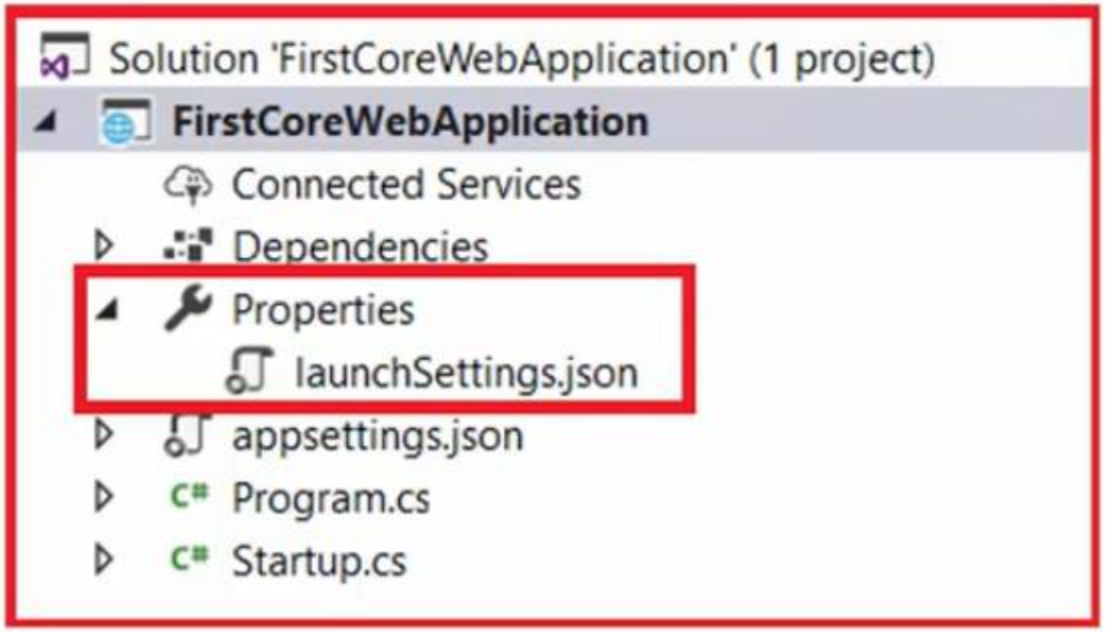
launchsettings.json file in ASP.NET CORE

Project

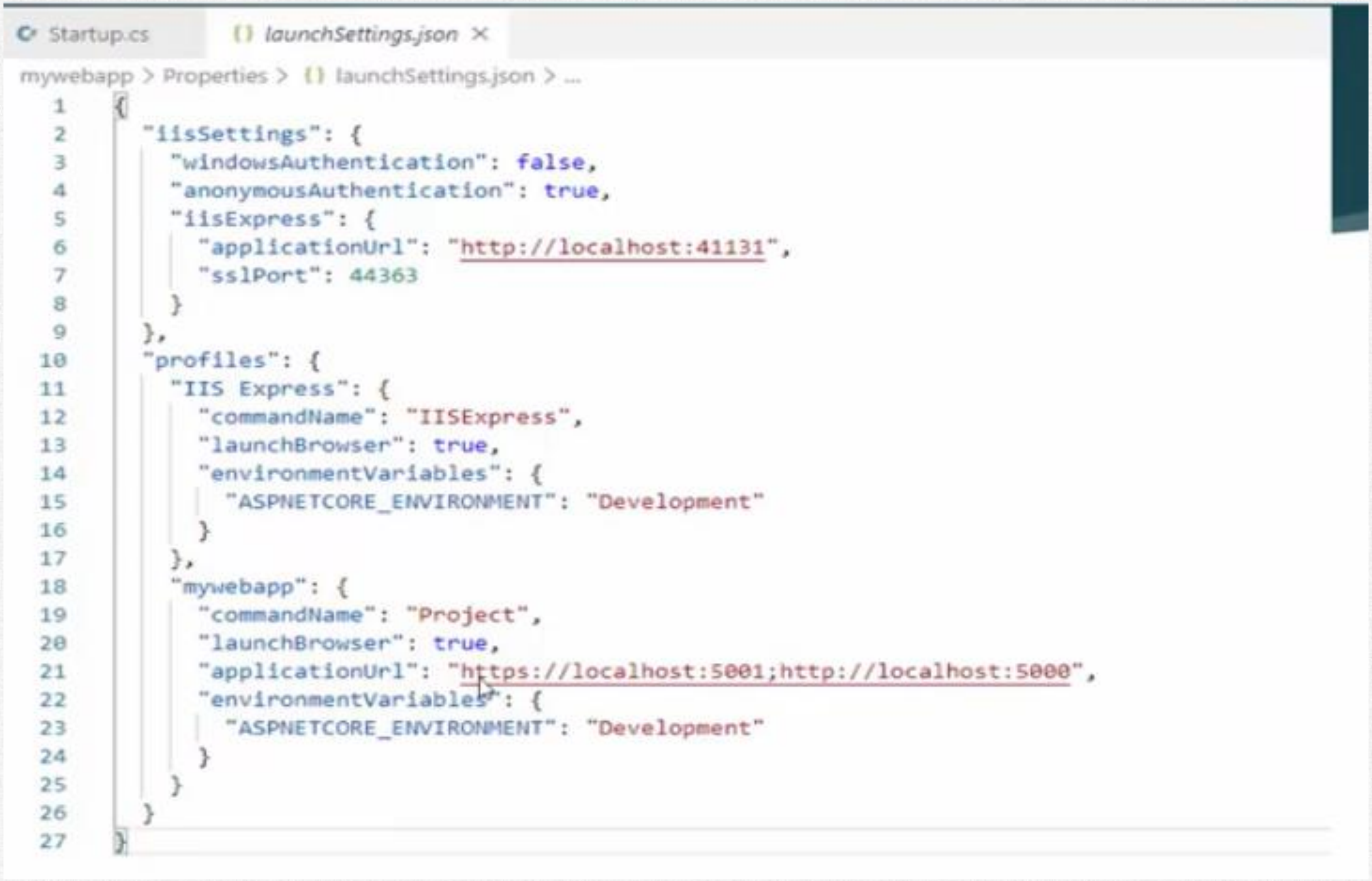


Solution 'FirstCoreWebApplication' (1 project)
- FirstCoreWebApplication
  - Connected Services
  - Dependencies
  - Properties
    - launchSettings.json
  - appsettings.json
  - Program.cs
  - Startup.cs

launchSettings.json file

- The settings that are present within this file are going to be used when we run the .NET core application either from Visual Studio or by using .NET Core CLI.
