Getting Started with Modern Mobile Development

Dino Esposito

@despos

2011

Content

- Mobile computing: Web and smartphones
- iPhone: tools, languages, deployment
- Android: tools, languages, deployment
- Windows Phone: tools, languages, deployment
- BlackBerry: tools, languages, deployment

A bit of background …

- First time I heard “mobile is the future” was in 2000…
  - …That future didn’t come for about a decade
- After years of .NET consulting/authoring, I’m switching to all-round mobile: strategy, models, development
- More and more companies in industry (not a specific sector) see mobile investments critical for their future
  - Both B2C and B2B
  - Printing, Editorial, Telecom, Banking, Mass retailers, Tourism, Entertainment, Hotels

Mobile Computing: Web and smartphones

Essence of mobile computing

- Devices, devices, devices
  - Cell phones, smartphones, tablets
  - Running applications on the go
- (Intricate) Jungle of different:
  - Device capabilities
  - Operating systems
  - Platforms and SDK
  - Computers
- Much worse than with desktop browsers
  - Heterogeneous audience, higher costs for startup & maintenance

Mobile computing: Web

- It all started ten years ago …
  - From a specific subset of HTML to HTML5

- 2000 ➔ WML
- 2002 ➔ primitive forms of XHTML
- 2004 ➔ XHTML

- Now quickly moving towards HTML5

Mobile Web: challenges

- Ad hoc design of web sites (m-sites)
- Different layout, different content, different idea
  - It’s just a new project

- **Reusability** is a great thing, if applied at the right level
  - Don’t succumb to the siren call that m-sites are the standard ASP.NET sites just with a different CSS/layout
  - Architect your site to expose reusable/queryable logic
  - Add a mobile service layer that serves right data to presentation

Mobile Web: challenges

2 of 3

- Mobile sites are generally simpler than classic sites
  - Logical/functional subset; not a physical subset
- In theory, mobile sites can provide you a huge audience
  - Millions of devices can browse the web
  - Nearly each device in its own way
  - Huge fragmentation (of capabilities)
- Know your users
  - Selling ringtones? Your target is the device; maximize audience
  - Selling services? Your target is (smart) consumer; focus on apps

Mobile Web: challenges

3 of 3

- Fragmentation is huge

- Don’t trust the device
  - Manufacturers want to make each device kind of unique
  - For years, they just customized the embedded browser
  - Net effect is that too many devices have a different set of capabilities

- Querying for capabilities
  - Test capabilities in JavaScript via DOM and browser OM
  - Acceptable results in desktop Web; not in mobile Web because of the different impact/size of fragmentation

WURFL at a glance

- XML-based repository of device capabilities
  - 500+ different capabilities of 7000+ devices
- Open-Source product with very strict (AGPL v3) license
  - AGPL = open-source all of the source code on your server
  - Commercial license from ScientiaMobile (**scientiamobile.com**)
- Adopters
  - Facebook, Google, AdMob
  - Numerous mobile (virtual) network operators
- WURFL in .NET
  - Official API from ScientiaMobile (scientiamobile.com/downloads)
  - API from 51Degrees (with uncertainties around the licensing terms for the WURFL repository)

WURFL at a glance

2 of 2

- Users, manufacturers, MNOs/MVNOs, content providers have different interests
- No easy way to agree on a set of standards
- If-then-else to output different CSS/script/layout doesn’t work because of the huge number of possible scenarios
  - Can’t fork a site per device and not even for classes of devices
  - Focus on capabilities and WURFL tells you about capabilities “known to be associated” with a given UA string
- WURFL is for the content provider
  - Let content providers know about real capabilities of the device

Mobile computing: smartphones

- Smartphones run their own OS/platform
  - Mobile OS is like a continent
  - Differences at various levels
  - Continents partition Earth; mobile OSs partition mobile space

- A few platforms you might want to address
  - iPhone/iPad
  - Android
  - BlackBerry
  - Windows Phone 7
  - Maybe Nokia QT

Mobile Web vs. Native Apps

- Follow-up session («Mobile Dilemma»)

- Decision boils down to your mobile strategy
  - Generally preferable to start with a m-site (large audience, lower development costs)
  - Advertise your m-site; redirect automatically to m-site
  - Upgrade to m-site subscriptions
  - Upgrade to smart-apps. For which platforms?

- Native apps may be cooler, but expensive
- Some middle ground being researched...

iOS

iOS: tools

- You need a Mac; the cheapest Macbook is fine
  - Mac is necessary to compile the code as it relies on libraries that simply don’t exist in Windows
  - Technically, can run OSX on a Win box; except that it is illegal ☺

- Join the iOS developer program ($99/year)
  - Free registration doesn’t not allow to test on real devices
  - In 2010, US declared jailbreaking lawful

- Get and install Xcode from Mac store
- Get and install the iOS SDK

- Get familiar (???) with Objective C

iPhone App Basics

- Starter method (main.m)
- Single window object is created by you in *main* or loaded from a XIB file
  - XIB files are where the UI behind the iPhone application is saved/packaged (i.e., form designer files in .NET)
  - Single window contains views and/or controls
  - One window, possibly multiple views
- Views are rectangular areas on top a window
  - Display content (controls, animation, text, drawing)
  - Handle events (touch)
  - Various specialized views: table, Web, alert, navigation
  - View behavior handled by **controller** classes

```
#import <UIKit/UIKit.h>
#import "MyWindow.h"

int main(int argc, char **argv)
{
  NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
  return UIApplicationMain(argc, argv, [MyWindow class], nil);
}
```

The window class to create.
This is **nil** if you rely on the main XIB file.

The name of the app-delegate to use.
If nil, then assumes "AppDelegate"

```
@interface MyWindow : UIApplication {
    UIView *mainView;
    UITextView *textView;
}
```

```
#import "MyWindow.h"
@implementation MyWindow

(void) applicationDidFinishLaunching: (id) unused
{
    UIWindow *window;
    struct CGRect rect = [UIHardware fullScreenApplicationContentRect];
    rect.origin.x = rect.origin.y = 0.0f;

    window = [[UIWindow alloc] initWithContentRect: rect];
    mainView = [[UIView alloc] initWithFrame: rect];
    textView = [[UITextView alloc]
        initWithFrame: CGRectMake(0.0f, 0.0f, 320.0f, 480.0f)];
    [textView setEditable:YES];
    [textView setTextSize:14];

    [window makeKey: self];
    [window _setHidden: NO];
    [window setContentView: mainView];
    [mainView addSubview:textView];

    [textView setText:@"Hello World"];
}
```

Concepts you must get used to ...

- An **app-delegate** controls the behavior of the application from start to end
  - Receives notifications when the app reaches certain states such as “finished launching” or “willterminate” or “memory warning”
- A **view-controller** class governs the behavior of a view
  - Handles touch events, initialization

Concepts you must get used to ...

- An **outlet** is an “object reference” through which the controller acts with an object (i.e., button) in the view
  - Similar to Button1 members in VS, must be created explicitly
  - Need outlets to be able to set a label after a button click
- An **action** is an action to be performed on an object
  - First add outlets and actions to XIB
  - Next connect them to actual objects so that action “btnClicked” is associated with an event on Button1 and outlet “Button1” is associated with a given button…
  - Finally, write the code for btnClicked in the view controller class

iOS: MonoTouch

- Use .NET for building iOS applications
    - Check out Xamarin.com ($399 license for individuals)
- A Mac computer is still required
    - Need: iPhone SDK, Mono, MonoTouch SDK
    - Use MonoDevelop to develop code
    - Use Interface Builder including Cocoa Touch thus having access to all the standard iPhone controls
- Limitations on generics and dynamic code (DLR)
- You get a native iPhone application that can be published as any other iPhone application
- Wrappers for iPhone native API (accelerometer, GPS, ...)

