**ASP.NET Core Main Method**

* The significance of **Main() method in an ASP.NET Core application**
* What happens behind the scenes when a .NET core application is executed

In an ASP.NET Core project we have a file with name **Program.cs**. In this file we have a **public static void Main()** method

public class Program

{

    public static void Main(string[] args)

    {

        CreateWebHostBuilder(args).Build().Run();

    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

        WebHost.CreateDefaultBuilder(args)

            .UseStartup<Startup>();

}

If you have any experience with previous versions of .NET, a console application has a **Main()** method and it is the entry point for that console application.  
  
But here, we are creating an **asp.net core web application** and not a console application. So the obvious question that comes to our mind is why do we have a Main() method.  
  
Well, the important point to keep in mind is that, an **asp.net core application initially starts as a console application** and the **Main()** method in **Program.cs** file is the entry point.   
  
So, when the runtime executes our application **it looks for this Main() method**and this where the execution starts.  
  
This **Main() method configures asp.net core**and starts it and at that point it becomes an asp.net core web application.  
  
So, if you take a look at the **Main()** method, it calls CreateWebHostBuilder() method passing it the command line arguments.  
  
As you can see, CreateWebHostBuilder() method returns an object that implements IWebHostBuilder.  
  
On this object, **Build()** method is called which builds a web host that hosts our asp.net core web application.  
  
On the web host **Run()**method is called, which runs the web application and it begins listening for incoming HTTP requests.  
  
**CreateWebHostBuilder()** method calls CreateDefaultBuilder() static method of the WebHost class.  
  
CreateDefaultBuilder() method creates a web host with pre-configured defaults. CreateDefaultBuilder() method does several things to create a web host. We will discuss all that the CreateDefaultBuilder() method does in detail in our next video. For now, just understand that the CreateDefaultBuilder() method **sets up a web host with certain defaults**.  
  
As part of setting up a web host, Startup class is also configured using the UseStartup() extension method of IWebHostBuilder class. If you are new to the concept of extension methods, please check out the following video.  
  
[Extension Methods in C#](https://www.youtube.com/watch?v=VkrKNXscoto)  
  
By convention, the startup class in ASP.NET Core is named Startup. This class has 2 methods.

public class Startup

{

    public void ConfigureServices(IServiceCollection services)

    { }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)

    {

        if (env.IsDevelopment())

        {

            app.UseDeveloperExceptionPage();

        }

        app.Run(async (context) =>

        {

            await context.Response.WriteAsync("Hello World!");

        });

    }

}

**Startup class does the following 2 very important things**

* **ConfigureServices()**method configures services required by the application
* **Configure()** method sets up the application's request processing pipeline

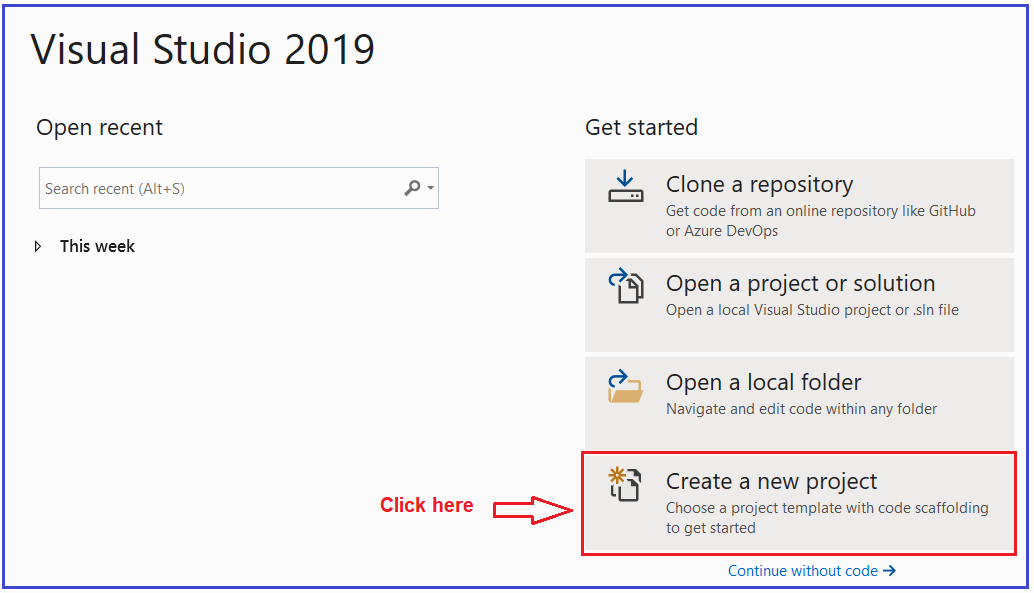
It is very important we understand what these 2 methods does. We will be revisiting these 2 methods several times as we progress through this course and discuss them in detail.

