What is .NET Standard?

.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.

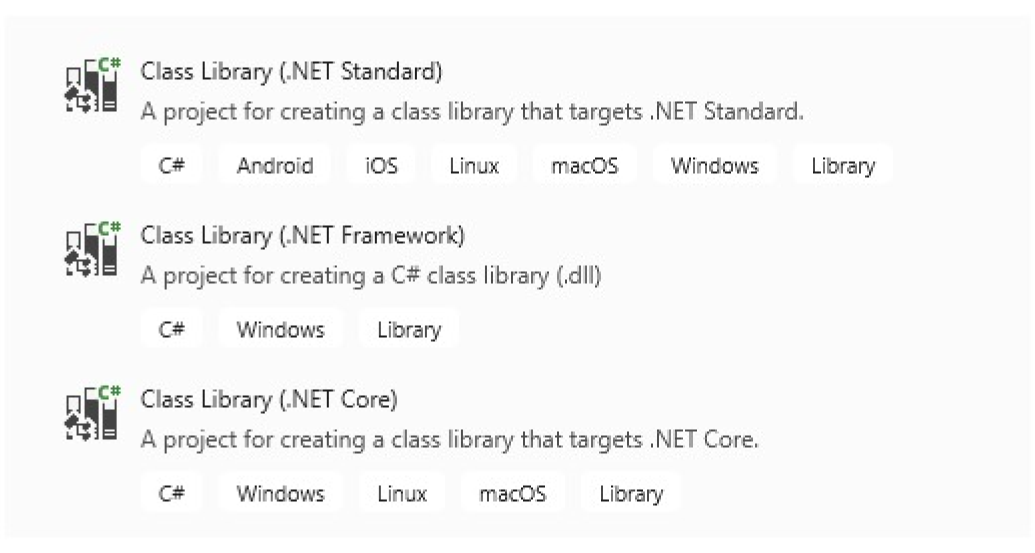
Confusion about .NET Standard

Newcomers 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:



Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
C# Android iOS Linux macOS Windows Library

Class Library (.NET Framework)
A project for creating a C# class library (.dll)
C# Windows Library

Class Library (.NET Core)
A project for creating a class library that targets .NET Core.
C# Windows Linux macOS Library

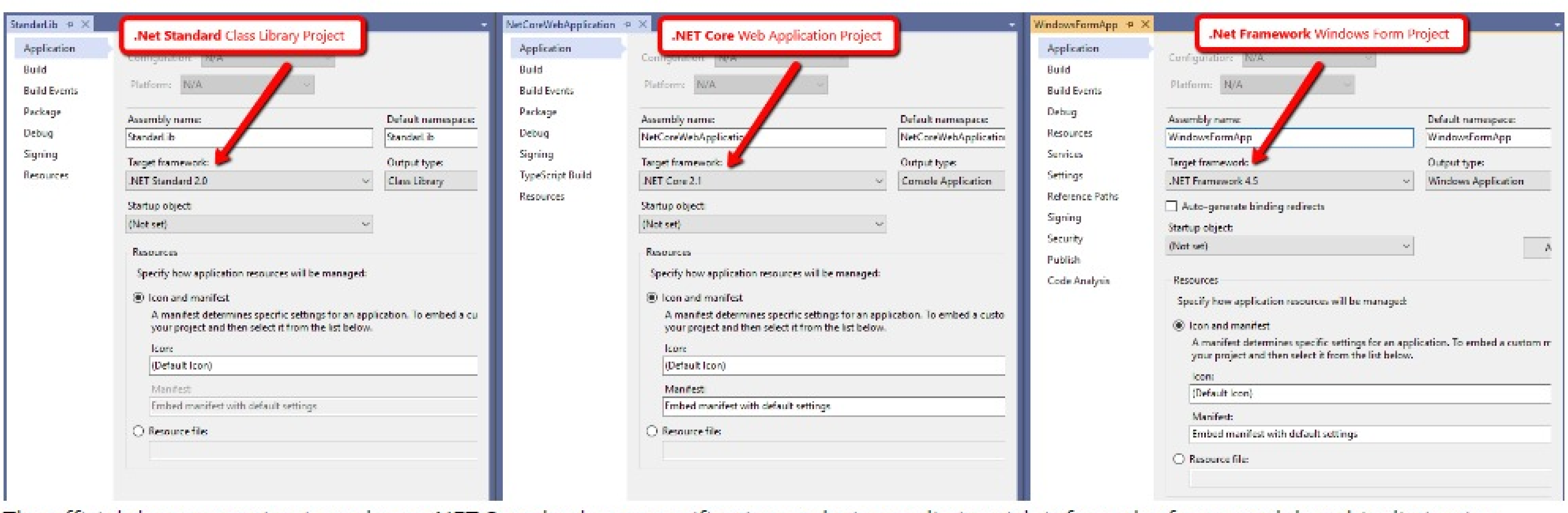
Wanna join Code Maze Team, help us produce more awesome .NET/C# content and **get paid?** >> JOIN US! <<

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.

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



StandarLib — Assembly name: StandarLib; Default namespace: StandarLib; Target framework: .NET Standard 2.0; Output type: Class Library; Startup object: (Not set); Icon: (Default Icon); Manifest: Embed manifest with default settings

NetCoreWebApplication — Assembly name: NetCoreWebApplication; Default namespace: NetCoreWebApplication; Target framework: .NET Core 2.1; Output type: Console Application; Startup object: (Not set); Icon: (Default Icon); Manifest: Embed manifest with default settings

WindowsFormApp — Assembly name: WindowsFormApp; Default namespace: WindowsFormApp; Target framework: .NET Framework 4.5; Output type: Windows Application; Startup object: (Not set); Icon: (Default Icon); Manifest: Embed manifest with default settings

Resources — Specify how application resources will be managed: Icon and manifest. A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

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, therefore, both can be compared.

.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

Wanna join Code Maze Team, help us produce more awesome .NET/C# content and **get paid?** >> JOIN US! <<

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 in-app deployment model
6. It is the best choice to work with docker containers

.NET Framework and .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 runtime which 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

Deciding Target Platform and Version

Let's discuss what point should we keep in mind while choosing the target platform and their versions.

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

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.