- The most important point that you need to keep in mind is this launchSettings.json file is only used within the local development machine.
- That means this file is not required when we publishing the asp.net core application to the production server.
- If you have certain settings and you want your application to use such settings when you publish and deploy your application to a production server, then you need to store such settings in an appsettings.json file.
- Generally, in the ASP.NET Core application, the configuration settings are going to be stored in the appsettings.json file.

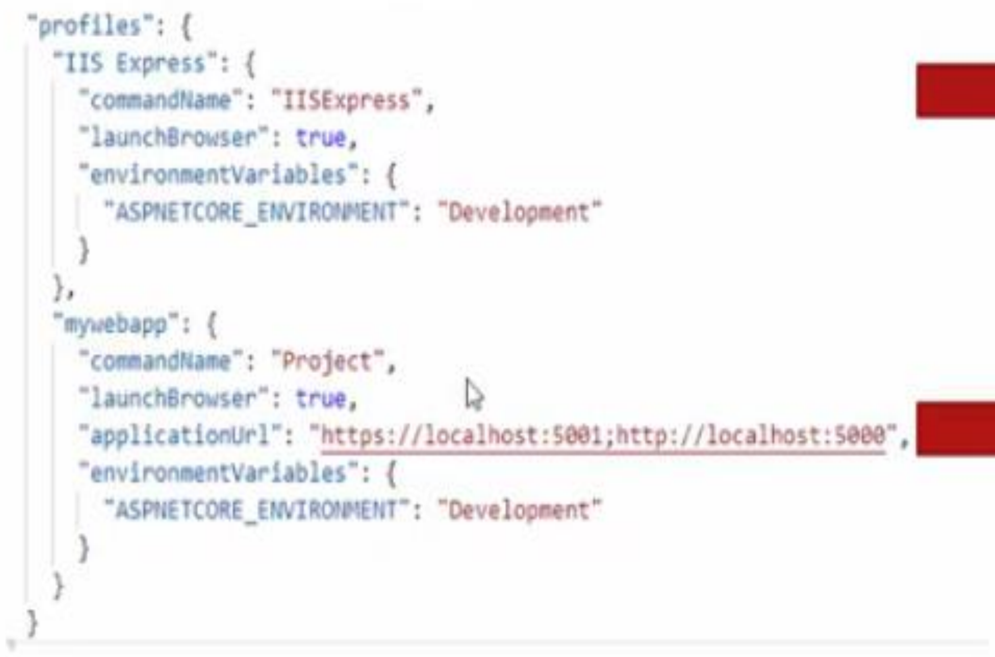
LaunchSetting.json:-This file only use for use for local development setting not for production server(remote server).



```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:41131",
      "sslPort": 44363
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "mywebapp": {
      "commandName": "Project",
      "launchBrowser": true,
      "applicationUrl": "https://localhost:5001;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

launchSettings.json

- As shown in the below **launchSettings.json** file, within the profiles we have two sections i.e. IIS Express and mywebapp.



