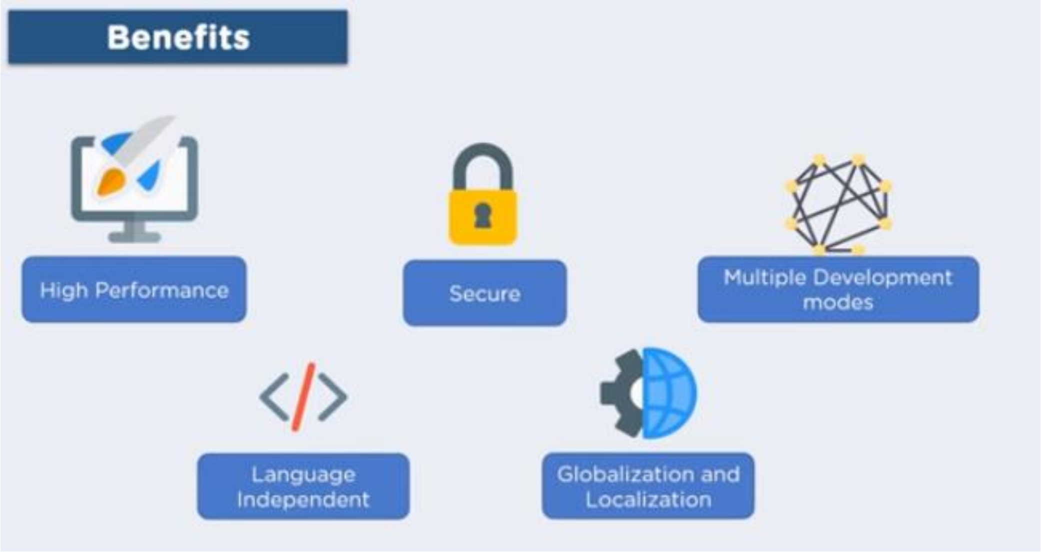
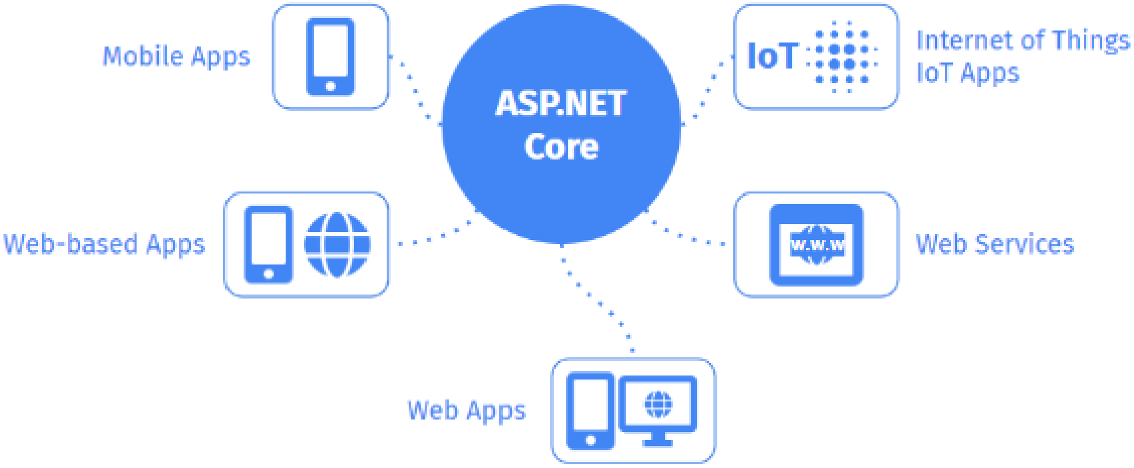
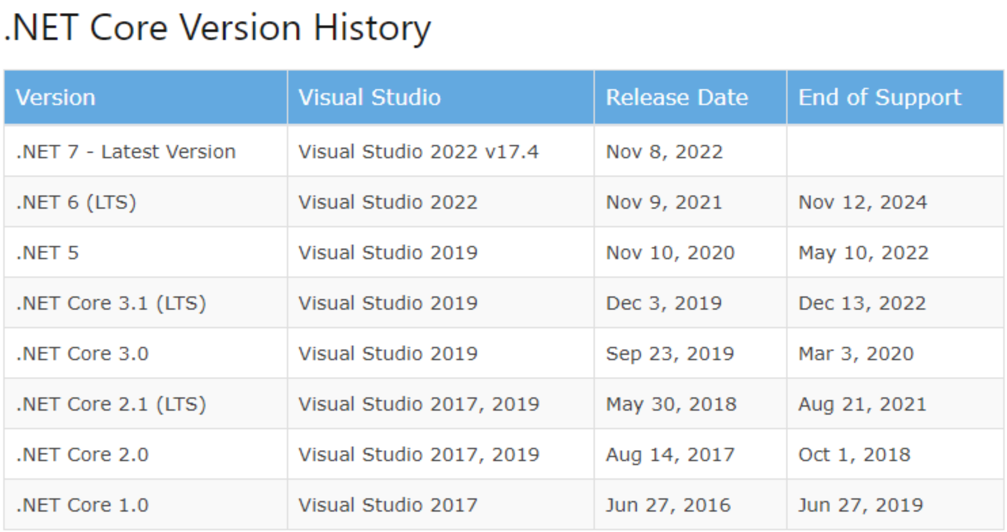
What is ASP.NET Core?

ASP.NET Core is an open-source modular web-application framework. It is a redesign of ASP.NET that unites the previously separate ASP.NET MVC and ASP.NET Web API into a single programming model.







.NET Core Version History

Version	Visual Studio	Release Date	End of Support
.NET 7 - Latest Version	Visual Studio 2022 v17.4	Nov 8, 2022	
.NET 6 (LTS)	Visual Studio 2022	Nov 9, 2021	Nov 12, 2024
.NET 5	Visual Studio 2019	Nov 10, 2020	May 10, 2022
.NET Core 3.1 (LTS)	Visual Studio 2019	Dec 3, 2019	Dec 13, 2022
.NET Core 3.0	Visual Studio 2019	Sep 23, 2019	Mar 3, 2020
.NET Core 2.1 (LTS)	Visual Studio 2017, 2019	May 30, 2018	Aug 21, 2021
.NET Core 2.0	Visual Studio 2017, 2019	Aug 14, 2017	Oct 1, 2018
.NET Core 1.0	Visual Studio 2017	Jun 27, 2016	Jun 27, 2019

.NET 5 unifies the previously separate .NET Core, .NET Framework, and Xamarin platforms, making it easier for developers to use a single platform for various application types.

