Creating Your First Blazor Application

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren – xpirit.com/gill

Module overview

Exploring a new Blazor project

Creating a first Blazor application

Improving the layout

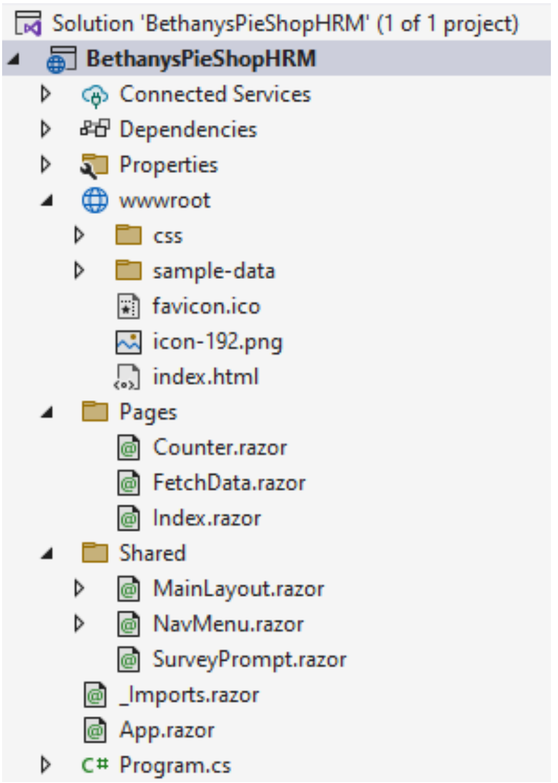
Debugging Blazor applications

Exploring a New Blazor Project

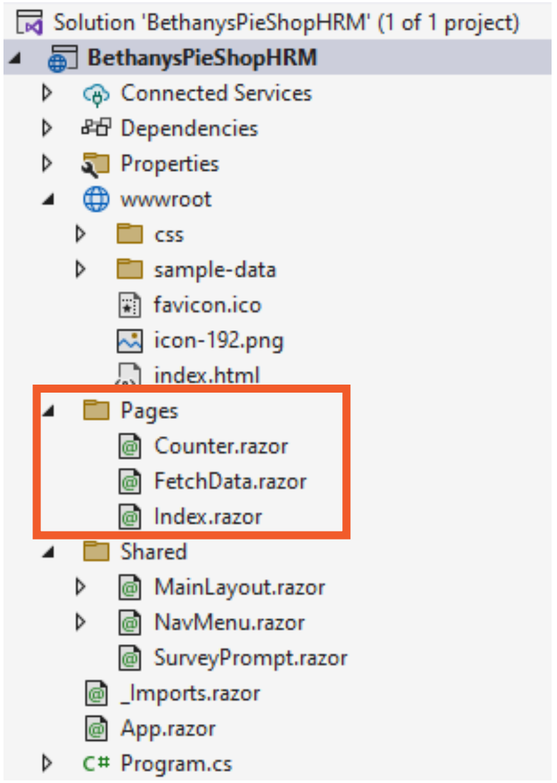
File → New Project

C# and .razor files

Structure similar to ASP.NET Core project



Razor Components in Blazor



Main building block in Blazor applications

***.razor files**

Razor syntax

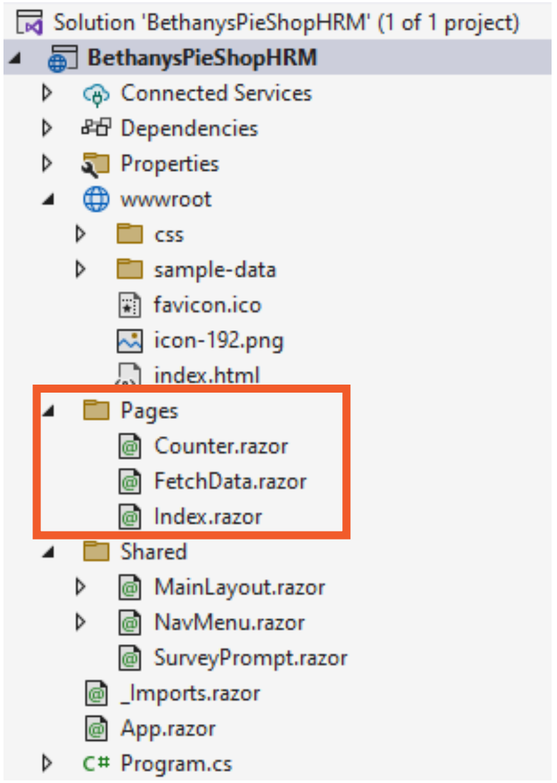
Contain UI and logic

- Handle events
- Logic typically in a component class

Rendered on the client

- Not a request/response model

Razor Components in Blazor



Often nested

Reused across the application or in a library

Class generated upon compilation

Component names must start with an uppercase!

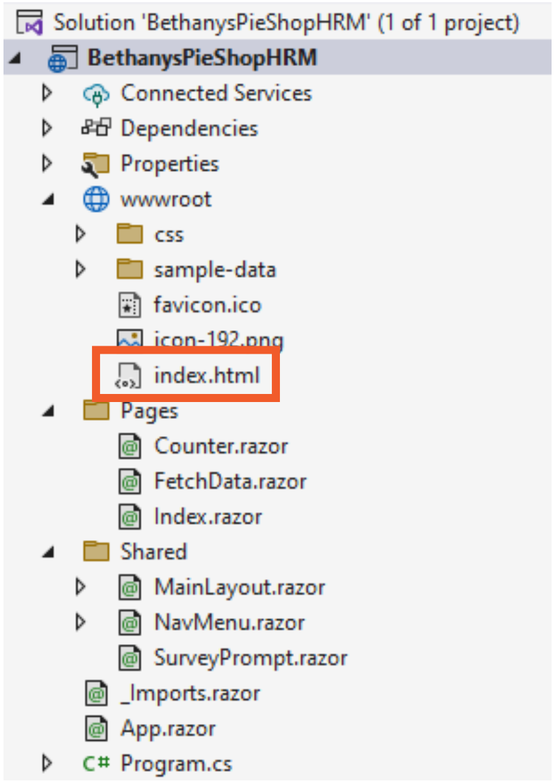
Looking at a First Component

Counter.razor

```
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    int currentCount = 0;
    void IncrementCount()
    {
        currentCount++;
    }
}
```

```
@page "/"
<h1>Hello, world!</h1>
Welcome to your new app.

<Counter />
```

Using a Component



Hosting page

Plain HTML

Triggers loading of the Blazor app through JavaScript

- blazor.webassembly.js

Demo

Creating a new Blazor WebAssembly project

Exploring the created files

New to ASP.NET Core?
Razor syntax confusing you?

Take a look at the
ASP.NET Core Learning Path!

Creating a First Blazor Application

Using Code

Mixed approach using @code

“Code behind” using partial

Using the Mixed Approach

```
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    int currentCount = 0;
    void IncrementCount()
    {
        currentCount++;
    }
}
```

```
public partial class EmployeeOverview
{
}
```

