.NET Core is the latest general purpose development platform maintained by Microsoft released in 2016.

It works across different platforms and has been redesigned in a way that makes .NET fast, flexible and modern.

This happens to be one of the major contributions by Microsoft. Developers can now build Android, iOS, Linux, Mac, and Windows applications with .NET, all in Open Source.

Characteristics of .NET Core

The following are the major characteristics of .NET Core −

Open source

- .NET Core is an open-source implementation, using MIT and Apache 2 licenses.
- .NET Core is a .NET Foundation project and is available on GitHub(cloud based repository)
- As an open source project, it promotes a more transparent development process and promotes an active and engaged community.

Cross-platform

- Application implemented in .NET Core can be run and its code can be reused regardless of your platform target.
- It currently supports three main operating systems (OS)
    - Windows
    - Linux
    - MacOS

Flexible deployment

- There can be two types of deployments for .NET Core applications –
-
    - Framework-dependent deployment
    - Self-contained deployment

- With framework-dependent deployment, your app depends on a system-wide version of .NET Core on which your app and third-party dependencies are installed. Mostly used for windows.

- With self-contained deployment, the .NET Core version used to build your application is also deployed along with your app and third-party dependencies and can run side-by-side with other versions. Mostly used for Linux and MacOS.

Command-line tools

- All product scenarios can be exercised at the command-line.
- The .NET Core command-line interface (CLI) is a new cross-platform tool for creating, restoring packages, building, running, and publishing .NET application

Compatible

- .NET Core is compatible with .NET Framework, Xamarin and Mono, via the .NET Standard Library.
- There is mono runtime for Xamarin application.

- Mono runtime is itself cross platform implementation of .net framework and can run for application like console and windows.
- Xamarin released in 2011, you can use it for creating application for iOS, mac and Android. This is used to build and run native mobile applications across mobile platforms.

.NET Standard Library - .Net standard is specification of .net APIs that has implementation for each runtime(.net f/w, .net core , Mono) because of this, code created for one runtime can be also executed by another runtime.

.Net Core Application –

1. Framework Dependent Application- Can run on only computer when .net core is installed on it.
2. Self-contained application- It contains all the best that you need to run application without installing .net core on computer

The .NET Core Platform

.NET Core Platform contains the following main parts –

- **.NET Runtime** – It provides a type system, assembly loading, a garbage collector, native interop and other basic services.

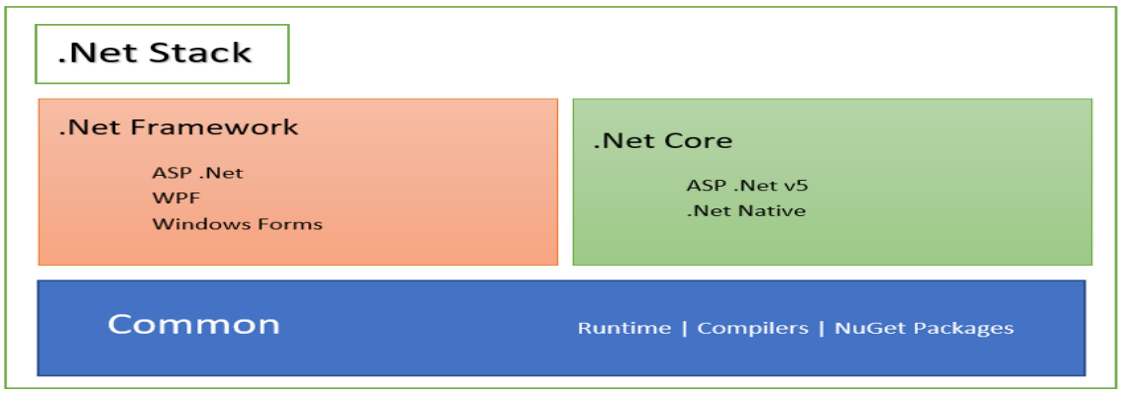
- **Fundamental Libraries** – A set of framework libraries, which provide primitive data types, app composition types and fundamental utilities.
  https://mindmajix.com/net-libraries

- **SDK & Compiler** – A set of SDK tools and language compilers that enable the base developer experience, available in the .NET Core SDK.

  https://learn.microsoft.com/en-us/dotnet/core/sdk

- **'dotnet' app host** – It is used to launch .NET Core apps. It selects the runtime and hosts the runtime, provides an assembly loading policy and launches the app. The same host is also used to launch SDK tools in much the same way.