1. **The significance of the ASP.NET Core Main Method.**
2. **Why do we have a Main() method in ASP.NET Core?**
3. **What happens behind the scenes when you run a .NET core application?**

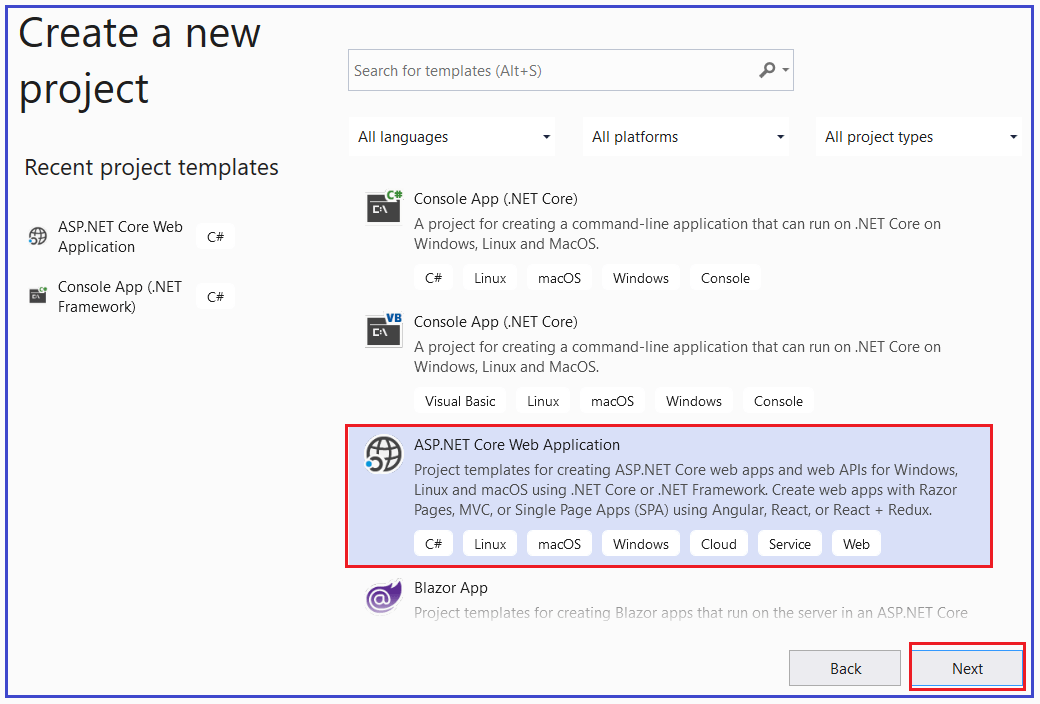
In order to understand the Main method of ASP.NET Core Web application, let’s first create an Empty ASP.NET Core Web application.

