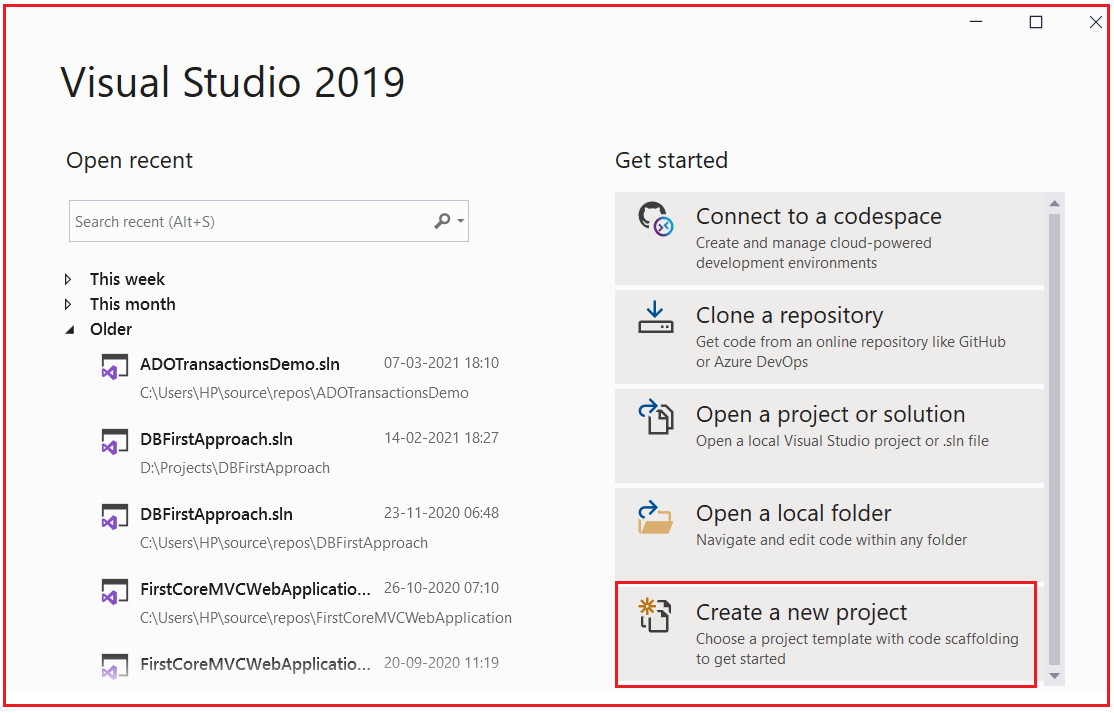
**Converting Console Application to Web API Application:**

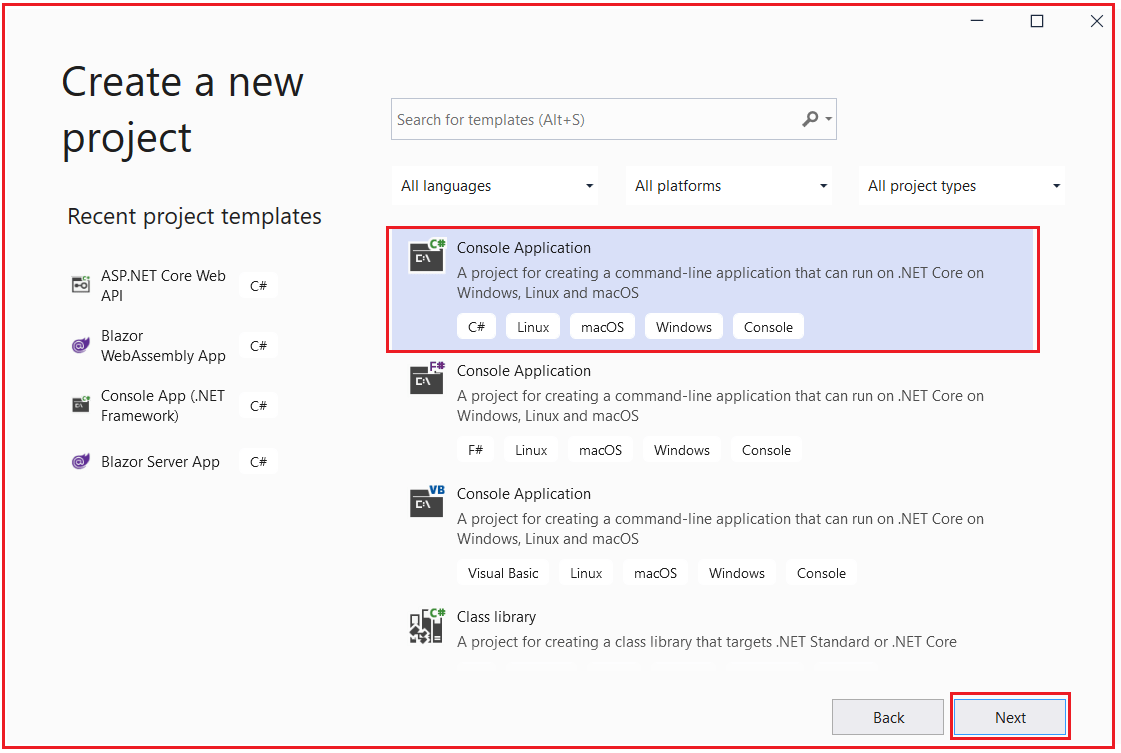
In the case of ASP.NET Core, by default, each .NET Core Application is a Console Application. This is the reason why you can see by default the Program.cs class file with the Main method in each .NET Core Web App. But then we need to add some services (specific to application type) and also need to do some changes to convert the console application to a Web API Project.

