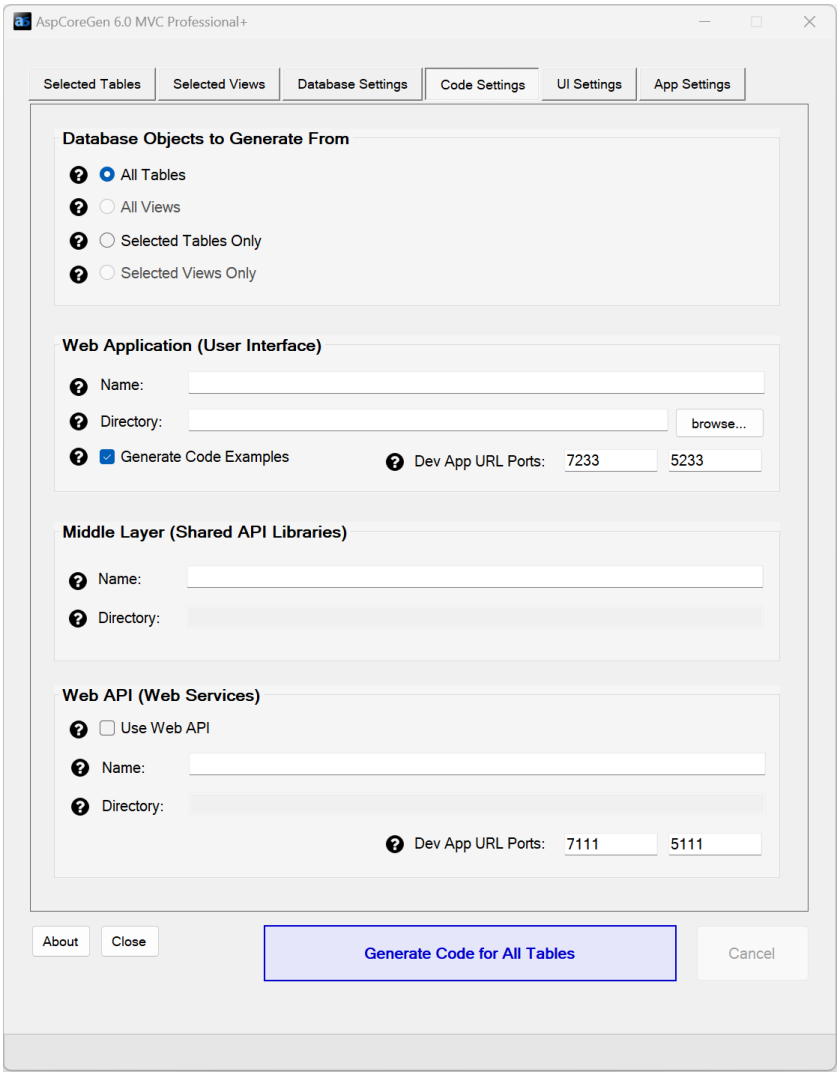
Code Settings Tab

1 CONTENTS

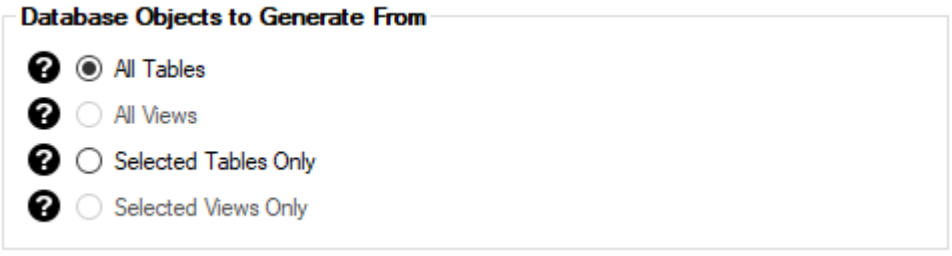
Code Settings Tab

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



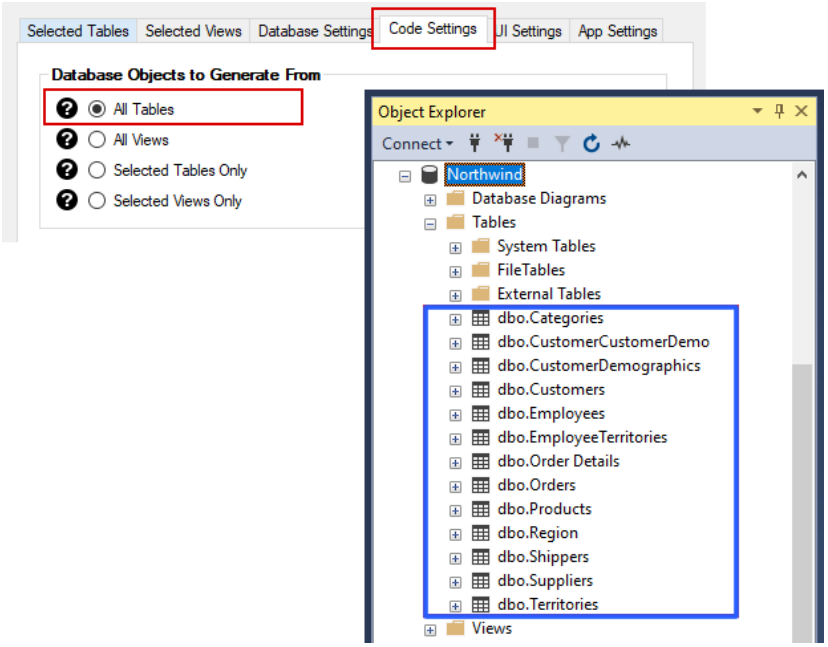
1 Database Objects to Generate From

This is where we set what we want to generate code from; Database Tables or Database Views.



