Xamarin University

Introduction to Mobile Development

Lecture will begin soon

Please log into your lab environment using the instructions provided via email

Please join in the conversation as we wait for the course to begin

Go Mobile Training

- Live online classes
- Self-paced tutorials
- 2x one-on-one sessions
- Private community forums

Certification Process

- Xamarin Certified iOS Developer
- Xamarin Certified Android Developer
- Xamarin Certified Mobile Developer

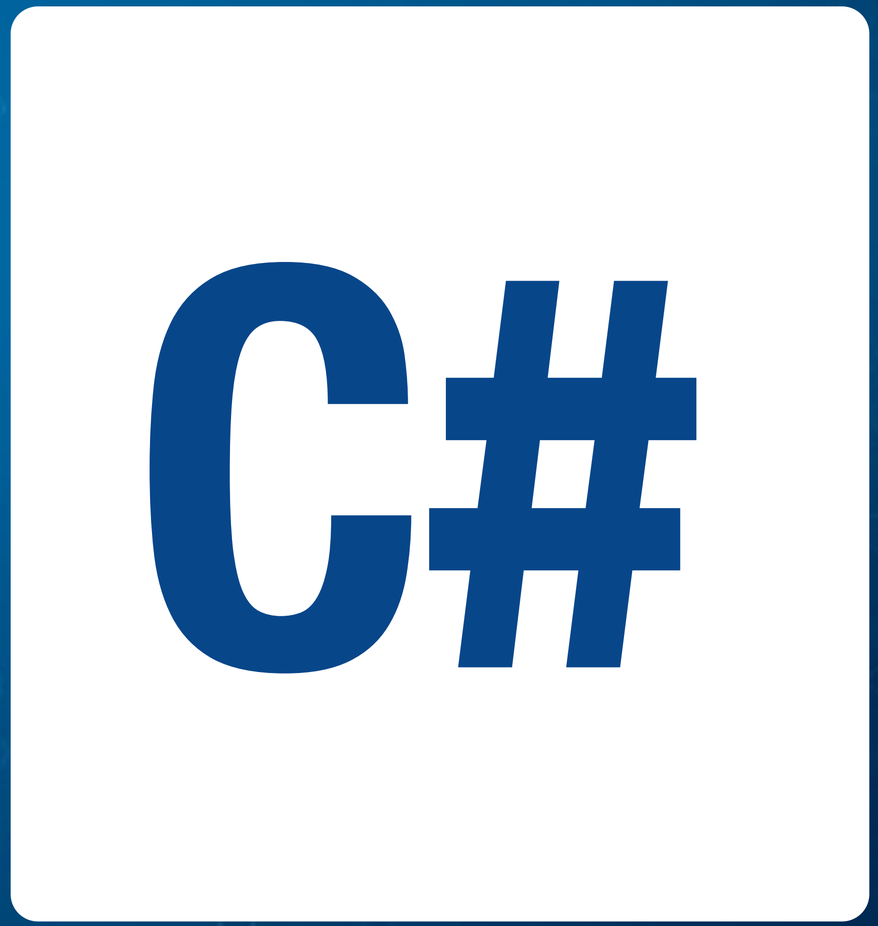
- Good for one year, renewable
- Verifiable online by clients

Xamarin Platform

One Language, One Framework

- Generics
- LINQ
- Lambdas
- Async / Task Parallel Libraries

- Type safety
- Garbage Collection



Designed for Performance

- Apps compiled to native code
- Optimizing compiler / linker
- Supports all platform APIs
- Uses native UI controls

Code Reuse

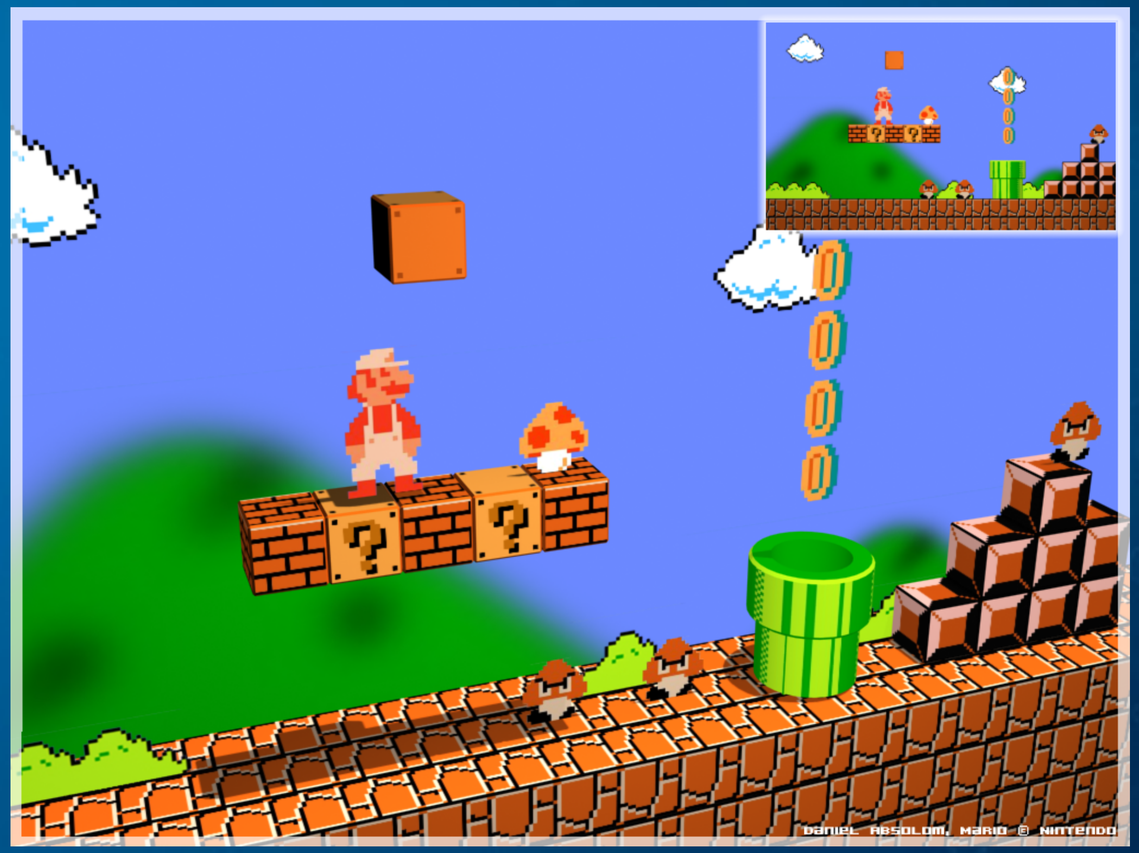
- Share C# code across platforms
  - Use existing C# libraries

- Native bindings projects
  - Obj-C bindings
  - .jar bindings

- C via P/Invoke, C++ via CPPSharp (github.com/mono/CppSharp)

Other Platforms

- Windows - via .NET
- Mac - via Xamarin.Mac
- Linux - via Mono
- Ouya, XBox, PSP, etc.