ASP.NET Core vs. ASP.NET MVC: Choosing the Right Framework For Your Project?

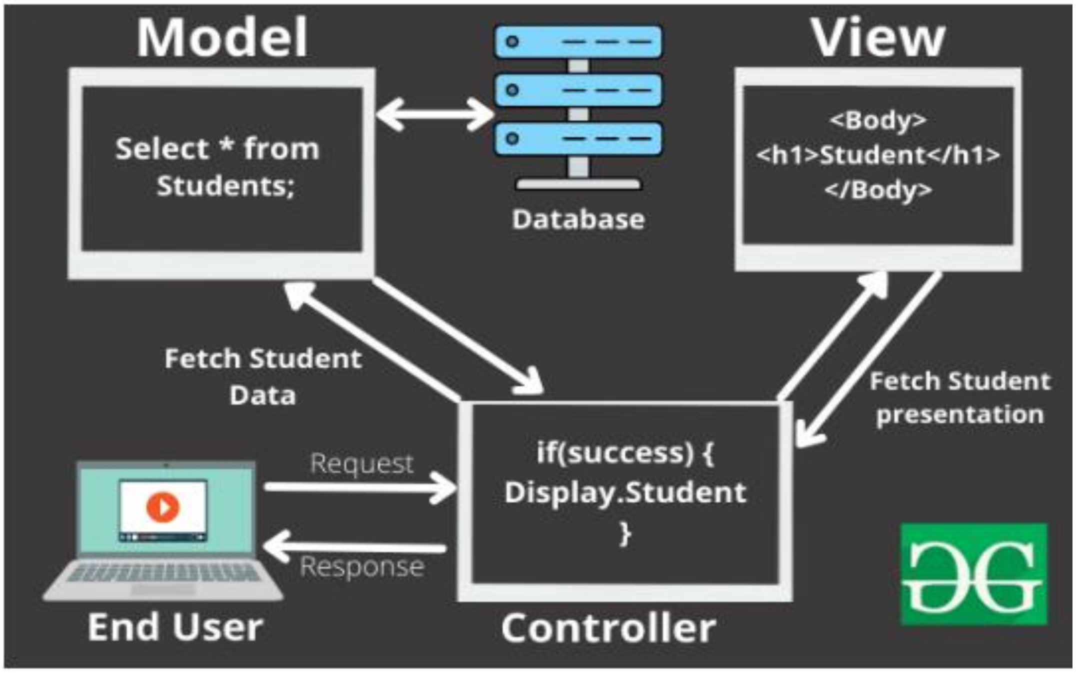
- ASP.NET Core is a new framework that is more portable and lightweight than ASP.NET MVC. It uses for developing web apps that run on Linux, Windows, and macOS.
- Conversely, ASP.NET MVC is an older performance intended to be considered with .NET Framework

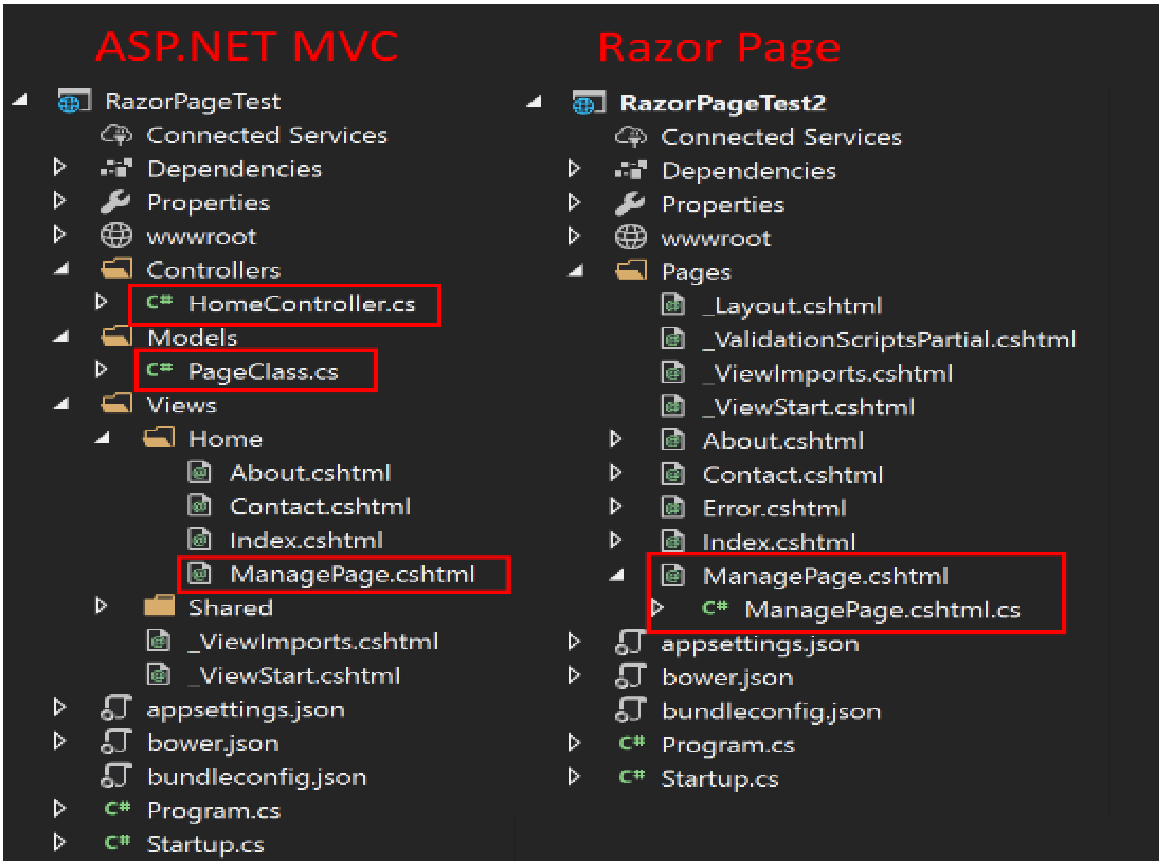
What is MVC?