**Create a new Console Application**

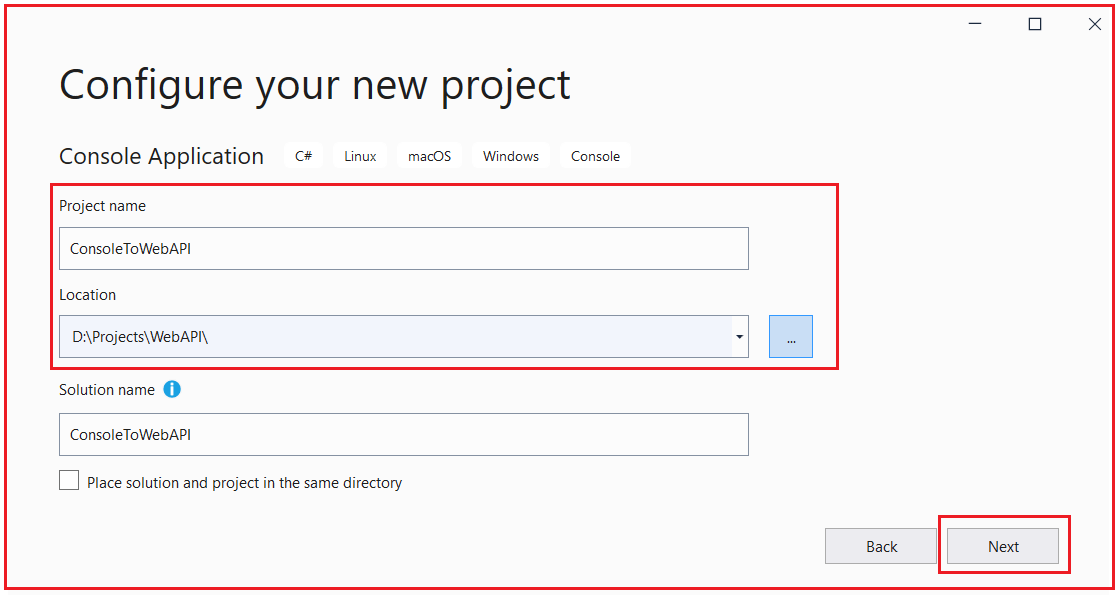
Open Visual Studio and then click on the Create a new Project option as shown in the below image.



