

Advanced iOS 4 Programming: Developing Mobile Applications for Apple iPhone, iPad, and iPod touch

Maher Ali, PhD

Bell Labs, Alcatel-Lucent



A John Wiley and Sons, Ltd., Publication

This edition first published 2010
© 2010 Maher Ali

Registered office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

Editorial office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com/wiley-blackwell.

The right of the author to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Trademarks: Wiley and the Wiley Publishing logo are trademarks or registered trademarks of John Wiley and Sons, Inc. and/or its affiliates in the United States and/or other countries, and may not be used without written permission. iPhone, iPad and iPod are trademarks of Apple Computer, Inc. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in the book. This book is not endorsed by Apple Computer, Inc.

ISBN 978-0-470-97123-9 (paperback),

ISBN 978-0-470-97144-4 (ebk),

ISBN 978-0-470-97165-9 (ebk),

ISBN 978-0-470-97954-9 (ebk)

A catalogue record for this book is available from the British Library.

Typeset in 10/12 Times by Laserwords Private Limited, Chennai, India

Printed in the United States of America

Contents

1	Getting Started	1
1.1	iOS SDK and IDE Basics	1
1.1.1	Obtaining and installing the SDK	1
1.1.2	Creating a project	1
1.1.3	Familiarizing yourself with the IDE	3
1.1.4	Looking closely at the generated code	4
1.2	Creating Interfaces	6
1.2.1	Interface Builder	6
1.2.2	Revising the application	9
1.3	Using the Debugger	14
1.4	Getting More Information	16
1.5	Summary	17
	Exercises	18
2	Objective-C and Cocoa	19
2.1	Classes	19
2.1.1	Class declaration	20
2.1.2	How do I use other declarations?	21
2.1.3	Class definition	21
2.1.4	Method invocation and definition	22
2.1.5	Important types	23
2.1.6	Important Cocoa classes	23
2.2	Memory Management	24
2.2.1	Creating and deallocating objects	24
2.2.2	Preventing memory leaks	25
2.3	Protocols	27
2.4	Properties	29
2.4.1	Property declaration	29
2.4.2	Circular references	33

2.5	Categories	36
2.6	Posing	37
2.7	Exceptions and Errors	38
	2.7.1 Exceptions	38
	2.7.2 Errors	42
2.8	Key-Value Coding (KVC)	44
	2.8.1 An example illustrating KVC	45
	2.8.2 KVC in action	46
2.9	Multithreading	51
2.10	Notifications	54
2.11	Blocks	56
	2.11.1 Declaration and definition	56
	2.11.2 Block literal	57
	2.11.3 Invocation	57
	2.11.4 Variable binding	58
2.12	Grand Central Dispatch (GCD)	59
	2.12.1 Queues	59
	2.12.2 Scheduling a task	60
	2.12.3 Putting it together	60
2.13	The Objective-C Runtime	61
	2.13.1 Required header files	62
	2.13.2 The NSObject class	62
	2.13.3 Objective-C methods	63
	2.13.4 Examples	66
2.14	Summary	82
	Exercises	83
3	Collections	87
3.1	Arrays	87
	3.1.1 Immutable copy	89
	3.1.2 Mutable copy	91
	3.1.3 Deep copy	93
	3.1.4 Sorting an array	96
3.2	Sets	101
	3.2.1 Immutable sets	101
	3.2.2 Mutable sets	102
	3.2.3 Additional important methods for NSMutableSet	104
3.3	Dictionaries	104
3.4	Summary	106
	Exercises	107
4	Anatomy of an iPhone Application	109
4.1	Hello World Application	109
	4.1.1 Create a main.m file	109

