Documentation for Web Application Step By Step Process

Company : Zuse Systems

Applicant : Shivek Dhurumpall

Summary : I had used MSSMS to create the table responsible for storing the data, the 3 columns I had created where ID , Description and Completion. I had used Visual Studio 2019 where I created 2 projects, The Api which I named zusesys and my MVC Project. The languages that I used to create this application where C#, Javascript, SQL and HTML.

1. Step 1 : Database Requirements:

Creating the database on my local computer, I had created a table called Tasks, The ID column was set as my primary key and I had set the Identity specification to Yes. This will allow the database to automatically increment the ID value, without me manually adding it.

A screenshot of a computer

Description automatically generated

I added a Description column with the datatype of varchar(50), My completed column was set as an integer, with the default value set to 0, I made this way to correspond with the check and uncheck of the check box, 1 meaning Yes when the check box is clicked and 0 meaning No when the check box is unchecked.

A screenshot of a computer

Description automatically generated

Successful creation of the database table named “tasks”.

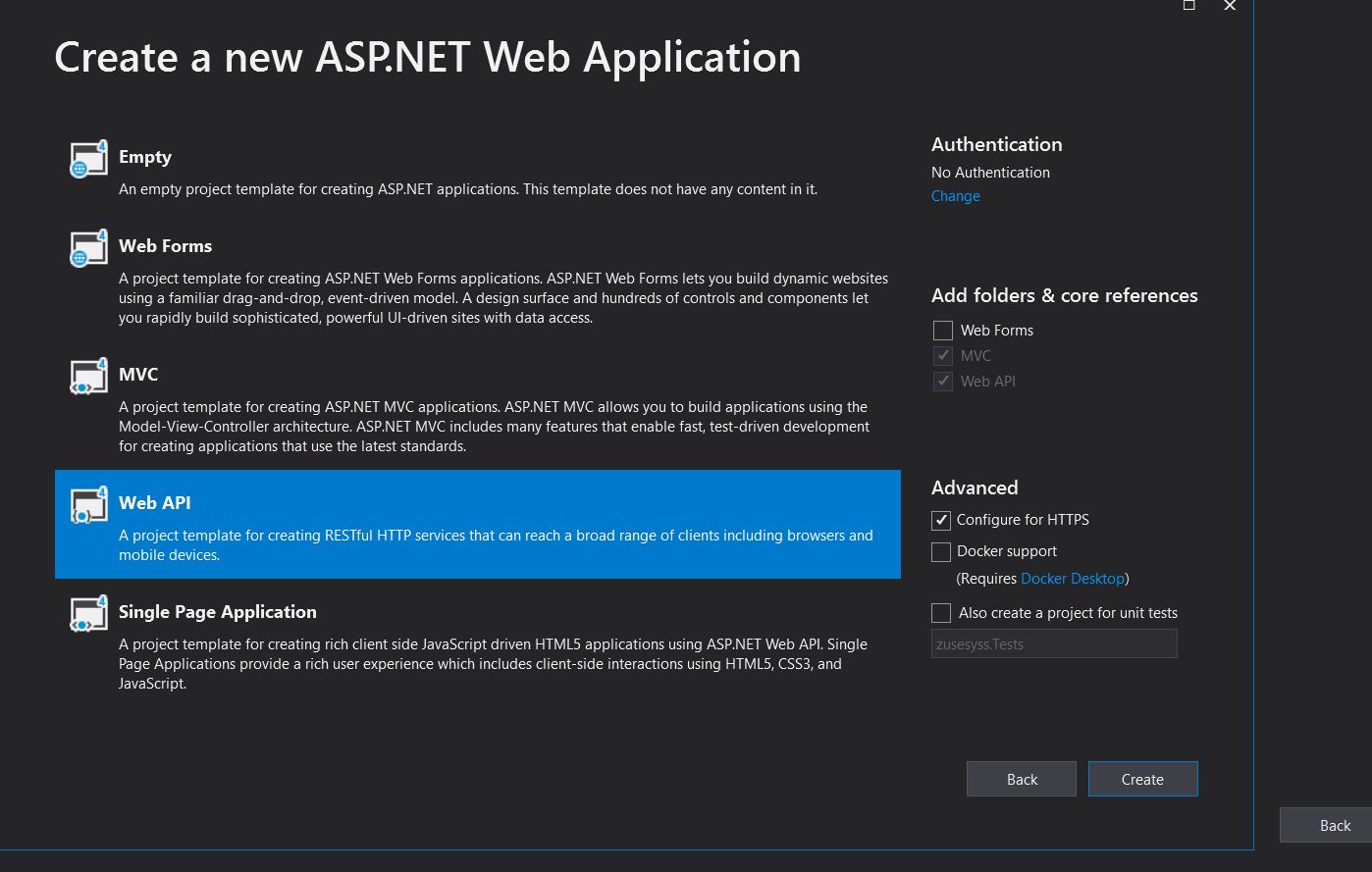
A computer screen with a blue box

Description automatically generated

Step 2 : Backend (Server) Requirements:

I had created 2 projects within the application, The API and the MVC.

First, I am creating the API Project, with the default values.



I had named the API Project zusesys

A screenshot of a computer

Description automatically generated

Once the Api Project has been created, I needed to connect my database to the API, I had right clicked on the Models Folder to add a new Item, I named the Model DBModels.

I followed the prompts, Here is where I connected the SQL server and the database for the generation of the connection string in the web config and generation of the Models files

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Once the database connection has been created in the API, Files are generated in the model folder, such as tt files and .cs files. After this process it is important to build the solution.

A black screen with white text

Description automatically generated

Now that I have made our model for the API, I would now need to create controllers, this is necessary for API end point specification that we would need to connect to our MVC Project.

These end points include the POST, PUT, GET And DELETE Functions.

