XAM110

Intro to Cross-Platform Mobile Development

Download class materials from university.xamarin.com

Microsoft

Xamarin University

Objectives

1. Work with shared components
2. Share code using Shared Projects
3. Share code using Portable Class Libraries
4. Share code using .NET Standard libraries

Work with Shared Components

Xamarin University

Tasks

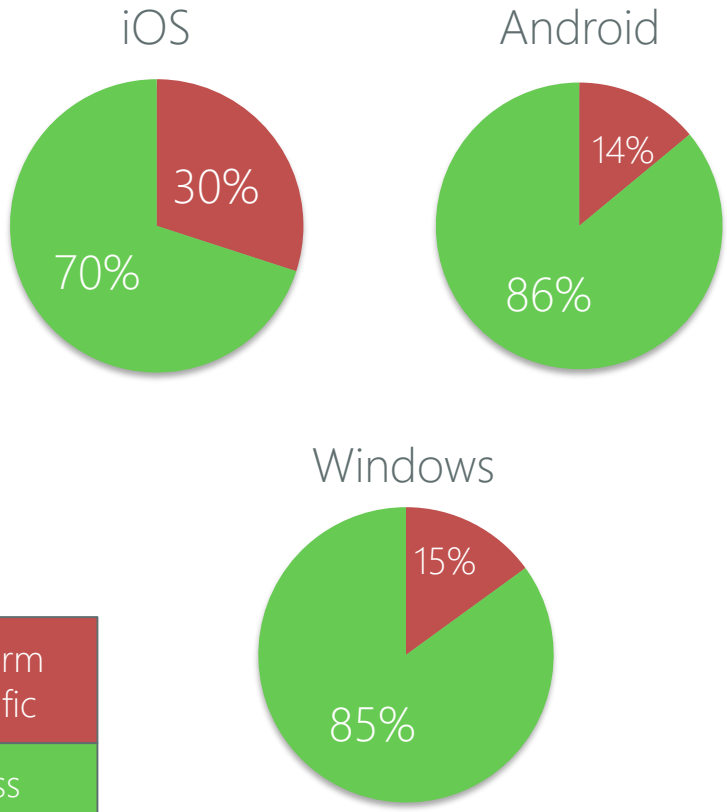
1. Add NuGet packages to your application
2. Add components to your application

Sharing code

- One of the main reasons to use Xamarin is the possibility of sharing a significant portion of your code across all your supported platforms

Sharable code

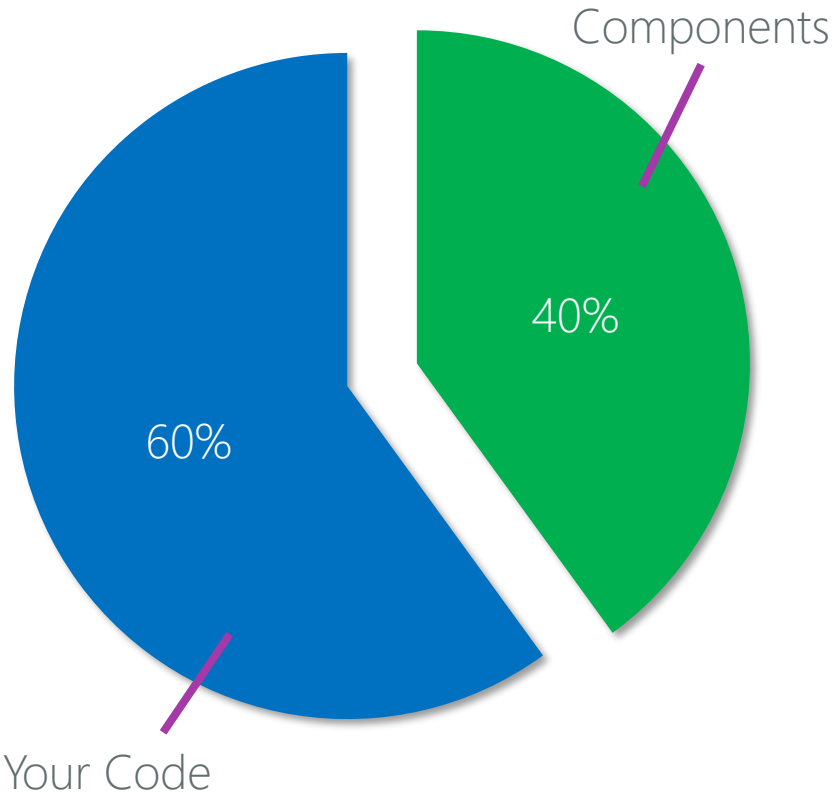
- Xamarin applications are native and therefore will *always* include some platform-specific code



statistics taken from iCircuit

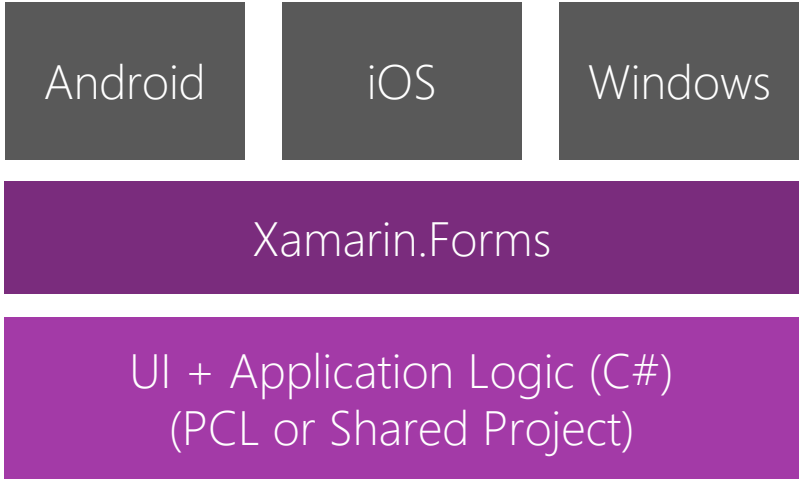
Sharable code

- Sharable code is split between reusable components and platform-independent code



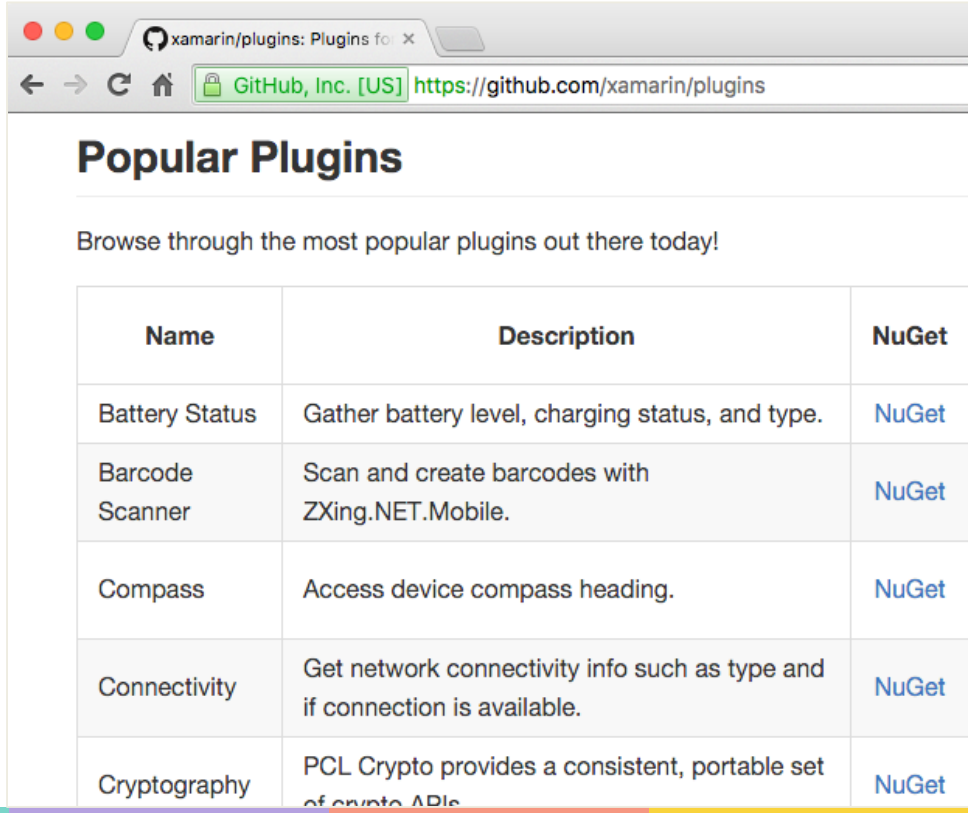
Xamarin.Forms

- Xamarin.Forms provides shared set of UI controls to design the user interface that ultimately render native UIs on iOS, Android and Windows



Other open-source plug-ins

- **github.com/xamarin/plugins** maintains a list of open-source components which you can use in your Xamarin based applications



Popular Plugins

Browse through the most popular plugins out there today!

