**ASP.NET MVC Web Application Development in Visual Studio**

**Configurations**:

.NET Framework Version 4.7

MS SQL EXRESS SERVER 2012

Visual Studio Community 2017 Version 15.9.11

**Create a Sample Project (MVC based auto generated code using ADO Data Entity Model and Razor)**

**Step 1**:

* Install Microsoft Visual Studio Community 2017

<https://visualstudio.microsoft.com/vs/community/>

* Install MS SQL EXPRESS SERVER 2012

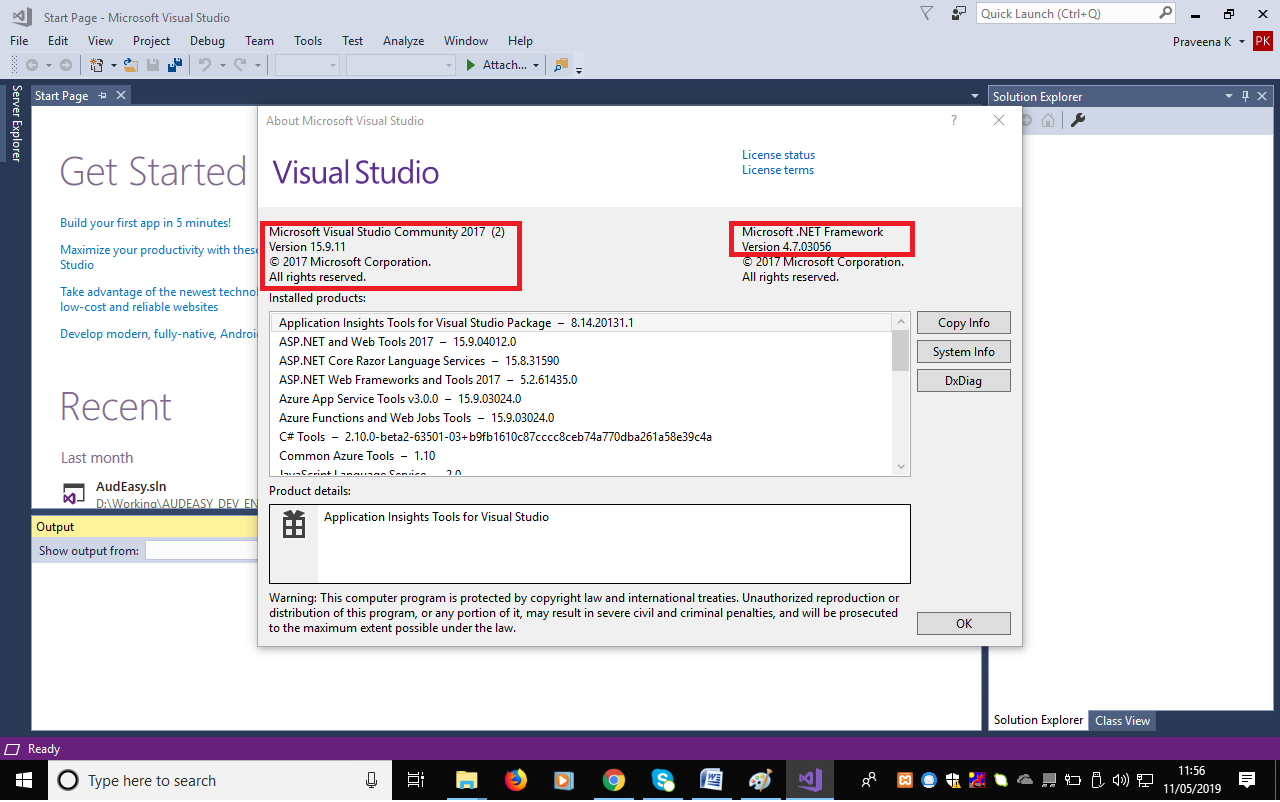
<https://www.microsoft.com/en-in/download/details.aspx?id=29062>