##### **Creating Empty ASP.NET Core Web Application**

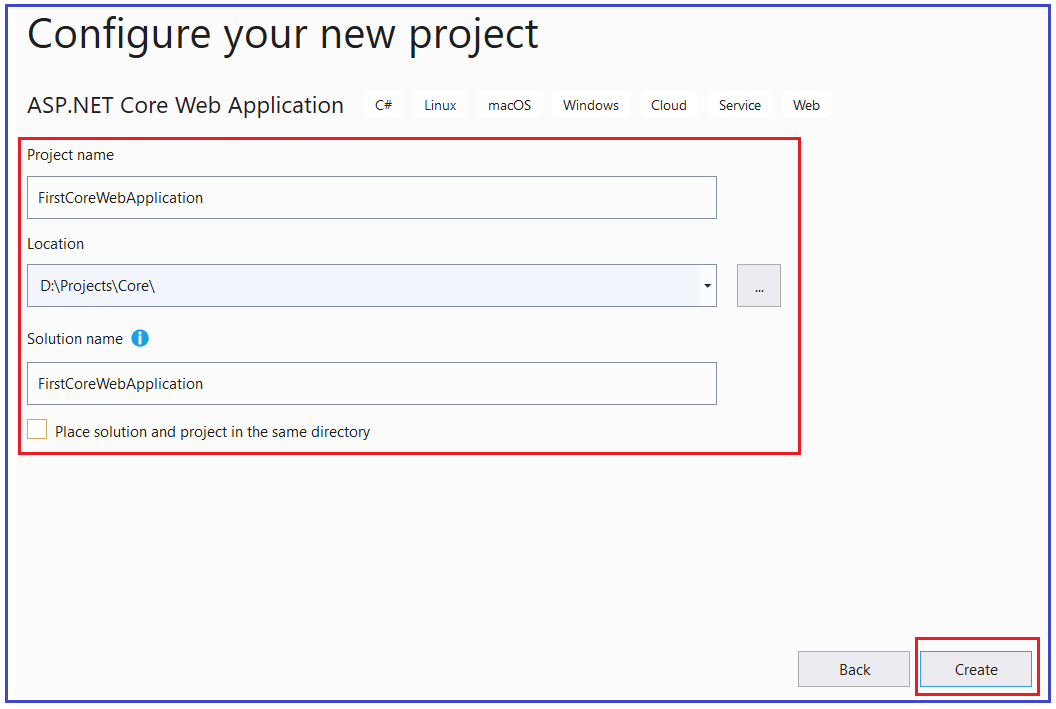
To create a new Empty ASP.NET Core Web Application, Open Visual Studio 2019, and then click on the Create a new project box as shown in the below image.



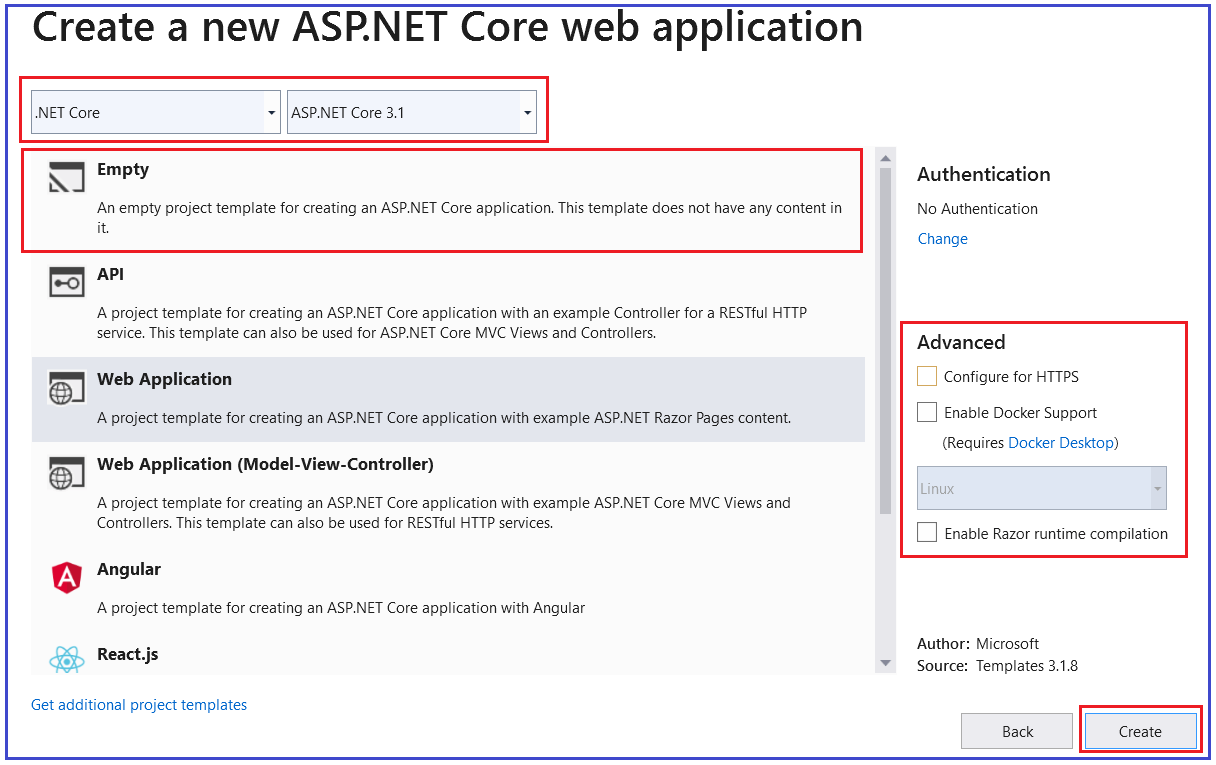
Once you click on the Create a new project box, it will open the Create a new project window. From this window select the **ASP.NET Core Web Application** template and click on the Next button as shown in the below image.