Name	Description	NuGet
Battery Status	Gather battery level, charging status, and type.	NuGet
Barcode Scanner	Scan and create barcodes with ZXing.NET.Mobile.	NuGet
Compass	Access device compass heading.	NuGet
Connectivity	Get network connectivity info such as type and if connection is available.	NuGet
Cryptography	PCL Crypto provides a consistent, portable set of crypto APIs	NuGet

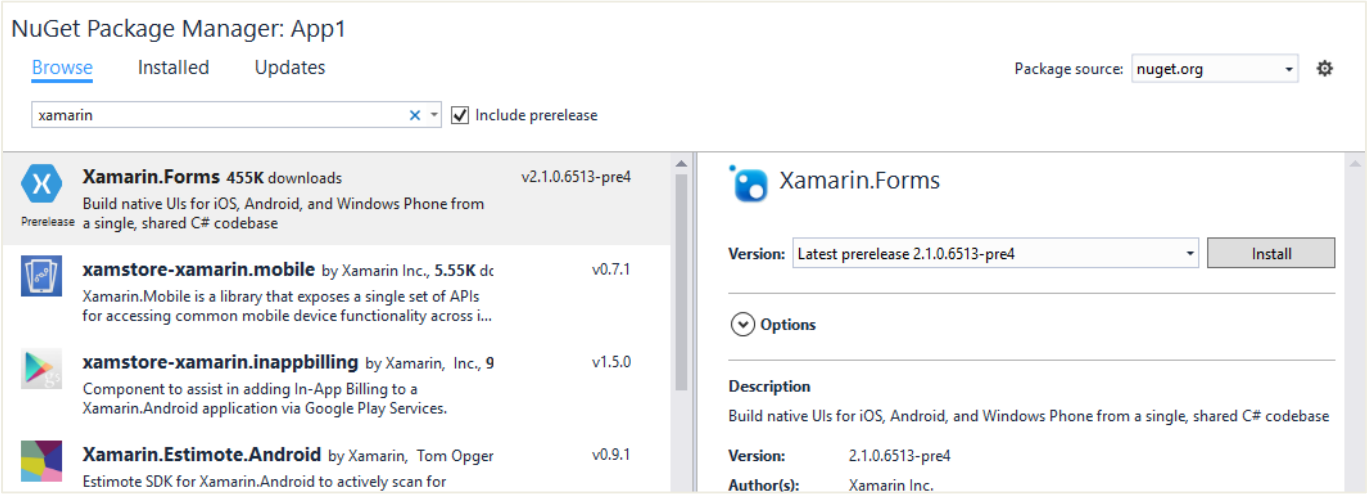
NuGet

- NuGet is a package manager for .NET that allows you to locate, install, update and remove shared components from within Visual Studio

www.nuget.org

Using NuGet in Visual Studio on Windows

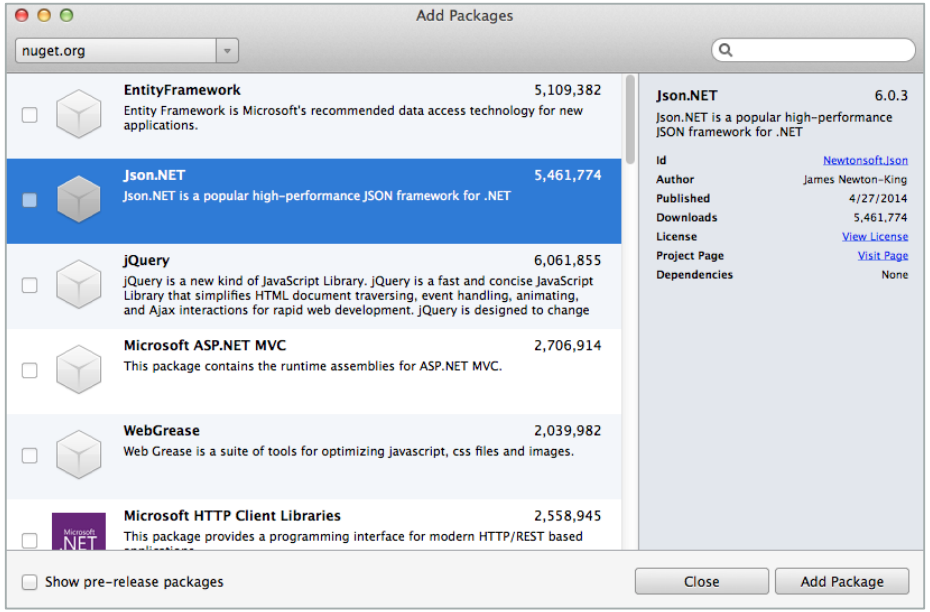
- Add NuGet packages in Visual Studio by right-clicking on the project and selecting **Manage NuGet Packages …**



Can search, update components and even revert to older revisions

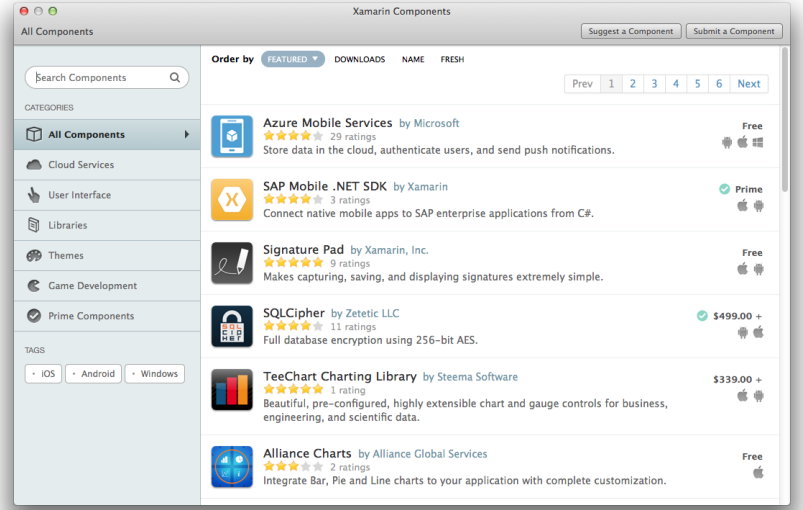
Using NuGet in Visual Studio for Mac

- Add NuGet packages in Visual Studio by right-clicking on the project and selecting **Add > Add NuGet Packages…**



Xamarin Component Store

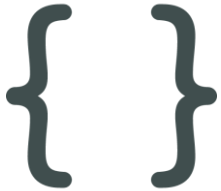
- Can also get reusable components from the **Xamarin Component Store** which is accessible through the **Components** folder in each platform project



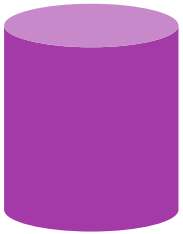
What can be shared?

- Anytime you are writing code which does not depend on a specific platform feature, it is potentially sharable, particularly if it:

Talks to a web service



Parses a data format



Uses a database

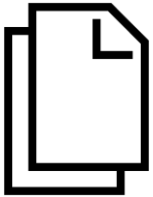
Performs processing or logic

Create shared classes + methods and then use them from your platform-specific code to maximize the shareable surface area

When is code *not* sharable?

- If the code you are writing depends on device or platform-specific APIs, or APIs not available in your project, then you will need to isolate it's use or provide some kind of *abstraction* to use it from your shared code

Access system information



Use files and folders on the device

Access personal information

Use external devices

Flash Quiz

Xamarin University

Flash Quiz

① All code you build with Xamarin is sharable across all platforms

a) True

b) False

Flash Quiz

① All code you build with Xamarin is sharable across all platforms

a) True

b) False

Flash Quiz

② The main thing that makes code sharable across platform is _____

a) When it is related to I/O

b) When it comes from NuGet

c) When it does not depend on any platform APIs

d) All of the above

Flash Quiz

② The main thing that makes code sharable across platform is _____

a) When it is related to I/O

b) When it comes from NuGet

c) When it does not depend on any platform APIs

d) All of the above

Flash Quiz

③ Which of the following might be possible candidates for sharing?

a) Code that accesses a web service with HttpClient

b) Validation rules for my UI which uses Regex and returns booleans

c) Code that uses local notifications on the device

d) Code that runs an algorithm to compare flight prices in parallel

Flash Quiz

③ Which of the following might be possible candidates for sharing?

a) Code that accesses a Web Service with HttpClient

b) Validation rules for my UI which uses Regex and returns booleans

c) Code that uses local notifications on the device

d) Code that runs an algorithm to compare flight prices in parallel

Summary

1. Add NuGet packages to your application
2. Add components to your application

Available project types

- There are two project styles available for sharing code – which one you select has an impact on *how* and *what kind* of code is shared