1.1 ALL TABLES

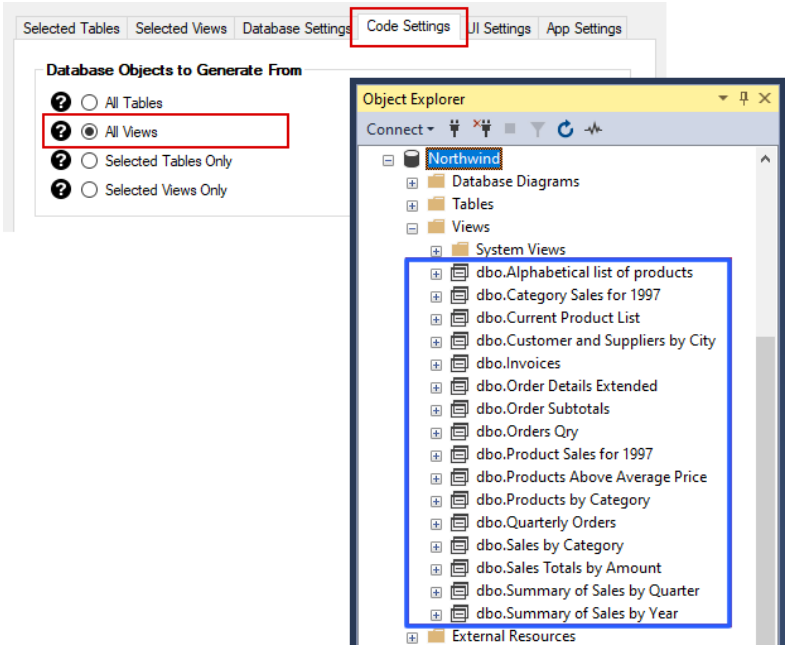
All Tables is the default. When this is selected, code will be generated for all the tables in your target MS SQL Server Database. In our examples, these would be all the tables listed in the Northwind database.



Code Settings Tab & Database Tables in MS SQL Server Northwind Database

1.2 ALL VIEWS

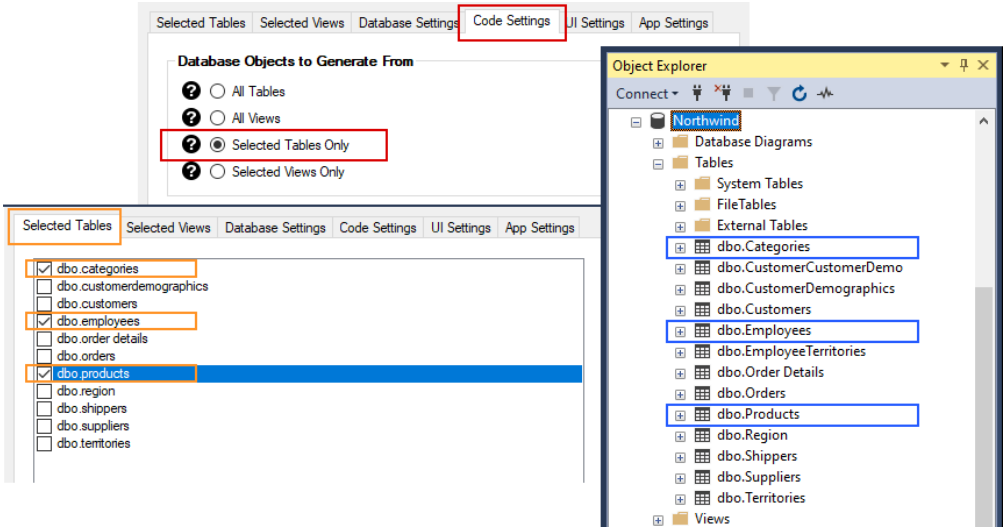
When this is selected, code will be generated for all the views in your target MS SQL Server Database. In our examples, these would be all the views in the Northwind database. This option is not available and will be disabled when *Use Linq-to-Entities (Entity Framework Core)* is selected under *Generated SQL* in the *Database Settings* tab.



Code Settings Tab & Database Tables in MS SQL Server Northwind Database

1.3 SELECTED TABLES ONLY

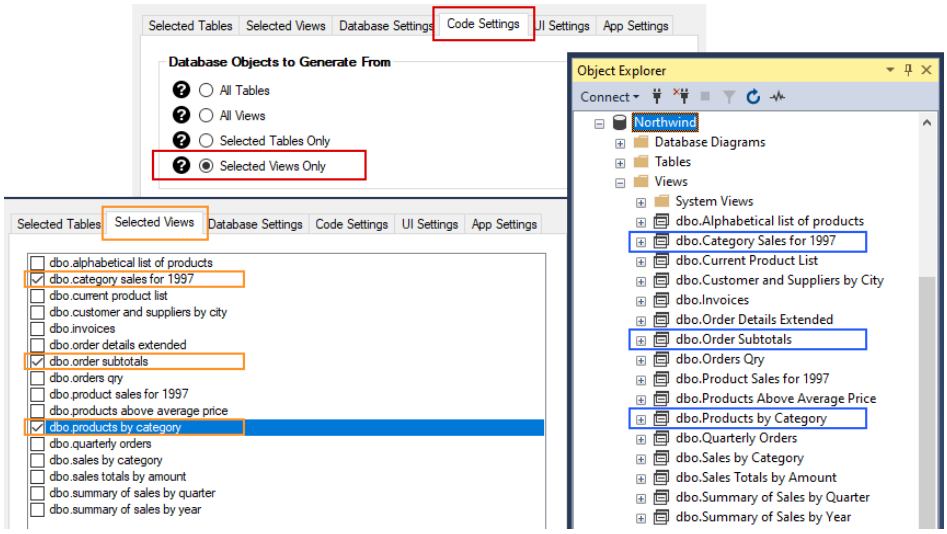
When this is selected, code will be generated for the selected tables in your target MS SQL Server Database. The *Selected Tables Only* option is directly related to the *Selected Tables* tab of the application. When you click on this option, the *Selected Tables* tab will automatically open*. You can select the tables you want to generate from on the *Selected Tables* tab. For more information on the *Selected Tables* tab, please see the documentation/tutorial for the *Selected Tables* tab titled "Selected Tables Tab".



Code Settings Tab, Selected Tables Tab, & Database Tables in MS SQL Server Northwind Database

1.4 SELECTED VIEWS ONLY

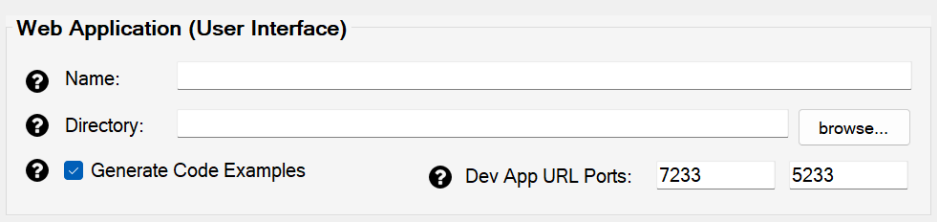
When this is selected, code will be generated for the selected views in your target MS SQL Server Database. The *Selected Views Only* option is directly related to the *Selected Views* tab of the application. When you click on this option, the *Selected Views* tab will automatically open*. You can select the views you want to generate from on the *Selected Views* tab. For more information on the *Selected Views* tab, please see the documentation/tutorial for the *Selected Views* tab titled "Selected Views Tab". This option is not available and will be disabled when *Use Linq-to-Entities (Entity Framework Core)* is selected under *Generated SQL* in the *Database Settings* tab.



