Cross-Platform Mobile Development with Xamarin
Wintellect Core Services

Consulting
Custom software application development and architecture

Instructor Led Training
Microsoft’s #1 training vendor for DevDiv for 14 years

On-Demand Training
World class, subscription based online training
INSTRUCTOR-LED TRAINING
Cross-Platform Mobile Development with Xamarin

- April 11 – 12
- Go to wintellect.com and click [ Training > Public Schedule ] from the menu
Agenda

- Mobile Landscape and Development Options
- Xamarin Development Platform
- Xamarin Application Architectural Options
- Building Shared Application Components
- Q & A
GitHub

- github.com/classroomCode/xamarinWebinar
Mobile Landscape

Worldwide Smartphone OS Market Share
(Share in Unit Shipments)

Source: IDC, Nov 2016
## Mobile Landscape

<table>
<thead>
<tr>
<th>Period</th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015Q4</td>
<td>79.6%</td>
<td>18.7%</td>
<td>1.2%</td>
<td>0.5%</td>
</tr>
<tr>
<td>2016Q1</td>
<td>83.5%</td>
<td>15.4%</td>
<td>0.8%</td>
<td>0.4%</td>
</tr>
<tr>
<td>2016Q2</td>
<td>87.6%</td>
<td>11.7%</td>
<td>0.4%</td>
<td>0.3%</td>
</tr>
<tr>
<td>2016Q3</td>
<td>86.8%</td>
<td>12.5%</td>
<td>0.3%</td>
<td>0.4%</td>
</tr>
</tbody>
</table>

Source: IDC, Nov 2016
Native Android Development

- Language
  - Java
- Development Tool
  - Android Studio
Native iOS Development

- Languages
  - Swift
  - Objective-C

- Development Tools
  - Xcode
  - AppCode (JetBrains)
UWP Development

- **Languages**
  - C#, VB, C++, JavaScript

- **Development Tools**
  - Visual Studio
Cross-Platform Development

- Web application
- Hybrid HTML-based cross-platform frameworks
  - Apache Cordova / Adobe PhoneGap
- Native cross-platform frameworks
  - Xamarin (C#)
  - React Native (JavaScript)
  - Appcelerator Titanium (JavaScript)
  - Qt (C++)
Cross-Platform Development (Pros)

- Ability to more easily reuse code for multiple platforms
- Leverage existing language/framework knowledge
Cross-Platform Development (Cons)

- GUI components may still need to be coded multiple times to obtain a platform-specific look and feel
  - Different platform design guidelines
- May not have access to all native functionality
- New native features may not be available right away
- Performance and offline functionality
Xamarin Development Platform

- Xamarin founded in 2011
  - Took over stewardship of Mono
- Released Xamarin Studio in 2013
- Acquired by Microsoft in 2016
Xamarin Platform Components

- Xamarin platform
  - Makes it possible to build native Android, iOS, macOS, and Windows applications with C# (and F#)

- Xamarin.Forms
  - Cross-Platform UI toolkit

- Xamarin Studio
  - Transitioning to Visual Studio and Visual Studio for Mac

- Xamarin Test Cloud
Xamarin on iOS

- Xamarin.iOS exposes Apple's Cocoa Touch frameworks as namespaces you can reference from C#
- Apple does not allow runtime code generation on iOS
- C# is ahead-of-time (AOT) compiled to ARM assembly language
- Framework components are embedded into your application
- Apple's tools are required to run an iOS application
Xamarin on Android

- Xamarin.Android exposes Google's Android SDK as namespaces you can reference from C#
- Android does not have the code generation limitations as iOS
- C# is compiled to IL and packaged with the Mono runtime
- Interactions with native types are accomplished via JNI
Xamarin on Windows

- C# is compiled to IL and uses the built-in runtime
- UWP has a .NET Native option which is similar to the AOT compilation used for iOS
Sharing Code

- "Business logic, database usage, network access, and other common functions can be written once and re-used on each platform, providing a foundation for platform-specific user interfaces that look and perform as native applications."

- Xamarin Documentation
Sharing Code

- To share code across native projects, you can use a ...
  - Shared project
  - Portable class library (PCL)
  - .NET Standard library

- PCLs are being phased out in favor of .NET Standard libraries
Shared Projects

- Code within a shared project is copied and compiled into each platform-specific application

```plaintext
^ #if PLATFORM compiler directives
```

![Diagram showing shared project structure with layers for iOS, Android, and Windows Phone applications.]
Shared Projects

- Different techniques can be used to handle platform-specific functionality when using a shard project
  - Conditional compilation
  - Class mirroring
  - Partial classes and methods
Conditional Compilation

- Each platform-specific project can define compile-time symbols
- Can be used to specify if which code should be included

Define Symbols: __MOBILE__; __IOS__; DEBUG;
public static string DatabaseFile
{
    get {
        var filename = "data.db";
        #if __IOS__
            var docPath = Environment.GetFolderPath(SpecialFolder.Person);
            var path = Path.Combine(docPath, "..", "Library", filename);
        #elif __ANDROID__
        #else
        else
            var path = Path.Combine(Environment.GetFolderPath(SpecialFolder.Personal, filename));
        #endif
        return path;
    }
}
Partial Classes and Methods

