Contents

What if your application must work with an existing database, perhaps with 300 tables or more? While you may want to write and maintain your entity model in Code First style, you're not looking forward to writing 300 entity classes by hand.

Fortunately, you don't have to. You can jump-start your code base by **generating the initial classes from the database schema** and then continue Code First from there. We call this approach "Code Second".

Platform: Silverlight
**Language:** C#
Download: Hello Code Second

Video

For a quick overview of *Code Second*, watch this 8-minute video.

Overview

As described in the video, there are a few simple steps to "Code Second" development.

Generate a Code First model

With the *Code Second* style of development, you start off with a database first approach: create your EDMX as usual, but before saving, choose the *CodeFirst template* from the EDM properties window:
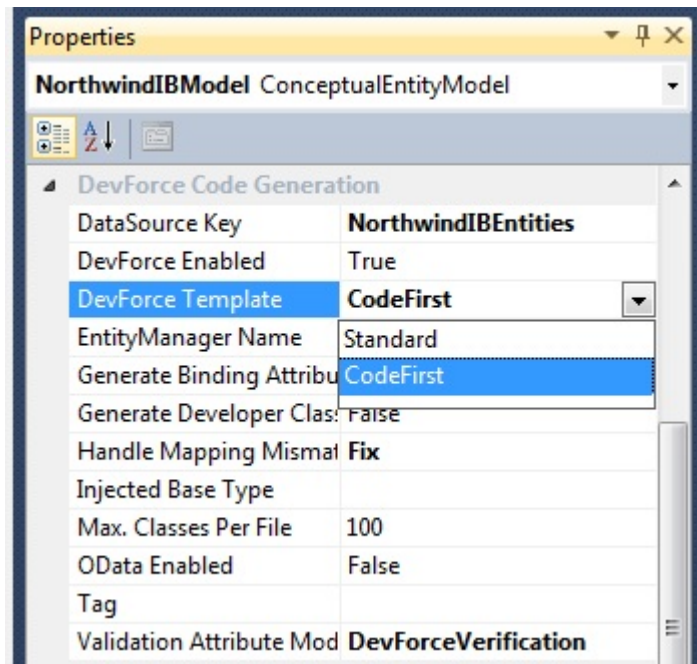
The *CodeFirst template* will generate the entities in your model as *Code First* entities. At this time you can also choose to generate data binding and validation attributes.

There are a few side affects to choosing the *CodeFirst template*: additional references have been added to the EntityFramework, IdeaBlade.Aop and PostSharp assemblies, and a DevForce "Code First marker file" has been added to the project to enable metadata generation when the project is built.

Take a moment to look at the generated code. As you've seen, Code First classes are simple POCO-style classes with a few minor changes, including the *ProvideEntityAspect* attribute which allows DevForce to inject entity behavior into your classes.

Here's a portion of the *Category* class from the NorthwindIB database:

```
[IbAop.ProvideEntityAspect]
[DataContract(IsReference=true)]
[Table("Category", Schema="dbo")]
public partial class Category {
 [Key]
 [DataMember]
 [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
 [Column("CategoryID")]
 public int CategoryID {
  get; set;
 }
 [DataMember]
 [Column("CategoryName")]
 public string CategoryName {
  get; set;
 }
 ...
}
```

You'll notice that DevForce has added data annotations to provide mapping between the database and your classes.

As with the *Standard* DevForce code generation template, you can modify the *CodeFirst template* as needed.

Remove the EDMX

Once you're done tweaking the model in the EDM designer to your satisfaction you must either remove the EDMX from your project or disable the *EntityDeploy* build action. Why? Because the Entity Framework will automatically generate "metadata artifact" files into the assembly, and your application won't work.

To remove the EDMX and associated files, you'll first need to copy the generated code to another file which won't be auto-generated when the EDMX is saved or the T4 template is run. You'll likely want to modify the generated code to suit your needs, and having DevForce auto-generate and wipe out your changes will be unwelcome. So do yourself a favor, and once you're happy with your model code, save it to another file.

If you think you might need the EDMX and want to keep it in the project, disable the *EntityDeploy* build action. This stops the generation of the artifact files.
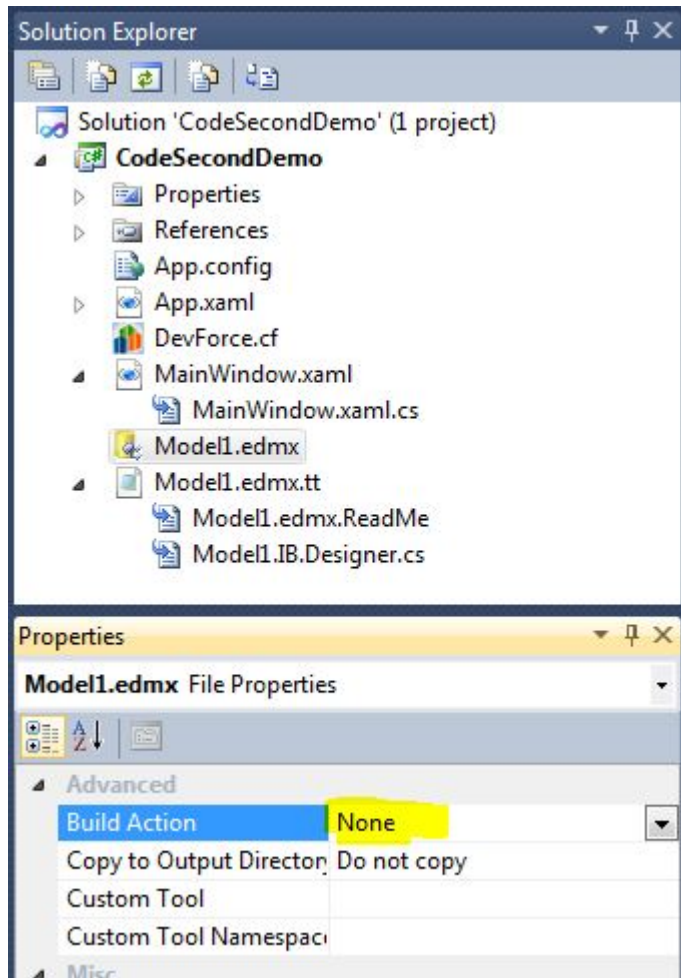
Fix the connection string

There's one last change to make before you're off and running doing *Code First* development. When you added the EDMX to your project the designer added an Entity Framework connection string to the project's configuration file. *Code First* doesn't understand this connection string so we'll need to modify it.

Here's the original Entity Framework connection string (with some editing for readability):

```
<add name="NorthwindIBEntities"
    connectionString="metadata=res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;
    provider=System.Data.SqlClient;provider connection string=&quot;
    data source=.;initial catalog=NorthwindIB;integrated security=True;multipleactiveresultsets=True;
    &quot;" providerName="System.Data.EntityClient" />
```

Note the **metadata** reference and the **providerName**. We'll remove these, so that we're left with a standard ADO.NET connection string:

```
<add name="NorthwindIBEntities"
    connectionString="data source=.;initial catalog=NorthwindIB;integrated security=True;multipleactiveresultsets=True;"
    providerName="System.Data.SqlClient" />
```

If your application configuration file is in another project, be sure to also copy this edited connection string to that file, otherwise your application will not run.