```
"profiles": {
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "mywebapp": {
    "commandName": "Project",
    "launchBrowser": true,
    "applicationUrl": "https://localhost:5001;http://localhost:5000",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
}
```

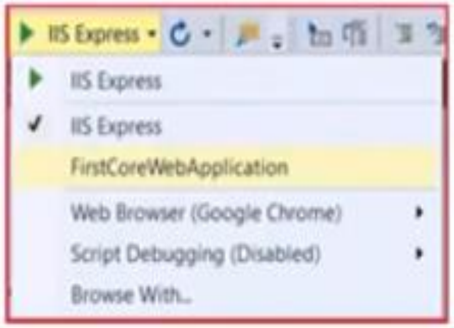
IIS Express Profile

mywebapp profile and this is our project name

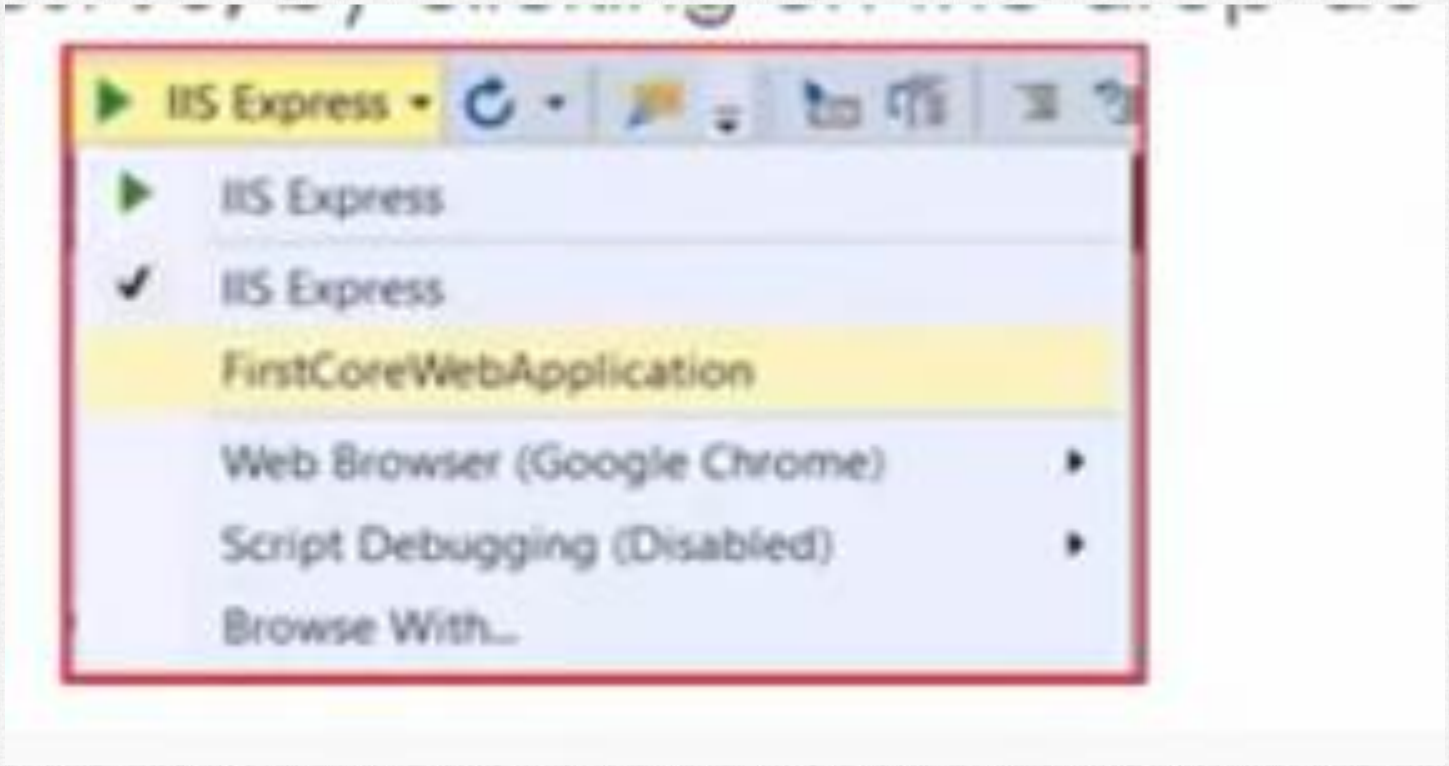
Time of running the project it will print command name:-
1)Run using VS it will print IISExpress
2)Run using CLI it will print Command name as Project name(Ex my project name Ex1_MainMethod .it print Ex1_MainMethod

Note

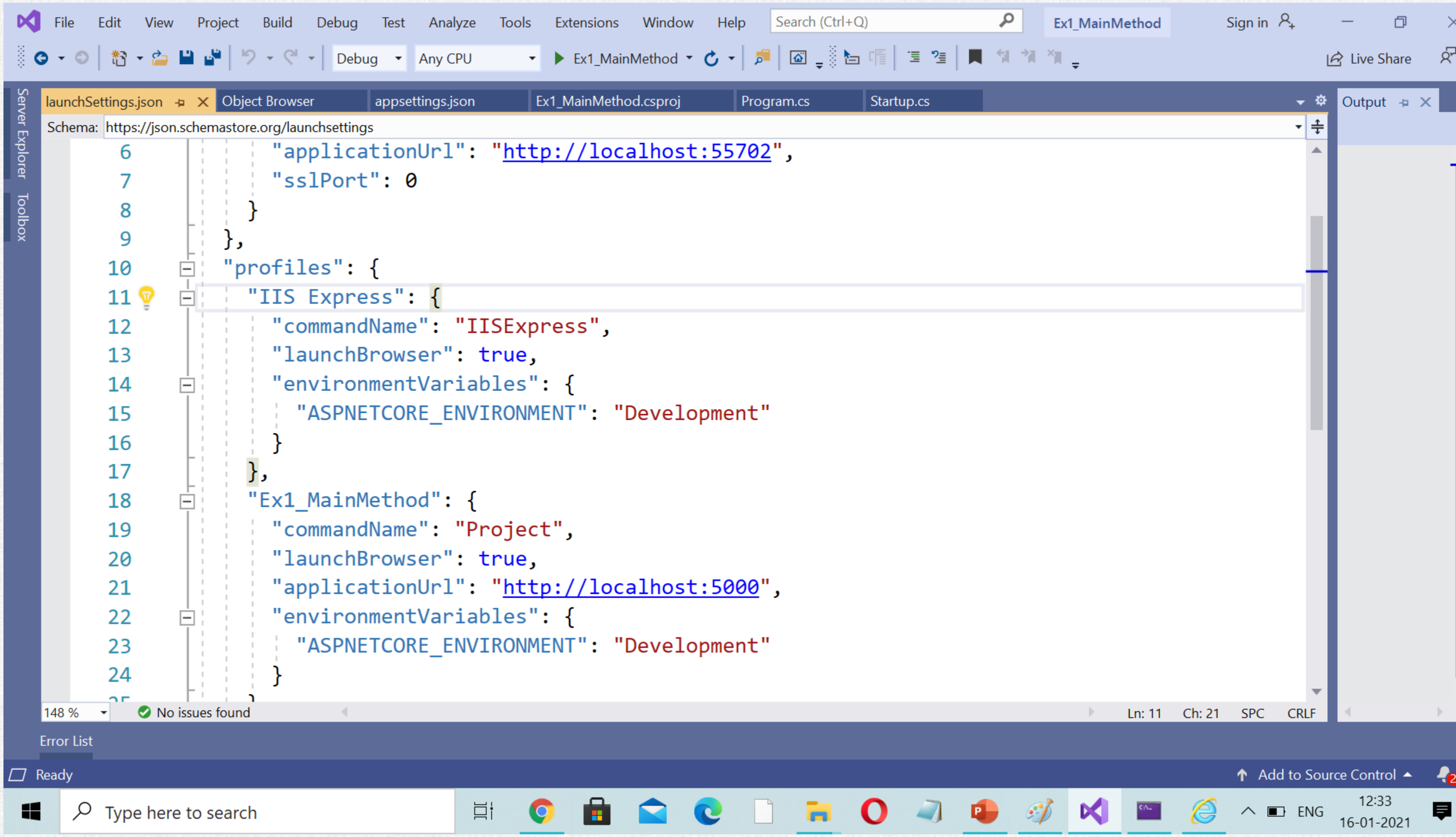
