Document Generated: 12/18/2025

Learning Style: Virtual Classroom

Technology:

Difficulty: Intermediate

Course Duration: 5 Days

Next Course Date: January 26, 2026

Developing ASP.NET Core 9.x MVC Web Applications (Intermediate C#) (TTCN20486)

About This Course:

ASP.NET Core MVC is a powerful framework that enables developers to build scalable, maintainable, and testable web applications. Using the Model-View-

Controller (MVC) pattern, it separates concerns between data, business logic, and user interface, making applications easier to manage and extend. For experienced C# developers, mastering ASP.NET Core MVC is essential for creating modern, dynamic web applications that integrate seamlessly with APIs, databases, and front-end technologies. Whether you're transitioning from desktop development, working with legacy .NET applications, or looking to build enterprise-grade web solutions, this course provides the structured guidance and hands-on experience needed to develop professional ASP.NET Core MVC applications efficiently.

Developing ASP.NET Core MVC Web Applications is an expert-led, hands-on course, where you will build real-world applications while applying best practices for designing and structuring MVC projects. You'll start by understanding the ASP.NET Core ecosystem and how it differs from legacy .NET frameworks. From there, you will configure middleware and services, use dependency injection to manage application components, and develop controllers, models, and views that follow clean architectural principles. With a strong emphasis on writing modern C# code, you'll leverage features like top-level statements, async/await, and record types to create efficient, maintainable applications.

Beyond core development, this course ensures you have the skills to handle database interactions using Entity Framework Core, implement authentication and authorization with ASP.NET Identity, and optimize security using best practices. You'll also learn how to improve performance with caching and state management, build and consume APIs, and troubleshoot issues with exception handling and logging. By the end of the course, you will have the confidence to build and deploy professional-grade ASP.NET Core MVC applications that meet real-world business needs.

Course Objectives:

- Design and structure ASP.NET Core MVC applications - Build scalable and maintainable web applications using the MVC pattern while following best practices for separation of concerns and clean architecture.

- Develop and manage controllers, models, and views - Implement business logic in models, handle user interactions with controllers, and create dynamic front-end views using Razor syntax, HTML helpers, and tag helpers.

- Implement middleware and dependency injection - Configure middleware to handle request processing and manage services efficiently using dependency injection to improve maintainability and testability.

- Secure applications with authentication and authorization - Implement ASP.NET Identity for user authentication, apply role-based and claims-based authorization, and protect applications from common security vulnerabilities.

- Integrate databases and APIs - Use Entity Framework Core to connect, query, and manage relational databases, and develop and consume Web

APIs to extend application functionality.

- Optimize performance and troubleshoot issues - Apply caching strategies, manage state, handle exceptions gracefully, and implement logging to monitor and debug ASP.NET Core MVC applications effectively.

Audience:

- This intermediate-level and beyond course is designed for experienced C# developers who are new to ASP.NET Core MVC and want to build full-featured web applications using modern development practices. Whether you're transitioning from desktop development, legacy .NET applications, or working in a team environment that requires scalable web solutions, this course provides the practical skills to get started.

Prerequsites:

Ideally you should have practical experience with the skills below in order to be successful with the hands-on portion of the course. If you are a little light in any of those areas, you are welcome to follow along with the labs.

- Strong C# programming experience, including concepts like LINQ, lambda expressions, and object-oriented principles.
- Familiarity with .NET Core and Visual Studio.
- Basic understanding of HTML, CSS, JavaScript, and working with relational databases.

Course Outline:

- Module 1: Overview of .NET Core & ASP.NET

  Gain a foundational understanding of ASP.NET Core and how it fits into modern web development using the MVC architecture.

- Explore Microsoft's web development technologies and how they compare.
- Understand the key differences between ASP.NET and ASP.NET Core.
- Learn how ASP.NET Core MVC enables scalable and maintainable applications.

- Recognize the advantages of cross-platform development with .NET Core.
- Identify the roles of models, views, and controllers in MVC applications.
- Set up an ASP.NET Core MVC project using Visual Studio.
- Understand how middleware and the request pipeline work in ASP.NET Core.
- Learn about the hosting options for ASP.NET Core applications.

- Module 2: Review of C# and New Features

  Reinforce your C# knowledge while exploring modern language features that simplify ASP.NET Core development.

- Review object-oriented programming principles in C#.
- Implement top-level statements to streamline application structure.
- Use global usings to simplify namespace management.
- Work with tuples and deconstruction for better data handling.
- Explore record types for immutable object modeling.
- Apply null operators and null reference types for safer code.
- Utilize async/await for efficient asynchronous programming.
- Observe these concepts in action through hands-on coding exercises.

- Module 3: Designing ASP.NET Core MVC Web Applications

  Plan and structure a well-architected ASP.NET Core MVC application that meets business and user requirements.

- Understand the importance of application planning and design.
- Define functional requirements and user interactions before development.
- Design models, controllers, and views for clear separation of concerns.
- Choose between different state management techniques.
- Plan data storage and database structure using Entity Framework Core.
- Identify common challenges in scaling and maintaining MVC applications.
- Optimize for performance, security, and maintainability from the start.
- Develop a roadmap for implementing and deploying your application.