4.1.2	Create the application delegate class	110
4.1.3	Create the user interface subclasses	111
4.2	Building the Hello World Application	112
4.3	Summary	116
	Exercises	117
5	The View	119
5.1	View Geometry	119
5.1.1	Useful geometric type definitions	119
5.1.2	The UIScreen class	120
5.1.3	The frame and center properties	121
5.1.4	The bounds property	123
5.2	The View Hierarchy	124
5.3	The Multitouch Interface	125
5.3.1	The UITouch class	126
5.3.2	The UIEvent class	126
5.3.3	The UIResponder class	127
5.3.4	Handling a swipe	131
5.3.5	More advanced gesture recognition	136
5.4	Animation	140
5.4.1	Using the UIView class animation support	140
5.4.2	Sliding view	145
5.4.3	Flip animation	145
5.4.4	Transition animation	146
5.5	Drawing	148
5.5.1	Fundamentals	148
5.5.2	The Summary View application	150
5.6	Summary	159
	Exercises	159
6	Controls	161
6.1	The Foundation of All Controls	161
6.1.1	UIControl attributes	161
6.1.2	Target-action mechanism	162
6.2	The Text Field	165
6.2.1	Interacting with the keyboard	167
6.2.2	The delegate	170
6.2.3	Creating and working with a UITextField	171
6.3	Sliders	172
6.4	Switches	173
6.5	Buttons	174
6.6	Segmented Controls	175
6.7	Page Controls	178
6.8	Date Pickers	179

6.9	Summary	181
	Exercises	181
7	View Controllers	183
7.1	The Simplest View Controller	183
	7.1.1 The view controller	183
	7.1.2 The view	185
	7.1.3 The application delegate	186
	7.1.4 A simple MVC application	187
7.2	Tab-Bar Controllers	189
	7.2.1 A detailed example of a tab-bar application	189
	7.2.2 Some comments on tab-bar controllers	194
7.3	Navigation Controllers	198
	7.3.1 A detailed example of a navigation controller	199
	7.3.2 Customization	204
7.4	Modal View Controllers	208
7.5	Summary	214
	Exercises	214
8	Special-Purpose Views	215
8.1	Picker View	215
	8.1.1 The delegate	216
	8.1.2 An example of picker view	217
8.2	Progress View	221
8.3	Scroll View	225
8.4	Text View	227
	8.4.1 The delegate	228
	8.4.2 An example of text view	228
8.5	Alert View	231
8.6	Action Sheet	233
8.7	Web View	235
	8.7.1 A simple web view application	235
	8.7.2 Viewing local files	240
	8.7.3 Evaluating JavaScript	244
	8.7.4 The web view delegate	251
8.8	Summary	256
	Exercises	257
9	Table View	259
9.1	Overview	259
9.2	The Simplest Table View Application	260
9.3	A Table View with Both Images and Text	265
9.4	A Table View with Section Headers and Footers	267
9.5	A Table View with the Ability to Delete Rows	269

9.6	A Table View with the Ability to Insert Rows	275
9.7	Reordering Table Rows	280
9.8	Presenting Hierarchical Information	285
9.8.1	Detailed example	288
9.9	Grouped Table Views	295
9.10	Indexed Table Views	298
9.11	Dynamic Table Views	304
9.12	Whitening Text in Custom Cells	307
9.13	Summary	311
	Exercises	313
10	File Management	315
10.1	The Home Directory	315
10.2	Enumerating a Directory	316
10.3	Creating and Deleting a Directory	318
10.4	Creating Files	319
10.5	Retrieving and Changing Attributes	323
10.5.1	Retrieving attributes	324
10.5.2	Changing attributes	325
10.6	Working with Resources and Low-Level File Access	327
10.7	Summary	330
	Exercises	331
11	Working with Databases	333
11.1	Basic Database Operations	333
11.1.1	Opening, creating, and closing databases	335
11.1.2	Table operations	335
11.2	Processing Row Results	337
11.3	Prepared Statements	340
11.3.1	Preparation	340
11.3.2	Execution	341
11.3.3	Finalization	341
11.3.4	Putting it together	341
11.4	User-Defined Functions	343
11.5	Storing BLOBs	347
11.6	Retrieving BLOBs	351
11.7	Summary	353
	Exercises	353
12	XML Processing	355
12.1	XML and RSS	355
12.1.1	XML	355
12.1.2	RSS	357
12.1.3	Configuring the XCode project	360