I had Created a controller called “TaskController” with actions using entity frame work, this had autogenerated a template controller with those CRUD Functions in connection with the Model

A screenshot of a computer

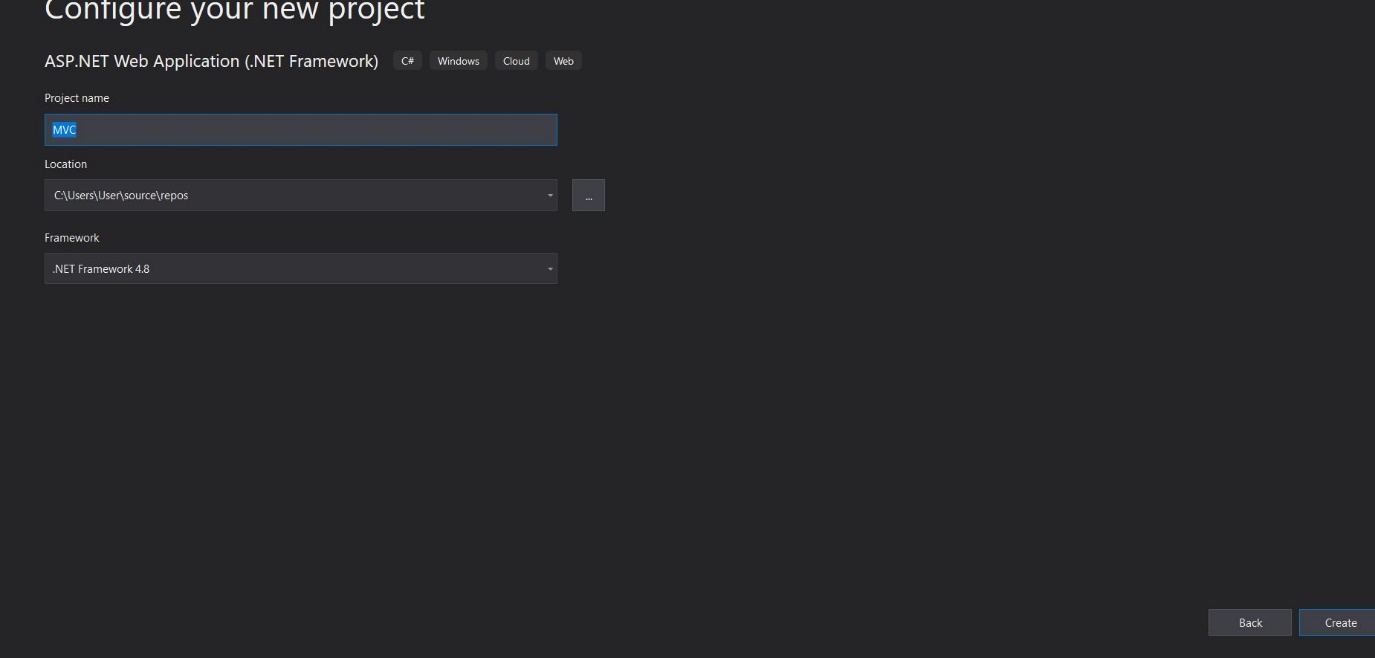
Description automatically generated

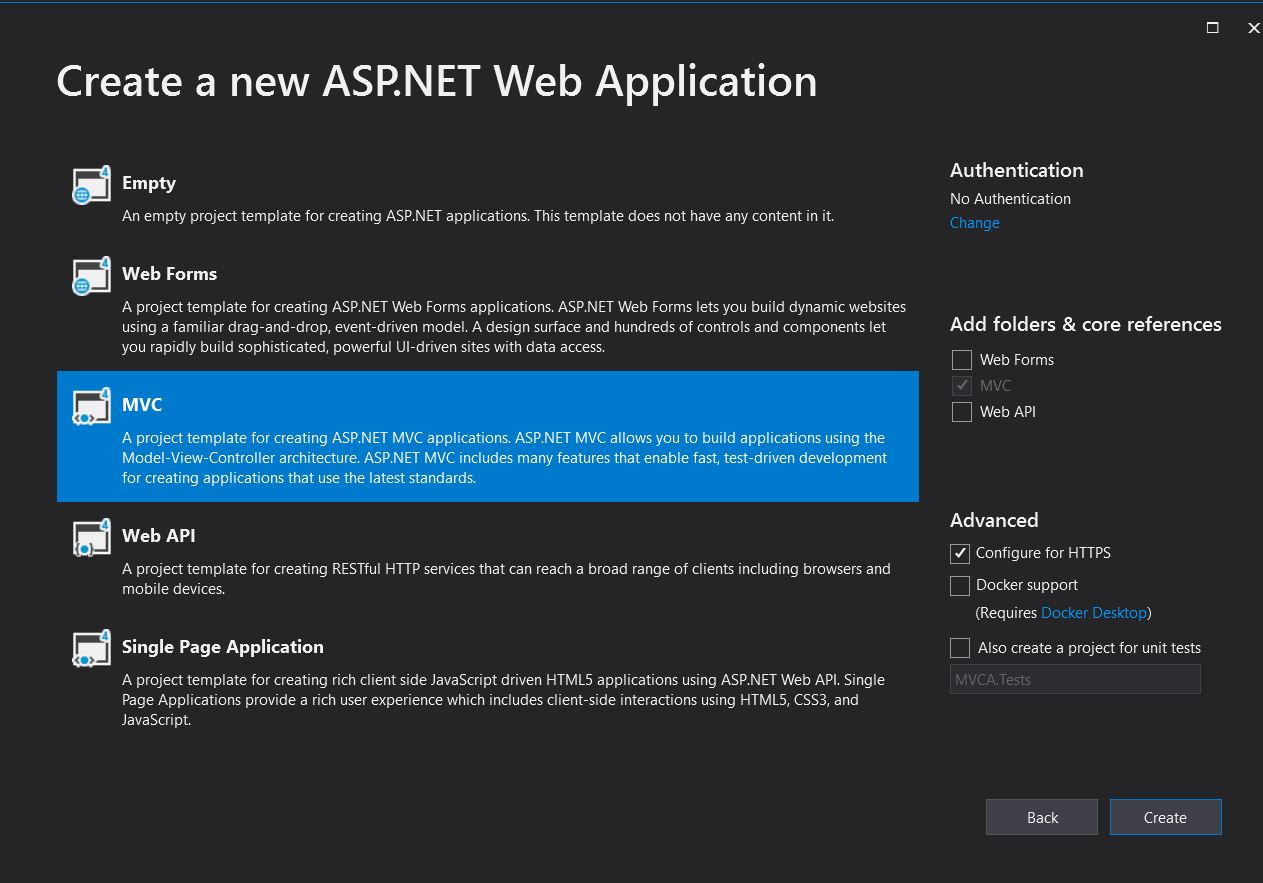
A screenshot of a computer

Description automatically generated