Using Partial Classes

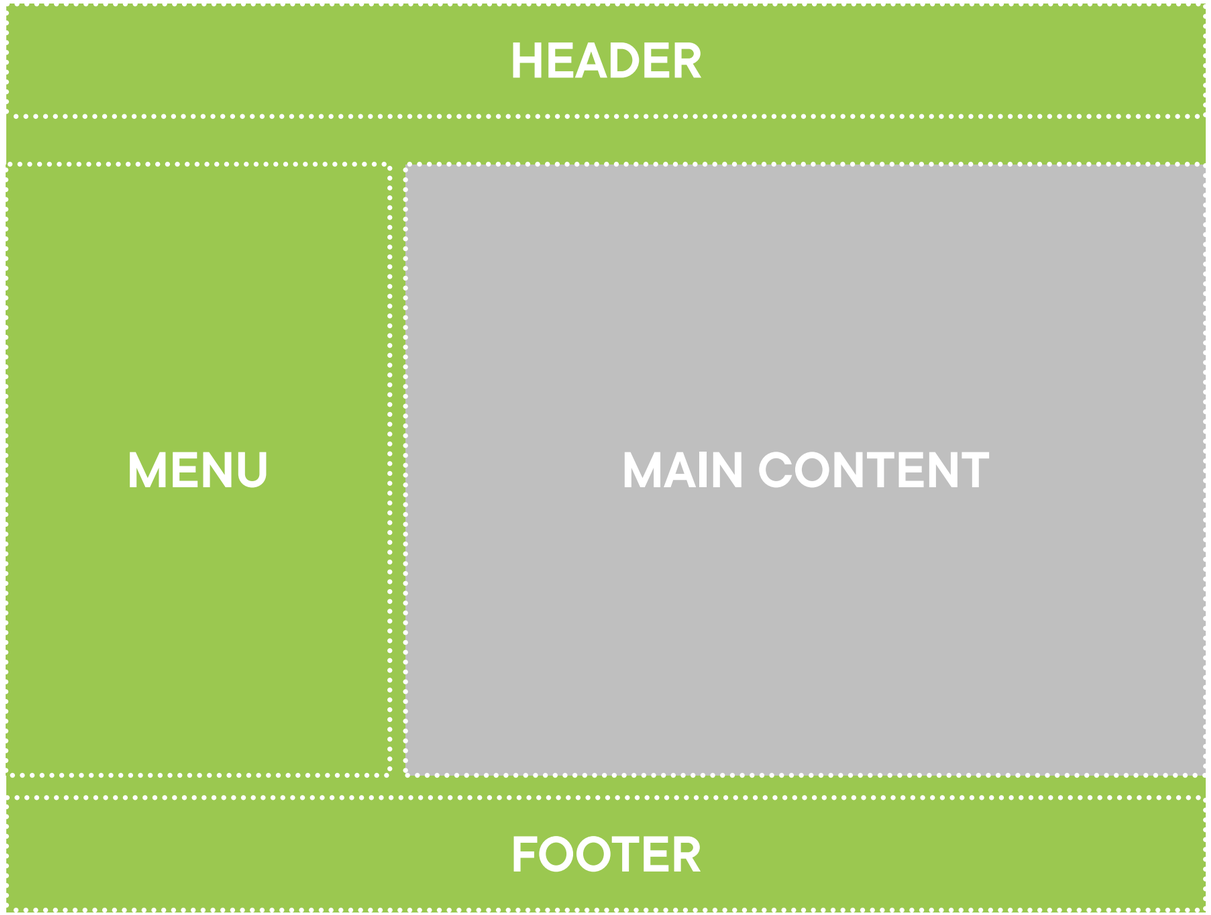
Code-behind

Demo

Creating the first page of our application

Improving the Layout

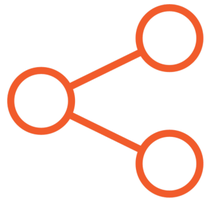
Sharing a Layout



Adding a Layout

Shared UI across several pages

Referenced by page components



Shared folder

Inherit from LayoutComponentBase

Creating a Layout Component

```
@inherits LayoutComponentBase

<div class="page">
    <div class="sidebar">
        <NavMenu />
    </div>

    <main>
        <div class="top-row px-4">
            <a href="https://docs.microsoft.com/aspnet/" target="_blank">About</a>
        </div>

        <article class="content px-4">
            @Body
        </article>
    </main>
</div>
```

```
@page "/employeeoverview"
@layout BethanysPieShopMainLayout
```

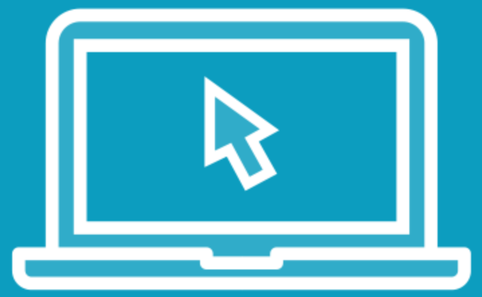
Applying a Specific Layout

```
<Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
     ...
</Router>
```

Using a Default Layout

Code from the App.razor

Demo



Improving the layout

Debugging Blazor Applications

Demo

Debugging Blazor applications

Summary

Components are the building block for any Blazor app

Can be nested, reused...

Layouts add consistency

Up next:
Working with components