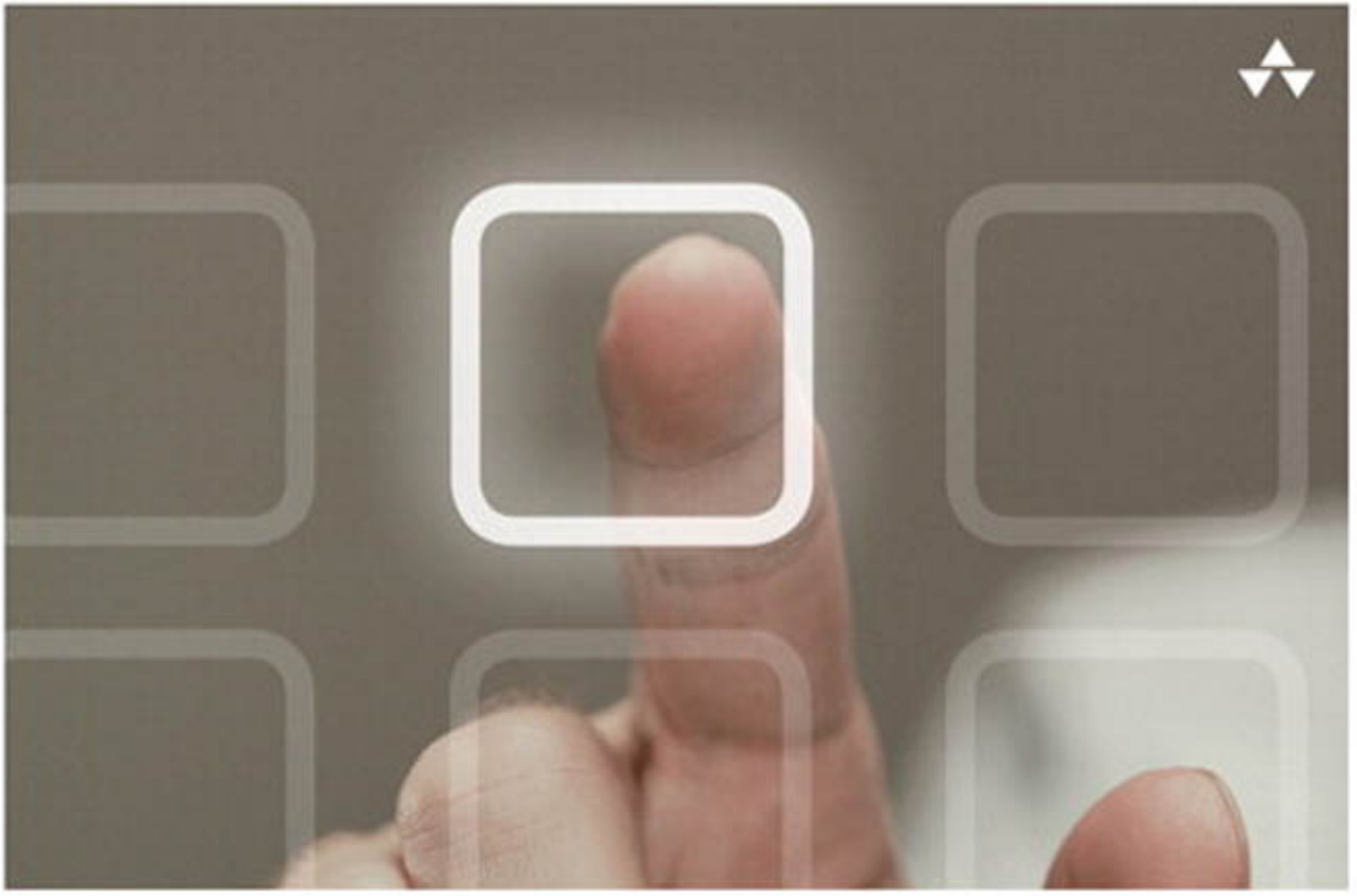


LEARNING iPad PROGRAMMING

A Hands-On Guide to Building iPad Apps with iOS 5

KIRBY TURNER
TOM HARRINGTON

Foreword by Mark Dalrymple

Praise for Learning iPad Programming

"This amazing, thorough book takes an interesting approach by working through the design and development of a simple, yet realistic iPad app from start to finish. It is refreshing to see a technical book that explains how and why without inundating you with endless toy examples or throwing you into a sea of mind-numbing details. Particularly amazing is that it does this without assuming a large amount of experience at first. Yet it covers advanced topics at sufficient depth and in a logical order for all developers to get plenty of valuable information and insight. Kirby and Tom know this material and have done a great job of introducing the various frameworks and the reasoning behind how, why, and when you would use them. I highly recommend *Learning iPad Programming* to anyone interested in developing for this amazing platform."

—Julio Barros
E-String.com

"This is a great introduction to iPad programming with a well-done sample project built throughout. It's great for beginners as well as those familiar with iPhone development looking to learn the differences in developing for the larger screen."

—Patrick Burleson
Owner, BitBQ LLC (http://bitbq.com)

"Kirby Turner and Tom Harrington's *Learning iPad Programming* provides a comprehensive introduction to one of today's hottest topics. It's a great read for the aspiring iPad programmer."

—Robert Clair
Author, *Learning Objective-C 2.0*

"*Learning iPad Programming* is now my go-to reference when developing apps for the iPad. This book is an absolute treasure trove of useful information and tips for developing on the iPad. While it's easy to think of the iPad as just a bigger iPhone, there are specific topics that need to be treated differently on the iPad, such as making best use of the larger display. *Learning iPad Programming* provides an incredible amount of depth on all areas of iPad programming and takes you from design to fully functioning application—which for me is a killer feature of the book. This should be in everyone's reference library."

—Mike Daley
Author, *Learning iOS Game Programming*
Cofounder, 71Squared.com

"A truly well-rounded book with something for every iOS developer, be they aspirant or veteran. If you are new to iOS, there is a solid foundation provided in Part I that will walk you through Objective-C, the core Apple frameworks, provisioning profiles, and making the best of Xcode. If you've been around the block but want solid insight into iPad programming, Part II has you covered: Rather than just providing canned example code, Kirby and Tom give you real code that incrementally builds and improves a real app. And if you've been working with iOS for a while, but would benefit from a walk-through of the plethora of new features that have come our way with iOS 5 and Xcode 4, dive into the chapters on Storyboards, iCloud, and Core Image. Best of all, the book is well-written and conversational, making it a joy to read. This book is stellar."

—Alexis Goldstein
Coauthor, *HTML5 & CSS3 for the Real World*

"*Learning iPad Programming* is one of the most comprehensive resources on the planet for those developing for Apple's iPad platform. In addition to coverage of the language, frameworks, and tools, it dives into features new in iOS 5, like Automatic Reference Counting, Storyboarding, and connecting your applications with iCloud. But where this book really shines is in the tutorials and the application you will build as you read through this book. Rather than being a toy that employs only off-the-shelf iOS user interface components from Interface Builder, the PhotoWheel app demonstrates custom view programming and view controller containment, nonstandard gesture/user input handling, and provides insight into how a complex iOS project comprised of multiple subsystems is assembled into a shipping application. In other words, *Learning iPad Programming* shows how to deal with the challenges you'll face in real iPad development."

—Erik Price
Senior Software Engineer, Brightcove

"A thoroughly crafted guide for learning and writing iOS applications, from the humble beginnings in Xcode and Interface Builder to creating a full-featured iPad application. There are many books that try to cover the gamut of knowledge required to take a reader from zero to app; Kirby and Tom have actually done it in this book. It is a fun and comprehensive guide to the world of developing apps for Apple's magical device."

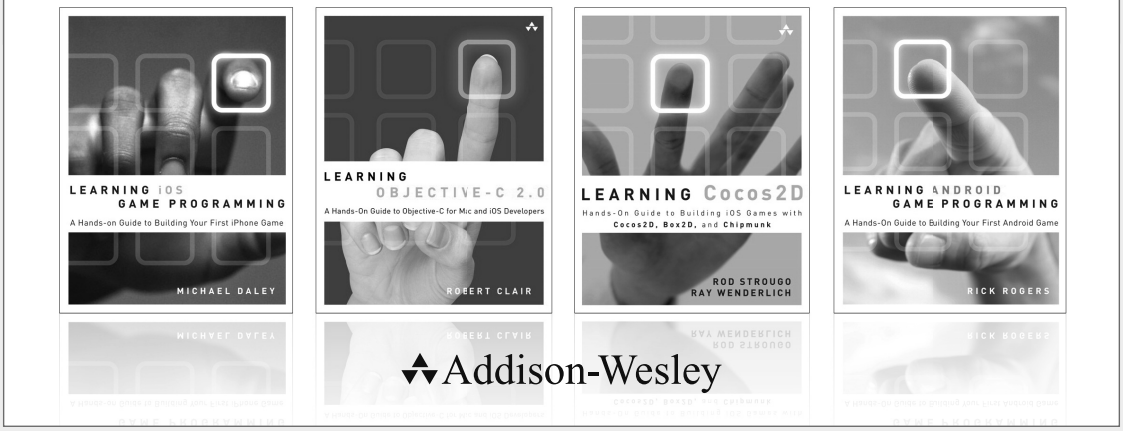
—Rod Strougo
Founder, Prop Group

"The iPad is changing the way we think about and use technology. *Learning iPad Programming* is one of the most in-depth and well-executed guides to get both new and seasoned developers up to speed on Apple's exciting new platform."

—Justin Williams
Crew Chief, Second Gear

Learning iPad Programming

Addison-Wesley Learning Series



Visit **informit.com/learningseries** for a complete list of available publications.

The **Addison-Wesley Learning Series** is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

Learning iPad Programming

A Hands-On Guide to Building iPad Apps with iOS 5

Kirby Turner
Tom Harrington

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Turner, Kirby, 1966–
Learning iPad programming : a hands-on guide to building iPad apps with iOS 5 / Kirby Turner, Tom Harrington.
p. cm.
Includes index.
ISBN 978-0-321-75040-2 (pbk. : alk. paper)
1. iPad (Computer)—Programming. 2. Application software—Development. 3. Mobile computing. 4. Laptop computers. 5. Macintosh (Computer) 6. iOS (Electronic resource) I. Harrington, Tom. II. Title.
QA76.8.I863T87 2012
005.258—dc23

2011042203

ISBN-13: 978-0-321-75040-2
ISBN-10: 0-321-75040-3
Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
First printing, December 2011

Editor-in-Chief
Mark Taub

Senior Acquisitions Editor
Chuck Toporek

Development Editor
Chuck Toporek

Managing Editor
John Fuller

Project Editor
Anna Popick

Copy Editor
Barbara Wood

Indexer
Ted Laux

Proofreader
Linda Begley

Technical Reviewers
Patrick Burleson
Matt Martel
Erik Price
Mike Shields

Publishing Coordinator
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
Rob Mauhar

❖

To Steve Jobs, who saw further than most.
— Kirby Turner and Tom Harrington

To Melanie and Rowan, for their continuous love and support.
And to my mom, the person who made me who I am today.
—Kirby Turner

To Carey, who gave me the courage to pursue my dreams.
—Tom Harrington

❖

Contents at a Glance

Contents

Foreword

Aren't books great?

Anyone who's known me for any amount of time knows I'm a total bookworm. I love books. Well-written books are one of the cheapest and fastest tools for self-education. I can remember a number of books that were hugely significant in my personal and professional development—books like *Object Oriented Software Construction* by Bertrand Meyer; Scott Knaster and Stephen Chernicoff's early Mac programming books; Dave Mark's C programming books; Robert C. Martin's horribly titled (but wonderfully full of aha! moments) *Designing Object Oriented C++ Applications Using the Booch Method*; and of course the late W. Richard Stevens's UNIX and network programming books. I remember lessons learned from these tomes, even those I read over 25 years ago.

Unfortunately not all books are created equal. I've seen some real stinkers in my time. When I first made the transition from Mac programming to iPhone programming, some of the books I got were great. And some were terrible. Really terrible, almost as if someone had filed the serial numbers off of *Instant Visual Basic Programming Guide for Complete Dummies in 24 Hours* and pasted on pictures of iPhones. There was one early iPad book that literally had an error on every page I skimmed at the bookstore. Some were just typos. Some were subtle errors, understandable if you haven't already lived in the Cocoa universe for a couple of years. Some of it was downright bad advice, obviously from someone who did not know what he or she was doing. There is a certain expectation of trust when you drop your hard-earned currency on a book, and violating that trust is unforgivable.

