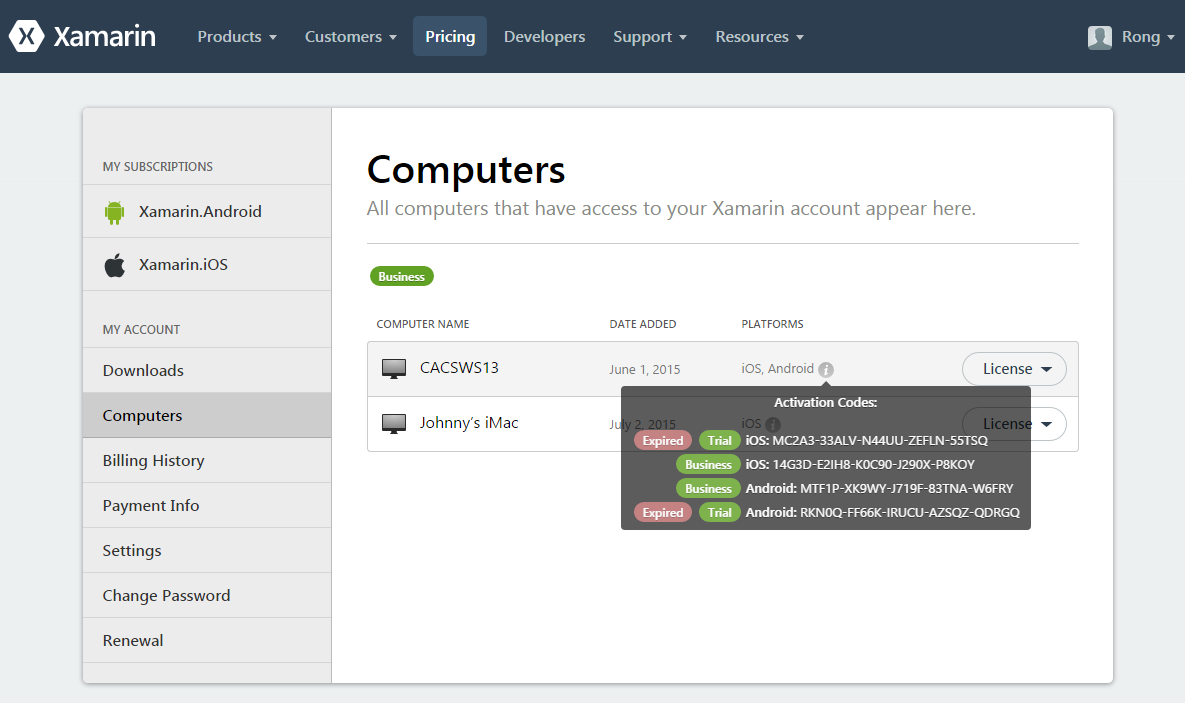
Research on Xamarin

By Rong Zhuang

August 19, 2015

1. Introduction
   1. Main Features

* Cross-Platform(C#)
* iOS(Xamarin.iOS)
* Android(Xamarin.Android)
* Mac(Xamarin.Mac)
* Xamarin Insights
* Xamarin Test Cloud(Xamarin.UITest)
  1. IDE
* Mac: Xamarin Studio, Xcode
* Windows: Xamarin Studio or Visual Studio+Xamarin’s plug-in for Visual Studio, Mac, Xcode, Xamarin.iOS Build Host
  1. Xamarin Account(per developer, per device platform)
* Trail(30days)-free
* Indie-$25/Month($300/Year)
* Business-$83/Month($999/Year)
* Enterprise-$158/Month($1899/Year)
* For Student-Free(Need to apply)



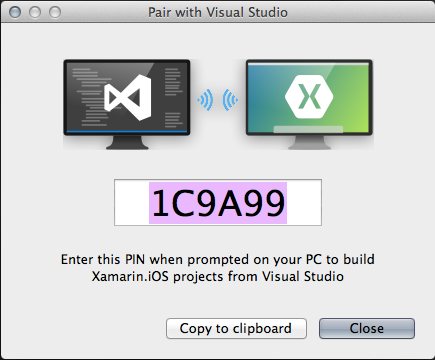
1. iOS Development

There are two options to setup the development environment for iOS:

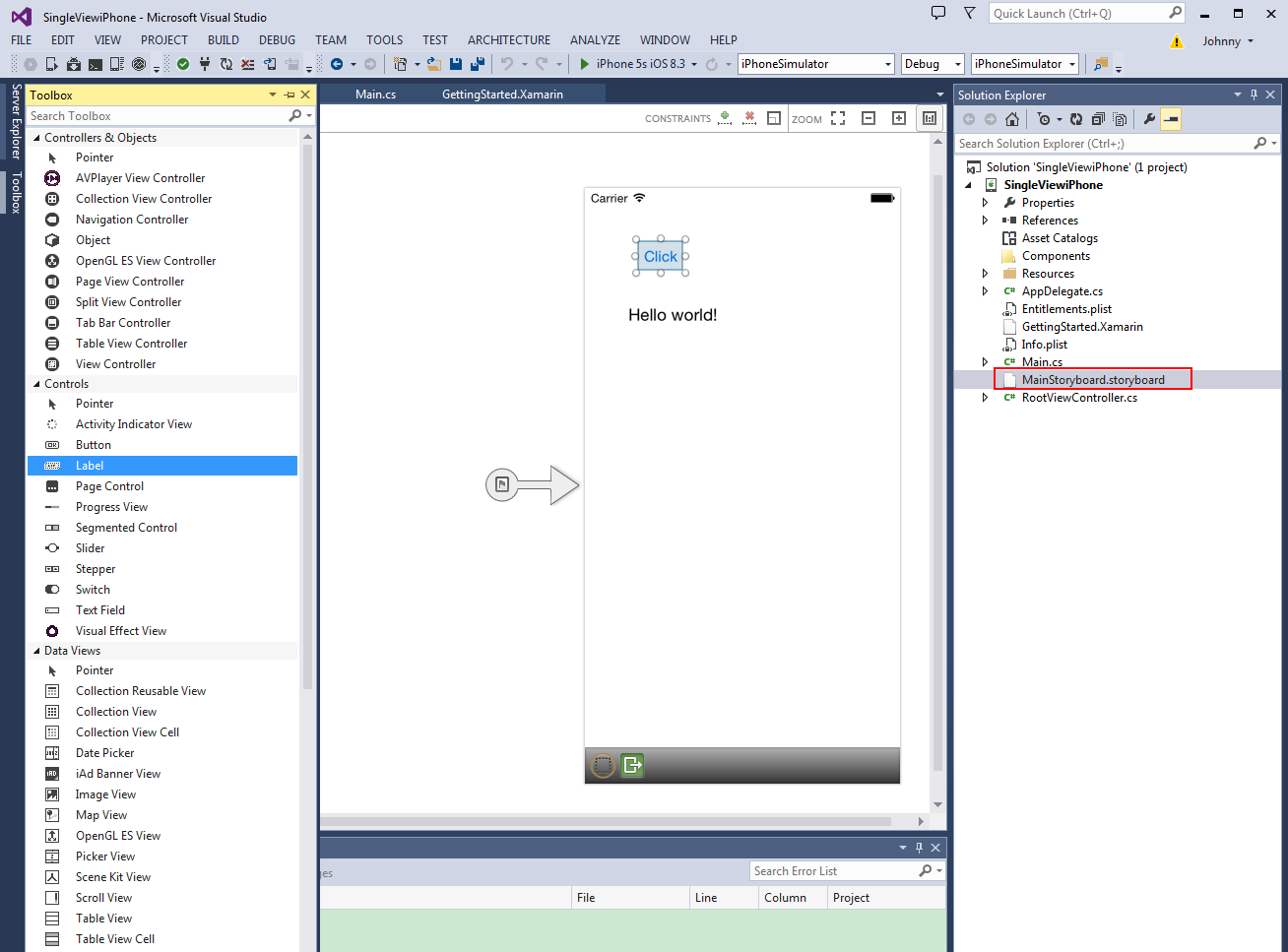
* Xamarin Studio on Mac
* Visual Studio on PC(with another mac connected to the same network of PC), see below.

*Note: Xamarin Studio on PC cannot be used for iOS development.*

* 1. Installation(Windows)
* Xamarin Platform for Windows
* Xamarin.iOS Build Host on Mac

