Pro Smartphone Cross-Platform Development

iPhone, BlackBerry, Windows Mobile, and Android Development and Distribution

Sarah Allen,
Vidal Graupera,
Lee Lundrigan

Apress®

Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution

ISBN-13 (pbk): 978-1-4302-2868-4

ISBN-13 (electronic): 978-1-4302-2869-1

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

President and Publisher: Paul Manning
Lead Editor: Mark Beckner, Ewan Buckingham
Technical Reviewer: Fabio Claudio Ferracchiati
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Jim Markham
Copy Editor: Ralph Moore
Compositor: MacPS, LLC
Indexer: BIM Indexing & Proofreading Services
Artist: April Milne
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/info/bulksales.

The source code for this book is available to readers at www.apress.com. You will need to answer questions pertaining to this book in order to successfully download the code.

To Bruce and Jack Allen for their love and support.

—Sarah Allen

To my loving wife, Tara, and my children Maggie, Grace, James, and Kathleen.

—Vidal Graupera

Contents at a Glance

Contents

Foreword

The year 2010 is an exciting time for those of us who have worked in and around the mobile industry since before the, now, decade-old 21st century. Some have referred to this year as "The Year of the Mobile Developer." It's true that, following the creation of frictionless paths to market through Apple's App Store, Google's Android Market, and the other handset or OS app stores, developers and brands alike are pursuing a market previously limited in reach. The options of distribution of applications until recently included carrier decks, handset portals, third-party channels such as Motricity, or even one's own web site.

Carriers once dominated and controlled which applications were allowed to reach eager end users via their portals—picking winners and losers by the weight of their business development and testing processes. Distribution via carriers has been difficult and costly, requiring direct relationships with carriers. Each carrier required a new business development effort and a different set of requirements for OSes and handsets supported, along with a unique testing process. Handset portals also required major effort from business development and also required joining expensive developer programs. The third-party and web-site options for distribution were easier but required individual marketing effort by developers, and the process for users to install downloaded apps on their own was a barrier for widespread adoption. Until recently, these challenges in the business of mobile development limited experimentation and innovation by all but a few hardy souls or the largest brands with the budgets to support it. Enter Apple's App Store.

The Apple App Store not only provided a path to market, but also, a dramatic change in marketing position for developers. Apple established the new industry standard with the "There's an App for That" campaign. Suddenly, instead of choosing a device for its hardware specs, end users considered what they could do with a phone beyond make calls and send text messages. The value of a device, now, has become its ability to run lots of applications. The iPhone didn't initially include an App Store. End users drove this innovation, as is often the case. Early adopters of the iPhone broke open the OS and began to extend it's capabilities with apps, but Apple was quick enough to leverage the iTunes connection for delivering $.99 songs to delivering $.99 applications.

The app store trend didn't and couldn't have happened without the availability of more capable devices. Nokia punctuated the importance of a new class of handset commonly referred to as smartphones in 2007 by calling their advanced handsets "Multimedia Computers." Smartphone as computers has become a more common analogy as smartphones grew in processing and storage capability. The steady increase of smartphone marketshare hit an inflection point in 2008 by crossing the magical 20% penetration rate in both the UK and the US. Historically, any technology mainstreams at the 20% penetration level, which has clearly been demonstrated by experience since 2008. According to Morgan Stanley analyst Mary Meeker, the rest of the world (ROW) will reach 20% smartphone penetration in 2012.

It is in this context of explosive growth in smartphone marketshare, a frictionless path to market through device and OS app stores, and a viable business model that the authors take us to the next step—cross-platform development. Cross-platform frameworks are still in the early

stages of technology evolution, but the timing is perfect for developers to add cross-platform frameworks to their tool box.
This is especially true for web developers and those serving brands that benefit most from the tradeoffs between wide distribution and deep integration.

In Part 1, the authors provide a survey of the top development and distribution options consisting of mainly handset and OS vendors including the iPhone, Android, BlackBerry, and Windows Mobile. Part 2 follows by introducing emerging cross-platform solutions covering both proprietary and open source frameworks with an emphasis on building native applications. And finally in Part 3, the authors address techniques for using HTML to create a native look-and-feel for web applications and services.

A key thread throughout the book is recognition that mobile development is a business endeavor and opportunity. There is a presentation of how-to instructions and code samples that will be useful to those just getting started with mobile development, but the audience that will benefit most from the pragmatic vision of the authors are professional developers and agencies. Certainly, many web developers are pursuing mobile development because it's a good decision to grow their business and if their clients aren't already requesting mobile applications, they will soon.

