Faysal Aziz

Blazor Development Bootcamp

Building Modern Interactive Web Applications with C# and .NET - 3 days

FA-0118 · 3 days

Software Development

Introduction

Blazor has become one of the most important technologies in the modern Microsoft development ecosystem because it allows developers to build rich, interactive web applications using C# instead of relying heavily on JavaScript. As organizations continue consolidating technology stacks and seeking faster development cycles, Blazor offers a compelling approach for full-stack development with a unified programming model across frontend and backend systems.

With the evolution of ASP.NET Core and .NET 9, Blazor now supports modern rendering models including Server, WebAssembly, Hybrid, and full-stack interactive web applications with improved performance, streaming rendering, enhanced authentication integration, and better scalability for enterprise-grade deployments.

This beginner-focused 3-day course is designed to help participants build a solid practical foundation in Blazor development while understanding how modern web applications are architected using the latest .NET ecosystem practices. The course emphasizes practical implementation, clean coding practices, reusable component design, and real-world development workflows rather than purely academic concepts. The instructor brings over 30 years of industry experience and incorporates practical enterprise-oriented development approaches, tools, and implementation strategies used in modern software projects.

Learning outcomes

By the end of this course, participants will be able to:

- Understand the role of Blazor within the ASP.NET Core ecosystem
- Differentiate between Blazor Server, Blazor WebAssembly, and Hybrid hosting models

- Install and configure a modern Blazor development environment
- Build interactive user interfaces using Razor components
- Work with component lifecycle methods and event handling
- Implement data binding and form handling in Blazor applications
- Create reusable and maintainable UI components
- Perform routing and navigation within Blazor applications
- Consume REST APIs and integrate backend services
- Work with dependency injection and application services
- Manage application state and user interactions
- Implement validation and error handling mechanisms
- Understand authentication and authorization fundamentals in Blazor
- Apply modern .NET 9 Blazor development practices and rendering models
- Build and structure a complete beginner-level Blazor application
- Understand deployment fundamentals and production considerations

Prerequisites

- Deep understanding of programming concepts
- Familiarity with C# syntax and coding
- Basic understanding of HTML and CSS
- Exposure to object-oriented programming concepts
- Working experience of web applications and client-server architecture
- Familiarity with Visual Studio or Visual Studio Code is beneficial but not mandatory
- No prior Blazor experience is required

Training outline

1. Introduction to Modern Web Development with Blazor

Evolution of Microsoft web development technologies

1. ASP.NET Web Forms overview
2. ASP.NET MVC evolution
3. ASP.NET Core modernization
4. Emergence of Blazor in modern development

Understanding Blazor Architecture

1. What is Blazor
2. Component-based architecture
3. Single-page application concepts
4. Role of Razor components
5. Interactive web application concepts

Blazor Hosting Models

1. Blazor Server fundamentals
2. Blazor WebAssembly fundamentals
3. Blazor Hybrid overview
4. Interactive rendering models
5. Static SSR concepts
6. Interactive Server rendering
7. Interactive WebAssembly rendering
8. Auto rendering mode concepts

Current Blazor and .NET Ecosystem

1. Overview of .NET 9 enhancements
2. Modern ASP.NET Core architecture
3. Performance improvements in modern Blazor
4. Real-world enterprise use cases
5. Industry adoption trends
6. Cross-platform development capabilities

2. Development Environment Preparation

Installing Development Tools

1. Visual Studio installation
2. Visual Studio Code setup
3. .NET SDK installation
4. Verifying SDK versions
5. Workload configuration

Configuring the Development Environment

1. Creating development projects
2. Solution structure overview
3. Project templates
4. Managing NuGet packages
5. Running local development servers

Understanding Blazor Project Structure

1. Program.cs
2. App.razor
3. Layouts
4. Pages
5. Shared components
6. Static assets
7. Configuration files

Developer Productivity Features

1. Hot reload functionality
2. Debugging Blazor applications
3. Logging fundamentals
4. Browser developer tools integration
5. Error diagnostics

3. Blazor Fundamentals and Razor Syntax

Razor Component Basics

1. Creating components
2. Component naming conventions
3. Razor syntax fundamentals
4. Mixing HTML and C#
5. Component rendering process

Working with Markup and Code

1. Inline expressions
2. Code blocks
3. Conditional rendering
4. Loop rendering
5. Dynamic content generation

Component Parameters

1. Parameter fundamentals
2. Passing data between components
3. Required parameters
4. Cascading parameters
5. Reusable component design

