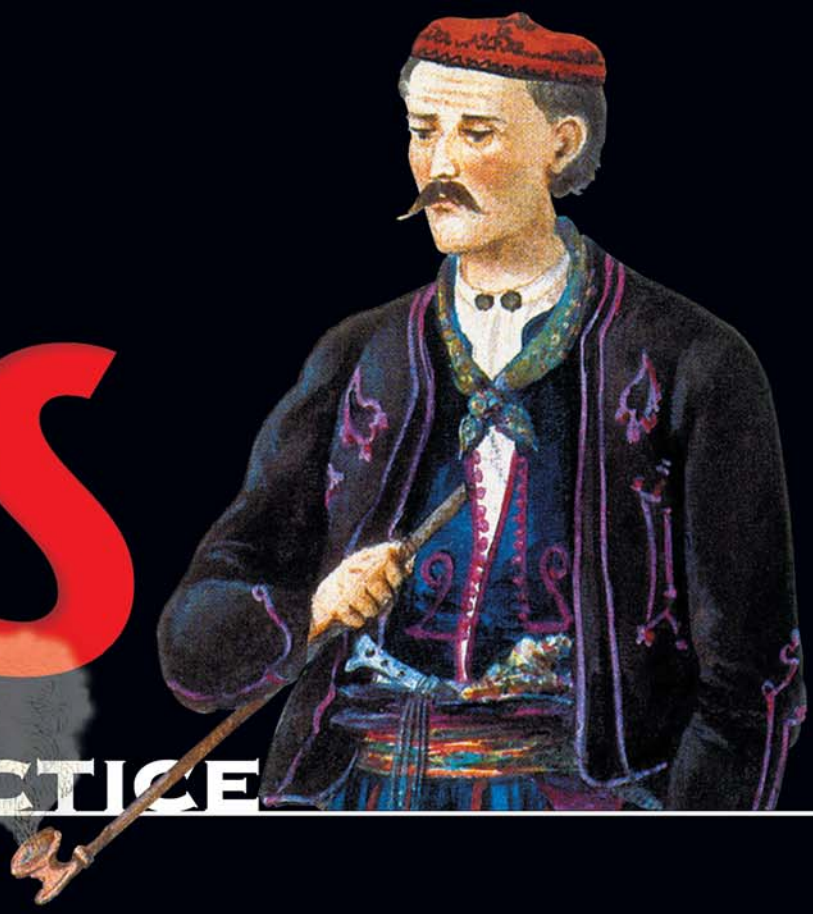
Bear Cahill

Covers iOS 6

iOS in Practice



Includes 98 Techniques

MANNING

目　录

1 ——开始 iOS 开发

本章包含：

- Xcode 和 Objective-C
- 了解 Xcode
- Hello World 的例子

我专职从事开发已经 20 年了，几乎学习了所有语言和平台，但是我相信 iOS 开发是所有开发中最令人激动，有趣，回报高，极具挑战性的。我爱 iOS 开发。

从开发者的角度来说他不但是最吸引人的，而且它同时也是领先的手机平台。这就意味着有很多可以做的，成长很多，变化很多，还可以从苹果，论坛还有其他开发者，书籍，会议等等方面得到很多的支持。

随着 iOS 系统和其他手机平台，平板的成长，这些很好的连接了传统电脑和只能手机的平台，现在都成为了巨大的市场。这些手机设备给开发者带来了更多的机会，iOS 让你能够同时为两个平台做开发。

在本章中，我们会开发一个 iOS 应用。我们需要一起来了解一些话题，包括搭建开发环境，在本章的最后，你会有你的第一个应用。让我们开始吧。

iOS 开发环境

Xcode 是开发 iOS 以及 OS X 应用的主要工具。苹果免费提供给开发者的，而且还提供了很多开发相关的工具，包括 UI 设计开发，版本控制等。

iOS 开发的主要语言叫做 Objective-C。Objective-C 是 C 的衍生语言，这就意味着所有的 C 代码都能够兼容，在 Ojective-C 中运行。但是，和 C 语言不同，Objective-C 是面向对象的。如果你了解 C++，Java 或者其他面向对象的编程语言，那么理解 Objective-C 就不会有问题。记住这本书的目的不是教你学习 Objective-C，如果你对这门语言的学习有问题的话，你可以使用其他的资源来研究 Objective-C。

苹果同时提供了很丰富的各种框架。有些框架开发所有的应用都需要使用，所以自动包含在了 Xcode 中。根据你自己的偏好，其他的框架可以选择性的添加到工程中。iOS 刚出来的时候，显示一个地点的地图非常难，需要很大的工作量。给地图增加锚点更加复杂。后来 MapKit 发布了，增加了一个地图，显示用户的位置变得基本毫无困难。

把 WebKit，StoreKit，MediaPlayer，Social，CoreData 这些框架添加到工程中很方便，功能也很强大。很多开源的第三方框架可以省去你很多的时间帮助完成复杂的功能。

iOS 开发很大程度上以来 Model-View-Controller（MVC）构架模式。MVC 把开发分成了三方面：模型（model），视图（view）以及控制器（controller）。Model 是数据层（比如说，工程中的数据库）。View 是和用户互动的 UI。Controller 是 view 和 model 的中间部分，它负责把用户行为翻译成逻辑，访问数据。

正如你所见，Xcode 做了很多来帮助你作为开发者需要做的工作，同时还让你可以用最适合 iOS 工程的方式去做。让我们来看看如何获取，安装，并熟悉 Xcode 吧，然后让我们来开发你的第一个应用。

1.2 使用 Xcode

正如上一节所说，Xcode 是 iOS 工程开发的主要工具。在这一节，我们会了解如何从苹果获取 Xcode，然后了解一下 Xcode 的各部分的内容方便以后的开发工作。

1.2.1 获取 Xcode

在 app Store 中搜索 Xcode，可以快速找到它。Xcode 是免费的，所以直接点击 FREE 按钮开始安装（见图 1.1）。由于文件比较大，下载需要花些时间，但是下载过程非常简单。Xcode 以及相关的应用可以在/Developer/applications 下安装，关键的应用会被添加到 Launchpad 中的 Developer 文件夹中。

图 1.1 app Store 中的 Xcode

你也可以到 http://developer.apple.com 下载 Xcode，但是这需要更多的操作。在网站上可以看到一些参加开发者项目的信息，比如说 Safari，iOS，以及 Mac 项目。

大部分情况下开发者项目都需要花钱才能加入，但是加入的话也可以获取到高级/测试的 iOS 固件或者开发工具，开发者论坛以及其他资源。如果你希望深入 iOS 开发的话，我强烈建议你加入。如果你想要发布任何应用的话，你必须加入。

现在你安装了 Xcode，让我们来看看它的组成部分吧。

1.2.2Xcode 之旅

Xcode 可以处理 iOS 工程开发中的所有主要方面开发。它可以管理代码的组织，链接框架，UI 设计，编辑，工程（比如说常规以及专业版本的给不同应用使用的相同的 code base），创建，测试以及提交应用给苹果审核。在本章中，我们会了解 Xcode 的基础部分。在接下来的章节中，我们会深入 Xcode 不同部分的细节问题。

Xcode 可以帮助我们这么多，那他拥有这么多的区域，面板，view 等内容就说得通了。左边的 Navigator 显示 了各种文件，框架，工程以及在你的工程中包含的项目(见图 1.2)。它可以让你选择文件进行编辑或者管理。

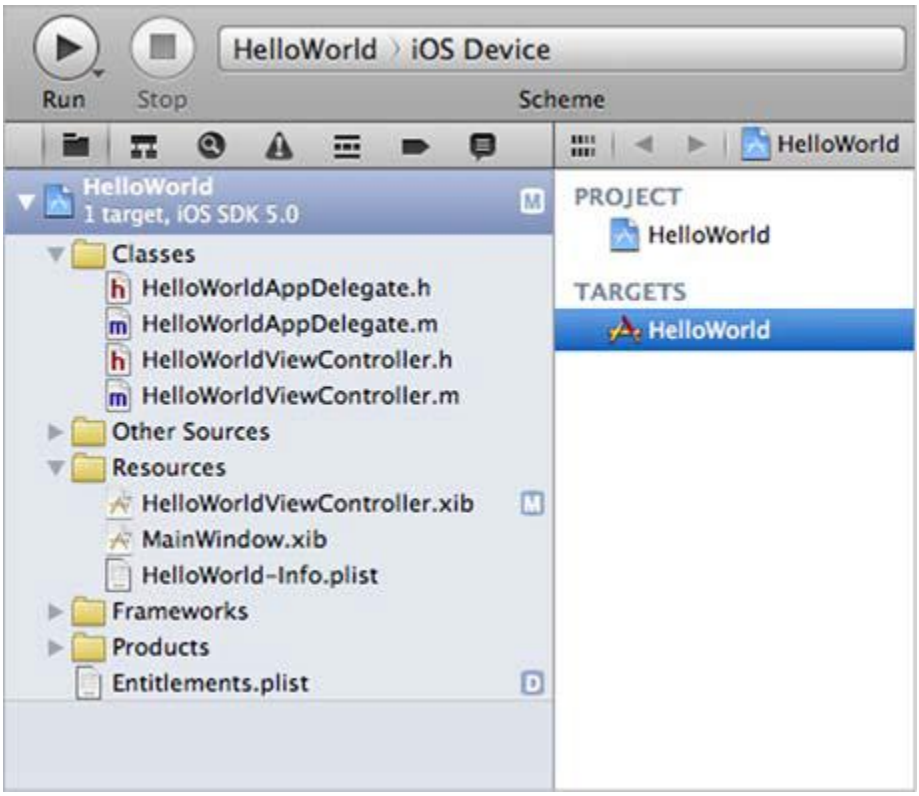


图 1.2 Xcode 中的 Navigator

Utilities 区域，点击右上方的的 View 按钮的右边按钮显示，它显示了选择项目（比如说一个文件）的各个方面以及设置（见图 1.3）。这儿你可以看到一个给定的项目是如何和其他项目相关联的，设置各种属性以及其他。当使用 Interface Builder（IB）编辑器来设置可视化项目的属性时特别有用。

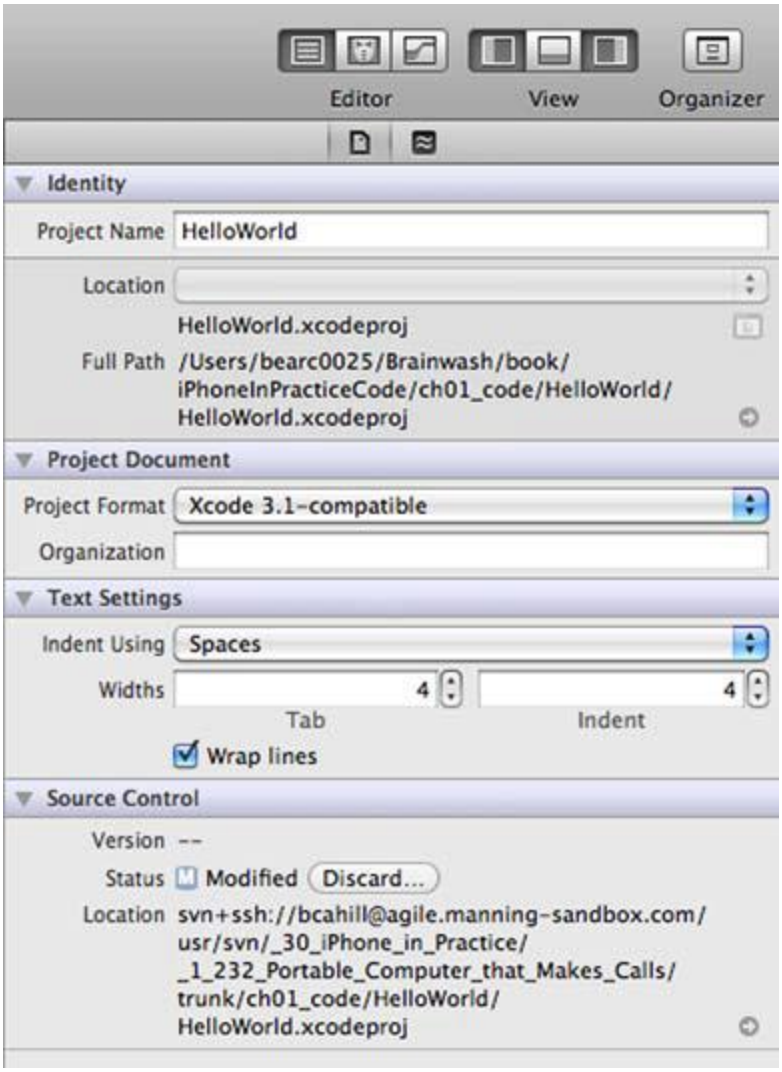


图 1.3 选择 HWViewController.m 文件之后，Utilities 区域的显示情况

由于所有的开发都需要一种方式来编辑代码，IB 编辑器可能是 Xcode 中看上去最熟悉的项目了（见图 1.4）。但是 IB 编辑器不仅仅可以用来编辑代码，还可以用来编辑 UI 和数据（比如说 CoreData 的数据库设计），全书的项目都会涉及到这个部分。

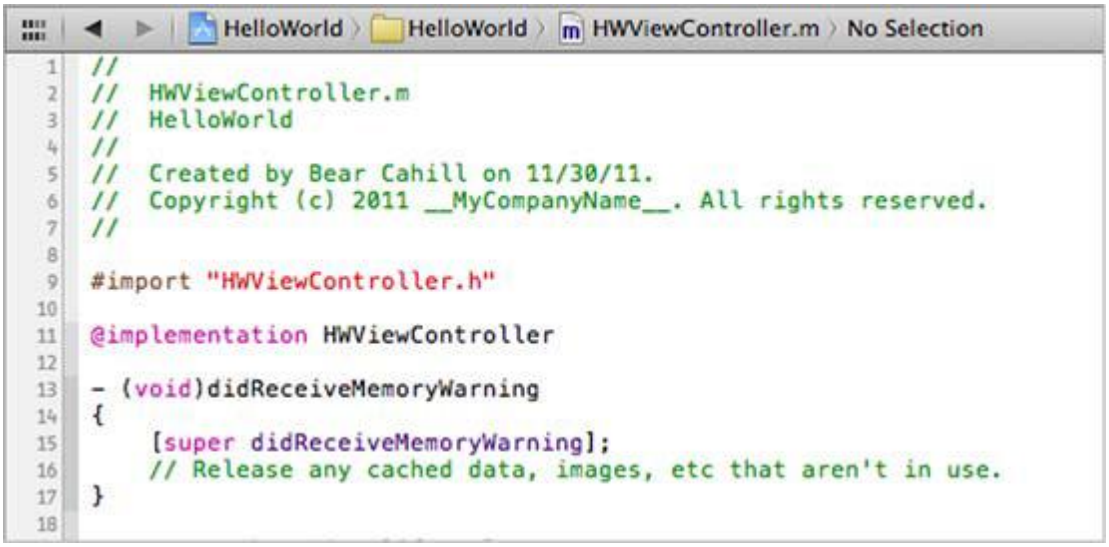


图 1.4 选择 HWViewController.m 文件后，编辑器的显示情况

Debug 区域在底部显示，这个区域可以分成两部分，右边可以显示 Console，查看 standard output（见图 1.5）。这两部分对在测试期间显示各种 value 和 output 非常有帮助 。

图 1.5Hello World 执行期间，Debug view 和 console 的显示情况

Toolbar 位于窗口的顶部，可以显示各种求，开始/停止测试，为创建工程选择 scheme（见图 1.6）。

图 1.6 顶部的 Toolbar

Organizer，在窗口 memu 部分显示，开发的很多方面都需要用到它。它可以显示框架还有其他有帮助的文件，方便提交你的二进制文件给 appStore 审核，组织各种设备等等其他功能（见图 1.7）。它可以帮助你记录 Provisioning 文件，也可以帮助获取设备上的崩溃报告（这不是说你的应用会崩溃，只是其他人需要这个）。

图 1.7 Organizer 显示框架文件

Organizer 可以通过使用 commard 键+点击代码中的文本，可以组织和和文本相关的文件。同时，Organizer 可以让你访问有用的文件比如说“Apple Human Interface Guidelines《苹果人机交互指南》”以及“Learning Objective-C:A Primer“《学习 Objective-C，入门指导》”。这两个文件都推荐你阅读。

现在已经大概了解了 Xcode 和它的开发环境，让我们来创建一个应用吧！

1.3Hello World 应用

作为探索 Xcode 以及了解 iOS 开发的一种方式，让我们先来创建一个基础的应用。这不需要花费很多工作，但是可以帮助你了解创建应用的基础。

首先，你可以创建一个新的工程，创建工程包括几个关键的步骤。然后你可以给应用创建 UI，然后运行应用。

1.3.1 创建一个新的工程

打开 Xcode，选择 Create a New Project（见图 1.8）。

图 1.8 使用 Xcode 创建一个新工程

然后可以选择工程的模板。请确定在顶部左边的 iOS 栏下选择了 Application。在右边会显示一些合适的选择。选择 Single View Application（见图 1.9），然后点击右下角的 Next。

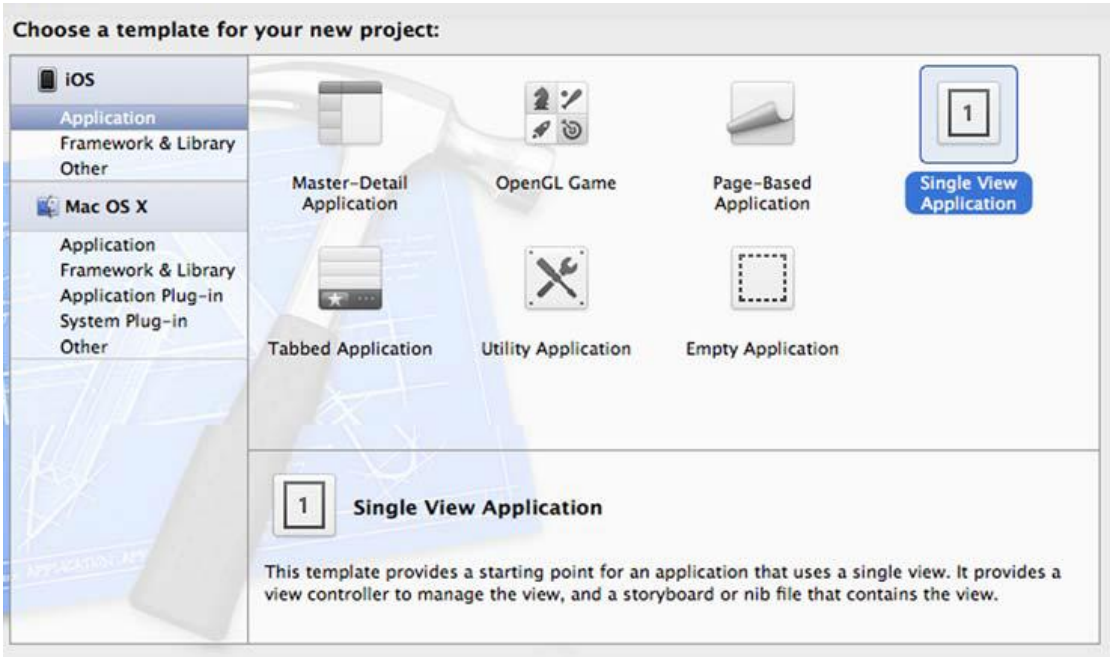


图 1.9 新工程的 Single View Application 模板

然后会有一个弹出框，让你输入产品的名称，设置 Company identifier，这通常是一个反向的 DNS 值。你还需要制定一个类前缀（class prefix）（这是为命名约定服务的），然后指定设备类型（比如说 iPhone）。最后，为了 UI 设计选择 Storyboard 选项，为了以后内存管理选择引用计数，最后再选择单元测试。

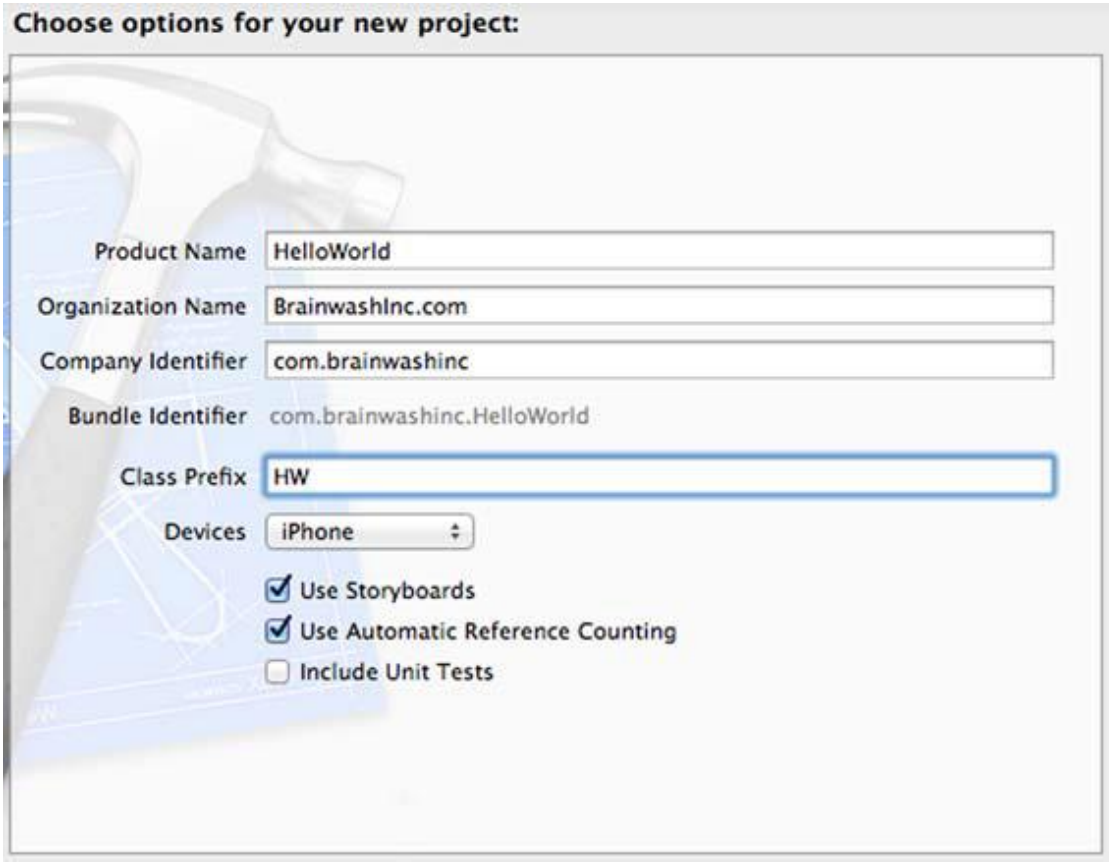


图 1.10 新工程的 Xcode 选项

点击 Next，进入到 Finder 窗口，这里可以指定工程的位置（见图 1.11）。在这步的时候，你还可以给工程创建一个本地的 git repository。（见图 1.11 的底部）。