The book isn't targeted at developers of gaming apps. While gaming is a leading category for all app stores, it's one of those categories that benefits most from deep integration into the OS or device. Cross-platform frameworks aren't likely to be the best solution for games. Productivity apps, branded apps, and some communications services such as social networking apps will benefit from using the tools and techniques covered in the book.

Several of the tools presented in the book are currently leading this emerging category. We are in the early days of cross-platform use on mobile devices. Of the estimated 17 million software developers worldwide, according to Motorola as quoted in Forbes, around 4 million of them are developing for mobile. While Rhodes, Appcelerator, and PhoneGap have been used to deliver applications via the Apple App Store, the total number of developers using these frameworks is in the low six figures. Like the early days of the web, and to some extent, still, experimentation is vital to moving the ecosystem forward. This book is an important contribution to that effort.

Debi Jones
Editor In Chief
Telefonica Developer Programs

About the Authors

Sarah Allen leads Blazing Cloud, a San Francisco consulting firm that specializes in developing leading-edge mobile and web applications. She is also co-founder and CTO of Mightyverse, a mobile startup focused on helping people communicate across languages and cultures. In both technical and leadership roles, Sarah has been developing commercial software since 1990 when she co-founded CoSA (the Company of Science & Art), which originated After Effects. She began focusing on Internet software as an engineer on Macromedia's Shockwave team in 1995. She led the development of the Shockwave Multiuser Server, and later the Flash Media Server and Flash video. An industry veteran who has also worked at Adobe, Aldus, Apple, and Laszlo Systems, Sarah was named one of the top 25 women of the Web by SF WoW (San Francisco Women of the Web) in 1998.

Website: `blazingcloud.net`
Personal Blog: `www.ultrasaurus.com`
Twitter: `@ultrasaurus`

Vidal Graupera has been developing award-winning mobile applications starting as far back as the Apple Newton in 1993. He founded and ran a successful software company that developed more than a dozen consumer applications on a variety of mobile platforms over a period of ten years. Vidal holds engineering degrees from Carnegie Melon University and the University of Southern CA, and an MBA from Santa Clara University. Vidal currently consults with clients on developing web and mobile applications.

Website: `vdggroup.com`
Personal Website: www.`vidalgraupera.com`
Twitter: `@vgraupera`

Lee Lundrigan, a founding engineer at Blazing Cloud, develops mobile applications using cross-platfrom frameworks on four platforms and Objective-C on the iPhone and iPad. He is an expert in CSS and HTML and also has experience creating dynamic UI in JavaScript. He has developed cross-browser CSS and HTML to run on iPhone, Android, BlackBerry, and Windows Mobile.

Website: `blazingcloud.net`
Personal Blog: `www.macboypro.com`

About the Technical Reviewer

Fabio Claudio Ferracchiati is a prolific writer on cutting-edge technologies. Fabio has contributed to more than a dozen books on .NET, C#, Visual Basic, and ASP.NET. He is a .NET Microsoft Certified Solution Developer (MCSD) and lives in Rome, Italy.

Acknowledgments

The authors received enthusiastic support from many of the creators of the software discussed herein. We would like to extend our thanks for technical review and enthusiastic support from the Rhomobile team: Adam Blum, Lars Burgess, Brian Moore, Evgeny Vovchenko, and Vladimir Tarasov; Brian LeRoux from Nitobi, David Richey and Jeff Haynie from Appcellerator; and Ed Spencer from Sencha. We also want to acknowledge Rupa Eichenberger's significant contribution to early technical reviews; Nola Stowe for initial work on the Android chapter; and Sarah Mei for her work on Rhodes geolocation. Jim Oser, Bruce Allen, and David Temkin each had a substantive impact in reviewing specific chapters.

Introduction

Developing mobile applications can be tricky business. Mobile developers need to use platform-specific tools and APIs and write code in different languages on different platforms. It is often hard to understand what it takes to develop and distribute an application for a specific device without actually building one. Each platform has different processes and requirements for membership in developer programs and documentation for different parts of the development process are often scattered and hard to piece together. Therefore, we have divided the book into three main topics: Platform Development and Distribution, Cross-Platform Native Frameworks, and HTML Interfaces.

Part 1: Platform Development and Distribution

In Chapters 1–5, we provide an overview of four platforms: iOS, for building iPhone, iPad, and iPod Touch applications; the Android open source platform, created by Google; Research in Motion's BlackBerry platform; and Windows Mobile from Microsoft. Each chapter follows the same outline:

- Building a Simple Hello World
- Running in the Simulator
- Adding a Browser Control
- Building for the Device
- Distribution Options and Requirements

This common outline allows for comparison across the operating systems and provides a feel for the patterns of the development process. If you decide to pursue native application development using only the vendor SDK, you will need a lot more details than any single chapter can provide, but this should provide the right amount of information to kick-off some experimentation or help make a decision about which platforms to pursue.