- The point that you need to remember is when you run the application from Visual Studio either by pressing **CTRL + F5** or just **F5** then by default the profile with **"commandName": "IISExpress"** is going to be used.
- On the other hand, if you run the ASP.NET Core application using .NET Core CLI (i.e. dotnet run command), then the profile with the **"commandName": "Project"** is going to be used.
- However, if you want then you can choose which profile to use when you run the application by pressing **CTRL + F5** or just **F5**, by clicking on the drop-down list in Visual Studio as shown below



- We can also decide the profile name in while profile you have to run the dotnet project.select from here

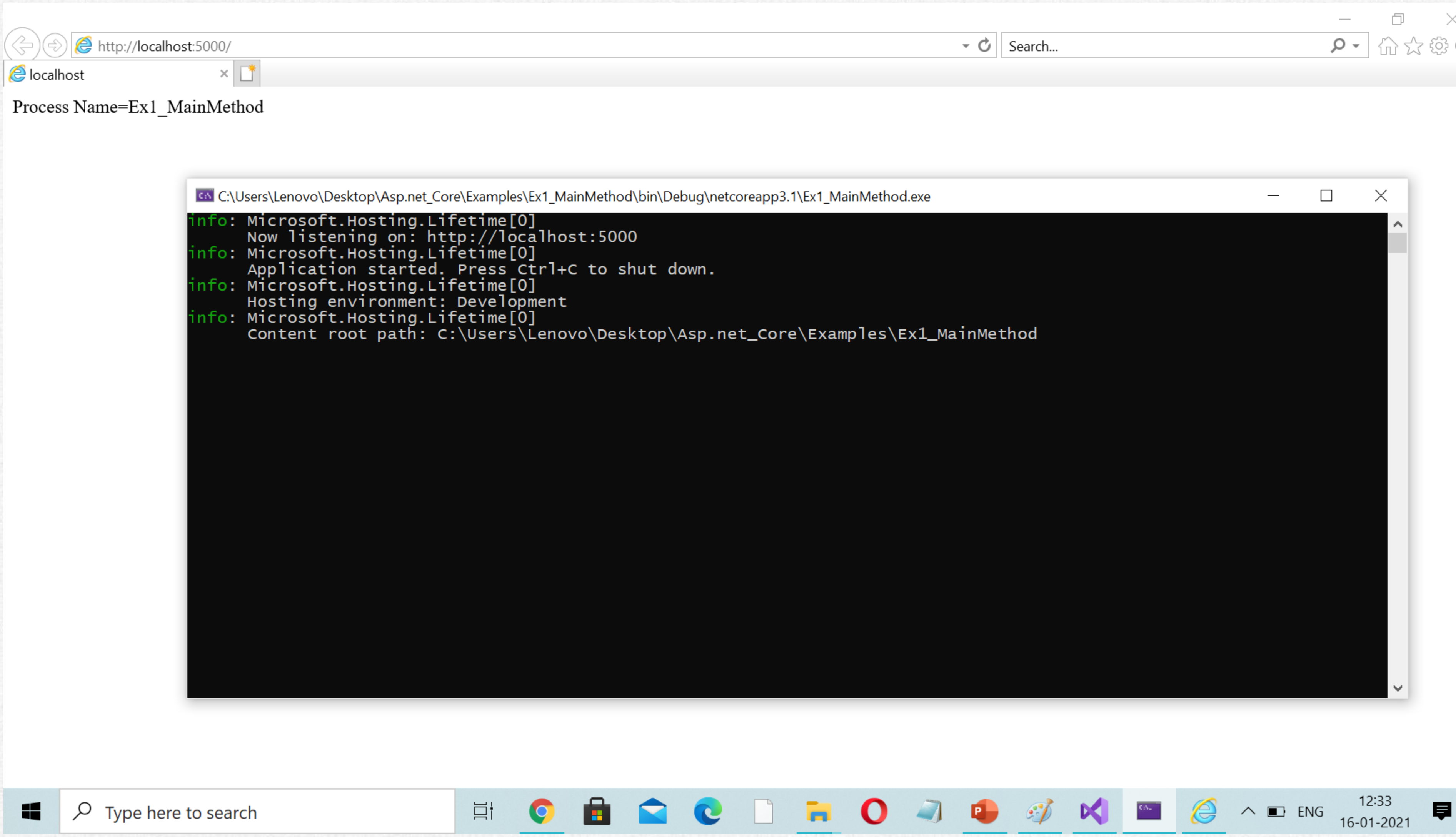


If run using Ex1_MainMethod :-Process name is print Ex1_MainMethod
If run using IISExpress :-Process name is print IISExpress.



