

**Comprehensive ASP.NET Core 8 Development**

**Course Number:** ASPNC-124
**Duration:** 5 days

**Overview**

This in-depth ASP.NET Core 8 training course teaches developers how to build modern, high-performance web applications using Microsoft's .NET 8 framework. Attendees learn how to create dynamic web interfaces with MVC and Razor Pages and build interactive real-time web apps with Blazor.

**Prerequisites**

All students must have:

* Experience with the C# programming language and object-oriented programming concepts
* Some knowledge of HTML, CSS, and JavaScript concepts

**Materials**

All ASP.NET Core training students receive comprehensive courseware.

**Software Needed on Each Student PC**

* .NET 8.0 SDK
* Visual Studio 2022 (17.8 or later), VS Code, or Rider
* Lab file bundle provided with the course

**Objectives**

* Understand the goals and benefits of ASP.NET Core 8.0
* Learn to make good decisions about application architecture and data access technology
* Use ASP.NET’s routing system to achieve a REST-style architecture
* Learn how to build a compelling and maintainable HTML user interface using the Razor view engine and client-side JavaScript
* Gain experience building a service that makes data available via a modern web API
* Understand the advantages of the new Minimal API Framework
* Learn best practices for employing unit testing, logging, and error handling
* Understand different authentication choices for securing a web API
* Get an introduction to Blazor, Razor Pages, and gRPC
* Understand the different cross-platform deployment options available including via Docker containers

**Outline**

* Introduction
	+ Evolution of .NET and .NET Core
	+ .NET SDKs and Runtimes
	+ IDE Choices
* .NET 8.0 SDK
	+ Installation
	+ Version Management
	+ Command-Line Interface (CLI)
* Modern C# and What's New in C# 12.0
	+ Multi-paradigm C#
	+ Features from Functional Programming
	+ Evolution of Nullability in .NET
	+ Immutability
	+ Designing for Concurrency
	+ Deferred Execution
* ASP.NET Core Application Architecture
	+ NuGet Packages
	+ Application Startup
	+ Hosting Environments
	+ Middleware and the Request Pipeline
	+ Services and Dependency Injection
* Application Configuration
	+ Configuration Providers and Sources
	+ Configuration API
	+ Options Pattern
	+ HTTPS and HTTP/2
* Request Routing
	+ RESTful Services
	+ Endpoint Routing
	+ Route Templates
	+ Route Constraints
	+ Route Template Precedence
	+ Attribute-Based Routing
* Models
	+ Persistence Ignorance
	+ Dependency Inversion
	+ Asynchronous Data Access
	+ Object-Relational Mapping
	+ Entity Framework Core
	+ Dapper ORM
* Controllers
	+ Responsibilities
	+ Requirements and Conventions
	+ Dependencies
	+ Action Results
	+ ApiController Attribute
* Views
	+ Responsibilities
	+ Conventions
	+ Razor Syntax
	+ Layouts
	+ ViewData and ViewBag
	+ Strongly-Typed Views
	+ Partial Views
	+ HTML and URL Helpers
	+ Tag Helpers
	+ View Components
	+ Client-Side Dependencies
	+ Razor Pages
	+ View Models
* HTML Forms
	+ Tag Helpers
	+ Form Submissions
	+ Model Binding
* Input Validation
	+ Introduction
	+ Data Annotations
	+ Model Binding
	+ Input Tag Helpers
	+ Validation Tag Helpers
* Application State
	+ Client-Side vs. Server-Side
	+ HttpContext.Items
	+ Session State
	+ TempData
* Web APIs
	+ API Controllers
	+ Minimal APIs
	+ OpenAPI / Swagger
	+ Testing APIs
	+ Cross-Origin Resource Sharing (CORS)
* Error Handling
	+ Best Practices
	+ HTTP Error Status Codes
	+ Developer Exception Page
* Logging
	+ Configuration
	+ ILogger
	+ Serilog and Seq
* Testing
	+ Unit Testing
	+ xUnit
	+ Testing Controllers
	+ Integration Testing
* Security
	+ Authentication
	+ ASP.NET Identity
	+ Bearer Tokens
	+ Authorization
	+ Web API Authentication
	+ OAuth 2.0 and OpenID Connect
	+ Secrets Management
* Remote Procedure Calls (gRPC)
	+ Introduction
	+ Protobuf
	+ Server
	+ Client
	+ Limitations
* Blazor
	+ Razor Components
	+ Blazor Server vs.WebAssembly
	+ Render Modes in .NET 8
* Deployment
	+ dotnet publish
	+ Kestrel
	+ IIS
	+ Docker
* Conclusion