.NET Core Fundamentals

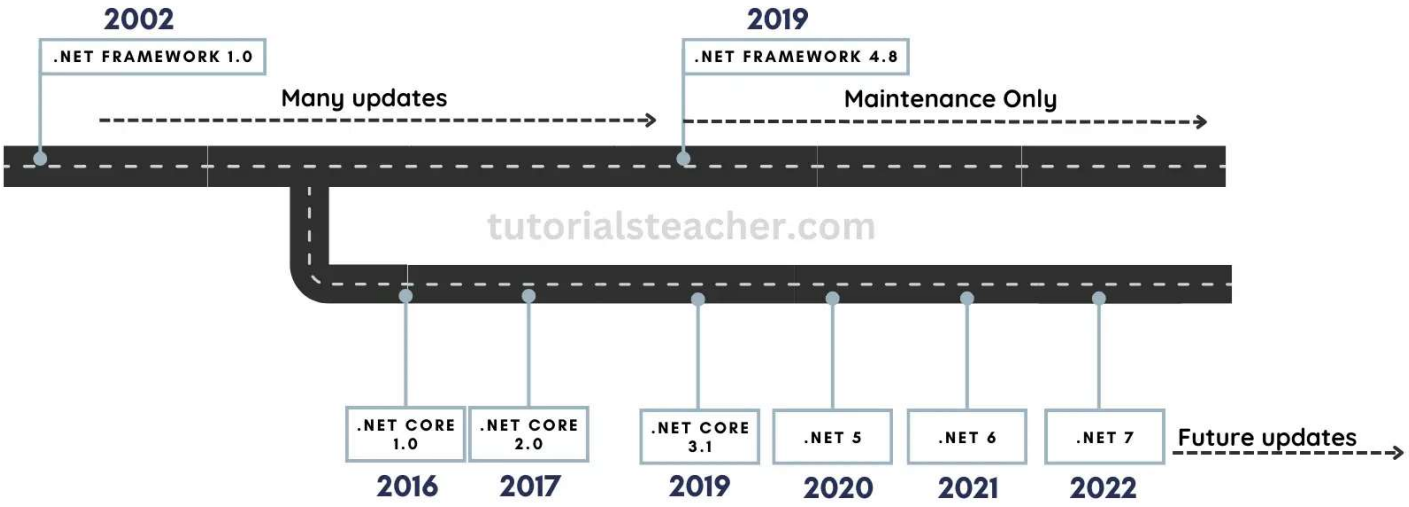
Base of .NET Core

Yash Lathiya

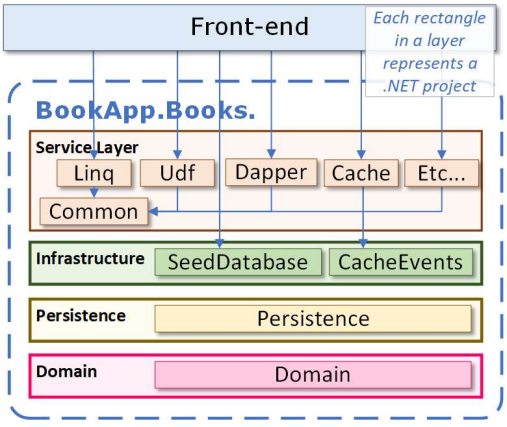
Table of Contents

.NET Core Overview

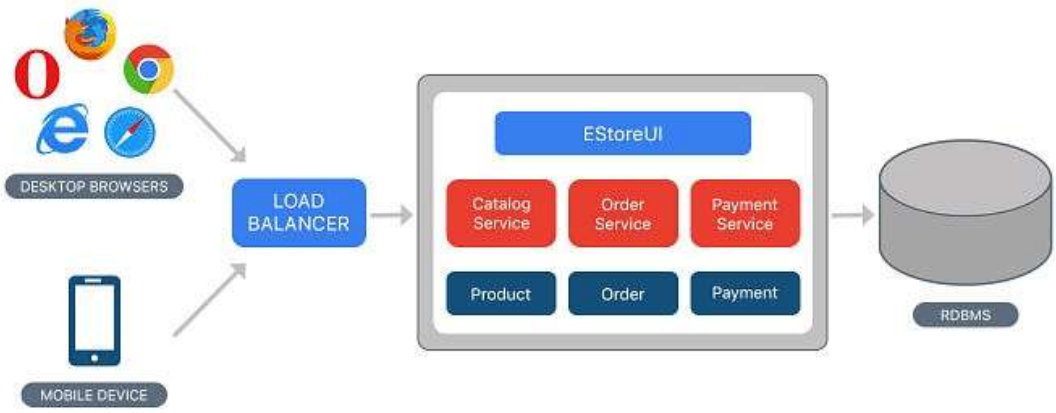
- .NET core is upgraded streamline of .NET framework.
- .NET framework support only windows OS while .NET core is cross-platform technology, It supports mac, windows & linux.



- Open Source
- Wide range of application