Code Settings Tab, Selected Views Tab, & Database Views in MS SQL Server Northwind Database

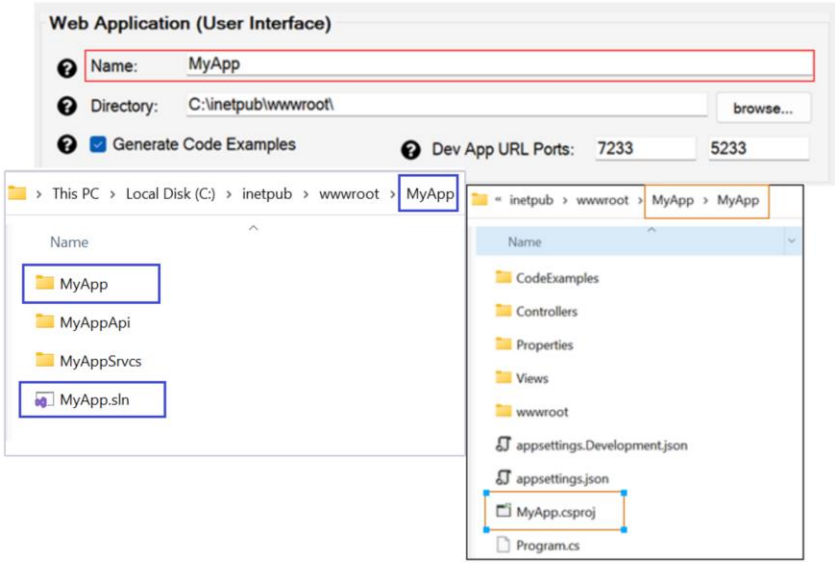
2 WEB APPLICATION

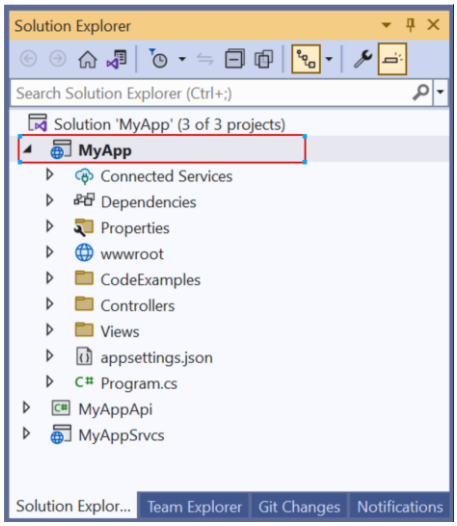
These fields/properties determines the web application Project's Name and the Directory where the code will be generated in.



2.1 NAME

The *Name* of the *Web Application Project (.csproj – ASP.NET MVC Core Project)*, the *Solution File (.sln)*, and the project's *Folder* that will be generated.

