

Session 2 - Building a Blazor Application Overview

A STEP-BY-STEP GUIDE



Building a Blazor Application for Hog Wild Biking

Sarah, the client, runs a small parts and service business named Hog Wild and wants to transition from a cash register to a point of sale (POS) system for improved inventory and sales management.

She researched various POS systems, focusing on features that would benefit her business, such as inventory management, customer data, employee management, invoicing, purchase orders, receiving, reporting, and reference data.

Sarah plans to start with a basic set of features and customize the POS system as her business needs evolve and staff become familiar with it.

The initial features she prioritizes include customers, employees, inventory, parts, invoicing, and maintenance.

Sarah expects the POS system to enhance inventory management, automate manual tasks, and save time.

The development of the POS system has been outsourced to a partner company, including the creation of the database and test data for the development team to work on.

Project Overview

Create our Application (Web Project and System Library) using .Net 8.x

Create a simple “Basics” web page and review common components

Review Web Project Folder Structure.

Setup System Library Folder Structure.

- Add supporting folders.
- Add dummy files.
- Add reference from HogwildSystem to HogWildWeb.
- Set HogwildWeb as a startup.

Install Components

- Paginator Component.
- Table Template Component.
- MudBlazor Component Library.

Project Overview (Continue)

Install Packages from NuGet for HogWildSystem.

- EntityFrameworkCore.Tools/SqlServer (Solutions).
- EF Core Power Tools.

Creating our first web page using the “Working Version” Data

- Create Menu.
- Create Web Page (WorkVersion.cs).

Create Entities

- Reverse engineer all tables.
- Refactor DbContext and all entity files from Public to Internal.

Project Overview (Continue)

- Add all view models.
- Create back-end dependencies.
- Update appsettings with “OLTP-DMIT2018” connection string.
- Update program.cs with connection string and dependencies injections.
- Create a service for Working Version (WorkVersionService)
- Update Back-end dependencies with reference to WorkVersionService.
- Add backend code for the “Working Version” web page.

Project Overview (Continue)

- Create Customer List
 - List and with straightforward feedback.
- Create Customer Edit
 - Dropdown lists
 - Validation using EditContext
 - Dynamic captions and enabling of buttons
 - CSS
- Create an Invoice List
 - Soft Delete
- Create Invoice Edit
 - Master/Child Relationship (Invoice & Invoice Lines)
 - Dynamic Totals







Project Overview (Continue)

- Create a Simple List To List
- Create Simple Non-Index List
- Security

Installing Visual Studio

1. Install Visual Studio from the “Microsoft Education” website
2. Follow the instructions in the “Installing Development Tools”
3. Under “Individual Components”, you will need to now include .NET 8.0 Runtime

Development Tools/Best Practices

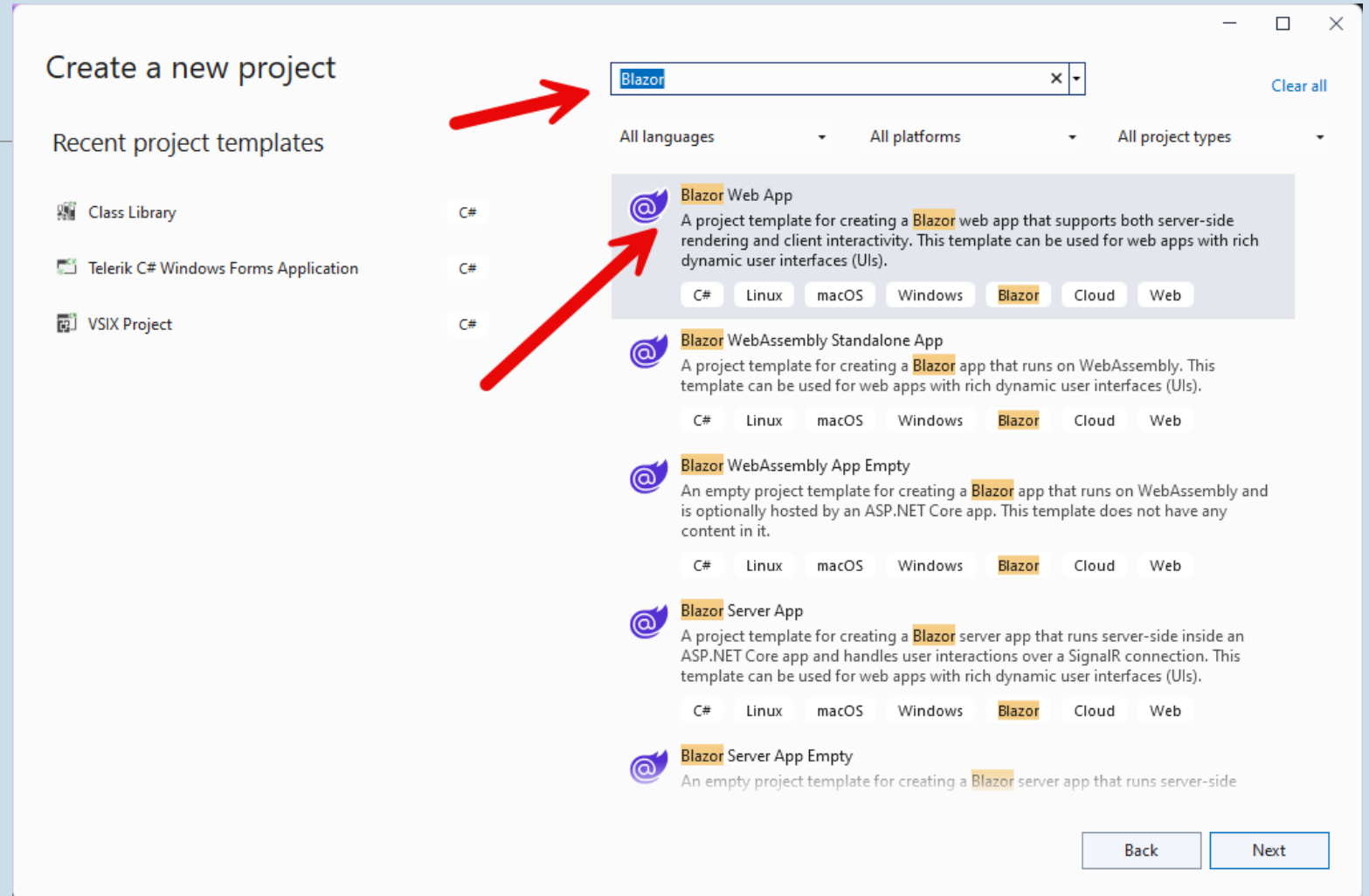
-  C# Coding guidelines
-  Microsoft Education Software
 - Visual Studio Enterprise Edition
 - SQL Server Developer
 - Visio Professional
-  CodeRush for Visual Studio
-  LINQPad - Student Version \$20.00 USD
-  Installing Development Tools
-  Installing Sql Server

