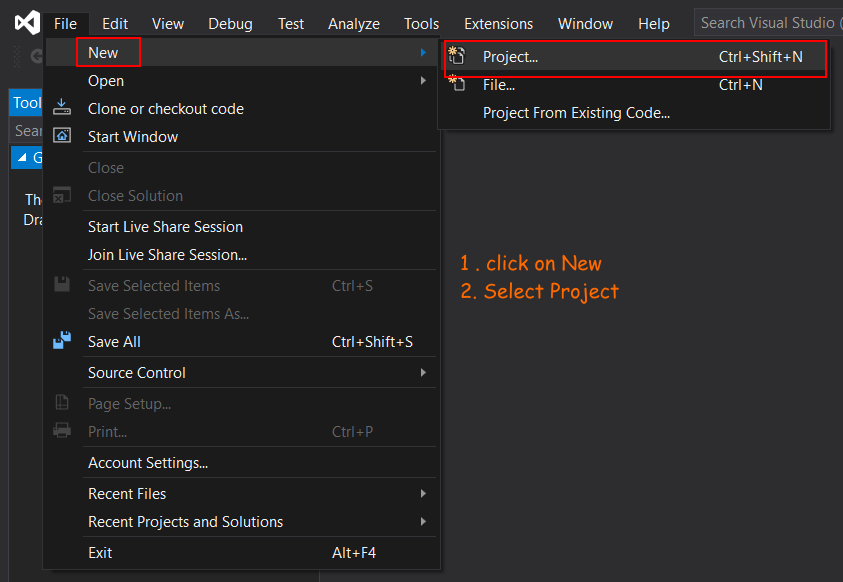
**MVC in ASP.net**

**Step 1**

Open Visual Studio. Here I use Visual Studio 2019, you can use any one as your system.

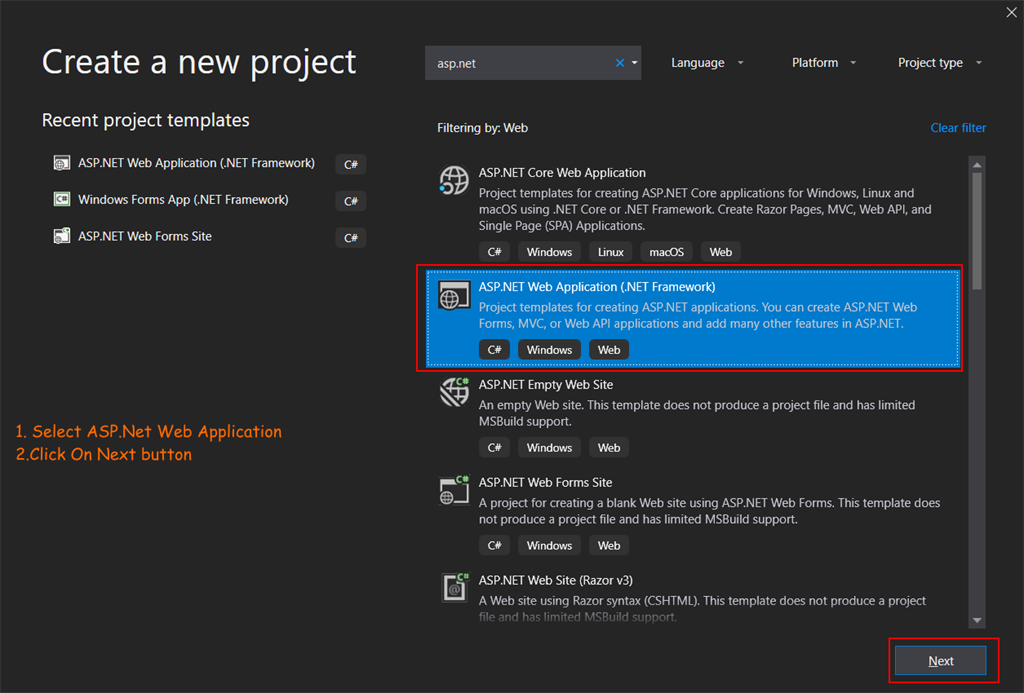
**Step 2**

Create a new project by clicking on File>New>Project.



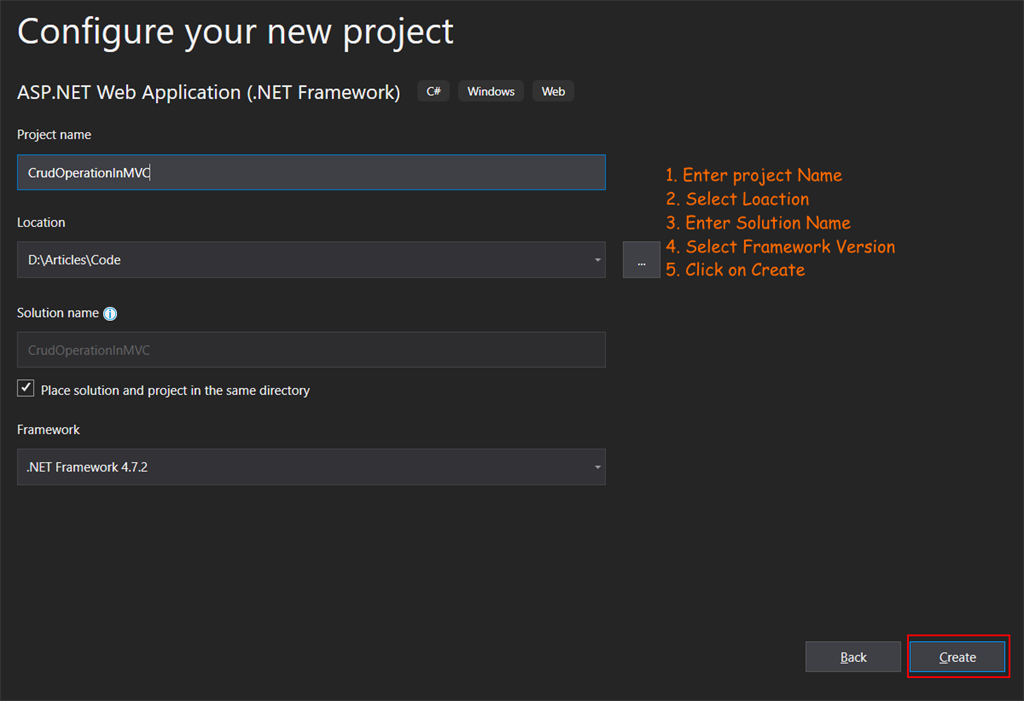
**Step 3**

Select Asp.Net Web Application and click on next button.



