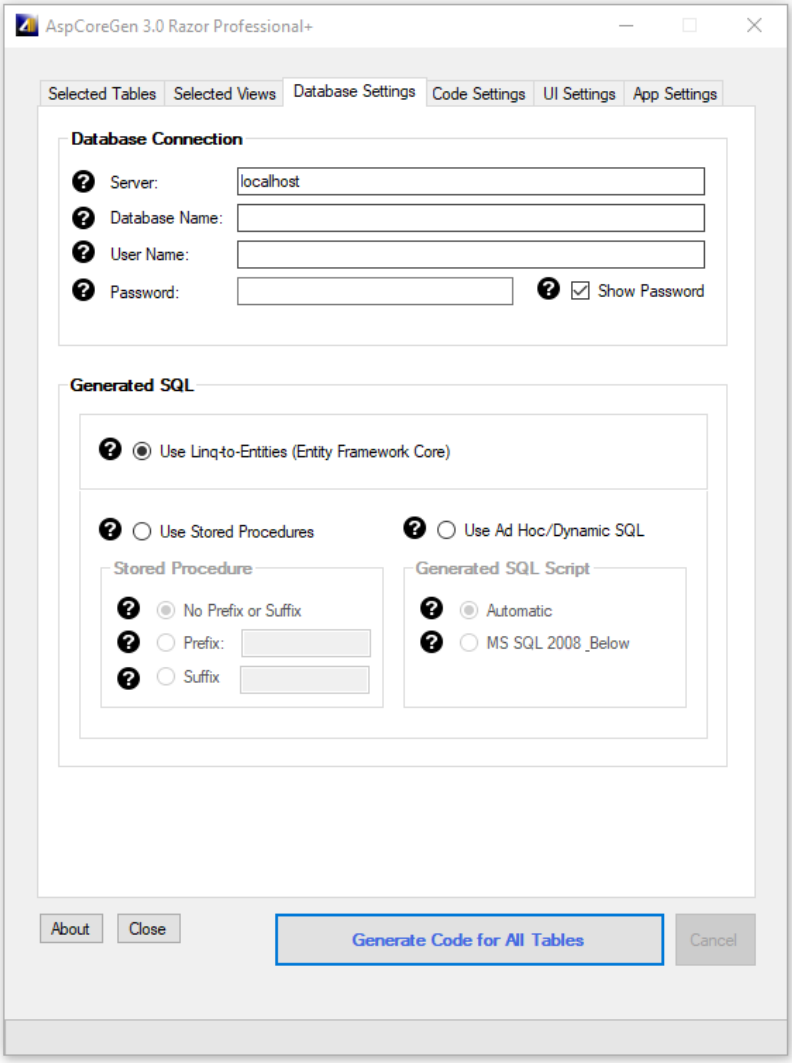
Database Settings Tab

1 CONTENTS

Database Settings Tab

The Database Settings Tab is where we set database-specific information such as the database connection properties and the type of SQL script to generate.



A. Database Connection

This is the MS SQL Server Database connection settings to the database. All of these items are required and cannot be blank.

1. **Server:** Server Name. If the MS SQL Server Database is local, you can just enter "*localhost*".
2. **Database Name:** The database you're trying to access.
3. **User Name:** An admin user name login that has access to the database.
4. **Password:** The password associated with the user name.
5. **Show Password:** Checking the *Show Password* in Figure 9 will remember the password you enter here so you don't have to enter it again the next time you open AspCoreGen 3.0 Razor.

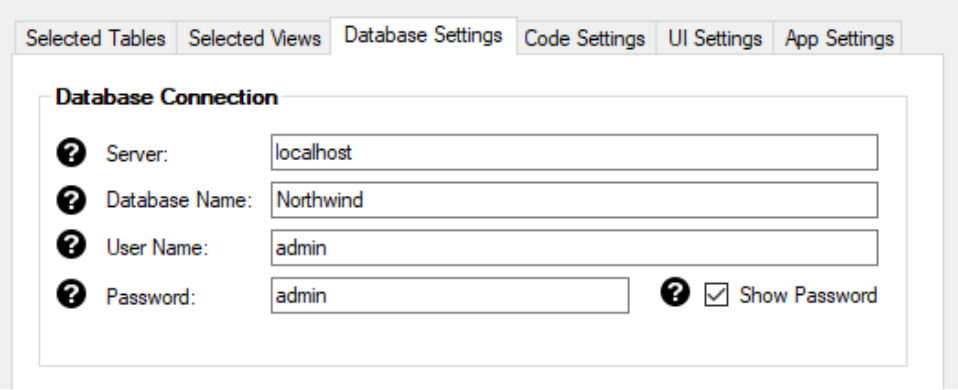
B. Generated SQL Script

The type of SQL script that will be generated.

1. **Use Linq-to-entities (Entity Framework Core):** Entity Framework.
2. **Use Stored Procedures:** SQL Scripts generated straight in your MS SQL database.
    a. **No Prefix or Suffix*:*** Names the generated script with no prefix or suffix. E.g. *Products_SelectAll*.
    b. **Prefix:** Names the generated script with a prefix. For example, when you enter *"sp_"* on the Prefix text box, the generated stored procs will start with an *"sp_"*. E.g. ***sp_****Products_SelectAll*.
    c. **Suffix:** Names the generated script with a suffix. For example, when you enter *"_sp"* on the Suffix text box, the generated stored procs will end with an *"_sp"*. E.g. *Products_SelectAll****_sp***.
3. **Use Ad-Hoc/Dynamic SQL:** SQL Scripts generated in your C# code.
    a. **Automatic:** Automatically determines your MS SQL Server version before generating scripts.
    b. **MS SQL 2008 Below:** Generates scripts for MS SQL Server versions 2008 and below. There's a slight difference with the generated script for older versions because some keywords are not available during this time.