Shared Project

Shared Binaries

Share code using Shared Projects

Xamarin University

Tasks

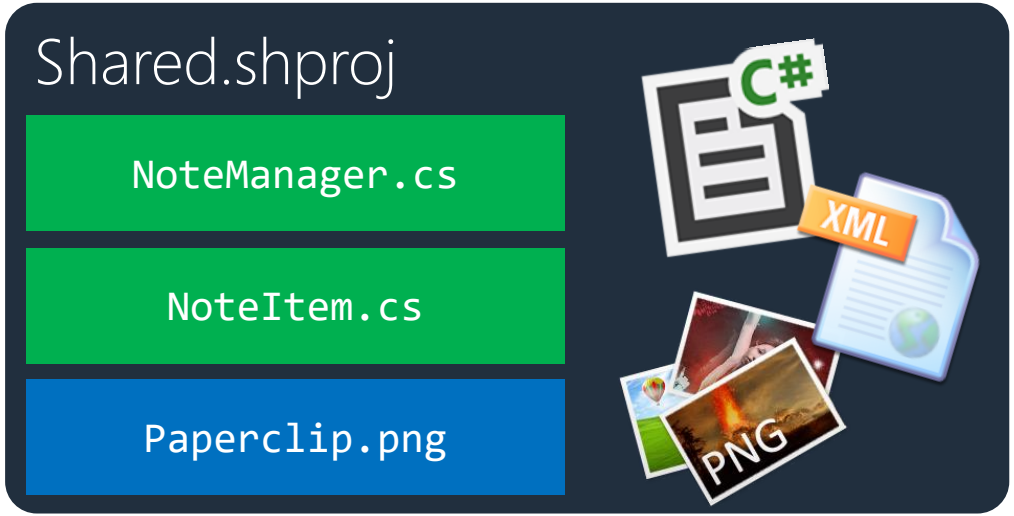
1. Share code across multiple projects with a shared project
2. Execute platform-specific code from a Shared Project

What is a Shared Project?

- Shared Projects enable project-level sharing of source + assets
  - ✓ single copy of source file
  - ✓ compiled uniquely into project
  - ✓ Normal refactoring + navigation works

Shared Project packaging

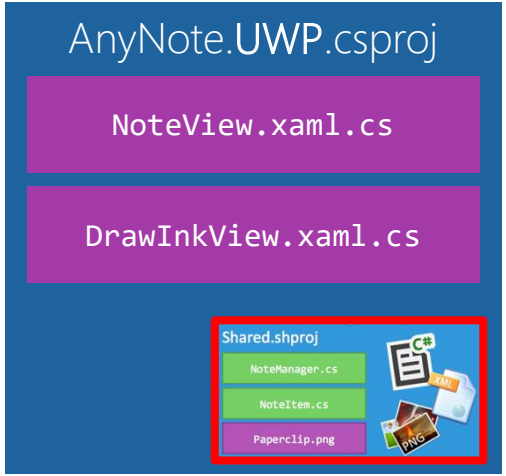
- Shared project defines the included files as well as the build type (Compile, None, etc.), but does not actually generate any output



Shared Projects defined by **.shproj** type

Shared Project internals

- Adding a reference to a shared project adds all the files to the target during the compile process, so each source file is compiled for the target



Demonstration

Using Shared Projects

Xamarin University

Platform-specific code strategies

- Several strategies for managing platform-specific code when using File Linking or Shared Projects

Conditional compilation

- Easiest strategy is to use conditional compilation to isolate platform specific code

```
#if __MOBILE__
#if __ANDROID__
#if __IOS__
#if WINDOWS_UWP
```

```
public static string DatabaseFilePath
{
    get
    {
        var filename = "HRdb.db3";
#if WINDOWS_UWP
        var path = filename;
#elif __ANDROID__
        var path = Path.Combine(
            Environment.GetFolderPath(
                Environment.SpecialFolder.Personal),
            filename);
#elif __IOS__
        string documentsPath = Environment.GetFolderPath(
            Environment.SpecialFolder.Personal);
        var path = Path.Combine(
            documentsPath,
            "..", "Library",
            filename);
#endif
        return path;
    }
}
```

Class mirroring

- Can provide specific implementation of a dependency used in the shared project – remember the shared project is not compiled on it's own

```
public class NoteManager
{
   void CloudBackupComplete() {
      Alert.Show("Success!",
            "Notes have been backed up.");
   }
}
```

Shared Project

```
class Alert
{
   internal static void Show(string title,
                             string message) {
      new UIAlertView(title, message, null, "OK")
                .Show();
   }
}
```

AnyNote.iOS

```
class Alert
{
   internal static void Show(string title,
                             string message) {
     new AlertDialog.Builder(Application.Context)
                .SetTitle(title)
                .SetMessage(message)
}
```

AnyNote.Droid

Partial classes

- Partial classes allow you to break your implementation into multiple source files
- Used primarily for generated code
- Can also be used to provide platform-specific implementations

```
partial class NoteManager
{
   void OnDeleteNote() {
      if (ShowAlert("Warning!", "...")) {
         ...
      }
   }
}
```

Shared Project

```
partial class NoteManager
{
   bool ShowAlert(
      string title, string msg) {
         ...
   }
}
```

AnyNote.iOS

Partial methods

- Can use partial methods to make the implementation *optional*
- If the method is not provided by the implementation, then the call to the method is omitted from the compiled code

```
partial class NoteManager
{
   partial void ShowPrintSettings();

   void PrintNote(NoteItem note) {
     ...
     ShowPrintSettings();
   }
}
```

Shared Project

```
partial class NoteManager
{
   // No definition of method
}
```

NoteManager.iOS

Individual Exercise

Working with Shared Projects

Xamarin University

Flash Quiz

Xamarin University

Flash Quiz

① Shared Projects create an output assembly directly

a) True

b) False

Flash Quiz

① Shared Projects create an output assembly directly

a) True

b) False

Flash Quiz

② What types of files can you add to a Shared Project?

a) Source Code only

b) Source and Image assets

c) Source and Data files

d) Anything supported by the targets using the project

Flash Quiz

② What types of files can you add to a Shared Project?

a) Source Code only

b) Source and Image assets

c) Source and Data files

d) Anything supported by the targets using the project

Flash Quiz

③ What techniques can be used to isolate platform specific code in a Shared Project?

a) Conditional Compilation

b) Partial classes

c) Both (a) and (b)

d) None of the above.

Flash Quiz

③ What techniques can be used to isolate platform specific code in a Shared Project?

a) Conditional Compilation

b) Partial classes

c) Both (a) and (b)

d) None of the above.

Summary

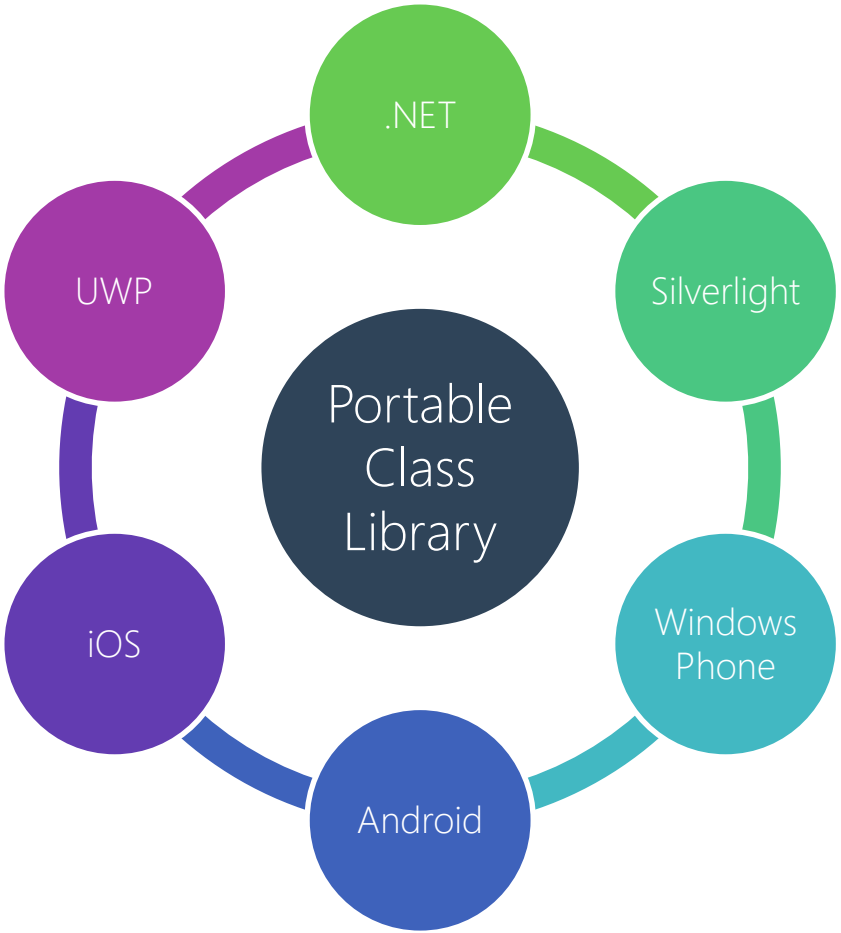
1. Share code across multiple projects with a shared project
2. Execute platform-specific code from a Shared Project