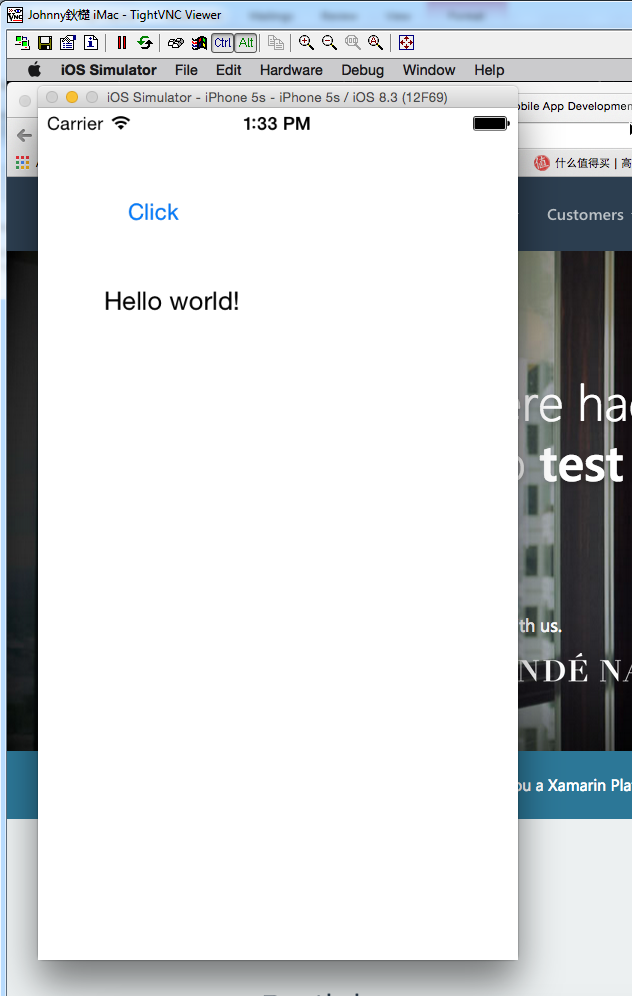


* 1. Developing in Visual Studio 2013

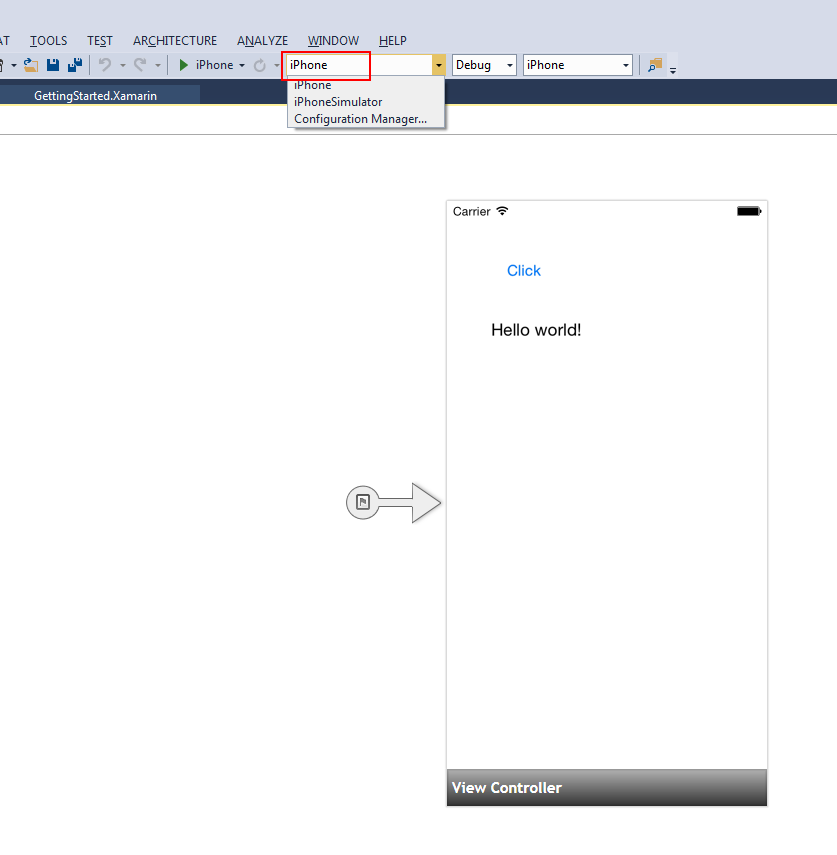


Note: Must be connecting to Build Host of Mac when editing the layout.

* 1. Test in simulator on Mac.

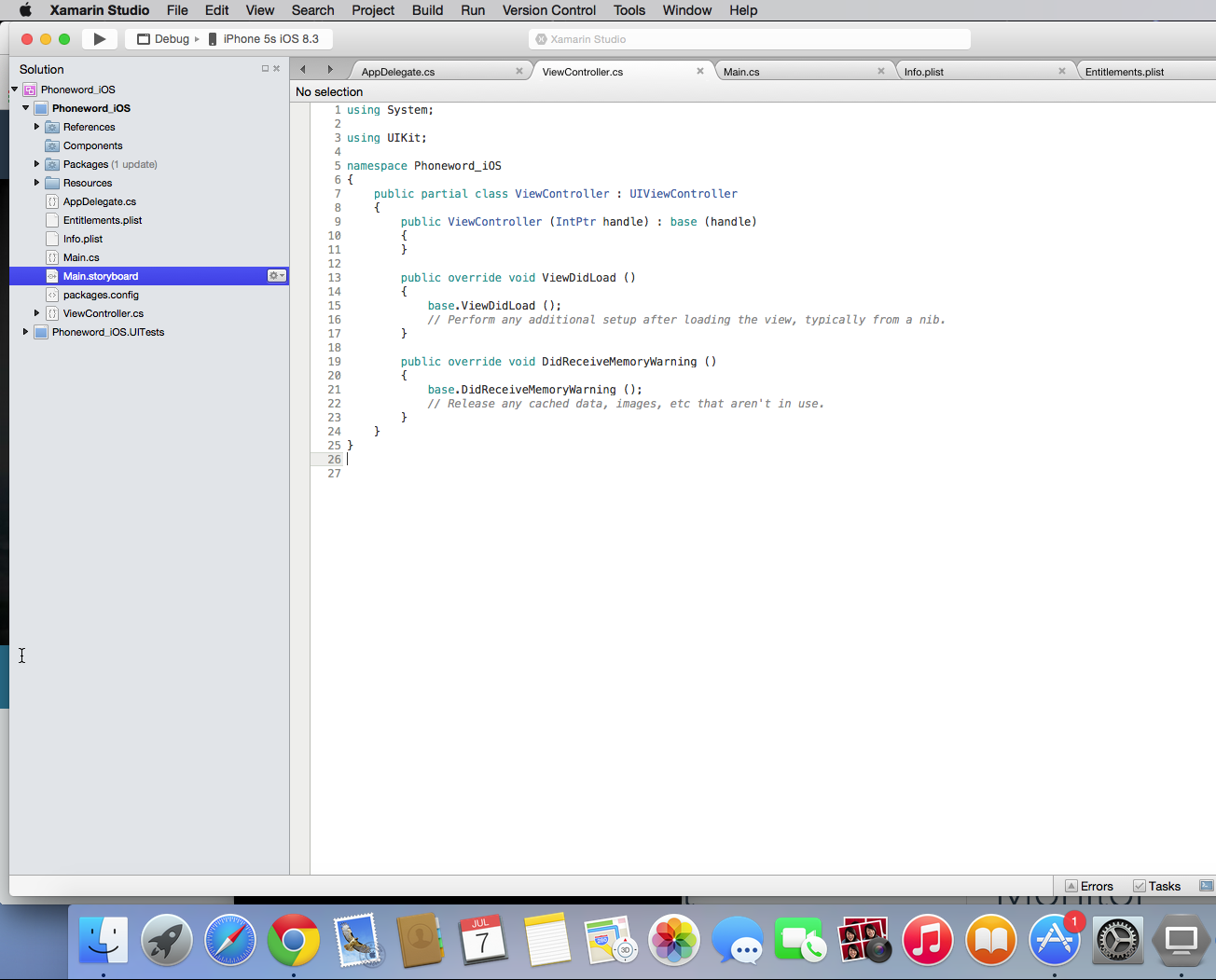


* 1. Test on real device.



Note: Plug in your iphone or ipad to MAC before running the test.

* 1. Developing in Xamarin Studio(Mac)



* 1. Deployment

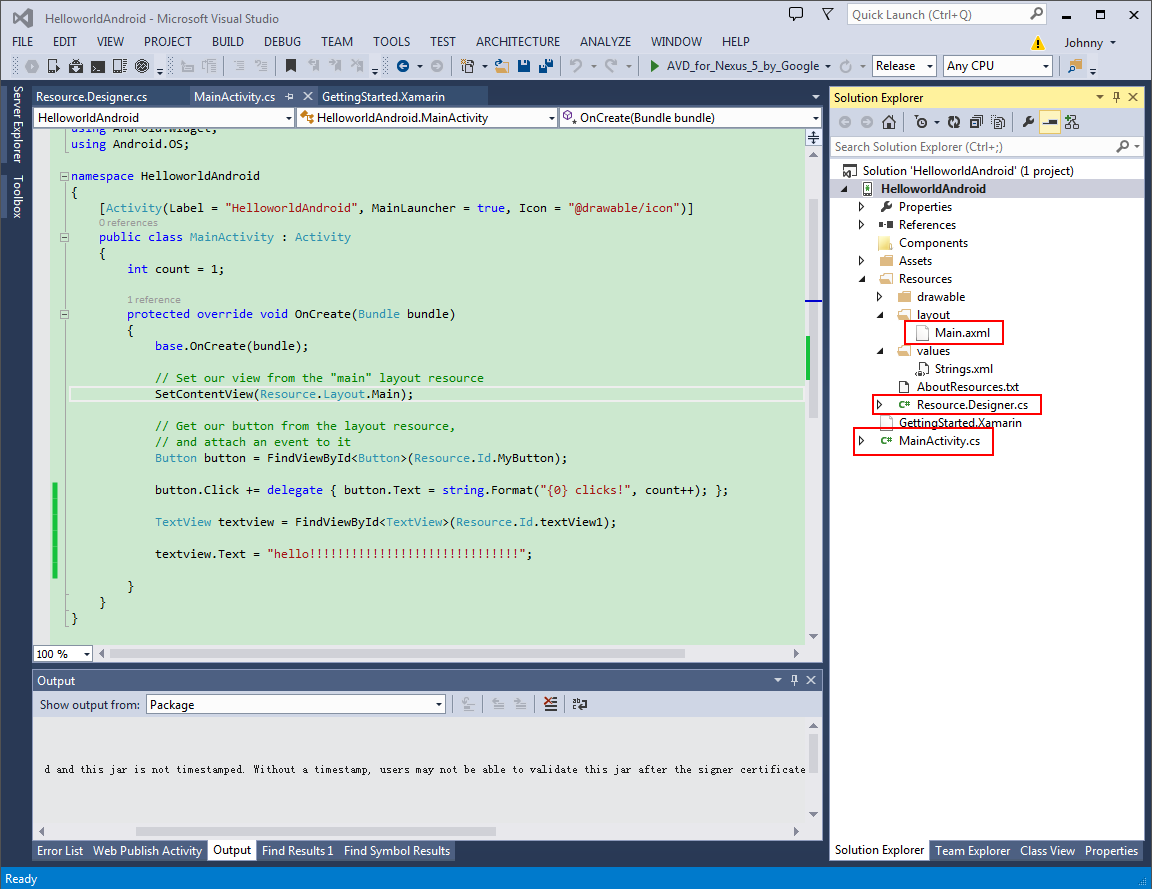
Publishing to the App Store

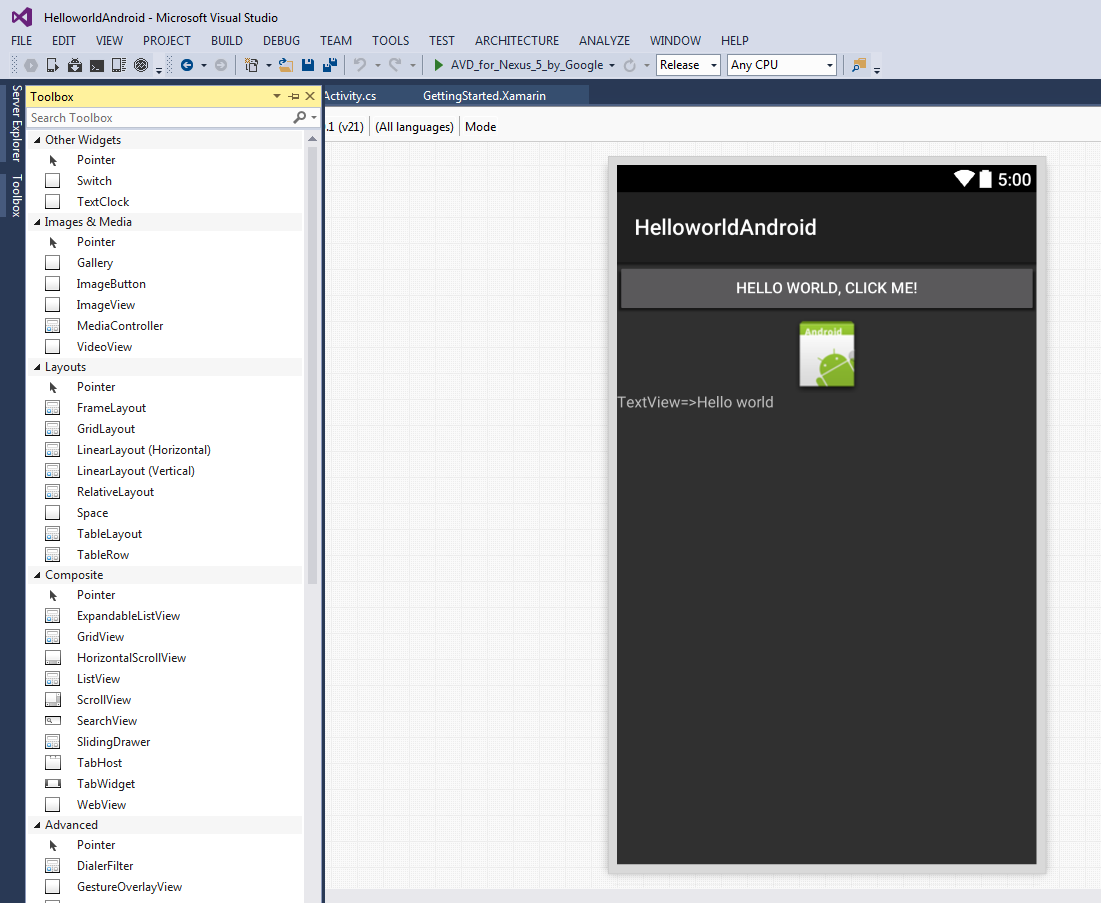
<http://developer.xamarin.com/guides/ios/deployment,_testing,_and_metrics/app_distribution_overview/publishing_to_the_app_store/>

1. Android Development

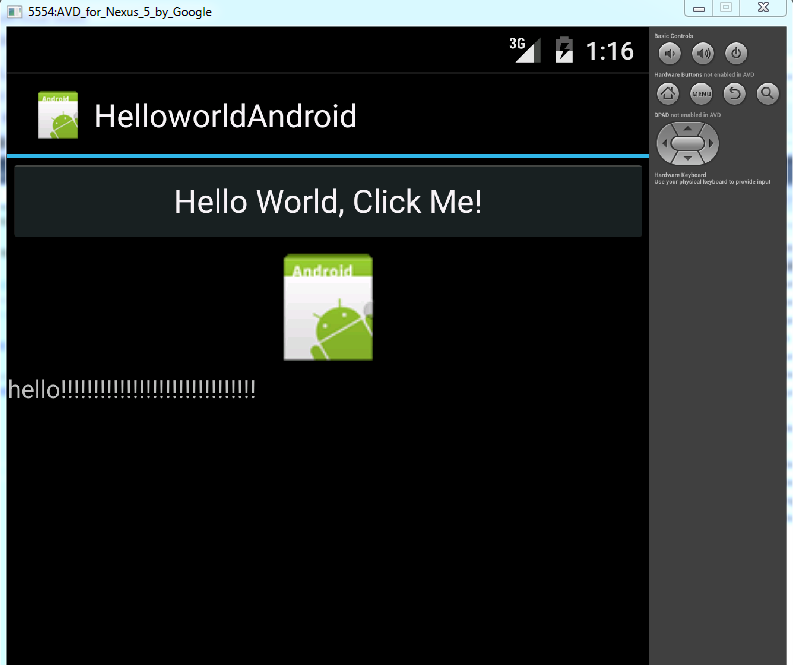
There are three options to setup the development environment for Android:

* Xamarin Studio on Mac
* Xamarin Studio on PC
* Visual Studio on PC
  1. Installation(Windows)
* Xamarin Platform for Windows
* Android SDK and Emulator
  1. Developing in Visual Studio 2013.

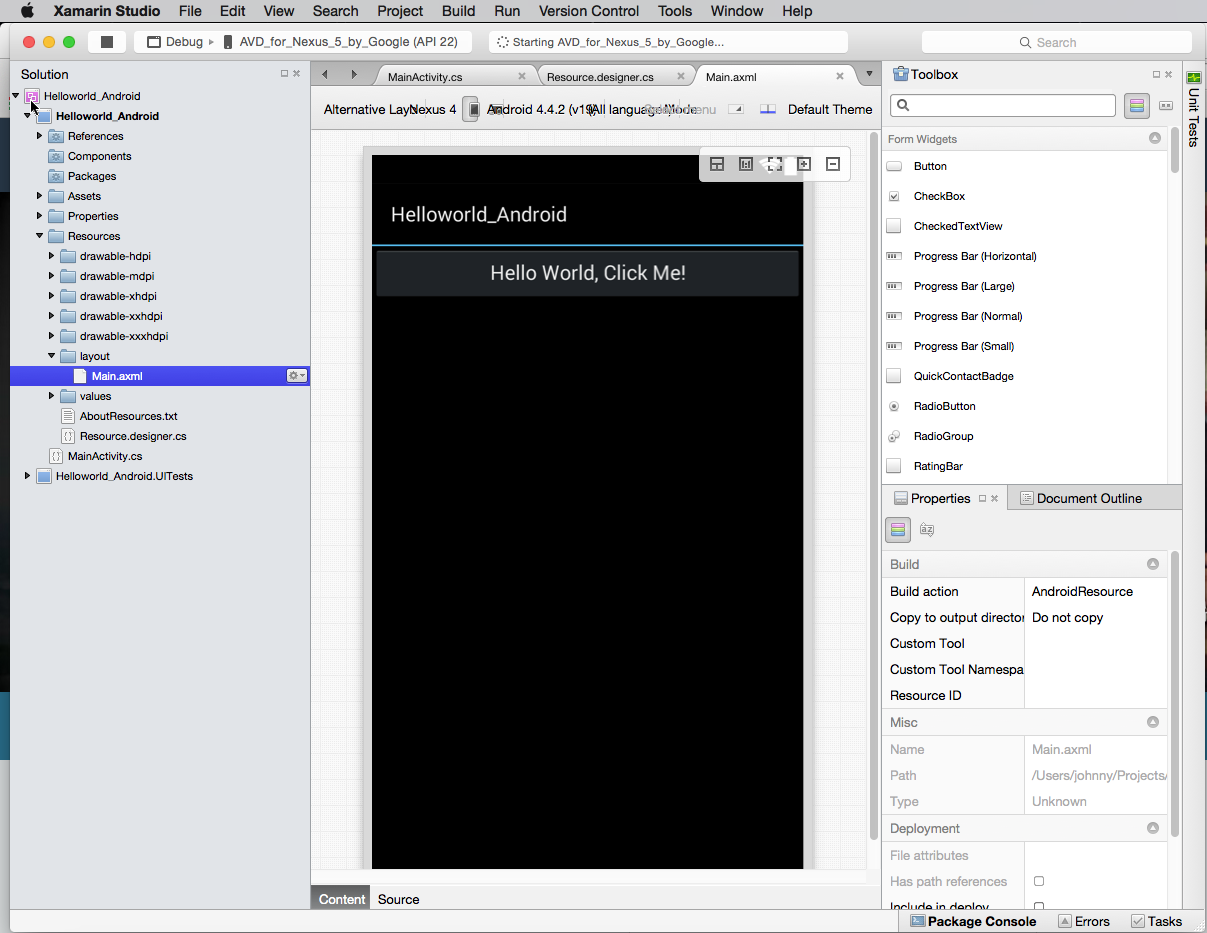




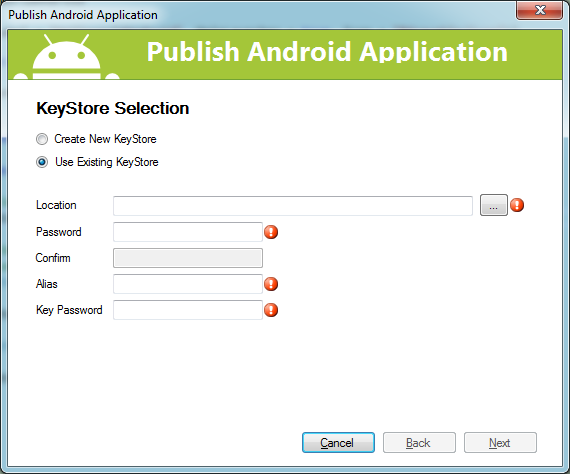
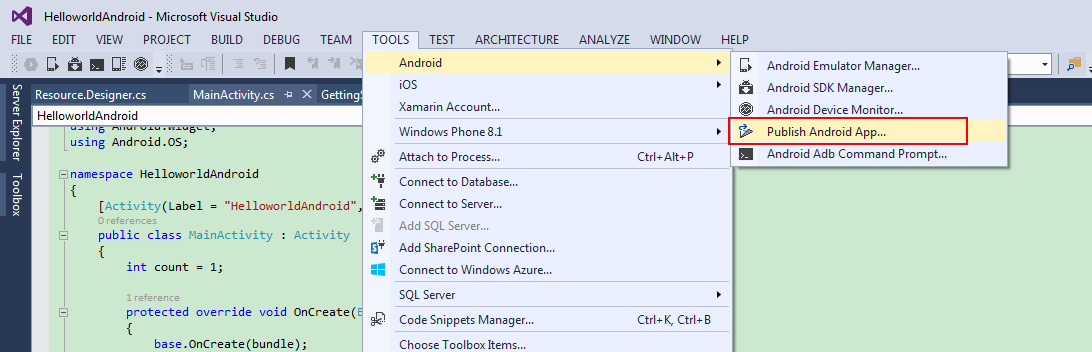
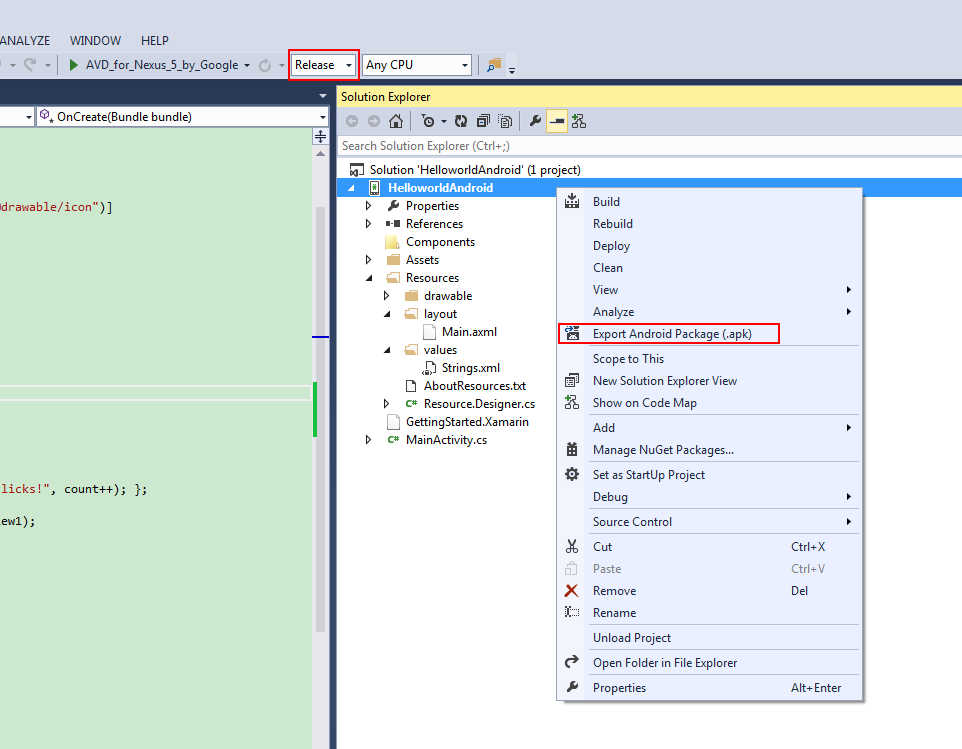
* 1. Test in emulator, AVD(Android Virtual Device).



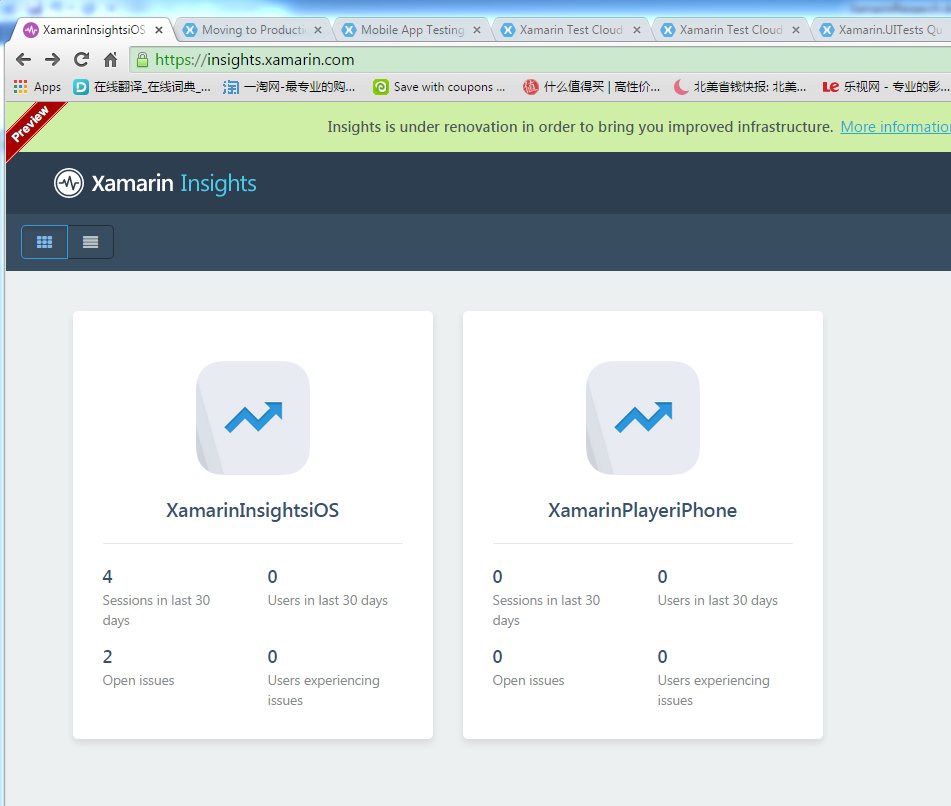
* 1. Developing in Xamarin Studio(Mac)

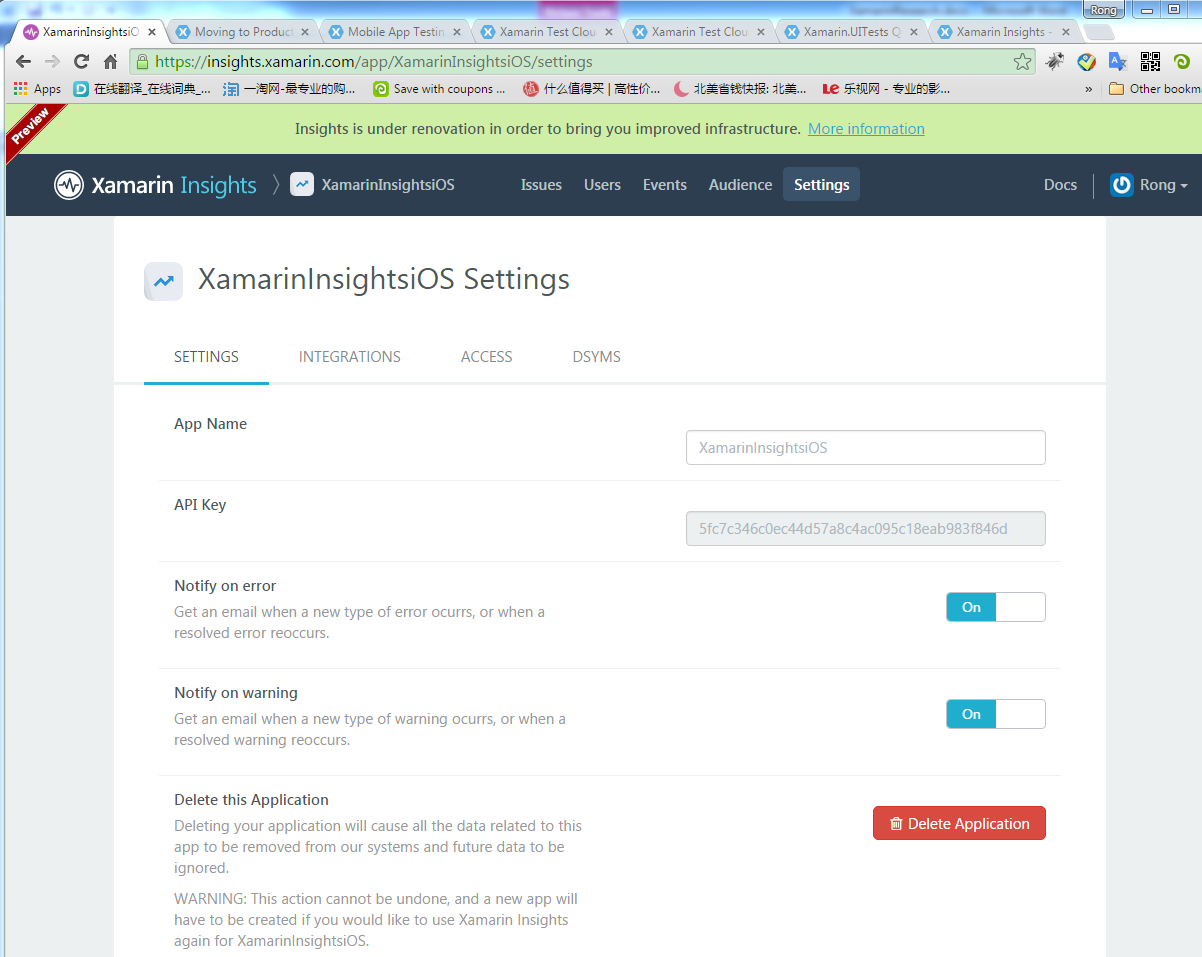


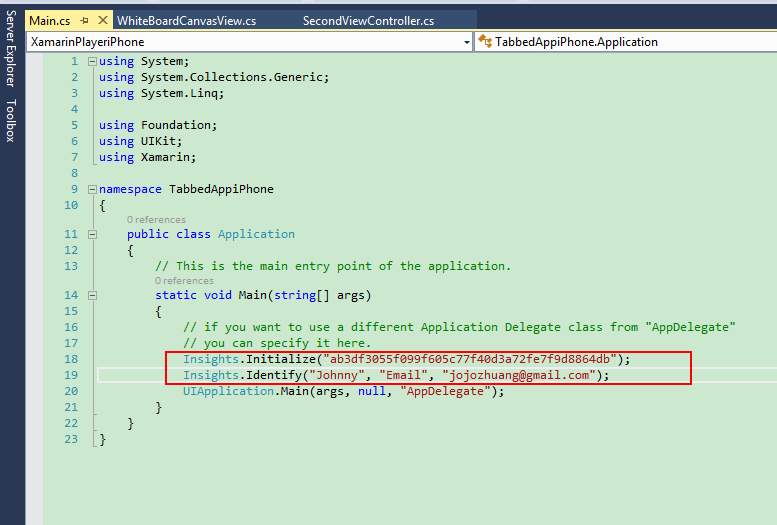
* 1. Publish and Deployment in Visual Studio.

