Walkthrough 2-B - ASP.NET Core - Introduction

Setup - IntroMVC

This part of the walkthrough will introduce MVC.

1. Click Create a new project.
2. Set language to **C#** and project type to **Web**.
3. Select the ASP.NET Core Empty template, click Next.
4. Set Project name to **IntroMVC**.
5. Set Location to **a folder of your choosing**.
6. Ensure Place solution and project in the same directory is not selected, click Next.
7. Set version to **.NET 5.0**, unselect Configure for HTTPS, click Create.
8. Run the site.

Program.cs

1. Break apart the Main method to better understand what it's doing.
2.
```
public static void Main(string[] args)
{
    CreateHostBuilder(args).Build().Run();
    var hostBuilder = CreateHostBuilder(args);
    var host = hostBuilder.Build();
    host.Run();
}
```

3. Convert the CreateHostBuilder lambda function call into a "regular" method.
4.
```
public static IHostBuilder CreateHostBuilder(string[] args) =>
{
    return Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
}
```

5. Break apart the CreateHostBuilder method to better understand what it's doing.
6.
```
public static IHostBuilder CreateHostBuilder(string[] args)
{
    return Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
    var hostBuilder = Host.CreateDefaultBuilder(args);
    hostBuilder.ConfigureWebHostDefaults(webBuilder =>
    {
        webBuilder.UseStartup<Startup>();
    });
    return hostBuilder;
}
```

7. Put the cursor on the CreateDefaultBuilder method of the var hostBuilder = Host.CreateDefaultBuilder(args); statement. Right-click it and select Go To Definition (F12).
8. Expand public static IHostBuilder CreateDefaultBuilder(string[] args); and peruse the comments. Close the tab.
9. Put the cursor on the var hostBuilder = CreateHostBuilder(args); line of the Main method. Press F9 to add a breakpoint.
10. Press F5 to run in debug mode.
11. Press F11 to step through the code until the browser appears with the output.
12. Press Shift+F5 to stop the debugger.
13. Remove the breakpoint.
14. Press Ctrl+F5 to run without debugging.
15. In the browser, navigate to **http://localhost:12345/hi** (12345 is an example TCP port, your value will be different); the page won't be found, because the existing middleware is only looking for the root page.

Startup.cs

1. Inspect the existing middleware and notice where the Hello World! response is written.
2. Delete the existing middleware.
3.
```
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!");
        });
    });
}
```

4. Run the site. A 404 will be returned because there is no middleware to respond.
5. Add a terminal middleware delegate to the application's request pipeline.
6.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```

7. Run the site, the heading will be returned.
8. In the browser, navigate to **http://localhost:12345/hi**; the URL won't matter, because the existing middleware responds to all requests.
9. Add another terminal middleware delegate to the application's request pipeline.
10.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));

    app.Run(async context => await context.Response.WriteAsync("Text"));
}
```

11. Run the site, the second middleware delegate isn't invoked because the first one terminates the pipeline.
12. Comment out the first middleware delegate.
13.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    //app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));

    app.Run(async context => await context.Response.WriteAsync("Text"));
}
```

14. Run the site, the other delegate responds.
15. Uncomment the first delegate and delete the second one.
16.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    //app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));