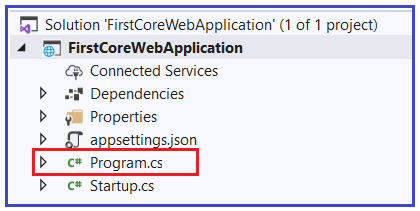
Once you click on the Next button, it will open the following **Configure Your New Project** window. Here, you need to give an appropriate name for your project, set the location where you want to create this project, the solution name for the ASP.NET Core Web application. In this example, we will give the name “**FirstCoreWebApplication**” and click on the Create button as shown in the image below.



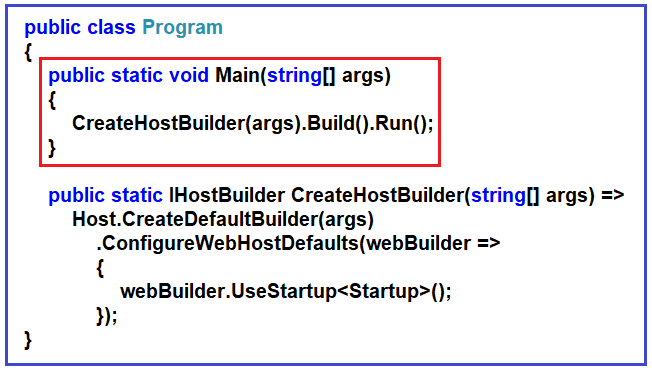
Once you click on the create button, it will open the **Create a new ASP.NET Core Web Application** window. Select the Empty Project template and uncheck all the checkboxes from the Advanced section and finally click on the Create button as shown in the below image.



Once you click on the **Create** button, it will take some time and will create the **Empty ASP.NET Core Web Application** with the following file and folder structure.



As you can see from the above image, we have a class file with the name **Program.cs**. The Program.cs class file of our ASP.NET Core Web Application contains the following code.



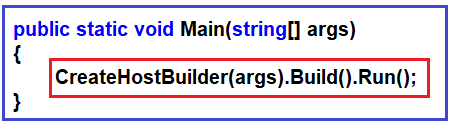
From the above image, you can see that the Program class contains a **public static void Main()**method. As we already know, when we create a console application in .net then by default the .NET Framework creates a class (i.e. Program class) with the Main Method. We also know that the Main method is the entry point for that console application execution.

Now the question is, here we are not creating a console application, here we create an ASP.NET Core Web Application. Then why do we have a Main() method in the ASP.NET Core Web Application?

