.NET Standard vs .NET Core

207

55

I have read about the difference between .NET Standard and .NET Core, but I really don't know what the difference is, or when to choose a .NET Standard library project and when to choose a .NET Core library project.

I have read that .NET Standard is to ensure that a set of APIs are always available, no matter the platform used (as long as that platform is compatible with the .NET Standard version that I have chosen). If I'm not mistaken, this means that I can create a class library of .NET Standard and then use it on any platform that is compatible with the .NET Standard version that I have chosen.

With .NET Core, I have read that it is intended for cross-platform use too, so if I choose a .NET Core library it seems that I can use it on many platforms too, just like .NET Standard.

So at the end, I don't see the difference. When should I use which? What is the difference between them?

.net .net-core .net-standard

edited Jun 14 '17 at 19:26
TriskalJM
1,827 1 13 18

asked May 20 '17 at 11:38
Álvaro García
6,773 19 65 120

28 In code terms: .net standard = interface, .net core = class; if you code against the class you nay get more methods (etc), but you are restricted to that concrete type (and descendants); if you use the interface you *might* get a smaller surface, but it will work against arbitrary implementations... as long as those implementations do what is expected :) yes, .net core targets multiple platforms, but there are *other* implementations of .net standard – Marc Gravell ♦ May 20 '17 at 11:59

7 .NETStandard is a replacement for PCL. A Portable Class Library helped you write a library that could run on more than one platform (phone, desktop, store, browser, xbox, etc). It did not scale very well, suffering badly from the n! problem, so they abandoned it. .NETCore is just the first framework they got done, it was the easiest, the rest has to catch up. Do keep in mind that this is very much work-in-progress, big changes ahead with .NETStandard v2.0. The standard to rule them all, for now :) – Hans Passant May 20 '17 at 12:45

Please don't add extra questions to an existing one. Your question in the edit is separate to this. – Jon Skeet May 21 '17 at 10:02

1 @JonSkeet Then I should open a new question? Thanks for the advise. because at first I have opened a new quesetion about .net Core multi target

and I have been downvoted because they said it is a duplicate question. – Álvaro García May 21 '17 at 11:03

6 Answers

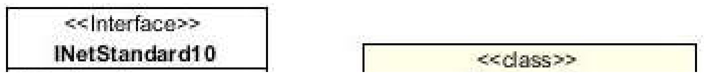
165

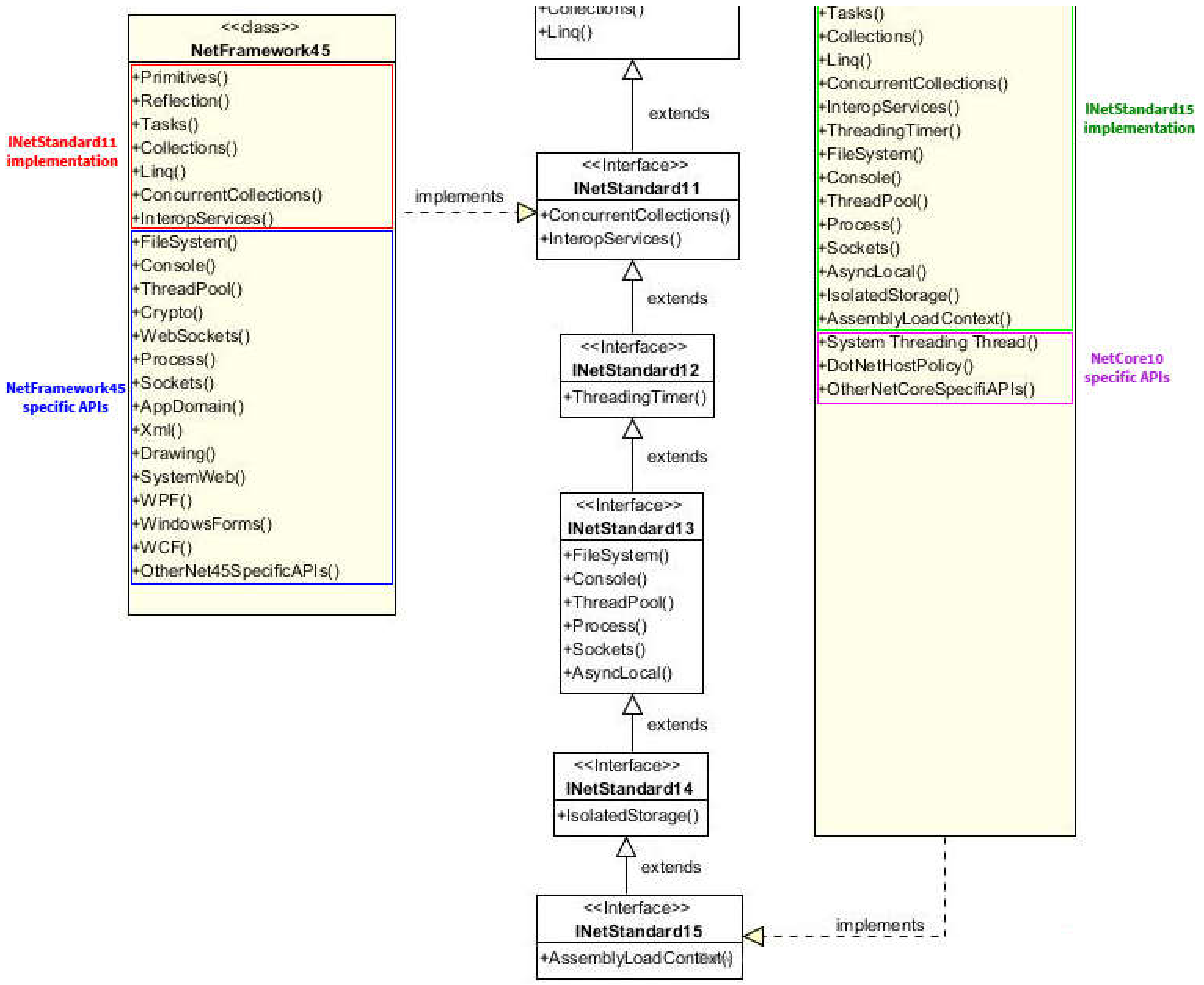
I will try to further clarify your doubts and extend Jon Skeet answer.

