

ASP.NET Core Blazor Applications

Razor components, interactivity, data access and security

FA-0638 · 5 days

Software Development

Develop and extend one Blazor to-do application while exploring Razor components, render modes, state, APIs and access control.

Introduction

This five-day course is for C#/.NET developers with web-development experience. It reviews relevant Razor foundations and develops one small to-do application to explore components, state, navigation, server and WebAssembly interactivity, data access and authentication.

Selected features are implemented in a prepared compatible environment; alternative render modes and tools use short comparisons or demonstrations. The source's explicit five-day title is used to resolve its conflicting four-day introductory sentence. The course teaches practical application development, not guaranteed mastery, production security or SEO gains.

Learning outcomes

The course teaches participants to:

- Distinguish Razor Pages from Razor components and review their relevant syntax.
- Explain static SSR, Interactive Server and Interactive WebAssembly in a Blazor Web App.
- Build and refactor a small to-do application using reusable components.
- Handle component parameters, lifecycle, navigation and selected state-management needs.
- Connect server-side data services and consume a prepared API from a client-side example.
- Review authentication and apply server-side authorisation to protected operations.
- Evaluate prerendering benefits, double-render/lifecycle considerations and limitations.

Prerequisites

- Solid understanding of .NET
- Skilled in C# programming.
- Skilled with web development concepts (HTML, CSS, JavaScript).
- Solid knowledge of Razor syntax and components.

A compatible supported .NET SDK, IDE and supplied project dependencies; Visual Studio or equivalent tooling appropriate to the selected SDK.

Permission to run the supplied local/training application and disposable database; production administrative access is not required.

Training outline

1. Day 1 – Razor Review, Blazor and Server Rendering

Razor Foundations Refresher

- Razor Syntax and Basics: Introduction to Razor syntax, how to write and use Razor markup within HTML.
- Review Razor Pages and distinguish .cshtml/page models from Blazor .razor components.
- Page Models: Understanding the role of Page Models and how to use them to handle requests.
- Handling Requests and Responses: Working with GET and POST requests, form submission, and handling responses.
- Razor Pages form binding versus Blazor component binding: distinguish the models.
- Using Tag Helpers: Introduction to built-in Tag Helpers and how to create custom Tag Helpers.
- Partial Views and Layouts: Creating and using partial views and layouts to structure your application.
- Recognise Razor Pages/MVC data-passing mechanisms and distinguish them from component parameters and services.
- Review selected JavaScript integration; jQuery is optional context, not a Blazor requirement.
- Validation and Forms: Implementing form validation using Data Annotations and the Validation Tag Helper.
- Routing: Understanding routing in Razor Pages and configuring custom routes.

Blazor Architecture and Setup

- Where Blazor fits in ASP.NET Core web development.
- Blazor Web App render modes and hosting concepts.

- Setting Up Your Development Environment
- Crafting Your First Blazor Project

Static Server-Side Rendering

- Static SSR: HTML rendering without component interactivity.
- Configure the prepared Blazor Web App for static server rendering.
- Server-Side Component Rendering
- SSR: Pros and Cons

2. Day 2 — Server Interactivity and Components

Interactive Server Rendering

- Events in Interactive Server rendering.
- Forms and validation in the selected render mode.
- Interactive Server circuit/SignalR communication and connectivity considerations.
- Data Binding on the Server

To-Do Project

- Project Kickoff and Requirements
- Setting Up Your To-Do List App
- Creating and Managing To-Do Items
- Implementing Basic CRUD Operations

Reusable, Non-Routable Components

- Responsibilities of reusable components that are not themselves routed pages.
- Crafting and Utilizing Non-Routable Components
- Passing Parameters and Managing State
- Component Communication Techniques

Project Refactoring

- Refactoring Your To-Do List App
- Creating Reusable Components
- Enhancing Component Interactions

- Review usability and feedback in the training application.

3. Day 3 – Lifecycle, Navigation and State

Component Lifecycle

- Unraveling Component Lifecycle Methods
- Initialization and Parameter Setting
- Handling State Changes and Events
- Cleanup and Disposal Best Practices

Routing and Navigation

- Configure and inspect application routing.
- Creating and Navigating Pages
- Route Parameters and Constraints
- Redirection and navigation checks; client-visible route guards are not an access-control boundary.

State Management

- The Essentials of State Management in Blazor
- Managing State with Services
- Implementing State Containers
- State across navigation, refresh and render modes; distinguish in-memory state from durable storage.

4. Day 4 – WebAssembly and Data Access

WebAssembly Interactivity

- Exploring Blazor WebAssembly
- Add an Interactive WebAssembly client example or inspect a prepared standalone alternative; distinguish their render-mode behaviour.
- Compare static SSR and interactive server/client rendering, including an overview of Auto where applicable.
- Implementing Client-Side Interactivity

Server-Side Data Services

- Data Access Strategies for Blazor SSR
- Connecting to Databases
- Using Entity Framework Core
- Implementing Data Services

Client-Side API Access

- Fetching Data in Blazor WebAssembly
- Consuming REST APIs
- Handling HTTP Requests and Responses
- Client-side storage limits; do not store secrets or treat browser data as trusted.

5. Day 5 — Access Control, Prerendering and Review

Authentication and Authorisation

- Implementing Authentication in Blazor
- Utilizing ASP.NET Core Identity
- Role-Based and Policy-Based Authorization
- Protect server operations/data and use UI/component checks for presentation; client-side checks can be tampered with.

Prerendering

- How prerendering delivers initial HTML before interactivity.
- Inspect prerendering configuration for the selected interactive components.
- Evaluate initial-content and potential SEO benefits against application behaviour; no ranking or universal performance promise.
- Handle double rendering, lifecycle differences and state transfer in selected examples.

Application Lab and Review

- Engaging Q&A Sessions
- Hands-On Exercises and Labs
- Practical Implementation Scenarios
- Participant Feedback and Discussion