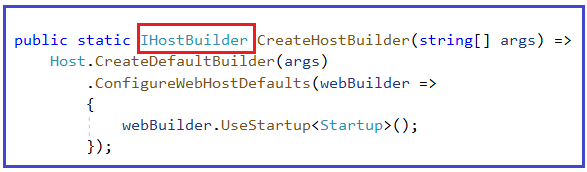
##### **Why do we have a Main() method in ASP.NET Core?**

The most important point that you need to keep in mind is, the ASP.NET Core Web Application initially starts as a Console Application and the Main() method is the entry point to the application. So, when we execute the ASP.NET Core Web application, first it looks for the Main() methodand this is the method from where the execution starts. The Main() method then configures ASP.NET Coreand starts it. At this point, the application becomes an ASP.NET Core web application.

If you further look at the body of the Main() method, then you will find that it makes a call to the CreateHostBuilder() method by passing the command line arguments args as an argument as shown in the below image.



As shown in the below image, the CreateHostBuilder() method returns an object that implements the IHostBuilder interface. The Host is a static class that can be used for creating an instance of IHostBuilder with pre-configured defaults. The CreateDefaultBuilder() method creates a new instance of the HostBuilder with pre-configured defaults. Internally, it configures Kestrel (Internal Web Server for ASP.NET Core), IISIntegration, and other configurations.

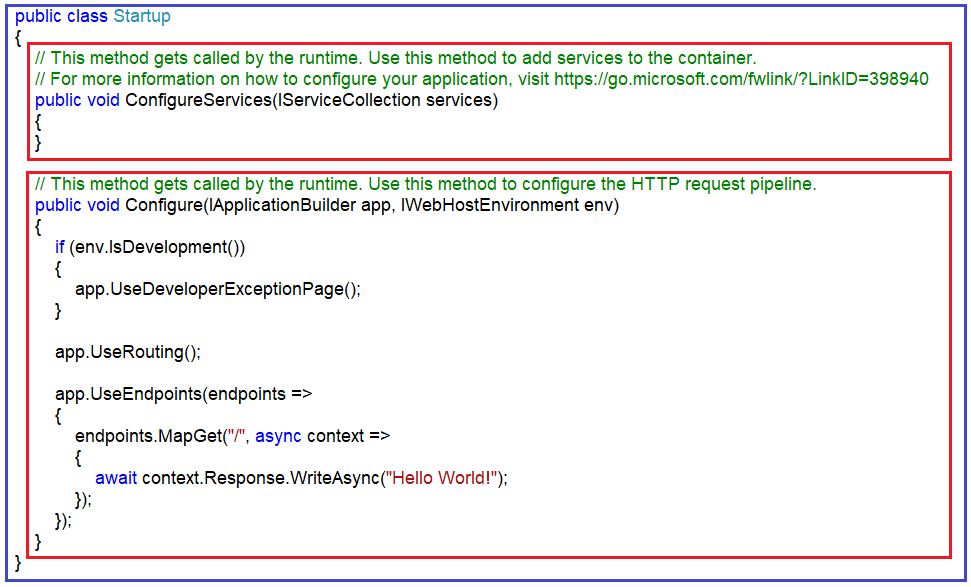


Within the Main() method, on this IHostBuilder object, the Build() method is called which actually builds a web host. Then it hosts our asp.net core web application within that Web Host. Finally, on the web host, it called the Run()method, which will actually run the web application and it starts listening to the incoming HTTP requests.

The CreateHostBuilder() method calls the static CreateDefaultBuilder() method  on the Host class. The CreateDefaultBuilder() method creates a web host with the default configurations. Behind the scene, to create a host, the CreateDefaultBuilder() method does several things. In the next article, we will discuss the **[CreateDefaultBuilder()](https://dotnettutorials.net/lesson/asp-net-core-inprocess-hosting/)****[method](https://dotnettutorials.net/lesson/asp-net-core-inprocess-hosting/)** in detail. For now, just understand that the CreateDefaultBuilder() method sets up a web host with default configurations.

##### **Startup Class**

While setting up the host, the Startup class is also configured using the UseStartup() extension method of the IHostBuilderclass. The Startup class has two methods as shown in the below image.



The ConfigureServices()method of the Startup class configures the services which are required by the application. The Configure() method of the Startup class sets up the pipeline of the application’s request processing. In a later article, we will discuss these two methods in detail.