# .NET Core Overview

.NET Core is a cross-platform, open-source framework developed by Microsoft to build modern applications. It’s designed to be lightweight, modular, and high-performance, offering the ability to build applications that run on Windows, Linux, and macOS.

.NET Core supports:

* **Web applications** (using ASP.NET Core).
* **Microservices** (REST APIs, gRPC, etc.).
* **Console applications**.
* **Cloud-based apps**.
* **Mobile and IoT applications** (through Xamarin and other tools).

It provides significant performance improvements over its predecessor, the .NET Framework, and supports modern development practices such as Dependency Injection, middleware, and async programming.

## ASP.NET Core

ASP.NET Core is a web framework for building modern web applications, APIs, and microservices. It’s a cross-platform version of ASP.NET, optimized for cloud and server-side performance. ASP.NET Core is designed to be faster, modular, and lightweight.

### Key Features:

* **Cross-platform**: It runs on Windows, Linux, and macOS.
* **Built-in Dependency Injection**: Allows for better testability and easier maintenance of your code.
* **Middleware pipeline**: It provides powerful request handling via a pipeline that can be customized and extended.
* **Unified MVC and Web API**: Combines MVC and Web API into a single framework.
* **Performance**: ASP.NET Core is optimized for high performance, including features like Kestrel (a lightweight, high-performance web server).

## Project Structure in ASP.NET Core

In ASP.NET Core, the default project structure is designed to be simple, but also modular and extensible.

1. **Program.cs**: The entry point of the application, which configures and starts the application.
2. **Startup.cs**: Contains configuration and services for the application (e.g., middleware, services, etc.).
3. **Controllers**: Contains classes that handle HTTP requests and return responses.
4. **Models**: Contains classes representing the data entities.
5. **Views**: Contains Razor Views used in MVC applications.
6. **wwwroot**: The web root folder where static files (CSS, JavaScript, images) are stored.
7. **appsettings.json**: Configuration file used for application settings.
8. **launchSettings.json**: Configuration file used for development environment settings (e.g., which browser to open, ports to listen on).

### Example of Typical Structure:

MyAspNetCoreApp/  
│  
├── Controllers/  
│ └── HomeController.cs  
│  
├── Models/  
│ └── Product.cs  
│  
├── Views/  
│ └── Home/  
│ └── Index.cshtml  
│  
├── wwwroot/  
│ ├── css/  
│ ├── js/  
│ └── images/  
│  
├── appsettings.json  
├── launchSettings.json  
├── Program.cs  
├── Startup.cs  
└── MyAspNetCoreApp.csproj

## wwwroot Folder

The **wwwroot** folder is the default directory for static files such as HTML, CSS, JavaScript, images, and other media. When a request is made to the server for a static file, the framework looks for the file in the **wwwroot** folder.

* **Static files** (like .css, .js, .png) are publicly accessible from the browser.
* The **wwwroot** folder can be customized for your application if necessary.

Example: To serve a CSS file in a web page, the folder structure would look like this:

wwwroot/  
└── css/  
 └── styles.css

You can link to it in your HTML like this:

<link href="~/css/styles.css" rel="stylesheet" />

## Program.cs

The **Program.cs** file is the entry point for your ASP.NET Core application. This file typically contains the Main method, where the application is started. This file is used to set up configurations and services for your application, and it configures the **Startup.cs** class.

Example:

public class Program  
{  
 public static void Main(string[] args)  
 {  
 CreateHostBuilder(args).Build().Run();  
 }  
  
 public static IHostBuilder CreateHostBuilder(string[] args) =>  
 Host.CreateDefaultBuilder(args)  
 .ConfigureWebHostDefaults(webBuilder =>  
 {  
 webBuilder.UseStartup<Startup>();  
 });  
}

* The CreateHostBuilder method sets up the web server and points to the **Startup.cs** class for further configuration.

## Startup.cs

The **Startup.cs** class is where the core configurations for your ASP.NET Core application are made. It defines how the app should respond to HTTP requests by configuring middleware, routing, and services (like database connections, logging, etc.).

The two main methods in **Startup.cs** are:

1. **ConfigureServices**: Used to register services with the dependency injection container (like database contexts, authentication services, etc.).
2. **Configure**: Used to define the middleware pipeline for processing HTTP requests.

