

EntitySpaces 2010 WCF Demo

Contents

About the WCF Demo.....	1
How We Generated the Lightweight Client Side Proxies	1
Generating the Client Side Proxies	3
Generating the Business Objects.....	6
The Generated Master Template	7
The WCF Service	11
How We Setup the Added the WCFConsumer_Thin Executable	13
How We Setup the Added the WCFConsumer_Thick Executable	15
Running the Demos	19

About the WCF Demo

The WCF Demo contains a single WCF Service that is accessed by what we call a thick client (that uses our full business objects on both sides of the conversation) and a thin client that uses our lightweight proxies on the client side and our full business objects on the server side. In both of these examples (the thick and thin client) we avoid using Visual Studio generated proxy classes when we add our Service Reference. Of course, you can use the Visual Studio generated service reference classes but our proxies are much more powerful to use.

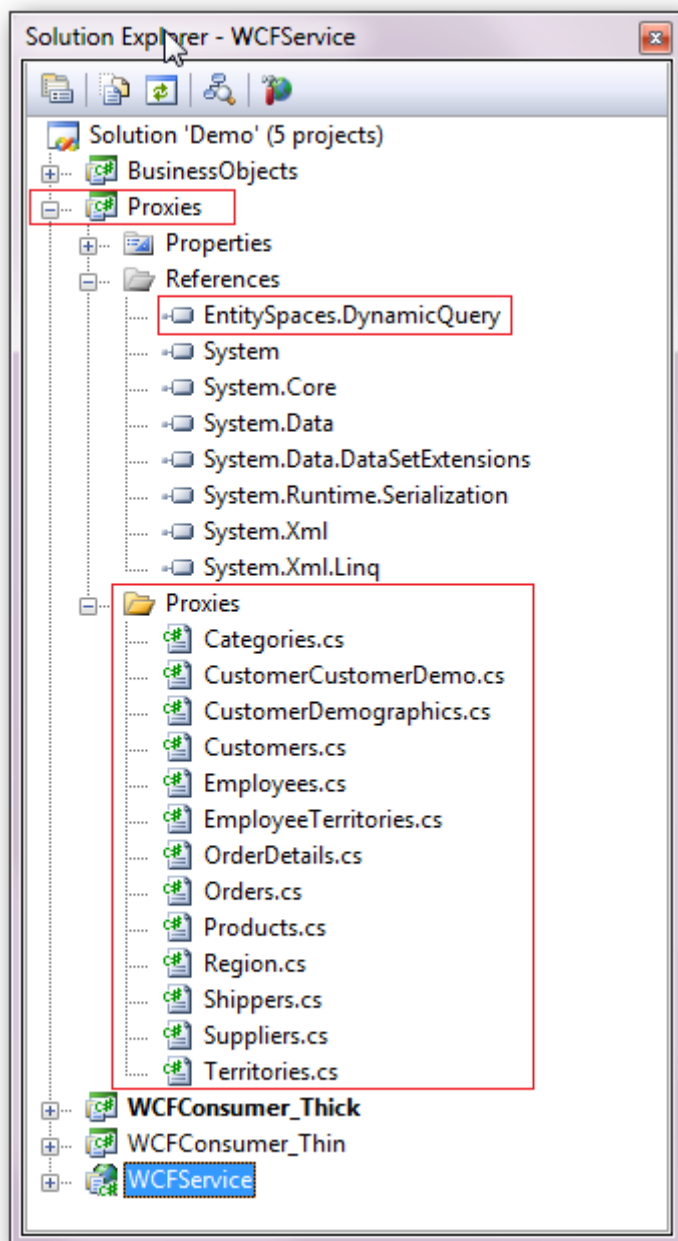
You could for instance, provide your clients with something similar to our thin client example and then consuming your WCF Reference would be very easy for them. IF you have any questions just register on our forums and ask.

- The EntitySpaces Team

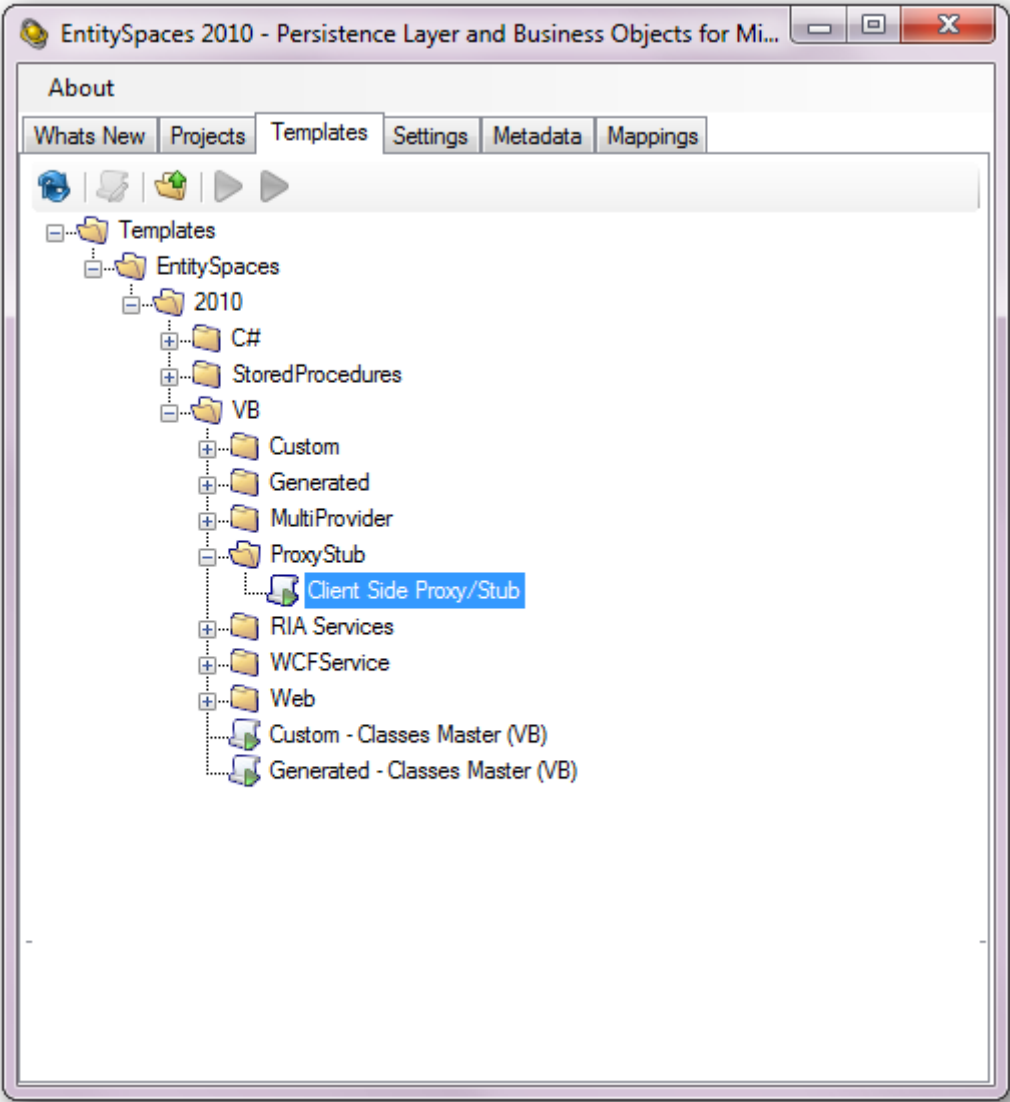
How We Generated the Lightweight Client Side Proxies

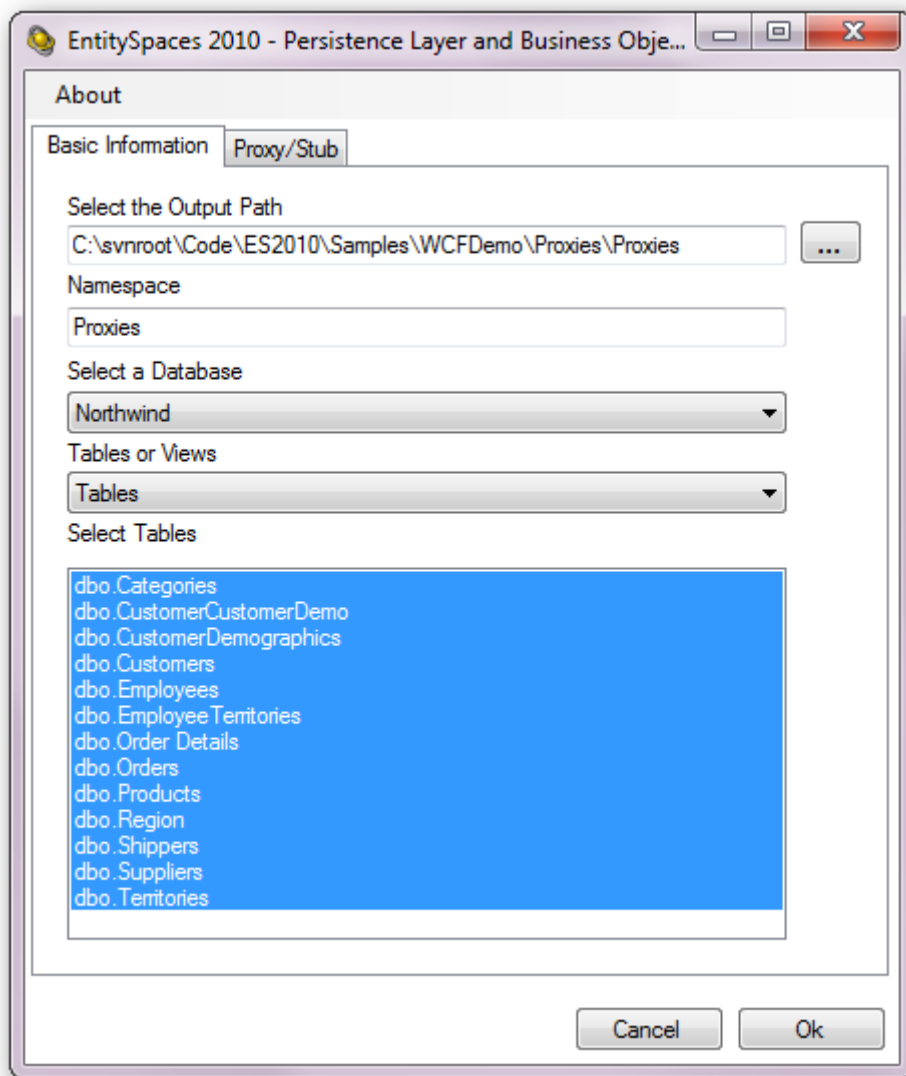
This is how we generated the “*Proxies*” Assembly. This must be created as a class library when you choose “New Project”. The reason we put the lightweight proxies in their own class library is so that we can choose the “Proxies” class library when we add the service reference, this will be shown later in this document.

Notice that we include the EntitySpaces.DynamicQuery assembly in our references. This assembly allows you to do full EntitySpaces DynamicQueries in your client side application. You actually serialize the query and send it to the server to execute. Obviously you do not access your database directly from your client side application. This is the only EntitySpaces assembly necessary for the thin client version and it’s only around 60k in size. You can even choose not to use the DynamicQuery feature and avoid needed to use any EntitySpaces class libraries. However, the DynamicQuery feature is very nice.



Generating the Client Side Proxies

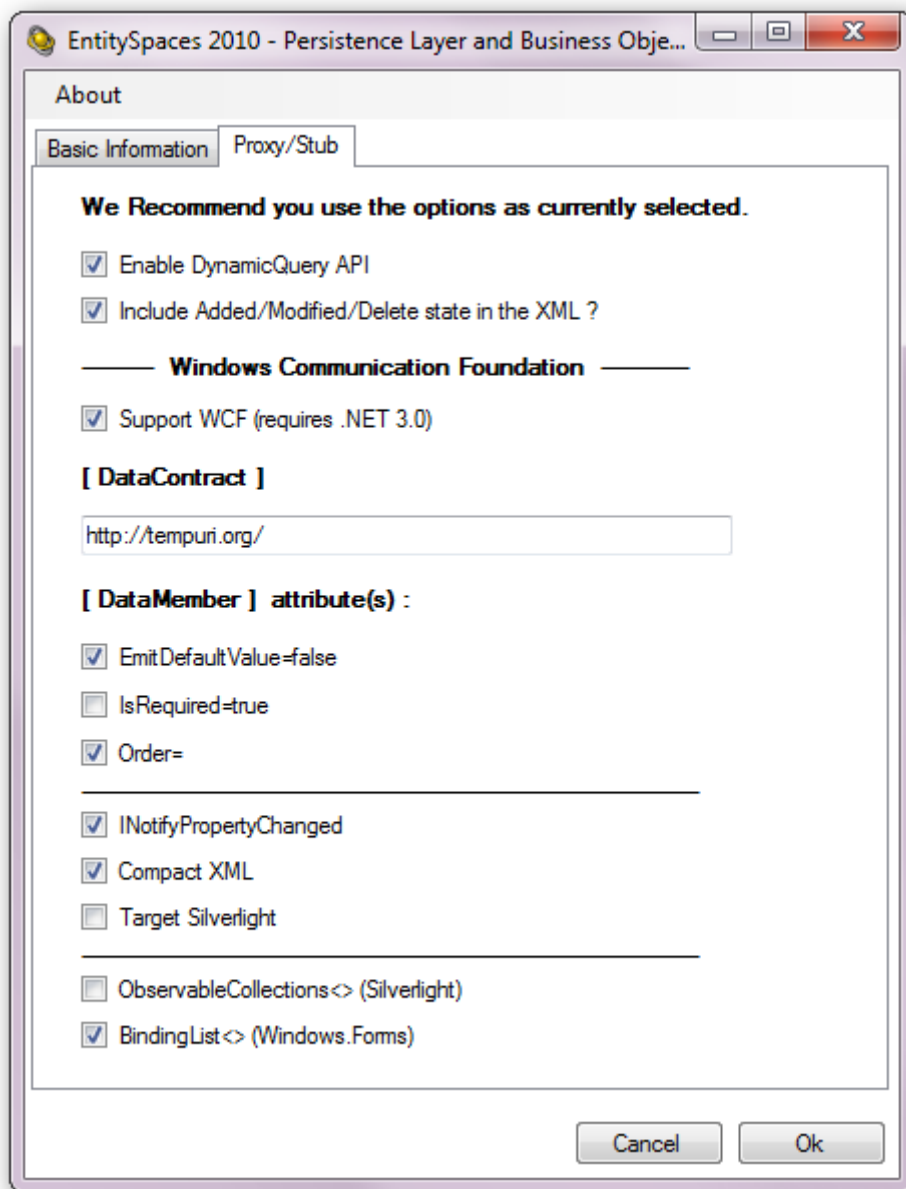




There wasn't much to do here on the "Basic Information" Tab.

We merely selected all of the tables in the Northwind database and we used "Proxies" as our namespace.

