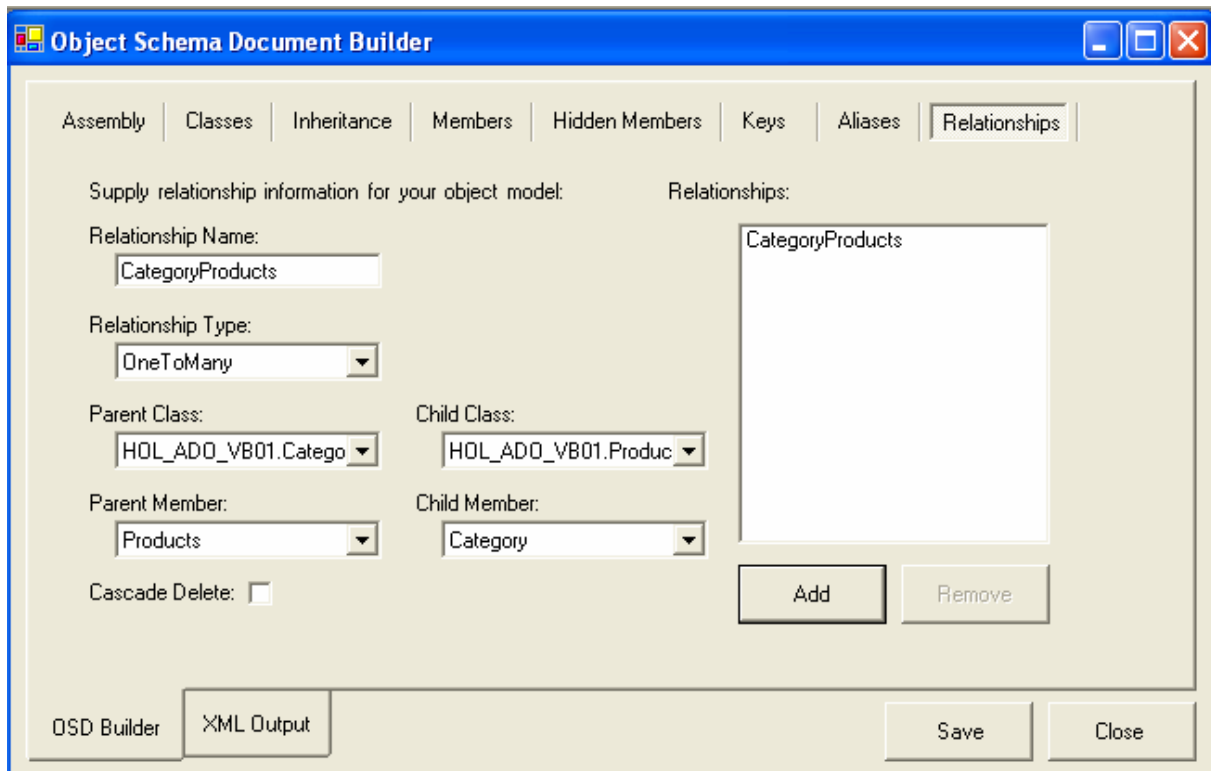
- ▶ Go to the Members tab. On this tab you select which members of each of the selected classes will be persisted. Normally each member will be selected, but make sure that this is the case.



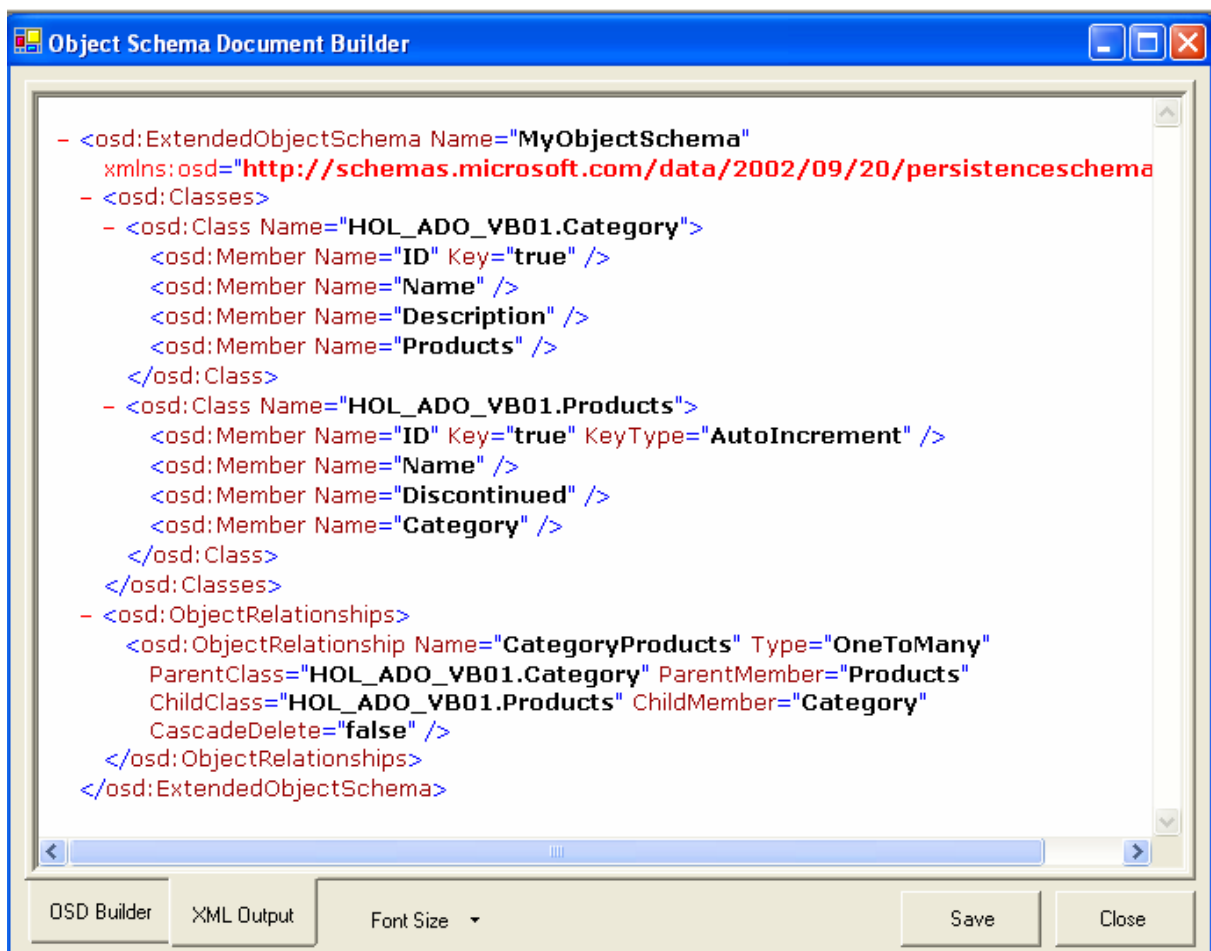
- Now define the keys of the objects on the Keys tab. Both the Category and the Product object must have ID as key as an auto increment key type.