- Supports C#, F# & VB
- Includes CLI tools
- Supports flexible deployments (docker, user-wide & system-wide)
- High Performance
- Consistent over all architecture
- Modular Architecture



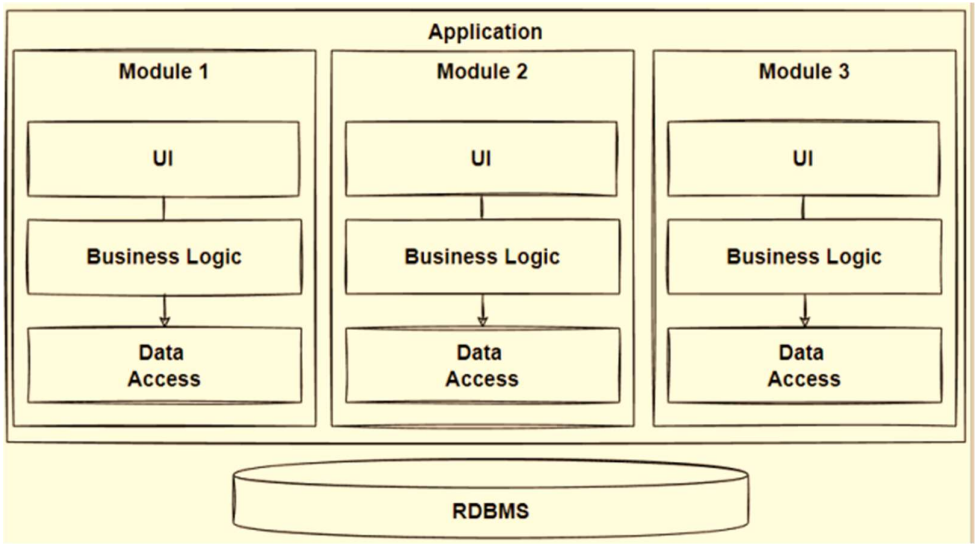
[Monolithic Architecture]
[mostly used for light-weight application]



[Microservice Architecture]
[mostly used for complexed application]

[Also needs inter service mechanism]

[Modular Monolithic Architecture]