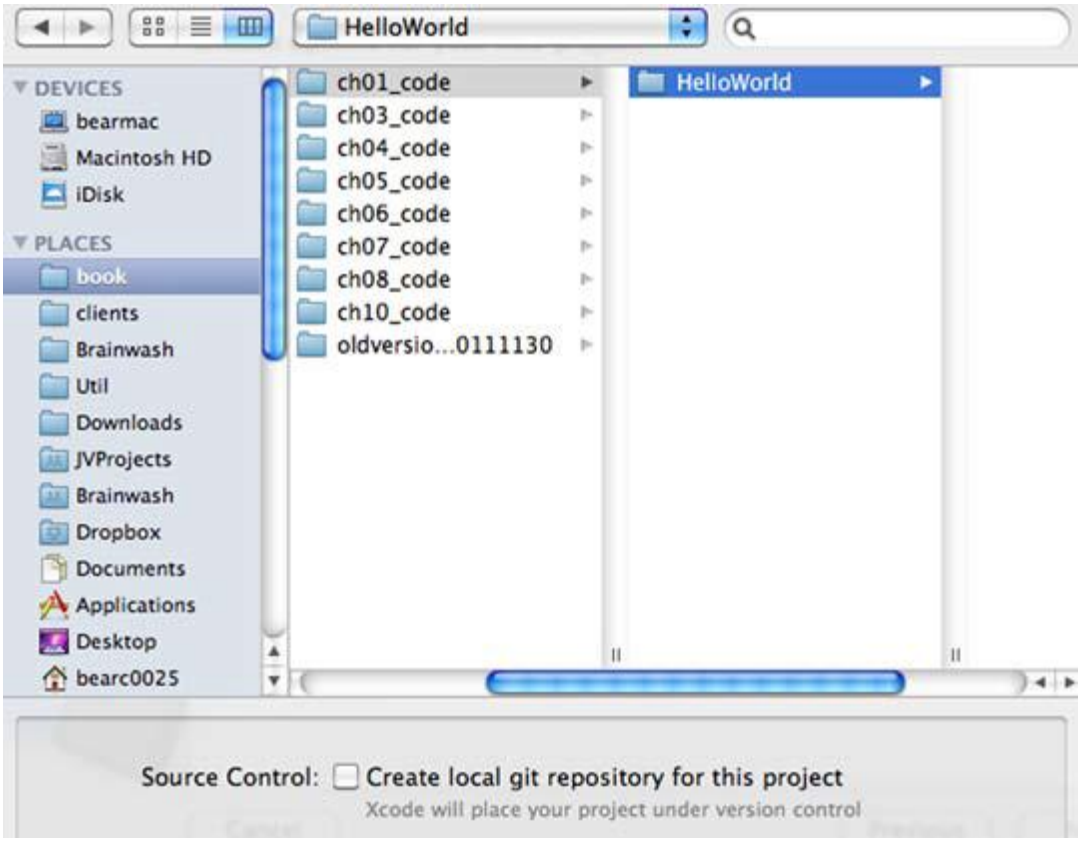


图 1.11 指定新工程的位置

点击 Create，现在你的工程的开发前准备工作做好了。Xcode 会显示默认的 Target 的 summary(见图 1.12)。你可以看到之前做的选择。请特别注意在 Navigator 中使用前缀做的命名约定。

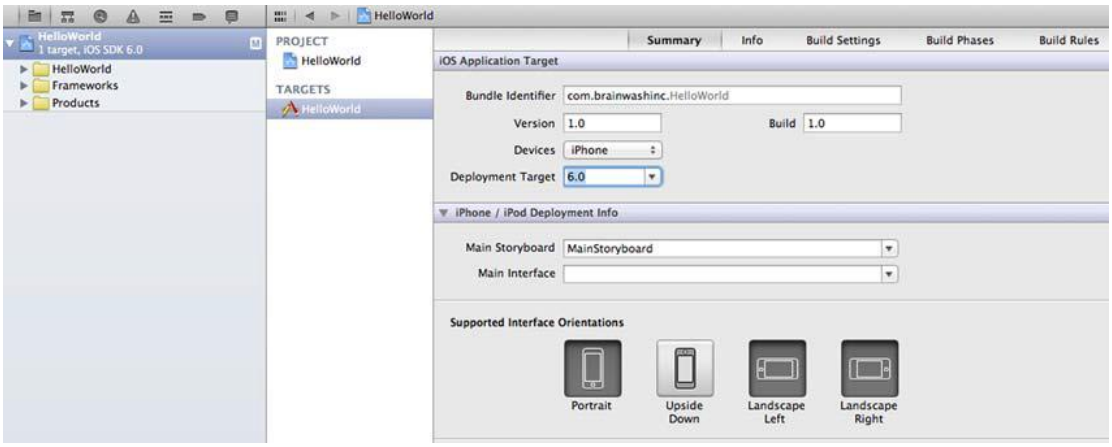


图 1.12 新项目的 summary

由于在创建应用的时候，你选择了使用 Storyboard 设置，所以还要注意 MainStoryboard 的 Main Storyboard 设置。如果你没有检查那个盒子，这个设定可能是空的，取而代之，就会使用一个 Main Interface 设置。

默认的 Main Interface 设置会将 UI 设计和 xib 文件想关联。然后，你的工程就会有一个.storyboard 文件。本书的大部分工程都会使用 XIB 文件来做 UI 设计，但是也会使用 storyboard。现在让我们来看看这个文件还有你第一个应用的 UI。

1.3.2 编辑 UI

基于模板对工程进行开发之前，首先让我们运行工程。是的，现在的状态已经可以进行编译和运行了。请确保在 scheme 的下拉目录的做上角选择 iPhone 模拟器（见图 1.13），然后点击运行。

图 1.13 选择 iPhone 模拟器

Xcode 将会编译，关联，使用 iOS 模拟器执行代码。现在只会显示一个空白的屏幕，因为应用现在还没有内容，让我们来改变它！

在工程中点击 storyboard 文件（比如说，MainStoryboard.storyboard）;UI 的内容会在编辑器中显示，以我们的情况来说，是 Interface Builder 做的 UI（见图 1.14）。白色的矩形是 view controller，之前在模拟器中运行项目的时候见过。

注意一下 Navigator 中的 HWViewController.h/.m 文件。注意编辑器左边表单中的项目（见图 1.14）。列出的 View Controller 是 HWViewController 类的实例。因此，在编辑器中改变它会影响应用的运行效果。

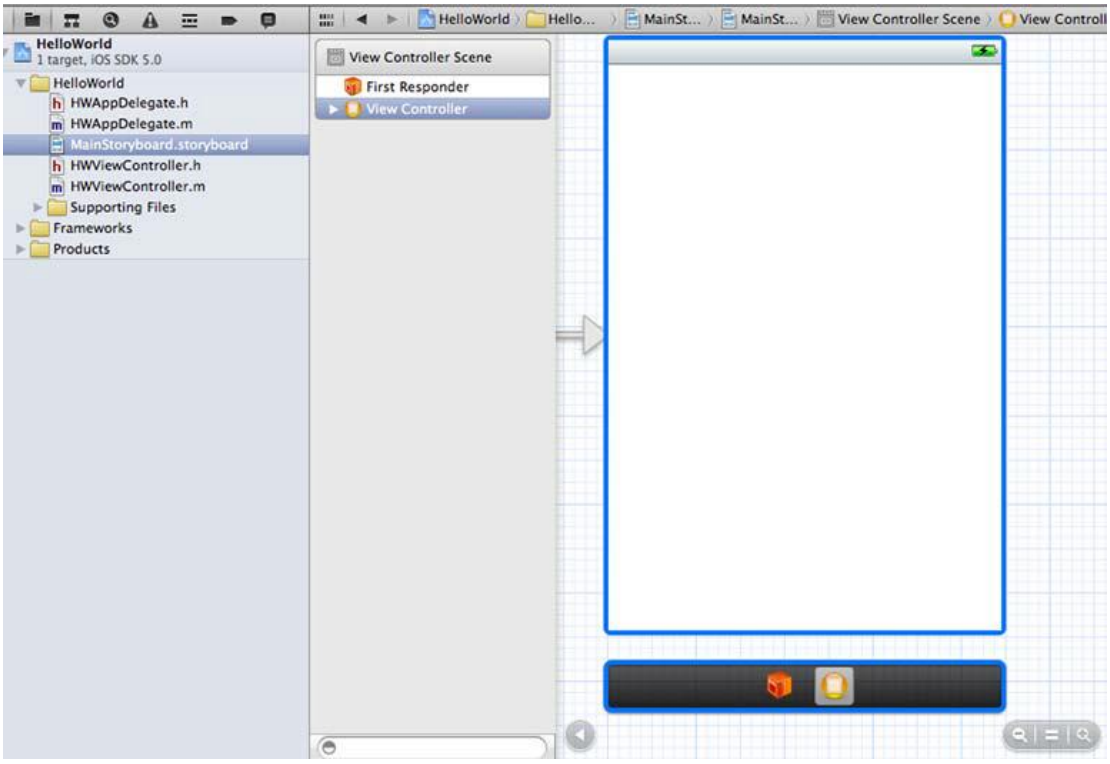


图 1.14 在编辑器中显示的 storyboard 文件

请确定 Utilities 是可见的（Xcode 的右上角，View 上的右边按钮），然后还要注意在底部有一个 UIKit 框架中的界面组件列表（见图 1.15）。

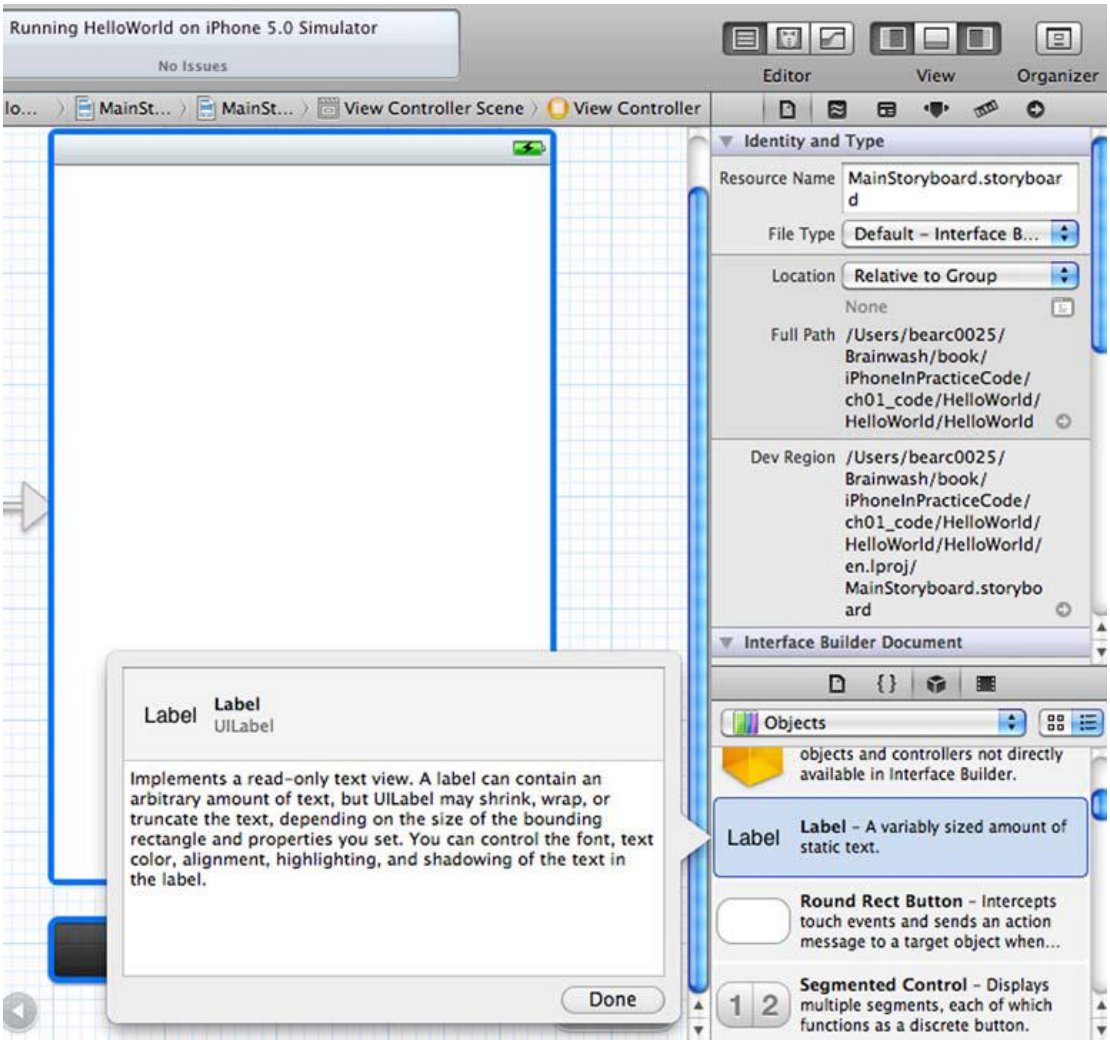


图 1.15 storyboard 文件，显示 Utilities

双击 label，然后输入 Hello World（见图 1.16）。

图 1.16 给工程的 UI 添加一个 label

现在再次运行应用，然后...恭喜！你有了一个具有一定功能的应用了！看看图 1.17，了解一下应用的 label 应该是什么样子。

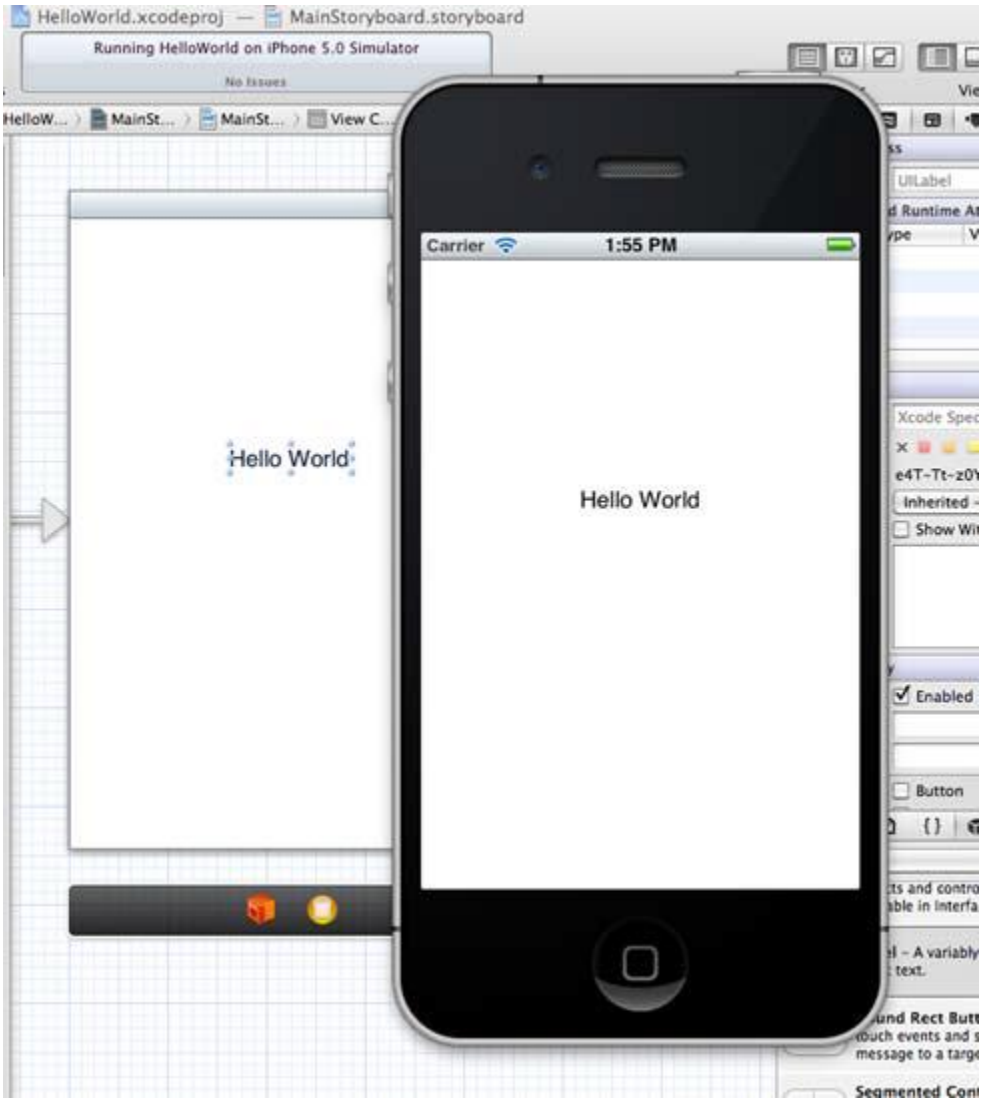


图 1.17 在 iOS 模拟器上运行的 Hello World 应用

现在再次运行应用，然后...恭喜 ！(见图 1.17)

1.4 总结

做的好！你的第一个应用完成了！不是很难，对吗？现在你是一个 iOS 开发者了（尽管我不指望你可以从你刚刚做的应用赚得什么钱）!

在本章中，你了解了 Xcode，包括如何获取 Xcode 还有它的各个部分，区域，views，编辑器等等。基于这些知识，我们快速的开发了一个 Hello World 应用来体验 iOS 开发。

但是这仅仅是开始。我相信你见过 iOS 应用做的一些很令人惊叹的事情，所以我们还可以实现更多的可能，而不仅仅只是刚刚做的那些。下一章，对 Hello World 应用我们还可以做一些更多的工作，包括 UI 内容。可以做的事情没有任何限制，现在就让我们开始吧！

2 ——创建一个 iOS 应用

本章包括：

- 实现头文件
- 制作按钮
- Action 和 outlet

上一章，你创建了你的第一个 iOS 应用。但是说你“开发”了一个应用有些夸大。因为 Xcode 做了大部分的工作，而你仅仅只是增加了一个 label，然后输入“Hello World”。你还没有做任何的 UI，甚至没有写任何代码。

在本章中，你将给 Hello World 应用添加一些基本的东西，把这个应用做的更接近一个正规的应用，而不是 Xcode 的模板。

首先，让我们来看看源码文件的类型（.h 和.m 文件）。然后你将学习如何响应用户操作并根据需要在代码中做出处理。最后，我将简要介绍一下委托模式，它可以将消息从一个对象传递到另一个对象。

2.1 源码文件

Objective-C，像其他面向对象的语言一样，有类来封装数据，消息来处理数据。

摘自《学习 Objective-C，入门指导》（LEARNING OBJECTIVE-C: A PRIMER）：在 Objective-C 中一个定义一个类需要有两个部分：第一接口，第二实现。接口文件包含了类的声明，定义了实例变量和方法。接口文件通常是.h 文件。实现文件包含了类的方法。实现文件通常是.m 文件。

如果你对 Objective-C 语言不熟悉的话，我推荐你阅读《学习 Objective-C，入门指导》（见 http://mng.bz/65W2)。 学习本书，你不需要完全掌握 Objective-C，但是如果你对 Objective-C 熟悉的话，那么本书中关于它的部分就不会成为你阅读本书的阻碍，学习会更容易一些。

在你的工程中，可以看到有两个类：HWappDelegate 和 HWViewController（见图 2.1），请注意创建工程的时候设定的 HW 前缀的文件名。每个类都有它的接口文件（.h）以及它们的实现文件（.m）。接口文件，通常被叫做头文件的内容不是很多。

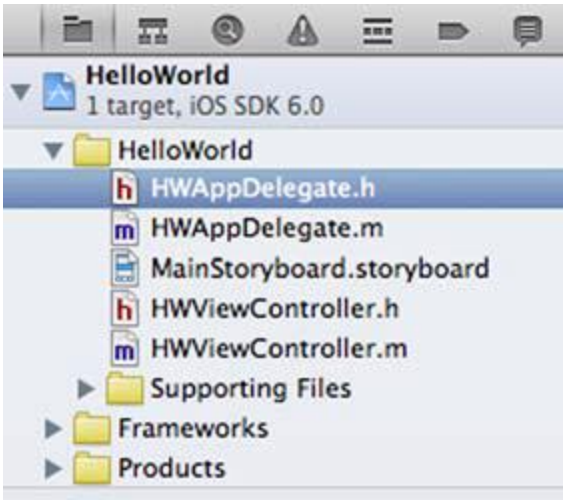


图 2.1 Hello World 工程的源文件

最有趣的是类的名称后面出现的东西（见下面的代码片段）。在冒号（ :）后面的项目指定了这个类继承的父类。以我们的 HWViewController 文件来说，这个类是继承了 UIViewController。

```
@interface HWappDelegate : UIResponder <UIapplicationDelegate>
```

HWappDelegate 继承了 UIResponder。但是更重要的东西是<>符号之间的内容<UIapplicationDelegate>。它标识类实现了文件中指定的接口（见图 2.2）。

```
@protocol UIApplicationDelegate<NSObject>

@optional

- (void)applicationDidFinishLaunching:(UIApplication *)application;
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions __OSX_AVAILABLE_STARTING(__MAC_NA,__IPHONE_3_0);

- (void)applicationDidBecomeActive:(UIApplication *)application;
- (void)applicationWillResignActive:(UIApplication *)application;
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url;  // Will be deprecated
    at some point, please replace with application:openURL:sourceApplication:annotation:
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *
    )sourceApplication annotation:(id)annotation __OSX_AVAILABLE_STARTING(__MAC_NA,__IPHONE_4_2);
    // no equiv. notification. return NO if the application can't open for some reason

- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;      // try to clean up as
    much memory as possible. next step is to terminate app
- (void)applicationWillTerminate:(UIApplication *)application;
- (void)applicationSignificantTimeChange:(UIApplication *)application;        // midnight, carrier
    time update, daylight savings time change

- (void)application:(UIApplication *)application willChangeStatusBarOrientation:
    (UIInterfaceOrientation)newStatusBarOrientation duration:(NSTimeInterval)duration;
- (void)application:(UIApplication *)application didChangeStatusBarOrientation:
    (UIInterfaceOrientation)oldStatusBarOrientation;

- (void)application:(UIApplication *)application willChangeStatusBarFrame:(CGRect)newStatusBarFrame
```