Example:

public class Startup  
{  
 public Startup(IConfiguration configuration)  
 {  
 Configuration = configuration;  
 }  
  
 public IConfiguration Configuration { get; }  
  
 public void ConfigureServices(IServiceCollection services)  
 {  
 // Register services here, such as MVC, authentication, etc.  
 services.AddControllersWithViews();  
 }  
  
 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  
 {  
 if (env.IsDevelopment())  
 {  
 app.UseDeveloperExceptionPage();  
 }  
 else  
 {  
 app.UseExceptionHandler("/Home/Error");  
 app.UseHsts();  
 }  
  
 app.UseHttpsRedirection();  
 app.UseStaticFiles(); // To serve static files from wwwroot  
 app.UseRouting();  
 app.UseAuthorization();  
  
 app.UseEndpoints(endpoints =>  
 {  
 endpoints.MapControllerRoute(  
 name: "default",  
 pattern: "{controller=Home}/{action=Index}/{id?}");  
 });  
 }  
}

* **ConfigureServices**: This method is used to add services like MVC, Entity Framework, Identity, etc., to the DI container.
* **Configure**: This method configures the middleware pipeline, which handles HTTP requests.

## launchSettings.json

The **launchSettings.json** file is used to configure the behavior of your application during development, including the URL for the local web server, environment variables, and how the app should be started.

Example:

{  
 "iisSettings": {  
 "windowsAuthentication": false,  
 "anonymousAuthentication": true,  
 "iisExpress": {  
 "environmentVariables": {  
 "ASPNETCORE\_ENVIRONMENT": "Development"  
 }  
 }  
 },  
 "profiles": {  
 "IIS Express": {  
 "environmentVariables": {  
 "ASPNETCORE\_ENVIRONMENT": "Development"  
 }  
 },  
 "ProjectName": {  
 "commandName": "Project",  
 "environmentVariables": {  
 "ASPNETCORE\_ENVIRONMENT": "Development"  
 },  
 "applicationUrl": "http://localhost:5000"  
 }  
 }  
}

* The applicationUrl property determines the URL on which the application will run during development.
* The ASPNETCORE\_ENVIRONMENT setting defines the environment (e.g., Development, Staging, Production).

## appSettings.json

The **appSettings.json** file is used to store application configuration settings, such as connection strings, logging settings, and custom configurations.

Example:

{  
 "ConnectionStrings": {  
 "DefaultConnection": "Server=localhost;Database=MyDatabase;User Id=myuser;Password=mypassword;"  
 },  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft": "Warning",  
 "Microsoft.Hosting.Lifetime": "Information"  
 }  
 },  
 "AllowedHosts": "\*"  
}

* The ConnectionStrings section is typically used to store database connection strings.
* The Logging section is used to configure logging levels.
* The AllowedHosts section is used for security purposes, specifying which hosts are allowed to access the app.

## Example of an ASP.NET Core Application

Let’s build a simple web application to illustrate the concepts.

1. **Program.cs**:

public class Program  
{  
 public static void Main(string[] args)  
 {  
 CreateHostBuilder(args).Build().Run();  
 }  
  
 public static IHostBuilder CreateHostBuilder(string[] args) =>  
 Host.CreateDefaultBuilder(args)  
 .ConfigureWebHostDefaults(webBuilder =>  
 {  
 webBuilder.UseStartup<Startup>();  
 });  
}

1. **Startup.cs**:

public class Startup  
{  
 public IConfiguration Configuration { get; }  
  
 public Startup(IConfiguration configuration)  
 {  
 Configuration = configuration;  
 }  
  
 public void ConfigureServices(IServiceCollection services)  
 {  
 services.AddControllersWithViews();  
 }  
  
 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  
 {  
 if (env.IsDevelopment())  
 {  
 app.UseDeveloperExceptionPage();  
 }  
 else  
 {  
 app.UseExceptionHandler("/Home/Error");  
 app.UseHsts();  
 }  
  
 app.UseHttpsRedirection();  
 app.UseStaticFiles();  
 app.UseRouting();  
 app.UseAuthorization();  
  
 app.UseEndpoints(endpoints =>  
 {  
 endpoints.MapControllerRoute(  
 name: "default",  
 pattern: "{controller=Home}/{action=Index}/{id?}");  
 });  
 }  
}

1. **appSettings.json**:

{  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft": "Warning",  
 "Microsoft.Hosting.Lifetime": "Information"  
 }  
 },  
 "AllowedHosts": "\*"  
}