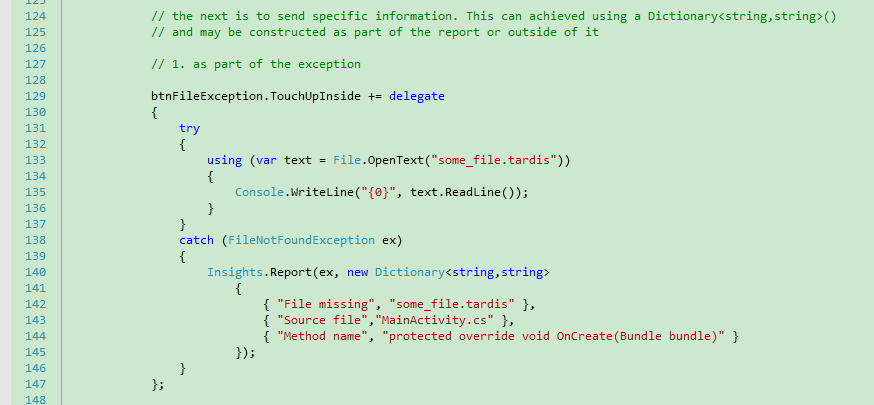


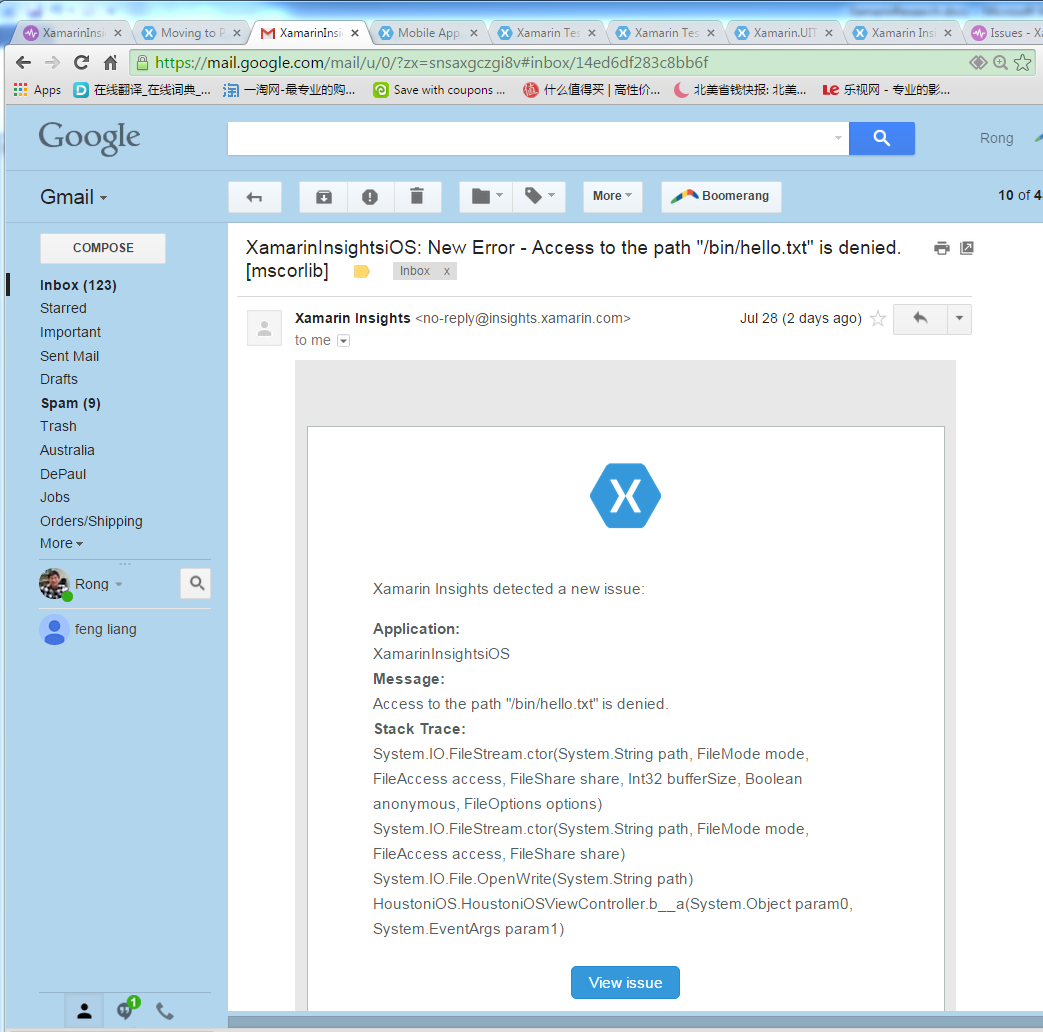
1. Xamarin Insights

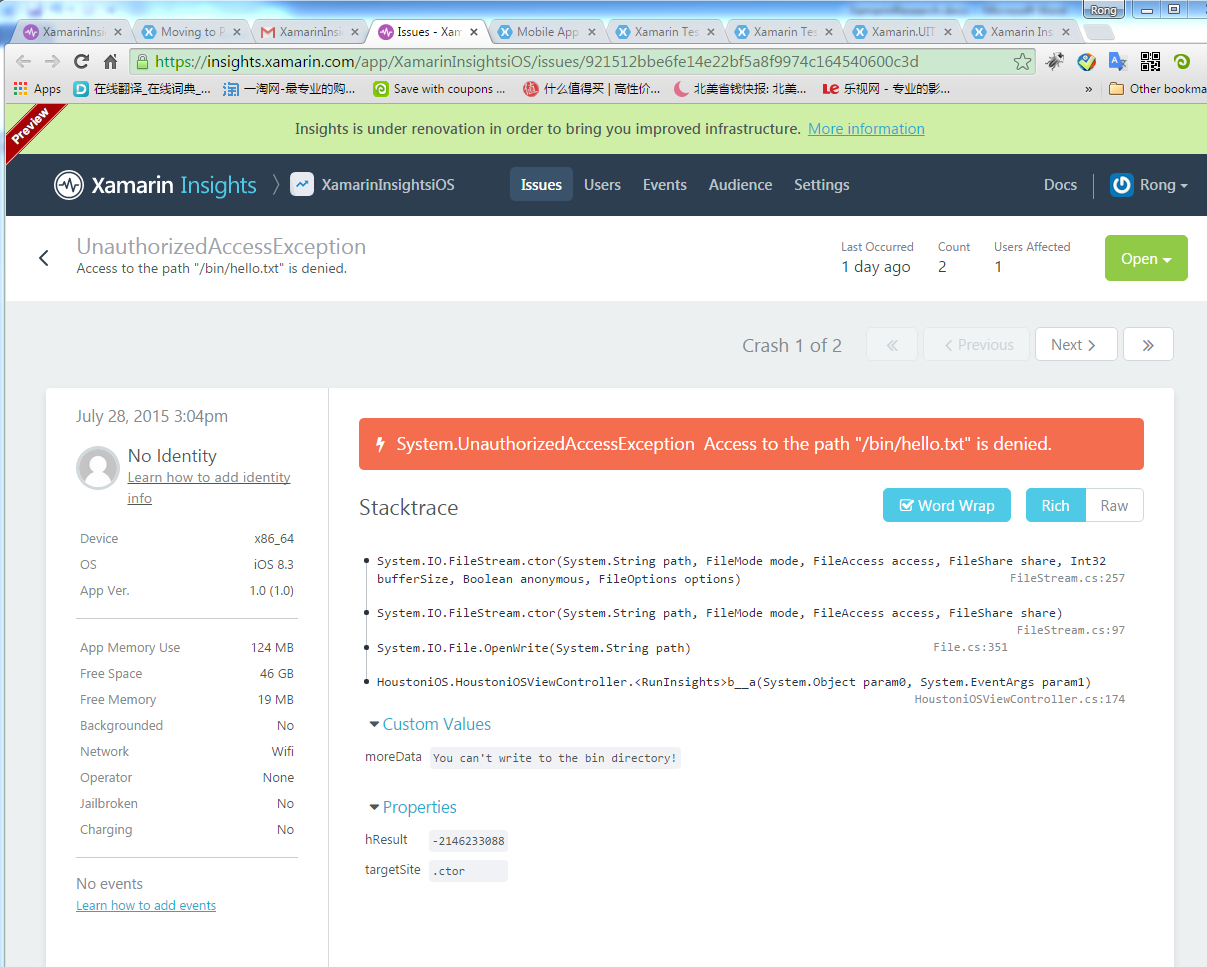






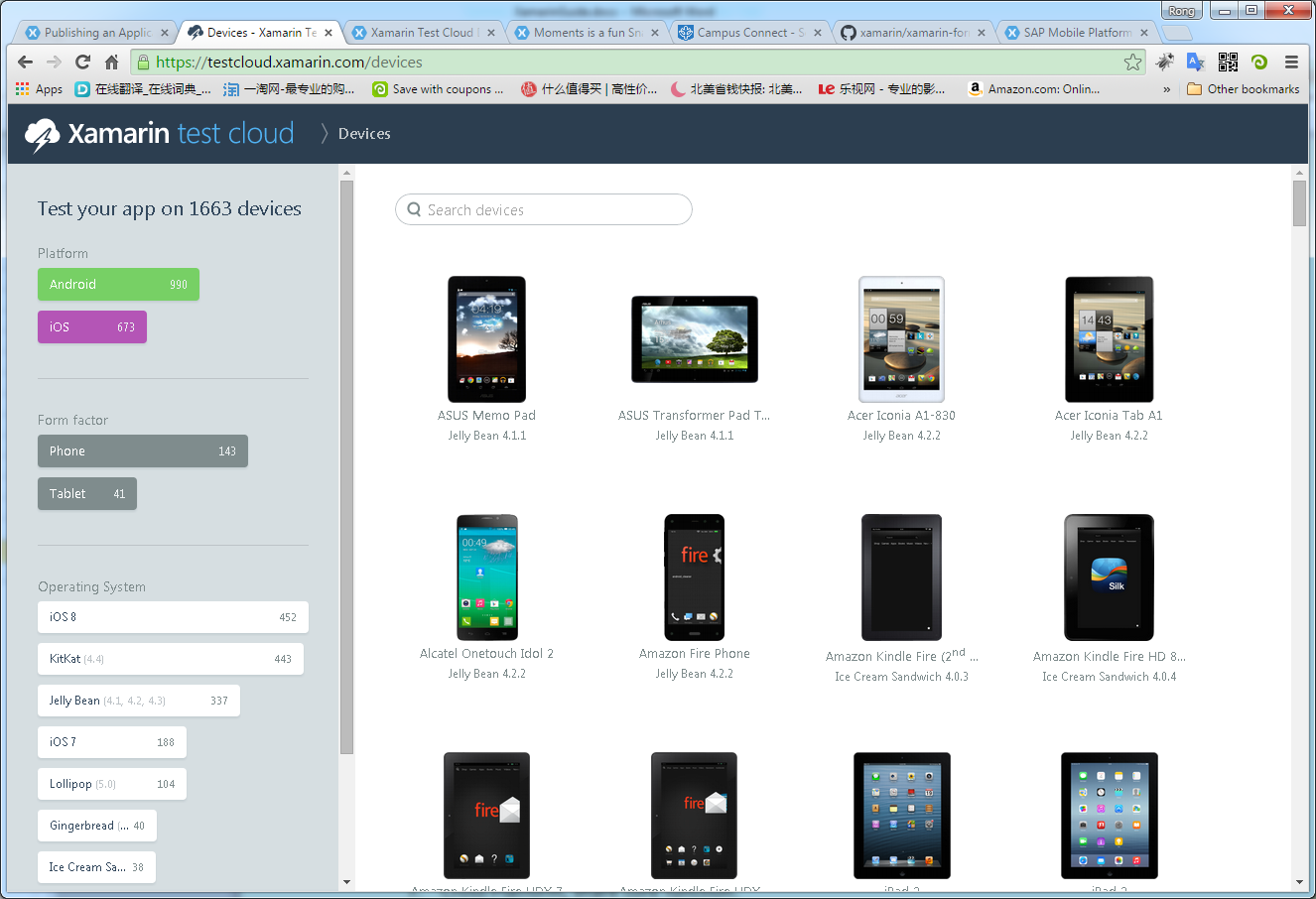




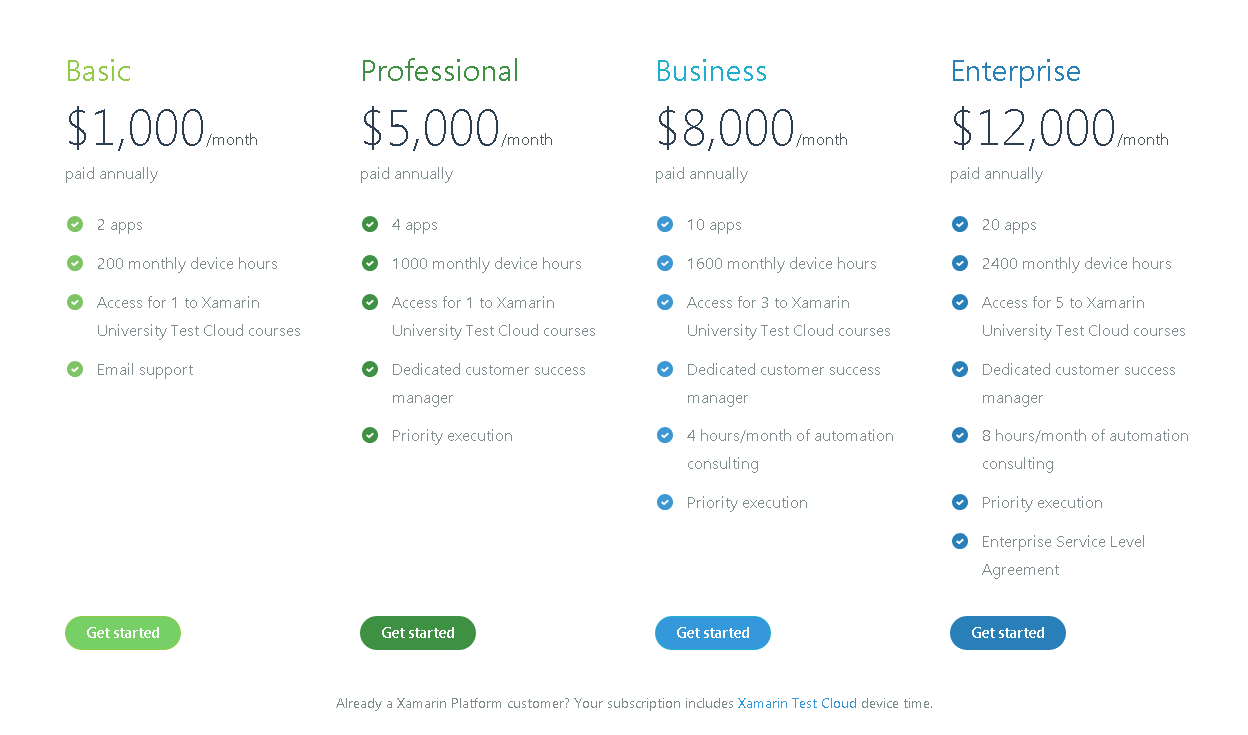


1. Xamarin Test Cloud

* Xamarin.UITest(C#)
* Calabash(Ruby)

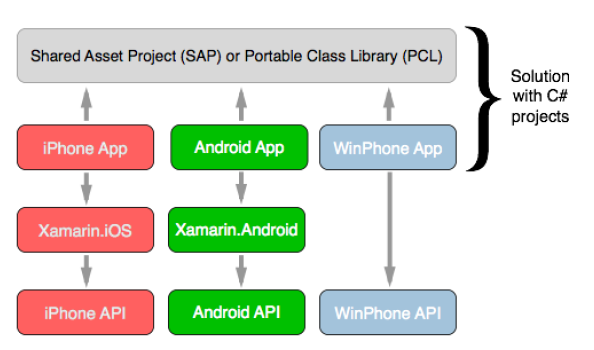


