How-To: Kestrel Web Server in ASP.NET Core Application

In this article, I am going to discuss the *Kestrel Web Server in ASP.NET Core Application*. Please read our previous article before proceeding to this article where we discussed *ASP.NET Core InProcess Hosting* Model. At the end of our previous article, we discussed that with the OutOfProcess hosting model, there are 2 web servers i.e. one internal web server and one external web server. The internal web server is called Kestrel and the external web server can be IIS, Apache, or Nginx. As part of this article, we are going to discuss the following two important concepts in detail.

1. What is a Kestrel Web Server?
2. How to Configure Kestrel Web Server?
3. How to run a .NET Core application using Kestrel Web Server?
4. How to run a .NET Core Application using .NET Core CLI.

What is a Kestrel Web Server?

As we already discussed ASP.NET Core is a cross-platform framework. It means it supports to develop and run applications on different types of operating systems such as Windows, Linux, or Mac.

The Kestrel is the cross-platform web server for the ASP.NET Core application. That means this Server supports all the platforms and versions that the ASP.NET Core supports. By default, it is included as the internal web server in the .NET Core application.
The Kestrel Web Server generally used as an edge server i.e. the internet-facing web server which directly processes the incoming HTTP request from the client. In the case of the Kestrel web server, the process name that is used to host and run the ASP.NET Core application is the project name.

As of now, we are using visual studio to run the ASP.NET Core application. By default, the visual studio uses **IIS Express** to host and run the ASP.NET Core application. So, the process name is **IIS Express** that we already discussed in our previous article.

How to run the application using Kestrel Web Server?

Before using the Kestrel server to run our application, let us first open the *launchSettings.json* file which is present inside the Properties folder of your application. Once you open the launchSettings.json file you will find the following code by default.

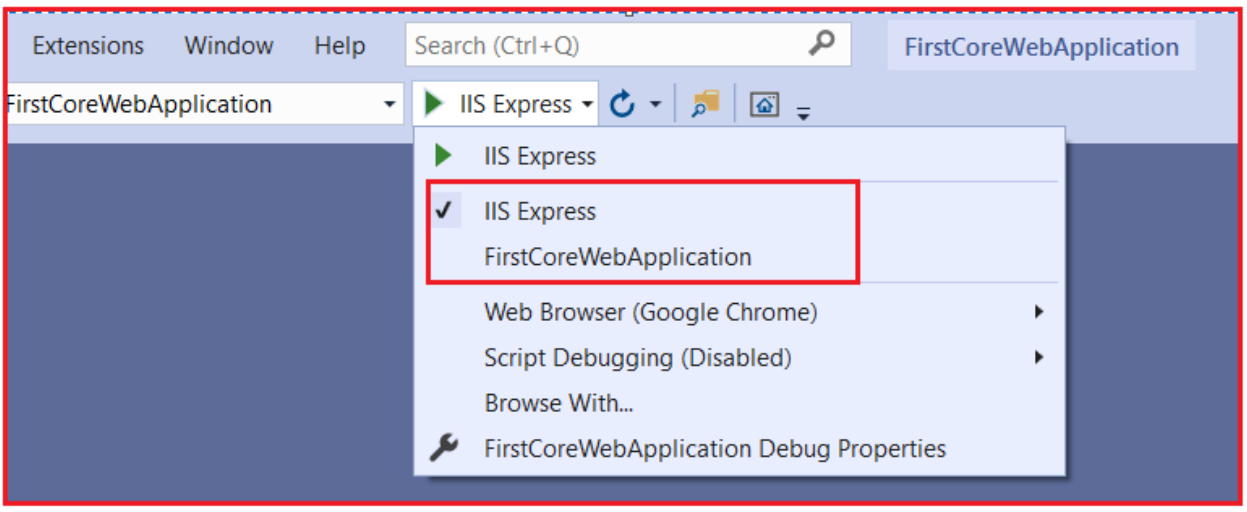
```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:60211",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "FirstCoreWebApplication": {
      "commandName": "Project",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      },
      "applicationUrl": "http://localhost:5000"
    }
  }
}
```