* Install Microsoft SQL SERVER MANAGEMENT STUDIO

<https://www.microsoft.com/en-in/download/details.aspx?id=29062>

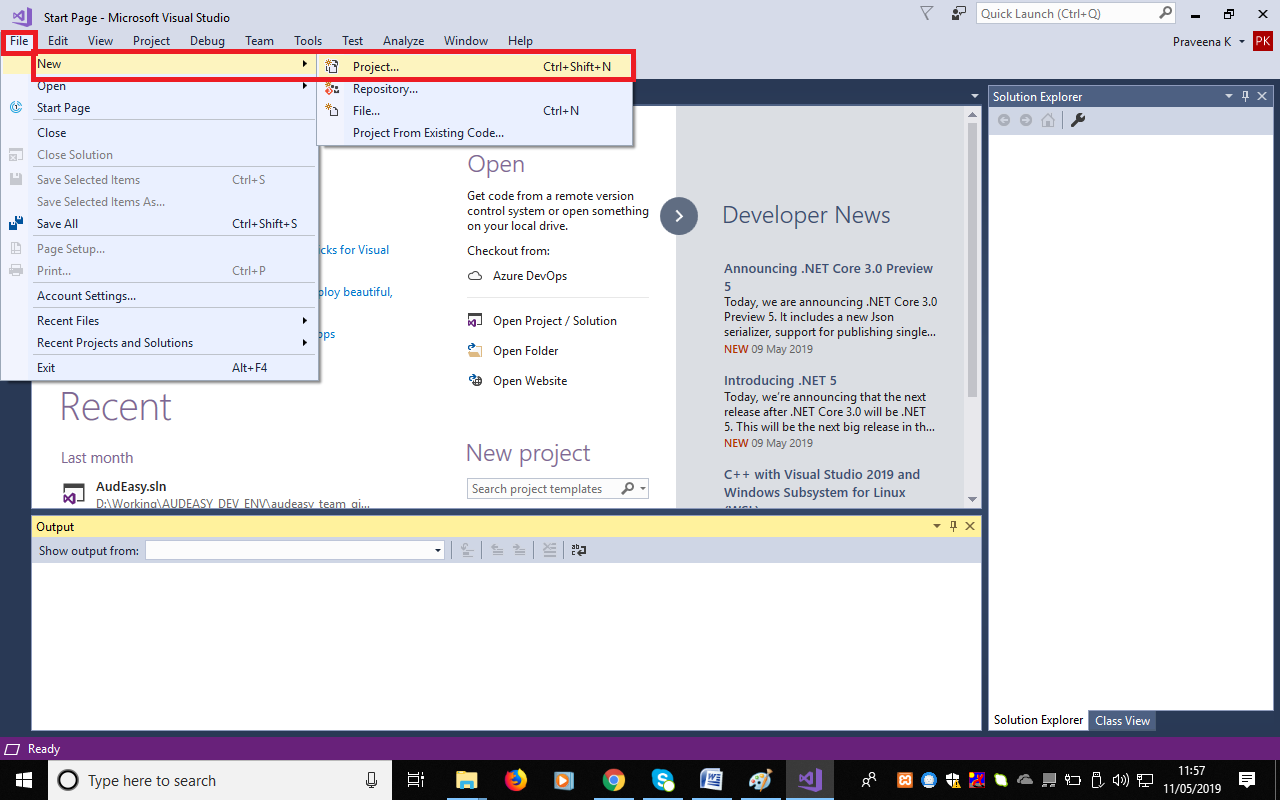
**References:**

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>



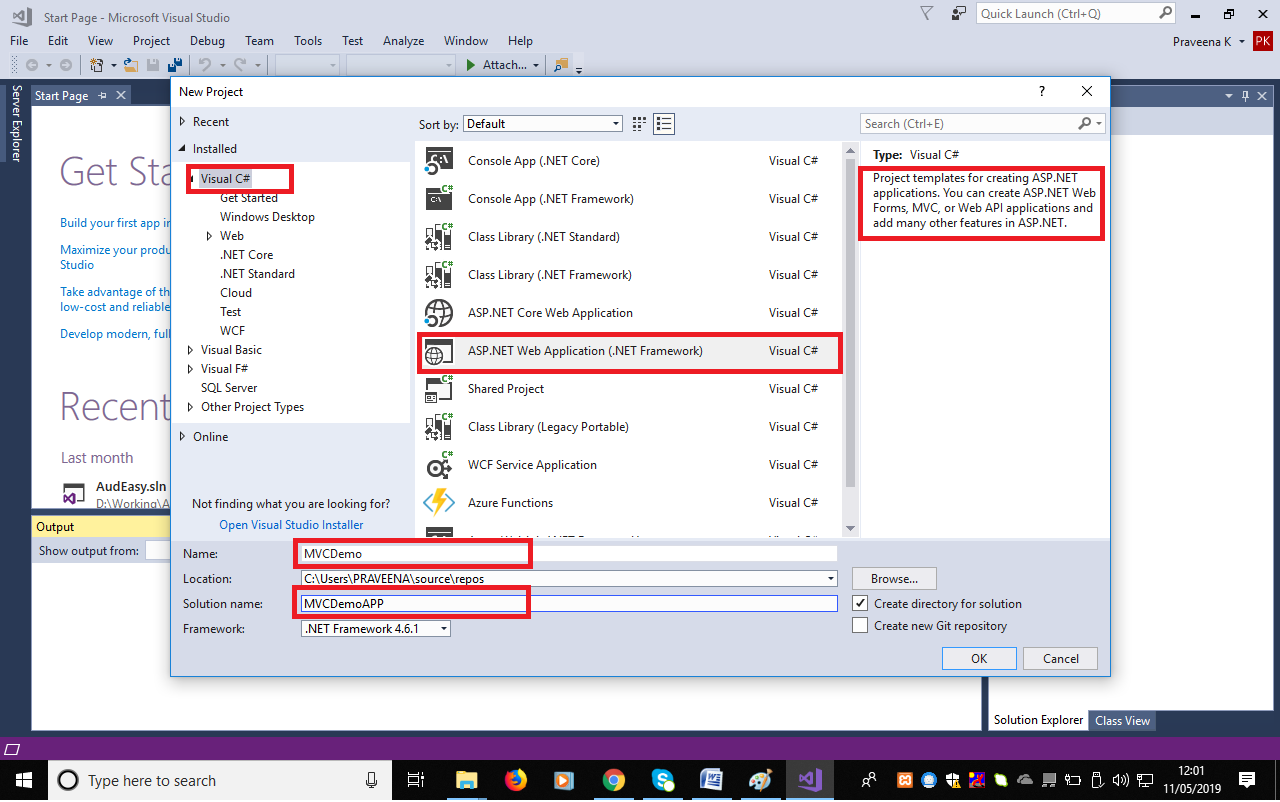
**Step 2:**

After installing Visual Studio Community 2017, double click on the icon to open IDE. Create a new Project as shown in screen shot. From the **File** menu, select **New** and then select **Project**.



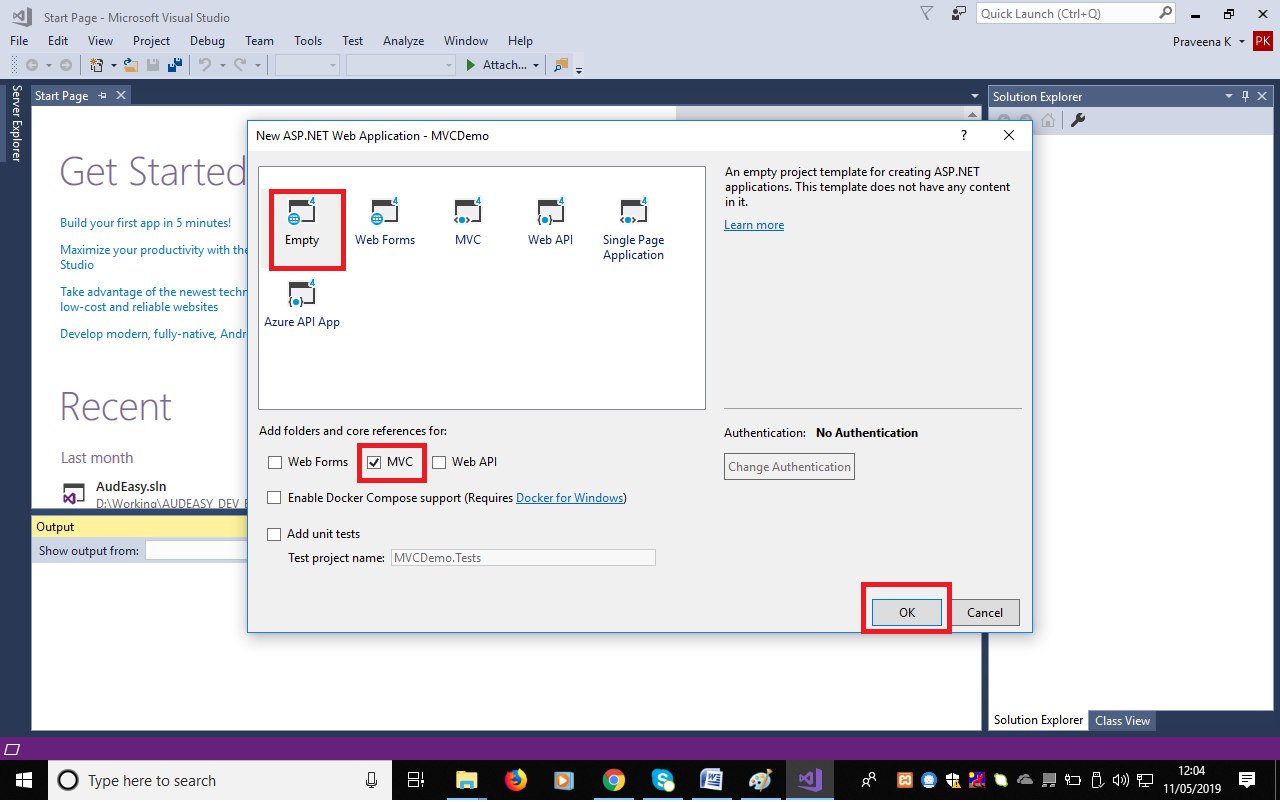
**Step 3:**

Select ASP. NET WEB APPLICATION and then provide the name of the project and solution name. One solution can contain ‘n’ number of projects. Each project can act as a DLL.

****

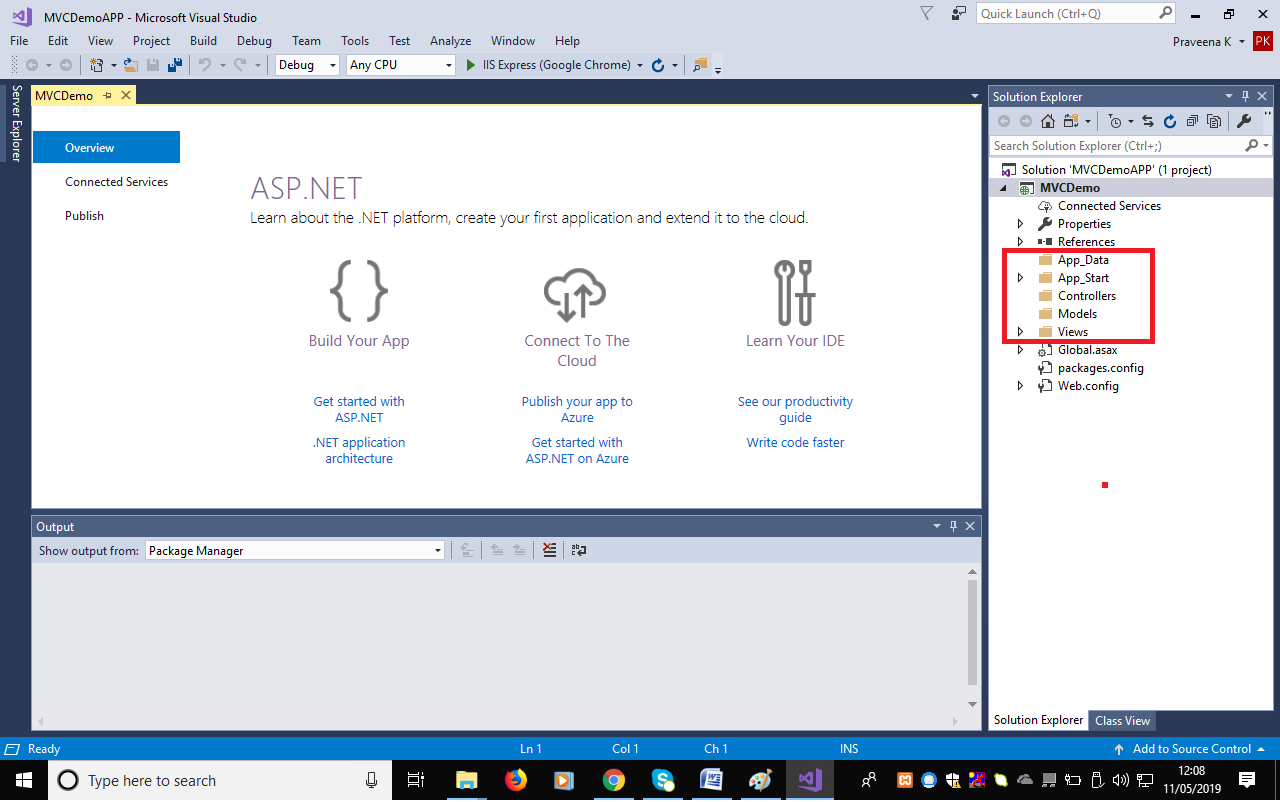
**Step 4:**

Upon clicking on **OK**, make below selections in NEXT screen. Make sure **MVC** is selected. This sample projects is a MVC based application.