- Finally define the relationship between the Category and Product entity using the Relationships tab: define the CategoryProduct relationship between Category and Product. You have to identify the members implementing the relationship as follows:



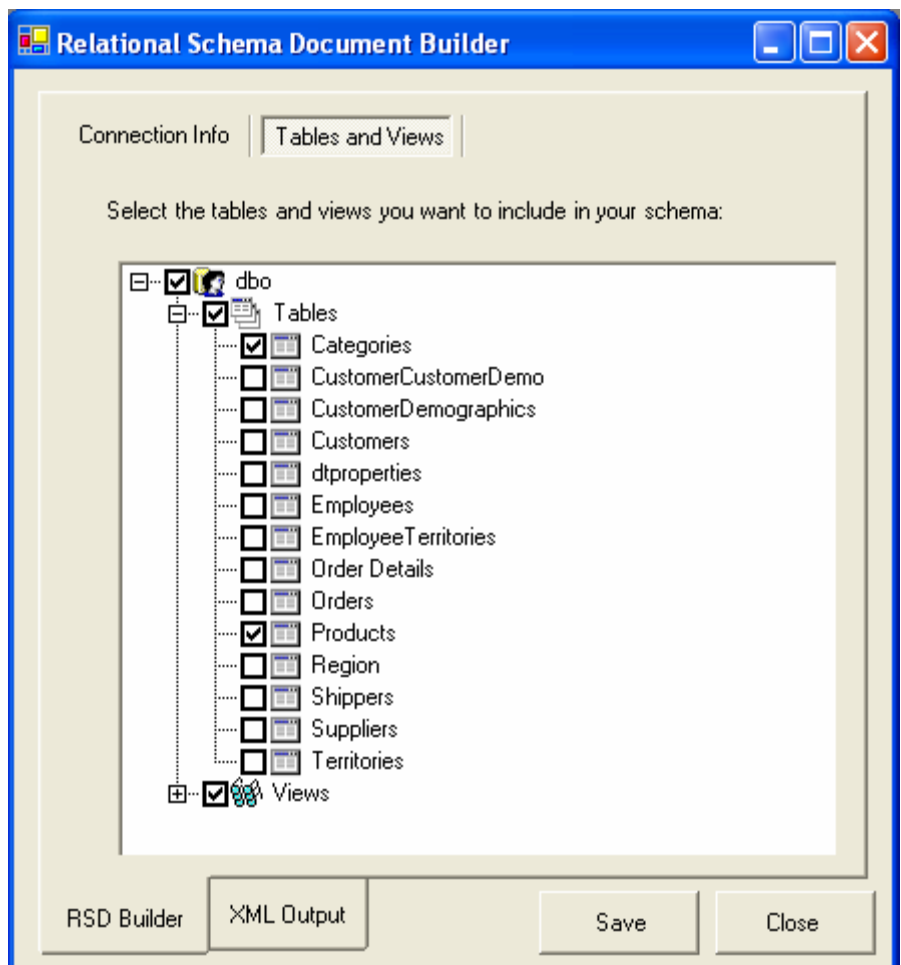
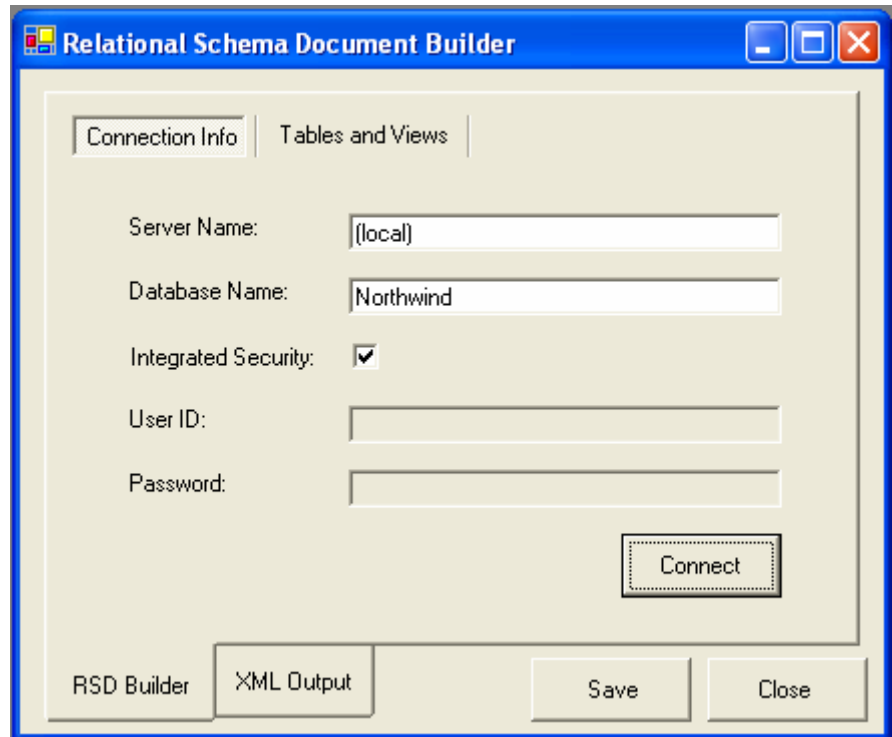
- ▶ Click the Save button ending up with the following xml schema.

