Table of Contents (Summary)

	Intro	xxi
1	Getting Started: *Going mobile*	1
2	iPhone App Patterns: *Hello @twitter*	37
3	Objective-C for the iPhone: *Twitter needs variety*	89
4	Multiple Views: *A table with a view*	131
5	plists and Modal Views: *Refining your app*	185
6	Saving, Editing, and Sorting Data: *Everyone's an editor...*	239
7	Tab Bars and Core Data: *Enterprise apps*	303
8	Migrating and Optimizing with Core Data: *Things are changing*	377
9	Camera, Map Kit, and Core Location: *Proof in the real world*	431
i	Leftovers: *The top 6 things (we didn't cover)*	487
ii	Preparing Your App for Distribution: *Get ready for the App Store*	503

Table of Contents (the real thing)

Intro

Your brain on iPhone Development. Here *you* are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing enough to develop your own iPhone apps?

Who is this book for?	xxii
We know what you're thinking	xxiii
Metacognition: thinking about thinking	xxv
Here's what YOU can do to bend your brain into submission	xxvii
Read me	xxviii
The technical review team	xxx
Acknowledgments	xxxi

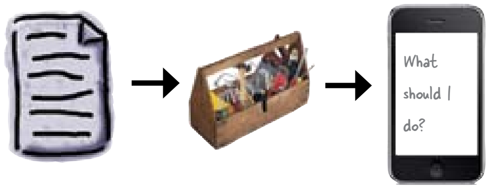
www.itbook.store/books/9780596803544

getting started

1 Going mobile

The iPhone changed everything.

It's a **gaming** platform, a personal **organizer**, a full **web browser**, oh yeah, and a **phone**. The iPhone is one of the most exciting devices to come out in some time, and with the opening of the App Store, it's an opportunity for independent developers to compete worldwide with big named software companies. All you need to release your own app are a couple of **software tools**, some **knowledge**, and **enthusiasm**. Apple provides the software and we'll help you the knowledge; we're sure you've got the enthusiasm covered.



There's a lot of buzz and a lot of money tied up in the App Store... 2
Mobile applications aren't just ported desktop apps 3
Anatomy of an iPhone app 5
Mike can't make a decision 6
Make a good first impression 7
It all starts with the iPhone SDK 8
Xcode includes app templates to help you get started 10
Xcode is the hub of your iPhone project... 12
...and plays a role in every part of writing your app 13
Build your interface using... Interface Builder 14
Add the button to your view 16
The iPhone Simulator lets you test your app on your Mac 17
What happened? 22
Use Interface Builder to connect UI controls to code 23
Interface Builder lists which events a component can trigger 24
Elements dispatch events when things happen to them 24
Connect your events to methods 25
Your iPhone Toolbox 35

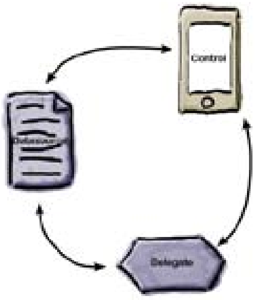
www.itbook.store/books/9780596803544

2 iPhone app patterns

Hello @twitter!

Apps have a lot of moving parts.

OK, actually, they don't have any real moving parts, but they do have lots of **UI controls**. A typical iPhone app has more going on than just a button, and now it's time to build one. Working with some of the **more complicated widgets** means you'll need to pay more attention than ever to how you **design** your app as well. In this chapter, you'll learn how to put together a bigger application and some of the **fundamental design patterns** used in the iPhone SDK.



First we need to figure out what Mike (really) wants	39
App design rules—the iPhone HIG	44
HIG guidelines for pickers and buttons	47
Create a new View-based project for Insta-Twit	48
The life of a root view	52
First, get the data from Mike	55
Use pickers when you want controlled input	56
Fill the picker rows with Mike's data	57
Pickers get their data from a datasource...	58
There's a pattern for that	59
First, declare that the controller conforms to both protocols	64
The datasource protocol has two required methods	66
Connect the datasource just like actions and outlets	67
There's just one method for the delegate protocol	68
The button needs to be connected to an event	72
Connect the picker to our outlet	78
Use our picker reference to pull the selected values	79
Your iPhone Toolbox	87

www.itbook.store/books/9780596803544

3 objective-c for the iPhone

Twitter needs variety

We did a lot in Chapter 2, but what language was that?

Parts of the code you've been writing might look familiar, but it's time you got a sense of what's really going on under the hood. The **iPhone SDK** comes with great tools that mean that you don't need to write code for everything, but you can't write entire apps without learning something about the underlying language, including **properties**, **message passing**, and **memory management**. Unless you work that out, all your apps will be just default widgets! And you want more than just widgets, right?