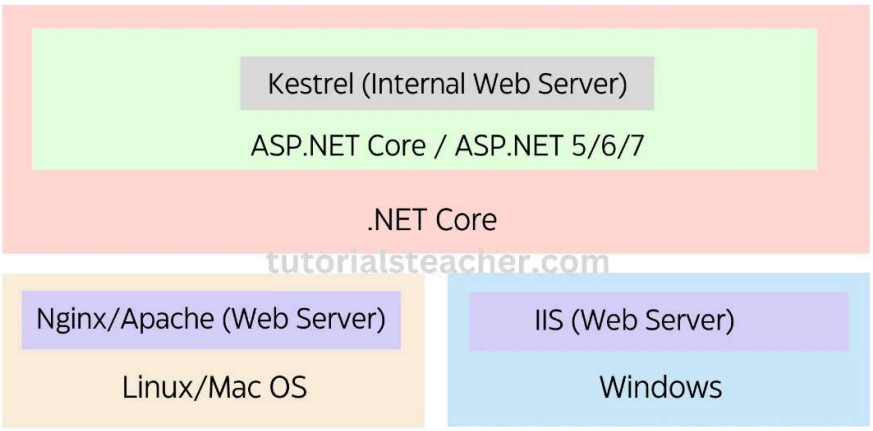


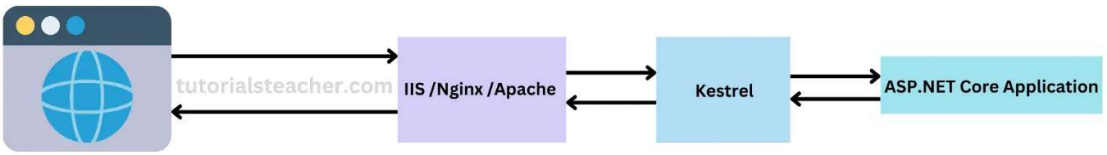
ASP .NET Core Overview

- ASP .NET is newly re-written from .NET framework,
- ASP .NET is for web application development.