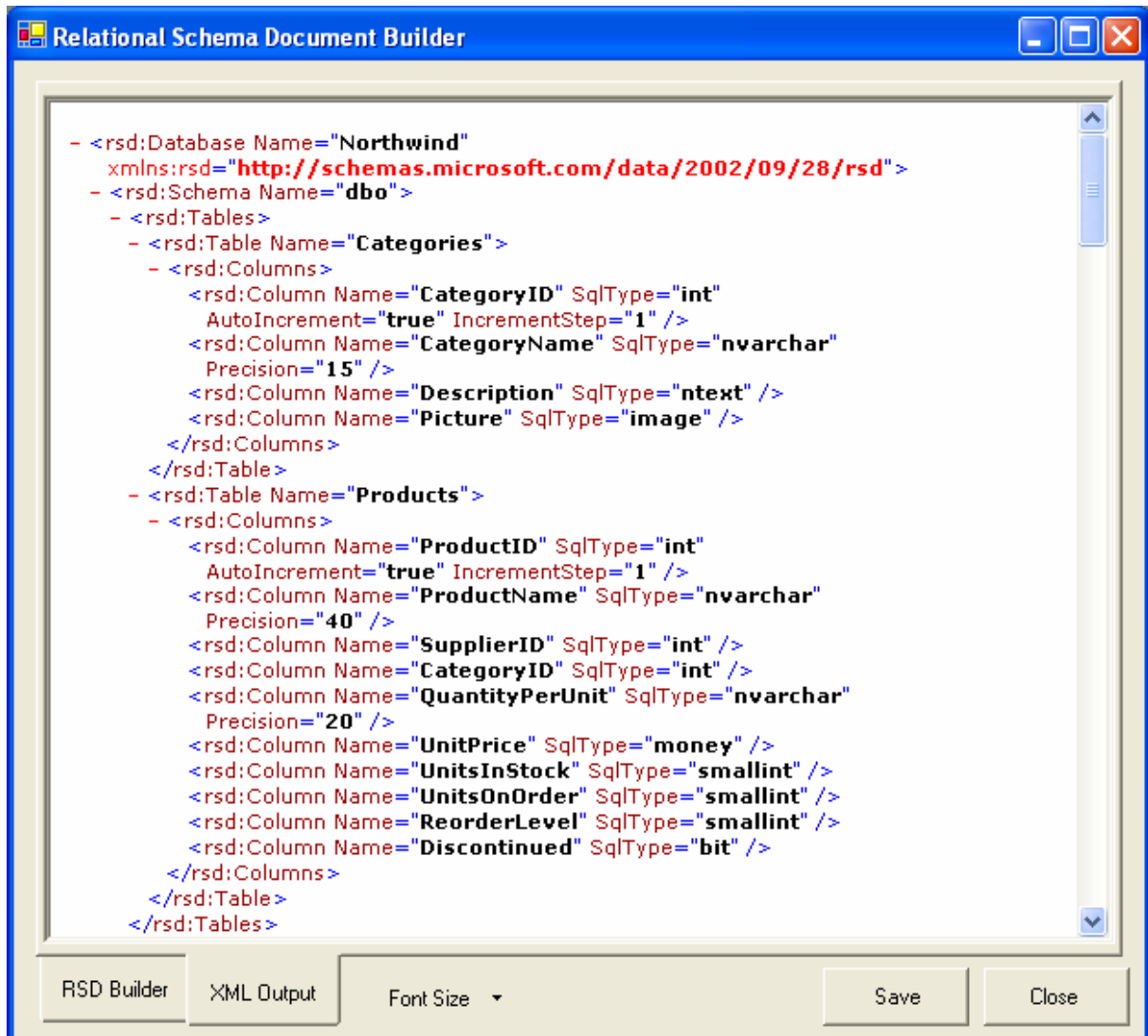


Exercise 2 – Preparing the Relational Model

In order to define the relational model, you will use the Categories and Products tables from the Northwind database. The only work you really need to do is to create an xml file describing the relational model. You will again use the ObjectSpaces Mapper Utility.