****

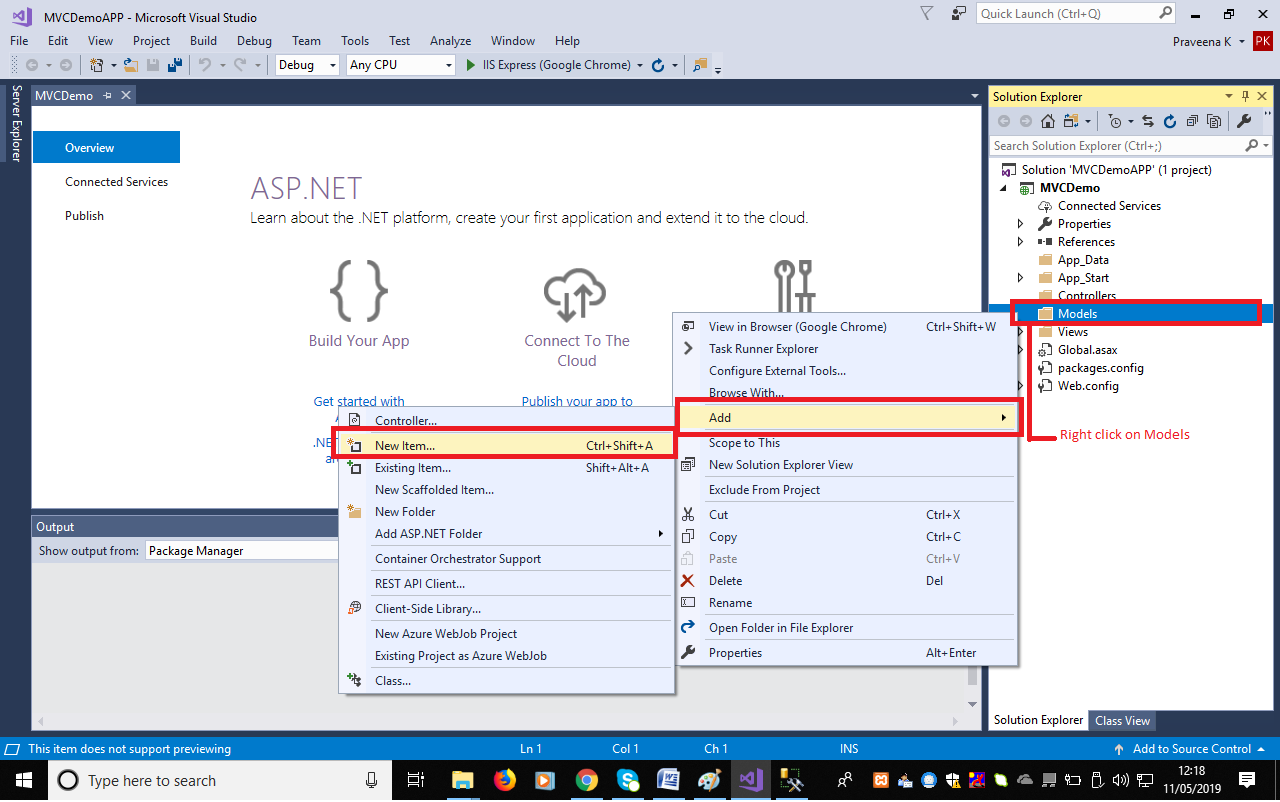
**Step 5:**

Upon clicking on **OK**, notice that the project directories get created for MVC Framework.



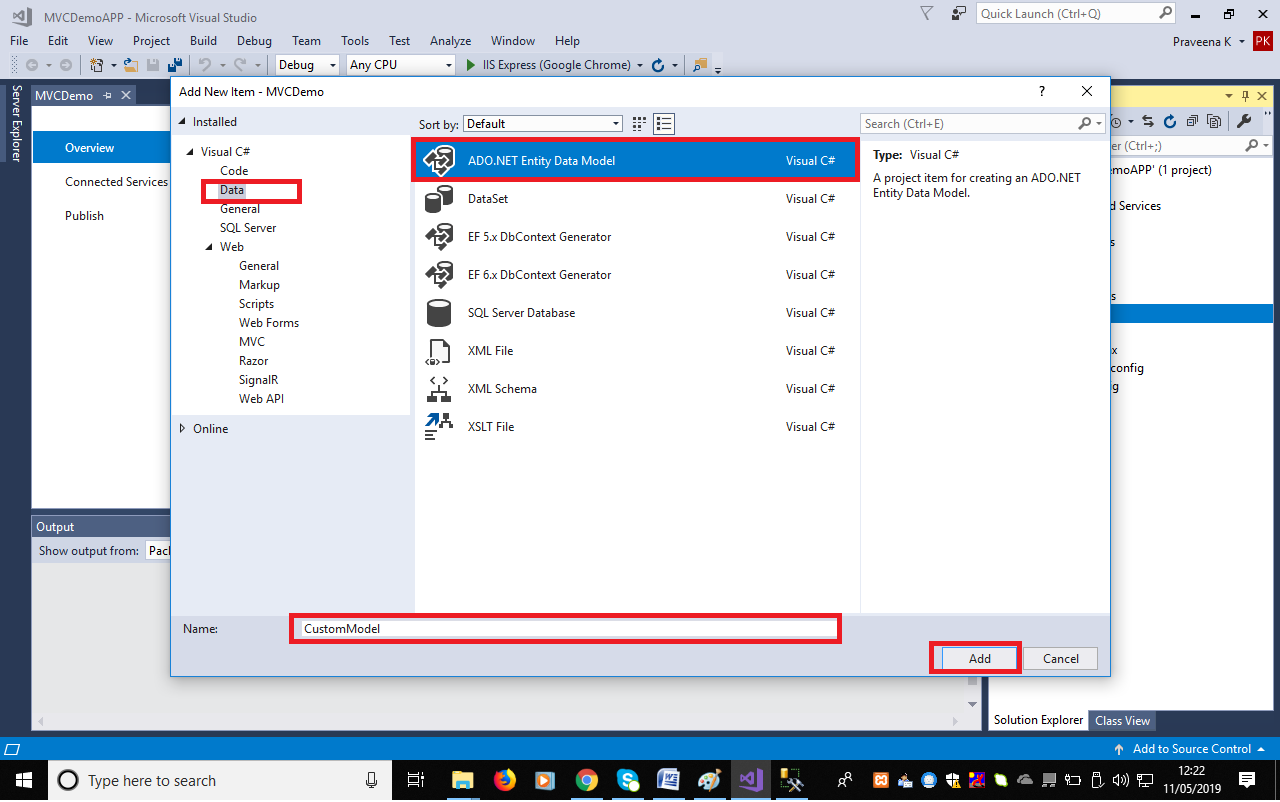
**Step 6:**

As a first step, link the DB server using the IDE ADO.NET Entity Wizard. Create a new Model for doing this.



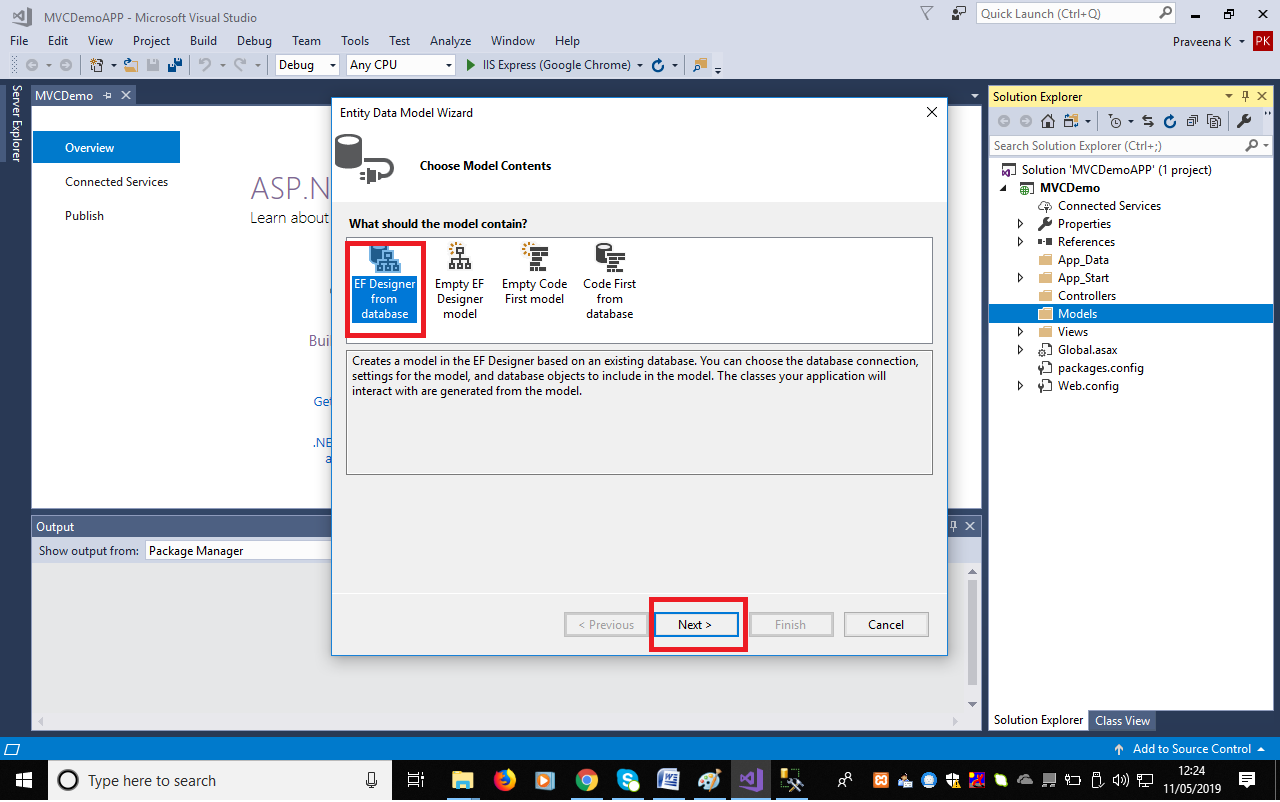
**Step 7:**

Make below selections and mention a user defined name for the model.