Search components (Ctrl+Q) 

.NET

- ☐ .NET 5.0 Runtime (Out of support)
- ☒ .NET 6.0 Runtime (Long Term Support)
- ☐ .NET 6.0 WebAssembly Build Tools
- ☒ .NET 7.0 Runtime

Create a New Project Blazor Server App using .Net 8.x



Configure Your New Project

Configure your new project

Blazor Server App C# Linux macOS Windows Blazor Cloud Web

Project name

HogWildWebApp

Location

C:\OD\OneDrive - NAIT\DMIT 2018 - Course Files\Blazor Demo\

Solution name ⓘ

HogWild

☐ Place solution and project in the same directory

Project will be created in "C:\OD\OneDrive - NAIT\DMIT 2018 - Course Files\Blazor Demo\HogWild\HogWildWebApp\"

Back Next

Additional Information

Additional information

Blazor Web App C# Linux macOS Windows Blazor Cloud Web

Framework ⓘ
.NET 8.0 (Long Term Support)

Authentication type ⓘ
Individual Accounts

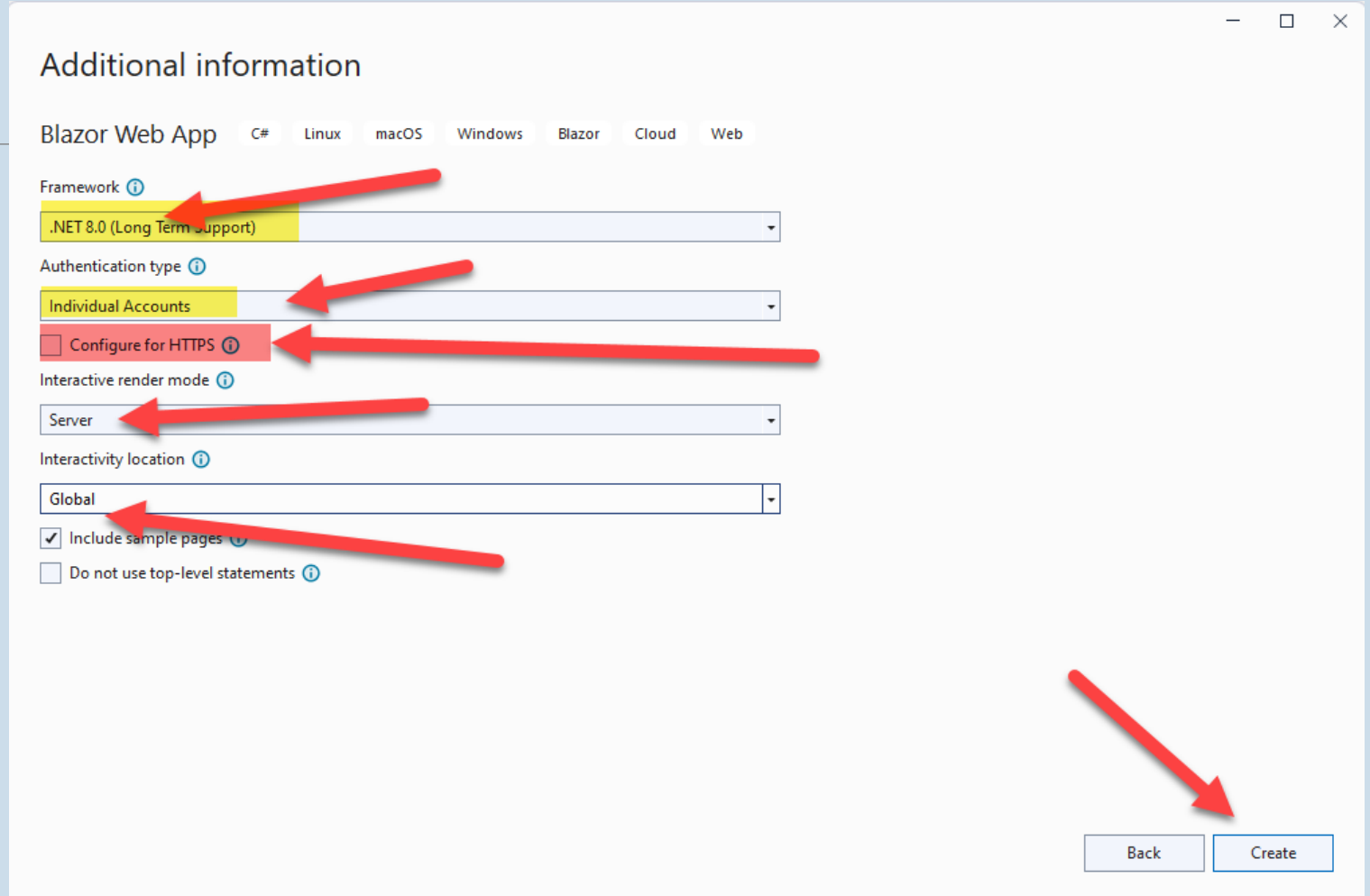
☒ Configure for HTTPS ⓘ

Interactive render mode ⓘ
Server

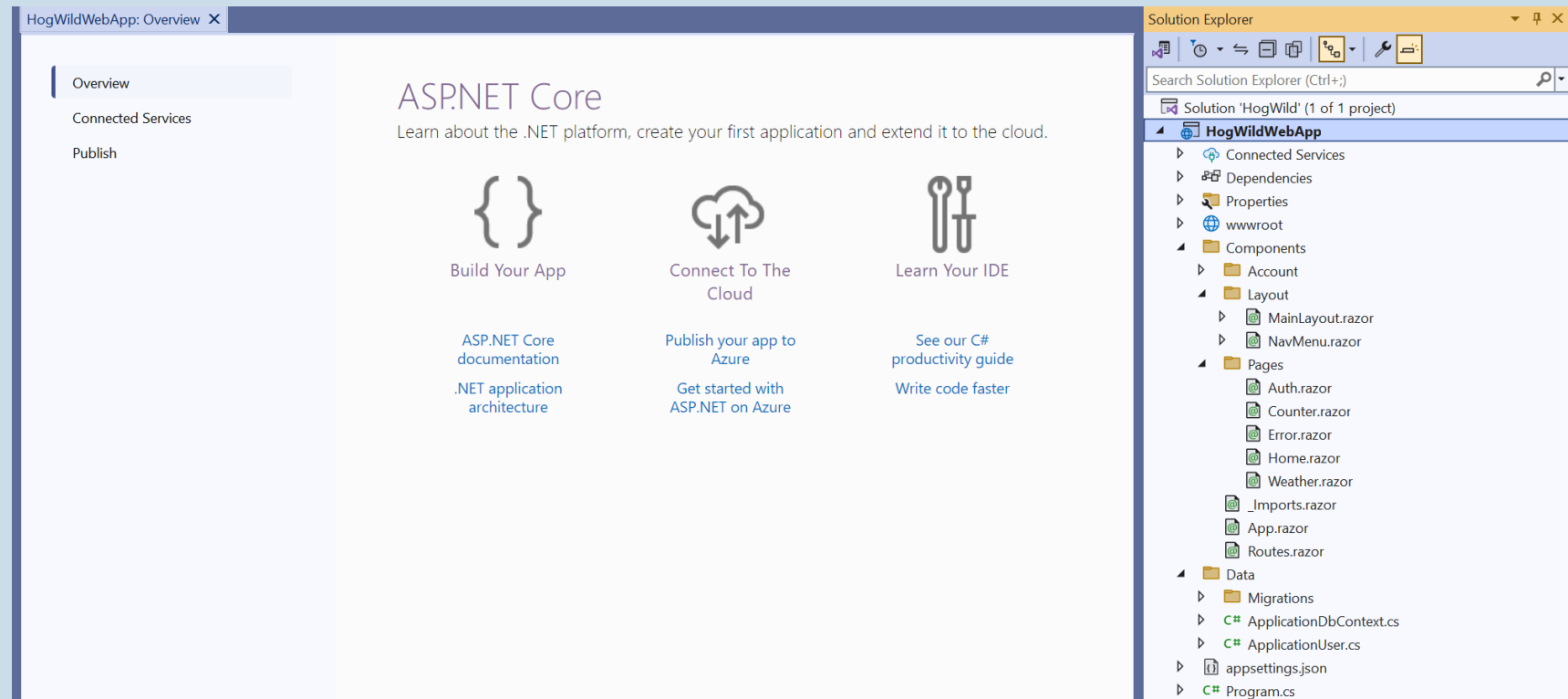
Interactivity location ⓘ
Global

☒ Include sample pages ⓘ
☐ Do not use top-level statements ⓘ

Back Create



Hog Wild Web Application



Web Application Layout

Properties Folder:

- **launchSettings.json:** Contains development environment configuration.

Components Folder:

Contains all reusable components for the app.

Layout Folder:

- **MainLayout.razor:** The main layout of the app.
- **MainLayout.razor.css:** Styles for the main layout.
- **NavMenu.razor:** Sidebar navigation with links to other pages.
- **NavMenu.razor.css:** Styles for the navigation menu.

Web Application Layout

appsettings.json: This file is used for configuration settings in your Blazor Server application. It typically includes settings such as database connection strings, API endpoints, and other configuration options. You can customize this file to suit the needs of your application.

Program.cs: This is the entry point for your Blazor Server application. It configures the host and starts the Blazor Server application. It includes the Main method, where the application is initialized and run.

Web Application Layout

Components\Pages: This is where the Razor Pages are located. Razor Pages are similar to traditional ASP.NET Web Forms or MVC Views. Each Razor Page (.razor file) combines HTML markup with C# code to create dynamic web pages.

- **_Host.cshtml** : The entry point for your Blazor Server application. It sets up the Blazor JavaScript runtime and initializes the Blazor application. This file is crucial for starting and configuring your Blazor app.
- **Counter.razor**: A sample page that demonstrates a simple counter component. It allows you to increment and decrement a counter value.
- **Error.razor**: This file that handles errors. It is displayed when an HTTP error occurs in the application. You can customize this page to provide user-friendly error messages or additional information about the error.
- **Login Razor Files**: Collection of razor pages needed to handle the login/authorization.
- **Home.razor**: The default landing page for the application.