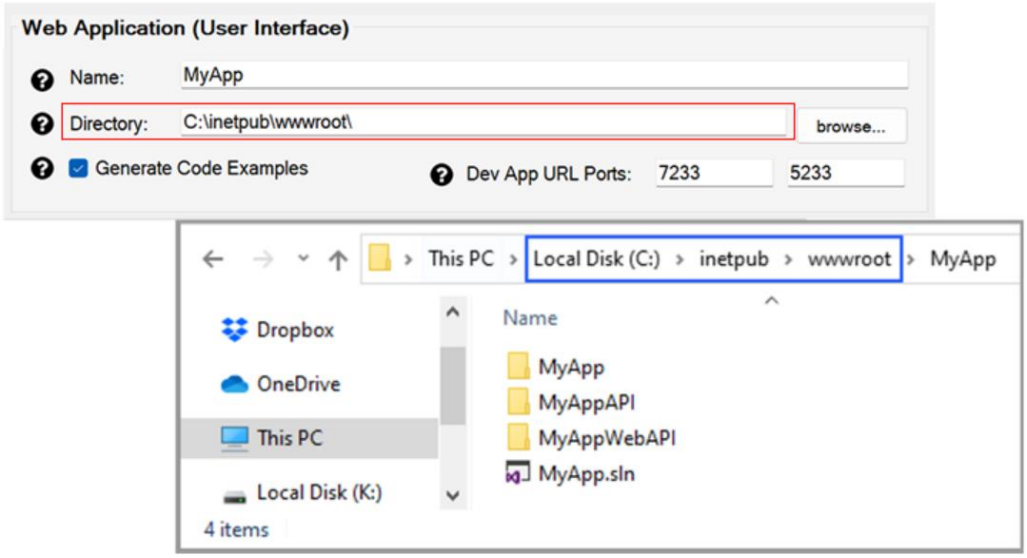




Web Application (Project) Name – Visual Studio

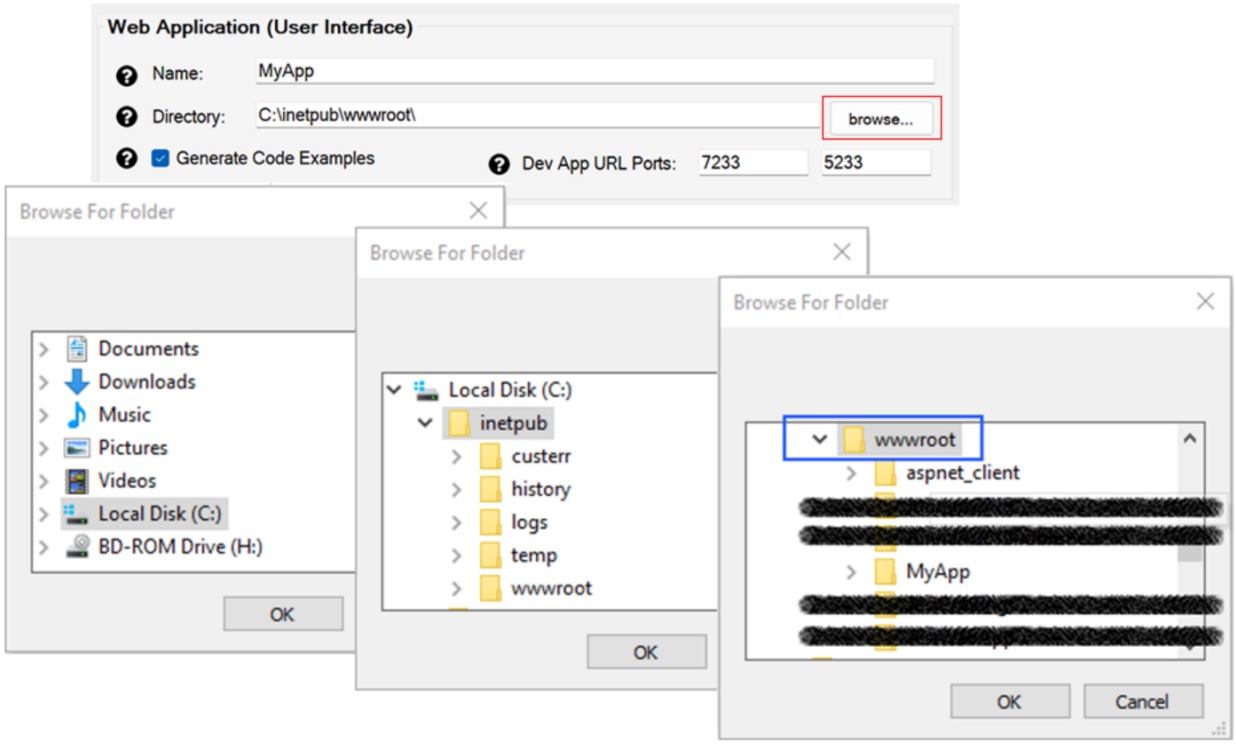
2.2 DIRECTORY

The *Directory* or *Folder* where the *Web Application Project* will be generated in. You can manually enter the Directory here, or use the *Browse* button as discussed below.



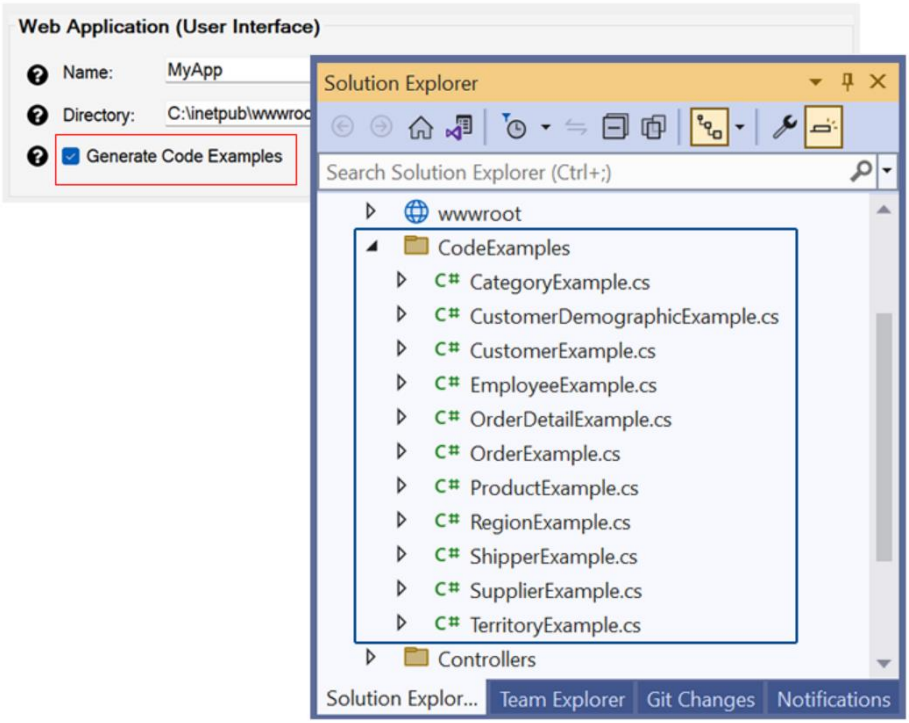
2.3 BROWSE BUTTON

The *Browse* button is optional. You can use it to choose the *Directory* where you want the generated Web Application to be generated in by browsing to a *Folder* in your computer.

