XAM110

Intro to Cross-Platform Mobile Development

- Lecture will begin shortly
- Download class materials from university.xamarin.com

Xamarin University

Objectives

1. Working with Shared Components
2. Using Shared Projects
3. Using Portable Class Libraries

Working with Shared Components

Xamarin University

Tasks

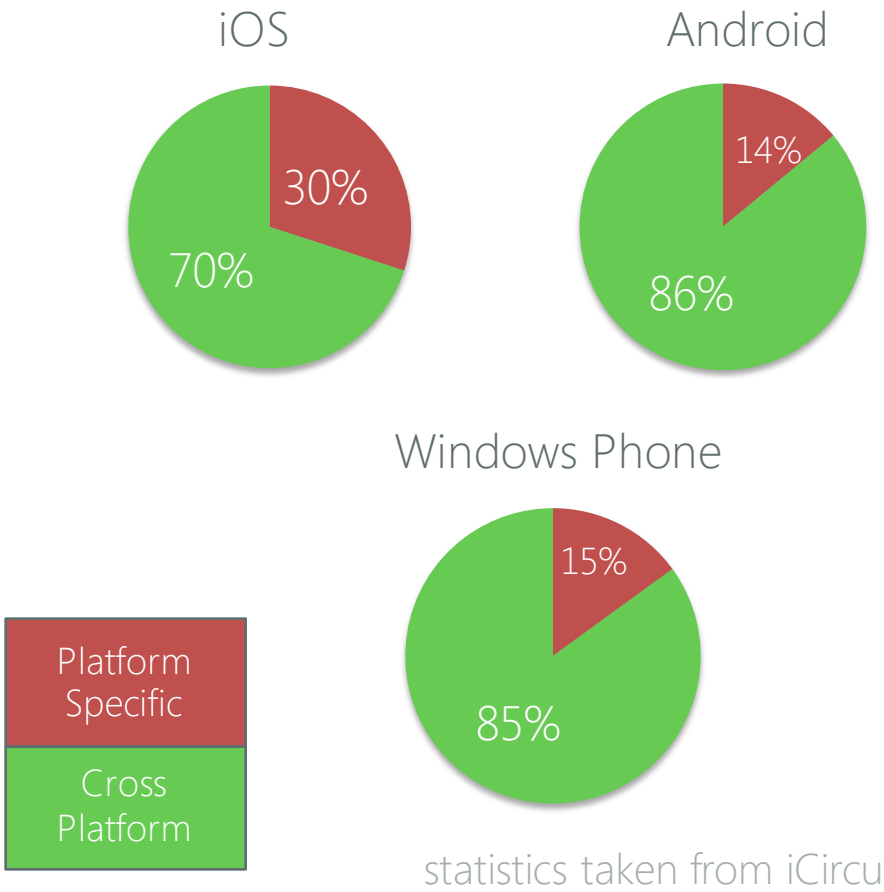
1. Sharable Code
2. Nuget in Xamarin Studio
3. Component Store

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

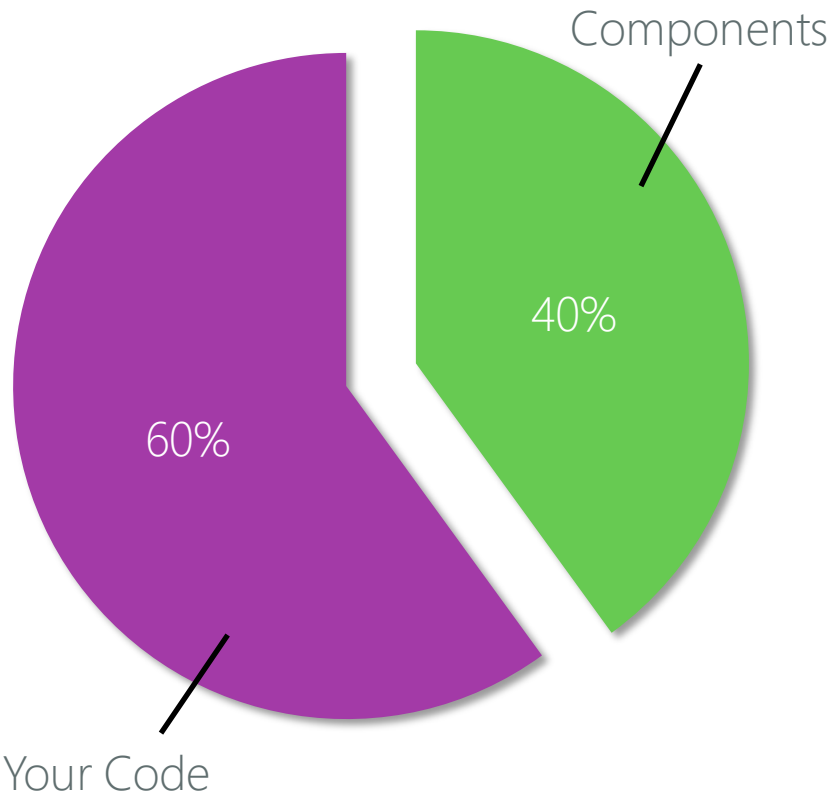
Sharable Code

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



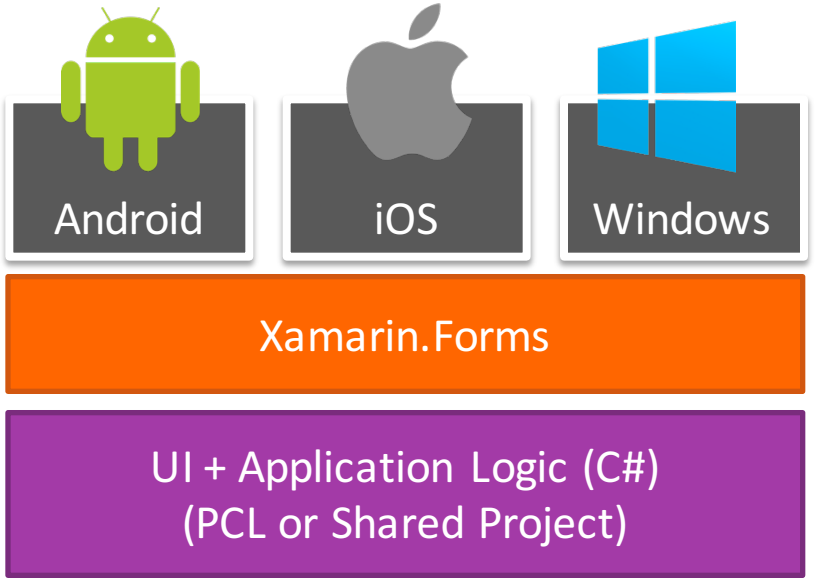
Sharable code

- 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
- Renders native UI on iOS, Android and Windows Phone 8
- Attend **XAM120** – Intro to Xamarin.Forms for more info