1 Database Connection

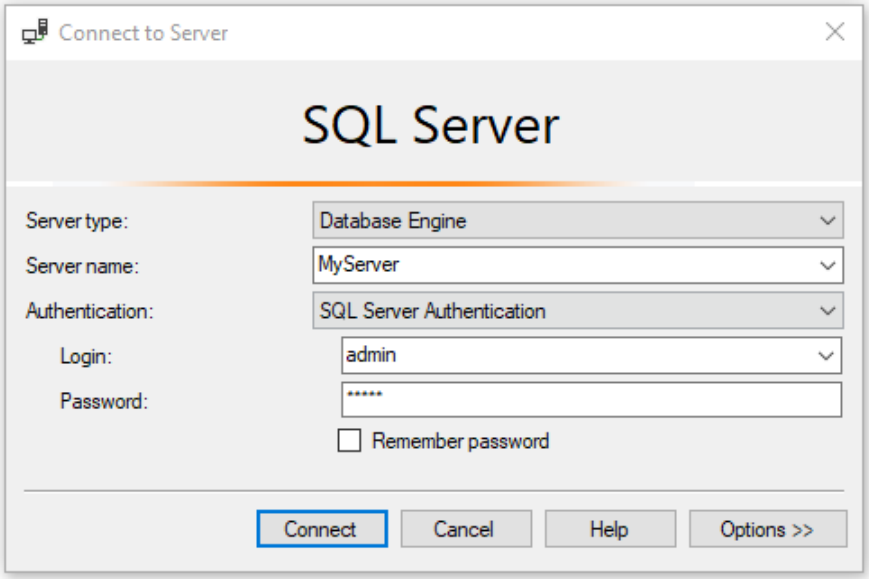
We use these fields/properties to connect to the MS SQL Server database, both to generate code and to perform CRUD (create, retrieve, update, delete) operations when you run the generated code.



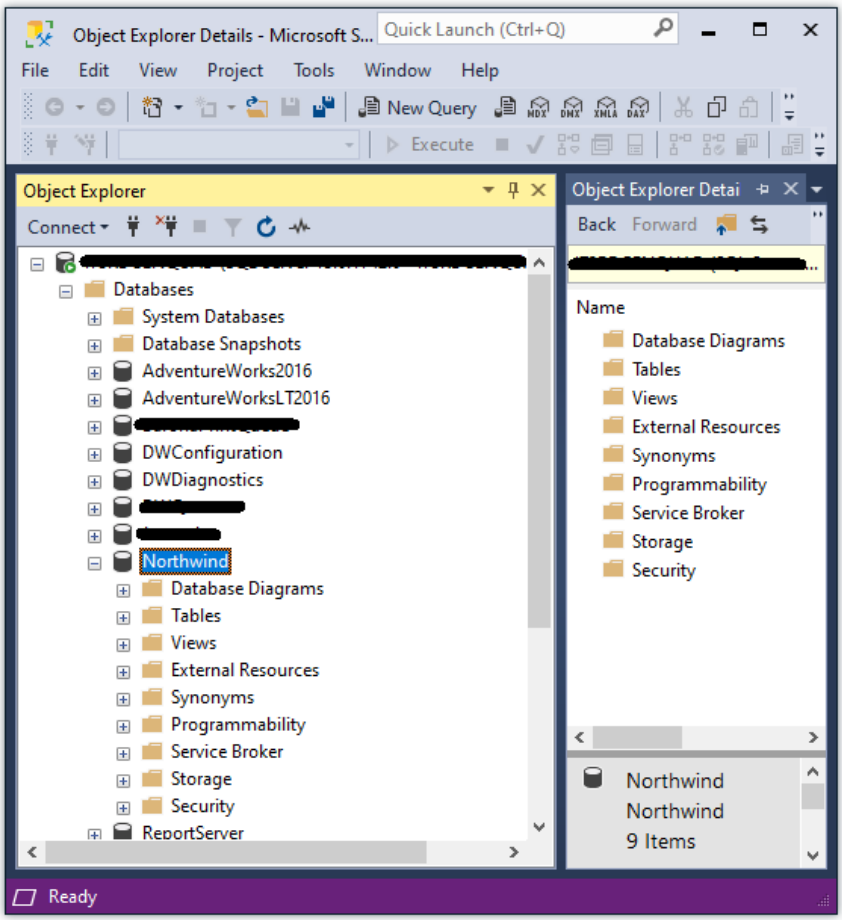
1.1 Using Database Connection to Connect to MS SQL Server Database to Generate Code

When you open your MS SQL Server Database it displays the *Connect to Server* dialog. Please see the direct relationship between the Database Connection fields you enter in AspCoreGen 3.0 Razor and your MS SQL Server Database below.