Schema: https://json.schemastore.org/launchsettings

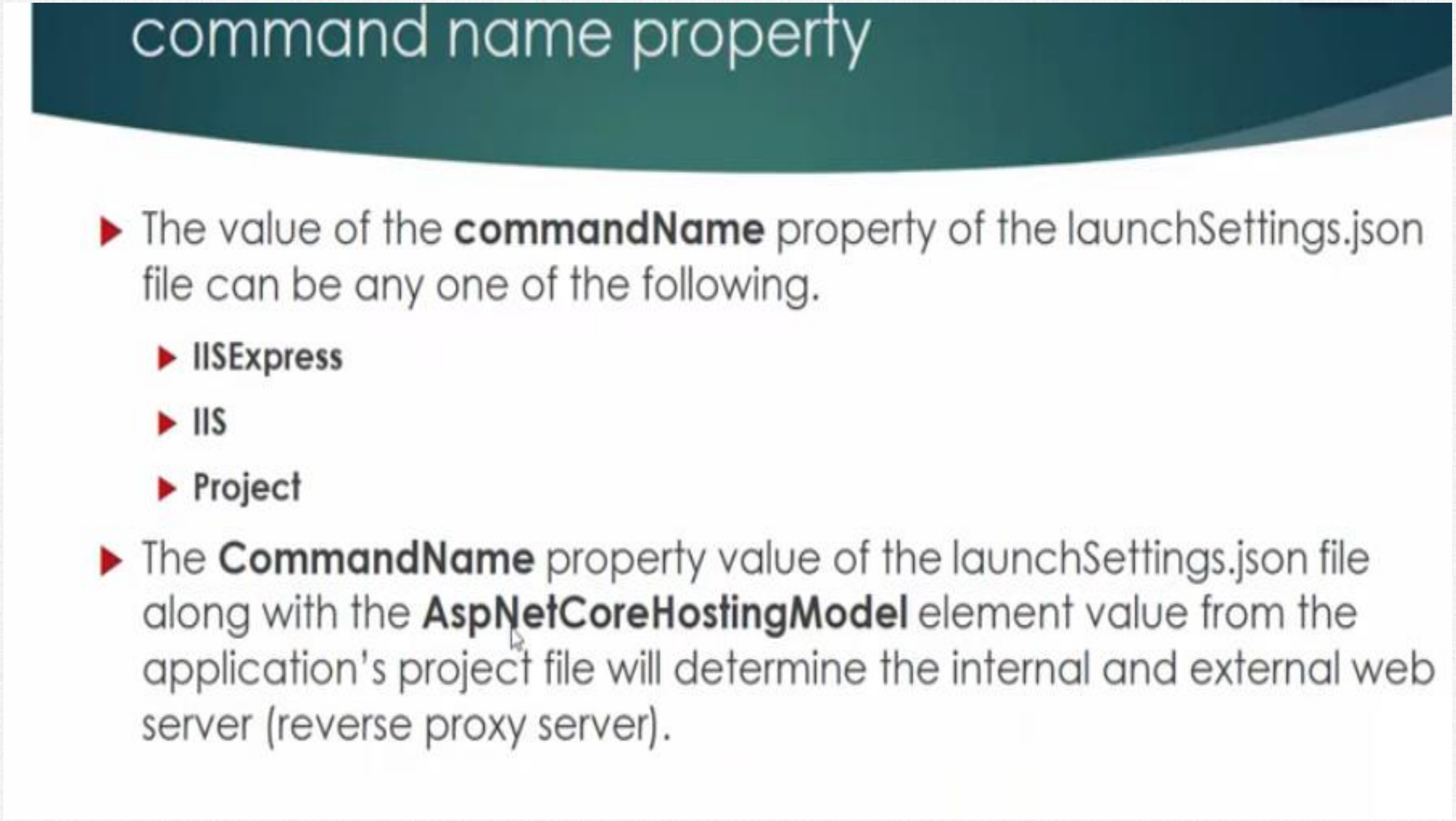
```
      "applicationUrl": "http://localhost:55702",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "Ex1_MainMethod": {
      "commandName": "Project",
      "launchBrowser": true,
      "applicationUrl": "http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
```



Process Name=Ex1_MainMethod

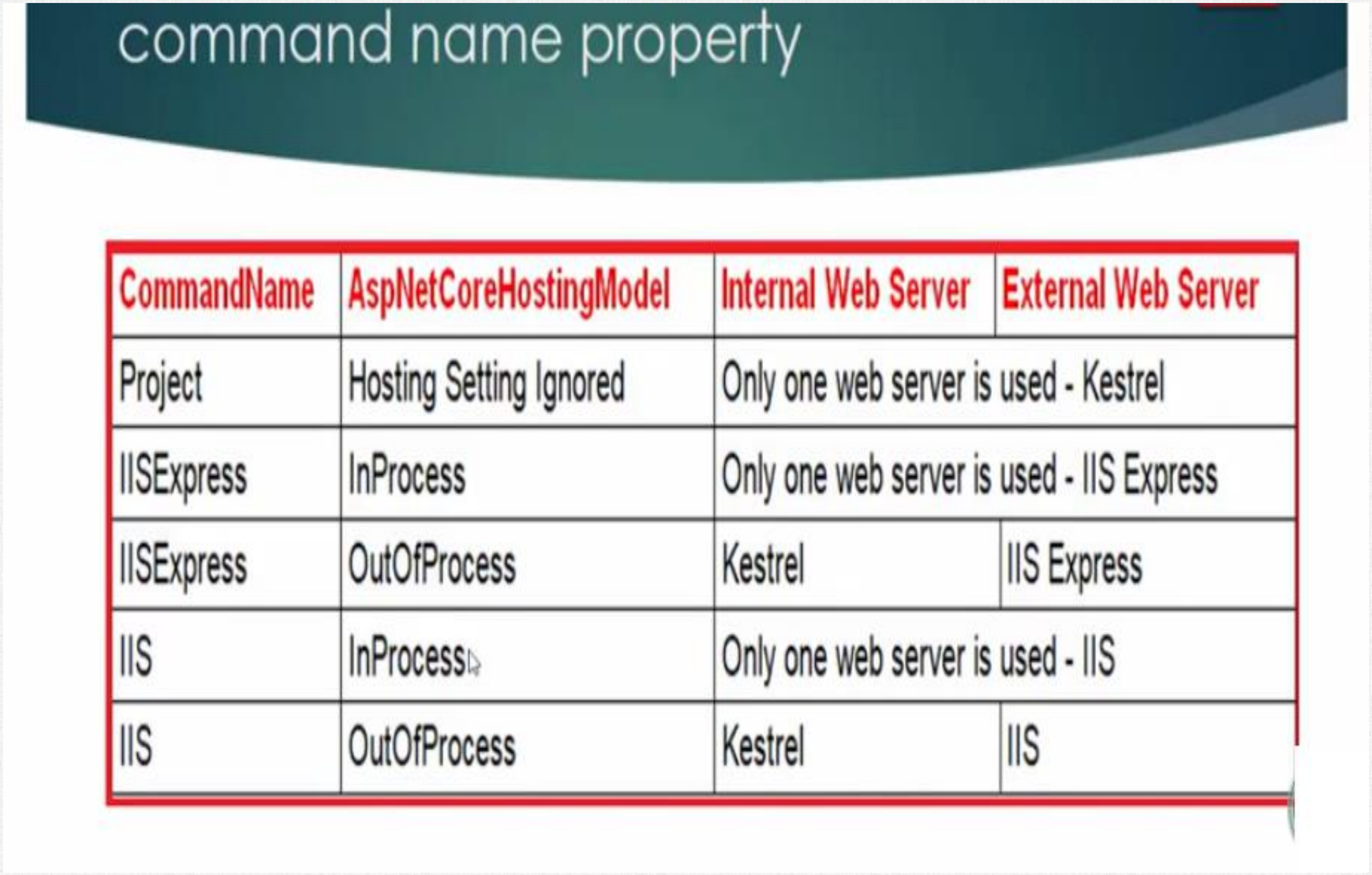
C:\Users\Lenovo\Desktop\Asp.net_Core\Examples\Ex1_MainMethod\bin\Debug\netcoreapp3.1\Ex1_MainMethod.exe

```
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Lenovo\Desktop\Asp.net_Core\Examples\Ex1_MainMethod
```



command name property

- The value of the **commandName** property of the launchSettings.json file can be any one of the following.
  - **IISExpress**
  - **IIS**
  - **Project**
- The **CommandName** property value of the launchSettings.json file along with the **AspNetCoreHostingModel** element value from the application's project file will determine the internal and external web server (reverse proxy server).