.Net Core Architecture



This is where .Net Core sits in the .Net Stack.

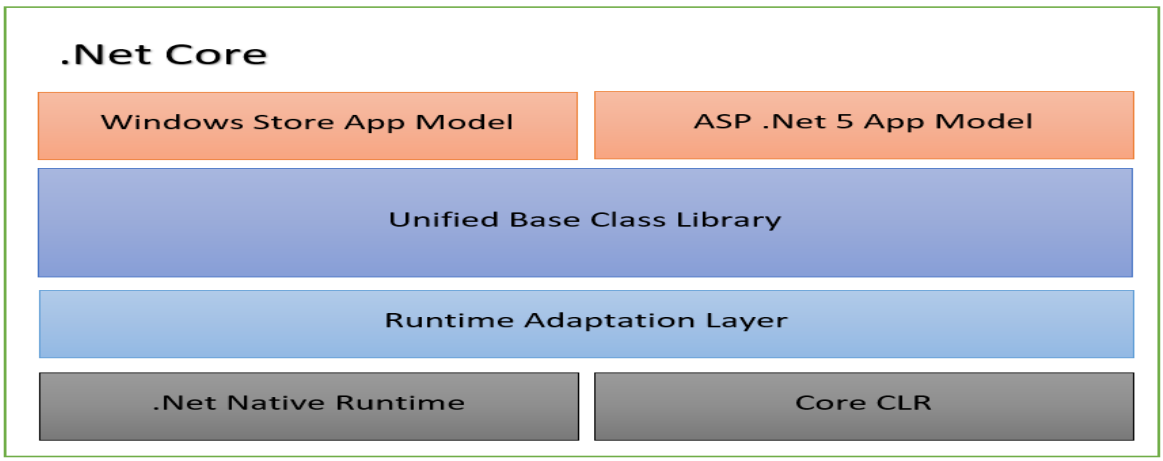
The various .Net Frameworks and libraries, till date, use the common runtime libraries, compilers, and NuGet Packages.

They build their own platform-specific libraries on top of these common packages.

The Common libraries contain the definitions for primitive stuff such as data types.

This hardly changes and is thus the base for all .Net stack frameworks.

The key terminology in architecture of .net core is as follows:

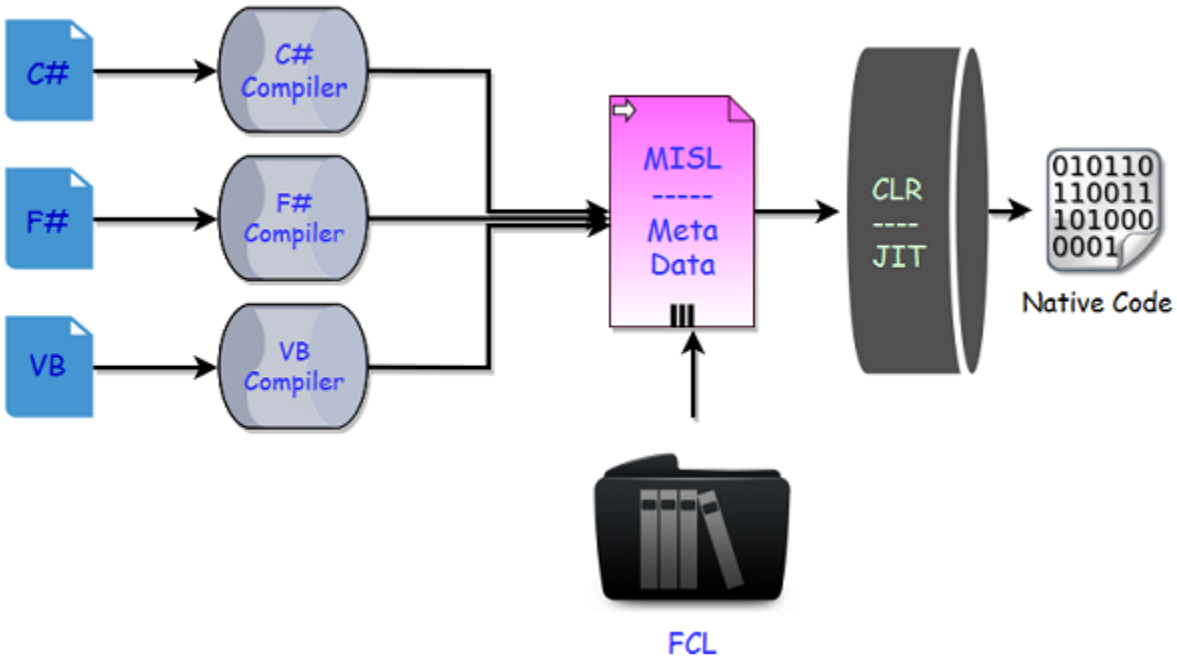


- **Core CLR:** Core CLR is the Common Language Runtime optimized for multiplatform and cloud-based deployments. This, along with .Net Native Runtime, forms the foundation of all .Net based platforms.
- **.Net Native Runtime:** Also referred to as Managed Runtime, .Net Native Runtime contains the native windows based libraries. This also contains Ahead Of Time (AOT) compilation instead of the erstwhile Just In Time (JIT) compilation. This improves the performance of the applications. The .Net Native Runtime and the Core CLR are the layers that contain implementations of primitive types as well as generic collections in .Net. These layers hardly change and are constant throughout the various .Net stacks. The various .Net stack APIs, thus, share the same implementations.
- **Unified BCL:** The Unified Base Class Library, also referred to as CoreFX, consists of the basic and fundamental classed that form the core of the .Net Core platform.
- **App Models:** On top of the BCL, sit the various App Models that developers leverage to develop platform-specific applications. Currently, .Net Core has the ASP .Net Model for web development and Windows Store Model for windows application development.

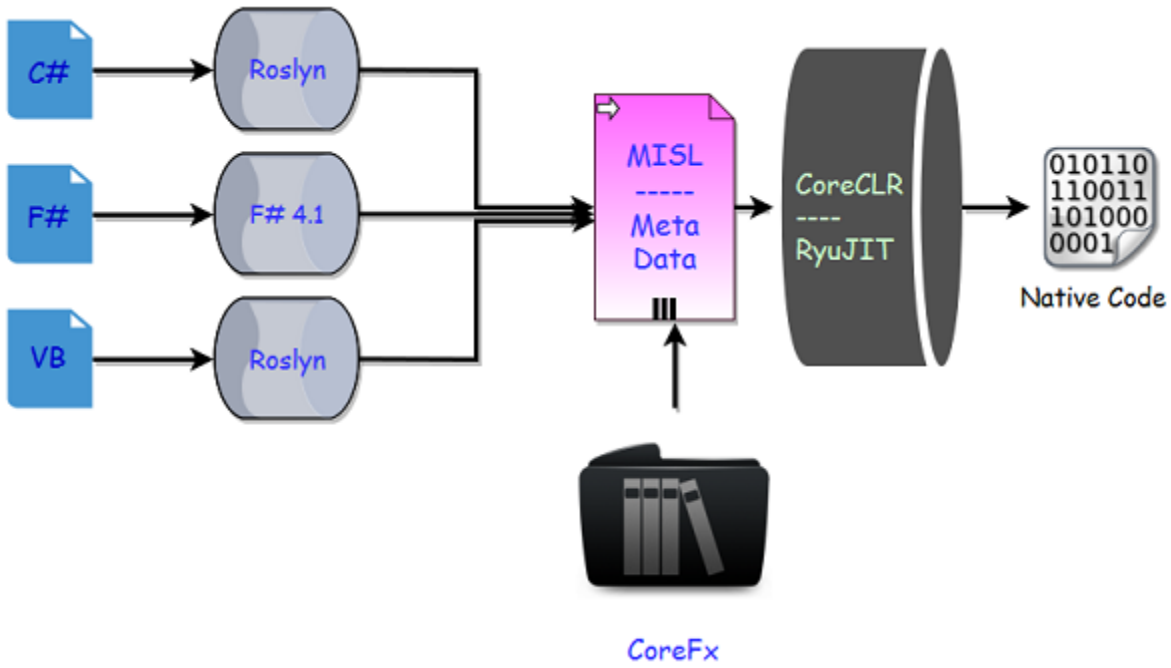
.Net Core Execution - understand the execution process of .NET Core and compare it with the .NET Framework. The managed execution process includes the following steps.