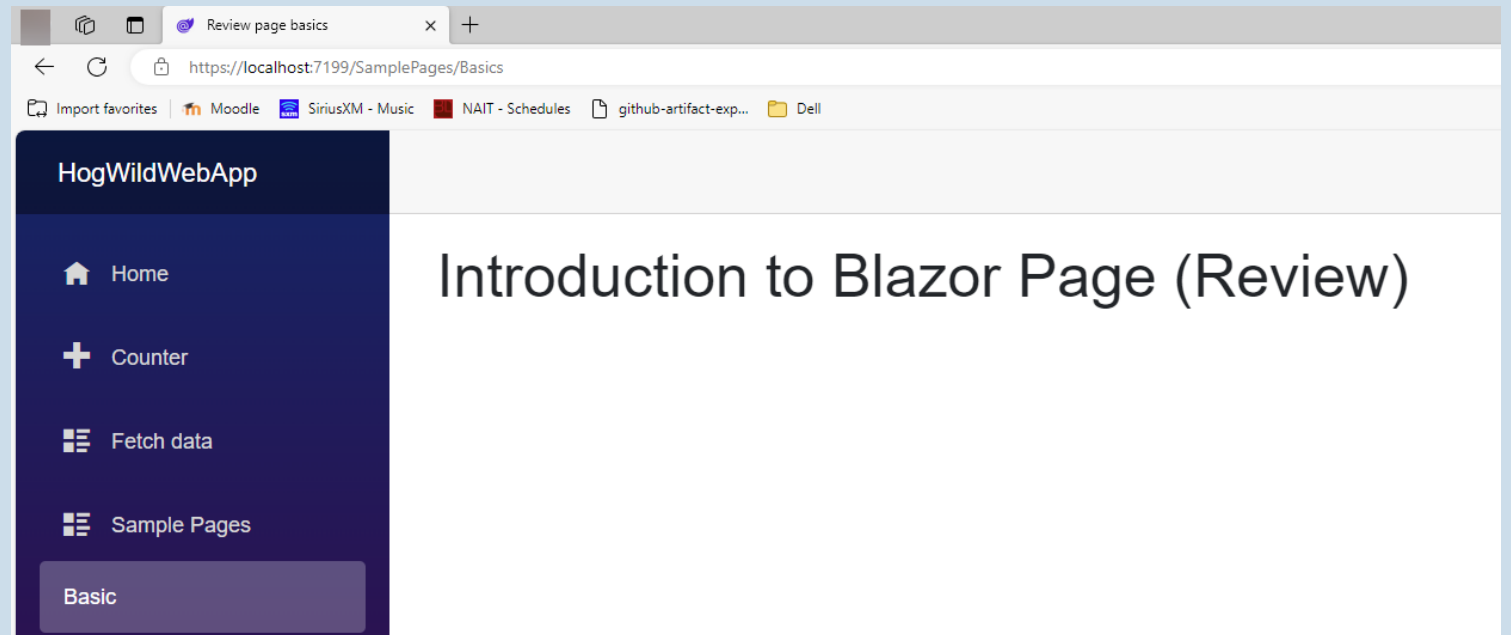
Web Application Layout

Components\Layout: This folder contains components and resources that can be reused across multiple pages or layouts in your Blazor Server application.

- **MainLayout.razor:** A shared layout component that defines the overall structure of your application's pages. It typically includes the header, navigation menu, and footer. You can customize this layout to suit your application's design and navigation needs.
- **NavMenu.razor:** A shared component representing the navigation menu for your application. It's often used within the MainLayout. You can customize the links and structure of the navigation menu to match your application's navigation requirements.

Building our First Page

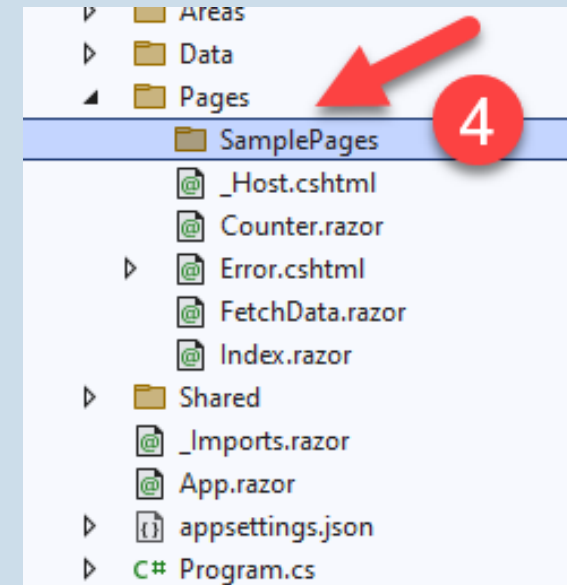
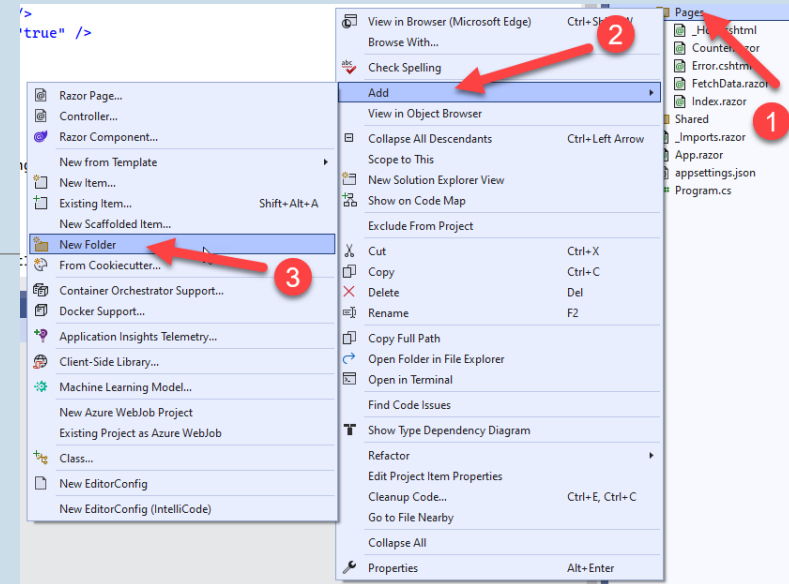
1. Create a folder for storing our pages: "SamplePages"
2. Add a new Razor Component: Basics.razor
3. Create code behind file: Basics.razor.cs
4. Create CSS file: Basics.razor.css
5. Add navigation for the "Basics" page.



Adding “SamplePages” Folder

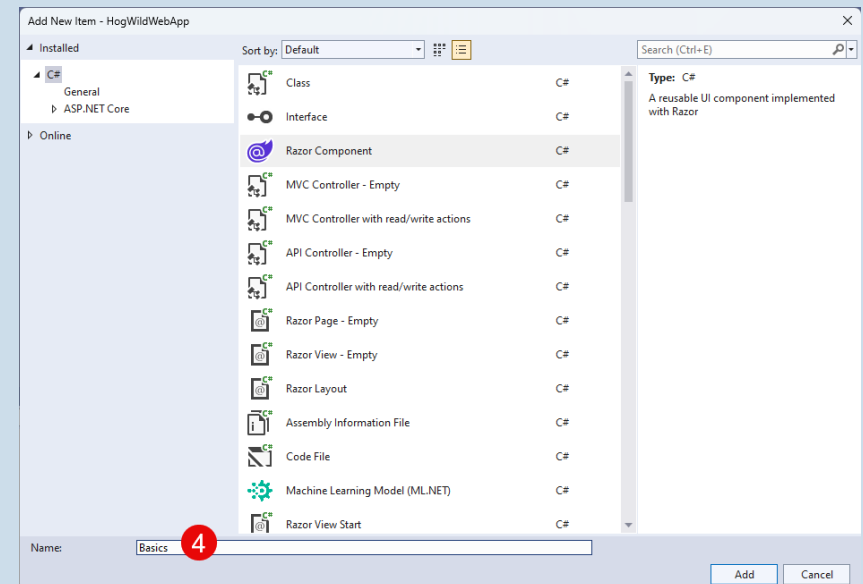
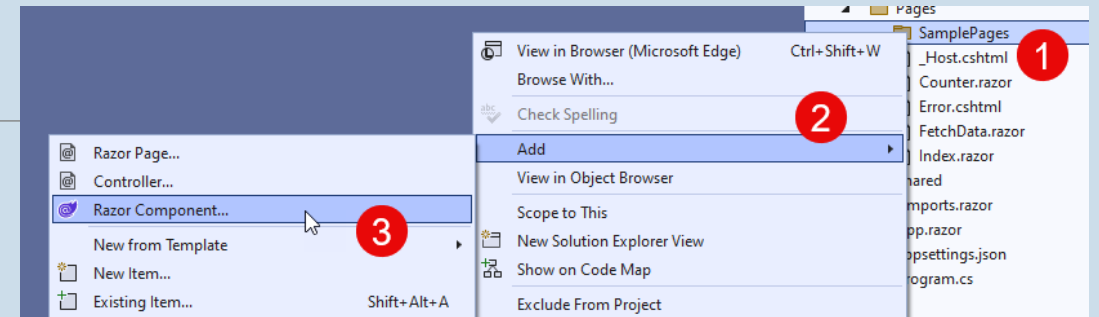
1. Right click on Pages
2. Select “Add”
3. Select New Folder
4. Type “SamplePages”