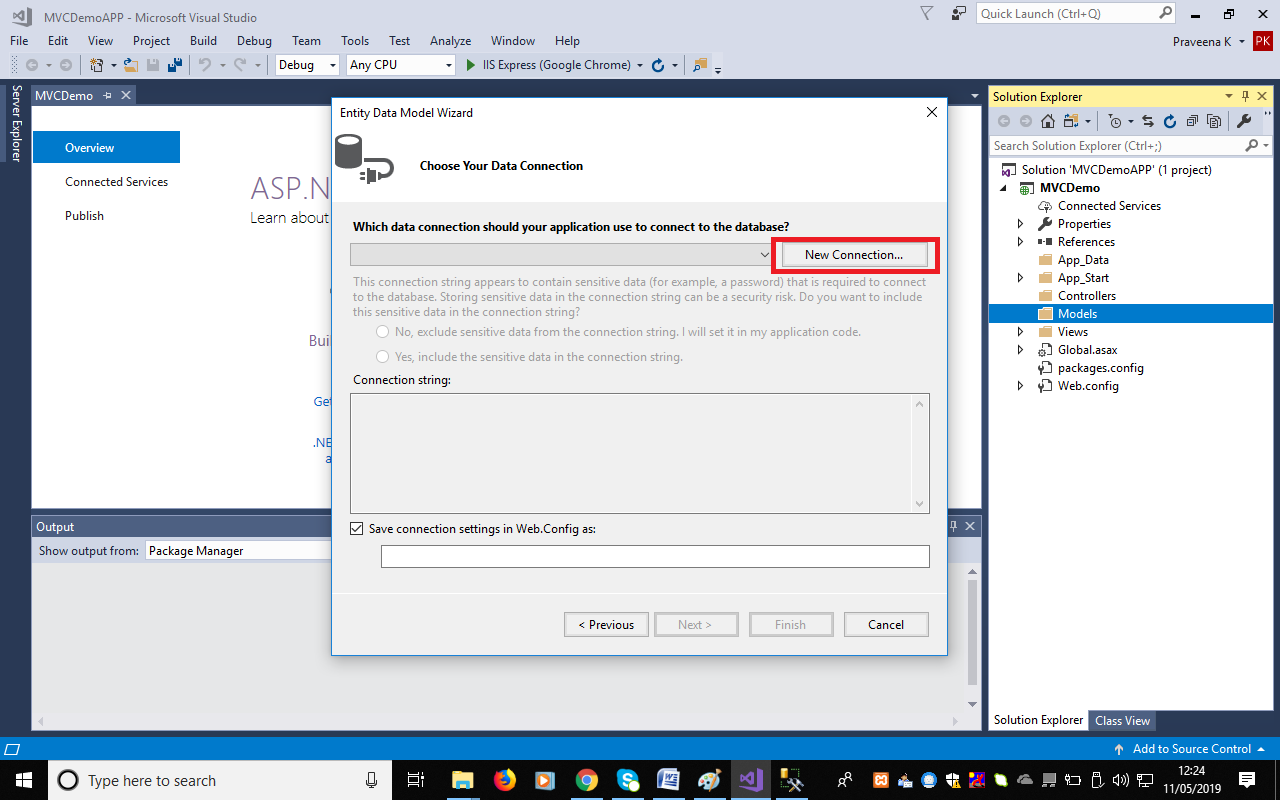


**Step 8:**

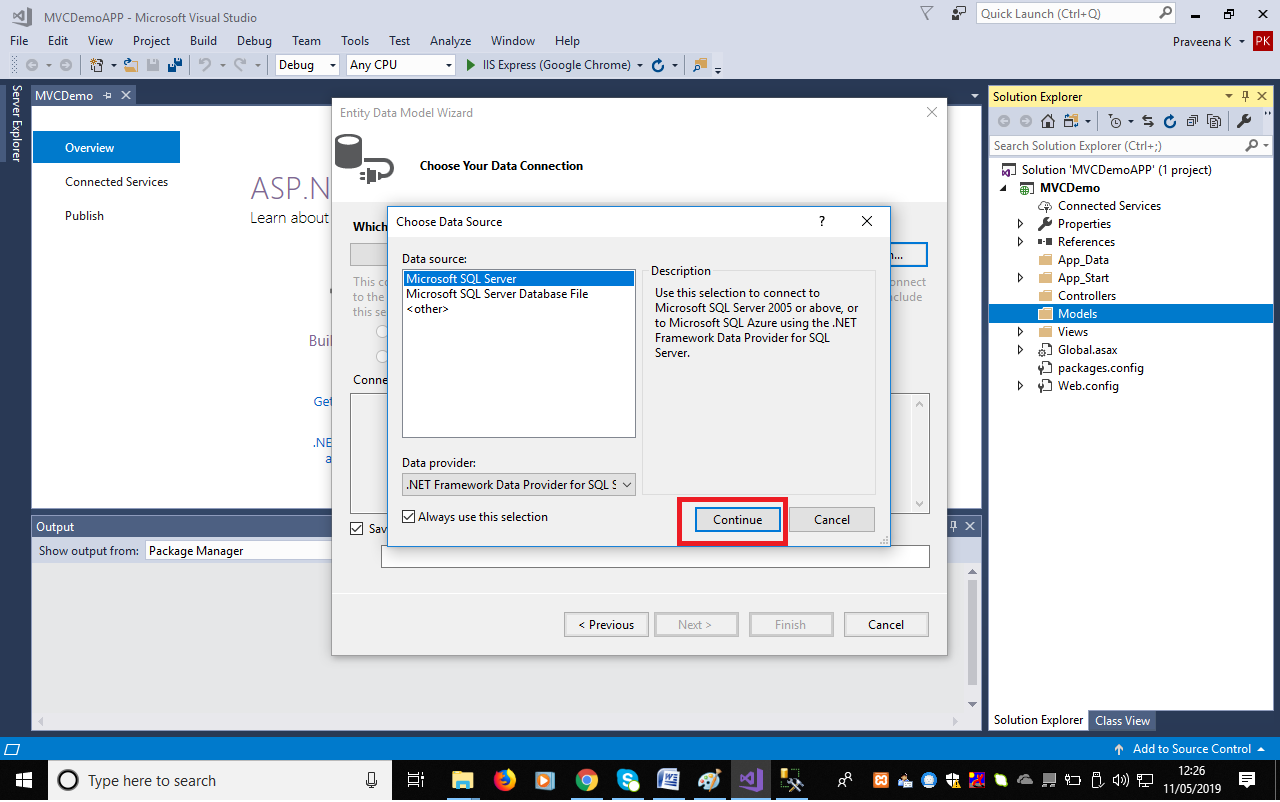
Adding a new model will take to below screen. Move to NEXT screen after making the selection.