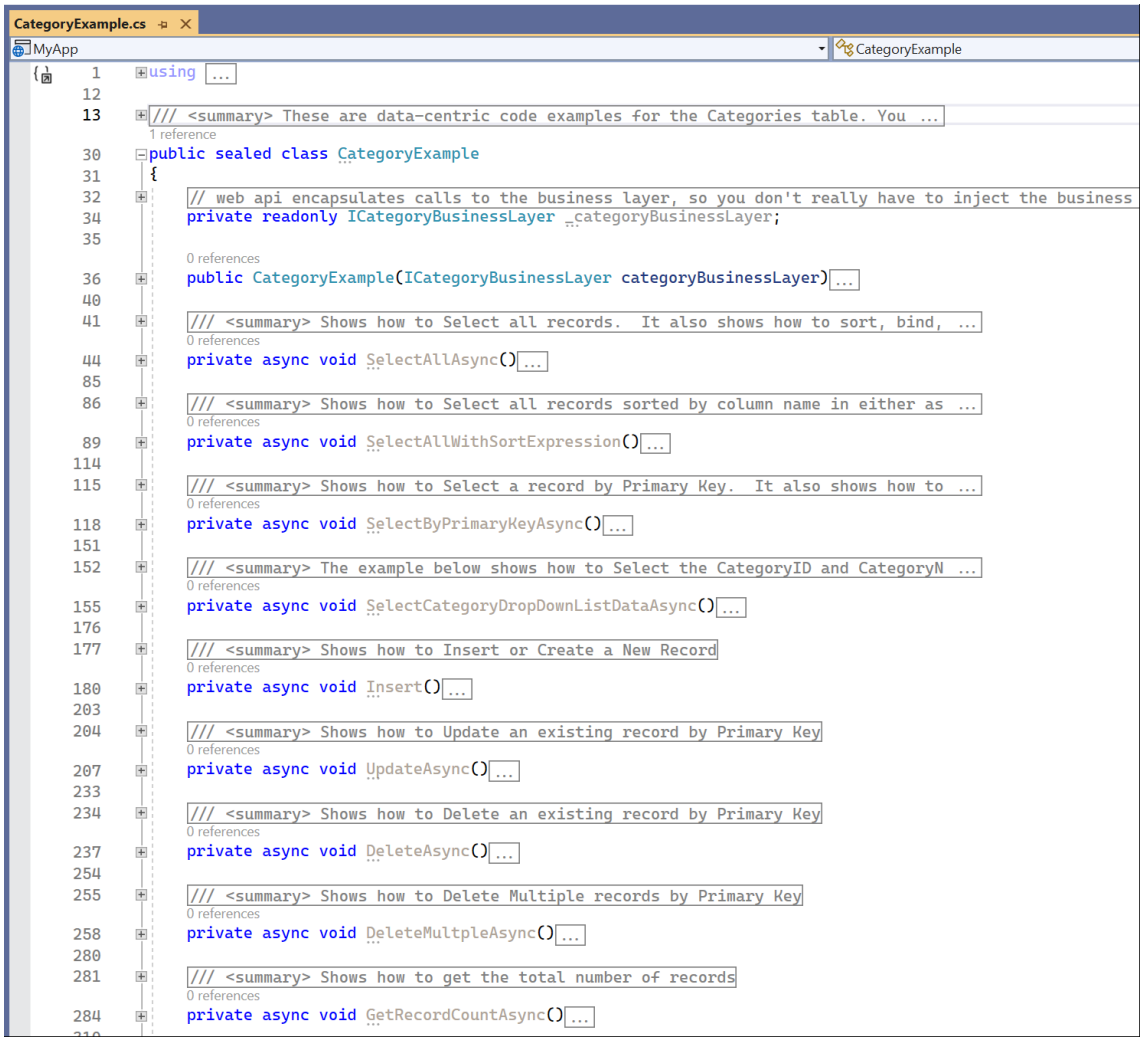


2.4 GENERATE CODE EXAMPLES

Check this box when you want to *Generate Code Examples*. The generated code examples will be under the *CodeExamples* folder. Each database table or view will generate an *Example* class file.



Each *Example* class file will contain various *CRUD* operations targeting the respective database table or view. An example of the generated *CategoryExample* class is shown below. The *CRUD* operation examples targets the *Categories* table in the *Northwind* database.



1. These *CRUD* operations can be accessed by any client, e.g. web forms, WCF class, Silverlight app, asp.net MVC controller, Windows Forms, Web API, etc.
2. Each *CRUD* operation example accesses the generated *Business Layer (middle tier)* class or *Web API* controller.
3. Every method shows an example for a specific operation.
4. Every method is documented by comments about the example being shown.
5. Every method is private because it's not meant to be used outside of the containing method, they are simply made as code examples.

Here's an expanded *SelectAll* (method) operation example for the *Categories* table. It shows the following:

1. *Select All* records from the *Categories* table using the *Web API*.
2. Or, if you like, you can uncomment the first line of code (and comment the *Web API* code) to *Select All* records from the *Categories* table using the *Business Layer* instead.
3. In *Example 1*, it shows how to *Sort* all the records in *Ascending* order *ByCategoryName*.
4. *Example 2* shows how to *Sort* all the records in *Descending* order *ByCategoryName*.
5. *Example 3* shows how to bind your data source (all the records that was retrieved) to a *GridView*. Of course this is just an example. You can bind your data source to any control that will take a *Generic List*, which is pretty much all the controls out there.
6. *Example 4* shows that you can use the data source in a *Foreach* loop. Again, this could have been any kind of loop, e.g. *For*, *While*, *Do While*, etc.

CategoryExample.cs

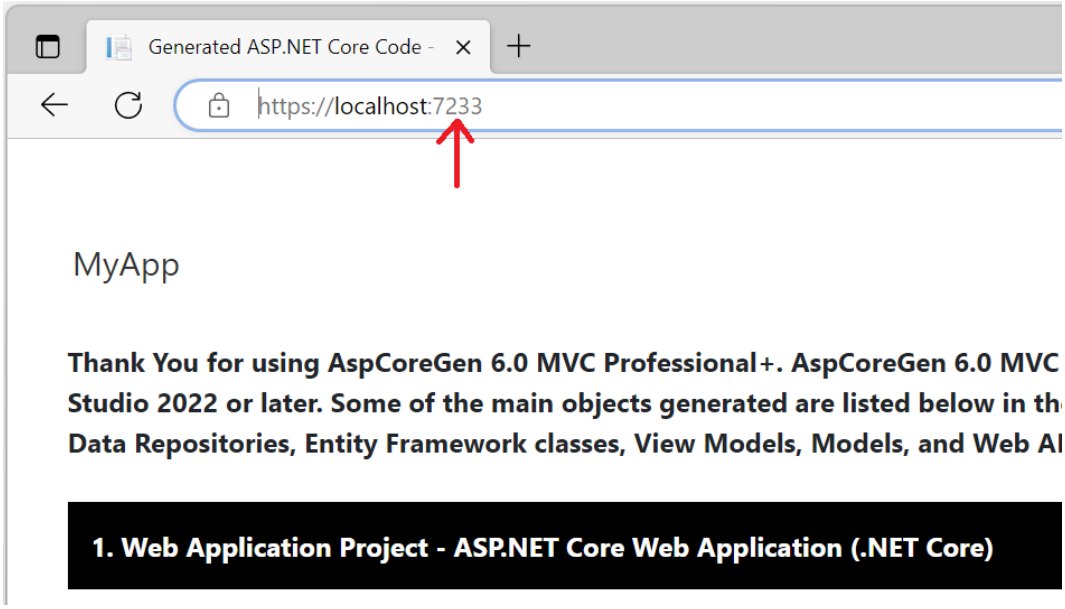
MyApp — CategoryExample

```
41      /// <summary> Shows how to Select all records.  It also shows how to sort, bind,  ...
        0 references
44      private async void SelectAllAsync()
45      {
46          // select all records
47          // uncomment this code if you would rather access the business layer instead of the web api
48          // List<Category> objCategoriesList = await _categoryBusinessLayer.SelectAll();
49
50          // ** code using the web api instead of the business layer ************************************
51          List<Category> objCategoriesList = null;
52
53          using (var client = new HttpClient())
54          {
55              client.BaseAddress = new Uri(Functions.GetWebApiBaseAddress());
56              HttpResponseMessage response = await client.GetAsync("CategoryApi/SelectAllAsync");
57
58              if (response.IsSuccessStatusCode)
59              {
60                  var responseBody = response.Content.ReadAsStringAsync().Result;
61                  objCategoriesList = JsonConvert.DeserializeObject<List<Category>>(responseBody);
62              }
63          }
64          // ** code using the web api instead of the business layer ************************************
65
66          // Example 1:  you can optionally sort the collection in ascending order by your chosen field
67          objCategoriesList.Sort(CategoryBusinessLayer.ByCategoryName);
68
69          // Example 2:  to sort in descending order, add this line to the Sort code in Example 1
70          objCategoriesList.Reverse();
71
72          // Example 3:  directly bind to a GridView - for ASP.NET Web Forms
73          // GridView grid = new GridView();
74          // grid.DataSource = objCategoriesList;
75          // grid.DataBind();
76
77          // Example 4:  loop through all the Categories
78          foreach (Category objCategory in objCategoriesList)
79          {
80              int categoryID = objCategory.CategoryID;
81              string categoryName = objCategory.CategoryName;
82              string description = objCategory.Description;
83          }
84      }
```

Note: Although none of the generated MVC views use the *Select All* operation, this is still included in the generated functions of the *Middle-Tier* or *Web API*. The industry does not recommend using select all operations unless you know that you're only getting a small amount of data.