MVC stands for model-view-controller. Here's what each of those components mean:

- **Model:** The backend that contains all the data logic
  - *Whether the data is from a database, API, or a JSON object, the model is responsible for managing it.*
- **View:** The frontend or graphical user interface (GUI)
  - *Through getter and setter functions, the controller pulls data from the model and initializes the views.*
- **Controller:** The brains of the application that controls how data is displayed

- *Through getter and setter functions, the controller pulls data from the model and initializes the views.*




ASP.NET MVC

```
◢ RazorPageTest
      Connected Services
   ▷ Dependencies
   ▷ Properties
   ▷ wwwroot
   ◢ Controllers
      ▷ HomeController.cs
   ◢ Models
      ▷ PageClass.cs
   ◢ Views
      ◢ Home
            About.cshtml
            Contact.cshtml
            Index.cshtml
            ManagePage.cshtml
      ▷ Shared
         _ViewImports.cshtml
         _ViewStart.cshtml
   ▷ appsettings.json
   ▷ bower.json
      bundleconfig.json
   ▷ Program.cs
   ▷ Startup.cs
```

Razor Page

```
◢ RazorPageTest2
      Connected Services
   ▷ Dependencies
   ▷ Properties
   ▷ wwwroot
   ◢ Pages
         _Layout.cshtml
         _ValidationScriptsPartial.cshtml
         _ViewImports.cshtml
         _ViewStart.cshtml
      ▷ About.cshtml
      ▷ Contact.cshtml
      ▷ Error.cshtml
      ▷ Index.cshtml
      ◢ ManagePage.cshtml
         ▷ ManagePage.cshtml.cs
   ▷ appsettings.json
   ▷ bower.json
      bundleconfig.json
   ▷ Program.cs
   ▷ Startup.cs
```

MVC Frameworks

- JavaScript has grown in popularity, and it's taken over the backend in recent years. More and more full-blown JavaScript applications have opted for the MVC architecture pattern in one way or another.
- Frameworks come and go, but what has been constant are the concepts borrowed from the MVC architecture pattern.
- Some of the early frameworks that applied these concepts were KnockoutJS, Django, and Ruby on Rails.

ASP.NET Core vs ASP.NET MVC

Both ASP.NET Core and ASP.NET MVC are Microsoft web app development frameworks. However, they have the following differences in some contexts:

1. Cross-Platform Support
   ASP.NET Core is an open-source, cross-platform framework that can be considered for developing apps. It runs on MacOS, Windows, and Linux. On the other side, the ASP.NET MVC framework runs only on Windows.

2. Dependency Injection (DI)
   ASP.NET Core focuses on dependency injection by default, making testing and maintaining web apps simple. In contrast, ASP.NET MVC does not consider dependency injection by default. However, it can be included in web apps.

3. Razor Pages
   ASP.NET Core added the new Razor Page framework, making developing single-page and simple web apps easier. In comparison, ASP.NET MVC does not add the Razor Pages.

4. Performance
   ASP.NET Core is commonly faster than ASP.NET MVC.

5. When you are developing a new web application project, it is recommended to use ASP.NET Core. Its highly lightweight, portable, and newer framework will provide you with highly flexible options for your project.

ASP.NET Core Tutorial
https://www.educative.io/blog/aspnet-core-tutorial

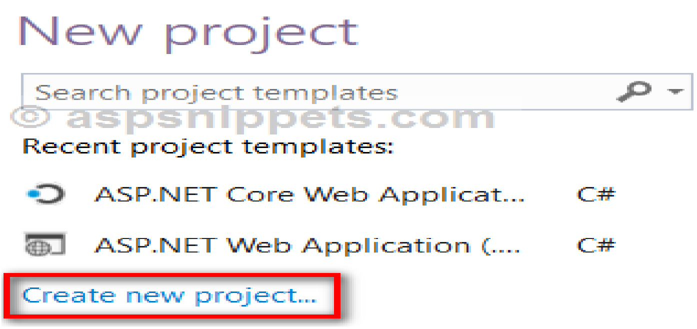
DEVELOPING WEBSITE USING RAZOR PAGES DESIGN PATTERNS

- Razor refers to the markup syntax allowing coders to embed server-based code. You can add code of C# and Visual Basic to the web pages.
- *Using the server-based code, it becomes possible to build dynamic web content*. While coders write the web page to the browser.

Creating a new ASP.Net Core 2.0 Project

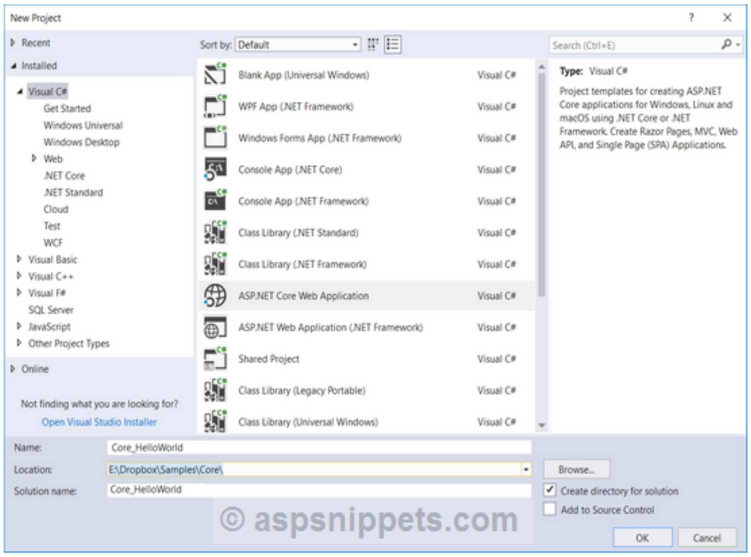
Let's get started with creating your first ASP.Net Core 2.0 Project.

1. Open Visual Studio and from Start section click on Create New Project.



New project
Search project templates
© aspsnippets.com
Recent project templates:
ASP.NET Core Web Applicat...   C#
ASP.NET Web Application (....   C#
Create new project...

2. From the New Project Dialog window, select ASP.Net Core Web Application option. Then you need to set a suitable Name for your project and also you can set its Location where you want to create the Project.



New Project
?
Recent
Installed
Visual C#
Get Started
Windows Universal
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual C++
Visual F#
SQL Server
JavaScript
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows)   Visual C#
WPF App (.NET Framework)   Visual C#
Windows Forms App (.NET Framework)   Visual C#
Console App (.NET Core)   Visual C#
Console App (.NET Framework)   Visual C#
Class Library (.NET Standard)   Visual C#
Class Library (.NET Framework)   Visual C#
ASP.NET Core Web Application   Visual C#
ASP.NET Web Application (.NET Framework)   Visual C#
Shared Project   Visual C#
Class Library (Legacy Portable)   Visual C#
Class Library (Universal Windows)   Visual C#
Browse...
OK
Cancel
© aspsnippets.com

Type: Visual C#

Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.