图 2.2UIapplicationDelegate 的定义

继承是面向对象开发的基本部分，如果你对继承不是很熟悉的话，我建议你研究好好的研究一下。实现接口是 iOS 开发中经常要遇到的，我们会在接下来的内容中来讨论它。

但是现在的话，记住一下就好了。下一步你要给 UI 添加一个按钮，然后继续开发应用。

2.2 给应用增加一个按钮

现在让我们回到 Hello World 工程，点击 Storyboard 文件来显示 UI。

正如之前对标签做的一样，来到 Utilities 面板，下来列表，到右下角找到 Round Rect Button（见图 2.3）。

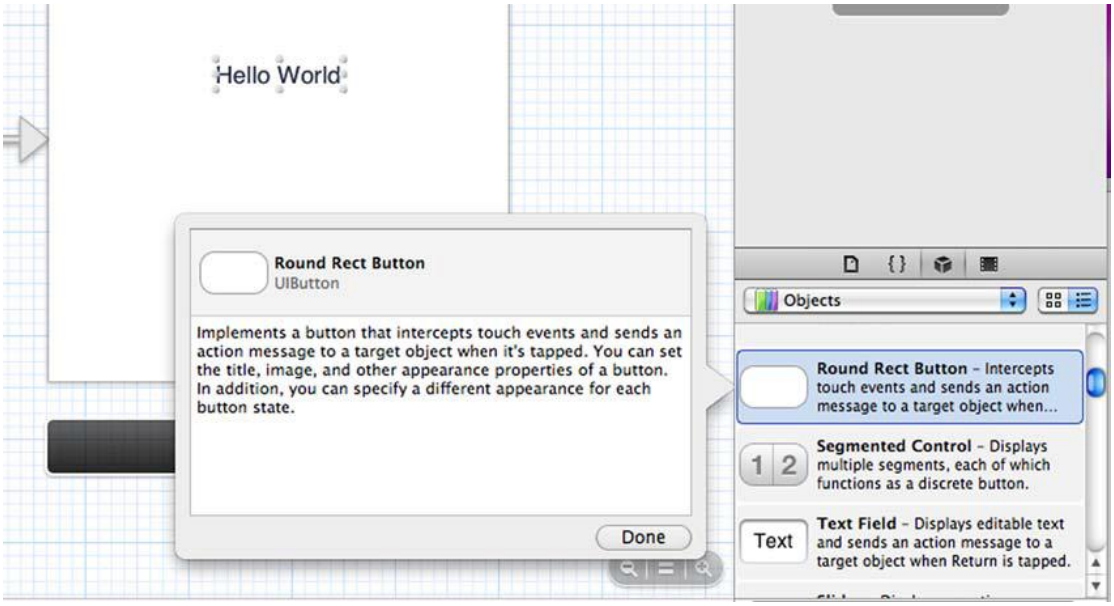


图 2.3Storyboard 的 Utilities 区域，选择 Round Rect Button

把按钮拖拽到 UI 界面上，放在 Hello World 标签的下面。双击按钮，输入 Hide（见图 2.4）。

如果你现在运行工程 ,就会显示这个按钮了。你可以点击它 ,但是现在什么反应都没有。这是因为还没有为这个按钮开发任何的功能。让我们现在来给这个按钮开发功能吧。

2.3 把按钮和一个 action 连线

为了接受用户事件（比如点击，缩放，长按，摇动，滑动，甚至说话等），我们需要在用户界面上添加一些组件，用户可以通过这些组件与应用交互，我们由此可以在代码中对这些事件进行处理，并添加自己需要的各种业务逻辑。这就回到了我之前提到过的 MVC 设计模式。UI 是给用户查看以及交互的视图。对于你的应用来说，你需要捕捉用户在按钮上的点击，调用代码，让用户点击按钮有作用。

在 iOS 开发中，你给按钮分配一个 action，当按钮被点击时则调用这个 action。UI 编辑器，IB，可以给按钮和 action 创建关系，它甚至可以帮你编码。

点击 Storyboard 文件，使其在编辑器中显示。下一步，点击 Assistant 窗格来使用它（见图 2.5）。

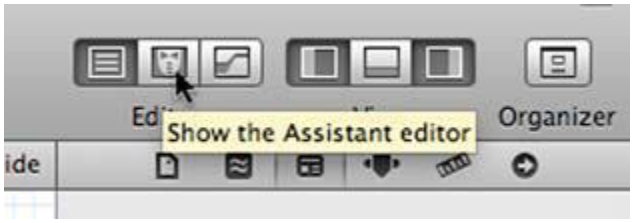


图 2.5 在 UI 编辑器上显示 Assistant 面板

现在 Storyboard 在编辑器的左边，HWViewController.h 文件在右边（见图 2.6）。如果你想要让 Storyboard 在右边显示的时候，可以使用编辑器上方的层级（hierarchy）目录。

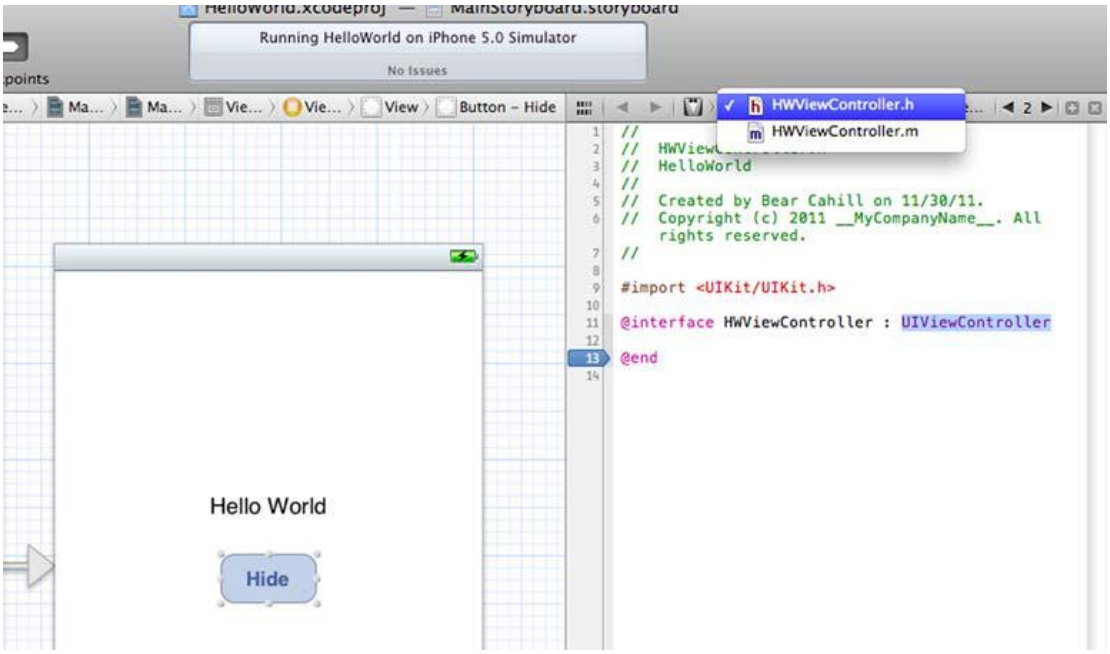


图 2.6 使用了 Assistant 模式的 Storyboard 和 HWViewController.h 文件

按住 Control 键，用鼠标拖动左边界面上按钮，这时按钮上会引出一条连接线，请您将此连接线拖动到@end 上方再释放鼠标，这时会弹出一个弹出框允许你设置创建的链接（见图 2.7）。

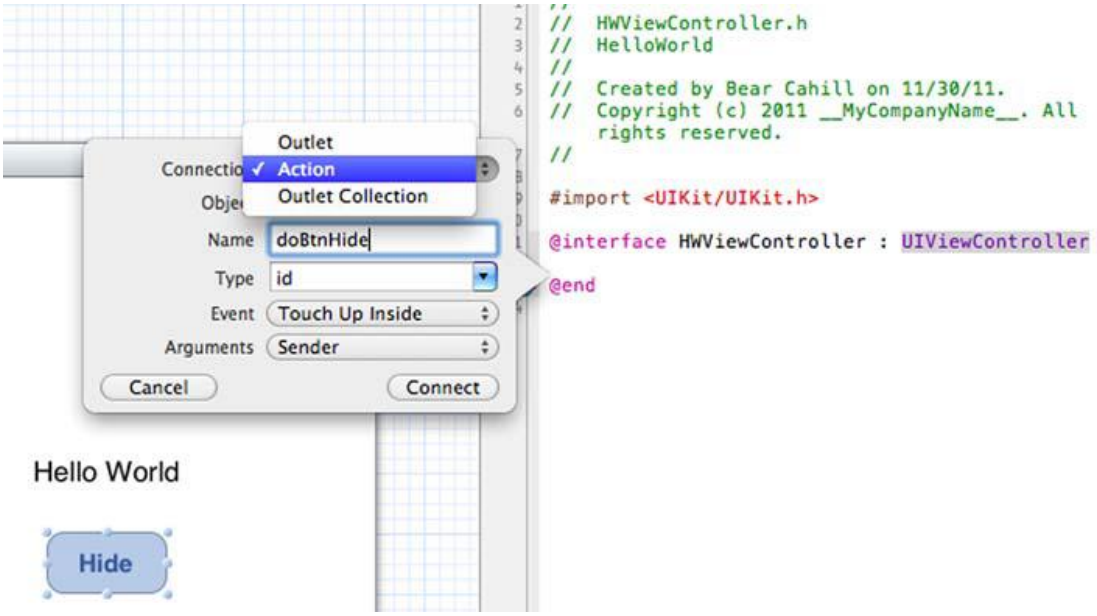


图 2.7 使用 IB 给 Hide 按钮创建一个 action

弹出窗口会让你从下拉目录中指定连线为一个 action。给 action 方法命名。按钮命名我通常采用以下方式：doBtn+一些比较合适的名字。

注意一下其他的设置。类型（Type）是 id，它表示传入事件的对象类型（译者：id 是 Objective-C 中的匿名类型，由于 UIResponder 中响应链设计模式的缘故，通常在 Xcode 中会默认设置为 id，因为事件可能被子类对象或者其他 UIKit 对象所捕获。因此，我们在代码中通常使用类似 UIButton *button = (UIButton *)sender;的代码将 id 类型的对象转化为需要的对象），事件（Event）是 Touch Up Inside，这就意味着如果用户触摸了按钮，然后把手指从按钮上离开，就会触发这个事件（如果用户把手指滑离按钮的话则不会触发这个事件），并执行这个 IBAction 方法。最后，参数是 sender，以我们的例子来看，就是我们所连接的这个 UIButton 对象，这样我们就获取了按钮的引用，并作为参数传递给这个 IBAction 方法。

点击 Connect，Xcode 就在头文件创建了该 IBAction 方法声明，同时在.m 实现文件中创建了该方法的方法块，现在我们可以看到头文件和实现文件中都有 -(IBAction)doBtnHide:(id)sender 这行代码，它们看起来几乎一样。唯一的区别是，实现文件中的这行代码后面有“{}”，头文件中的则是“;”。现在方法体为空，这表示在您点击按钮时，虽然会执行这个 IBAction 方法，但不会发生任何事情，接下来就让我们把事情变的有趣一点儿，您只需要直接在方法块的{ }中插入代码就可以了。

另一个方法是先创建 IBAction 声明，然后右击按钮，从 event 列表中选择事件连接到该 IBAction 方法。

2.4 连线标签（label）作为 outlet

和连线 action 和 UI 元素一样，你也可以在代码中把 UI 元素和 outlet 连线。这么做可以让你在 UI 编辑器中创建的 UI 元素在代码中有一个引用。

给标签在代码中增加一个 outlet 连线于和 action 连线比较相似。打开 Storyboard 文件，显示 Assistant 面板，按住 Control 键把 label 和头文件连线（见图 2.8）。你可以使用默认设置，然后给变量输入名称：lblHelloWorld。

现在你的按钮有了一个 action 连线，标签有了一个应用。那么当点击 Hide 按钮时你希望发生什么事情呢？

正如和 action 一样，还有一种方式就是先创建 IBOutlet 声明，然后按住 Control 键，用鼠标拖动该标签，并将其连接到该 IBOutlet。

2.5 实现按钮的 action

现在你可以实现 doBtnHide:action 的功能了。最明显的就是把某个东西隐藏起来。由于你有一个 Hello World 标签的 outlet，这个 outlet 是作为 Hello World 标签的引用，让我们隐藏这个 Hello World 标签。

这个方法唯一要添加的代码即是：[lblHelloWorld setHidden:YES]，现在完成好了。运行它，隐藏按钮（见图 2.9）。

图 2.9 点击 Hide 按钮把 Hello World 按钮隐藏起来

如果再点击一下按钮，可以重新显示标签就再好不过了。使用[lblHelloWorld

setHidden:![lblHelloWorld isHidden]]可以很简单的完成这个功能。点击的时候如果是隐藏状态 则显现 ,如果是显现状态 则隐藏。但是我不是很喜欢按钮的名称一直是 Hide。我们可以基于隐藏的设置来对此做改变：

```
[sender setTitle:[lblHelloWorld isHidden]

? @"Show" : @"Hide" forState:UIControlStateNormal];
```

2.6 委托

有一个概念我们之后还会遇到 ,但是现在我想提一下。这个概念就是委托(delegation)。

我之前提到过，在头文件的@interface 声明中，“<”和“>”之间指定了该类所实现的协议。以我们的 HWAppDelegate 类来说，它实现了 UIApplicationDelegate 协议。

委托就像是对象之间传递消息的通道，它可以将消息从一个对象传递到另一个对象，因此，一个对象可以通过委托调用另一个对象的方法。简单的一个例子就是文本框。每次在文本框输入文字时，用户会先触碰 Return 键，输入框会询问其 delegate 它是否需要返回。文本框通过调用自己的委托中的方法：textFieldShouldReturn:(UITextField*)textField;来进行询问的。

这个方法返回了一个 Boolean 值，这个值决定 输入框是否需要回车创建新的一行。大部分情况下，它的 action 会结束输入，然后保存数据，提交数据，发送数据，或者根据用户的输入决定到底应该做什么。

以你的应用来说，HWappDelegate 类是应用的 delegate，所以会调用大部分生命周期相关的方法，比如说 applicationWillTerminate:。

iOS 开发中会经常使用到这个概念，这也是 MVC 模式的另一种应用。

2.7 总结

现在我们已经对 Hello World 应用做了一些深入的开发。应用的功能不多，但是你用到了很多关键的概念，应用方法等，这些会在大部分的应用开发中使用到。使用 UI 编辑器，

你可以创建图像视窗，开发，文本框或者其他的界面元素。通过使用 Utilities 视窗，你可以设置任意的属性。

通过给你的各种 UI 元素创建 action 和 outlet ,你可以使用户和这些 UI 元素进行交互，基于这些交互让应用完成任务。

本书的第二部分，每章你都可以从头开发一个新的项目，然后我们一起探索更多的 UI 元素以及各种框架提供的功能。这些从开发你的第一个应用学习的基本步骤会在以后的开发中帮助你很多。

——通过教你制作一个上架应用 PicDecor 来教你使用 view controllers 以及创建展示图片

3

本章包括：

- UI 设计概念
- 视图控制器
- 使用照相机功能
- 创建以及显示图片

本章教大家开发一个应用 PicDecor，这个应用可以允许用户从相册上传图片或者使用照相机拍照，然后用应用中系统自带的装饰图片来对图片来进行加工。图片加工完成之后，用户可以保存图片，邮件完成的图片给朋友。尽管这个应用有点傻，但它的功能你可以应用在其他应用中。PicDecor 的下载地址是：http://itunes.com/apps/picdecor。 这章将给你展示我是如何一步步制作这个应用的，App Store 的上架版本我只另外添加了一些图片和一个 About 页面。

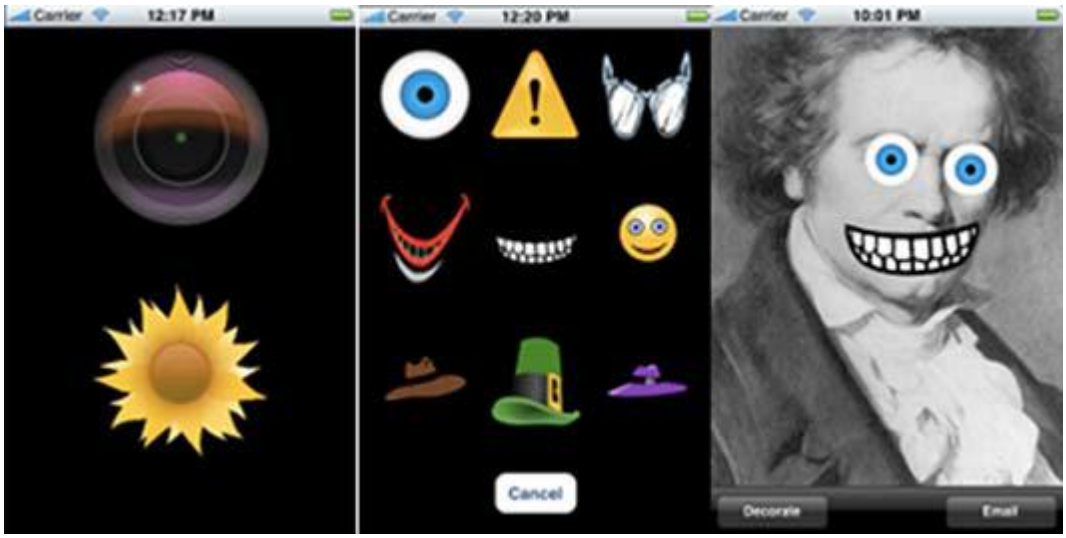


图 3.1 PicDecor 的界面：从左到右依次是，选择来源图片，应用自带图片，装饰好的图片

首先让我们看看 PicDecor 的界面，见图 3.1。第一个界面让用户选择来源图片——照相机拍照或是相册。用户选择好图片之后，他们就可以从应用自带的图片中选择一些小元素添加到来源图片。通过手指操作，用户可以加工来源图片。

在本章中，你可以看到 PicDecor 是如何设计和编码的。你可以学到视图控制器（view controllers），图片视图（image views），图像处理，在视图控制器（view controller）中的 delegation 委托类以及发送消息（例如说，发送邮件）。学习这些基本，在以后开发应用的过程中都可以用的上。你可以知道问题的解决方式有哪些，在什么情况下使用它们。同时，还可以学到用的不是特别多的功能，比如说应用中访问照相机以及相册。

我们先开始设计 UI。在设计的过程中，我们需要保持一个开发的心态，这样才能制作出一个风格的 UI。

然后，我们可以开始编码，给设计好的 UI 控件准备相对应的类，使用 IB 声明类和方法可以让你把代码和设计好的 UI 连接起来。

接下来我们可以开始将接下来我们可以开始将 UI 和相关代码连线，从而达到关联用户界面设计和相关功能的目的。

在一些案例中，UI 界面设计驱动功能开发，或功能开发驱动 UI 界面设计。有时候，在一些案例里，两种情况都会出现。基于以上情况，你可能会考虑该先做 UI 界面设计还是先开始代码功能设计。

PicDeor 是个小工程，所以我们可以边完善 UI 设计边编码，这样不会浪费时间。

现在可以开始设计 UI 界面了，让我们看看如何在 Xcode 的 IB 中使用 UI 编辑器。

3.1 UI 设计概念

在考虑 UI 的外观和操作的时候，要记住操作要比外观更加重要。在某些情况下，很多东西都能使用不同技术完成同一个功能。基于你的应用，用户，功能，以及其他方面，需要做不同的决策。我们需要考虑的东西不止有 UI 元素的大小，形状，放置位置，按钮或者表单的颜色。

你可能倾向于不同的 UI 设计方式， 手绘，使用白板或者 IB。他们都各有优点。无论怎样，当你确定了你的设计方式，确保多思索，而不是仅仅做到实现功能以及给它弄出来就行。就像前文提到过的，设计 UI，Xcode 的 IB 是一个非常好用的工具，它不仅提供了苹

果的标准控件，而且当你决定以编程的方式去实现你的 UI 的时候，IB 提供了例如坐标，大小，颜色等可以在代码中使用的属性。

Xcode 提供了很多非常棒的 UI 控件。我们可以使用它提供的控件，同时也需要想些办法来定制这些控件，让我们的应用更独特。

3.1.1 创建自定义代码块

我们可以自定义Apple Library里的所有item。大部分的item都是由其他的发展而来的，而一些最基本的item也可以相应的更改属性。比如说UIImageView, UITableView,以及UIWebView都是UIView，它们都是必须的。

UIView（引自Apple官方文档）

UIView给绘图以及操作事件提供了一个架构。一个UIView对象定义了一个屏幕上的一个矩形区域，同时处理该区域的绘制和触屏事件。一个视图也可以作为其他视图的父视图。

在 IB 的 Attributes（属性）窗口中，可以显示被选择的类（包括它们的父类）的属性。比如一个 UITableView 的属性窗口，在顶部可以看到 table view 的属性，每个相继父类的属性会在它子类属性的下方显示。如图 2，你可以看到，scroll view 的属性在 table view 属性的下方显示，UIView 的属性显示在最底部。

图 3.2 Attributes Inspector 中显示一个类属性的层次（hierarchy of a class’s attributes）

在 Attributes Inspector（属性检查器）中改变不同的属性，可以实时的看到这些 UI 属性的变化（同样的，如果你改变 UI，他们的属性也会随之一起调整）。在 UI 设计中，除了尺寸和外观，还需要考虑其他方面的东西。比如说，对一个用户来说，什么样的界面才吸引人？

3.1.2 产品定义声明

好的 UI 是根据产品项目设计的。大部分工程都要求 UI 满足功能性的需求。许多苹果的应用都使用结构简单，功能完备的 UI 设计。这并不是说他们看上去不美观或是设计的非常好，只是它们通常来说都不是自定义的 table views，而且背景图不会使人分心。

在开始设计 UI 之前，你需要确定产品定义。谁是你的用户?他们当然是 iPhone 或是 iPod Touch 的用户，但是使用我们应用的人有什么特别的需要呢？他们想要加工照片，分享照片。

对此，我们需要提供什么功能呢？我们需要有给图片增加小装饰，把处理好的图片电邮给其他人。基于此，你就有了一个初步的产品定义。

现在，让我们来完善我们的产品定义，重点要问这几个问题：我们的 app 是怎么样的，可以做什么，目标用户是谁。比如说，我们希望我们的应用简单，可以提供给想要分享照片的用户有趣的图片装饰功能。