iOS: MonoTouch

- Compile standard **.NET 4** code using the MonoTouch core assemblies
- Reuse is possible via a new compile step for the MonoTouch profile
  - Non UI-code
  - Code can potentially be shared between .NET, Android, and iPhone/iPad
- Currently, C# only

- With some work, it is possible to write iPhone code in Windows under Visual Studio and use the same project to compile on Mac

iOS: Deployment

- Applications must be published to the AppStore
  - Internal distribution is possible only with an Enterprise developer account
- Application code must be signed against a distribution certificate (to identify the publisher) and a distribution provisioning profile
  - For companies, only the Team Agent can get the certificate
  - Get the AppStore distribution provisioning profile from the portal
  - Compile against that with your distribution certificate and submit

iOS: Testing on devices

- Get a Development Certificate to sign your code
    - Create a **Certificate Signing Request** using the Keychain Access application on your Mac
    - Log on to the portal and upload the CSR
    - This needs be done once (per developer)
    - Install the certificate in the Mac keychain
- Get a provisioning profile (Pprof) for each device
    - Register a device manually through the portal or connect them to Xcode and let it do what's required (only a few devices)
    - If you do it manually, you need the device UDID
    - Can get UDID via Xcode, iTunes, or the device itself (settings)
    - UDID != IMEI or serial number

iOS: Getting the Pprof

- Xcode
  - Once the certificate is installed, you simply build the app and tell Xcode you want to test on the attached device
  - Xcode gets the Pprof automatically (if the device is registered)
  - In alternative, do it manually through the portal and download the Pprof to the device
- Ad hoc provision profiles
  - To test on non-registered devices (up to 100) create an ad-hoc provision profile manually on the portal
  - Indicate UDID and App ID and download the Pprof as a file
  - Compile the app against this Pprof and send both profile and app to the tester
  - Tester doesn’t even need to have a Mac and install via iTunes

Over-the-Air Beta Testing

- Upload your IPA file to **https://testflightapp.com**
- IPA = .app + ad hoc provisioning profile (create in Xcode)
- Get a TestFlightApp account and email testers
- Testers get the IPA from the Web

Android

Android: Tools

- Pay a fee only to publish to the Market (one-time $25)

- Get and install the Java SDK
- Get and install the Android SDK

- Eclipse or IntelliJ Community Edition as the IDE

- Get familiar with Java

Android App Basics

- Starter class
- Create main view
- View based on XML file(s)
- Event handling

- Manual binding of handlers to controls
- Manual definition of control references (outlets)

- Easy match with C#/VB

Android Deployment

- Just compile and distribute the APK executable
- Everything is at your own risk

Phone7

WP7: Tools

- Join the developer program ($99/year)
- Get and install the SDK
- Visual Studio / Blend
- Get familiar with WPF

WP7 App Basics

- Silverlight application

- Special aspects
  - Tombstoning
  - Launchers/Choosers
  - Pivot/Panorama
  - Tiles
  - Application bar
  - Multiple options for storage

WP7 Deployment

- Log on to the portal and submit the app
  - The app will go through the certification process and if approved it is published to the marketplace
- Limit to the number of free apps you can upload

WP7 Testing

- Just install any application you want on unlocked devices
  - Max 10 sideloaded applications at a time
- Each developer can unlock up to 3 devices
- Install XAP files via a tool
- No way to install on locked devices

- Private Beta Marketplace to test pre-release apps
- Private,non publicly discoverable marketplace?

