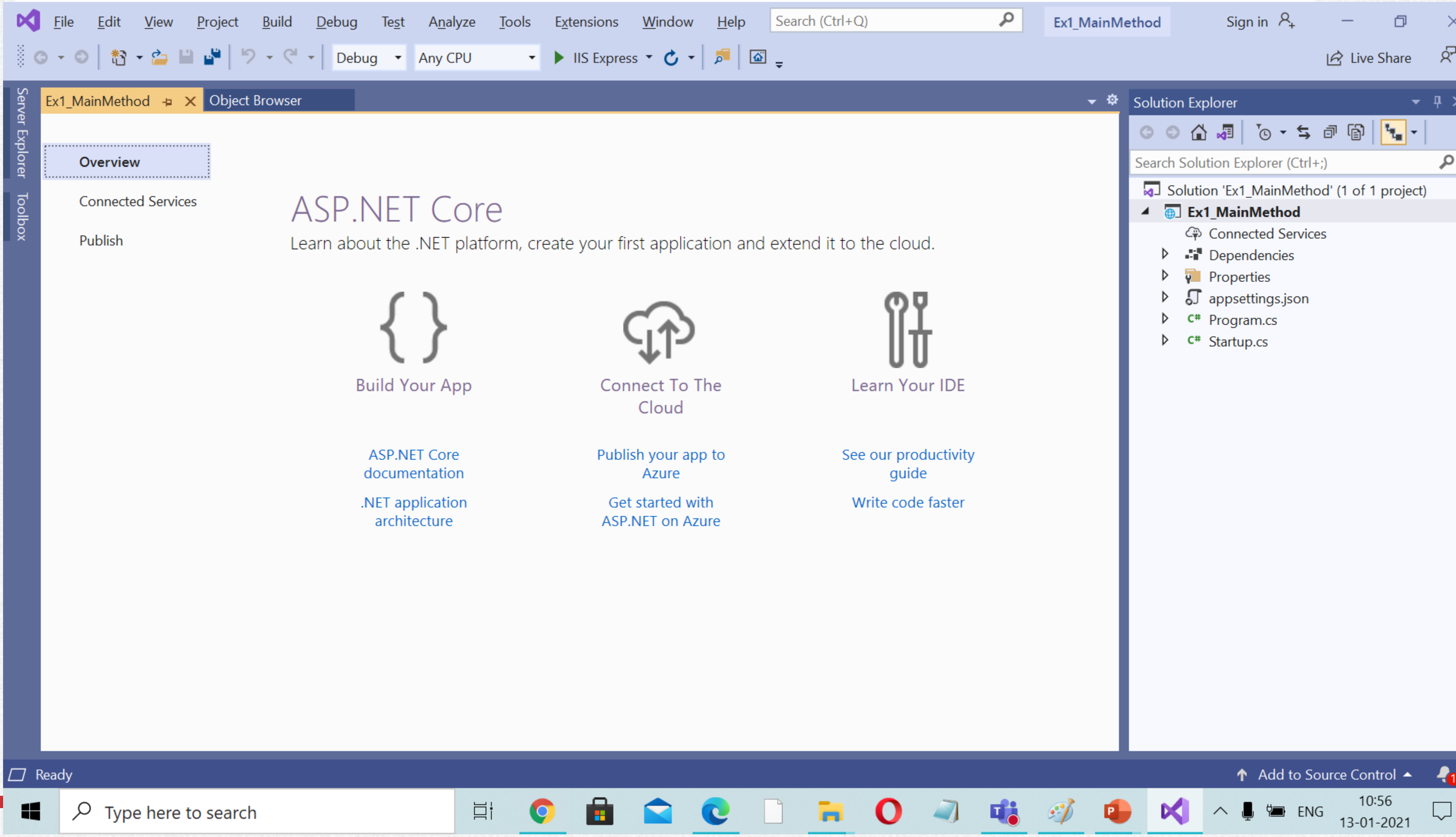
ASP.NET Core Main Method

- Significance of Main Method
- Why we have Main Method in ASP.NET Core?
- What happens behind when we run an ASP.NET Core Application?



