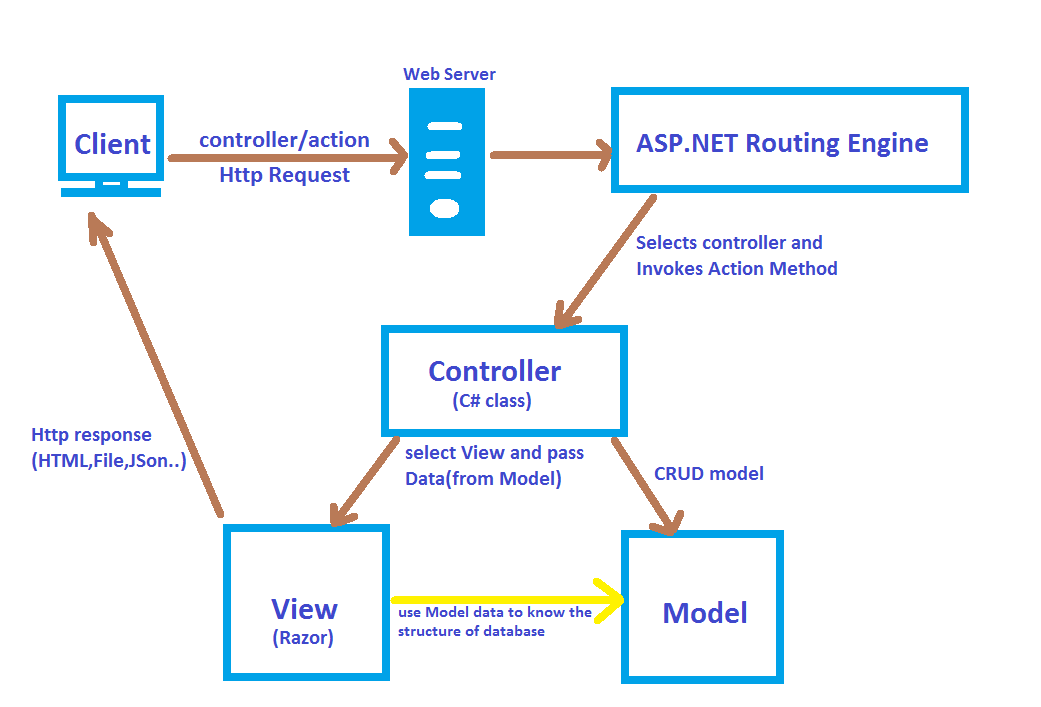
**Objective: Create a ASP.NET Core MVC Web application**

**Theory:**

**MVC Framework:**

The Model-View-Controller (MVC) architectural pattern separates an application into three main groups of components: Models, Views, and Controllers. This pattern helps to achieve separation of concerns. Using this pattern, user requests are routed to a Controller which is responsible for working with the Model to perform user actions and/or retrieve results of queries. The Controller chooses the View to display to the user, and provides it with any Model data it requires.



The ASP.NET Core MVC framework is a lightweight, open source, highly testable presentation framework optimized for use with ASP.NET Core.

ASP.NET Core MVC provides a patterns-based way to build dynamic websites that enables a clean separation of concerns. It gives you full control over markup, supports TDD-friendly development and uses the latest web standards.

## Features

ASP.NET Core MVC includes the following:

* Routing
* Model binding
* Model validation
* Dependency injection
* Filters
* Web APIs and etc.