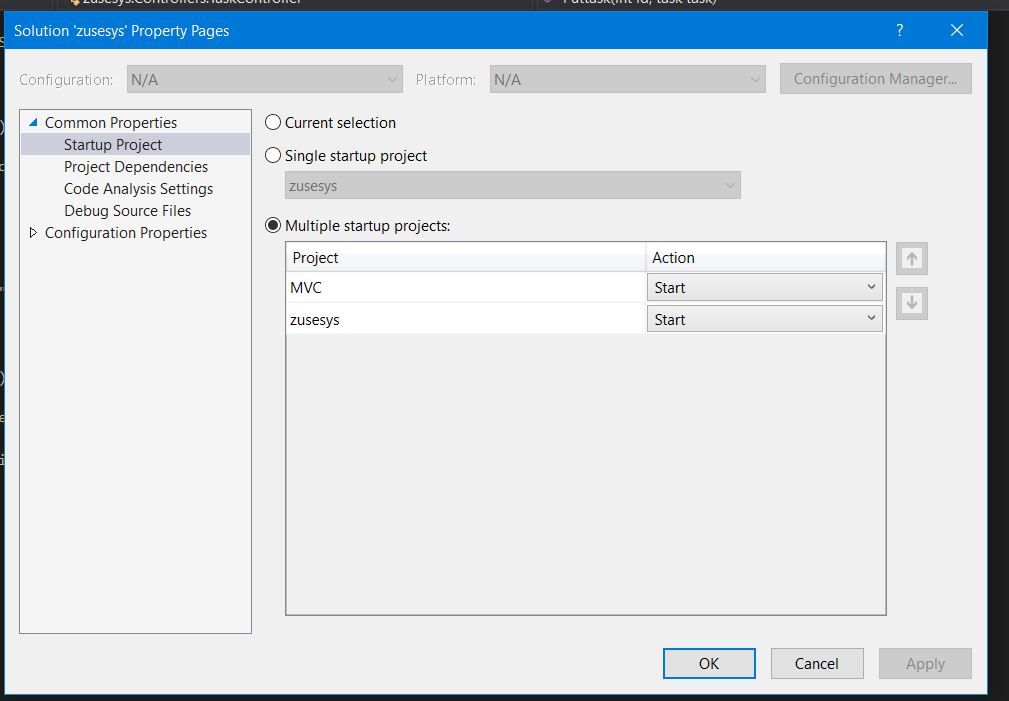
Now that our API Controller has been created, I am going to create my MVC Project, this will allow me to retrieve the Data from the API from the database, as well as push data from the API to the Database. The MVC will also allow me to create my front end to view the list of tasks.

I created a MVC projected called MVC.

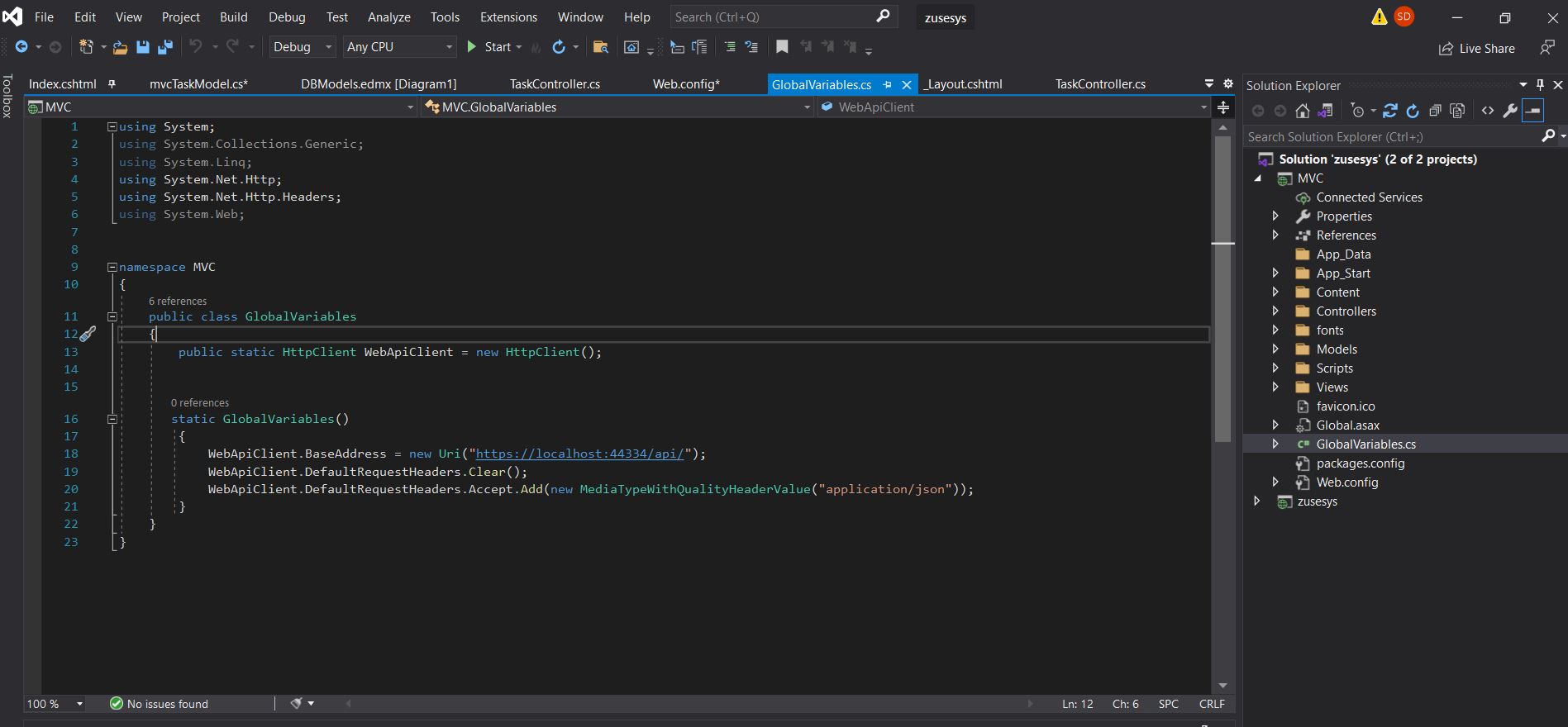


I had used the default settings.

