C# & .Net core Learning Path

AI summary	This learning path for C# and .NET Core covers C# basics, OOP concepts, advanced C# concepts, database connectivity, .NET Core Web API, and Entity Framework Core.
Created	@September 15, 2023 10:02 AM
Learning references	C# Basics ASP .net core Web API Entity Framework Miscellaneous
Tags	.net core C# Training entity framework

Week 1: C# Basics

Day 1-2: C# Fundamentals

- Introduction to C# and the .NET framework
- Data types, variables, and operators in C#
- Control constructs: if statements, loops, and switch statements
- Exception handling and error management

Assignments

- Create a C# console application that takes user input for two numbers and performs basic arithmetic operations (addition, subtraction, multiplication, division).Write code to handle exceptions that may occur during user input or calculations

Day 3-4: Object-Oriented Programming (OOP)

- OOP concepts: classes, objects, inheritance, encapsulation, polymorphism
- Defining and implementing classes, methods, properties
- Interfaces and abstract classes
- Extension methods and custom attributes

Assignments

- Implement a class hierarchy for a simple shape system (e.g., Circle, Rectangle, Triangle) with methods to calculate area and perimeter. Create instances of these shapes and demonstrate polymorphism.
- Implement same assignment with interfaces and abstract classes

Day 5-6: Advanced C# Concepts

- IDisposable pattern and using keyword
- Generics in C#
- Collections in C# (List, Dictionary, IEnumerable)
- Task Parallel Library (TPL)
- Using async and await keywords
- Delegates and events
- Lambda expressions (Expression class)
- Func, Action, Predicate delegates
- LINQ basics (where, joins, select, group)

Assignments

- Build a program that reads data from a CSV file and performs LINQ queries to filter and sort the data. Use generics to create a reusable collection class for managing a list of objects

Day 7: Database Connectivity

- Connecting to a database using ADO.NET
- Introduction to SQL queries in C#
- Querying data using prepared statements
- Invoking stored procedure
- Introduction to Entity Framework Core

Assignment

- Connect to a local SQL Server database using ADO.NET and retrieve data from a table. Display the data in a console application. Experiment with different SQL queries to modify and retrieve specific data

Week 2: .NET Core Web API and Entity Framework

Day 1: .NET Core Web API Introduction

- RESTful principles and HTTP verbs
- Introduction to .NET Core Web API
- Setting up a basic Web API project

Assignment

- Create a basic .NET Core Web API project with an endpoint that returns a list of dummy data (e.g., a list of products).Test the API using tools like Postman or Swagger.

Day 2: Web API Architecture

- Understanding host builder and program.cs file
- Hosting Web API in Kestrel and IIS server
- Services and dependency injection (Singleton, Scoped, Transient)
- Middleware and request pipeline

Assignment

- Add additional endpoints to the Web API for creating, updating, and deleting items from the dummy data. Implement dependency injection to inject a service into the controller and use it for data operations.

Day 3: Advanced Web API Concepts

- Filters and their usage (Action, Authorization, Exception, Result, Service filter)

- Content negotiation for response format
- Configuration management using Appsettings (reading configurations from appsettings)
- Logging in .NET Core (ILogger Interface)

Assignment

- Implement global exception handling and logging for your Web API. Create custom Authorization filter using API key.

Day 4: Caching and Validation

- Caching mechanisms in .NET Core
- Model validation via annotations
- Automapper for object mapping

Assignments

- Implement caching for frequently requested data in your Web API. Add model validation annotations to your API models and test validation with invalid data.
- Add memory cache to get API’s and experiment with different caching options

Day 5: Background Processing

- Background processing using hosted services
- CORS configuration for cross-origin requests
- Introduction to Swagger UI for API documentation

Assignments

- Create a background service that performs a specific task at regular intervals (e.g., health check of API). Configure CORS to allow cross-origin requests from a different domain.

Day 6-7: Entity Framework Core

- Understanding ORM and its importance
- Db context class and its role
- Code-first approach and migrations
- Db-first approach and scaffolding entities
- Fluent API for advanced configuration
- IQueryable and querying data
- Implementing a basic repository pattern with EF Core

Assignments

- Extend your Web API to perform CRUD operations on a database using Entity Framework Core. Implement the repository pattern to separate data access code from controllers.

Final Assignment

Building a Task Management API

Objective: Create a RESTful Task Management API using .NET Core Web API and Entity Framework Core. This assignment will cover a range of topics, including routing, database operations, middleware, validation, and more.

Requirements:

1. **Task Model:** Create a Task model class with properties like `Id`, `Title`, `Description`, `DueDate`, and `IsCompleted`.
2. **Database Setup:** Use Entity Framework Core to set up a database for storing tasks. You can choose either a Code-First or Database-First approach.
3. **API Endpoints:**
   - Implement CRUD operations (Create, Read, Update, Delete) for tasks.
   - Design endpoints like `GET /api/tasks`, `GET /api/tasks/{id}`, `POST /api/tasks`, `PUT /api/tasks/{id}`, and `DELETE /api/tasks/{id}`.

4. **Validation:** Implement validation for task creation and updating. Ensure that fields like `Title` and `DueDate` are required, and the `DueDate` should be in the future.
5. **Middleware:**
    - Use middleware to log incoming requests.
    - Implement global exception handling middleware to return meaningful error responses.
6. **Dependency Injection:** Utilize dependency injection for services responsible for handling tasks and database operations.
7. **CORS:** Configure CORS to allow requests from specific origins.
8. **Swagger:** Implement Swagger UI for API documentation. Document your endpoints and models using XML comments.
9. **Background Processing:** Create a background service that marks tasks as overdue when their due date has passed.
10. **Authentication and Authorization (Optional):** Implement API key based authentication and authorization to secure your API.