**Implementation using Visual Studio 2019 IDE:**

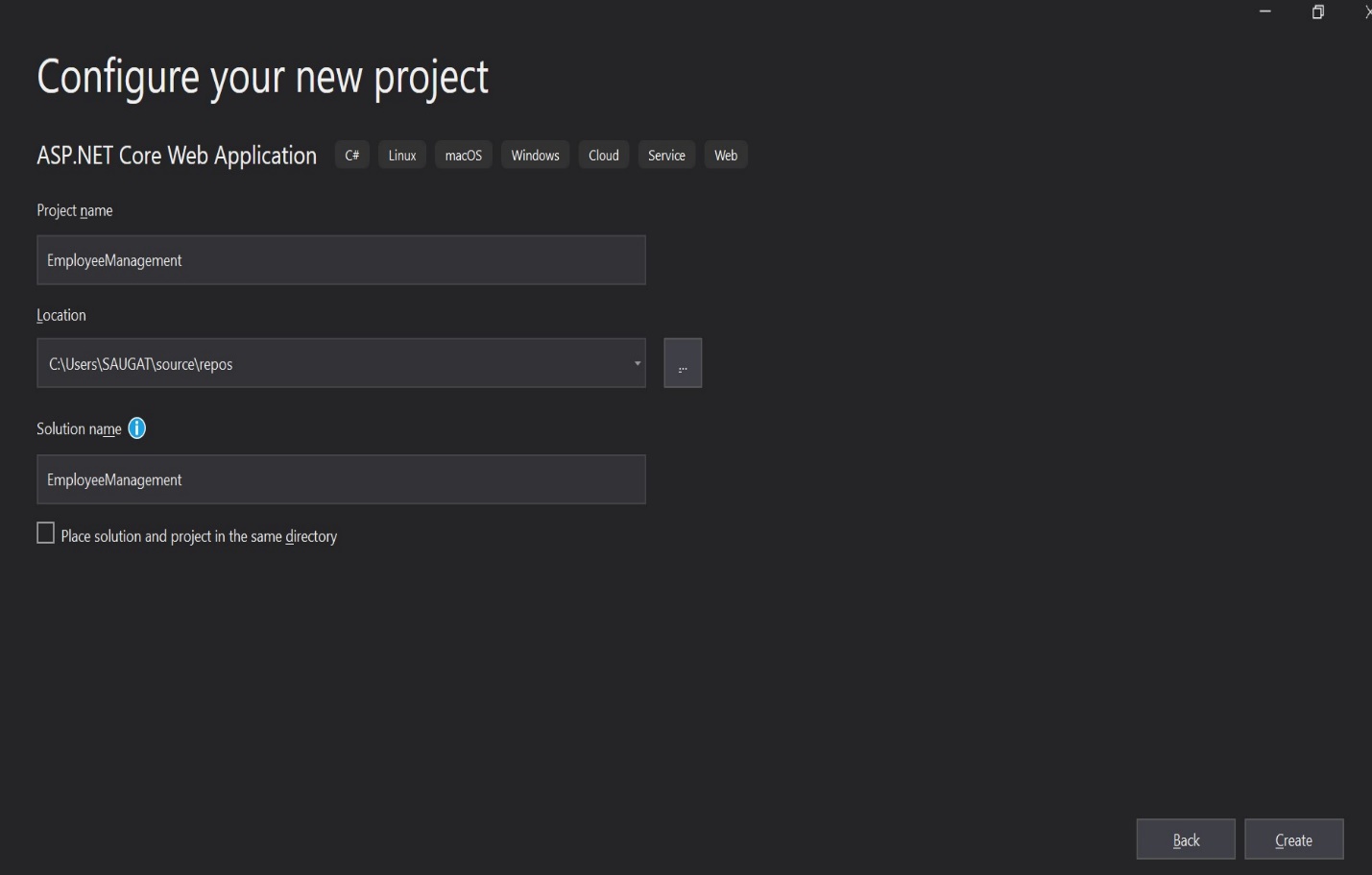
* **Creating a Web App & Run**

 From the Visual Studio, select Create a new project.

 Select ASP.NET Core Web Application and then select Next.