- ▶ Select the menu Relational Schema->New.
- ▶ Create a relational schema file with name myRelationalSchema.rsd.xml (again in the C:\XML folder).
- ▶ Enter the details for connecting to the SQL server database: Server Name and Database Name.
- ▶ Use the Connect button to verify the connection to the database.
- ▶ Next select the tables to use in the relational model; do this with the Tables and Views tab.
- ▶ Select the Categories and Products tables. Leave the other tables unchecked.
- ▶ Click the Save button and verify the XML output.

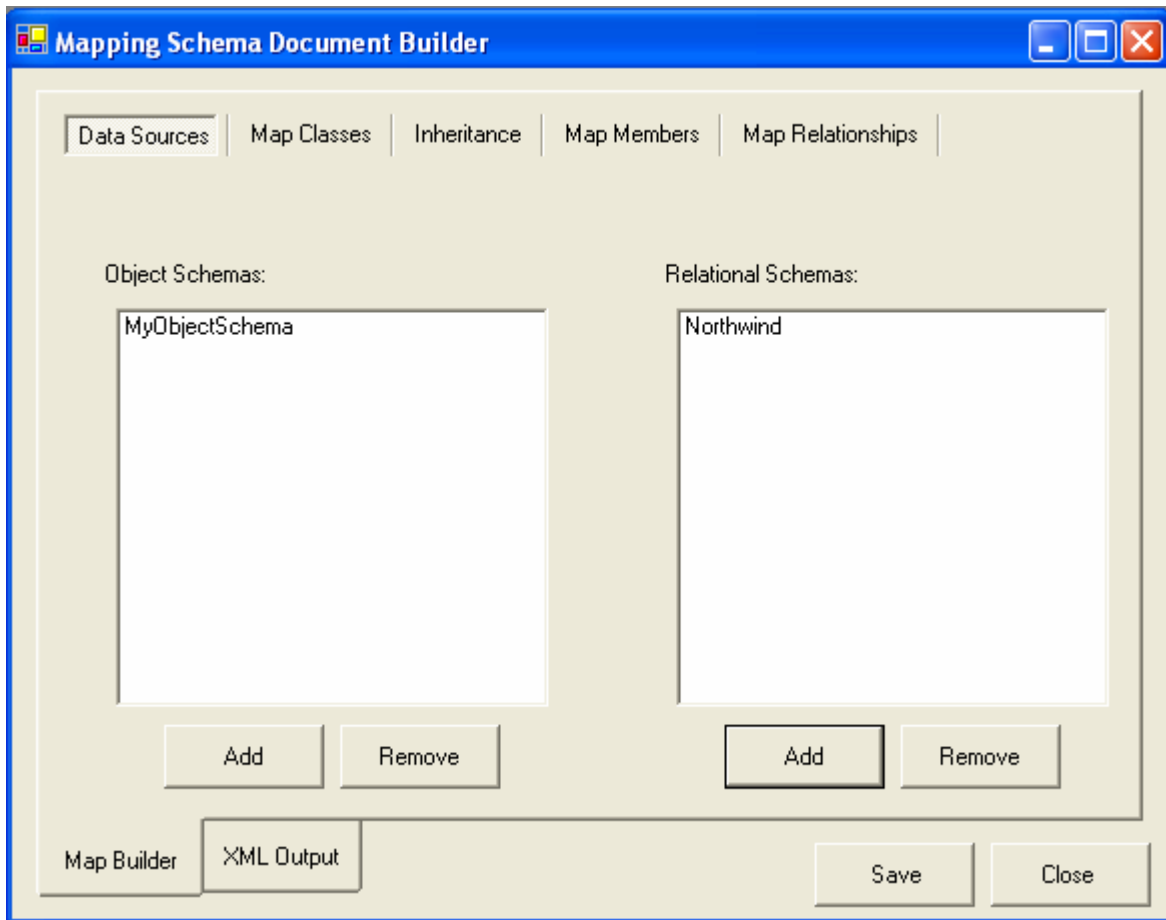




Exercise 3 – The Object/Relational Mapping

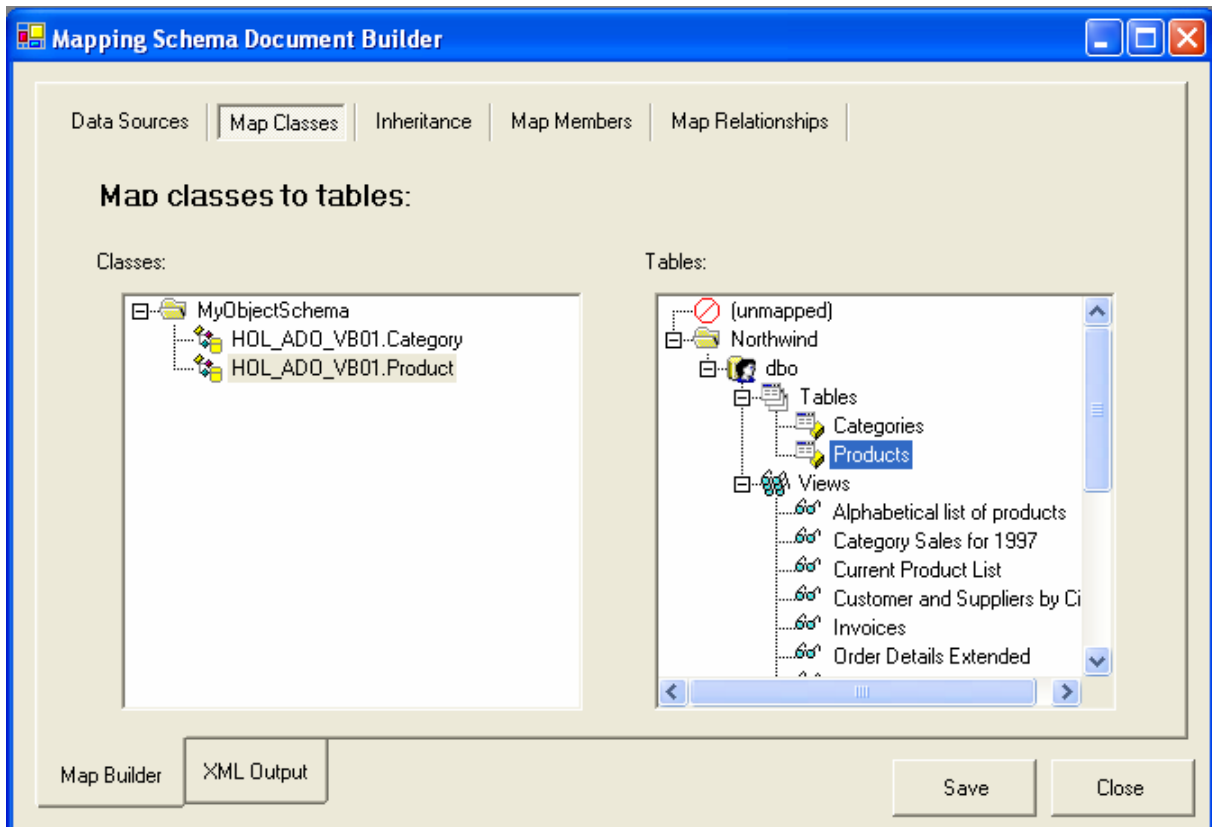
You have the Object Schema and Relational Schema Documents ready now. In this exercise you will combine them in a new Mapping Schema Document defining the actual Object/Relation Mapping.

- ▶ In the Mapper Utility select the menu Mapping Schema->New
- ▶ Create a myMapping.msd.xml file, again in the C:\XML folder
- ▶ Add the OSD and RSD files we created in the previous steps:

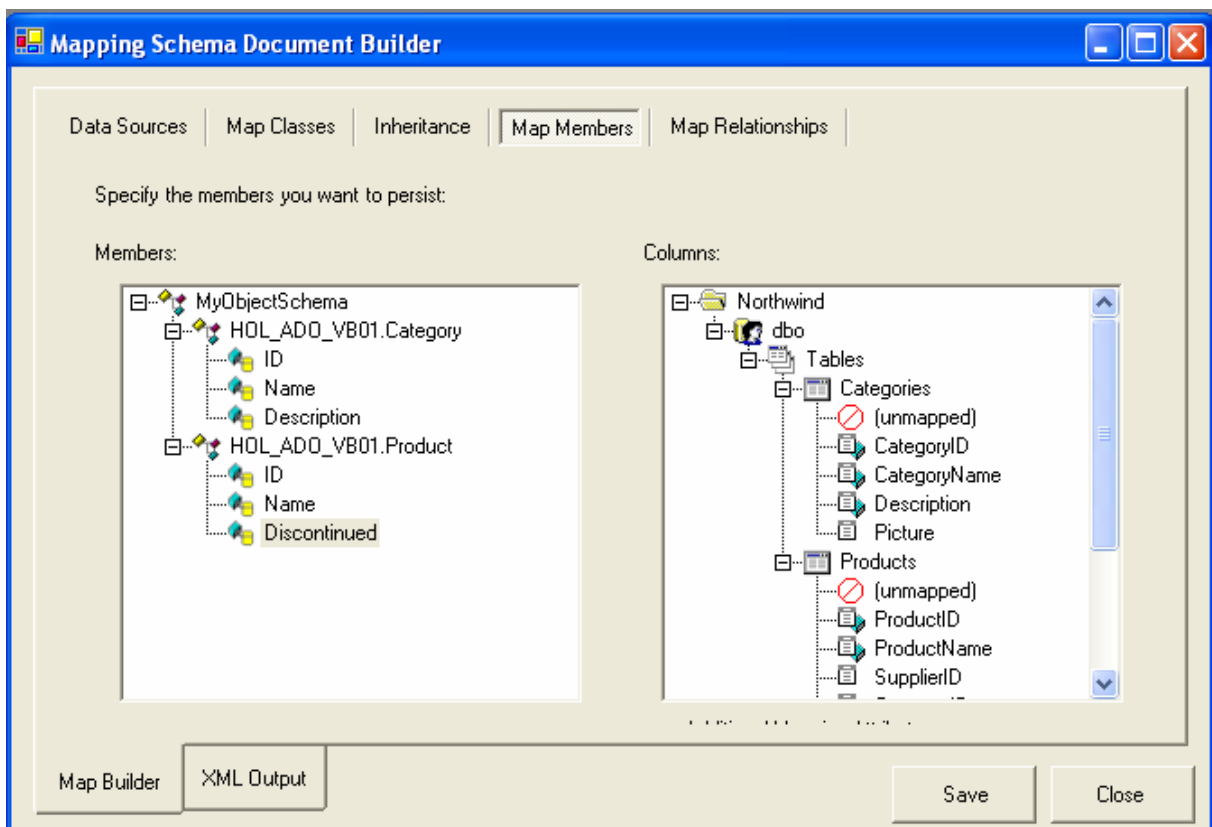


On the Map Classes tab you might now map your class onto a relational table.

- ▶ Select the Categories class and then click on the Categories table.
- ▶ Do the same thing for the Products class and the Products table.
You will see the icons of the nodes changed.

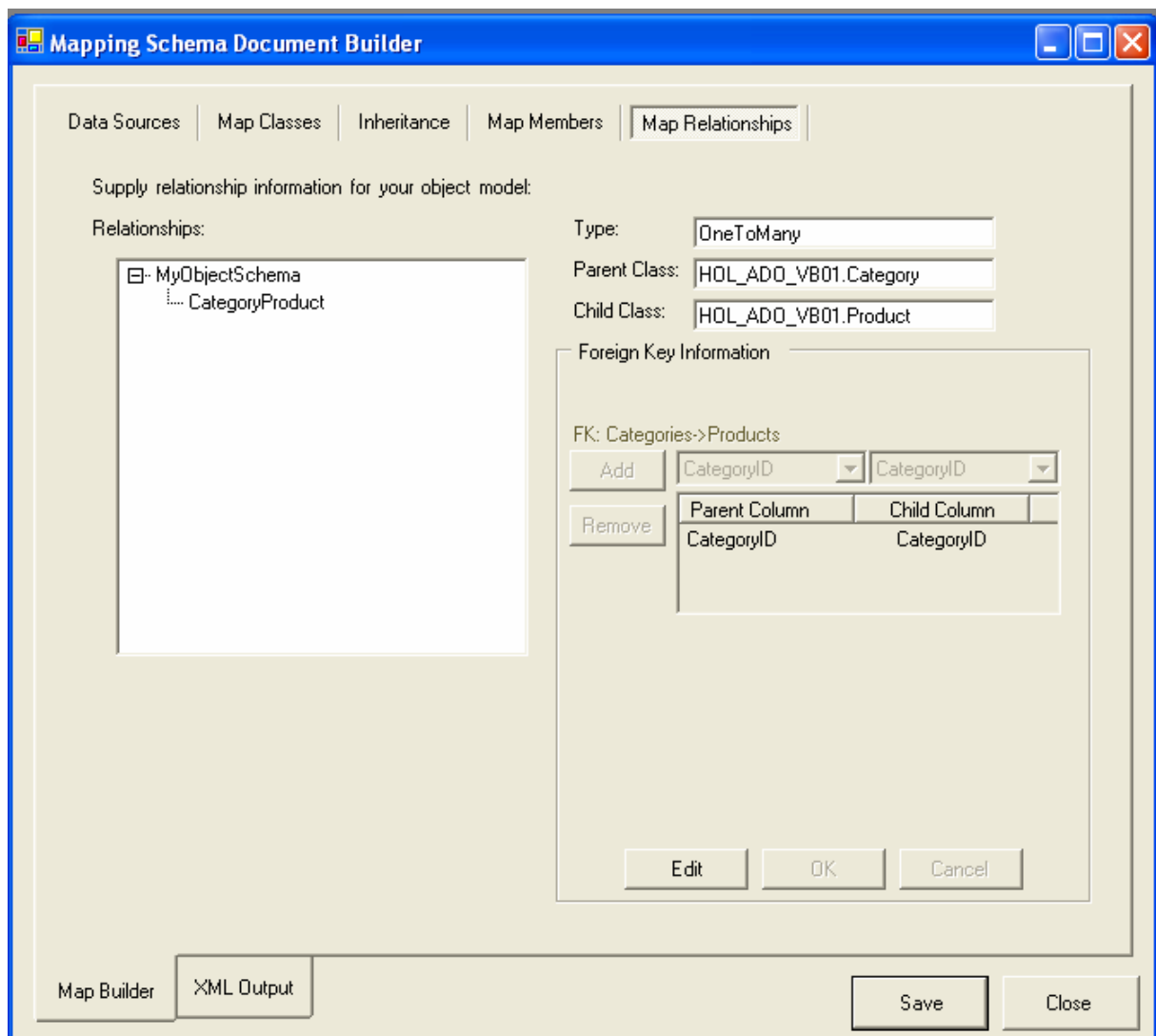


- ▶ Map the members of these classes to the columns of the relational tables. Select the member and then the matching column.