WP7 Private Beta Marketplace

- Create a private beta marketplace for your app
  - Expires after 90 days
  - Non updatable; create a new beta marketplace for new releases
- Adds the tester Live ID to the beta marketplace and sends an invitation email to the tester
  - The invitation email includes a link to the beta XAP
- Up to 100 testers per marketplace
- Testers login to Zune and install the beta application (only if they are on the approved list)
- Beta software doesn’t go through certification
  - Instant uploading of a new fresher XAP

BlackBerry

BB: tools

- Get and install the Java SDK
- Get and install the BlackBerry JDE Component Packs

- Eclipse with the BB plugin

- Get familiar with Java

BB App Basics

- Each UI application is made of a stack of screens
- Topmost screen gets input focus
- Adding a screen displays it; removing a screen displays the new topmost screen

- Overall similar to Android
- Can even reuse some code that implements logic

```java
public class HelloWorld extends UiApplication
{
    public static void main(String[] args)
    {
       HelloWorld theApp = new HelloWorld();
       theApp.enterEventDispatcher();
    }

    public HelloWorld()
    {
        // Display a new screen
        pushScreen(new HelloWorldScreen());
    }
}
```

BB Deployment

- Over the Air
  - Users go to your web site with the phone and install the application from it
- Desktop Manager
  - Install downloaded apps from PC to a USB-connected device
- Application Web Loader
  - Installs the app from a web site to a USB-connected device
- BES administration
  - Pushes the application to the devices connected to it
- Virtual Preloads
  - Carriers add links to preconfigured apps; users go and install

