Xamarin MonoTouch

(A C# DotNet to iOS compiler for the Mac)

Created for JaxMUG
December 2012

Irv Adams

Intended audience

- This presentation is aimed primarily at Dot Net developers who want a jumpstart at developing iPhone and iPad apps, using a familiar environment.
- I had never attempted to code anything for the iPhone more than about two months ago, and using this tool made it straightforward.
- The tools provided by Apple for software development are excellent, but may require a longer and steeper learning curve.

Bio

- Irv Adams
  - BA/math, worked on Master's and stay current with User Groups.
  - 27 years in the software industry.
  - Have worked on mainframes, midframes, PCs and most recently mobile devices, mostly smart phones.
  - Languages: ForTran, COBOL, RPG, BASIC, dBase, Clipper, FoxBase, FoxPro, Visual BASIC, C#, Delphi, ASP, HTML,etc.
  - Became interested in iPhone apps by inheriting a complicated app involving several partners that includes a web service back end and involves multiple screens, push technology and automated dialing.

History of The Xamarin company

- Boston, MA based, created May 2011
- Miguel de Icaza of Ximian launched an open source project for a Linux version of dotNet in July 2001.
- Ximian bought by Novell in August 2003.
- Attachmate buys Novell in April 2011, lays off many workers including the XIMIAN Mono open source team.
- They regrouped as a new company and in July 2011 were granted a perpetual license from Novell to create and market Mono, MonoTouch (iOS), and Mono for Android.
- The rest is recent history.

• MonoTouch technical details

- It's not free! MonoTouch Costs about $400 for a single-user license. Runs on a Mac (or Mac Mini, which is what I'm using).
- You create middle-tier code in pure C# inside an IDE that resembles VS. Screens are still designed using XCode and Interface Builder, although Monotouch makes the 'code behinds' for the controls somewhat easy (I will demo this later).
- You can compile and deploy, or just debug directly to an attached iPhone (or iTouch, or iPad), or you can run to a simulator (I'll demo this later). It's a pretty mature product, but there are still a few minor bugs (I'll give an example later).

- My experiences and observations as a DotNet guy in an iOS jungle.

- If you just want to get your feet wet, recommend a Mac Mini ($650-750). It has all the same software, is a simple small box, and you attach keyboard, monitor and mouse and away you go. Powerful and compact.
- You will have to learn, at a minimum, the following applications on the Mac: Finder, Launchpad, iTunes Connect (this is where you upload and register apps for the App store), Safari (the browser), and how to select, use and store things in the active Tray area.
- The Mac is more cloud-oriented throughout, especially for developers. It makes more use of drag-drop operations that what I'm used to with MS. In short, it's more visual.
- The forms are laid out differently, and cut/paste operations are done differently.

• The Apple Way versus the MS Way

- Apple tightly controls the distribution of mobile apps. They require that all the devices and all the users who will beta test your software be identified and provide the UDIDs (unique identifiers) to Apple.
- The distribution of test apps (ad hoc) is controlled through a free website called TestFlight. This is where versions of your app are stored, and the names and IDs of your beta testers are kept. You may have up to 100 testers.
- When your app is ready, build some screen documentation, compile the app for release, and submit it via the iTunes Connect website. Apple will review and test it, then email you when it goes to the App Store.

• Preparing your apps for distribution.

- You will have to create Provisioning and Distribution profiles for every app. These files contain the unique App ID, Certificate(s) associated with your account, and allowed Devices (along with their UDIDs) associated with this provision or distribution.
- You will have to choose icon(s) (PNG files) and resize them to at least 4 different pixel dimensions. These are stored in the Options area of your project or in the info.plist file.
- There's a very long and complete list of do's and don'ts for your apps on the Apple Developers website. They specifically prohibit certain content and themes, and require that your app do every function exactly as advertised. Make certain it is thoroughly debugged and tested.
- You have to make the choice whether to charge for your app. They should be free unless they provide a complete and complicated service. There's an incredible amount of competition out there. Either way, you gain exposure, experience and advertisement.

• MonoTouch Links to get you started.

- I found all the MonoTouch documentation to be excellent - patient and thorough, aimed at people still on the steep part of the learning curve like myself. Below are some searches that I recommend you look into:
- Google (or Bing) "Hello, iPhone - a first MonoTouch application". Go carefully all the way through this app, it takes a while but is well worth it!
- Google "Publishing to the app store - A guide to distributing MonoTouch applications" - this also takes some time to work through but tells you exactly how to publish to the App Store, when you're ready.
- Google "Working with images - A guide to iOS Icons and Images".

Code demo.

- OK, let’s see some code!
- I will demo and explain the C#, XCode and IB setup behind a single-screen gratuity calculation app that I wrote in MonoTouch and uploaded to the Apple store (“Tippy the Tip Calculator”). Then, we will create our own simple app from scratch.