Name: Core_HelloWorld

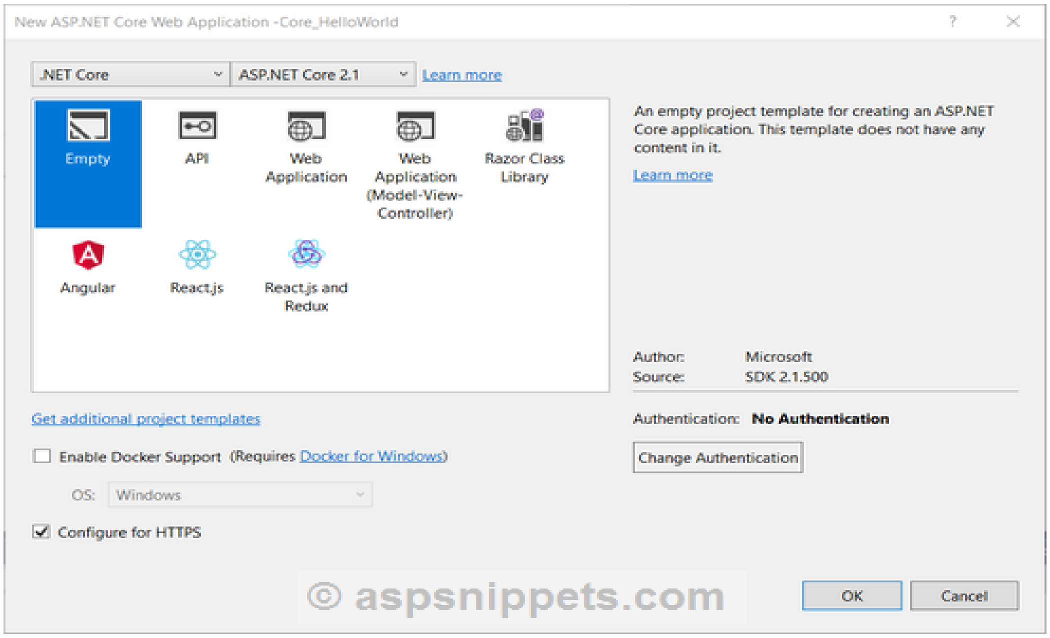
Location: E:\Dropbox\Samples\Core\

Solution name: Core_HelloWorld

☑ Create directory for solution

□ Add to Source Control

3. From the New ASP.Net Core Web Application Dialog, select Empty and click OK.



New ASP.NET Core Web Application -Core_HelloWorld

.NET Core | ASP.NET Core 2.1 | Learn more

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

Learn more

Author: Microsoft
Source: SDK 2.1.500

Get additional project templates

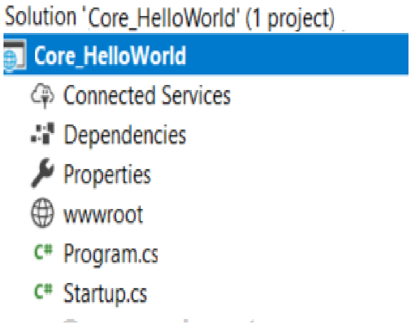
Authentication: **No Authentication**

☐ Enable Docker Support (Requires Docker for Windows)

OS: Windows

☑ Configure for HTTPS

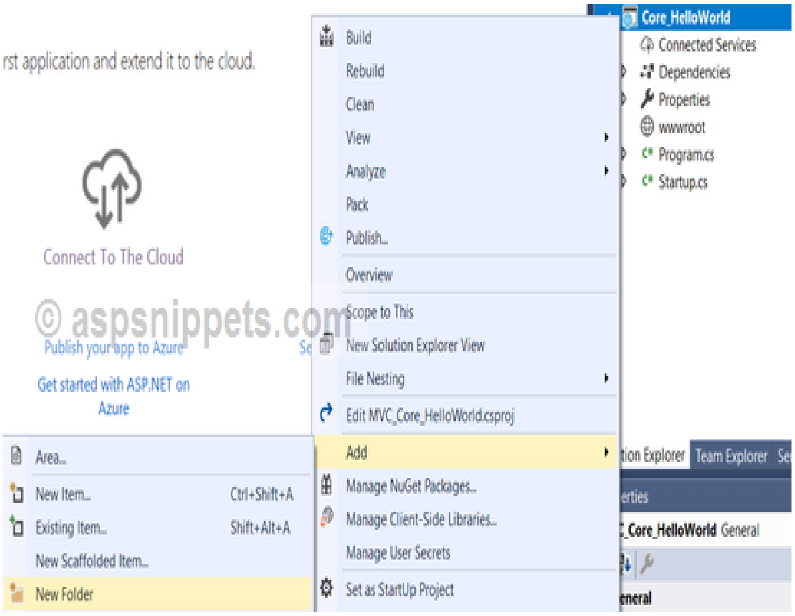
4. Your first Hello World ASP.Net Core Web Application Project is now ready and you should see the following folders and files in your Solution Explorer window.



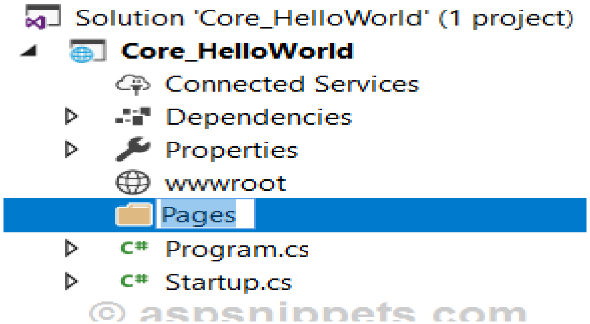
```
Solution 'Core_HelloWorld' (1 project)
Core_HelloWorld
    Connected Services
    Dependencies
    Properties
    wwwroot
    Program.cs
    Startup.cs
```

Adding Razor Page to the ASP.Net Core Project

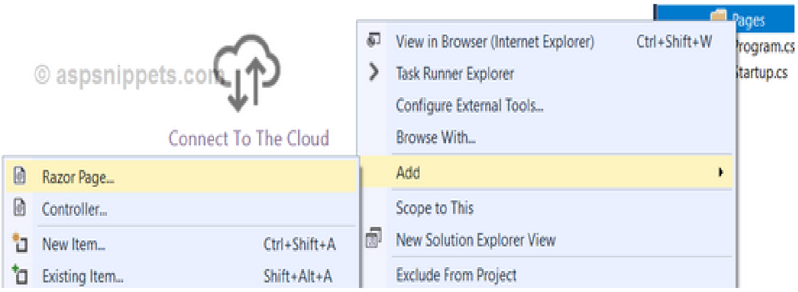
1. Inside the Solution Explorer window, Right Click on the Project and then click Add and then New Folder option from the Context Menu.