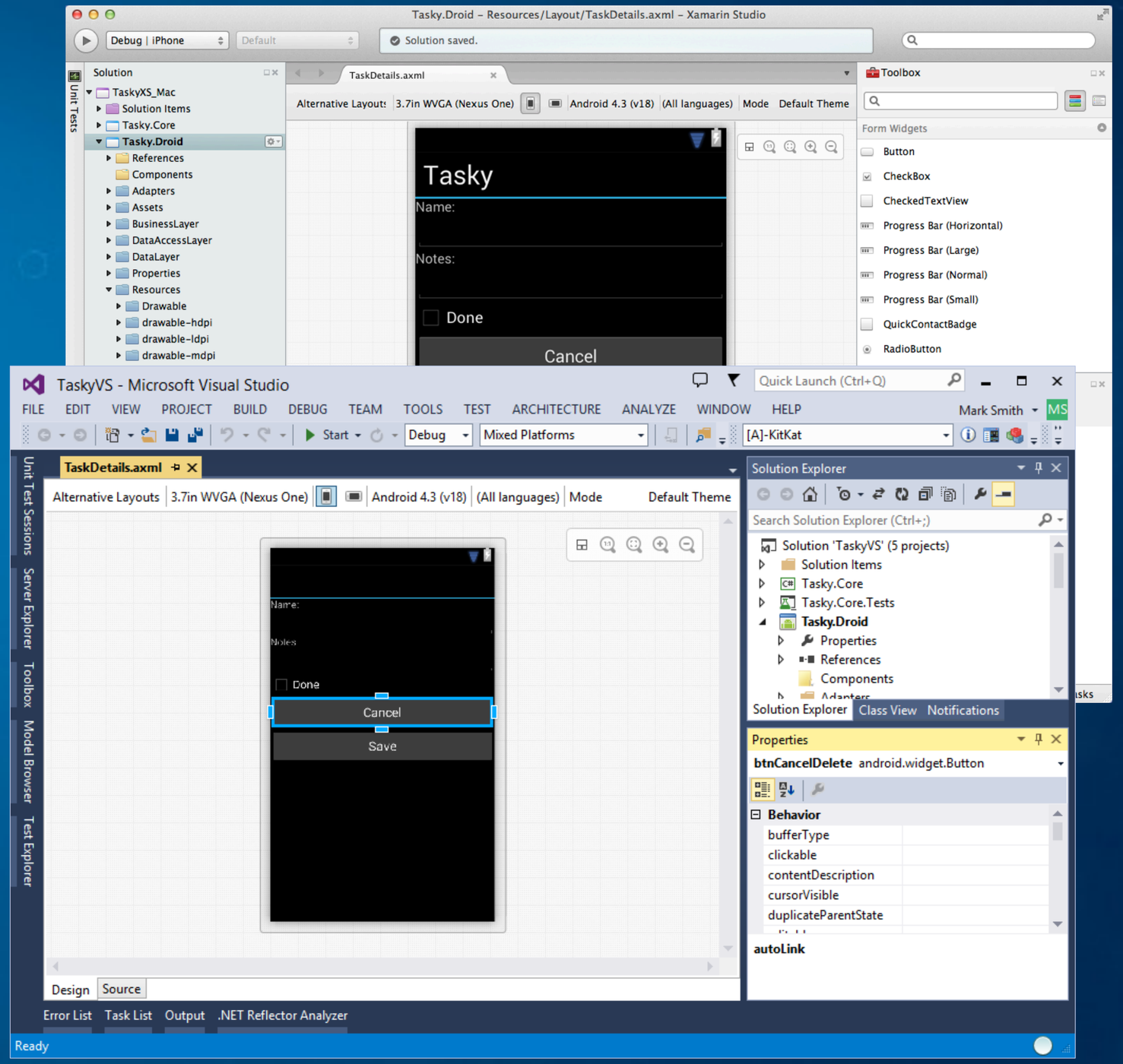


Xamarin Community

- Public forums
  - http://forums.xamarin.com
- Huge C# Community
  - http://stackoverflow.com
- Xamarin Support