ASP.NET Core

Learn about the .NET platform, create your first application and extend it to the cloud.

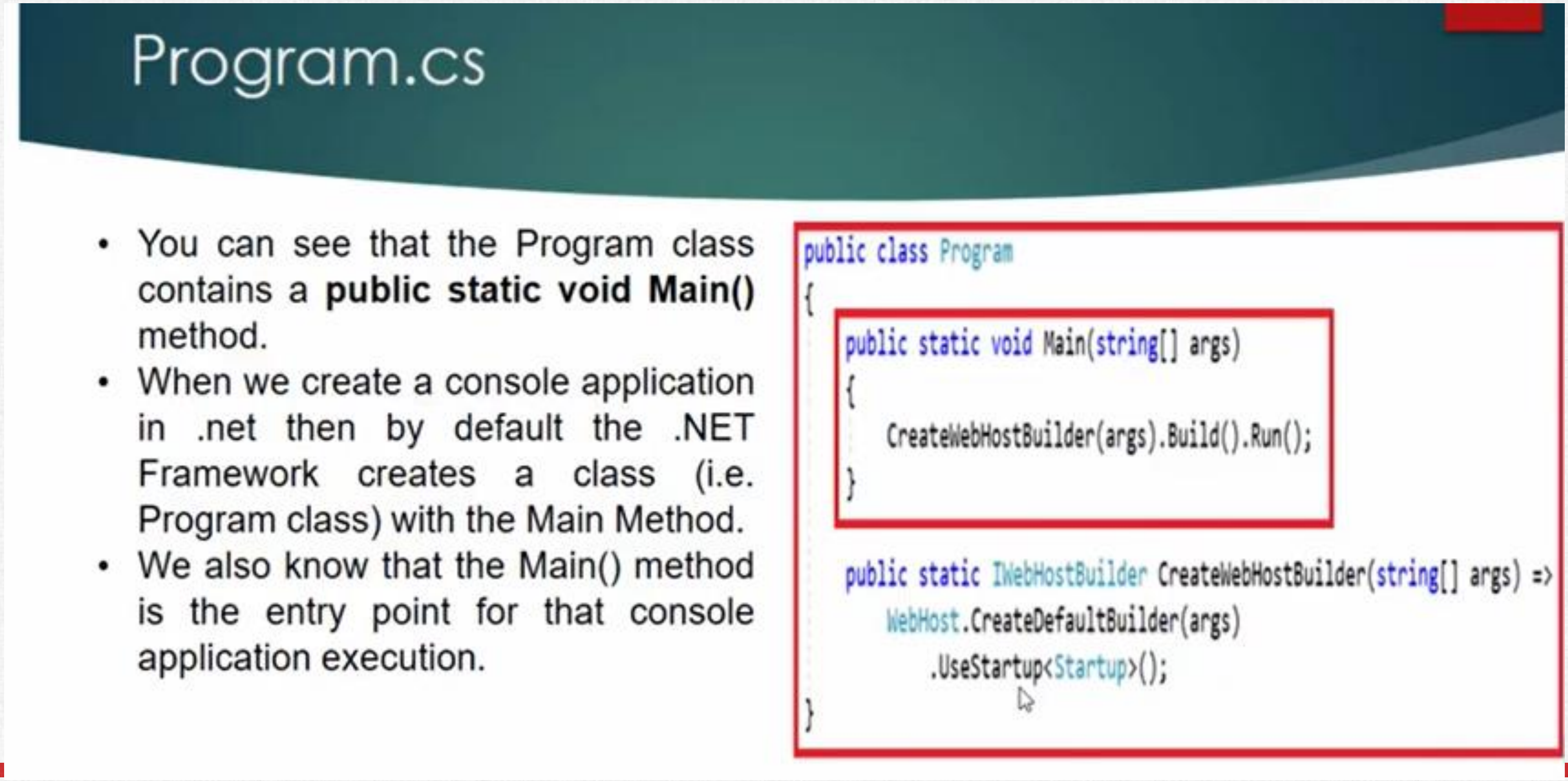
Build Your App
ASP.NET Core documentation
.NET application architecture

