Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Full-Stack Web Apps with Blazor and ASP.NET Core APIs

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- HTML, CSS, and JavaScript programming experience
- C# programming experience

Target Audience

Web developers with C# experience who build Blazor front ends backed by ASP.NET Core APIs, Entity Framework Core, and SignalR. Teams that only need the front end can take *Build Web Apps with Blazor*. Teams ready to work test-first on the same stack can continue with *Test-Driven Development with Blazor and ASP.NET Core APIs*.

Description

This course covers the full ASP.NET Core stack, from Blazor components to the APIs and database behind them. Students build Razor components, forms, routing, and passkey sign-in on the client side. On the server they build Minimal APIs, generate OpenAPI documents, and access data with Entity Framework Core. Controller-based Web APIs are covered for teams that maintain them. Students also use SignalR for real-time updates, test with bUnit and WebApplicationFactory, and run the whole app locally with Aspire.

Learning Outcomes

- Choose render modes and apply the C# features that shape full-stack .NET code
- Build composable UIs with Razor components, data binding, and validated forms
- Configure page routing and integrate client-side JavaScript
- Build REST APIs with Minimal APIs, typed results, endpoint filters, and validation

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Generate OpenAPI documents with Microsoft.AspNetCore.OpenApi
- Maintain controller-based Web APIs alongside Minimal APIs
- Model and query relational data with Entity Framework Core
- Consume server data from Blazor with HttpClient and SignalR
- Secure the app with passkeys, ASP.NET Core Identity, and authorization policies
- Test Razor components with bUnit and APIs with WebApplicationFactory
- Run the app locally with Aspire and deploy it to production

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install the .NET 10 SDK and the .NET 11 SDK, plus Visual Studio 2026 or Visual Studio Code with the C# Dev Kit, and a container runtime (Docker Desktop or Podman) on their computers. Also, students will need permission to install NuGet Packages. If students are unable to configure a local environment, a cloud-based environment can be provided.

Training Topics

Getting Started with Blazor

- What is Blazor?
- Comparison to Other Web Frameworks
- The Blazor Web App Model
- Server-Side & WebAssembly
- LTS and STS .NET Releases

Modern C# for Full-Stack .NET

- Records and Pattern Matching for DTOs
- Union Types for Results and Errors
- Closed Hierarchies
- Extension Properties and Indexers
- The `field` Keyword

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Null-Conditional Assignment
- Collection Expressions and Their Arguments

Running Blazor Code

- Single Page Applications
- What is WebAssembly?
- WebAssembly vs. JavaScript
- How does .NET run in a web browser?

Blazor Application

- Project Templates
- Create a New Application
- Hosting Blazor in an ASP.NET Core 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 Triggering Re-rendering

Rendering

- Render Modes
- Static Server Rendering
- Interactive Server Rendering
- Interactive WebAssembly Rendering
- Interactive Auto Rendering
- AOT vs. JIT Compilation
- Streaming Rendering
- Output Caching and the CacheView Component
- Render Modes and Consuming Data

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Render Modes and JavaScript

Composing Razor Components

- Splitting a Component into Smaller Components
- One-Way and Two-Way Data Binding
- Parent-to-Child Data with Parameters
- Child-to-Parent Data with Event Callbacks
- Refs to DOM Elements
- Razor Component Libraries
- Required Component Parameters
- Lift State Up
- Managing State in General
- Persistent Component State
- Circuit Reconnection, Pause, and Resume
- Section Components

Displaying Data with QuickGrid and Virtualize

- QuickGrid Columns, Sorting, and Paging
- URL-Based Paging and Sorting
- Conditional Row Styling with RowClass
- Row Click Handling
- QuickGrid over Entity Framework Core
- Virtualizing Variable-Height Items
- Scrolling to an Item

Razor Component Forms

- Form, Input, Select, and TextArea Elements
- The EditForm Component
- The Edit Context
- Specialized Input Components
- The DisplayName Component
- Validation Attributes on the View Model
- Custom Validation Attributes
- Validating Nested Objects and Collections
- Client-Side Validation for Static Forms
- Asynchronous Validation Rules
- Antiforgery Protection

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Razor Component Pages and Routing

- Using a Razor Component as a Page
- The Router Component
- Configuring Page Routing
- Routing to Components in Other Assemblies
- Route Parameters and the Query String
- Enhanced Navigation for Static Pages
- Not Found Pages
- Relative Navigation and NavLink Matching

Authentication, Authorization, and Auditing

- What is Authentication?
- What is Authorization?
- What is Auditing?
- Authentication Models for Blazor
- Passkey Sign-In with ASP.NET Core Identity
- Component Attribute-based Authorization
- Authorization Components
- Authorization with C# Code
- Roles and Claims
- Policies Shared by Components and APIs
- Securing APIs Called from Blazor
- Blazor Identity UI

Interacting with JavaScript

- What is JavaScript Interop?
- When is JavaScript Needed?
- Synchronous vs. Asynchronous Calls
- JavaScript Initializers
- Calling JavaScript from a Component
- Calling .NET from JavaScript
- Collocated JavaScript Modules

ASP.NET Core Minimal APIs

- What is a Minimal API?
- REST Conventions and Resource Design
- Route Handlers and Route Groups

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Parameter Binding
- TypedResults and Problem Details
- Endpoint Filters
- Handling Binding Failures in Filters
- Built-in Validation with `AddValidation`
- Asynchronous Validation Rules
- Union Types in Request and Response Bodies
- The `[ShortCircuit]` Attribute
- CORS, Rate Limiting, and Output Caching
- API Versioning Strategies

OpenAPI

- Generating Documents with Microsoft.AspNetCore.OpenApi
- Choosing the OpenAPI Specification Version
- Document, Operation, and Schema Transformers
- Describing Problem Details and File Results
- Describing Server-Sent Events
- Build-Time Document Generation
- Interactive API Documentation UIs
- Generating Typed Clients

Controller-Based Web APIs

- What is a Controller API?
- Differences from Minimal APIs
- Controllers, Actions, and Attribute Routing
- Model Binding and ApiController Conventions
- Controller Filters
- Moving Endpoints to Minimal APIs

Data Access with Entity Framework Core

- DbContext, Entities, and Model Configuration
- Migrations and Schema Evolution
- Querying with LINQ
- Tracking, Projections, and Split Queries
- Relationships and Loading Related Data
- Saving Changes and Transactions
- Optimistic Concurrency

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- DbContext Lifetimes with `IDbContextFactory`
- Raw SQL and Dapper

Calling APIs from Blazor

- HttpClient with IHttpClientFactory
- Calling a REST API from a Component
- Sharing DTOs between Client and Server
- Loading States, Errors, and Cancellation
- Direct Services vs. HTTP Calls

SignalR

- Connect a Blazor App to SignalR
- Hubs
- Two-Way Data Transfer
- Connect to SignalR with C#
- Connect to SignalR with JavaScript
- Scaling Out SignalR

Testing Blazor Apps and APIs

- Principles of Unit Testing
- Test Doubles and Coverage
- xUnit v3 on Microsoft.Testing.Platform
- Facts, Theories, and Fixtures
- Testing Razor Components with bUnit
- API Integration Tests with WebApplicationFactory
- Testing EF Core against a Real Database

Aspire and Deployment

- The Aspire AppHost
- Orchestrating the App, API, and Database
- Service Defaults for OpenTelemetry and Health Checks
- Service Discovery
- Publishing and Containerizing the Application
- Deploying to Production Hosting

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved