# DOT NET CORE Introduction

## DOT NET FW

* Windows only: Runs on windows only, tightly coupled with windows.
* Extensive library support and 3rd party integration.
* Compatible with existsing .net applications.
* Rich window specific APIs.
* Deployed using GAC (Global Assembly Cache).

## DOT NET Core

* Runs on windows, macOS, Linux.
* Light weight compared to .NET fw. So utilized less disk space and memory usage at runtime.
* Modular architecture => allowing developers to include only the necessary libraries.
* Gets latest language features and improvements faster.
* Self contained deployment.

## .NET Core Overview

.NET Core is a cross-platform, high-performance framework for building modern, cloud-based, and internet-connected applications. It is an open-source project maintained by Microsoft and the .NET community. .NET Core applications can run on Windows, macOS, and Linux. Key features of .NET Core include:

**Cross-Platform**: Write applications that can run on multiple operating systems.

**Performance**: High-performance server runtime for scalable and efficient applications.

**Unified Framework**: One framework for building web, mobile, desktop, cloud, and IoT applications.

**Modular**: Install only the libraries needed for your application.

**Side-by-Side Versioning**: Install multiple versions of .NET Core on the same machine.

**Open Source**: Open development and contributions from the community.

## ASP.NET Core

ASP.NET Core is a framework for building web applications and APIs. It is a redesigned version of ASP.NET, optimized for modern web development. ASP.NET Core supports the following features:

**Unified MVC and Web API**: Combines MVC and Web API into a single programming model.

**Razor Pages**: Simplified, page-focused approach to web development.

**Dependency Injection**: Built-in support for dependency injection to manage application dependencies.

**Middleware Pipeline**: Flexible request handling pipeline with middleware components.

**Cross-Platform**: Develop and run applications on Windows, macOS, and Linux.

**High Performance**: Designed for high throughput and low latency.

**Cloud-Ready**: Built with cloud deployments in mind, including support for modern hosting platforms and containers.

## Project Structure

An ASP.NET Core project typically follows a standard structure to organize code and resources. Key directories and files include:

**Controllers**: Contains controller classes responsible for handling HTTP requests and returning responses.

**Models**: Defines the data models used in the application.

**Views**: Contains Razor view files for rendering HTML UI.

**wwwroot**: Stores static files such as CSS, JavaScript, and images.

**App\_Data**: Used for application data storage, if needed.

**Properties**: Contains project properties and configuration settings.

**Startup.cs:** Configures services and the request pipeline.

**Program.cs**: Entry point for the application.

**appSettings.json**: Configuration settings file.

## wwwroot Folder

The wwwroot folder is the default location for serving static files in an ASP.NET Core application. It is the root directory for publicly accessible web content. Common subfolders within wwwroot include:

Css, js, lib, images.

Files placed in wwwroot are accessible via URL, and no server-side processing is applied to them.

## Program.cs

The Program.cs file is the entry point of an ASP.NET Core application. It contains the Main method, which is the starting point of the application. This file sets up the application's host and runs it. Key elements include:

**Host Creation**: Uses CreateHostBuilder method to configure and create the host.

**Host Configuration**: Configures web server settings, logging, and other host-specific options.

**Host Run**: Starts the host and begins listening for incoming requests.

## Startup.cs

The Startup.cs file is where you configure services and the application's request handling pipeline. It includes two key methods:

**ConfigureServices**: Used to add services to the application's dependency injection container.

**Configure**: Defines the middleware pipeline for handling HTTP requests.

## launchSettings.json

The launchSettings.json file contains settings that define how the application is launched. It includes configurations for different environments, such as development and production. Key settings include:

**profiles**: Defines different launch profiles, each with specific settings.

**environmentVariables**: Specifies environment variables for the application.

**applicationUrl**: The URL the application will use.

## appSettings.json

The appSettings.json file is used to store configuration settings for an ASP.NET Core application. It supports hierarchical structures and is used to configure various aspects of the application, such as:

**Connection Strings**: Database connection information.

**Logging**: Configuration for logging providers and levels.

**Custom Settings**: Any other application-specific settings.