It is inevitable that developers create ways to share code across plaforms when CPU power is fast enough and there is sufficient memory to support some kind of abstraction and demand fuels faster time to market. We saw this with cross-platform desktop frameworks that emerged in the 1990s, and now with cross-platform mobile frameworks.

Part 2: Cross-Platform Native Frameworks

Chapters 6–9 provide an overview and examples of applications written in three popular native frameworks. In categorizing as a "native framework," we selected software that allows a common development approach across platforms but that build to an application that is indistinguishable by a user from one built with native code (as described in Part 1). Note that to build using these frameworks, you will still need the vendor SDK described in Part 1 and use vendor-specific techniques for code signing and distributions.

There are two chapters on the Rhomobile platform, one for the client-side Rhodes and one for the RhoSync server framwork. Rhodes is covered in more depth than the other two platforms: Titanium Mobile and PhoneGap. Rhodes is at version 2 at this writing, Titanium v1.2 and PhoneGap 0.9. As with the rest of the book, these chapters are designed to provide a feel for what it is like to develop for each platform, to kick-start some experimentation, and aid in deciding what platform to spend more time with.

Part 3: HTML Interfaces

You can use the technique of adding a browser control in combination with the HTML and CSS patterns and frameworks presented in Chapters 10–14.

To develop a mobile application user interface, a mobile developer must typically learn a platform-specific language and SDK. This can become quite cumbersome if you need your application to run on more than one platform. Fortunately, there is an alternative; all smartphone platforms today include a browser control component (also known as a web view) that a developer can embed in their application that will allow them to write some or all of their app in HTML, CSS, and JavaScript.

Leveraging HTML and CSS for mobile application UI gets even better with the introduction of the mobile WebKit browser. WebKit is an open source browser engine originally created by Apple. WebKit introduces a partial implemention of HTML5 and CSS3 with full support for HTML4 and partial implementation CSS2. Note that as of this writing, HTML5 and CSS3 are still in "working draft;" however, these emerging standards have been aggressively adopted by multiple web browsers and the latest versions of WebKit-based browsers include most HTML5 and CSS3 features. The WebKit mobile browser is currently the native browser for iPhone/iPod Touch/iPad, Android, Palm, and many Symbian phones. BlackBerry plans to catch up with its own WebKit-based browser, recently demonstrated at Mobile World Congress in February 2010. Windows Mobile ships with an IE-based browser, which includes a better implemention of CSS1 and 2 compared with BlackBerry, but still has limitations. It is possible, though sometimes challenging, to build cross-platform UI in HTML and CSS that works across WebKit, mobile IE, and BlackBerry broswers. The most challenging part is differing levels of support for current HTML and CSS standards.

Chapter 1

The Smartphone is the New PC

The mobile phone is the new personal computer. The desktop computer is not going away, but the smartphone market is growing fast. Phones are being used as computers by more people and for more purposes. Smartphones are generally cheaper than computers, more convenient because of their portability, and often more useful with the context provided by geolocation.

Already there are more mobile phones than computers connected to the Internet. While a minority of those phones would be considered smartphones, we're seeing a fast-moving landscape where today's high-end phones become next year's mid-range or even low-end phones. With profits from applications growing, we'll see continued subsidies of the hardware and operating systems by manufacturers and carriers, keeping new phones cheap or free.

We're seeing a change in how people use computers. Desktop applications that we use most frequently are centered around communications, rather than the more traditional personal computer task of document creation. In the business world, we file expense reports, approve decisions, or comment on proposals. As consumers, we read reviews, send short notes to friends, and share photos. E-mail is the killer app of the late 20th century, not the word processor or spreadsheet. Both in the business world and in our personal lives, these communication-centered tasks translate effectively into mobile applications.

As smartphones gain widespread adoption, the desktop computer will be relegated to the specialist and elite professional, much as the mini-computer and supercomputer are today. Many of the routine tasks we currently perform on a desktop or laptop, we will be able to accomplish on a smartphone. More importantly, new applications will meet the needs of people who don't use a computer today. Software development will shift toward mobile development as the majority of people who use computers will use them indirectly through a mobile phone. The center of gravity of the software industry will be mobilized.