12.2	Document Object Model (DOM)	361
12.3	Simple API for XML (SAX)	368
12.4	An RSS Reader Application	377
12.5	Putting It Together	380
12.6	Summary	381
	Exercises	381
13	Location Awareness	383
13.1	The Core Location Framework	383
13.1.1	The CLLocation class	385
13.2	A Simple Location-Aware Application	387
13.3	Google Maps API	390
13.4	A Tracking Application with Maps	396
13.5	Working with Zip Codes	401
13.6	Working with the Map Kit API	404
13.6.1	The MKMapView class	404
13.6.2	The MKCoordinateRegion structure	404
13.6.3	The MKAnnotation protocol	405
13.6.4	The MKAnnotationView class	407
13.6.5	The MKUserLocation class	409
13.6.6	The MKPinAnnotationView class	409
13.7	Summary	411
	Exercises	411
14	Working with Devices	413
14.1	Working with the Accelerometer	413
14.1.1	Basic accelerometer values	413
14.1.2	Accelerometer example	414
14.2	Working with Audio	418
14.2.1	Playing short audio files	418
14.2.2	Recording audio files	420
14.2.3	Playing audio files	421
14.2.4	Using the media picker controller	422
14.2.5	Searching the iPod Library	424
14.3	Playing Video	427
14.4	Accessing Device Information	428
14.5	Taking and Selecting Pictures	429
14.5.1	Overall approach to taking and selecting pictures	429
14.5.2	Detailed example of taking and selecting pictures	430
14.6	Monitoring the Device Battery	432
14.6.1	Battery level	432
14.6.2	Battery state	433
14.6.3	Battery state and level notifications	434
14.6.4	Putting it together	434

14.7	Accessing the Proximity Sensor	435
14.7.1	Enabling proximity monitoring	435
14.7.2	Subscribing to proximity change notification	436
14.7.3	Retrieving the proximity state	436
14.8	Summary	437
	Exercises	437
15	Internationalization	439
15.1	String Localization	439
15.2	Date Formatting	445
15.3	Number Formatting	448
15.4	Sorted List of Countries	450
15.5	Summary	451
	Exercises	451
16	Custom User Interface Components	453
16.1	Text Field Alert View	453
16.2	Table Alert View	457
16.3	Progress Alert View	462
16.4	Summary	467
	Exercises	467
17	Advanced Networking	469
17.1	Determining Network Connectivity	469
17.1.1	Determining network connectivity via EDGE or GPRS	470
17.1.2	Determining network connectivity in general	471
17.1.3	Determining network connectivity via Wi-Fi	471
17.2	Uploading Multimedia Content	472
17.3	Computing MD5 Hash Value	475
17.4	Multithreaded Downloads	477
17.4.1	The Multithreaded Downloads application	477
17.4.2	Asynchronous networking	479
17.5	Push Notification	484
17.5.1	Configuring push notification on the server	484
17.5.2	Configuring the client	490
17.5.3	Coding the client	493
17.5.4	Coding the server	496
17.6	Local Notification	497
17.7	Large Downloads and Uploads	497
17.8	Sending Email	499
17.9	Summary	502
	Exercises	503

18 Working with the Address Book Database	505
18.1 Introduction	505
18.2 Property Types	506
18.3 Accessing Single-Value Properties	506
18.3.1 Retrieving single-value properties	507
18.3.2 Setting single-value properties	508
18.4 Accessing Multivalued Properties	508
18.4.1 Retrieving multivalued properties	508
18.4.2 Setting multivalued properties	510
18.5 Person and Group Records	512
18.6 Address Book	513
18.7 Multithreading and Identifiers	515
18.8 Person Photo Retriever Application	515
18.9 Using the ABUnknownPersonViewController Class	517
18.10 Using the ABPeoplePickerNavigationController Class	518
18.11 Using the ABPersonViewController Class	520
18.12 Using the ABNewPersonViewController Class	522
18.13 Summary	523
Exercises	524
19 Core Data	525
19.1 Core Data Application Components	525
19.2 Key Players	525
19.2.1 Entity	526
19.2.2 Managed object model	526
19.2.3 Persistent store coordinator	527
19.2.4 Managed object context	527
19.2.5 Managed object	527
19.2.6 The Core Data wrapper class	528
19.3 Using the Modeling Tool	531
19.4 Create, Read, Update, and Delete (CRUD)	536
19.4.1 Create	536
19.4.2 Delete	537
19.4.3 Read and update	537
19.5 Working with Relationships	539
19.6 A Search Application	540
19.6.1 The UISearchDisplayController class	540
19.6.2 Main pieces	543
19.7 Summary	547
Exercises	548
20 Undo Management	549
20.1 Understanding Undo Management	549
20.1.1 Basic idea	549

