

**Web Apps with Blazor and ASP.NET Core Web API**

**Course Number:** ASPNC-128
**Duration:** 3 days

**Overview**

Embrace the future of web development with Blazor, the framework that blends the power of .NET with the simplicity of the web. This Web Apps with Blazor and ASP.NET Core Web API training course teaches attendees how to master crafting modern, single-page applications with razor-sharp precision.

**Prerequisites**

Students need HTML, CSS, JavaScript, and C# programming experience.

**Materials**

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of documentations and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

**Software Needed on Each Student PC**

Students will need a free, personal GitHub account to access the courseware and will also need to pre-install Visual Studio 2022. In addition, students will need permission to install NuGet Packages. If students are unable to configure a local environment, a cloud-based environment can be provided.

**Objectives**

* Understand Blazor's core concepts, history, and hosting models (Server, WebAssembly, Unified, Hybrid)
* Use WebAssembly and explore its compatibility and advantages over JavaScript and C#
* Build robust Razor components.
* Use form functionalities
* Understand Razor Component Pages
* Implement authentication, authorization, and auditing using best practices and new .NET 8 features
* Master new .NET 8 Render Modes
* Interact with REST APIs
* Understand JavaScript Interop, call JavaScript functions from C# (and vice versa), and organize JavaScript code
* Write unit tests for Blazor apps using xUnit and bUnit frameworks

**Outline**

* Introduction
	+ What is Blazor?
	+ History of Blazor Hosting Models
	+ Blazor Server
	+ Blazor WebAssembly
	+ Blazor Unified
	+ Blazor Hybrid
	+ What’s new for Blazor in .NET 8
* Running Blazor Code
	+ Single Page Applications
	+ What is WebAssembly?
	+ Browser Compatibility
	+ WebAssembly vs. JavaScript
	+ How does .NET Core / C# run in a web browser?
* Blazor Application
	+ Project Template
	+ Create a New Application
	+ Hosting Blazor with a ASP.NET Core MVC Server
	+ Configuration
	+ Dependency Injection
	+ Environments
	+ Logging
	+ Handling Errors
	+ Debugging WebAssembly in the Browser
* 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 Refs to Access DOM Elements
	+ Razor Component Libraries
	+ Razor Component Design Patterns
	+ Parameters are Immutable
	+ Lift State Up
	+ Managing State in General
	+ New .NET 8 Persistent Component State
	+ New .NET 8 Section Components
	+ New .NET 8 QuickGrid Component
* 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
	+ Specialized Edit Form Controls
	+ Applying Validation to the Form
	+ Decorating the View Model with Validation Attributes
	+ Code Custom Validation Attributes
	+ New .NET 8 Antiforgery Tokens
* 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
	+ New .NET 8 Enhanced Navigation for Static Rendered Pages
* Authenication, Authorization, and Auditing
	+ What is Authentication?
	+ What is Authorization?
	+ What is Auditing?
	+ Authentication Models for Blazor
	+ Component Attribute-based Authorization
	+ Authorization Components
	+ Authorization with C# Code
	+ Roles and Claims
	+ New .NET 8 Blazor Identity UI
* Rendering
	+ New .NET 8 Render Modes
	+ Static Server Rendering
	+ Interactive Server Rendering
	+ Interactive WebAssembly Rendering
	+ Interactive Auto Rendering
	+ AOT vs. JIT Compilation
	+ New .NET 8 Output Caching
	+ New .NET 8 Streaming Rendering
* Using Server Data
	+ REST APIs with ASP.NET Core Web API
	+ Web Sockets with SignalR
* ASP.NET Core MVC Web API
	+ What is ASP.NET Core MVC?
	+ What is a REST API?
	+ What is an API Controller?
	+ Injecting the Http Client
	+ Exploring the Http Client
	+ Calling a REST API from a Blazor Component using the HttpClient
	+ Build a REST API with ASP.NET Core MVC
	+ Using Dapper to access SQL Server Data
* SignalR
	+ Connect a Blazor App to SignalR
	+ Hubs
	+ Two-Way Data Transfer
	+ Connect to SignalR with C#
	+ Connect to SignalR with JavaScript
* Interacting with JavaScript
	+ What is the JavaScript Interop?
	+ When is JavaScript needed?
	+ Synchronous vs. Asynchronous Calls
	+ New .NET 8 JavaScript Initializers
	+ 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 App
* Unit Testing Blazor Apps
	+ What is Unit Testing?
	+ Testing Frameworks
	+ Integration with IDE
* 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 Framework
	+ What is xUnit?
	+ Testing Framework
	+ Facts vs. Theory
	+ Assertions
	+ Integration with Visual Studio
* Testing 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
* ASP.NET Core Web API
	+ What Should be Tested on a Web API?
	+ Testing Controllers
	+ Testing APIs
	+ Integration Testing of APIs
* Conclusion