

# **Phase-5**

**Name** : Kotadiya Amisha K.

**College** : Government Engineering College, Rajkot

**Branch** : Computer Engineering

## **Understanding**

➤ **Introduction to WEB Dev.**

- ✓ MVC
- ✓ Rest Web API

## ➤ **MVC:**

- ✓ ASP stands for **A**ctive **S**erver **P**ages
- ✓ ASP is a development framework for building web pages.
- ✓ ASP and ASP.NET are server side technologies.
- ✓ Both technologies enable computer code to be executed by an Internet server.

### ✓ **ASP supports many different development models:**

- ✓ Classic ASP
- ✓ ASP.NET Web Forms
- ✓ ASP.NET MVC
- ✓ ASP.NET Web Pages
- ✓ ASP.NET API
- ✓ ASP.NET Core

### ✓ **Model**

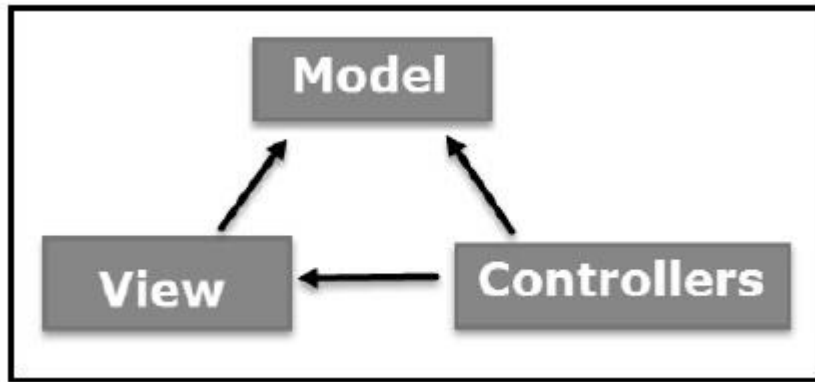
The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

### ✓ **View**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

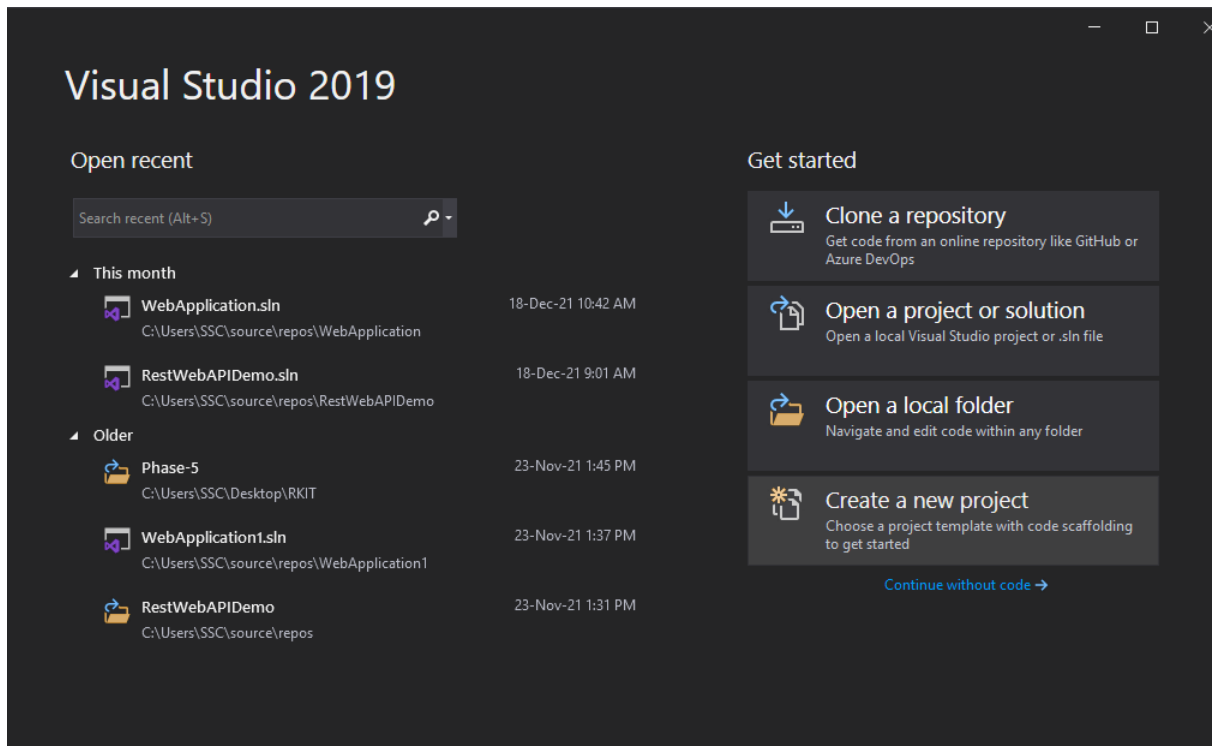
### ✓ **Controller**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

