Northwind Code First Migration

# The data access layer using Entity Framework

The Northwind database is a demo created by Microsoft.

The database definition and seed data is in Resources/instnwnd2005\_original.sql.

You will need to install the database before starting. The EF setup depends on the database.

### EF mapping

Since the database is first, let’s map it. Anything which lives in a database is stored in the project Northwind.DAL (the data access layer).

Migrations info can be found here:

https://docs.microsoft.com/en-gb/ef/ef6/modeling/code-first/migrations/existing-database

On the package manager console:

> Install-Package Microsoft.EntityFrameworkCore.Tools

> Install-Package Microsoft.EntityFrameworkCore

> Install-Package Microsoft.EntityFrameworkCore.SqlServer

> Scaffold-DbContext "Server=[My database instance];Database=Northwind;Trusted\_Connection=true;MultipleActiveResultSets=true;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

The scaffold command creates models and database context in the Models folder.

The database context uses EF fluent API to configure the models. If it causes a problem, I will convert the same features to data annotations later. Data annotations might be better because the features like indexes, primary key and foreign keys are in the same place as the model and it will make the code more readable.

Data annotations can be found here:

https://docs.microsoft.com/en-gb/ef/ef6/modeling/code-first/data-annotations

To make the database and EF talk to each other a migration needs to be created:

> Add-Migration InitialCreate

This creates a migration folder and migration classes.

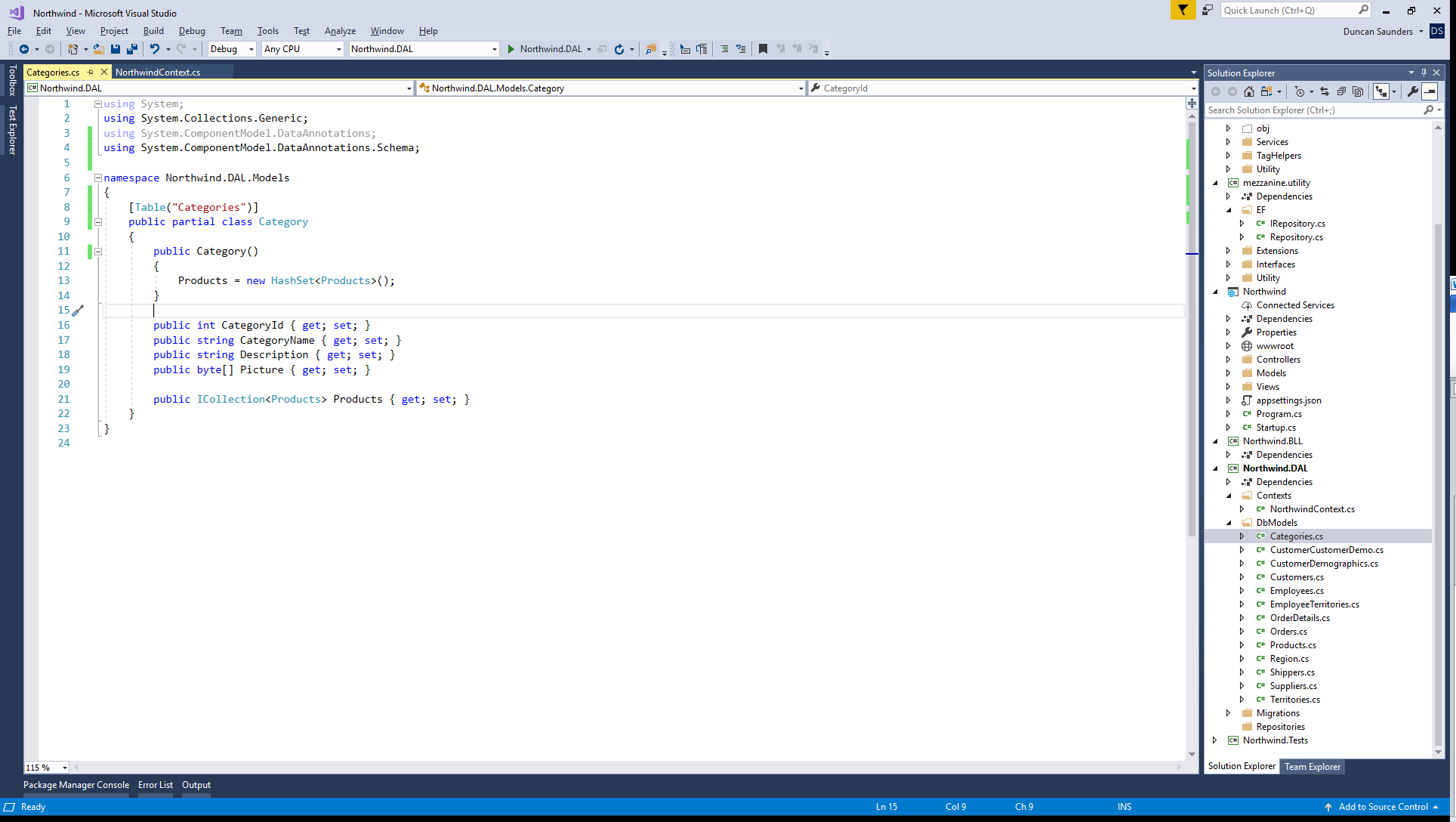
However, the migration assumes we don’t have a database so comment out all the code in the Up() method.

Finally, run a command to update the database and let EF know we have migrated it.

> Update-Database

You can see that the model names are the same as the table names. This will cause confusion later on. The models need to be renamed but we still need to tell EF that it should point to the correct table.

To do this, I need to add annotations:



Rename the class, then rename the file.

Check the renaming has worked by creating a migration

> Add-Migration RenameModels

Look in the up method. If there is anything in the Up() or Down() methods you have an error in your code. Both must be empty.

Remove the migration with

> Remove-Migration

## Updating the context and creating the repositories

Comment out the ‘OnConfiguring’ and empty constructor methods in the NorthwindContext. The connection string will be managed differently via a config file later on.

Each model needs a repository.

Repositories are exposed in the application via an interface and a concrete implementation.

This means that the repositories can be tested with any datasource.

There is some work to do creating each repository. They are similar in form but not similar enough to be made completely by an abstract class

A consistent format is being used. The abstract class mezzanine.EF.Repository<TModel, TKey> is used for the repositories.

All the repository does is provide CRUD operations and get records.

Any business logic takes place in Northwind.BLL.

To make managing the repositories easier, a startup class has been included. Your main application will need to call the methods in this class to configure the database context and automatically create and upgrade the database.

Automatic database schema and seed data from nothing will not work with this project as the database was first. You must create the database using the script detailed above.

Keeping the startup class in the data access layer means it can be managed separately from the main application.