Costs:



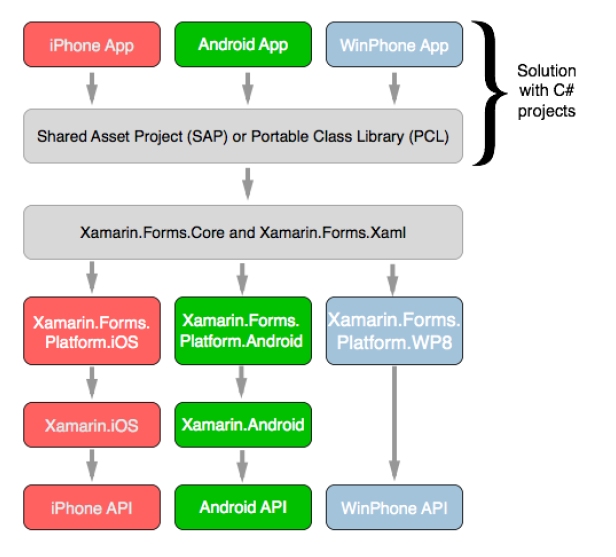
1. Cross-Platform
   1. Separated components

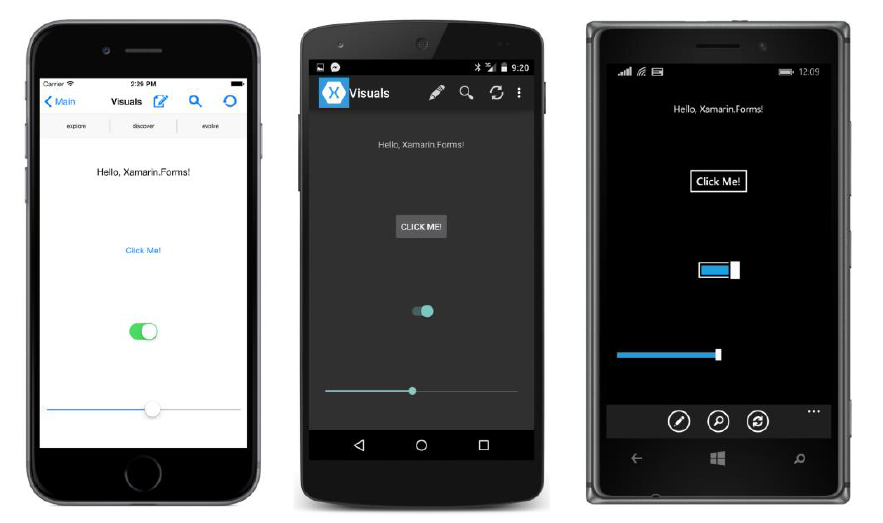
* Xamarin.Android
* Xamarin.iOS
* Xamarin.WP8



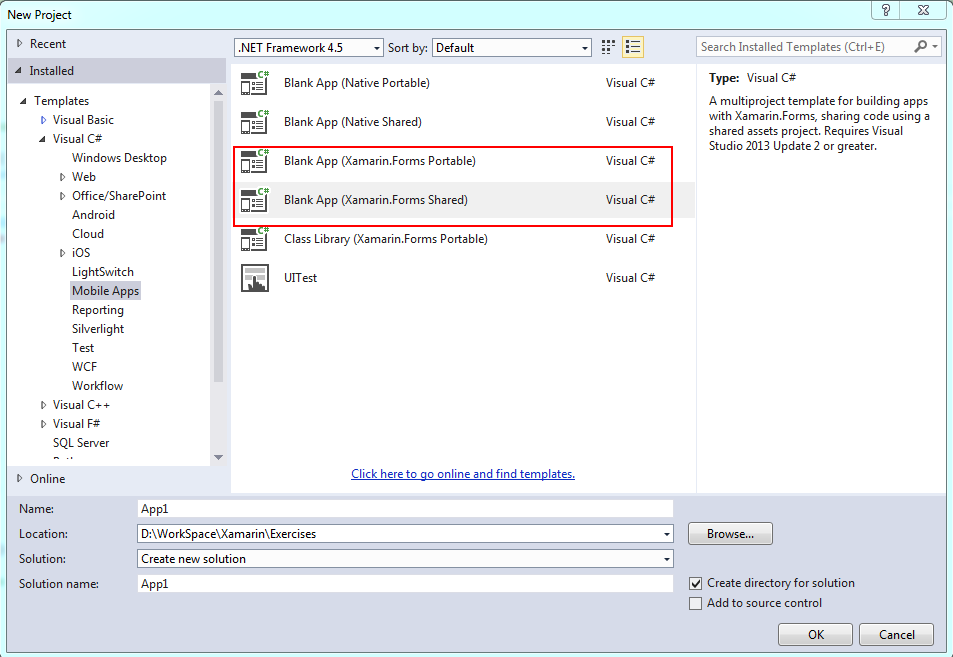
* 1. Xamarin.Form(Unified UI development process)

Xamarin.Forms integrates with Xamarin.iOS and Xamarin.Android.