Must use PCL for XAML pages; can use Shared Projects or PCLs for other shared code

Data Access (Database)

- SQLite support available for iOS, Android and Windows
- Can also store in the cloud – Azure Mobile Services, Amazon, Dropbox, etc.
- Attend **XAM160** – SQLite and Data in Mobile for more information

Web Services

- Use **`HttpClient`** for REST services, can then process with
  - **`System.Xml`** / **`System.Json`**
  - LINQ to XML
  - Json.NET component
- Use WCF or **`.asmx`** for SOAP
- Attend **XAM150** and watch **XAM151** for more info

Xamarin.* Libraries

- Open-Source, Cross-Platform APIs available from **Github.com/Xamarin**
  - Xamarin.Social
  - Xamarin.Auth
  - Xamarin.Mobile
- Check out .NET Foundation for more great open source libraries like MailKit and Rx

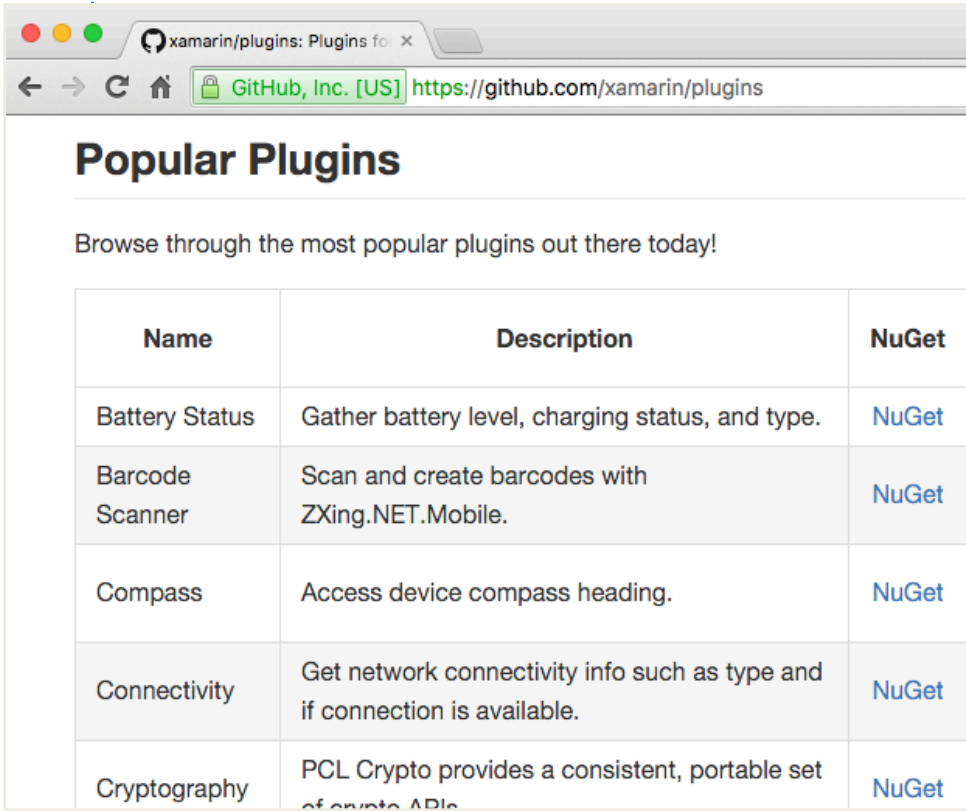
Example: take a picture

- **Xamarin.Mobile** supports accessing the camera in a cross-platform fashion

```
async void OnTakePicture(object sender, EventArgs e)
{
    var picker = new MediaPicker();
    if (picker.IsCameraAvailable) {
        MediaFile photo = await picker.TakePhotoAsync(
            new StoreCameraMediaOptions {
                Name = "photo.jpg",
                DefaultCamera = CameraDevice.Rear
        });
        string filePath = photo.Path;
        Stream photoStream = photo.GetStream();
        ... // Load stream or file into native image
    }
}
```

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



NuGet

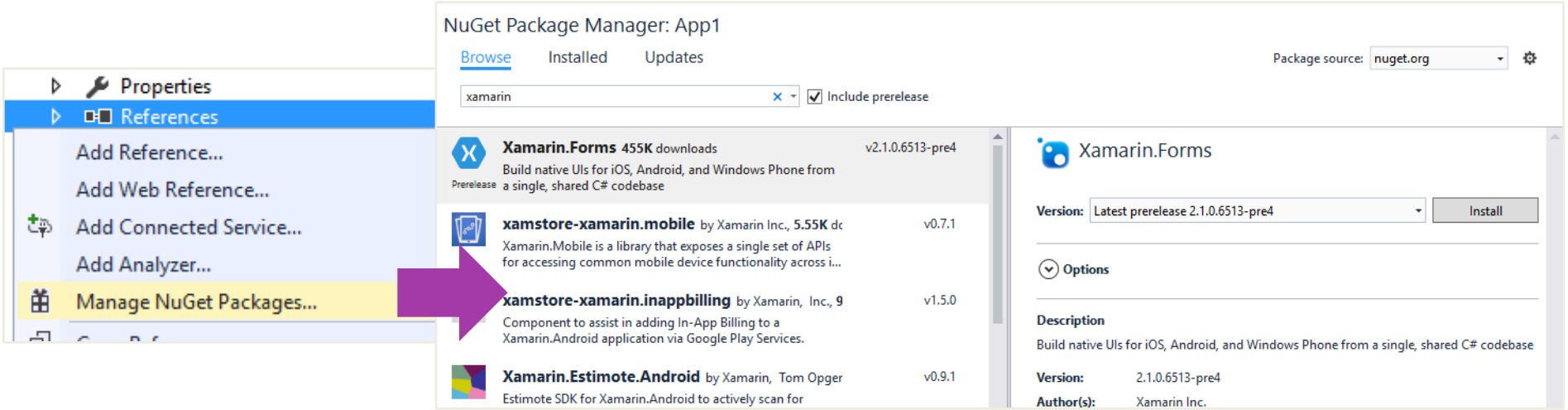
- NuGet is a package manager for .NET that allows you to locate, install, update and remove shared components from your projects right in your IDE (either Visual Studio or Xamarin Studio)