2.5 DEV APP URL PORTS

The *Dev App Url Ports* are used during development only. These settings can be found in the *launchSettings.json* file under the *Properties* folder in the Web Application.

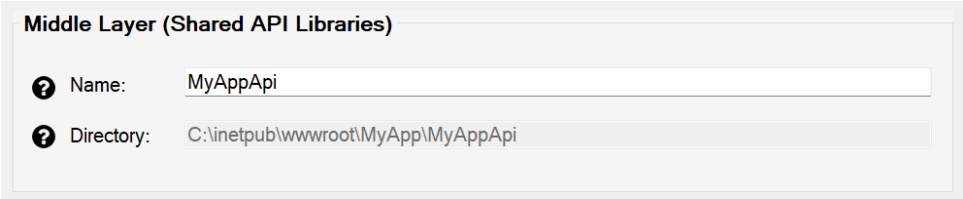


https://localhost:7233 (Default)

applicationURL opens in https://localhost:7233 by Default

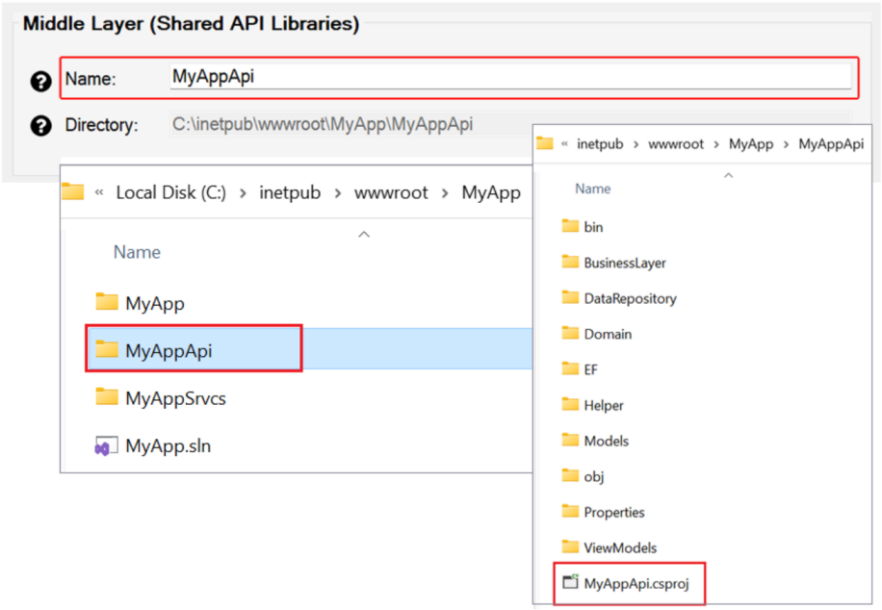
3 MIDDLE LAYER (SHARED API LIBRARIES)

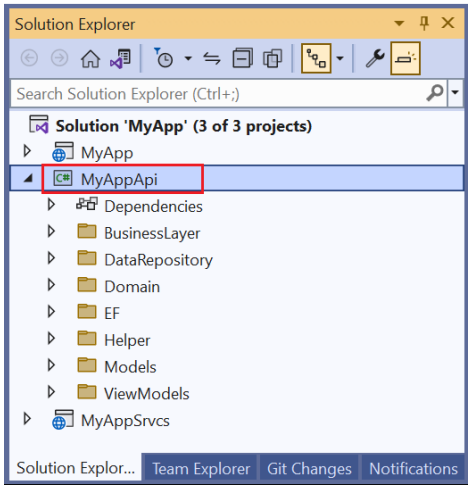
These fields/properties determines the Middle Layer's (Class Library Project) Name and the Directory where the code will be generated in. This project contains the *Business Layer*, *Data Repository* and *Shared Libraries*.



3.1 NAME

The *Name* of the *Middle Layer Project (.csproj – Class Library Project)* and the project's *Folder* that will be generated.

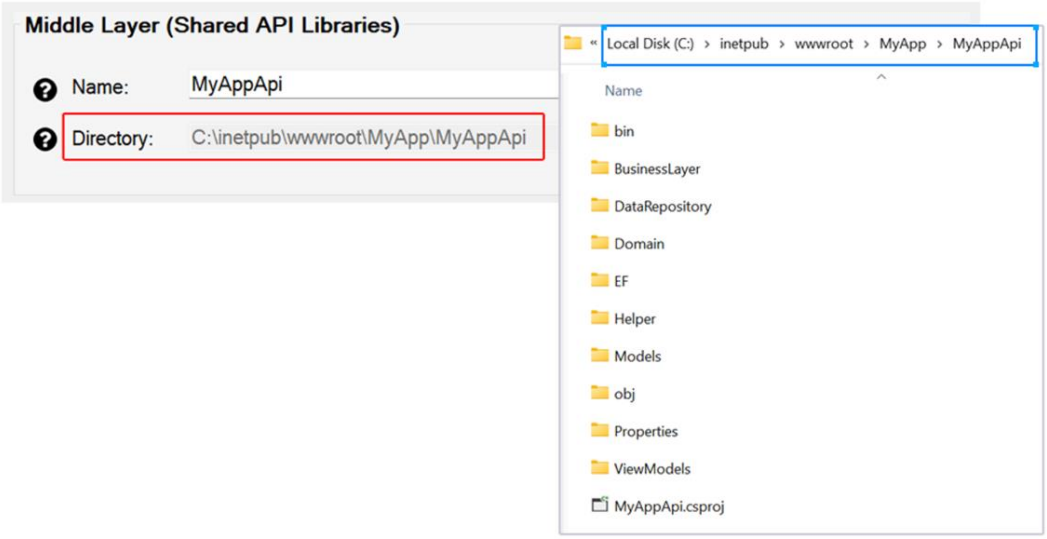




Middle Layer Project Name – Visual Studio

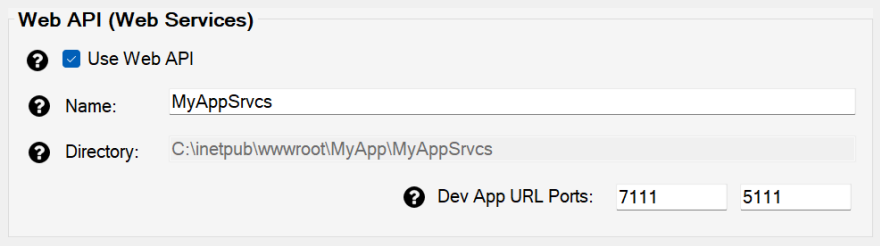
3.2 DIRECTORY

The *Directory* or *Folder* where the *Middle Layer Project* will be generated in. The *Directory* is automatically filled relative to the *Web Application's Directory* and placed in the same-named *Directory* as the Middle Layer Project *Name*.