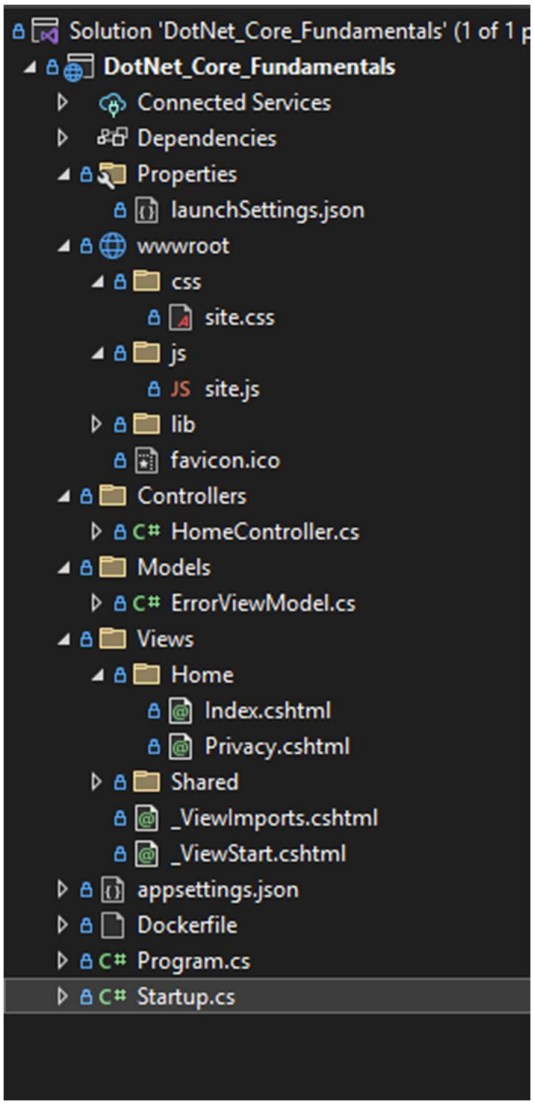
Features

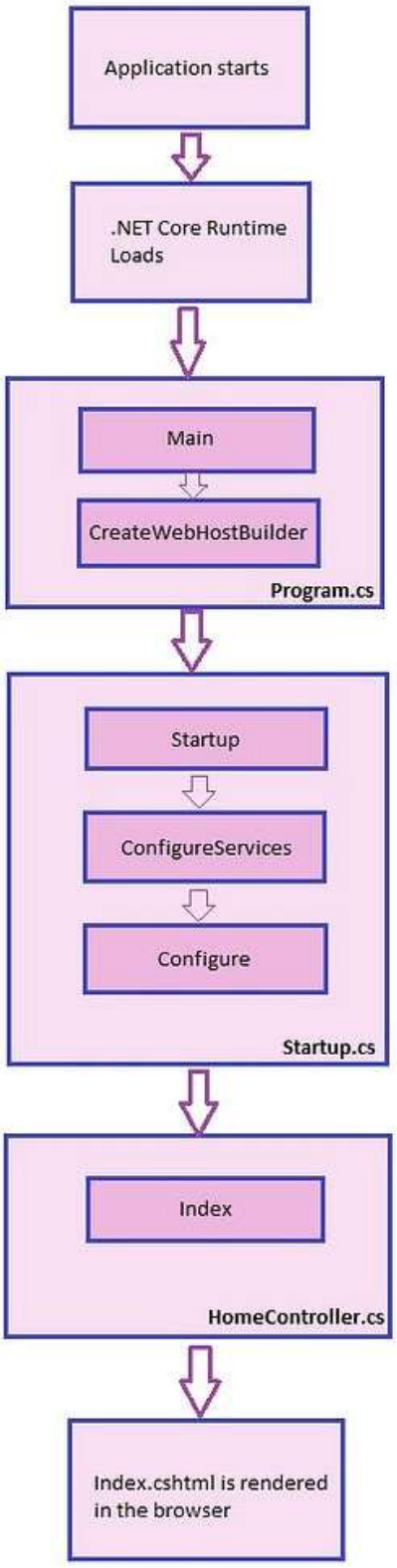
- Supports Multiple Platforms – OS
- Fast
- IoC Container (built-in container for automatic DI)
- Integration with modern UI Frameworks (AngularJS, ReactJS)
- Code Sharing across versions
- Side-by-side app versioning
- Smaller deployment footprint (size)
- Hosting





Project Structure





wwwroot Folder

- **Static Content Hosting**:

  The wwwroot folder in ASP.NET Core serves as the hosting location for static web assets.

- **Direct Client Access**:

  Files stored in wwwroot are directly accessible to clients via their URLs without any server-side processing.

- **File Types**:

  Common static assets stored in wwwroot include HTML, CSS, JavaScript, images, fonts, and other files that don't require server-side computation.

- **No Processing Overhead**:

  ASP.NET Core serves files from wwwroot efficiently, without any additional processing, improving performance and responsiveness.

- **Deployment**: During deployment, contents of the wwwroot folder are typically deployed to the web server or hosting environment to ensure availability to clients.

- **Organization**: Developers commonly organize static assets within subdirectories of wwwroot based on their type or purpose for better management and maintainability.

- **Efficient Content Delivery**: Using wwwroot allows for optimal content delivery, facilitating the building of modern web applications with ease.

Program.cs:

- Entry Point:

  Program.cs is the entry point for an ASP.NET Core application.

- Main Method:

  It contains the Main method, which is the starting point of the application.

- Host Builder:

  Main method typically creates a host builder, configures the web host, and runs the application.

- Configuration:

  It's responsible for configuring the application's startup and initializing necessary resources.

Startup.cs:

- Pipeline Configuration:

  Startup.cs configures the application's request processing pipeline.

- ConfigureServices Method:

  This method is used to set up services like dependency injection, logging, and configuration.

- Configure Method:

  It defines how the application handles HTTP requests by adding middleware, setting up routing, and configuring authentication and authorization.

- Middleware Configuration:

  It's where you define middleware components to process HTTP requests and responses.

launchSettings.json:

- Debugging Configuration:

  launchSettings.json defines settings for launching and debugging an ASP.NET Core application.

- Profiles:

  It contains profiles with configurations for different environments such as development, staging, and production.

- Application URL:

  Profiles specify properties like the application URL, environment variables, and command-line arguments used during debugging.

- Customization:

  Developers can customize how the application runs locally during development using this file.

appSettings.json:

- Configuration Storage:

  appSettings.json is used to store application settings in key-value pairs.

- Application Settings:

  It contains settings that configure the behavior of the application such as database connection strings, logging settings, and feature toggles.

- IConfiguration:

  Settings defined in appSettings.json can be accessed programmatically throughout the application using the IConfiguration interface.