有了这么一个产品定义， UI 设计应该怎么样呢?首先，我们需要可以让用户可以选择一张需要装饰的图片。你可能会首先想到可以制作一个按钮，让用户可以点击选择图片来源，然后再选择图片。但是我们为什么不可以直接让用户选择图片来源呢？用户知道他们需要从相册获取图片或者直接使用照相机照一张。对用户来说，操作越少越好。

基于此，苹果的 image picker 或相机功能是被不同的控件控制的。选择图片然后把图片加载到 app 中的同时，你需要让用户可以装饰图片，电邮分享。在底部的工具栏增加着两个功能应该会很好。你需要两个按钮：Decorate（装饰）和 Send（发送）。

由于我们提供了一套装饰图，所以需要让用户可以从中选择一种来装饰。为了简化这个任务，我们可以使用在装饰图上增加按钮。用户点击图片上的按钮来选择想要的图片，然后这个图片就加到原始图上了。

最后，电邮图片需要使用到默认的邮件控制器（email message controller）。好了，UI 设计好了。

3.1.3 跳出框架看问题

作为一个开发人员，我们的设计往往很直接。每次都是功能重于形式。但是我们也会理解和欣赏吸引人的美观设计。苹果一直坚持提供给我们创造丰富界面同时保证界面的性能和功能的方法。

我们需要创建三个感觉简单的视图，我们也想到了它们的设计方式，但是我们的设计是否有趣呢？有没有方法可以让我们的设计既满足功能上的需求，同时也满足趣味性呢？如果你想到的选择图片来源的视图的形式是采用两个文字按钮——照相机以及相册，那么你和我一样，注重功能大于形式。

如果取而代之的，我们使用两张图片作为视图，比如说照相机的图片和相册的图片会怎么样呢？这样也很直观，同时也吸引人。这需要用户不使用 iPhone 自带的照相机功能而在 PicDecor 中使用照相功能。需要把使用 iPhone 的体验延续到使用我们应用。

我们现在是根据苹果提供的东西来进行项目，我们需要让我们的应用突出。界面的良好可以体验可以让用户对应用的接受度更高。苹果已经建立了良好的界面概念，我们可以向他们学习。

3.1.4 像苹果一样思考

苹果创建的交互界面大部分都是非常直观而有创造力的。但是也有一些情况下，界面并不是都功能非常清晰。在 Lion 系统之前，在 Mac OS 中来调整窗口大小的唯一方式是使用右下角的箭头。

不过只要操作过一次，你就会用户记住怎么做了。没有必要反复说明该如何操作。我们希望 UI 直观，但是我们不需要反复告诉用户如何进行操作。你第一次使用 iPhone 的时候，有看过用户手册吗？什么用户手册？那时候如何知道怎么删除一个应用，拍照，查看相册的呢？

所以，重点是，选择你的目标用户然后去迎合他们。不要试图去迎合所有人。不要试图去取悦所有人 ,取悦所有人的意思是为了 5%什么功能都想要的用户而放弃那 95%只需要一个简单应用的用户。无论如何，不要创建一个装有所有功能的水桶，要坚持你最初的设计。

现在我们已经定下了基本的 UI 方案，接下来让我们看看如何创建 UI。

3.2 创建 View Controllers（视图控制器）以及其他控件

PicDecor 项目，我们有了三个视图，图片源文件选择，图片编辑，装饰物选择。视图控制器（View Controller）可以处理视图上添加其他的视图如图片，按钮还有其他的视图控件的情况。我们需要创建各种不同的视图控制器，把相关的视图和控制器连接起来，然后在这些视图上设计 UI。在本小节，我们会学习这些视图控制器，视图以及视图控件。

UIViewController 可以用来管理工具栏(toolbar)，导航栏（Navigation Bar）,以及应用视图。UIViewController 类同时支持模拟视图（modal view）以及当设备方向旋转时可以同时旋转视图。

技巧 1 使用 IB 设计一个视图控制器

大部分工程都会使用视图控制器。开始工程的最佳办法就是创建一个视图控制器。实际上，大部分的苹果工程模板都会创建一个默认的视图控制器。让我们看看怎么做。

问题

需要给我们的工程创建一个视图控制器，可以允许用户选择图片来源，拍照或是相册。

解决方法

首先，我们需要在 Xcode 中创建一个新项目。项目默认带有的一个 XIB 文件。然后，使用 UI 编辑器，为了方便用户选择，我们可以添加一些必要的按钮。最后，设置视图的背景色为黑色。

讨论

用户来源选择的视图，开始我提到了使用两个标准的按钮，这么做功能胜过形式。在头脑风暴后 ,我提到了可以使用两个大 icon 供用户点击。我们需要照相机以及相册的两个 icon。我们可以选择苹果有的图片，摄像头代表照相机，向日葵代表相册。

这么做功能性达到了 ,而且美观 ,我们借用了用户已经很熟悉的 iPhone 中就有的概念。

首先，在 Xcode 中创建一个新工程。选择 Single View Application（见图 3.3）。选择 view-based 工程，接下来会询问位置以及工程名，让我们使用 PicDecor 作为工程名。

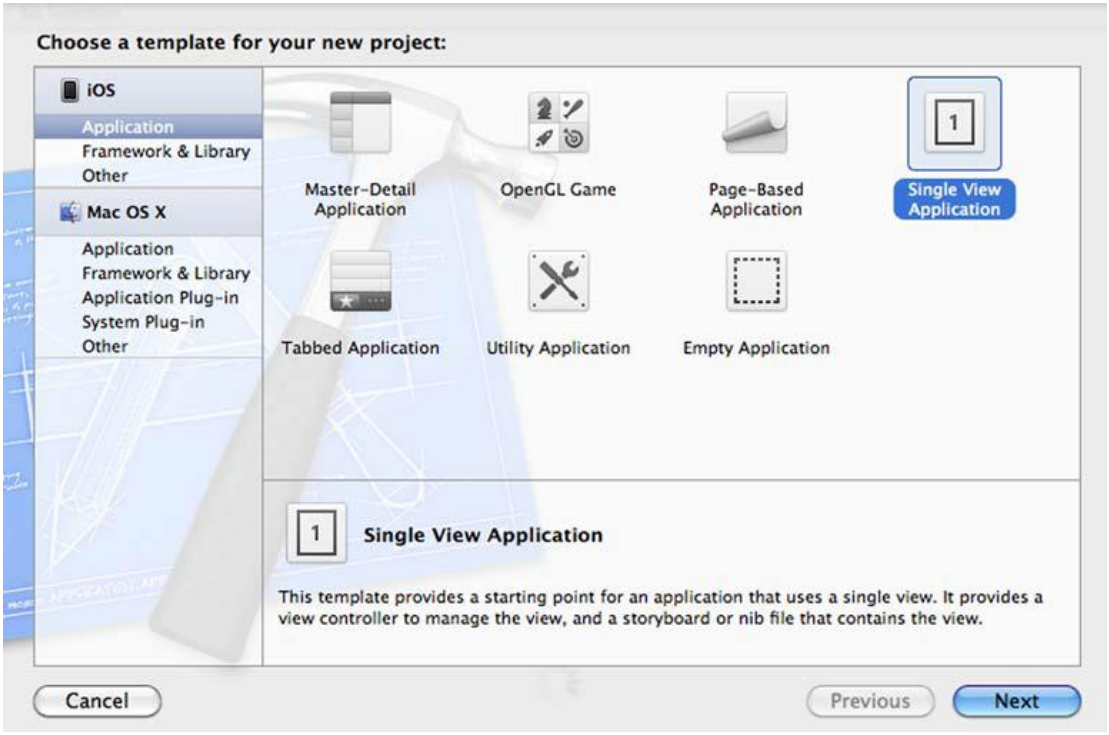


图 3.3 在 Xcode 中创建一个新的 view-based 项目

PicDecor工程下，展开Resources group，点击打开PicDecorViewController.xib 文件。（见图3.4）

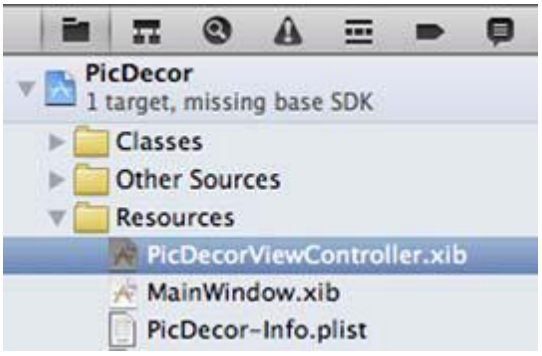


图 3.4 展开 Resources group，定位到 PicDecorViewController.xib f 文件

Interface Builder 显示了一块空白的视图，这个空白的视图就是我们应用现在的 UI。同样，如果我们在模拟器中运行（*Command-R*）应用，显示的就是这片空白（见图 3.5）。

由于我们想要应用启动的时候让用户进行图片来源的选择，需要添加按钮以及一个标签。在 Object Library 中，拖拽一个 Round Rect 按钮到视图中。（见图 3.6）

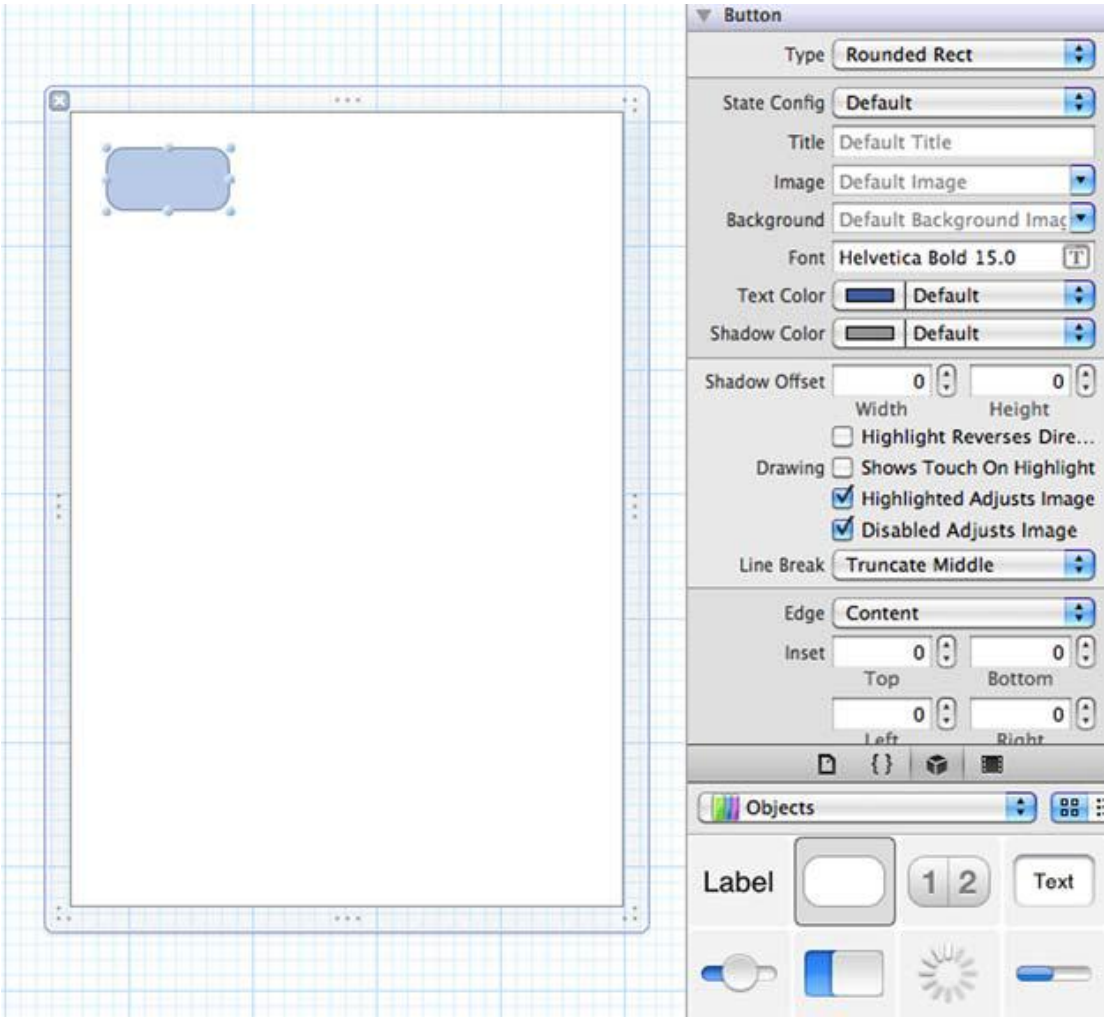


图 3.6 拖拽一个 Round Rect 按钮到视图中

现在对按钮进行自定义大小，可以通过两种方法完成，一是点击按钮，按钮会显示可供拖拽调整大小的点，二是可以使用 Size Inspector 来调整大小。让我们使用 Size Inspector，设置尺寸的数值。

点击按钮来进行选择。然后从 Tools 目录中选择 Size Inspector（或是在顶部的 Inspector 的视窗中点击尺子的 icon）。分别设置 X，Y，宽度，高度 的值为 80，20,160,160。（见图 3.7）

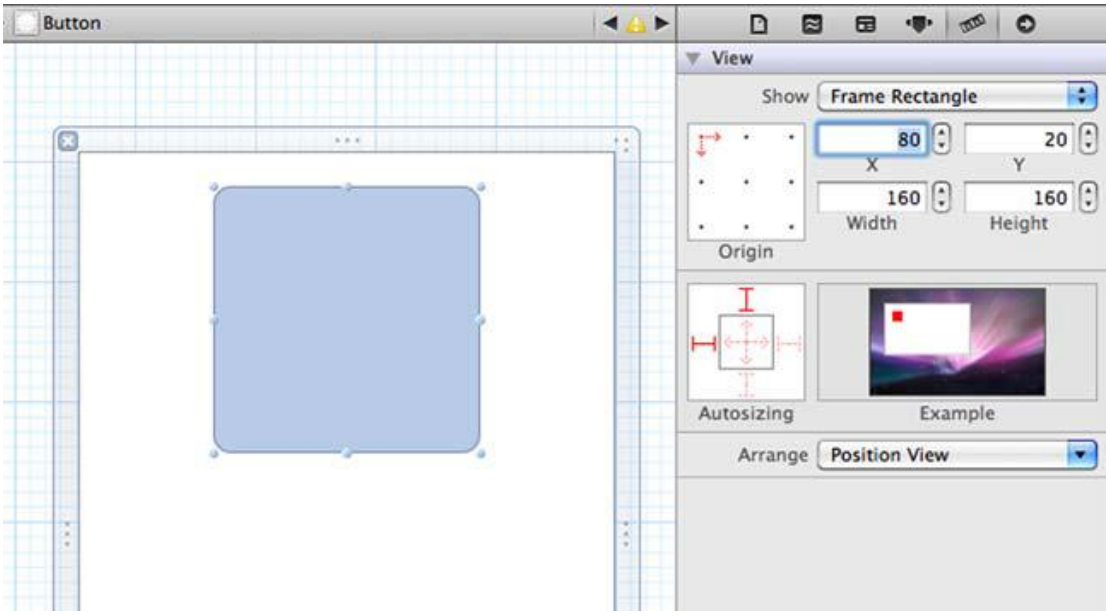


图 3.7 设置按钮的 X，Y，宽度，高度值

在视图中插入第二个按钮(或者可以直接复制黏贴第一个按钮)。按钮的数值基本相同，出来 Y 值设为 230（见图 3.8）。稍后我们可以给这两个按钮设置为图片显示，但是现在就先这么放在这儿。

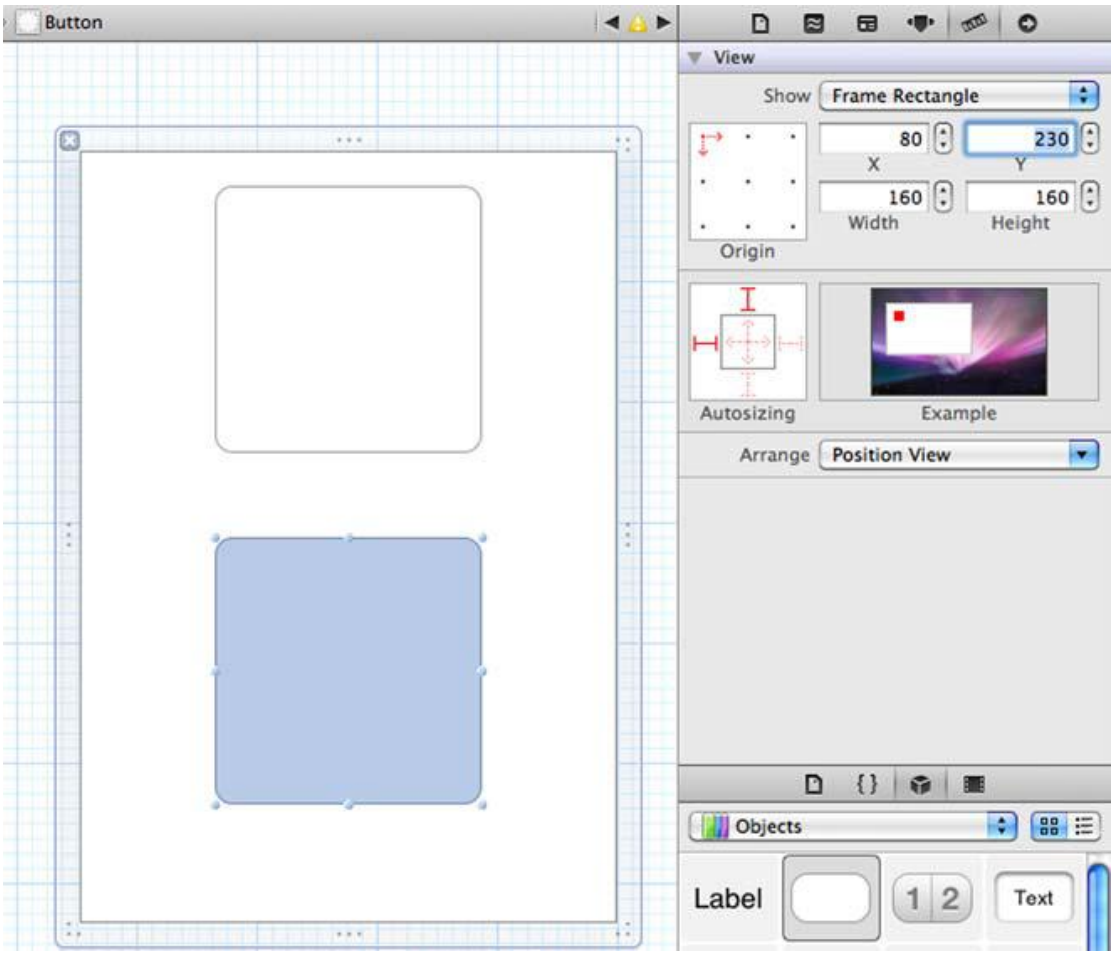


图 3.8 复制黏贴第一个按钮，第二个按钮的 y 坐标值设置为 230

最后，通过以下步骤把视图的背景色设置为黑色：

点击按钮外的灰色背景区域

选择 Tool 目录中的 Attribute Inspector 来查看属性

在 View group of attributes 中，点击背景颜色，然后就会显示 Color 工具

如果之前没有选择，在顶部选择 Color Palettes 的 icon（中间的 icon），然后在下拉目录中选择 Apple

点击 Black，然后视图就会呈现黑色（见图 3.9）

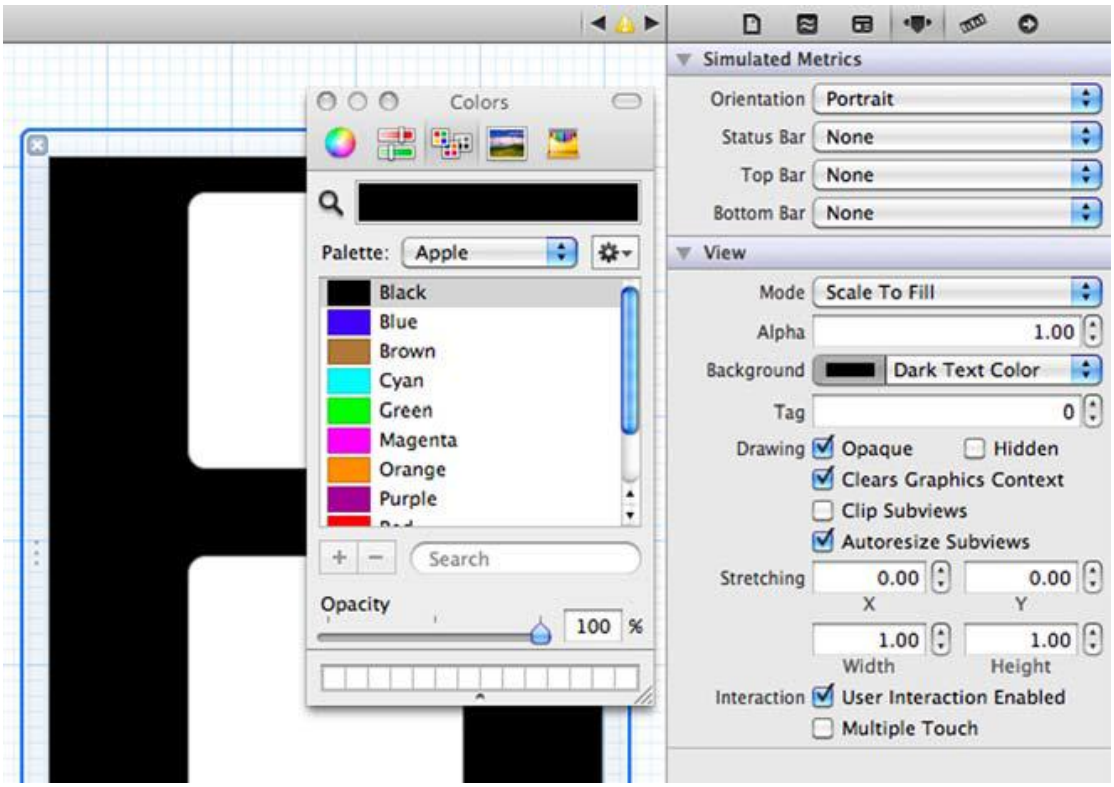


图 3.9 把视图的背景色设置为黑色

技巧 2 给视图增加一个 imageView 和工具栏

ImageViews 和 Toolbars 是 Apple SDK 提供的两个通用控件。当然，UIImageView 会显示一幅图以及一条工具栏类的按钮供用户使用。开发者使用起来很简单，用户也容易理解。我们可以让我们选择的图片在提图片视图上显示，允许用户使用工具栏操作。