20.1.2	Creating an undo manager	550
20.1.3	Registering an undo operation	550
20.1.4	Hooking into the undo management mechanism	551
20.1.5	Enabling shake-to-edit behavior	552
20.2	Detailed Example	552
20.2.1	The view controller class	553
20.2.2	First-responder status	553
20.2.3	Editing mode and the NSUndoManager instance	553
20.2.4	Registering undo actions	554
20.3	Wrapping Up	555
20.4	Summary	556
	Exercises	556
21	Copy and Paste	557
21.1	Pasteboards	557
21.1.1	System pasteboards	557
21.1.2	Creating pasteboards	557
21.1.3	Properties of a pasteboard	558
21.2	Pasteboard Items	558
21.2.1	Pasteboard items	558
21.2.2	Manipulating pasteboard items	559
21.3	The Editing Menu	560
21.3.1	The standard editing actions	561
21.3.2	The UINavigationController class	561
21.3.3	The role of the view controller	562
21.4	Putting It Together	562
21.4.1	The image view	563
21.4.2	The view controller	564
21.5	Summary	568
	Exercises	569
22	Offline Mode	571
22.1	Setting Up the Project	571
22.1.1	Adding support for libxml2	572
22.1.2	Adding the TouchXML Objective-C wrapper	573
22.2	Parsing XML Using the TouchXML Wrapper	574
22.2.1	The structure of the RSS feed	574
22.2.2	Obtaining the XML document	575
22.2.3	Extracting parking availability	575
22.2.4	Monitoring the feed and disseminating the updates	577
22.3	Showing a Screen Shot of the Last Session	579
22.4	The TableView Controller	580
22.5	Summary	584
	Exercises	584

23 Peer-to-Peer Communication	587
23.1 Basic Chat Application	587
23.1.1 Peer discovery and connection establishment	587
23.1.2 Creating the session	588
23.1.3 Setting up a data-receive handler	589
23.1.4 Sending data	591
23.2 Exchanging Pictures	592
23.2.1 Sending an image	592
23.2.2 Receiving an image	592
23.3 Summary	593
Exercises	593
24 Developing for the iPad	595
24.1 The Cities App: Iteration 1	595
24.1.1 The application delegate class	595
24.1.2 The CitiesViewController class	596
24.1.3 The StatesViewController class	598
24.1.4 Creating the UI	600
24.1.5 Wrapping it up	604
24.2 The Cities App: Iteration 2	604
24.2.1 Initializing the popover view controller with a navigation controller	605
24.2.2 Showing the popover	607
24.2.3 Wrapping it up	607
24.3 Split View Controller	608
24.3.1 An example of the split view controller	608
24.3.2 Dissecting the split view controller	610
24.4 Modal View Controller Presentation Styles	612
24.5 Summary	612
Exercises	614
Appendix A Saving and Restoring App State	615
Appendix B Invoking External Applications	619
Appendix C App Store Distribution	621
Appendix D Using XCode	623
D.1 XCode Shortcuts	623
D.2 Creating Custom Templates	623
D.3 Build-Based Configurations	626
D.4 Using Frameworks	629
Appendix E Unit Testing	633
E.1 Adding a Unit Test Target	633
E.2 Adapting to Foundation	634

E.3	The Model	636
E.4	Writing Unit Tests for the Employee Class	638
	E.4.1 The setUp and tearDown methods	639
	E.4.2 Testing for equality	640
	E.4.3 Testing for nullity	640
E.5	Adding a Build Dependency	641
E.6	Running the Tests	641
Appendix F Working with Interface Builder		643
F.1	National Debt Clock Application	643
	F.1.1 Creating the project	643
	F.1.2 Creating the view controller class	643
	F.1.3 The application delegate class	646
	F.1.4 Building the UI	647
F.2	Toolbar Application	661
	F.2.1 Writing code	661
	F.2.2 Building the UI	663
	F.2.3 Putting it together	669
References and Bibliography		671
Index		673

