KanaiyaKatarmal

.NET Developer Roadmap for 2025

2025
EDITION

→

newsletter.kanaiyakatarmal.com

NET Developer Roadmap

To stay ahead in the rapidly evolving world of .NET Core, following a structured learning path is crucial. Here's the .NET Core Developer Roadmap 2025 to guide you from the basics all the way to advanced topics.

1. C# Fundamentals

Before diving into .NET Core, mastering C# fundamentals is crucial.

- **Basic Syntax and Data Types:** Learn about variables, data types, operators, and control structures.
- **Object-Oriented Programming (OOP):**
  - **Classes and Objects:** Understand how to define classes and create instances.
  - **Inheritance:** Grasp the concept of class hierarchies and inheritance.
  - **Polymorphism:** Learn how the same method can behave differently based on the object type.
  - **Encapsulation:** Understand how to hide internal object details and provide access through methods.
  - **Abstraction:** Learn how to define abstract classes and interfaces. them.

- **Collections and LINQ:** Master collections such as lists, dictionaries, and LINQ queries to manipulate data.
- **Exception Handling:** Learn how to handle errors using try, catch, finally, and custom exceptions.
- **Asynchronous Programming:**
  - **async/await:** Grasp the fundamentals of asynchronous programming to improve app performance.
  - **Task Parallel Library:** Learn about parallel and concurrent programming using tasks.
  - **Threading:** Understand how threads work and how to manage them.

KanaiyaKatarmal

→

2. .NET Core Basics

Now that you're comfortable with C#, dive into the .NET Core framework.

- **Understanding .NET Core Architecture:** Learn about the core components of .NET Core and how they interact.
- **CLI Commands:** Get familiar with the .NET Core Command-Line Interface (CLI) to manage projects and dependencies.
- **Project Structure:** Understand the layout of a typical .NET Core project and the role of each file/folder.
- **Configuration:**
  - **appsettings.json:** Learn how to manage application settings.
  - **Environment Variables:** Set up environment-specific configurations.
  - **User Secrets:** Secure sensitive data for development purposes.
- **Dependency Injection:** Master the built-in DI container for decoupling classes.

- **Middleware:** Learn how middleware works in handling HTTP requests.
- **Logging:** Understand logging best practices and how to log events.
- **Error Handling:** Implement global error handling to enhance user experience.

KanaiyaKatarmal

newsletter.kanaiyakatarmal.com

3. Web Development

3.1 ASP.NET Core

ASP.NET Core is a robust web framework for building scalable, secure web applications.

- **MVC Pattern:** Understand the Model-View-Controller pattern and its application.
- **Razor Pages:** Use Razor Pages to build dynamic web pages.
- **Routing:** Learn how URL patterns map to controllers and actions.
- **Controllers and Views:** Master how to create and render views from controllers.
- **Model Binding:** Bind data from views to model objects.
- Validation: Implement validation logic for user inputs.
- **Filters:** Learn how to use filters for logging, authentication, and authorization.
- **Areas:** Organize your application into areas for better modularity.

3.2 Web APIs

Building APIs with .NET Core is a vital skill for modern web development.

- **RESTful Services:** Learn how to create scalable and stateless REST APIs.
- **HTTP Methods:** Understand GET, POST, PUT, DELETE, and PATCH methods.
- **Status Codes:** Use HTTP status codes correctly for API responses.
- **API Versioning:** Implement versioning to maintain backward compatibility.
- **Content Negotiation:** Handle different response formats (JSON, XML).
- **CORS:** Configure Cross-Origin Resource Sharing for security.
- **API Documentation (Swagger/OpenAPI):** Use Swagger for interactive API documentation.
- **Rate Limiting:** Prevent abuse by limiting the rate of requests.
- **API Security:** Secure your API using best practices, including authentication and authorization.

4. Database Technologies

4.1 Entity Framework Core

Entity Framework Core (EF Core) is the ORM tool for interacting with databases.