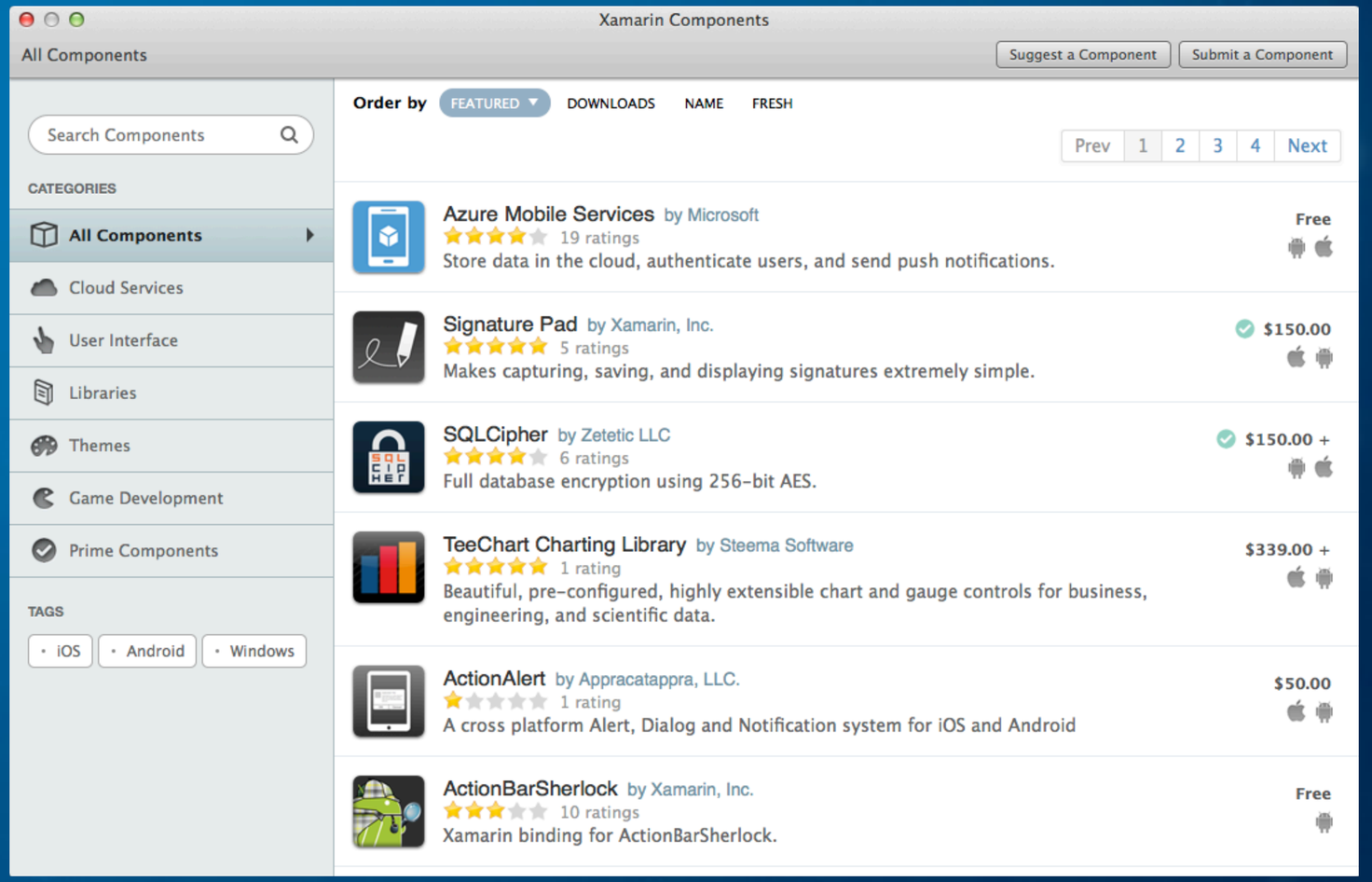
Choice of IDEs

- Visual Studio
  or Xamarin Studio
- Mac or PC
- Android + iOS designer



Xamarin Component Store

- Cross-platform components
- Search and add to project right from the IDE
- Revenue model for user-submitted components