So, this book, *Learning iPad Programming*. Is it worth the price? Does it fall into my first category of books (awesome) or the second (unequivocally lame)? Good question. Glad you asked.

First, a good book needs to cover its topic, and cover it well. *Learning iPad Programming*, just by judging its heft, contains a lot of material. Well, assuming you've got the print version in hand. *War and Peace* weighs the same as *The Little Prince* in ebook form, so it's hard to tell. Just skim through the table of contents and you can see that it covers a lot of stuff. A metric freakload of stuff. And it's all relevant stuff. It covers the basics like installing the development tools. Model-View-Controller. Master-Detail. Table views. `UIViewController`. Navigation views. Handling device rotation. There are also more advanced topics such as consuming Web services, the media library, touch gestures, data persistence, and the raw unpleasantness that is Apple's device provisioning. And there's some cutting-edge stuff, such as storyboards, AirPrint, AirPlay,

iCloud, and Core Image. Kirby and Tom have suffered the arrows in their backs dealing with months of flaky prerelease software so that you don't have to.

Very good books are timely but not exploitative. I saw the first iPad programming book about three months after the device was announced. There was no way this book could convey the iPad gestalt to the reader because nobody had had a device in hand for that long. It was pumped out as fast as possible to hit the market, and it showed. The core of *Learning iPad Programming* has been in development for well over a year as I write this. Good books take time to achieve high levels of awesomeness.

Great books transcend their subject matter. This book is called *Learning iPad Programming*. It would be easy to assume that it just covers introductory iPad programming in a simplistic manner. "Views are cool!" "Yay! Tapping a button!" But it's more. Not many books have a single project that lives and evolves through the entire narrative. The reason not many books do this is because it is difficult to do well. Important toolkit features get shoehorned into weird places because the author didn't do enough up-front design. This book, though, takes you from design, to a throw-away prototype, to the Real Deal.

And then it goes further. Not many books talk about the inner game of design. This one does. Even fewer books talk about the inner game of debugging. Debugging is a fundamental part of the day-to-day life of a programmer, and few books devote more than a paragraph or two to it. *Learning iPad Programming* has an entire chapter on the topic, and it's much more than how to single-step with the debugger. As I was reading preproduction versions of the chapters so that I could sound halfway intelligent in this foreword, I emitted an audible "SQUEE" when I hit Chapter 25. I love debugging, and love seeing such an important topic covered in detail in what is ostensibly a beginner's book. And as you can tell, I love learning stuff. I learned some stuff from Chapter 25.

Finally, those who *create* the great books transcend the ordinary. The Mac and iPhone community is pretty small and well connected. You tend to learn quickly who the trusted players are. Many of the lame books I alluded to earlier are by individuals I had never heard of before, and never heard from again. No blogs, no appearances at conferences, no footprint on the community. Get in, crank out something, and get out.

Kirby and Tom are different. They're known entities. They have blogs. Tom has his name on a Core Data book. They've shipped products. They have happy customers. They answer questions online. They organize and speak at conferences. They organize CocoaHeads chapters. They have invested a great deal of their time into the betterment of the community. It is why I am honored and humbled that they asked me to write this foreword.

As you can probably tell, I'm pretty excited about this book. There are many excellent introductory iOS programming books. I recommend reading all of them (at least the good ones) because iOS is such a huge topic that even Kirby and Tom can't cover everything you need to know in one volume. But if you're specifically targeting the

iPad, this one is the one to get. I have the feeling it's destined to become one of those influential books for some of you out there.

—Mark Dalrymple
Cofounder of CocoaHeads, the international Mac and iPhone programmer community
Author of *Advanced Mac OS X Programming: The Big Nerd Ranch Guide*
November 12, 2011

Preface

In October 2011, Apple CEO Tim Cook shared some interesting facts about the iPad, including

- Ninety-two percent of Fortune 500 companies are testing or deploying iPads.
- Over 80 percent of U.S.-based hospitals are testing or piloting the iPad.
- Every state in the United States has some type of iPad deployment program in place or in pilot.

And the news about the iPad doesn't stop there. The FAA has approved the use of the iPad instead of paper charts for on-duty airline pilots. Without a doubt, the iPad is changing the way people think about (and use) computers today. And it continues to get better with the release of iOS 5, the latest operating system for iPad and iPhone devices.

Make no mistake, the iPad packs a punch. With its patented multi-touch interface, an onboard graphics chip, the powerful A5 processor, and 3G and/or WiFi networking, the iPad is the benchmark in a post-PC world. More important, though, is how the iPad 2 fits into the Mac/iOS ecosystem. Mac OS X Lion and iOS 5 users can use FaceTime for video chat from desktop to device. What's more, iOS 5's iMessage enables users to text from their iPad with other iPad and iPhone users. The iPad is a unique marriage of hardware and technology, and it is the Gold Standard for tablets.

This book is written with iOS 5 in mind and is aimed at new developers who want to build apps for the iPad. The book will also appeal to iPhone developers who want to learn more about how to make their apps sing on the iPad. While some people look at the iPad as just a bigger iPhone, it really isn't. There is a lot more that you as a developer can do with the iPad from a user interface perspective that you just can't do on the iPhone.

While the book will include brief discussions of iPhone programming where appropriate, the primary focus of the book is the iPad. The book highlights those areas of the iOS 5 SDK that are unique to the iPad, and it isn't a rehash of similar books targeting the iPhone. Additionally, the book covers new features in iOS 5, such as container view controllers, iCloud, and Core Image, as well as some of the great new features in Xcode 4.2, such as storyboarding. Apple has gone to great lengths to make it easier for you to develop for iOS and OS X, and the plan for this book is to make it even easier for you to get there.

What Will I Learn?

This book will teach you how to build apps specifically for the iPad, taking you step by step through the process of making a real app that is freely available in the App Store right now! The app you'll build in this book is called PhotoWheel.

> **Download the App!**
>
> You can download PhotoWheel from the App Store: **itunes.apple.com/app/photowheel/id424927196&mt=8**. The app is freely available, so go ahead—download PhotoWheel, and start playing around with it.

PhotoWheel is a spin on the Photos app that comes on every iPad (pun intended). With PhotoWheel you can organize your favorite photos into albums, share photos with family and friends over email, and view them on your TV wirelessly using AirPlay. But more important than the app is what you will learn as you build the app.

You will learn how to take advantage of the latest features in iOS 5 and Xcode, including storyboarding, Automatic Reference Counting, iCloud, and Core Image. You will learn how to leverage other iOS features such as AirPrint, AirPlay, and Grand Central Dispatch (GCD). And you will learn how to extend the boundaries of your app by communicating with Web services hosted on the Internet.

Think of this book as an epic-length tutorial, showing you how you can make a real iPad app from start to finish. You'll be coding along with the book, and we'll explain things step by step. By the time you have finished reading and working through this book, you'll have a fully functional version of PhotoWheel that you can proudly show off to friends and family (you can even share it with them, too). Best of all, you'll have confidence and the knowledge of what it takes to design, program, and distribute iPad apps of your own.

What Makes the iPad So Different?

While the iPad runs the same version of iOS that runs on the iPhone, iPod touch, and Apple TV, the iPad is significantly different from those other iOS-based devices. Each device is used differently, and iOS brings certain things to the table for each of them. For example, the version of iOS that runs in your Apple TV doesn't yet offer the same touch interface; in fact, the interface is totally different. Apple TV's UI runs as a layer on top of iOS, providing a completely different user experience.

But the iPad is so different. It is not something you can hold in the palm of your hand, like the iPhone and iPod touch. You use both hands. You swipe. You touch. You interact with it more than with most iPhone apps. It's easy to dismiss the iPad as "just a large iPhone," but it really isn't.

While the physical size is the obvious difference between the iPad and iPhone, the real difference, the difference that sets the iPad apart from the iPhone, is conceptual. The

conceptual differences stem from how an iPad application is designed and how the user interacts with the application. And the conceptual differences start with the bigger display.

Bigger Display

The iPad's bigger screen provides more than double the screen real estate found on the iPhone. This means that your application can display more information, giving you more space to work with for your user interface. A good example of this is WeatherBug.

WeatherBug HD has been designed to take full advantage of the iPad's larger screen. As you can see in Figure P.1, the iPad version of WeatherBug displays much more weather-related information on a single screen than you can get on the iPhone

Figure P.1 On the left is the WeatherBug app displayed on the iPad. The screen shot on the right is the same WeatherBug app running on the iPhone. (Used with permission of Earth Networks.)

version. Instead of your having to touch and swipe (and sometimes pray) to find additional weather information, WeatherBug HD on the iPad gives you everything you need to know right on the main screen. No additional touching or swiping needed. Of course, additional detail is still available at a touch.

Less Hierarchical

Because of the smaller screen, many iPhone applications tend to sport a hierarchical navigation system. You see this throughout many iPhone apps. The user taps an item and a new screen slides into view. Tap another item and another view slides in. To navigate back, you tap a back button, usually found in the upper left corner of the screen.

The Dropbox app illustrates the hierarchical navigation system quite well. Dropbox, for those who may not know, is an online service that allows you to store your data files, documents, and images in the cloud. Stored files are then synced across all of your computers and devices that run the Dropbox client software. Say, for example, you are working on a text document from your laptop. You save the text document to your Dropbox folder. Later you need to review the text document, so you open the same text document on your iPhone. Dropbox makes this possible.

When you use the Dropbox app on your iPhone, you see a list of files and folders sorted alphabetically. Tapping a file or folder will open it, causing the new screen to slide into view. If you open a file, you see the contents of the file. If, however, you open a folder, you see a new list of files and folders. Continue tapping folders to navigate further down the hierarchy.

To move back up the hierarchy, tap the back button in the upper left corner. The text label for this button can vary. Usually it displays the name of the previous item on the stack, but sometimes it displays the word *Back*. While the text label may vary, the style of the back button does not. The back button has a pointy left side. This almost arrowlike style conveys a sense of moving backward through the screens.

The forward and backward navigation through the hierarchy is illustrated in Figure P.2.

Dropbox is also available for the iPad. So how did the developers redesign an app that obviously requires hierarchical navigation to make it feel flatter, less hierarchical? They took advantage of an iOS object available only to the iPad called `UISplitViewController`, shown in Figure P.3.

The split view controller is a nonvisual object that controls the display of two side-by-side views. When you hold your iPad in landscape mode, the two views are displayed side by side. Rotate your iPad to portrait and the left-side view disappears. This allows the user to focus his attention on the main content displayed on the right side.

> **Note**
>
> You get hands-on writing a split-view-based application in Chapter 8, "Creating a Master-Detail App."

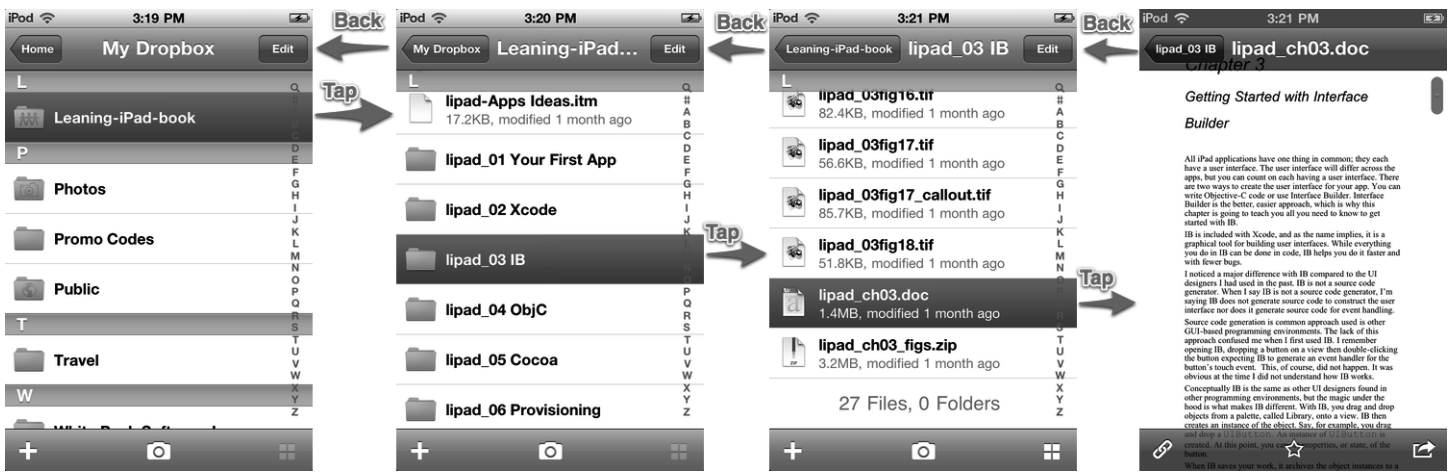


Figure P.2 Example of navigating the hierarchy of folders and files using the Dropbox app on the iPhone. You tap to move forward, or drill down, to more content, and you tap the back button to move backward.