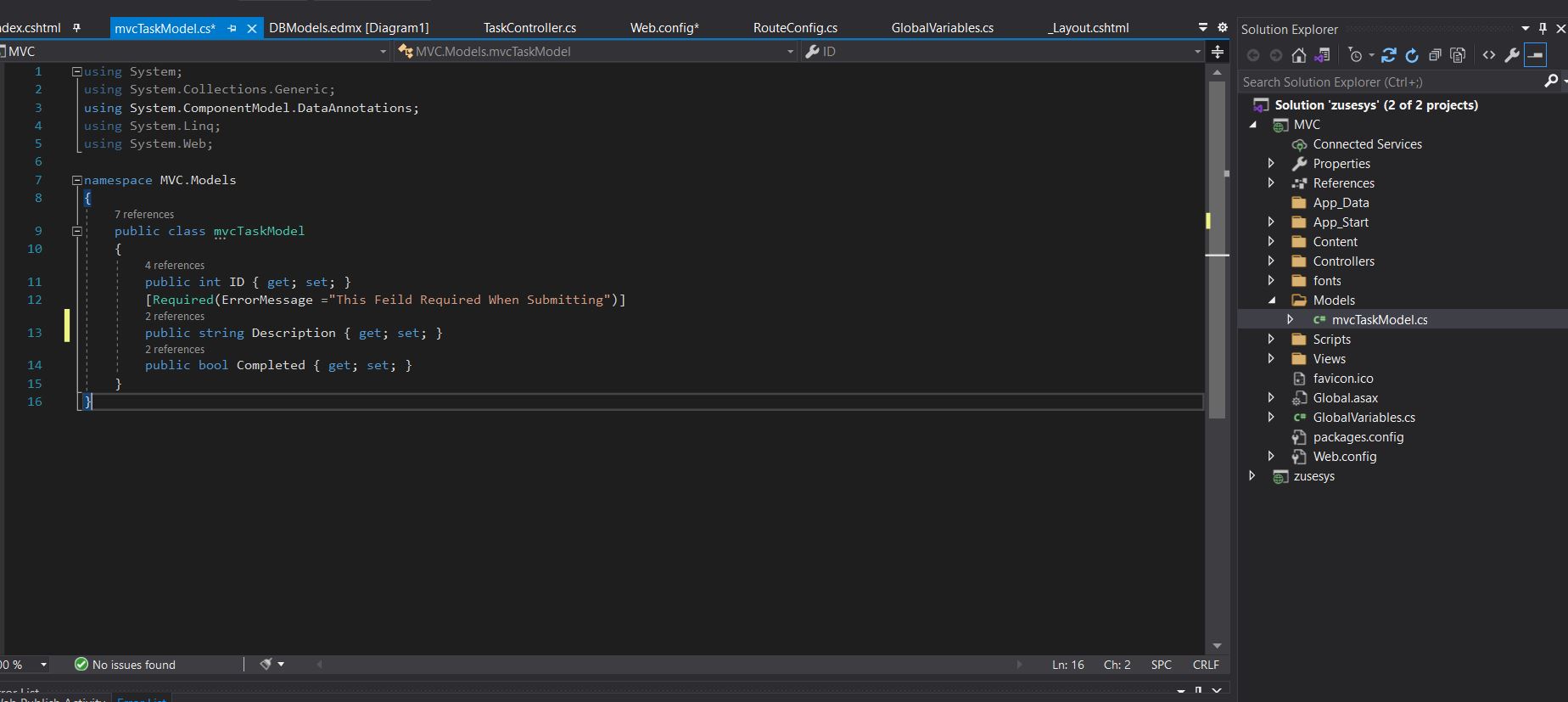
Once both my Projects where created, I set the API and the MVC to both run when the solution is executed, This is because we would need the API to run to allow database connection to the frontend.



I had created a class withing the MVC Project called GlobalVariables, I created this to act as a bridge between the MVC and the API, This is where I specified the API localhost address together with the API end point, that would be called as the base address to retrieve information.