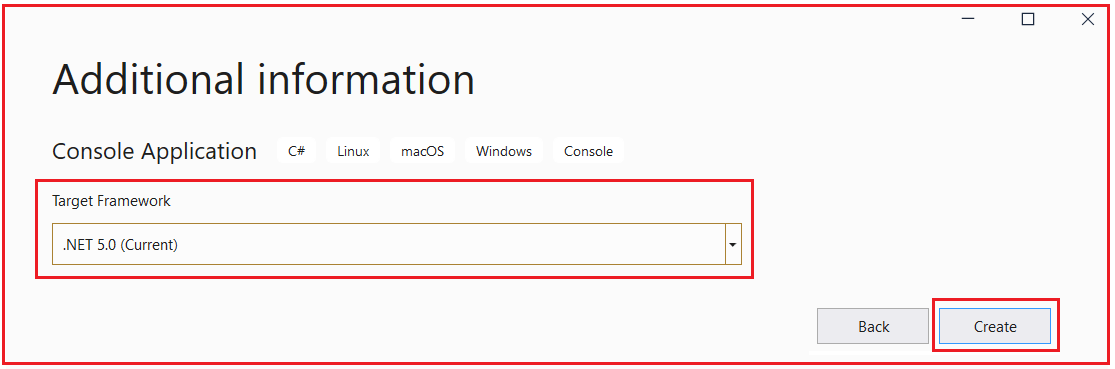
Then from the next Create a Project window, select Console Application that targets .NET Core Framework and click on the Next button as shown in the below image.



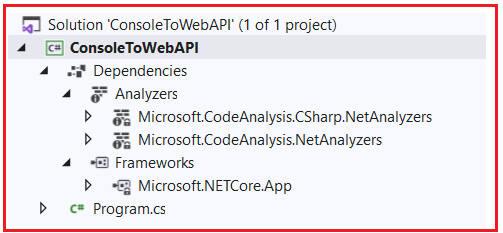
From the next Configure your new project window, give a meaningful name to your project (I am giving the name ConsoleToWebAPI), and select the location where we want to create the console application. Then if you want you can give a different name to your solution but I am going with the default one. Finally, click on the Next button as shown in the below image.



From the next Additional Information window, select the target .NET Core Framework. As we are working with .NET 5, so select .NET 5 and click on the Create button as shown in the below image.



Once you click on the Create button, it will create the Console Application with the following structure. As you can see, it contains Program.cs class file and Dependencies Folder. Inside the Dependencies folder, it also has the Analyzers and Frameworks folder. At this moment the Frameworks folder contains only Microsoft.NETCore.App. Also, notice the project type as C#.



Now open the Program.cs class file and you should the following in it which includes the Main method which is the entry point to this console application.

**using** *System;*

**namespace** *ConsoleToWebAPI*

**{**

**class** Program

**{**

**static** **void** Main**(**string**[]** args**)**

**{**

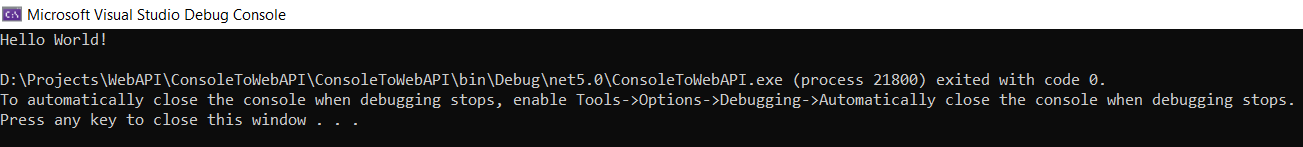
Console.WriteLine**(**"Hello World!"**)**;

**}**

**}**

**}**

Now if you run the application, then you will get the following output in the console window. That means our console application is working fine.



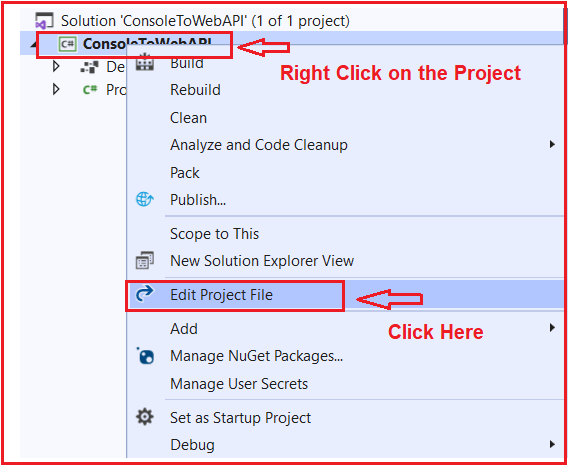
Now we will do some changes to this ASP.NET Core Console Application in order to make it an ASP.NET Core Web API Application.

**Changing in CSPROJ file (Project file)**

In order to make the console application a Web API Application, first of all, we need to make the following changed to the csproj file.

1. Change the Project SDK to Web
2. Remove the Project Output Type
3. Verify the Target Framework Moniker (TFM)

In order to open the csproj file of this console application, we have two options either by double-clicking on the project or by right-clicking on the project and select the **Edit Project file** option as shown in the below image.