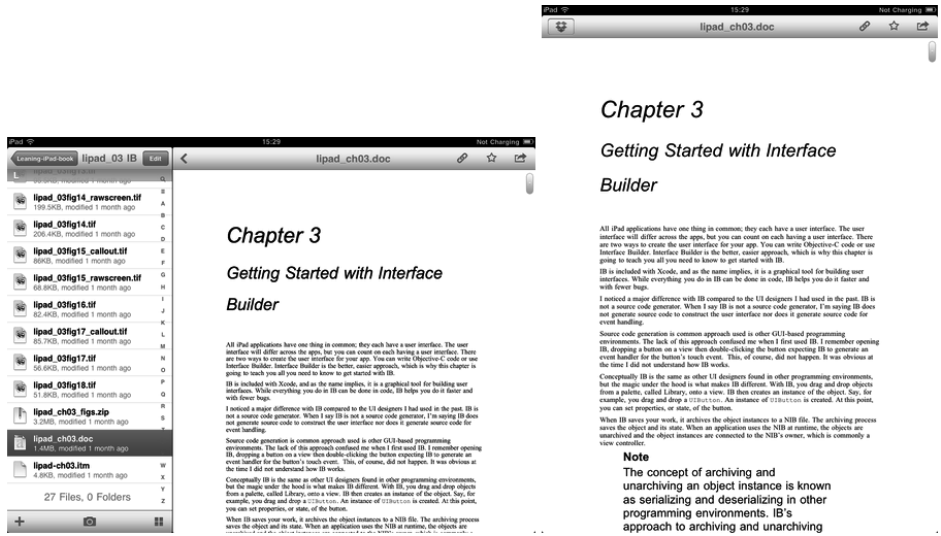


Figure P.3 Screen shots of Dropbox running on the iPad. Notice how the navigation is displayed in the left-side view when the device is held in a landscape orientation and is hidden when the iPad is rotated to portrait.

This view pattern is often called "master-detail," where the master view is displayed on the left side and the detail view is displayed on the right side. The master view is used to navigate the hierarchy of data, or in the case of Dropbox, the master view is used to navigate the list of files and folders. When you find the file you want to view, tap it in the master view and the file contents are displayed on the right in the detail view. Rotate your iPad to a portrait position to focus your attention on the file's content, hiding the master view.

Orientation Matters

Most iPhone applications support only a single orientation. Many iPhone games are played in landscape mode, while many other iPhone apps are displayed in portrait. Like the iPad, the iPhone does support rotation and orientation, but the small size of the device makes supporting different orientations unnecessary. Most users hold their iPhones in portrait mode with the Home button at the bottom when using applications, rotating to landscape only to play a game.

The iPad is different. With the iPad, users grab the device and turn it on without regard to a certain orientation. This is even truer when the iPad is not in a case. Try this little experiment...

Place your iPhone, or iPod touch, on your desk or table with the Home button pointing at 10 o'clock. Walk away or turn around. Come back to the device and pick it up. Take a look at the device as you hold it in your hand. There's a good chance that

as you picked up the device, you rotated it so that the Home button is at the bottom. You did this rotation even before turning on the device. It is an almost natural instinct to hold your iPhone with the Home button at the bottom.

Now try the same experiment, but this time use your iPad. Place it on your desk or table. Make sure the Home button is positioned away from you, say, at 10 o'clock, then walk away. Come back and pick up your iPad. Chances are good you did not rotate the device. Instead, you are likely holding your iPad in the same orientation it was in before you picked it up.

Multi-Touch Amped Up

Did you know that the iPad and iPhone support the same multi-touch interface? They do. As a matter of fact, the iOS multi-touch interface supports up to 11 simultaneous touches. This means that you can use all your fingers—and maybe one or two more if you have a friend nearby—to interact with an application.

The iPad with its larger screen makes multi-touch more feasible. While two-handed gestures have limited use on the iPhone, they can become a natural part of interacting with an iPad application. Take, for example, Apple's own Keynote app for the iPad. It takes advantage of the multi-touch interface to provide features once reserved for the point-and-click world of the desktop. Selecting multiple slides and moving them is just one example of how Keynote on the iPad maximizes the user experience with multi-touch.

So you already know that the multi-touch interface supports up to 11 simultaneous touches, but how can you confirm this? Write an iPad app that counts the number of simultaneous touches. That is exactly what Matt Legend Gemmell did. He wrote a really neat iPad app, shown in Figure P.4, that shows the number of simultaneous touches. But Matt went beyond just showing the touch count. He made the app sci-fi-looking, which also makes it fun to play with.

You can read more about Matt's iPad multi-touch sample and download the source code from his blog posting (**mattgemmell.com/2010/05/09/ipad-multi-touch**).

Another way to explore the iPad multi-touch interface is to play with Uzu for iPad, only $1.99 in the App Store (**bit.ly/learnipadprog-UzuApp**). Uzu is a "kinetic multi-touch particle visualizer" and it's highly addicting. (Figure P.5 doesn't do the app justice; you should really download and play around with Uzu if you want to see some clever use of multi-touch.)

The iPad Bridges the Gap between the Phone and the Computer

So, everyone agrees that the iPad is not an oversize iPhone. Great, glad to have you on the same page here. Now on to the larger question: Is the iPad a replacement for a laptop or desktop? No, not yet, but it's pretty darn close.

For many, the iPad represents a mobile device bridging the gap between the smartphone and a full-fledged computer, whether a laptop or desktop computer. While many individuals use the iPad for content consumption, the iPad is also used

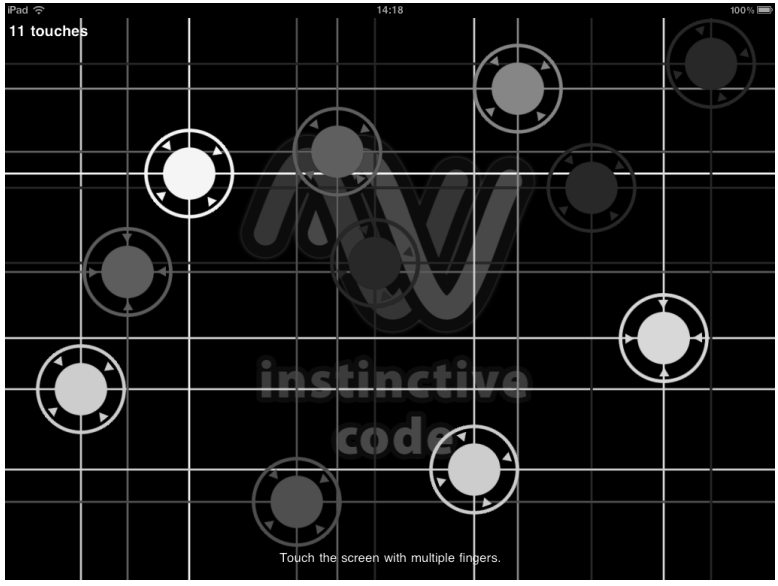


Figure P.4 Matt Legend Gemmell's multi-touch sample app for the iPad illustrating 11 touches

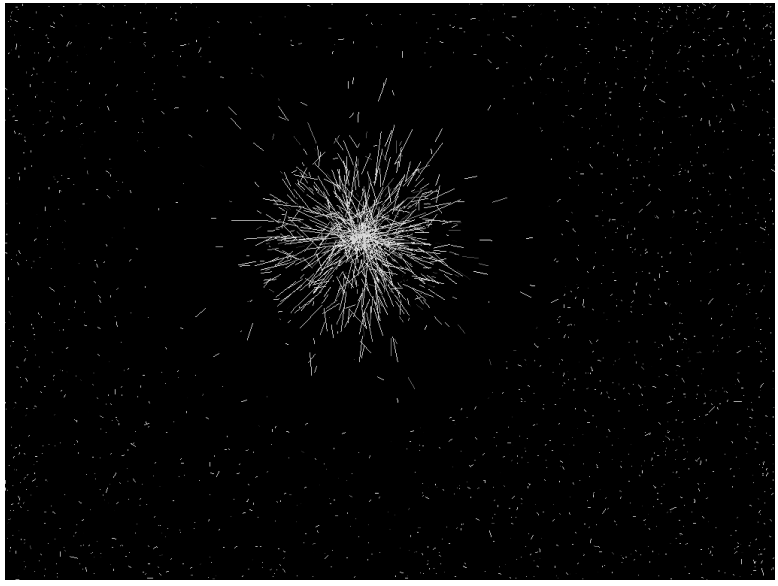


Figure P.5 Uzu, the particle visualizer for the iPad

to perform a good number of tasks previously left to the desktop or laptop computer. This causes iOS developers to rethink how to implement software concepts that have been around for eons. Word-processing software is one such concept that is seeing new life on the iPad.

The iPad opens the door to a wide range of applications not feasible on the small form factor of the iPhone. Word processing, again, is one such application that comes to mind.

While the iPhone is great for capturing quick notes, it is not ideal for writing lengthy documents. And while it is technically possible to implement a full-featured word processor on the iPhone, why would you? The screen is too small, and even in landscape mode, typing two-thumbed on a tiny screen would be less than productive. The iPhone is ideal for performing simple, quick tasks—writing a note, scheduling an event, marking a to-do item as complete—but it is less than ideal for lengthier tasks such as writing a book.

Enter the iPad

The iPad provides an experience similar to a small laptop. And when combined with a wireless keyboard, your iPad becomes a nice setup for writing long documents. I'm speaking from experience. A lot of the text in this book was originally written on an iPad. I can't imagine what writing a book on an iPhone would be like, but I know what it is like on the iPad, and it is a joy. Best of all, the iPad allows you to concentrate on a single task. This eliminates distractions and gives you better focus on the task at hand.

Organization of This Book

This book provides you with a hands-on guide for, as the book's title states, learning iPad programming. It walks you through every stage of the process, from downloading and installing the iOS SDK to submitting the first application to Apple for review.

There are 27 chapters and one appendix in the book, as follows:

- Part I, "Getting Started"

  Part I introduces you to the tools of the trade. Here you learn about developer tools such as Xcode and Interface Builder. You learn how to write code using Objective-C and the Cocoa framework. And you learn what it takes to provision your iPad as a development device.

  - Chapter 1, "Your First App"

    This chapter immediately immerses you in creating your first application. The chapter provides a step-by-step guide to creating a simple, but functional, iPad application that runs in the iPad Simulator. You'll use Xcode to create the application, which means there is also some light coding to be done, but knowledge of Objective-C is not required at this point in the book. The goal

of this chapter is for you to immediately get your hands on the tools and the code you'll use to create iPad apps.

- Chapter 2, "Getting Started with Xcode"

  Xcode is the developer's IDE (Integrated Development Environment) used to write Objective-C code for iPad applications. This chapter highlights key features of Xcode, including recommended preference settings, commonly used shortcut keys, and descriptions of the various windows you will see when using Xcode.

- Chapter 3, "Getting Started with Interface Builder"

  In this chapter, you explore Interface Builder (IB). Interface Builder is the tool used to create an application user interface (UI) with no programming required. This chapter explains how to use IB and many of its useful features. In addition, the chapter warns you about common mistakes made when using IB, such as forgetting to associate an event to an `IBAction`.

- Chapter 4, "Getting Started with Objective-C"

  This chapter introduces you to Objective-C with a brief overview of the programming language of choice for iPad programming. The goal for this chapter is not to be a comprehensive review of the programming language but instead to provide enough information to get you started writing your first real iPad app.