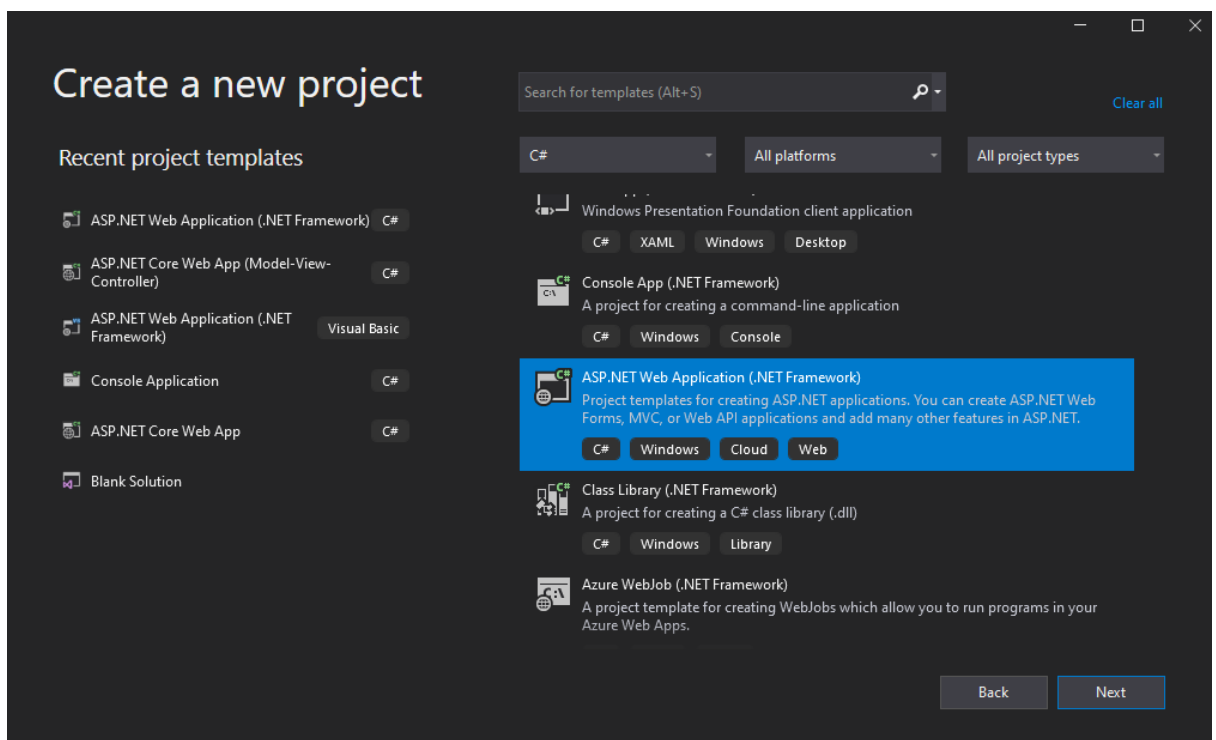


## ➤ Create ASP.NET MVC Application

**Step 1:** Open Visual Studio 2019 and select Create a new project, as shown below.



**Step 2:** Now Select Console Application and Choose Asp.net web application and click on the next button.



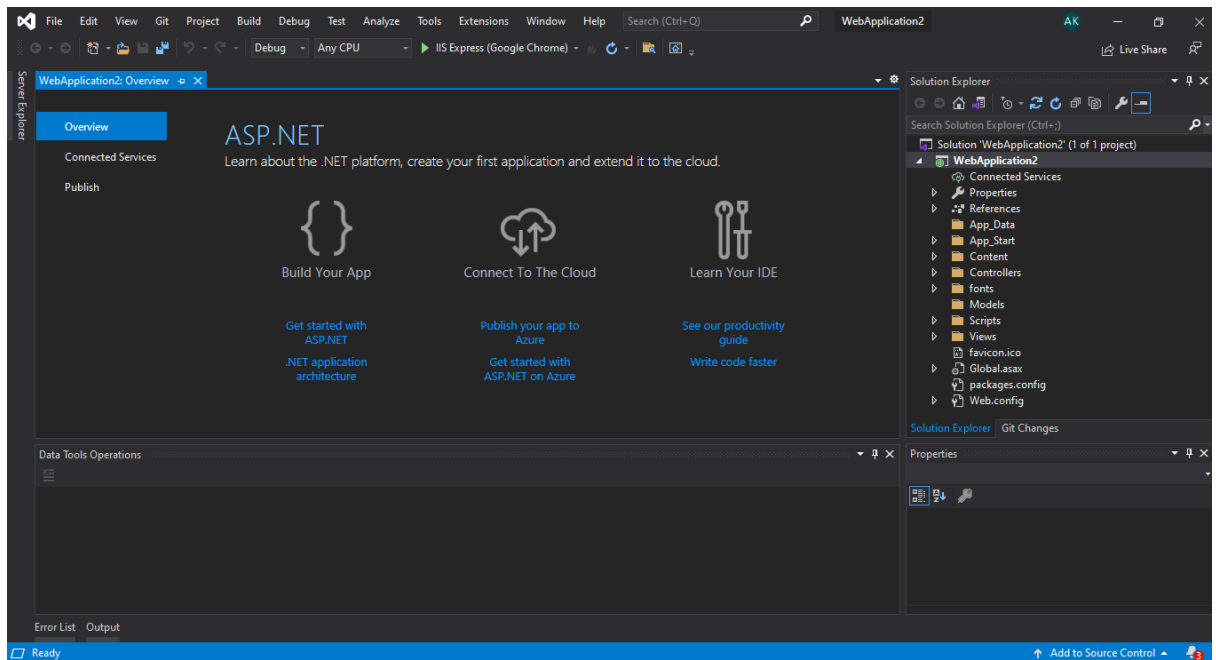
**Step 3:** Now Write the Project name, choose Project Solution, Write Solution name and the Framework.

The screenshot shows the 'Configure your new project' dialog box. At the top, it says 'ASP.NET Web Application (.NET Framework)' with tabs for 'C#', 'Windows', 'Cloud', and 'Web'. The 'Project name' field contains 'WebApplication2'. The 'Location' field shows 'C:\Users\SSC\source\repos'. The 'Solution name' field also contains 'WebApplication2'. There is an unchecked checkbox for 'Place solution and project in the same directory'. The 'Framework' dropdown is set to '.NET Framework 4.7.2'. At the bottom right, there are 'Back' and 'Create' buttons.

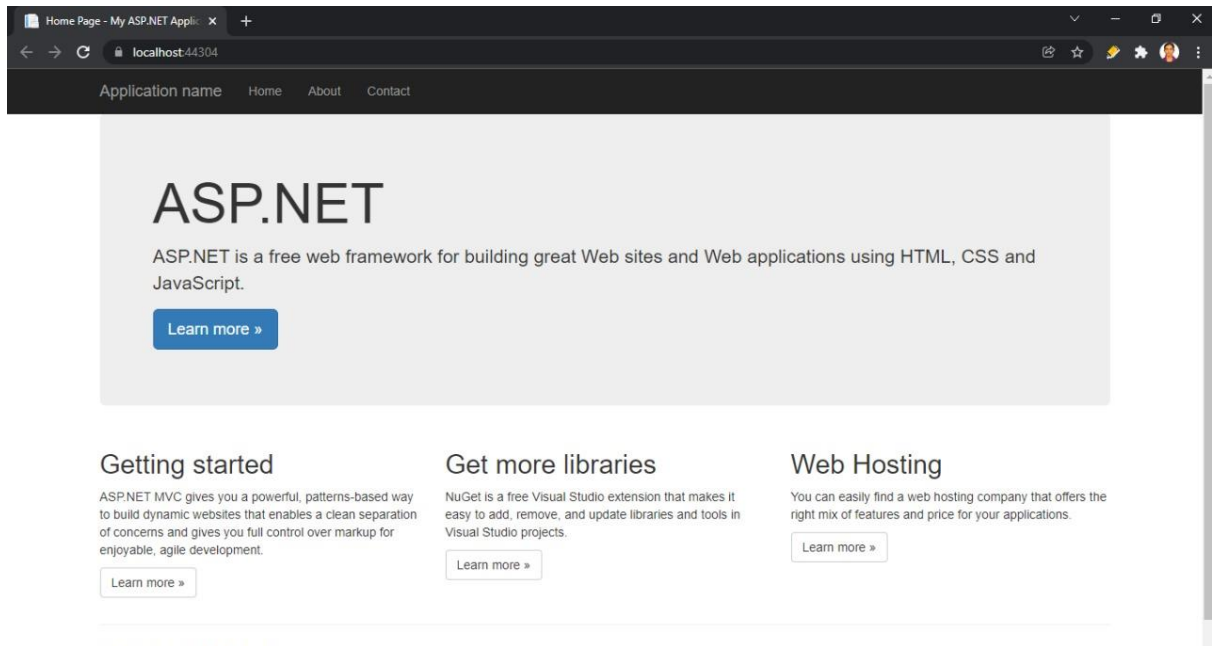
**Step 4:** Now We have to Choose MVC Application. And Click on the Create Button You can also change the authentication by clicking on Change button. You can select appropriate authentication mode for your application.

The screenshot shows the 'Create a new ASP.NET Web Application' dialog box. On the left, there are five project templates: 'Empty', 'Web Forms', 'MVC' (which is highlighted with a blue background), 'Web API', and 'Single Page Application'. Each template has a brief description. On the right, there are three sections: 'Authentication' with 'No Authentication' selected and a 'Change' link; 'Add folders & core references' with checkboxes for 'Web Forms', 'MVC' (checked), and 'Web API'; and 'Advanced' with checkboxes for 'Configure for HTTPS' (checked), 'Docker support' (unchecked), and 'Also create a project for unit tests' (unchecked). Below the 'Advanced' section, there is a text box containing 'WebApplication2.Tests'. At the bottom right, there are 'Back' and 'Create' buttons.