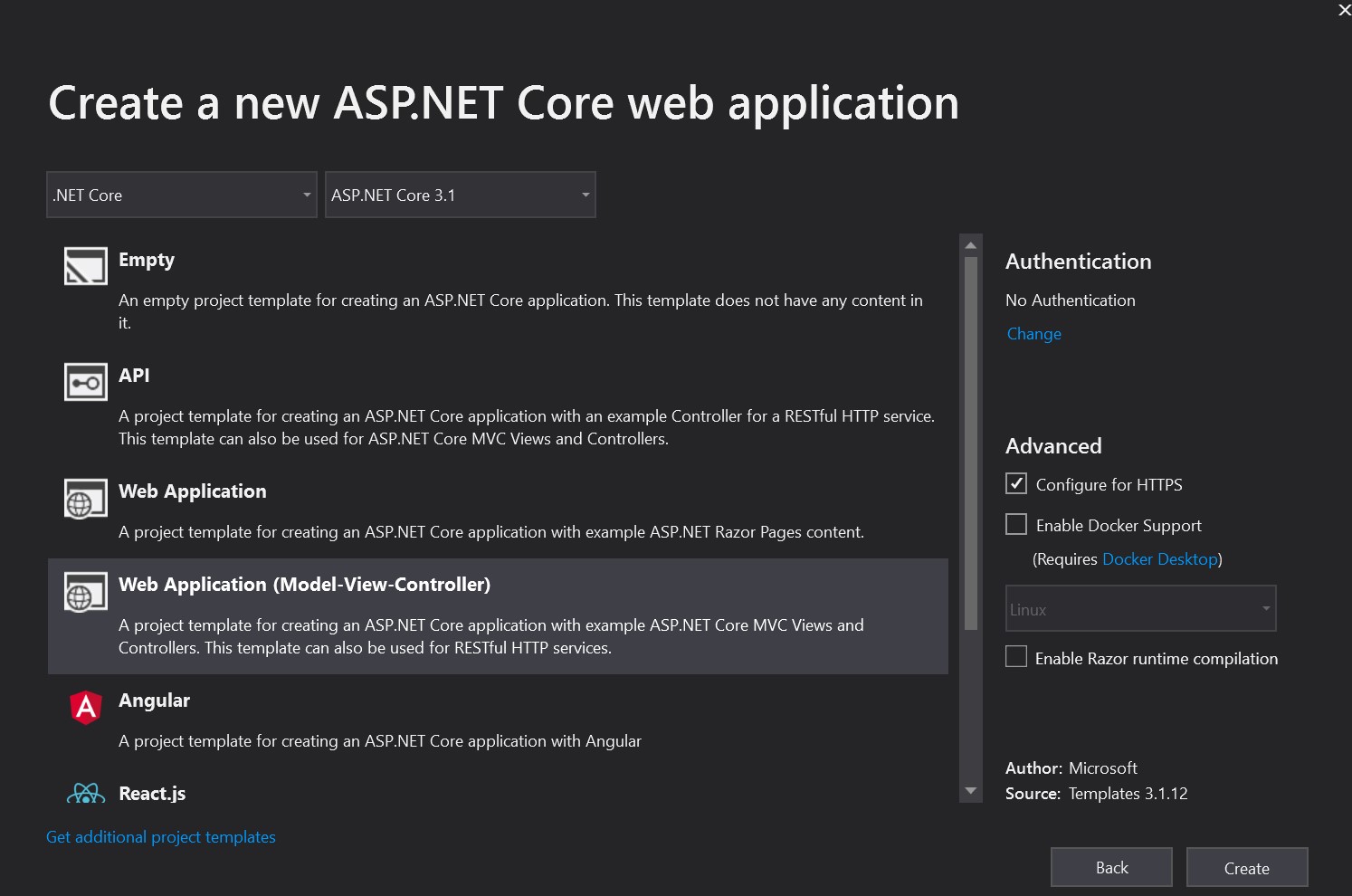
 Name the project as you like or EmployeeManagement as below:

 Choose the location path to save your project.