- Lots of shared components available which support mobile development

www.nuget.org

Using Nuget in Visual Studio

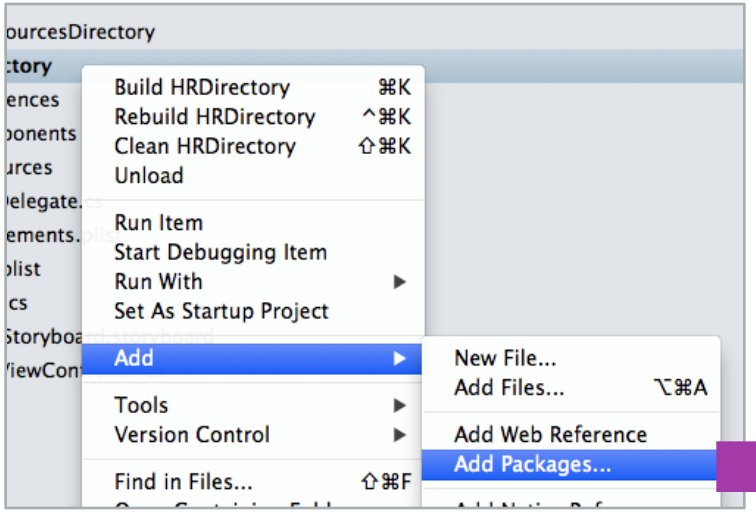
- Can add Nuget components in Visual Studio using **References** folder

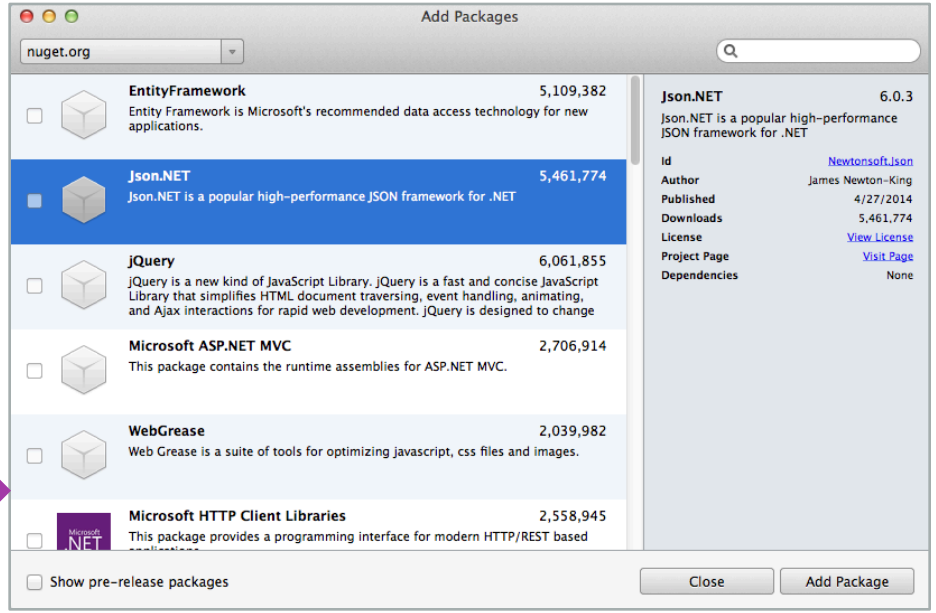


Can search, update components and even revert to older revisions

Using Nuget in Xamarin Studio

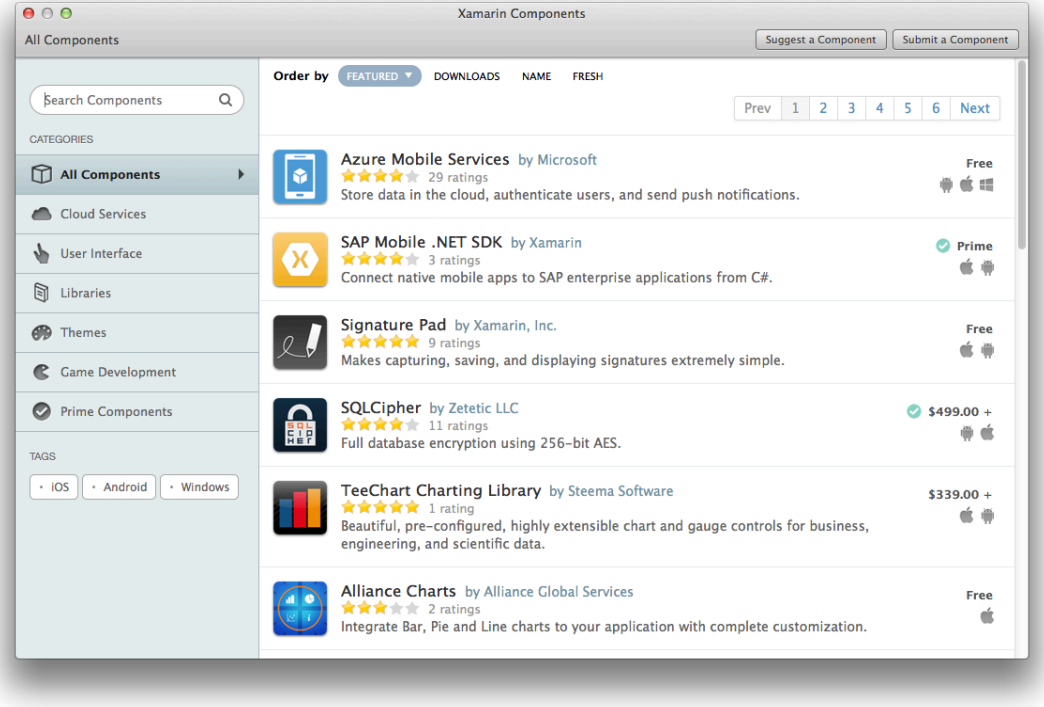
- Right-click on the Project and select Add > Packages…





Xamarin Component Store

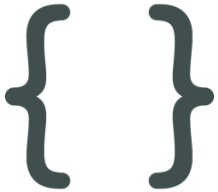
- Can also get reusable components from the **Xamarin Component Store** which is accessible through the **Components** folder in each project



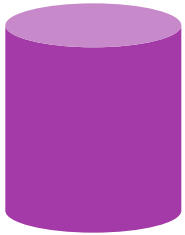
Where can I use shared code?

- 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

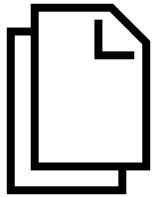
Performs processing or logic

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

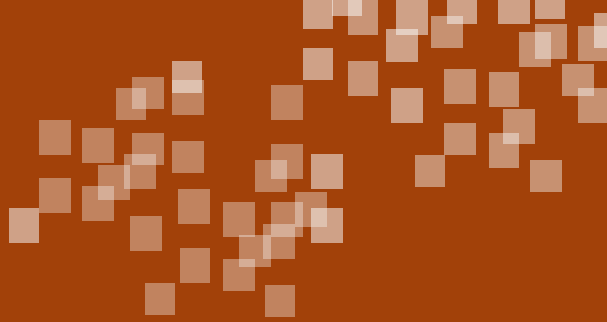
③ Which of the following might be possible candidates for sharing?