## Step 5: So Here Our Project is Created.

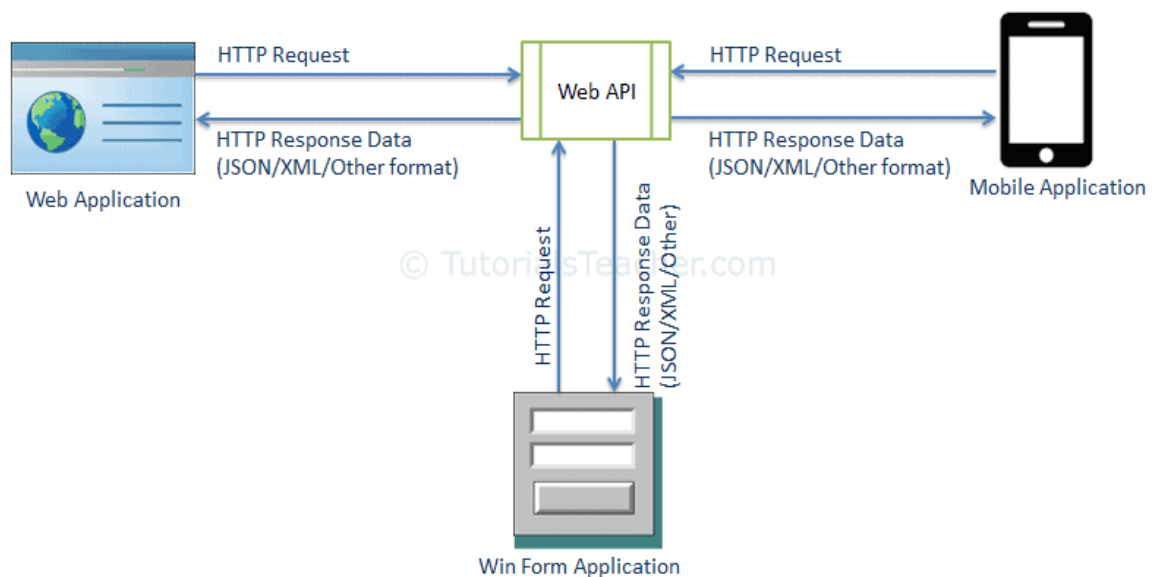


**Step 6:** press F5 to run the project in debug mode or Ctrl + F5 to run the project without debugging.



## ➤ Web API:

- ✓ An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other.
- ✓ A REST(REpresentational State Transfer) API is an API that conforms to the design principles of the REST, or representational state transfer architectural style. For this reason, REST APIs are sometimes referred to RESTful APIs.
- ✓ ASP.NET Web API is a framework for building HTTP services that can be accessed from any client including browsers and mobile devices. It is an ideal platform for building RESTful applications on the .NET Framework.
- ✓ API: Before we understand what is Web API, let's see what is an API (Application Programming Interface).
- ✓ As per Definition of API: In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.
- ✓ To put it in simple terms, API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.
- ✓ ASP.NET Web API: The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc.
- ✓ It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view.
- ✓ It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.





## ✓ **ASP.NET Web API Characteristics**

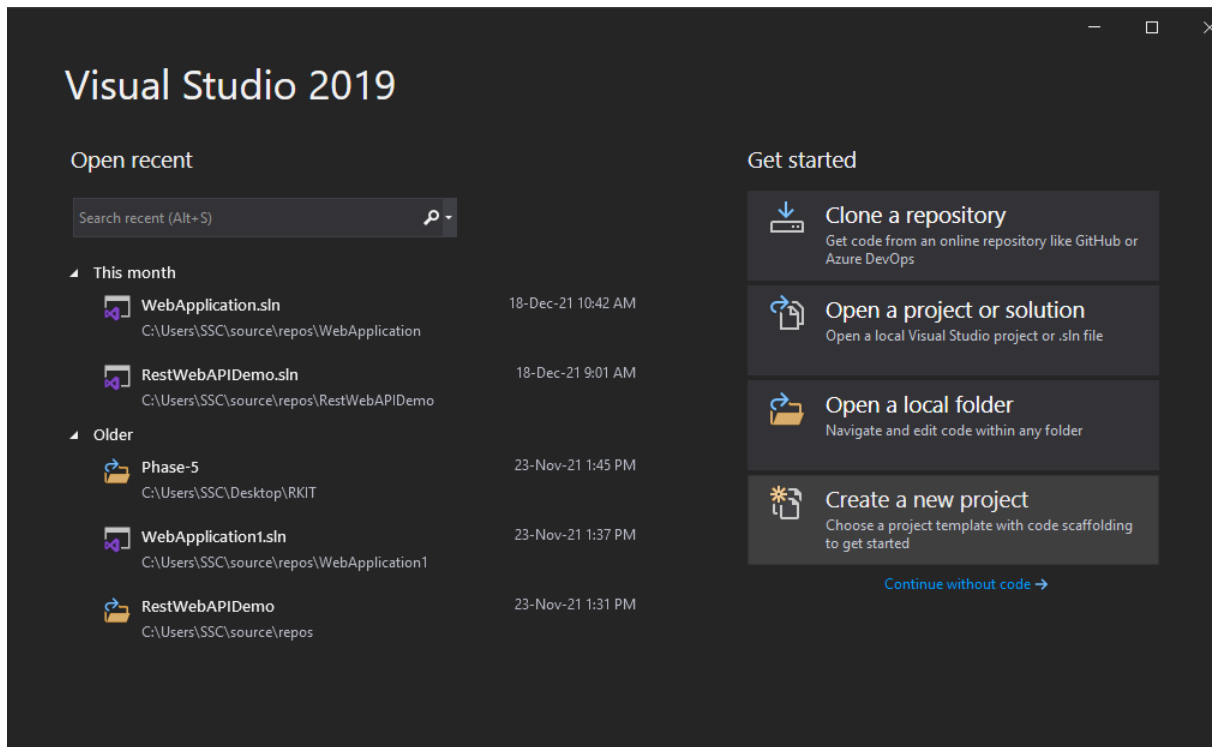
- ✓ ASP.NET Web API is an ideal platform for building RESTful services.
- ✓ ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ✓ ASP.NET Web API maps HTTP verbs to method names.
- ✓ ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
- ✓ ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
- ✓ ASP.NET Web API framework includes new HttpClient to communicate with Web API server.
- ✓ HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

## ➤ **Start with Project**

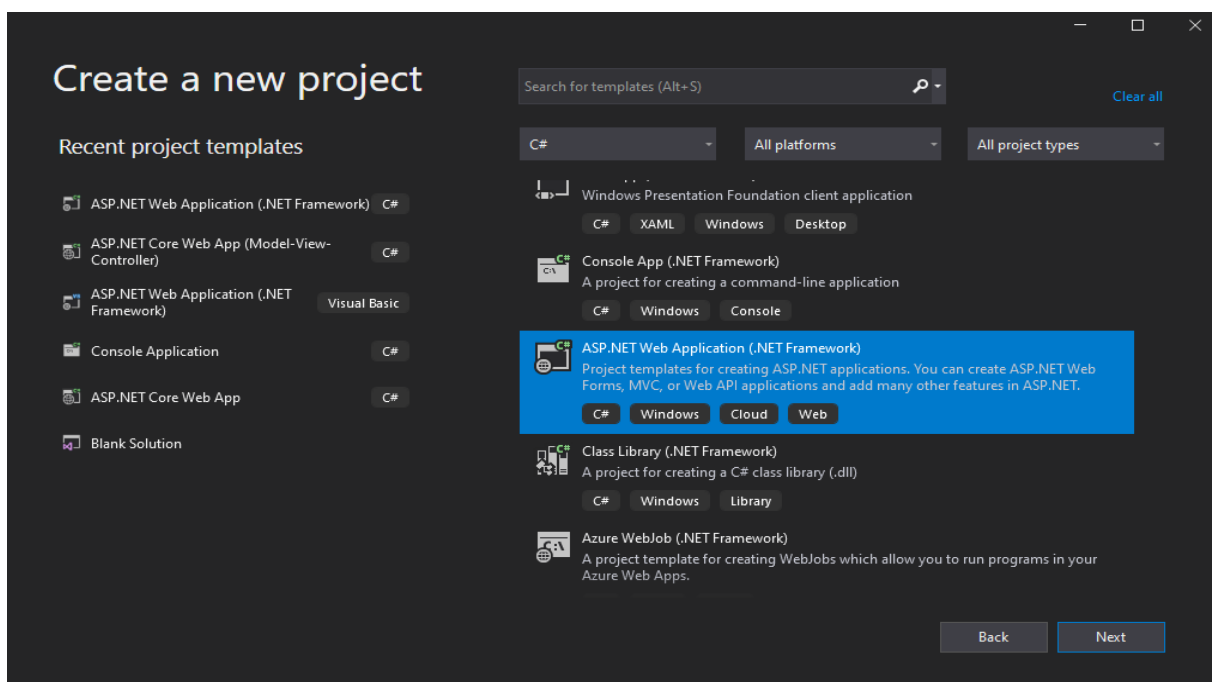
- ✓ Create new Web API Project
- ✓ Setting up infrastructure
- ✓ Create Controller, model
- ✓ Parameters (From URI, From Body)
- ✓ Serialization
- ✓ Routing
- ✓ Config