Application Marketplace

In September 2009, Apple announced that more than two billion applications had been downloaded from its App Store. With more than 100,000 applications available, Apple has transformed the mobile phone market by dramatically increasing consumer spending on applications and successfully shifting independent developer mindshare toward mobile application development. By the end of 2009, Google Android's open platform was reported to have over 20,000 apps in the Android Market online store.[1]

Mobile applications are not new. Even in the late 90s, mobile development was considered to be a hot market. While there were independent application developers and most of the high-end phones supported the installation of applications, the process of application install was awkward and most end users did not add applications to their phone. Examples of early smartphone and PDA devices from this era included the Apple Newton Message Pad, Palm Pilot, Handspring (and later Palm) Treo, Windows Pocket PC, and others. Almost all mobile developers worked directly or indirectly for the carriers.

The iPhone revitalized the landscape for mobile application development. Apple created an easy-to-use interface for purchasing and installing third-party applications, and more importantly, promoted that capability to their users and prospective customers.

Smartphone operating systems actively innovate to keep up with advances in hardware and ease development with improved tools and APIs. As we've seen with the iPhone App Store, often the most significant innovations are not purely technical. The App Store reduced barriers to application development by providing easy access to distribution. Unsurprisingly, people develop more apps when there is an accessible market and distribution channel. Google's App Market, Blackberry App World, and Windows Marketplace for Mobile are likely to drive the success of existing applications for those operating systems and draw new developers as well.

Increase in Mobile Usage and Trend Toward Smartphones

Six in 10 people around the world now have cell-phone subscriptions, according to a 2009 UN Report,[2] which surpasses the quarter of the world's population with a computer at home. Smartphones are still a small minority of mobile phones, but growth is strong and the numbers are particularly interesting when compared to computer sales. Mobile Handset DesignLine reports that smartphones represent 14% of global device sales, but Gartner projections note that smartphone shipments will overtake unit

---

[1] http://www.techworld.com.au/article/330111/android_market_hits_20_000_apps_milestone

[2] International Telecommunications Union (a UN agency), "The World in 2009: ICT facts and figures," http://www.itu.int/newsroom/press_releases/2009/39.html, 2009.

sales of notebook computers in 2009 and that by 2012, smartphones will grow to 37% of mobile device sales.[3]

Looking at how people use their mobile phones today suggests patterns of behavior that will drive smartphone sales in the future. Increasingly, people are using their phones for more than phone calls: web browsing and the use of other mobile applications are growing. Market researcher comScore reports that global mobile Internet usage more than doubled between January 2008 and January 2009.[4] In Africa, a recent sharp increase in mobile phone adoption is attributed to the use of phones for banking and sending money to relatives via text messaging.

Even lower-end mobile phones typically bundle web browser, e-mail, and text messaging, but the power of the smartphones enables a wider array of applications. Smartphones are not just little computers that fit in your pocket. For many applications, they are actually more powerful devices than a laptop due to their built-in capabilities of camera, connectedness, and geolocation. Business people who can afford a laptop often prefer the longer-lasting battery power and portability of the smaller device. In an *Information Week* article, Alexander Wolfe collected real-world use cases of businesses adopting smartphones for applications that used to be only accessible with a desktop or laptop computer:

> At Dreyer's Grand Ice Cream, the Palm Treo 750 is being used by some 50 field sales representatives to access the company's back-end CRM database.
>
> The company's field-sales reps tried laptops and tablet PCs, but their battery life was too short and rebooting took too much time on sales calls, which number 20 to 25 a day, says Mike Corby, director of direct store delivery. Dreyer's reps also found the laptops to be too bulky to tote around, "not to mention the theft worries with notebooks visible on their car seats."
>
> At Astra Tech, a medical device maker, some 50 sales reps access Salesforce CRM apps on their smartphones. "Salespeople say they now check yesterday's sold or returned products plus the overall revenue trends, five minutes before meeting with a customer," says Fredrik Widarsson, Astra Tech's sales technology manager, who led the deployment on Windows Mobile smartphones (and is testing the app on iPhones). "Another interesting effect is that once a salesperson is back home for the day, the reporting part of their job is done. During waiting

---

[3] Christoph Hammerschmidt, "Smartphone market boom risky for PC vendors, market researchers warn," http://www.mobilehandsetdesignline.com/news/221300005;jsessionid=1JYPKFPGNOGE1QE1GHPCKH4ATMY32JVN, October 28, 2009.