NOTE: The image is from .Net 7.0



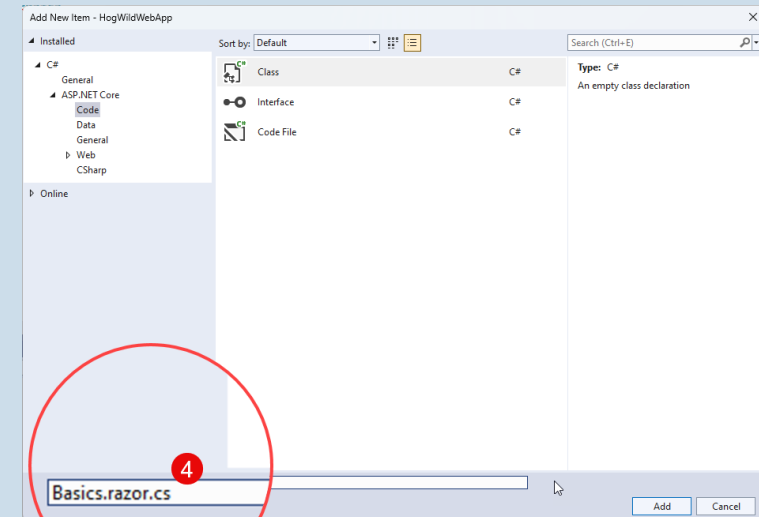
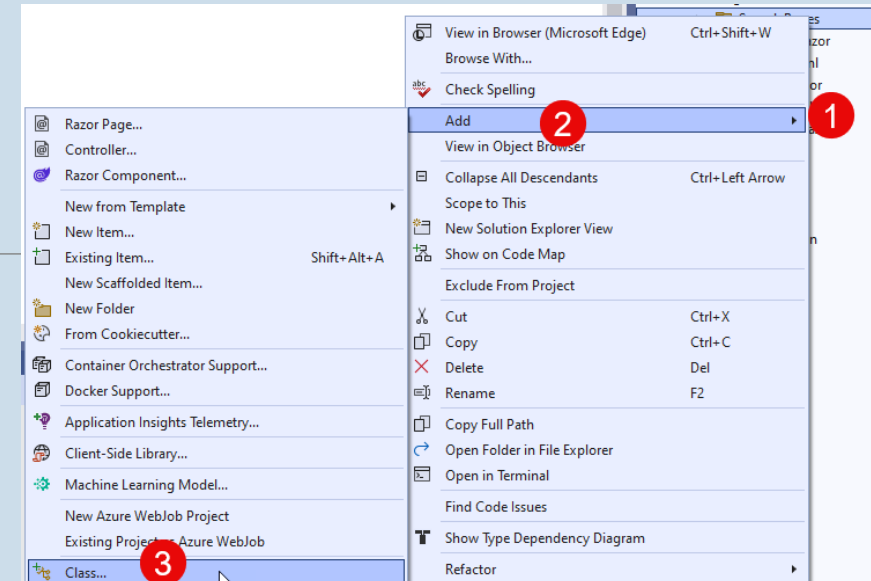
Adding “Basics” Razor Component

1. Right click on “SamplePages:
2. Select “Add”
3. Select “Razor Component...”
4. Type “Basics”



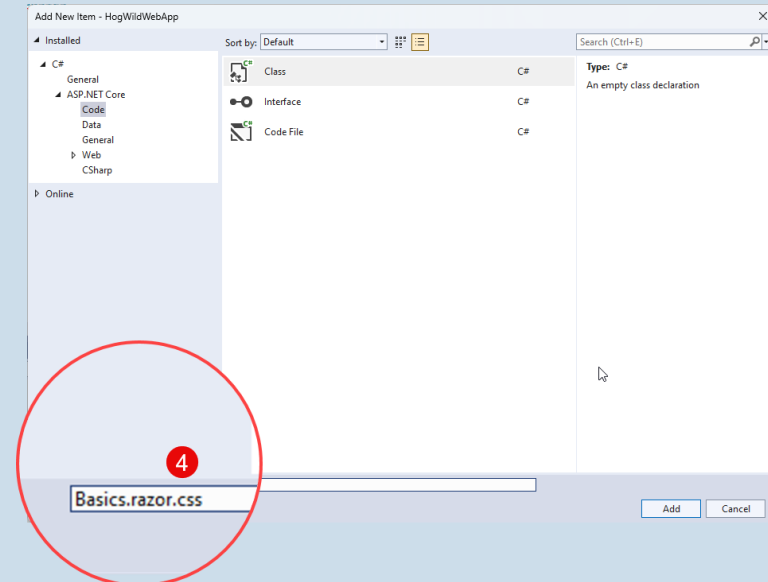
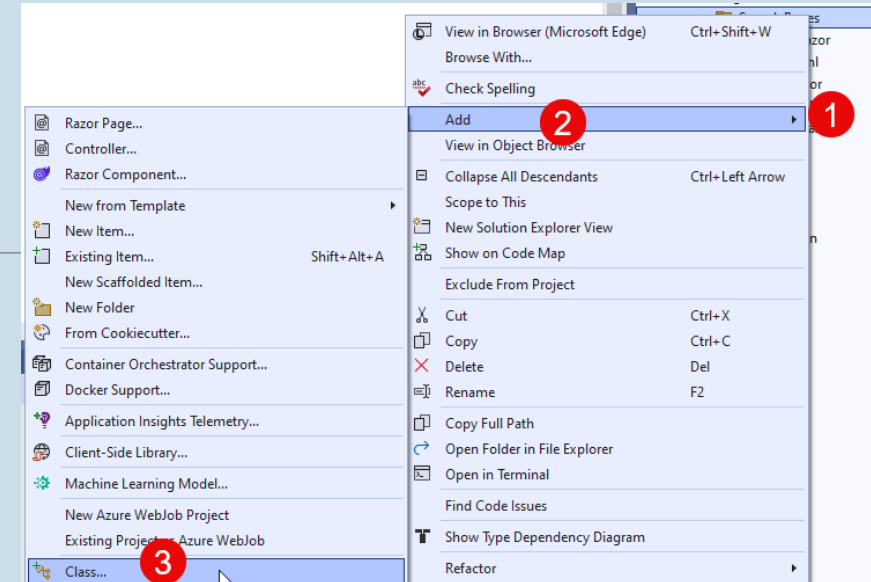
Adding “Basics” Code Behind File