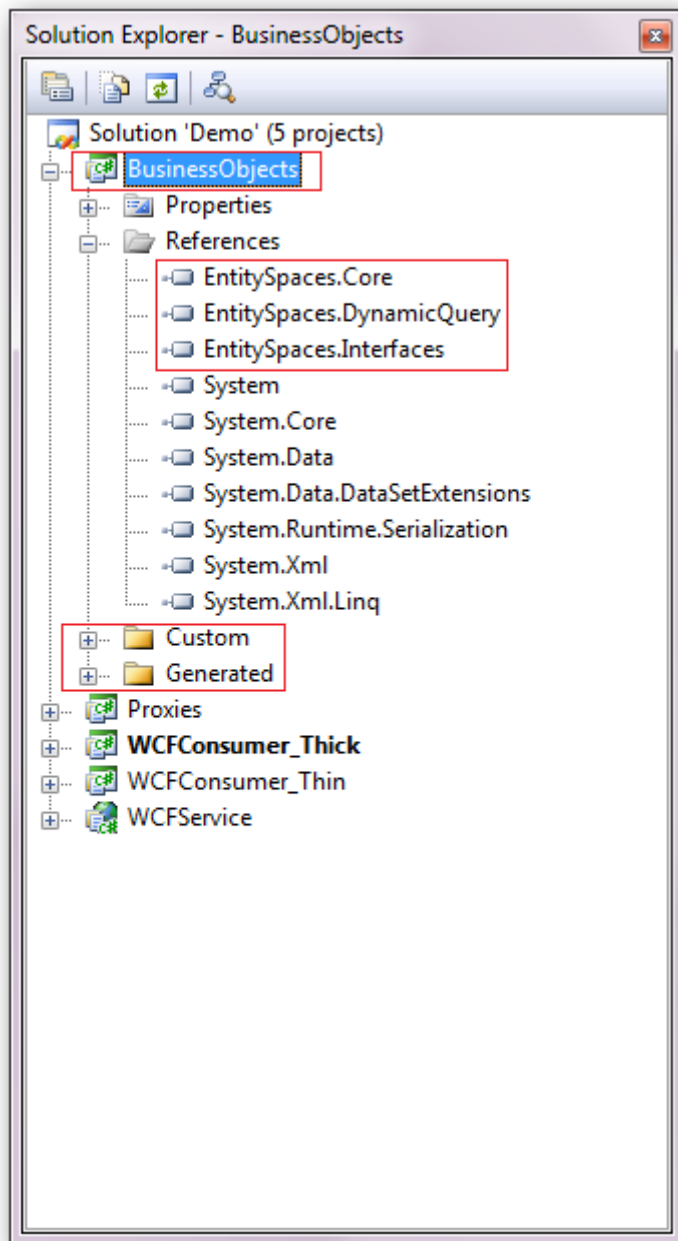
It is important that the proxies be in their own class library. See the next page for the interesting UI choices ...



These are the choices we recommend for WCF development on the lightweight Proxy/Stub tab. That's it, merely generate the classes, include them in your "Proxies" class library, add a reference to the EntitySpaces.DynamicQuery assembly

Generating the Business Objects

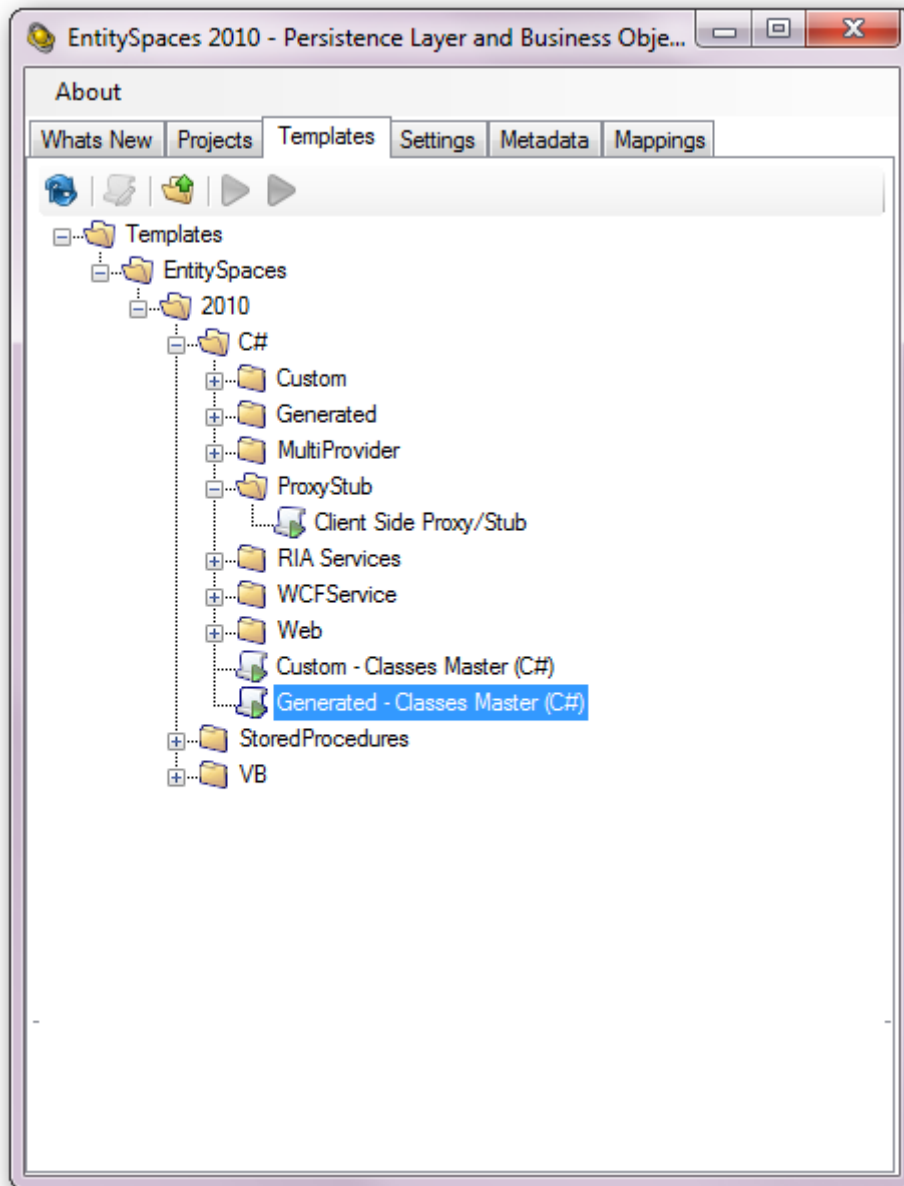
These objects are housed in our “BusinessObjects” class library. These are your normal full business objects that you would use in any normal EntitySpaces Application.



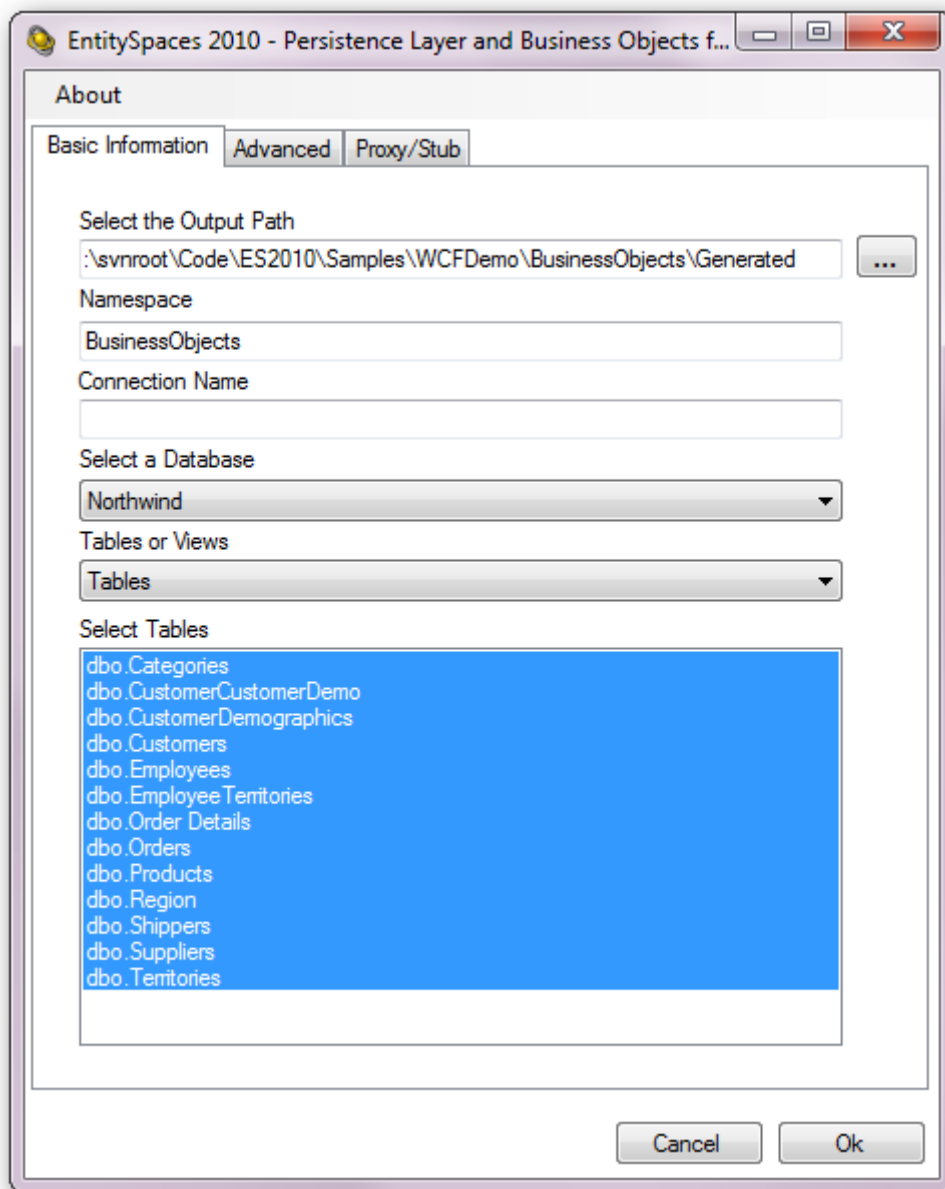
Since this class library will house our full business objects we add the three main references shown on the left. Also, we create our “Custom” and “Generated” classes and include them in this class library.