[4] Dawn Kawamoto, "Mobile Internet usage more than doubles in January," http://news.cnet.com/8301-1035_3-10197136-94.html

> periods throughout the day, they put notes into the CRM system, using their smartphone."[5]

In a recent article by Gary Kim, Forrester analyst Julie Ask identifies three things as the killer advantages of mobile devices: "immediacy, simplicity, and context."[6] When those are combined with usefulness, we're going to start to see a different flavor of software application emerge that will transform the way we use mobile phones. The use of software applications as "computing" will become archaic. The age of software as communications medium will have arrived.

What is a Smartphone?

Cell phones today are generally divided between the low-end "feature phones" and higher-end "smartphones." A smartphone has a QWERTY keyboard (either a physical keyboard or soft keyboard like the iPhone or BlackBerry Storm) and is more powerful than the feature phone with larger, high-resolution screens and more device capabilities.

Smartphone Landscape

Relative to desktop computers, smartphones have a diverse set of operating systems (see Table 1–1). Moreover, unlike desktop operating systems, the OS in mobile computing typically determines the programming language that developers must use.

When developing an application for the desktop, such as Microsoft Word or Adobe PhotoShop, application developers create their core application in a language such as C++ and share that core code across platforms, but then use platform-specific APIs to access the filesystem and develop the user interface. In the 1990s, a number of cross-platform desktop frameworks emerged, making it easier for companies to develop a single codebase that they could compile for each target platform (typically, just Mac and Windows). For mobile development, this is a bigger challenge.

---

[5] Wolfe, Alexander. "Is The Smartphone Your Next Computer?" October 4, 2008. http://www.informationweek.com/news/personal_tech/smartphones/showArticle.jhtml?articleID=210605369, March 16, 2009.

[6] Gary Kim, "Can Mobile Devices Replace PCs?" http://fixed-mobile-convergence.tmcnet.com/topics/mobile-communications/articles/66939-mobile-devices-replace-pcs.htm, October 19, 2009.

Table 1–1. *Smartphone Operating Systems and Languages*

OS	Symbian	RIM BlackBerry	Apple iPhone	Windows Mobile	Google Android	Palm webOS
Language	C++	Java	Objective-C	C#	Java	Javascript

Even focusing only on smartphones, there are four major operating systems that make up over 90% of the market: Symbian, RIM BlackBerry, Apple iPhone, and Windows Mobile, with the rest of the market shared by Linux and emerging mobile operating systems, Google Android and Palm's webOS. For most of these operating systems, there is a native development language, which is required to develop optimally for that platform, as illustrated in Table 1–1. While it is possible to develop using other languages, typically there are drawbacks or limitations in doing so. For example, you can develop a Java application for Symbian; however, several native APIs are unavailable for accessing device capabilities. Besides the differences in languages, the software development kits (SDKs) and paradigms for developing applications are different across each platform. While the device capabilities are almost identical, such as geolocation, camera, access to contacts, and offline storage, the specific APIs to access these capabilities are different on each platform.

Cross-Platform Frameworks

The fast-growing market for applications drives the need for faster time to market. Just as market opportunities led vendors to release cross-platform applications on desktop computers in the 1990s, mobile applications are more frequently available across devices. Operating systems vendors vie for the attention of developers and application vendors, but improve their tools incrementally. Where such dramatic challenges exist in developing across multiple platforms, it is natural for third party cross-platform frameworks to emerge.

The innovation in cross-platform frameworks for smartphone applications surpasses the patterns of abstraction seen in the cross-platform desktop frameworks of the 1990s. These new smartphone frameworks are influenced by the rapid application development techniques we are seeing in web development today. There are three specific techniques in web application development that are borrowed for these non-web frameworks: 1) layout with mark-up (HTML/CSS); 2) using URLs to identify screen layouts and visual state; and 3) incorporating dynamic languages, such as Javscript and Ruby.

A generation of designers and user interface developers are fluent in HTML and CSS for layout and construction of visual elements. Additionally, addressing each screen by a unique name in a sensible hierarchy (URL) with a systemized way of defining connections between them (links and form posts) has created a *lingua franca* understood by visual and interactions designers, information architects, and programmers alike. This common language and its standard implementation patterns led to the development of frameworks and libraries that significantly speed application development on the Web. These patterns are now being applied to the development of

mobile applications as common techniques by individual developers as well as in cross-platform frameworks.

The new cross-platform frameworks (and the native Palm webOS) leverage these skills using an embedded web browser as the mechanism for displaying application UI. This is combined with a native application that transforms URL requests into the rendering of application screens simulating the web environment in the context of a disconnected mobile application.