    app.Run(async context => await context.Response.WriteAsync("Text"));
}
```

17. Add a Use middleware delegate.
18.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```

19. Run the site, the headings appear in the browser.
20. Add UseFileServer which will serve static pages from wwwroot.
21.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseFileServer();

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```

22. Run the site, the headings still appear in the browser because there are no static files yet.

wwwroot

1. In Solution Explorer, right-click the project and select Add / New Folder. Name it **wwwroot**.
2. Download **wwwroot.zip (https://mycanvas.mohawkcollege.ca/courses/66435/files/11082238?wrap=1)** . Uncompress the contents and drag them into the wwwroot folder of the Solution Explorer.
3. Run the site, wwwroot/index.html appears.
4. Click the link for Page One, pageone.html appears.
5. Click the link for Page Two, since there is no pagetwo.html, the other middleware responds.

Startup.cs

Error Handling

1. Add some middleware that will throw an exception if the request contains the word invalid.
2.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseFileServer();

    app.Use(async (context, next) =>
    {
        if (context.Request.Path.Value.Contains("invalid"))
        {
            throw new Exception("ERROR!");
        }

        await next();
    });

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```
3. Run the site, navigate to **http://localhost/invalid**. A 500 will be returned indicating a server error.
4. Add some middleware to present a more detailed error message, while in development mode.
5.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseFileServer();

    app.Use(async (context, next) =>
    {
        if (context.Request.Path.Value.Contains("invalid"))
        {
            throw new Exception("ERROR!");
        }

        await next();
    });

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```
6. Run the site, navigate to **http://localhost/invalid**. A diagnostic error page will be returned.
7. Add middleware to handle an exception.
8.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseExceptionHandler("/error.html");

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseFileServer();

    app.Use(async (context, next) =>
    {
        if (context.Request.Path.Value.Contains("invalid"))
        {
            throw new Exception("ERROR!");
        }

        await next();
    });

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```
9. Run the site, navigate to **http://localhost/invalid**. The error diagnostic page still appears.
10. Notice the first if statement in the method is checking for development mode.
11. In Solution Explorer, right-click the project and select Properties. Click the Debug tab. Change the ASPNETCORE_ENVIRONMENT environment variable to **Production**. Close the Properties page.
12. Run the site, navigate to **http://localhost/invalid**. A friendly error page now displays.

13. Change ASPNETCORE_ENVIRONMENT back to **Development**. Refresh the invalid page to ensure it is changed back correctly.
14. Delete the default exception handler, the middleware that throws an error and "Hello" middleware delegates.
15.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseExceptionHandler("/error.html");

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseFileServer();

    app.Use(async (context, next) =>
    {
        if (context.Request.Path.Value.Contains("invalid"))
        {
            throw new Exception("ERROR!");
        }

        await next();
    });

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<h1>Hello from Use</h1>");
        await next();
        await context.Response.WriteAsync("<h1>Hello again from Use</h1>");
    });

    app.Run(async context => await context.Response.WriteAsync("<h1>Hello from Run</h1>"));
}
```
16. Add middleware to specify routing and a default route.
17.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

    app.UseFileServer();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```
18. Run the site. A different error occurs. This one indicates that a call to add controllers is missing.
19. Update the ConfigureServices method to configure MVC.
20.
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```
21. Run the site, its functional again.

HomeController.cs

1. Create the **Controllers** folder under the project.
2. Right-click the Controllers folder and select Add / Controller..., select MVC Controller - Empty and accept the default name of HomeController.cs.
3. Run the site, a hard refresh may be necessary. A new error will appear. This error indicates that the view for the Index action wasn't found.
4. Change the return type to a string and return one.
5.
```
public IActionResultstring Index()
{
    return View()"Hello from HomeController / Index";
}
```
6. Run the site, the output from the Index method appears.
7. Change the return type back to IActionResult and return a ContentResult.
8.
```
public stringIActionResult Index()
{
    return new ContentResult { Content = "Hello from HomeController / Index" };
}
```
9. Run the site, the output is unchanged.

Startup.cs

1. Note that the Home controller responds before the static files.
2. In the Configure method, move app.UseFileServer statement before the app.UseRouting statement.

3.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseFileServer();

    app.UseRouting();

    ~~app.UseFileServer();~~

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

4. Run the site. Note that the static page is served.
5. Move app.UseFileServer statement back to follow the app.UseRouting statement.
6.
```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    ~~app.UseFileServer();~~

    app.UseRouting();

    app.UseFileServer();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

7. Run the site. Note that the Home controller responds again.