* 1. Sharing Codes
* Portable Class Library (PCL): dlls
* Shared Asset Project (SAP): code and assets files

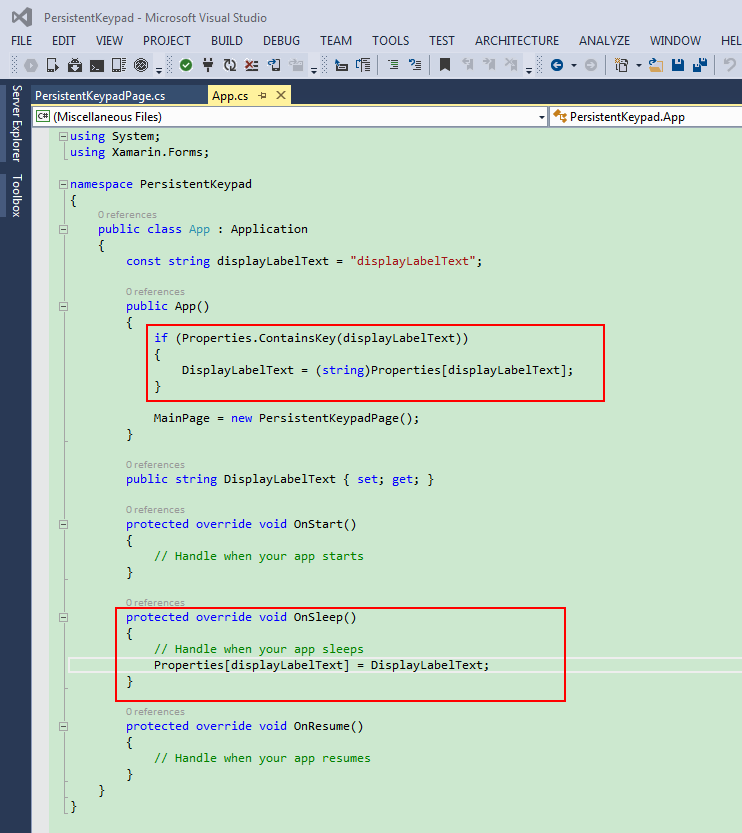


|  |  |
| --- | --- |
|  |  |

The iOS and Android projects have access to pretty much the same version of .NET, but it is **not** the same version of .NET that a Windows Phone project uses. This means that any .NET classes accessed by the shared code might be somewhat different depending on the platform. As you’ll discover later in this book, this is the case for some **file I/O classes in the System.IO** namespace.

1. Store Data
   1. Transient Data

* Application.Properties(string key, object item)
* Application Events(OnStart, OnSleep, OnResume)

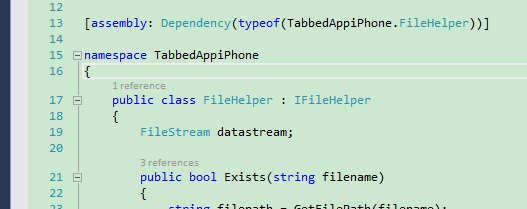


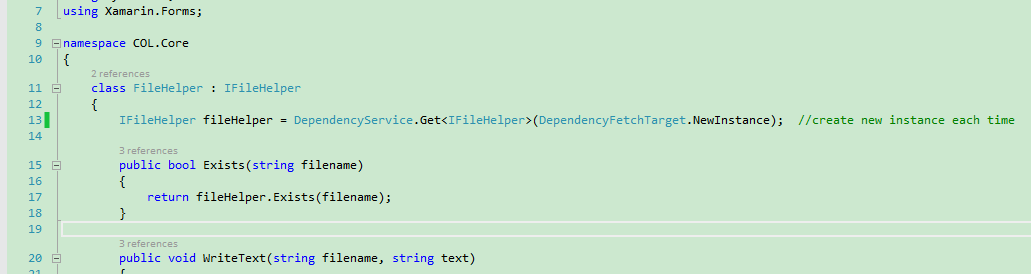
Note: If each item of this data is an entry in the Properties dictionary, each item needs a dictionary key. However, if a program needs to save a large file such as a word-processing document, it shouldn’t use the Properties dictionary, but in-stead should access the platform’s file system directly.

* 1. Large File

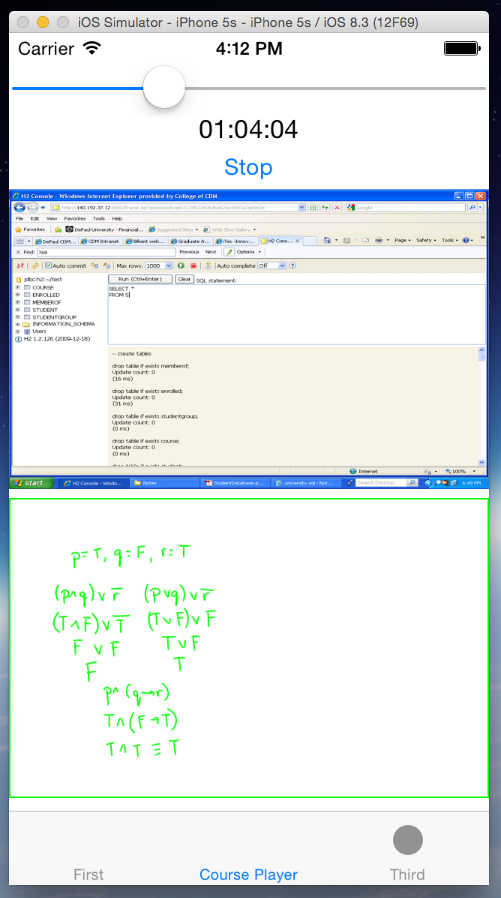
Platform’s file system: Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

Xamarin.Forms.DependencyService



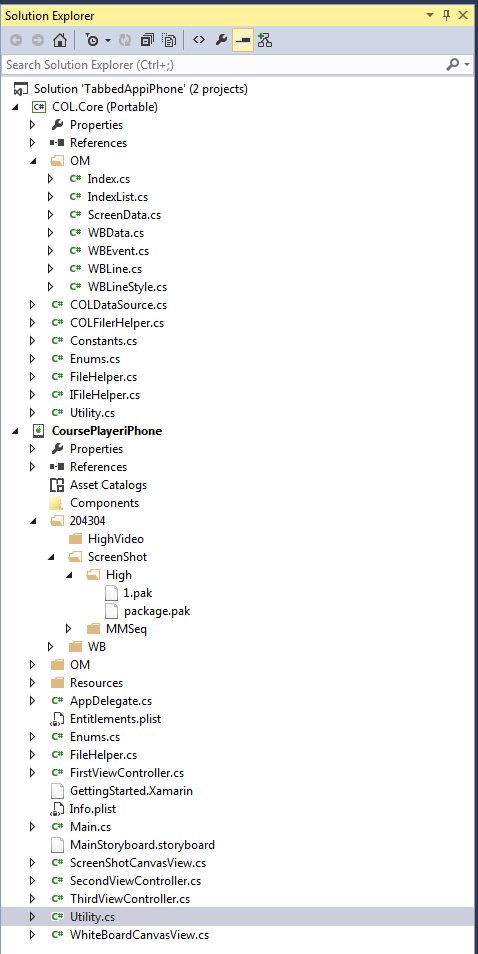


1. Prototype of Course Player
   1. Layout

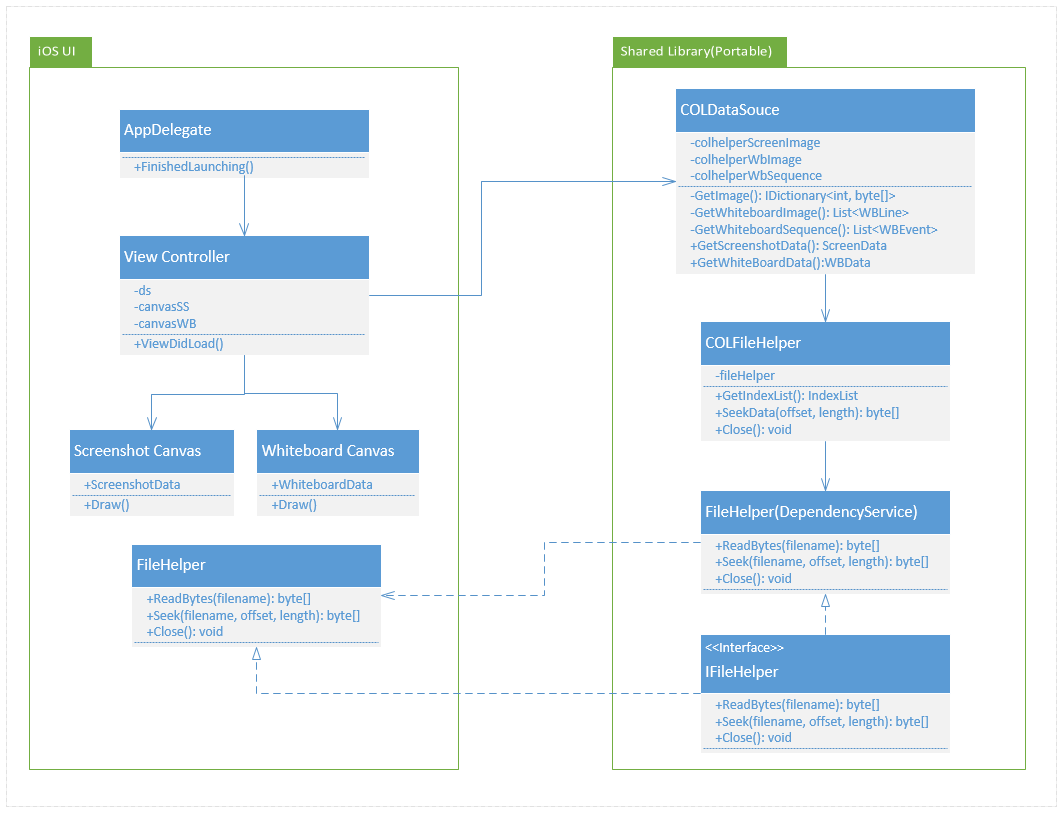


Use a slider, label and button to simulate video playing.

* 1. Components
* COL.Core contains the core function to get and convert data.
* CoursePlayeriPhone contains the course related files and render the UI.



* 1. UML class diagram



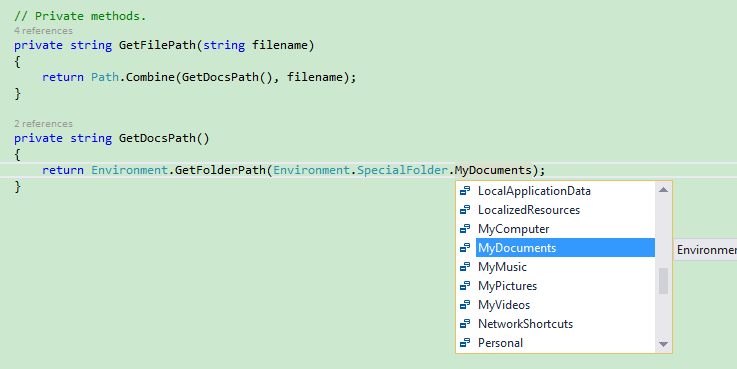
1. Important Concerns
   1. Does Xamarin provide the support of the third-party libraries? How?