The creation of the generated classes is shown on the following pages.

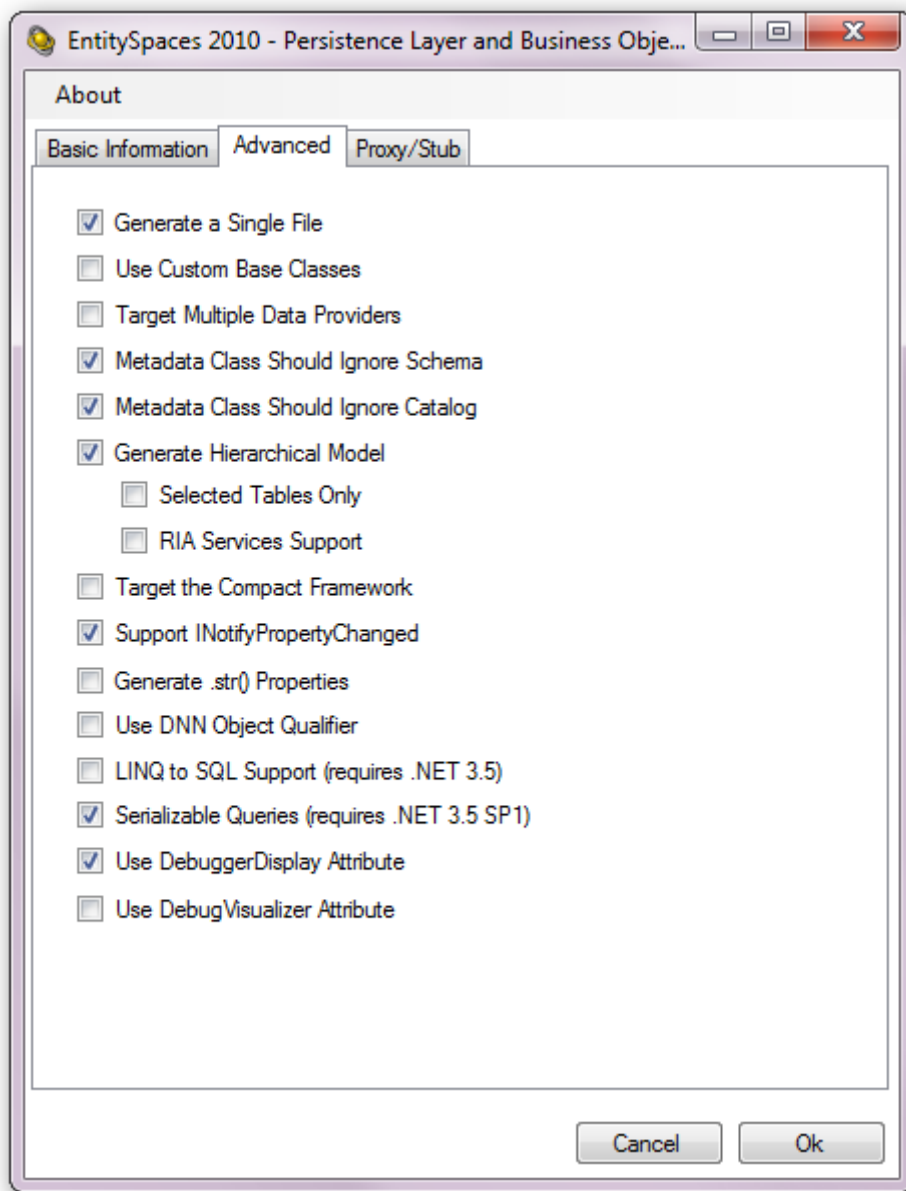
The Generated Master Template



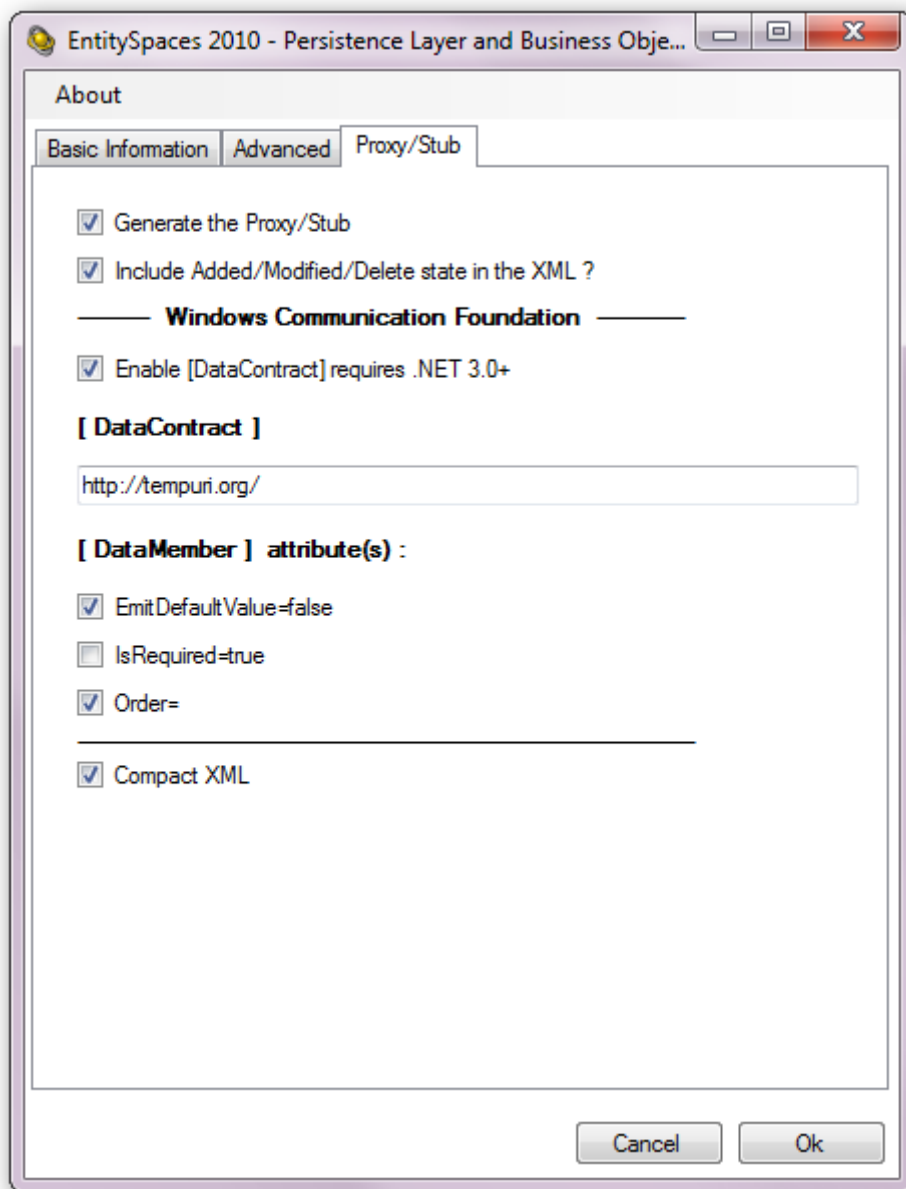
First we merely select all of the tables in our Northwind Database.