1

Getting Started

This chapter serves as a quick introduction to the tools bundled with the iOS SDK. It also shows you basic development steps used on the iOS operating system that include coding, user interface (UI) design, and debugging. You do not have to understand everything in this chapter as we will go over these concepts throughout the book. What you need to get from this chapter is a feeling for iOS development using XCode.

We start with some basics of the XCode IDE in Section 1.1. Next, Section 1.2 talks about the UI design tool Interface Builder. After that, we show you how to use the built-in debugger in XCode in Section 1.3. Next, Section 1.4 shows you different sources of information for obtaining additional help. Finally, we summarize the chapter in Section 1.5.

1.1 iOS SDK and IDE Basics

In this section, we walk you through the process of creating your first iPhone application. But first, you need to obtain the iOS SDK and install it on your Intel-based Mac.

1.1.1 *Obtaining and installing the SDK*

Obtaining and installing the iOS SDK is easy. Just follow these steps:

1. Get your iPhone developer Apple ID and password from:
<http://developer.apple.com/iphone/>
2. Download the latest iOS SDK from the site mentioned above.
3. Install the iOS SDK on your Mac.

Now, you're ready to create your first project — read on!

1.1.2 *Creating a project*

Let's use XCode to create an iOS project targeting the iPhone device. First, locate XCode and launch it. You can use Spotlight to find it or you can navigate to `/Developer/Applications/XCode`.

XCode is the central application for writing, designing, debugging, and deploying your iOS applications. You will use it a lot, so go ahead and add it to the Dock.

From XCode, select `File` → `New Project`. You should see a window, similar to the one shown in Figure 1.1, asking you for the type of project you want to create. Choose the default and create a window-based application. This is the most generic type of iPhone project and the one that can be customized for different needs.



Figure 1.1 Choosing a window-based application in the project creation process.

Click `Choose`, enter the name of your project (here, we're using `My Project`), and click `Save`. A new directory is created with the name you entered, and several files are generated for you. You should now see the newly created iPhone project as shown in Figure 1.2.

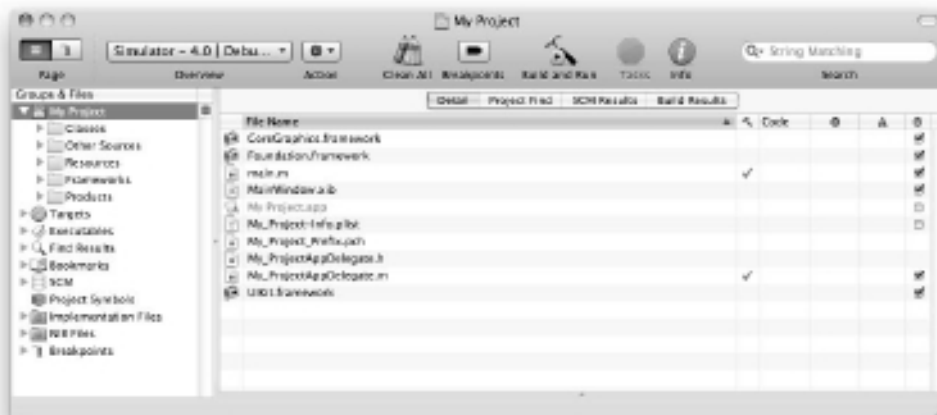


Figure 1.2 A newly created iPhone project in XCode.

1.1.3 Familiarizing yourself with the IDE

As you can see from Figure 1.2, the main window is divided into several areas. On the top, you will find the toolbar (Figure 1.3). The toolbar provides quick access to common tasks. It is fully configurable; you can add and remove tasks as you want. To customize the toolbar, Control-click it and choose *Customize Toolbar*. A window with a set of items will be shown so you can drag your favorite task on the toolbar. Click *Done* when you're finished. To remove an item, Control-click on it and choose *Remove Item*.