- Chapter 5, "Getting Started with Cocoa"

  A programming language is only as powerful as the frameworks that support it, and Cocoa provides an impressive stack of frameworks and a library that make it possible for you to build your iPad app in less time.

- Chapter 6, "Provisioning Your iPad"

  Walking down the yellow brick road to the wonderful world of iPad development can have its own set of scary moments. One of the scariest is dealing with provisioning profiles, certificates, and registering a device for testing. Xcode 4 provides improvements in this area, but it is still far from perfect. This chapter guides you through the scary forest of provisioning profiles, certificates, and device registration.

- Chapter 7, "App Design"

  You can't build an app if you don't know what you're building. This chapter shares tips on designing an application before the first line of code is ever written.

- Part II, "Building PhotoWheel"

  Part II is the heart of the book. This is where you get hands-on building a real iPad app. The app you build is no simple Hello World app. It is PhotoWheel, a full-featured photo app. In Part II, you learn everything from custom animations for view transitions to iCloud syncing to viewing your photos on TV.

- Chapter 8, "Creating a Master-Detail App"

  You start building PhotoWheel by first building a prototype of it. While building the prototype you get a chance to learn about the split-view controller used in master-detail apps.

- Chapter 9, "Using Table Views"

  In this chapter, you learn the basics of displaying data using table views. You also learn how to reorder, delete, and even edit data displayed in a table view.

- Chapter 10, "Working with Views"

  In this chapter, you dive into the world of views. Here you learn how to create a custom wheel view for displaying photos.

- Chapter 11, "Using Touch Gestures"

  In this chapter, you learn how to take advantage of the iPad's multi-touch screen. You learn to use touch gestures so that users can interact with your app.

- Chapter 12, "Adding Photos"

  PhotoWheel is about photos, so it is only natural that you need to learn how to add photos to the app. In this chapter, you learn how to retrieve photos from the Photos app library and how to take new photos using the built-in camera.

- Chapter 13, "Data Persistence"

  PhotoWheel won't be very useful if people can't save their work. There are numerous methods for storing and retrieving app data. In this chapter, you explore two of them, and you learn to use Core Data.

- Chapter 14, "Storyboarding in Xcode"

  A storyboard is an exciting new way for designing an app's user interface. In this chapter, you get hands-on with storyboarding, and you learn how you can do more with less code using Interface Builder.

- Chapter 15, "Doing More with View Controllers"

  A storyboard can take you only so far. At some point in time, you must write code to make your app really shine. In this chapter, you learn how to take advantage of view controllers to do more.

- Chapter 16, "Building the Main Screen"

  In this chapter, you dive into PhotoWheel. Prototyping is over and you have the basic UI in place with a storyboard. Now it's time to build the main screen, and that's exactly what you do in this chapter. You also learn how to use container view controllers, and you build a custom grid view that can be used in other projects.

- Chapter 17, "Creating a Photo Browser"

  In this chapter, you learn how to use a scroll view to create a full-screen photo browser. You also learn how to use a pinch gesture to zoom in and out on a photo.

- Chapter 18, "Supporting Device Rotation"

  Users expect iPad apps to display properly regardless of how the device is being held. A user may hold his iPad with the Home button on the left or right, or maybe on the top or bottom. And it is your job to ensure that your app displays properly regardless. That is what you learn in this chapter: how to support device rotation.

- Chapter 19, "Printing with AirPrint"

  This chapter gets straight to the point and teaches you how to print from your app using AirPrint.

- Chapter 20, "Sending Email"

  Virtually everyone has an email account these days, and everyone loves looking at photos. So it only makes sense that PhotoWheel users want to share photos with family and friends using email. In this chapter, you learn how to send email from your app.

- Chapter 21, "Web Services"

  Adding photos already found on your iPad to PhotoWheel is a nice exercise, but many people keep their photos stored elsewhere. In this chapter, you learn how to communicate between an iPad app and a Web server to search for and download photos from Flickr.

- Chapter 22, "Syncing with iCloud"

  Many people have multiple iOS devices, and it would be great if they could use PhotoWheel with the same data on all of them. Syncing can be hard, but with iCloud it is a lot easier than it could be. In this chapter, we add online syncing of photos and albums.

- Chapter 23, "Producing a Slideshow with AirPlay"

  The iPad has a great screen, but you might want to show photos to a group, and it's awkward to gather everyone around a hand-held device. In this chapter, you see how to make use of external wireless displays—a large TV set, maybe—from an iPad app. And you do it using AirPlay, so you don't need to run cables across the room.

- Chapter 24, "Visual Effects with Core Image"

  Core Image is an amazingly cool framework for analyzing and changing images. As if color effects and automatic photo enhancement weren't enough, you can also use Core Data Image to locate the faces of any people in the picture. You add all of this to PhotoWheel in a convenient user interface that allows people to preview effects before committing to them.

- Part III, "The Finishing Touches"

  In the final part of the book, you learn tips on debugging your app. But more important, you learn how to distribute your app to others.

- Chapter 25, "Debugging"

  At this point you know how to create an iPad application, but what happens when a problem occurs? This chapter is devoted to application debugging. It introduces you to the GDB and shows you how to turn on and off breakpoints, and how to use sounds to debug. The chapter also introduces you to more advanced debugging techniques such as using Instruments to track down memory leaks.

- Chapter 26, "Distributing Your App"

  The application is written. It has been debugged and tested. The next step is getting the application into the hands of users. This chapter explores the options for distributing iPad applications, focusing on the two most commonly used distribution methods: Ad Hoc and App Store.

- Chapter 27, "The Final Word"

  The book ends with some final words of encouragement for the new iPad programmer.

- Appendix A, "Installing the Developer Tools"

  This appendix walks you through the steps needed to start programming for the iPad. This includes setting up an iOS developer account, downloading the iOS SDK, and installing the developer tools on your Mac.

Learning iPad Programming takes you from app design to the App Store. Along the way you learn about the developer tools, the programming language, and the frameworks. But more important, you learn how to build a full-featured iPad app that you can show off to family and friends.

Audience for This Book

This book is intended for programmers who are new to the iOS platform and want to learn how to write applications that target the iPad. The book assumes that you are new to iPad programming and have little to no experience with Xcode and the Objective-C programming language. However, the book assumes that you have some prior programming experience with other tools and programming languages. It is not intended for individuals with absolutely no prior programming experience.

This book is targeted to programmers who want to learn how to develop sophisticated applications for the iPad using iOS 5. You are expected to have a Mac on which you can program using Xcode and Interface Builder, as well as an iOS developer account and an iPad. Some programming experience is helpful, particularly knowledge of C, although there is a chapter on object-oriented programming with Objective-C to give you a head start.

The book will also appeal to experienced iOS developers, people who have programmed and have shipped apps to the App Store for the iPhone and iPod touch. If

you are an experienced reader, you can skip over the basics, if you so choose, and quickly get to work on the example projects used throughout the book.

Getting the Source Code for PhotoWheel

The source code from each chapter as well as the source code for PhotoWheel as presented in this book is available from the book's Web site (**learnipadprogramming.com/source-code/**). Work on PhotoWheel doesn't stop at the end of this book either. There is so much more to do with the app and so much more to learn. The most up-to-date source code is available on github (**github.com/kirbyt/PhotoWheel**).

You will also find more how-to articles and tips for improving PhotoWheel at the book's blog site (**learnipadprogramming.com/blog/**).

And should you have additional questions, or want to report a bug or contribute a new feature to PhotoWheel, feel free to send email to kirby@whitepeaksoftware.com or tph@atomicbird.com, or send a message to @kirbyt or @atomicbird on Twitter.

There is plenty of code to review throughout the book along with exercises for you to try, so it is assumed that you have access to the Apple developer tools such as Xcode and the iOS SDK. Both of these can be downloaded from the Apple iOS Dev Center.[1]

Artwork Provided by

Matt McCray is the swell guy who provided the artwork in PhotoWheel. Reach out to Matt if you're looking for a designer for your next app. He can be reached at matt@elucidata.net and his Web site is at **www.elucidata.net**.

1. Apple's iOS Dev Center: **developer.apple.com/ios.**

Acknowledgments

As for any book that gets written, there's an entire cast and crew who remain hidden from the limelight; please take a moment to hear us out as we thank the supporting cast...

Acknowledgments from Kirby Turner

I want to first thank my wife, Melanie, and my son, Rowan, for their support and patience while I focused on completing this book. I want to thank Tom for agreeing to be coauthor during the final stages of this book, which would have been delayed even more if not for his help. I want to give a huge THANKS to Chuck Toporek for giving me this opportunity to write a book. And, of course, I want to say thanks to the technical reviewers and the production team for all their hard work in a short amount of time.

I want to send an extra thanks to Daft Punk for the *TRON: Legacy* album. It was the soundtrack for most of this book.

I want to thank Steve Jobs and the amazing Apple engineers for bringing the fun back to programming for me. Last, I want to thank the Mac and iOS developer community. None of this would be possible if not for the passion and spirit of this unique community.

Acknowledgments from Tom Harrington

I'd like to thank Kirby for inviting me to be part of this book. I'd especially like to thank Chuck Toporek, our technical reviewers, and the rest of the production team for all their hard work making me look good in print. Kirby and I wrote this book while iOS 5 was still in beta testing, and we often found it necessary to make revisions before we were finished, just to keep up with the pace of change in iOS and the developer tools. Everyone involved has done a great job dealing with the challenges of writing a book on a topic that's constantly in flux.

On a closely related note, thanks to everyone at Apple for their hard work on iOS 5 and the iPad. Without them we wouldn't have such a cool topic to write about.

Finally, as of this writing we've only just learned that Steve Jobs has passed away after a long and highly successful tenure as CEO of Apple. I got started writing software nearly 30 years ago on an Apple II, and it set the tone for my future career. Thanks for all you've done over the years, Steve.

About the Authors

This book is brought to you by...

Kirby Turner

Kirby Turner is an independent software developer and business owner focusing on Mac and iOS programming. Kirby has been programming since the early 1980s. He sells his own apps through his company, White Peak Software, and he does contract programming when time allows.

When Kirby is not sitting behind the keyboard, he can be found hanging out with his wife, Melanie, and son, Rowan, hiking the mountains of New England, kayaking the waters in and around Salem, Massachusetts, and snowboarding down mountains in search of perfect powder.

Tom Harrington

Tom Harrington switched from writing software for embedded systems and Linux to Mac OS X in 2002 when he started Atomic Bird, LLC. After six years of developing highly regarded Mac software, he moved to iPhone in 2008. He now develops iOS software on a contract basis for a variety of clients. In addition to this book, Tom is coauthor of *Core Data for iOS*. Tom also organizes iOS developer events in Colorado. When not writing software, he can often be found on his mountain bike. You can find Tom on Twitter as @atomicbird.

6

Provisioning Your iPad

Before you ship the next killer app, you must test it on an iPad. Using the iPad Simulator to test and debug your application will only get you so far. That's why it's important to always test your app on the real thing. But before your app will run on a real device, you must set up your iPad as a development device. This is where provisioning comes in, and that's exactly what you will do in this chapter: provision your iPad for development purposes.

The steps involved are tedious. Fortunately you do not have to repeat them often, only a few times a year. And Apple is continuously improving the provisioning process. Provisioning a device today using Xcode is easy-peasy compared to what one had to do back in the day … the dark days of 2008.

About the iOS Provisioning Portal

The iOS Provisioning Portal is the Web site, shown in Figure 6.1, used to request and download certificates, register device IDs, create App IDs, and create and download provisioning profiles. The Web site and all of its features are not available to everyone. You have access to the portal if you are one of the following:

- An individual registered as a paid member of the iOS Developer Program. You can be registered as an individual or a company.
- An individual registered as a team member for a company that is a paid member of the iOS Developer Program. Depending on your team member role, your access to the Provisioning Portal may be limited. There is no cost to the individual team member, and team members have access to all of the same developer resources that a paid member has, courtesy of the paid company membership.

If you are not a paid member or a team member of a company, you will not have access to the iOS Provisioning Portal. Without access, you will not be able to provision your device.

> **Note**
>
> You can read more about joining the iOS Developer Program in Appendix A, *Installing the Developer Tools*.