- By using the partial class feature of .NET, part of a class can be defined in the shared project with another part of the class defined in each platform-specific project.
- Partial methods can also be used to achieve something similar to the template method design pattern.
Portable Class Library (PCL)

- When creating a PCL project, you select a profile that specifies the target platforms.
- The set of .NET APIs available to use is an intersection of what is supported on each platform.
- Many NuGet packages are distributed as PCLs.
Portable Class Library (PCL)

- It is a common practice to use dependency injection when sharing code via a PCL
  - Define an interface within the PCL
  - Code to that interface within the PCL
- Each platform-specific project can provide the PCL with an object that implements the required interface
.NET Standard Library

- PCLs are being replaced by libraries that target .NET Standard
- .NET Standard is a set of APIs that all .NET platforms have to implement
- Unifies the .NET platform and prevents future fragmentation
.NET Standard

**.NET FRAMEWORK**
- WPF
- Windows Forms
- ASP.NET

**.NET CORE**
- UWP
- ASP.NET Core

**XAMARIN**
- iOS
- OS X
- Android
- Mono Class Library

**COMMON INFRASTRUCTURE**
- Compilers
- Languages
- Runtime components
.NET Standard

.NET FRAMEWORK
- WPF
- Windows Forms
- ASP.NET

.NET CORE
- UWP
- ASP.NET Core

XAMARIN
- iOS
- OS X
- Android

.NET STANDARD LIBRARY
One library to rule them all

COMMON INFRASTRUCTURE
- Compilers
- Languages
- Runtime components
## .NET Standard

<table>
<thead>
<tr>
<th>Platform Name</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Standard</td>
<td>netstandard</td>
</tr>
<tr>
<td>.NET Core</td>
<td>netcoreapp</td>
</tr>
<tr>
<td>.NET Framework</td>
<td>net</td>
</tr>
<tr>
<td>Mono/Xamarin Platforms</td>
<td></td>
</tr>
<tr>
<td>Universal Windows Platform</td>
<td>uap</td>
</tr>
<tr>
<td>Windows</td>
<td>win</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>wpa</td>
</tr>
<tr>
<td>Windows Phone Silverlight</td>
<td>wp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
<th>1.5</th>
<th>1.6</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Standard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.NET Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.NET Framework</td>
<td></td>
<td>4.5</td>
<td>4.5.1</td>
<td>4.6</td>
<td>4.6.1</td>
<td>4.6.2</td>
<td>vNext</td>
<td>4.6.1</td>
</tr>
<tr>
<td>Mono/Xamarin Platforms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Universal Windows Platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10.0</td>
<td></td>
<td></td>
<td>vNext</td>
</tr>
<tr>
<td>Windows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.0</td>
<td>8.1</td>
<td></td>
</tr>
<tr>
<td>Windows Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.1</td>
</tr>
<tr>
<td>Windows Phone Silverlight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Xamarin.Forms

- Xamarin.Forms is a cross-platform UI framework
- Separate platform-specific projects are still required but the UI can now be defined in a single shared project
- Provides a library of common controls that create native controls at runtime
- Supports the use of XAML for defining the UI
  - Implementation of the same standard used by WPF
Xamarin.Forms

- Xamarin.Forms project contains a single instance of a class that inherits from a class named Application
  - Provides lifecycle and navigation events
- Application object created by the platform-specific project
  - iOS: FinishedLaunching method in AppDelegate.cs
  - Android: OnCreate method in MainActivity.cs
  - Windows: OnLaunched method in App.xaml.cs
Xamarin.Forms

• The UI of a Xamarin.Forms-based application consists of a collection of pages

• MainPage property of the Application object determines the first page to be displayed

• Several different subclasses of the Page class exist
  • ContentPage, NavigationPage, TabbedPage, MasterDetailPage, CarouselPage
Xamarin.Forms

- Content pages contain views
- View can be a control or contain other controls
- Each control has a renderer that produces the appropriate native control at runtime
The Device class can be used to execute different code or provide a different value at runtime based on the underlying platform.

```csharp
Device.OnPlatform(
    iOS: () => { ... },
    Android: () => { ... },
    WinPhone: () => { ... },
    Default: () => { ... });

var greeting = Device.OnPlatform<string>(
    iOS: "Hi", Android: "Yo", WinPhone: "Hello");

new Thickness(5, Device.OnPlatform(20, 0, 0), 5, 5);
```
Device.OnPlatform can also be used in XAML

```xml
<StackLayout.BackgroundColor>
  <OnPlatform x:TypeArguments="Color"
              iOS="Silver"
              Android="Green"
              WinPhone="Blue" />
</StackLayout.BackgroundColor>
```
Conclusion

- Recording of this webinar
- github.com/classroomCode/xamarinWebinar
- Two-day instructor-led training (April 10-12)
- jason.bell@treeloop.com

Q & A