AspCoreGen 3.0 Razor	MS SQL Server Database
Server: *localhost* (when local) or *MyServer*	**Server Name:** *localhost* (when local) or *MyServer*
Database Name: *Northwind*	**Databases:** *Northwind*
User Name: *admin*	**Login:** *admin*
Password: *admin*	**Password:** *admin*
Show Password: *true*	n/a

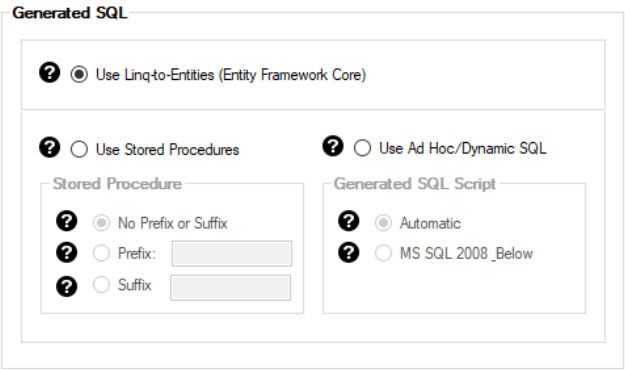


Once you're connected to MS SQL Server Database, you will see a list of *Databases* in the *Object Explorer*. The *Northwind* database is what we have in the example above (Database Name), the database we want to generate code from.

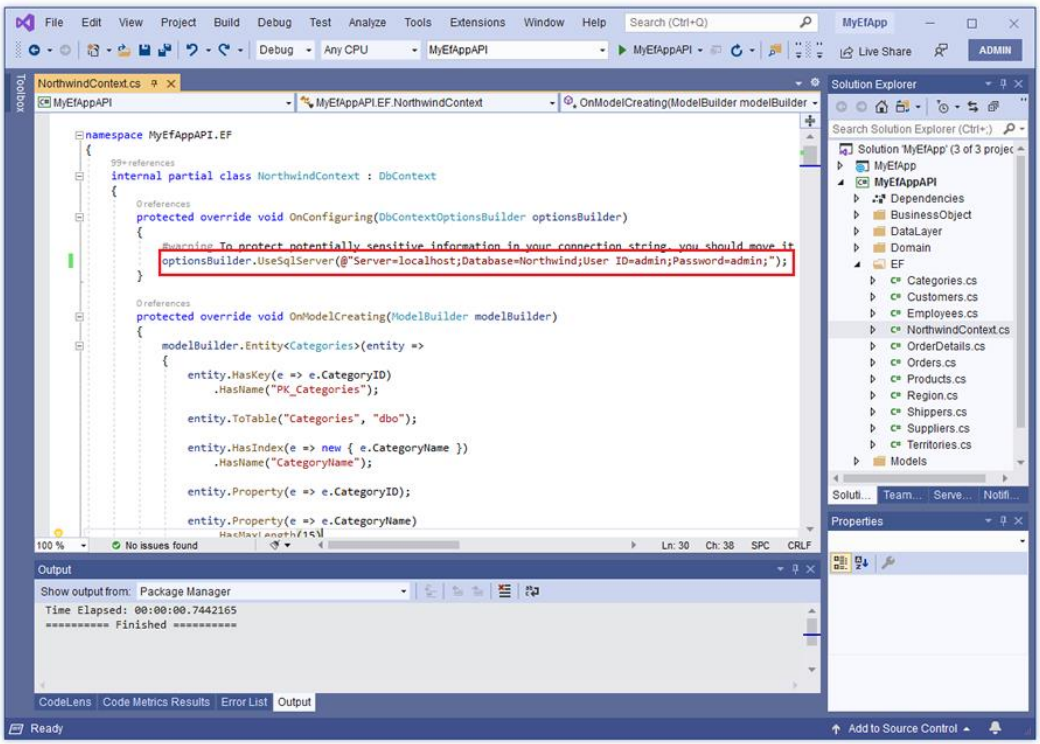


1.2 Using Database Connection to Connect to MS SQL Server Database to Perform CRUD

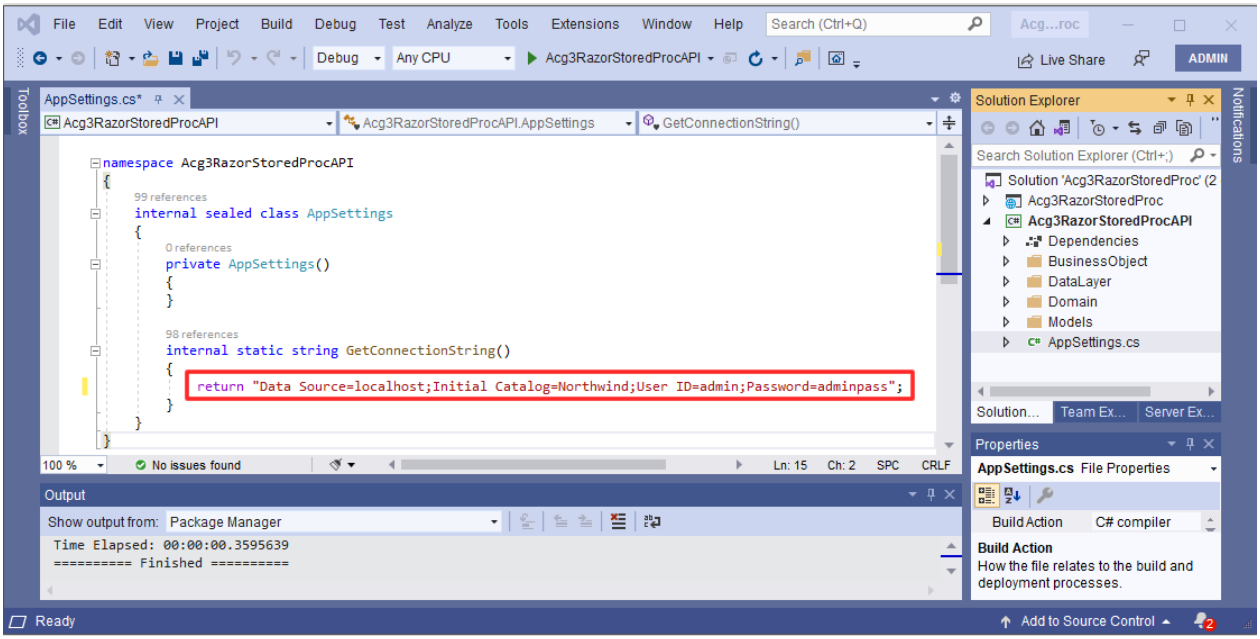
Once the code is generated, a part of the code is used to connect to the MS SQL Server Database you've chosen (*Northwind*) to perform CRUD operations on that database. In the generated code you will find the *Connection String* based on the *Generated SQL* option you have chosen.