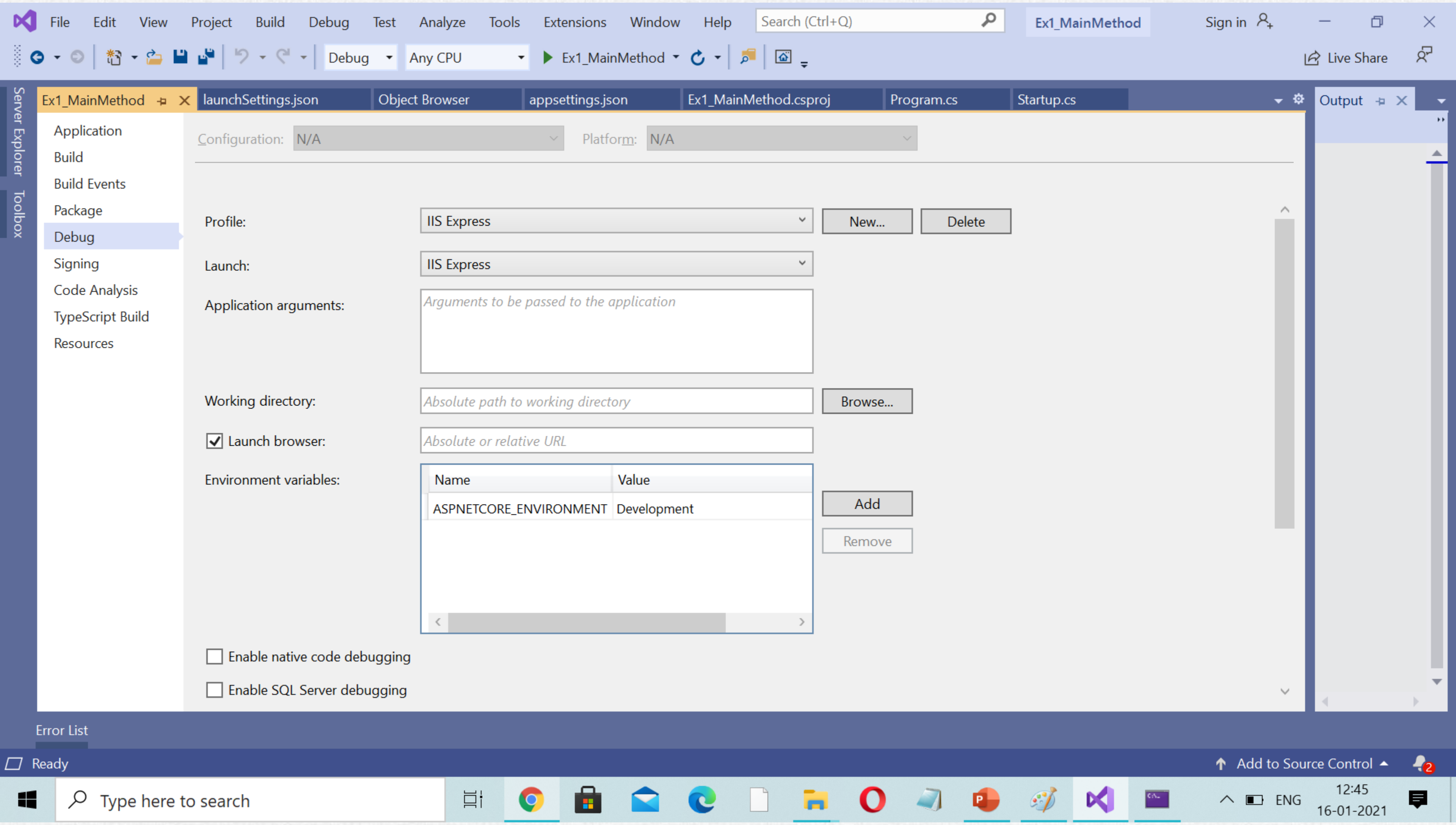


command name property

CommandName	AspNetCoreHostingModel	Internal Web Server	External Web Server
Project	Hosting Setting Ignored	Only one web server is used - Kestrel	
IISExpress	InProcess	Only one web server is used - IIS Express	
IISExpress	OutOfProcess	Kestrel	IIS Express
IIS	InProcess	Only one web server is used - IIS	
IIS	OutOfProcess	Kestrel	IIS

How to access the Graphical User Interface (GUI) in Visual Studio?

- If you want then you can also change the settings of launchSettings.json using the Graphical User Interface (GUI) provided by Visual Studio.
- Right-click on the project name in Solution Explorer and then select the "**Properties**" option from the context menu. Click on the "**Debug**" tab on the project "**Properties**" window



Configuration: N/A Platform: N/A

Profile: IIS Express

Launch: IIS Express

Application arguments: *Arguments to be passed to the application*

Working directory: *Absolute path to working directory*

[✓] Launch browser: *Absolute or relative URL*

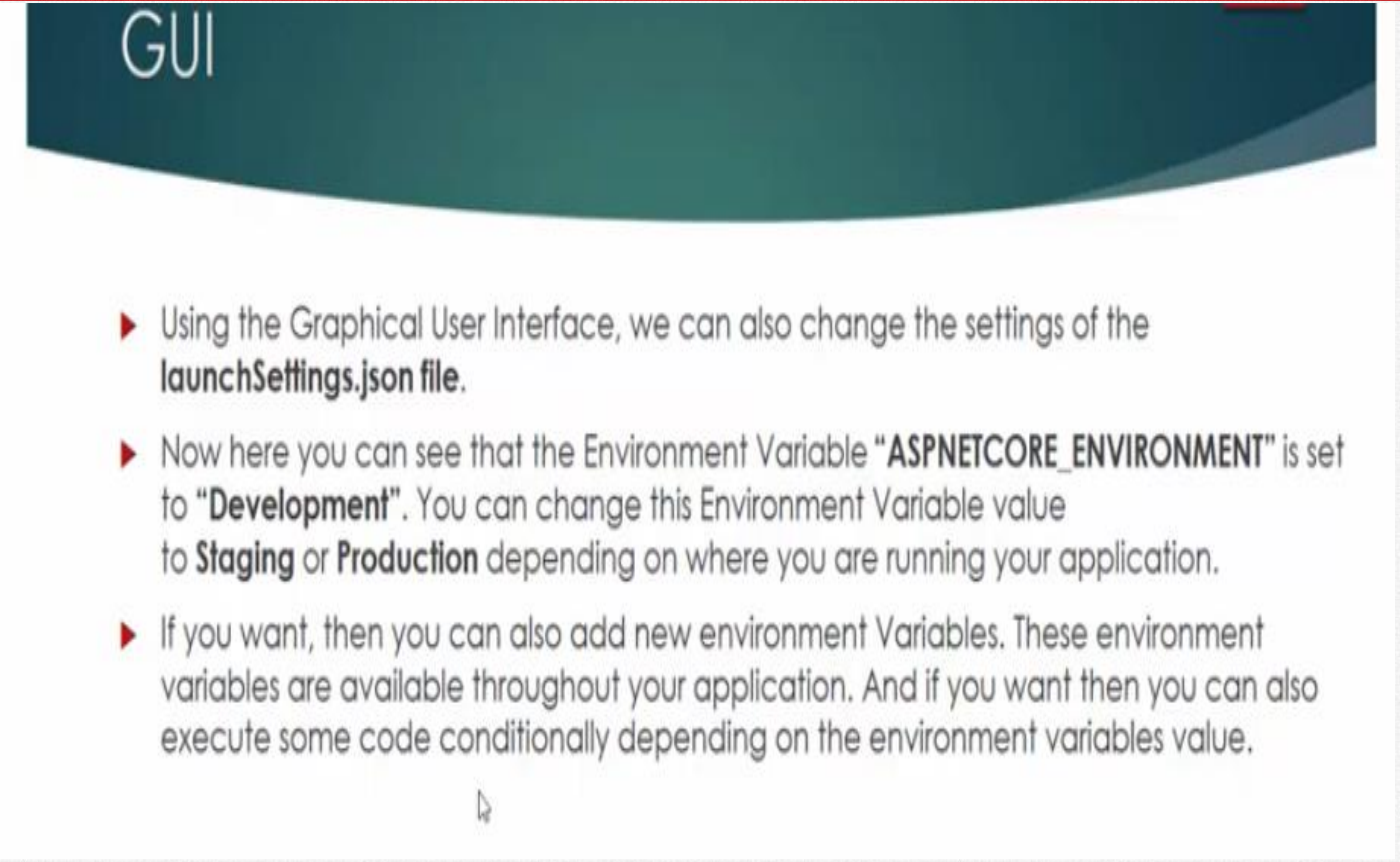
Environment variables:

Name	Value
ASPNETCORE_ENVIRONMENT	Development

[ ] Enable native code debugging