.NET Standard is a *specification*, so a library compiled for a specific .NET Standard version can be used in different .NET Standard implementations.

As said in my other comment, a good analogy for the relationship between .NET Standard and other .NET Standard Implementations (.NET Core, .NET Framework, etc) is this gist by David Fowler: .NET Standard versions are `Interfaces` , while frameworks are implementations of those interfaces.

This simplified diagram may help to understand this relationship:





Anything targetting `NetCore10` has access to `INetStandard15` APIs *and* `NetCore10` **specific** APIs (such as `DotNetHostPolicy`).

If you, instead, need access only to `INetStandard15` APIs (and target that specification instead of a concrete framework) your library may be used by **any** framework which implements it (`NetCore10` , `NetFramework462` , etc.)

Note: in the original analogy David Fowler used interfaces for both .NET Standard versions and frameworks implementations. I believe that using interfaces and classes is, instead, more intuitive and better represents the relationship between specifications and concrete implementations.

edited Sep 15 '17 at 22:16

answered May 20 '17 at 17:07
Federico Dipuma
12.1k 3 28 42

2 Thanks so much for this. But I have a doubt. If .net standard is an interface and can be implemented for example with .net framework and .net Core, when I create a .net standard class library and I use this library in another project, what implementation is using, net framework or .net Core? – Álvaro García May 20 '17 at 18:10

6 It will use the implementation of the compiled application (whatever it is). If you compile a NET core app then it will use NET core libraries (that are an implementation of NET standard) – Federico Dipuma May 20 '17 at 18:19

3 That diagram is a fantastic help in illustrating the core/std/framework relationship. – jasper Aug 10 '17 at 17:08

So if I create a net461 console app and target a netstandard2.0 library, nothing from that standard lib resolves in the console app. So... ?? – Sinaesthetic Aug 30 '17 at 18:45

1 A picture is worth a thousand words – Nikaas Oct 4 '18 at 12:20

164

.NET Core is *an implementation* of .NET Standard. It's available on multiple operating systems, but that's not the same thing - there are other implementations of .NET Standard as well.

So if you create a .NET Core library, it will have access to things that are implemented in .NET Core, but *aren't* part of .NET Standard, and your library won't be compatible with *other* implementations of .NET Standard, such as Xamarin, Tizen, full .NET desktop framework etc.