Share code using Portable Class Libraries

Xamarin University

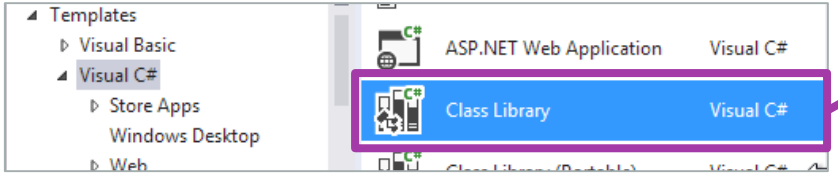
Tasks

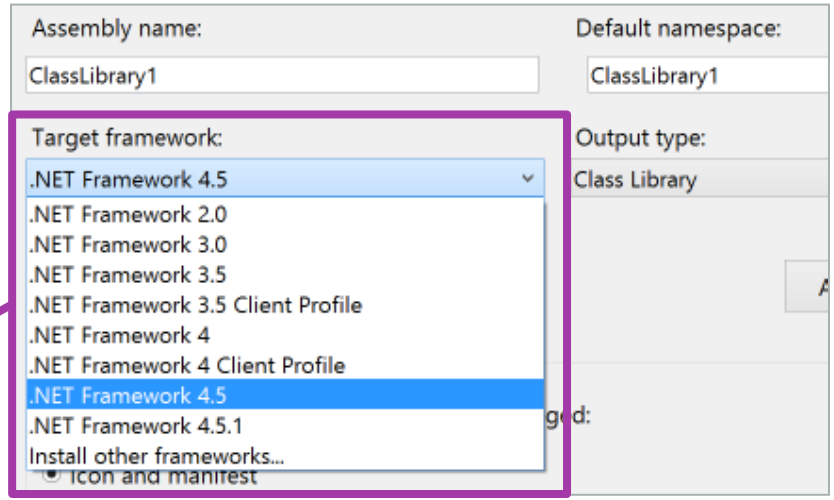
1. Portable Class Libraries
2. Profiles
3. Handling Platform Abstractions

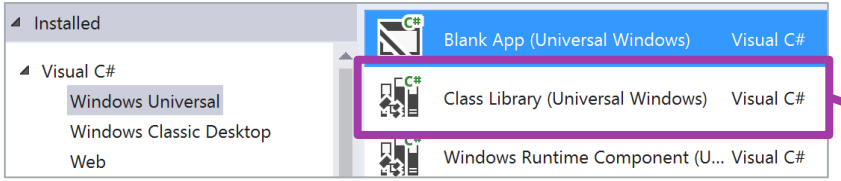


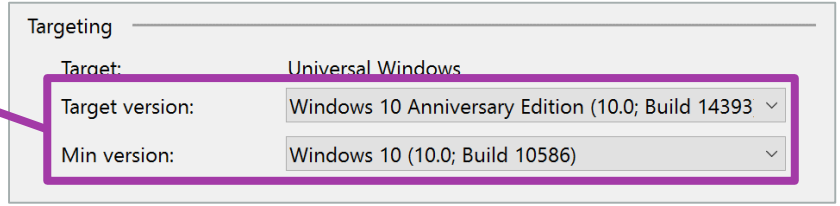
Class Library projects

- Class Library projects are tied to a specific platform + frameworks



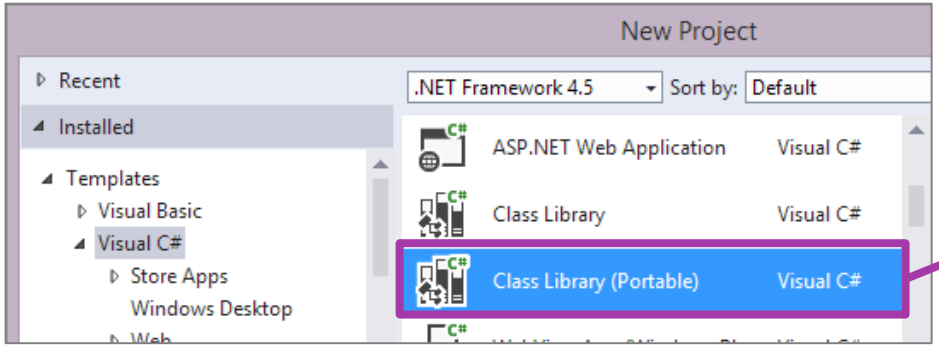


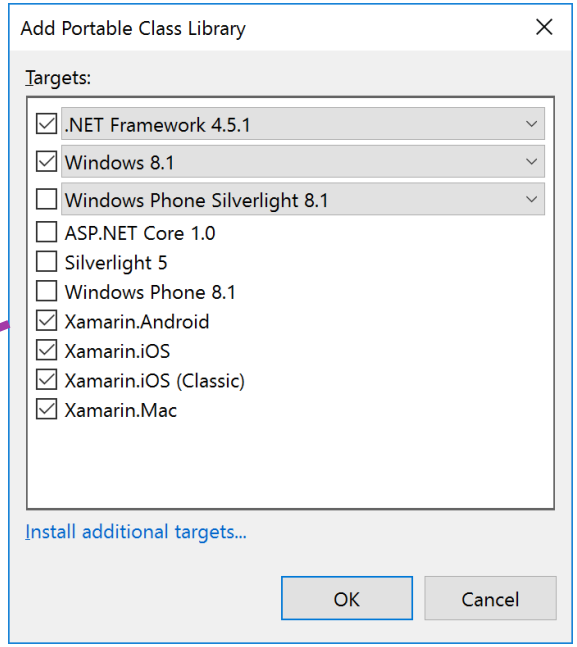




Portable Class Libraries (PCL)

- Portable Class Libraries are assemblies that can be used by different flavors of .NET without recompiling



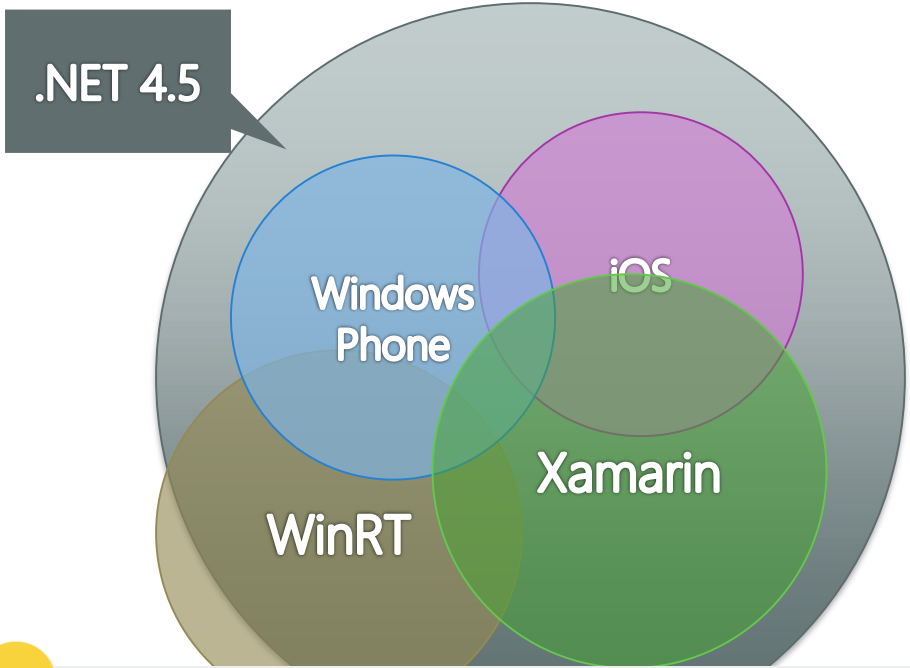


How does it work?

- A PCL is tied to a specific *profile* which defines the specific APIs it can use

Feature	.NET Framework	Windows Store	Silverlight	Windows Phone (SL)	Windows Phone (Store)	Xamarin
Core Libraries	✓	✓	✓	✓	✓	✓
LINQ	✓	✓	✓	✓	✓	✓
IQueryable	✓	✓	✓	7.5+	✓	✓
Compression	4.5+	✓	✗	✗	✓	✓
Data Annotations	4.0.3+	✓	✓	✗	✗	✓
System.IO.File	✗	✗	✗	✗	✗	✗