- **Code First Approach:** Define models in code and generate the database schema.
- **Database First Approach:** Reverse engineer models from an existing database.
- **Migrations:** Manage database changes over time.
- **CRUD Operations:** Learn how to create, read, update, and delete data.
- **Relationships:** Master 1-to-1, 1-to-many, and many-to-many relationships.
- **Lazy Loading vs Eager Loading:** Understand when to load related data lazily or eagerly.
- **Query Optimization:** Optimize queries for better performance.

4.2 Database Systems

In addition to EF Core, you should be familiar with various databases.

- **SQL Server:** The most widely used relational database in .NET.
- **PostgreSQL:** An open-source relational database known for its reliability.
- **MongoDB:** A NoSQL database for flexible data storage.
- **Redis:** Use Redis for caching to enhance performance.

5. Security

Security is a crucial aspect of any application.

- **Authentication:**
  - **JWT:** Use JSON Web Tokens for stateless authentication.
  - **OAuth 2.0:** Implement authorization using OAuth 2.0.
  - **OpenID Connect:** Use OpenID Connect for secure authentication.
- **Authorization:**
  - **Role-based:** Implement role-based access control.
  - **Policy-based:** Use policies to handle complex authorization logic.
  - **Claims-based:** Use claims to manage user-specific data.
- **Data Protection:** Encrypt sensitive data to ensure privacy.
- **HTTPS:** Use HTTPS to secure communications between clients and servers.
- **Cross-Site Scripting (XSS):** Prevent XSS attacks by sanitizing user inputs.
- **Cross-Site Request Forgery (CSRF):** Protect against CSRF attacks with anti-forgery tokens.
- **SQL Injection Prevention:** Safeguard your application from SQL injection attacks.

6. Testing

Testing is essential for delivering robust applications.

- **Unit Testing:**
  - **MSTest:** Use MSTest for unit testing .NET applications.
  - **NUnit:** NUnit is a popular testing framework.
  - **xUnit:** Another widely used testing framework.
- **Integration Testing:** Test how different parts of your application work together.
- **Mocking:**
  - **Moq:** Use Moq to mock dependencies in unit tests.
  - **NSubstitute:** An alternative to Moq for mocking.
- **Test Coverage:** Ensure your tests cover all critical parts of your application.
- **TDD (Test Driven Development):** Adopt a test-first approach to writing code.

7. Advanced Concepts

7.1 Design Patterns

Understanding design patterns is crucial for writing clean, maintainable code.

- **Repository Pattern:** Abstract data access into repositories.
- **Factory Pattern:** Use factories to create objects without exposing the instantiation logic.
- **Singleton Pattern:** Ensure only one instance of a class is created.
- **Observer Pattern:** Implement event-driven architecture.
- **Strategy Pattern:** Use different algorithms interchangeably.
- **SOLID Principles:** Follow the SOLID principles for better object-oriented design.

7.2 Architecture

Learn modern architectural approaches to building scalable and maintainable applications.

- **Clean Architecture:** Separate concerns to improve code maintainability.
- **Microservices:** Design applications as a set of loosely coupled services.
- **Domain-Driven Design (DDD):** Focus on the core business logic.
- **CQRS:** Implement Command Query Responsibility Segregation for better scalability.
- **Event Sourcing:** Use events to capture state changes in the system.

8. Background Tasks

At some point, every application will require background tasks.

- **IHostedService and BackgroundService:** Use the native IHostedService and BackgroundService interfaces for simple task scheduling.
- **CRON Concepts:** Learn and implement CRON expressions
- for scheduling.
- **Hangfire:** Utilize Hangfire for easy and reliable background job processing.
- **Quartz:** Implement Quartz for more advanced job scheduling needs.

9. Logging

Enhance Your Application's Observability and Bug Tracking

- **Microsoft ILogger Interface:** Utilize the ILogger interface for logging in .NET applications.
- **Serilog:** A popular logging package, often the only one you'll need. It's my go-to tool for every new C# project.
- **Sinks:** Learn about various sinks for directing log output to different destinations.
- **Structured Logging:** Implement structured logging for better log management and querying.
- **Serilog SEQ:** Use SEQ for structured log storage and querying during development.
- **Serilog Configurations and Correlation ID:** Configure Serilog effectively and use correlation IDs for tracing requests.
- **Kibana and the ELK Stack:** Explore Kibana and the ELK stack (Elasticsearch, Logstash, Kibana) for advanced log analysis and visualization.
- **OpenTelemetry:** Integrate Open Telemetry for comprehensive observability, including metrics, logs, and traces, to monitor the performance and behavior of your applications.

