What is .NET?

.NET is a framework for building web, desktop, and mobile apps. It supports multiple languages, but C# is the most common for web development. We use ASP.NET Core to build modern web applications.

ASP.NET Core

ASP.NET Core is a modern, open-source, cross-platform framework for building web applications and APIs using C# and the .NET ecosystem.

- Works on Windows, Linux, and macOS.
- It is the successor of the ASP.NET Framework and is faster and more modular.
- Supports MVC, Razor Pages, Web APIs, and Blazor.
- High performance and lightweight.

Understanding the Project Structure

```
📂 MyWebApp (Project folder)
|-- 📂 wwwroot (Static files like CSS, JS, images)
|-- 📂 Pages (Razor Pages - optional)
|-- 📂 Controllers (MVC logic - if using MVC)
|-- 📄 Program.cs (Main entry point of the app)
|-- 📄 appsettings.json (Configuration settings)
```

Note:

- This file is the starting point of the app.
- Generates production-ready files inside the publish folder.
- ✔ Controllers handle requests.
- ✔ Views display UI.
- ✔ Models manage data.

Key Features of ASP.NET Core

1️⃣ Cross-Platform & Open Source

- Can run on Windows, Linux, and macOS.
- Fully open-source on GitHub.

2️⃣ High Performance & Lightweight

- Built for speed and efficiency (faster than older ASP.NET versions).
- Uses Kestrel web server for lightweight hosting.

3️⃣ Modular & Flexible

- Uses dependency injection (DI) for better code management.

- Middleware-based pipeline (only load what you need).

4️⃣ **Supports Multiple Architectures**

- MVC (Model-View-Controller) – for structured web apps.
- Razor Pages – for simpler page-based development.
- Web API – for building RESTful APIs.
- Blazor – for interactive web UIs using C# instead of JavaScript.

5️⃣ **Cloud & Microservices Ready**

- Works seamlessly with Docker, Kubernetes, and Azure.
- Can be used for microservices architectures.

6️⃣ **Built-in Security**

- Supports authentication & authorization (JWT, Identity, OAuth).
- Data protection, CSRF, and HTTPS enforcement out of the box.

Step-by-Step Guide

Step 1: Install Required Tools

Make sure you have the following installed:

- .NET SDK (Check by running `dotnet --version` in the terminal)
- VS Code with the C# Extension (Install from the Extensions tab in VS Code)

Step 2: Create a New .NET Web Application

1. Open VS Code.
2. Open the Terminal (press `Ctrl + ~` or go to **View** → **Terminal**).
3. Run the following command to create a new ASP.NET Core web app:

   ```
   dotnet new webapp -n MyWebApp
   cd MyWebApp
   ```

4. Open the project in VS Code:

   ```
   code .
   ```

Step 3: Run the Application

1. In the VS Code Terminal, run:

   ```
   dotnet run
   ```

2. Open a browser and go to `http://localhost:5000` (or the port shown in the terminal).

Step 4: Modify the Application

1. Add a Controller

Inside the `MyWebApp` folder, create a new folder named **Controllers**. Inside **Controllers**, create a new file **HomeController.cs** and add:

```
using Microsoft.AspNetCore.Mvc;

namespace MyWebApp.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

2. Create a View

1. Inside `MyWebApp`, create a **Views** folder.
2. Inside `Views`, create a **Home** folder.
3. Inside `Views/Home`, create a new file **Index.cshtml** and add:

```
<h1>Welcome to My Web App</h1>
<p>This is a simple .NET Core web application.</p>
```

3. Configure Routing

Open `Program.cs` and modify:

```
var builder = WebApplication.CreateBuilder(args);

// Add MVC support
builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages(); // Keep if Razor Pages are needed

var app = builder.Build();

// Middleware setup
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

// app.UseHttpsRedirection();
// app.UseStaticFiles();

app.UseRouting();
app.UseAuthorization();

// Ensure MVC Controllers are mapped properly
app.MapControllerRoute(
    name: "default",
```

```
    pattern: "{controller=Home}/{action=Index}/{id?}");

// app.MapRazorPages(); // Keep if Razor Pages are needed

app.Run();
```

Step 5: Run & Debug the App

- Run the app in the terminal:

  ```
  dotnet run
  ```

- Debugging in VS Code:
  1. Press `F5` to start debugging.
  2. Select `.NET Core` if prompted.
  3. The app should launch in your browser.

Publish the Application

If you want to deploy your application, publish it using:

```
dotnet publish -c Release -o ./publish
```

Then, deploy it to **IIS, Azure, or Docker**.