## ✓ Create new Web API Project

**Step 1:** Open Visual Studio 2019 and select Create a new project, as shown below.



**Step 2:** Now Select Console Application and Choose Asp.net web application and click on the next button.



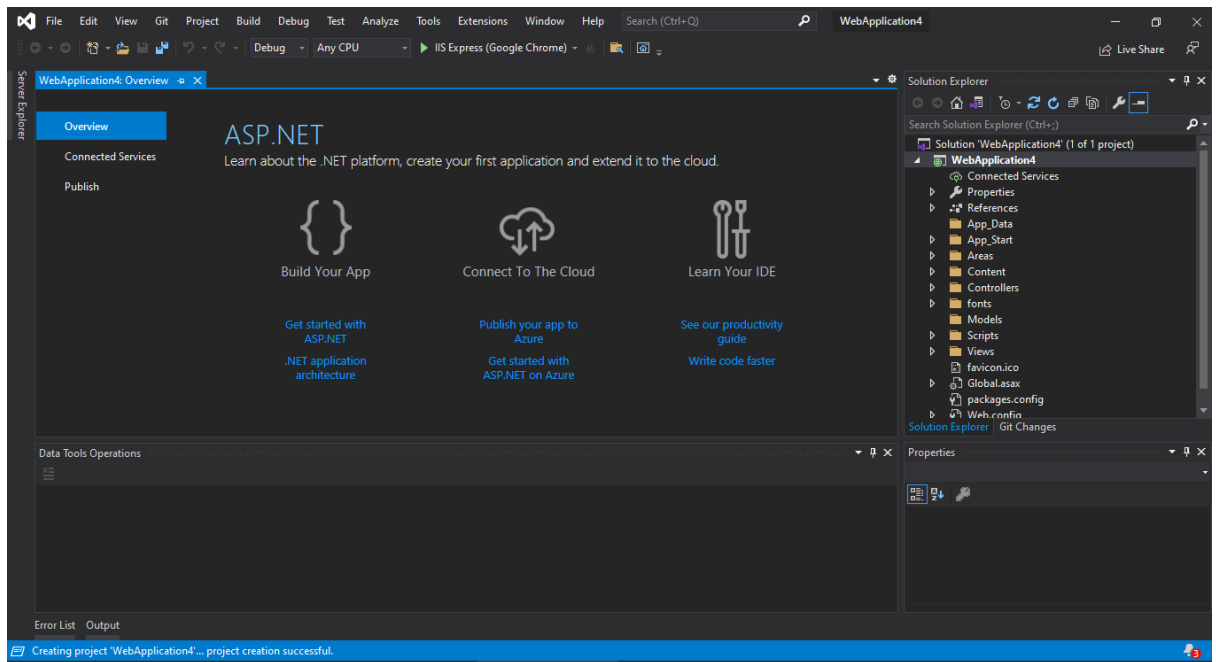
**Step 3:** Now Write the Project name, choose Project Solution, Write Solution name and the Framework.

The screenshot shows the 'Configure your new project' dialog box. At the top, it says 'ASP.NET Web Application (.NET Framework)' with tabs for 'C#', 'Windows', 'Cloud', and 'Web'. The 'Project name' field contains 'WebApplication4'. The 'Location' field shows 'C:\Users\SSC\source\repos'. The 'Solution name' field also contains 'WebApplication4'. There is an unchecked checkbox for 'Place solution and project in the same directory'. The 'Framework' dropdown is set to '.NET Framework 4.7.2'. At the bottom right, there are 'Back' and 'Create' buttons.

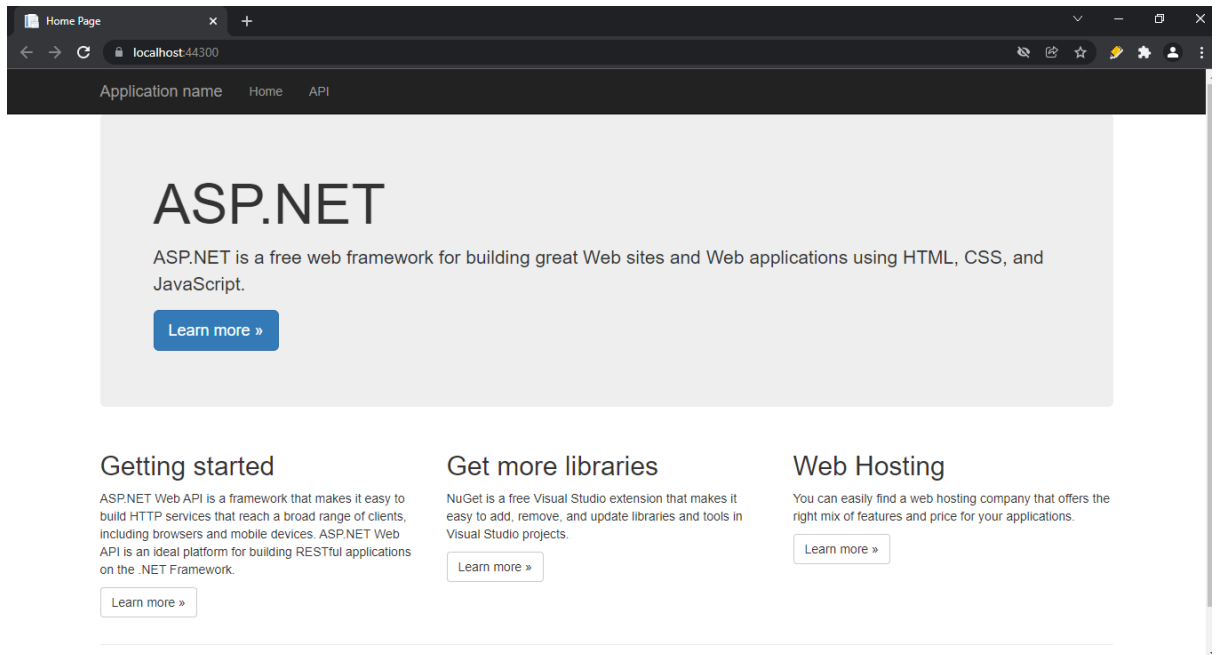
**Step 4:** Now We have to Choose Web API. And Click on the Create Button You can also change the authentication by clicking on Change button. You can select appropriate authentication mode for your application.

The screenshot shows the 'Create a new ASP.NET Web Application' dialog box. On the left, there are five templates: 'Empty', 'Web Forms', 'MVC', 'Web API' (which is highlighted in blue), and 'Single Page Application'. On the right, there are three sections: 'Authentication' (set to 'No Authentication' with a 'Change' link), 'Add folders & core references' (with checkboxes for 'Web Forms', 'MVC', and 'Web API', where 'MVC' and 'Web API' are checked), and 'Advanced' (with checkboxes for 'Configure for HTTPS', 'Docker support', and 'Also create a project for unit tests', where 'Configure for HTTPS' is checked). Below the 'Advanced' section, there is a text box containing 'WebApplication4.Tests'. At the bottom right, there are 'Back' and 'Create' buttons.