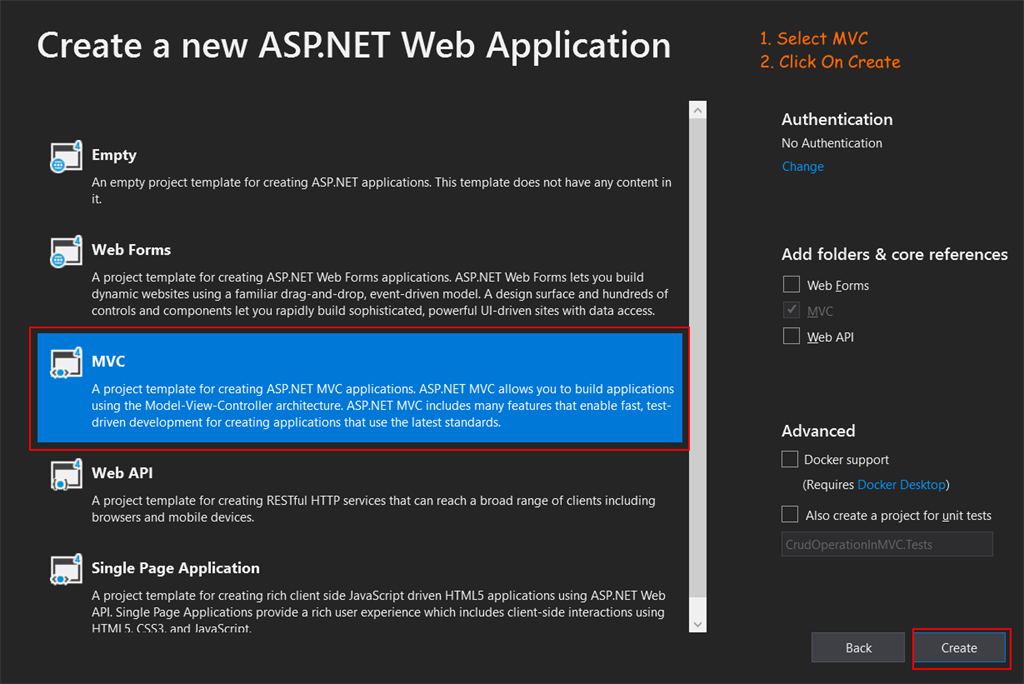
**Step 4**

In the next screen you need to enter a few details like your project name, project location where you want to save your project, solution name and .Net Framework version. After entering all details click on Create button.

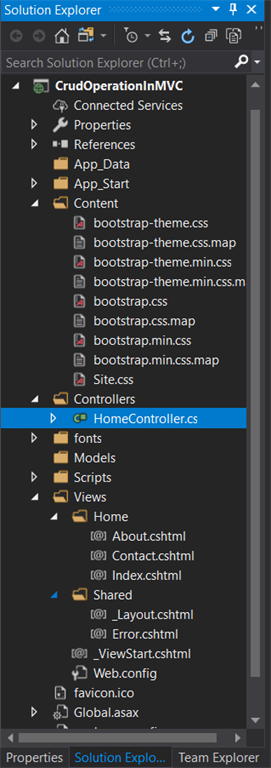


**Step 5**

In Next select MVC project and click on Create button.

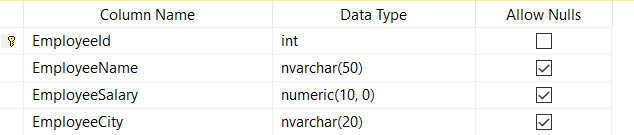


Now your project is ready and you can see project structure in below image. Here I deleted all view from home controller because we created a  new for our requirement.



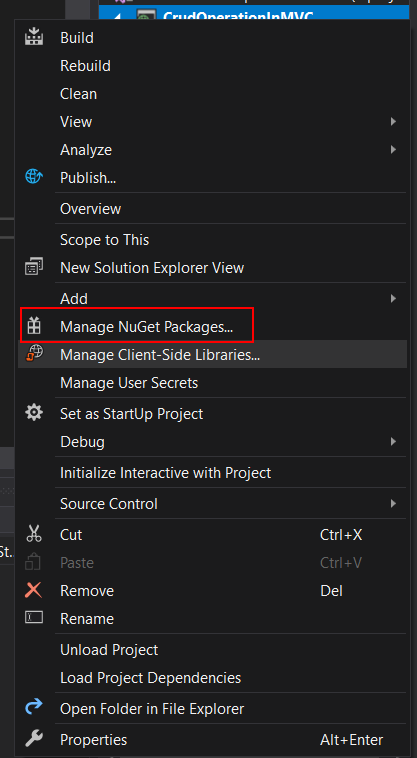
**Step 6**

Now in this project we will use database first approach so we need to create a database and a table. Here I created a database name with Tutorials and table Employee with the  following columns as you can see in the below image.



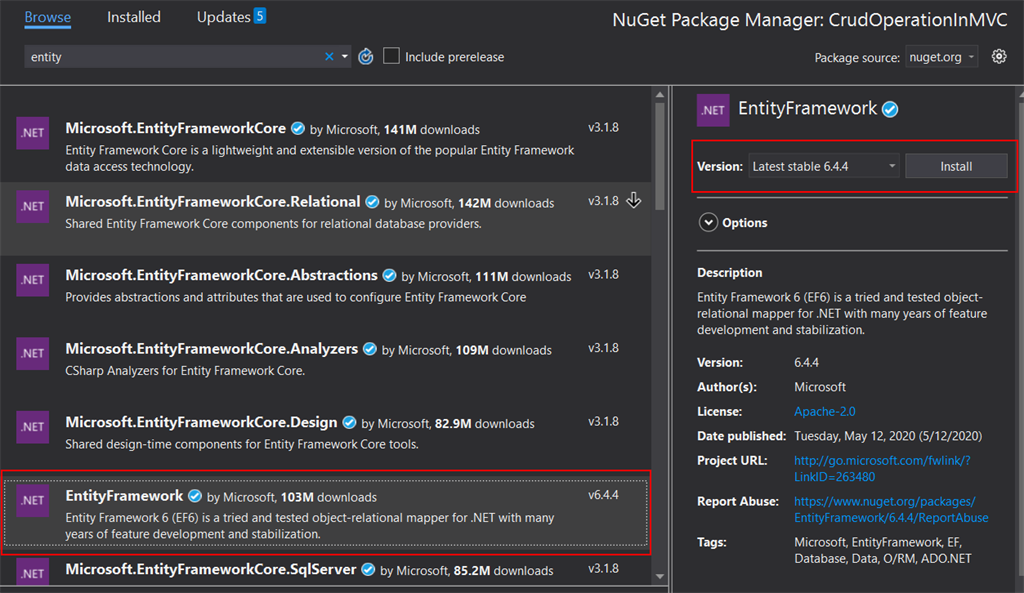
**Step 7**

Now we need to add Entity Framework in our project for performing some operation. For adding Entity Framework click on project name in solution explore then click on Manage NuGet Packages.