Generated SQL	Location in Code
Use Linq-to-Entities (Entity Framework Core)	Business Layer and Data Layer API Project - In the *EF Directory*, In the …*Context.cs* class - In the example, you will find the *Connection String* in the *NorthwindContext.cs* class
Use Stored Procedures or, Use Ad Hoc/Dynamic SQL	Business Layer and Data Layer API Project - In the example, you will find the *Connection String* in the *AppSettings.cs* class



Use Linq-to-Entities (Entity Framework Core) – NorthwindContext.cs



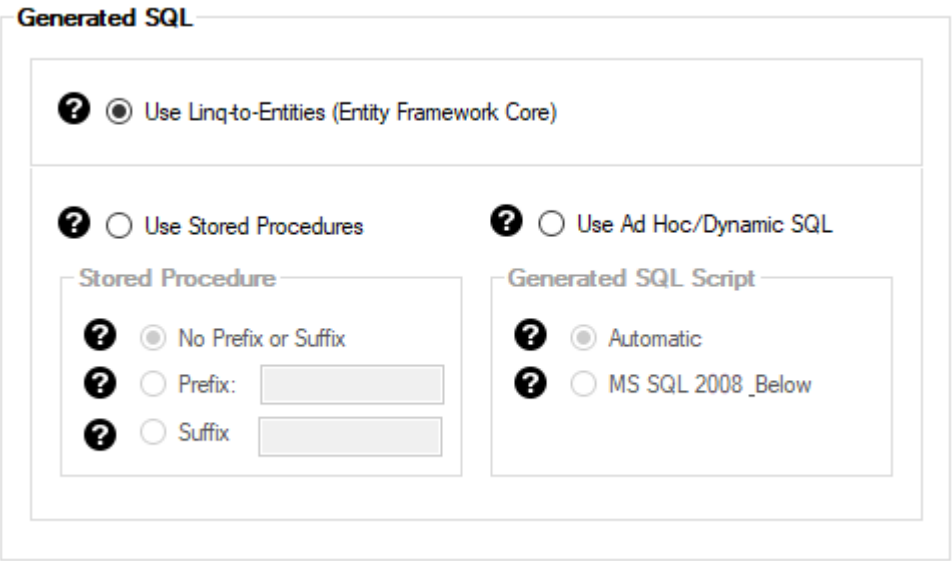
Use Stored Procedures or Use Ad Hoc/Dynamic SQL – AppSettings.cs

Note: You can change the *Connection String* values once you push your web application to your production environment. Make sure to point it to your Production Database instead.

2 GENERATED SQL

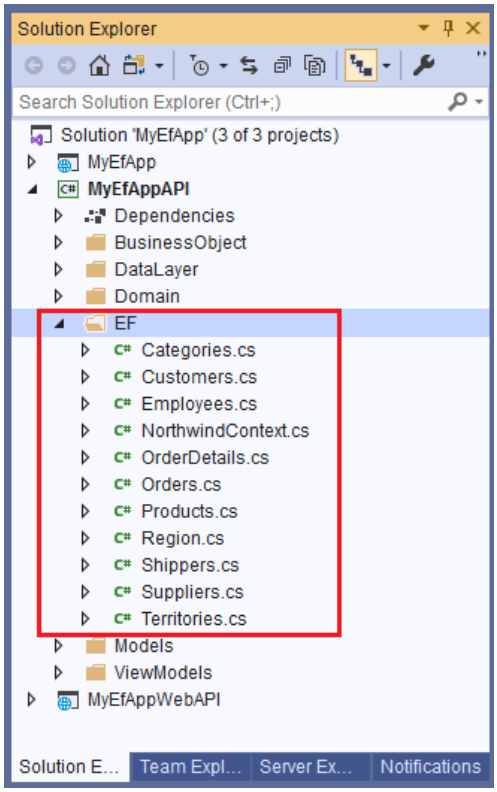
We use these fields/properties to determine the type of SQL Script that will be generated.