1. Right click on “SamplePages:
2. Select “Add”
3. Select “Class...”
4. Type “Basics.razor.cs”



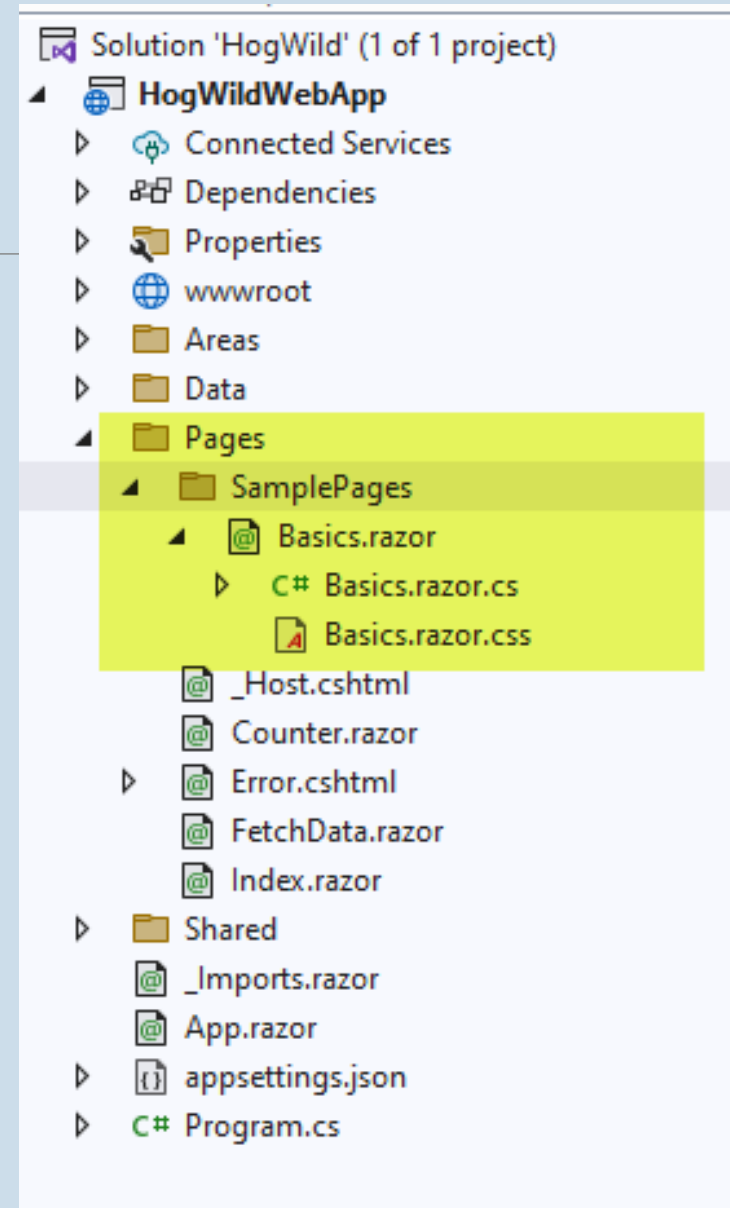
Adding “Basics” CSS File

1. Right click on “SamplePages:
2. Select “Add”
3. Select “Class...”
4. Type “Basics.razor.css”



Final Layout “Basics” Razor Component Files

NOTE: The image is from .Net
7.0

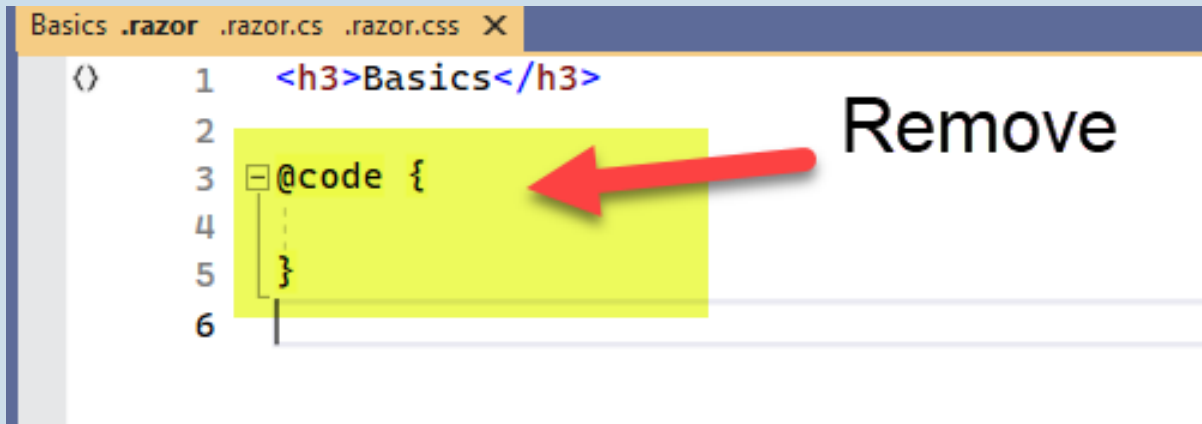


Initial Cleanup

Basics.razor

In the Blazor framework, the `@code` block defines the C# code associated with a Blazor component. This code block is typically placed within a `.razor` file and is used to implement the logic, properties, methods, and event handlers specific to that component.