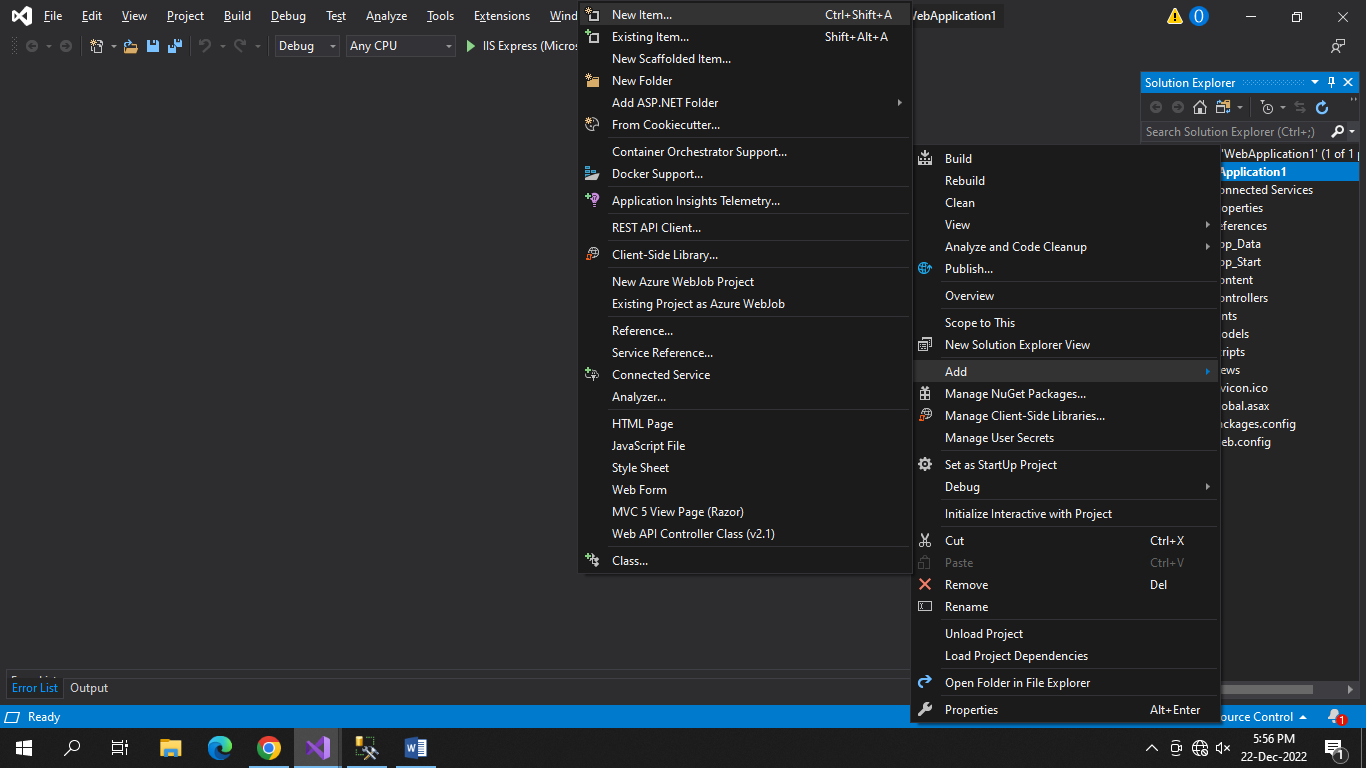
**Step 8**

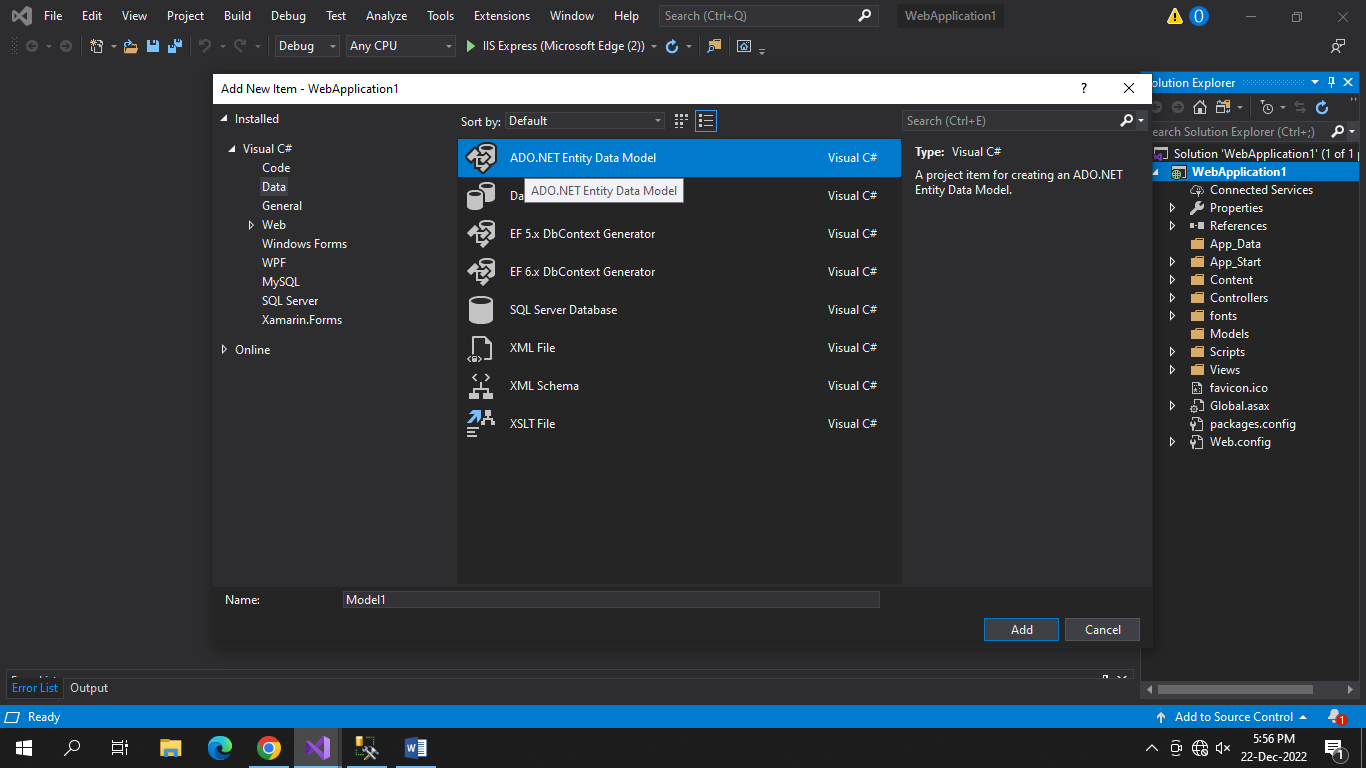
Now you see a window with many NuGet Packages. Search for Entity framework and click on install button and after a few seconds Entity Framework is  installed in your project.