In short: to achieve maximum portability, make your library target .NET Standard.

answered May 20 '17 at 11:41
Jon Skeet
1113k 708 8113

5 @AlvaroGarcia: What do you mean by "it" should be compatible? .NET Core 1.0? Not necessarily - because .NET Core 1.0 can still include *extra* things. That entry means that if you target .NET Standard 1.6, you can run the code under both Mono 4.6 *and* .NET Core 1.0. – Jon Skeet May 20 '17 at 11:47

4 You can't run a .NET Core assembly on anything other than .NET Core (CoreCLR & CoreFX). You can run a .NET Standard assembly on any framework that meets the contractual obligations for the relevant Standard (1.3, 1.6, 2.0, etc). – Mark Rendle May 20 '17 at 11:55

13 As an analogy, try imagine .NET Standard as an Interface (e.g. `INetStandard16`). .NET Core 1.0 and Mono 4.6 both implement `INetStandard16` . You cannot convert .Net Core 1.0 to Mono 4.6 (and vice-versa), but anything that uses `INetStandard16` will work on both. (credits to David Fowler) – Federico Dipuma May 20 '17 at 11:56

4 This surprised me. The names seem backwards. You would think something named "core" would be the more minimal of the two... – jpmc26 May 21 '17 at 5:39

2 I'm thinking entropy has begun. – JustJohn May 25 '17 at 21:09

4

.NET Core Class library is basically subset of .NET Framework library, which just contains less APIs. Sticking to .NET Core Class library makes difficult to share code between runtimes. This code might not work for different runtime (Mono for Xamarin), because it doesn't have the API that you need. To solve this there is **.NET Standard, which is just set of specification that tells you which APIs you can use**. Main purpose of .NET Standard is to share code between runtimes. And important that this specification implemented by all runtimes.(.NET Framework, **.NET Core** and Mono for Xamarin).

So if you sure that you will use your library only for .NET Core projects, you can ignore .NET Standard, but if have even tiny chance that your code will be used by .NET Framework or Mono for Xamarin then better stick to .NET Standard

Also note that higher version of .NET Standard contain more APIs, but lower version supported by more platforms. **Therefore if you create .NET Standard library that you want to share between runtimes then target the lowest version you can**, which help you reach maximum amounts of platforms. For example, if you want to run on .NET Framework 4.5 and .NET Core 1.0, the highest .NET Standard version you can use is .NET Standard 1.1. This this great table from documentation for more info about it.

PS: Also if you want to convert you library to .NET Standard, .NET Portability Analyzer could help you with that.

edited Aug 22 '17 at 12:18

answered Aug 16 '17 at 20:12
user2771704
4,093 5 27 36

> **.NET Core** is an implementation of the .NET Standard that’s optimized for building console applications, Web apps and cloud services using ASP.NET Core. Its SDK comes with a powerful tooling that in addition to Visual Studio development supports a full command line-based development workflow. You can learn more about them at aka.ms/netstandardfaq and aka.ms/netcore.

The above, together with a very clear explanation of most of the stuff discussed in this question can be found in the following extremely helpful article by Microsoft (MSDN - September 2017): .NET Standard - Demystifying .NET Core and .NET Standard

answered Oct 5 '17 at 0:17
steliosalex
41 3

0

Did you mean .NET Framework? Because .NET standard is an implementations, such as .NET Framework, .NET Core and Xamarin.

I love .NET Core because we can host it on Linux (use nginx in my experience). It's different than .NET framework which is you can only host on IIS. You can consider about hosting budget in this case (Because windows server is expensive for me).

In the **development environment perspective**, .Net core is lightweight. So, you can use VSCode, Sublime, for IDE (not only visual studio).

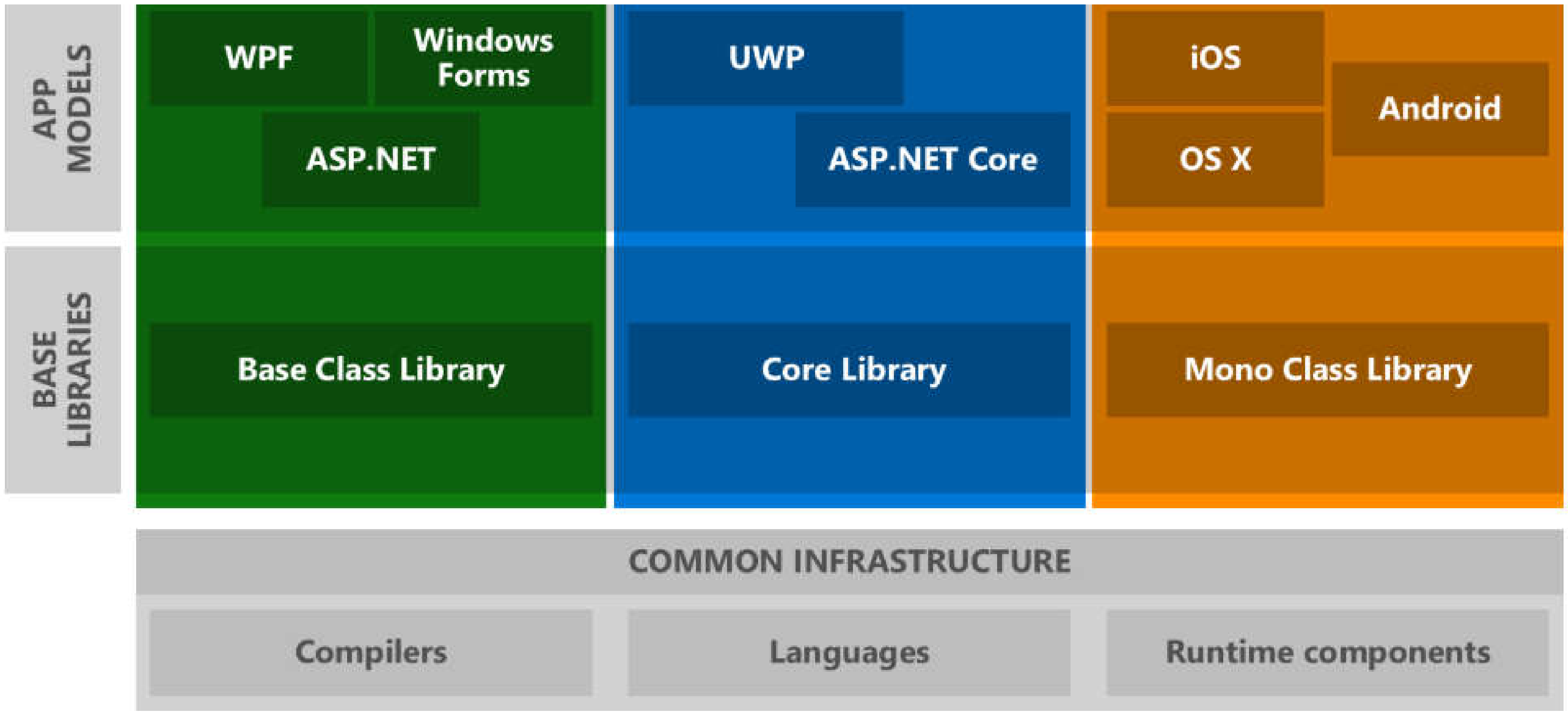
answered Feb 17 '18 at 2:39
Fityan Aula
77 9

0

.NET Standard is a specification of .NET APIs intended to be available on .NET implementations. This enables to define uniform set of BCL APIs for all .NET implementations.

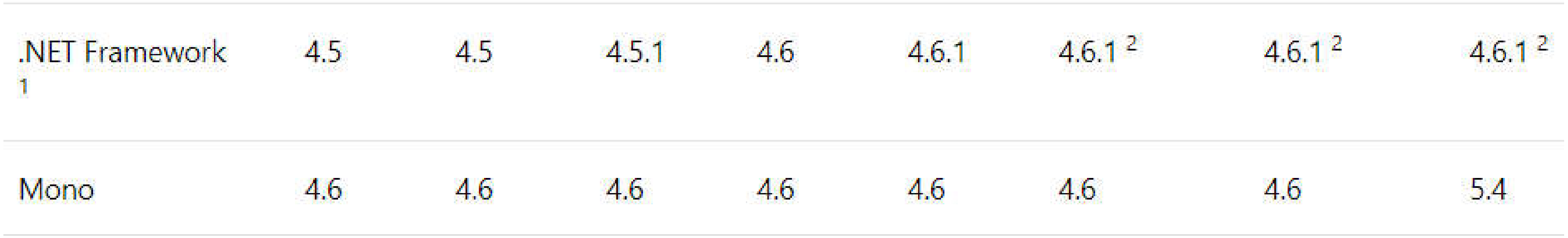
.NET Core is one such implementation of .NET Standard. .NET Framework is another implementation of .NET Standard.

Image from .NET Blog



Federicos answer gives you a graphical overview of how each framework evolve with versions. Take a look at below diagram from Microsoft Docs.

.NET Standard
1.0
1.1
1.2
1.3
1.4
1.5
1.6
2.0

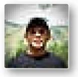


.NET Framework [1]	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1 [2]	4.6.1 [2]	4.6.1 [2]
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4

> Targeting .NET Standard increases your platform support whereas targeting a particular .NET platform such as .NET Core (or .NET Framework) will allow you to use all the platform features for that platform.

edited Apr 27 at 10:25

answered Apr 24 at 7:31



Nipuna
2,726 7 41 72