Renee is catching on.... 90

Make room for custom input 91

Header files describe the interface to your class 93

Auto-generated accessors also handle memory management 99

To keep your memory straight, you need to remember just two things 101

But when Mike's finished typing... 111

Customize your UITextField 113

Components that use the keyboard ask it to appear... 114

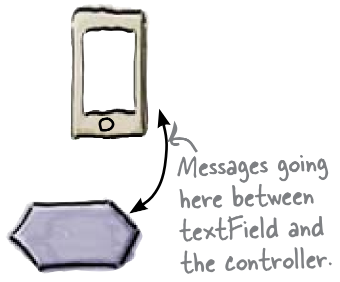
Ask the textField to give up focus 115

Messages in Objective-C use named arguments 117

Use message passing to tell our view controller when the Done button is pressed 118

Something's still not right 122

Your Objective-C Toolbox 129



www.itbook.store/books/9780596803544

4 multiple views

A table with a view

Most iPhone apps have more than one view.

We've written a cool app with one view, but anyone who's used an iPhone knows that most apps aren't like that. Some of the more impressive iPhone apps out there do a great job of moving through complex information by using multiple views. We're going to start with navigation controllers and table views, like the kind you see in your Mail and Contact apps. Only we're going to do it with a twist...

Look, I don't have time for posting to Twitter. I need to know a ton of drink recipes every night. Is there an app for that?

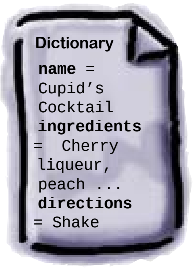
So, how do these views fit together?	135
The navigation template pulls multiple views together	136
The navigation template starts with a table view	137
A table is a collection of cells	140
Just a few more drinks	148
Plists are an easy way to save and load data	150
Arrays (and more) have built-in support for plists	153
Use a detail view to drill down into data	156
A closer look at the detail view	157
Use the navigation controller to switch between views	167
Navigation controllers maintain a stack of views	168
Dictionaries store information as key-value pairs	172
Debugging—the dark side of iPhone development	175
First stop on your debugging adventure: the console	176
Interact with your application while it's running	177
Xcode supports you after your app breaks, too	178
The Xcode debugger shows you the state of your application	179
What the heck is going on?	181
Your iPhone Toolbox	183

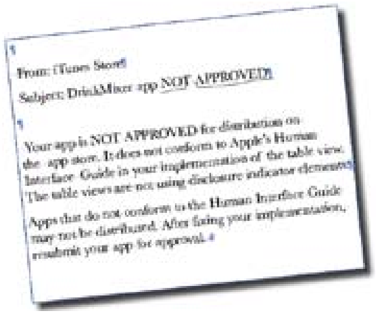
www.itbook.store/books/9780596803544

5 plists and modal views: Refining your app

So you have this almost-working app...

That's the story of every app! You get some functionality working, decide to add something else, need to do some **refactoring**, and respond to some **feedback** from the App Store. Developing an app isn't ~~always~~ ever a linear process, but there's a lot to be learned in that process.


Dictionary
name = Cupid's Cocktail
ingredients = Cherry liqueur, peach ...
directions = Shake


From: iTunes Store
Subject: DrinkMixer app NOT APPROVED

Your app is NOT APPROVED for distribution on the app store. It does not conform to Apple's Human Interface Guide in your implementation of the table view. The table views are not using disclosure indicator elements.

Apps that do not conform to the Human Interface Guide may not be distributed. After fixing your implementation, resubmit your app for approval.

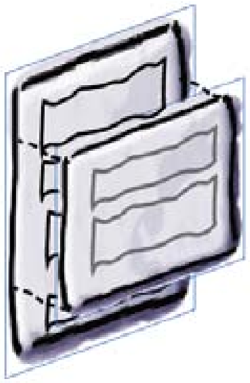
It all started with Sam...	186
Use the debugger to investigate the crash	188
Update your code to handle a plist of dictionaries	191
The detail view needs data	194
Each dictionary has all the information we need	195
We have a usability problem	201
Use a disclosure indicator if your cell leads to more information	203
Sales were going strong...	206
Use navigation controller buttons for editing	211
The button should create a new view	215
We need a view... but not necessarily a new view	216
The view controller defines the behavior for the view	217
A nib file contains the UI components and connections...	218
You can subclass and extend views like any other class	219
Modal views focus the user on the task at hand...	224
Any view can present a modal view	225
Our view doesn't have a navigation bar	230
Create the save and cancel buttons	232
Write the save and cancel actions	233
Your iPhone Toolbox	237

www.itbook.store/books/9780596803544

6 saving, editing, and sorting data

Everyone's an editor...

Displaying data is nice, but adding and editing information is what makes an iPhone app really rock. DrinkMixer is great—it uses some **cell customization**, and works with **plist dictionaries** to display data. It's a handy reference application, and you've got a good start on adding new drinks. Now, it's time to give the user the ability to modify the data—**saving, editing, and sorting**—to make it more useful for everyone. In this chapter we'll take a look at **editing patterns** in iPhone apps and how to guide users with the nav controller.



