Walkthrough 19 - Blazor Server Introduction

Setup

This lab will explore Blazor Server.

1. Start Visual Studio.
2. Click Create a new project.
3. Set language to **C#** and project type to **Web**.
4. Select the Blazor Server App template, click Next.
5. Set Project name to **BlazorServerIntro**.
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. Set version to **.NET 5.0**, unselect Configure for HTTPS, click Create.
9. Run the site and test each of the 3 pages.
10. Close the browser.

Startup.cs

1. Open Startup.cs.
2. Notice the differences in the ConfigureServics and Configure methods.
3. In Configure, notice the endpoints.MapFallbackToPage("/_Host") statement.

_Host.cshtml

1. Open Pages / _Host.cshtml.
2. In the body tag, notice the <component type="typeof(App)" ... tag.

App.razor

1. Open App.razor.
2. Notice the RouteView component and the DefaultLayout="@typeof(MainLayout)" attribute.

MainLayout.razor

1. Open Shared / MainLayout.razor.
2. Notice the @Body property. This is where other Blazor components will render.
3. In the sidebar div, notice the NavMenu tag.

NavMenu.razor

1. Open Shared / NavMenu.razor.
2. Notice the NavLink components.

Index.razor

1. Delete the existing Pages / Index.razor file.
2. Right-click the Page folder and select Add / Razor Component..., name it **Index.razor**.
3. Delete the existing code.
4.
```
<h3>Index</h3>

@code {

}
```
5. Add a page directive that establishes the routing.
6.
```
@page "/"
```
7. Add an h1 tag and a greeting.
8.
```
@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.
```
9. Add the SurveyPrompt component with the title property.
10.
```
@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.
```

```
<SurveyPrompt Title="How is Blazor working for you?" />
```

11. Run the site, it should function as before.

SurveyPrompt.razor

1. Open Shared / SurveyPrompt.razor.
2. Notice how the Title property is received and used.

Counter.razor

1. Delete the existing Pages / Counter.razor file.
2. Right-click the Page folder and select Add / Razor Component..., name it **Counter.razor**.
3. Delete the existing code.
4. ```
   ~~<h3>Counter</h3>~~

   ~~@code {~~

   ~~}~~
   ```
5. Add a page directive that establishes the routing and a header.
6. ```
   @page "/counter"

   <h1>Counter</h1>
   ```
7. In the browser, click the Counter link.
8. Add a code block.
9. ```
   @page "/counter"

   <h1>Counter</h1>

   @code {

   }
   ```
10. Declare a variable to hold the count and a method to interact with it.
11. ```
    @page "/counter"

    <h1>Counter</h1>

    @code {
        private int currentCount = 0;

        private void IncrementCount()
        {
            currentCount++;
        }
    }
    ```
12. Display the variable and add a button to call the method.
13. ```
    @page "/counter"

    <h1>Counter</h1>

    Current count: @currentCount

    <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

    @code {
        private int currentCount = 0;

        private void IncrementCount()
        {
            currentCount++;
        }
    }
    ```
14. Save the file, refresh the browser if necessary. The counter functions as before.
15. To illustrate how components can be nested, add the Index component.
16. ```
    @page "/counter"

    <h1>Counter</h1>

    <p>Current count: @currentCount</p>

    <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

    <Index />

    @code {
        private int currentCount = 0;

        private void IncrementCount()
        {
            currentCount++;
        }
    }
    ```
17. Save the file, refresh the browser if necessary. Notice the Index component.

18. Remove the Index component.
19.
```
@page "/counter"

<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

~~<Index />~~

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```
20. Save the file, refresh the browser if necessary.

WeatherForecast.cs

1. Open Data / WeatherForecast.cs.
2. This is the class that the WeatherForecastService returns.

WeatherForecastService.cs

1. Open Data / WeatherForecastService.cs
2. Notice that it is just a Plain Old C# Object (POCO) and not a web service, because with Blazor Server, all of the code runs server-side.

FetchData.razor

1. Delete the existing Pages / FetchData.razor file.
2. Right-click the Page folder and select Add / Razor Component..., name it **FetchData.razor**.
3. Delete the existing code.
4.
```
~~<h3>FetchData</h3>~~

~~@code {~~

~~}~~
```
5. Add a page directive that establishes the routing, a header, some text, and a code block.
6.
```
@page "/fetchdata"

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@code {

}
```
7. Add an @using directive to the Data namespace.
8. Declare an array of WeatherForecasts.
9.
```
@page "/fetchdata"

@using BlazorServerIntro.Data

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@code {
    private WeatherForecast[] forecasts;
}
```
10. Inject the WeatherForecastService.
11. Override the OnInitializedAsync method to get the forecasts.
12.
```
@page "/fetchdata"

@using BlazorServerIntro.Data
@inject WeatherForecastService ForecastService

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@code {
    private WeatherForecast[] forecasts;

    protected override async Task OnInitializedAsync()
    {
        forecasts = await ForecastService.GetForecastAsync(DateTime.Now);
    }
}
```

13. Display a loading message or the count of forecasts.
14.
```
@page "/fetchdata"

@using BlazorServerIntro.Data
@inject WeatherForecastService ForecastService

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@if (forecasts == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <p>Retrieved @forecasts.Count() forecast(s).</p>
}

@code {
    private WeatherForecast[] forecasts;

    protected override async Task OnInitializedAsync()
    {
        forecasts = await ForecastService.GetForecastAsync(DateTime.Now);
    }
}
```
15. Save the file, in the browser, click the Fetch data link.
16. Replace the count output with a table of forecasts.
17.
```
...
@if (forecasts == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <p>Retrieved @forecasts.Count() forecast(s).</p>
    <table class="table">
        <thead>
            <tr>
                <th>Date</th>
                <th>Temp. (C)</th>
                <th>Temp. (F)</th>
                <th>Summary</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var forecast in forecasts)
            {
                <tr>
                    <td>@forecast.Date.ToShortDateString()</td>
                    <td>@forecast.TemperatureC</td>
                    <td>@forecast.TemperatureF</td>
                    <td>@forecast.Summary</td>
                </tr>
            }
        </tbody>
    </table>
}
...
```
18. Save the file, refresh the browser if necessary. Fetch data functions as before.