Configuring Portable Class Libraries



You select the platforms the library will be used on – this decides the profile

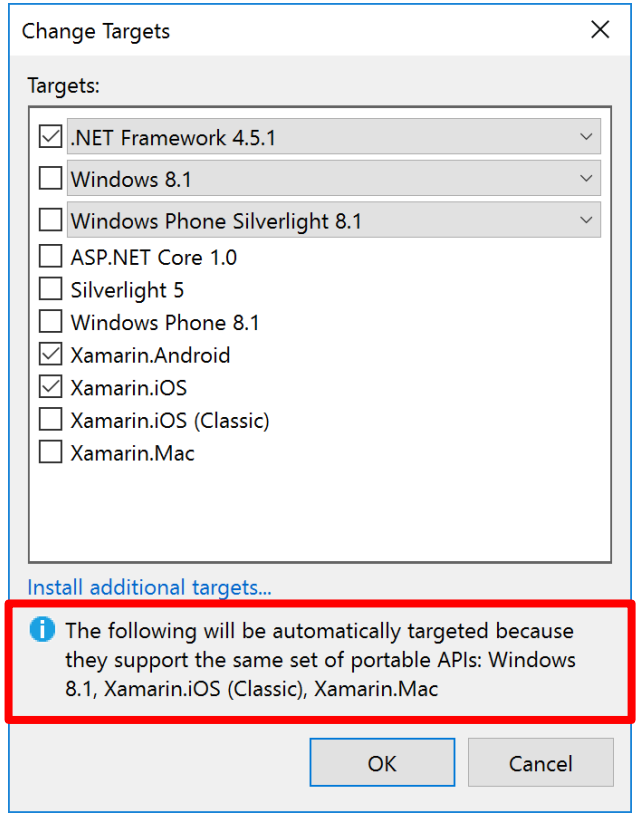
The available combinations are controlled by the profiles Microsoft has defined

The *more platforms* you choose, the *less APIs* you will be able to use

Pick only the framework targets you *need right now* to give you the broadest API reach as possible, can always add other targets later if you expand your platforms

Missing profiles

- Some platform combinations are not allowed because Microsoft has not defined a profile for that combination
- IDE will attempt to pick the closest variation, or give an error and require that you add an additional target



Demonstration

Creating a Portable Class Library

Xamarin University

PCL limitations

- PCLs are limited to features which are common to the targeted frameworks; this means a lot of classes will be *missing* when you are in a PCL project

```
partial class NoteManager
{
    void LoadNotes(string filename) {
        var reader = new System.IO.StreamReader(filename);
    }
}
```

```
public StreamReader(
  System.IO.Stream stream
)
```

▲ 1 of 6 ▼

Parameter

stream: The stream to be read.

Summary

Initializes a new instance of the *StreamReader* class for the specified stream.

Selected profile has no constructor on **`StreamReader`** which takes a string

Platform-specific code strategies

- Several approaches you can take to passing data between the platform-specific code and the shared (PCL) code

Fill in properties with loaded data

Call the API in the platform-specific code and fill in exposed public properties in the shared code with the results

Platform-specific code strategies

- Several approaches you can take to passing data between the platform-specific code and the shared (PCL) code

Fill in properties with loaded data

Open and pass supported types to PCL

Decide the location + filename, open a `Stream` and let shared code parse/load the data

Platform-specific code strategies

- Several approaches you can take to passing data between the platform-specific code and the shared (PCL) code

Fill in properties with loaded data

Open and pass

Design higher-level abstractions

Can use an abstraction such as an interface or an event and provide an implementation of that abstraction in the platform-specific project(s)

Callbacks

- PCLs can **expose events or delegates** to request extensibility from the platform code, particularly effective if requirements are small

```
public class Dialer
{
   public static
       Func<string,bool> MakeCallImpl;

   public bool MakeCall(string number) {
     if (MakeCallImpl(number)) {
        ...
     }
   }
}
```

PCL

```
Dialer.MakeCallImpl = number =>
{
  return UIApplication
        .SharedApplication
        .OpenUrl(new NSUrl(
            "tel:" + number));
}
```

Xamarin.iOS

Platform abstractions

- Complex requirements can be described by an abstraction that is defined in the PCL

```
public interface IDialer
{
    bool MakeCall(string number);
}
```

PCL

Shared code defines **IDialer** interface to **represent required functionality** – this is what the PCL uses to get to the API

PhoneDialerIOS

PhoneDialerDroid

PhoneDialerWin

Platform projects **implement the shared dialer interface** using the platform-specific APIs

Injecting dependencies

- Can supply concrete implementation to PCL via constructor, method or property setter; this technique is often called *Dependency Injection*

```
Dialer.Instance = new Dialer(new iPhoneDialer());
```

OR

```
Dialer.Instance.Initialize(new AndroidDialer());
```

OR

```
Dialer.Instance.Platform = new WindowsDialer();
```

Individual Exercise

Working with Portable Class Libraries

Xamarin University

Flash Quiz

Xamarin University

Flash Quiz

① Portable Class Libraries share source code files across projects

a) True

b) False

Flash Quiz

① Portable Class Libraries share source code files across projects

a) True

b) False

Flash Quiz

② When you define your platform targets, you are selecting a _____.

a) Configuration
b) Platform Group
c) Profile
d) Grouping

Flash Quiz

② When you define your platform targets, you are selecting a _____.

a) Configuration
b) Platform Group
c) Profile
d) Grouping

Flash Quiz

④ What techniques can I use to add platform-specific code to a PCL?

a) Dependency Injection (DI)
b) Service Locator
c) Publisher / Subscribe (events or messaging system)
d) Any of the above

Flash Quiz

④ What techniques can I use to add platform-specific code to a PCL?

a) Dependency Injection (DI)

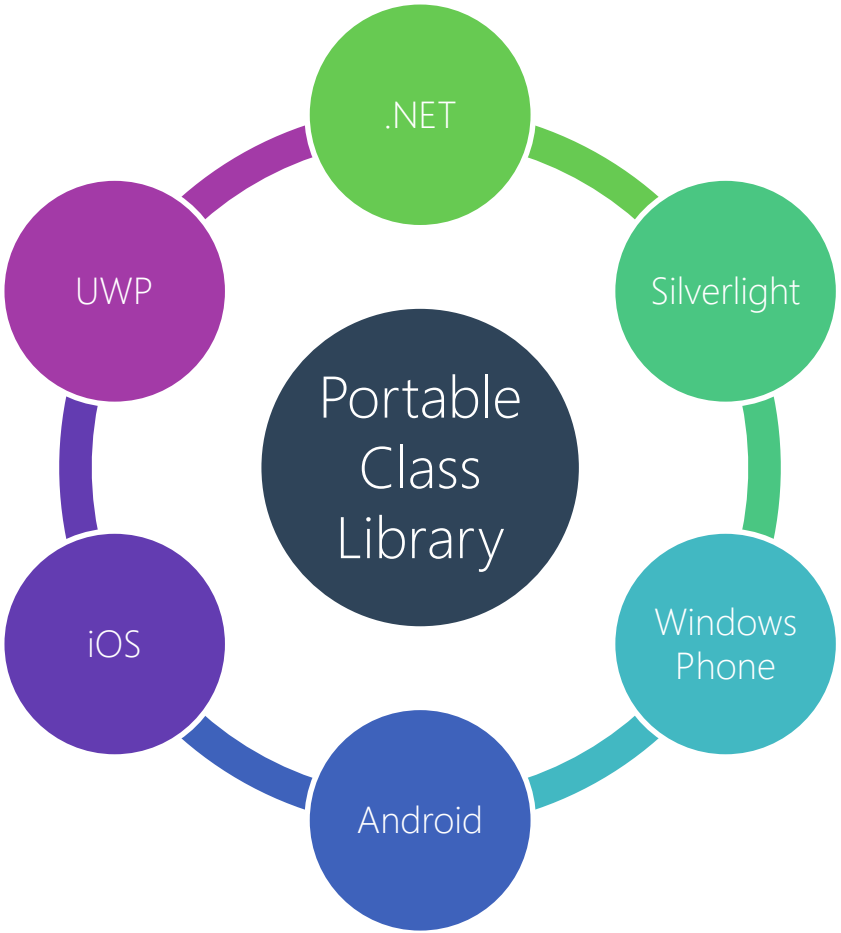
b) Service Locator

c) Publisher / Subscribe (events or messaging system)