**Step 9:**

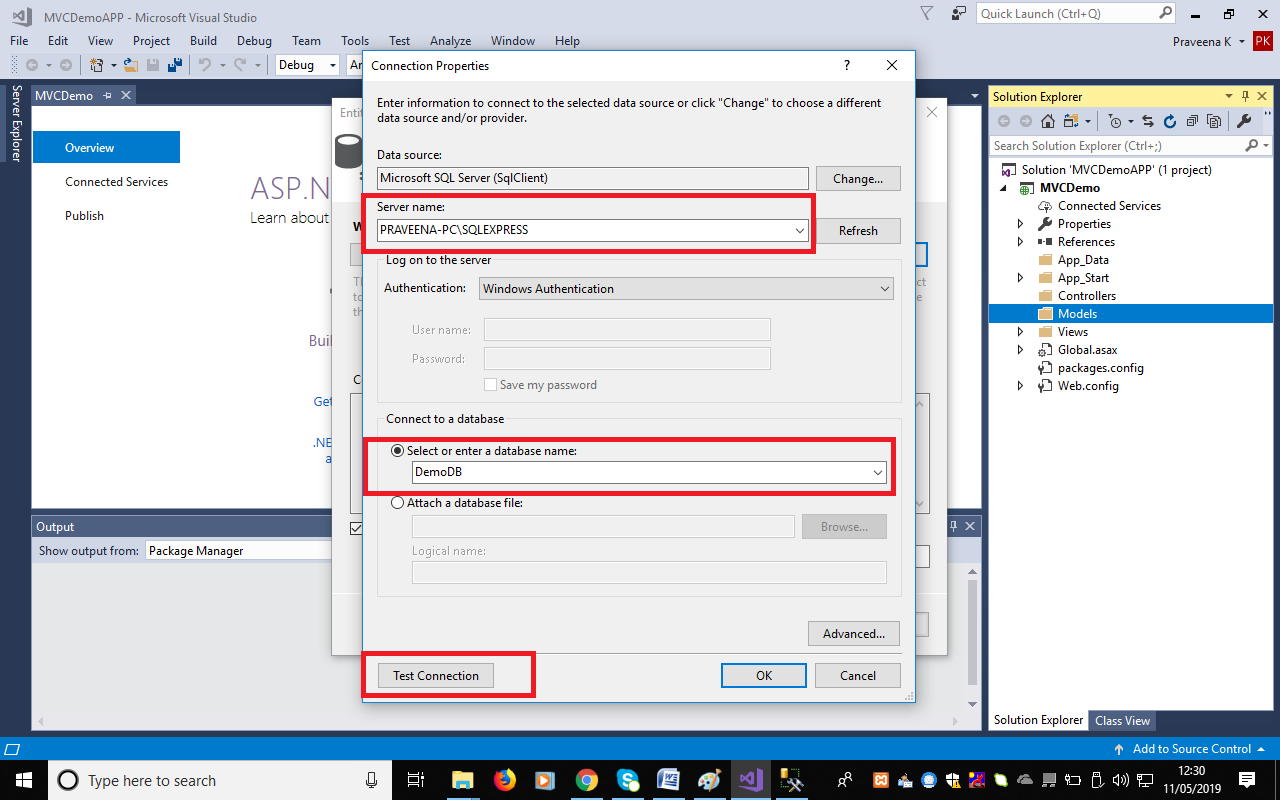


**Step 10:**



**Step 11**: Test MS SQL Server DB Connectivity.

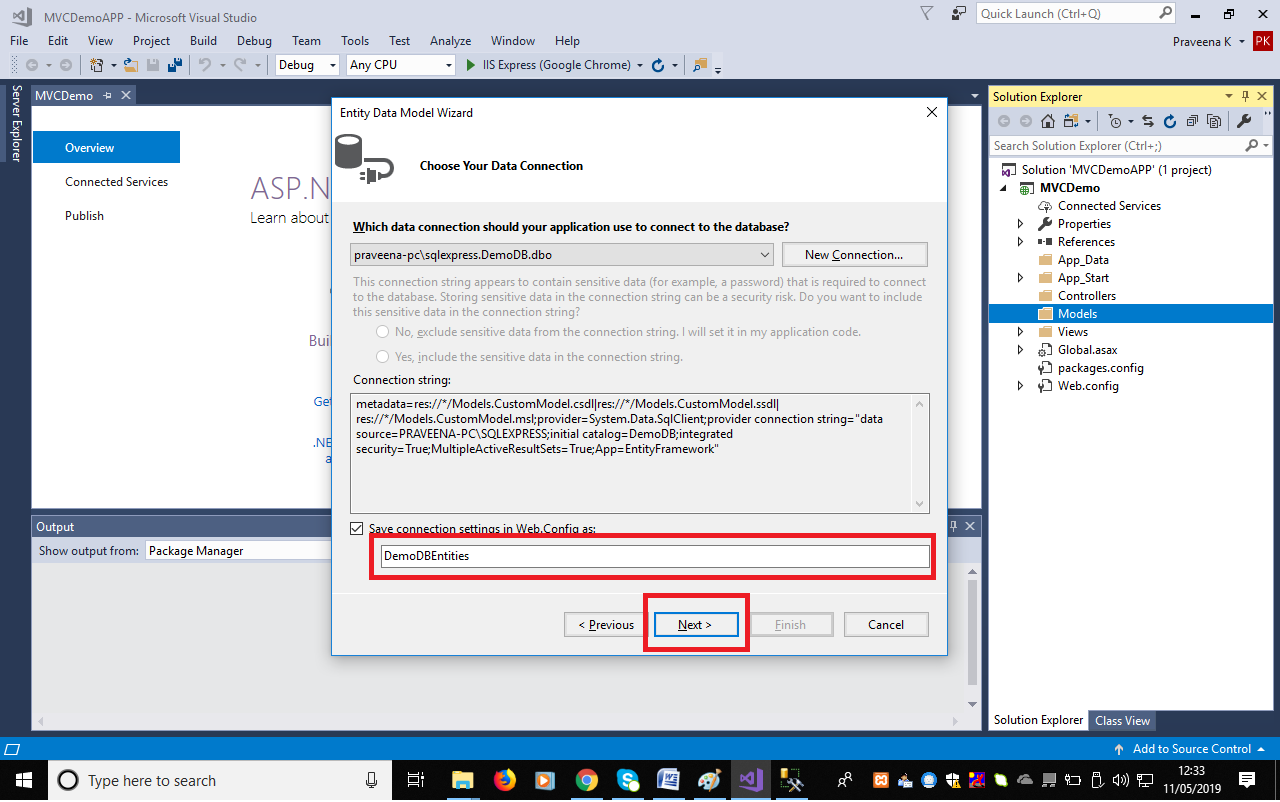
Clicking on ‘**Test Connection**’ will return success message if the connectivity is established with server. After confirming connectivity of server, select **OK.**

****

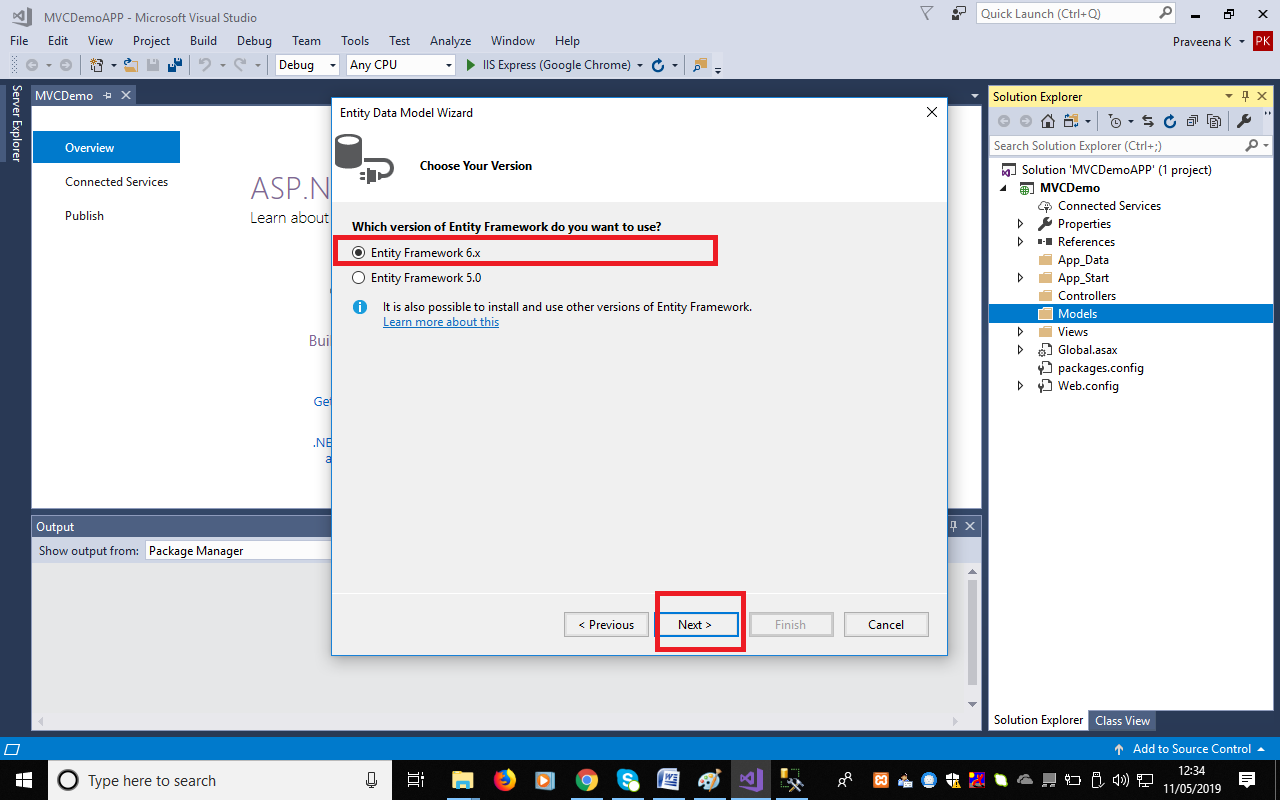
**Step 12**:

Web.Config is a configuration file which holds application specific data as well as connection specific information. If credentials of the DB are to be retained in config file, it can be chosen accordingly in below screen.