Red-Headed School Girl
Canadian Whiskey
Cream Soda
Add the whiskey, then the cream soda to a shot glass and drink

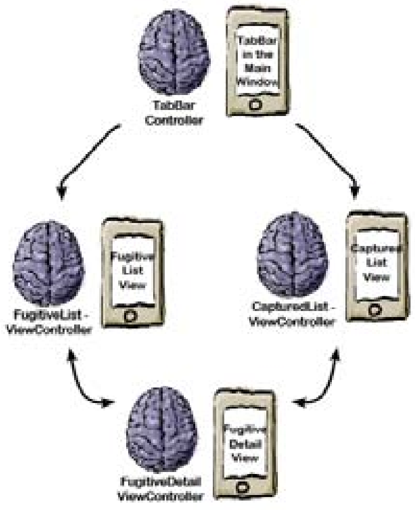
Sam is ready to add a Red-Headed School Girl... 240
...but the keyboard is in the way 241
We need to wrap our content in a scroll view 243
The scroll view is the same size as the screen 245
The keyboard changes the visible area 248
iPhone notifies you about the keyboard 250
Register with the default notification center for events 251
Keyboard events tell you the keyboard state and size 257
The table view doesn't know its data has changed 276
You need to ask the table view to reload its data 276
The array is out of order, too 280
Table views have built-in support for editing and deleting 288
Your iPhone Development Toolbox 301

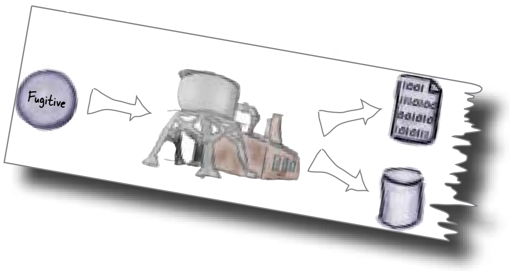
www.itbook.store/books/9780596803544

7 tab bars and core data

Enterprise apps

Enterprise apps mean managing more data in different ways. Companies large and small are a significant market for iPhone apps. A small handheld device with a **custom app** can be huge for companies that have **staff on the go**. Most of these apps are going to manage **lots of data**, and iPhone 3.x has built in Core Data support. Working with that and another new controller, the **tab bar controller**, we're going to build an app for justice!


TabBar in the Main Window
TabBar Controller
Fugitive List View
FugitiveList-ViewController
Captured List View
CapturedList-ViewController
Fugitive Detail View
FugitiveDetail ViewController



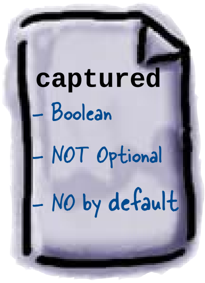
HF bounty hunting	304
Choose a template to start iBountyHunter	308
Drawing how iBountyHunter works...	310
Build the fugitive list view	316
Next up: the captured view	318
After a quick meeting with Bob...	327
Core Data lets you focus on your app	329
Core Data needs to know what to load	330
Core Data describes entities with a Managed Object Model	333
Build your Fugitive entity	334
Whip up a Fugitive class without writing a line	341
Use an NSFetchRequest to describe your search	344
Add the database as a resource	354
The template sets things up for a SQLite DB	355
The iPhone's application structure defines where you can read and write	358
Copy the database to the correct place	359
To be continued...	373
Your Core Data Toolbox	375

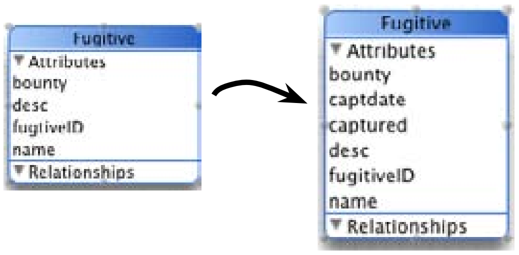
www.itbook.store/books/9780596803544

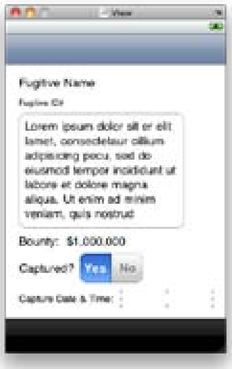
8 migrating and optimizing with core data

Things are changing

We have a great app in the works. iBountyHunter successfully loads the data that Bob needs and lets him view the fugitives in an easy way. But what about when **the data has to change**? Bob wants some **new functionality**, and what does that do to the **data model**? In this chapter you'll learn how to handle **changes** to your data model and how to take advantage of more Core Data features.