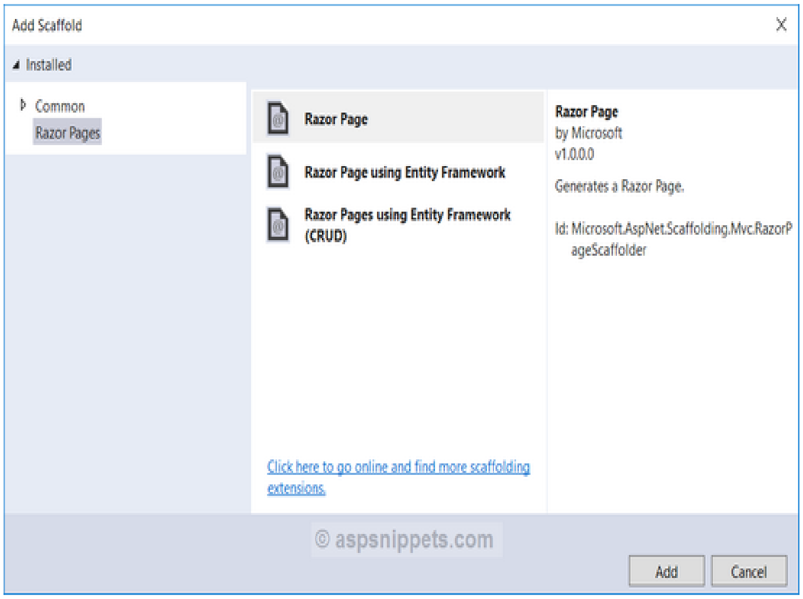
2. Name the newly added Folder as Pages.



3. Now, inside the Solution Explorer window, Right Click on the Pages folder and then click on Add and then Razor Page option of the Context Menu.

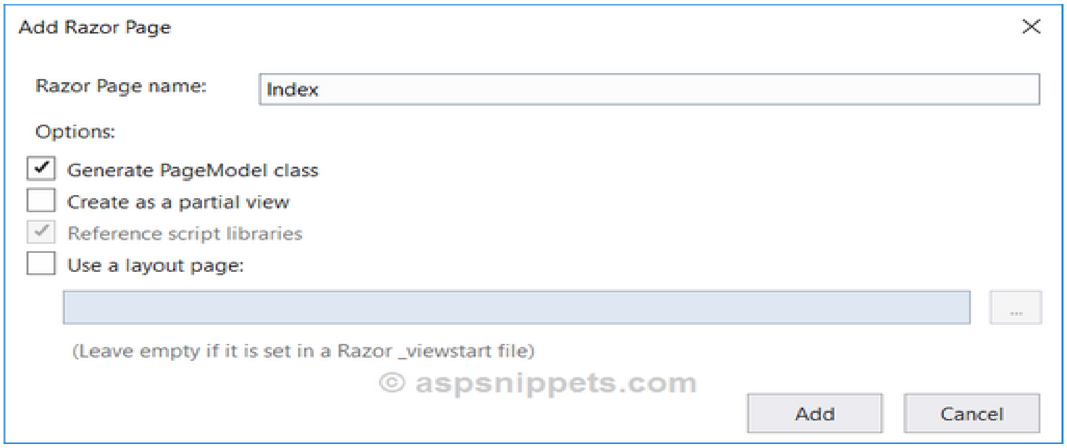


4. From the Add Scaffold Dialog window, select the "Razor Page" option and click Add button.


Add Scaffold
Installed
Common
Razor Pages
Razor Page
Razor Page using Entity Framework
Razor Pages using Entity Framework (CRUD)
Click here to go online and find more scaffolding extensions.
Add
Cancel

Razor Page
by Microsoft
v1.0.0.0
Generates a Razor Page.
Id: Microsoft.AspNet.Scaffolding.Mvc.RazorPageScaffolder

5. You will now be asked to provide a suitable Name to the new Razor Page.


Add Razor Page

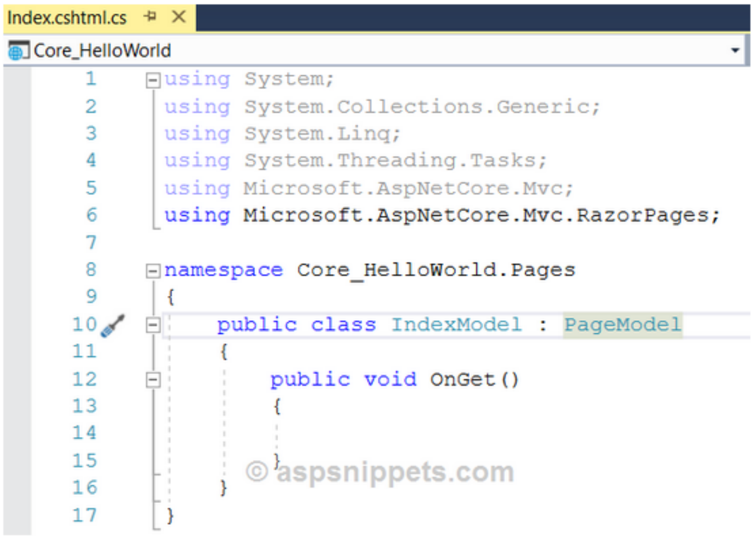
Razor Page name: Index

Options:

☑ Generate PageModel class
☐ Create as a partial view
☑ Reference script libraries
☐ Use a layout page:

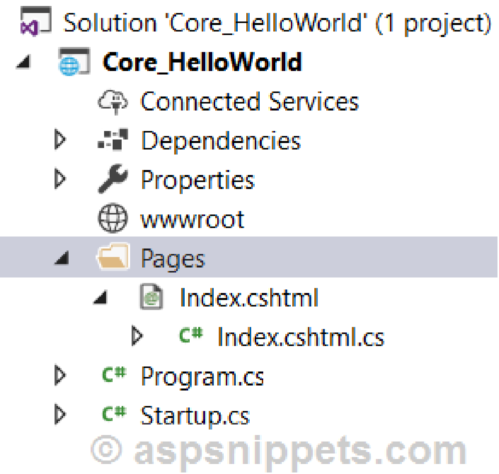
(Leave empty if it is set in a Razor _viewstart file)

Once you click add, the following Razor Page is created. The default handler method is “Get” in the Razor PageModel class.