2.1 USE LINQ-TO-ENTITIES (ENTITY FRAMEWORK CORE)



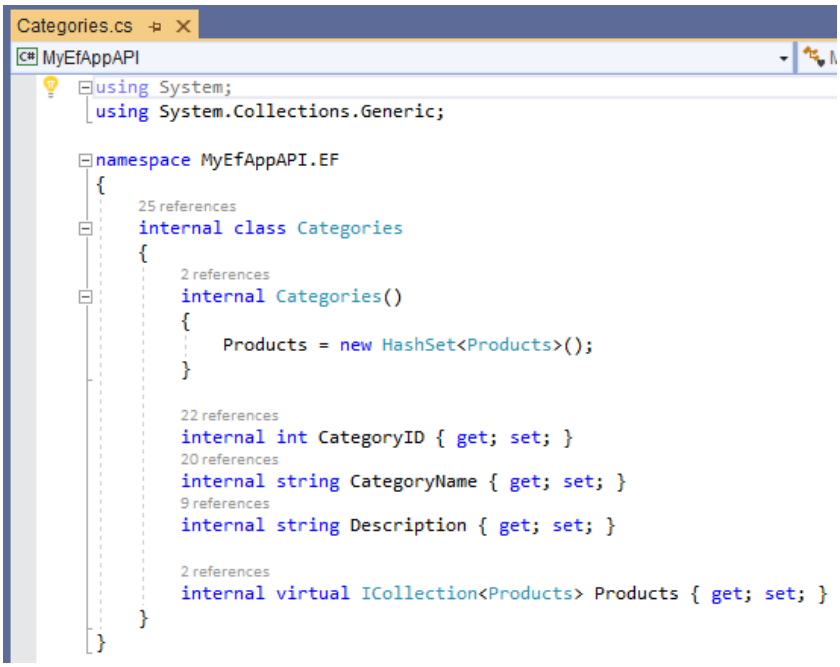
When you select *Use Linq-to-Entities (Entity Framework Core)* (default), the generated *Data Layer* code will be in an *Entity Framework Core Linq-to-Entities* format. The code will be generated in the:
Business Layer and Data Layer API Project

- EF Directory



Generated Entity Classes

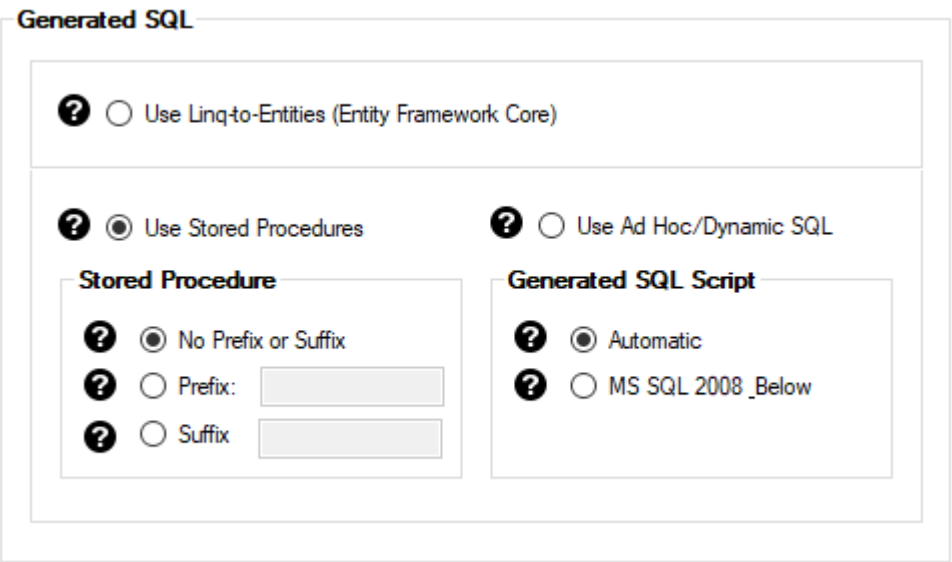
Each table selected for code generation will also have their respective class generated in the EF folder. E.g. *Categories* table will generate *Categories.cs* class, etc.



```
Categories.cs
MyEfAppAPI
using System;
using System.Collections.Generic;

namespace MyEfAppAPI.EF
{
    25 references
    internal class Categories
    {
        2 references
        internal Categories()
        {
            Products = new HashSet<Products>();
        }

        22 references
        internal int CategoryID { get; set; }
        20 references
        internal string CategoryName { get; set; }
        9 references
        internal string Description { get; set; }

        2 references
        internal virtual ICollection<Products> Products { get; set; }
    }
}
```

Generated Categories.cs Entity Class

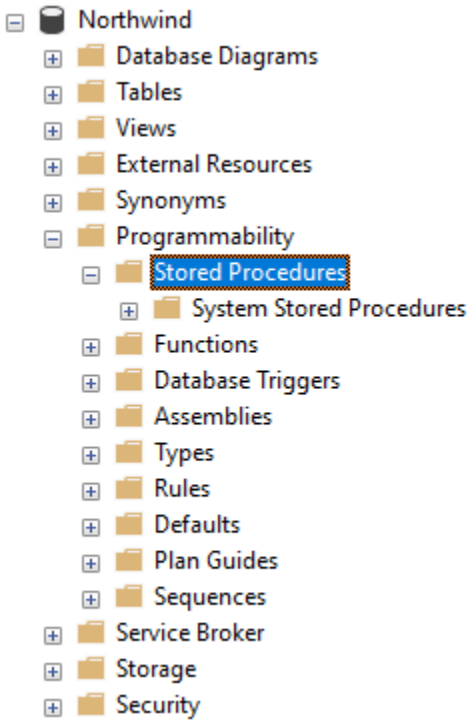
2.2 Use Stored Procedures



When you select *Use Stored Procedures*, the generated *SQL Script* code will be in a *Stored Procedure* form. The code will be generated in the MS SQL Server Database, in our example, SQL Scripts will be generated in the *Northwind* database. You will find the generated Stored Procedures:

Under the *Northwind* database

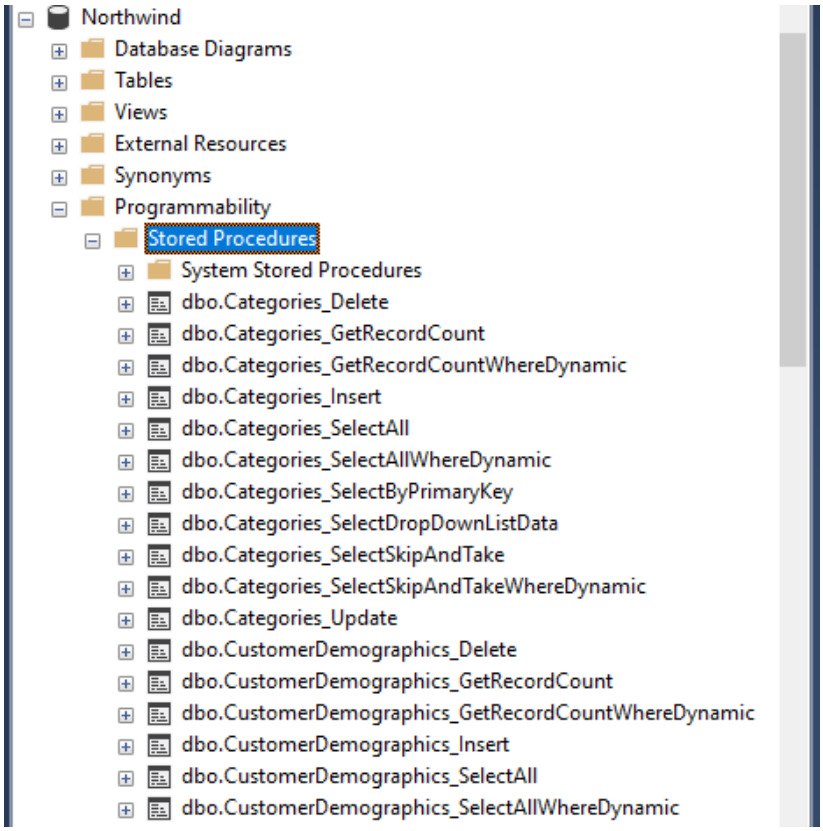
- *Programability* folder
    - *Stored Procedures* folder



Generated Stored Procedures will be under Stored Procedures folder

2.2.1 Use Stored Procedures – No Prefix or Suffix

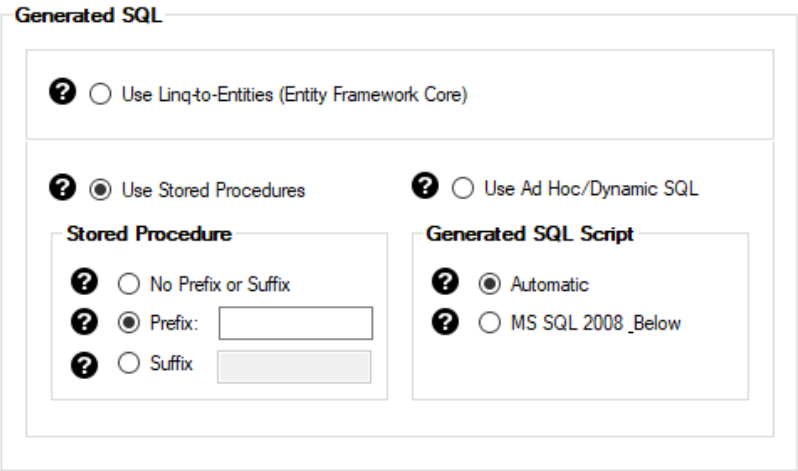
When you select *No Prefix or Suffix* (default), the generated *Stored Procedure Names* will have no prefixes or suffixes. E.g. *Categories_Delete, Categories_GetRecordCount*



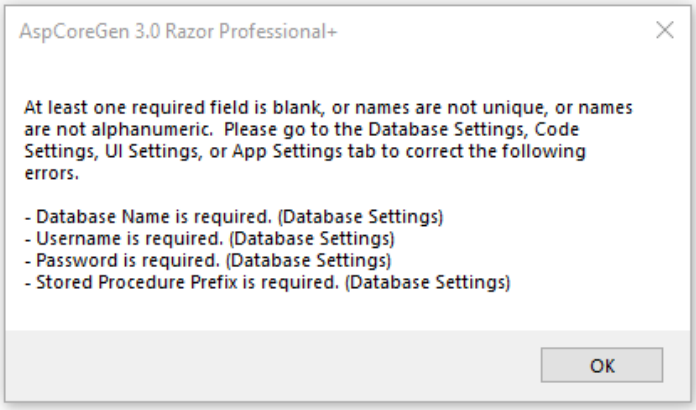
Stored Procedure Names with No Prefix or Suffix

2.2.2 Use Stored Procedures – Prefix

When you select *Prefix*, the generated *Stored Procedure Names* will have a prefix. The *Prefix* box is **required** when *Prefix* under *Stored Procedure* is selected.

