Contents

The Tour of DevForce Silverlight walks you through the basics of using DevForce to build a simple Silverlight application. In this sample we'll take the application built in the tour and swap the existing *Database First* model for a Code First one.

- **Platform:** Silverlight
- **Language:** C#
- **Download:** Code First Silverlight tour

Problem

If you're interested in the Code First style of development, but worry that you'll sacrifice DevForce features, worry no more. In this sample we'll take the completed application from the Tour of DevForce Silverlight and show how easy it is to use *Code First* instead.

Solution

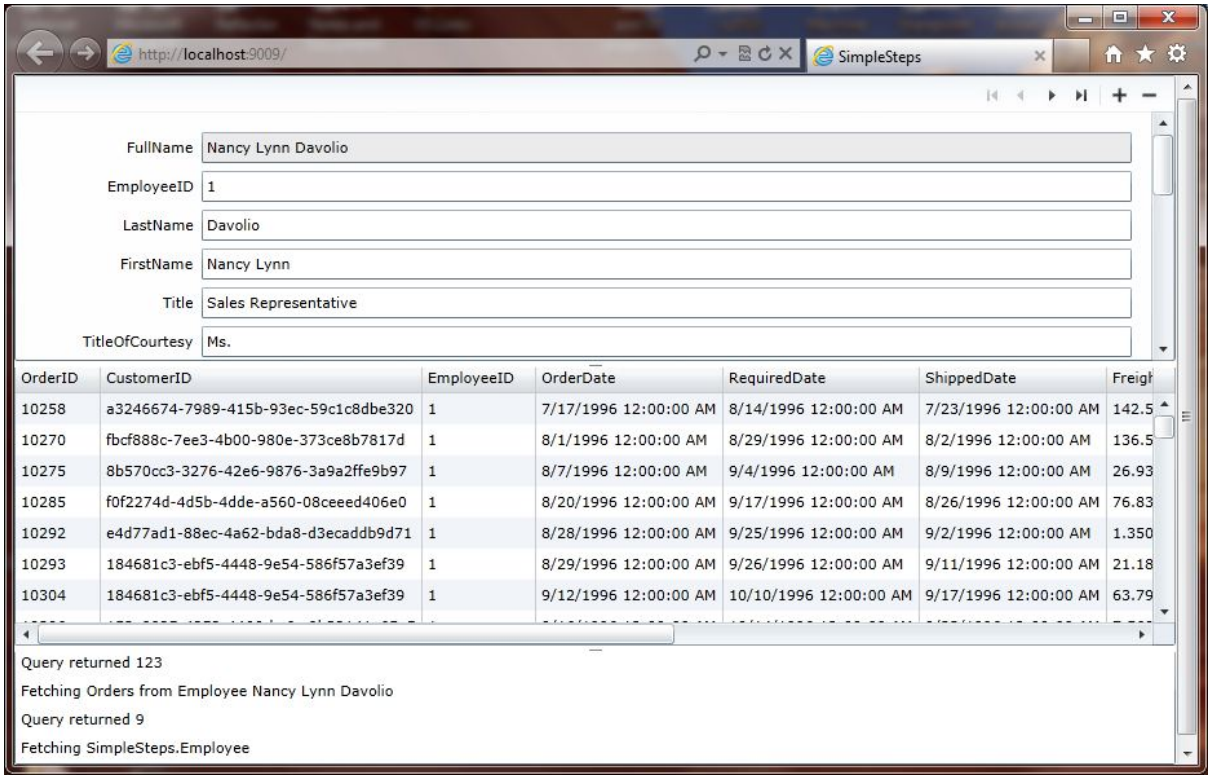
Our starting point is the completed solution from part 5 of the Silverlight tour. We made only a few changes.

1. Let's first install the DevForce Code First NuGet package to both projects. This will add all necessary assembly references and files, and inject both PostSharp and DevForce into the MSBuild pipeline.
2. Since we already have a database, we took the Code Second approach and generated our *Code First* model from the existing EDMX file.
3. We then copied the generated code to a file named *NorthwindEntities.cs* and removed the EDMX and .TT files.
4. We then customized the generated code - we can do this because it won't be automatically re-generated, and we'd like to make a few changes.
    1. We added a base class named *BaseEntity*. Using a base class is an elegant way of implementing *Code First* entities, since you can add the *ProvideEntityAspect* attribute on the base class to have entity infrastructive injected into all sub-classes.
    2. All entity classes were changed to inherit from the *BaseEntity*.
    3. We didn't generate with the various binding attributes enabled in order to simplify the generated code, but we found that disabling auto-generated field display of the navigation properties is useful, especially in a demo application using a *DataForm* and *DataGrid* which are set to auto-generate fields. So we added [Display(AutoGenerateField = false)] to all navigation properties.
    4. We kept the validation attributes, since they are part of the business logic you would add.
    5. Simplified the *NorthwindIBEntities* constructors.
    6. Removed the *Employee.cs* file and placed its custom property, *FullName*, directly in the *Employee* class. We also found we needed to set the display order for auto-generation of fields since we defined this property after all other *Employee* properties, yet wanted to display it first.
    7. We left the various data annotations intact.
5. When we installed the *CodeFirst* package, it also added a marker file, "DevForce1.cf", which will trigger (re)generation of the ".ibmmx" metadata file when the model changes. Additionally, it added references for the *EntityFramework.dll*, *IdeaBlade.Aop.dll* and *PostSharp.dll* to the web project, and *IdeaBlade.Aop.SL.dll* and *PostSharp.SL.dll* to the Silverlight application project.
6. Modified the connection string in the web.config from an Entity Framework connection string to a SQL connection string.

A few other things to notice -

- "DevForce1.cf" is the *Code First* marker file that triggers (re)generation of the ".ibmmx" DevForce metadata file.
- "NorthwindIBEntities.ibmmx" is the generated DevForce metadata file. It's included in both the web and Silverlight projects as an embedded resource.

When you run the application you'll see the the simple Employee-Order viewer you built in the Silverlight Tour, but this time it's using *Code First* under the hood!



Prerequisites

- The Silverlight 5 Toolkit
- Entity Framework Code First