Figure 6.1 iOS Provisioning Portal home page

iOS Developer Program Team Roles

When you join the iOS Developer Program as a paid member, you have the option to join as an individual or a company. The main difference between the two is that a company can include team members and an individual cannot. As a company, you can invite other developers to join your team and assign roles to them. The roles are as follows:

- **Team Agent**: A Team Agent is the individual who originally enrolled in the iOS Developer Program. A Team Agent has full access to the iOS Provisioning Portal, privileges not available to others. A Team Agent can invite others to be Team Admins or Team Members. A Team Agent can also approve certificate requests, register devices, create App IDs, create Push Notification Service SSL certificates, enable In App Purchase, retrieve the distribution certificate, and create development and distribution provisioning profiles.
- **Team Admin**: A Team Admin can invite new Team Admins and Team Members to joint the team. The Team Admin can also approve certificate requests, register devices, and create development provisioning profiles.
- **Team Member**: A Team Member can request and download a development certificate and download development provisioning profiles.

- **No Access**: No Access is a role given to developers to prevent them from accessing the iOS Provisioning Portal. This role is used when the company is enrolled in multiple Apple developer programs, that is, the iOS Developer Program and Mac Developer Program.

The Provisioning Process: A Brief Overview

Provisioning a new device is not difficult in and of itself, but there are a number of steps that must be performed first. You must be a paid member of the iOS Developer Program or be a team member of a company that is a paid member. You must request and install a development certificate on your development machine. You must register your device ID. You must create an App ID. You must create and install a development provisioning profile. All of this must be done before you can provision your iPad and run your application on it.

> **Note**
>
> It should go without saying, but you really should have an iPad if you plan to write iPad applications. You can learn iPad programming without an iPad, but you will be limited in what you can test. It's my opinion that having a physical device is a necessity for anyone serious about iPad programming.

Luckily you do not have to perform each of these steps often. Some of the steps, such as requesting and installing a development certificate, need to be done only once a year, and Xcode performs other steps for you as needed. Still, knowing about each step helps when a problem does come up, and yes, you will encounter a problem with the process at some point in your iOS career. It's just the nature of writing apps for mobile devices.

> **Note**
>
> A development certificate is valid for one year. You must request a new development certificate when the old one expires.

You've had a number of new terms thrown your way, such as device ID, App ID, and provisioning profile, but what exactly are these?

What Is a Device ID?

A device ID, also known as the Unique Device Identifier or UDID for short, is a 40-character string that uniquely identifies the device. The UDID is tied to a single device; no two devices will ever share the same device ID. The device ID is added to a provisioning profile. This restricts applications built with the provisioning profile to run on only the devices associated to the profile.

You register devices in the iOS Provisioning Portal. You can register up to 100 devices per year. The devices you register are for development and testing purposes only. You do not register the devices of your customers who download and install your application through the App Store.

Registering a device counts toward your yearly limit even if you remove the device. Say you register your iPad, then delete it from the list of registered devices. It still counts toward your annual limit of registered devices. If after deleting your iPad UDID from the list you decide to add it back, it will count again toward your yearly limit. Put another way, deleting a device does not reduce the number of registered devices for the year.

What Is an App ID?

The App ID is used during the development and provisioning processes to share Keychain data among a suite of applications, and it is used for document sharing, syncing, and configuration of iCloud. The App ID also allows the application to communicate with the Push Notification Service and external hardware accessories.

The App ID is the combination of a Bundle Seed ID and a Bundle Identifier. The Bundle Seed ID is a universally unique, ten-character, alphanumeric string. Its value is generated by Apple within the iOS Provisioning Portal. The Bundle Identifier is a string value you add to your application bundle. The Bundle Identifier is used by the operating system to identify your application for tasks such as applying application updates.

The common naming convention for the Bundle Identifier is the reverse domain name style. If your company name is Acme and your application name is Awesome App, you would define the Bundle Identifier as com.acme.awesomeApp.

The Bundle Identifier portion of the App ID can contain the wildcard character (*). The wildcard character, if used, must be the last character of the Bundle Identifier, for example, com.acme.*. Using the wildcard character allows the App ID, and its associated provisioning profiles, to be used with multiple applications.

> **Note**
>
> The Bundle Identifier portion of the App ID can be a wildcard only. In other words, the Bundle Identifier value can be a single asterisk character. As you will learn momentarily, the wildcard App ID created by Xcode's Organizer window is a single asterisk character.

A wildcard App ID is convenient because it can be used for multiple applications. Xcode's Organizer window creates a wildcard App ID that can be used by any application. The App ID looks like the last example in Table 6.1.

> **Note**
>
> When creating an App ID in the Provisioning Portal, you add a description of the purpose of the App ID. The App ID created by Organizer has the description "Xcode: Wildcard AppID."

Table 6.1 **App ID Examples**

App ID	Remarks
ABCDE12345.com.acme.awesomeApp	An explicit App ID for Acme's Awesome App
ABCDE12345.com.acme.*	A wildcard App ID for Acme applications
ABCDE12345.*	A wildcard App ID for all applications

You can also create an explicit App ID. An explicit App ID restricts the provisioning profile to a single application. Certain Apple services such as Push Notification and In App Purchase require an explicit App ID. If you plan to use an Apple service, you must use an explicit App ID.

Changing from a Wildcard to an Explicit App ID

You can change from a wildcard App ID to an explicit ID. You may find that you need to do this if, for example, you decide to add Game Center support to your application after it has been released. While you will be changing the App ID, you must not change the Bundle ID as defined in your application's *info.plist*. The Bundle ID defined in the app is used to identify your application for new releases, that is, updates to your application. Changing this breaks the app update process, and Apple will reject your app update submission if the Bundle ID changes.

Changing the Bundle ID suffix in the App ID is not the same as changing the Bundle ID in the application. This is why you can safely switch from a wildcard App ID to an explicit App ID. Changing the App ID's Bundle ID suffix does not change the application's real Bundle ID.

To change from a wildcard App ID, you must create a new App ID. You can use the same Bundle Seed ID or generate a new one. For the Bundle ID suffix, enter the bundle ID exactly as it is defined in your application's *info.plist*.

Because you now have a new App ID, you need to create a new provisioning profile that uses the new App ID.

Note

If your existing application uses the Keychain, select the same Bundle Seed ID for the new App ID. If you use a different Bundle Seed ID, your application will not be able to access any existing Keychain data.

What Is a Development Provisioning Profile?

A development provisioning profile ties developers, devices, and App IDs to a development team. A provisioning profile makes it possible for the developer to install and run the application on a device for the purpose of debugging and testing. To make this possible, the development provisioning profile must be installed on the device. A device, however, can contain more than one development provisioning profile.

> **Note**
>
> There is a second type of provisioning profile, the distribution provisioning profile. A distribution provisioning profile is used for Ad Hoc distribution—distribution to registered devices for the purpose of testing an application—and App Store distribution. More information on the distribution provisioning profile is included in Chapter 26, "Distributing Your App."

Now that you understand the pieces, let's get your development computer and device ready.

Do I Need a Dedicated Development Device?

A common question asked by new iOS developers is whether a dedicated development device is needed or not. The answer to this question depends entirely on you. Many iOS developers own only one device, but many other developers own multiple devices.

I have seven devices at the time of this writing: two iPads, three iPod touches (a first gen, a second gen, and a fourth generation), one iPhone 3GS, and one iPhone 4 CDMA. I use the older iPod touches to test my applications on older versions of iOS. I use the fourth-gen iPod touch to test on the latest version of iOS and to test retina display images. I usually keep one iPad up-to-date with the latest public release of iOS, and I use the other iPad, which is my primary iPad development device, to run beta versions of iOS. And I use my iPhone 3GS for testing on a phone, which can behave differently from an iPod touch, receiving incoming phone calls and SMS messages, for example.

I write a number of client apps, and my clients have different requirements, which is why I have multiple devices—and that number is growing every few months as I continue to buy new devices. It's doubtful that the average iOS developer needs this many devices. You can produce your application using only one device. However, there are advantages to having a dedicated development device.

I find I often need to reset my device to factory-install state, or I want to test my app on the latest beta release of iOS. You can still do this while owning only one device, but it does mean your device is constantly changing. That game you bought last night and got halfway through gets deleted when you reset your device, or your favorite app crashes on the new iOS beta release. This can be frustrating if your development device is your only device and you use the device for personal reasons.

Because of this, it is helpful—and it saves you time—to own devices dedicated to development purposes. Owning multiple devices, however, can get expensive, especially for the iPad. That's why I typically buy used devices for development. The Apple Refurbish Store is a good place to shop for used devices at an affordable price. Also, ask family and friends what they plan to do with their old devices if they are upgrading to newer hardware. This might be an opportunity to score a development device for free or close to free.

Setting Up Your Development Machine

The first thing you must do is set up your Mac development machine for code signing. Coding signing your application serves two purposes: It confirms the app author, and it guarantees that the app has not been altered since it was signed. iOS requires each application to be digitally signed before the app can run on a device. Code signing is never a joy, but it is a necessity, ensuring that the application comes from a trusted source.

To code sign your app, you must have a public and private key pair and a digital certificate. When your application is in development, you use a development certificate to code sign the application. This allows you to run and test your app on your own device. When you are ready to deploy your application to other devices, whether through the App Store or through Ad Hoc and enterprise distribution, you use a distribution certificate to code sign the app.

> **Note**
>
> Chapter 26, "Distributing Your App," covers distribution of your application. Therefore, I do not cover the distribution certificate in this chapter. Steps to request and install the distribution certificate are covered in Chapter 26.

Requesting a Development Certificate

To prepare your development Mac for code signing, you must first request a development certificate. To request a development certificate you need to generate a Certificate Signing Request (CSR). You use the Mac desktop application Keychain Access to generate the CSR. As Keychain Access creates your CSR, it also generates a public and private key pair for you and stores the pair in the login Keychain. The key pair identifies you as an iOS developer and is associated to the development certificate.

The Keychain Access application is available in the **Applications > Utilities** folder. Alternatively, you can launch the app using Spotlight. Press **⌘-Space** and start typing "Keychain," without the quotes, in the Spotlight box. Spotlight will find the Keychain Access application for you. All you need to do then is press the **Enter** key to launch the application.

The first thing you need to do in Keychain Access is select **Preferences** from the menu (or type **⌘-,**). Click **Certificates** and turn off Online Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) as shown in Figure 6.2.

Close the Preferences window, then select **Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority** from the menu bar. At this point, you are ready to enter the certificate information, as seen in Figure 6.3. Enter your email address in the User Email Address field. This must be the same email address you submitted when registering as an iOS Developer.

Enter your name in the Common Name field. The name must match the name submitted when you registered as an iOS Developer. Leave the CA Email Address field blank. Mark the options "Saved to disk" and "Let me specify key pair information."

Figure 6.2 Turn off the OCSP and CRL settings in the **Preferences > Certificates** screen.

Figure 6.3 Certificate Assistant window in Keychain Access

When you have finished, click the **Continue** button. The Assistant will ask where to save the CSR. Your desktop is as good a place as any, so save the CSR to your desktop.

Make sure you selected the "Let me specify key pair information" option in the Certificate Information window (Figure 6.3). This tells the Certificate Assistant to display the Key Pair Information window, shown in Figure 6.4. Here you set the options for generating the key pair. Select 2048 bits for the Key Size and RSA for the Algorithm. Click **Continue**.

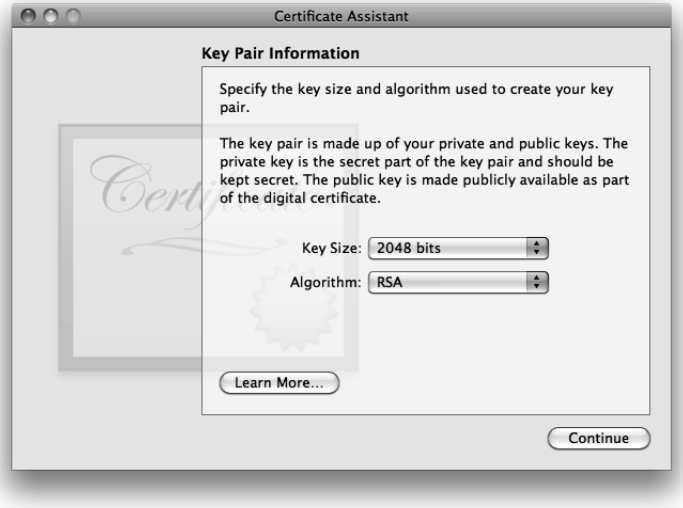


Figure 6.4 Key Pair Information window

> **Note**
>
> Your certificate request will be rejected if you do not specify the Key Size as 2048 bits and the Algorithm as RSA.