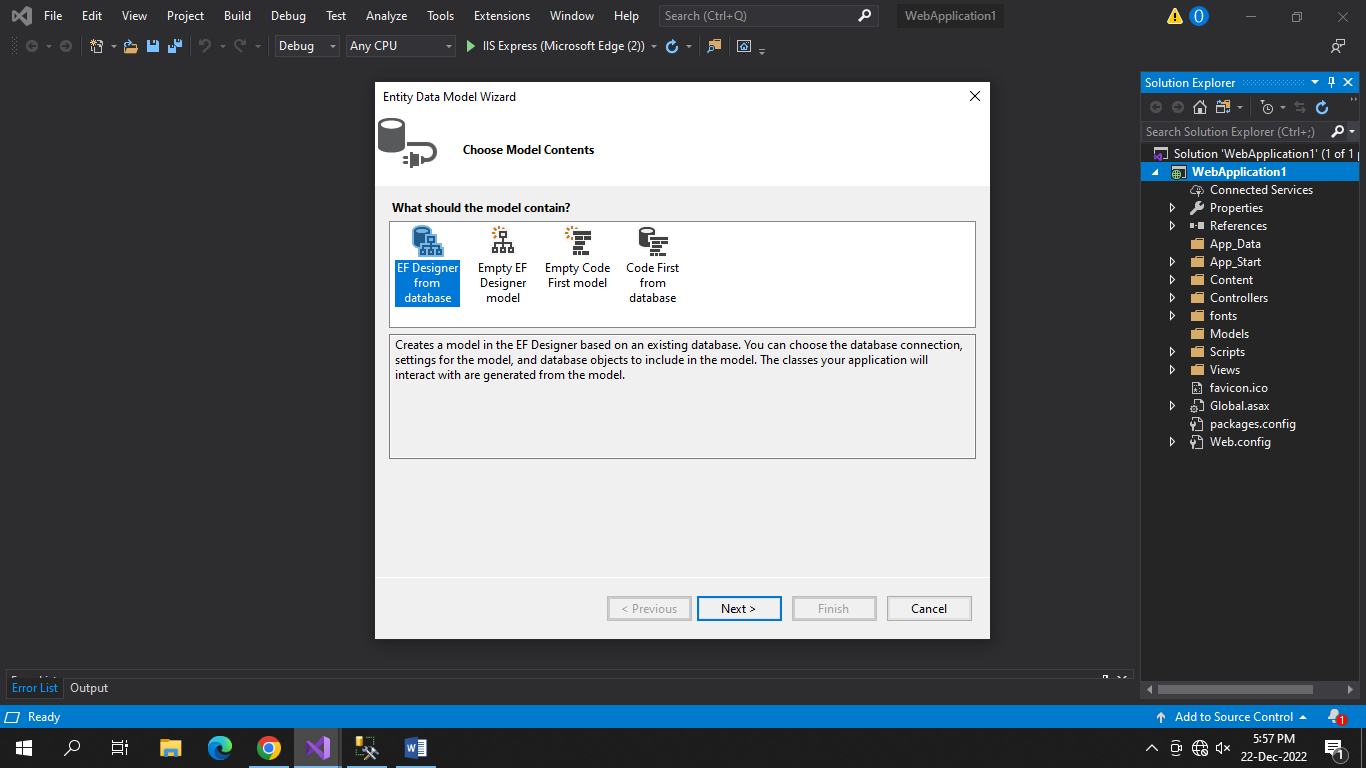


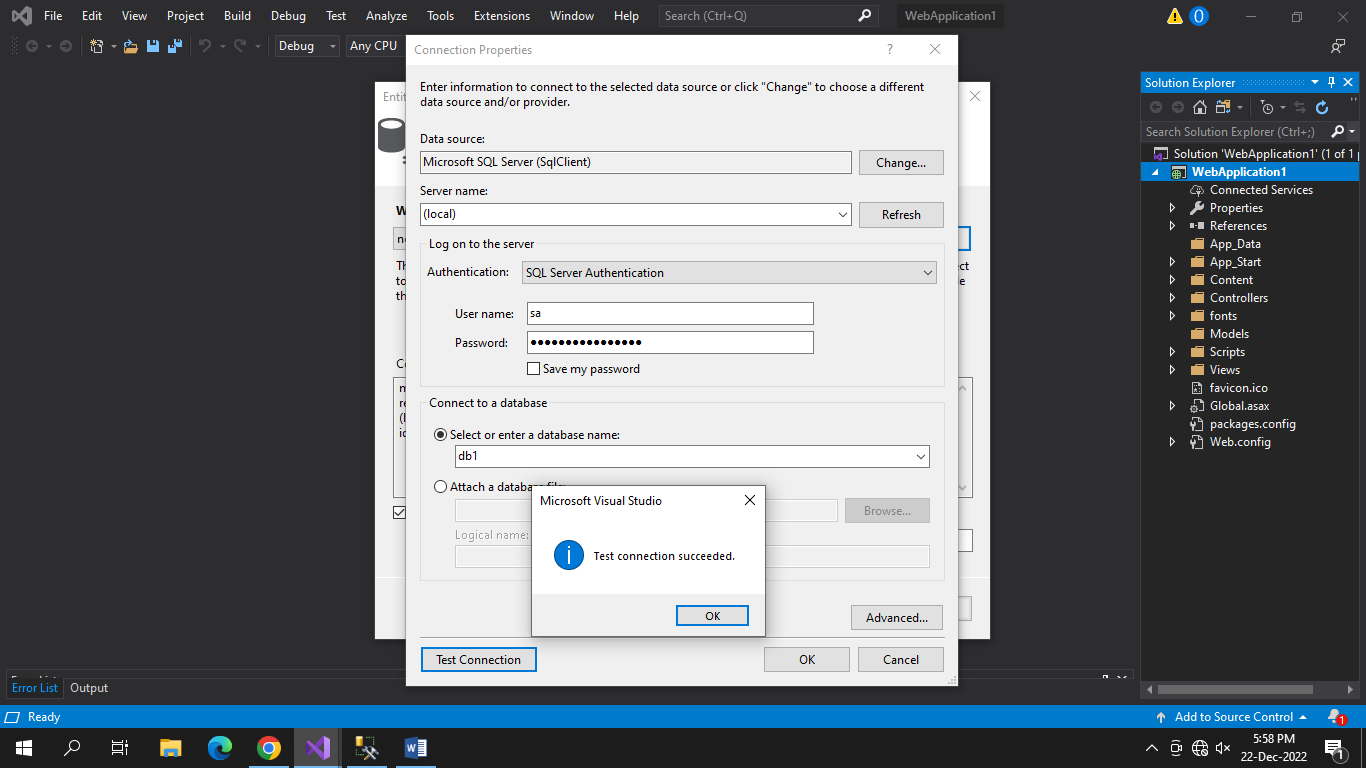
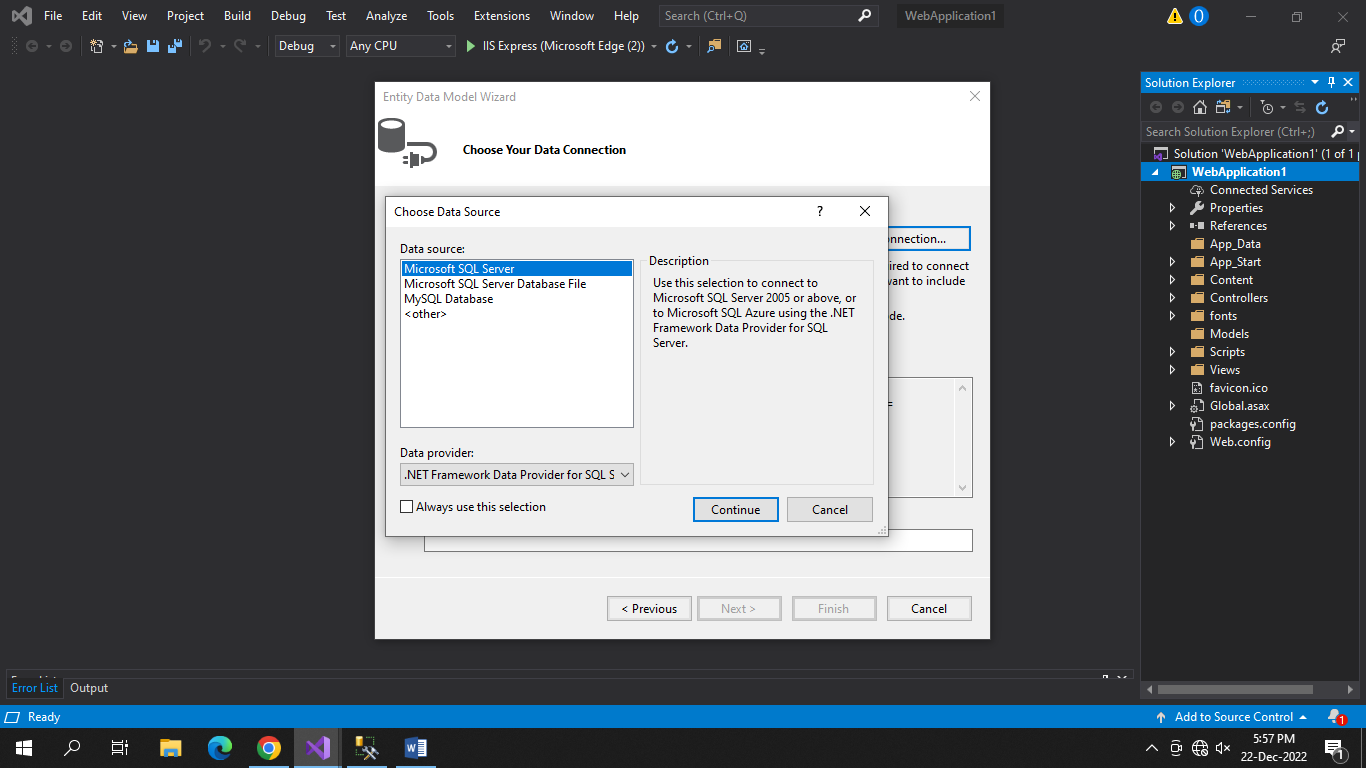
**Step 9**

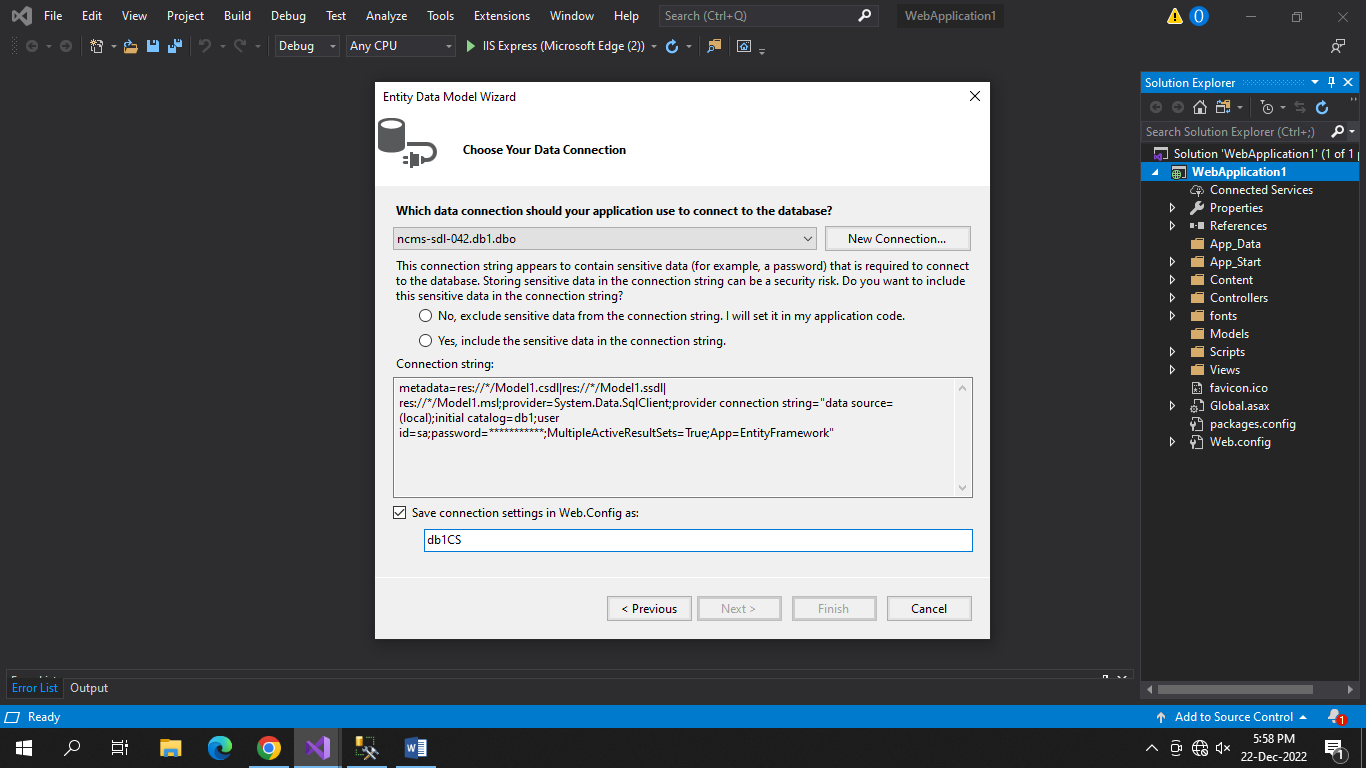
Now we need to add ADO.Net Entity Data Model in our project. For adding ADO.Net Entity Data Model click on your project name then click on Add and then click on new Item or you can use short cut key Ctrl+Shift+A .