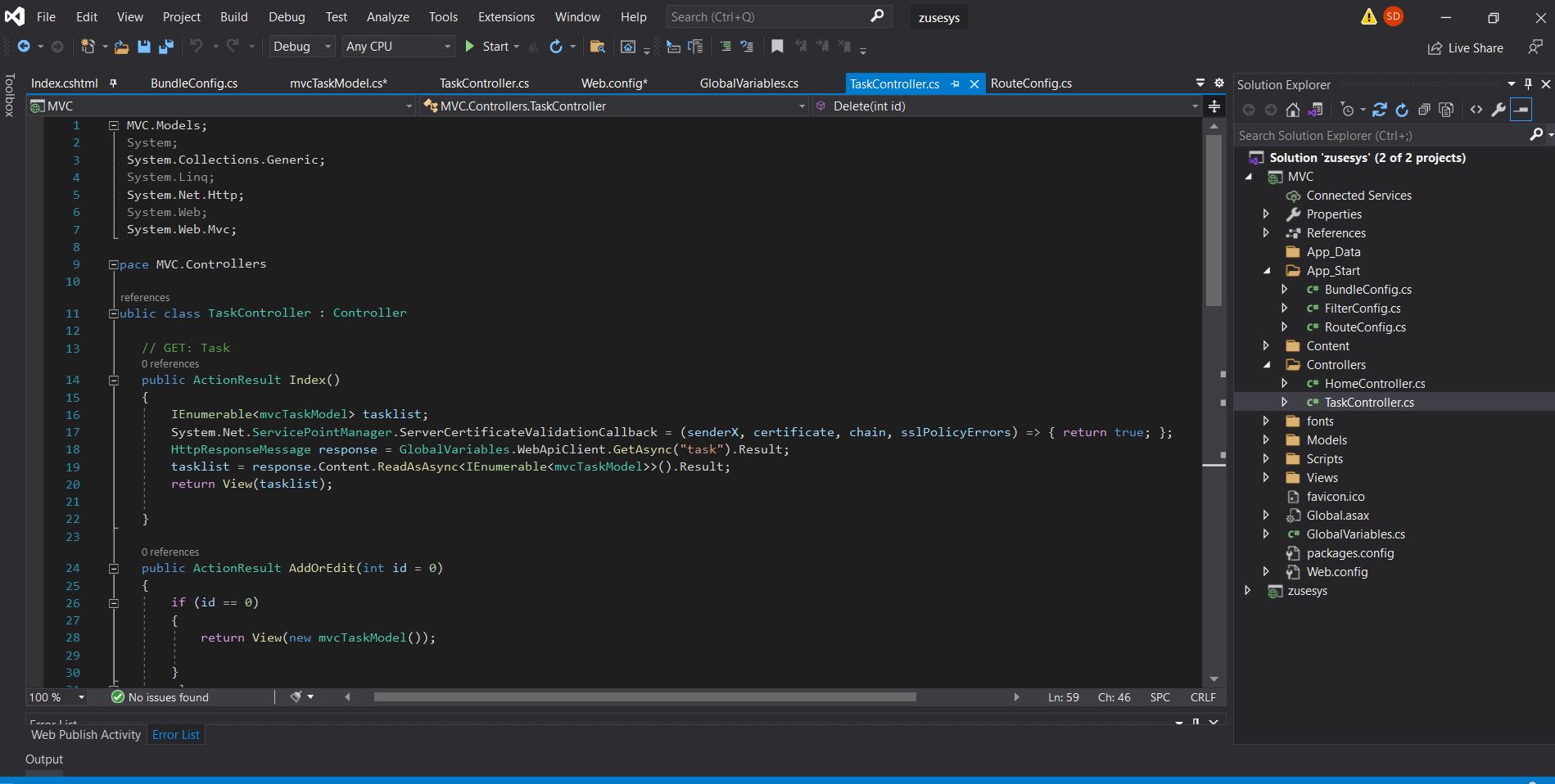


Within the MVC Project, I had a model class within the models folder, with the necessary variables that I need.

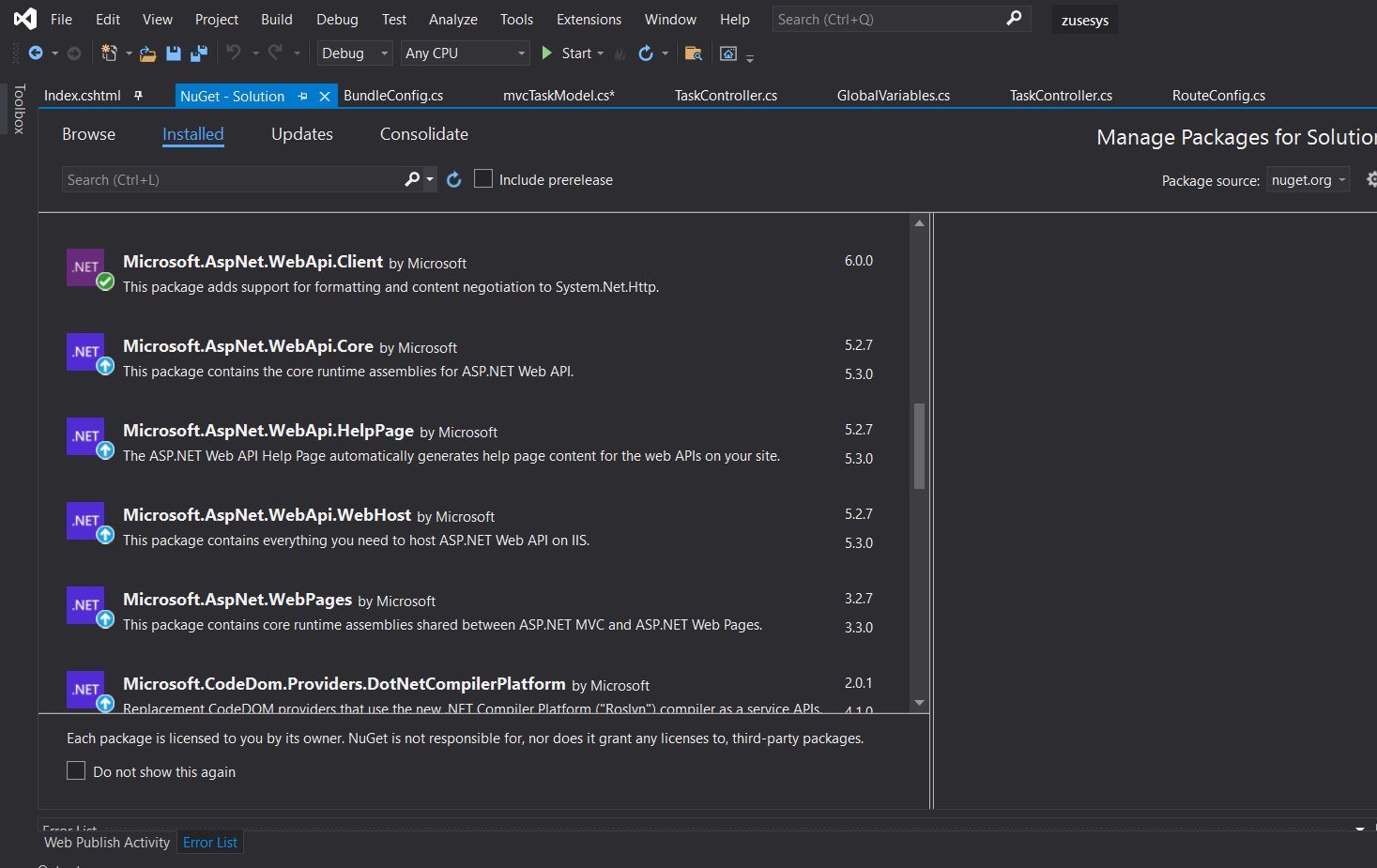


There after I had created a new controller to display the list of tasks on to the front end, this controller will be responsible for the Edit,Update , Delete and List functions.

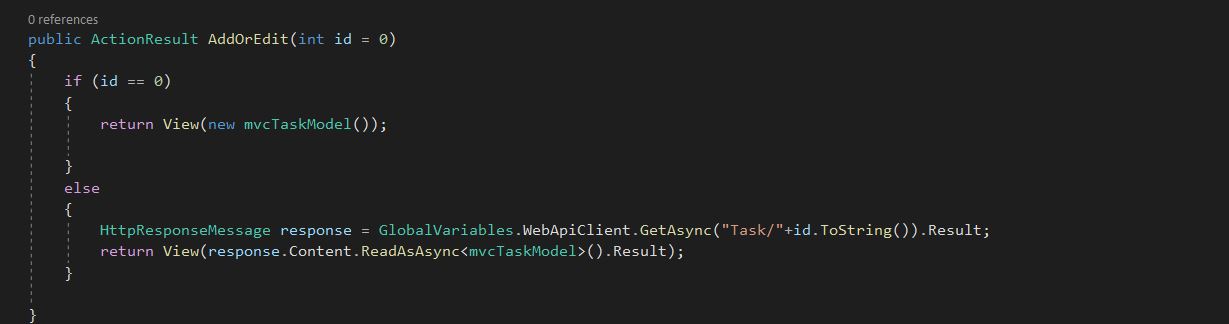
I had created a Get method used to list the tasks, by connecting to the to webApiClient requesting a GET function for the task controller, that is then returned to be shown on a view.