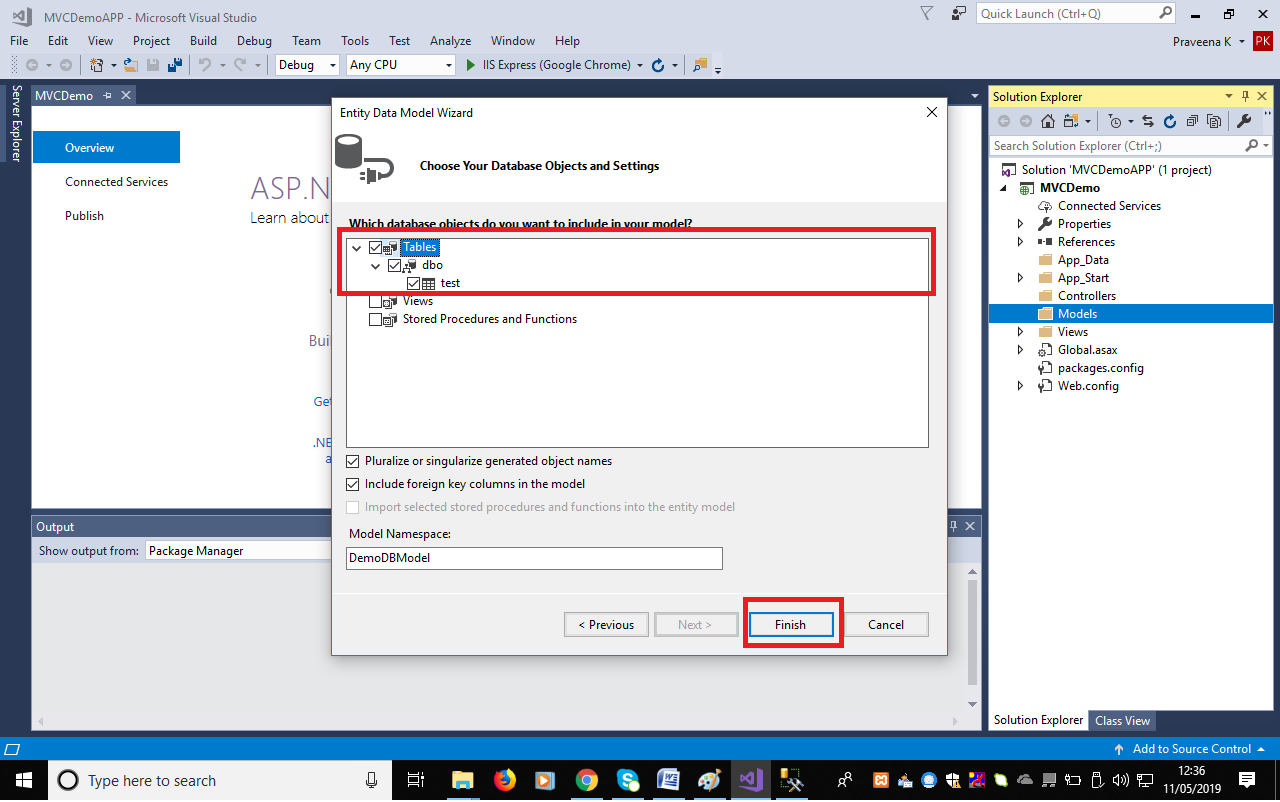
Provide a user defined name for storing connection related information as settings in the Web.Config file.



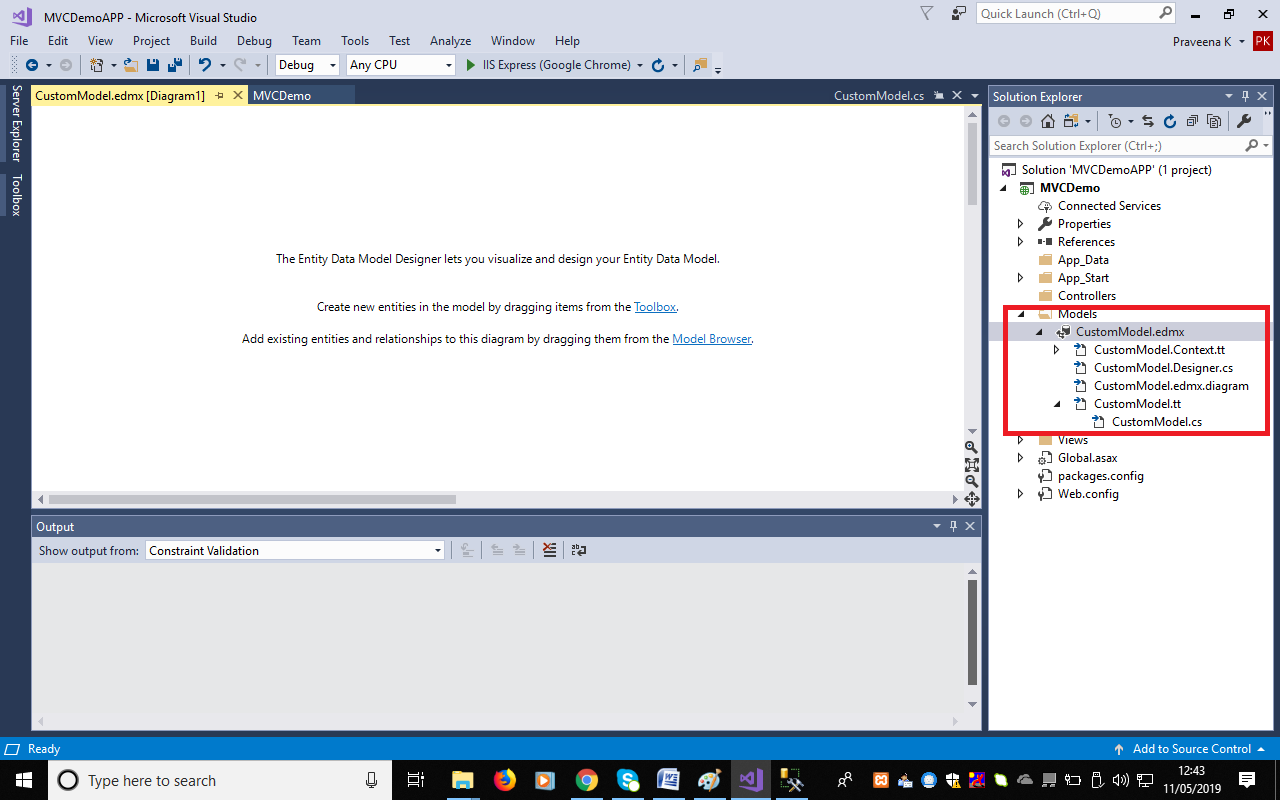
**Step 13**:



**Step 14**: Select the Database table that will be used for querying. Click on **Finish**. With this step, model gets created successfully and a link to Database is established. Default code will also get auto generated, in this approach.