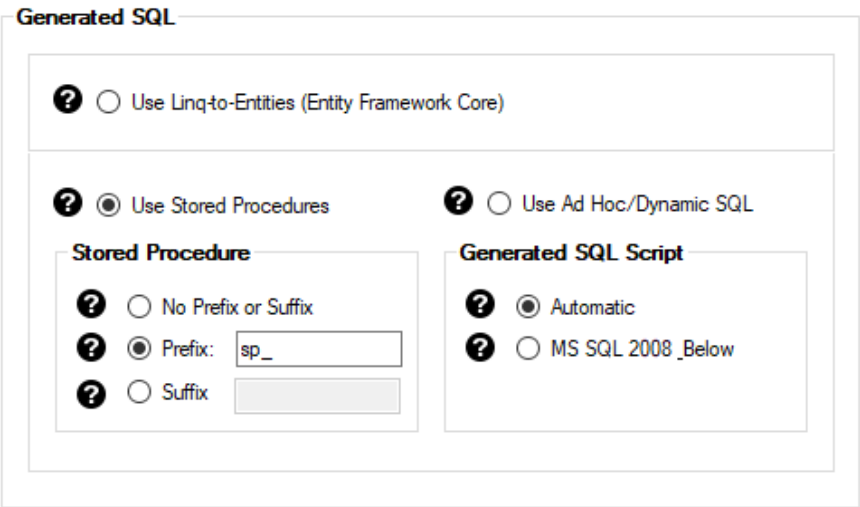


Prefix with Blank Box

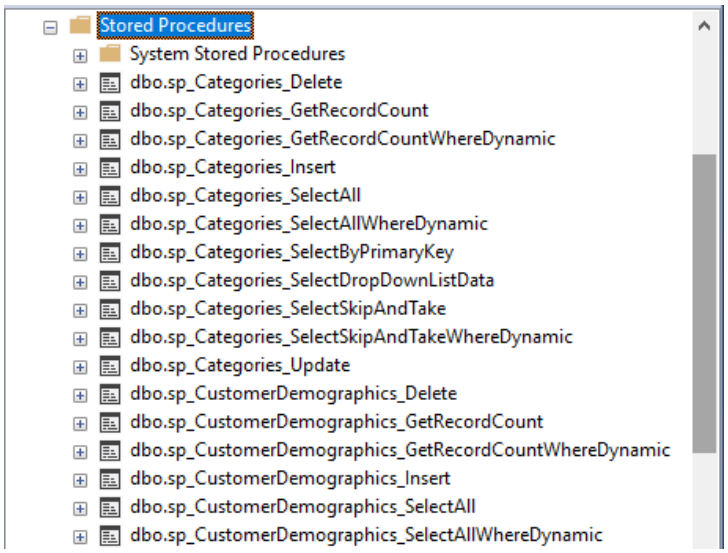


Prefix is required

The *Prefix* you enter will be used as the *Prefix* for the generated *Stored Procedure Names*. E.g. ***sp_****Categories_Delete,* ***sp_****Categories_GetRecordCount*.



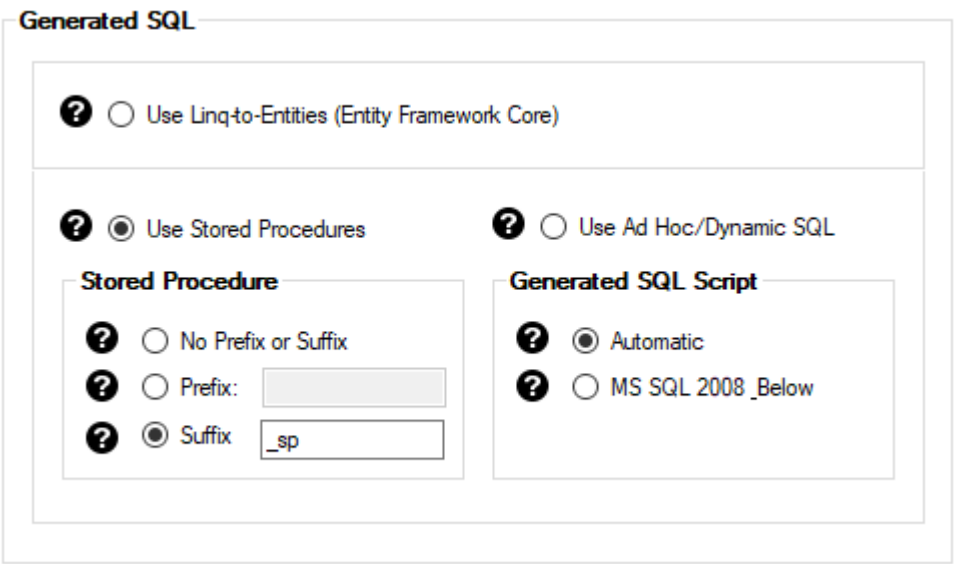
Prefix (filled)



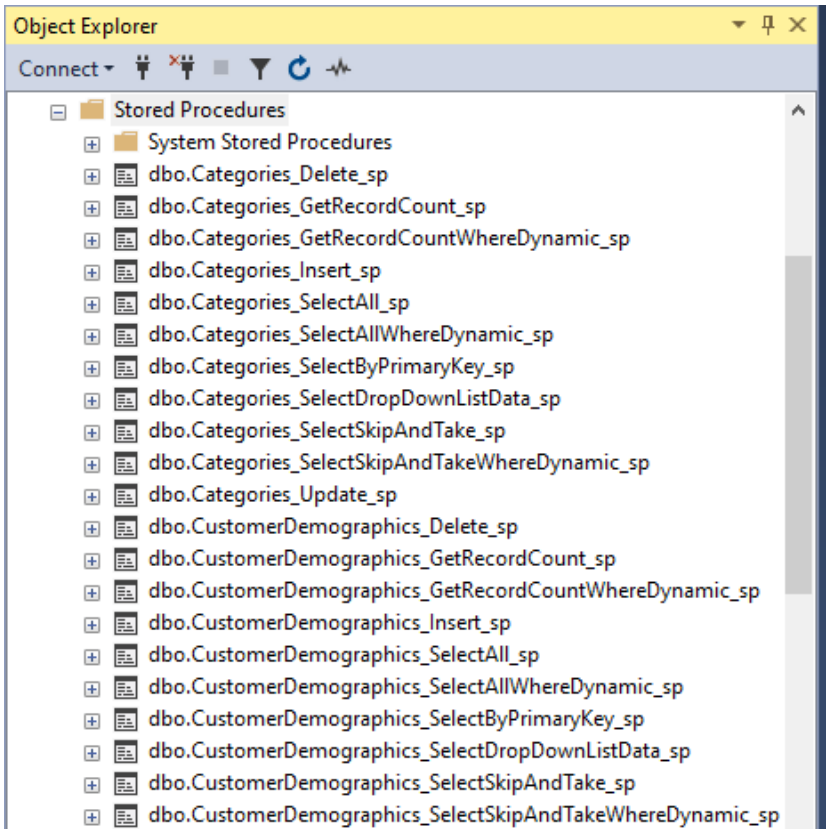
Stored Procedure Names with Prefix

2.2.3 Use Stored Procedures – Suffix

When you select *Suffix*, the generated *Stored Procedure Names* will have a prefix. The *Suffix* box is **required** when *Suffix* under *Stored Procedure* is selected. The *Suffix* you enter will be used as the *Suffix* for the generated *Stored Procedure Names*. E.g. *Categories_Delete_**sp**, Categories_GetRecordCount_**sp***.