## Step 5: So Here Our Project is Created.



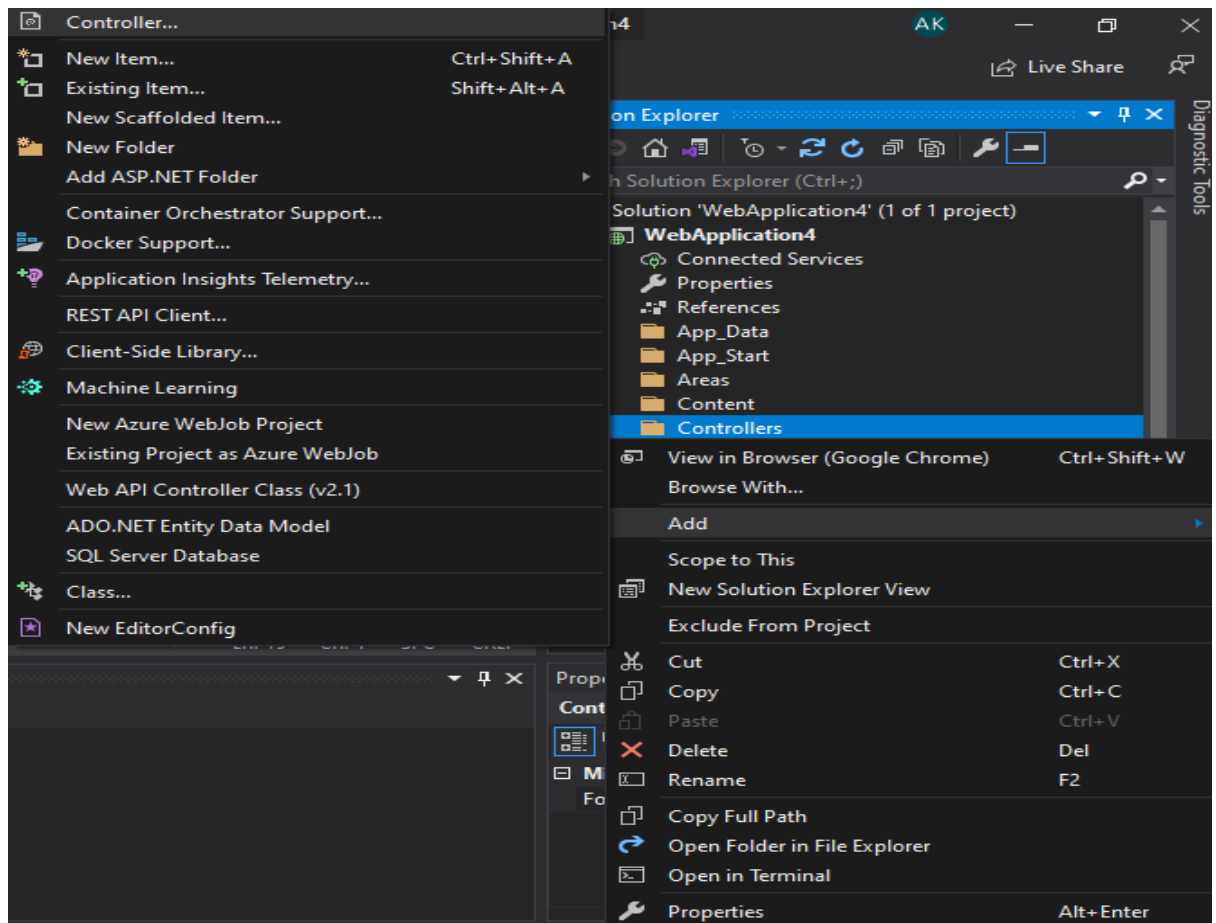
## Step 6: press F5 to run the project in debug mode or Ctrl + F5 to run the project without debugging.



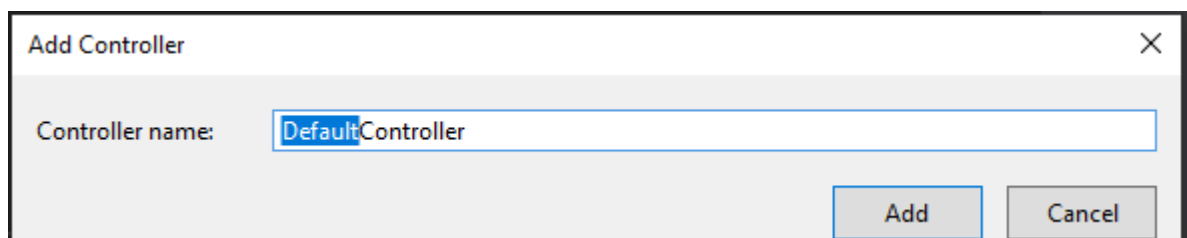
## ➤ Create Controller, model

### ✓ Create Controller:

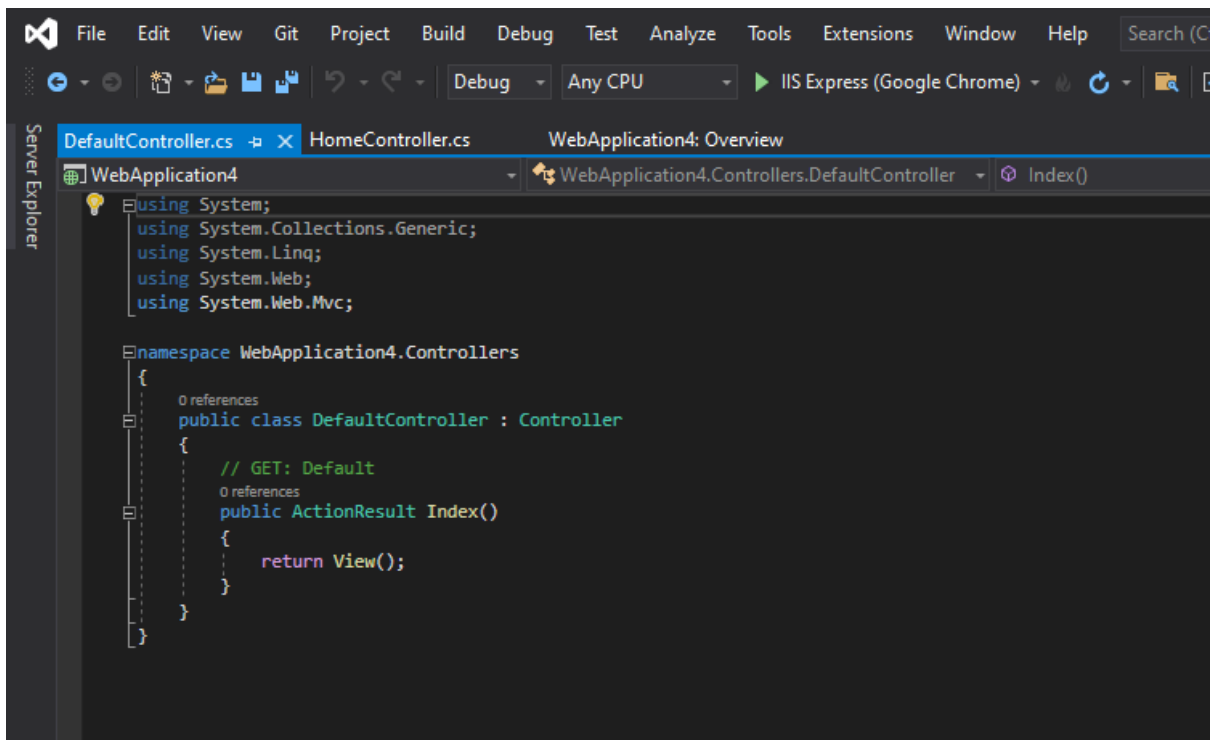
Step:1 We can create controller for the application by adding a new item into the controller folder. Just right click on the controller folder and click add -> controller as given below.