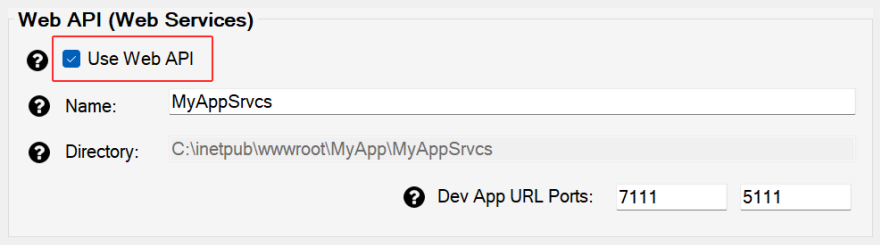
4 WEB API

These fields/properties determines the Web API Project's Name and the Directory where the code will be generated in, and whether to generate this optional Web API Project.



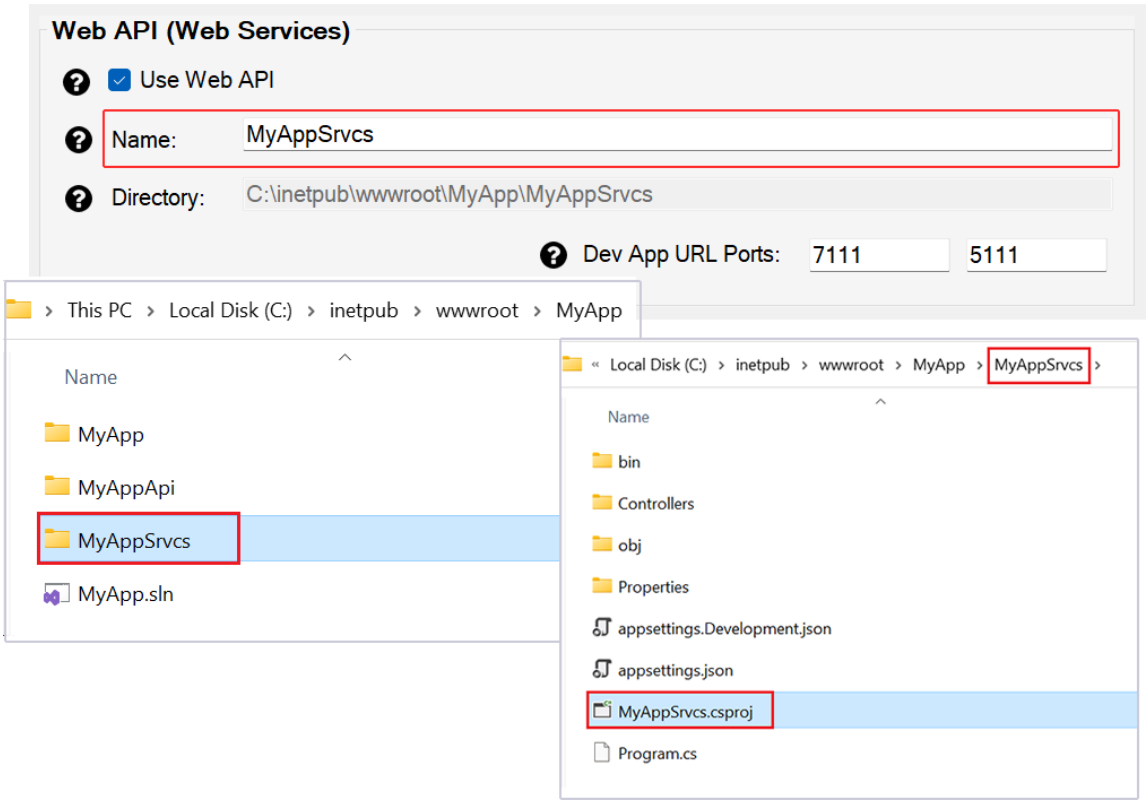
4.1 USE WEB API

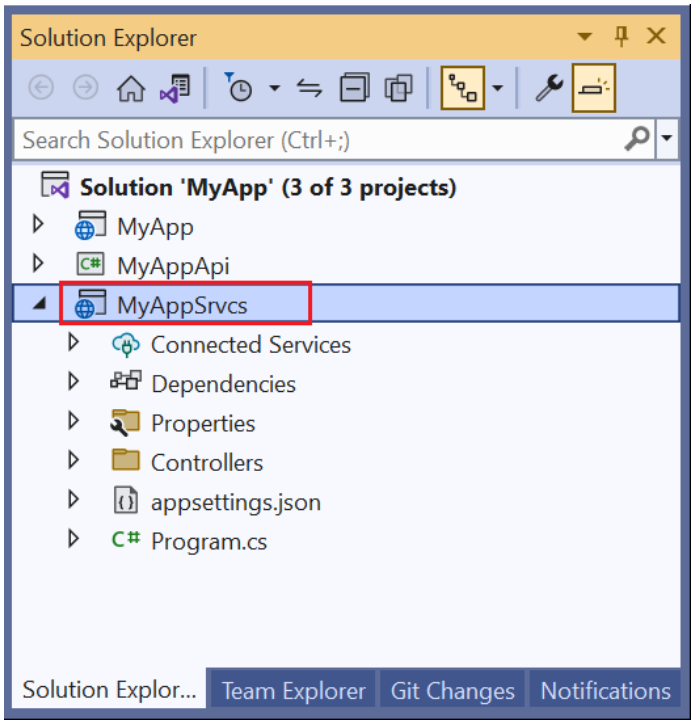
Generates the optional *Web API* project when checked.



4.2 NAME

The *Name* of the *Web API Project (.csproj – ASP.NET Core MVC API Project)* and the project's *Folder* that will be generated.