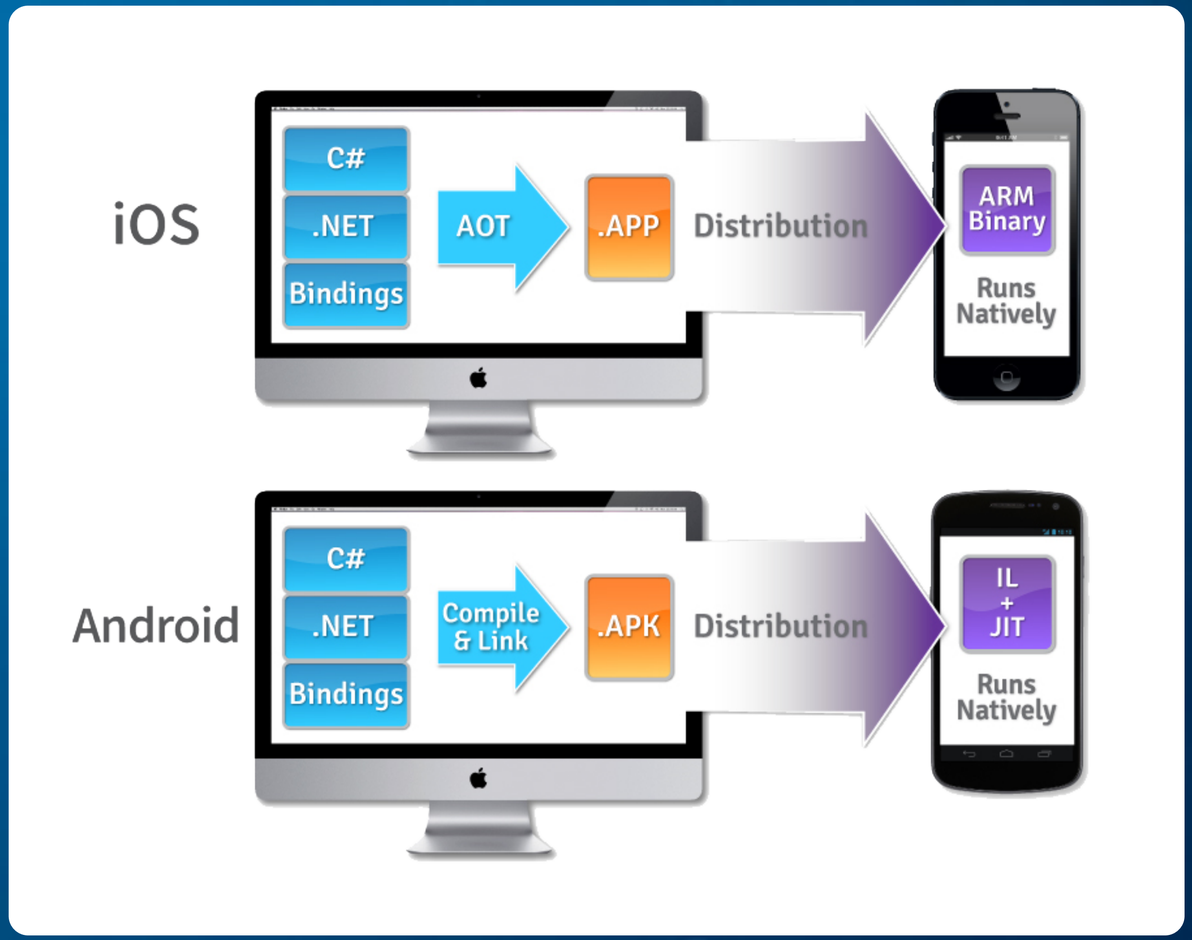
Demo: Tasky Pro

How Xamarin Works

Build + Execution Model

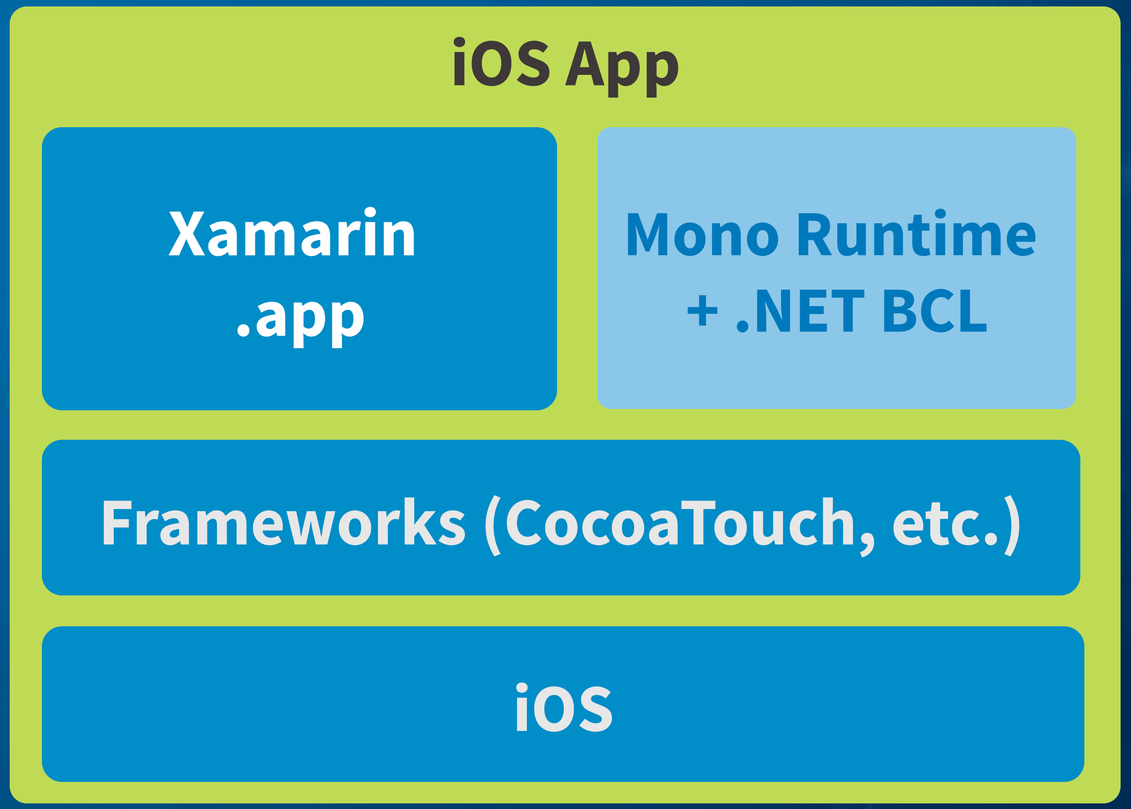
Xamarin apps are native on both iOS & Android but process differs slightly

- iOS: Ahead-of-Time (AOT) compiler turns code into ARMvX binary .app file
- Android: Mono compiler creates native .apk file that contains IL that gets JIT'd at runtime



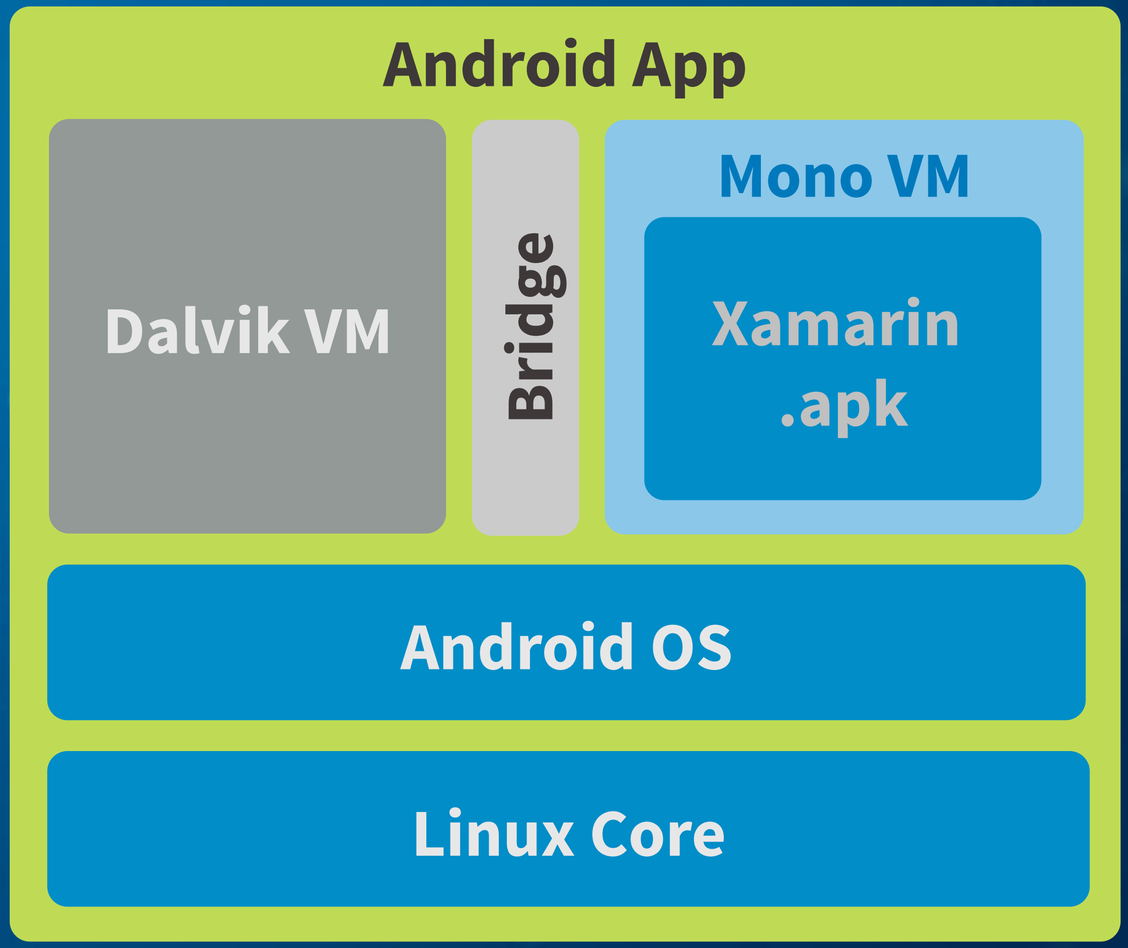
App Runtime Model: iOS

- Native ARMvX code – no JIT used
- Mono Runtime provides system services
- App has full access to iOS frameworks