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



Demonstration

Nuget and the Xamarin Component Store

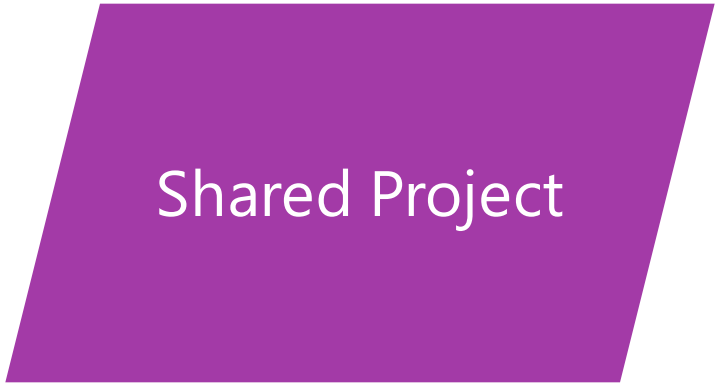
Xamarin University

Summary

1. Sharable Code
2. Nuget in Xamarin Studio
3. Component Library

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



Using Shared Projects

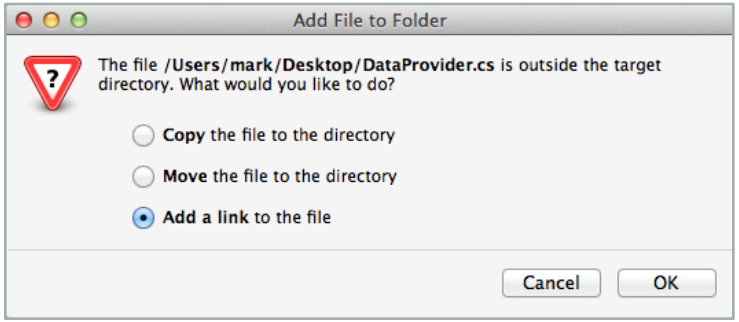
Xamarin University

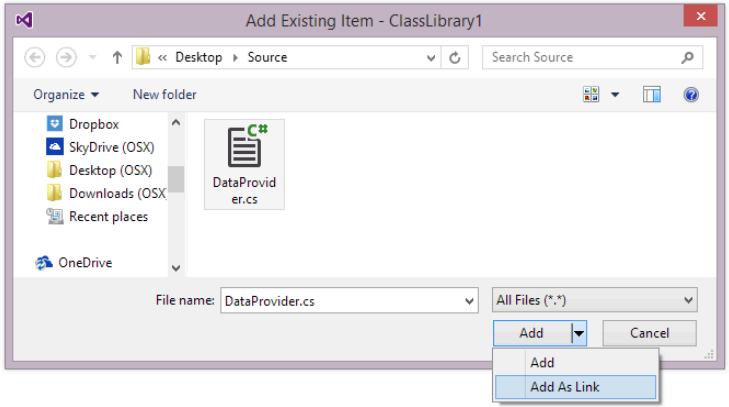
Tasks

1. File Linking
2. Introduction to Shared Projects
3. Shared Project Internals
4. Platform Specific Code Strategies

The Old Way …

- Can use *File Linking* to share source files between projects
  - ✔ single copy of source file
  - ✔ can use compiler directives to isolate platform-specific code
  - ✖ Refactoring + Navigation limited
  - ✖ Testing painful



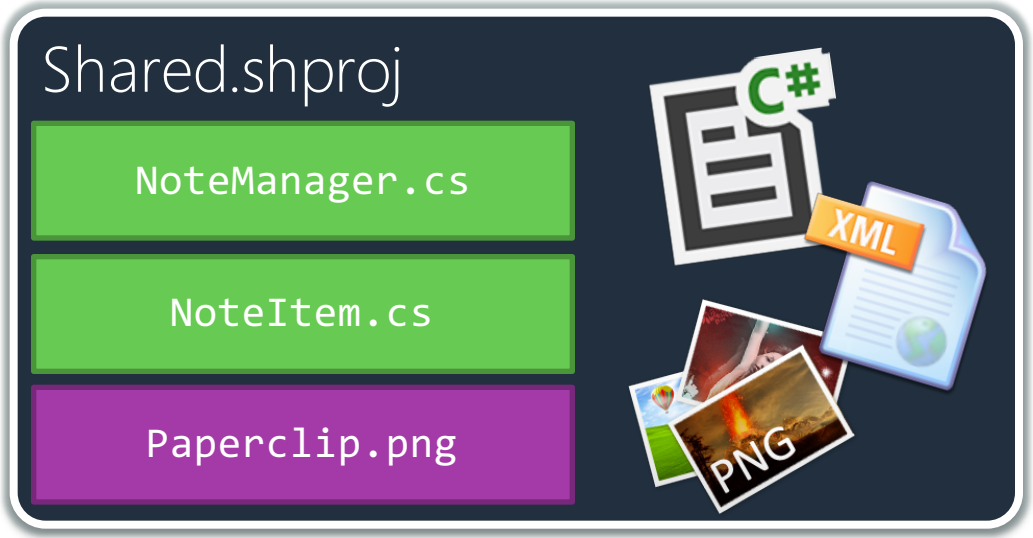


What is a Shared Project?

- Shared Projects enable project-level sharing of source + assets
  - ✔ single copy of source file
  - ✔ compiled uniquely into project
  - ✔ Normal refactoring + navigation works
  - Requires VS2013 + Update 2

Shared Projects packaging

- Shared Projects defined by new **.shproj** type

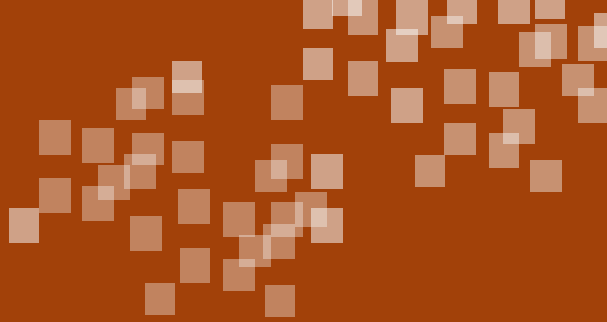


Shared project defines the **included files** as well as the **build type** (Compile, None, etc.), but does not actually generate any output