- Choosing a compiler
- Compiling your code to MSIL
- Compiling MSIL to native code
- Running code

https://www.thetechplatform.com/blog



.NET Core Code Execution Process



- In .NET Core now we have a new series of compilers, like we have Roslyn for C# and VB.
- You can also make use of the new F# 4.1 compiler if you want to use F# with .NET Core.
- Actually these tools are different and we can use Roslyn with .NET Framework as well if we are using C# 6 or later, because C# compiler can only support up to C# 5.
- In .NET Core, we don't have a framework class libraries (FCL), so a different set of libraries are used and we now have CoreFx.
- CoreFx is the reimplementation of the class libraries for .NET Core.
- We also have a new run time with .NET Core known as CoreCLR and leverages a JIT Compiler.

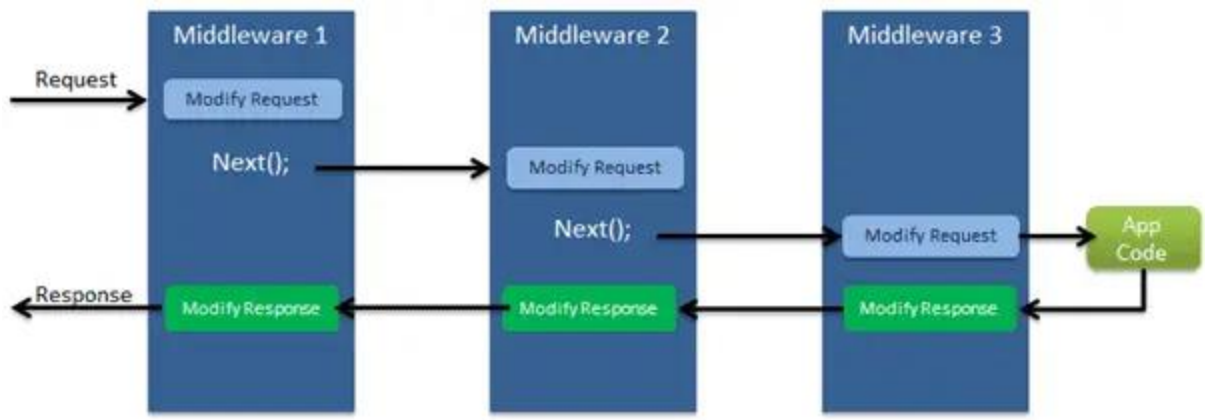
What is Middleware in ASP.NET Core

In ASP.NET Core, Middleware is a piece of software that can handle an HTTP request or response.

A given middleware component in ASP.NET Core has a very specific purpose.

For example we may have a middleware component that authenticates a user, another piece of middleware to handle errors,

yet another middleware to serve static files such as JavaScript files, CSS files, Images etc.

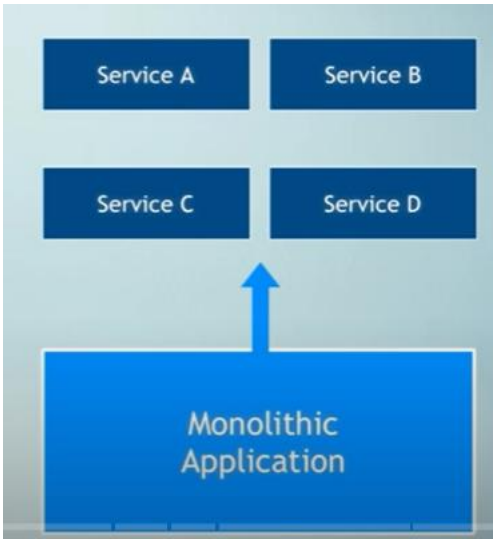


Microservices -

Monolithic App : All components are part of single unit or application

App must written in 1 tech. stack

Microservices – spilt application into smaller , independent services


Service A
Service B
Service C
Service D
Monolithic Application


products
v2.2
shopping-cart
v2.0
payment
v5.1
checkout
v5.5