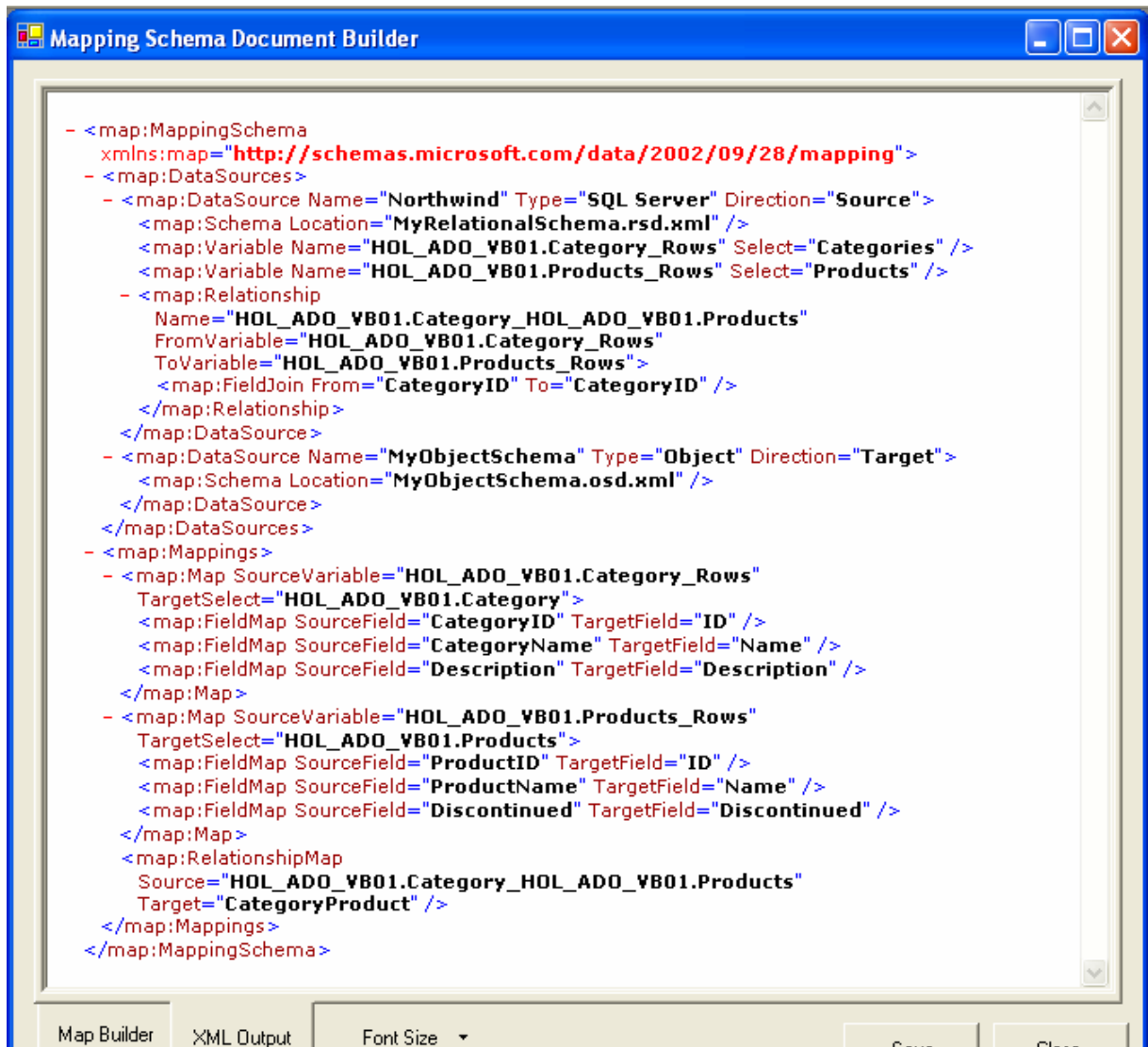


Finally define the relationship.

- ▶ Click on the Map Relationships selecting your CategoryProduct relationship.
- ▶ Push the Edit button at the bottom of the window.
- ▶ Select in the Foreign Key Information panel the primary key and the foreign key.
- ▶ Click the Add button.
- ▶ Additionally click the OK button.