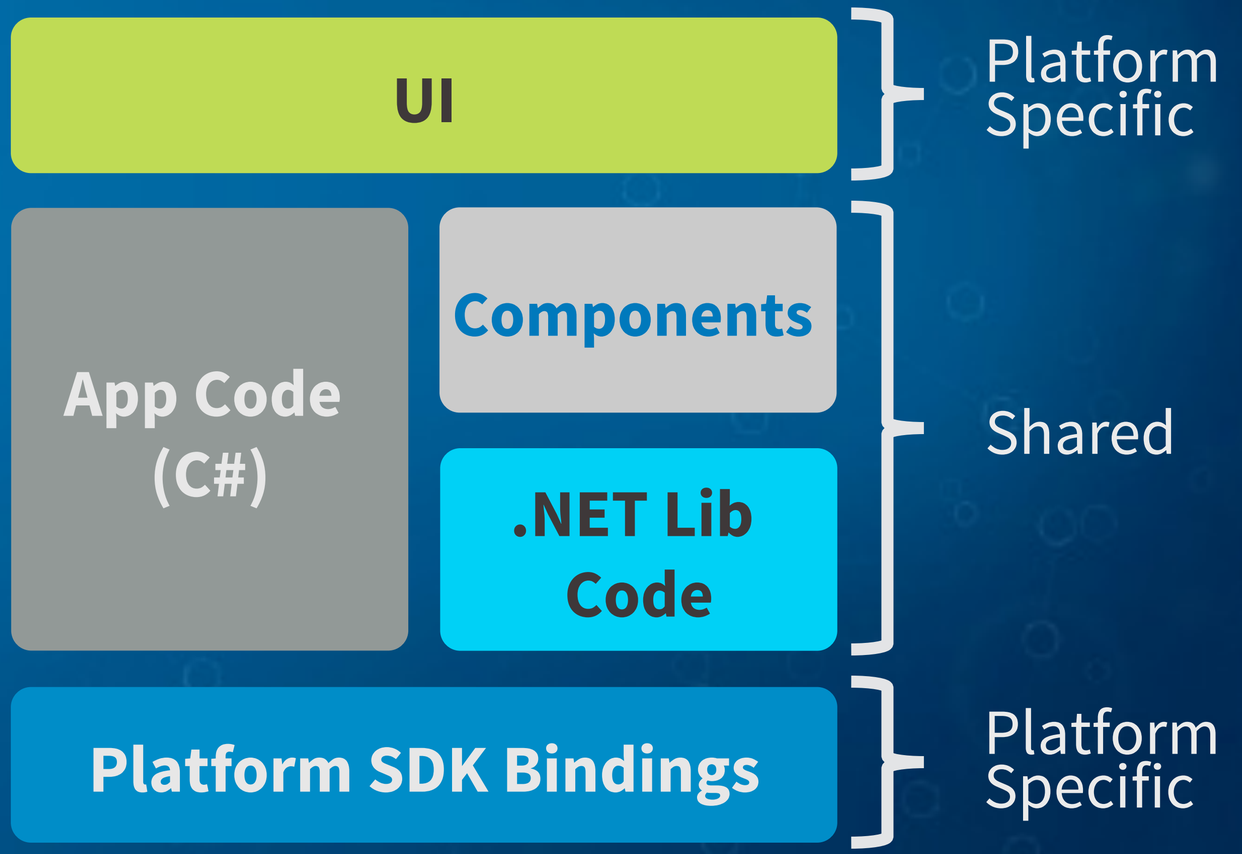
App Runtime Model: Android

- Mono VM + Dalvik execute side-by-side
- Mono VM JITs IL into native code and executes most of your code
- Interop directly with Android OS + Dalvik



Code Sharing Across Platforms

- Same app structure across all platforms
  - UI is native & platform-specific
  - App code is C# and potentially sharable
- Not a "Write-Once-Run-Anywhere" solution but can have up to 95% shared code, depending on patterns used



Platform Comparisons

App Package	**.app**	**.apk**	**.xap**
typeof(Screen)	**Controller**	**Activity**	**Page**
typeof(Control)	**View**	**Widget**	**Control**
UI Files (xml)	**.storyboard**	**.axml**	**.xaml**
UI Pattern	**MVC**	**MVC**	**MVVM**

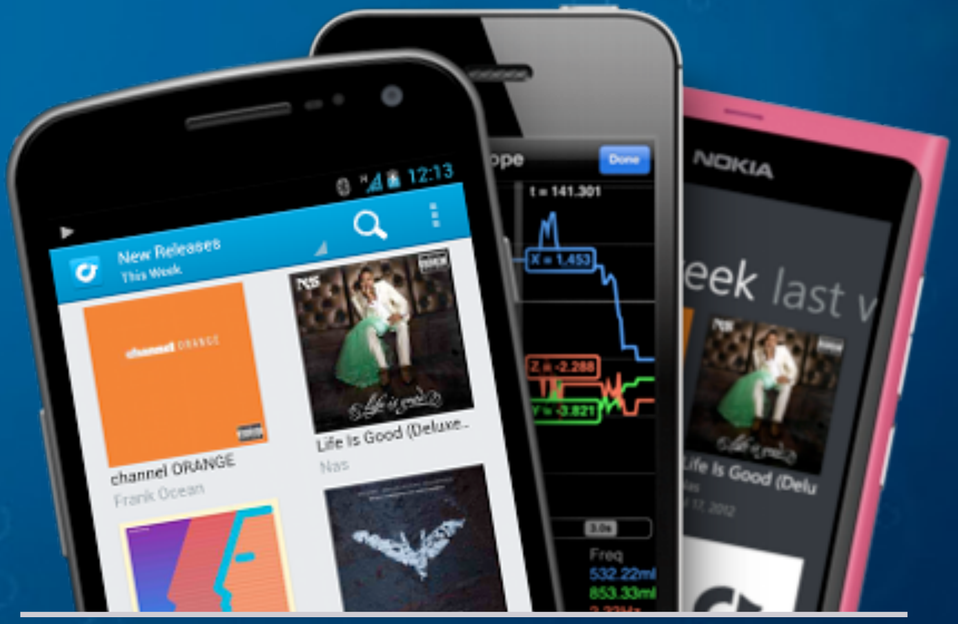
Distribution

- All major vendors have public distribution model
- iOS: App Store, Enterprise, Ad-Hoc
- Android: Google Play, Amazon Marketplace, .apk distribution
- Windows: Windows Phone Store