The Branded Experience of Mobile Applications

New cross-platform smartphone frameworks support a trend where mobile applications, such as web applications, are a branded experience. The Web is a varied, diverse place, where the lines between application functionality, content, and branding blur. Web applications do not express the native operating systems of Mac, Windows, or whatever desktop happens to host the browser. Web applications are liberal with color and graphics, defying the UI conventions of the desktop as well as avoiding the blue underlined links of the early Web that Jacob Nielson erroneously identified as the key to the Web's usability.

As an example, the NBA released its NBA League Pass Mobile app for both iPhone and Android. "Multiplatform is a key tenet of our philosophy," said Bryan Perez, GM of NBA Digital. "We want our content available to as many fans as possible, and with more and more carriers adopting Android around the world, it's important to be there now."[7] Most businesses simply can't afford to focus on the niche of a single operating system or device. To reach customers, more companies are developing mobile applications, and the customers they want to reach are divided across the wide array of mobile platforms. Despite the challenges, businesses are driven to communicate with their customers through their mobile phones because of the enormous opportunity presented by such connectedness.

It may be effective shorthand to say that smartphones are the new personal computer; however, in reality they represent a new communications medium. This book covers frameworks and toolkits that make it easier than ever before to develop applications for multiple mobile platforms simultaneously. Leveraging these tools, you can take advantage of the widespread adoption of smartphone devices to broaden the reach of your business.

To provide some perspective on how application interfaces vary across platform, Figures 1–1 to 1–5 illustrate how two applications, WorldMate and Facebook, are realized across various platforms. These specific applications are not implemented using cross-platform frameworks, but are included to provide context on design decisions made in cross-platform implementation. As you will see, the two applications look quite

---

[7] Todd Wasserman, "So, Do You Need to Develop an Android App Too Now?," `http://www.brandweek.com/bw/content_display/news-and-features/direct/e3iebae8a5c132016bcab88e37bc3948a44`, October 31, 2009.

different from each other, even on the same platform. As is typical, these mobile applications choose a color scheme that is consistent with their brand, rather than adhering to defaults provided by the smartphone operating system.