IIS Server (i.e. IISExpress) will use the below URL to run your application
http://localhost:60211

This Profile Settings for IISExpress

This profile is used by Kestrel Server and it will use the below URL
http://localhost:5000

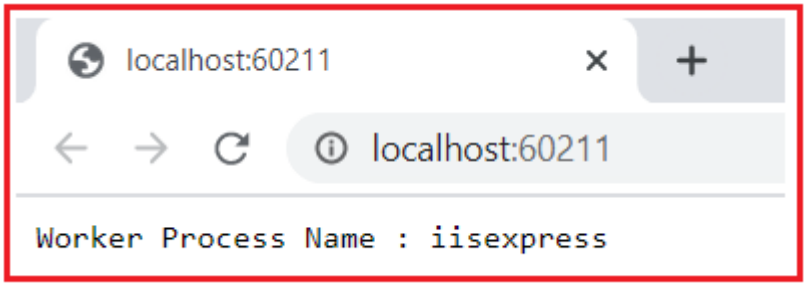
In our upcoming article, we will discuss *launchSettings.json* in detail. But for now, just have a look at the Profiles section. Here, you can see, we have two sections. One is for IIS Express (IIS Server) and the other one is for the Kestrel server. In visual studio, you can find the above two profiles (IIS Express and FirstCoreWebApplication) as shown below.



If you select IIS Express then it will use the IIS server and if you select FirstCoreWebApplication, then it will use Kestrel Server.

Running the application using IIS Express:

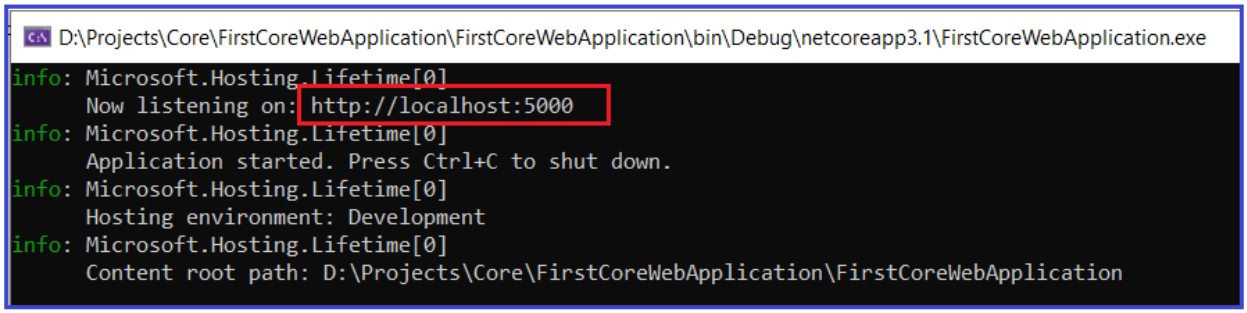
If you run the application using IIS Express, then it will use the URL and port number mentioned in the iisSettings of your launchSettings.json file. To prove this run the application using IIS Express and see the output as shown below.



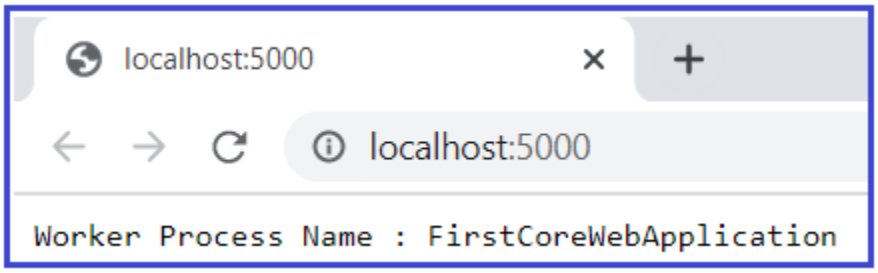
Running the application using Kestrel Server:

In order to use the Kestrel server to run your application in Visual Studio, first, you need to select the FirstCoreWebApplication profile as shown below.