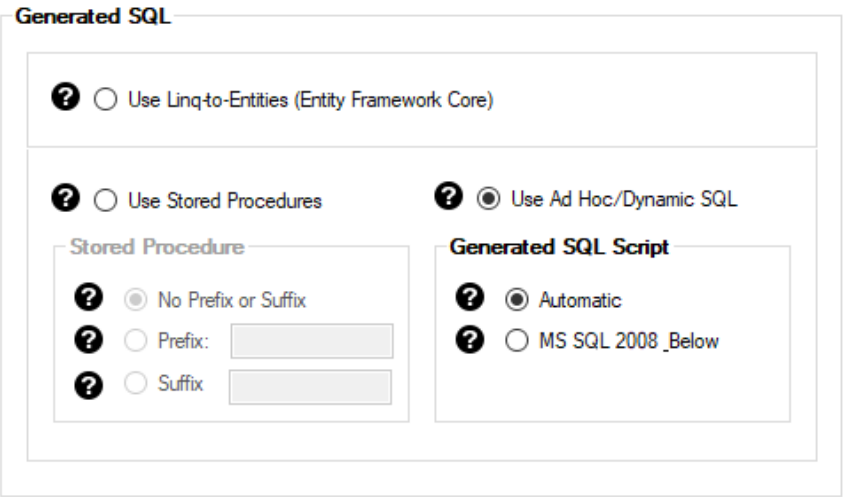


Suffix (filled)



Stored Procedure Names with Prefix

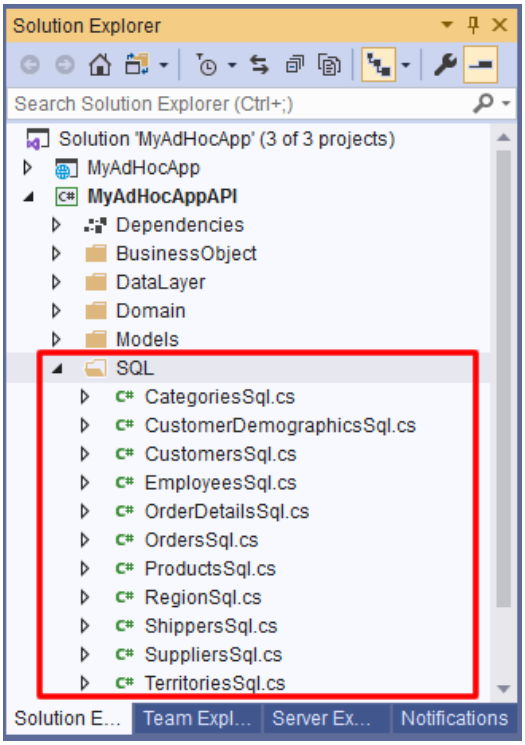
2.3 Use Ad Hoc/Dynamic SQL



When you select *Use Ad Hoc/Dynamic SQL*, the generated *SQL Script* code will be in generated in a class file. The code will be generated in the:

Business Layer and Data Layer API Project

- SQL Directory



Generated Ad Hoc SQL Classes

```
CategoriesSql.cs
MyAdHocAppAPI | MyAdHocAppAPI.DataLayer.Base.Ca | CategoriesSql()

using System;
using System.Text;

namespace MyAdHocAppAPI.DataLayer.Base
{
    12 references
    internal sealed class CategoriesSql
    {
        0 references
        private CategoriesSql()
        {
        }

        1 reference
        internal static string SelectByPrimaryKey()
        {
            string selectStatement = GetSelectStatement();
            StringBuilder sb = new StringBuilder();

            sb.Append(selectStatement);
            sb.Append(" WHERE ");
            sb.Append("[CategoryID] = @categoryID ");

            return sb.ToString();
        }

        1 reference
        internal static string GetRecordCount()
        {
            StringBuilder sb = new StringBuilder();

            sb.Append("SELECT COUNT(*) AS RecordCount FROM [dbo].[Categories]");

            return sb.ToString();
        }
```

CategoriesSql.cs – Generated SQL – Ad Hoc

2.3.1 Automatic

Automatic is selected by default. This will automatically determine the MS SQL Server you're using and generate scripts based on the version. Newer generated SQL script keywords are used for MS SQL Server versions that is 2008 and above.

2.3.2 MS SQL 2008 Below

When selected, SQL script keywords used in older (older than 2008) versions are used in the generated scripts. Although these keywords may still run using the newer versions of MS SQL Server, they may not be as fast.

You can read end-to-end tutorials on more subjects on using AspCoreGen 3.0 Razor Professional Plus that came with your purchase. These tutorials are available to customers and are included in a link on your invoice when you purchase AspCoreGen 3.0 Razor Professional.

Note: Some features shown here are not available in the Express Edition.

End of tutorial.