The Certificate Assistant will generate the CSR and save it to your desktop. A public and private key pair is also generated for you and stored in the login Keychain. The key pair can be viewed in the Keychain Access application under the Keys category as seen in Figure 6.5.

> **Note**
>
> If you are doing development from more than one Mac, you must copy the private key to each development machine. You will not be able to sign your application and test on your iPad without your private key. Use Keychain Access to export your private key and import it to your other development machines.
>
> For a step-by-step guide on exporting your private key, visit **developer.apple.com/ios/manage/certificates/team/howto.action**, scroll to the bottom of the Web page, and read the information under the "Saving your Private Key and Transferring to other Systems" section.

Click **Done** to close the Certificate Assistant. The generated CSR file is on your desktop. Your next step is to submit the CSR for approval.

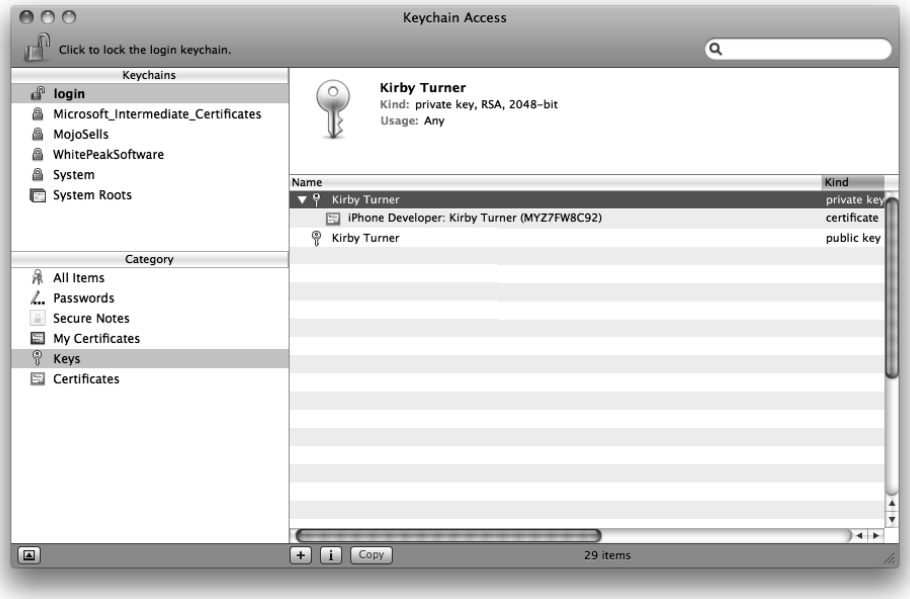


Figure 6.5 Public and private key pairs stored in the login Keychain

Getting More Help

The iOS Provisioning Portal requires you to perform a number of steps before you can test your application on your iPad and prepare your applications for distribution. A set of helpful resources is available in the Portal Resources.

The Portal Resources include a detailed user guide and how-to videos for requesting and installing your development certificate, assigning devices, creating App IDs, and creating provisioning profiles. If you need additional help with the iOS Provisioning Portal or you prefer seeing the steps performed, you should check out the Portal Resources.

The Portal Resources are available on the iOS Provisioning Portal home page under the section Portal Resources, found on the upper right side of the page, as seen in Figure 6.6.

Submit Your CSR for Approval

The next step is submitting your CSR for approval, and it is less involved than the previous step. A Team Agent or Admin approves or rejects your CSR. You will receive an email notifying you of your certificate status. If your request is approved, you download your digital certificate from the Provisioning Portal and install it on your development machine.

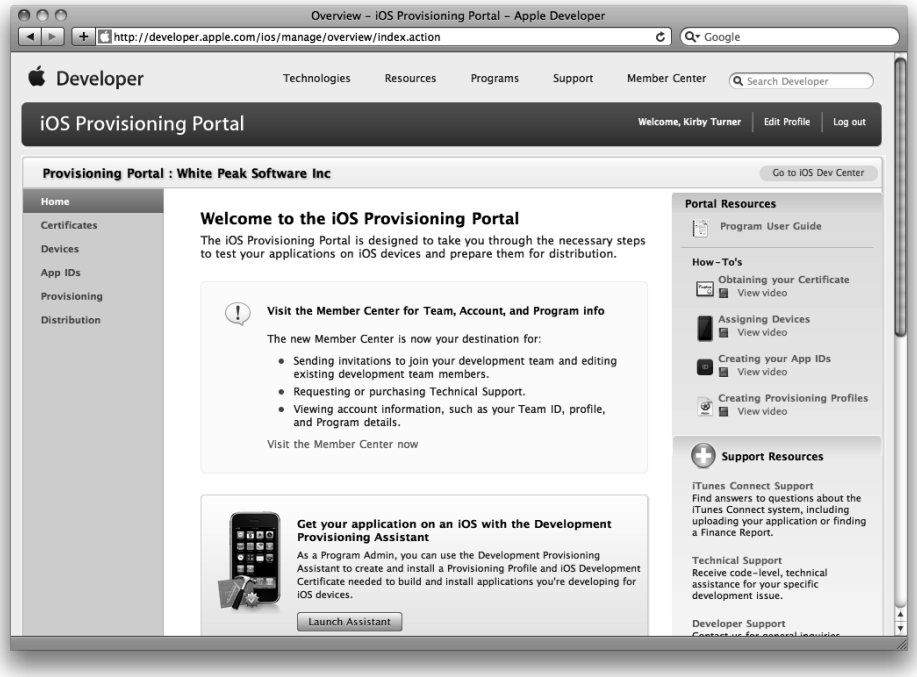


Figure 6.6 Portal Resources containing a user guide and how-to videos

> **Note**
>
> All certificate requests must be approved through the iOS Provisioning Portal. If you are the Team Agent or Team Admin, you still must approve your own certificate request.

To submit your CSR, sign in to the iOS Provisioning Portal. If you have trouble remembering the URL of the iOS Provisioning Portal, sign in to the iOS Dev Center (**developer.apple.com/ios**). Toward the upper right side of the iOS Dev Center home page there is a section titled iOS Developer Program, shown in Figure 6.7. This section includes links for the iOS Provisioning Portal, iTunes Connect, Apple Developer Forums, and the Developer Support Center. Click the **iOS Provisioning Portal** link to be transported to the portal Web site.

From the iOS Provisioning Portal home page, click the **Certificates** link found in the left-side menu bar. Next click the **Development** tab, then the **Add Certificate** button. Scroll down to find the **Choose file** button. Click the button and select the CSR file that you saved to the desktop. Click the **Submit** button to upload your CSR. If you are unable to submit your CSR through the Web site, email the CSR file to the Team Agent.

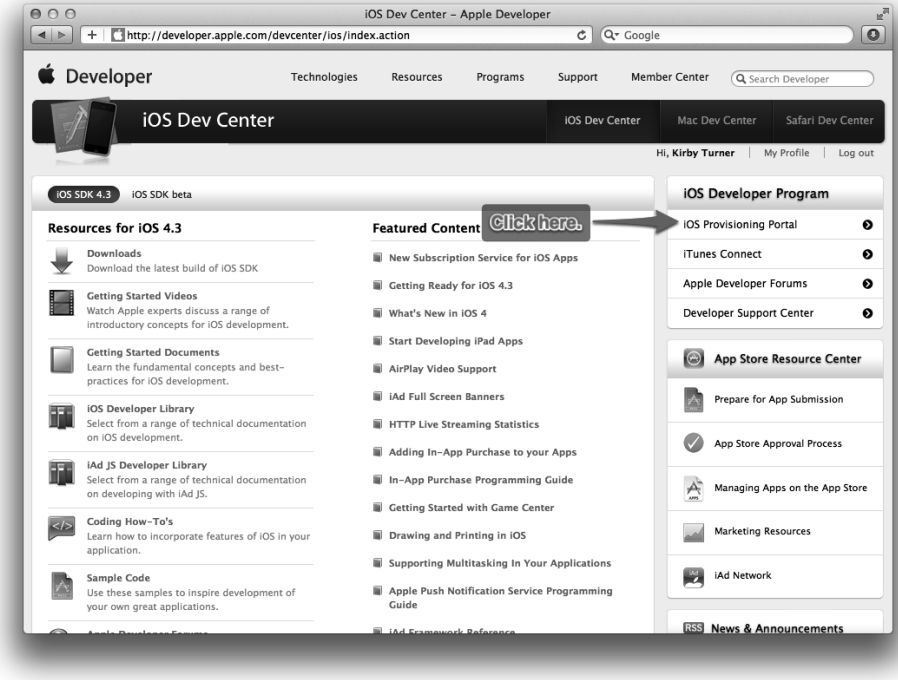


Figure 6.7 iOS Dev Center home page with a link to the iOS Provisioning Portal

Download and Install Your Certificate

The Team Admin will be notified by email after your submitted CSR has been received. Once the Admin approves or rejects your request, you will receive a notification email with your certificate status. When it has been approved, you sign in to the Provisioning Portal again, then click **Certificates > Development**. You'll see your approved certificate listed at the top. Click the **Download** button under the Action column to save the certificate to your development machine.

> **Note**
>
> If this is your first time setting up your development machine, you need to download and install the WWDR intermediate certificate. Click the download link and save the *AppleWWDRCA.cer* file to your development machine. Use Finder to navigate to the saved *AppleWWDRCA.cer* file, and double-click the file to launch Keychain Access; this installs the certificate on your machine.

On your development machine, use Finder to locate the saved *.cer* file. Double-click the *.cer* file to launch Keychain Access and to install your certificate. Save the

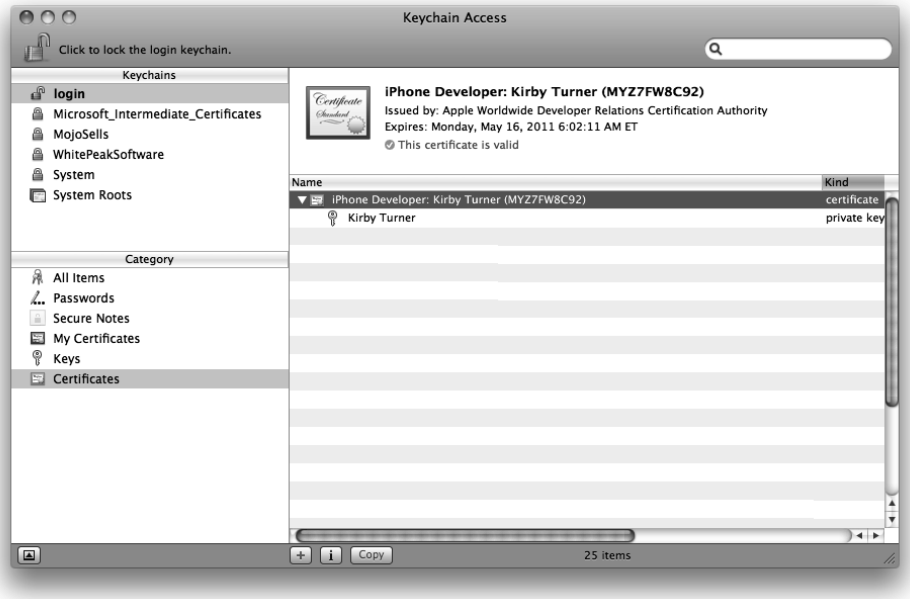


Figure 6.8 You can view your certificate in the Certificates category.

certificate to your login Keychain. Once it is installed, you can view the certificate by selecting the Certificates category for the login Keychain in Keychain Access (Figure 6.8). Your certificate name will be "iPhone Developer: Your Name."

While still in Keychain Access, click the Keys category for the login Keychain. Here you will see your public and private keys generated by the Certificate Assistant. Expand the private key by clicking the disclosure triangle. You will see that the certificate has been associated to your private key. Apple never receives your private key when you submit the CSR. Your private key is available only to you. This is why it is important that you not lose it.

> **Note**
>
> Make sure you have a backup of your key pair. If you do not have a backup and you lose the private key, you must go through the certificate request process all over again. I use Time Machine to make hourly backups of my primary development machine. I also use SuperDuper! to make complete system backups weekly. This provides a suitable backup of my public-private key pair. You should do something similar.