no assembly is produced

Shared Projects 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

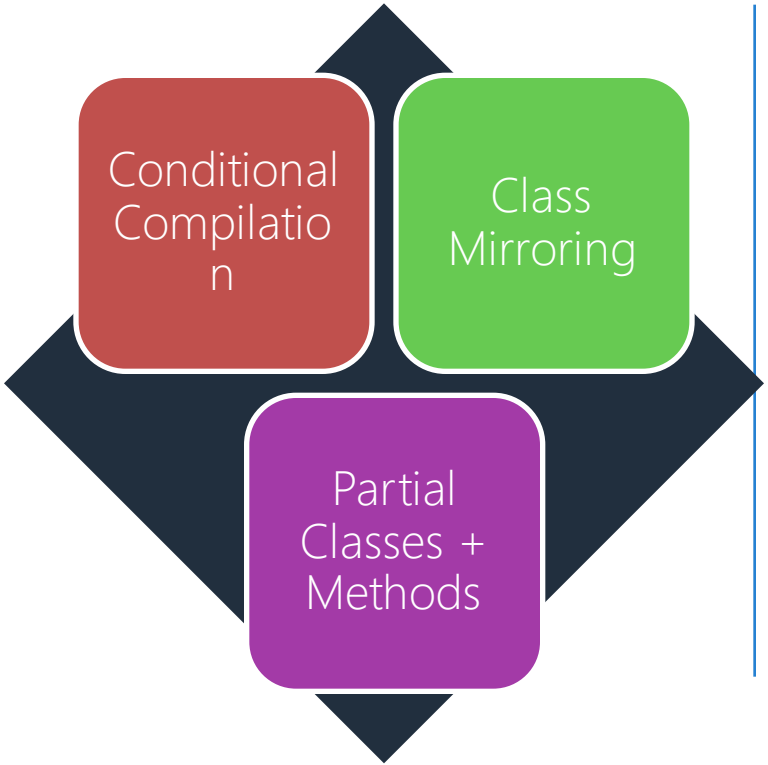


Demonstration

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_PHONE
  - #if SILVERLIGHT

```
public static string DatabaseFilePath {
    get {
        var filename = "HRdb.db3";
    #if WINDOWS_PHONE
        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

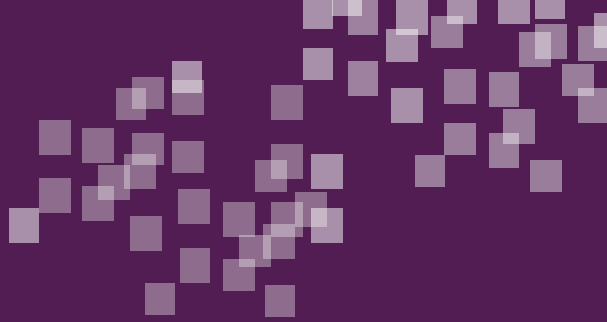
```
partial class NoteManager
{
   partial void ShowPrintSettings();

   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

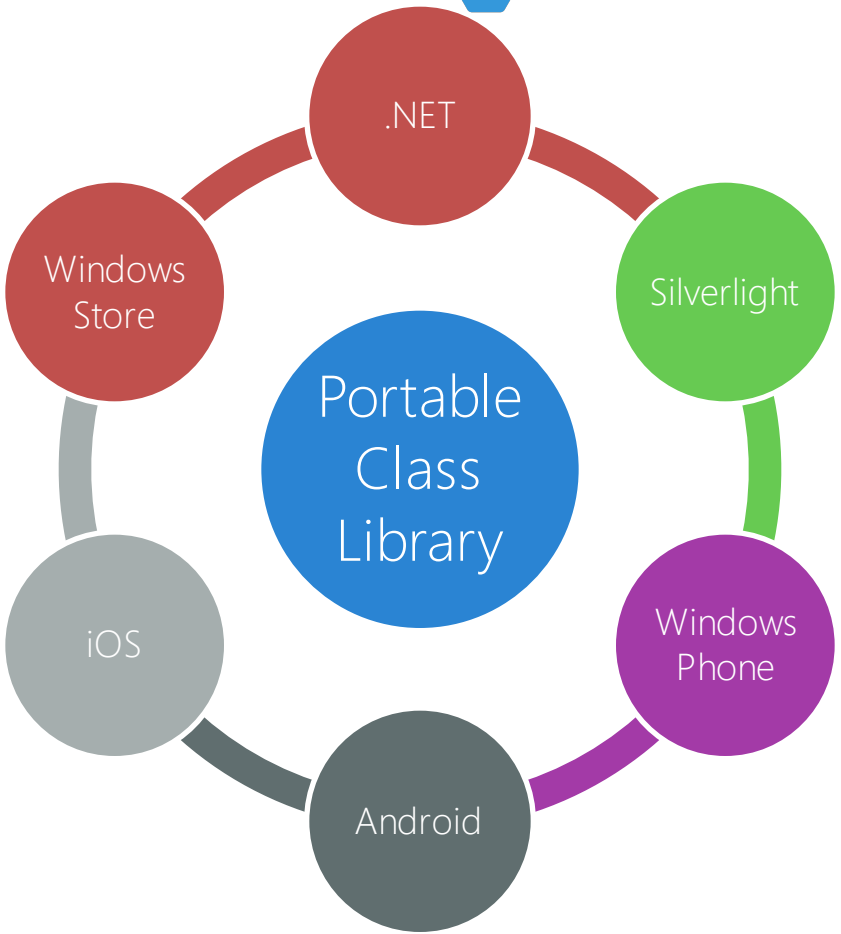
1. File Linking
2. Introduction to Shared Projects
3. Shared Project Internals
4. Platform Specific Code Strategies

Using Portable Class Libraries

Xamarin University

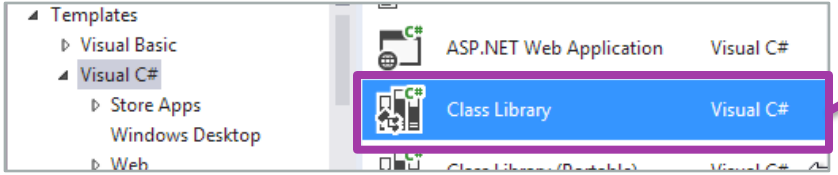
Tasks

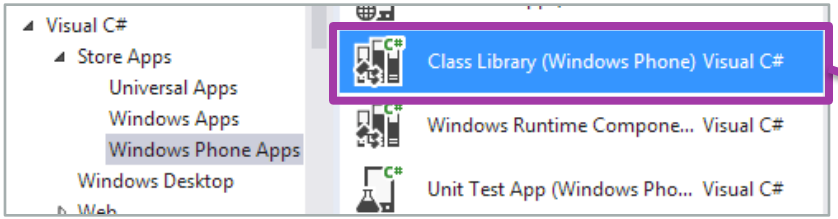
1. Portable Class Libraries
2. Profiles
3. Handling Platform Abstractions