用户选择好图片后，我们需要让图片显示出来方便用户编辑图片。编辑图片视图（Image editing view）是很基本的东西，因为大部分用户见到的内容都是基于选择以及交互。编辑

图片视图需要一个新的视图控件，因为项目的根视图控制器（root view controller）在处理图片来源的选择。

问题

我们需要创建一个带视图的试图控制器，这个视图用来显示用户选择的图片，用户在这个视图上可以装饰图片。同时，还需要增加一个带按钮的工具栏，方便用户装饰图片以及 email 图片。

解决方法

给工程的 xib 文件添加一个视图控制器。然后可以添加一个图片视图，工具栏，以及新的视图控制器上的按钮。同时，必要时还需要设置文本和颜色。

讨论

从 Library 选择 Controllers，然后拖拽一个视图控制器到 xib 文件（见图 3.10）

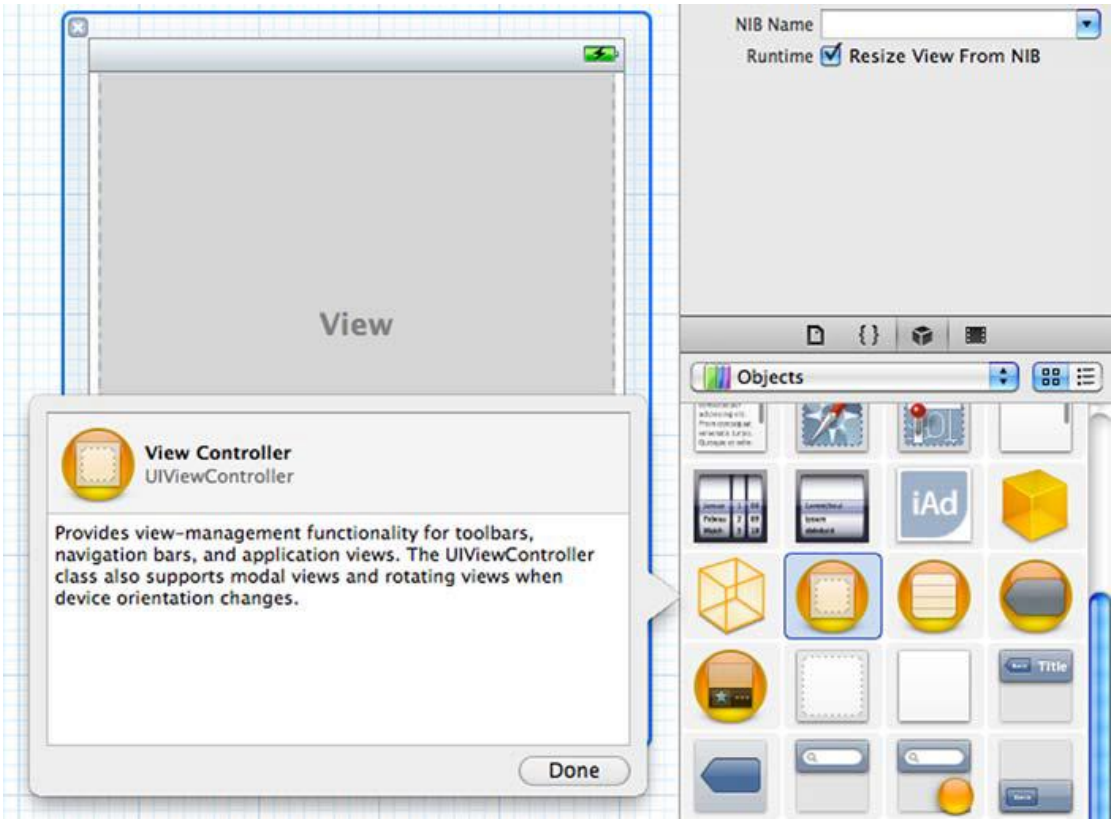


图 3.10 增加一个 view controller 到 xib 文件/工程

窗口中间有一个 word view。现在我们还没有告诉控制器这个视图是用来做什么的。这是因为很多东西都是视图，比如说表单，图片，按钮，标签以及文本框，我们需要做选择。在这里我们的选择是一个图片视图，因为我们需要编辑一张图片。

从 Object Library，拖拽一个 image view 到 controller 窗口。它会自动调试到合适的全屏尺寸（见图 3.11）。

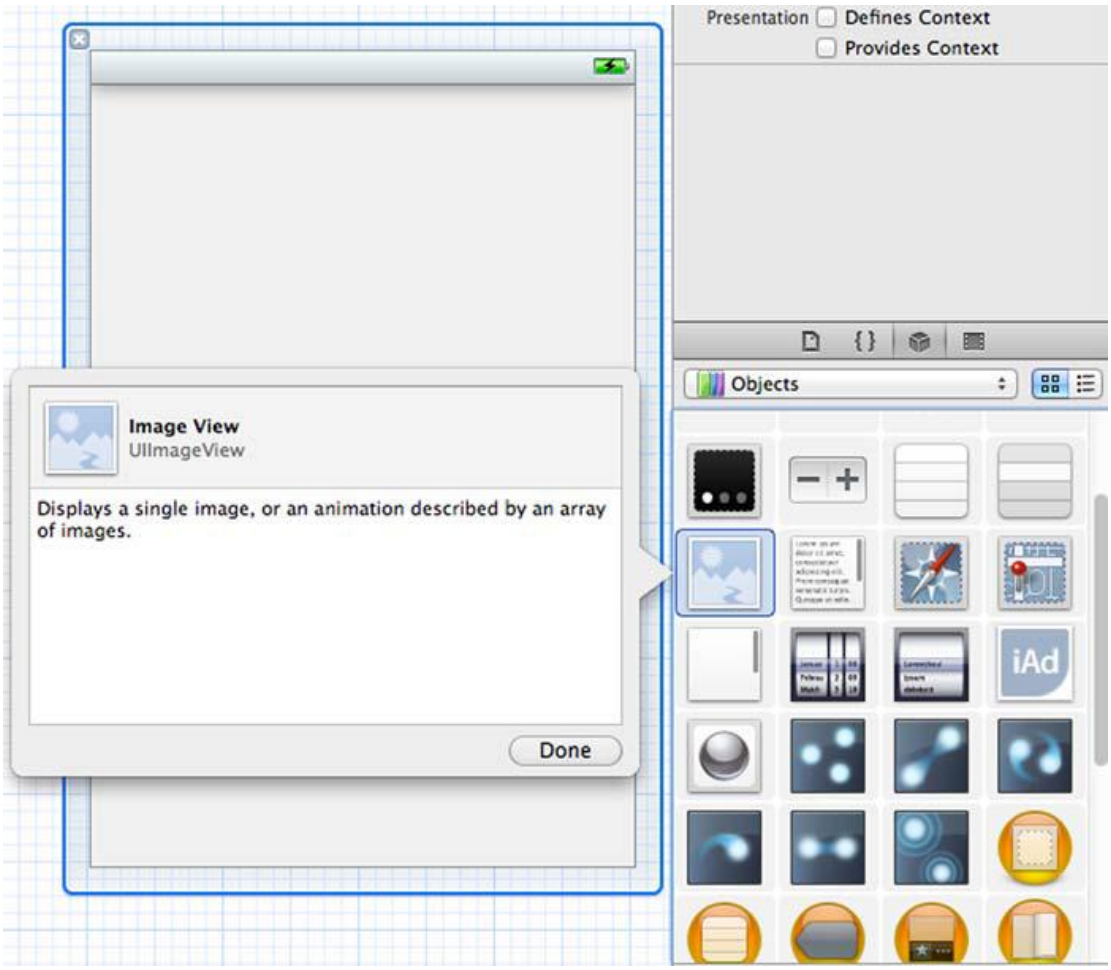


图 3.11 把一个 image view 添加到 view controller

你同样还需要一个工具栏来放置按钮。在 Library 中选择 Windows and Bars，然后拖拽一个工具栏到图片视图的顶部。（见图 3.12）。这样会发生什么事呢？

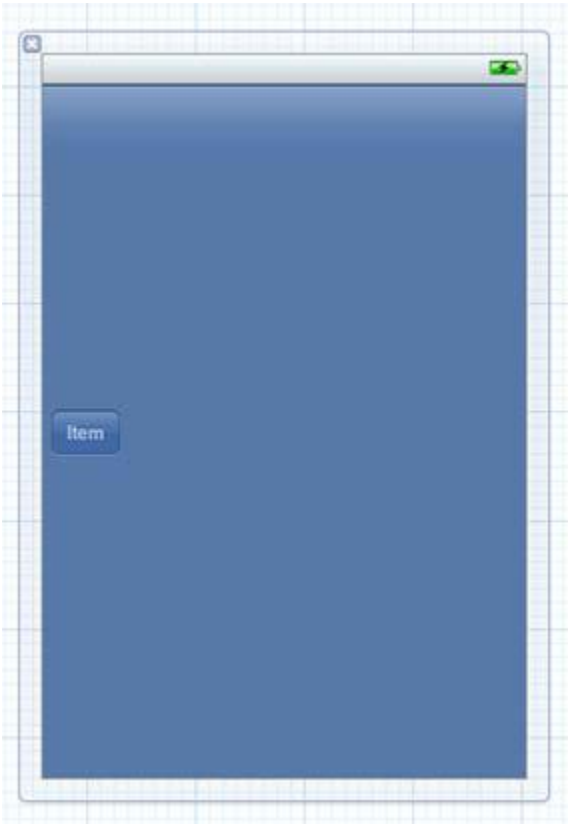


图 3.12 添加一个工具栏到 viewcontroller，代替 image view

工具栏替代了 image view。一个视图控制器只能控制一个视图，所以它把之前的 image view 移除了，新的视图工具栏取而代之。但是我们两个视图都需要。那么我们怎么来设计界面让它可以拥有两个或更多的 item 呢？

选择工具栏然后删除，输入 Command-x。在 controller 视窗中拖拽进来一个视图，不要是图片视图。现在我们再来添加工具栏和图片视图。拖拽一个工具栏放在视图的下部。最后，拖拽一个图片视图到视图左边的白色区域。由于我们是先添加的工具栏，所以 IB 会自动调整图片视图的大小让它填充剩余的视图的空间（见图 3.13）。

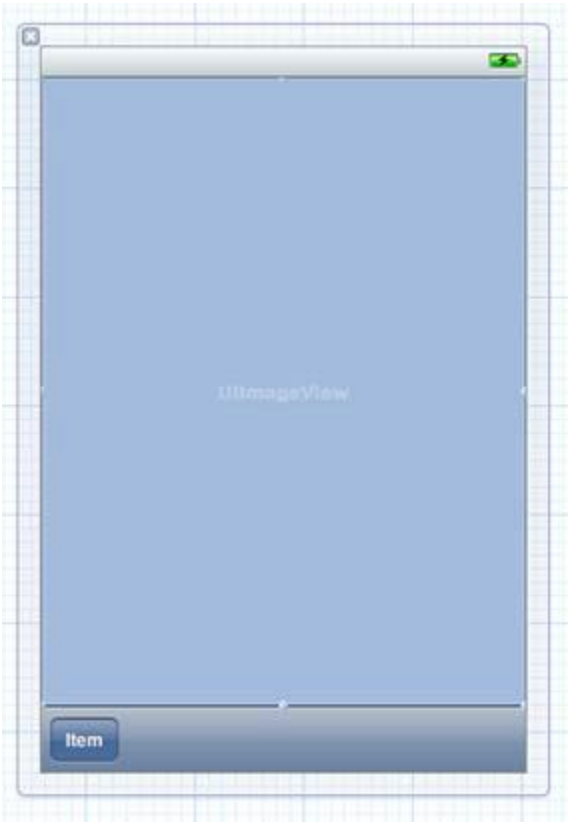


图 3.13 在添加工具栏之后再添加 image view，它会自动调整大小

现在视图控制器的视图是一个 UIView。UIView 可以包含多个子视图。在我们的案例中，我们有一个图片视图和一个工具栏。UIToolbar 也是 UIView 的子类。

通过 Object Library，再添加一个 Flexible Space Bar Button Item 到工具栏的正中心。然后添加一个 Bar Button Item 到工具栏的左边。如果工具栏上的 items 很多，flexiblespace bar 可以自动调试空间。以 PicDecor 来说，flexiblespace bar 允许在 bar 的两端都可以防止按钮。双击每个按钮，然后在左边的按钮中输入 Decrate，右边的输入 Email。（见图 3.14）

图 3.14 两个按钮被一个 Flexible Space BarButton Item 分隔开

为了让每个 bar button 的宽度相同，分别选择两个按钮，然后在 Size Inspector 中设置宽度为 100。另一个方法是选择其中一个按钮，然后按住 commard 点击另一个按钮。当两个按钮都选择到了之后，在 Size Inspector 中设置宽度大小。

SIZE INSPECTOR 视窗标题 Inspector 工具的视窗的标题会使用 item 的描述加 inspector 的类型。比如说，控制 bar button item 的大小，Size Inspector 视窗的标题会是 Bar Button ItemSize。

待会儿我们会把图片和代码连线，也会给每个按钮分配相应的功能动作。现在，我们开心会儿，基本的 UI 框架已经做好了，看上去也还不错。但是，为了配适第一个视图的黑色背景，选择工具栏，然后设置在 Attribute（属性）中设置 Style（类型）为黑色透明（Black Opaque）。（见图 3.15）设置好之后按钮就自动调整好了。

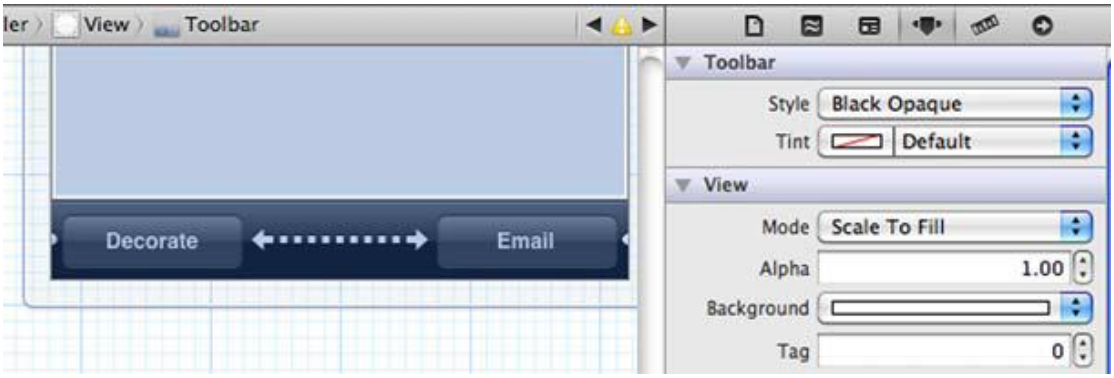


图 3.15 设置工具栏的 style 为黑色透明

现在我们创建设计好了图片来源选择的视图和图片编辑视图，现在我们可以定义装饰选择的视图。我们可以采用之前创建带有按钮的图片来源选择的视图的相似方法。这样视图的整体风格会统一。

技巧 3 给视图添加按钮

按钮是最常见的 UI 元素，它们给用户提供了一种交互。无论是一个标准样式的按钮或是非典型样子的按钮（比如说一副图或是文本的形式），按钮对用户来说很有用，而且开发者添加起来也很容易。

问题

我们需要创建一个视图，这个视图可以允许用户点击选择各种装饰图，然后添加到之前的图片中来编辑。

解决方法

我们可以创建一个新的视图控制器。对于图片选择功能，我们添加一些按钮（见技巧 5）然后再添加一些用来装饰的图片。

讨论

就像之前的图片来源选择的视图（image source selection view）一样，装饰图片选择视图（decoration image selection view）设计起来应该非常快。视图上有一组按钮链接着装饰图片，当按钮被点击的时候，可以把图片添加到主图片上来进行装饰，视图上的按钮可以各自显示相应的图片。从 Objects & Controllers group 中拖拽一个新的视图控制器到 xib 文件。从 Object Library 中的 Windows& Bars group 中拖拽一个视图到新的控制器上。使用 AttributesInspector，调整背景色为黑色。

下一步，从 Inputs & Values group 拖拽一个按钮到视图中。我们如果使用 3*3 的网格按钮，那么我们就可以添加 9 张图片，而这 9 张图片可以非常好的在按钮中 显示。View

的宽是 320，每个按钮中间的间隙是 20 像素，这就需要 3 个按钮的宽度为 240 像素。设定按钮的 X，Y，宽度，高度值分别为为 20，20，80，80。

复制这三个按钮，制作第二排按钮，Y 值设为 150。继续复制，做第三排按钮，Y 值设为 280。最后，在底部中央添加一个按钮。双击，然后输入 Cancel（见图 3.16）

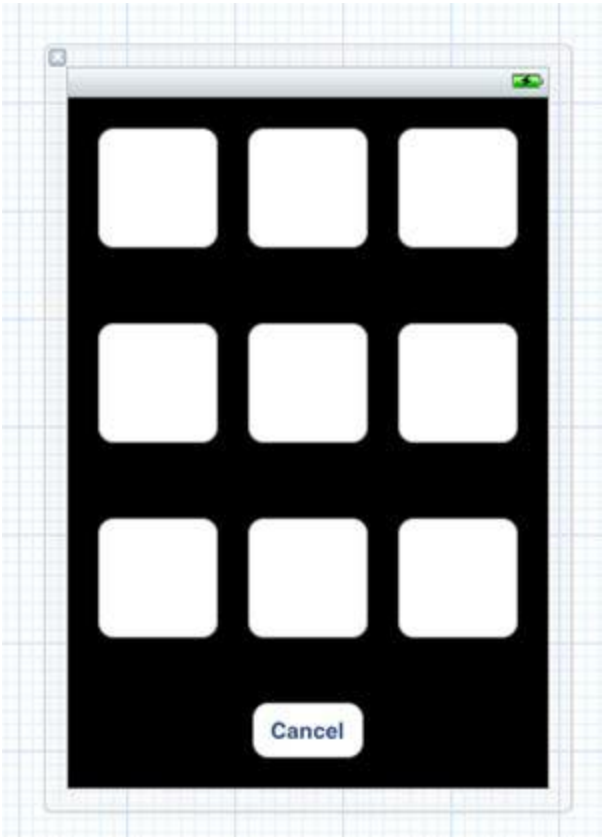


图 3.16 按钮网格以及一个取消按钮

现在我们的界面看上去有模有样了。取决于应用的发展方向不同，关于界面的制作会有不同的做法。比如说，如果你想要增加装饰图片库，那么你可能会需要一个 table view，让用户可以下拉看到其他的装饰图片。

现在让我们先假定我们对项目代码，界面和代码的结合很满意。

现在我们的 UI 还只是个骨架，之后我们要填充内容。现在，我们只需要确定我们之前的设计想法已经做出了大概的样子。然后让我们在 Xcode 创建好类之后返回 UI 编辑器。

现在我们已经设计好了界面元素的面积，位置，尺寸以及其他属性。现在让我们开始编码。下一步我们需要告诉 IB 创建的这些视图如何和代码相连，和用户的交互是怎样的。

3.3 开发 actions 和 outlets

在这个章节中，我们将在 iPhone 模拟器中第一次运行应用。虽然还有一些事情要做，但是感到欣慰的是我们的应用将会一直陪伴着我们。

在一些情况下，在 XIB 中也可以使用 Ctrl+拖拽的方式，就像在 IB 中操作的那样。可以设置一些属性和其他的一些内容，这就足够了。例如，你想要在一个图片视图中表现一个单一的图片的话，你可以拖拽一个图片视图到 XIB 文件中，并且设置它的图片源为你已添加到在项目中的图片，这就 OK 了。

同样的，在一些情况下你可以不写任何代码而添加一些按钮。拖拽按钮到一个视图中，设置文本，颜色，背景图，指定它要调用的方法，这样就完成了按钮的添加。

但是其他情况下我们需要在代码开发上做些工作。通常，这意味着要定义一个这个控件的子类，并且告诉IB拖拽到XIB中的这个控件是新创建的类的实例。 其中一个常见的例子是定义UIViewController的子类，并且告诉IB一个适当的视图控制器（view controller）来作为指定类的实例。在这种情况下，Inspector 分析这个类，并且在头文件

中提供一些可以自定义的outlets和actions的权限。例如，如果你指定了一个UIButton作为一个outlet，并且指定了一个方法作为按钮的action。IB允许让你把代码跟你拖拽到XIB中的按钮相关联起来。而且可以指定当按钮轻触的时候调用我们提供的action方法。

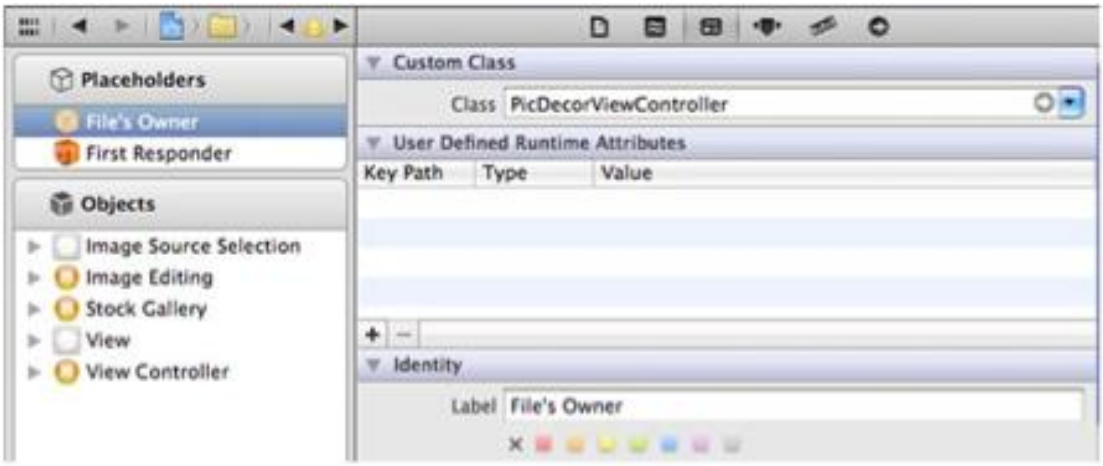
而如果在UI编辑器中先定义好了UI，那可以让Xcode通过按Ctrl并拖拽的方式把控件拖拽到头文件中来声明actions和outlets。

在 PicDecor 项目中，我们将使用类似 UIButton 的标准的 UI 组件作为 outlets。因为不需要对按钮进行自定义，所以代码不需要实现子类或者是其他一些工作。在这个小节中，我们将看看在 PicDecor 项目中如何定义合适的 outlets 和 actions。