Your development machine is now set up to code sign builds of your application, but you cannot run your app on your iPad yet. You still have a few more steps to follow. Next up is setting up your device.

Setting Up Your Device

Now that your development machine is set up, it's time to set up your iPad for development. Here's what needs to happen:

1. You need to register your device ID.
2. You need to create an App ID.
3. You need to create a development provisioning profile.
4. You need to download and install the development provisioning profile.

These steps can be performed in one of two ways: by using the Xcode Organizer window or the iOS Provisioning Portal. Using Organizer is by far the easier way to set up your iPad for development. It performs the steps automatically for you. This approach is, however, not without its limits.

Organizer creates a wildcard App ID, and as you may recall, a wildcard App ID cannot be used if you plan to use Apple services like Game Center, In App Purchase, and Push Notification. That said, you should still let Xcode do its magic. While you may not use a wildcard App ID for your next awesome iPad app, it can be used to build and run sample apps and to test proofs of concept and prototype applications on your iPad.

Use for Development

The steps to set up your iPad for development are quite easy using Xcode. Tether your iPad to your development computer. Launch Xcode and open the Organizer window (**Windows > Organizer** or **Shift-⌘-2**). Organizer shows the list of registered and attached iOS devices. Attached devices have a status icon displayed to the right of the device name. A white status icon means the device is not ready for development. A green status icon means the device is ready for development. A yellow status icon means the device is busy.

Click the name of your iPad in the Devices list. You should see a screen similar to the one shown in Figure 6.9. Click the **Use for Development** button. Xcode will prompt you for your iOS Provisioning Portal credentials. Enter your Apple ID and password used for your iOS Developer account.

Xcode automatically sets up your device for development. It registers your device with the iOS Provisioning Portal, creates a wildcard App ID and development provisioning profile, if needed, and last downloads and installs the provisioning profile.

> **Note**
>
> Figure 6.9 shows a new iPod touch. The setup instructions are the same regardless of the device type. I chose to use an iPod touch to capture the screen shot because my development iPad was in use at the time.

The process can take a few minutes. While the process is running, the status icon is yellow. Do not disconnect your iPad from your computer while the process is

Figure 6.9 Organizer window with a new Apple device attached to the computer

running. Once the process is compete, the status icon changes to green and you will see a screen similar to the one in Figure 6.10.

You are now ready to build and run iOS applications on your iPad. To test that everything has been set up correctly, create a new project in Xcode. You can select any iOS application template; it does not matter which one. Make sure you select your iPad as the device for the active scheme (seen in Figure 6.11). Build and run (**⌘-R**) the project. Assuming your development machine and iPad are set up correctly, you will see the sample app running on your iPad.

> **Note**
>
> Sign in to the iOS Provisioning Portal to see the App ID and development provisioning profile created by Xcode. The App ID has the description "Xcode: Wildcard AppID," and the development provisioning profile has the name "Team Provisioning Profile: *." The status for the profile will also show "Managed by Xcode."

As you just learned, using Xcode is the easiest way to set up a new device. But what happens if there is a problem? And what if you plan to use Apple services like iCloud, Game Center, or In App Purchase, which require an explicit App ID? You need to use the iOS Provisioning Portal to manually perform the steps.

Figure 6.10 Organizer window with an attached Apple device ready for development

Figure 6.11 Use the scheme popup menu to set the run destination.

Using the iOS Provisioning Portal

As you already know, the iOS Provisioning Portal is used to request and download developer certificates, to register devices, to create App IDs, and to create and download provisioning profiles. You have already gone through the steps for requesting and downloading the developer certificate; the rest of this chapter will focus on the other areas of the Provisioning Portal.

Adding a Device ID

You can add a device ID individually, or you can upload a batch of device IDs in a *.deviceids* file generated by the iOS Configuration Utility. The iOS Configuration Utility is available to enterprise members only, so it is not covered here. Instead, the following walks you through adding an individual device ID.

Start by signing into the iOS Provisioning Portal Web site (**developer.apple.com/ios/manage/overview/index.action**), then click Devices in the left-side menu bar. Click the **Add Devices** button found on the right side of the Devices Web page. Enter a device name followed by the 40-character device ID, as seen in Figure 6.12. (I tend to be descriptive with the device name, adding the device owner's name and device type, for example, "Kirby Turner's iPad2.") Click the plus sign (**+**) button to

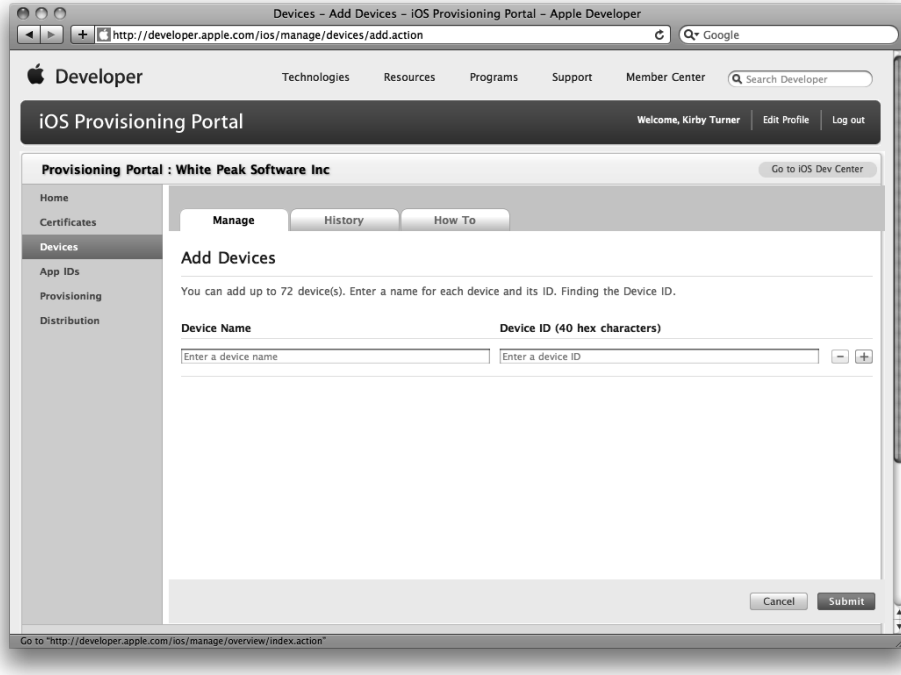


Figure 6.12 Add Devices page in the iOS Provisioning Portal

enter another device. When you have entered all the devices, click the **Submit** button. That's it. The device IDs are now registered.

> **Note**
>
> Only the Team Agent and Team Admin can register a device ID. A Team Member must send his device ID to the agent or admin to be registered.

You can edit the name of a registered device, but you cannot change the device ID. If a device ID is no longer valid—for example, you no longer own the device—you can remove the device from the list by marking the check box next to the device name, then clicking the **Remove Selected** button at the bottom of the registered device list.

How to Find the UDID for a Device

Finding the UDID for a device is not hard if you know where to look. There are different ways to find the ID of a device. The easiest for developers is to use Xcode's Organizer window. Open the Organizer window (**Shift-⌘-2**) and select the device. The device ID is displayed in the Identifier field as shown in Figure 6.13.

Another way to find the ID, albeit more obscure, is to use iTunes. Connect your device to your computer, then launch iTunes. In iTunes, select the device to see the device information screen, shown in Figure 6.14. Click the serial number. This will display the

Figure 6.13 Device information displayed in the Organizer window

Figure 6.14 Device information displayed in iTunes

UDID in place of the serial number. Once the UDID is showing, type ⌘-C to copy it to the clipboard. Click the field again to return to the serial number.

> **Note**
>
> You can also click the software version to see the build number.

Another common way to retrieve the UDID is to use one of the many free UDID apps available in the App Store. These apps not only retrieve the UDID from the device but also provide options to copy the ID to the clipboard and to email the device ID to the recipient of your choosing. Using a UDID app is the easiest way for nondevelopers to send you their device IDs.

To download a UDID app, go to the App Store and search on "UDID." Select the app that appeals to you the most.

Adding an App ID

Adding an App ID is almost as easy as registering a device ID. Once again, sign in to the iOS Provisioning Portal. Click the App IDs menu item in the left-side menu bar, then click the **New App ID** button found on the right side of the page, shown in Figure 6.15. This takes you to the Create App ID page.

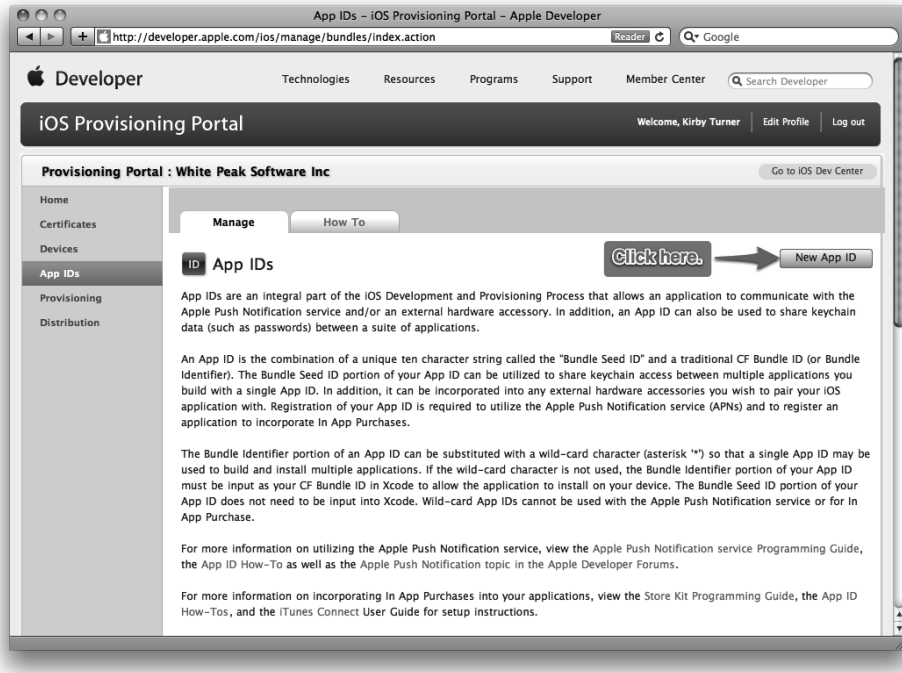


Figure 6.15 Click the **New App** button found on the App IDs page in the iOS Provisioning Portal.

Note

The Team Agent is the only team member who can add a new App ID.

On the Create App ID page, seen in Figure 6.16, enter the description for the App ID. Use a name or description that makes sense to you. For instance, you might use the name of your application when creating an explicit App ID. The description you enter is used throughout the portal, so enter a name or description that will help you identify the App ID.

Next, set the Bundle Seed ID, aka the App ID prefix. You can have the portal generate a new seed for you, or you can select from a seed that you previously generated. If you plan to share Keychain data access among multiple applications, you need to use the same Bundle Seed ID for each application's App ID.

Last, enter the Bundle Identifier, aka the App ID suffix. Remember to use the reverse domain name naming convention. Include an asterisk (*) as the last character if you wish to create a wildcard App ID.

Click the **Submit** button to save the App ID. Clicking the button returns you to the App IDs page. Scroll down the page to see the newly created App ID in the list of App IDs.