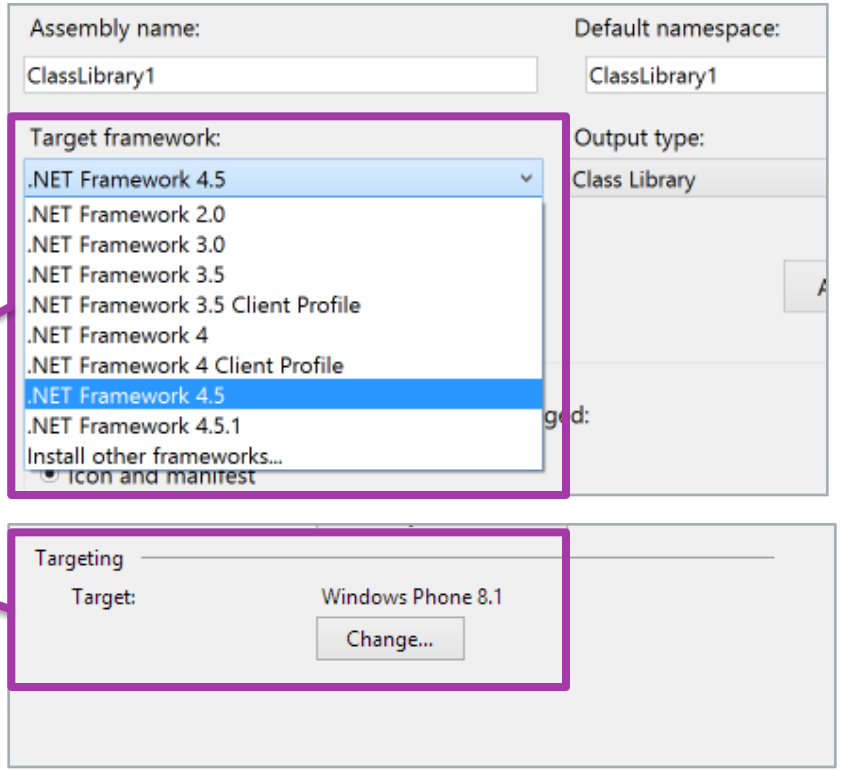


Class Library Projects

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

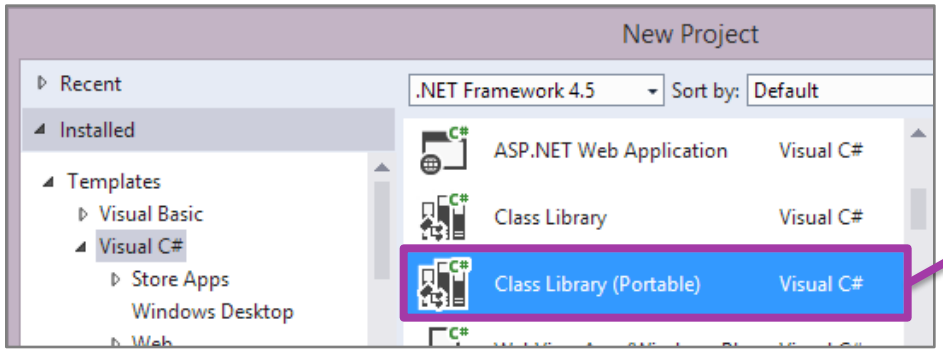


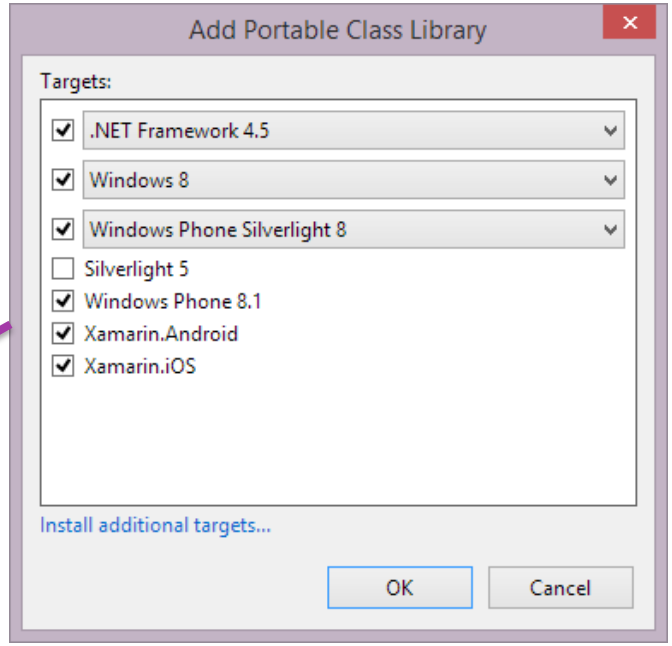




Portable Class Libraries (PCL)

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



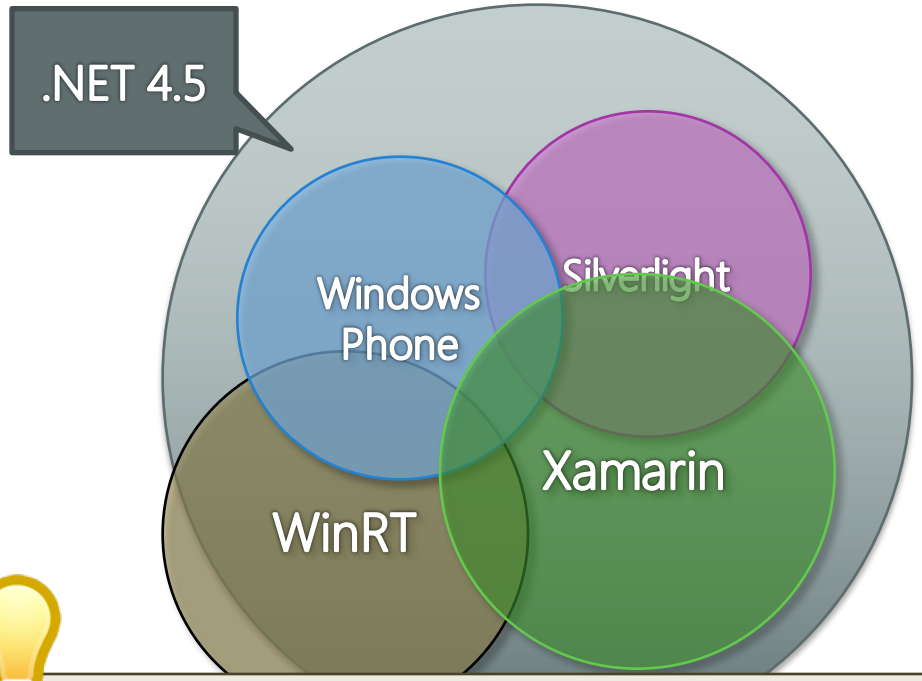


How does it work?