d) Any of the above

Summary

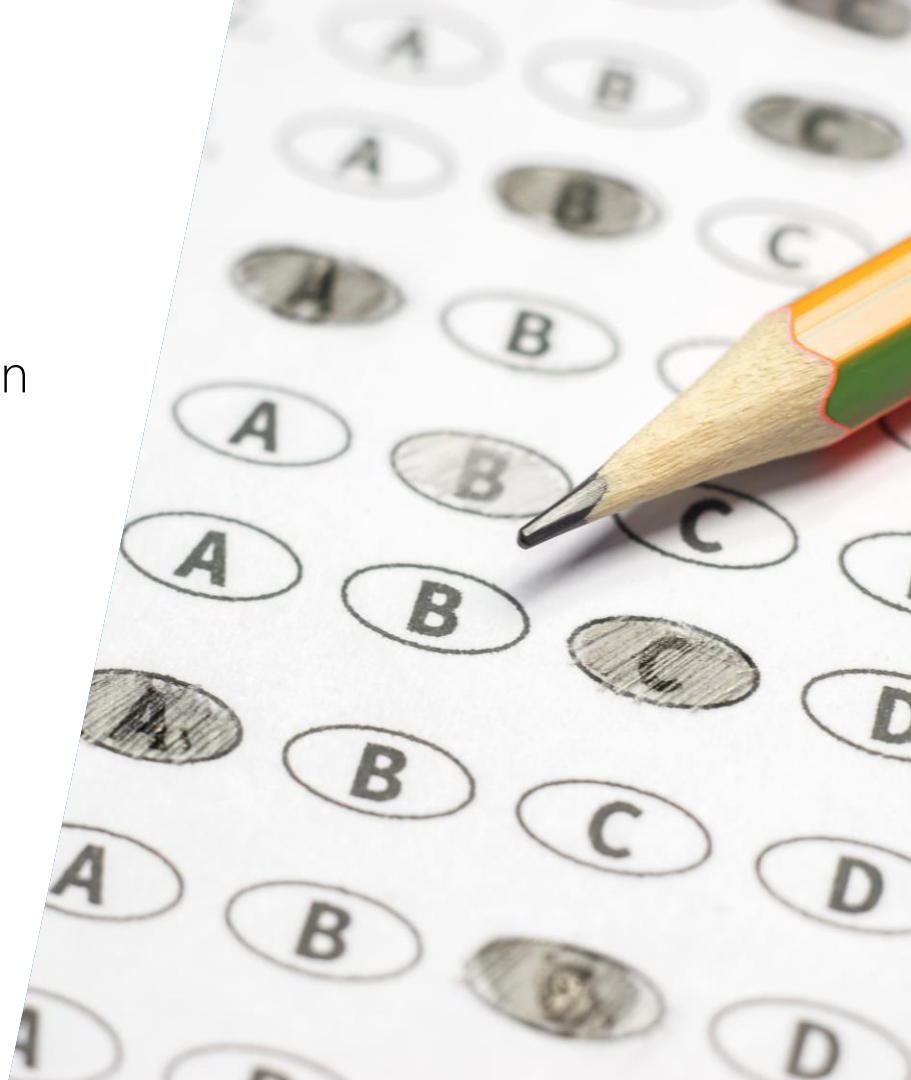
1. Portable Class Libraries
2. Profiles
3. Handling Platform Abstractions



Share code using .NET Standard libraries

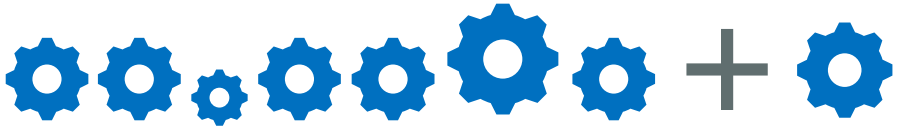
Tasks

1. Create a .NET Standard library
2. Select a .NET Standard target version for your library
3. Use a .NET Standard library with a Xamarin app



Motivation

- Sharing code across platforms has some common challenges

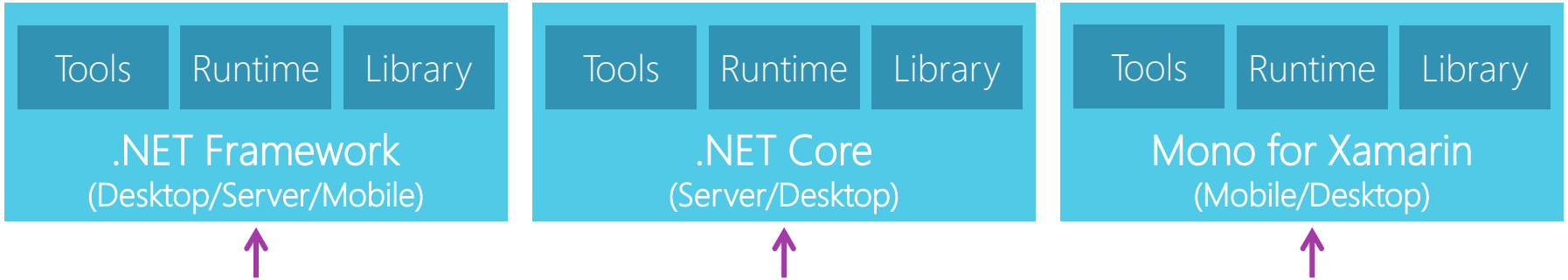


API surfaces are not uniform

New platforms may not have the APIs your library needs

Platforms

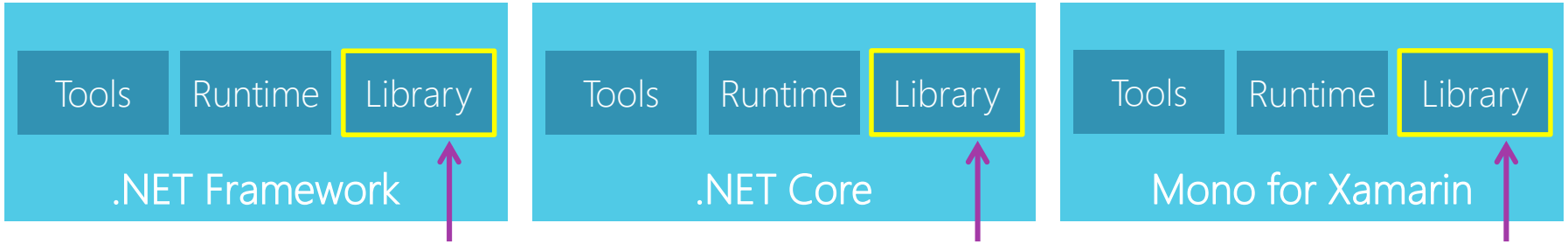
- In .NET, a *platform* is a set of software components capable of building and executing an application on some target operating systems



The three primary Microsoft platforms let you run apps on a wide range of targets

Library variation

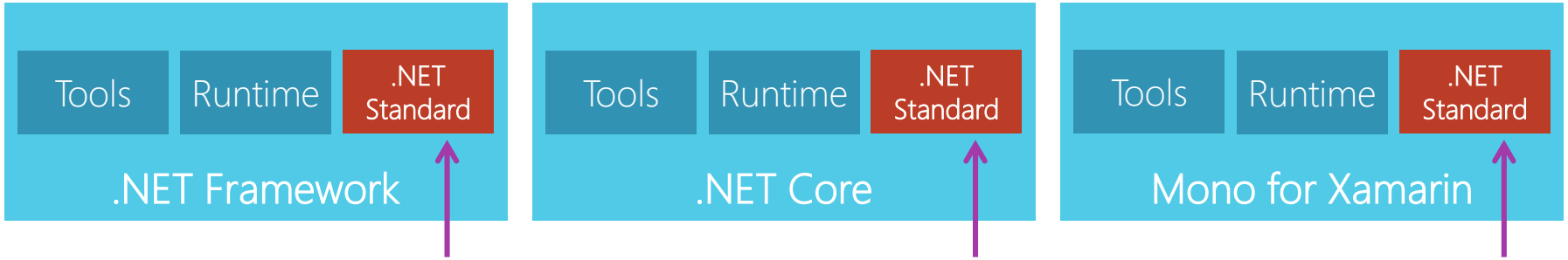
- The libraries offered by each platform are different – they do not provide a uniform API surface for you to code against



This makes it difficult for you to write one library that works on all these platforms – what if you need to use an API that's not available everywhere?

What is .NET Standard?

- *.NET Standard* is a library specification that provides the same APIs across all .NET platforms

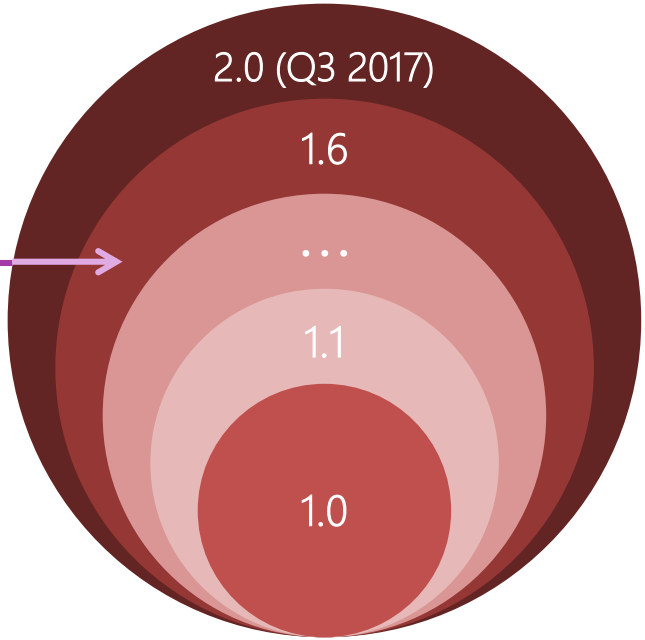


This makes it easier to build a library that works on all these platforms since the available APIs are standardized

.NET Standard versions

- .NET Standard is under active development and has many versions – each new version increases the number of available APIs

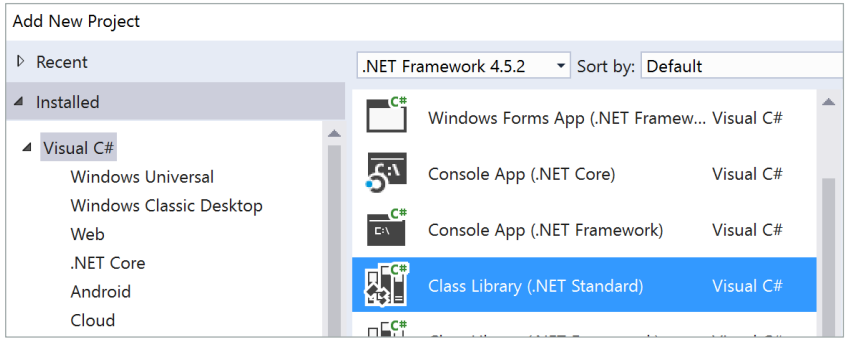
New versions are supersets, they do not remove any existing APIs



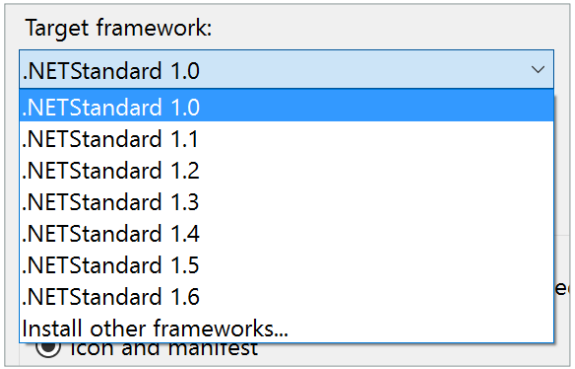
How to create a .NET Standard library?