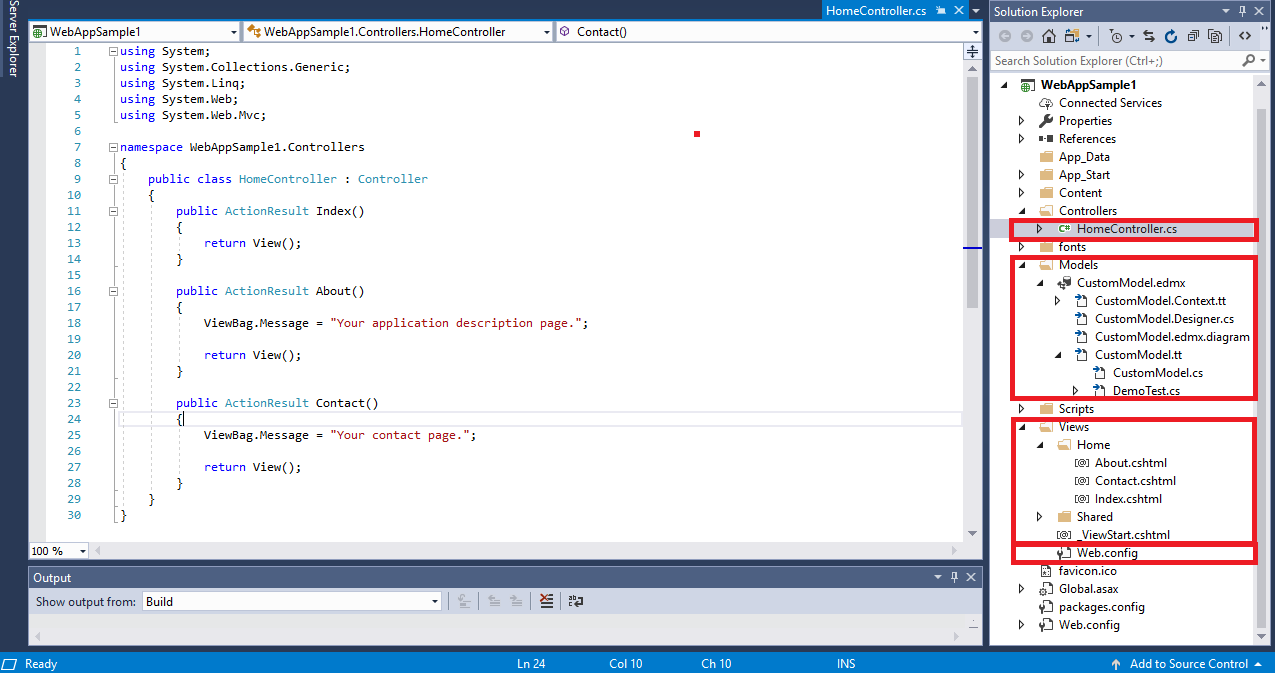


**Step 15**:

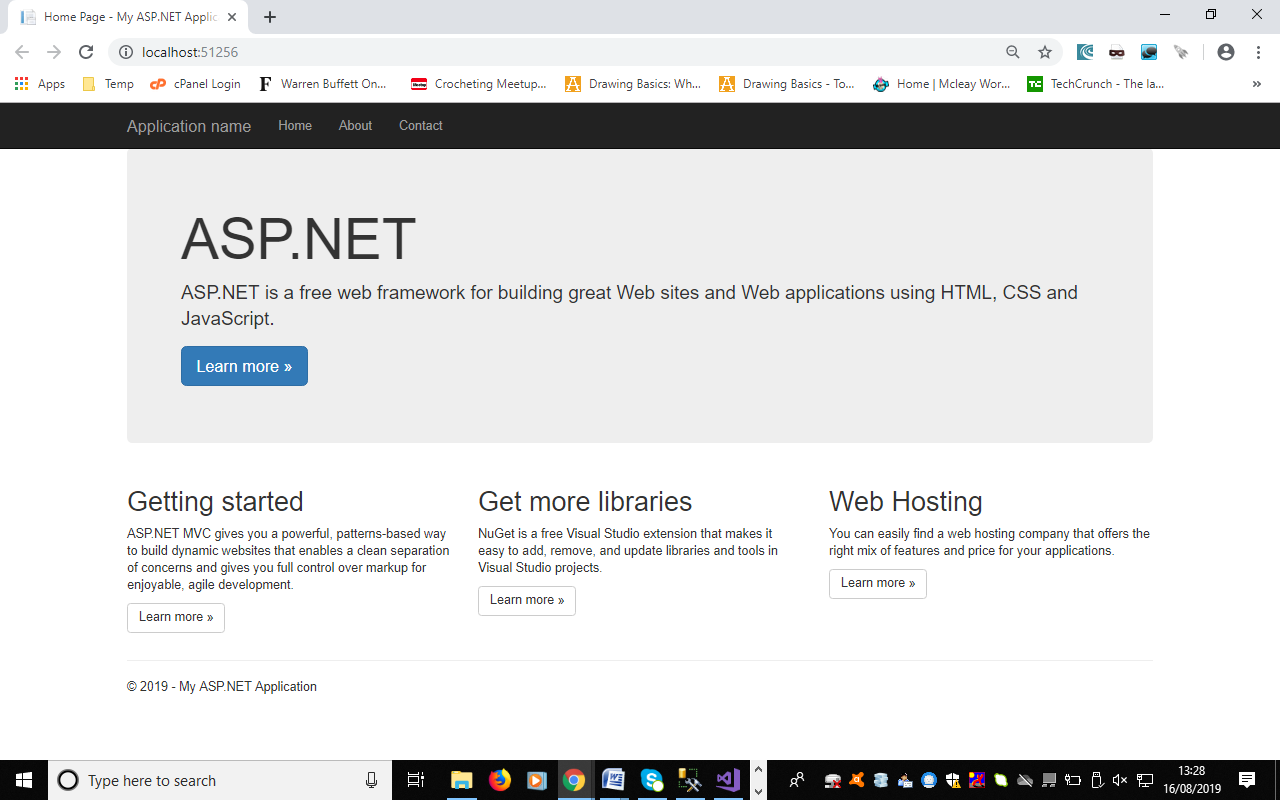


**Step 16**: Default Controller and Default Views would have got created for MVC framework. Hence the application can be “**Run**” to view the default index file from browser.

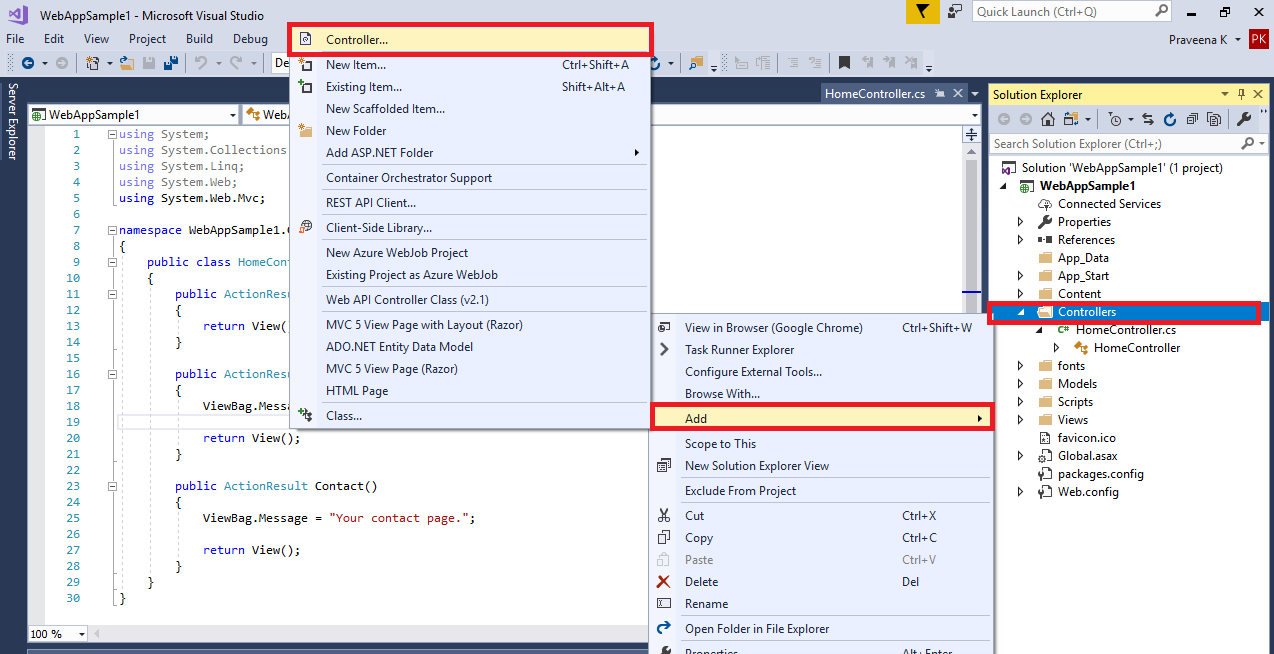
From this step, a new project named WebAppSample1 will be available in all the screens. Do not be confused on the project and related files.



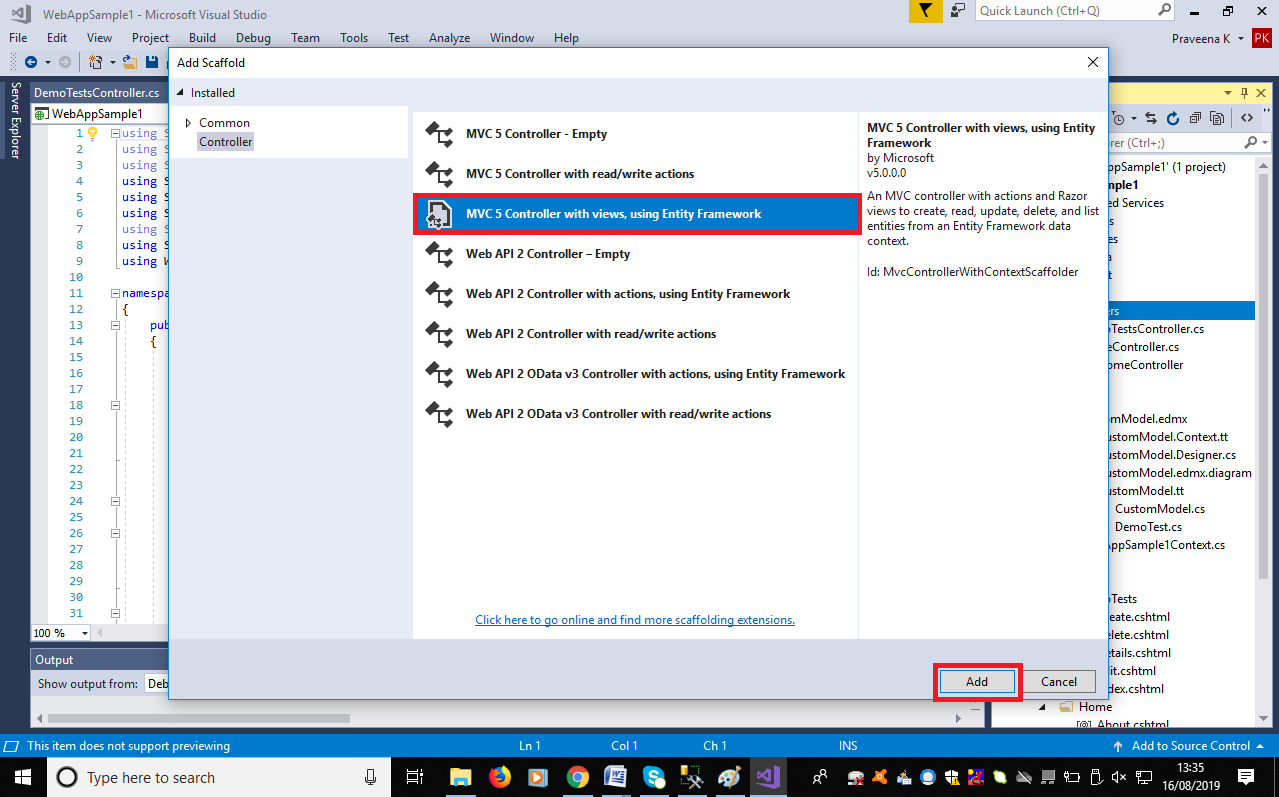
**Step 17**: Click on “**IIS Express (Google Chrome)**” from standard menus.