 Then, Click Create

 Select Web Application (Model-View-Controller),



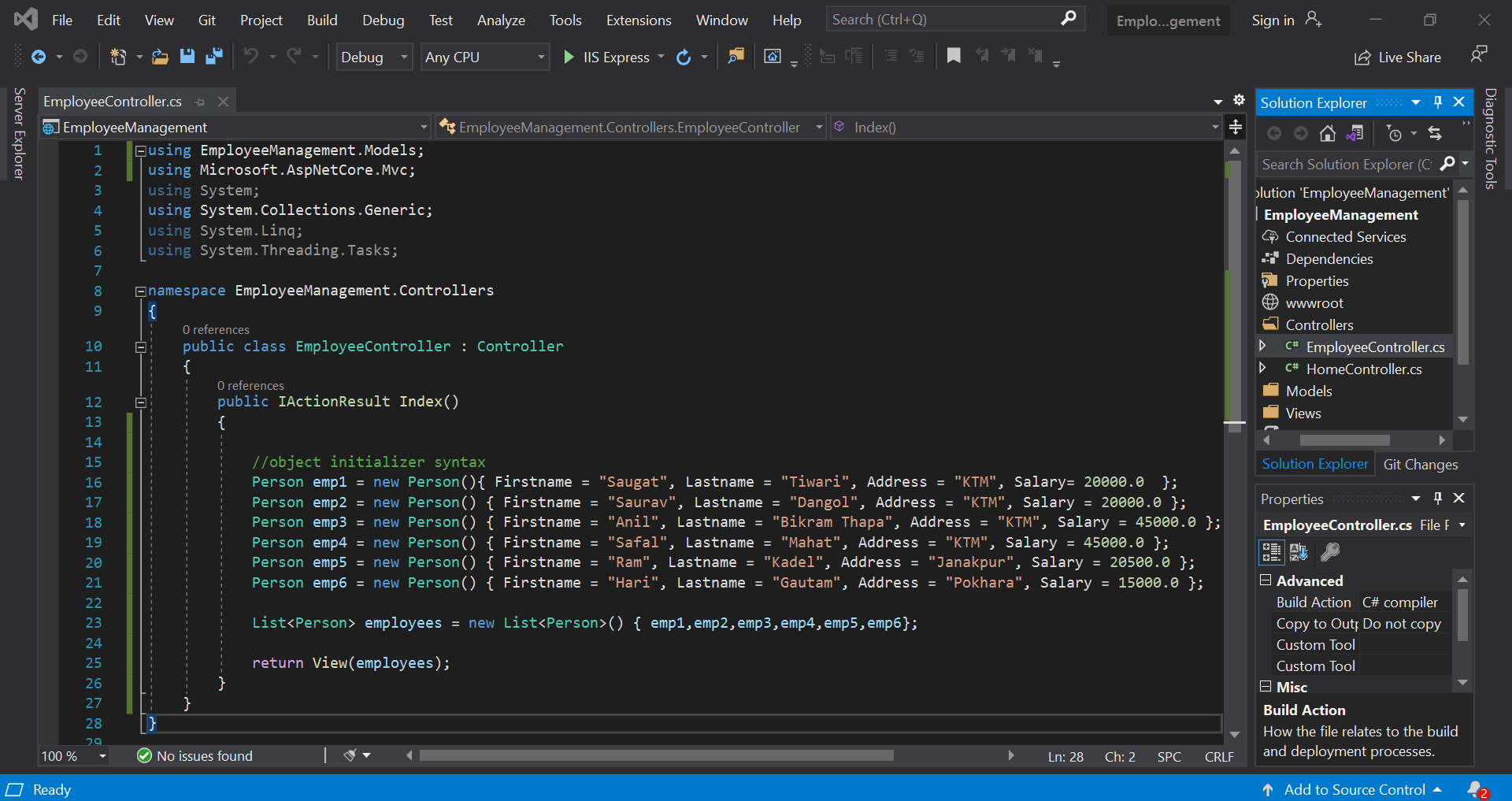
and then select Create.

