2021/2022(2)
IF184605 Framework-Based Programming

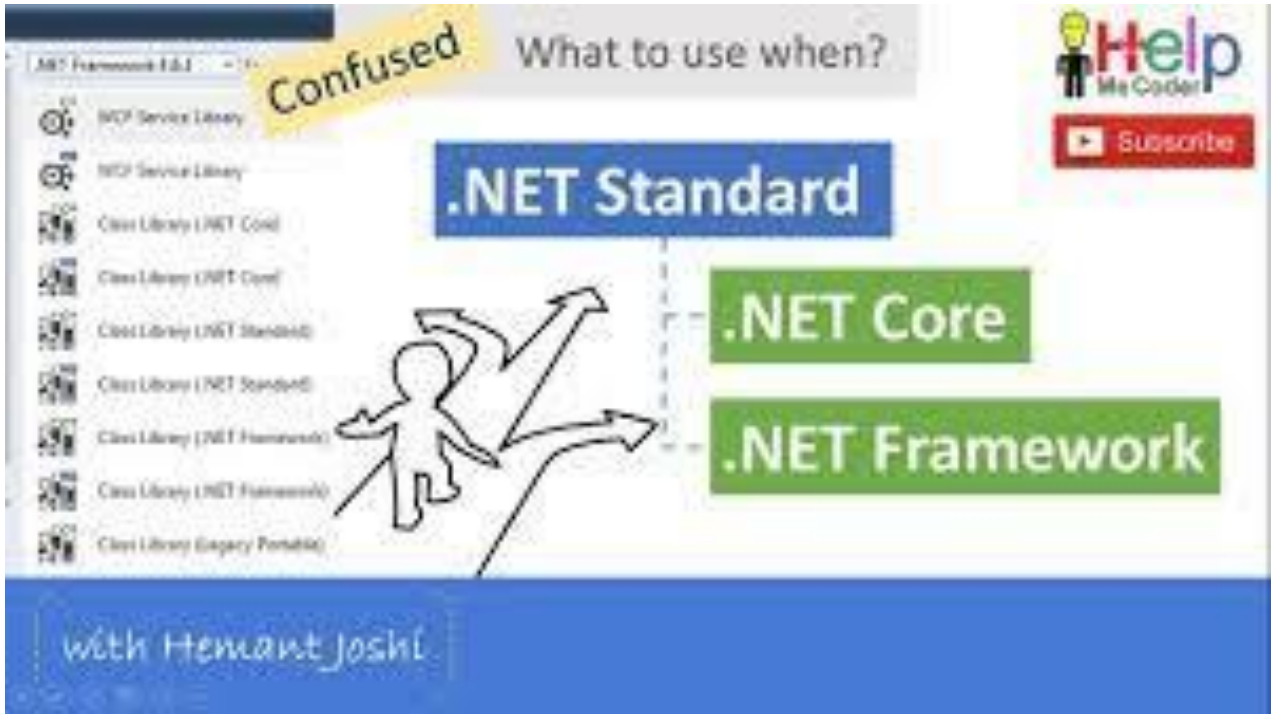
Lecture #3b

.NET Framework & .NET Core vs .NET Standard

Misbakhul Munir IRFAN SUBAKTI
司馬伊凡
Мисбакхул Мунир Ирфан Субакти

.NET: The comparison

- .NET Framework & .NET Core vs .NET Standard
  - Ref: code-maze.com



.NET Framework: The importance

- .NET Framework is the original implementation of .NET which was developed by Microsoft in the early 2000s to build Web and Desktop applications for Windows. It allows us to write applications in C#, Visual Basic, and F#.
- It is an execution environment that provides a variety of services to its running applications and also an extensive class library to write different kinds of applications.
- .NET Framework consists of two major components:
  1. **Common language runtime (CLR)** – which handles the execution of applications
  2. **Base Class Library (BCL) –** which provides a library of tested and reusable code that developers can use in their applications
- The code written in any .NET supported language is compiled into a special language called Intermediate Language (IL) and stored in assembly files with a .dll or .exe file extension. When an application runs, the CLR takes the assembly and turns it into machine code through a process called just-in-time (JIT) compilation.
- The .NET Framework provides many services to its running applications. It provides memory management, a common type system, and also language interoperability.

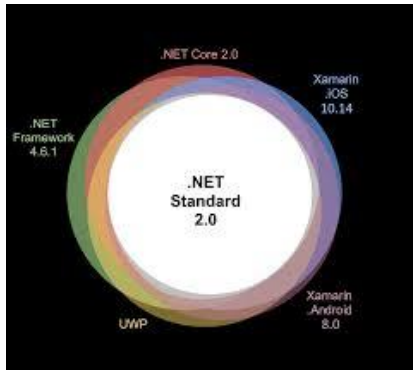
.NET Core: The rise of

- Due to the rise of different devices and cloud computing, in the past few years, software development trends have changed quite rapidly. Along with Windows, other operating systems usage has also increased.
- A large number of software and services are now being developed for different platforms. Because of this trend, Microsoft started developing a new framework that can fulfil the requirements of the current as well as future software development needs.
- Hence, they developed a new framework called .NET Core. **It is an open-source and cross-platform implementation of .NET.** It shares many of its characteristics with .NET Framework but there are differences as well.
- All aspects of .NET Core are open-source including class libraries, runtime, compilers, languages as well as application frameworks. .NET Core also supports C#, Visual Basic, and F#. It can run the application code with the same behaviour on multiple architectures, including x64, x86, and ARM. It has a flexible deployment model in which it can be included in the application or installed side-by-side (user-wide or system-wide).
- .NET Core can also be used with Docker. It also includes command-line tools that can be used during local development and most importantly in continuous integration.
- Currently, all the innovations and major enhancements are being made in .NET Core. However, the .NET Framework is also being developed but at a much slower pace.

ASP.NET Core .NET Core

docker

.NET Standard: What is it?



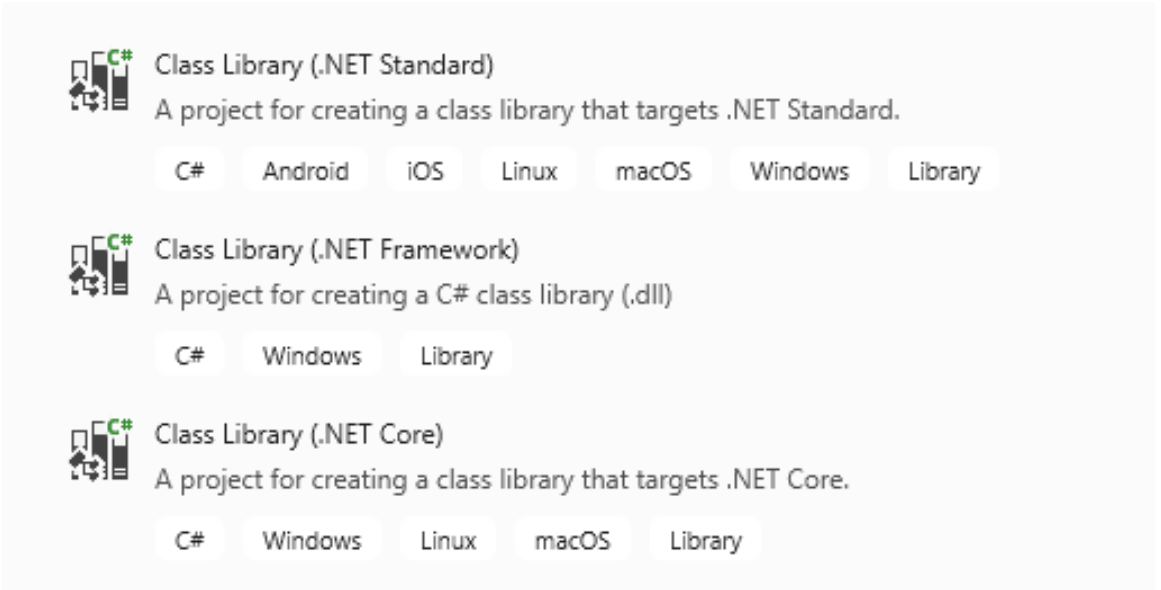
- .NET Standard is a specification (not an implementation of .NET) that defines the set of APIs that all .NET implementations must provide. It addresses the code sharing problem for .NET developers across all platforms by bringing APIs across different environments.
- We can think of it as another .NET Framework, except that we use it to develop class libraries only. .NET Standard is a successor of the portable class library.

.NET Standard: The confusion about

- Newbies to .NET who try to understand the .NET Standard can easily get confused in the beginning.
- And here's one of the reasons why that might be.
- When we open up Visual Studio 2019 and go to the **Create a new project** window, we'll see three types of class library projects; Class Library (.NET Standard), Class Library (.NET Framework), and Class Library (.NET Core).
- If we carefully read the description which is written under Class Library (.NET Standard) and Class Library (.NET Core), it says that target platforms are .NET Standard and .NET Core respectively. This might give us a feeling that the .NET Standard is another framework or implementation of .NET:

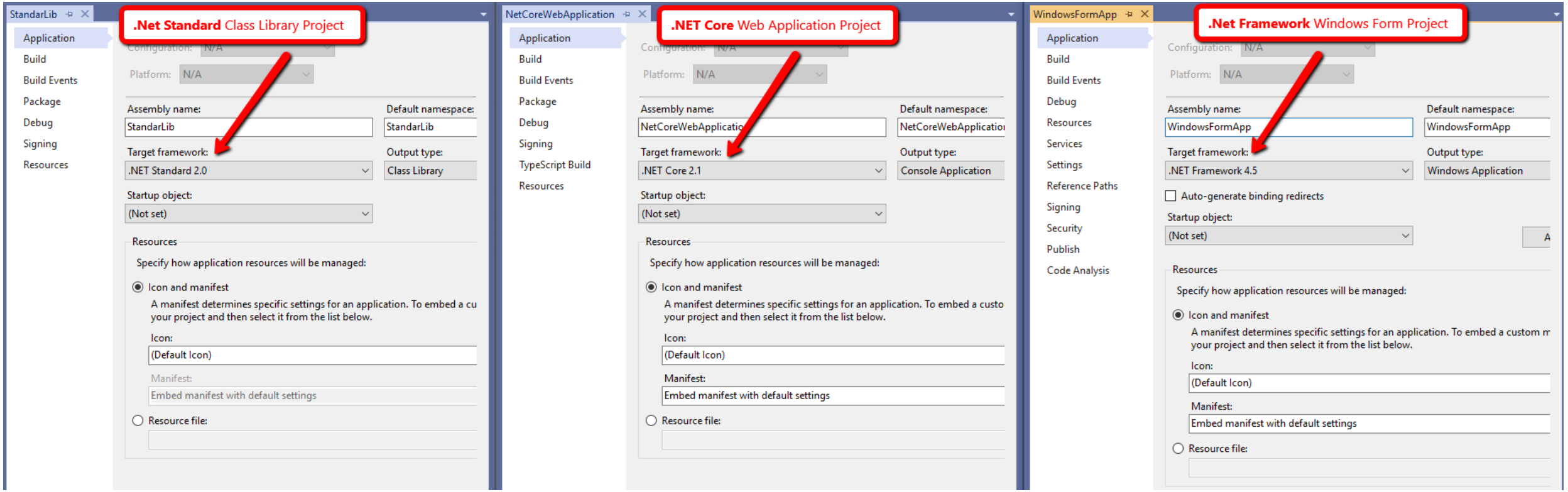
.NET Standard: The confusion about (cont'd)

- Furthermore, let's create three projects for each platform:
  1. **StandarLib** is a Class Library project type that targets the .NET Standard
  2. **NetCoreWebApplication** is a Web Application type project which targets the .NET Core
  3. **WindowFormApp** is a Windows Form type project which targets the .NET Framework
- If we take a look at the properties of all three projects, we can see each property window has a field named **Target framework;** which is right for the .NET Framework and .NET Core projects.



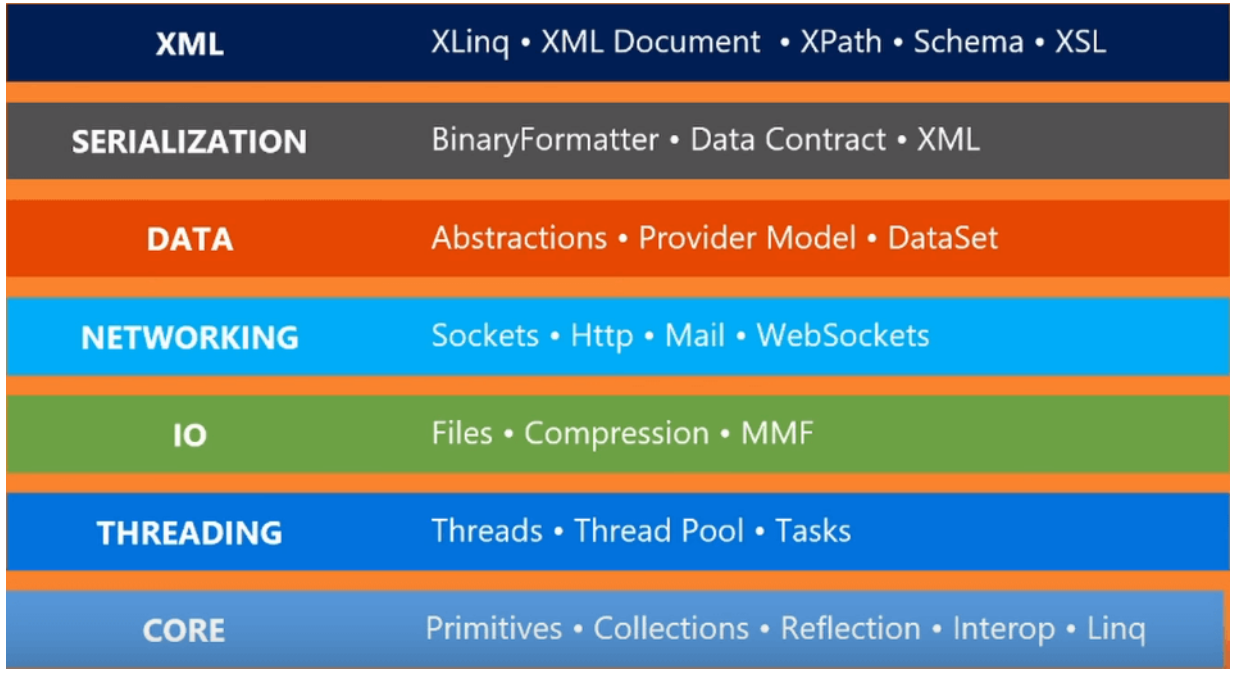
.NET Standard: The confusion about (cont'd)

- But for the .NET Standard project, it might be misleading, because, newbies might think of it as a framework and not a specification:



.NET Standard: The confusion about (cont'd)

- The official documentation introduces .NET Standard as a specification and tries to distinguish it from the framework but this distinction doesn't seem to be reflected in the development tools.



.NET Framework vs .NET Core

- Now we understand that .NET Framework and .NET Core are two different .NET implementations,

.NET Framework

1. The .NET Framework is the first implementation of .NET which works on Windows only
2. Its source code is public but Microsoft doesn't accept third party contributions for it
3. It has a very rich desktop top development framework for windows which include Windows Forms and WPF
4. A huge third-party packages library is also available for it
5. It doesn't support the in-app deployment model
6. Although it can be used with a docker container, its image size is large and can only be deployed on Windows containers

.NET Core

1. .NET Core is the latest implementation of .NET which runs on Windows, Linux, and macOS
2. Its open-source and Microsoft accepts third party contributions to .NET Core
3. It supports desktop frameworks like Windows Forms and WPF from version 3.0
4. The .NET Core also has support for a large number of third party packages as well but still, it doesn't compete with .NET Framework in this area
5. It does support an in-app deployment model
6. It is the best choice to work with docker containers

.NET Framework & .NET Core vs .NET Standard

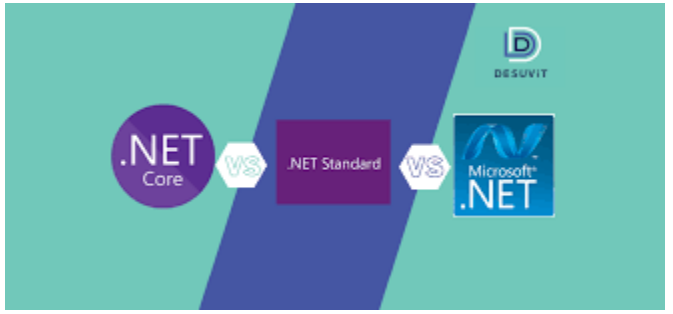
- Because .NET Framework and .NET Core are .NET implementations, therefore, we can compare them together against the .NET Standard.

.NET Framework and .NET Core

1. The .NET Framework and .NET Core are implementations of .NET
2. Both frameworks have a runtime that manages the execution of applications
3. The base class library is also a part of both frameworks
4. We can create different types of projects in either framework

.NET Standard

1. The .NET Standard is a specification and not a .NET implementation
2. It specifies a set of APIs that all the .NET implementations have to implement
3. We can create only class library type projects with it



Target platform & version: Decision

We should use the .NET Core:

1. While developing applications for cross-platform
2. For the development of microservices
3. When we want to use Docker containers
4. To develop high-performance and scalable systems

Use the .NET Framework when:

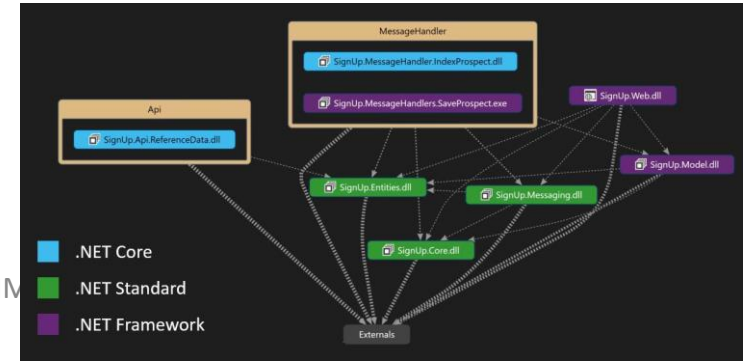
1. We want to target only Windows
2. Our application uses some third party packages which are not supported by .NET Core
3. The application uses .NET technologies that are not available for .NET Core

.NET Standard should be the choice when:

1. We want to share our common code across different .NET implementations

Target platform & version: Decision (cont'd)

- Once we choose the right platform and project type for our application, the next step is to use the correct version. Let's suppose that we have to target both, the .NET Framework and .NET Core and we also want to share the common code across both of these platforms.
- Of course, the .NET Standard class library will be our choice here. Choosing the correct version for this scenario would have been quite challenging, but fortunately, Microsoft has provided us with a nice helpful chart:



Target platform & version: Decision (cont'd)

The following table lists the **minimum** platform versions that support each .NET Standard version. That means that later versions of a listed platform also support the corresponding .NET Standard version. For example, .NET Core 2.2 supports .NET Standard 2.0 and earlier.

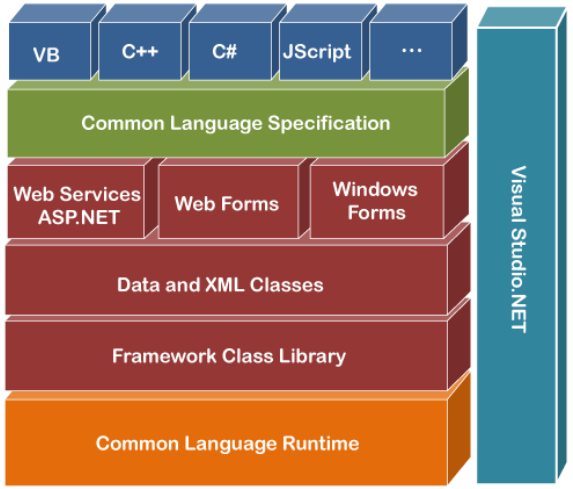
.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework [1]	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1 [2]	4.6.1 [2]	4.6.1 [2]
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Unity	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1

1 The versions listed for .NET Framework apply to .NET Core 2.0 SDK and later versions of the tooling. Older versions used a different mapping for .NET Standard 1.5 and higher. You can download tooling for .NET Core tools for Visual Studio 2015 if you cannot upgrade to Visual Studio 2017.

2 The versions listed here represent the rules that NuGet uses to determine whether a given .NET Standard library is applicable. While NuGet considers .NET Framework 4.6.1 as supporting .NET Standard 1.5 through 2.0, there are several issues with consuming .NET Standard libraries that were built for those versions from .NET Framework 4.6.1 projects. For .NET Framework projects that need to use such libraries, we recommend that you upgrade the project to target .NET Framework 4.7.2 or higher.

Target platform & version: Decision (cont'd)

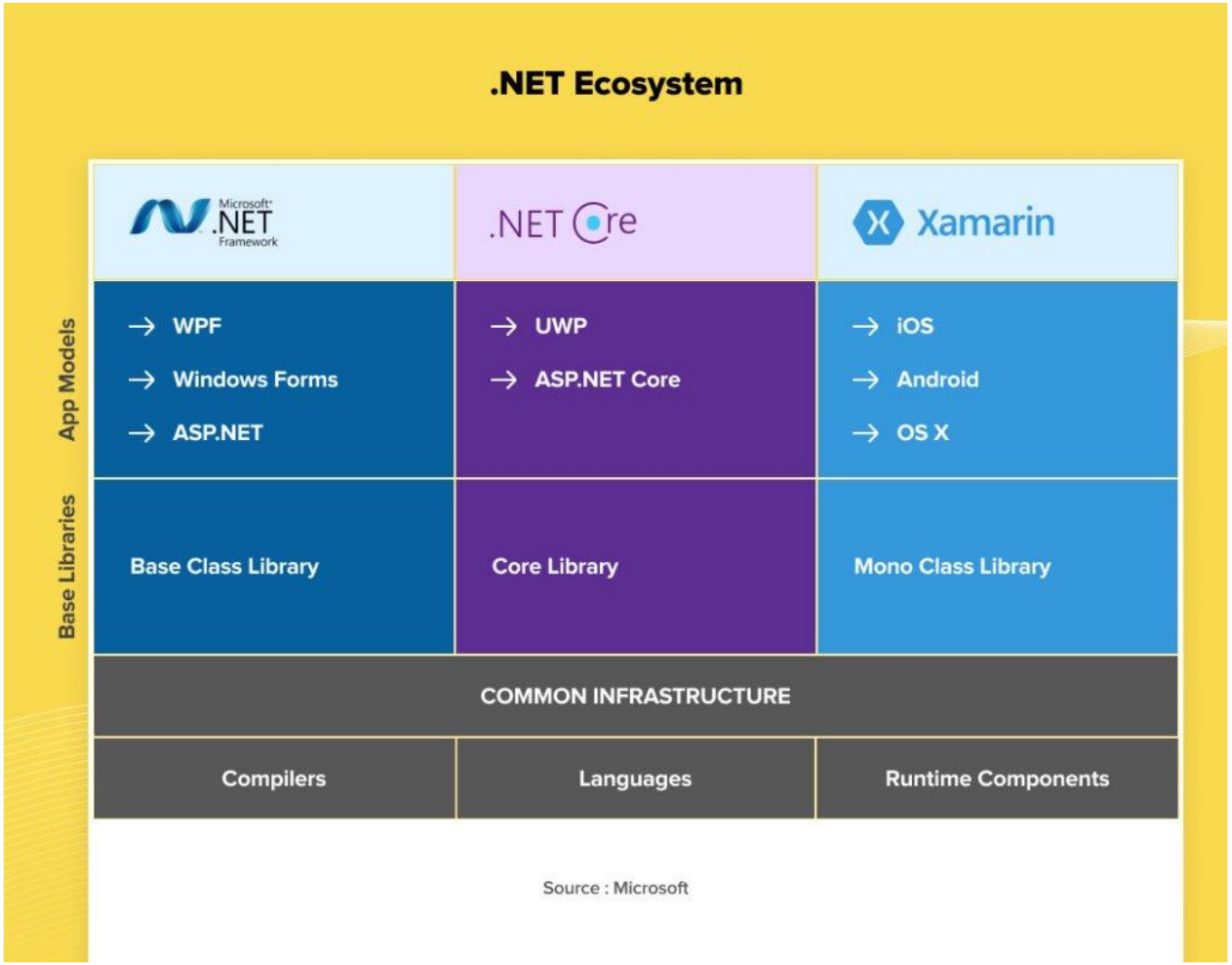
- As shown in this chart, the first row mentions the different versions of .NET Standard and all other rows mention the different versions of various platforms supported by it.
- For example, if we choose the 1.6 version of .NET Standard for our class library project and reference that library in .NET Core and .NET Framework projects then according to this chart we must choose the 1.0 version for .NET Core project and 4.6.1 version of .NET Framework project.
- The same goes for all other versions.

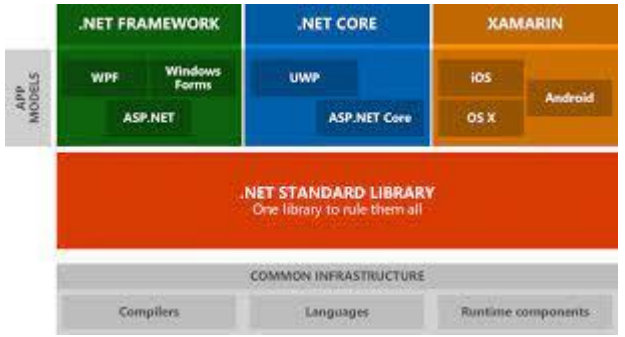


Xamarin

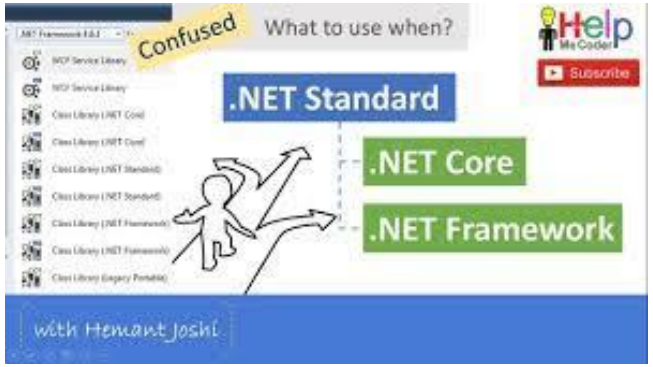
- Xamarin is a Microsoft-owned San Francisco-based software company founded in May 2011 by the engineers that created Mono, Xamarin.Android and Xamarin.iOS, which are cross-platform implementations of the Common Language Infrastructure and Common Language Specifications.
- Xamarin is an open-source platform for building modern and performant applications for iOS, Android, and Windows with .NET. Xamarin is an abstraction layer that manages communication of shared code with underlying platform code.
- In May 2020, Microsoft announced that *Xamarin.Forms*, a major component of its mobile app development framework, would be *deprecated* in November 2021 in favour of a new .NET based product called MAUI - Multiform App User Interface.
- .NET MAUI Essentials arrived in fall 2021. Alongside more details of MAUI, Microsoft says it will end updates to the Xamarin mobile app development platform in November 2022. Xamarin has been the Microsoft technology for developers to use if they want to develop apps for iOS and Android using C#.

.NET Framework | .NET Core | Xamarin





Conclusion



- The confusion among the .NET Framework, .NET Core, and .NET Standard has been removed by clarifying these concepts and comparing them with each other.
- Choosing the target platform and its version depending on our needs.
- For developing apps for iOS and Android using C#, there is Xamarin → .NET MAUI