Once you select the FirstCoreWebApplication, now run the application. Here, we need to observe two things. First, it will launch the command prompt and host the application using the Kestrel server as shown below. Here, you need to focus on the URL and port number and it should be the URL and port number mentioned in your FirstCoreWebApplication profile of launchSettings.json file.



Secondly, it opens the default browser and listening to that URL and Port Number as shown below.

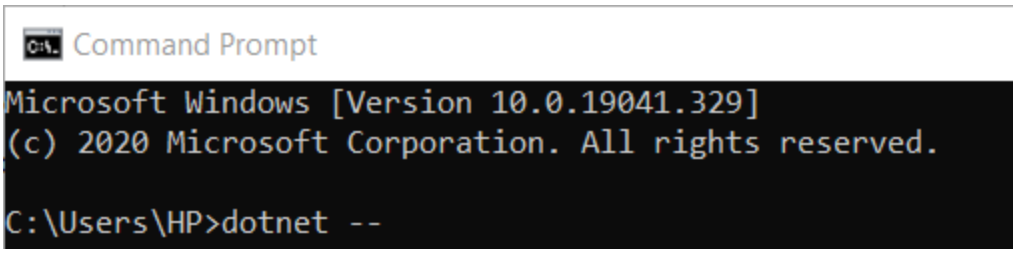


Note: In our example, for IIS Express the port number is 60211, and worker process is iisexpress while for Kestrel server the port number is 5000 and the worker process name is FirstCoreWebApplication (It is nothing but your application name).

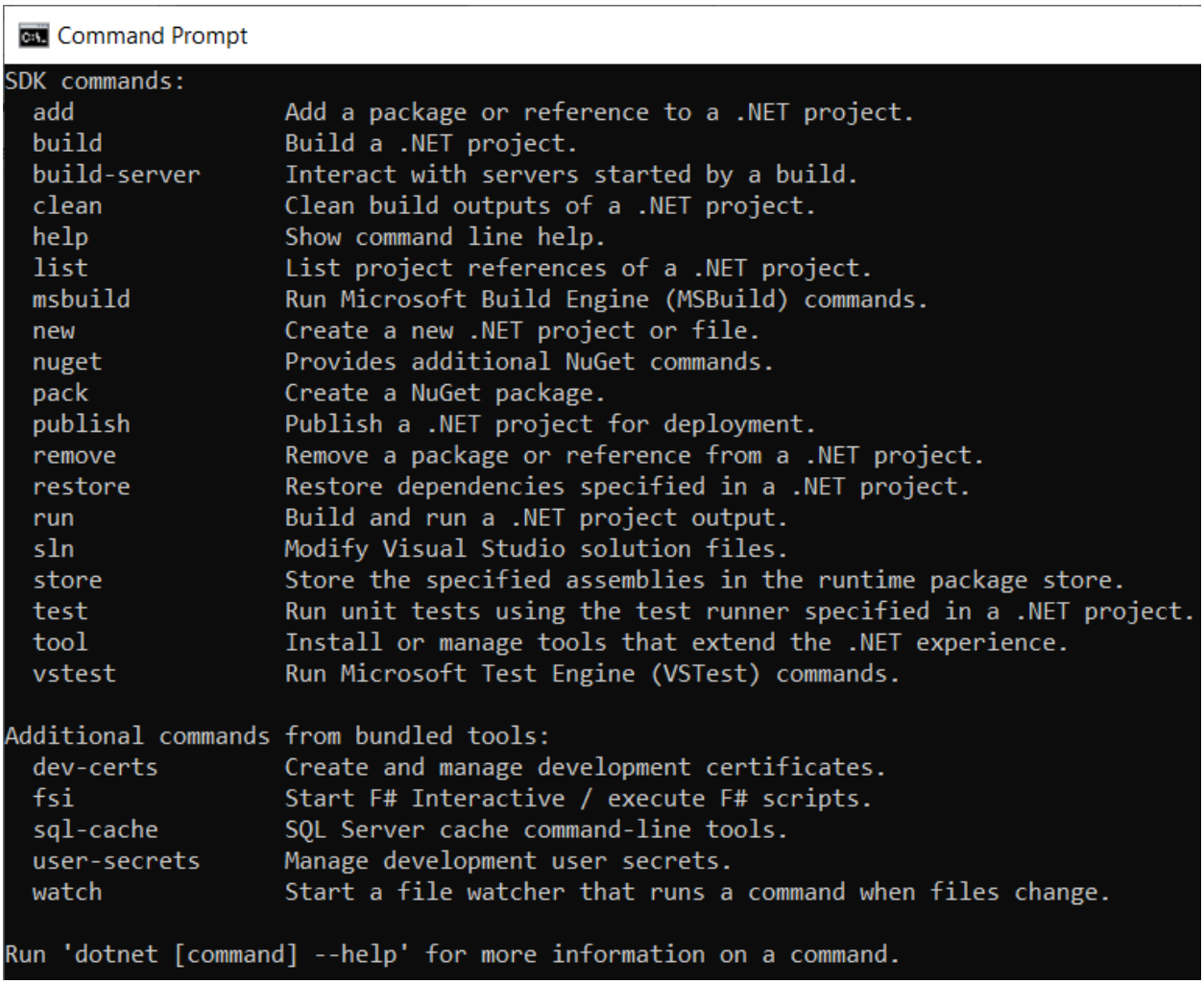
How to run .NET Core application using .NET Core CLI?