Figure 1–1. *WorldMate iPhone*

Figure 1–2. *WorldMate 2009 Symbian*

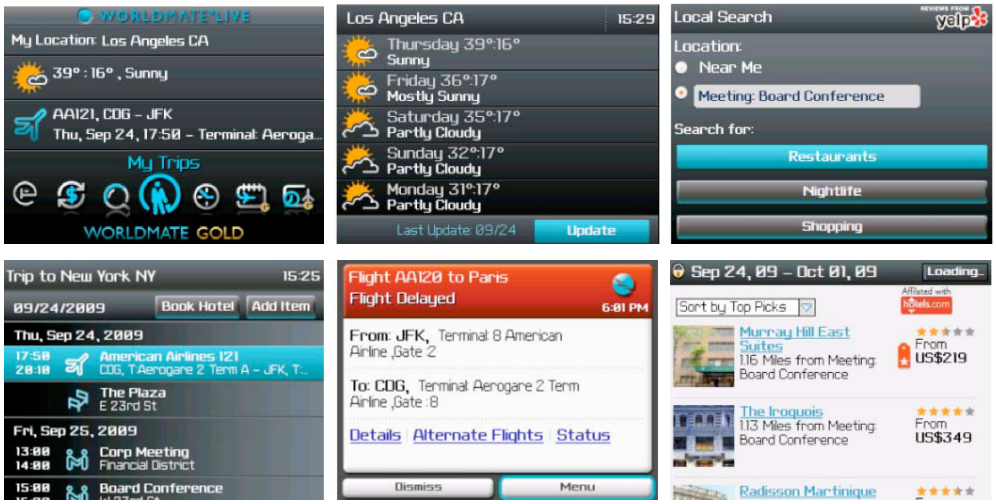

Figure 1–3. *WorldMate BlackBerry*

Figure 1–4. *Facebook BlackBerry*

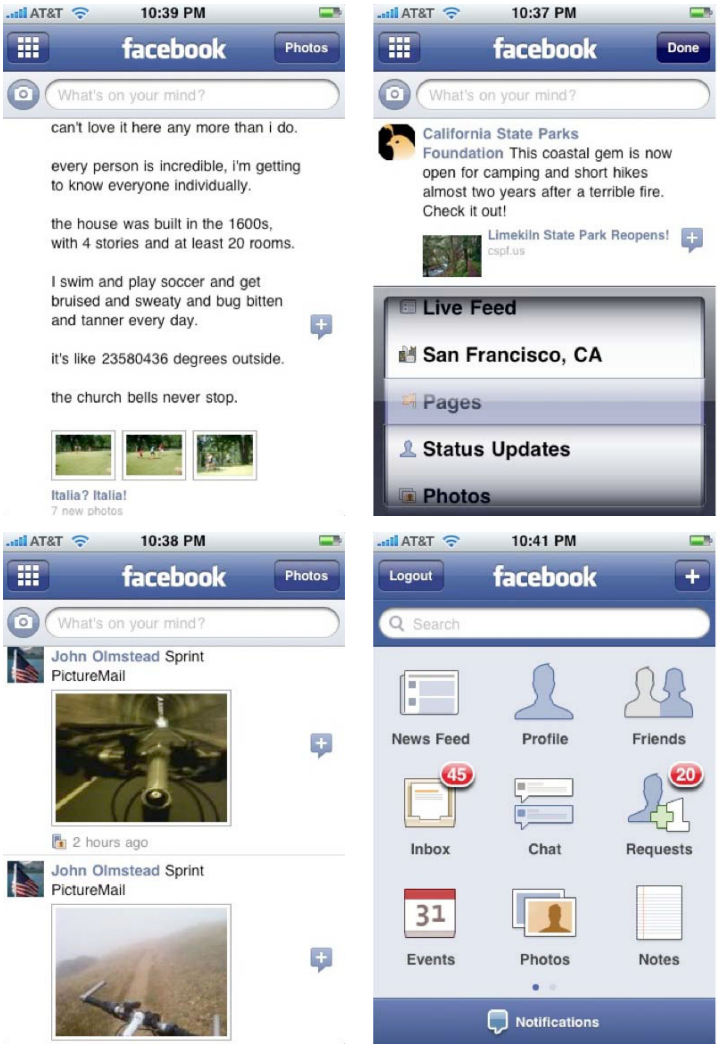

Figure 1–5. *Facebook iPhone*

Cross-Platform Development

Frequently, the industry produces multiple platforms that essentially provide the same solutions for different market segments. In the 1990s, Microsoft Windows and the Apple Macintosh provided GUI platforms with windows, mouse input, menus, and so forth. Software vendors needed to create applications for the both platforms and, inevitably software developers created libraries and frameworks that abstracted the differences, making it easier to develop one application that ran across platforms. In the 2000s, as more applications moved to the Web and browser syntax diverged, software developers created cross-platform libraries and frameworks, such as jQuery, Dojo, and OpenLaszlo. When there exists both a market for applications and enough processor speed and

memory to support a layer of abstraction, developers naturally create cross-platform tools to speed time to market and reduce maintenance costs.

With the phenomenal growth of mobile, which has seen broad adoption across a diverse array of platforms, it is inevitable that software developers would create cross-platform mobile solutions. However, the challenge with mobile operating systems today is the diverse set of languages, in addition to platform-specific API syntax. Mobile cross-platform frameworks are addressing that challenge by leveraging the ubiquitous browser Javascript or scripting languages such as Lua or Ruby.

Web Techniques

We are seeing the influence of web development on emergent cross-platform techniques for mobile. Before any cross-platform frameworks existed, many developers found that embedding Web UI in a native application was a practical way to develop mobile applications quickly and make cross-platform applications easier to maintain. The user interface for mobile applications tends to be presented as a series of screens. From a high level, the mobile UI can be thought of as having the same flow-of-control as a traditional web site or web application.

It is common in a mobile application for every click to display a new screen, just as a click in a traditional web application displays a new page. By structuring the UI of the mobile application such as a web application, the coding can be simplified. By actually using Web UI controls, the implementation of the user interface can be created with a single source that renders and behaves appropriately across platforms. Also, it is much easier to hire designers and UI developers who are familiar with HTML and CSS than for any specific mobile platform, let alone finding developers who can develop a UI across multiple platforms using native toolkits.

What does it mean to have a web application architecture for an app that may not even access the network? Every smartphone platform has a web browser UI control that can be embedded into an application just like a button or a check box. By placing a web browser control in the application that is the full size of the screen, the entire UI of the application may be implemented in HTML. In reality, this has nothing to do with the Web, and everything to do with the sophisticated layout and visual design flexibility that even a bare-bones web browser is capable of rendering.

Cross-Platform Frameworks

In the past few years, many cross-platform frameworks have emerged. There has been an explosion of activity in this area as mobile devices become faster and more widely adopted, and particularly with a fast-growing market for applications. This book covers many of the popular frameworks that are focused on application development. The frameworks fall into two categories: those that let you create a native mobile application using cross-platform APIs, and HTML/CSS/Javascript frameworks that let you build cross-platform interfaces that run in a web browser. It is common practice to combine

these to create cross-platform native applications. This book covers the native cross-platform frameworks of Rhodes, PhoneGap, and Titanium. These are listed below along with a number of frameworks that are not covered in this book.