Once you open the Project file, you will get the following configuration inside the project file. As you can see in the below code, the Project SDK is Microsoft.NET.Sdk and the OutputType is Exe and the TargerFramework is net5.0.

**<Project** Sdk="Microsoft.NET.Sdk"**>**

**<PropertyGroup>**

**<OutputType>**Exe**</OutputType>**

**<TargetFramework>**net5.0**</TargetFramework>**

**</PropertyGroup>**

**</Project>**

So, we need to do some changes to the above CSPROJ file in order to make this project a Web API Project. As we are going to deal with ASP.NET Core Web Application, so we need to set the SDK type as Microsoft.NET.Sdk.Web. Again, as it is no longer a console application, so we need to remove the OutputType which is set to Exe from the Project file. Then we need to set the TargetFranmework. As already it is set to net5.0, which means it is dealing with .NET 5, so we don’t require to do any changes. In case if it is target to some lower version of the .NET Framework, then change it to net5.0. So, with the above changes in place, your CSPROJ file should looks as shown below.

**<Project** Sdk="Microsoft.NET.Sdk.Web"**>**

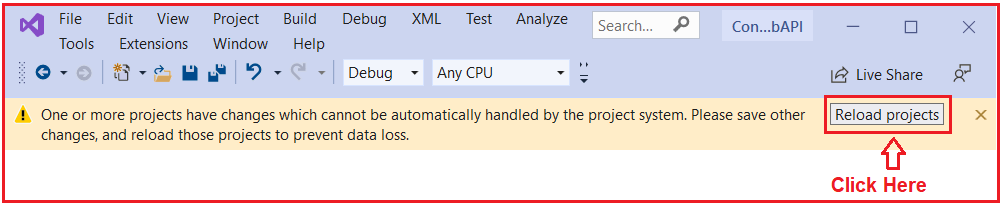
**<PropertyGroup>**

**<TargetFramework>**net5.0**</TargetFramework>**

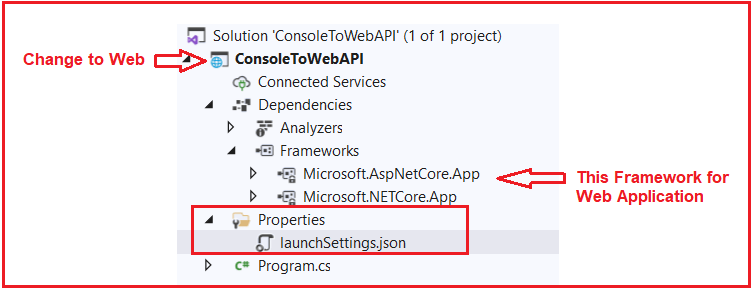
**</PropertyGroup>**

**</Project>**

Once you save the changes, it will give you one warning that one or more projects have changes that cannot be automatically handled by the project system. Please save other changes and reload those projects to prevent data loss. Simply click on the Reload Projects button as shown in the below image.



Once you click on the Reload Projects button, the warning gone and you can also observe the project structure also has been changed. Now, the project type is changed to the web, it also adds **Microsoft.AspNetCore.App** inside the Frameworks folder which provides support for ASP.NET Core Web App as well as it also adds **Properties** folder with **launchSettings.json** file as shown in the below image.



The launchSettings.json file contains all the settings that are required to run this application in the Development environment. You will find the following settings in the launchSettings.json file.

**{**

"iisSettings": **{**

"windowsAuthentication": **false,**

"anonymousAuthentication": **true,**

"iisExpress": **{**

"applicationUrl": "http://localhost:60654/"**,**

"sslPort": 44337

**}**

**},**

"profiles": **{**

"IIS Express": **{**

"commandName": "IISExpress"**,**

"launchBrowser": **true,**

"environmentVariables": **{**

"ASPNETCORE\_ENVIRONMENT": "Development"

**}**

**},**

"ConsoleToWebAPI": **{**

"commandName": "Project"**,**

"launchBrowser": **true,**

"environmentVariables": **{**

"ASPNETCORE\_ENVIRONMENT": "Development"

**},**

"applicationUrl": "https://localhost:5001;http://localhost:5000"

**}**

**}**

**}**

These are the changes required in the project file. But this is not enough to run this application at this moment. We need to do some changes to the Program.cs file