Contents

This application demonstrates basic DevForce operations such as logging in, querying, and saving.

- **Platform:** Silverlight
- **Language:** C#, VB
- **Download:** Simple Silverlight application

Problem

You want to see basic DevForce operations running in a application that you can examine and play with.

Solution

SimpleSilverlightApp is an example which shows how to get a simple DevForce Silverlight application quickly up and running.

Steps used to build this application:

1. Open Visual Studio and choose "File - New - Project" to start a new project. Select either Visual C# or Visual Basic => DevForce => DevForce Silverlight Application template. The template will create two projects - one for a Silverlight application project, and one for the web application to host it. The projects differ from those created by the "Silverlight Application" template from Microsoft in that they contain references to the DevForce assemblies you'll need, and the web project also hosts the DevForce BOS.

The web project is also set to use the Visual Studio Development Server. This makes development easier, since Visual Studio will stop and start the web server as needed, but of course cannot be used outside of development and unit testing. You'll need to deploy to IIS when ready to move your application beyond your desktop.

One other thing to notice is the port number of 9009. We use that in all samples, and this is the default set by the DevForce templates, but you can use any open port you'd like. Remember when you move to IIS the port number will be 80.

2. To keep things simple in this application we're going to use only 2 projects, so we won't create additional class libraries to hold our Entity Model or Domain Model. This isn't a robust architecture for your real development efforts, but helps keep the moving parts to a minimum here.

We first create the Domain Model in the web project, by adding a new item for the "ADO.NET Entity Data Model". This data model is using the NorthwindIB database used in other tutorials. Why did we add the item to the web project? The Entity Model is used only on the "server side" of a DevForce application. In this case the web project, hosting the BOS, is the "server side". A Silverlight application, running as a browser plug in, cannot access databases.

When working in the EDM designer, note the Properties Window contents. If you sort the properties by category, you'll see "DevForce Code Generation" attributes for the model, for entities, and for entity properties. We've let everything default here.

When you save the EDMX, DevForce will generate several additional files: a *.edmx.tt file containing the T4 template used in code generation, and a *.IBDesigner.xx file containing the code for generated model. We also see that DevForce created a link in the Silverlight application project to the generated model file. DevForce will do this automatically for any generated code: using assembly name matching between Silverlight and standard .NET assemblies to perform the linkage.

- Now that we have a domain model we can create a page to retrieve some entities. Page.xaml contains a simple DataGrid, with some buttons highlighting some of the steps needed to use the EntityManager in a DevForce Silverlight application. You'll also notice a ComboBox loaded with several queries.
- Build and run the application (be sure that the web project is the startup project). First notice that a browser window opens to something like http://localhost:9009/Default.aspx. If you look in the Default.aspx file in your web project you'll see that it contains a Silverlight control with height and width set to fill the page, and whose source is a "XAP" file loaded from the server. This XAP file is a compressed archive containing your Silverlight application and all is dependencies and resources.

You'll first notice that you need to press the "Connect" button in this application - this creates a connection to the BOS and performs some other housekeeping to initialize communications but does this asynchronously, as required in a Silverlight application when communicating with a service. Remember that the BOS here is also hosted by the web application, at an address similar to http://localhost:9009/EntityService.svc.

Next we Login. This must also be done asynchronously, since the BOS is responsible for authenticating users. In the sample, the user will not be authenticated and is logged in as a guest user.

We can now run the several queries provided in the ComboBox to fetch data to the Silverlight application. Fetches must also be performed asynchronously in Silverlight if they will go to the BOS to be fulfilled. If a fetch can be completed from cache, you can issue either a synchronous or asynchronous query.

If you make any changes to grid data you can save those modifications by pressing the "Save" button. Here an asynchronous save operation is performed to send the changes to the BOS for saving to the database.

You can also use the "Logout" button to perform an asynchronous logout from the BOS, and the "Reset" button to both logout and disconnect.