- 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

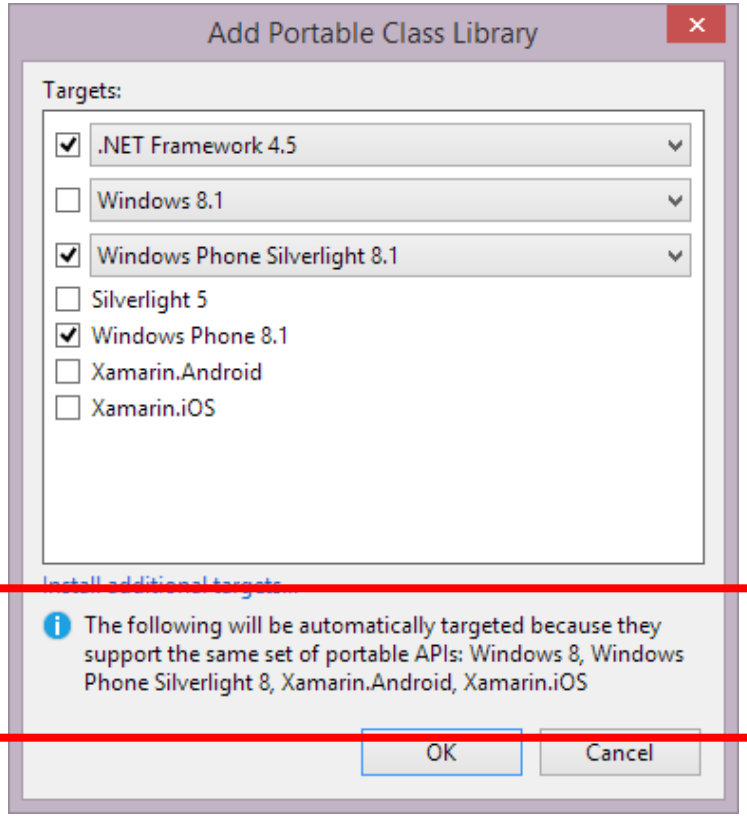
The available combinations are controlled by the profiles Microsoft has defined

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



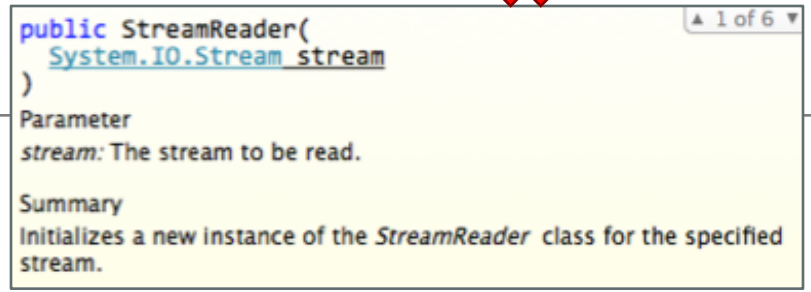
Demonstration

Creating a Portable Class Library

Platform Abstractions

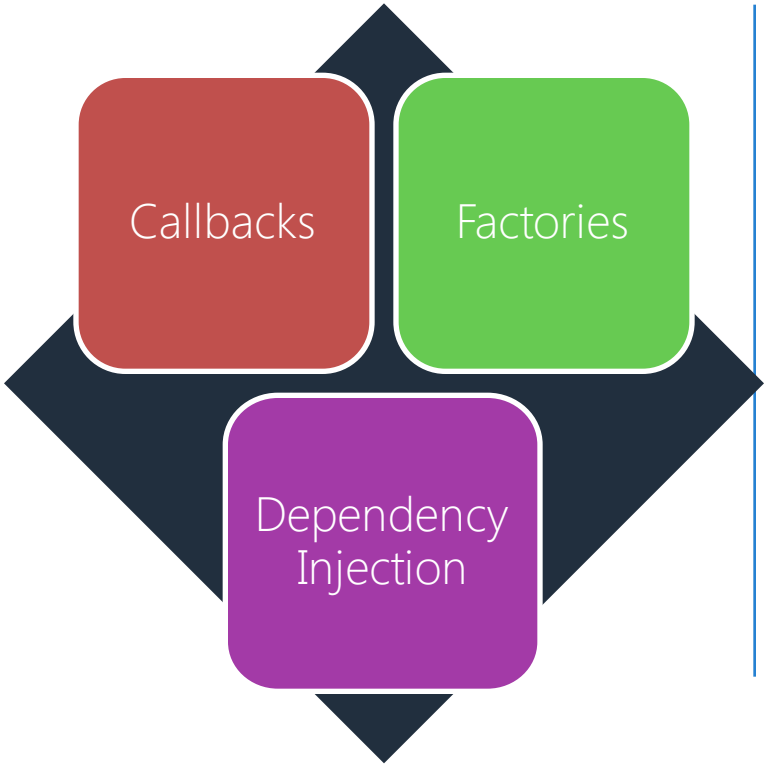
- PCLs are limited to features which are available on all target frameworks

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



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

Platform Specific Code Strategies



- Can restrict your API usage to the lowest common denominator, or use an abstraction such as an interface or an event and implement that abstraction in the platform-specific project(s)
- Need some way to get to the abstraction in your PCL code

Event-Based Extensibility

- PCLs can use events on types to request extensibility from the consumer, particularly effective if platform-specific requirements are small

```
public class NoteManager
{
   public event bool ShowAlert(string title,
                               string message);

   void OnDeleteNote(NoteItem note) {
      if (ShowAlert("Warning!", "...")) {
         ...
      }
   }
}
```