Step:2 Providing controller name, then click Add.



Step:3 After adding this controller, as per the conventions project will create a folder with the same name as the controller name in the view folder to store the view files belongs to the controller. This controller contains the default code like below.

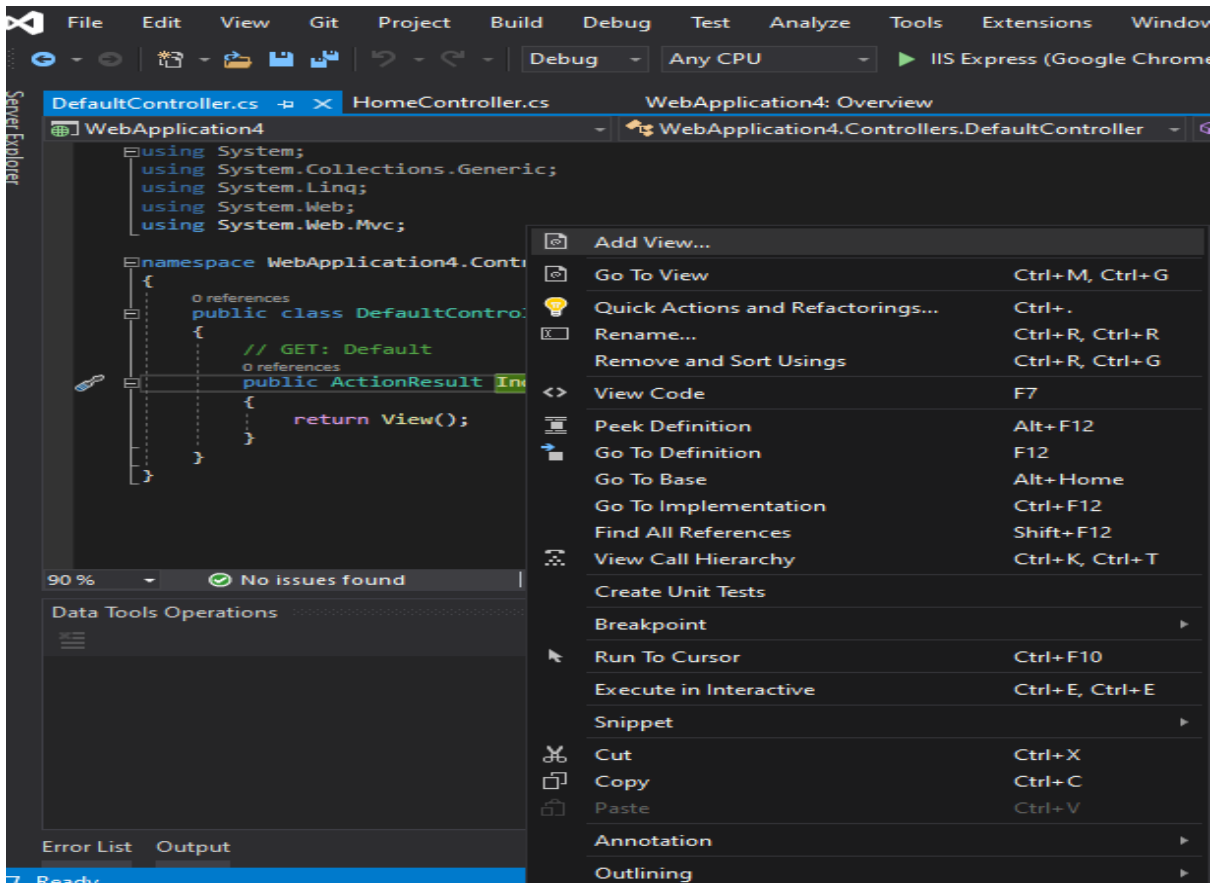


The screenshot shows the Visual Studio IDE with the 'DefaultController.cs' file open. The file is located in the 'WebApplication4.Controllers' namespace. The code includes several using statements at the top: 'using System;', 'using System.Collections.Generic;', 'using System.Linq;', 'using System.Web;', and 'using System.Web.Mvc;'. Below these, the 'namespace WebApplication4.Controllers' is defined, containing a 'public class DefaultController : Controller'. Inside the class, there is a comment '// GET: Default' above a 'public ActionResult Index()' method. The method body consists of a single line 'return View();'.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication4.Controllers
{
    public class DefaultController : Controller
    {
        // GET: Default
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Step:4 After creating view. Right click on Index -> Add View method.

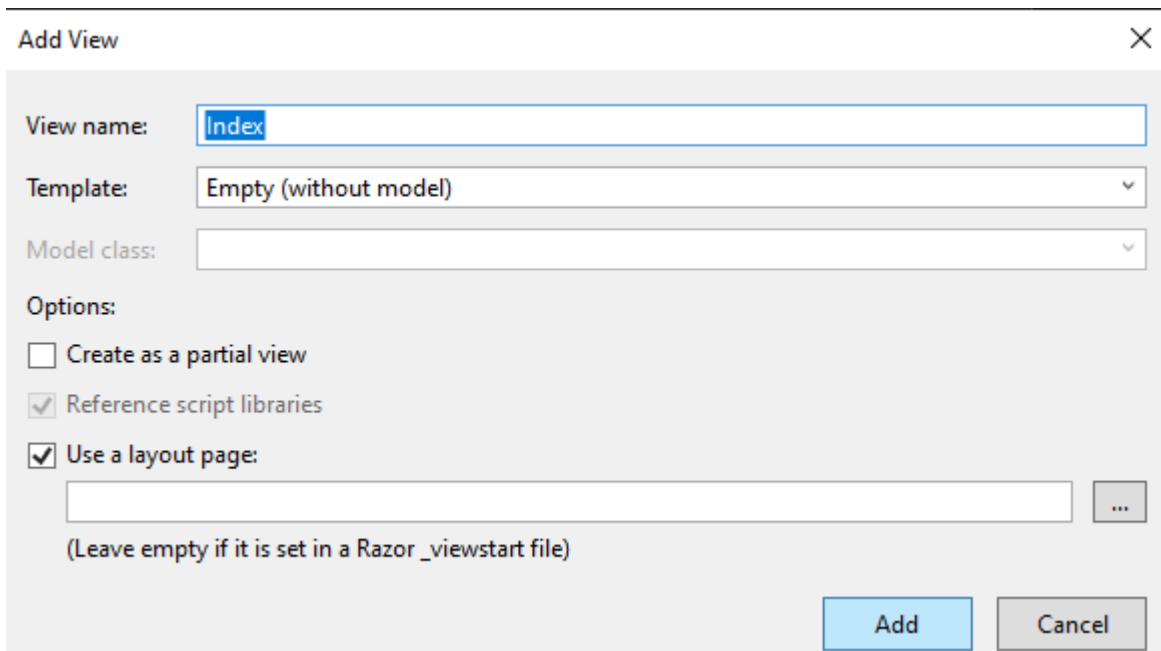


This screenshot shows the same Visual Studio IDE as the previous one, but with a right-click context menu open over the 'Index' method. The menu lists various actions such as 'Add View...', 'Go To View', 'Quick Actions and Refactorings...', 'Rename...', 'Remove and Sort Usings', 'View Code', 'Peek Definition', 'Go To Definition', 'Go To Base', 'Go To Implementation', 'Find All References', 'View Call Hierarchy', 'Create Unit Tests', 'Breakpoint', 'Run To Cursor', 'Execute in Interactive', 'Snippet', 'Cut', 'Copy', 'Paste', 'Annotation', and 'Outlining'. Each item is accompanied by its respective keyboard shortcut. The 'Add View...' option is at the top of the menu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication4.Controllers
{
    public class DefaultController : Controller
    {
        // GET: Default
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Step:5 Then click add button.