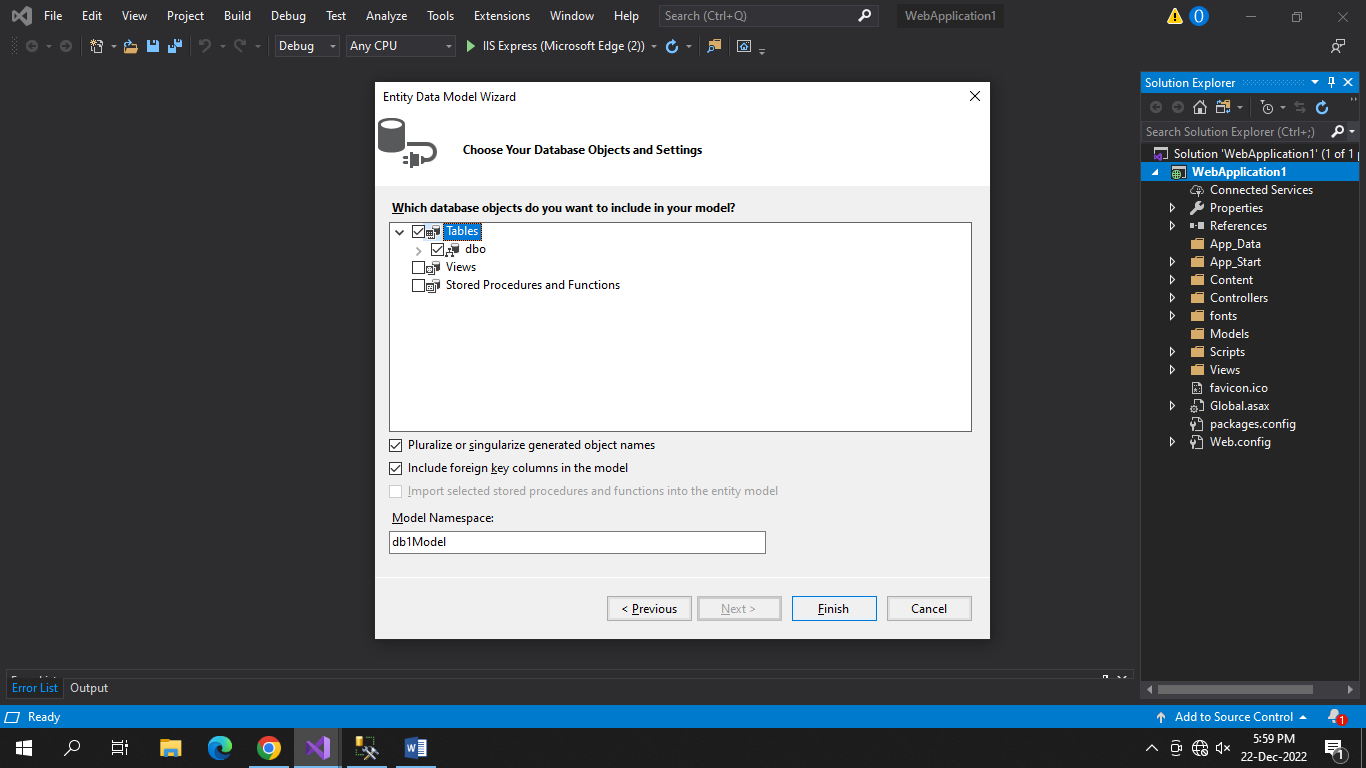


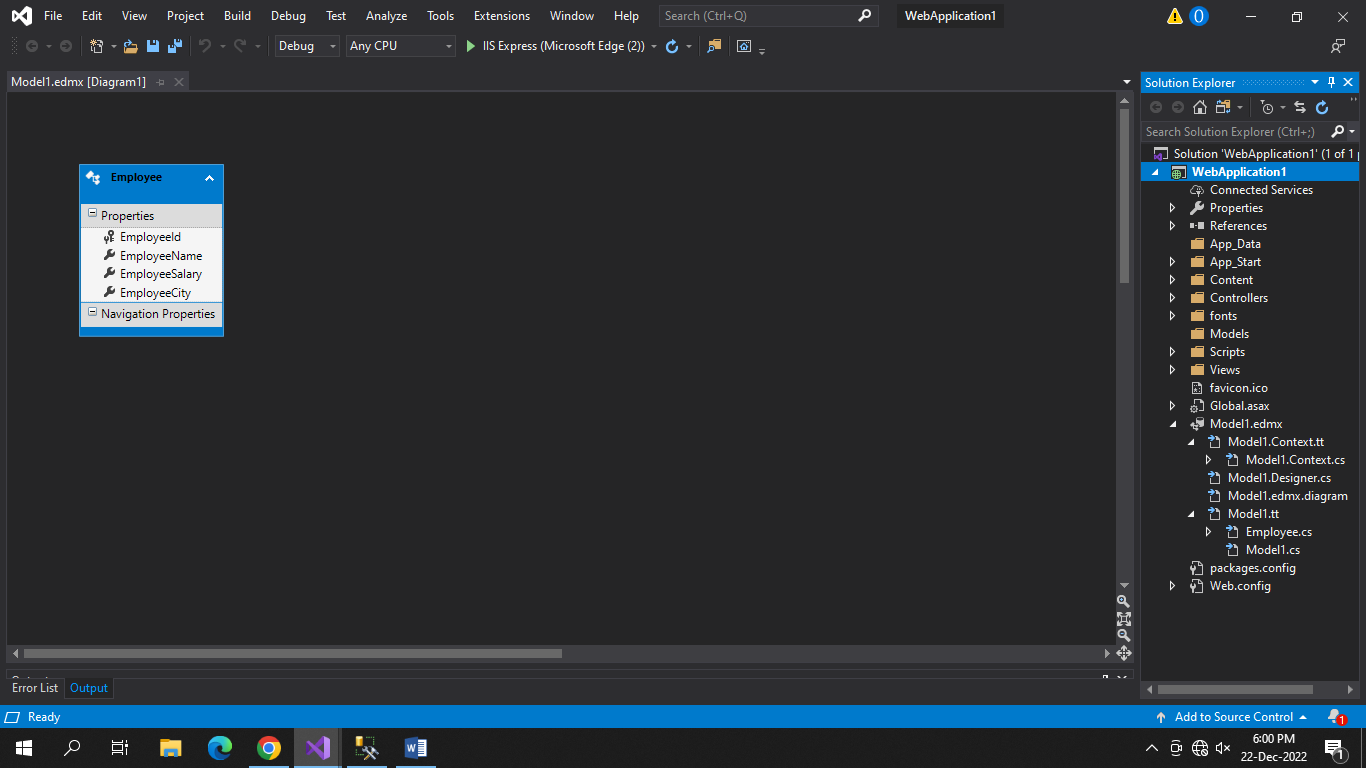












Here is Employee Model.