技巧 4 在 Xcode 中声明 actions 和 outlets

Actions 是在 UI 被触发时调用的方法。Outlets 指的是 UI 控件，代码需要知道指的是哪个控件。很明确的，你需要 actions——否则的话当 UI 被触发时代码知道要去做什么事情么？声明 Outlets 有时候没有太大的必要，但是如果想要改变组件的属性的话，需要相关联一下。

Interface Builder 允许你通过按 Ctrl+拖拽 UI 组件或者是它们相关联的 actions 到头文件中的方式来进行声明。但是，如果在头文件中已经对它们有了声明，IB 就会意识到它们并且可以更方便的来关联两者。



图示 3.17 File’s Owner 是 PicDecor 项目的 PicDecorViewController 中的一个实例

问题

需要连接不同的 UI 控件和代码中的实例变量。并且，需要连接 UI 中的 actions 和代码里中要调用的方法。

解决方案

在头文件中，需要声明那些要用 IB 连线的各种的 actions 和 outlets。

讨论

点击 XIB 中的 File’s Owner。注意看在 Identity Inspector 窗口中，类的类型为 PicDecorViewController（见图 3.17）。注意我通过点击窗口底部有个指向右边的图标扩展了文件列表。

接着来看下 PicDecorViewController.h 文件（见图 3.18），它基本上是空的。但其实这个类的视图上已经有黑色的背景和两个大按钮了。所以说代码不需要知道视图的背景色，而且不需要知道这些按钮。需要知道的就是什么时候这些按钮被点击了。

在 Xcode 中声明 actions ,需要两个 actions :一个按钮一个。我命名为 doCameraBtn 和 doPhotoAlbumBtn。在头文件中定义它们（看下面的代码片段）。需要指定返回值类型为 IBAction。这样 IB 就知道它们已经被声明为可用的了。没有我们所感兴趣的真实的返回类型，因为我们不是在代码中直接触发这些方法的。当然也可以在代码直接调用这些方法，调用的时候就当它们的返回类型是 void 就可以。

```
-(IBAction)doCameraBtn:(id)sender;

-(IBAction)doPhotoAlbumBtn:(id)sender;
```

这些方法传递一个参数：id。这是 sender class（派发事件的类）的引用，在这个例子中就是指被点击的按钮。可以像使用其他的 UIButton 的引用一样使用这个引用。可以改变标题，颜色，位置或者是其他想改变的内容。

现在轮到在代码中声明视图控制器了，这将跟你在 IB 中所做的事情相关联。Xcode 提供了第一个视图控制器和它的视图，但是我们还需要创建并声明其他附加的类。PicDecor 项目中 ,我们还有其他两个视图控制器。现在在 Xcode 中创建这两个类。右键点击 Claases，选择 Add > New File....在弹出的窗口中左边列表中选择 Cocoa Touch 类，右边则选择 UIViewController 子类（见图 3.19）。

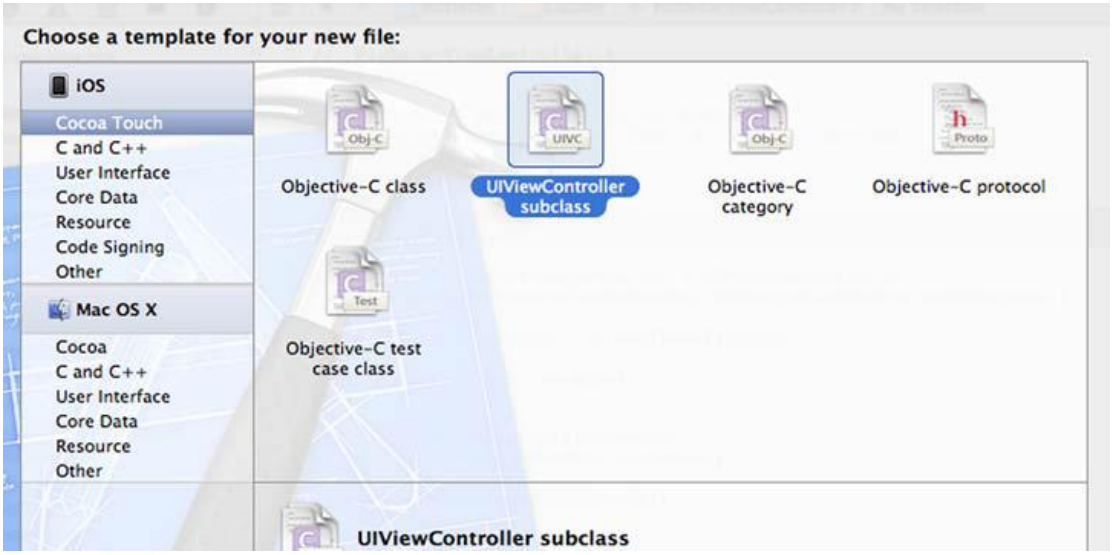


图 3.19 在 Xcode 中创建一个新的 view controller 子类

点击 Next,给类命名为 VCImageEditing,它将会自动添加到项目中。重复这个操作添加 VCDecorations 类，这样就创建了两个 view controller 类。

VCImageEditing 类有一个 outlet（指向图片视图）和两个 actions （指向装饰和发邮件按钮）。在头文件中声明这些（见以下代码）

VCImageEditing.h 中定义 outlet 和 actions

```
@interface VCImageEditing : UIViewController {

    IBOutlet UIImageView *ivEditingImage;

}

-(IBAction)doDecorateBtn:(id)sender;

-(IBAction)doEmailBtn:(id)sender;

@end
```

VCDecorations 类有两个 actions。一个 action 处理打开所有的装饰性图片库的按钮，另一个处理取消按钮。因为是通过点击按钮来执行方法，所以按钮上显示什么图片可以由你来决定。在头文件中声明这些（看下面所列代码）。

VCDecorations.h 定义 action

VCDecorations.h defining the actions

```
@interface VCDecorations : UIViewController {

}

-(IBAction)doImageBtn:(id)sender;

-(IBAction)doCancelBtn:(id)sender;

@end
```

现在我们来给这些视图控制器连线。图片源选择视图控制器（image source selection controller）需要知道图片编辑视图控制器（image editingview controller），同样的图片编辑视图控制器需要知道装饰图片的视图控制器（decorations controller）。这些同样需要在头文件中定义outlets（见以下代码）。这意味着头文件必须导入image editing controller .h文件。

PicDecorViewController 头文件中全部内容

```
#import "VCImageEditing.h"

@interface PicDecorViewController : UIViewController {

    IBOutlet VCImageEditing *vcImageEditing;
```

```
}

-(IBAction)doCameraBtn:(id)sender;

-(IBAction)doPhotoAlbumBtn:(id)sender;

@end
```

在 VCImageEditing.h 文件中，导入 VCDecorations.h 文件 ，并且声明一个这个类型的 IBOutlet。

现在代码已经完成了声明（虽然还没有实现）,我们可以开始连接 UI 中的其余元素到代码中了。之前在 UI 中已经定义了按钮跟 image view，而且在代码中已经声明了对应的 actions 和 outlets。可以开始在 IB 中相互关联它们了。

技巧 5 连接 actions 跟 outlets 到代码中

之前的技巧中有提到过，有两种方式把 outlets 和 actions 从 IB 连接到代码中。如果 outlets 和 actions 在头文件中声明了，IB 可以很方便的将它连接到对应的 UI 组件。另一种方式就是按着 Control 拖拽 UI 组件或者是它们的事件到头文件中来分别声明 outlets 或者是 actions。

问题

需要连接在代码中创建的 actions 和 outlets 到 IB 中定义的 UI 事件和组件。

解决方案

可以检查一下声明了 actions 和 outlet 的代码。然后连接合适的 UI 组件到代码的声明部分。

讨论

在 XIB 文件中选择 File’s Owner。然后打开 Connections Inspector（中间白色箭头图标的圆圈）窗口，你会发现包含 outlets 的列表（见图 3.21）。在每个 outlet 的右边的圆圈中拖拽空的点到视图中合适的按钮上面，如图示的那样。在弹出的列表中，你可以选择让用户以何种方式来调用 action 方法。选择 Touch Up Inside。这及意味着当且仅当用户点击按钮，然后手指离开按钮，但手指还在按钮的范围内时，这个方法会被调用。

图 3.21 把 PicDecorViewController.h 文件中的 action 和按钮连接起来

现在运行应用，按着这些按钮的时候会引发崩溃，因为还没有实现这些方法。

和 PicDecorViewController 类似，你需要告诉 IB 编辑图片控制器是 VCImageEditing 类的实例， decorations view controller 是 VCDecorations 对象（见图 3.22）。当然，设置名字（下侧 Label 属性）可以使得类在 XIB 窗口中看起来更加直观。

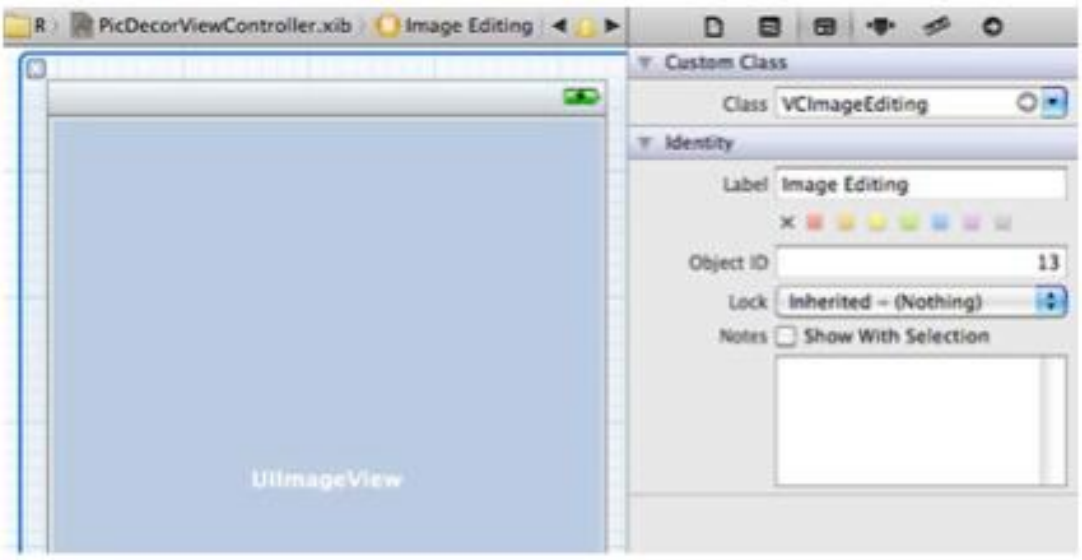


图 3.22 指定类，并且给 image editing view controller 来定义名称

点击 VCImageEditing 实例，打开 Connections Inspector 视图，拖拽 ivEditingImage 这个 outlet 右侧的空白点到之前在视图上添加的那个图片视图上（见图 3.23）。同样的，从 actions/methods 上拖拽连接点到 toolbar 上的正确的按钮上。Toolbar 上按钮不像普通的按钮那样有一系列不同的 actions，所以不会弹出一个可选择 actions 的列表。

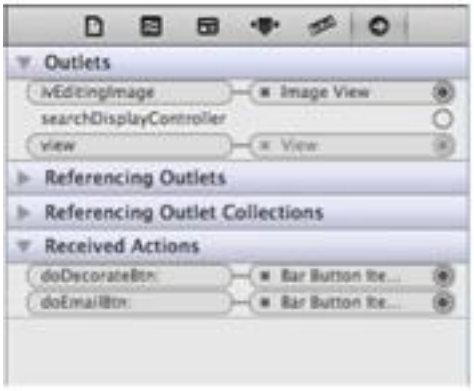


图 3.23 为 VCImageEditing 连接 outlet 和 actions。

选择 VCDecorations 视图并设置它的类为 VCDecorations。查看连接并且连接 doCancelBtn action 到视图上的取消按钮（同样在弹出列表中选择 Touch Up Inside）。拖拽 doImageBtn action 中的空白点到左上角的按钮（选择 Touch Up Inside）现在圆圈已经填充满了。拖拽填充好的点到第二个按钮，并且同样选择 Touch Up Inside。现在指定的 action 关联了多个组件。对所有的 image 按钮都重复操作这个步骤（见图 3.24）。

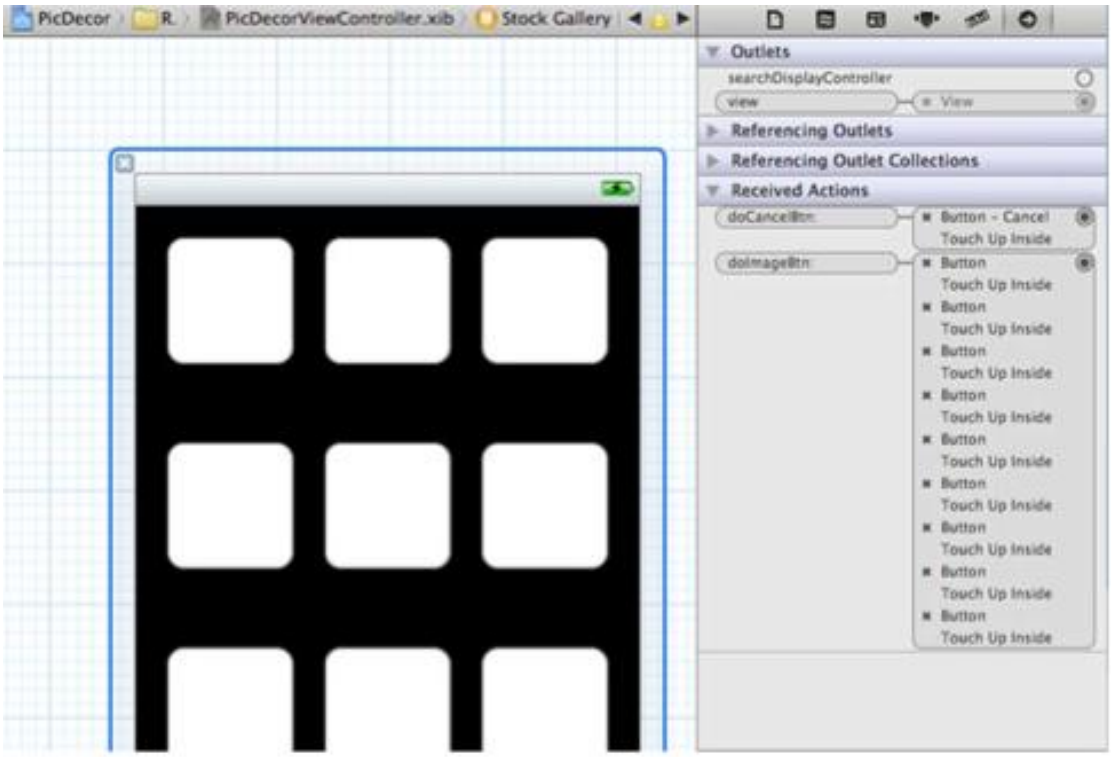


图 3.24 VCDecorations actions 连接到所有的图片按钮上

下个步骤是完善界面的外观了。我们需要图片来完成这件事。我将使用在网上找到的图片。拖拽这些文件到 Xcode 中 Resource 目录下(见图 3.25)，确保弹出窗口中点击了 Copy Items into Destination Group’ s Folder 选项。

图 3.25 拖拽图片文件到 Xcode 中项目目录的 Resource 文件夹下

在 UI 编辑器中，选择 image source selection view（有两个大按钮的那个视图）并且选择顶部的按钮。在 Attributes Inspector 中，设置类型为 Custom（见图 3.26）。这样就清除了当前按钮的外观，使之变成了黑色/空白的。从 Background setting 的下拉列表中选择 lens 的图片（例如 lens.png）。底下的按钮做同样处理——设置类型为 Custom 并选择 sunflower 图片。运行应用，看结果（见图 3.2.7）。如果你点击了其他的按钮会崩溃，因为我们还没实现对应的方法。

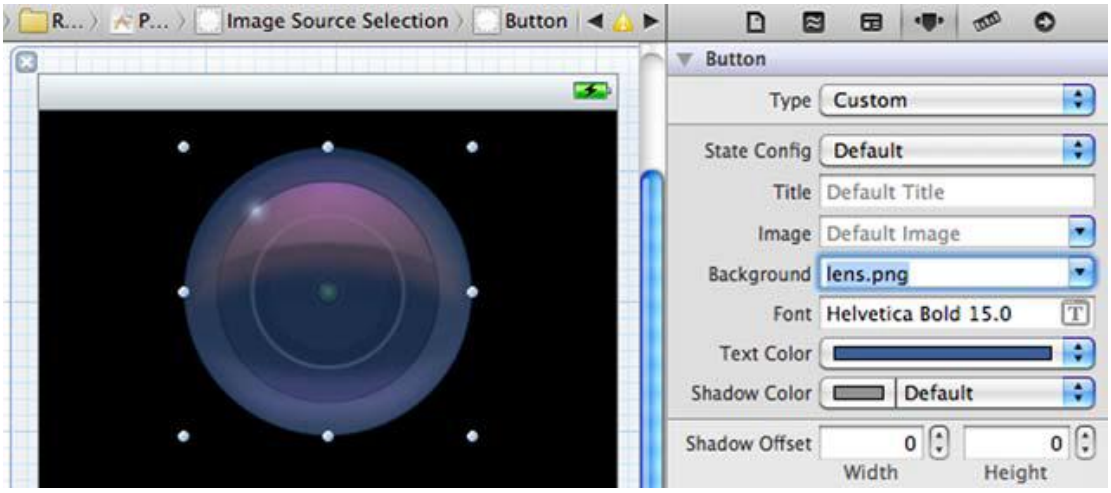


图 3.26 设置照相机按钮的类型和背景图

图 3.27 在模拟器中的图片源选择视图控制器(Image source selections view)

正如你之前对图片源选择视图做的那样，再找一些图片来美化九个装饰作用的图片按钮。设置每个按钮为 Custom 自定义类型，并且从拖拽到项目的图片中选择图片来改变按钮的背景图。

最后，需要相互关联视图控制器。在 UI 编辑器中。选择 File’ s Owner，并且拖拽 vcImageEditing 到 XIB 中的 VCImageEditing。（见图 3.2.8）

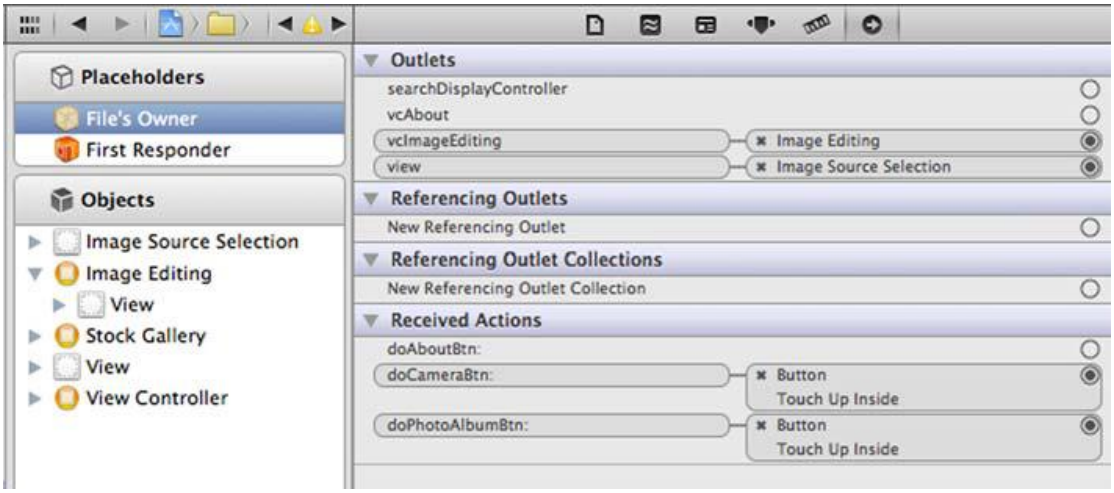


图 3.28 在 File’s Owner 中连接 vcImageEditing 到 VCImageEditing 实例

重复之前的操作连接VCDecorations视图控制器到XIB文件中的image editing实例。

或者如果你没在头文件中声明 outlets 和 actions，就仅需要按着 Control 并拖拽 Interface Builder 中的组件生成 widgets（见图 3.29）或者 events （见图 3.30）到头文件中即可。通过打开 View > Editor 菜单打开 Assistant。这样 XIB 文件跟头文件可以挨着。