- **Visual Studio 2017** and **Visual Studio for Mac v7.0** include a project template for creating .NET Standard class libraries

1 Use the template to create the project



2 Choose the version in the properties



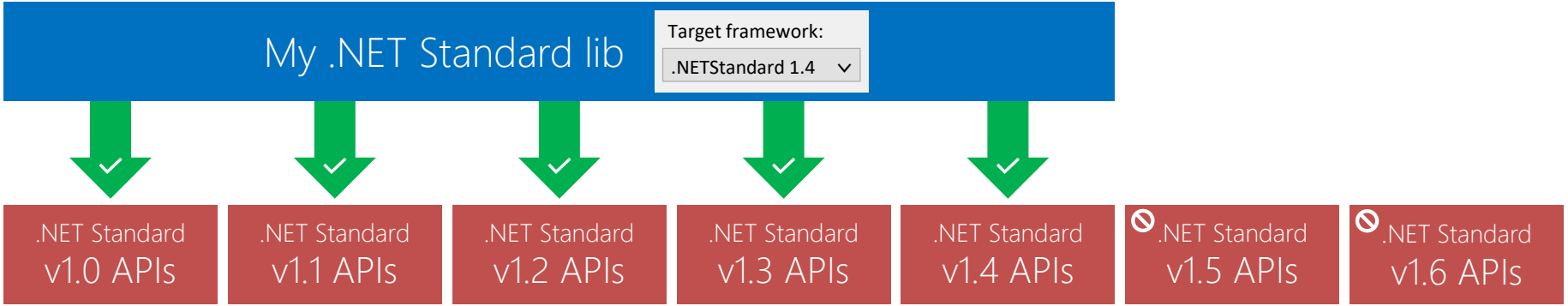
Available APIs

- The .NET team at Microsoft decides which APIs to add to each .NET Standard release (this includes a public feedback process)

Version	APIs	API count
1.0	Primitives, reflection, tasks, collections, LINQ to Objects, XML, …	7,949
1.1	Concurrent collections, interop, HTTP interactions, …	10,239
1.2	Threading timer, more interop, …	10,285
1.3	Console, file system, thread pool, sockets, cryptography, …	13,122
1.4	More cryptography, …	13,140
1.5	More assembly members, more streams, …	13,355
1.6	Even more cryptography, more regex, expression compiling, …	13,501
2.0 (Q3 2017)	Data classes, drawing, pipes, caching, SMTP, web sockets, more serialization, XPath, expanded many existing classes, …	32,638

.NET Standard API use

- Your library can use .NET Standard APIs added in versions less than or equal to your target version



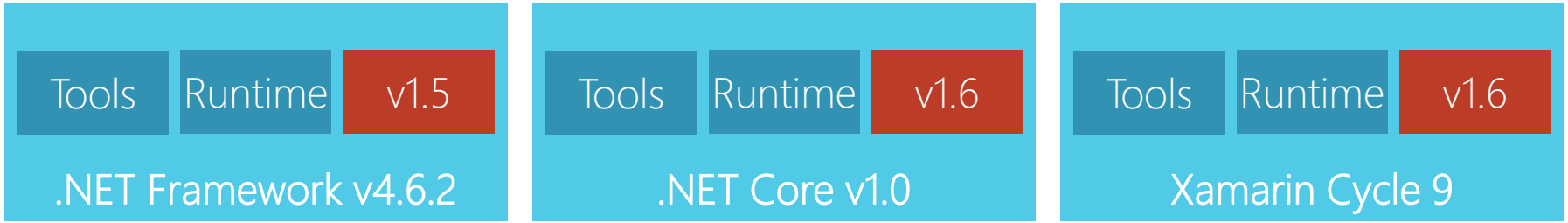
Demonstration

Create a .NET Standard library and use .NET Standard APIs

Xamarin University

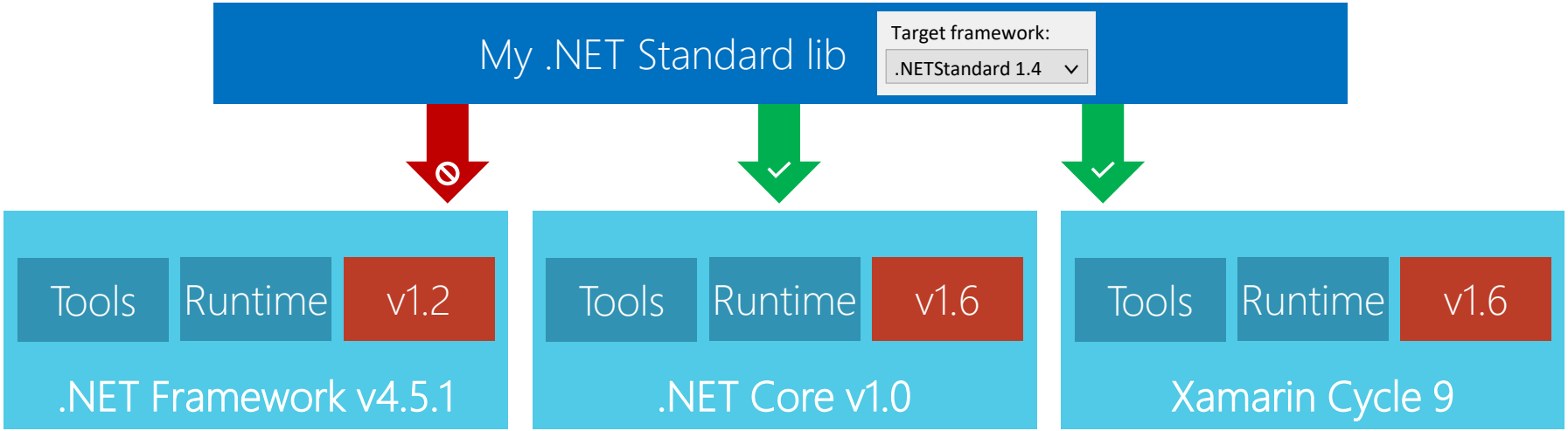
Platform support

- Each platform can support its preferred version of .NET Standard



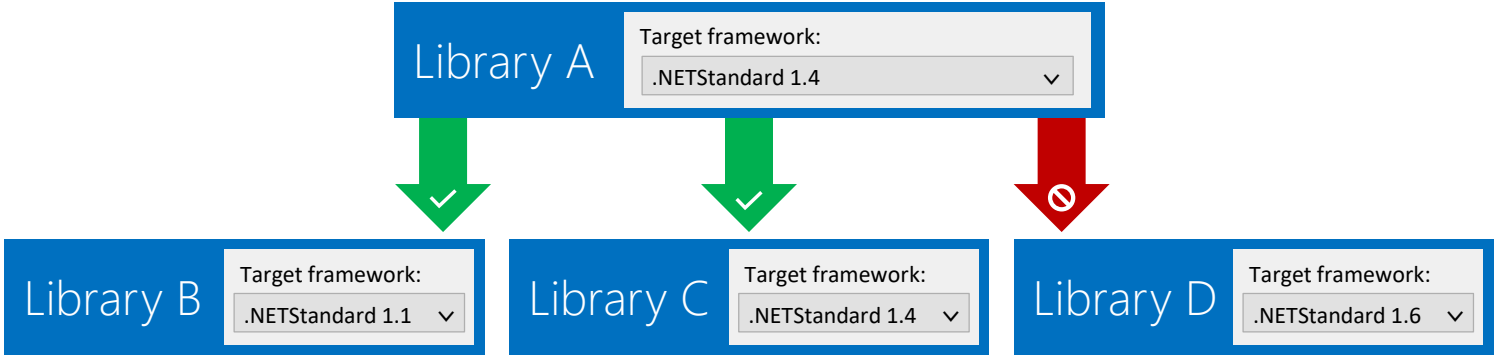
Where can your library be used?