We will be removing this from our “Basics.razor” file.



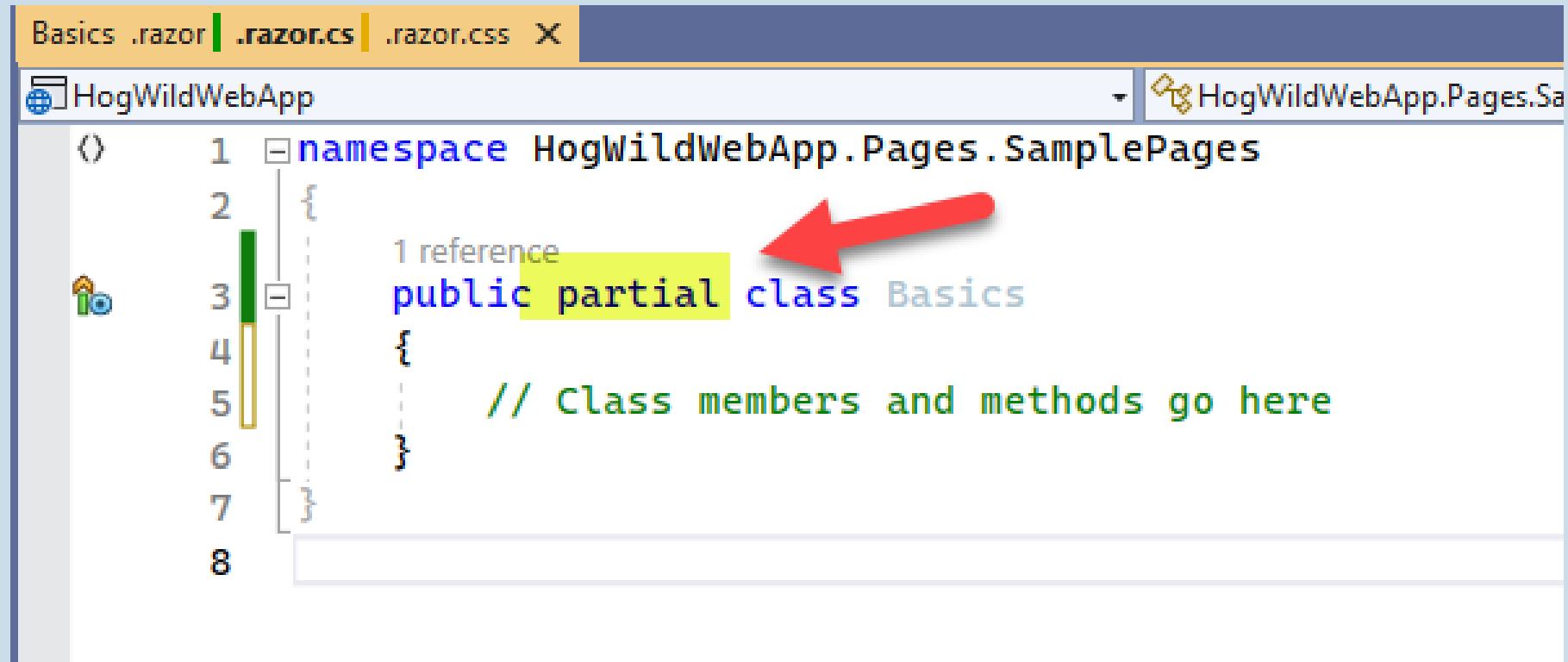
```
Basics .razor .razor.cs .razor.css X
{
  1 <h3>Basics</h3>
  2
  3 @code {
  4
  5 }
  6
```

Remove

Adding “Partial” to Basics.razor.cs

In C#, classes can be declared as partial classes, which means that their definition can be split across multiple files. This is often used in large projects to organize and maintain code. However, when you declare a class as partial, you should make sure that all parts of the class have the **partial** modifier.

Basics.razor.cs



```
Basics .razor | Basics.razor.cs Basics.razor.css X
HogWildWebApp
1 namespace HogWildWebApp.Pages.SamplePages
2 {
3     public partial class Basics
4     {
5         // Class members and methods go here
6     }
7 }
8
```


Adding Navigation

NavMenu.razor

Adding a collapse area for storing our pages.

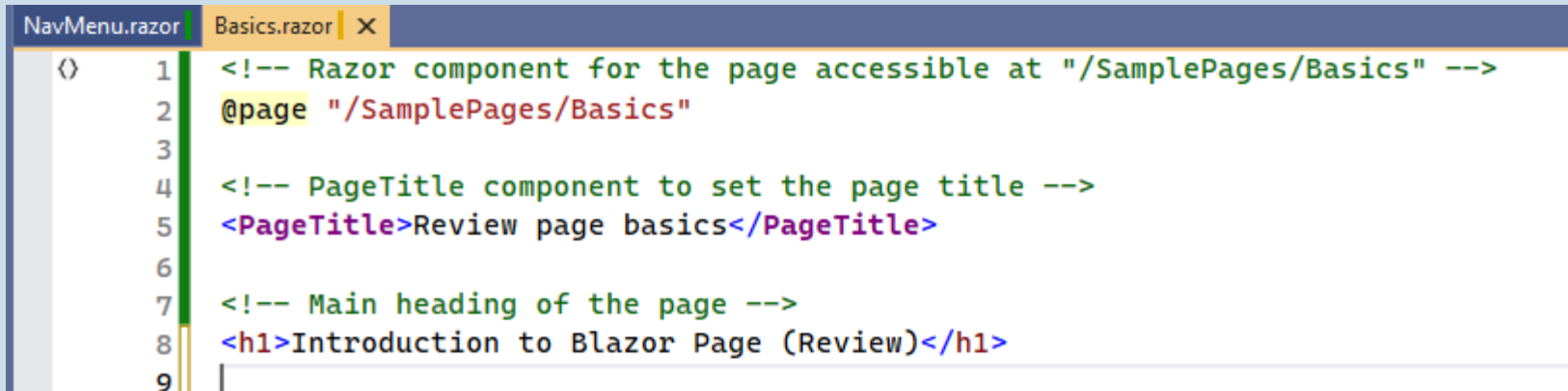
Adding navigation to “Basics” page.