Building Cross-Platform Mobile Apps

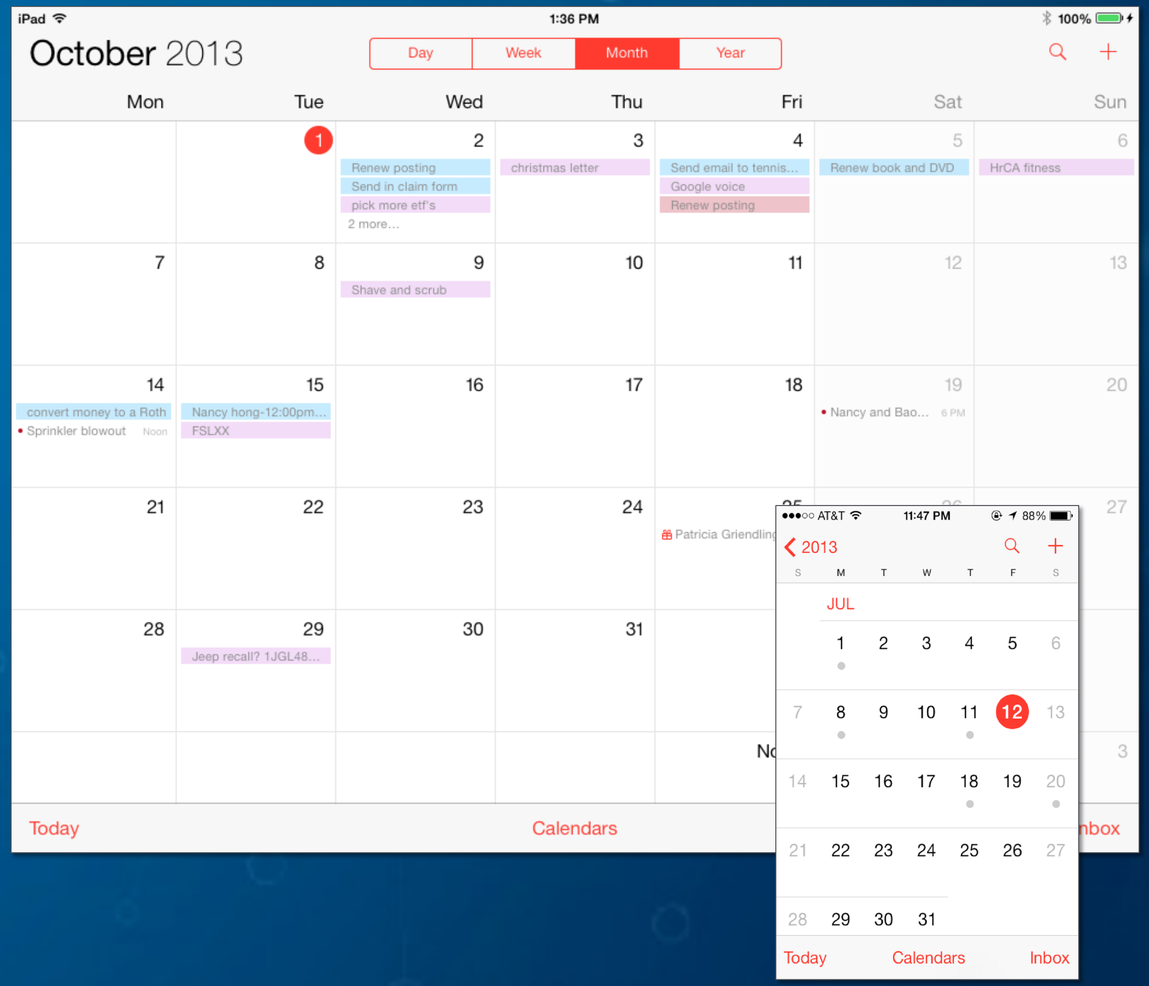
Design Platform-Specific UI

- UI in iOS != Android != Windows Phone
- Metaphors are different:
  - Navigation Controller vs. Back Button
- Users expect native UI



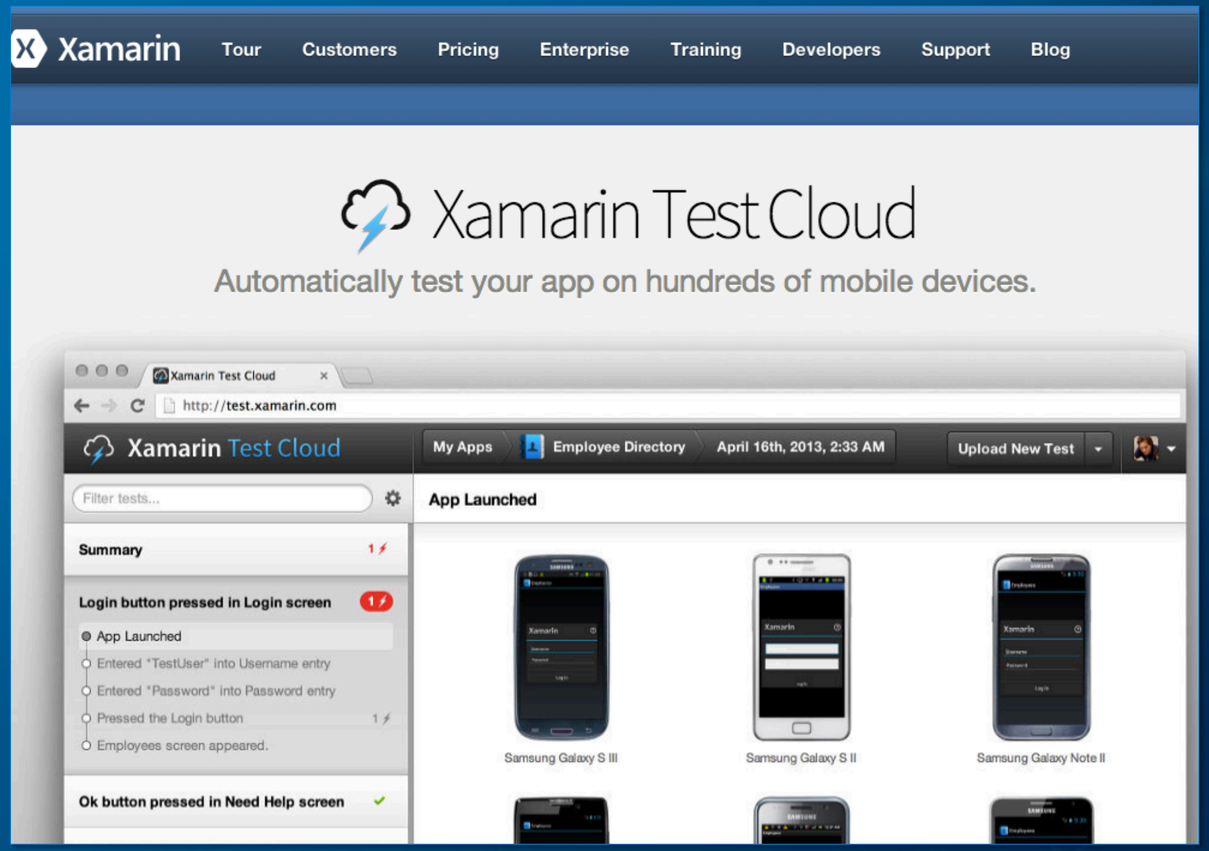
Choose Device Targets

- Phone != Tablet
- Beware Android fragmentation
- Form factor + capability differences
- Try to minimize targets