10. Additional Skills

10.1 Frontend Technologies

Having knowledge of frontend technologies can be a valuable addition.

- **HTML/CSS:** Master the basics of web markup and styling.
- **JavaScript:** Learn JavaScript to manipulate the DOM and handle events.
- **TypeScript:** Use TypeScript for better type safety in JavaScript development.
- **Angular/React/Vue.js:** Get familiar with popular JavaScript frameworks.
- **Blazor:** For .NET developers, Blazor allows you to build interactive web UIs using C# instead of JavaScript.

11 Best Practices & Tips

- Always follow coding standards and conventions.
- Write clean, maintainable code.
- Implement proper error handling.
- Use async/await where appropriate.
- Implement proper logging.
- Write comprehensive documentation.
- Follow security best practices.
- Optimize performance.
- Write unit tests.
- Keep learning and staying updated.

12 Recommended Learning Path

- Start with C# fundamentals.
- Move to .NET Core basics.
- Learn web development with ASP.NET Core.
- Master database operations with Entity Framework.
- Implement security features.
- Practice testing.
- Study advanced concepts.
- Explore cloud deployment.
- Learn monitoring and optimization.
- Keep practicing and building projects.

13 Project Ideas for Practice

- Task Management System
- E-commerce Platform
- Blog Engine
- API Gateway
- Authentication Service
- Real-time Chat Application
- File Storage Service
- Payment Processing System
- Booking System
- Social Media API

14. Good to Know Libraries

At some point, every application will require background tasks.

- **Refit:** Utilize Refit for making HTTP calls.
- **FluentValidation:** Implement FluentValidation to validate incoming requests.
- **ProblemDetails:** Use Problem Details for structured error handling.
- **SignalR:** Use SignalR for real-time communication.
- **API Versioning:** Implement API versioning to manage changes over time.
- **Scrutor:** Automate dependency injection with Scrutor.
- **Carter:** Enhance minimal API routing with Carter.
- **AutoMapper/Mapster/Mapperly:** Leverage AutoMapper,
- Mapster, or Mapperly for object mapping.
- **Sonar Analyzers:** Integrate Sonar Analyzers for code quality analysis.
- **YARP Reverse Proxy:** Use YARP as a reverse proxy.
- **CQRS Pattern with MediatR:** Learn the CQRS pattern usingthe MediatR library.
- **Benchmark.NET:** Evaluate your application performance
- with Benchmark.NET.

KanaiyaKatarmal

15 .LLMs & AI Tools

Leveraging Large Language Models (LLMs) and AI tools can boost productivity, automate coding tasks, and even enhance applications with intelligent features.

AI Coding Assistants:

- **GitHub Copilot / Copilot Labs** – AI code suggestions directly in your IDE.
- **Tabnine** – AI autocompletion and suggestions for .NET and other languages.
- **Kite** – AI code completions and documentation hints.

LLM SDKs & Frameworks for .NET:

- **OpenAI .NET SDK** – Integrate OpenAI's GPT models into .NET applications.
- **LangChain .NET** – Build advanced LLM-powered workflows, chatbots, and agents.
- **Semantic Kernel (Microsoft)** – Embed LLMs into .NET apps for reasoning, planning, and task automation.
- **Azure OpenAI Service** – Microsoft's managed OpenAI service for .NET developers.

AI-Powered Tools for Development & Productivity:

- **CodiumAI** – AI-based testing suggestions and code quality improvements.
- **Sorcerer** – Automated code documentation and refactoring assistance.
- **Cogram** – AI pair programmer for faster coding and debugging.

Want more tips?

Follow us and share this post with others who might find it helpful!

Get weekly tips straight to your inbox

newsletter.kanaiyakatarmal.com

KanaiyaKatarmal

←