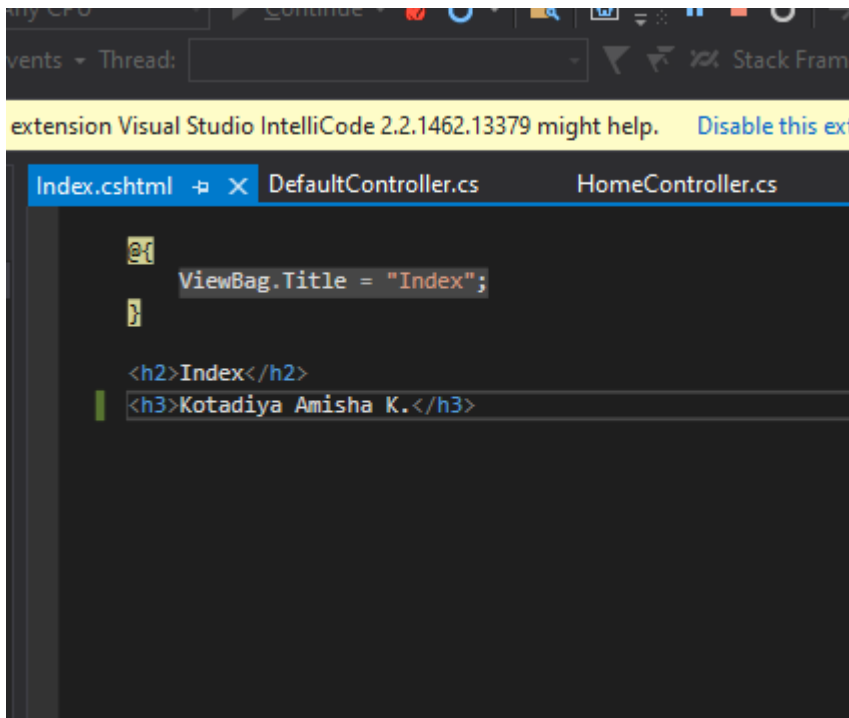


The 'Add View' dialog box is shown with the following fields and options:

- View name:** Index
- Template:** Empty (without model)
- Model class:** (empty)
- Options:**
  - ☐ Create as a partial view
  - ☒ Reference script libraries
  - ☒ Use a layout page:
    - (Leave empty if it is set in a Razor \_viewstart file)

Buttons: Add, Cancel

Step:6 To access this controller to the browser, we are adding an index file to the default folder inside the view folder. This index file contains the following code.

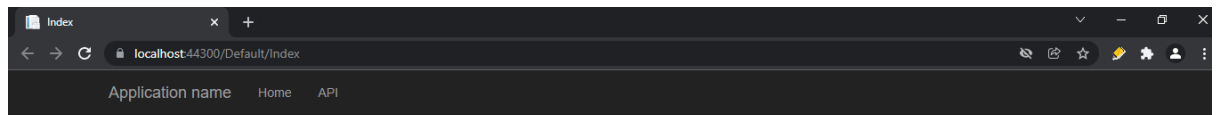


The code editor shows the following code in the `Index.cshtml` file:

```
<@if  
    ViewBag.Title = "Index";  
</@if>  
  
<h2>Index</h2>  
<h3>Kotadiya Amisha K.</h3>
```



Output:



Index

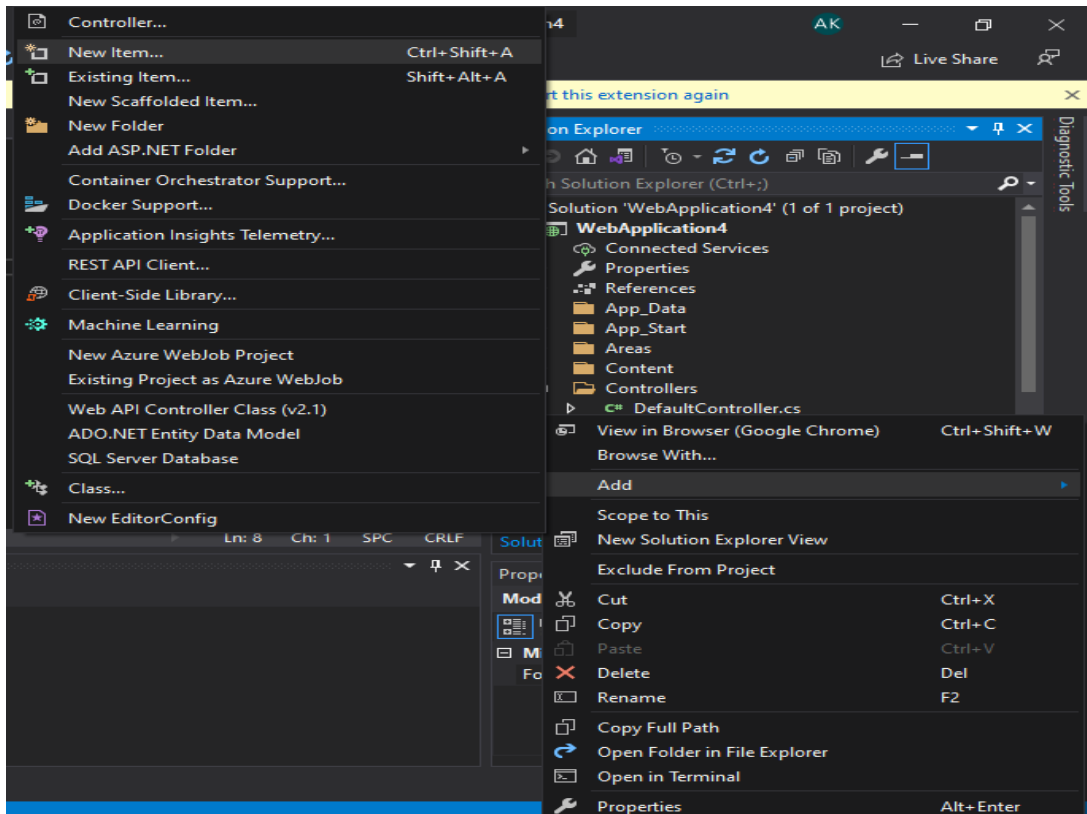
Kotadiya Amisha K.

© 2022 - My ASP.NET Application

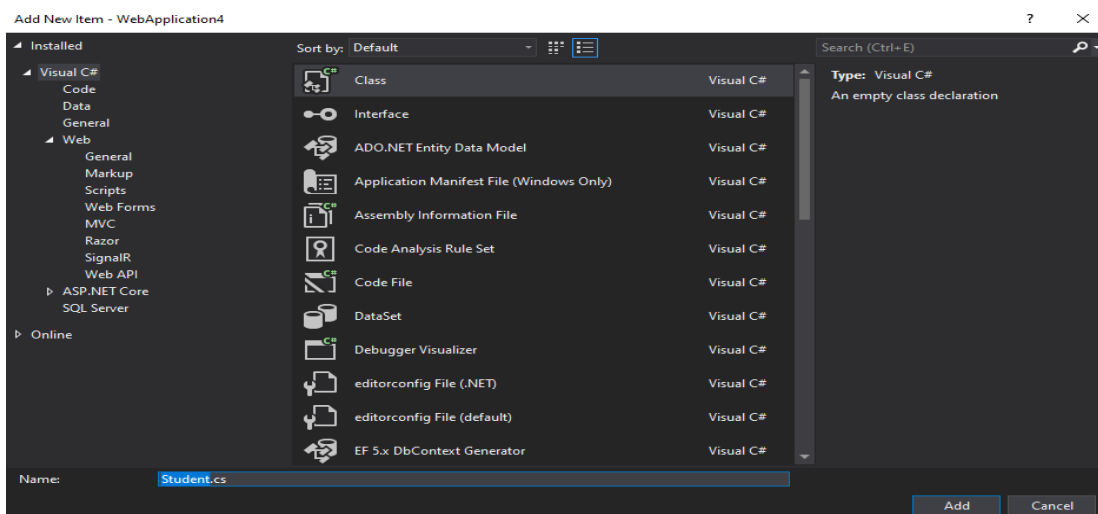
## ✓ Create Model:

Let's create the model class that should have the required properties for the Student entity.