- Module 4: Configuring Middleware and Services in ASP.NET Core

  Manage request processing, services, and dependencies in ASP.NET Core applications.

- Configure built-in middleware components to handle requests.
- Create custom middleware for handling application-specific needs.
- Implement dependency injection for better service management.
- Register and configure services within the application.
- Understand service lifetimes (transient, scoped, singleton) and when to use them.
- Inject dependencies into controllers, views, and other components.

- Use middleware for authentication, logging, and exception handling.
- Ensure modular, reusable service configurations for scalability.

- Module 5: Developing Controllers

  Create and manage controllers to handle user requests and manage routing.

- Understand the role of controllers in the MVC pattern.
- Write action methods to process incoming requests.
- Use route attributes and convention-based routing to map URLs.
- Implement model binding to pass data between views and controllers.
- Handle HTTP requests using GET, POST, PUT, and DELETE methods.
- Return JSON responses for API-style communication.
- Use dependency injection within controllers for cleaner architecture.
- Implement asynchronous actions to improve responsiveness.

- Module 6: Developing Views

  Build dynamic user interfaces using Razor syntax and reusable UI components.

- Create views using Razor to dynamically generate HTML.
- Use HTML helpers and tag helpers to simplify UI development.
- Implement partial views for reusable components.
- Work with strongly typed views and view models.
- Implement form submissions and handle user input.
- Pass data between controllers and views using ViewData, ViewBag, and TempData.
- Apply conditional rendering and loops within Razor templates.
- Debug and optimize views for performance and readability.

- Module 7: Using Layouts, CSS, and JavaScript in ASP.NET Core MVC

  Ensure a consistent look and feel while integrating front-end enhancements.

- Implement layouts to standardize headers, footers, and navigation.
- Use sections and view components to create modular UI elements.
- Apply CSS styles to enhance the appearance of applications.
- Integrate JavaScript and jQuery for interactive functionality.
- Load and manage external client-side libraries.
- Handle client-side validation to improve form usability.
- Implement responsive design principles for mobile support.
- Optimize front-end performance using bundling and minification.

- Module 8: Developing Models

Define the data structure and implement business logic in MVC applications.

- Create model classes that represent application data.
- Implement data validation with annotations.
- Use model binding to transfer data between views and controllers.
- Work with forms to collect user input.
- Apply client-side and server-side validation for accuracy.
- Utilize business logic within models to enforce rules.
- Implement computed properties and data transformations.
- Organize models for better maintainability and testability.

- Module 9: Using Entity Framework Core in ASP.NET Core

  Connect applications to databases and manage data efficiently.

- Configure Entity Framework Core for database access.
- Define and manage entity relationships using code-first modeling.
- Use LINQ to query and manipulate data.
- Implement repository patterns for structured data access.
- Apply migrations to modify database schema.
- Perform CRUD operations with database entities.
- Optimize database interactions for performance.
- Secure data access using best practices.

- Module 10: Testing and Troubleshooting

  Ensure application reliability with unit testing, logging, and exception handling.

- Write unit tests for controllers, models, and services.
- Debug MVC applications effectively using Visual Studio tools.
- Implement structured exception handling.
- Use built-in logging providers to capture application events.
- Configure logging levels for debugging and monitoring.
- Identify and fix common performance issues.
- Set up structured error pages and messages.
- Automate testing to streamline development workflows.

- Module 11: Managing Security

  Protect applications with authentication, authorization, and secure coding practices.

- Implement ASP.NET Identity for user authentication.
- Secure routes and resources using authorization policies.
- Use role-based and claims-based access control.
- Store and manage user credentials securely.

- Prevent SQL injection, CSRF, and XSS attacks.
- Apply HTTPS and secure cookies for better protection.
- Monitor security vulnerabilities and patch issues.
- Audit and log user activity for compliance.

- Module 12: Performance and Communication

  Improve application efficiency with caching, state management, and real-time communication.

- Implement caching strategies to reduce server load.
- Manage session and application state effectively.
- Optimize query performance for data-heavy applications.
- Implement SignalR for real-time client-server communication.
- Use WebSockets for persistent connections.
- Reduce memory and CPU overhead with best practices.
- Minimize unnecessary database queries and computations.
- Monitor and profile performance bottlenecks.

- Module 13: Implementing Web APIs

  Build RESTful APIs for seamless integration with external applications.

- Develop ASP.NET Core Web APIs using controllers.
- Handle HTTP requests and responses effectively.
- Implement versioning for maintainable APIs.
- Secure API endpoints with authentication and authorization.
- Return JSON and XML formatted data.
- Consume APIs from JavaScript and server-side applications.
- Document APIs using OpenAPI/Swagger.
- Optimize API performance with caching and compression.

- Module 14: Hosting and Deployment

  Prepare applications for production environments with deployment best practices.

- Configure applications for IIS and cloud hosting.
- Set up automated deployment pipelines.
- Manage application settings and configurations.
- Implement logging and monitoring for deployed applications.
- Optimize applications for cloud environments.
- Secure deployments with encryption and firewall rules.
- Scale applications using load balancing and containerization.
- Maintain and update live applications with minimal downtime.