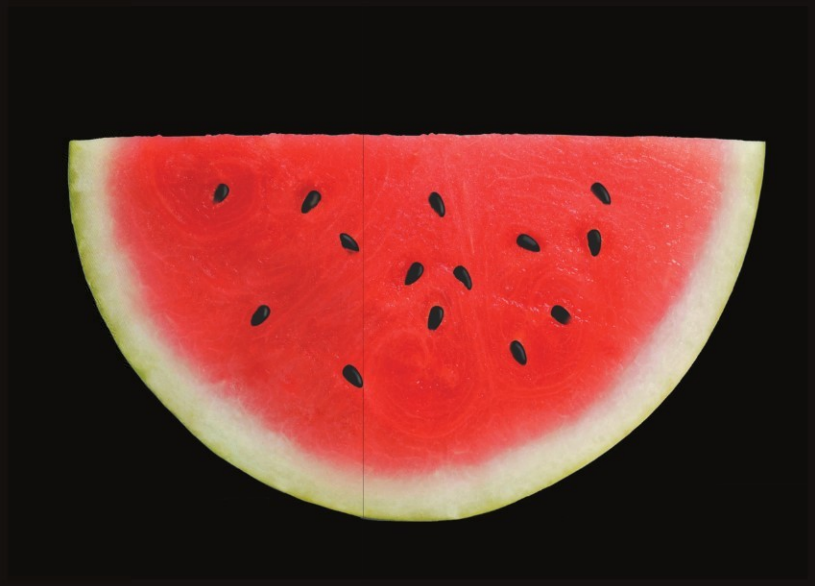
Core code concepts and techniques for
iPhone and iPad app developers



iOS 6 Recipes

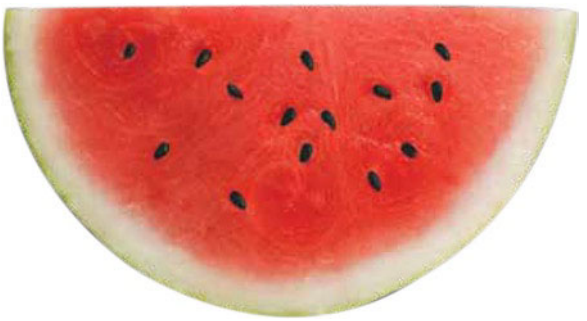
A Problem-Solution Approach

Hans-Eric Grönlund | Colin Francis | Shawn Grimes

Apress®

iOS 6 Recipes

A Problem-Solution Approach



Hans-Eric Grönlund
Colin Francis
Shawn Grimes

Apress®

iOS 6 Recipes

ISBN 978-1-4302-4599-5

ISBN 978-1-4302-4600-8 (eBook)

President and Publisher: Paul Manning
Lead Editor: Steve Anglin
Developmental Editor: Douglas Pundick
Technical Reviewer: Anselm Bradford
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editor: Anamika Panchoo
Copy Editor: Linda Seifert
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

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

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

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

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code.

I dedicate this book to my wife, Esra, who supports my whimsical endeavors.

—Hans-Eric Grönlund

Contents at a Glance

About the Authors xix

About the Technical Reviewer xxi

Acknowledgments xxiii

Introduction xxv

Chapter 1: Application Recipes 1

Chapter 2: Layout Recipes 79

Chapter 3: Table and Collection View Recipes 113

Chapter 4: Location Recipes 175

Chapter 5: Motion Recipes 217

Chapter 6: Map Recipes 243

Chapter 7: Social Network Recipes 303

Chapter 8: Camera Recipes 343

Chapter 9: Multimedia Recipes 383

Chapter 10: Image Recipes 417

Chapter 11: User Data Recipes 457

Chapter 12: Data Storage Recipes 503
Chapter 13: Data Transmission Recipes 563
Chapter 14: Game Kit Recipes 595

Index 651

Contents

About the Authors xix

About the Technical Reviewer xxi

Acknowledgments xxiii

Introduction xxv

Chapter 1: Application Recipes 1

Recipe 1-1: Setting Up a Single-*V*iew Application 1

Recipe 1-2: Linking a Framework 5

Recipe 1-3: Adding a User Interface Control View 8

Recipe 1-4: Creating an Outlet 10

Recipe 1-5: Creating an Action 14

Recipe 1-6: Creating a Class 17

Recipe 1-7: Adding an Info.plist Property 21

Recipe 1-8: Adding a Resource File 22

Recipe 1-9: Using Storyboards 24

So What's in a Story(board)? 25

Setting Up the Application with a Storyboard 26

Adding a New Scene to the Storyboard 29

Adding a Table View Scene 33

Adding a Detail View 37
Setting Up a Custom View Controller 41
Using Cell Prototypes 45
Recipe 1-10: Handling Errors 48
Setting Up a Framework for Error Handling 49
Notifying the User 52
Implementing Recovery Options 56
Recipe 1-11: Handling Exceptions 59
A Strategy for Handling Exceptions 60
Setting Up a Test Application 60
Intercepting Uncaught Exceptions 61
Reporting Errors 64
Adding the Button 65
Emailing the Report 67
A Final Touch 69
Recipe 1-12: Adding a Lite Version 70
Adding a Build Target 70
Coding for a Specific Version 72
Recipe 1-13: Adding Launch Images 73
Launch Image Files 73
Designing Launch Images 76
Summary 77

Chapter 2: Layout Recipes 79
Recipe 2-1: Using Autolayout 79
Autolayout Constraints 79
Constraint Priorities 84
Adding a Trailing Button 87
Recipe 2-2: Programming Autolayout 89
Setting Up the Application 89
The Visual Format Language 91

Adding Image Views 95
Defining the Image Views' Constraints 95
Recipe 2-3: Debugging Autolayout 101
Dealing with Ambiguous Layouts 101
Handling Unsatisfiability 105
Summary 111

Chapter 3: Table and Collection View Recipes 113
Recipe 3-1: Creating an Ungrouped Table 113
Setting Up the Application 113
Adding a Model for Countries 117
Displaying Data in a Table View 120
A Note on Cached Cells 122
Configuring the Cells 123
A Note on Rounded Corners 126
Implementing the Accessory Views 127
Enhanced User Interaction 133
A Note on Cell View Customization 135
Recipe 3-2: Editing a UITableView 136
UITableView Row Animations 138
But Wait, There's More! 139
Recipe 3-3: Reordering a UITableView 142
Recipe 3-4: Creating a Grouped UITableView 143
Recipe 3-5: Registering a Custom Cell Class 150
Creating a Custom Table View Cell Class 151
Registering Your Cell Class 153
Recipe 3-6: Creating a Flag Picker Collection View 154
Setting Up the Application 155
Creating a Data Model 158
Building the Flag Picker 159
Defining the Collection View Interface 163

Displaying the Flag Picker 169
Using Autolayout to Center the Headers 172
Summary 173

Chapter 4: Location Recipes 175
About Core Location 175
Standard and Significant Change Services 176
What's New in iOS 6 176
Requiring Location Service 177
Recipe 4-1: Getting Basic Location Information 178
Setting Up the Application 178
Starting and Stopping Location Updates 180
Receiving Location Updates 183
Testing Location Updates 185
Recipe 4-2: Significant Location Changes 186
Setting Up the Application 187
Enabling Background Updates 188
Adding Local Notifications 190
Recipe 4-3: Tracking Magnetic Bearing 191
About Heading Tracking 191
Setting Up the Application 191
Starting and Stopping Heading Updates 193
Implementing Delegate Methods 194
Recipe 4-4: Tracking True Bearing 197
Adding True Bearing 197
Recipe 4-5: Region Monitoring 202
A Thing or Two About Regions 202
Welcome to Baltimore! 203

Recipe 4-6: Implementing Geocoding 208
Implementing Reverse Geocoding 208
Implementing Forward Geocoding 213
Best Practices 216
Summary 216

Chapter 5: Motion Recipes 217
Recipe 5-1: Recognizing Shake Events 217
Intercepting Shake Events 217
Subclassing the Window 218
Implementing Shake Notifications 220
Testing Shake Events 221
Recipe 5-2: Accessing Raw Core Motion Data 222
The Core Motion Sensors 222
Setting Up the Project 223
Accessing Sensor Data 225
Pushing or Pulling 230
Selecting an Update Interval 230
The Nature of Raw Motion Data 230
Recipe 5-3: Accessing Device Motion Data 231
The Device Motion Class 231
Setting Up the Application 232
Accessing Device Motion Data 234
Setting a Reference Frame 235
Recipe 5-4: Moving a Label with Gravity 237
Setting Up the Application 237
Moving the Label with Gravity 239
Adding Acceleration 241
Summary 242

Chapter 6: Map Recipes 243

Recipe 6-1: Showing a Map with the Current Location 243

Setting Up the Application 243

User-Controlled Tracking 248

Recipe 6-2: Marking Locations with Pins 250

Adding Annotation Objects 251

Changing the Pin Color 253

Recipe 6-3: Creating Custom Annotations 255

Setting Up the Application 255

Creating a Custom Annotation Class 257

Creating a Custom Annotation View 258

Customizing the Callouts 262

Adding a Detailed View 264

Recipe 6-4: Dragging a Pin 267

Adding a Draggable Pin 268

Recipe 6-5: Adding Overlays to a Map 270

Creating the Overlays 270

Recipe 6-6: Grouping Annotations Dynamically 273

A Forest of Pins 273

Implementing a Solution 278

Adding Color Coding 281

Recipe 6-7: Starting Maps from Your App 285

Adding Map Items 286

Launching in Directions Mode 289

Recipe 6-8: Registering a Routing App 291

Declaring a Routing App 291

Handling Launches 293

Testing the Routing App 295

Specifying Coverage Area 298

Summary 301

Chapter 7: Social Network Recipes 303

Recipe 7-1: Sharing Content with the Activity View 303

Setting Up an Activity View Controller 304

Excluding Activity View Items 306

Including Activity View Items 309

Recipe 7-2: Sharing Content Using a Compose View 313

Recipe 7-3: Sharing Content Using SLRequest 316

Setting Up the Main View 317

Requesting Access to Twitter Accounts 318

Handling Multiple Accounts 322

Recipe 7-4: Retrieving Tweets 325

Setting Up a Navigation-Based Application 326

Displaying Available Feeds 327

Displaying Tweets 331

Showing Individual Tweets 337

Summary 342

Chapter 8: Camera Recipes 343

Recipe 8-1: Taking Pictures 343

Setting Up the User Interface 343

Accessing the Camera 344

Retrieving a Picture 348

Implement Basic Editing 349

Saving Picture to Photos Album 350

Recipe 8-2: Recording Video 351

Recipe 8-3: Editing Videos 353

Recipe 8-4: Using Custom Camera Overlays 357

Recipe 8-5: Displaying Camera Preview with AVCaptureSession 361

Recipe 8-6: Capturing Still Images with AVCaptureSession 364

Adding a Capture Button 365

Recipe 8-7: Capturing Video with AVCaptureSession 370
Adding a Video Recording Mode 370
Recipe 8-8: Capturing Video Frames 377
Summary 381

Chapter 9: Multimedia Recipes 383
Recipe 9-1: Playing Audio 383
Setting Up the Application 383
Setting Up the Audio Player 386
Handling Errors and Interruptions 388
Recipe 9-2: Recording Audio 389
Setting Up an Audio Recorder 391
Handling Interruptions 394
Recipe 9-3: Accessing the Music Library 394
Setting Up a Basic Music Player 394
Handling Notifications 397
Picking Media to Play 398
Querying Media 402
Recipe 9-4: Playing Background Audio 405
Setting Up the User Interface 405
Declaring Background Mode Playback 406
Implementing the Player 408
Summary 415

Chapter 10: Image Recipes 417
Recipe 10-1: Drawing Simple Shapes 417
Recipe 10-2: Programming Screenshots 422
Recipe 10-3: Using Image Views 426
Recipe 10-4: Scaling Images 433
In Review 441

Recipe 10-5: Manipulating Images with Filters 441
Combining Filters 446
Creating Thumbnail Images for the Table View 449
Recipe 10-6: Detecting Features 451
Summary 455

Chapter 11: User Data Recipes 457
Recipe 11-1: Working with NSCalendar and NSDate 457
Recipe 11-2: Fetching Calendar Events 463
Recipe 11-3: Displaying Events in a Table View 467
Recipe 11-4: Viewing, Editing, and Deleting Events 473
Recipe 11-5: Creating Calendar Events 476
Creating Recurring Events 478
Recipe 11-6: Creating Reminders 480
Setting Up the Application 480
Requesting Access to Reminders 482
Creating Time-Based Reminders 484
Creating Location-Based Reminders 487
Recipe 11-7: Accessing the Address Book 492
Recipe 11-8: Setting Contact Information 497
Summary 502

Chapter 12: Data Storage Recipes 503
Recipe 12-1: Persisting Data With NSUserDefaults 503
Recipe 12-2: Persisting Data Using Files 508
Recipe 12-3: Using Core Data 513
Understanding Core Data 513
Setting Up Core Data 514
Designing the Data Model 516
Setting Up the Vocabularies Table View 522

Implementing the Words View Controller 530
Adding a Word Edit View 534
Persisting Data on iCloud 543
Recipe 12-4: Storing Key-Value Data in iCloud 543
Setting Up iCloud For an App 546
Persisting Data in iCloud Key-Value Store 550
Caching iCloud Data Locally Using NSUserDefaults 553
Recipe 12-5: Storing UIDocuments in iCloud 555
Summary 562

Chapter 13: Data Transmission Recipes 563
Recipe 13-1: Composing Text Messages 563
Recipe 13-2: Composing Email 568
Attaching Data to Mail 571
Recipe 13-3: Printing an Image 576
Recipe 13-4: Printing Plain Text 583
Recipe 13-5: Printing a View 586
Recipe 13-6: Formatted Printing with Page Renderers 588
Summary 594

Chapter 14: Game Kit Recipes 595
Recipe 14-1: Making Your App Game Center Aware 595
Implementing the Game 595
Registering with iTunes Connect 607
Authenticating Local Player 610
Displaying Game Center From Your App 614
Recipe 14-2: Implementing Leaderboards 616
Defining the Leaderboards 616
Reporting Scores to Game Center 619

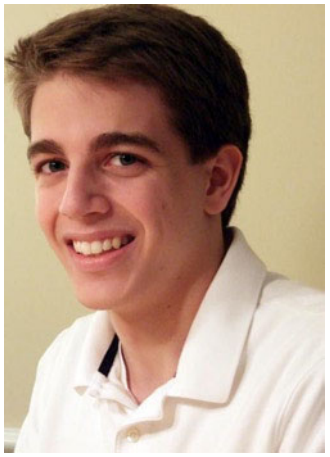
Recipe 14-3: Implementing Achievements....621
Defining Achievements in iTunes Connect....621
Reporting the Achievements....623
Recipe 14-4: Creating a Simple Turn-Based Multiplayer Game....630
Building the Tic Tac Toe Game....630
Preparing the Game for Game Center....633
Implementing Matchmaking....636
Encoding and Decoding Match Data....641
Handling Turn-Based Events....646
Summary....649

Index....651

About the Authors

Hans-Eric Grönlund has developed software professionally since 1990. He's currently an employee at Knowit, a leading Scandinavian IT Consultancy company, where he helps teams develop software with agile methods.

Software development is not only Hans-Eric's profession; it's also his hobby. In his spare time, he has taught himself how to write programs in many different languages on various platforms. His latest passion, obviously, is Objective-C on iOS. Hans-Eric's Twitter ID is @hansEricG.



Colin Francis is an iOS developer originally from Maryland. After studying iOS development with the assistance of Shawn Grimes, the two of them authored iOS 5 Recipes (Apress). He currently lives in Miami, where he works primarily on music-based applications for iOS.

Shawn Grimes and his wife, Stephanie, run Campfire Apps, LLC, a mobile app development company focused on apps for children and families. Together, they have started the APPlied Club program which teaches mobile app development to high school students. Shawn is active in the Baltimore, Maryland development scene and co-runs the Baltimore Mobile Developers group with Chris Stone.

About the Technical Reviewer

Anselm Bradford lectures in digital media at AUT University in New Zealand. In 2013 he will be joining Code for America as a 2013 fellow.

Additionally, he has worked with Apress/friends of ED, O'Reilly Media, Peachpit Press, and Lonely Planet on books in the areas of visual communication, web, and interactive media. He has authorship credit on HTML5 Mastery and CSS3 Solutions (Apress). He may be found on Twitter @anselmbradford, and he occasionally blogs at `AnselmBradford.com`.

Acknowledgments

I'd like to start by thanking Shawn Grimes and Colin Francis, who wrote the iOS 5 version of this book. Without their great effort, I'd still be struggling with it.

I also must thank Matthew Campbell, who helped me get over the Xcode threshold and get rid of the initial confusion; anyone who has made the transition from other IDEs knows that this can be, well, tricky.

Additionally, I'd like to send a big thank you to the Apress team without whose help this book would not have been possible. Anamika Panchoo for keeping me on track at all times, Douglas Pundick and Anselm Bradford for their extremely helpful technical reviews, and last but not least, Linda Seifert, for correcting my language.

Finally, I'm inclined to thank my wife, Esra, my son, Måns, and my daughter, Aylin, to whom I've been more or less inaccessible during the last few months. I promise I'll make it up to you (and hopefully already have when you read this).

—Hans-Eric Grönlund

Introduction

The easy part of software development is knowing how to write code in the programming language at hand. The tougher part is mastering the programming interfaces of the platform and getting to the level where you can effectively turn ideas into working features with real values.

iOS 6, although extremely powerful and easy to use, is no exception to this. Objective-C, by many considered a rather "funky" programming language, is something you'll get your head around rather quickly, even learn to appreciate. However, you're likely to spend a lot of time learning the various APIs and frameworks.

We believe the best way to acquire the necessary knowledge and reach that plateau of high productivity, is through hands-on experience. We think the best way to learn is to code along, creating small projects in which you can test and tweek the features, get a feeling for them before you implement them in your real projects.

With this idea in mind, we created *iOS 6 Recipes*. It contains over 600 pages of sample code accompanied by instructions on how to create small test apps that allow you to run the code on your iOS 6 device or in the iOS Simulator.

We have tried to cover as many topics as possible using the features of iOS 6. We hope it provides the basic fundament you need to start converting your great ideas into fantastic apps.

Who This Book Is For

When you read this book, it will help if you have a basic knowledge of Objective-C, have taken your first steps in Xcode, and written a couple of Hello World apps. If you haven't, don't worry; just pay extra attention to the first eight recipes of Chapter 1. They should provide most of the basics you need to follow along.

How This Book Is Structured

The example-based chapters of this book do not particularly build off of one another, in the hope that you can simply open up to any chapter of specific interest and start building a certain type of

application. It is recommended that you at least skim Chapter 1, “Application Recipes,” and then Chapter 2, “Autolayout Recipes,” before moving on. The first chapter contains recipes for common tasks, such as creating outlets and actions, which are referenced throughout the text and should be fully understood. The second chapter provides basic knowledge of the new layout scheme of iOS 6. Reading that chapter might prove helpful when you create the user interfaces of the recipes later on.

Throughout this book, it is assumed that you are developing in the latest versions of iOS (6.0) and Xcode (4.5) at the time of writing. This means that every recipe in this text assumes that you will be using Automatic Reference Counting (ARC), and as such does not include significant memory management. This also means that depending on when you are reading this, your results may look slightly different, although the basic functionality should remain similar.

Many of the recipes in this book cannot be fully tested on the iOS simulator, and as such will require both an iOS device and a provisioning profile, which can be acquired when you subscribe to the iOS Developer Program. We’ve pointed out each recipe that cannot be tested in the iOS Simulator.

> **Note** With the introduction of iPhone 5, Apple has added a new screen size to the iPhone family. The new 4 inch screen has the same width but is slightly taller than the old 3.5 inch screen. The recipes in this book use the `Retina 3.5 Full Screen` size metric for their user interfaces. However, thanks to the new Autolayout feature of iOS 6, they will work just as well with the new `Retina 4 Full Screen` size metric. You can freely choose whichever metric works for you.

Downloading the Code

The code for the examples shown in this book is available on the Apress web site, www.apress.com. A link can be found on the book’s information page under the Source Code/Downloads tab. This tab is located underneath the Related Titles section of the page.

Contacting the Author

If you have any questions or comments regarding this book, I’d be happy to hear them. Contact me at hasse42g@gmail.com, or write a comment at my blog, http://www.hans-eric.com.

Chapter 1

Application Recipes

We're going to start this book with a set of recipes dealing with the iOS application, its project, and various basic Xcode tasks. The first eight recipes are fundamental, showing things like how to setup an application, how to connect and reference user interface elements to your code, and how to add images and sound files to your project. If you're new to iOS development we suggest you go through those first before moving on.

We also recommend that you take a closer look at Recipe 1-9 to see whether Storyboards is something for you. Storyboards is the new way of designing user interface in iOS, allowing you to gather several views in one file. Although the examples in this book are based on the old way of creating user interfaces, having one `.xib` file per view controller, you could just as easily do them the storyboards way.

The last four recipes in this chapter deal with miscellaneous topics like how to set up simple APIs for default error and exception handling; how to include a Lite version of your app in your projects, and how to make the app launch seem shorter in the eyes of the user.

Recipe 1-1: Setting Up a Single-View Application

Many of the recipes in this book are implemented in a test application with a single view. Such a project is easy to setup in Xcode using the Single View Application template.

To create a new single-view application project in Xcode, go to the main menu and select File ➤ New ➤ Project. This brings up the dialog with available project templates (see Figure 1-1). The template you're looking for is located in the Application page under the iOS section.

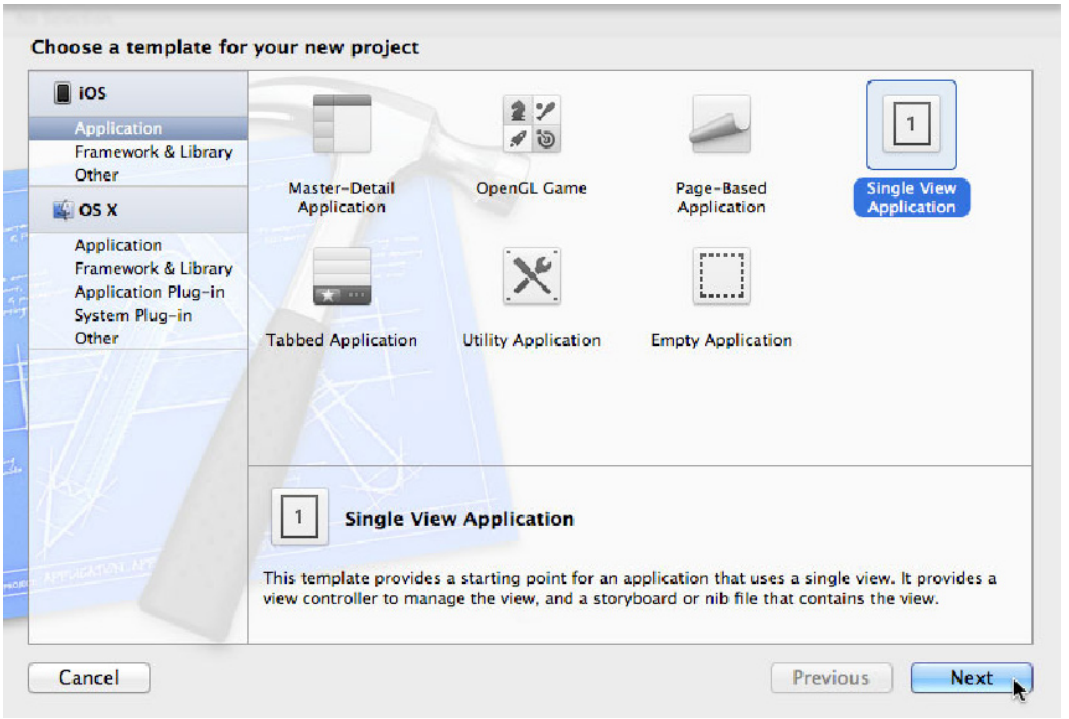


Figure 1-1. *The single view application template in the iOS application section*

After you've selected the Single View Application template and clicked Next, you need to enter a few properties for your application:

- A *Product Name*, for example **My Test App**
- An *Organization Name*, which unless you have one can be your name
- A *Company Identifier*, preferably your Internet domain if you have one

If you like, you can also enter a class prefix that will be applied to all classes you create using the Objective-C file template. This can be a good idea if you want to avoid future name conflicts with third party code, but if this app is only meant for testing a feature, you can leave it blank.

You also need to say which device type your application is for: iPad, iPhone, or both (Universal). Pick iPhone or iPad if you're testing. You can also pick Universal, but then the template will generate more code, which you probably don't need if your only purpose is trying a new feature.

All the code examples in this book assume you're using ARC (Automatic Reference Counting) so make sure that Use Automatic Reference Counting is checked. Also, if you're not planning on using Storyboards (see Recipe 1-9) or unit tests, be sure that the corresponding options are unchecked. Figure 1-2 shows an example of this configuration.

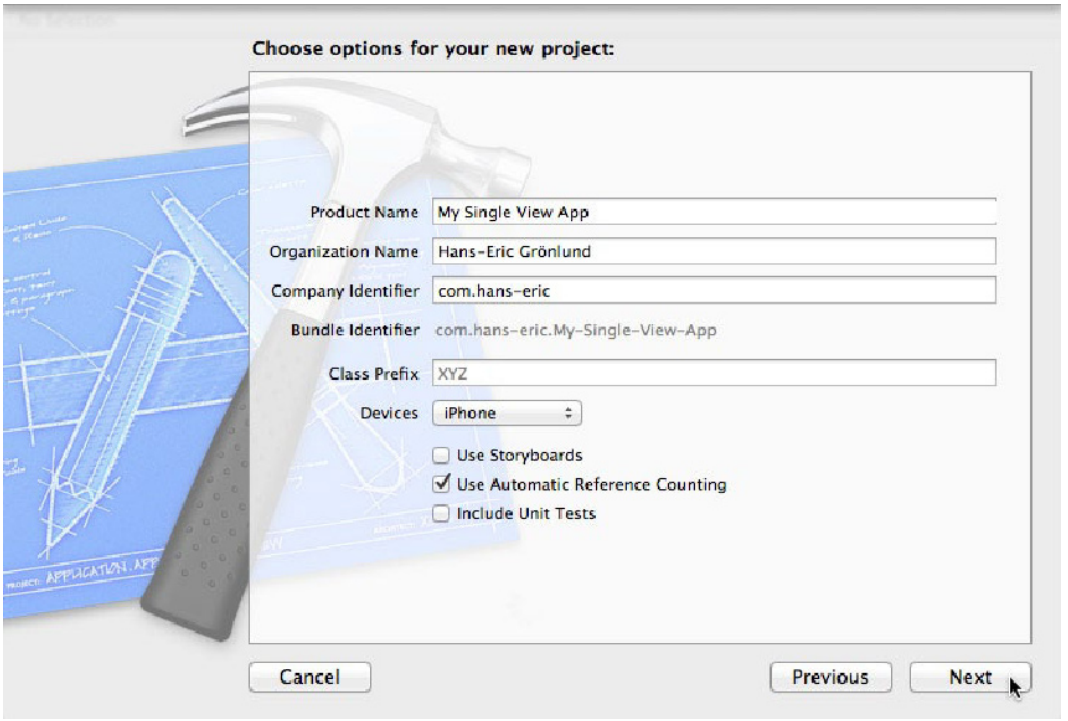


Figure 1-2. *Configuring the project*

Finally, click the Next button and then select a folder where the project is stored. Bear in mind that Xcode creates a new folder for the project within the folder you picked, so select the root folder for your projects.

There's often a good reason to place the project under version control. It allows you to check changes to the code so that you can go back to a previous version if something goes wrong or you just want to see what's been done. Xcode comes with Git, a well-spread open-source version control system. To initialize it for your project, check the Create local git repository for this project checkbox, as in Figure 1-3.