First we merely select all of the tables in our Northwind Database.

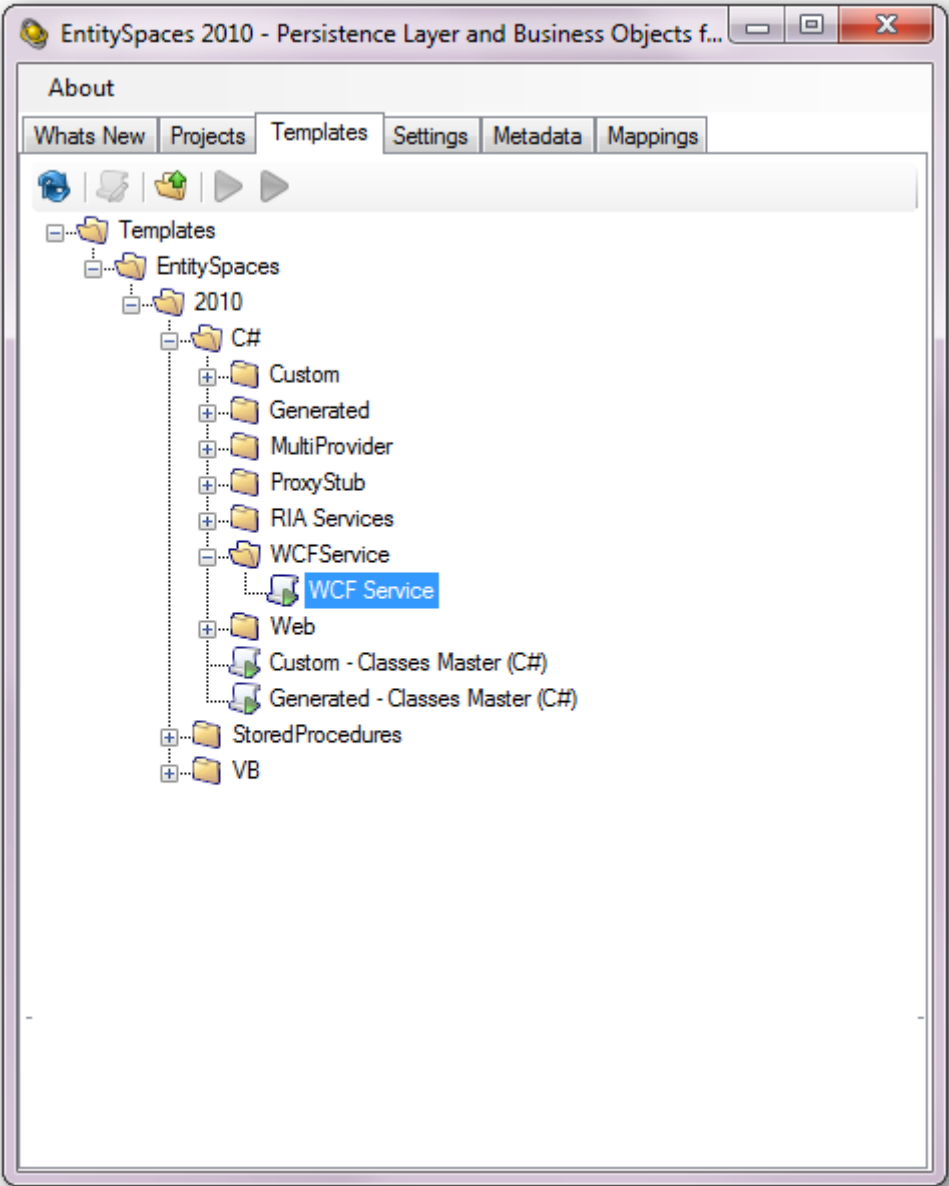


The key things to notice here is that we checked **Serializable Queries** which is necessary if we want to be able to deserialize the queries sent to us from our client Application. This is not required, you don't have to allow queries from the client side, but for this sample we wanted to show how this feature works.



Notice here we make the same choices as we did when created the lightweight proxies. The important thing here is that we need to be sure to check the “Compact XML” checkbox too since we also made this choice for our lightweight proxies.

The WCF Service



EntitySpaces 2010 - Persistence Layer and Business Objects f...

About

Basic Information

Select the Output Path
C:\svnroot\Code\ES2010\Samples\WCFDemo\WCFService ...

EntitySpaces Namespace
BusinessObjects

WCF Service Class Namespace
WCFService

WCF Service Class Name
Service1

Connection Name

Select a Database
Northwind

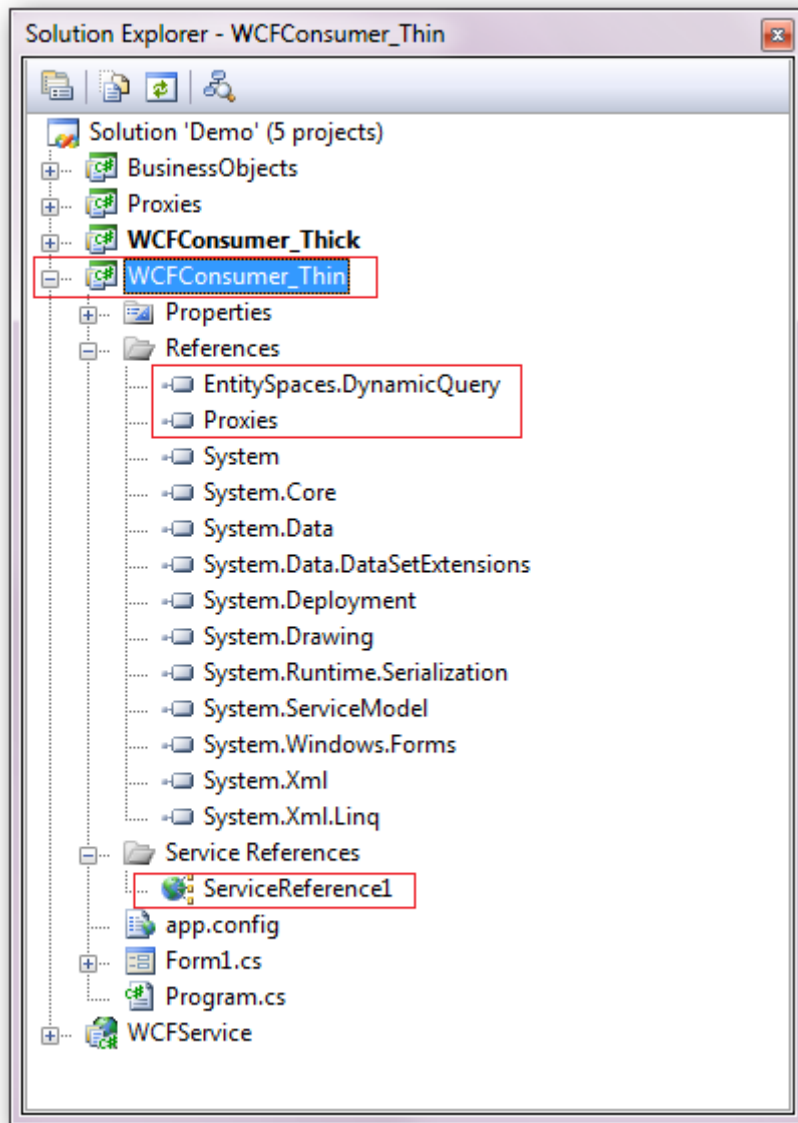
Tables or Views
Tables

Select Tables

- dbo.Categories
- dbo.CustomerCustomerDemo
- dbo.CustomerDemographics
- dbo.Customers
- dbo.Employees
- dbo.EmployeeTerritories
- dbo.Order Details
- dbo.Orders
- dbo.Products
- dbo.Region
- dbo.Shippers
- dbo.Suppliers
- dbo.Territories