## **Add a controller**

 In Solution Explorer, right-click Controllers > Add > Controller

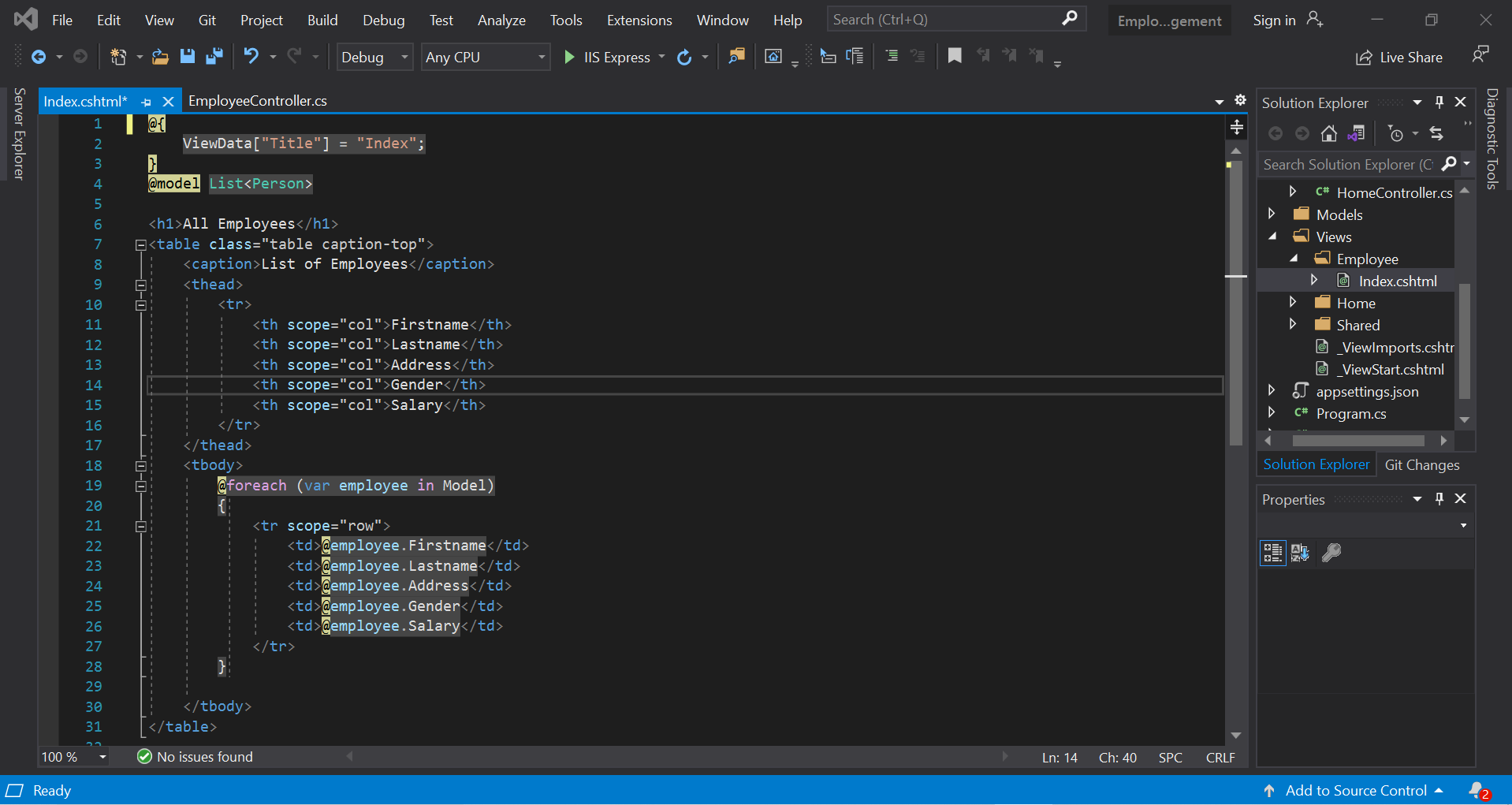
 In the Add Scaffold dialog box, select Controller Class – Empty

 In the Add Empty MVC Controller dialog, enter Controller name and select Add.