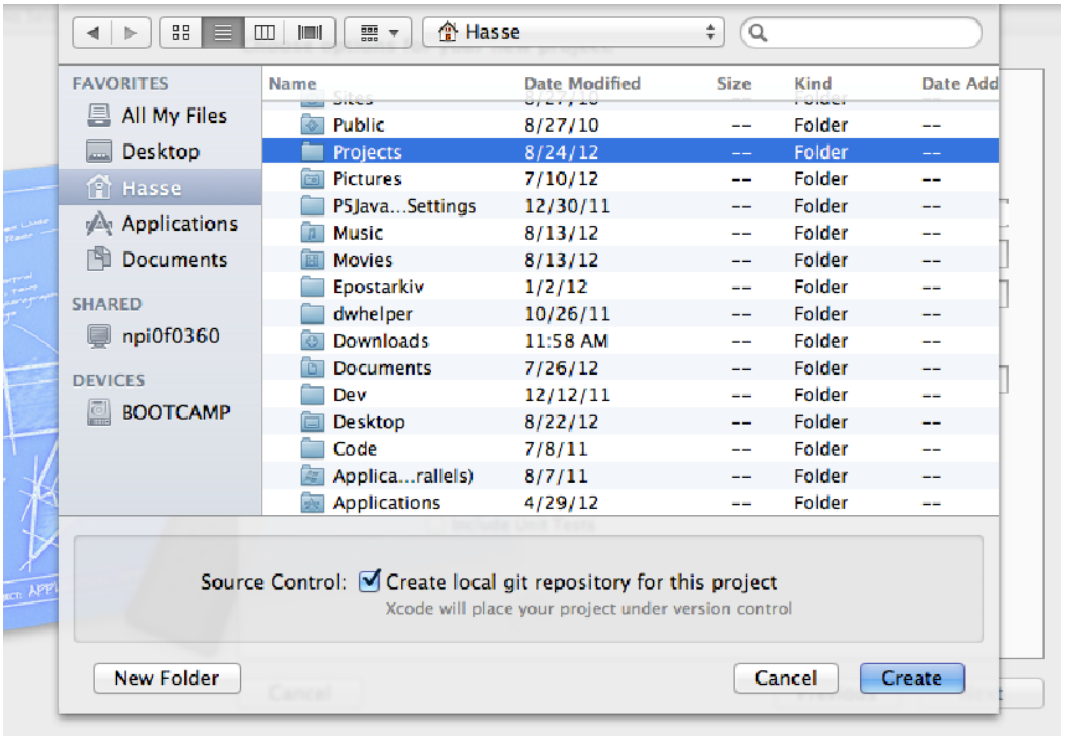


Figure 1-3. *Selecting the parent folder for the project*

Now when you click the Create button, an application with an app delegate and a view controller will be generated for you (see Figure 1-4). The setup is complete and you can build and run the application (which of course at this point only shows a blank screen).

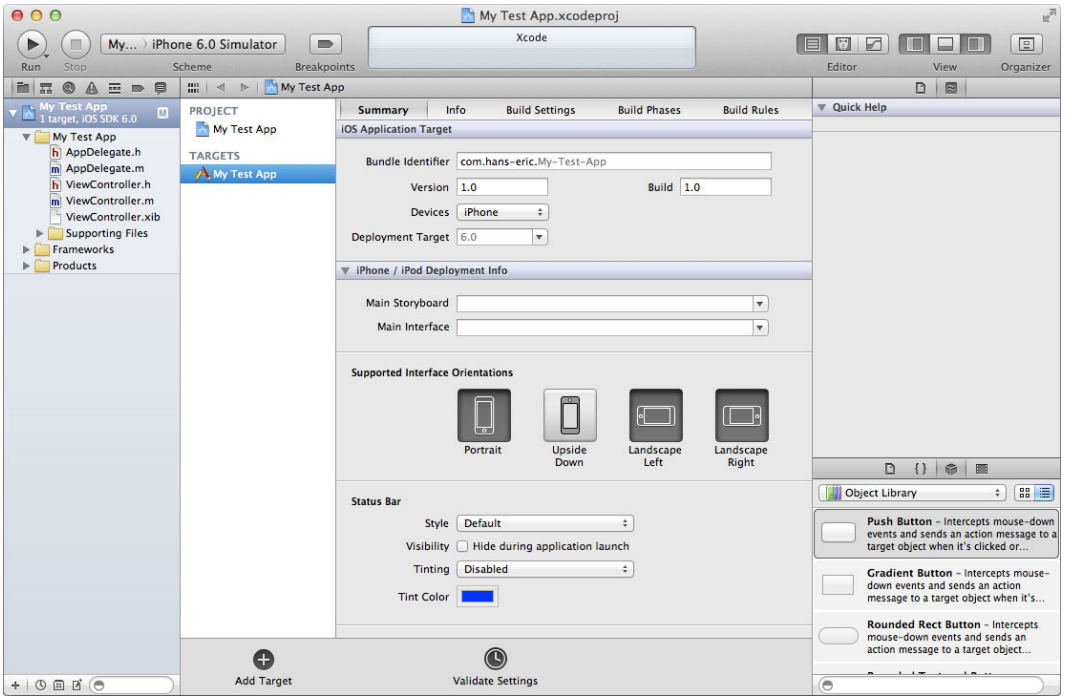


Figure 1-4. A basic application with an app delegate and a view controller

Recipe 1-2: Linking a Framework

The iOS operating system is divided into so called frameworks. To use the functionalities of a framework you need to link the corresponding binary to your project. For the `UIKit`, `Foundation` and `CoreGraphics` frameworks Xcode does this automatically when you create a new project. However, many important features and functions reside in frameworks like `CoreMotion`, `CoreData`, `MapKit`, and so on. For those other frameworks you need to follow these steps to add them.

1. Select the project node (the root node) in the Project navigator panel on the left-side of the Xcode project window. This brings up the Project editor panel.
2. Select the target in the Targets dock. If you have more than one target, for example, a unit test target, you need to perform these steps for all of them.
3. Navigate to the Build Phases tab and expand the section called Link Binary With Libraries. There you see a list of the currently linked frameworks.
4. Click on the Add items (+) button at the bottom of the list. This brings up a list of available frameworks.
5. Select the framework you want to link and use the Add button to include it (see Figure 1-5).

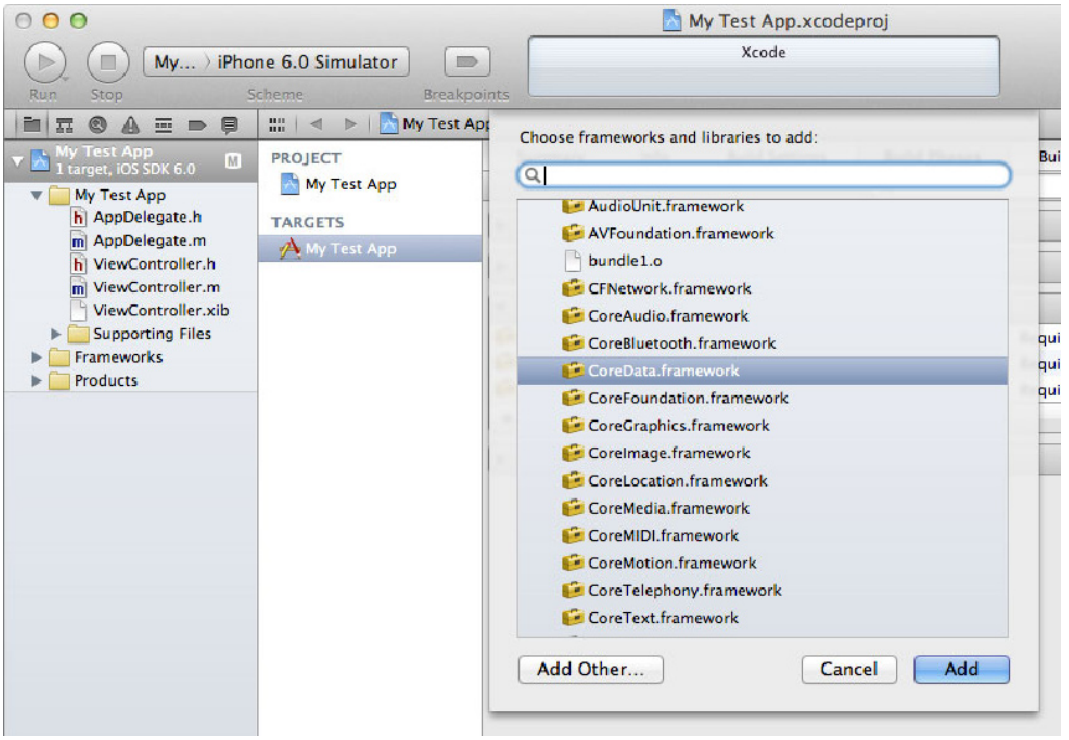


***Figure 1-5.** Adding the Core Data framework*

> **Tip** To make it easier to find a particular framework you can use the search field to filter the list.

When you add a framework to your project, a corresponding framework reference node is placed in your project tree (see Figure 1-6). What you may want to do is to drag and drop the node to the Frameworks folder where the other framework references reside. It's not strictly necessary but helps your project tree stay organized.

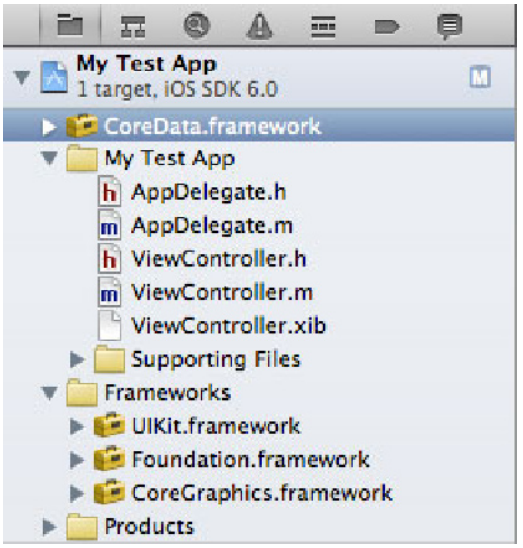


***Figure 1-6.** When adding a framework, a reference node is created at the top of your project tree*

Now, to use the functions and classes from within your code you only need to import the framework API (Application Programming Interface.) This is normally done in a header file (.h) within your project, as in this example where we import the `CoreData` API in the `ViewController.h` file.

```
//
//  ViewController.h
//  My Test App
//

#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>

@interface ViewController : UIViewController

@end
```

> **Note** If you don't know the header file for a framework, don't worry, all framework APIs follow the same pattern, namely `#import <FrameworkName/FrameworkName.h>`.

With the framework binary linked, and the API imported, you can start using its functions and classes in your code.