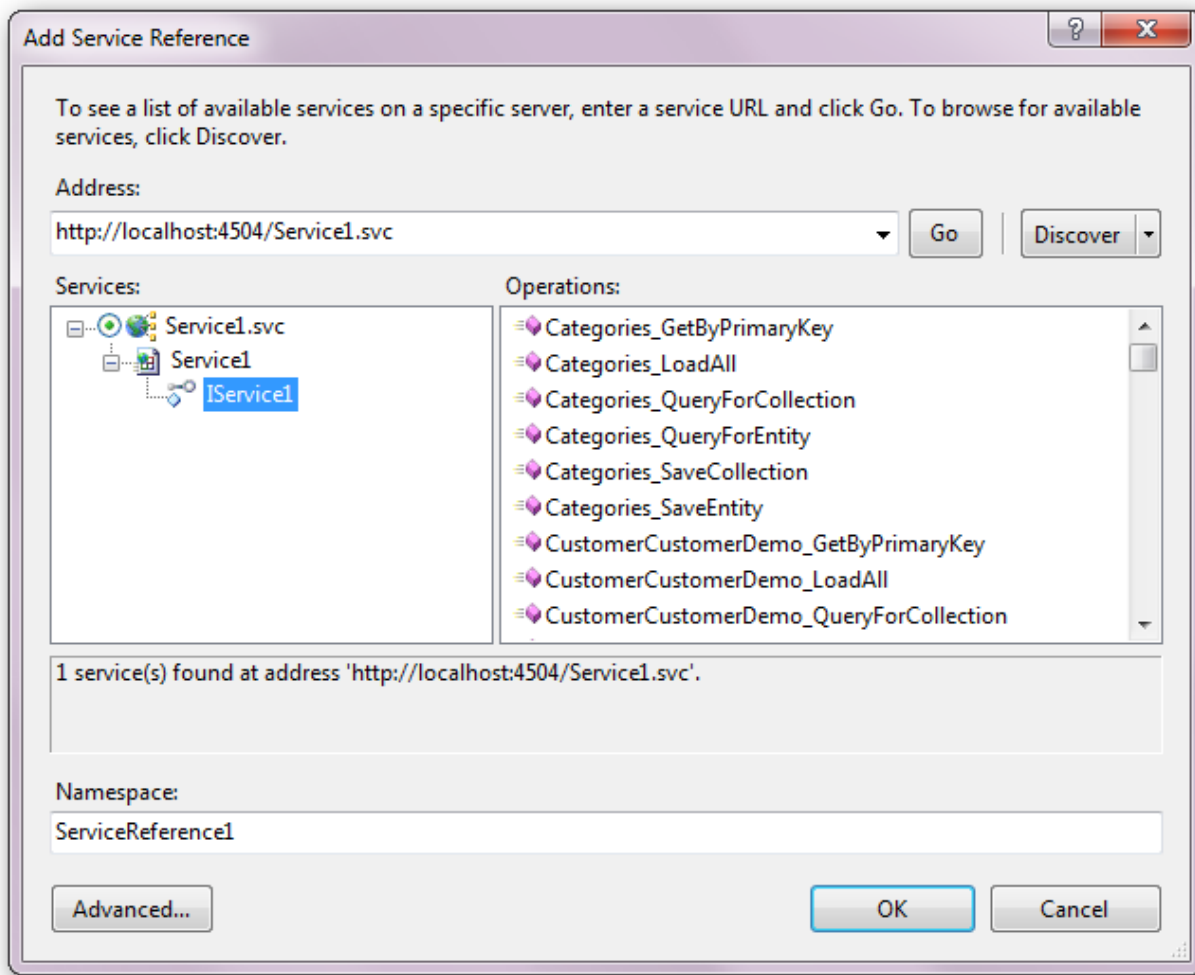
Cancel Ok

How We Setup the Added the WCFConsumer_Thin Executable

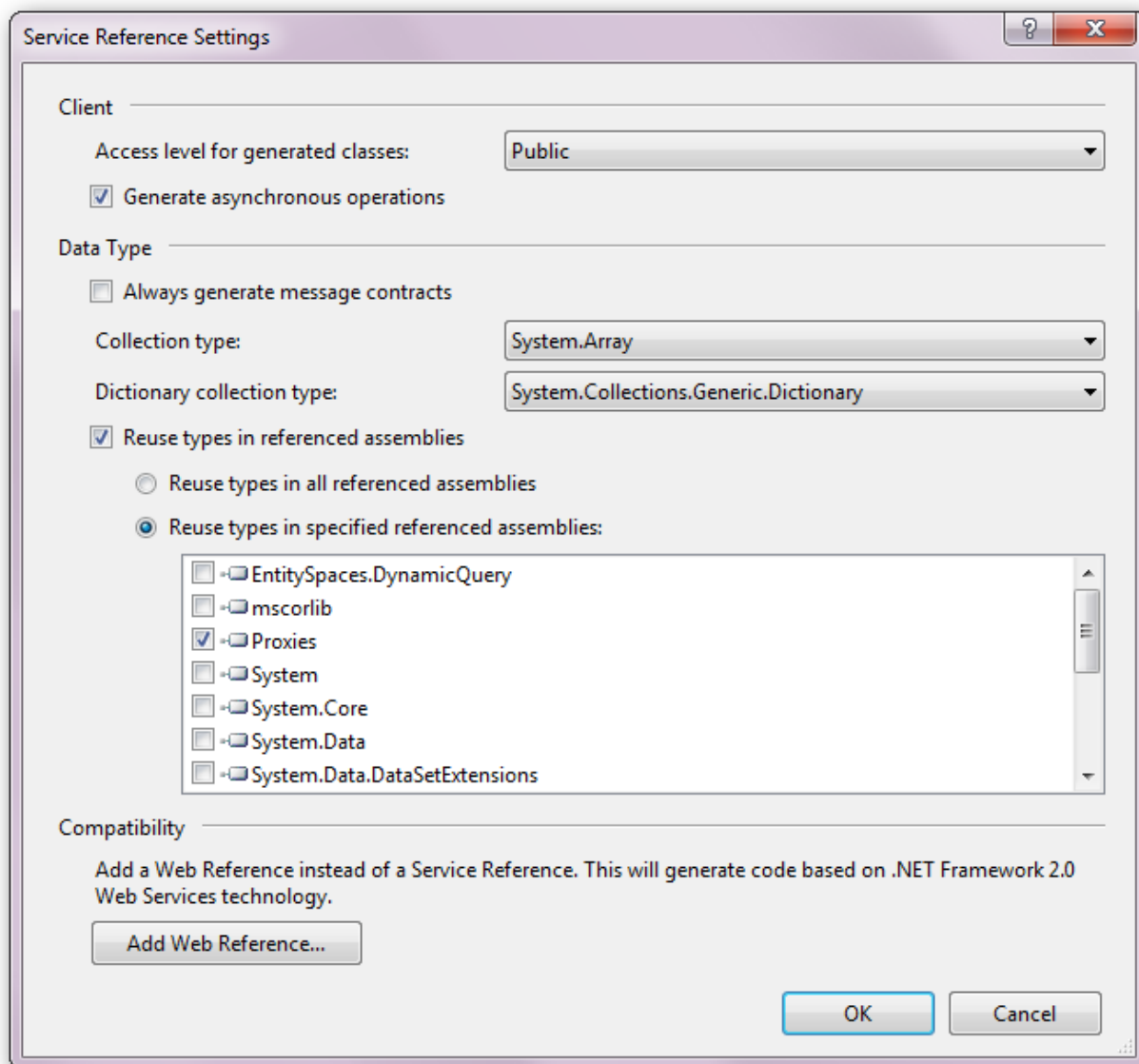


This is a very simple executable that uses the lightweight client side proxies on the client side. We include two references here, our “Proxies” class library and the EntitySpaces.DynamicQuery class library since we plan to use the serializable query feature.

We also needed to add a reference to our WCF Service. Let’s take a look on the following two pages and see how we added our Service Reference.