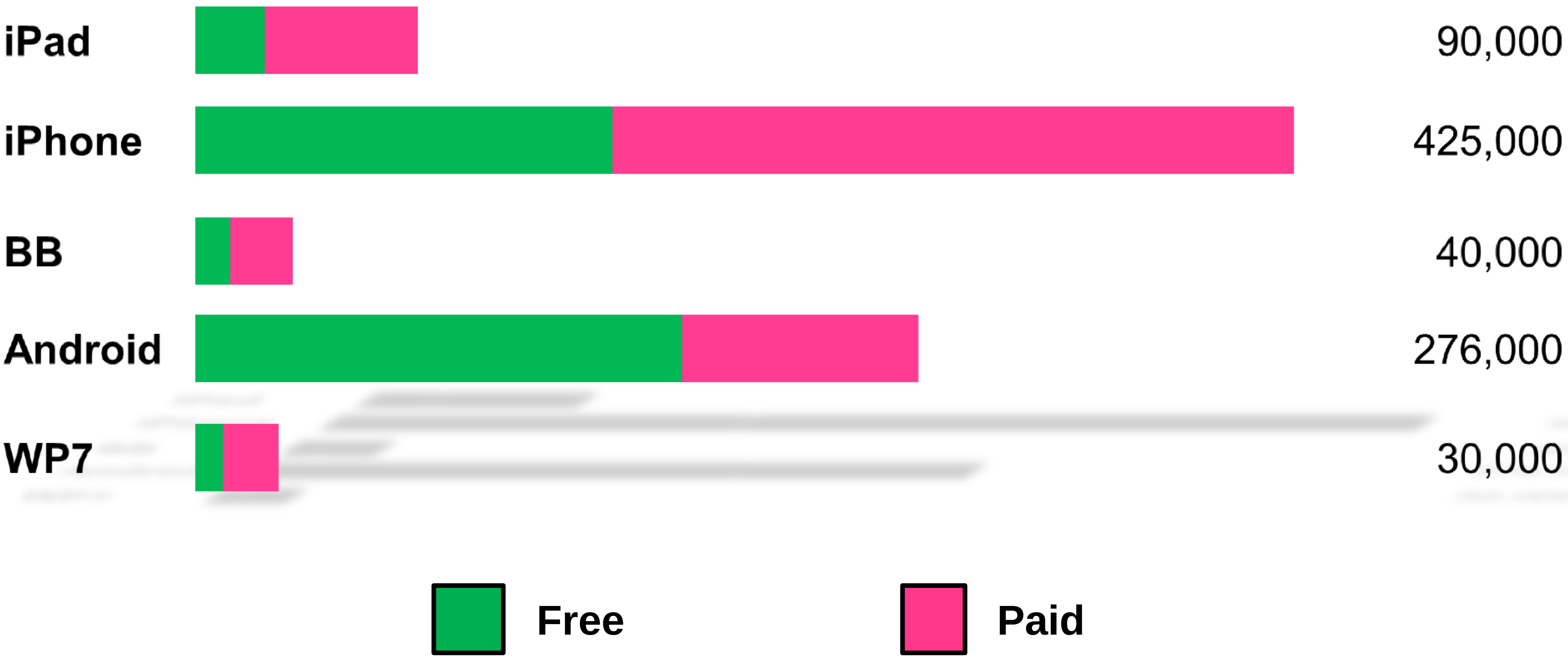
BB AppWorld

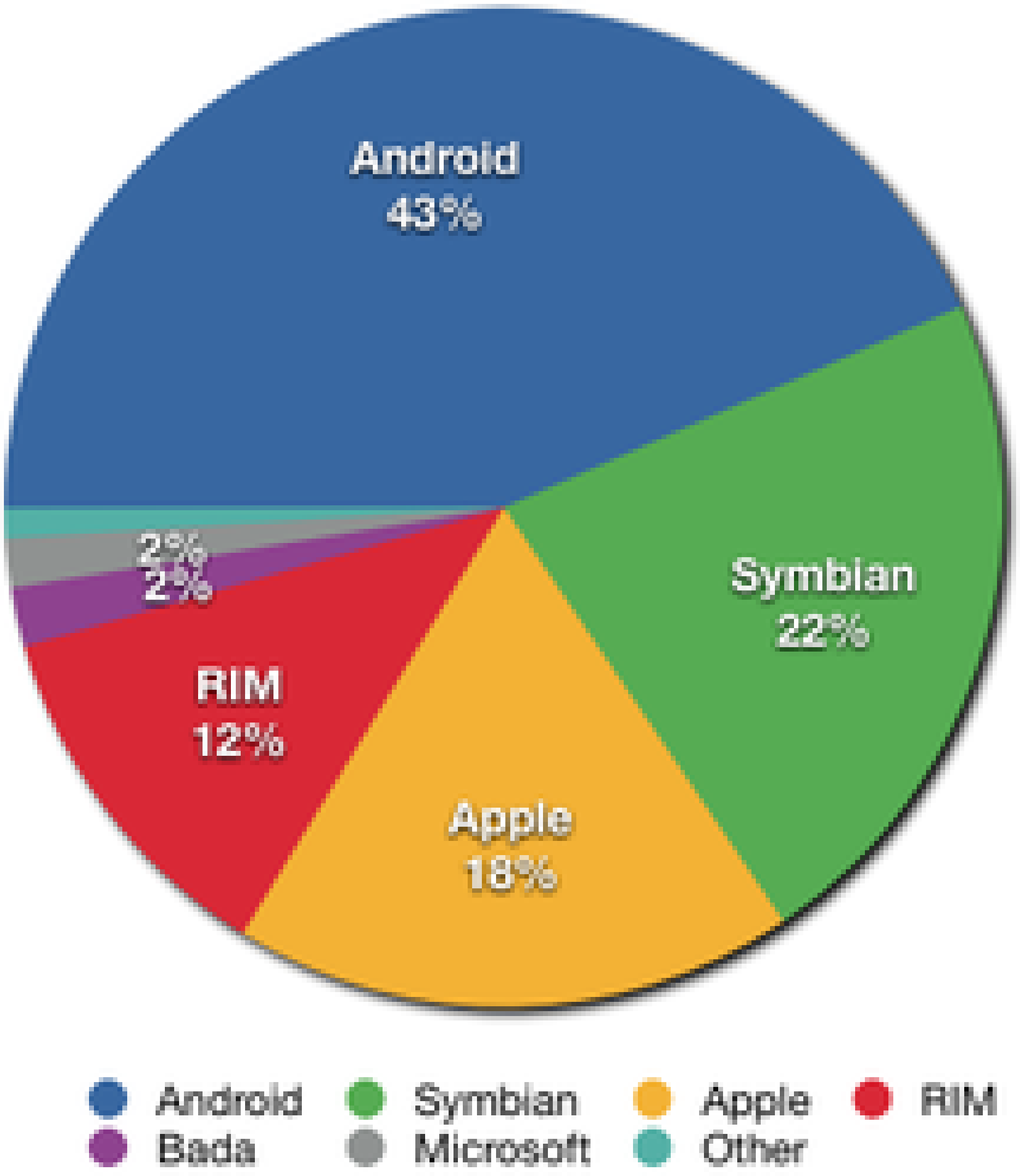
- Register a vendor account for the AppWorld for $0
- Approval process for submitted applications
- Paypal account to get paid