Connect To The Cloud
Publish your app to Azure
Get started with ASP.NET on Azure

Learn Your IDE
See our productivity guide
Write code faster

Solution 'Ex1_MainMethod' (1 of 1 project)
- Ex1_MainMethod
  - Connected Services
  - Dependencies
  - Properties
  - appsettings.json
  - Program.cs
  - Startup.cs

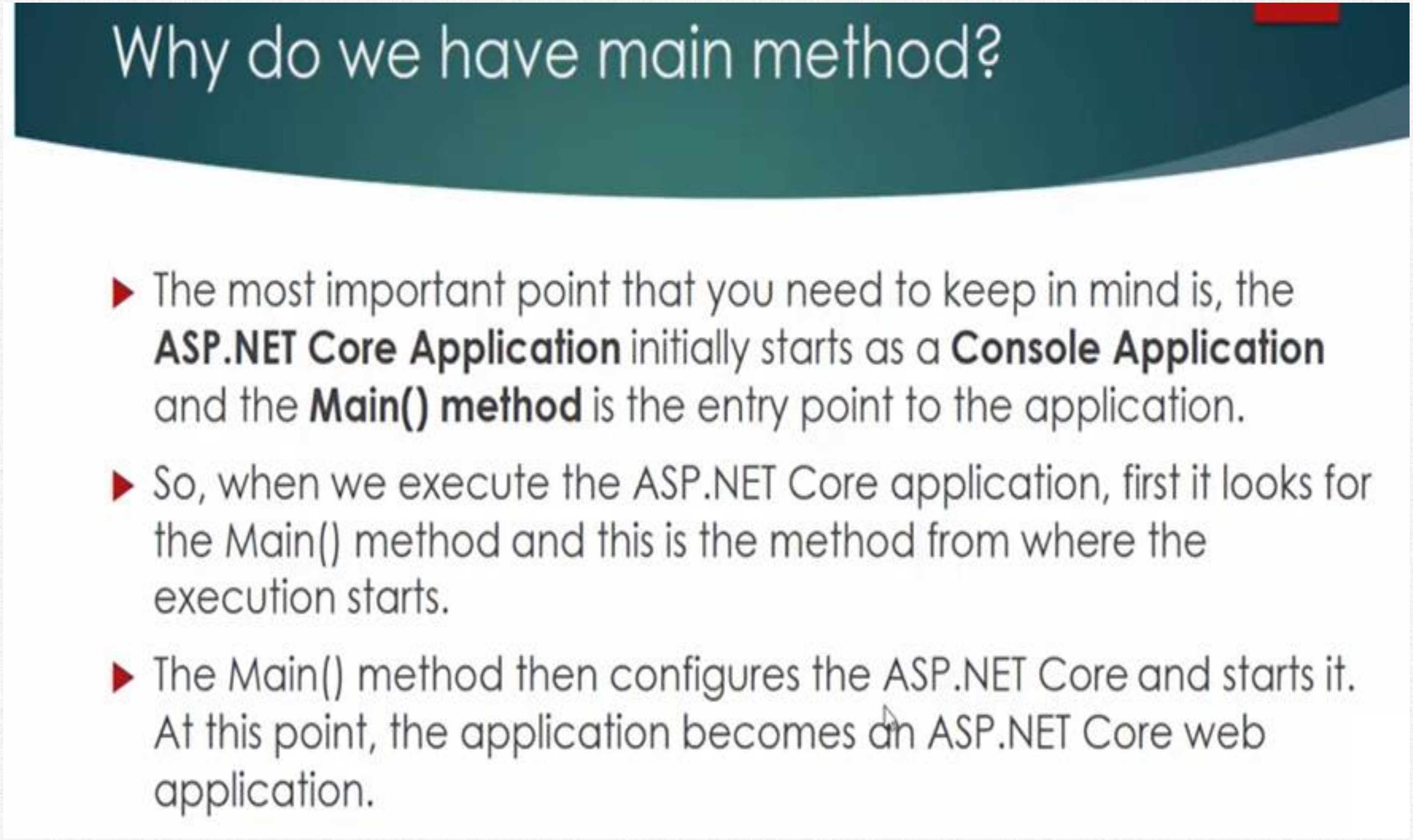
- Two .cs file is created by default Program.cs and Startup.cs.
- Program.cs:-Inside that Main Method is there