1. **public** partial **class** Employee
2. {
4. **public** **int** EmployeeId { **get**; **set**; }
5. **public** **string** EmployeeName { **get**; **set**; }
6. **public** Nullable<**decimal**> EmployeeSalary { **get**; **set**; }
7. **public** **string** EmployeeCity { **get**; **set**; }
8. }

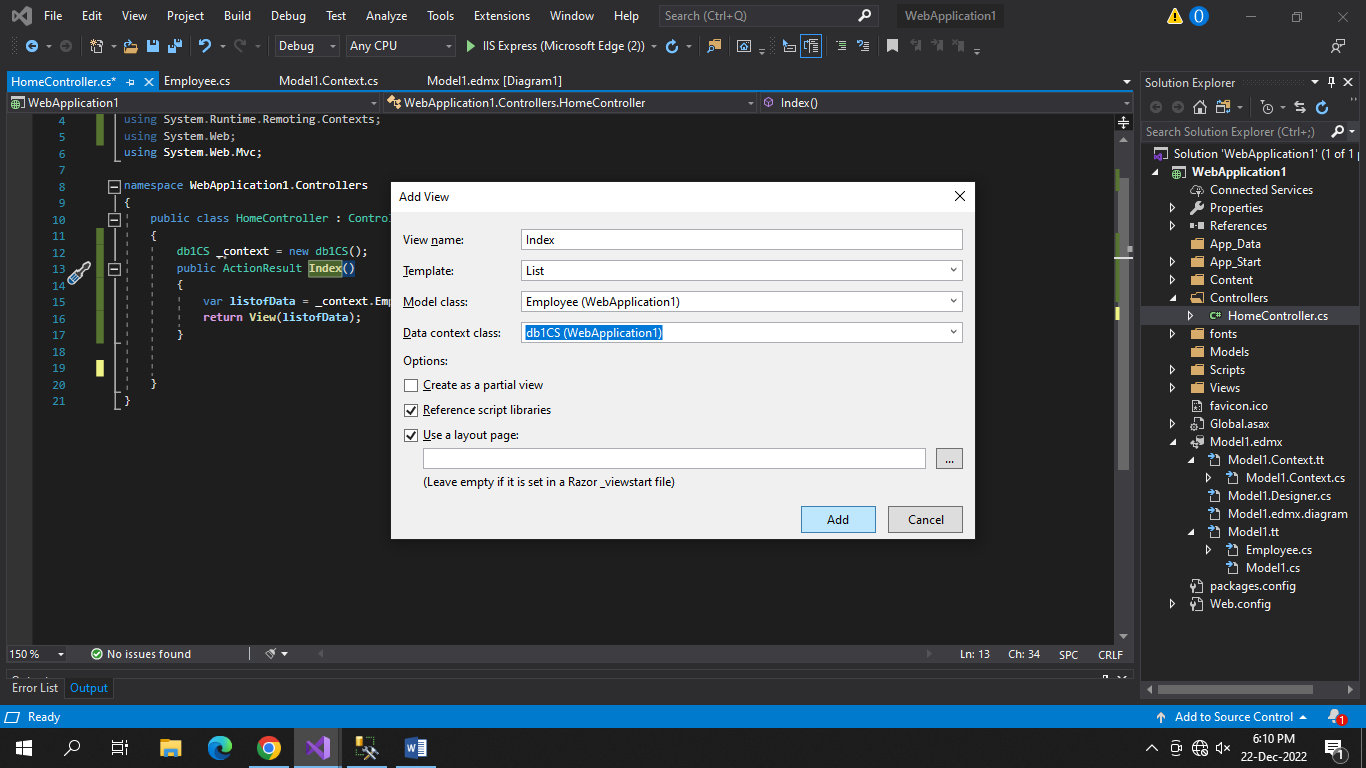
**Step 14**

Now delete all method from your controller and add index method as show in the below code.

1. db1CS \_context=new db1CS();
2. **public** ActionResult Index()
3. {
4. var listofData = \_context.Employees.ToList();
5. **return** View(listofData);
6. }

**Step 15**

Now add view by right clicking in method then click on Add View and you will see a new popup window with the following options. And then click on Add button.



* View Name: Name of your Cs Html file
* Template: There are many templates which give pre defined design like list, create, edit, delete, details select list for index method
* Model Class: Select your model class which we generated in the previous step.
* Data Context Class: Select your data context class, here I have TutorialCS whose name we gave in previous step.
* Use Layout Page: if you check this item compiler add layout file from \_Shared folder in this view. You can select different layout file if you have more than one .

**Step 16**

Now we create a view for adding a new record. Here we need to create two Action methods, one is get method and the second is post method. Get method is called when we click on create button and return view. Post method is used for saving data in the table.

[HttpGet]

**public** ActionResult Create()

 {

**return** View();

 }

 [HttpPost]

**public** ActionResult Create(Employee model)

 {

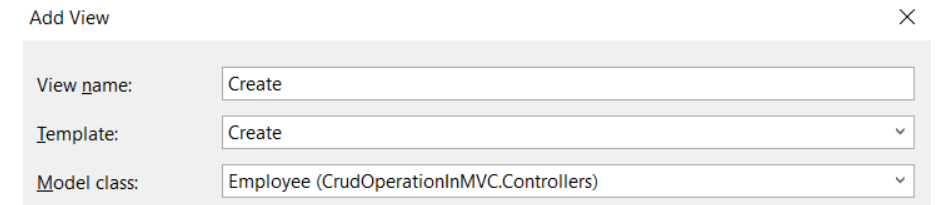
     \_context.Employees.Add(model);

     \_context.SaveChanges();