[ ] Enable SQL Server debugging

If we upload the application in main server we have to change the ASPMETCORE_ENVIROMENT AS Staging or Production in place of Development.

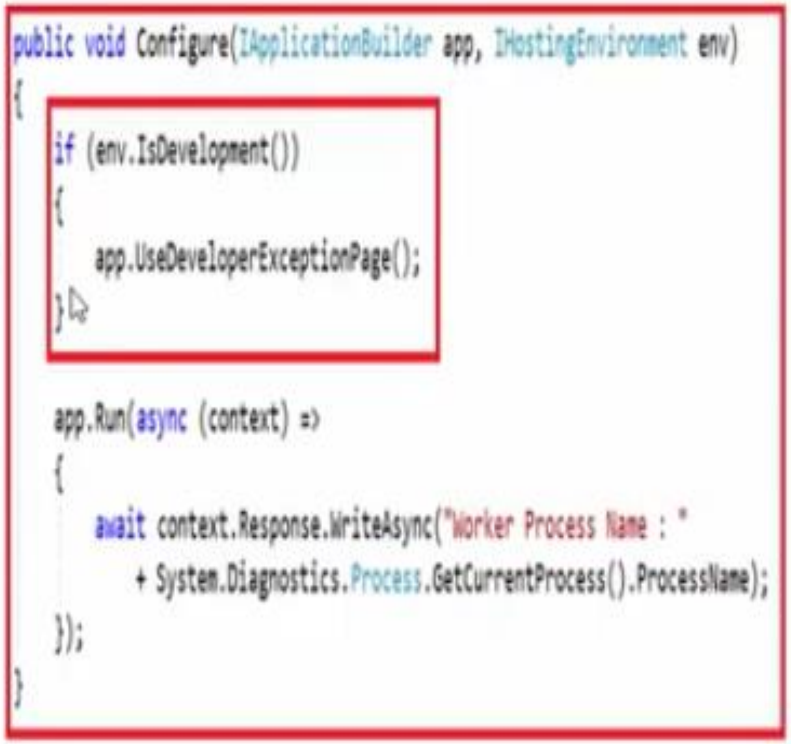


GUI

- Using the Graphical User Interface, we can also change the settings of the **launchSettings.json file**.
- Now here you can see that the Environment Variable "**ASPNETCORE_ENVIRONMENT**" is set to "**Development**". You can change this Environment Variable value to **Staging** or **Production** depending on where you are running your application.
- If you want, then you can also add new environment Variables. These environment variables are available throughout your application. And if you want then you can also execute some code conditionally depending on the environment variables value.

Example

It checks if the environment is Development, then it is going to display the Developer Exception Page. In our upcoming articles, we are going to discuss more these environment variables.



```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Worker Process Name : "
            + System.Diagnostics.Process.GetCurrentProcess().ProcessName);
    });
}
```

Thanks