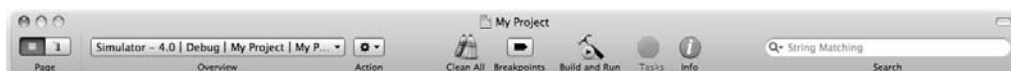


Figure 1.3 The XCode toolbar.

On the left-hand side, you'll see the *Groups & Files* list (Figure 1.4).

This list is used to organize the source code, frameworks, libraries, executables, and other types of files in your project.

The list shows several files and groups. Groups can contain other groups and files. You can delete a group as well as create a new one. The group indicated by the blue icon whose name is the same as the name you've chosen as the project name is a *static group*. Underneath it, you see all your headers, implementations, resources (images, audio files, and so on), and other related files. The folderlike

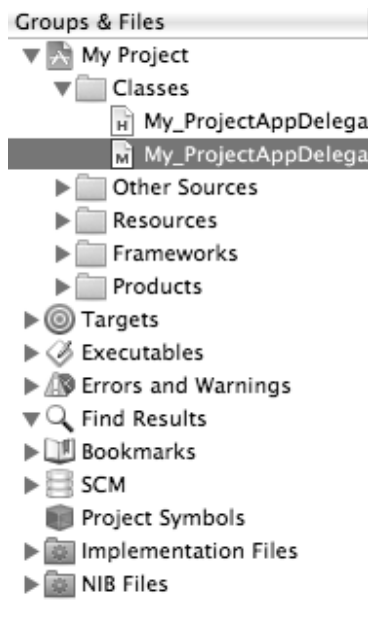


Figure 1.4 The Groups & Files list in XCode.

yellow groups act conceptually as containers. You can have containers inside other containers and all files inside these containers live in the same directory on the disk. The hierarchy only helps you organize things. You have full freedom to organize your project's layout as you like. The compiler will pick up the resources, headers, and implementation files when it builds your application.

The other kind of groups that are listed below the project group are called *smart groups*. There are two types of smart groups: (1) built-in smart groups and (2) custom smart groups. The content of the built-in smart groups cannot be customized. Examples of these groups include executables, bookmarks, errors/warnings, and targets. Customized smart groups are shown in purple, and two predefined groups are created for you when you create a new project. The first group is named Implementation Files, and all implementation files are listed underneath it. The other is called Nib Files, underneath which all UI files are listed.

Figure 1.5 shows the Details view and the text editor beneath it.

Selecting an item in the Groups & Files list will result in its details being shown in the Details view. You can go to a full-editor window using `Command-Shift-E`.

1.1.4 Looking closely at the generated code

Expand the Classes and Other Sources groups. You will notice several files that live underneath these two groups. Click on the `main.m` file and expand to a full-editor view.



Figure 1.5 The Details view with the text editor view.

The `main.m` file looks very similar to a C file with a `main()` function. As we will see later in this book, all that `main()` does is prepare for memory management and launch the application.

Click on the `My_ProjectAppDelegate.h` file under the `Classes` group. You will notice that the editor changes its content. This file contains the declaration of the application delegate class. Every application that runs on iOS has a delegate object that handles critical phases of its life cycle.

Click on `My_ProjectAppDelegate.m`. This file with the `.m` extension is the counterpart of the previous `.h` file. In it, you see the actual implementation of the application delegate class. Two methods of this class are already implemented for you. The `applicationDidFinishLaunching:` method is one of those methods; it handles a particular phase of the application life cycle. The other method, `dealloc`, is a method where memory used by this object is released. In iOS, you manage the allocation and freeing of memory as there is no garbage collection. Memory management is crucial in iOS development, and mastering it is very important. The first chapters in this book are dedicated to teaching you exactly that—and much more.

The generated files and resources are adequate for starting the application. To launch the application, click on `Build and Go` in the toolbar or press the `Command-Enter` key combination. You'll notice that the application starts in the simulator and it shows only a white screen with the status bar on top. Not very useful, but it works!

1.2 Creating Interfaces

To be useful, an iPhone application needs to utilize the amazing set of UI elements available from the SDK. Our generated iPhone application contains a single UI element: a window.

All iPhone apps have windows (usually one). A *window* is a specialized view that is used to host other views. A *view* is a rectangle piece of real estate on the 320×480 iPhone screen. You can draw in a view, animate a view by flipping it, and receive multitouch events on it. In iPhone development, most of your work goes towards creating views, managing their content, and animating their appearance and disappearance.

Views are arranged into a hierarchy that takes the shape of a tree. A tree has a root element and zero or more child elements. In iOS, the window is the root element and it contains several child views. These child views can in turn contain other child views and so on and so forth.

To generate views and manage their hierarchy, you can use both Interface Builder (IB) and Objective-C code. IB is an application that comes with the SDK that allows you to graphically build your view and save it to a file. This file is then loaded at runtime and the views stored within it come to life on the iPhone screen.

As we mentioned before, you can also use Objective-C code to build the views and manage their hierarchy. Using code is preferred over using IB for the following reasons. First, as a beginner, you need to understand all aspects of the views and their hierarchy. Using a graphical tool, although it simplifies the process, does hide important aspects of the process. Second, in advanced projects, your views' layouts are not static and change depending on the data. Only code will allow you to manage this situation. Finally, IB does not support every UI element all the time. Therefore, you will sometimes need to go in there and generate the views yourself.

The following section teaches you how to use IB. However, for the most part in this book, Objective-C code is used to illustrate the UI concepts. For extensive coverage of Interface Builder, please see Appendix E.

1.2.1 Interface Builder

The project has a basic window resource file. This file can be found under the `Resources` group. Expand the `Resources` group and locate the file `MainWindow.xib`. This file contains the main window of the application. This file is an `.xib` file that stores the serialized objects in the interface. When the project is built, this file is converted to the more optimized format `.nib` and loaded into memory when one or more of the UI components stored in it are requested.

Double-click on the `MainWindow.xib` file to launch IB. IB starts by opening four windows. The first window shows the main window stored in the file. The second window shows the document window listing the different objects stored in the file. The third window is the Library window containing all the UI objects that you can add to the file. The fourth and final window is the Inspector window with its four panes.

The Inspector window shows the attributes of the currently selected object. If you click on an object, the Inspector window shows you its attributes distributed among four different panes. Each pane has several sections. You can change these attributes (such as color, position, and connections) and the changes will propagate to your project's user interface.

The main window of the application is white; let's change it to yellow. Click on the window object in the document window. In the Inspector window, make sure that the leftmost pane is selected. In the View section of this pane, change the background color to yellow as shown in Figure 1.6.

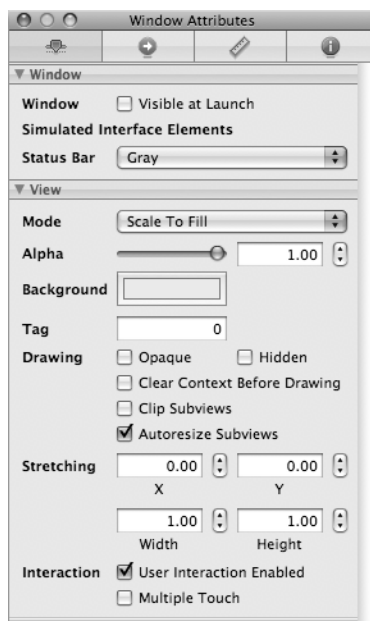


Figure 1.6 The attributes pane in the Inspector window of Interface Builder.

Go to XCode and run the application. Notice how the main window of the application has changed to yellow.

It is important to keep the project open in XCode while working with IB. XCode and IB communicate well when both applications are open.

To build a user interface, you start with a view and add to it subviews of different types. You are encouraged to store separate views in separate `.xib` files. This is important, as referencing one object in a file will result in loading all objects to main memory. Let's go ahead and add a label view to our window. This label will hold the static text `Hello iPhone`.

A label is one of the many UI components available for you. These components are listed under several groups in the Library. Select `Tools` → `Library` to show the Library window if it's not shown. As shown in Figure 1.7, locate the `Inputs & Values` section.