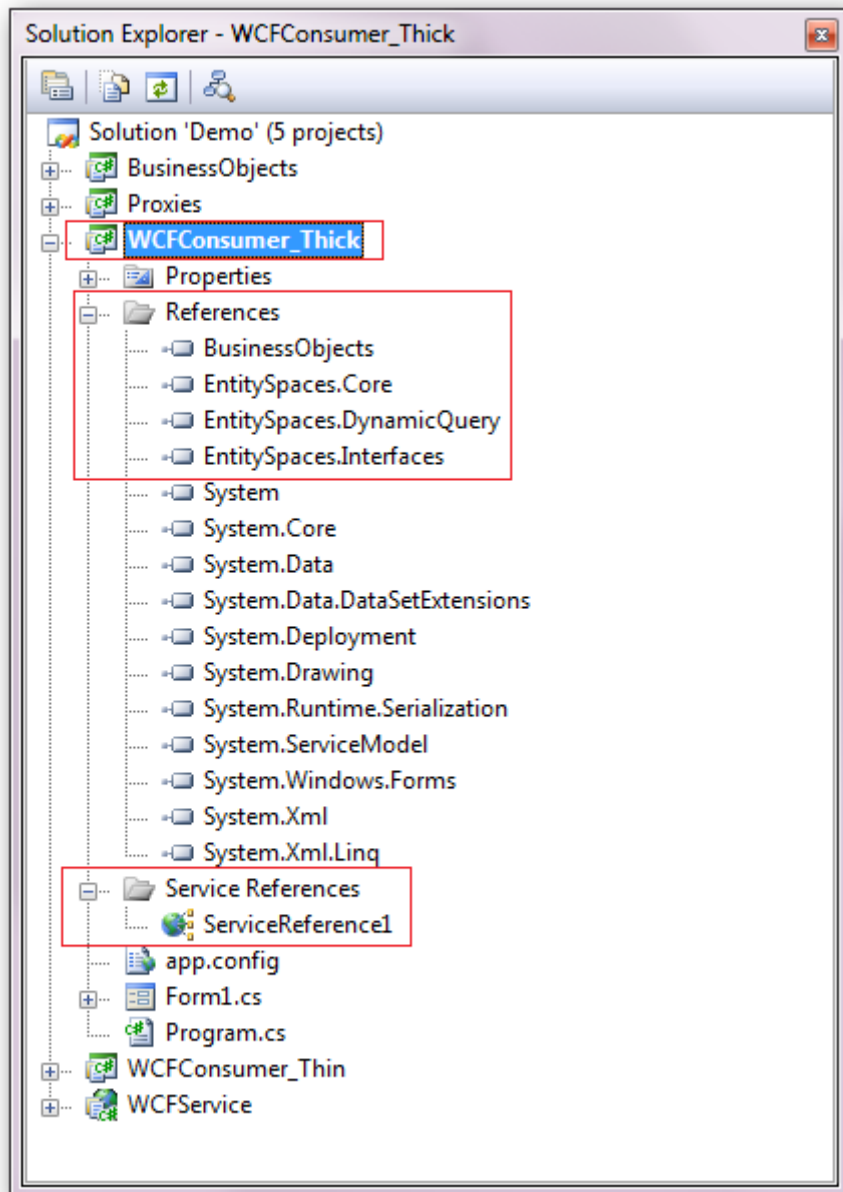
The above is pretty standard, since our Service is in the same solution we can merely use the “Discover” button and it will automatically find our Northwind service. The next thing we need to do is let Visual Studio know we want it to use our Proxies and not the Visual studio generated proxies. To do this we must press the “Advanced” Button.



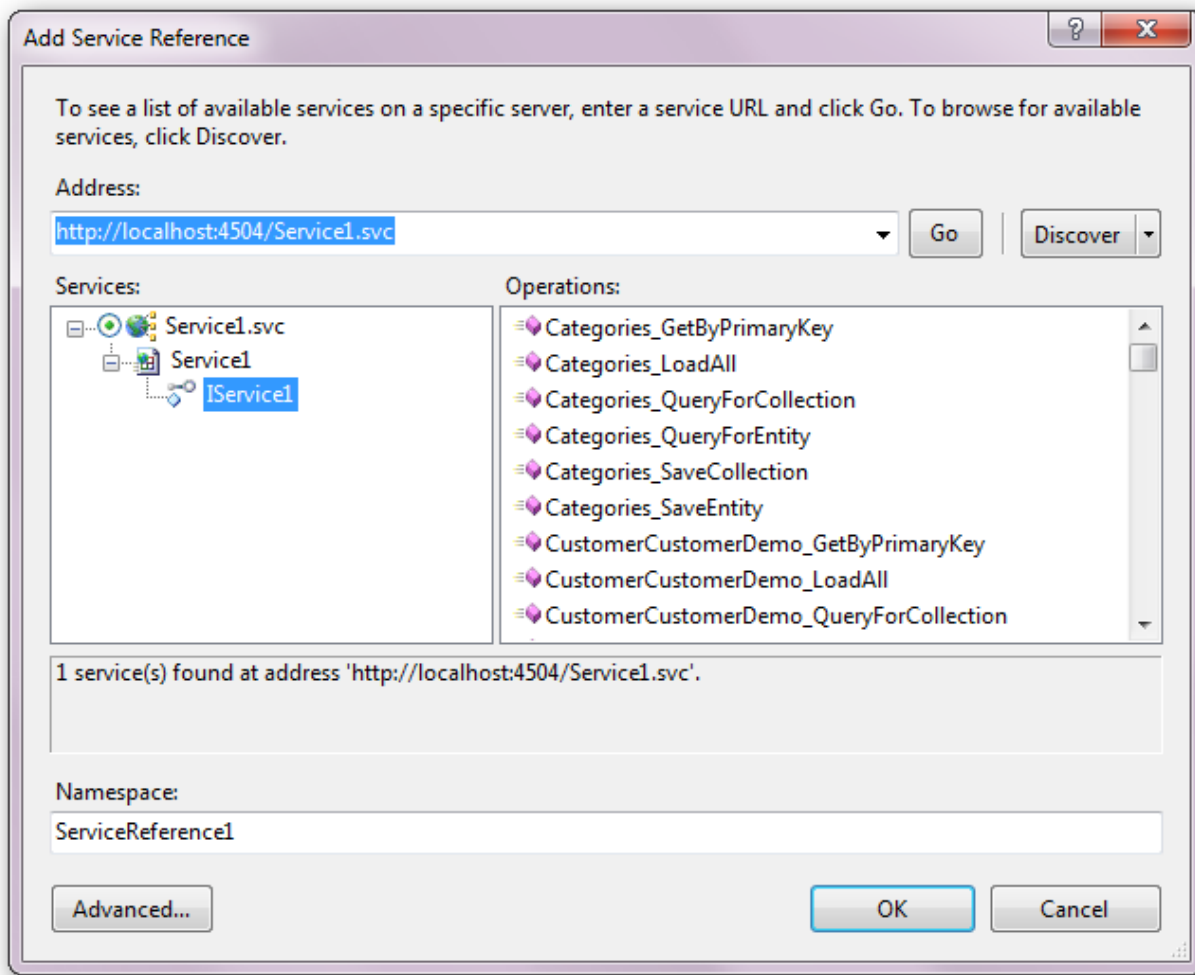
The importing thing to note here are the choices we made to indicate that we want to use the “Proxies” class library. That’s it, click “OK”