- Your library can be used by platforms supporting your targeted .NET Standard version or higher



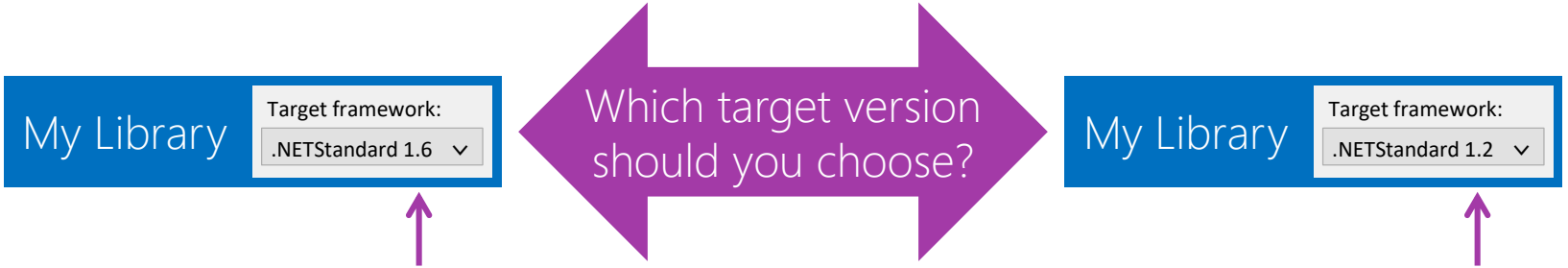
Which libraries can you reference?

- A library targeting .NET Standard can reference a library targeting the same .NET Standard version or lower



How to choose target version

- You should target the lowest .NET Standard version that gives you all the APIs you need – this maximizes the number of platforms you can run on



Targeting higher means you can use more APIs (e.g. any APIs added in 1.0 through 1.6)

Targeting lower means you are compatible with more platforms (e.g. all that support at least 1.2)

PCL Profile compatibility

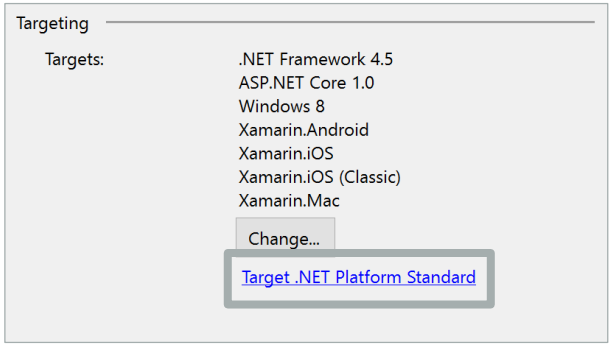
- Most PCL Profiles popular in Xamarin development are paired with .NET Standard versions, allowing those libraries to reference each other

PCL Profile (all support Xamarin)	.NET Standard
Profile151 (.NET Framework 4.5.1, Windows 8.1, Windows Phone 8.1)	1.2
Profile44 (.NET Framework 4.5.1, Windows 8.1)	1.2
Profile111 (.NET Framework 4.5, Windows 8, Windows Phone 8.1)	1.1
Profile7 (.NET Framework 4.5, Windows 8)	1.1
Profile259 (.NET Framework 4.5, Windows 8, Windows Phone 8.1, Windows Phone Silverlight 8)	1.0
Profile78 (.NET Framework 4.5, Windows 8, Windows Phone Silverlight 8)	1.0

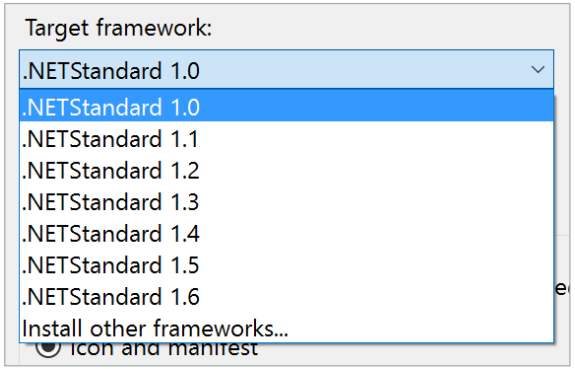
Convert a PCL to .NET Standard

- **Visual Studio** lets you convert a PCL to a .NET Standard library

1 Retarget in the project properties



2 Choose your version



Handle platform differences

- You use the same techniques to handle platform-specific code in .NET Standard libraries as in PCLs

Demonstration

Upgrading from PCL to .NET Standard Libraries

Xamarin University

Flash Quiz

Xamarin University

Flash Quiz

① Libraries targeting .NET Standard versions share source code files across projects

a) True

b) False

Flash Quiz

① Libraries targeting .NET Standard versions share source code files across projects

a) True

b) False

Flash Quiz

② A .NET Standard library targeting .NET Standard version 1.4 can use APIs from .NET Standard 1.2

a) True

b) False

Flash Quiz

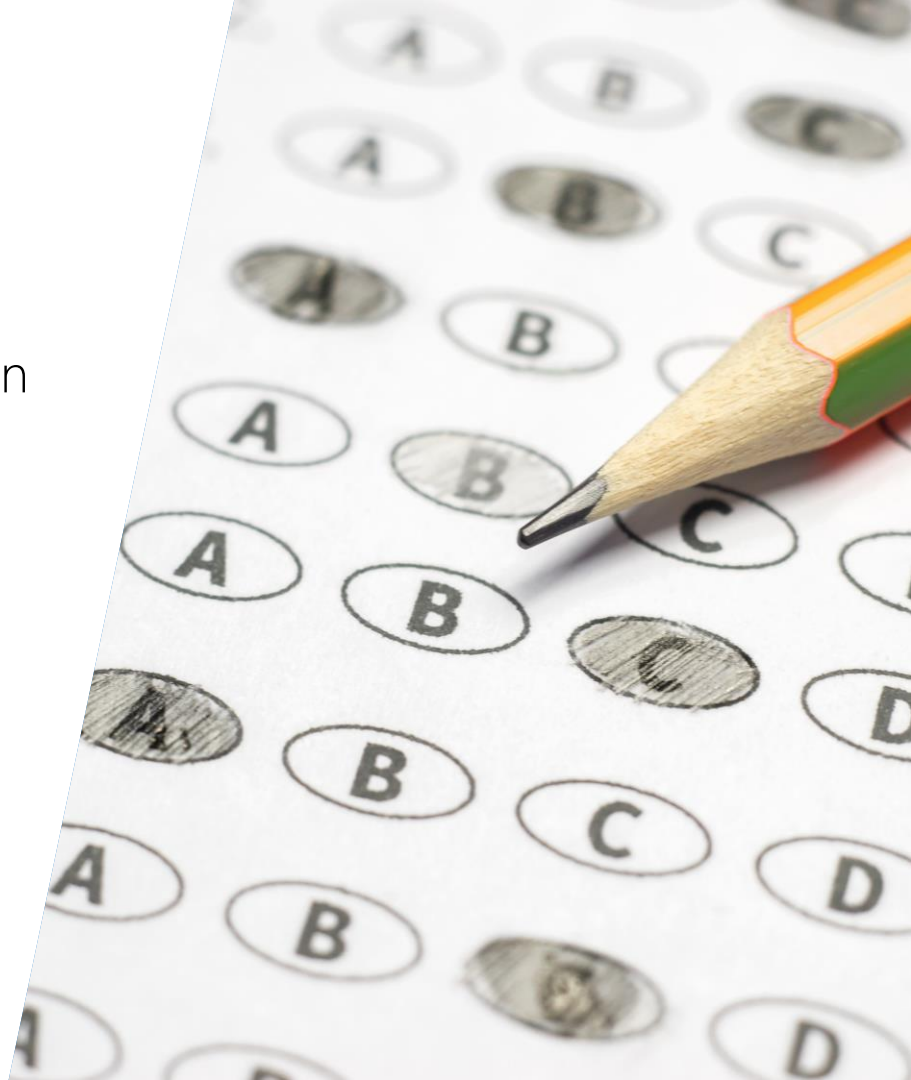
② A .NET Standard library targeting .NET Standard version 1.4 can use APIs from .NET Standard 1.2

a) True

b) False

Summary

1. Create a .NET Standard library
2. Select a .NET Standard target version for your library
3. Use a .NET Standard library with a Xamarin app



Code-sharing comparison

- Selecting a code-sharing technique depends on many factors – each style has advantages and disadvantages

Shared Projects	
Pros	Cons
All APIs available	Can lead to spaghetti code
Platform-specific logic can be added directly	Difficult to unit test due to conditional code
All file types can be shared	Must be shipped in source form
Smaller package sizes/platform-specific optimizations	

PCL / .NET Standard libraries	
Pros	Cons
Enforces architectural design	Limited APIs available
Can be unit tested separately	Difficult to share non-code files
Can be shipped in binary form (NuGet)	Limited to target platforms or APIs
	Requires more work to integrate platform-specific code

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile

Microsoft