We can also run the ASP.NET Core application from the command line using the .NET Core CLI. The CLI stands for Command Line Interface.
When we run an ASP.NET Core application using the .NET Core CLI, then the .NET Core runtime uses Kestrel as the webserver. We will discuss the .NET Core CLI in detail in our upcoming article. Now, let us see how to run a dot net core application using .NET Core CLI Command.

First. open the command prompt and type "**dotnet —**" and press enter as shown below.



Once you type the "**dotnet —**" and click on the enter button then you will find lots of commands as shown below.



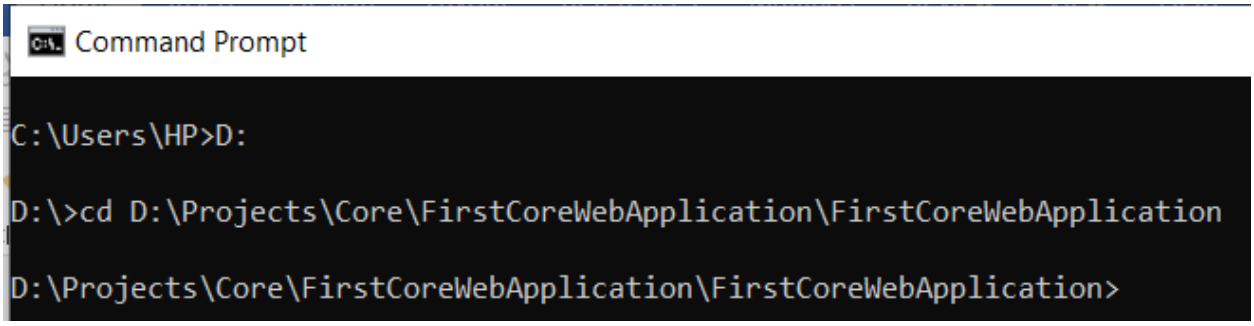
Using the CLI (above commands)

You can create a new project using the **new** command, you can also build the project using the **build** command, or you can publish the project using the **publish** command. It is possible to restore the dependencies and tools which are required for a .net core project using the CLI.

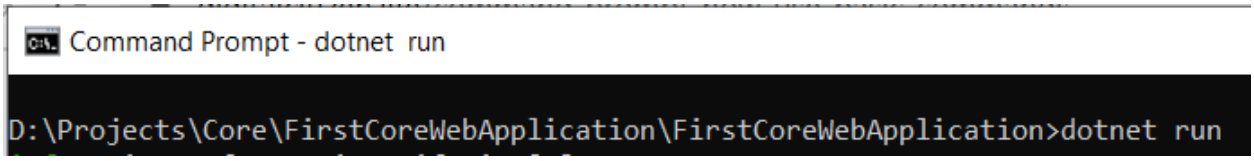
Running .NET Core application using .NET Core CLI