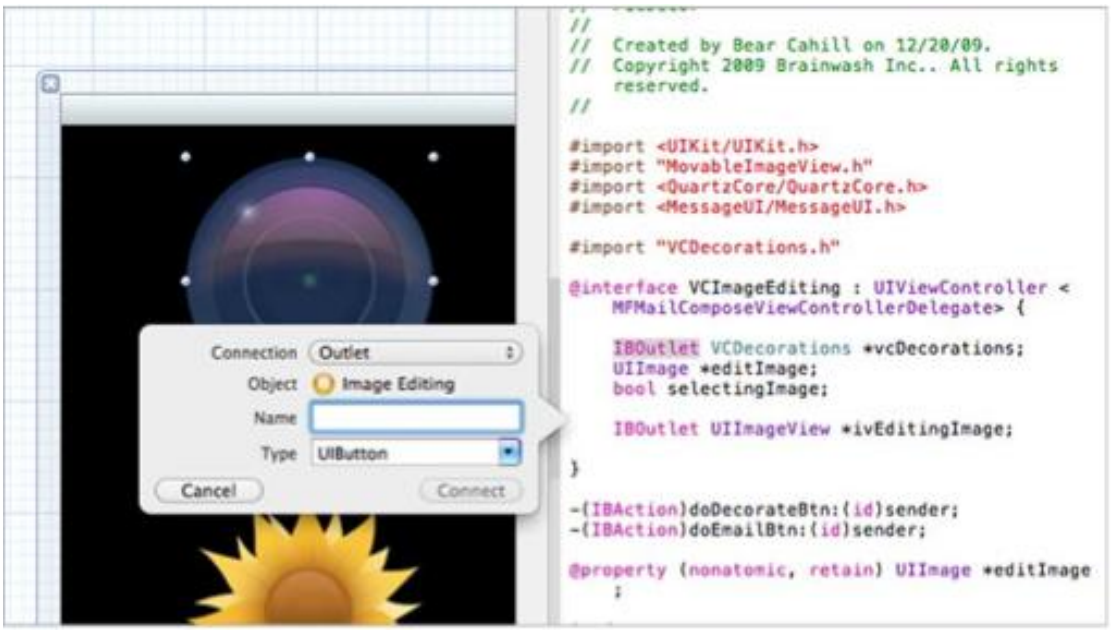


图 3.29 在 IB 中按 Control 拖拽 UI 组件到头文件中来声明 outlet

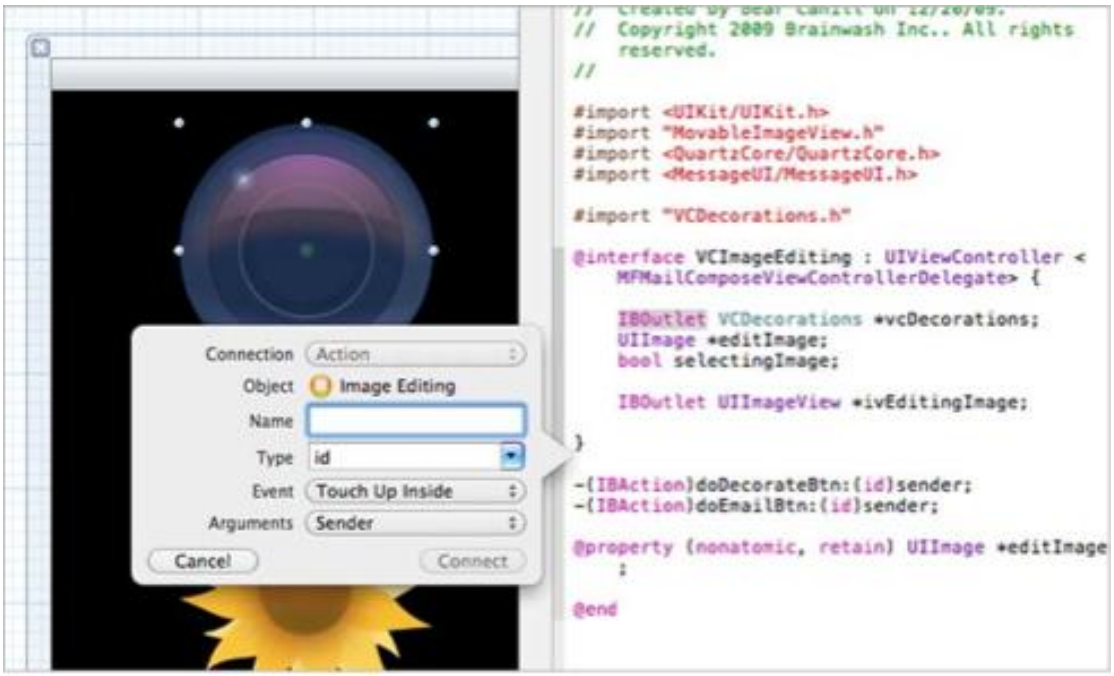


图 3.29 在 IB 中按 Control 拖拽 UI 组件到头文件中来声明 action。

目前你的 UI 已经设计完成并且关联了代码。这是一个今后你做 iOS 开发会做无数次的一个过程。不久以后，你可能在五分钟就能完成这些，并且会觉得做这些事很轻松。但是这只是冰山一角，真正的主要功能都在代码中和隐藏在 UI 后的一些东西。UI 对于用户来说是一切，重要性不言而喻。但是冰山的其余部分，隐藏在视图之后的才是我们将要继续完成的事情。

3.4 使用照相机 / 相册

目前我们已经声明了 actions 并已经把它们跟 UI 连接起来了，现在则需要实现这些方法。在某些情况下，轻触一个按钮会加载一个新的 view controller 或者表现用户的一个选择。其余情况下，action 会处理一个过程或者是连接服务器。大多数 actions 由一个界面元素来会启动执行相关的方法。

PicDecor 有很多 actions，但是这些都不复杂。因为我们已经做好连接 UI 的工作，所以让我们继续深入，实现选择图片按钮的代码。

技术 6 访问照相机 / 相册

无论何时应用想要获取图片，我们都可以使用一个简单的控制器。通过一个设置来访问照相机或相册。给应用添加这个控制器并且显示它不难实现，但它需要在应用中添加很多代码。这个控制器需要一个委托来针对用户的选择调用不同的方法。

问题

为了让用户装饰一个图片 ,需要显示这些装饰图片 ,并且可以让用户可以选择这些图片。

解决方案

要允许用户通过拍照或者从相册中获取图片源。需要实现按钮的 actions 并用正确的方式来打开图像选择器。

讨论

第一个需要定义的控制器是 PicDecorViewController，那我们就从这里开始把。我们需要实现两个按钮的 actions 来处理选择图片的逻辑。每个方法都将实例化并且显示它们的 UIImagePickerController 实例来允许用户选择一张图片。

在一个图片被选择后，UIImagePickerController 需要发通知给应用，来实现你设置的图片控制器的委托中的方法。明白这点后，我们就需要更新 PicDecorViewController 的头文件来声明它执行 UIImagePickerControllerDelegate 的协议。

```
@interface PicDecorViewController : UIViewController
<UIImagePickerControllerDelegate, UINavigationControllerDelegate> {
```

UIImagePickerController 协议 UIImagePickerController 的 setDelegate

方法需要一个实现了 PickerControllerDelegate 和 UINavigationControllerDelegate 的接口的对象。但是 UINavigationControllerDelegate 现在还没有所需的方法。

现在让我们来实现方法，从.h 文件中复制声明到.m 文件的@implementation PicDecorViewController 代码下方。唯一的区别是 doCameraBtn 方法要标明来源是照相机。（见以下代码）

PicDecorViewController 执行 action

```
-(IBAction)doCameraBtn:(id)sender;

//从.h 文件复制

{

UIImagePickerController *ipController =

[[UIImagePickerController alloc] init];

if ([[[UIDevice currentDevice] model]

rangeOfString:@"Sim"].location == NSNotFound)

//检查模拟器

[ipController setSourceType:UIImagePickerControllerSourceTypeCamera];

[ipController setDelegate:self];

[self presentModalViewController:ipController animated:YES];

}

//image picker 类
```

```
-(IBAction)doPhotoAlbumBtn:(id)sender;

//从.h 文件复制

{



UIImagePickerController *ipController =

[[UIImagePickerController alloc] init];

[ipController setDelegate:self];

//image picker 类

[self presentModalViewController:ipController animated:YES];

}
```

因为模拟器无法实现模拟照相机，所以检测 Sim 设备模型是多余的代码。如果有找到的话，它会设置照相机类型，否则会向其他方法一样使用相册。

需要告知 image editing controller 选择的图片是哪张。我们可以通过声明了一个 UIImage 成员变量——editImage ，并且把它设置成属性（property）(见以下代码)来完成告知 image editingcontrollers。不要忘记写 synthesize ,并且在.m 的其他地方 dealloc 它。

VCImageEditing 的头文件

```
#import "VCDecorations.h"
```

```
@interface VCImageEditing : UIViewController {

UIImage *editImage;

//需要编辑的图片

IBOutlet UIImageView *ivEditingImage;

}

-(IBAction)doDecorateBtn:(id)sender;

-(IBAction)doEmailBtn:(id)sender;

@property (nonatomic, retain) UIImage *editImage;

@end
```

基于 UIImagePickerControllerDelegate 接口 ,PicDecorViewController 可以执行某个方法来处理图片的选择（见以下代码）。在这个例子中，需要获取选择的图片并且把它传递给 editing image controller。然后在 editing image controller 视图上显示出来（在关掉 image picker controller 之后）。

PicDecorViewControllers 执行图片选择方法

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info

{

    [self dismissModalViewControllerAnimated:NO];

    UIImage *i =
```

[info objectForKey:UIImagePickerControllerOriginalImage]; [vcImageEditing setEditImage:i]; [self presentModalViewController:vcImageEditing animated:YES];

}

PicDecorViewController 目前处理了两个按钮的 actions 并且把选择后图片传递到 editing image controller 后并显示出来。接下来 ,我们需要实现 editing image controller 的 actions 方法。

技巧 7 模态的呈现一个视图控制器

无论是 About 页面 ,设置页面还是一个选择图片对象的列表 ,或者是许多其他的选择，使用和应用的主交互方式不同的交互方式来呈现一个视图控制器的方法很好。为用户侧面滑出的方式呈现一个视图控制器是很有用的，这不像应用中其余的部分是水平滑出来的。针对这个应用，你需要表现给用户不同的装饰图片的选择让它们来选。

问题

在你的 image editing controller 视图中 ,当按下装饰按钮的时候需要显示装饰图片的选择列表。

解决方案

需要实现之前声明过的名为 doImageBtn :的 action ,并且要模态的打开 decorations controller 视图。

讨论

在这个章节之后的部分，我们将详细讨论如何发邮件，尤其是如何把图片放到 email 中。我们先来集中精力看下选择装饰图片的 action。VCImageEditing 需要一个 VCDecorations 实例的引用，其次还需要使用一个标记——selectingImage 来判断是否已选中一个装饰图。(见以下代码)。doDecorationsBtn 方法的主要功能就是打开 decorations controller 视图控制器。

VCImageEditing 头文件以及 doDecorateBtn action 实现

```
#import "VCDecorations.h"

@interface VCImageEditing : UIViewController {

    UIImage *editImage;

    IBOutlet VCDecorations *vcDecorations;

bool selectingImage;

    IBOutlet UIImageView *ivEditingImage;

}

-(IBAction)doDecorateBtn:(id)sender;

-(IBAction)doEmailBtn:(id)sender;

@property (nonatomic, retain) UIImage *editImage;

@end

-(IBAction)doDecorateBtn:(id)sender;

{
```

```
selectingImage = YES;

    [self presentModalViewController:vcDecorations animated:YES];

}
```

类似这样模态的提交一个视图控制器 r 将会使视图从界面中由下而上滑动出来。这是个伟大的技术可以使得用户可以进行输入却不改变当前 navigation 控制器的层级结构，或者是脱离 navigation 控制器。一旦用户交互完成后，视图将会以相反的滑动方向消失。接着咱们看如何让视图控制器消失。

技巧 8 让一个模态的显示中的视图控制器消失

让视图控制器使用完后消失，重要性跟如何模态的打开它是一样的。这有可能是用户点击一个 Done 按钮触发，或者选择用户选择了它们喜欢的图片，或者是其他的 action。

问题

decorations view controller 需要储存选择的图片。可以通过设置一个变量来做到。但是还需要把它从父视图模态打开显示它的状态中销毁。

解决方案

储存选择的图片并且通过父视图控制器来销毁视图控制器。

讨论

decorations view controller 需要一个成员变量来存储选择的按钮图片以便之后调用 image editing controller 中处理（看如下代码）。

VCDecorations.h 声明selectedImage作为UIImage 实例

```
@interface VCDecorations : UIViewController {
    UIImage *selectedImage;
}
@property (nonatomic, retain) UIImage *selectedImage;
-(IBAction)doImageBtn:(id)sender;
-(IBAction)doCancelBtn:(id)sender;
@end
```

跟之前的类实现 action 类似，实现这个 action 方法。首先复制这些声明到.m 文件中。cancel 按钮就简单调用 dismiss 方法来从父视图控制器中移除掉自身（见以下代码）。图片按钮的方法则存储选择按钮的背景图作为 selected image，这个变量稍后再作处理。

VCDecorations.m 整个实现文件

```
@implementation VCDecorations
@synthesize selectedImage;
-(IBAction)doImageBtn:(id)sender;
{
    [selectedImage release];
selectedImage =
```

```
    [sender backgroundImageForState:UIControlStateNormal]; [self
dismissModalViewControllerAnimated:YES];

    }

    -(IBAction)doCancelBtn:(id)sender;

    {

        [self dismissModalViewControllerAnimated:YES];

    }

    @end
```

image editing controller 如何来获取到选择的图片呢？VCDecorations 存储图片，并且 VCEditingImage 类有 decorationsview controller 的引用（vcDecorations 这个变量）。但是如何使得 VCEditingImage 类知道一个图片已经被选择了呢？有很多种方式来解决这个问题。VCDecorations 类可以发送一个普通的通知。任何的类只要收听这个通知就会知道这个状态。或者 VCDecorations 可以定一个委托的接口让 VCEditingImage 可以实现这个接口。那么 VCDecorations 需要有一个委托属性好让 VCEditingImage 来设置这个属性为 self。图片选择后，decorations view controller 会让它的委托属性（editing image view controller）来执行它的委托方法，并传递选择的图片作为参数。

对于这个小应用来说这些方法显得有些难以处理了。我们尝试去获取的图片仅仅是一个标记而已。当 Decorations 视图中按钮被按下的时候，在 VCEditingImage 类中可以简单设置 selectingImage 为 yes。在视图控制器中可以使用标准的委托的方法来操作这个标记，并且给 selected image 赋值。那让咱们继续来看那些东西。

3.5 显示并控制图片

视图控制器的一些标准的委托方法一般用来初始化控制器的视图的某些方面。这些方法可以允许我们同时运行不同的操作。这些委托方法中最重要的有：viewDidLoad, viewWillAppear,以及 viewDidAppear。

这些方法不光可以帮助在视图加载之后，视图即将显示时，视图在屏幕上完全显示时升级 UI，还可以帮助确定在这些时间运行哪些功能。

PicDecor 可以利用调用这些方法来了解什么时候来显示被选择的图片和装饰。现在让我们来看看 PicDecor 中的视图控制器委托方法是怎么工作的。

3.5.1 进程中的交互

ViewDidLoad 方法会在视图加载后被调用。这通常是在应用启动的时候，所以要注意 performance-intensive 的操作。每次执行的时候再进行操作这种方式非常好。

在视图即将显示之前，viewWillAppear 会被调用。这个时候是更新或重置 UI 的好机会。你也许希望可以移除键盘或者初始化输入框。

视图显示之后，viewDidAppear 会被调用。这个时候是可以处理动画类的操作。

所有的这些方法调用的时候，他们的父方法也都需要被调用，需要传递参数。这个动作需要任何的代码在覆写方法执行之前做好。

技巧 9 显示选择的图片

就像之前说的一样，图片视图是 UI 中非常常见的控件。iOS SDK 让我们可以很方便的添加它们到工程中，显示图片也很方便。当一个图片视图在 UI 上时，设置图片和显示图片一样重要。不同的设定可以让图片显示的方式不同。

问题

我们需要处理两种可能性：用户自己选择一副图来编辑，拍照或是从相册中选择，或是用户选择自带的装饰图片。

解决方法

我们可以采用 viewWillAppear 方法来处理选择拍照或是从相册的图片。应用在 image selection controller 中使用图片的属性，vcDecoration 来创建 displayed image view。

同时，还可以是使用 viewWillAppear 方法来关注什么时候 VCEditingImage controller 的视图会显示。这只会在两种情况下出现：PicDecorViewController 显示的时候，还有 VCDecoration 被移除的时候。显示 VCDecoration 的时候我们可以把 selectingImage 标志设为 Yes。当 VCDecoration 被移除的时候，你可以决定是否选择一幅图。

在 viewWillAppear 方法中，你可以检查 selectingImage 标志。如果是 true，可以很容易的接触（access）到 selectedImage 的 member，使用它来创建一个 image view，然后把它添加到主 view 上。确保设置 selectingImage 标志回 NO。（见以下代码）。

VCImageEditingviewWillAppear 访问选择的图片

```
-(void)viewWillAppear:(BOOL)animated

{

[super viewWillAppear:animated];

if (editImage != nil)

{

[ivEditingImage setImage:editImage];

[self.view sendSubviewToBack:ivEditingImage];

}

if (selectingImage)

{

MovableImageView *iv =

[[MovableImageView alloc]

initWithImage:[vcDecorations selectedImage]];

[iv setUserInteractionEnabled:YES];

[self.view addSubview:iv];

}

selectingImage = NO;

}
```

注意创建的图片视图是 MovableImageView 的一个实例。这是一个自定义的类（需要在头文件导入）允许装饰图片可以随着用户手指的移动而移动。这个类起源于 ImageView，并且可以拦截触摸事件让图片随之移动（见以下代码）

MovableImageView 实现文件拦截触摸事件让图片随之移动

```
#import "MovableImageView.h"

@implementation MovableImageView

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event

{

[super touchesBegan:touches withEvent:event];

}

-(void)touchesEnded:(NSSet*)touches withEvent:(UIEvent *)event

{

[super touchesEnded:touches withEvent:event];

}

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event

{

[super touchesMoved:touches withEvent:event];

float deltaX = [[touches anyObject] locationInView:self].x

- [[touches anyObject] previousLocationInView:self].x;
```

```
float deltaY = [[touches anyObject] locationInView:self].y
- [[touches anyObject] previousLocationInView:self].y;
self.transform = CGAffineTransformTranslate(self.transform,
deltaX, deltaY);
}
@end
```

现在如果运行应用，我们可以选择图片来源（在模拟器中只能使用相册），选择一张图片，然后添加装饰（见图 3.31）。如果在模拟器的相册中没有任何图片的话，你可以在模拟器中打开 Safari，然后下载一张图片到相册。

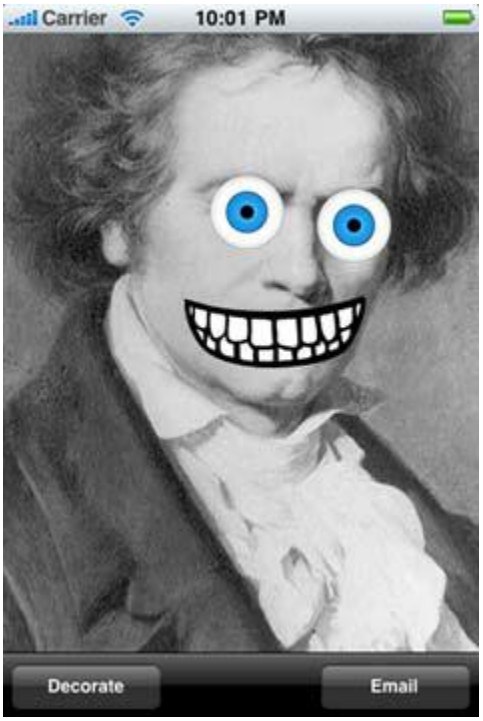


图 3.31 用装饰图装饰了的图片

技巧 10 不通过相机检测一个设备

不是所有的 iOS 设备都有相机。应用可以在有无相机时都能良好运行很重要。所以，也许需要给用户相关的信息或者选择，或者默认做某种操作。

问题

由于我们不希望在设备没有相机的时候（（比如说老版的 iPod 和 iPad）还可以选择相机选项，所以需要为无相机的设备跳过这个步骤。

解决方法

我们可以使用内置的方法来检测照相机是否可用。如果不可用 ,我们可以默认打开相册。

讨论

委托方法还有另一个用法。我们可以在第一个控制器中接入 viewDidAppear，如果用户没有照相机的话，则忽略这个请求（见以下代码）。我们可以在文件顶部定义一个静态的 Bool 命名为 startedUp。在 viewDidAppear 方法中，如果 startedUp 设置为 No，就检测相机是否可用。如果相机不可用，则自动调用 doPhotoAlbumBtn 方法。它传递 nil 作为发送器，这很好，因为这个方法这没有处理参数。

如果相机不可用则自动选择相册

```
-(void)viewDidAppear:(BOOL)animated
```

```
{
[super viewDidAppear:animated];
if (!startedUp)
if (![UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera])
[self doPhotoAlbumBtn:nil];
startedUp = YES;
}
```

现在我们已经创造了一个幅杰作，其他人如何来分享你的艺术品呢？如果有什么方法可以直接在设备上点子传输你的作品该多好！

3.6 提供电邮功能

使用 SDK 有两种方法可以用应用发送 email。一个方法是使用 mailto：URL，这样可以打开系统电子邮件。你可以指定主题，收件人等等内容。这样做来发送邮件当然可以，但是没有理由让用户离开你的应用。另一个方法是使用 email composer controller 来使用同样的邮件 UI，但是用户留在我们的应用中。

无论何时需要联网，都要好好认真考虑。关于图片的处理，你也许会想到缩小或是压缩图片来提高性能。同时，处理图片，电邮，上传，或是其他处理都需要花费时间，时间量无法确定。在这种情况下，我们需要给用户一些反馈。

使用编程的方法来缩小，压缩，给用户反馈都是来帮助我们的应用实现功能的好方法。苹果也提供了标准的框架来处理带附件的电邮。现在让我们的 PicDecor 电邮装饰图片。

技巧 11 增加 in-app 电邮

通常来说实现发送邮件的功能有几个原因。用户也许希望给应用反馈，和朋友分享 app 的 itunes 链接，发送文字信息或者附件。MessageUI 框架不仅提供了内置的 UI，同时帮你实现了很多功能。

问题

我们希望用户可以在应用内发送他们的作品给其他人。

解决方法

我们可以使用内置的 MessageUI 框架来给应用增加发送邮件功能。

讨论

发送email，VCImageEditing需要接入MFMailComposeViewControllerDelegate协议（见以下的代码片段）。在这个委托协议中只有一个方法。无论出于什么原因controller完成之后就会被调用。它的标识是mailComposeController:didFinishWithResult:error:。这个代码可以检查结果和错误然后以此继续运行。