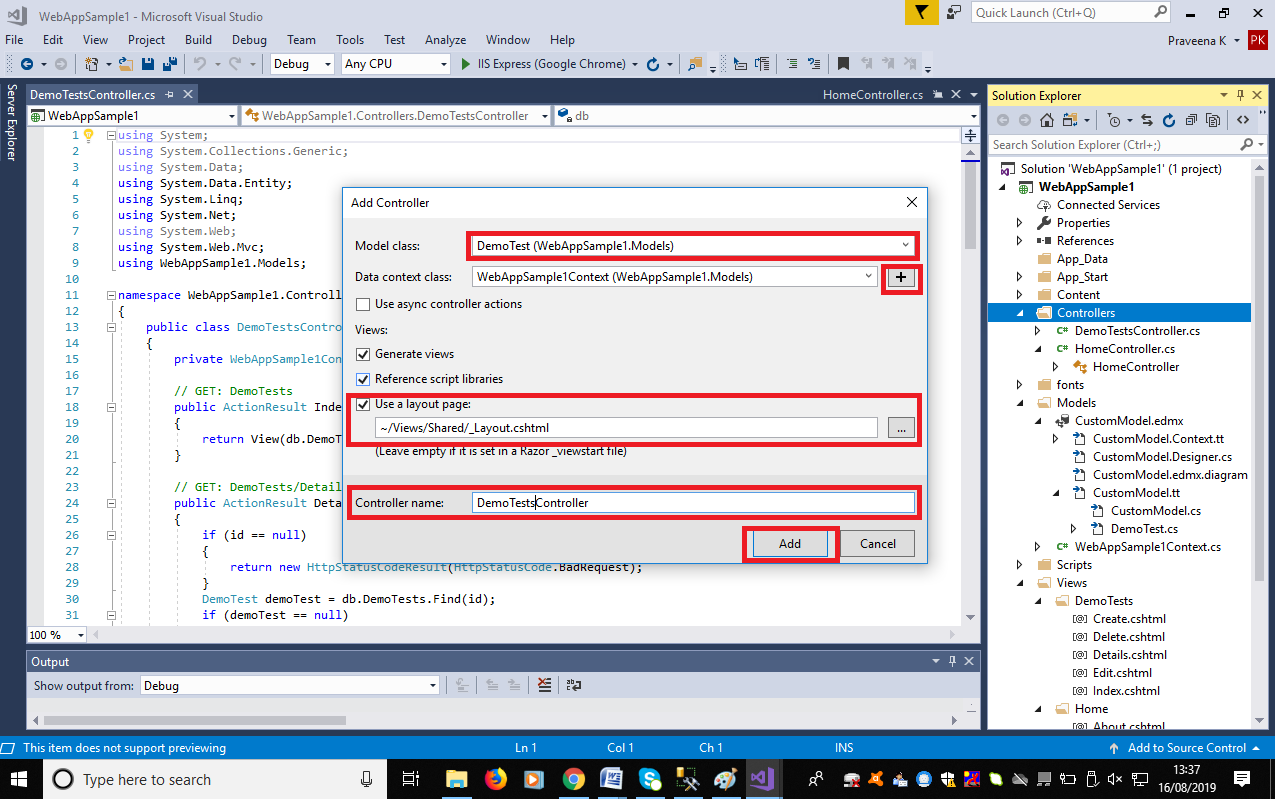
**Step 18**: Create a new Custom Controller.



**Step 19**:

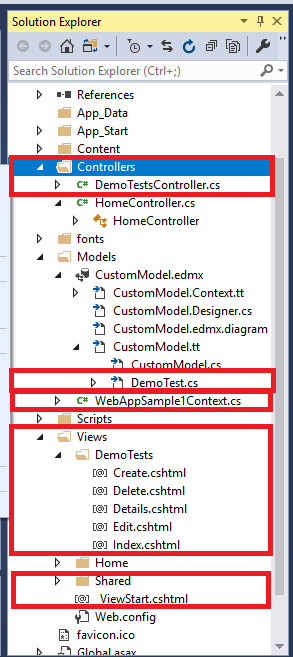


**Step 20**: Choose the Model class that was created for the Database. Create a new Data Context class and use the default shared layout file available. Provide name to the controller and click on “**Add**”.



**Step 21**:

Code gets auto generated for all database operations based on the DB connectivity that was linked. Adding a controller based on Data Entity also creates a default code in **views** folder.



**Step 22**:

Since the controller used in the application is the default controller it always displays the default home page. To display custom controller and custom views created, RouteConfig.cs file is to be modified to point to DemoTestsController.cs.

App\_Start/ RouteConfig.cs.

Sample Code

namespace WebAppSample1

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "DemoTests", action = "Index", id = UrlParameter.Optional }

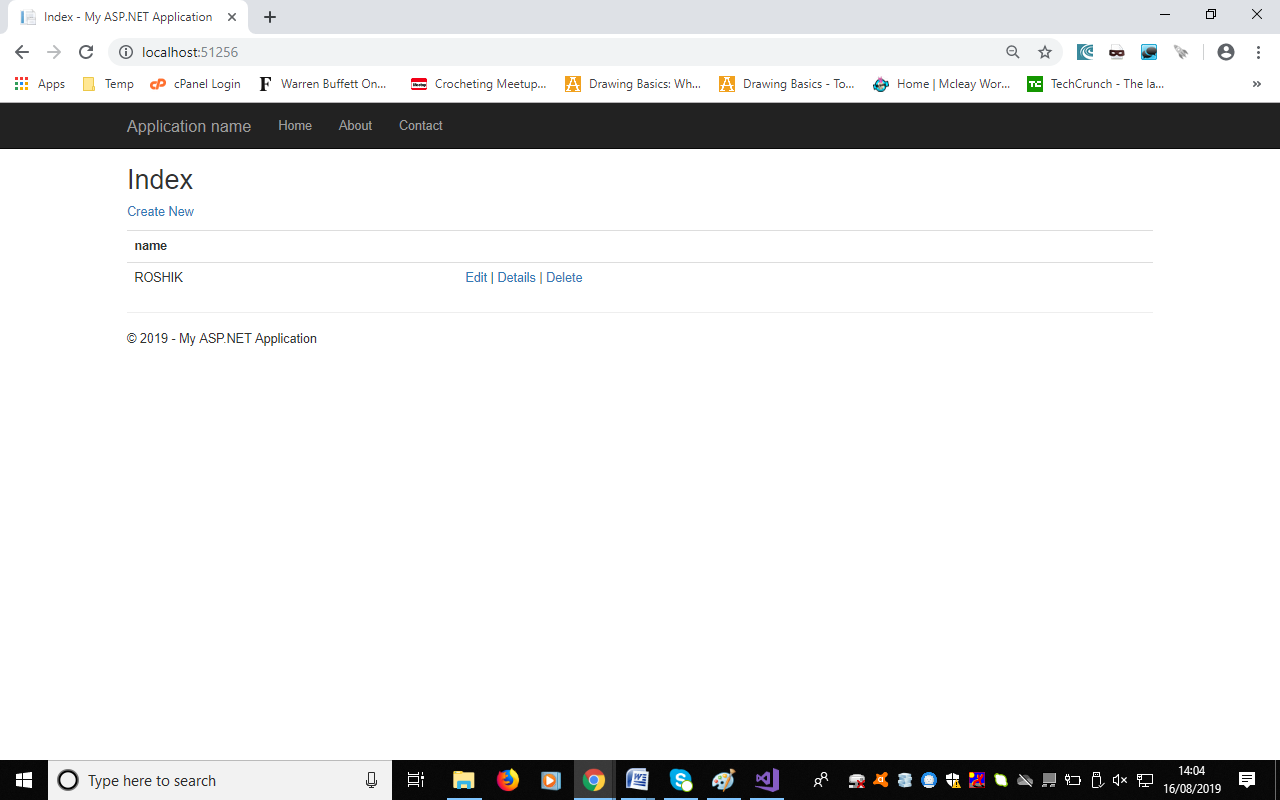
);

}

}

}

**Step 23**: Run the application by clicking on “**IIS Express (Google Chrome)**” from standard menus.



Click on “Create New” and create new record, try other operations subsequently to verify all the operations to the DB table.