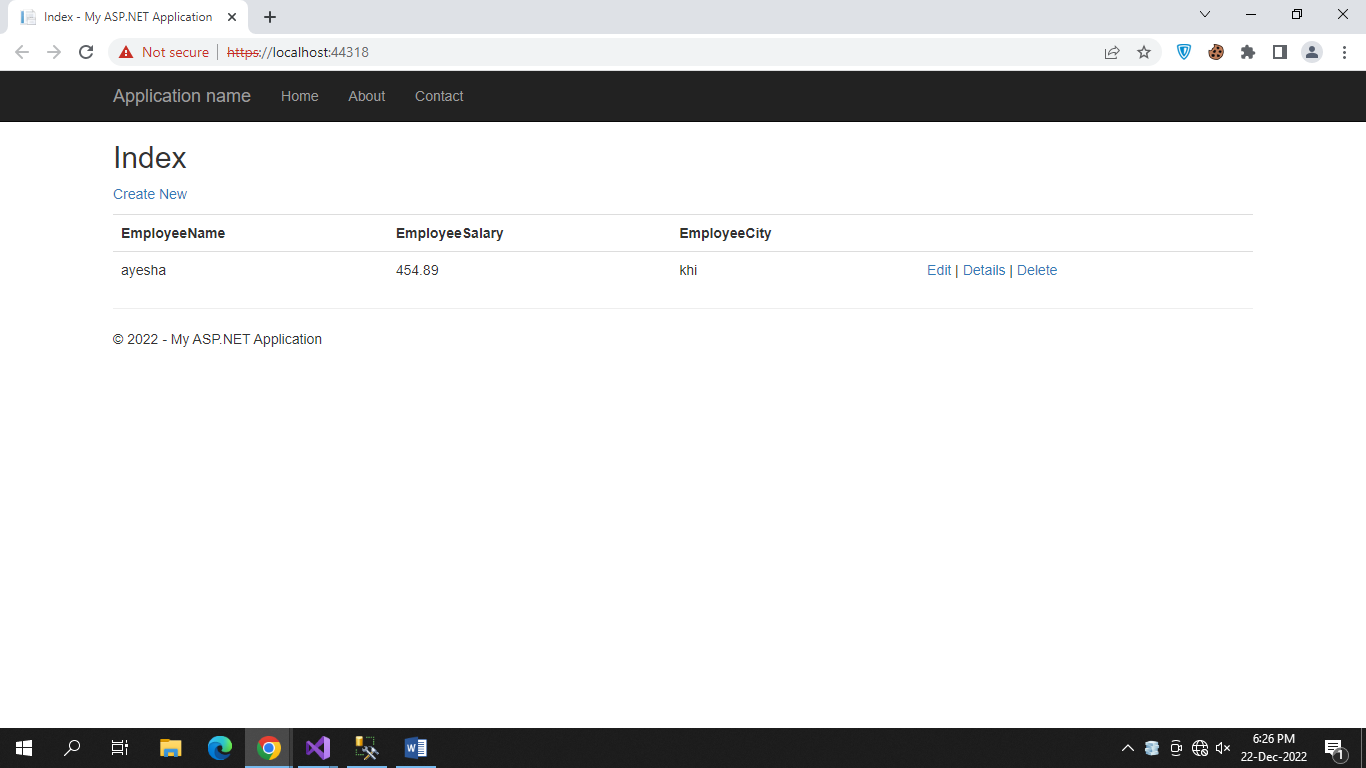
     ViewBag.Message = "Data Insert Successfully";

**return** View();

 }



Output:



**Step 17**

Now we will implement Edit functionality. For edit we also create two methods, get and post type. Get type method gets data from database by recording it and passing it in view. Post method saves updated data in database. Code for both methods is below.

[HttpGet]

**public** ActionResult Edit(**int** id)

{

    var data = \_context.Employees.Where(x => x.EmployeeId == id).FirstOrDefault();

**return** View(data);

}

[HttpPost]

**public** ActionResult Edit(Employee Model)

{

    var data = \_context.Employees.Where(x => x.EmployeeId == Model.EmployeeId).FirstOrDefault();

**if** (data != **null**)

    {

        data.EmployeeCity = Model.EmployeeCity;

        data.EmployeeName = Model.EmployeeName;

        data.EmployeeSalary = Model.EmployeeSalary;

        \_context.SaveChanges();

    }

**return** RedirectToAction("index");

}

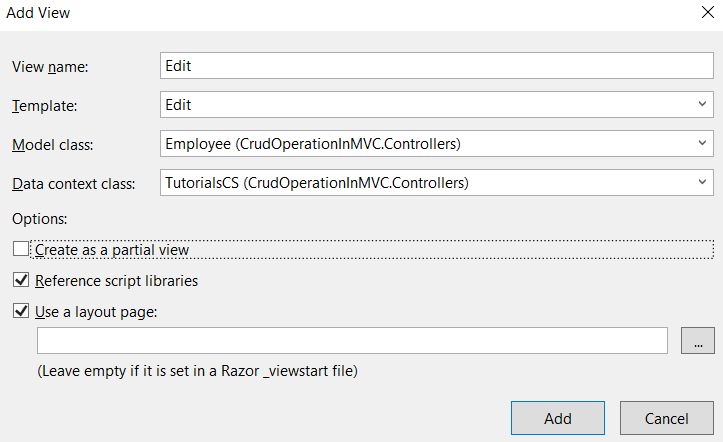
**Code Explanation**

* In get type method we get an id as a parameter using that we get first record from database and initialize in data variable then return that variable in view
* In post type method we get Employee type model as a parameter; we get record from database with EmployeeId which comes with parameter.
* If record is found in database with that id then we initialize upcoming data in database data.
* And save changes using SaveChanges Method. And return to index method using

RedirectToAction method.

**Step 18**

Now we need to add view for editing. For this  right click in Edit method click on Add View and select Edit Template' all other options are the same as index.



**Step 19**

Now we create view for viewing employee data. So create Action method detail with parameter id . Using this we get data from database and pass it in view. Add view by right clicking in method and selecting template Details.

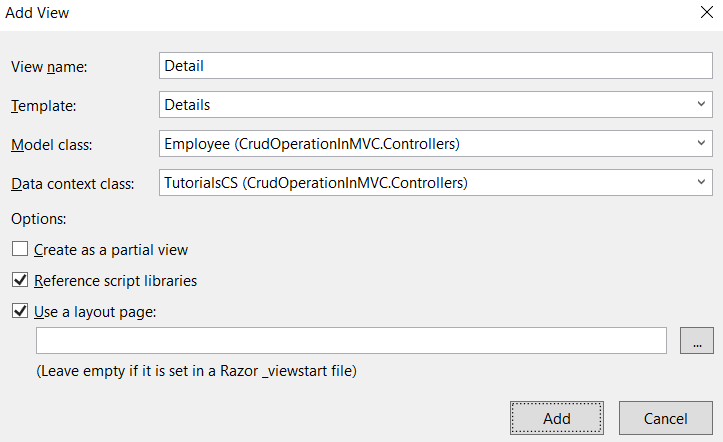
**public** ActionResult Detail(**int** id)

{

    var data = \_context.Employees.Where(x => x.EmployeeId == id).FirstOrDefault();

**return** View(data);

}



**Step 20**

Now we create Action Method for delete which takes id as n parameter.

**public** ActionResult Delete(**int** id)

{

    var data = \_context.Employees.Where(x => x.EmployeeId == id).FirstOrDefault();

    \_context.Employees.Remove(data);

    \_context.SaveChanges();

    ViewBag.Messsage = "Record Delete Successfully";

**return** RedirectToAction("index");

}

**Code Explanation**

* Here first we get data by id from database.
* Remove that entity which we get from database.
* Save changes using SaveChanges() method.
* And redirect to Index method.

