**Comprehensive ASP.NET Core 6 Development**

**Duration:** 5 days

**Overview**

This ASP.NET Core 6 training course provides comprehensive coverage of how to develop web applications with Microsoft’s ASP.NET Core 6 framework. Coverage of Web UIs includes the MVC pattern as well as Razor Pages. For Web APIs, attendees learn to build a traditional API, implement a microservice architecture, and use the new minimal API feature. An introduction to Blazor is included but it is not covered in-depth.

**Prerequisites**

* Previous experience developing web-based applications with C#
* Some familiarity with HTML, CSS, and JavaScript

**Software Needed on Each Student PC**

* Windows 10 or later with at least 32 GB RAM
* Visual Studio 2022
* .NET 6.0 SDK
* LocalDB or another version of SQL Server
* If you have purchased this class, please contact us for more detailed setup specifications

**Objectives**

* Understand the goals and benefits of ASP.NET Core 6.0
* 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
  + Visual Studio and Visual Studio Code
* .NET 6.0 SDK
  + Installation
  + Version Management
  + Command-Line Interface (CLI)
* What’s New in C#
  + Record Types
  + Init Only Setters
  + Nullable Reference Types
  + Global Using Directives
  + File-Scoped Namespace Declarations
  + Top-Level Statements
* 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
  + Content Negotiation
  + CRUD Operations
  + Microservice Architecture
  + 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
  + Authorization
  + Web API Authentication
  + JSON Web Tokens (JWT)
  + OAuth 2.0 and OpenID Connect
  + Secrets Management
* Remote Procedure Calls (gRPC)
  + Introduction
  + Protobuf
  + Server
  + Client
  + Limitations
* Blazor
  + Razor Components
  + Blazor Server
  + Blazor WebAssembly
* Deployment
  + dotnet publish
  + Kestrel
  + IIS
  + Docker
* Conclusion