Adding variable for storing the current state of the “Sample Page” toggle.

```
NavMenu.razor X
6      </button>
7      </div>
8  </div>
9
10 <div class="@NavMenuCssClass nav-scrollable" @onclick="ToggleNavMenu">
11   <nav class="flex-column">
12     <div class="nav-item px-3">...
17     <div class="nav-item px-3">...
22     <div class="nav-item px-3">...
27     <div class="nav-item px-3">
28       <!-- NavLink to expand or collapse the Sample Pages section -->
29       <NavLink class="nav-link" @onclick="()=>expandSamplePages = !expandSamplePages">
30         <span class="oi oi-list-rich" aria-hidden="true"></span> Sample Pages
31       </NavLink>
32
33       <!-- Check if the Sample Pages section is expanded -->
34       @if (expandSamplePages)
35       {
36         <!-- Nested NavLink within the Sample Pages section -->
37         <NavLink class="nav-link" href="/SamplePages/Basics">
38           <span>Basic</span>
39         </NavLink>
40       }
41     </div>
42   </nav>
43 </div>
44
45 @code {
46   // Holds the current state of the Sample Pages section toggle
47   private bool expandSamplePages;
48
49   private bool collapseNavMenu = true;
```

Refactoring Basics.razor

Basics.razor



```
1 <!-- Razor component for the page accessible at "/SamplePages/Basics" -->
2 @page "/SamplePages/Basics"
3
4 <!-- PageTitle component to set the page title -->
5 <PageTitle>Review page basics</PageTitle>
6
7 <!-- Main heading of the page -->
8 <h1>Introduction to Blazor Page (Review)</h1>
9
```

@Page Directive

In the provided Razor component, `@page "/SamplePages/Basics"` is a directive used in Blazor to define the route or URL at which the component can be accessed. Let's break down its significance:

1. `@page`: This is a special directive in Blazor used to define the routing information for the component. It tells the Blazor framework that this component should be associated with a specific URL or route.

2. `"/SamplePages/Basics"`: This is the route or URL associated with the Razor component. It indicates that when a user navigates to the `"/SamplePages/Basics"` URL in the application, this particular Razor component will be rendered and displayed in the browser.

So, in simple terms, `@page "/SamplePages/Basics"` is specifying that this Razor component is accessible at the `"/SamplePages/Basics"` URL within the Blazor application. When a user visits that URL, the content defined within this component, including the page title and the main heading, will be displayed in their browser.

@ Symbol

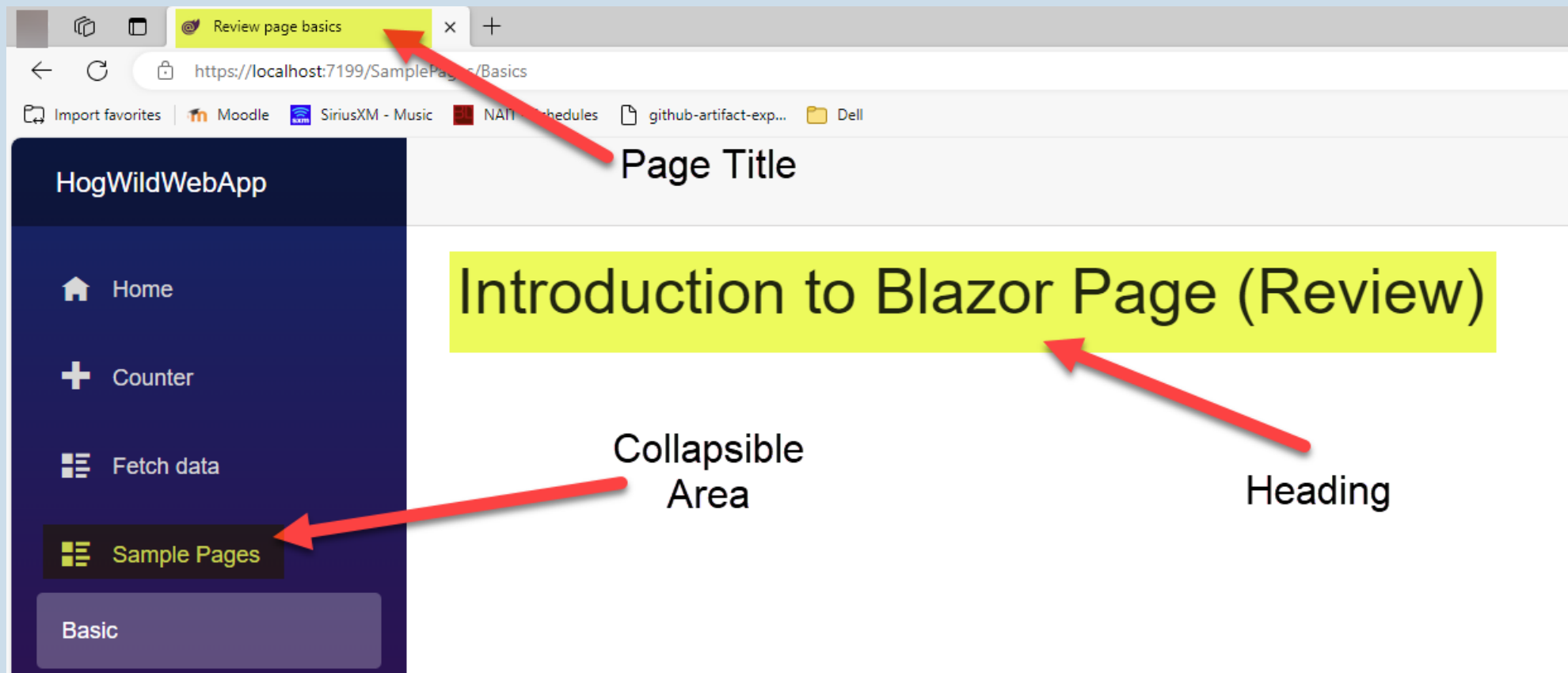
Blazor components combine C# and HTML code in a single file.

The '@' symbol is used to switch between the HTML and C# environments within the component.

When using '@', you're in the C# environment, allowing for the execution of C# code.

Without '@', you're in the HTML environment, where you write standard HTML markup.

Blazor Page Output





Statement Regarding Slide Accuracy and Potential Revisions

Please note that the content of these PowerPoint slides is accurate to the best of my knowledge at the time of presentation. However, as new research and developments emerge, or to enhance the learning experience, these slides may be subject to updates or revisions in the future. I encourage you to stay engaged with the course materials and any announcements for the most current information