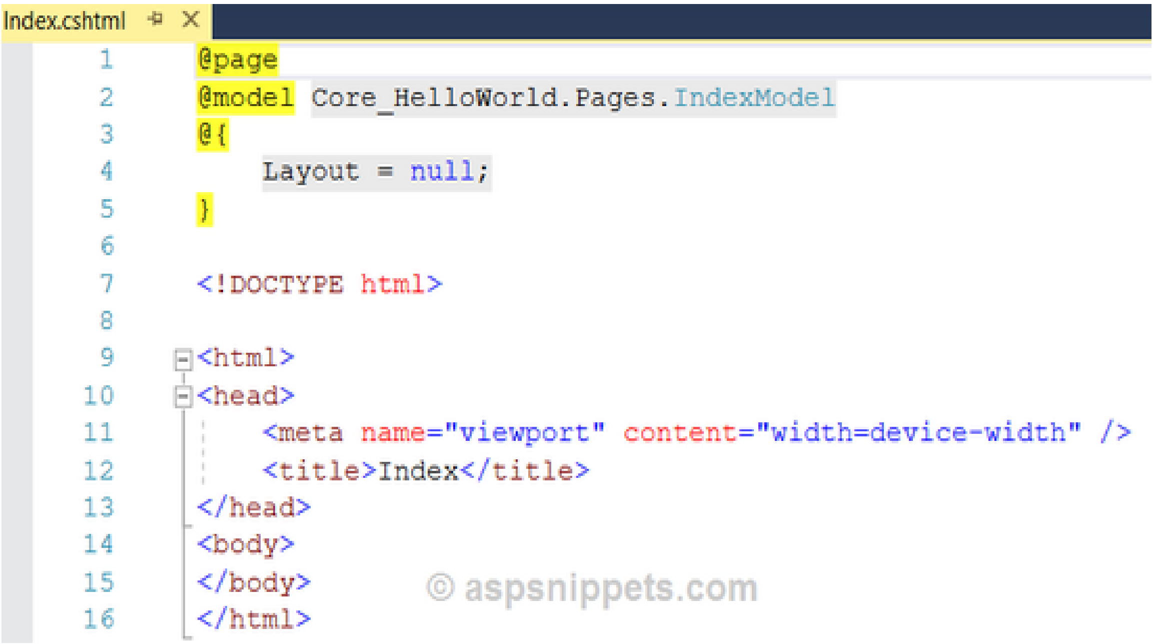
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace Core_HelloWorld.Pages
{
    public class IndexModel : PageModel
    {
        public void OnGet()
        {

        }
    }
}
```

Inside the Solution Explorer window, the corresponding Razor HTML Page is also shown.



```
Solution 'Core_HelloWorld' (1 project)
◢ Core_HelloWorld
      Connected Services
  ▷   Dependencies
  ▷   Properties
      wwwroot
  ◢   Pages
    ◢   Index.cshtml
      ▷   Index.cshtml.cs
  ▷   Program.cs
  ▷   Startup.cs
```

And when the Razor HTML Page is opened it looks as shown below.

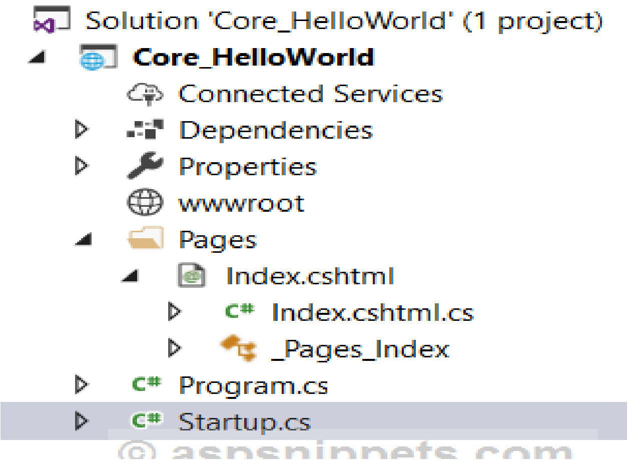


```
@page
@model Core_HelloWorld.Pages.IndexModel
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
</body>
</html>
```

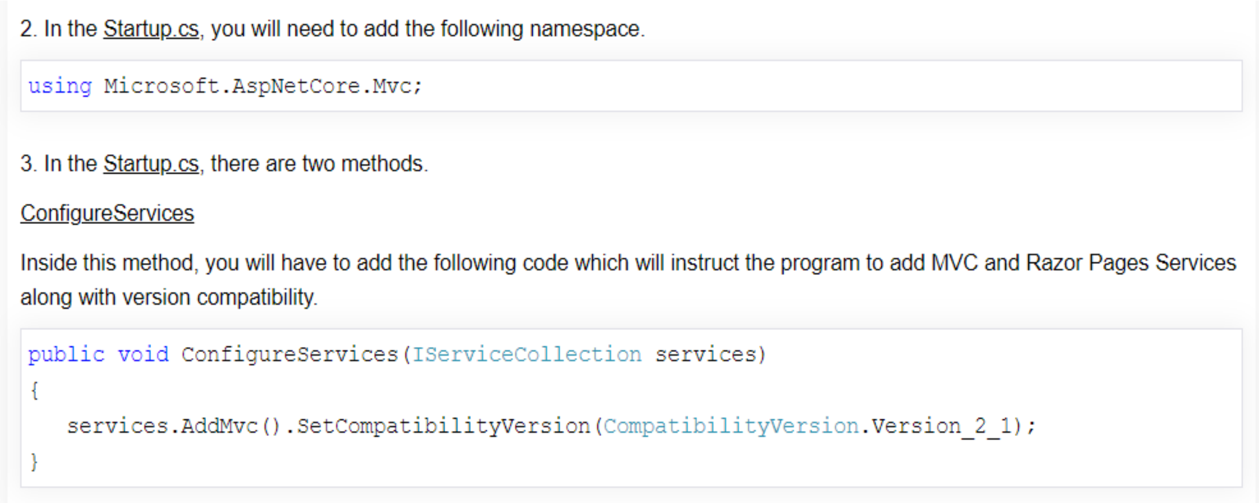
Configuring the Routes

The last important part is to configure the Routes in the Startup.cs file.

1. Open the Startup.cs class from the Solution Explorer window.