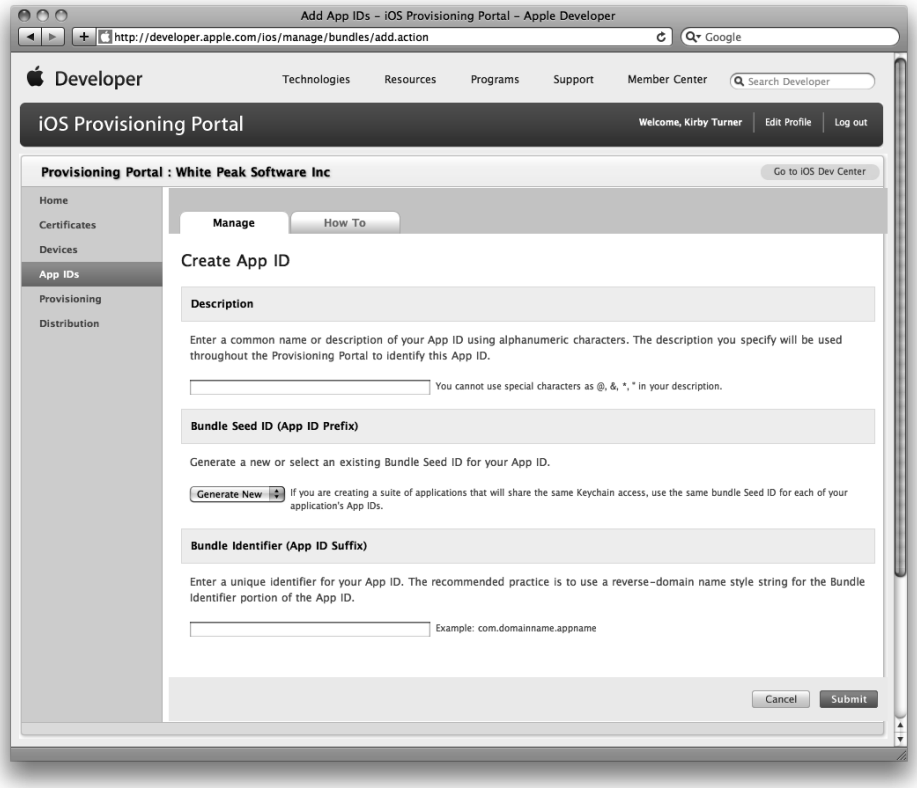


Figure 6.16 The Create App ID page

In the list of App IDs, you can choose between two actions: Details and Configure. The Details action is available only for wildcard App IDs. Click the action to see the details of the wildcard App ID.

The Configure action is available only for explicit App IDs. Click the action to configure the App ID for Push Notification. You do not need to configure the explicit App ID for In App Purchase and Game Center services as these services are enabled by default for explicit App IDs.

> **Note**
>
> You cannot delete an App ID once it has been created.

Creating a Development Provisioning Profile

The development provisioning profile brings you (the developer), your device, and your App ID together so that you can run and test your application on your iPad. To create a new development provisioning profile, log in to the iOS Provisioning Portal

and click the Provisioning link found in the left-side menu bar. Click the **Development** tab at the top of the Provisioning screen, then click the **New Profile** button. This will take you to the Create iOS Development Provisioning Profile page, seen in Figure 6.17.

> **Note**
>
> Only Team Agent and Team Admin members can create development provisioning profiles.

Enter a name for the provisioning profile. (I like to use descriptive names, for example, "Hey Peanut Dev Profile.")

Select the development certificates that will be associated to the provisioning profile. This identifies the development certificates used to code sign the application for development. You should select the developer certificate for each team member who will be using the development provisioning profile.

Select the App ID for the profile. Each profile can have only one App ID. You can use a wildcard App ID if you wish to use the same development provisioning profile for more than one application.

Finally, select the devices that the developers can use for running and testing the application on a device. Then click the **Submit** button to generate the development provisioning profile.

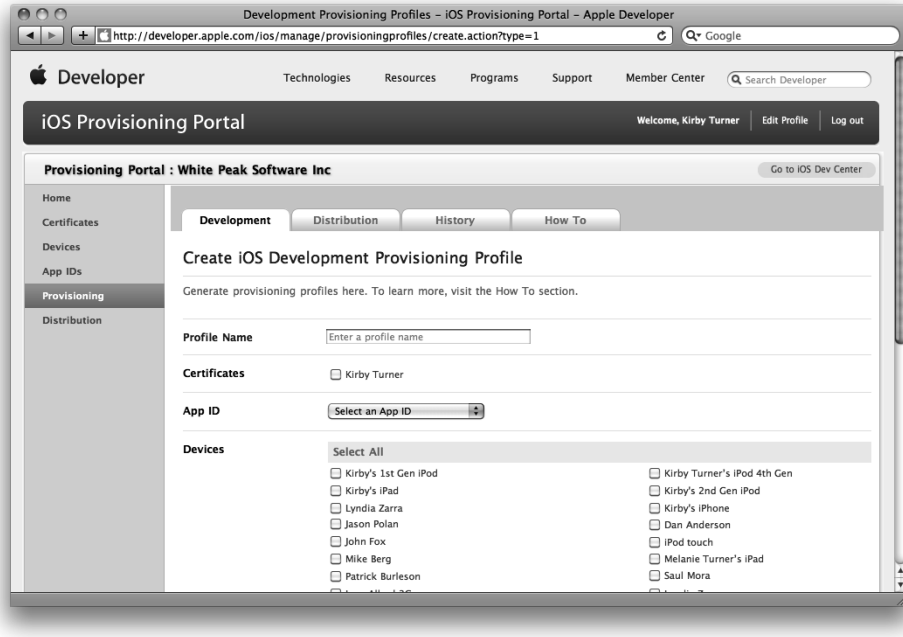


Figure 6.17 Create iOS Development Provisioning Profile page

Downloading a Development Provisioning Profile

Once a development provisioning profile has been generated, team members can download and install the profile. To download the development provisioning profile, log in to the iOS Provisioning Profile, click the **Provisioning** menu item, then click the **Development** tab on the Provisioning page. You will see a list of development provisioning profiles, as shown in Figure 6.18. Click the **Download** button for the profile you wish to install.

Now you are ready to install the provisioning profile.

Installing a Development Provisioning Profile

There are a few ways to install a development provisioning profile. You can copy the profile file to *~/Library/MobileDevice/Provisioning Profile*, but the most common way is to use the Organizer window in Xcode. This will ensure that the profile is stored in the proper directory, and it will create the directory if it does not already exist.

> **Note**
>
> The development provisioning profile you download has the file extension *.mobileprovision*.

To install using Organizer, launch Xcode and open the Organizer window (**Shift-⌘-2**). Select a device on which to install the provisioning profile, then click the

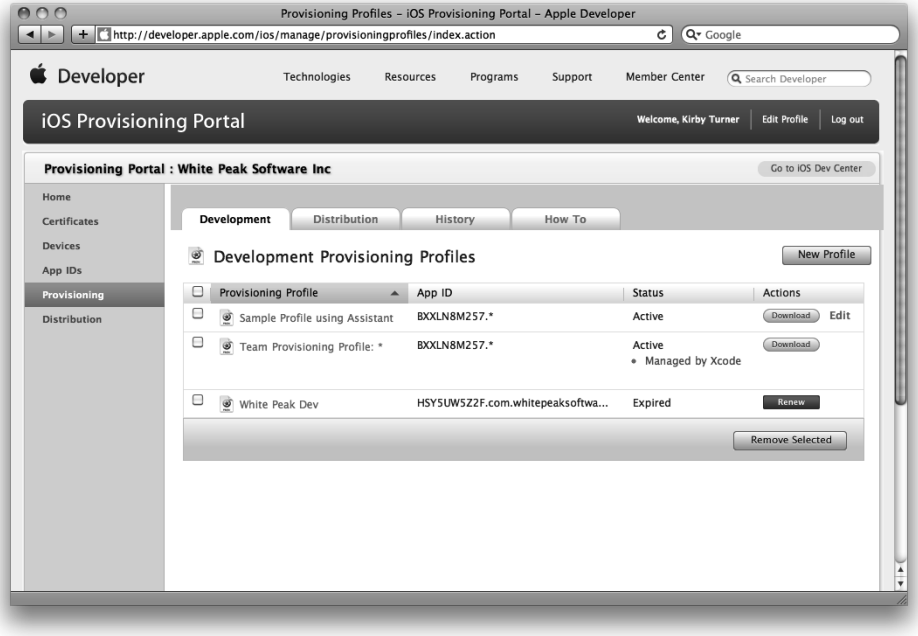


Figure 6.18 List of development provisioning profiles

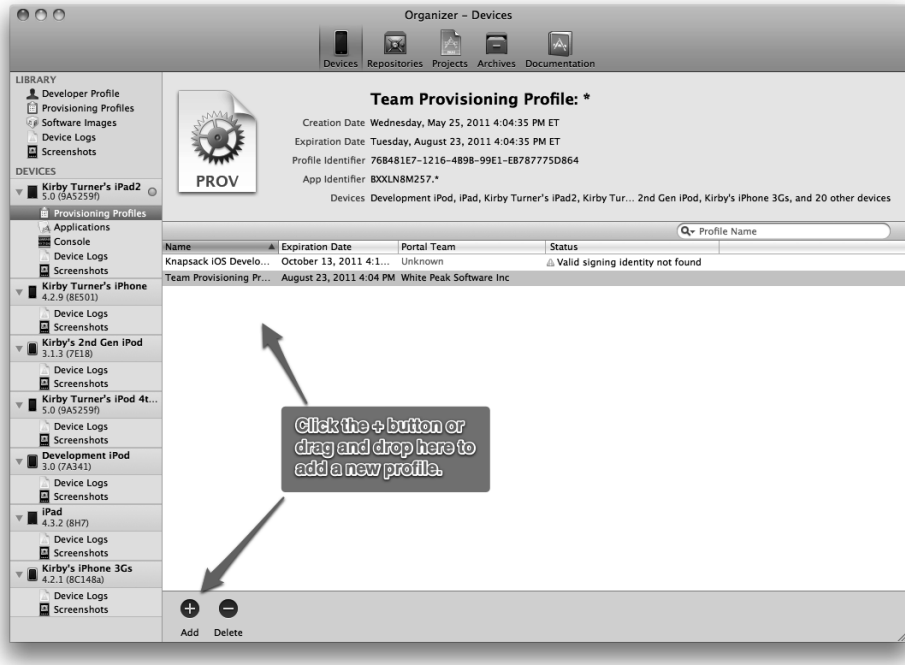


Figure 6.19 Click the + button or drag and drop the *.mobileprovision* file to add the development provisioning profile.

+ button to select the *.mobileprovision* file you downloaded from the iOS Provisioning Portal. Alternatively, you can drag and drop the *.mobileprovision* file onto the Provisioning list in the Organizer window; see Figure 6.19.

Another way to install a development provisioning profile is to drag and drop the file onto the iTunes icon in the dock. Note, however, that this will fail if the provisioning profile directory does not exist.

You can also copy the *.mobileprovision* file directly to the *~/Library/MobileDevice/Provisioning Profiles* directory. To make it easier, add a shortcut for the directory to Finder. When you need to install a new provisioning profile, simply drag and drop the downloaded file onto this shortcut; see Figure 6.20.

A third option is to drag and drop the provisioning file onto the Xcode icon in the Dock. This will copy the provisioning profile to the appropriate directory and to your iPad device if connected.

That's it for installing a development provisioning profile.

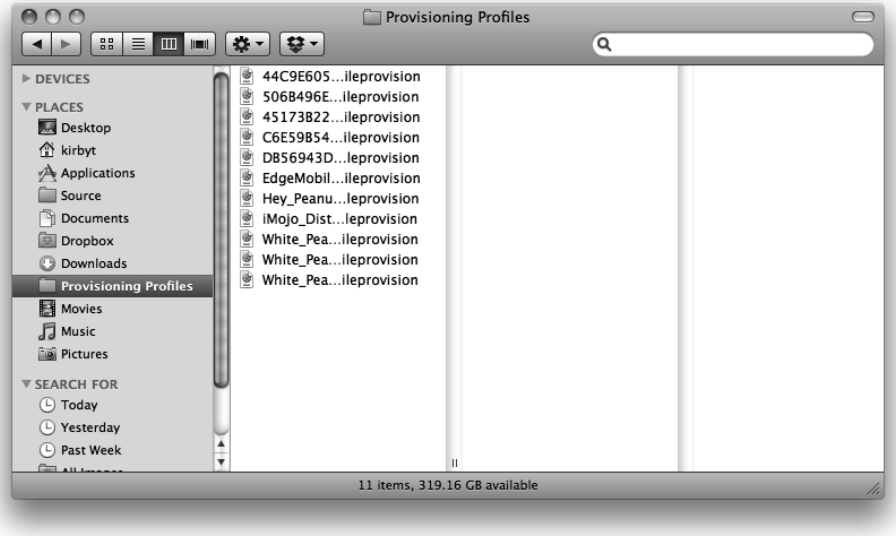


Figure 6.20 Finder window with a shortcut folder to the provisioning profile directory

Summary

At this point, your development machine should be set up for code signing iOS applications, and your iPad should be ready for development. You're now ready to write iPad applications that you can run on your own device. But first, let's talk a bit about app design as it relates to the iPad.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z