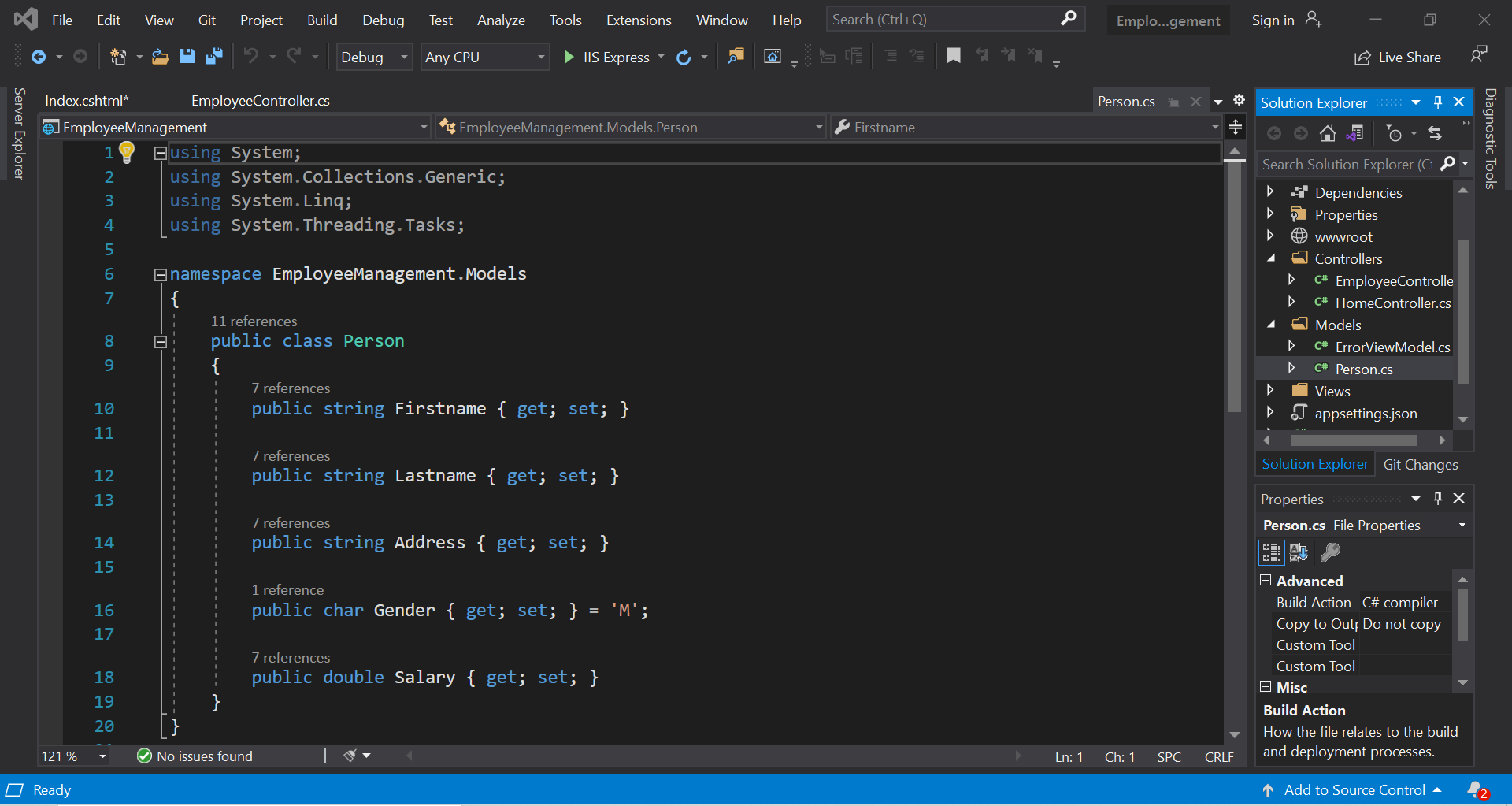


* **Add a view**
* Right-click on the Views folder, and then Add > New Folder and name the folder. Here, in our case, Employee folder
* Right-click on the Views/foldername , i.e. Views/Employee, and then Add > New Item.
* In the Add New Item - Employee dialog:
* In the search box in the upper-right,
* enter view Select Razor View Keep the Name box value, Index.cshtml and Select Add

Replace the contents of the Views/Employee/Index.cshtml Razor view file with the following:



* Add a data model class
* Right-click the Models folder > Add > Class. Name the file Person.cs.
* Update the Person.cs file with the following code:



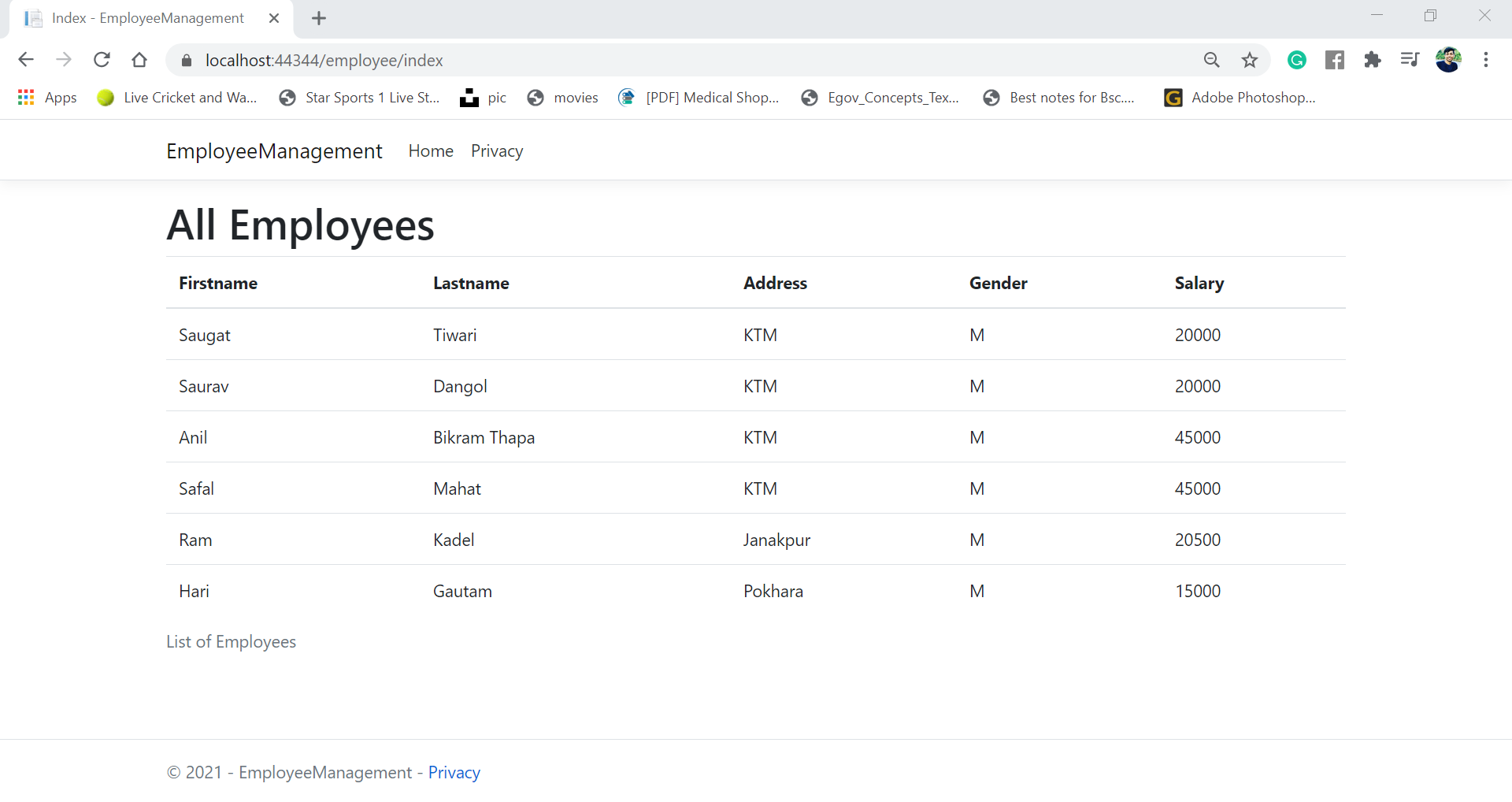
**Result on Chrome Browser:**

Now, to see the result the App,

◦ Select Ctrl-F5 to run the app in non-debug mode, Or

◦ Select IIS Express Button

Then the app is run on web browser and we gave the URL as: /employee/index to see the result.



**Conclusion:**

In this lab of Net Centric Computing, we learned about the MVC framework on ASP.NET core. We successfully created a project called 'EmployeeManagement' and we add a controller named 'EmployeeController.cs' which contains a public class Index in which we initialize the object of List<Person> using object initializer syntax and returns an employee as a object. And then we create a view of the above Index class as 'Index.cshtml' which shows the employees details as a table. Also, we created a model named 'Person.cs' which manages the data come from the controllers.

Thus, we successfully created a simple ASP.NET app which show the list of employee details on the web page using the concept of MVC framework.