```
Solution 'Core_HelloWorld' (1 project)
◢ Core_HelloWorld
      Connected Services
  ▷   Dependencies
  ▷   Properties
      wwwroot
  ◢   Pages
    ◢   Index.cshtml
      ▷   Index.cshtml.cs
      ▷   _Pages_Index
  ▷   Program.cs
  ▷   Startup.cs
```



2. In the Startup.cs, you will need to add the following namespace.

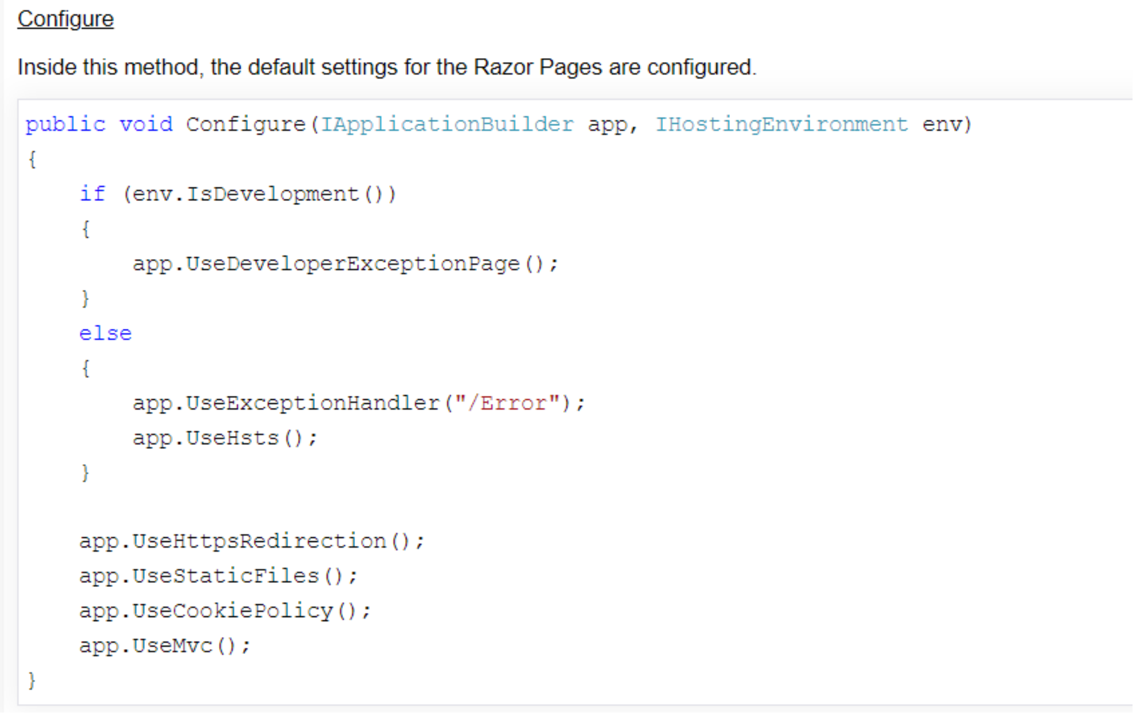
```
using Microsoft.AspNetCore.Mvc;
```

3. In the Startup.cs, there are two methods.

ConfigureServices

Inside this method, you will have to add the following code which will instruct the program to add MVC and Razor Pages Services along with version compatibility.

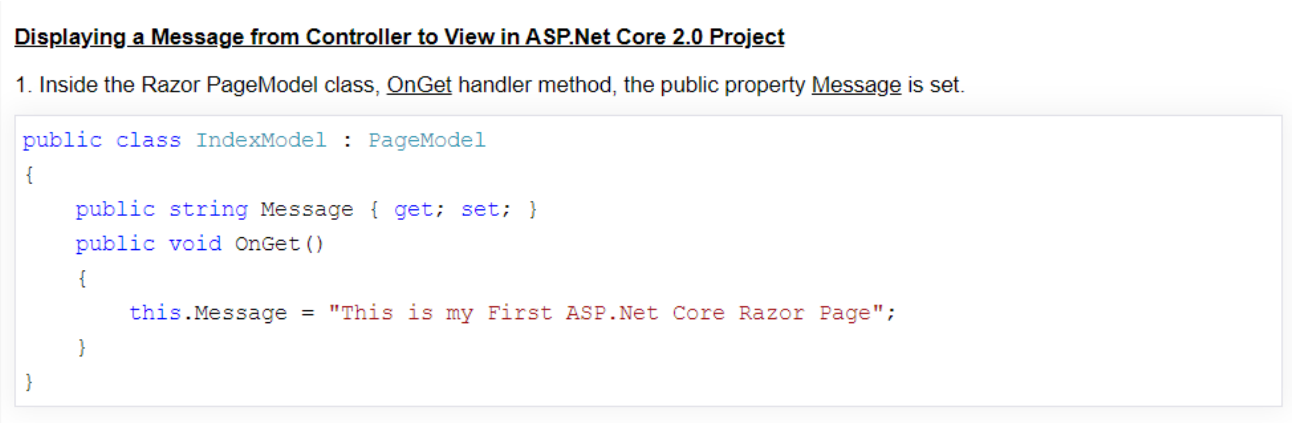
```
public void ConfigureServices(IServiceCollection services)
{
   services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
}
```



Configure

Inside this method, the default settings for the Razor Pages are configured.

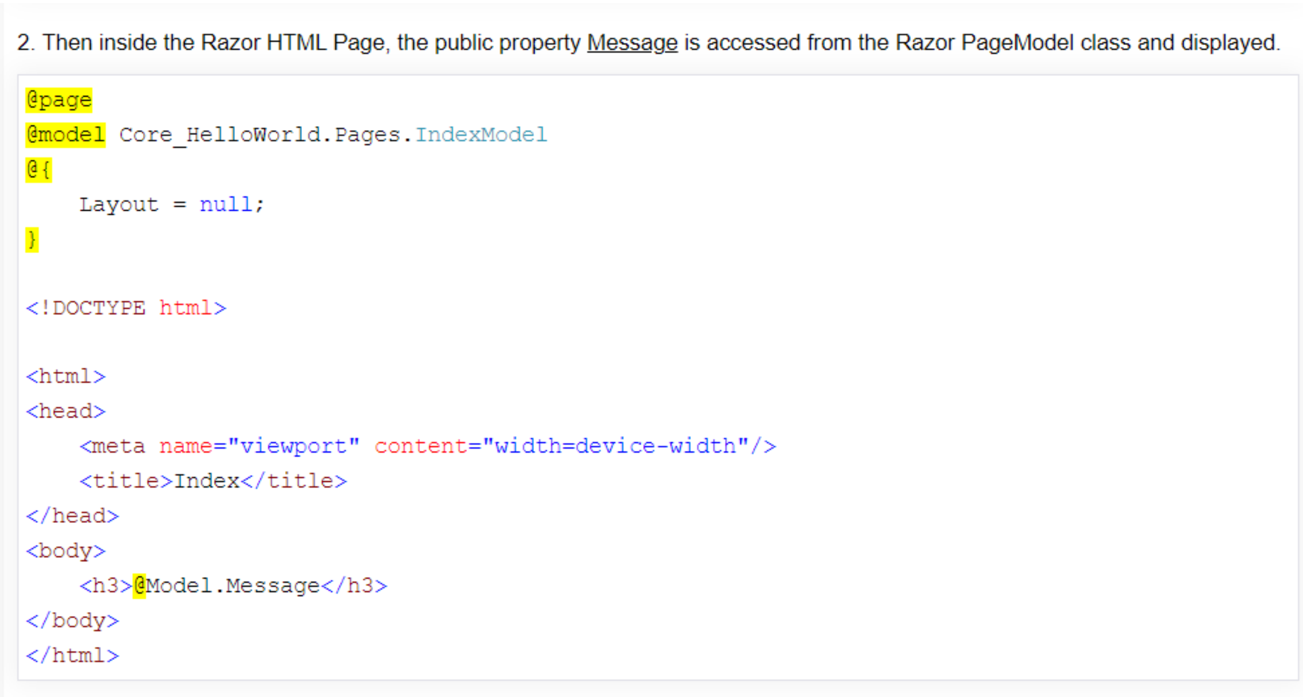
```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();
    app.UseMvc();
}
```