Yes, Xamrain supports not only .NET based class libraries, but also Objective-C, Java, HTML or Javascript to reuse in Xamarin.

Walkthrough: Binding an Objective-C Library

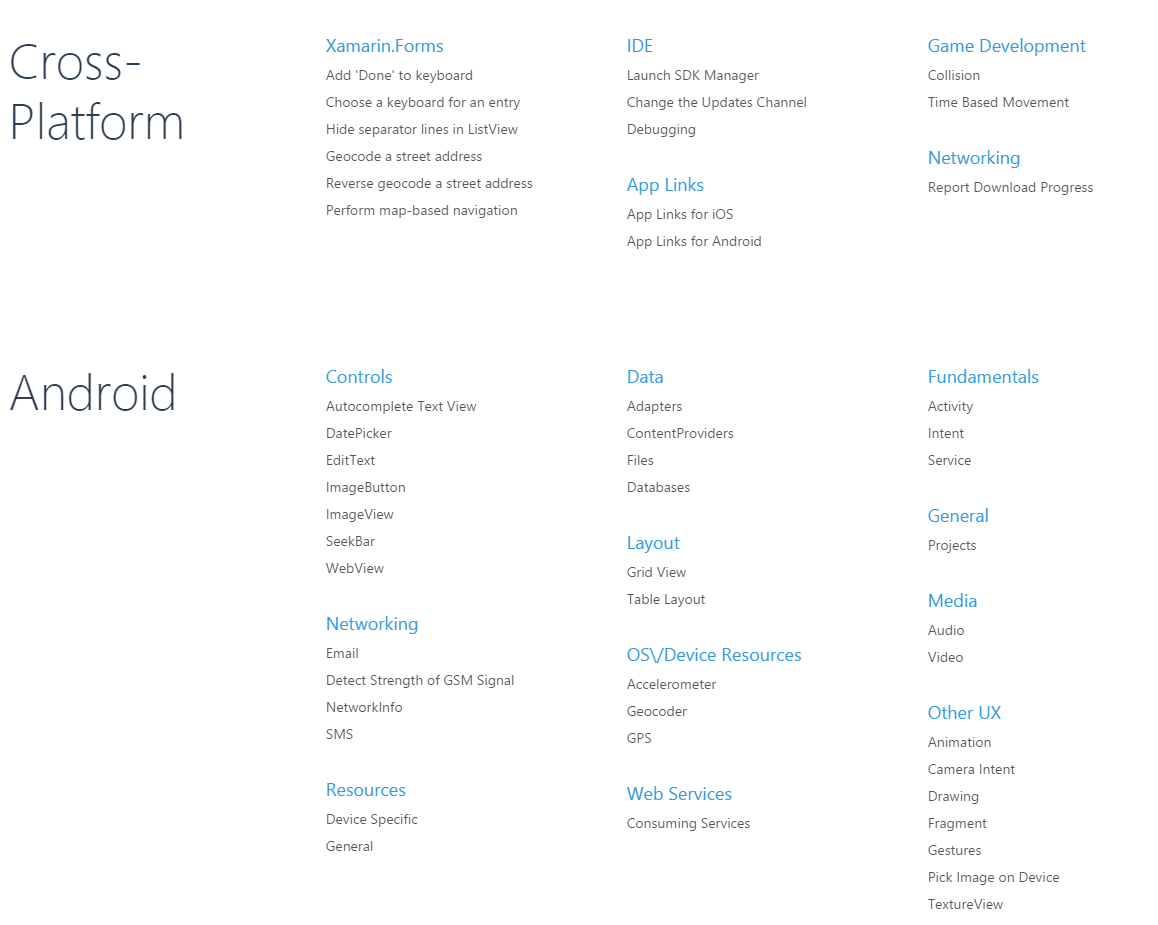
<http://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/Walkthrough_Binding_objective-c_library/>

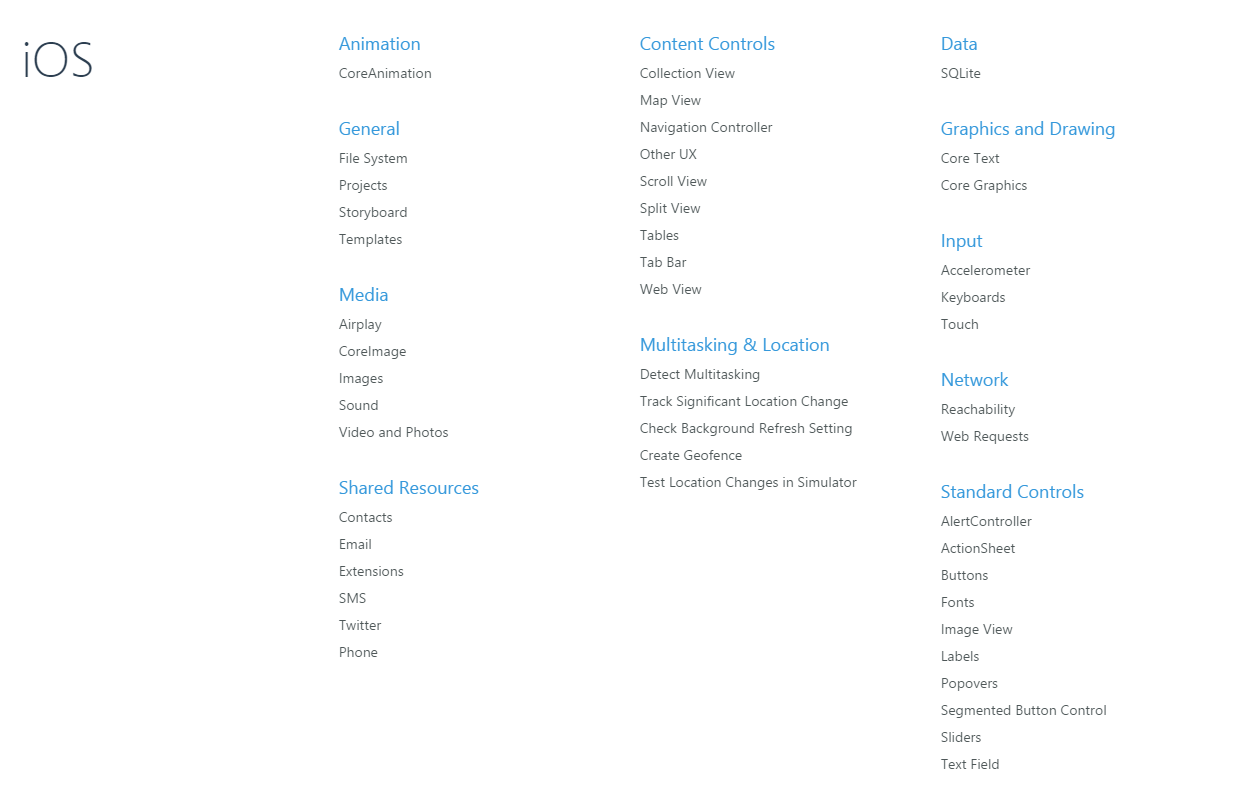
1. First, we'll create an Objective-C static library using Xcode.
2. Then we'll binding this static library with Xamarin.iOS.
3. Next, show how Objective Sharpie can reduce the workload by automatically generating some (but not all) of the necessary API definitions required by the Xamarin.iOS binding.
4. Finally, we'll create a Xamarin.iOS application that uses the binding.
   1. IO interfaces, eg. file read/write



* 1. What features, especially the system relevant functionalities, does Xamarin provide?

Briefly, Xamarin supports unified cross-platform features. Separately, it supports platform aimed features. <http://developer.xamarin.com/recipes/>





* 1. How does Xamarin follow the new releases of native platforms, iOS and Android?



1. Overview Conclusion
2. Xamarin has great support for cross-platform mobile development.
3. The development cost is high by using Xamarin.
4. Developers are required to have the knowledge of C#(.Net), Xamarin(Mono), iOS development and Android Development.
5. For iOS, the layout design is not supported enough (eg. auto layout), maybe need to create separate projects for iPhone and iPad.
6. Feature Comparison from Portfolio Perspective

|  |  |  |
| --- | --- | --- |
|  | **Native** | **Xamarin** |
| Development Community | Matured mobile platform with plenty of materials/documents. | Lack of resources, like, books, third-party libraries, etc. |
| Features/Functionalities | Can get the entire features/functionalities provided by the SDK. | Some of the native features are not supported very well(eg. Auto layout). |
| Third-Party Library | Fully supported | Needs some extra effort to achieve the same purpose. |
| Coding | Pure native codes | Hybrid of Mono, iOS and Android |
| Debugging | Comfortable with Xcode, Android Studio | Comfortable with Visual Studio |
| Deployment | Standard procedure | A little more extra work required. |
| Extendibility | Easy to add new features | In most cases, it is possible to add new feature, but need do some researching work first. |
| Reusable | Low | High, if more platforms are required. Besides, the shared library can be enhanced to support web application. |
| Effort in Development phase | High | Low, if more platforms are required |
| Effort in maintenance phase | High | Low, especially when the bug is not relevant with UI. |
| Required Developers | 2 (1 for iOS, 1 for Android) | 3 (1 for Xamarin shared library, 1 for iOS UI, 1 for Android UI) |
| Developer’s Qualification | Platform specified knowledge | Cross-platform knowledge, Xamarin platform, plus C#, Mono |
| License Fee(Per Year) | iOS $99; Android free | iOS developer account $99  Xamarin iOS: $999  Xamarin Android: $999 |
| Potential Risk | None | Uncertainty of Xamarin’s future. |

*The comparison is based on the assumption that our APP only supports iOS and Android.*

1. Reference

* <http://xamarin.com/faq>
* <https://components.xamarin.com/>
* <http://developer.xamarin.com/recipes/ios/general/file_system/load_a_file/>
* <http://developer.xamarin.com/recipes/android/data/files/>
* <http://developer.xamarin.com/recipes/>
* <http://developer.xamarin.com/guides/cross-platform/insights/>