Walkthrough 2-A - ASP.NET Core - Introduction

Setup - MinWebServer

This part of the walkthrough will set up a minimum web server.

1. Start Visual Studio.
2. Click Create a new project.
3. Set language to **C#** and project type to **Console**.
4. Select the Console Application template, click Next.
5. Set Project name to **MinWebServer**.
6. Set Location to **a folder of your choosing**.
7. Ensure Place solution and project in the same directory is not selected, click Next.
8. Ensure Target Framework is .NET 5.0, click Create.

Program.cs

1. Press Ctrl+F5 to run the program. The output is as expected.
2. Change the name of the class to App.
3.
```
class ~~Program~~App
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```
4. Run the program. The output is unchanged.
5. In the Solution Explorer, change the name of the file to **App.cs**.
6. Run the program. The output is unchanged.
7. Remove the Main method's parameters.
8.
```
class App
{
    static void Main(~~string[] args~~)
    {
        Console.WriteLine("Hello World!");
    }
}
```
9. Run the program. The output is unchanged.
10. Change the return type to string and return a string.
11.
```
class App
{
    static ~~void~~string Main()
    {
        Console.WriteLine("Hello World!");
        return "hello";
    }
}
```
12. Run the program. It doesn't compile; inspect the error.
13. Change the return type to int and update the return statement.
14.
```
class App
{
    static ~~string~~int Main()
    {
        Console.WriteLine("Hello World!");
        return ~~"hello"~~0;
    }
}
```
15. Run the program. The output is unchanged.
16. Change the return type back to void and remove the return statement.
17.
```
class App
{
    static ~~int~~void Main()
    {
        Console.WriteLine("Hello World!");
        ~~return 0;~~
    }
}
```
18. Run the program. The output is unchanged, but notice that it still returns a 0.
19. Add a second Main method with the original method signature.
20.
```
class App
{
    static void Main()
    {
        Console.WriteLine("Hello World!");
```

```
    }

    static void Main(string[] args)
    {
        Console.WriteLine("Main");
    }
}
```

21. Run the program. It doesn't compile; inspect the error.
22. Remove the second Main method.
23.
```
class App
{
    static void Main()
    {
        Console.WriteLine("Hello World!");
    }

    ~~static void Main(string[] args)~~
    ~~{~~
        ~~Console.WriteLine("Main");~~
    ~~}~~
}
```

24. Run the program. The output is unchanged.
25. Change the name of the Main method.
26.
```
class App
{
    static void ~~Main~~First()
    {
        Console.WriteLine("Hello World!");
    }
}
```

27. Run the program. It doesn't compile; inspect the error.
28. Change the name back to Main.
29.
```
class App
{
        static void ~~First~~Main()
    {
        Console.WriteLine("Hello World!");
    }
}
```

30. Run the program. The output is unchanged.
31. In Solution Explorer, right-click the project and select Manage NuGet Packages...
32. Click the Browse tab and search for **kestrel**. Install Microsoft.AspNetCore.Server.Kestrel.
33. Delete the console write and begin creating a new web server. Invoke quick actions (Ctrl+.) to add the **using Microsoft.AspNetCore.Hosting;** directive.
34.
```
using Microsoft.AspNetCore.Hosting;
using System;

namespace MinWebServer
{
    class App
    {
        static void Main()
        {
            ~~Console.WriteLine("Hello World!");~~
            new WebHostBuilder()
        }
   }
}
```

35. Continue, use quick actions to add the **using Microsoft.AspNetCore.Builder;** directive when you get to Run().
36.
```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using System;

namespace MinWebServer
{
    class App
    {
        static void Main()
        {
            new WebHostBuilder()
                .UseKestrel()
                .Configure(app => app.Run(context => context.Response))
        }
    }
}
```

37. Continue, use quick actions to add the **using Microsoft.AspNetCore.Http;** directive when you get to WriteAsync.
38.
```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using System;

namespace MinWebServer
{
    class App
    {
        static void Main()
```

```
        {
            new WebHostBuilder()
                .UseKestrel()
                .Configure(app => app.Run(context => context.Response.WriteAsync))
        }
    }
}
```

39. Finish the statement.
40.
```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using System;

namespace MinWebServer
{
    class App
    {
        static void Main()
        {
            new WebHostBuilder()
                .UseKestrel()
                .Configure(app => app.Run(context => context.Response.WriteAsync("<h1>Minimum Web Server</h1>")))
                .Build()
                .Run();
        }
    }
}
```

41. Run the program.
42. From the console window that appears, copy one of the URLs and paste it into a browser.
43. In the console window, shut down the application.
44. From the File menu, close the solution.