Before running to Test if the list will show, I needed to download the appropriate Nuget packages, the Main one being is the Microsoft.AspNet.WebApiClient Nuget package.

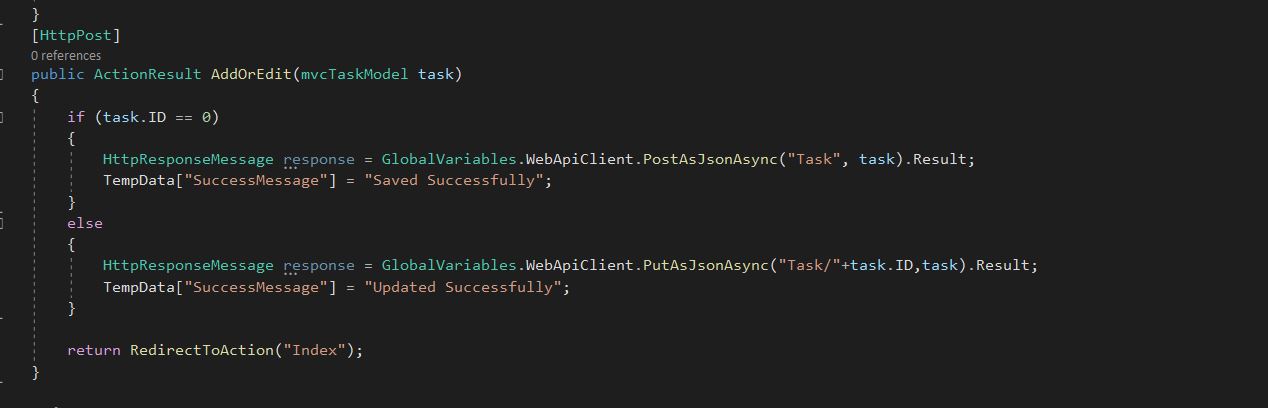


I created an IF statement, since the id variable created is always equal to 0, this will force the code to run the else statement, where it is calling A GET request when the user wishes to create or edit a task.

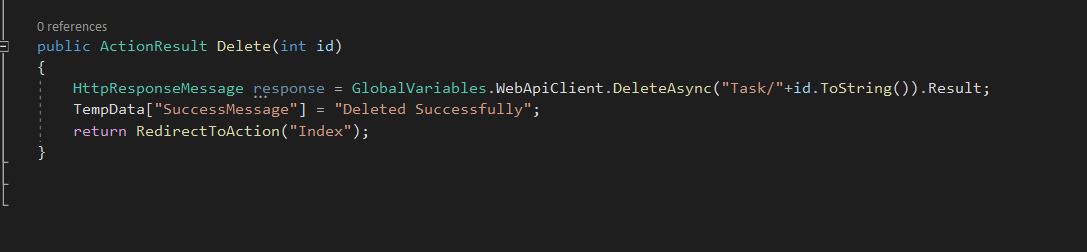


Here is an HTTPPost Request being made, I had created a HTTPPost methods, where if the user had to press “Create task” the system will direct to the IF statement as a POST function towards the WebAPIClient, to the API’s taskcontroller.

Or if the task ID is not 0 (which means its a already created task) , it will go to the PUT function where the task can be updated, with that specific task ID specified.

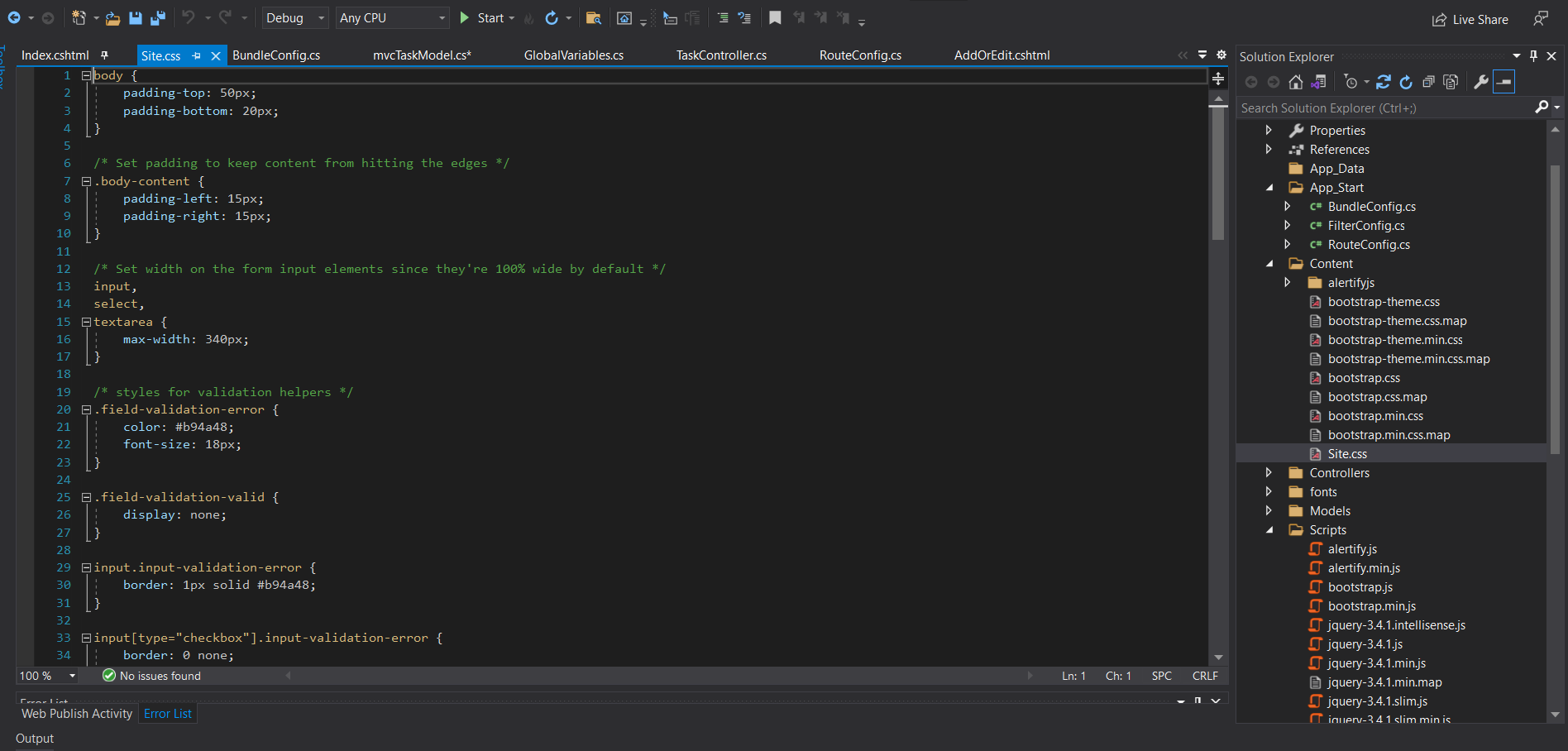


Delete Method , same as PUT and POST, but it would be calling a Delete function in our API.