Apps and Stores

September 2011

Approx number of apps per store




Android
43%
Symbian
22%
Apple
18%
RIM
12%
2%
2%
Android
Symbian
Apple
RIM
Bada
Microsoft
Other

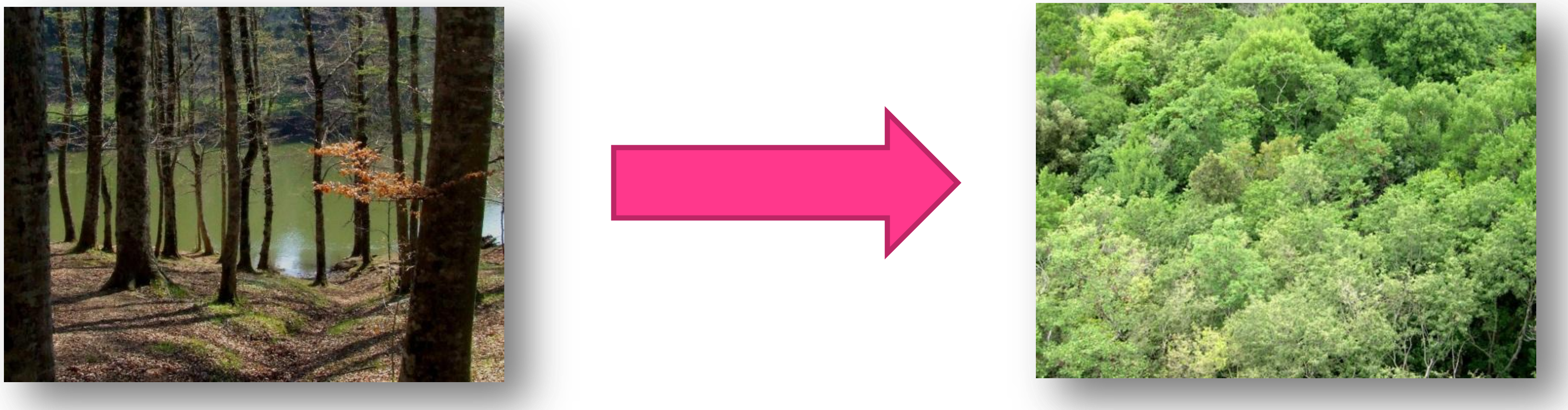
App Analytics

Mopapp.com

- Measure of Performance application (**mopapp.com**)
- Web-based analytics tool integrated with most stores
  - App Store, Android Market, AppWorld, WP7 Marketplace
  - GetJar, Handango, MobiHand
- Control app sales and downloads
  - Get store's daily raw sales data
  - Get sales data dressed as snazzy reports and charts
- Currently under **free public beta**
  - Sign-up, start using all features, and pay nothing
  - There will always be a free plan …

Summary

- *Need a strategy far before you need m-sites or iPhone apps.*
- *So far mobile grew as a forest of individual trees; it’s time to see it as a uniform forest.*



- ***Architecting Mobile Solutions for the Enterprise**, MS Press, scheduled for Jan 2012*