Displaying a Message from Controller to View in ASP.Net Core 2.0 Project

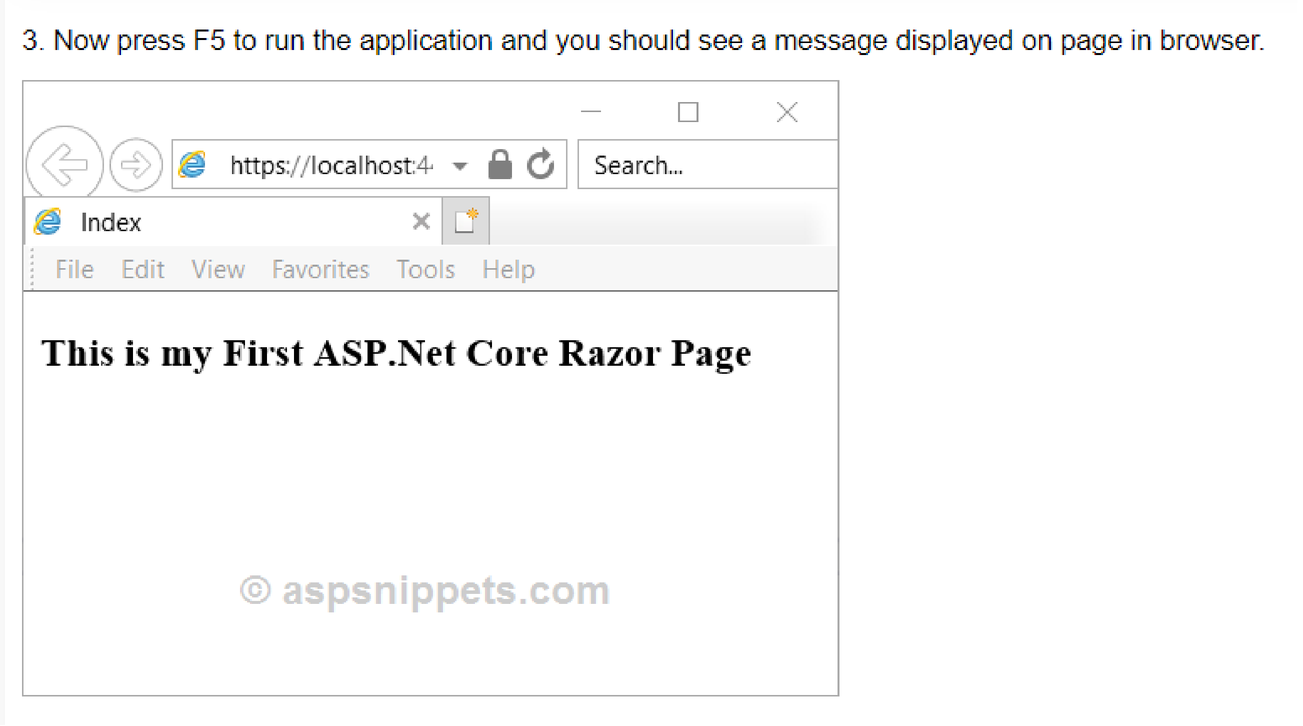
1. Inside the Razor PageModel class, OnGet handler method, the public property Message is set.

```
public class IndexModel : PageModel
{
    public string Message { get; set; }
    public void OnGet()
    {
        this.Message = "This is my First ASP.Net Core Razor Page";
    }
}
```



2. Then inside the Razor HTML Page, the public property Message is accessed from the Razor PageModel class and displayed.

```
@page
@model Core_HelloWorld.Pages.IndexModel
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>Index</title>
</head>
<body>
    <h3>@Model.Message</h3>
</body>
</html>
```



3. Now press F5 to run the application and you should see a message displayed on page in browser.

This is my First ASP.Net Core Razor Page

How to Run Hello world in ASP.net

https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-aspnet-core?view=vs-2022

How to Submit (Post) Form in ASP.Net Core Razor Pages

https://www.youtube.com/watch?v=gKVMf1nEfHs&t=18s