Let's see how to run a .NET Core application using .NET Core CLI command. To do so please follow the below steps

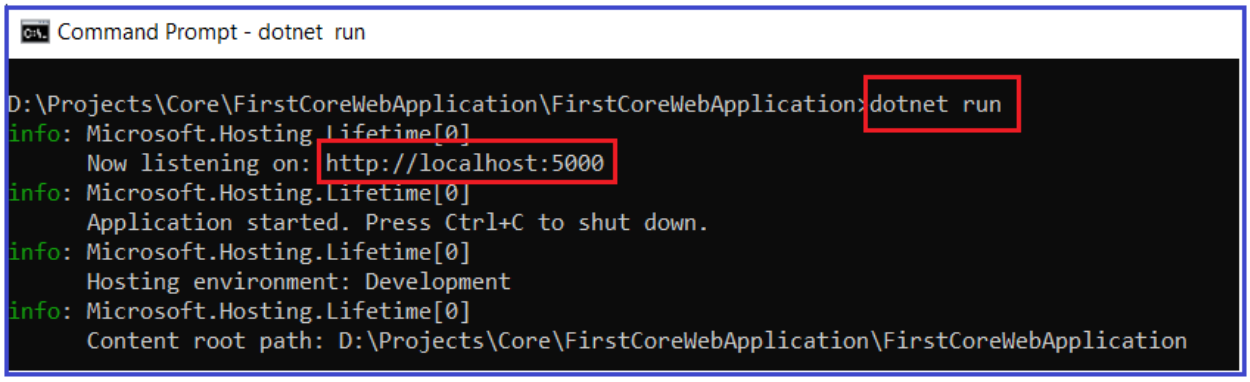
First, open the Windows Command Prompt. To do so, open the run window and then type cmd and click on the enter button which will open the command prompt. Then you need to change the directory to the folder which contains your asp.net core application. My project is present in the "**D:\Projects\Core\FirstCoreWebApplication\FirstCoreWebApplication**" folder so I change the current directory to my project file by using the following command.



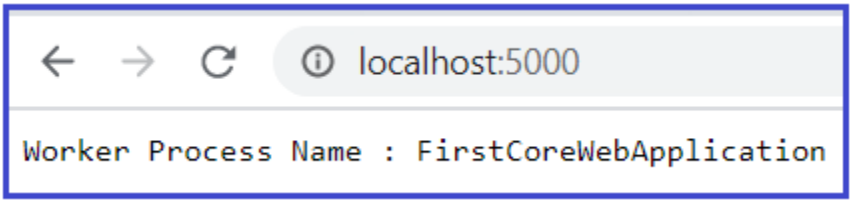
Once you change the directory to your project folder, then execute the "**dotnet run**" command as shown in the below image.



Once you type the **dotnet run** command, press the enter key, then the .NET Core CLI builds and runs the application. It also shows the URL and you can use this URL to access your application as shown in the below image.



Here, in my case, the application is available at **http://localhost:5000.** If you remember this port is configured in the launchSettings.json file of your application inside the FirstCoreWebApplication profile which is nothing but the profile for the Kestrel server. Now open the browser and navigate to the **http://localhost:5000** URL and It should display the worker process name as **dotnet** as shown below.



Changing the Port Number:

If you want then you can also change the Port number for Kestrel Server. To do so open the launchSettings.json file and give any available Port number as shown below. Here, I am changing the Port number to 60222.

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:60211",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "FirstCoreWebApplication": {
      "commandName": "Project",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      },
      "applicationUrl": "http://localhost:60222"
    }
  }
}
```

Now, save the changes and run the application using Kestrel Server and you should see the changed port number in the URL.

In the next article, we will discuss the *OutOfProcess hosting in the ASP.NET Core* application. Here, in this article, I try to explain the *Kestrel Web Server in ASP.NET Core* application in detail. I hope this article will help you to understand the Kestrel Web Server in ASP.NET Core Application.

Courtesy: https://dotnettutorials.net/lesson/kestrel-web-server-asp-net-core/

Modified: 2021.10.04.7.29.AM
Dököll Solutions, Inc.