Event Handling in Blazor

1. Button click events
2. Form events
3. Keyboard events
4. Mouse events
5. Event callback patterns

Data Binding Fundamentals

1. One-way binding
2. Two-way binding
3. Binding expressions
4. Binding form controls
5. Change tracking concepts

4. Building User Interfaces with Blazor

Page Creation and Routing

1. Creating pages
2. Route templates
3. Navigation manager
4. URL parameters
5. Navigation workflows

Layouts and Shared UI

1. Main layouts
2. Nested layouts
3. Shared components
4. Navigation menus
5. Reusable UI structures

Styling Blazor Applications

1. CSS integration
2. Component-scoped CSS
3. CSS isolation
4. Responsive design concepts
5. Bootstrap integration

Forms and User Input

1. EditForm component
2. Input components
3. Form submission handling

4. Validation message handling
5. User interaction workflows

Validation Techniques

1. Data annotations
2. Validation rules
3. Client-side validation
4. Error message handling
5. User experience considerations

5. Component Lifecycle and Application Behavior

Component Lifecycle Fundamentals

1. Initialization lifecycle
2. Rendering lifecycle
3. Parameter updates
4. State changes
5. Disposal workflows

Managing Component State

1. Local component state
2. Shared state concepts
3. State containers
4. Data persistence fundamentals
5. User session concepts

Dependency Injection in Blazor

1. Service registration
2. Service lifetimes
3. Injecting services into components
4. Shared services
5. Application architecture considerations

Working with Services

1. Business logic services
2. Utility services
3. Configuration services
4. Data access services
5. Separation of concerns

6. Data Access and API Integration

Understanding APIs in Modern Applications

1. REST API fundamentals
2. JSON data structures
3. Client-server communication
4. API consumption workflows

HTTP Communication in Blazor

1. HttpClient fundamentals
2. Sending GET requests
3. Sending POST requests
4. Handling API responses
5. Error handling strategies

Working with Data Models

1. Model creation
2. Serialization concepts
3. Deserialization workflows
4. Data transfer objects
5. View model concepts

Displaying Dynamic Data

1. Rendering collections
2. Loading states
3. Empty states
4. Conditional UI rendering
5. Data refresh workflows

7. Authentication and Security Fundamentals

Introduction to Web Security

1. Authentication concepts
2. Authorization concepts
3. Identity fundamentals
4. Secure application practices

Blazor Authentication Fundamentals

1. Authentication state
2. Authentication providers

3. Protected pages
4. Role-based authorization
5. Secure routing

Modern Authentication Enhancements

1. Authentication state serialization
2. Secure rendering workflows
3. Protected interactive components
4. User session management

8. Advanced Beginner Blazor Concepts

JavaScript Interoperability

1. JS interop fundamentals
2. Calling JavaScript from Blazor
3. Calling .NET from JavaScript
4. Browser API integration
5. Interoperability use cases

Reusable Component Design

1. Component libraries
2. Shared UI architecture
3. Parameter-driven components
4. Template components
5. Maintainable UI design

Error Handling and Diagnostics

1. Exception handling
2. Error boundaries
3. Logging strategies
4. Troubleshooting rendering issues
5. Debugging techniques

Performance and Optimization Basics

1. Rendering optimization
2. Component efficiency
3. Lazy loading concepts
4. WebAssembly performance considerations
5. Scalability awareness

9. Building a Complete Beginner Blazor Application

Application Planning

1. Defining application requirements
2. UI structure planning
3. Component organization
4. Navigation planning

Implementing Application Features

1. Creating reusable layouts
2. Building forms
3. Consuming APIs
4. Managing state
5. User interaction handling

Organizing Application Architecture

1. Folder structures
2. Service organization
3. Separation of concerns
4. Maintainability practices

Testing and Validation

1. Manual testing workflows
2. Functional verification
3. Debugging application issues
4. Validation checks

10. Deployment and Operational Fundamentals

Preparing Applications for Deployment

1. Build configurations
2. Publish workflows
3. Environment configuration
4. Production readiness concepts

Hosting Options

1. IIS hosting overview
2. Azure hosting concepts
3. Cloud deployment fundamentals
4. Static hosting considerations

Monitoring and Maintenance

1. Logging fundamentals
2. Error monitoring
3. Application updates
4. Version management

Future Learning Paths

1. Advanced Blazor development
2. .NET Aspire overview
3. MAUI and Hybrid development
4. Enterprise application architecture
5. Full-stack .NET development pathways

This training outline is intended as a general guideline for the proposed course delivery. Topics, sequence, depth of coverage, hands-on activities, and technical focus areas may be adjusted, expanded, condensed, or otherwise modified by the trainer based on participant skill levels, organizational requirements, technological updates, classroom pace, or other operational considerations without prior notice.

© 2026 Faysal Aziz · Kuala Lumpur