View
Fugitive Name
Fugitive ID
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
Bounty: $1,000,000
Captured? Yes No
Capture Date & Time:

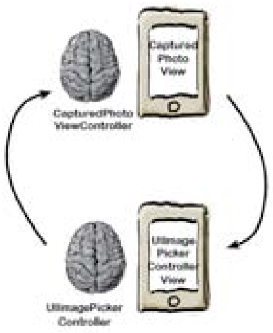
Bob needs documentation	378
Everything stems from our object model	381
The data hasn't been updated	384
Data migration is a common problem	385
We need to migrate the old data into the new model	386
Xcode makes it easy to version the data model	387
Core Data can "lightly" migrate data	389
Bob has some design input	394
A quick demo with Bob	406
Use predicates for filtering data	408
We need to set a predicate on our NSFetchRequest	409
Core Data controller classes provide efficient results handling	416
Time for some high-efficiency streamlining	417
Next we need to change the search to use the controller...	417
Refactor viewWillAppear to use the controller	418
We need to refresh the data	423
Your Data Toolbox	429

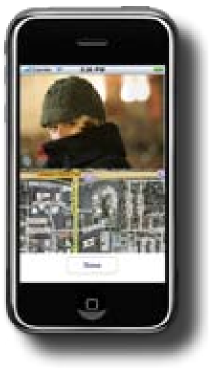
www.itbook.store/books/9780596803544

9 camera, map kit, and core location

Proof in the real world

The iPhone knows where it is and what it sees. As any iPhone user knows, the iPhone goes way beyond just **managing data**: it can also take **pictures**, figure out your **location**, and put that information together for use in your app. The beauty about incorporating these features is that just by tapping into the tools that iPhone gives you, suddenly you can import pictures, locations, and **maps** without much coding at all.





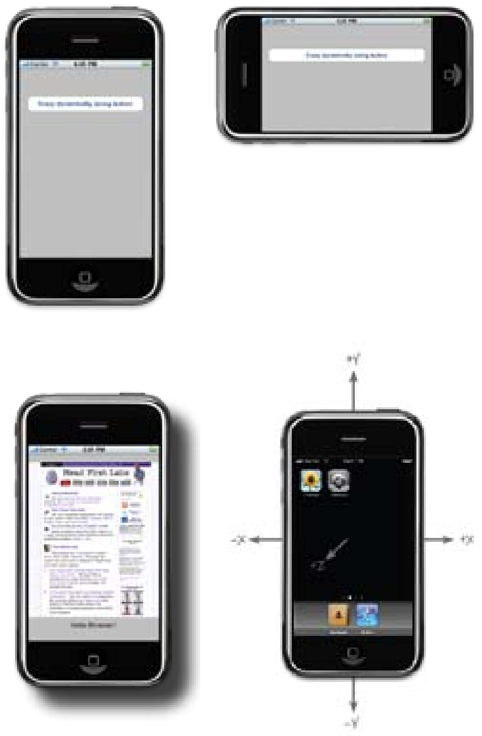
For Bob, payment requires proof!	432
The way to the camera...	441
There's a method for checking	451
Prompt the user with action sheets	452
Bob needs the where, in addition to the when	458
Core Location can find you in a few ways	464
Add a new framework	466
Just latitude and longitude won't work for Bob	472
Map Kit is new with iPhone 3.0	473
A little custom setup for the map	474
Annotations require a little more finesse	479
Your extras Toolbox	485
It's been great having you here!	486

www.itbook.store/books/9780596803544

appendix i, leftovers

The top 6 things (we didn't cover)

Ever feel like something's missing? We know what you mean... Just when you thought you were done, there's more. We couldn't leave you without a few extra details, things we just couldn't fit into the rest of the book. At least, not if you want to be able to carry this book around without a metallic case and castor wheels on the bottom. So take a peek and see what you (still) might be missing out on.



#1. Internationalization and Localization	488
Localizing string resources	490
#2. UIWebView	492
#3. Device orientation and view rotation	494
Handling view rotations	495
Handling rotation with two different views	496
#4. View animations	497
#5. Accelerometer	498
Understanding the device acceleration	499
#6. A word or two about gaming...	500
Quartz and OpenGL	501

www.itbook.store/books/9780596803544

appendix ii, preparing your app for distribution

Get ready for the App Store

You want to get your app in the App Store, right? So far, we've basically worked with apps in the simulator, which is fine. But to get things to the next level, you'll need to **install an app** on an actual iPhone or iPod Touch before applying to get it in the App Store. And the only way to do that is to **register** with Apple as a developer. Even then, it's not just a matter of clicking a button in Xcode to get an app you wrote on your personal device. To do that, it's time to **talk with Apple**.

Apple has rules	504
The Provisioning Profile pulls it all together	505
Keep track in the Organizer	506

www.itbook.store/books/9780596803544