Test on All Targeted Devices

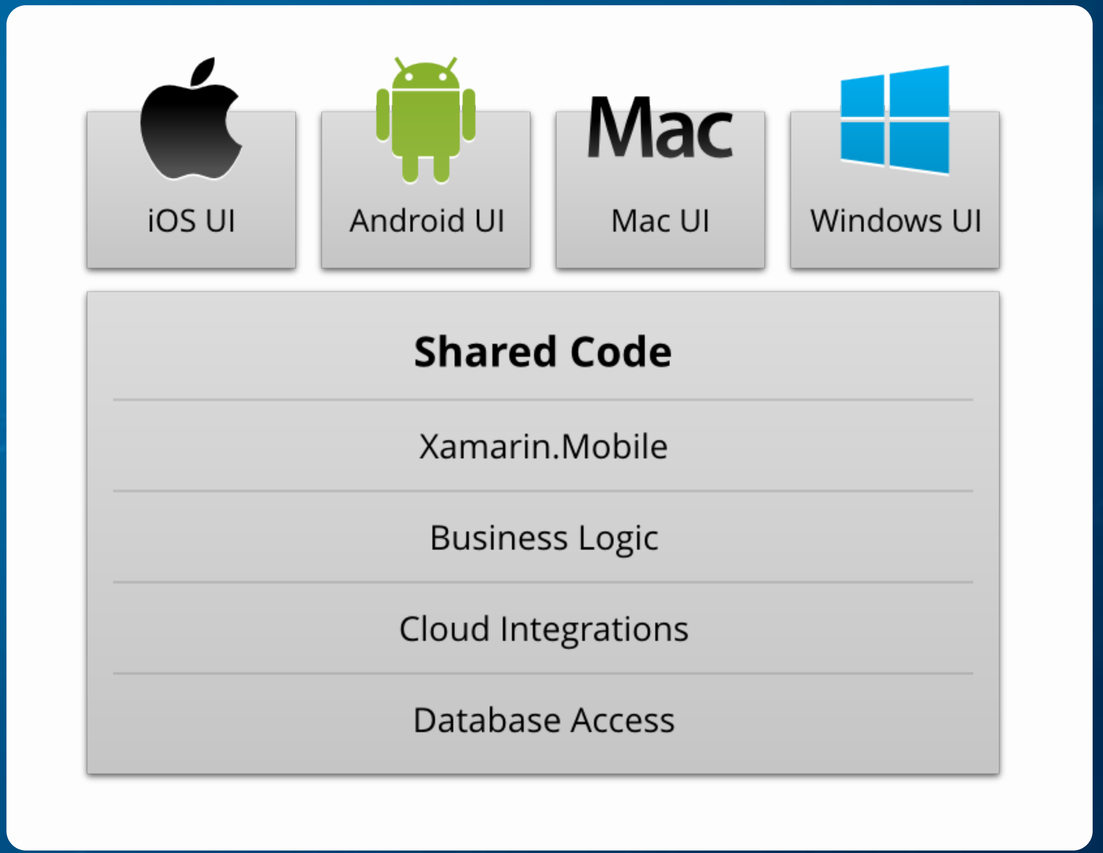
- Device != Emulator/Simulator
- Xamarin TestCloud
  - Hundreds of actual devices
  - Supports CI workflow