- **Rhodes and RhoSync** from Rhomobile. Use Ruby for cross-platform business logic in this MVC framework and leverage HTML, CSS, and JavaScript for the UI. The optional RhoSync server supports synchronization of client-server data. With Rhodes, you can build applications for iPhone/iPad, Android, BlackBerry, and Windows Mobile. The client framework is MIT License; their RhoSync server framework is GPL with a commercial option. `http://rhomobile.com/`
- **PhoneGap** from Nitobi. Use HTML, CSS, and Javascript along with projects and libraries that support native application development to create applications that run on iPhone/iPad, Android, BlackBerry, Palm, and Symbian. Open-source MIT License. `http://www.phonegap.com/`
- **Titanium Mobile** from Appcelerator. Use JavaScript with custom APIs to build native applications for iPhone and Android. Titanium is an open-source framework, released under the Apache 2 license. `http://www.appcelerator.com`
- **QuickConnectFamily**. Use HTML, CSS, and JavaScript to build an application that runs on iPhone/iPad, Android, BlackBerry, and WebOS. The QuickConnectFamily templates give you access to behavior normally restricted to "native" apps. You can have full database access across all the supported platforms. `http://www.quickconnectfamily.org/`
- **Bedrock** from Metismo. A cross compiler converts your J2ME source code to native C++, simultaneously deploying your product to Android, iPhone, BREW, Windows Mobile, and more. Bedrock is a set of proprietary libraries and tools. `http://www.metismo.com`
- **Corona**. Develop using the Lua scripting language for native iPhone, iPad, and Android apps. Corona is a proprietary framework. `http://anscamobile.com/corona/`
- **MoSync SDK**. Use C or C++ to develop using MoSync libraries to build for Symbian, Windows Mobile, j2me, Moblin, and Android. MoSync is a proprietary framework. `http://www.mosync.com/`
- **Qt Mobility**. Use C++ and Qt APIs to target S60, Windows CE, and Maemo. Qt (pronounced "cute") is a cross-platform application development framework widely used for the development of GUI programs. The Qt mobility project moves it to mobile platforms. It is distributed as open source under the LGPL. `http://labs.trolltech.com/page/Projects/QtMobility`

- **Adobe Flash Lite**. Use ActionScript, a JavaScript-like proprietary scripting language, to build cross-platform application files (SWF) that will run as applications on a variety of devices that support Flash Lite. Adobe Flash Lite is a proprietary platform. `http://www.adobe.com/products/flashlite/`
- **Adobe AIR**. Adobe is working toward having the full features of Flash Player 10 work across a wide array of mobile devices; however, those efforts seem to be focused on web-based applications rather than native applications. Adobe AIR (as of this writing, in beta for Android) allows developers to run Flash applications outside of the mobile browser as stand-alone applications. `http://www.adobe.com/products/air/`
- **Unity**. A popular game development platform which allows you to deploy to Mac, Windows, or iPhone. Unity supports three scripting languages: JavaScript, C#, and a dialect of Python called Boo. They have announced support of Android, iPad, and PS3 to be released in Summer 2010. `http://unity3d.com/`

In addition to these frameworks for developing native applications, there are also many frameworks to create HTML, CSS, and JavaScript for mobile web applications. Many of these frameworks are little more than a collection of commonly used styles and graphical elements; however, when developing cross-platform applications using the techniques discussed in this book, these cross-platform HTML frameworks are essential time-savers. The last section of the book introduces Sencha, jqTouch, and iWebKit. These and others not covered in this book are listed as follows:

- **Sencha Touch**. A JavaScript framework that allows you to build native-looking mobile web applications in HTML5 and CSS3 for iOS and Android. Sencha Touch is an open-source framework available under the GNU GPL license v3, with a commercial license option available. `http://sencha.com`
- **JQTouch**. A JQuery plug-in for making iPhone-like applications that are optimized for Safari desktop and mobile browsers. Released under the MIT License. `http://jQTouch.com`
- **iWebKit**. An HTML5 and CSS3 framework targeting iOS native and web applications. iWebkit has been released under the GNU Lesser General Public License. `http://iWebkit.net`
- **iUI**. A JavaScript and CSS framework to build mobile web applications that run on iOS. iUI has been released under the New BSD License. `http://code.google.com/p/iui/`
- **xUI**. A lightweight JavaScript framework currently being used by PhoneGap. Currently targeting iOS applications with tentative future support for IE mobile and BlackBerry. Currently released under a GNU GPL license. `http://xuijs.com`