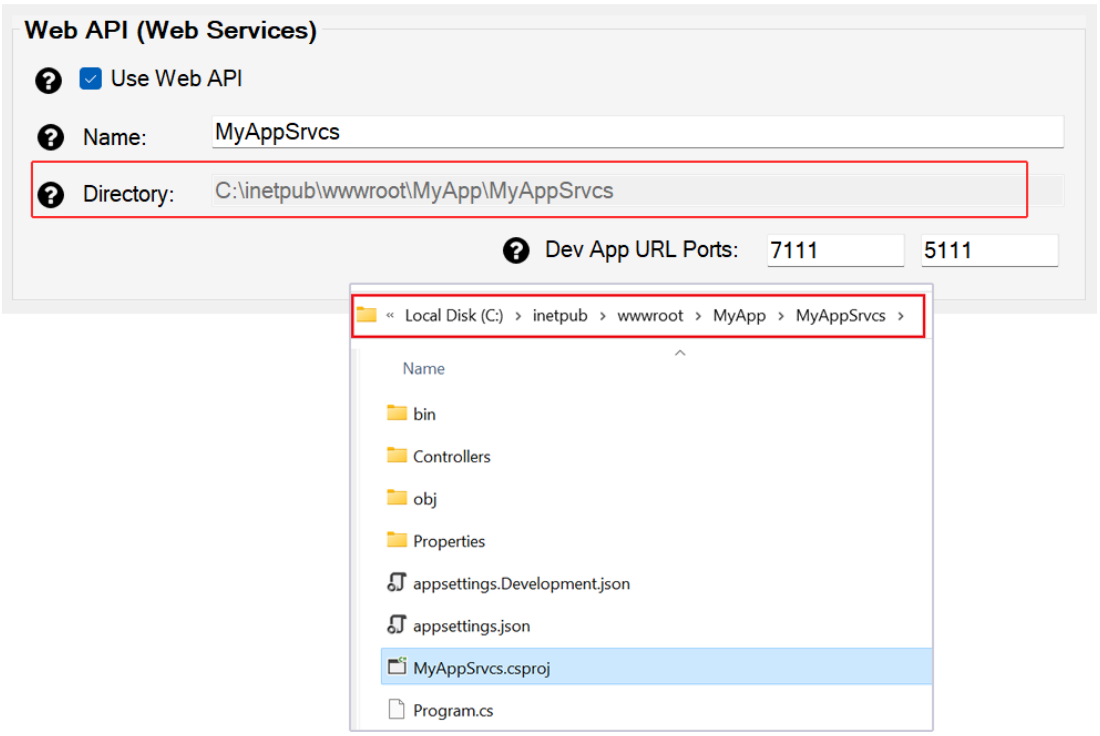




Web API (Project) Name – Visual Studio

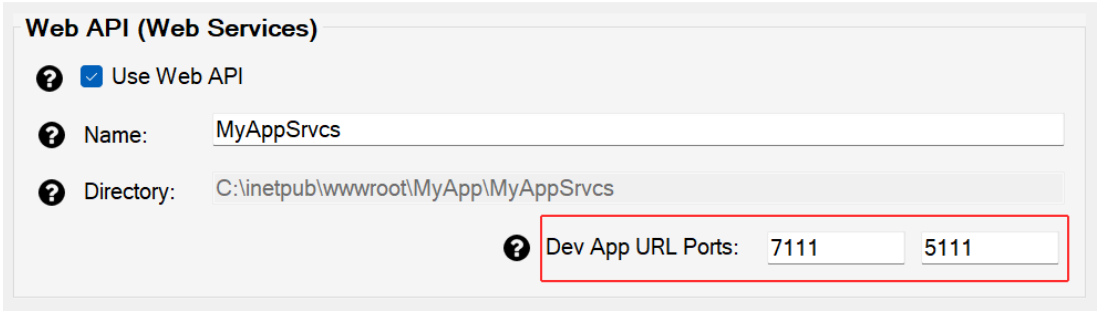
4.3 DIRECTORY

The *Directory* or *Folder* where the *Web API Project* will be generated in. The *Directory* is automatically filled relative to the *Web Application's Directory* and placed in the same-named *Directory* as the Web API *Name*.



4.4 DEV APP URL PORTS

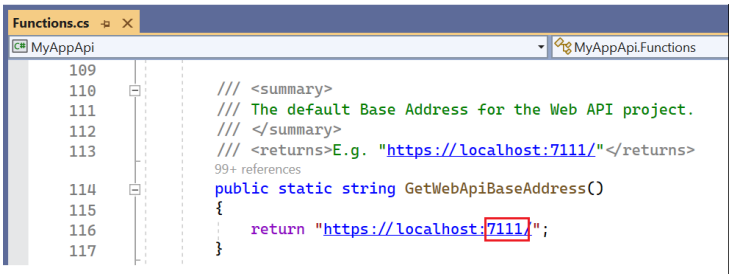
The *Dev App Url Ports* are used during development only. These settings can be found in the *launchSettings.json* file under the *Properties* folder in the Web API project. When both the Web App and Web API projects are ran as *Multiple Projects* in Visual Studio, the web app will run using its respective Dev App Url Ports, and that same behavior happens with the Web API project.



4.4.1 Accessing Web API From A Client

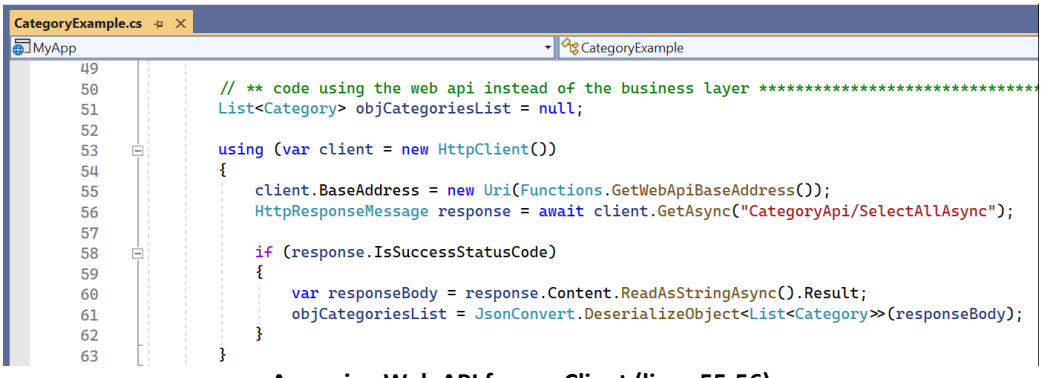
The Web API can be accessed by the Web App (or any client) using the Web API project's web address, which includes the *Dev App URL Ports*. The Web API's base web address can be found in 2 places:

1. **Functions Class**, a helper class from the Web App project. The *GetWebApiWebAddress* method can be called from a client (like the web app)
2. **launchSettings.json** under the Properties folder in the Web API project.



Functions.cs (Web App Helper Class)

launchSettings.json in the Web API Project



Accessing Web API from a Client (lines 55-56)

* The *Selected Tables* tab will only automatically open when the *Automatically Open Selected Tables or Selected Views Tab check box* is *Checked*, which is the default.

You can read end-to-end tutorials on more subjects on using AspCoreGen 6.0 MVC 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 6.0 MVC Professional.

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

End of tutorial.