Step 3 : Frontend (Web) Requirements:

Within the Content folder in the MVC Project, there is a css file called Site.css, I edited the file to make it more user friendly to the user.



I had then downloaded Alertify JS Nuget Package, this will allow me to show user friendly prompts to users when an execution has been made.

A screenshot of a computer

Description automatically generated

Within the Shared Layout folder I had added the JavaScript for the Alertify Nuget package

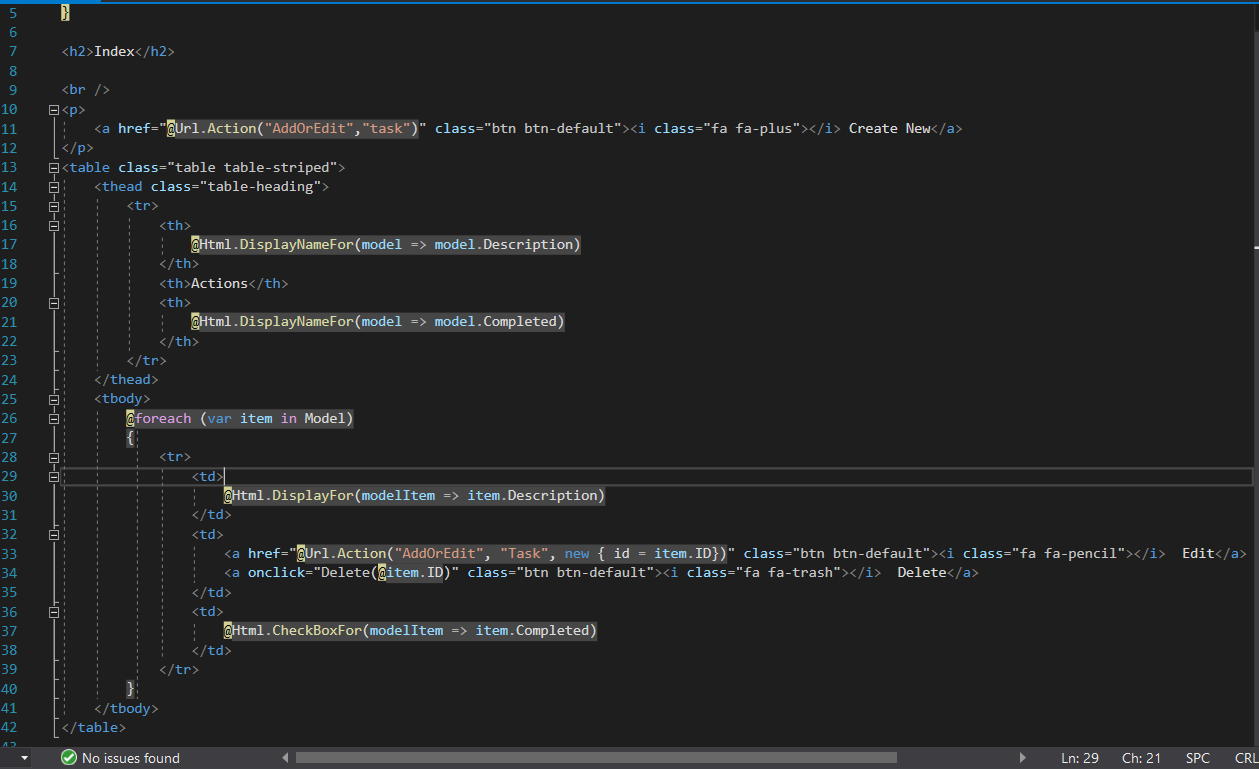
A screenshot of a computer

Description automatically generated

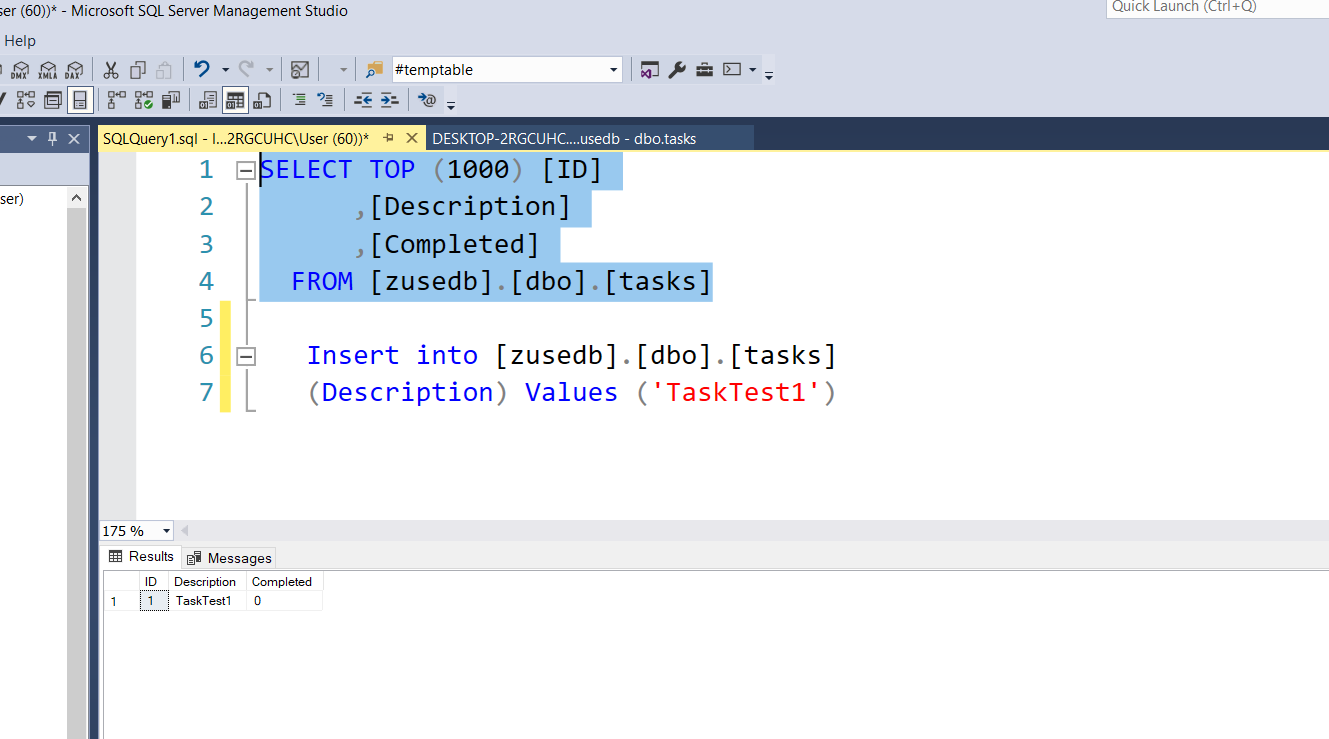
A screenshot of a computer

Description automatically generated

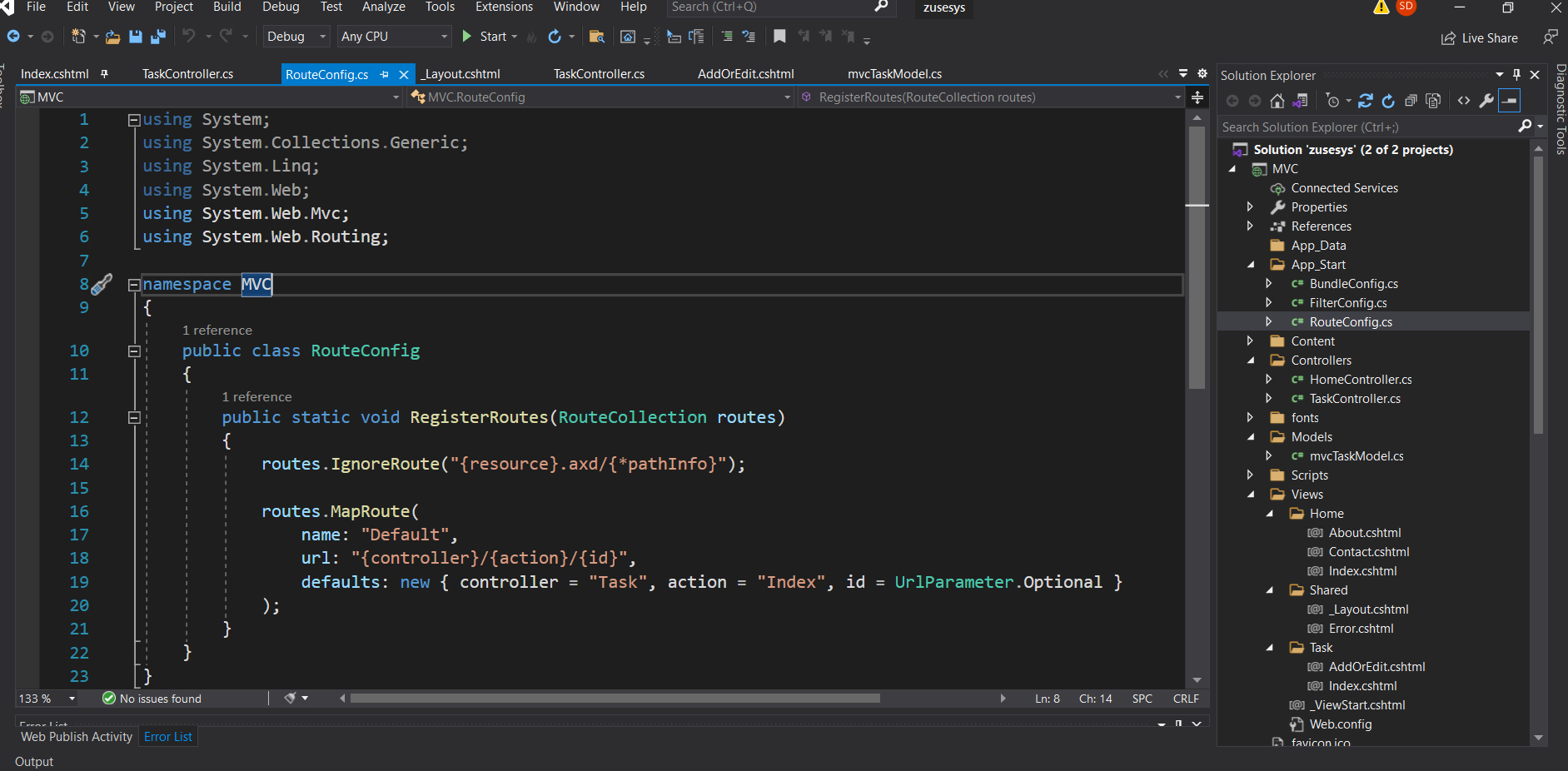
There after, I needed to display this information on view, I had created an Indez.cshtml, where I specified my button events for the create and edit method , aswell as made the layout to display my Tasks as list on a table format, and the checkbox.



I added Test Data on SQL, To check the connectivity from my Database to API to MVC



Within the RoutConfig, I had set the default start up controller as my task controller and index. This will allow the system to run the controller I want to run for the MVC project upon start up.



I ran the project, it is important to note that the API must be running when trying to perform functions on the front end, as it works as a middleman between the frontend to the database. This is why I set the API and MVC to both Run, when the solution is ran.

Below shows the test data successfully showing on the frontend.

A screenshot of a computer

Description automatically generated

Now Lets Test the data, I had pressed the delete button, due to alertify Nuget package a user friendly message had been shown, this will also show for updating and creating.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Check Box Functionality added in the Index.chhtml

Each checkbox is given a class completed-checkbox and a data-id attribute containing the task ID. A JavaScript function handles the checkbox change event to determine whether the checkbox is checked or unchecked. An AJAX POST request is sent to update the task's Completed status on the server. And will update the completed variable in the model to 1 or 0.

A computer screen shot of a program

Description automatically generated

For this to work and Update on the database side, I added a POST request in the task controller in the MVC Project.

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated