.NET Core

.NET Core is a free, cross-platform, open source implementation of the managed framework. It supports four types of applications: console, ASP.NET Core, cloud, and Universal Windows Platform (UWP). Windows Forms and Windows Presentation Foundation(WPF) are not part of .NET Core. Technically, .NET Core only supports console applications. ASP.NET Core and UWP are application models built on top of .NET Core. Unlike the .NET Framework, .NET Core is not considered a Windows component.

Therefore, updates come as NuGet packages, not through Windows Update. Since the .NET Core runtime is installed App-Local, and applications are updated through the package manager, applications can be associated with a particular .NET Core version and be updated individually.

.NET Standard

Each implementation of the managed framework has its own set of Base Class Libraries. The Base Class Library (BCL) contains classes such as exception handling, strings, XML, I/O, networking, and collections. .NET Standard is a specification for implementing the BCL. Since a .NET implementation is required to follow this standard, application developers will not have to worry about different versions of the BCL for each managed framework implementation.

Framework Class Libraries (FCL) such as WPF, WCF, and ASP.NET are not part of the BCL, and therefore are not included in .NET Standard.

The relationship between .NET Standard and a .NET implementation is the same as between the HTML specification and a browser. The second is an implementation of the first.

Hence, the .NET Framework, Xamarin, and .NET Core each implement .NET Standard for the BCL in their managed framework. Since the computer industry will continue to introduce new hardware and operating systems, there will be new managed frameworks for .NET. This standard allows application developers to know that there will be a consistent set of APIs that they can rely on.

Each .NET version has an associated version of the .NET Standard.

By providing consistent APIs, porting applications to different managed implementations, as well as providing tooling, is easier.

.NET Standard is defined as a single NuGet package because all .NET implementations are required to support it. Tooling becomes easier because the tools have a consistent set of APIs to use for a given version. You can also build a single library project for multiple .NET implementations.

You can also build .NET Standard wrappers for platform specific APIs.

Conclusions

.NET Standard is an API specification that defines, for a given version, what Base Class Libraries must be implemented.

.NET Core is a managed framework that is optimized for building console, cloud, ASP.NET Core, and UWP applications. It provides an implementation of .NET Standard for the Base Class Libraries.

.NET Standard is an API specification that defines, for a given version, what Base Class Libraries must be implemented.

.NET Core is a managed framework that is optimized for building console, cloud, ASP.NET Core, and UWP applications. It provides an implementation of .NET Standard for the Base Class Libraries.

1. Делаем DLL в .NET Standard:

```
using System;

namespace MyStandardLibrary
{
    ссылка: 1
    public static class MyLogger
    {
        ссылка: 1 | 1/1 пройдены
        public static string PrintGreeting(string name) => $"Hello, {name}, {DateTime.Now}";
    }
}
```

2. Пишем для нее юнит-тест:

```
namespace MyLibraryTest
{
    Ссылок: 0
    public class GreetingTest
    {
        [Fact]
        | Ссылок: 0
        public void Is_Correct_Greeting()
        {
            //arrange
            string name = "Denis";

            //act
            string expected = $"Hello, Denis, {DateTime.Now}";
            string actual = MyLogger.PrintGreeting(name);

            //assert
            Assert.Equal(expected, actual);
        }
    }
}
```

3. Теперь ее можно подключать в другие проекты (.NET Core, WPF):