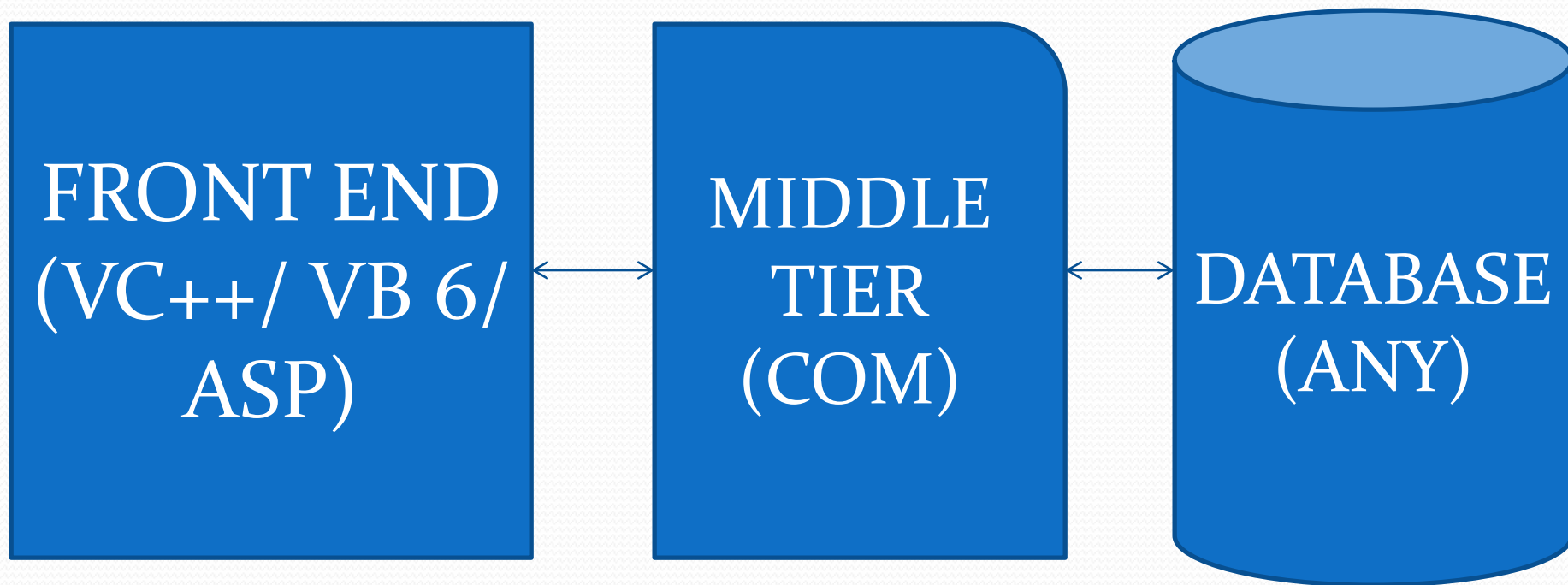


Introduction to .Net

Before .Net (Microsoft)



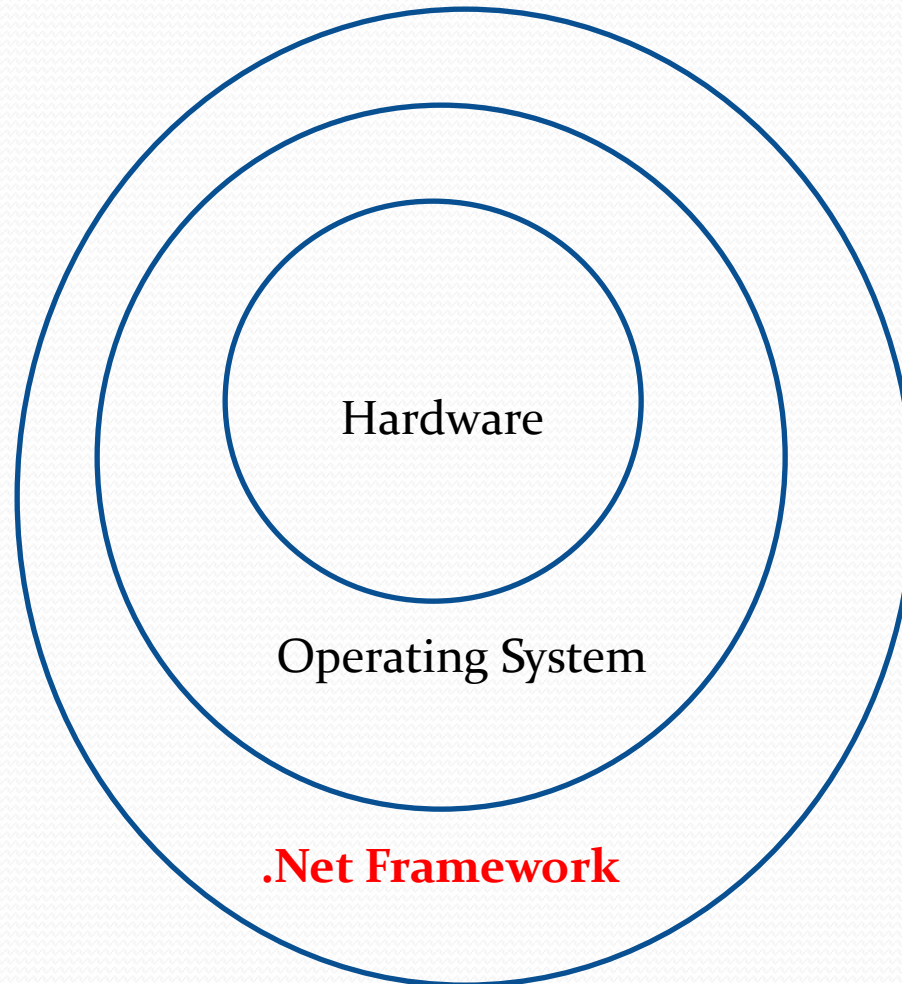
Problems (Pre .Net)

- VC++ -> Had OO and threading but Complex
- VB6 -> Not OO and no threading but Simple
- ASP -> Script based, Interpreted, Late Bound, Difficult to maintain and debug, not OO
- COM -> DLL HELL! (Mainly versioning and deployment)

.Net Features

- OO Code
- Multiple Languages
- Multiple platforms*
- Multiple project types(eg web based, desktop based, etc)
- Better Security
- Improved Performance

.Net framework



SOURCE CODE

(Any .Net
Language)
Eg, C#, Vb.Net

COMPILE

ASSEMBLY (EXE/DLL)

BYTE CODE
MSIL/CIL/IL

Web Application, Windows Forms, Console Apps, Web Services, WCF, WPF, Workflow, ASP.Net MVC, .Net Core, Xamarin, Windows Services, Web API

.Net Base Class Library

System.Dll, System.Data.Dll, System.Xml.Dll etc

Common Language Runtime(CLR)

JIT Compilation

Memory Management

Garbage Collection

App Domain Management*

Common Language Specification

Common Type System

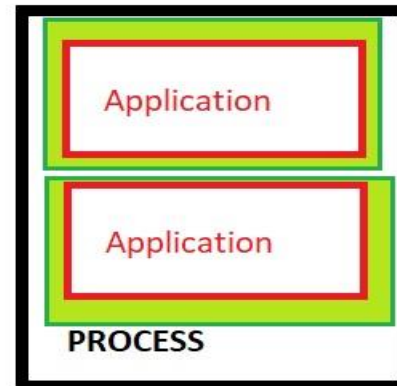
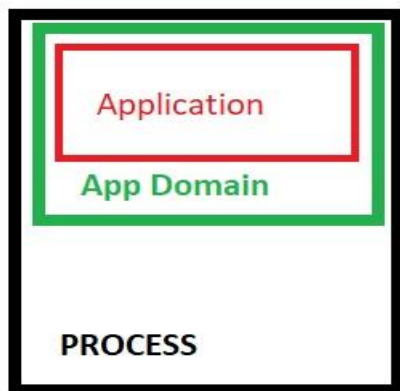
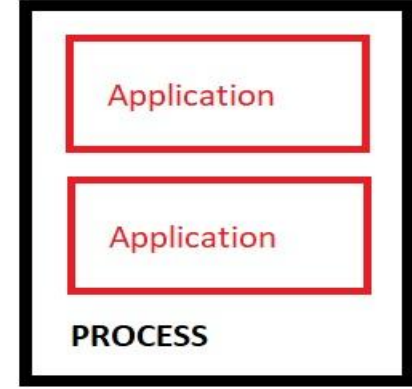
Thread Management

Security Management

Debugging


Exception Handling

App Domains



So what is the .Net Framework???

- .Net Base Classes
- +
- CLR
- +
- Utilities



And what would you need to run
your code on other platforms???

Difference between .Net Framework, .Net Core, Mono and Xamarin

- .Net Framework used for Windows platform mainly
- Mono used for Linux
- Xamarin used for Mobile platforms(Android, iOS and Windows)
- .Net Core used for all platforms

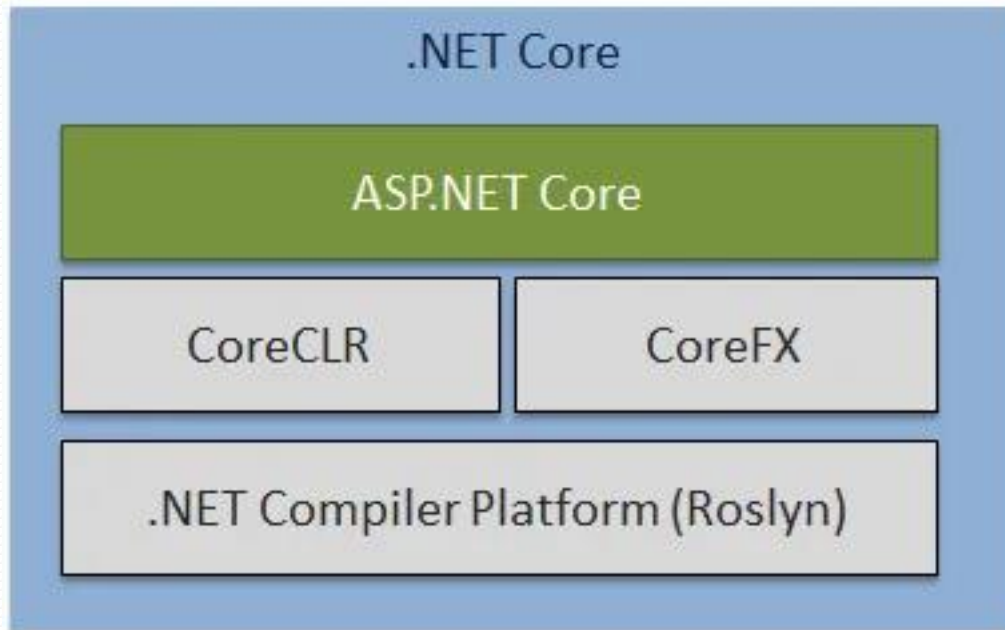
What is .NET Standard?

- There are various implementations of .NET. Each implementation allows .NET code to execute in different places—Linux, macOS, Windows, iOS, Android, and many more. .NET Standard is a formal specification of the APIs that are common across all these .NET implementations.
- .NET Standard allows libraries to build against the agreed on set of common APIs, ensuring they can be used in any .NET application—mobile, desktop, IoT, web, or anywhere you write .NET code.
- .Net Standard 2.0 specification targets all the latest versions

.Net Core Key Features

- Open-source
- Cross-platform
- Lightweight
- Extensible

.Net Core



CoreCLR

- CoreCLR is the runtime for .NET Core. It includes the garbage collector, JIT compiler, primitive data types and low-level classes.

CoreFX

- CoreFX is platform-neutral code that is shared across all platforms.
- CoreFX is the foundational class libraries for .NET Core. It includes types for collections, file systems, console, JSON, XML, async and many others.

Roslyn

- Roslyn is a set of open-source compilers and code analysis APIs for C# and Visual Basic (VB.NET) languages from Microsoft.
- .NET Compiler Platform is the technical name popularly called Roslyn

.Net Framework/Core History

- Net Framework 1 – Feb 2002
- .Net Framework 2 – Oct 2005
- .Net Framework 3 – Nov 2006
- .Net Framework 3.5 – Nov 2007
- .Net Framework 4 – Apr 2010
- .Net Framework 4.8 – Aug 2022
- .Net Core 1 – June 2016
- .Net Core 2 – Aug 2017
- .Net Core 3 – Sep 2019
- .Net 5 – Nov 2020
- .Net 6 – Nov 2021
- .Net 7 – Nov 2022
- .Net 8 – Nov 2023 (proposed)

.Net Core

- Upto Version 2.2 only supports ASP.NET MVC and Web Apis
- Version 3+ supports Winforms and WPF also
- Course covers version 6/7 (.Net 7)

What will we be working with?

- Visual Studio 2022 Community Edition- .Net 7
- C#
- Console Apps/Class Library
- Asp.Net Core MVC
- Web Api

Managed vs Unmanaged code

- Managed code is run by CLR (All .net code)
- Unmanaged code is not run by CLR
 - Egs Windows DLLs (PInvoke) and COM Apps (using COM Interop)

Assembly Structure

MyAssembly.dll

Assembly manifest

Type metadata

MSIL code

Resources

Assembly Manifest

Contains...

- Assembly Name
- Version Number
- Culture
- Strong Name
- List of files contained
- References
- Type Reference information



Can't wait to start coding 😊