

# Swift iOS 应用开发实战

刘铭 著

---

Swift iOS in Action

---

- 本书为广大想从Objective-C转向Swift的iOS程序员们提供了捷径
- 基于Swift语言，通过大量案例，详细讲解和演示了iOS中的各种功能和组件的用法，iOS开发中的各种技巧，以及完整iOS项目的开发方法，实战性强



机械工业出版社  
China Machine Press

---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**

# Swift iOS 应用开发实战

---

Swift iOS in Action

---

刘铭 著



机械工业出版社  
China Machine Press

## 图书在版编目 ( CIP ) 数据

Swift iOS 应用开发实战 / 刘铭著. —北京: 机械工业出版社, 2015.4  
(iOS/ 苹果技术丛书)

ISBN 978-7-111-49955-8

I. S… II. 刘… III. 程序语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2015) 第 076221 号

---

## Swift iOS 应用开发实战

---

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 姜 影

责任校对: 殷 虹

印 刷: 三河市宏图印务有限公司

版 次: 2015 年 5 月第 1 版第 1 次印刷

开 本: 186mm×240mm 1/16

印 张: 15.75

书 号: ISBN 978-7-111-49955-8

定 价: 59.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

## 为什么要写这本书

苹果公司在 2014 年 6 月的 WWDC (Worldwide Developers Conference, 苹果全球开发者大会) 上向公众展现了全新的 iOS 8 移动操作系统和 Yosemite 桌面操作系统。作为开发者, 大会开幕之前的任何爆料, 我都是不会错过的。在大会开幕的前一天, 我得知本届大会的宣传标语为: 编写代码 改变世界 (Write the Code. Change the World)。历届 WWDC 的宣传标语都是以全新、卓越、领先、创新、引领等作为关键词, 而这次却使用了非常接地气的“编写代码”一词, 心里感觉怪怪的。在 WWDC 结束的那一刻, 我深深地体会到苹果为什么使用这个词, 因为此时此刻, 苹果做出了一个令所有程序员都为之惊讶的改变——推出了全新的程序设计语言 Swift。Swift 无疑是本届 WWDC 中贴近开发者的最大亮点。

为什么苹果要抛弃已经使用了几十年的 Objective-C, 而去开发一门全新的程序设计语言呢? 按照苹果官方的说法, Objective-C 这门语言太老了, 它无法提供现代语言所具有的那些功能, 而 Swift 语言具有现代、安全、快速等特点。其实在我看来, 这是苹果极其强烈的控制欲在作怪。

在写作本书之前, 我已经写过两本关于 iOS 开发的书, 毫无疑问它们都是在讲如何使用 Objective-C 开发 App 的。可以说 Swift 是我写作这本书的最大挑战, 并且它给我的身心造成了极大的摧残。因为在 WWDC 开幕之前, 我基本上已经完成了第三本关于 Objective-C 图书的初稿。可谁又知道, Swift 的出现如同晴天霹雳、飞来横祸, 让我有种被秒杀的感觉。就这样放弃吗? 不行, 此时此刻, 我想到了八九年前每晚玩《魔兽世界》的情景——不断地“跑尸”。为了完成自己的 iOS 开发三部曲, 必须发扬当年“跑尸”跑不死的精神。总的算下来我的第三本书前后共写了两年的时间。

作为苹果公司独立发布的支持型开发语言，Swift 语言的语法内容混合了 Objective-C、JavaScript、Python 的特点，语法简单、使用方便、易学，大大降低了开发者的入门门槛。同时 Swift 语言还可以与 Objective-C 混合使用，对于用惯了高难度 Objective-C 语言的程序员来说，学会 Swift 更不在话下！

Swift 允许开发者通过更简洁的代码来实现更多的内容。在 WWDC 2014 发布会上，苹果演示了如何只通过一行简单的代码完成一个完整图片列表加载的过程。另外，Swift 还可以让开发人员一边编写程序，一边预览自己的应用程序，从而快速测试应用在某些特殊情况下的反应。

相信对使用 Objective-C 语言开发过 App 的程序员来说，Objective-C 有着诡异的语法，并且是一门与其他 C 语言风格迥异的编程语言，学习难度可想而知。但是，在过去的 20 年里，苹果只支持 Objective-C，这迫使广大程序员不得不学习和使用艰涩难懂的 Objective-C 语言。

随着 Swift 语言的到来，这种简单、好用又安全的编程语言将吸引更多的开发者加入，让苹果软件生态圈更加繁荣。如此，“果粉”将可以在 App Store 和 Mac Store 中下载到更多称心如意的 App。从某种意义上说，Swift 语言是苹果的一项新的商业战略。

对 Swift 语言来说，相信大部分的读者都是从 2014 年 6 月开始接触的，而之后的几个月也应该在刻苦地学习 Swift 这门语言。就像实际生活中我们学习英语一样，精通英语的语法和使用英语进行相互沟通并不完全是一回事。学习程序语言也是如此，虽然程序员可以在短时间内掌握 Swift 的语法，但是使用它来开发 App 是另一个层面的事情了。本书每一章都通过各种各样相对独立的项目，让读者了解 iOS 开发中最常用的几个方面的程序设计技能，包括 Interface Builder、自动布局和 Size Class、表格视图、导航控制器和标签控制器、远程访问及 Facebook 集成等。本书能让那些已经掌握 Swift 语言的程序员尽快上手开发实现各种功能的 App，体验开发的乐趣。

## 读者对象

本书是为想通过学习基本的工具和技术，开发出具有实用功能的、可以在 iOS 平台上面完美运行的 App 的人所准备的。

本书读者主要为：

- iOS 设备的用户和爱好者。
- iOS 应用程序业余开发者，使用 Objective-C 开发过简单 App。
- iOS 应用程序专业开发者，使用 Objective-C 开发过商业 App。
- 已经掌握苹果最新的 Swift 语言，正准备进行项目开发的程序员。
- 开设相关课程的大专院校学生。

## 如何阅读本书

在阅读本书之前，需要具备以下几方面的知识和硬件条件。

□ 面向对象的开发经验，熟悉类、实例、方法、封装、继承、重写等概念。

□ 有 Swift 或 Objective-C、C、C++ 的开发经验。

□ 有 MVC 设计模式开发经验。

□ 有简单的图像处理经验。

□ 一台 Intel 架构的 Mac 电脑（Macbook Pro、Macbook Air、Mac Pro 或 Mac Mini）。

如果加入了 iOS 开发者计划，还可以准备一台 iOS 移动设备。

本书通过大量的实例项目来讲解如何使用 Swift 开发简单的应用程序，虽然每个 App 所实现的功能都不复杂，但是都能帮读者了解每章重点讲授的知识点和技巧，只有“打通”每个点以后，一个完美的 App 才能流畅运行。如果你是一名初学者，请一定从第 1 章开始学习。

本书内容共 12 章，下面概述各章内容。

第 1 章介绍了 Swift 语言的特性、Playground 编辑器和 iOS 模拟器，重点介绍了开发 iOS 应用程序的集成开发环境 Xcode。

第 2 章和第 3 章通过一个简单计算器应用程序向大家介绍 Xcode 的用户界面搭建工具 Interface Builder、Outlet 与 Action 关联、MVC 设计模式、应用程序委托和视图控制器。

第 4 章通过购物应用程序向大家介绍如何使用故事板组织和管理视图。

第 5 章介绍表格视图的相关知识，包括与表格相关的委托协议，并且继续完善购物应用程序。

第 6 章介绍自动布局的相关知识，当程序员使用 Interface Builder 搭建 App 的用户界面时，往往要考虑不同分辨率和屏幕尺寸的设备，有时候一个场景需要做出 10 套左右的界面。但是通过自动布局可以让我们只需搭建好一套用户界面，就可以在所有的设备上完美运行。

第 7 章介绍集合视图的相关知识，通过在购物应用程序中使用集合视图来显示各种商品的缩略图。

第 8 章通过制作 IMDb 电影信息查询程序，向大家介绍如何使用 Swift 语言进行远程服务器调用，并将获取的 XML 数据进行整理并显示到屏幕上。

第 9 章使用 Photos.framework 框架实现在应用程序中获取照片库或摄像头所拍摄的照片。

第 10 章介绍如何在应用程序中整合 Facebook 和 Twitter 社交分享功能。

第 11 章介绍如何进行应用程序的调试。

第 12 章介绍如何在应用程序中进行文件和文件目录的管理。

## 勘误和支持

由于作者的水平有限，加之编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，特意留下联系的电子邮件 [liuming\\_cn@qq.com](mailto:liuming_cn@qq.com)。你可以就书中的错误和我进行沟通，当然，遇到任何技术问题也可以与我联系，我将尽力为你提供最满意的解答，期待能够得到你的真挚反馈。另外，书中的资源文件可以从华章网站（[www.hzbook.com](http://www.hzbook.com)）下载。

## 致谢

首先要感谢的是一直在背后默默支持我的老婆——刘颖。因为没有她就没有现在的我，非常感谢她在我写书的时候没有像平时那样唠唠叨叨，为我创造了一个轻松的环境。

感谢我身边的每一位充满创意和活力的朋友——张燕、卢红玲、秦琼、杨晓龙、陈雪峰、朱舸、赵顺利、吴永新，以及名单之外的更多朋友，感谢你们长期以来对我的支持和帮助。

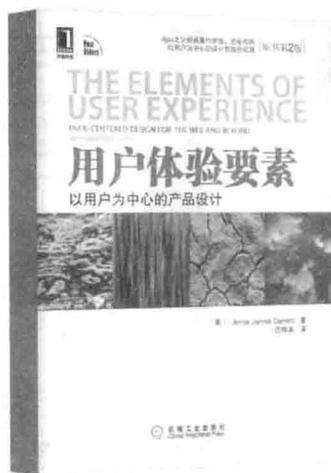
感谢机械工业出版社华章公司的编辑杨福川老师，在这三年多的时间中你始终支持我的写作，你的鼓励和帮助引导我能顺利完成全部书稿。

最后感谢我的爸爸、妈妈、岳父、岳母、小舅子和小舅子的女朋友，感谢你们时时刻刻为我灌输信心和力量！

谨以此书献给我最亲爱的儿子——乐乐，以及众多热爱苹果公司的朋友们！

刘 铭

## 推荐阅读



用户体验要素：以用户为中心的产品设计（原书第2版）

作者：Jesse James Garrett ISBN：978-7-111-34866-5 定价：39.00元



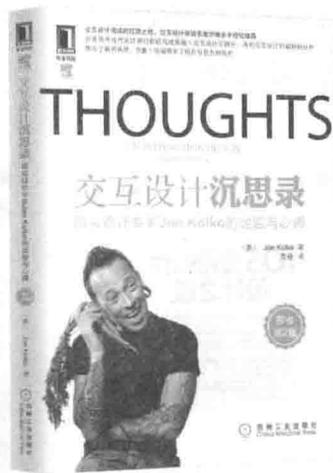
敏捷用户体验设计：用户体验设计应用敏捷方法的技巧与最佳实践

作者：Diana Brown ISBN：978-7-111-44721-4 定价：59.00元



优秀网站设计：打造有吸引力的网站（原书第3版）

作者：Patrick J. Lynch等 ISBN：978-7-111-39959-9 定价：69.00元



交互设计沉思录：顶尖设计专家Jon Kolko的经验与心得（原书第2版）

作者：Jon Kolko ISBN：978-7-111-39678-9 定价：49.00元

# iOS畅销经典著作



# Contents 目 录

## 前 言

<b>第 1 章 Swift 简介</b> .....	1
1.1 初识 Swift.....	2
1.2 了解 Playground.....	4
1.2.1 Playground 的编辑器模式.....	5
1.2.2 时间轴简介.....	6
1.2.3 Quick Look 所支持的类型.....	10
1.2.4 为什么要用 Playground.....	10
1.2.5 Playground 的一些限制.....	10
1.3 了解 Xcode.....	11
1.3.1 使用 Xcode 创建 iOS 项目.....	11
1.3.2 Xcode 的工具栏.....	14
1.3.3 导航区域.....	15
1.3.4 编辑区域.....	17
1.3.5 实用工具区域.....	18
1.4 使用代码编辑器.....	19
1.5 iOS 模拟器.....	22
1.5.1 iOS 模拟器的特性.....	22
1.5.2 模拟器中 iOS 系统的基本设置.....	24
1.5.3 在模拟器中安装和卸载应用程序.....	25
1.5.4 iOS 模拟器的限制.....	25

<b>第 2 章 搭建计算器的界面</b> .....	26
2.1 为移动平台开发应用.....	26
2.2 了解故事板.....	29
2.2.1 创建用户界面工具 Interface Builder.....	30
2.2.2 故事板.....	30
2.3 创建用户界面.....	36
2.3.1 设置界面的预览窗口.....	36
2.3.2 向视图添加界面元素.....	38
2.3.3 Interface Builder 的布局工具.....	39
2.4 理解 iOS 8 的视图和窗口.....	41
2.4.1 视图概述.....	41
2.4.2 UIWindow 类.....	41
2.4.3 视图的层次.....	41
2.4.4 视图的类型.....	43
2.5 与代码进行关联.....	43
2.5.1 要完成的效果.....	44
2.5.2 理解 Outlet 和 Actions.....	44
2.5.3 使用快速检查器查看关联.....	49
<b>第 3 章 设计模式和视图控制器</b> .....	51
3.1 MVC 设计模式简介.....	51
3.2 Calculator 项目中的控制器.....	54
3.2.1 实现计算器运算符的相关代码.....	54
3.2.2 实现计算结果的相关代码.....	56
3.3 关于 Application Delegate.....	57
3.4 了解视图控制器.....	59
3.4.1 视图控制器简介.....	59
3.4.2 不同类型的视图控制器.....	61
<b>第 4 章 使用故事板组织和管理视图</b> .....	64
4.1 创建购物应用程序.....	64
4.1.1 创建应用程序.....	64

4.1.2	创建 Shopping 的用户界面	66
4.1.3	创建数据模型	70
4.1.4	添加导航控制器	71
4.1.5	创建其他的视图控制器	73
4.1.6	在故事板中连接视图	77
4.2	调整数据模型	79
4.2.1	重建商品信息的数据模型	79
4.2.2	改变商品的购买状态	81
4.2.3	创建欲购买的商品	83
<b>第 5 章 自定义表格视图</b>		<b>86</b>
5.1	剖析表格视图	87
5.2	使用 UITableViewController 创建表格	89
5.2.1	创建超市的特价商品列表	89
5.2.2	创建特价商品的数据模型	91
5.2.3	通过 data source 传递数据	93
5.2.4	在 IB 中自定义单元格	94
5.3	表格视图中的选择与删除	99
5.3.1	删除表格中的单元格	99
5.3.2	单元格的选择和取消选择	101
5.4	委托	101
5.4.1	协议	102
5.4.2	协议方法	103
5.5	设置应用程序启动画面	103
5.5.1	直接设置启动画面	104
5.5.2	通过 LaunchScreen.xib 设置启动画面	105
<b>第 6 章 自动布局</b>		<b>107</b>
6.1	自动布局的概念	107
6.1.1	约束	109
6.1.2	约束的关系	109
6.1.3	创建约束	110

6.2	在 IB 中创建约束	110
6.2.1	为视图元素创建约束	111
6.2.2	通过预览查看实时效果	112
6.2.3	通过工具栏添加约束	113
6.2.4	改变约束的值	115
6.2.5	拖曳出约束	116
6.3	竖屏下的自动布局	117
6.3.1	对于约束的考虑	117
6.3.2	添加浏览特价商品详情的功能	118
6.3.3	为竖屏界面添加相关约束	120
6.3.4	显示相关商品信息	123
6.4	横屏下的完美布局	125
6.4.1	Size Class	125
6.4.2	使用 Size Class	126
<b>第 7 章 使用集合视图</b>		<b>130</b>
7.1	集合视图简介	130
7.2	使用集合视图显示数据	132
7.2.1	在故事板中添加新的场景	132
7.2.2	为集合视图提供数据	133
7.2.3	创建自定义的集合视图单元格	136
7.3	自定义集合视图的布局	138
7.4	标签栏控制器	139
7.4.1	标签栏及其中的标签	139
7.4.2	在故事板中添加标签栏控制器	140
7.4.3	设置标签栏配置条目	143
<b>第 8 章 获取远程数据</b>		<b>145</b>
8.1	使用故事板创建用户界面	145
8.2	使用 NSURLSession 获取数据	148
8.2.1	生成 URL 连接	150

8.2.2	使用异步方式解决等待问题	152
8.3	构建自定义 API 控制器	154
8.4	添加影片搜索功能	157
8.5	设计 IMDb 的用户界面	160
8.5.1	为场景添加虚化背景效果	161
8.5.2	通过类的扩展增加 String 的功能	163
8.5.3	自定义 label 字体和图像视图	165
<b>第 9 章</b>	<b>在程序中获取照片</b>	<b>168</b>
9.1	框架简介	169
9.2	搭建项目的用户界面	170
9.2.1	搭建用户基本界面	170
9.2.2	导航栏控制器的 Navigation Item	173
9.2.3	为 Photos Gallery 项目建立关联	174
9.3	完成 Photos Gallery 项目的逻辑代码	177
9.3.1	使用 PHAssetCollection 管理照片集	177
9.3.2	获取照片集中的照片	181
9.3.3	完善 PhotoViewController 控制器	184
9.3.4	完成 PhotoViewController 的交互	186
9.3.5	使用 UIImagePickerController 多方式获取照片	189
<b>第 10 章</b>	<b>使用 Facebook 和 Twitter 整合社交</b>	<b>192</b>
10.1	使用 Accounts Framework	192
10.1.1	访问 Twitter 账号和账号的属性	193
10.1.2	访问 Facebook 账号和账号的属性	199
10.2	使用 Social Framework 发布内容	203
10.2.1	创建 Stream 控制器	203
10.2.2	使用 Tweet Composer 视图发布消息到 Twitter	207
10.2.3	发送消息到 Facebook	209
10.3	使用 Social.framework 进行 API 调用	210
10.3.1	使用 SLRequest 检索 Twitter 流	210

10.3.2 获取 Facebook 信息	213
<b>第 11 章 调试你的应用程序</b>	<b>217</b>
11.1 为什么调试很重要	217
11.2 编译时候的问题	218
11.2.1 错误	219
11.2.2 警告	219
11.3 运行时候的问题	220
11.3.1 断点	220
11.3.2 使用调试器	221
11.4 帮助文档	222
11.5 与帮助文档有关的技巧	223
11.6 通过帮助文档了解应用程序图标	224
<b>第 12 章 文件和文件目录管理</b>	<b>227</b>
12.1 iOS 文件系统简介	227
12.2 在磁盘中搜索常用目录	229
12.3 读写文件的操作	231
12.3.1 将文件写入到目录中	231
12.3.2 读取文件内容	233
12.4 在磁盘上创建目录	234
12.5 遍历目录和文件	235
12.5.1 简单地遍历目录和文件	235
12.5.2 遍历并获取需要的信息	236
12.6 删除文件和目录	239

# Swift 简介

北京时间 2014 年 6 月 3 日凌晨 1 点，苹果公司（以下简称苹果）在美国旧金山 Moscone 中心举行了 WWDC 开发者大会开幕式，会上正式发布了 iOS 8 移动操作系统以及 OS X Yosemite 桌面操作系统。抛开 Yosemite 不谈，苹果在 iOS 8 中更新了很多内容，其中包括：

- ❑ 锁屏状态下直接快速删除通知。
- ❑ 支持发送语音信息。
- ❑ 输入法支持预判联想功能。
- ❑ 新增 Healthkit 健康应用。
- ❑ Siri 支持流媒体识别，可以直接通过它购买音乐。
- ❑ 自带相册集成更强大的图片处理软件。
- ❑ 大幅改善中国的地图体验。
- ❑ Spotlight 支持搜索音乐、电影、餐厅、App Store 中的应用。
- ❑ 可以接入第三方键盘，将是否授权输入法的选择留给用户。
- ❑ 邮件支持更多的手势操作。

如果说 iOS 8 中新增加的这些特性可以让广大用户激动不已，那么此次大会推出的全新程序设计语言——Swift（中文翻译为雨燕，爱称为小燕子）就可以让所有的 iOS 程序员发飙。为什么是发飙而不是疯狂呢？道理很简单，大部分的程序员可能会在之后的一段时间内放弃使用了多年的已经驾轻就熟的 Objective-C 语言，辛苦地学习一门新的程序设计语言。

另外，按照苹果的一贯风格，只要推出了一款新的产品来替代旧的产品，那么对应的旧产品就离退市不远了。有过 iOS 开发经历的“程序猿”都知道，当初苹果使用自动引用

计数器 (Automatic Reference Counting, ARC) 特性来代替手工管理内存, 以及使用自动布局 (Auto Layout) 来代替 Resizing Layout, 最终的结果都是这种情况。

那么 Swift 好学吗? 已经掌握了 Objective-C 的“程序猿”是否可以平稳地过渡到 Swift? 笔者现在还不能给出明确的答案, 相信在看完这本书以后, 你就能得到答案。

## 1.1 初识 Swift

Swift 是苹果在 WWDC 2014 所发布的一门编程语言, 用于开发 iOS 和 OS X 应用程序。

2010 年 7 月 LLVM 编译器的原作者暨苹果开发工具部门总监克里斯·拉特纳 (Chris Lattner, 就是在 WWDC 2014 大会上亲自演示 Swift 代码的那位仁兄) 开始着手开发 Swift 语言, 一直到 2014 年 6 月发布, Swift 大约经历了 4 年的开发期。在 WWDC 2014 大会中, 苹果宣称 Swift 的特点是: 快速、现代、安全和具有交互性。

Swift 的处理速度非常快。在 WWDC 上, 苹果展示了 Swift、Objective-C 以及 Python 的速度对比, Swift 比 Objective-C 快 1.4 倍, 比 Python 快 3.9 倍, 如图 1-1 所示。在进行 RC4 加密算法测试中, Swift 则是 Python 的 220 倍。但笔者认为苹果在这里使用了障眼法, 因为每门编程语言都有其优缺点, 如果非要用自己的长处与别人的短处相比, 明显有些小气了。

在笔者看来, Swift 就像是一门可以被编译的脚本语言。因为在很多语法特性上 Swift 和一些脚本确实非常相似。但是, 在应用程序开发中, Swift 不是以一门脚本语言来运行的, 所有的 Swift 代码都会被 LLVM 编译为本地代码, 然后以极高的效率运行。

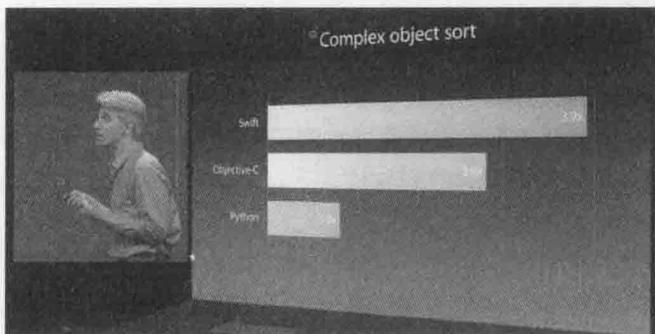


图 1-1 Swift 与 Objective-C 和 Python 的速度对比

Swift 和 Objective-C 都是类型安全的语言, 变量和方法都有明确的返回, 并且变量在使用前需要进行初始化。而在语法方面, Swift 迁移到了业界公认的非常先进的语法体系, 其中包含了闭包、多返回、泛型和大量的函数式编程的理念, 函数终于可以作为变量保存了<sup>①</sup>。初步看来, Swift 在语法上借鉴了 Ruby 的很多人性化设计, 但借助苹果自己手中强大

<sup>①</sup> Swift 语言上的特性不在本书的介绍范围之内, 你可以下载苹果官方的《The Swift Programming Language》一书, 或者在 CocoaChina 上面下载其中文版本。

的 LLVM 编译器，在性能上必然要甩开 Ruby 很远。

从另一方面说，Swift 的代码又是可以通过交互来“解释”执行的。Xcode 6 加入了所谓的 Playground 功能来对开发者输入的 Swift 代码进行交互式响应，当然，我们也可以使用 Swift 的命令行工具交互式地执行 Swift 语句。这里之所以把“解释”两个字打上双引号，是因为即使在命令行中，Swift 其实不是被解释执行的，而是在每条指令后都从开始的 Swift 代码全部进行编译，然后执行。这样的做法依然可以让人“感到”是在做交互解释执行，由此这门语言的编译速度和优化水平可见一斑。同时 Playground 还顺便记录了每条语句在执行时的各种情况，称做一组 Timeline。我们可以使用 Timeline 对代码执行逐步检查，省去断点调试的时间，也很方便，如图 1-2 所示。

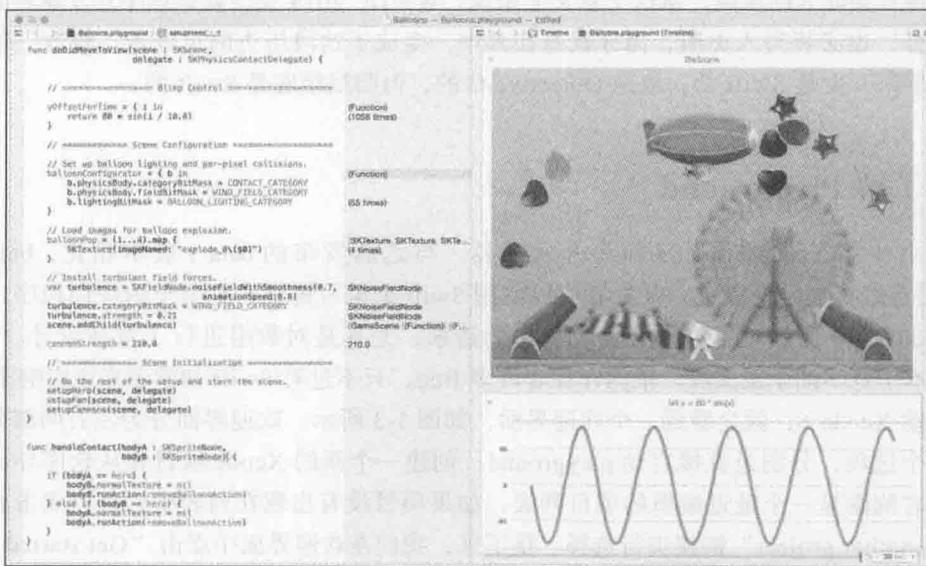


图 1-2 Swift 的 Playground 功能

不知大家是否有这样的想法：既然苹果的生态系统做得这样出色，为什么还要推出一门全新的语言，这不是自找麻烦吗？在 WWDC 2014 大会上，克雷格·费德里吉<sup>①</sup>（Craig Federighi）给出了这样的答案：苹果一直使用 Xcode 作为 Mac 和 iOS 平台的开发工具，而 Xcode 的核心是我们用来开发应用的 Objective-C 语言，它已经为我们服务了 20 多年，我们很喜欢它。但我们必须问自己一个问题，一个没有 C 的 Objective-C 会是什么样子？

苹果不仅仅想了，还将其付诸实践，因此出现了 Swift 语言，并且苹果希望使用 Swift 来主导该领域。苹果将 Swift 视为“亲生儿子”，并把它视为 Objective-C 的继承者。作为 iOS 或 Mac 的开发者，笔者深感学习和使用 Swift 的必要性。现在 Swift 可以和原来的 Objective-C 或 C 代码混用（不同于 Objective-C 和 C++ 或 C 在同一个 .mm 文件中的混编，

<sup>①</sup> 在苹果的 iOS 操作系统从拟物化转变为扁平化的过程中，克雷格·费德里吉起到了很大的作用。

Swift 文件不能和 Objective-C 代码写在同一个文件中，需要将两种代码分开)。编译出来的二进制文件是可以运行在 iOS 7 和 iOS 8 设备上（不支持 iOS 6 及之前的系统）。

现在 Xcode 6 中所有的文档都有 Objective-C 和 Swift 两种语言版本，并且按照苹果开发者社区的一贯跟进速度，有理由相信在不久的将来，苹果很可能会逐步废弃对 Objective-C 的支持，而全面支持 Swift。所以，关于到底是学 Swift 还是 Objective-C 的问题，笔者的建议是，尽快学习 Swift，尽快使用 Swift。在苹果无数工程师和语言设计天才的努力下，Swift 吸收了众多语言的精华，应该是当下最新也是最先进的一门编程语言之一。我想，也正是苹果对这门语言有这样的自信，才会在公司全盛的时候，不墨守成规，如此大胆地推出新的语言。因为苹果必定比你我都更明白，更换语言带来的利必须远大于弊，才会值得冒如此大的风险。从这个意义上来说，WWDC 2014 大会就是程序开发业界的一枚重磅炸弹，也必将写入史册，而你我身在其中，变成了这段历史的见证者。修改一位伟人的话：苹果开发是 Swift 的，也是 Objective-C 的，但归根结底是 Swift 的。

## 1.2 了解 Playground

本书使用 Xcode 6 beta 4 作为开发工具，与之前发布的 beta 1 版本相比，beta 4 对 Swift 语言做了大幅度改进。开发者可以使用 Swift 来编写更好、更安全的应用程序，而且新版本的 Swift 也修正了许多开发者提出的请求，尤其是对数组进行了重新设计。但是，beta 版本毕竟不同于正式版，依然存在着许多 Bug，只不过对 Swift 初学者来说影响不大。

启动 Xcode 6，就会看到一个欢迎界面，如图 1-3 所示。欢迎界面分为左右两部分，左侧有三个选项，分别是直接启动 playground、创建一个新的 Xcode 项目和从仓库导出一个项目。右侧则是一个最近编辑的项目列表，如果项目没有出现在列表中，可以点击底部的“Open another project”链接进行选择。接下来，我们在欢迎界面中点击“Get started with a playground”链接，启动 Playground。

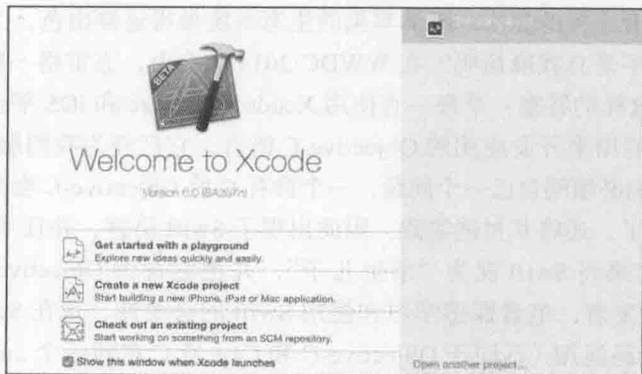


图 1-3 Xcode 6 的欢迎界面

## 1.2.1 Playground 的编辑器模式

Playground 是什么呢？它是 Xcode 6 中新引入的一种文档类型，在该文档类型的文件中封装了一些有用的东西，其中包括全部的 Swift 代码。Swift 代码会在 Playground 环境中实时运行，并且只要我们在 Playground 中编辑 Swift 代码，就会马上自动显示结果。Swift 文档也能包含一个文件夹，里面可以内嵌那些供代码使用的资源。另外，我们还可以在 Playground 文件中引用那些外部的、存储在系统中的可利用资源。最后，Playground 还可以包含一个时间轴（Timeline），时间轴可以在边栏中通过可视化方式显示结果，这个特性对开发者来说非常实用。下面分步讲解如何创建一个文件。

**步骤 1** 在 Xcode 的欢迎界面中点击“Get started with a playground”，在弹出的文件选项面板中设置 Name 为“HelloWorld”，设置 Platform 为“iOS”，如图 1-4 所示。

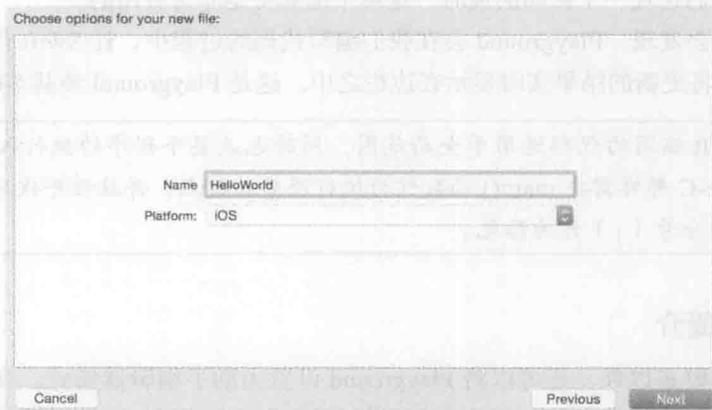


图 1-4 设置新建文件的选项

**步骤 2** 确定好文件的保存位置以后，会自动打开 Playground 编辑器，其中左侧是 Swift 代码，而在右侧的边栏中会显示运行的结果。将之前的 Swift 代码修改成下面这样：

```
// Playground - noun: a place where people can play
import UIKit
var string = "hello" + " " + "world"

for i in 0..<10 {
    string += "\(i)"
}
string

for i in 0..<20 {
    var j = i % 4
}
```

在上述代码中，首先定义了变量 string，然后通过加号将 3 个字符串连接并赋值给它。

变量 `string` 是 `String` 类型，当我们声明常量或变量的时候，通过加上类型标注来说明常量或变量中要存储的值的类型。方法是在常量或变量名后面加上一个冒号和空格，再加上类型名称。之前的声明语句也可以写成下面这样：

```
var string: String = "hello" + " " + "world"
```

**注意** 一般来说，我们很少需要写类型标注。如果在声明常量或者变量的时候赋了一个初始值，Swift 可以推断出这个常量或者变量的类型。在上面的例子中，因为为 `String` 赋了初始值，所以编译器推断其类型为 `String`。

接下来在两段循环代码中使用了半闭区间运算符 (`a..<b`)，它定义一个包含从 `a` 到 `b` 但不包含 `b` 的区间。如果需要定义一个包含从 `a` 到 `b` 的所有值的区间，可以使用闭区间运算符 (`a..b`)。当我们迭代一个区间的值时，这两个运算符是非常有用的。

细心的读者会发现，Playground 会在我们编写代码的过程中，让 Swift 代码从头到尾地反复运行，并且将更新的结果实时显示在边栏之中。这是 Playground 最基本的编辑器模式。

**说明** 使用 Swift 编写的代码适用于全局范围，同时也是整个程序的执行入口，不像 C 或 Objective-C 那样需要 `main()` 函数作为执行程序的起点，并且程序代码也不需要像它们那样用分号 (`;`) 作为结尾。

## 1.2.2 时间轴简介

除了编辑器模式以外，还可以将 Playground 设置为助手编辑器模式。在该模式下，开发者可以调出时间轴 (Timeline)，并通过时间轴以可视化的方式了解运行结果的更多细节信息。

**步骤 1** 在默认情况下，Playground 的工具栏是隐藏的，在菜单中选择“View → Show Toolbar”，然后点击工具栏右侧的“Show the Assistant editor”，如图 1-5 所示。

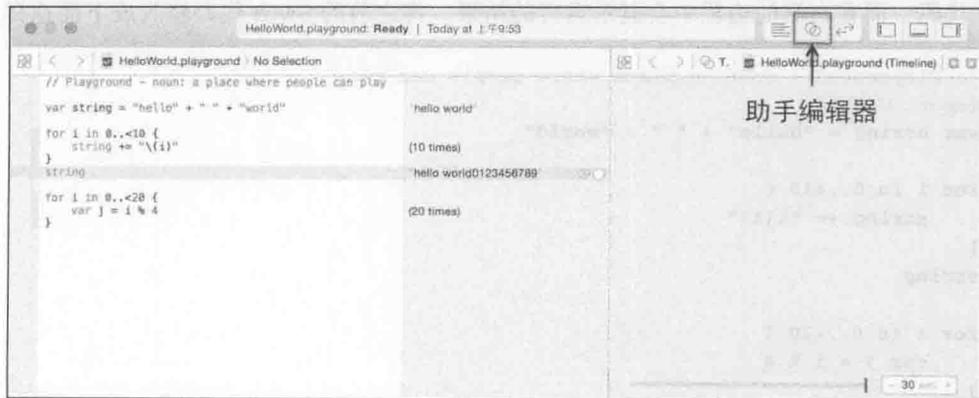


图 1-5 将 Playground 切换到助手编辑器模式

**步骤 2** 观察“`var j = i % 4`”代码行右边栏中的信息，“(20 times)”代表该循环体一共执行了 20 次。而这 20 次的结果是什么呢？点击该行右边栏中的“Value History”按钮  后，会出现与代码行对应的项目图表，如图 1-6 所示。在项目图表中所呈现的点代表每次该代码行被循环执行时所产生的值。此时的“Value History”按钮变成了 ，表明它的项目已经出现在时间轴中。

Playground 会根据不同类型的值显示不同类型的项目。如果是数值则会显示一个图表，在图表中的 x 轴表示执行的时间，y 轴记录代码行中的值，点击图表中的某个点，就会显示在该时间点的值。

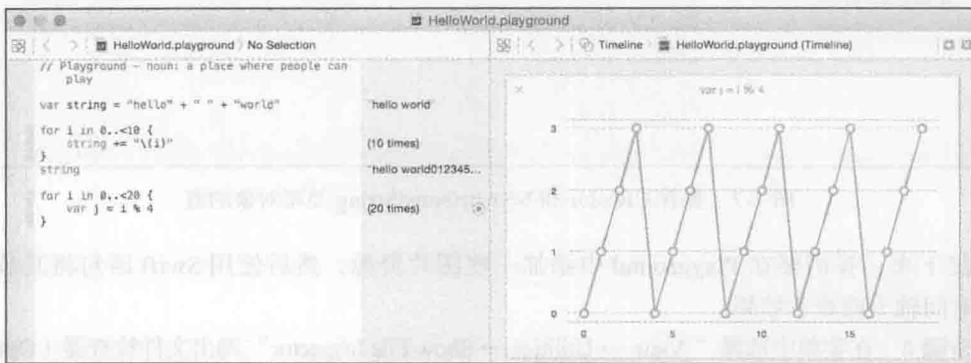


图 1-6 Playground 中的时间轴

除了数字类型，在时间轴中还可以显示字符串（string）、颜色（color）和图像（image）类型。当我们使用 `println()` 函数的时候，在时间轴上还会显示控制台输出项目。

当时间轴中有太多的 Value History 项目，想要收起（不是关闭）某个临时不用的项目时，点击该项目顶端的表达式  即可。

**步骤 3** 在当前 Swift 代码的结尾处添加下面的代码：

```
let color = UIColor.blueColor()

let attribStr = NSAttributedString(string: string, attributes:
    [NSForegroundColorAttributeName:color,
     NSFontAttributeName:UIFont.systemFont(ofSize(32))])
```

输入这几行代码以后，在结果边栏中会显示 color 的颜色和 attribStr 的值。你可以点击“Quick Look”按钮观看字符串的实际显示效果，也可以点击“Value History”按钮在时间轴中查看结果，如图 1-7 所示。

⊖ 因为笔者使用的是 beta 版本，发现点击数值图表中的表达式并不起作用，这时可以点击表达式与左侧 x 图标之间的空白处来解决。

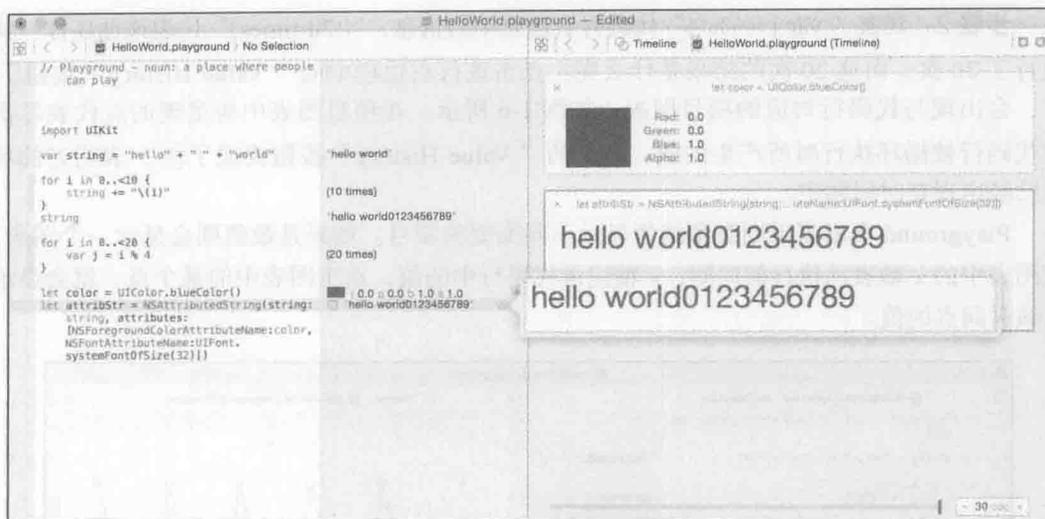


图 1-7 查看 UIColor 和 NSAttributedString 类型对象的值

接下来，我们要在 Playground 中添加一些图片资源，然后使用 Swift 语句将其载入，并在时间轴上面查看结果。

**步骤 4** 在菜单中选择“View → Utilities → Show File Inspector”调出文件检查器（Option+Command+I 快捷键），在“Resource Path”部分中，点击其中的按钮打开项目的资源文件夹，如图 1-8 所示，将素材文件夹中的 1-1.png、1-2.png 和 1-3.png 这 3 个图像文件拖曳到其中。

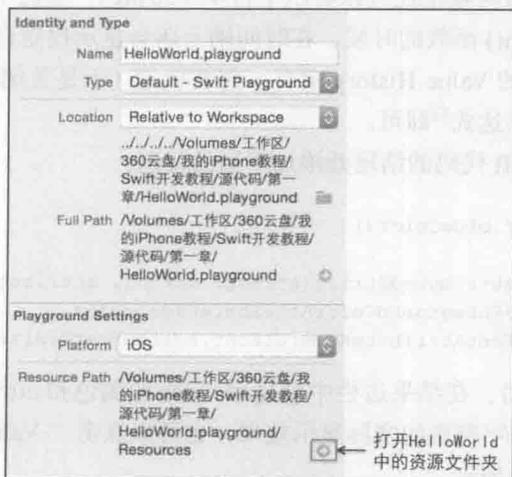


图 1-8 调出 File Inspector

**步骤 5** 在 Playground 中添加下面的代码：

```
let imageNames = ["1-1", "1-2", "1-3"]
```

```

let images = imageNames.map{ UIImage(named: $0) }
images
let image = images[0];
let imageView = UIImageView(frame: CGRectMake(0, 0, 512, 512))
imageView.image = image

```

上述代码首先定义了一个字符串数组 `imageNames`，数组中的字符串对象都是刚刚保存在 `Resources` 文件夹中的文件名（不带扩展名）。然后，通过数组的 `map` 方法依次遍历其中的所有元素（字符串对象），再通过这些字符串获取资源文件夹中的所有图像，最后将这些图像以数组的形式赋值给 `images`。

截止到目前，我们只能在边栏中看到 `map` 方法运行了 4 次，并不能判断是否成功载入了图像，所以接下来加入了一行 `images` 代码。此时，在边栏中会看到包含 3 个图像信息的数组，每个图像都是  $512 \times 512$  像素，并且点击“Quick Look”按钮还可以看到它们，如图 1-9 所示。另外，也可以通过“Value History”按钮将它们显示在时间轴上。

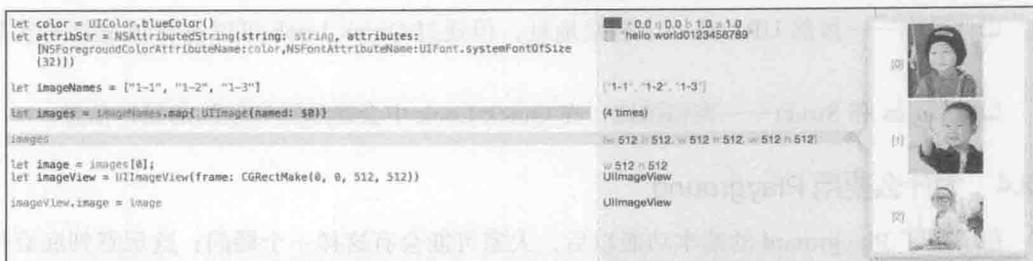


图 1-9 通过“Quick Look”查看载入的图像

在后面的代码中我们又创建了一个 `UIImageView` 对象，并将 `images` 数组中的第一个对象赋值给 `imageView` 的 `image` 属性。此时，点击“let imageView”行的“Quick Look”，可以发现它是一个空白的 `UIImageView` 对象。但点击“imageView.image”行的“Quick Look”，则会发现 `UIImageView` 对象已经载入了图像。



**提示** 在 Playground 中可以随时更新 `Resources` 中的资源，如更改现有的图像文件，添加音视频文件等。只要在更新完成后选择菜单中的“Editor → Execute Playground”即可。

除了在 Playground 项目中添资源以外，我们还可以利用资源的绝对路径来载入图像。

**步骤 6** 将素材文件夹中的 `1-1.png` 文件复制到用户的图片文件夹中。在 Playground 中添加下面的代码：

```

imageView.image = image

let absoluteImagePath = "/Users/liuming/Pictures/1-1.png"
let image2 = UIImage(contentsOfFile: absoluteImagePath)

```

在“Quick Look”中同样可以看到 `absoluteImagePath` 路径所定位的图像。`absoluteImagePath` 指定了一条绝对路径，其中的“`/Users/liuming/`”部分需要修改为你所指定的路径。

### 1.2.3 Quick Look 所支持的类型

利用 Quick Look 特性，我们可以在 Playground 中快速查看欲了解的值的的信息。那么 Quick Look 都支持哪些类型呢？它包括下面这些：

- ❑ 颜色——`UIColor` 类型的对象
- ❑ 字符串——包括无格式（`String`）和带属性的（`NSAttributedString`）
- ❑ 图像——`UIImage`
- ❑ 视图——各种视图对象，例如 `UISlider`、`UIButton`、`UILabel` 等
- ❑ `Array` 和 `Dictionary`——列表显示数组和字典对象
- ❑ `Points`、`rects` 和 `sizes`——点、矩形和大小的信息
- ❑ 贝赛尔曲线——显示所绘制的曲线
- ❑ `URLs`——虽然 `URL` 是一个链接地址，但通过 Quick Look 可以查看该链接的实际内容
- ❑ `Classes` 和 `Struct`——类和结构，在 Quick Look 中会显示类和结构的属性信息。

### 1.2.4 为什么要用 Playground

在了解了 Playground 的基本功能以后，大家可能会有这样一个疑问：这玩意到底有什么用？用它来开发一个项目那简直就是“扯淡”。它只能编写一些小打小闹的东西呀！

苹果发布 Swift 这门全新的语言是需要巨大勇气的，毕竟让上百万开发者从使用了 20 多年的 Objective-C 迁移到 Swift，就像下了一个天大的赌注。弄不好赔钱是小事（苹果不缺钱），丢了面子可就是大事了。因此，苹果在 Xcode 开发工具中独立出 Playground 链接，就是为了让程序员能够快速掌握 Swift，这里面包括：

- ❑ 通过 Playground 学习 Swift。
- ❑ 苹果在 iBooks Store 中推出了《*The Swift Programming Language*》一书，大家可以在 Playground 中边看书边学习。
- ❑ 方便初学者学习程序设计语言，让他们真正地去关注代码本身，只要在 Playground 编辑器中编写代码，就可以立即得到结果，从而省去了学习项目的配置、调试、构建和在模拟器或真机上运行的麻烦。

### 1.2.5 Playground 的一些限制

虽然 Playground 是初学者学习 Swift 的理想工具，但是它也有一些限制。最主要的一个限制就是它不能用于性能测试。Playground 主要用在编写代码的过程中实时显示运行结

果，它执行速度的快慢完全取决于所编写代码的行数，行数越多执行所花费的时间也就越多。除此以外，Playground 也不能实现下面的这些功能：

- ❑ 用户的交互——Playground 可以实时得到运行结果，但是不能有交互的操作。
- ❑ 授权——Playground 不支持任何形式的授权。
- ❑ 基于设备的执行——当开发 iOS 应用的时候，不能将 Playground 项目安装到 iOS 设备上执行。
- ❑ 自定义的框架——在 Playground 中我们只能使用 iOS SDK 所提供的基本框架库，无法使用自己开发的框架。如果非要使用，则需要将该框架复制到 Playground 项目的资源文件夹中。

## 1.3 了解 Xcode

作为一名开发者，不管在什么平台上进行开发，总需要一大堆的软件支持，才能将自己的想法变成可以让成千上万人受益的应用程序。苹果不希望这样，它为开发者提供了一个优雅的、功能强大的、光鲜亮丽的开发工具包，这个包就叫做 Xcode。通过 Xcode，我们可以创建、测试、部署和发布 iOS 或 OS X 应用程序。在 2014 年 6 月，苹果发布了 iOS 8 和 Xcode 6 的 beta 版本，在使用的过程中笔者能够感受到苹果力求让开发工具无比简单、实用。通过其强大的功能和全新技术，帮助开发者在创建应用程序的过程中体会到无限的乐趣和前景。

接下来使用 Xcode 创建一个简单的、具有交互功能的 iOS 应用“HelloWorld”。通过前面的学习我们知道 Playground 无法实现交互功能，所以下面我们要创建一个真正的 iOS 项目。

### 1.3.1 使用 Xcode 创建 iOS 项目

如果你之前在其他平台上开发过应用程序，可能会有一个疑问：在 iOS 和 OS X 开发领域中，为什么 Xcode 会成为一枝独秀？最主要的原因可能就是简单。化繁至简是苹果始终遵循的原则，而简单的背后却体现了其强大的内涵。Xcode 提供了在开发中你需要的所有东西：一个直观的代码编辑器、高级的调试器，集成界面编辑功能以及苹果不断更新和维护，这些使得它变成了真正的独一无二。

**步骤 1** 点击欢迎画面中的“Create a new Xcode project”或者选择菜单“File → New → Project”。此时会出现项目模板选择面板，在面板的左侧选择开发项目的目标平台。利用 Xcode 不仅可以开发 iOS 项目，还可以开发 OS X 应用程序。

**步骤 2** 在确定 iOS 中的 Application 类别后，在右侧的主面板中会列出几种基本的项目模板。选择“Single View Application”模板，如图 1-10 所示。该模板只提供一个用于显示的视图和一个管理这个视图的控制器，而且视图一般会存储在故事板或 nib 文件中。除此

以外，该模板还包含了一个应用程序委托类型（`UIApplicationDelegate`）的对象，它用于响应一些系统事件，比如应用程序的启动、进入到后台和程序终止退出等。

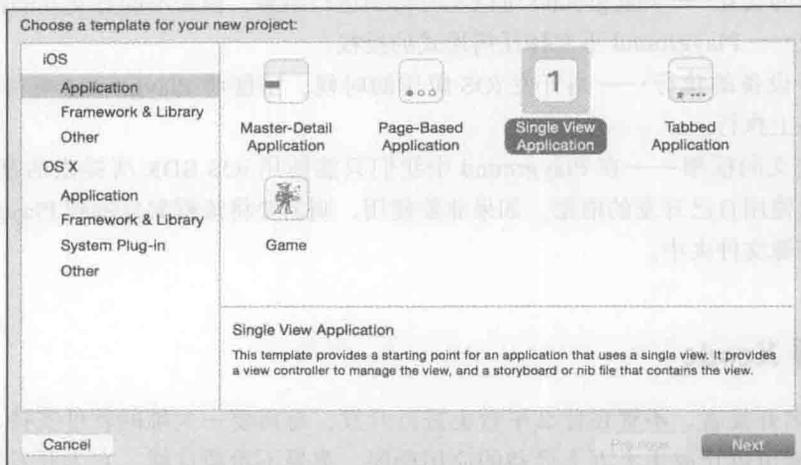


图 1-10 选择项目模板

在这里我们对 Application 中所列出的项目模板进行简单介绍。

- ❑ **Master-Detail Application**：创建一个基于列表的应用程序。当我们从主列表中选择一个条目后，就会在另一个视图中显示相应的详细内容。此模板还会提供一个返回按钮让用户可以从详细视图返回到主列表。在苹果内置的邮件应用和很多的新闻类应用中我们可以看到该技术，当用户点击列表中的某条新闻后，就可以看到详细的新闻内容。当该模板用在 iPad 项目中的时候，会形成分离的视图效果。
- ❑ **Page-based Application**：使用页面视图控制器（`page view controller`）创建的项目模板，用户可以在屏幕上进行翻页。
- ❑ **Single View Application**：创建一个基本的应用程序模板，它包含一个单独的视图和相应的视图控制器。
- ❑ **Tabbed Application**：创建一个有标签栏的应用程序。在一般情况下，标签栏会包含几个不同的类别并出现在屏幕的下方。当用户进行选择时，就会显示不同的视图。iPhone 中的电话应用程序就使用了该技术，使用它的用户可以在个人收藏、最近通话、通讯录、拨号键盘和语音留言之间切换。
- ❑ **Game**：如果你想开发一款游戏项目，就可以使用该模板。在之后的配置选项中，还可以选择使用 `SceneKit`、`SpriteKit`、`OpenGL ES` 或 `Metal` 技术。

**步骤 3** 点击“Next”按钮后，需要指定以下几个配置选项。

- ❑ **Product Name**：应用程序的名称。例如，我们想创建一个交互应用，就可以指定 `Product Name` 为“HelloWorld”。按照“潜规则”，在定义 `Product Name` 的时候，可以忽略单词间的空格，而每个单词的首字母要大写（不是必需的）。`Product Name`

在项目开发的过程中是可以修改的，但同时它也是一个非常重要的细节，所以名字和功能要尽量保持一致。

- ❑ **Organization Name**：不管你是一个独立开发者还是软件公司的某个开发部门，你需要指定一个组织机构或单位的名称。这里，我们可以随意起一个名字。但是如果你想要将应用程序提交到 App Store，就需要填写正确的信息内容，虽然不是必需，但是笔者强烈推荐填写。此外，每当我们在项目中新建一个文件，组织名称都会出现在文件顶部的版权详细信息中。
- ❑ **Organization Identifier**：组织标识，如果我们计划将应用程序发布到 App Store，这个设置非常重要。例如，将应用程序发布到 App Store，就需要指定一个 App ID 和 Bundle Identifier。其中 Bundle Identifier 就依赖于这个 Organization Identifier。Organization Identifier 的格式为一个反向域名，这里我们将其设置为 `cn.project`。
- ❑ **Bundle Identifier**：我们不能编辑此选项，它完全依赖于 Organization Identifier 和 Product Name。
- ❑ **Language**：这是在 Xcode 6 中的一个新选择，因为苹果推出了一个全新的编程语言 Swift，所以在项目中我们可以选择使用 Objective-C 还是 Swift。在 HelloWorld 项目中设置为 Swift。
- ❑ **Devices**：选择 iPhone，代表应用程序项目是专门为 iPhone（包含 iPod Touch）设备开发的。如果选择 Universal，则代表该项目可以同时支持 iPhone 和 iPad 设备。
- ❑ **Use Core Data**：该复选框指定项目是否支持 Core Data。如果勾选 Core Data，则可以在项目中使用数据库特性。

设置完成的配置选项面板如图 1-11 所示，确认无误后点击“Next”按钮。

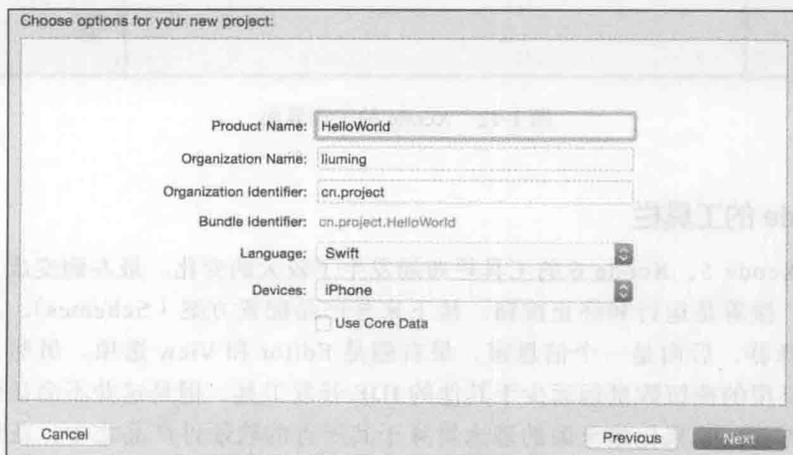


图 1-11 在项目选项面板中设置选项

**步骤 4** 接下来会询问保存项目的本地位置，可以将它保存到任意的位上。

**步骤 5** 注意到在项目保存面板中的底部有一个“Create git repository on”选项，当我们希望创建一个本地仓库的时候可以勾选此项，在默认情况下它是未勾选的。如果我们是以团队形式进行多人项目开发，推荐勾选此项。

**步骤 6** 最后点击“Create”按钮创建 HelloWorld 项目。

在创建好项目以后，就会看到 Xcode 的工作界面。在界面的最上方是工具栏，其下方分别包含导航区域、编辑区域、调试区域和实用工具区域四部分，如图 1-12 所示。其实 Xcode 工作界面的每个区域还有更细致的划分。接下来，我们将会针对每个区域进行详细介绍。

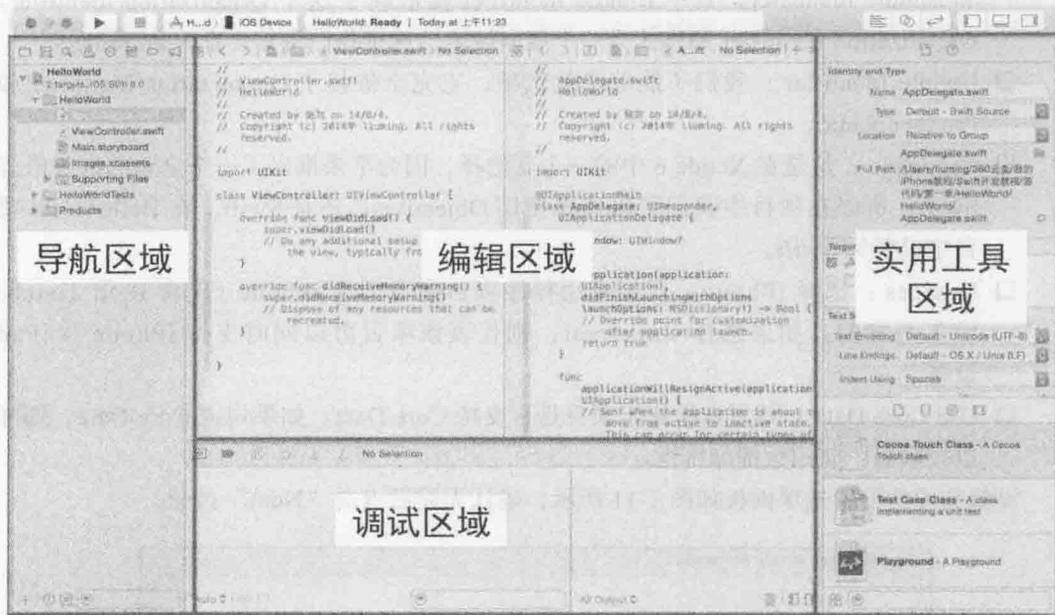


图 1-12 Xcode 的工作界面

### 1.3.2 Xcode 的工具栏

相比于 Xcode 5，Xcode 6 的工具栏布局发生了较大的变化。最左侧变成了操作窗口的三个按钮，接着是运行和终止按钮。接下来是产品配置方案（Schemes），其中包含了构建配置选择器，后面是一个信息窗，最右侧是 Editor 和 View 选项。虽然 Xcode 6 整个工具栏所呈现的按钮数量远远少于其他的 IDE 开发工具，但是这并不会让开发者在使用时感觉不方便。苹果化繁至简的理念贯穿于其所有的软硬件产品之中，花费的心思可见一斑。

### 1.3.3 导航区域

Xcode 的导航区域位于工作界面的左侧，整个区域共包含 8 种不同的导航模式，如图 1-13 所示。除了可以点击相应按钮切换不同导航模式以外，还可以通过 Command+1 至 Command+8 快捷键或在菜单中选择“View → Navigators”进行切换。

所有的导航器都具有过滤和范围限定功能，该功能大多位于导航区域的底部（只有搜索导航位于顶部，导航切换按钮的下面）。我们可以通过 Command+Option+J 快捷键访问它。在按下该快捷键以后，编辑光标就会出现在过滤框之中，而且该快捷键适用于所有 8 种导航方式。



图 1-13 Xcode 导航区域中的 8 种导航方式

#### 1. 项目导航器

项目导航器 (Project Navigator) 会显示项目的所有内容，通过树形结构，将一些相关的文件组织到类似于文件夹的组之中。从名称就可以猜到，项目导航可以帮助我们快速定位源代码文件、框架文件和项目目标，如图 1-14 所示。

若要在项目中创建一个新组 (Group)，我们可以在项目导航中点击鼠标右键选择一个现存的节点，然后在快捷菜单中选择“New Group”。也可以通过拖曳方式将文件移出或移入某个组，但需要说明的是，在项目导航中移动文件并不会改变本地磁盘中文件的物理存储位置。

要想删除一个文件，我们只要在该文件上按 Delete 键即可。

此时，Xcode 会询问要删除本地磁盘中的物理文件，还是只删除在项目导航中对文件的引用。删除组的操作与文件相同，只不过删除一个组时，会删除组中的所有文件。

虽然我们可以在项目导航中随意创建任意数量的组，但是有三个组是在应用程序创建的时候就已经存在的。第一个是项目组，它与我们创建的项目名称相同，其中包含了所有的代码文件和资源。第二个是测试 (Tests) 组，其中包含了项目的测试代码。第三个是产品 (Products) 组，其中包含了真正的 iOS 应用程序。

在项目导航器的最下方有一组图标，通过这组图标我们可以过滤显示在项目导航中的内容，如图 1-15 所示。其中，通过第一个增加新文件的按钮，我们不仅可以为项目创建一个新的文件，还可以将项目位置以外的文件添加到当前项目中。

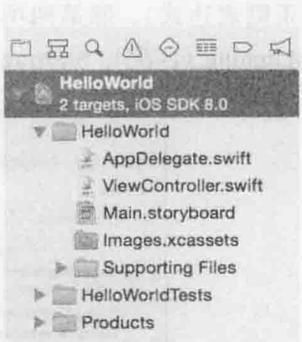


图 1-14 项目导航器中的树形结构列表

#### 2. 符号导航器

通过 Command+2 快捷键可以跳转到符号导航器

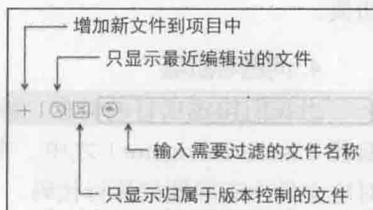


图 1-15 项目导航器中最下方的图标按钮功能

(Symbol Navigator)。符号导航器可以帮助我们快捷地定位项目中的某个类、方法或属性。因为从 Xcode 4.5 以后，LLVM 编译器使用了 Clang 作为前端，所以在符号导航中浏览类、方法和属性的速度要快很多。

### 3. 搜索导航器

除了可以使用 Command+3 快捷键切换到搜索导航器 (Search Navigator) 以外，我们还可以使用 Command+Shift+F 快捷键快速进入搜索功能。使用搜索导航器可以帮助我们在指定的范围内搜索指定的内容，搜索结果将会显示在其下方的区域中，如图 1-16 所示。

点击搜索框上方的 Find 字段，之后可以在 Find 和 Replace 之间切换。以 Find 模式为例，其中包含了四种不同的搜索范围：Text (项目中的所有文本)、References (项目中所有的引用)、Definitions (项目中所有自定义的，不包含从外部引用的)、Regular Expression (正则表达式)。除第四项以外，它们又都包含四种不同的搜索方式：Containing (包含)、Matching (匹配)、Starting with (从开头搜索)、Ending with (从结尾搜索)，如图 1-17 所示。

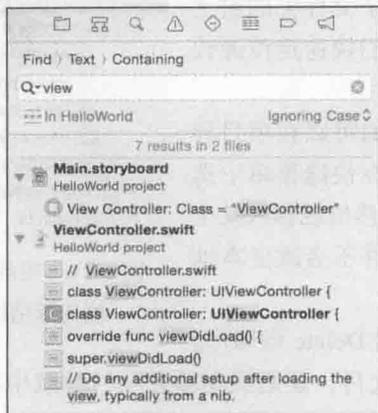


图 1-16 搜索导航器的工作界面



图 1-17 限定搜索方式

另外，在搜索框的下方左右两侧各有一个字段可以进行切换。当点击“In Project”以后，导航部分会被切换到范围选择界面，我们可以在这里确定在项目中搜索或替换的范围。当点击右侧的“Ignoring Case”后，可以在忽略大小写和匹配大小写之间进行切换。

### 4. 问题导航器

当我们构建项目的时候，编译器所产生的警告、错误消息和分析警告会显示在问题导航器 (Issue Navigator) 之中。在列表中选择某个警告或错误后，代码编辑器就会快速定位对应文件中有问题的那行代码。

### 5. 测试导航器

使用 Command+5 快捷键可以切换到测试导航器 (Test Navigator)。单元测试对软件开

发平台来说是非常重要的一个方面。毕竟，单元测试可以帮助我们修复程序中的 Bug 以及找到运行时意外崩溃的原因。如果提交到 App Store 中的应用程序在运行时总是出现崩溃的情况，那么苹果在审核时会毫不犹豫地将其拒绝。

## 6. 调试导航器

使用 Command+6 快捷键可以切换到调试导航器 (Debug Navigator)。它除了可以监控 CPU 和内存的使用情况以外，还可以监控磁盘的读写情况和应用程序在网络方面的接收发送情况。更重要的是，我们可以通过它来监控应用程序每个线程的队列执行情况。使用调试导航器可以帮助我们了解线程中对象、函数的前后调用关系。

## 7. 断点导航器

第 7 个导航器是断点导航器 (Breakpoint Navigator)，我们可以通过它管理在项目中所设置的断点。在点击导航器中的断点以后，编辑器会快速定位到该位置。同时也可以在这里禁止或删除某个断点。

值得注意的是，我们可以在断点导航器中共享断点用于其他的项目。在列表中选择某个断点后点击鼠标右键，选择“Share Breakpoints”即可。

## 8. 日志导航器

当我们在日志导航器 (Log Navigator) 中选择一个构建条目的时候，它的结果会显示在编辑区域之中。如果双击其中的警告和错误，还可以直接在代码编辑器中将其打开。当我们点击日志最后面的命令行图标时，还可以看到更详细的信息日志，帮助我们查找问题和错误，如图 1-18 所示。

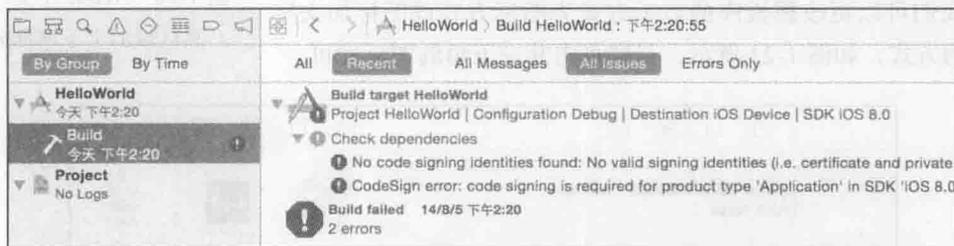


图 1-18 日志导航器中条目的详细信息

### 1.3.4 编辑区域

编辑区域的功能相信大家很清楚，它用于编辑项目中各种类型的文件。Xcode 6 提供了 3 种不同类型的编辑模式。我们可以通过工具栏中的 Editor 选择器进行切换，如图 1-19 所示。除了标准编辑器 (Standard editor) 以外，另外两个分别是助手编辑器 (Assistant editor) 和版本编辑器 (Versions editor)。



图 1-19 工具栏中的 Editor 选择器

在选择助手编辑器以后，编辑区域会被分割为左右两个编辑窗口。我们经常会使用助手编辑器从 Interface Builder（搭设用户界面的工具）增加 IBAction 或 IBOutlet 声明到代码文件之中并建立相应的关联。

Xcode 的版本编辑器可以帮助我们轻松对比两个不同版本的源代码。如果项目使用的是 Git 或 SVN，我们就可以通过编辑器对比当前正在编辑的文件和仓库中保存的之前版本代码的不同。

### 1.3.5 实用工具区域

实用工具区域位于整个工作界面的右侧，它分为上下两个部分。上面的部分是检视窗，下面则叫做库，如图 1-20 所示。

检视窗可以帮助我们对编辑器里选中的对象进行属性、行为等方面的设置，如控件的大小、位置、背景色、字体的样式和字号的大小等。

在实用工具区域的下半部中又分为文件模板库、代码片段库、对象库和媒体库四个部分，我们可以使用 Control+Option+Command+(1 ~ 4) 快捷键进行切换。

#### 1. 文件模板库

文件模板库（File Templates Library）提供了我们在开发中经常会用到的一些文件模板。若要使用它们，直接用鼠标将其拖曳到项目导航中相应的位置即可。

我们可以更改模板库的显示方式为图标方式或图标加文字说明的方式，如图 1-21 所示，只要点击其左下角的图标即可。



图 1-20 Xcode 中实用工具区域的上下两部分

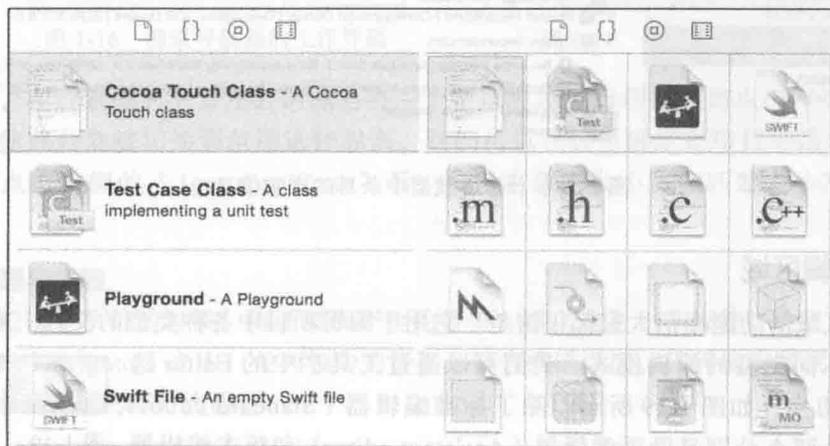


图 1-21 库区域中的文件模板库

---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**