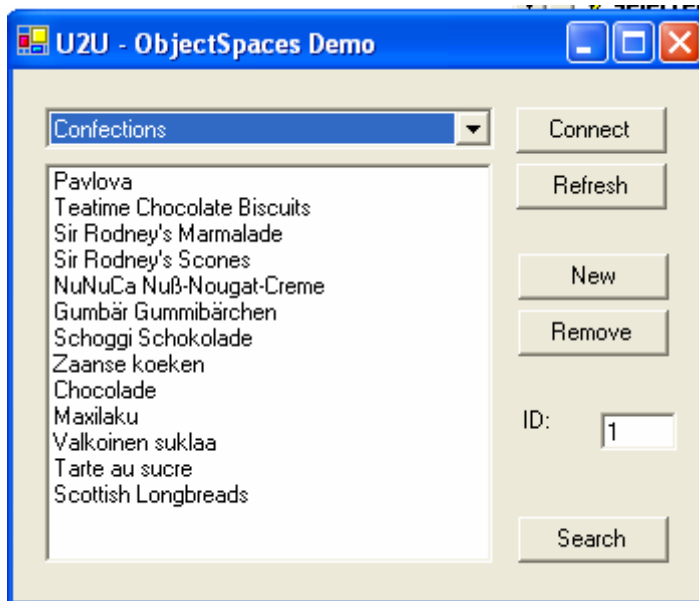


- ▶ Save and verify the XML Output.



You are ready now to work with these files in your Whidbey project.

Exercise 4 – Using ObjectSpaces



Now you can use ObjectSpaces in your Whidbey project. All the code has already been prepared for it. The only thing you have to do is step through the code.

One of the first things you must do is to create an instance of System.Data. ObjectSpaces. ObjectSpace class. This class is available in the System.Data.ObjectSpaces assembly.

This instance needs a connection to the database and the mapping file.

```
Private con As SqlConnection
Private constr As String = _
    "Server=.;Integrated Security=true;Initial Catalog=Northwind"
Private os As ObjectSpace

Dim con As New SqlConnection(constr)
Dim os As New ObjectSpace("C:\XML\MyMappingSchema.msdxml", con)
```

- ▶ Run the application and click the Connect button.
- ▶ Step through the code using the F11 button.

A second button (the Refresh button) demonstrates how to load the list of categories.

```
Dim categories As ObjectSet = _
    os.GetObjectSet(GetType(Category), "", "Products")
listCategories.DisplayMember = "Name"
For Each cat As Category In categories
    listCategories.Items.Add(cat)
Next
```

In the above code, an ObjectSet is filled up by means of the GetObjectSet method. The first argument is the type of object to retrieve from the database. The second argument is an OPath expression. Think of it as the WHERE part of a normal SQL query. The third argument tells ObjectSpaces to retrieve the Product object from the database and also to get the category of the product.

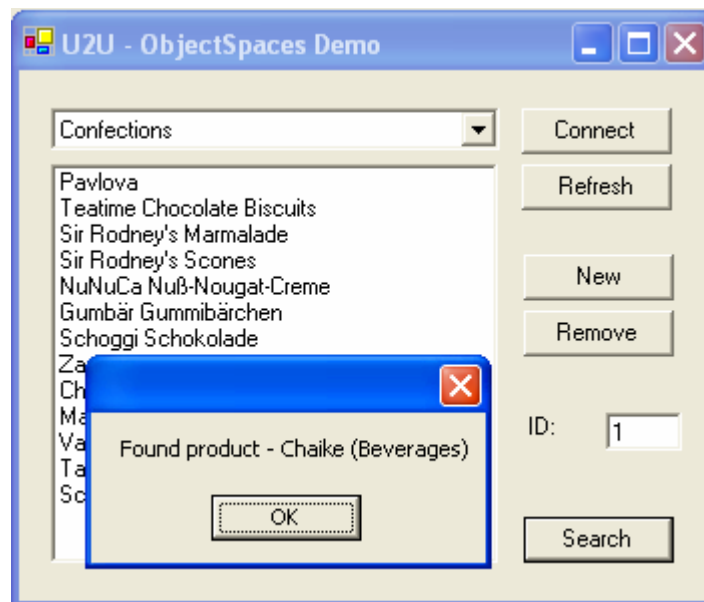
Selecting a category means displaying the products belonging to it.

- ▶ Select a category and step through the code.

```
Dim cat As Category = DirectCast(listCategories.SelectedItem, Category)
listProducts.Items.Clear()
listProducts.DisplayMember = "Name"
For Each pr As Product In cat.Products
    listProducts.Items.Add(pr)
Next
```

ObjectSpaces introduce a new query language called OPath. A simple OPath statement is used to find the product using the ID entered. The GetObject() method of the ObjectSpace is used to retrieve a single object.

```
Dim prod As Product = _
    DirectCast(os.GetObject(GetType(Product), _
        "ID=" & textBoxID.Text, "Category"), Product)
If Not (prod Is Nothing) Then
    MessageBox.Show("Found product - " & prod.Name & " (" & _
        prod.Category.Name & ")")
Else
    MessageBox.Show("Product not found!")
End If
```



Finally, some code can be executed for creating and deleting rows in the database using ObjectSpaces.

A new row is created as a new object, optionally with a child object. All the new objects must be registered with the ObjectSpace object; the PersistChanges method is called in order to update the whole graph of the object.

- ▶ Click on the New button to step through the code.

```
Dim newCat As New Category
newCat.Name = "Belgian Beers"
newCat.Description = "The best beer in the world!"
os.StartTracking(newCat, InitialState.Inserted)
Dim newPr As New Product
newPr.Name = "Jupiler"
newPr.Discontinued = 0
newPr.Category = newCat
os.StartTracking(newPr, InitialState.Inserted)
newCat.Products.Add(newPr)
os.PersistChanges(newCat, New PersistenceOptions(Depth.ObjectGraph))
```