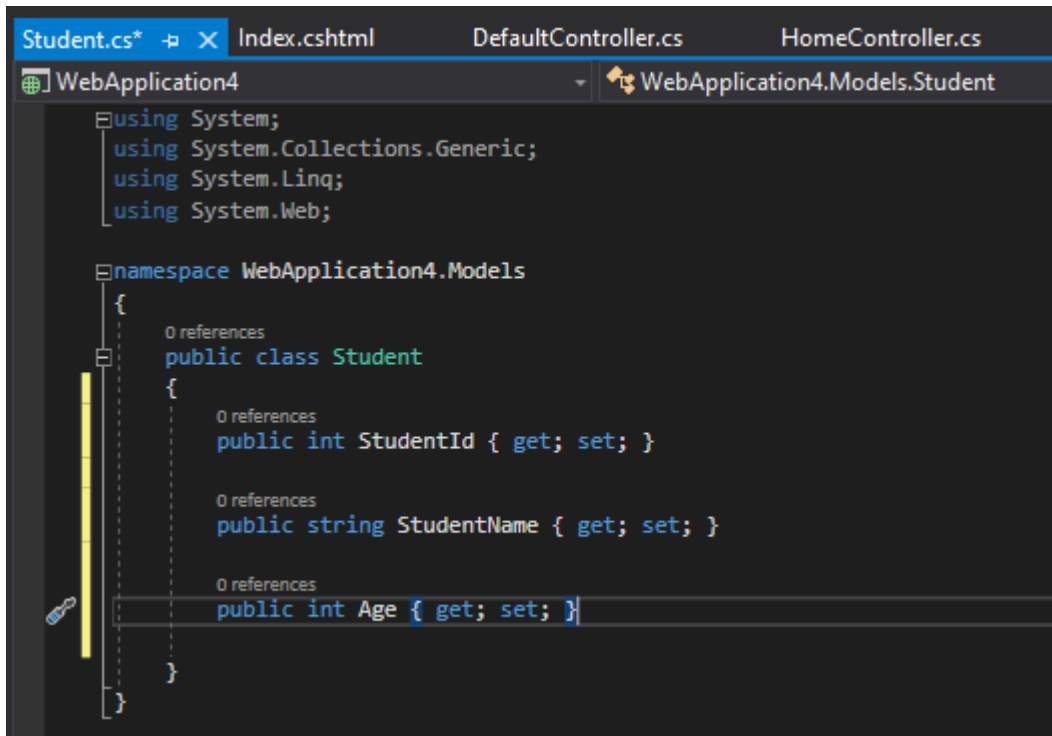
In the MVC application in Visual Studio, and right-click on the Model folder, select Add -> and click on New Item.



It will open the Add New Item dialog box. In the Add New Item dialog box, enter the class name Student and click Add.



This will add a new Student class in model folder. We want this model class to store id, name, and age of the students. So, we will have to add public properties for Id, Name, and Age, as shown below.



```
Student.cs*  Index.cshtml  DefaultController.cs  HomeController.cs
WebApplication4
  WebApplication4.Models.Student
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Web;

    namespace WebApplication4.Models
    {
        0 references
        public class Student
        {
            0 references
            public int StudentId { get; set; }

            0 references
            public string StudentName { get; set; }

            0 references
            public int Age { get; set; }
        }
    }
```

The model class can be used in the view to populate the data, as well as sending data to the controller.

## ➤ **Parameters (From URI, From Body)**

- ✓ When Web API calls a method on the controller it must set parameters by a process called parameter binding.
- ✓ So for simple datatypes like int, bool, double, decimal, string Web API tries to get values from URI by default.
- ✓ For complex datatypes like user defined objects it takes Web API tries to read value from message body.
- ✓ These are default cases but these methods can be overridden by for getting complex data from URI itself and for getting simple data from the message body.
- ✓ These are used just to override in given conditions otherwise in default case other methods can be used.

## ➤ **Serialization**

- ✓ As the name suggest it is used to convert object in to the JSON stream format.
- ✓ We can convert object in JSON stream format and also can get back that object from JSON stream using the concept of the serialization and de-serialization.
- ✓ The quickest method of converting between JSON text and a .NET object is using the JsonSerializer.
- ✓ The JsonSerializer converts .NET object into their JSON equivalent and back again by mapping the .NET object property names from the JSON property names and copies the values for you.

## ➤ **Routing**

### **Web API Routing:**

- ✓ In ASP.NET Web API, a controller is a class that handles HTTP requests. The public methods of the controller are called action methods or simply actions. When the Web API framework receives a request, it routes the request to an action.
- ✓ This route is defined in the WebApiConfig.cs file, which is placed in the App\_Start directory.

Web API supports two types of routing:

1. Convention-based Routing
2. Attribute Routing

#### **1. Convention-based Routing**

In the convention-based routing, Web API uses route templates to determine which controller and action method to execute. At least one route template must be added into route table in order to handle various HTTP requests.

When we created Web API project using WebAPI template in the Create Web API Project section, it also added WebApiConfig class in the App\_Start folder with default route as shown below.

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        // Enable attribute routing
        config.MapHttpAttributeRoutes();

        // Add default route using convention-based routing
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

```

    );
}
}

```

In the above WebApiConfig.Register() method, config.MapHttpAttributeRoutes() enables attribute routing. The config.Routes is a route table. The "DefaultApi" route is added in the route table using MapHttpRoute() extension method.

You can configure multiple routes in the Web API using HttpConfiguration object. The following example demonstrates configuring multiple routes.

```

public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        // Enable attribute routing
        config.MapHttpAttributeRoutes();

        // WebAPI route
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/WebAPI/{id}",
            defaults: new { controller = "WebAPI",
                           id = RouteParameter.Optional }
        );

        // Default route
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}

```

## 2. Attribute Routing

Attribute routing is supported in Web API 2. As the name implies, attribute routing uses [Route()] attribute to define routes. The Route attribute can be applied on any controller or action method.

In order to use attribute routing with Web API, it must be enabled in WebApiConfig by calling config.MapHttpAttributeRoutes() method.

Consider the following example of attribute routing.

```
// Use Route attribute
[Route("api/WebAPI/{id}/courses")]

// GET api/<controller>/<id>/courses
public IEnumerable<string> GetCourses(int id)
{
    if (id == 1)
        return new List<String>() { "C", "ASP.NET", "JAVA" };
    else if (id == 2)
        return new List<String>() { "C#", "PHP", "ANDROID" };
    else
        return new List<String>() { "C++", "HTML", "SQL" };
}
```

## ➤ **Config**

- ✓ A configuration file (web.config) is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code.
- ✓ In this way you can configure settings independently from your code.
- ✓ Generally, a website contains a single Web.config file stored inside the application root directory. The behavior of an ASP.NET application is affected by different settings in the configuration files: o machine.config o web.config
- ✓ The machine.config file contains default and the machine-specific value for all supported settings.
- ✓ The machine settings are controlled by the system administrator and applications are generally not given access to this file.
- ✓ An application however, can override the default values by creating web.config files in its roots folder.
- ✓ The web.config file is a subset of the machine.config file.
- ✓ If the application contains child directories, it can define a web.config file for each folder. Scope of each configuration file is determined in a hierarchical top-down manner