How We Setup the Added the WCFConsumer_Thick Executable

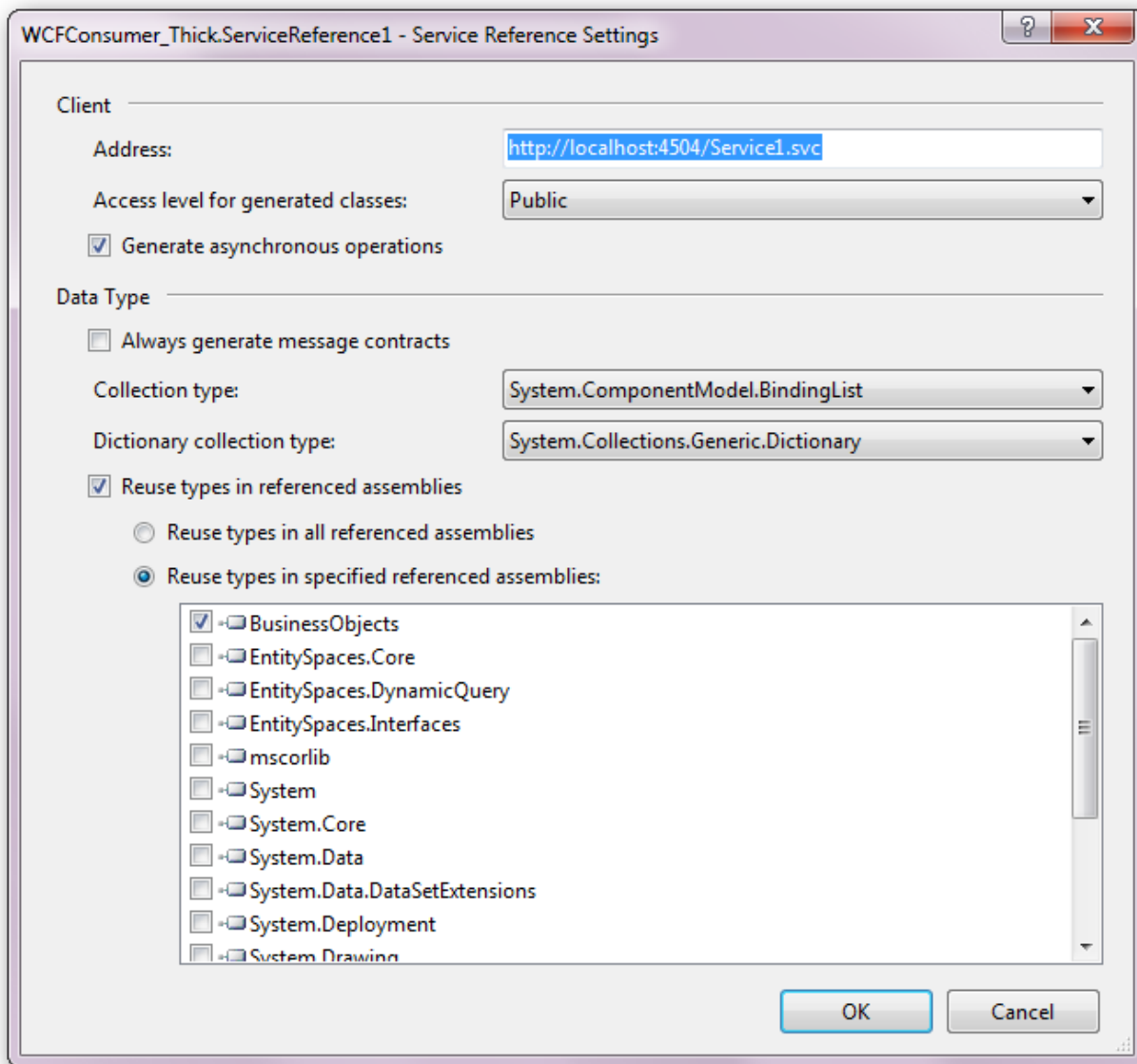
This is a very simple executable that uses the lightweight client side proxies on the client side. We need to include the BusinessObjects class library since this is our thick client example. We are going to use our full business entities on both sides of the conversation. We also include the main three EntitySpaces class libraries as shown on the left.



We also needed to add a reference to our WCF Service. Let's take a look on the following two pages and see how we added our Service Reference.



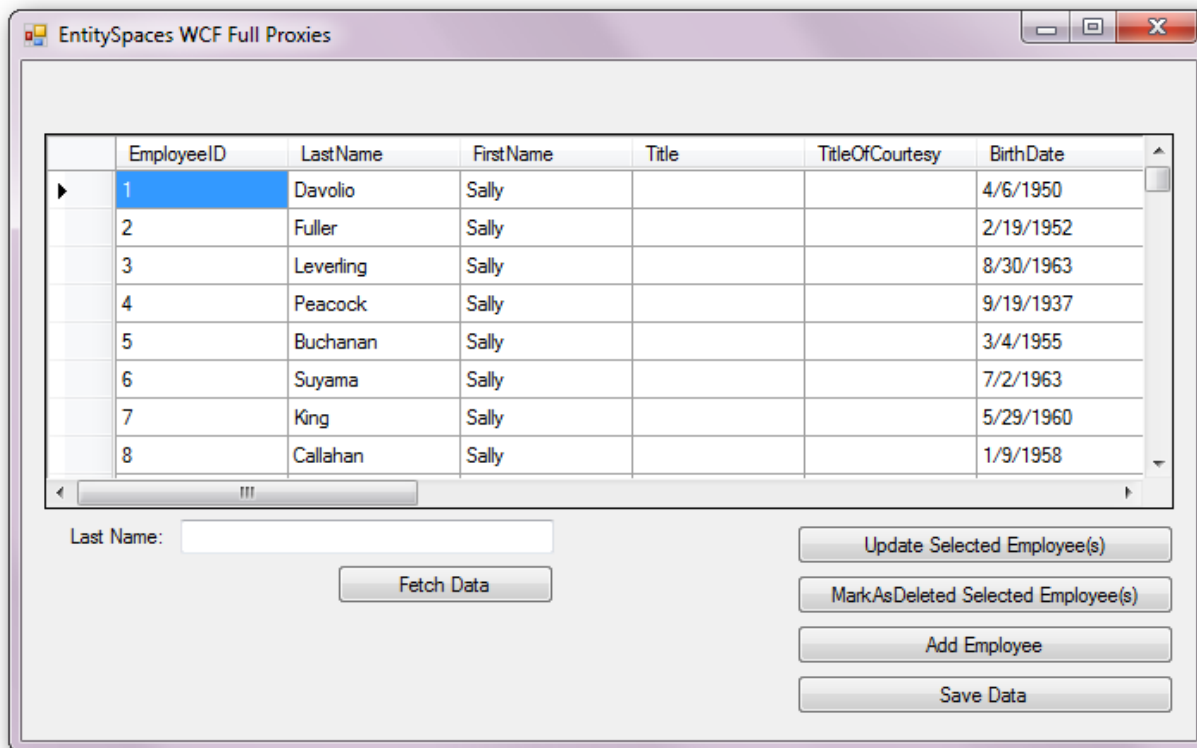
The above is pretty standard, since our Service is in the same solution we can merely use the “Discover” button and it will automatically find our Northwind service. Then we changed the Namespace to “NorthwindClient”. The next thing we need to do is let Visual Studio know we want it to use our Proxies and not the Visual studio generated proxies. To do this we must press the “Advanced” Button.



Notice that we set the collection type to "ObservableCollection" the same type that we chose when generating our lightweight proxy stubs. We also indicate that we want it to use the proxies in our EntitySpaces.Proxies.Silverlight assembly (that's just what we happened to name our assembly that contains the lightweight proxies).

That's it, click "OK"

Running the Demos



The screenshot shows a Windows application window titled "EntitySpaces WCF Full Proxies". Inside the window is a data grid with the following columns: EmployeeID, LastName, FirstName, Title, TitleOfCourtesy, and BirthDate. The grid contains 8 rows of employee data. The first row is selected. Below the grid is a search field labeled "Last Name:" and a "Fetch Data" button. To the right of the search field are four buttons: "Update Selected Employee(s)", "MarkAsDeleted Selected Employee(s)", "Add Employee", and "Save Data".

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	Davolio	Sally			4/6/1950
2	Fuller	Sally			2/19/1952
3	Leverling	Sally			8/30/1963
4	Peacock	Sally			9/19/1937
5	Buchanan	Sally			3/4/1955
6	Suyama	Sally			7/2/1963
7	King	Sally			5/29/1960
8	Callahan	Sally			1/9/1958

Last Name:

Fetch Data

Update Selected Employee(s)

MarkAsDeleted Selected Employee(s)

Add Employee

Save Data

The first thing you need to do is make sure the connection string in the web.config file is correct for your Northwind database. The web..config file in in the WCFService project. Find the **connectionStrings** section in your web.config file and set it up appropriately.

You can run either the Thick or Thin clients and add data, delete, or update data in the grid and then press the Update button. You can test the search feature and test the serializable DynamicQuery feature. The search feature searches on both FirstName and LastName fields.