

**Comprehensive Blazor Server**

**Course Number:** ASPNC-116  
**Duration:** 4 days

**Overview**

[Blazor](https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor) Server is a .NET web technology for hosting applications that appear to be traditional single-page applications (SPAs), but actually run on the server, enabling traditional Microsoft authentication/authorization schemes. Users get the SPA experience, while developers avoid writing JavaScript and focus on traditional server-side C# implementation.

In this Comprehensive Blazor Server training course, attendees learn how to use C# programming to build UI apps that leverage the same component-based patterns popularized by libraries such as Angular and React. Students learn how to reduce dependence on JavaScript UI libraries and frameworks, using JavaScript only when needed and not as the primary basis of their client-side applications. In addition, attendees learn how to consume data sources and perform authentication and authorization.

**Prerequisites**

All students must have:

* C# programming experience
* HTML, CSS, and JavaScript development experience
* ASP.NET Core experience

**Materials**

All Blazor Server training students receive courseware covering the topics in the class.

**Software Needed on Each Student PC**

* Windows 10 or later with at least 8 GB RAM
* Visual Studio 2019 or later with the latest update
* .NET 5 or later SDK
* LocalDB or another version of SQL Server
* Additional lab files that Accelebrate provides

**Objectives**

* Understand the Blazor platform
* Build UIs with components
* Utilize data binding and event handling
* Compose components
* Use a 3rd party component library
* Build Blazor pages and configure routing
* Deploy a Blazor Server application to production
* Consume server data
* Handle authentication and authorization
* Unit test Blazor apps

**Outline**

* Introduction
  + What is new in .NET 5 (relative to .NET 3.1)
  + What is new in C# 9.0?
  + What is Blazor?
  + Blazor Hosting Models
  + What is Blazor Server vs. Blazor WebAssembly?
  + Server Model
    - Server Model vs. JavaScript
    - UI Server Execution and Web Sockets
    - Scalability and Deployment
  + WebAssembly Model
    - What is WebAssembly?
    - Browser Compatibility
    - WebAssembly Model vs. JavaScript
    - How does .NET Core / C# run in a web browser?
    - Scalability and Deployment
* Blazor Server Application
  + Project Template
  + Create a New Application
  + Hosting Blazor Server
  + Configuration
  + Dependency Injection
  + Environments
  + Logging
  + Handling Errors
  + Debugging
* Razor Components and Data Binding
  + What is a Component?
  + Creating a Data Model
  + Binding the Data Model to the HTML
  + Passing Arbitrary Attributes
  + Handling Events
  + Manually Trigger State Updates and Re-rendering
* Composing Razor Components
  + Decompose a Component into Smaller Components
  + One-Way Data Binding
  + Two-Way Data Binding
  + Pass Data from a Parent Component to a Child Component using Parameters
  + Pass Data from a Child Component to a Parent Component using Event Callbacks
  + Use Keys to Optimize Performance
  + Use Refs to Access DOM Elements
  + Razor Component Libraries
  + Razor Component Design Patterns
    - Parameters are Immutable
    - Lift State Up
    - Managing State in General
* Razor Component Forms
  + What is the purpose of Form?
  + Collecting Data using a Form, Input, Select and Textarea Elements
  + Explore Form Element Two-Data Binding
  + Build Forms with the Blazor Edit Form Razor Component
  + Explore the Concept of the Edit Context
  + Use the Specialized Edit Form Controls
    - Input Text
    - Input TextArea
    - Input Select
    - Input Number
    - Input Checkbox
    - Input Date
  + Applying Validation to the Form
  + Decorating the View Model with Validation Attributes
  + Code Custom Validation Attributes
* Razor Component Pages
  + What is the Page model?
  + Differences between Razor Pages and Razor Components
  + Using a Razor Component as a Page
  + Explore the Router Component
  + Configuring Page Routing
  + Route to Components from Multiple Assemblies
  + Using Route Parameters
  + Using the Query String
  + Applying Authorization to a Razor Component Page
  + Using Authorization within the Component Tree
* Blazorise
  + What is Blazorise?
  + Configuring Blazorise for Blazor Server
  + Configuring CSS for Blazorise
  + Blazorise Theming
  + Components
  + Forms
  + DataGrid
* Using Server Data
  + Connect to SQL Server with EF Core
  + Utilize EF Core in a Blazor Server application
* Security and Identity
  + Compare/Contrast Authentication between Blazor Server and ASP.NET Core
  + Compare/Contrast Authorization between Blazor Server and ASP.NET Core
  + Implement a Login Form in Blazor Server
  + Integration with ASP.NET Core’s HttpContext.User
  + Handle Authorization in Blazor Server
  + Threat mitigation
* Interacting with JavaScript
  + What is the JavaScript Interop?
  + When is JavaScript needed?
  + Synchronous vs. Asynchronous Calls
  + How to call a JavaScript function from a Component
  + How to call C# code from JavaScript
  + Calling Static Methods
  + Calling Instance Methods
  + Organizing JavaScript Code within a Blazor Server App
  + Explore JavaScript Ecosystem
    - Client-Side Libraries
    - NPM & Yarn
    - Webpack
    - Useful Libraries
* Unit Testing
  + What is Unit Testing?
  + Principles of Unit Testing
    - Defining a Unit
    - Setup/Teardown
    - Testing in Isolation
    - Determining What to Test
    - Code Coverage
    - Test Frameworks
    - Stubs, Mocks and Spies
  + xUnit
    - What is xUnit?
    - Testing Framework
    - Facts vs. Theory
    - Assertions
    - Integration with Visual Studio
  + Razor Components
    - What Should be Tested on a Razor Component?
    - What is bUnit?
    - Using bUnit with xUnit
    - Setup and define components under tests in C# or Razor syntax
    - Verify outcome using semantic HTML comparer
    - Interact with and inspect components
    - Trigger event handlers
    - Provide cascading values
    - Inject services
    - Mock IJsRuntime
    - Perform snapshot testing
* Blazor WebAssembly Demo
  + Create a Blazor WebAssembly Project
  + Compare/Contrast Blazor WebAssembly to Blazor Server
  + Explore the WebAssembly and ASP.NET Core Server Architecture
  + Demonstrate how to use a Component from the Blazor Server project with the Blazor WebAssembly project
* Conclusion