Program.cs

- You can see that the Program class contains a **public static void Main()** method.
- 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.

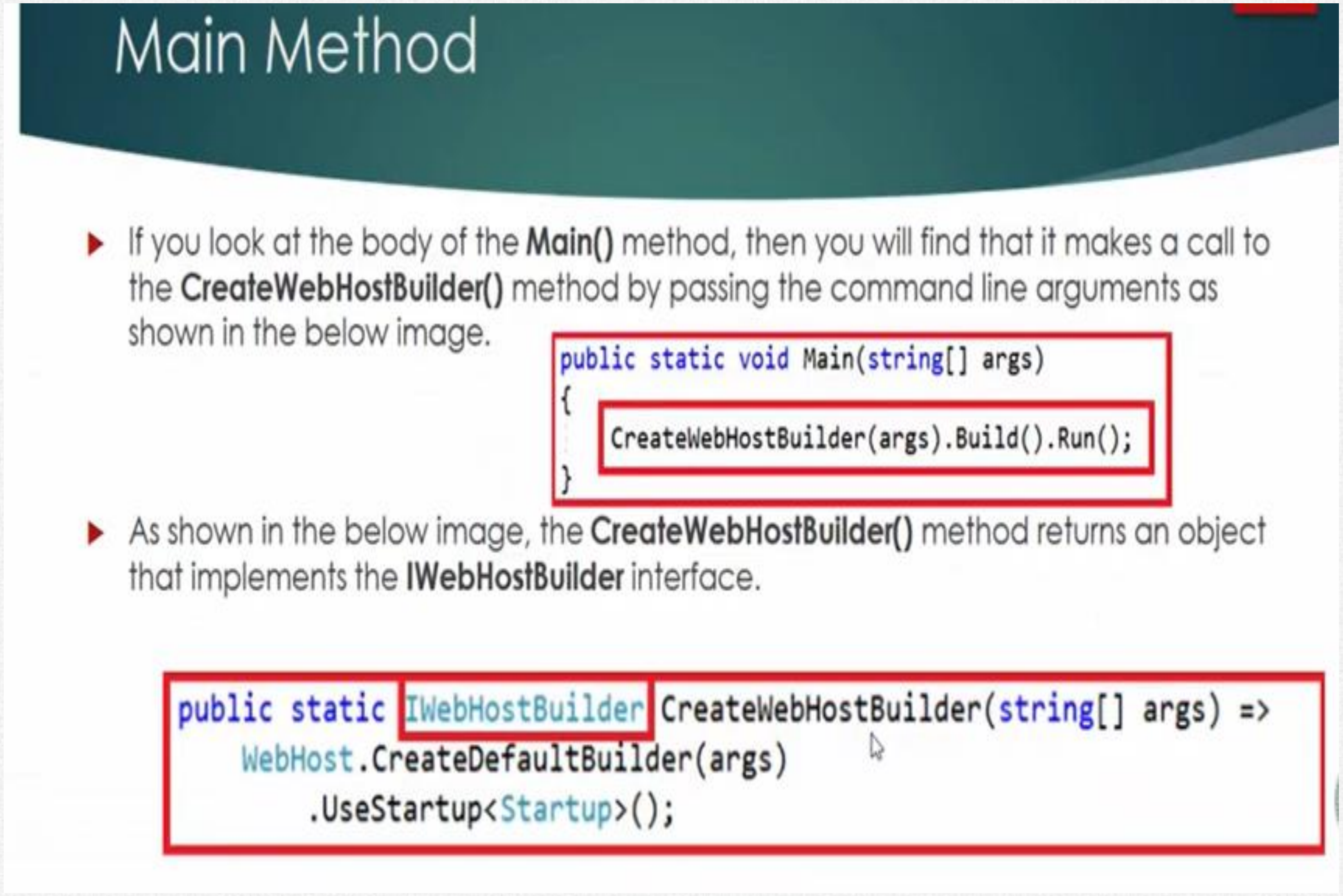
```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```



Why do we have main method?

- The most important point that you need to keep in mind is, the **ASP.NET Core 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 application, first it looks for the Main() method and this is the method from where the execution starts.
- The Main() method then configures the ASP.NET Core and starts it. At this point, the application becomes an ASP.NET Core web application.

Webhost:-build the web host, host the web application within the webhost.
Build.Run():- Run Method, which actually runs the web application and start listening to incoming http request.



Main Method

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