PCL

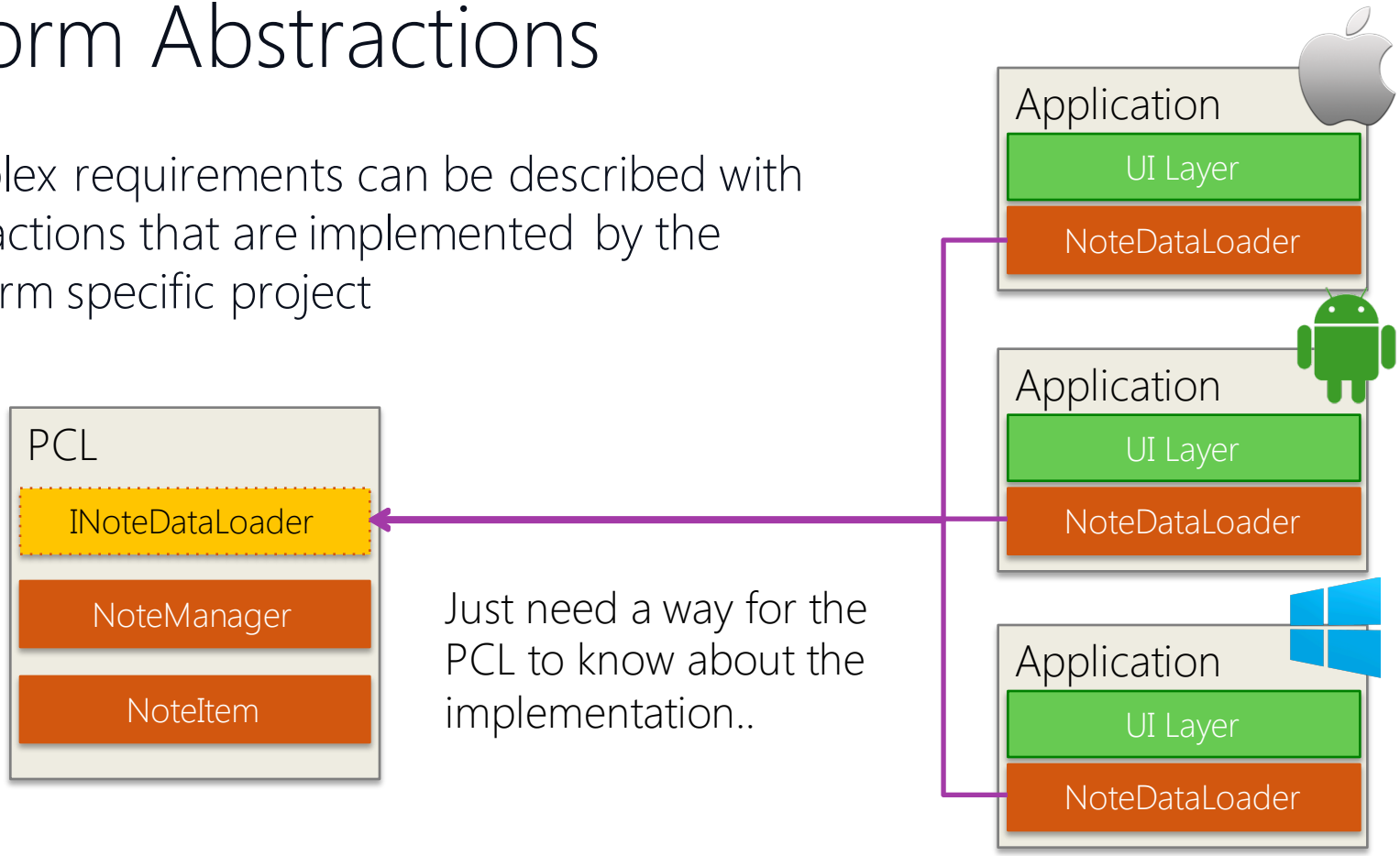
```
var notes = new NoteManager();

notes.ShowAlert += (title, message) => {
      new UIAlertView(title, message,
                      null, "OK").Show();
      ...
};
```

NoteManager.iOS

Platform Abstractions

- Complex requirements can be described with abstractions that are implemented by the platform specific project



Providing concrete dependencies

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

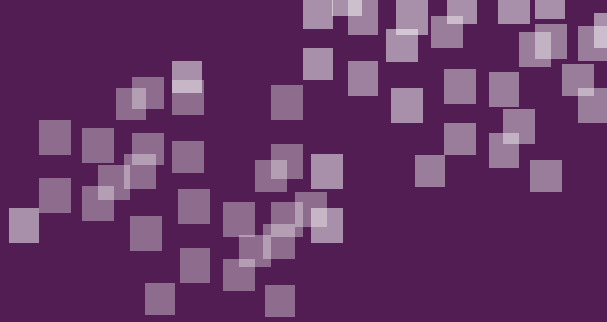
```
var manager = new NoteManager(new NoteDataLoader());
```

OR

```
var manager = new NoteManager ();
manager.Initialize(new NoteDataLoader ());
```

OR

```
var manager = new NoteManager ();
manager.Loader = new NoteDataLoader ();
```



Individual Exercise

Working with Portable Class Libraries

Xamarin University

Which one should I use?

Shared Projects

PROS	CONS
All APIs available	Can lead to spaghetti code
Platform-specific logic can be added directly	Cannot be unit tested on it's own
All file types can be shared	Must be shipped in source form

Portable Class 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)	Requires more work to integrate platform-specific code

Flash Quiz

Xamarin University

Flash Quiz

① Portable Class Libraries share source code across projects

a) True

b) False

Flash Quiz

① Portable Class Libraries share source code 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

③ Which of the following is not a platform supported by PCLs.

a) .NET 2.0
b) Windows Phone 8.0 (Silverlight)
c) Windows Phone 8.1
d) Windows Store Apps

Flash Quiz

③ Which of the following is not a platform supported by PCLs.

a) .NET 2.0
b) Windows Phone 8.0 (Silverlight)
c) Windows Phone 8.1
d) Windows Store Apps

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)

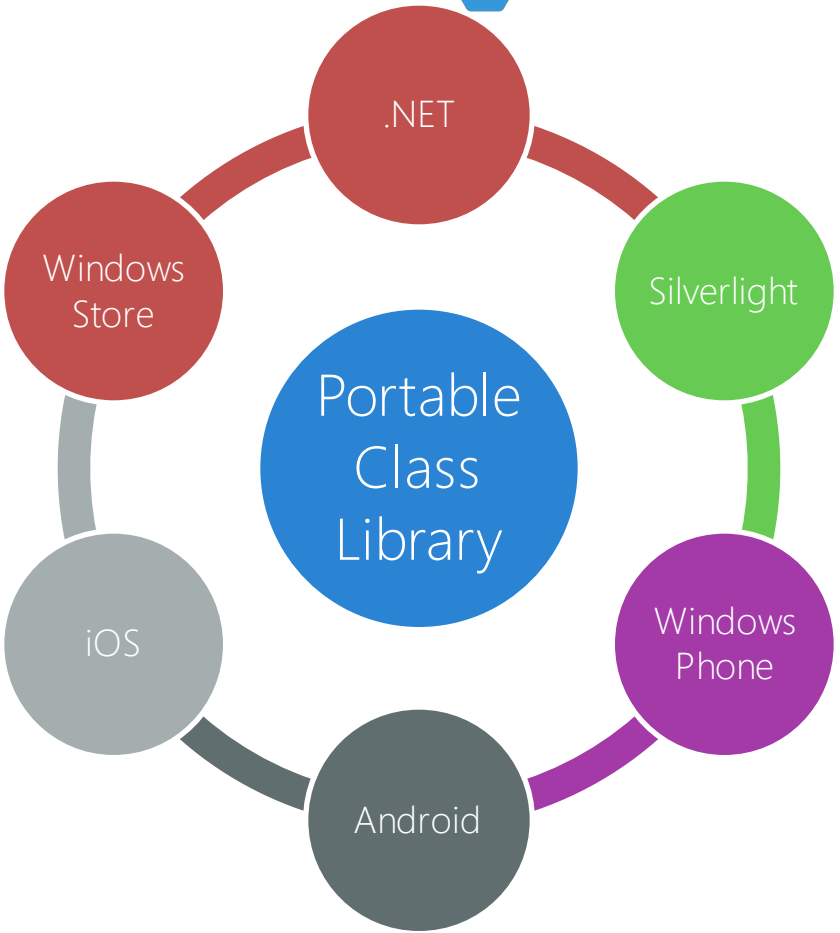
b) Service Locator

c) Publisher / Subscribe (events or messaging system)

d) Any of the above.

Summary

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



Thank You!

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