http://xamarin.com/test-cloud

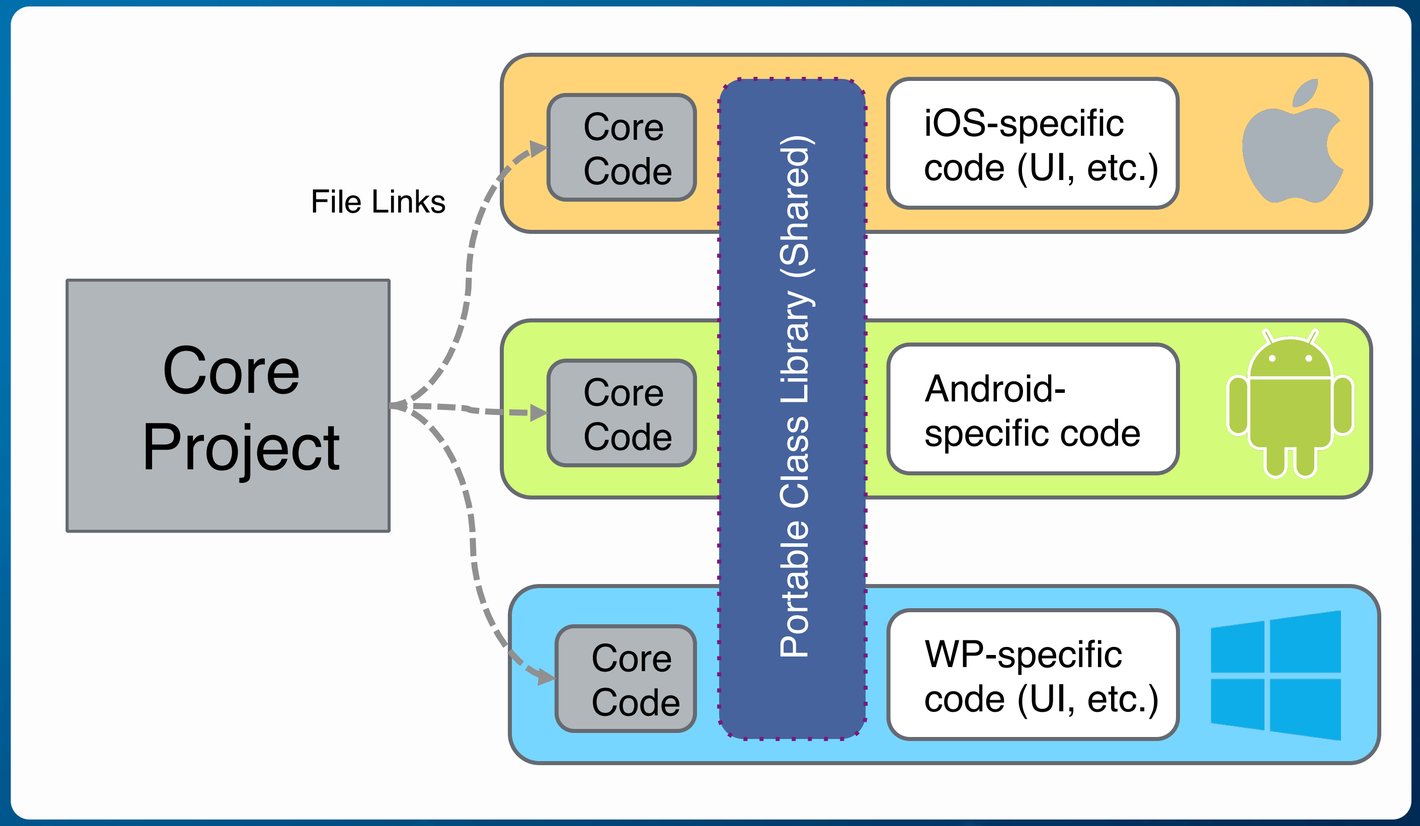
Architecture

- Use Layers
  - Separation of Responsibility
- Use Encapsulation
  - Enables core code sharing
- Same architectural patterns used in other software projects



Sharing Code Across Projects

- Portable Class Libraries – great for components
- File Linking – share code at the source level
- Use multiple projects



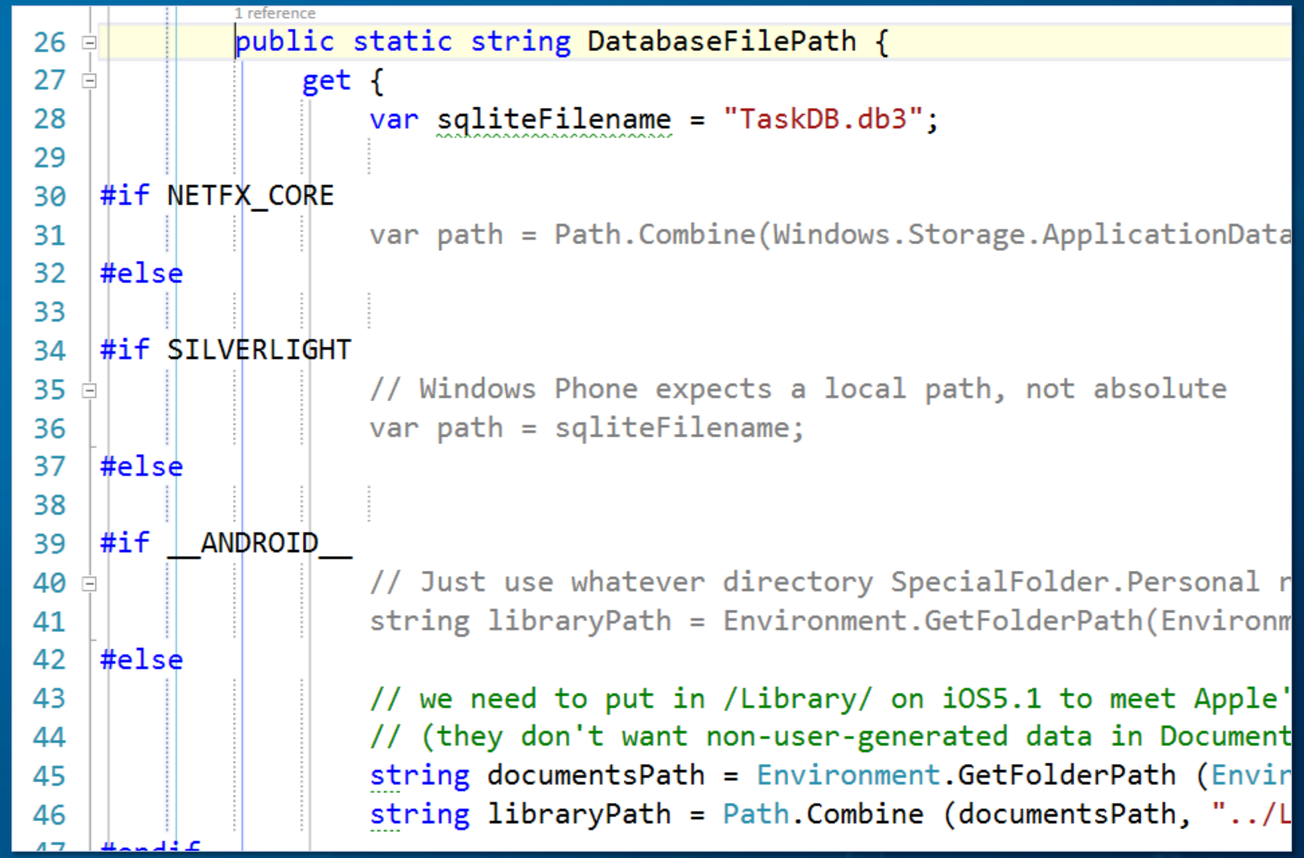
Demo: Creating a Cross Platform Mobile Solution

Handling Platform Differences

File Linking + Conditional Compilation

- Pre-Defined Symbols:

```
#if __MOBILE__
#if __ANDROID__
#if __IOS__
#if WINDOWS_PHONE
#if SILVERLIGHT
```



- Can add custom symbols in build settings

Architectural Abstraction

- Use Interfaces and provide different implementation based on platform

- Can abstract with patterns
  - Dependency Injection
  - Inversion of Control (IoC)
  - Service Provider

Xamarin.* Libraries

- Cross-Platform API Abstractions
- Open Source
- `github.com/Xamarin/Xamarin.*`

Mobile **Social** **Auth**

Sharing the Data Access Layer

- SQLite available for iOS + Android
  - Can use ADO.NET
- C#-SQLite for Windows
  - `code.google.com/p/csharp-sqlite/`
- SQLite.NET ORM
  - available in the Component Store

Sharing the Web Services Layer

- Use HttpClient for REST services
  - also supports WebClient or HttpWebRequest
- Supports WCF services
  - w/ BasicHttpBinding
- Legacy SOAP services
  - .asmx compatibility

Further Reading

http://docs.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/

Xamarin University

Introduction to Mobile Development

Thank You

Please complete the class survey:

`https://www.surveymonkey.com/s/7RBSQS9`