```
public static void Main(string[] args)
{
    CreateWebHostBuilder(args).Build().Run();
}
```

- As shown in the below image, the **CreateWebHostBuilder()** method returns an object that implements the **IWebHostBuilder** interface.

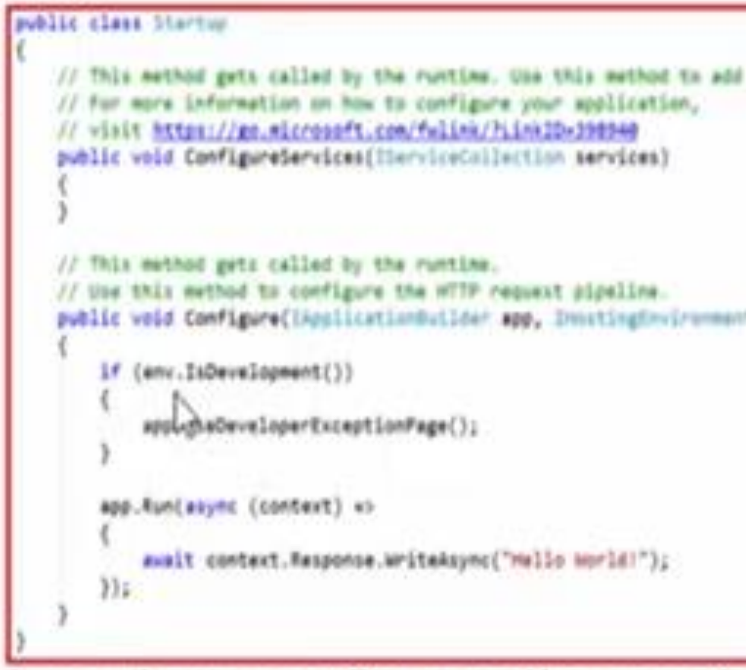
```
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>();
```

Startup.cs:-

- This class have two method
- 1) Configureservices:-All the services call inside this method those are required in our web application.
- 2) Configure:-Setup the request pipeline processing.

Startup Class

- While setting up the web host, the Startup class is also configured using t
  extension method of the **IWebHostBuilder** class.
- It has two method as shown in the picture.



```
public class Startup
{
    // This method gets called by the runtime. Use this method to add
    // For more information on how to configure your application,
    // visit https://go.microsoft.com/fwlink/?LinkID=398940
    public void ConfigureServices(IServiceCollection services)
    {
    }

    // This method gets called by the runtime.
    // Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.Run(async (context) =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    }
}
```

- The **ConfigureServices()** method of the Startup class configures the servic
  required by the application.
- The **Configure()** method of the Startup class sets up the pipeline of the a
  request processing

```
public void ConfigureServices(IServiceCollection services)
        {
        }
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            app.UseRouting();
            app.UseEndpoints(endpoints =>
            {    endpoints.MapGet("/", async context =>
                {
                    await context.Response.WriteAsync("Hello World!");
                });
            })
```

Thanks