```
@interface VCImageEditing : UIViewController
<MFMailComposeViewControllerDelegate> {
```

在应用内发送电邮的功能还需要一些其他的item。我们需要增加两个及以上的框架。在Xcode的左上角点击PicDecor工程item。然后，在顶部选择PicDecor target以及Build Phases标签。展开Link Binary with Libraries项（见图3.32）

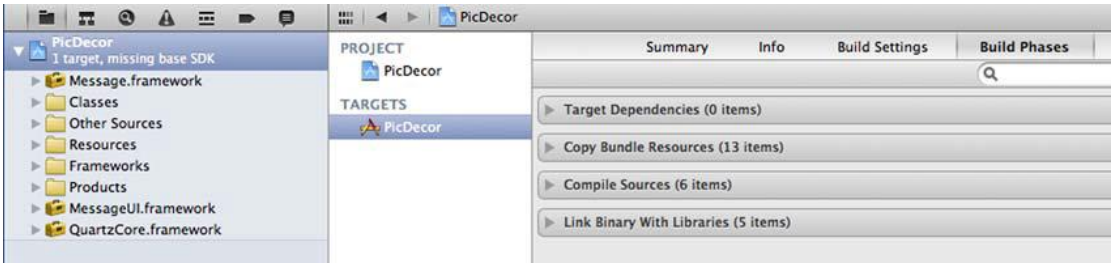


图3.32 添加框架到工程

点击+然后从表(可以在顶部输入文字来筛选列表)中选择Message。框架(见图3.33)。采取相同的方法，添加QuartzCore框架，这个框架可以用来进行view的转换以及动画类的东西。同时在，在头文件导入QuartzCore/QuartzCore.h 和MessageUI/MessageUI.h。

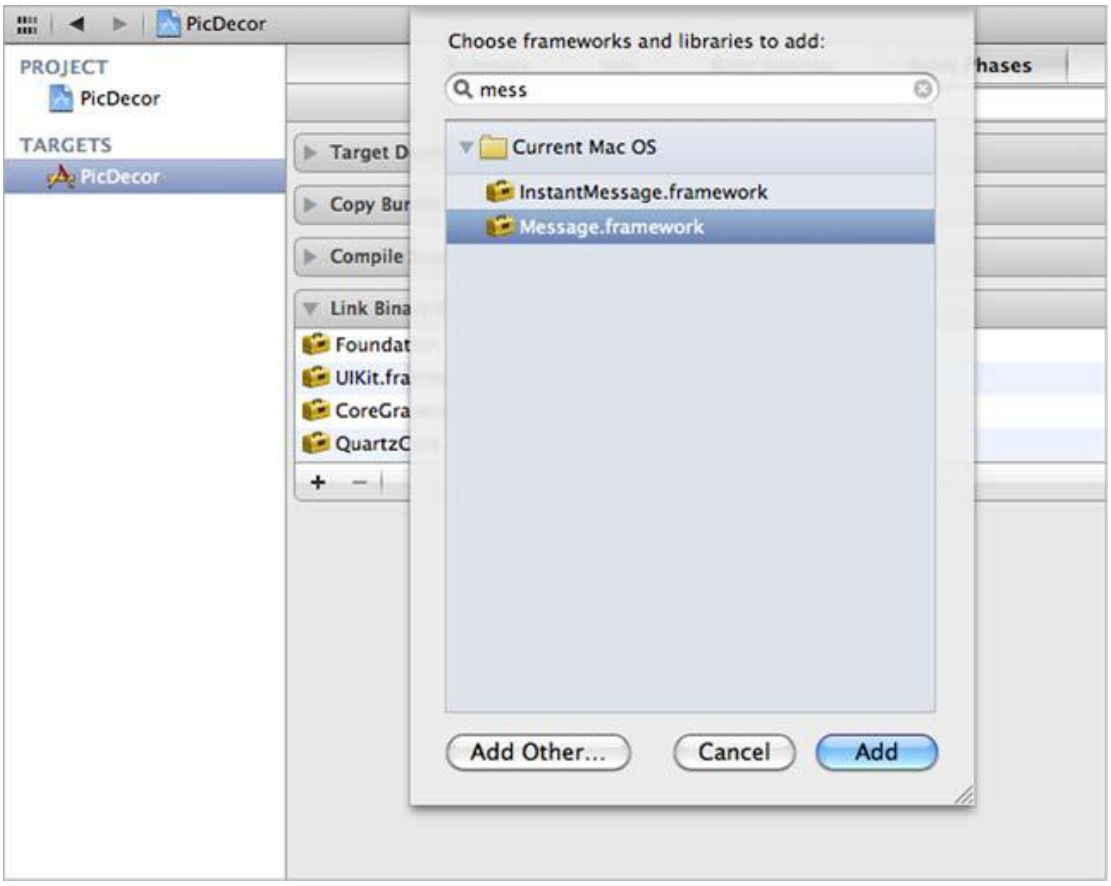


图3.33 添加MessageUI框架到Xcode工程

需要的item都有了之后，我们只需做两件事情，一是从editing image view中创建一个图片，然后把它作为附件电邮（见以下代码）

VCImageEditing通过图片视图创建一副图片并将它作为附件电邮

```
-(void)mailComposeController:(MFMailComposeViewController*)controller

didFinishWithResult:(MFMailComposeResult)result

error:(NSError*)error

{
```

```
[self dismissModalViewControllerAnimated:YES];

}

//常规检查是否有错

-(UIImage *)saveImage:(UIView *)view {

CGRect mainRect = [[UIScreen mainScreen] bounds];

UIGraphicsBeginImageContext(mainRect.size);

CGContextRef context = UIGraphicsGetCurrentContext();

[[UIColor blackColor] set];

CGContextFillRect(context, mainRect);

[view.layer renderInContext:context];

UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();

//创建一个图片

UIGraphicsEndImageContext();

return newImage;

}

-(IBAction)doEmailBtn:(id)sender;

{

MFMailComposeViewController *mailController
```

```
= [[MFMailComposeViewController alloc] init];

mailController.mailComposeDelegate = self;

//隐藏toolbar

for (UIView *v in [self.view subviews])

if ([v isKindOfClass:[UIToolbar class]])

[v setHidden:YES];

UIImage *i = [self saveImage:self.view];

//获取图片

for (UIView *v in [self.view subviews])

if ([v isKindOfClass:[UIToolbar class]])

[v setHidden:NO];

NSData *imageAsData = UIImagePNGRepresentation(i);

[mailController addAttachmentData:imageAsData

//添加图片电邮

mimeType:@"image/png" fileName:@"PicDecor.png"];

[mailController setSubject:@"My PicDecor Image"];

[self presentModalViewController:mailController animated:YES];

[mailController release];
```

}

以上的代码通过saveImage方法从编辑的图像中创建了一个新的图片。SaveImage方法通过doEmailBtn action方法调用。图片会转化为NSData，然后使用MFMailComposeViewController实例作为附件邮件。主题设定好了，message UI controller显示出来。MFMailComposeViewControllerDelegate拒绝处理controller。

注意之前的代码，saveImage方法调用来隐藏工具栏。这样做可以让工具栏不在被发送的图片上显示。另一个做到同样功能的方法是在头部声明一个IBOutlet UIToolbar *tbButtons，然后连线IB中的工具栏，在方法中设置工具栏为隐藏属性。

技巧12 缩小电邮发送的图片

无论是为了UI，还是邮件的需要，或是其他用法，缩小图片都很有用。有时候你也许会使用一张图片作为应用里的icon（比如说，让用户设置图片为个人头像）。这样的话，就没有必要保持在设备中的图片的完整大小，那么做会占用很多空间以及可能的带宽（如果需要传输或者上传的话）。

问题

在许多情况下，图片也许会很大，这时候就需要你不仅来考虑设备的联网性能，还需要考虑电池的寿命。这样的话，你会需要缩小图片。

解决方法

缩小图片尺寸只需要几行代码（见以下代码）。但是这么做在某些时候非常有用，有的时候则是必要的。

调整图片尺寸

```
+ (UIImage*)imageWithImage:(UIImage*)image
scaledToSize:(CGSize)newSize;
{
UIGraphicsBeginImageContext( newSize );
[image drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
UIImage* newImage = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return newImage;
}
```

实际上，相机照出照片的分辨率并不适配我们的图片视图。我们怎么来缩小图片呢？

另一个方法是把图片压缩成JPEG格式：

```
[UIImageJPEGRepresentation(imageObj, 0.4f)
writeToFile:self.myFilePath atomically:YES];
```

技巧13 使用activity indicator

让用户等待应用运行是非常不好的用户体验，如果在等待过程中什么反馈都没有那就更糟糕了。用户会奇怪应用怎么停止运行了，这样就会造成用户流失。所以如果不得不让用户等待必须给他们反馈。

问题

在应用进行配置的时候，我们想要给用户展示一些进程让用户知道应用正在进行配置中。这样做不光给用户提供了反馈，还可以防止用户退出应用造成用户流失。

解决方法

让我们看看如何给用户提供进程的反馈。

讨论

显示活动有两个常用的方式，一种是activity indicator（活动监控）以及network activity indicator（网络流量监控器）。

对一个标砖的activity indicator，你可以给view添加一个，然后在代码中声明IBOutletlet引用。需要的时候，就可以把它的动画打开或者设置为隐藏动画。当完成的时候，就可以把设置改回来。

设置network activity indicator，可以获取分享的UIApplication实例然后把network activity indicator设置为YES：

```
[[UIApplication sharedApplication]
setNetworkActivityIndicatorVisible:YES];
```

当应用在主线配置的时候，要防止用户对应用进行操作，所以我们需要显示activity indicator来给用户直接的反馈。这种方式的一个例子就是加载web页面的时候的显示以及刷新数据库的表单时候的显示。

Network activity indicator（见图3.34）可以让用户了解设备上的音频设施现在正在被用户在线交流。

当应用在进行另一个线程的时候非常有效。用户可能不会知道应用现在正在进行配置如果他们还可以继续在界面上操作。

图3.34 在状态栏上显示的Network activity indicator信号指示

3.7总结

现在你已经理解了基本的视图控制器，可以基于此来使用table view以及navigation controllers。现在已经知道了关于UI设计我们需要从不同角度去思考，不仅仅只是考虑功能性或只是考虑美观，需要功能与美观并重。

使用IB来做UI设计为应用打好基础。之后可以写界面相关的代码然后用IB把代码和界面连线。

PicDecor是一个很基础的应用，但它应用了iOS开发中的许多概念和基础。以后有很多开发工作都可以利用这些知识。掌握了这些知识以后，进行比较复杂的项目也可以较快上手。

通过定义outlet以及action，我们可以开发UI的交互功能。然后，通过代码，可以让这些功能实现。然后，我们考虑了用户的设备没有照相机功能（比如说老版的iPod Touch）的情况，对此作了额外的工作。这类的思考对我们设计应用非常有帮助。

4 ——通过教你制作一个上架应用Dial4来学习访问地址簿／联系人列表

本章包括：

- 主从表模板（Master-Detail applications）
- 表视图（Table View）
- 运用地址簿（address book）

在本章中，我们将会创建一个叫做Dial4的应用，此应用将会运用到iPhone的地址簿。虽然苹果不允许应用访问设备的所有信息（比如用户的电话号码），但有些在地址簿中的信息还是允许访问甚至修改的。这个功能使得应用能够显示地址簿中的联系人信息，甚至可以根据用户的操作作修改。这个功能对很多应用都有用，比如说带有根据地址寻找联系人功能的应用，或直接拨打联系人电话功能的应用。

iOS SDK提供给开发者现成的地址簿UI，其中显示了地址簿的相关内容，并且用户可以直接选择这些内容。虽然这很方便，但对于内容展示方面还是存在一定的限制。介于这个原因，在创建Dial4这个应用的时候，你将允许用户直接浏览并查看联系人列表，并把允许的用户联系人信息直接显示在表视图里。这样你就避免了显示一些禁止的内容，比如这些联系人的电子邮箱地址。用表视图显示的另一个优点则是方便用户筛选内容。

在使用地址簿框架（framework）的时候，指针的管理需要注意。使用一个从数组（array）里面来的地址簿记录时，objc_unretainedPointer将会在未传送存储空间所有权（ownership of the memory）的情况下被调用。有时，__bridge_transfer被用来进行对指针的类型强制转换（typecast）以实现对Objective-C的兼容性。

现在我们就要创建这个叫做Dial4的应用了。还在用着老式电话的时候，我喜欢只要拨出电话四位尾数就能自动拨号的功能。这样我只需要记住一堆四位数，就可以很快拨打我朋友和家人的号码

了。Dial4将会通过显示联系人列表，并且根据打上去的数字对联系人进行筛选来重现这项功能。不管是四位尾数、地域编码还是什么别的，都能用来筛选列表，而且只显示符合要求的记录。这项了不起的技术也可以应用在其他应用上。

4.1 创建带有表视图的主从表模板

用View Controller来展示一个view（视图）是非常方便的，甚至还能展示其他view来让用户进行操作。但是当用来展示层次（hierarchy）的时候，还是使用navigation controller更方便。在Dial4里面，我们将从表视图中选择一行，然后滑出一个新的view。如果你是个纯粹的iPhone用户，你就会很熟悉这种表现方式。

表视图本身非常适合层次浏览（hierarchy navigation），对于一层层打开浏览并修改资料这种用户互动方式来说，表视图是最适合不过的了。浏览表视图中的列表不需要运用到什么其他机制，这都是自动的。

我们将从新建工程这步开始。可以将其定为master-detail application模版，这样就会有一些项目是默认设定好的。然后，我们就可以专注于如何操作表视图了。

技巧14 创建主从表样板

需要在工程中用表视图的方式显示地址簿中的联系人。点击列表中的其中一项将会使另外一个view controller和其附带的view从右侧滑出。这种层次浏览方式是使用navigation controller来操作的。

问题

要创建一个显示内容的表视图以供浏览，需要选择合适的工程类型。所以现在请在Xcode里的工程类型中选择Master-Detail Application模版。

UINAVIGATIONCONTROLLER（从苹果官方说明文档中摘抄）使用navigation controller来对层次数据的展示进行管理。一个navigation controller管理自身的view的层次关系（也叫做navigation interface），这包括了一部分由navigation controller本身直接管理的view和一些你自己设定的view controller。设定的view controller负责显示各个页面的资料，而navigation controller则负责管理这些页面的跳转。

解决方案

我们将使用navigation controller作为最底层的view controller。

讨论

请在Xcode里的工程类型中选择Master-Detail Application模版。然后点击Next，将产品名定为Dial4，选择the other settings，点击Next，然后设定所在地。打开工程Resources文件夹下的Dial4MasterViewController.xib文件，我们就会看到一个表视图出现了（图4.1）。

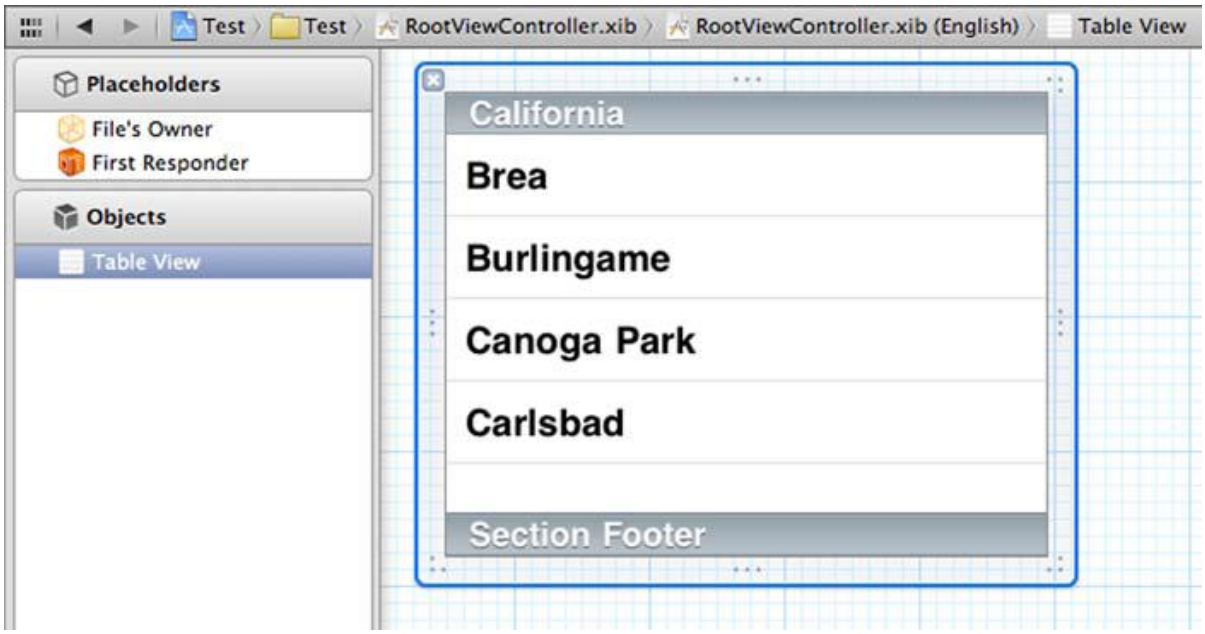


图4.1 Master-Detail Application模版默认使用表视图

其他解决方案

UIVIEWCONTROLLER（从苹果官方说明文档中摘抄）在Model View Controller（MVC）设计模板中，一个controller对象提供了可自定义的逻辑关系，比如连接view和controller，连接应用的有关数据到view等。在iOS 应用中，view controller对象就是专门用来展示和管理view的。View controller对象继承了UIViewController类，而这个类是在UIKit framework中定义的。

在第三章中，我们使用了 view controller 来制作屏幕从底部滑出的效果。对于那个工程来说

这种做法是可以的，因为其主要目的在于选择选项，浏览则相对次要。但在 Dial4 中，我们需要让用户浏览表视图并且查看里面的数据。在这种情况下，navigation controller 就更加合适了。

UITABBARCONTROLLER（从苹果官方说明文档中摘抄）使用 tab bar controller 将会给应用带来一种至多种的操作方式。一个 tab bar controller 管理自身的 view 的层次关系（也叫做 tab bar interface），这包括了一部分由 tab bar controller 本身直接管理的 view 和一些我们自己设定的 view controller。

另一种方式就是使用 tab bar application 模版。这个模版将会在屏幕下方创建一个 tab bar。它可以含有多个按钮，每个都对应一个 controller 以及相应的 view。这种效果可以在 iPhone 自带的 iPod 应用里面看到（图 4.2）。

图4.2 苹果在iPod 应用界面中使用了tab bar controller

如果没有这么多种数据需要展示，就没有必要使用tab bar controller。根据这次的情况，只需要用到一个列表显示数据，在点击后能查看更详细的数据。在这种情况下，navigation controller和table view controller是最佳选择。

即使没有navigation controller，你也可以照常显示表视图，但是你要如何展示表视图中的数据呢？从屏幕下方滑出一个view看似是一种解决方法，但是这样做看起来会很别扭。使用tab bar controller允许使用多个不同的controller，但这些controller之间并无多大关联，使得“浏览”这一主要目的不够明确。

只有navigation controller和tab bar controller能够管理和容纳多个其他controller。他们都允许用户以从下方滑出或从边上滑出的方式查看其他controller的view。

使用view controller和tab bar controller都太小题大作了，要实现点击表视图中的其中一项进入另一个controller用不着那么复杂。

4.2 使用表视图展示数据

表视图本身并不起眼，如果直接运行工程只会看到一个空的表格。它不仅需要从别处获得可供展示的数据，还需要另一个类来管理用户操作。表视图使用data source（数据源）来提供其数据以及用delegate来对用户操作做反应。

表视图使用行或者单元格（cells）来纵向显示数据。每个单元格都能显示文字或图片，并且可以让用户点击。当然有时候点击并没用，因为该数据可能仅供展示。你应该可以看到类似（>）的标示来指定这个单元格是否允许点击并展示更多东西。

表视图的类UITableView处理包括卷动、高亮、选择等功能。哪些部分是需要工程指定的呢？一般情况下只是需要展示的数据和当单元格被点击时需要作何反应。

在第二章中，我们在按钮上使用了delegate，你应该还记得delegate是对一个对象的操作进行反应的吧（比如按钮被点击）。接下来的内容就和之前做的相似，但我们还是要看看表视图是如何获取并展示数据的。

技巧14 在表视图中展示数据

表视图需要另外一个对象来为其提供供展示的数据。这个对象叫作datasource。它还需要一个delegate来处理用户点击。根据需求，datasource和delegate可以是同一个对象。在大部分情况下，我直接使用默认的view controller来作为datasource和delegate。如果你使用IB来创建表视图，IB会自动自动将表视图、datasource和delegate的关系连接到table view controller。

DATASOURCE datasource和delegate很相似。唯一的区别是与被联系的对象的关系。如果说delegate是用来联系UI的，那么datasource就是用来联系数据的。一般被联系的对象都是一个view、，比如表视图，随时与它的datasource联系并且获得需要展示的数据。一个datasource，必须像delegate一样作为一个协议（protocol）被implement。至少这个协议中需要用到的方法必须被implement。Datasource负责管理向对象view分配的内存。

问题

我们要确认默认的表视图是否已经设定了相应的datasource和delegate。Dial4MasterViewController或RootViewController都可以，选择哪一个取决于你。

解决方案

用Xcode打开默认XIB文件并查看设定是否正确。

讨论

从图上看来表视图与IB的连接已经建立了（图4.3）。而表视图的delegate和datasource都设定为了File’s Owner。查看File’s Owner的时候，你会发现它其实是RootViewController的实例（instance）。

图4.3 IB默认将表视图连接到File’s Owner

打开RootViewController.m文件，会发现当创建工程的时候，Xcode已经创立了一些针对datasource界面的默认方法（请看下面的代码）。

默认的表视图方法

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView { return 1;
}
                                                    //表视图中有几个段
// Customize the number of rows in the table view.
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section {
return 0; }                                         //每个段有多少个单元格

// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"Cell";

UITableViewCell *cell =
[tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (cell == nil) {
cell = [[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
}
    // Configure the cell.
return cell;
}                                                   //每个单元格的对应字符
```

numberOfSectionsInTableView:这个方法会返回表视图中段的数目，比如说要展示一系列汽车的数据，我们肯定需要先根据汽车的品牌分成很多个段，然后在每个段中展示该品牌的不同车型。

numberOfRowsInSection:这个方法会根据传入的段返回行数（单元格数），用之前打得比方来说，就是显示某品牌的汽车有几种车型。

cellForRowAtIndexPath:这个方法会根据传入的编号（indexPath），也就是段和行，返回对应的单元格。可以根据自己的需要自行设定表视图的相关内容以供显示。

我们需要建立一个数组来储存地址簿中需要的数据。在头文件中将其定义为NSArray的指针，并命名为myContacts。现在，在numberOfRowsInSection这个方法中就会返回myContacts这个数组的大小。

在这个数组里，我们会有所有在设备中的联系人数据的位置。确认表视图的指针是指向RootViewController的，这样表视图才能从正确的位置获得它所需要的datasource和delegate。这些数据可能有些多，我们需要决定到底要显示哪些以及如何显示。

4.2.1 苹果的单元格风格

苹果提供了4种单元格的风格（图表4.1）。默认的单元格风格只显示一行的文字。Value1和Value2单元格在左边显示标题，在右边显示内容。副标题显示两行字。最顶上的一条是粗体的，一般是用来显示标题。所有的单元格风格都在左边有内置的图片view和右边的装饰view（用来做各种标示符，按钮或者是自定义view）。

图表4.1 表视图单元格的默认风格（从苹果官方说明文档中摘抄）

UITableViewCellStyleDefault	这种风格在左边有一个居左的黑色文字标识，在右边有一个居右较小的蓝色文字标识。iPhone 自带的 Setting App（系统设置）的单元格就使用了这种风格。
UITableViewCellStyleValue1	这种风格在左边有一个居左的黑色文字标识，在右边有一个居右较小的蓝色文字标识。iPhone自带的Setting App（系统设置）的单元格就使用了这种风格。
UITableViewCellStyleValue2	这种风格在左边有一个居左的蓝色文字标识，在右边有一个居右较小的黑色文字标识。iPhone自带的Phone / Contacts App（通讯录）的单元格就使用了这种风格。
UITableViewCellStyleSubtitle	这种风格在顶部有一个居左的标识，在下面则有一个居左的较小灰色文字标识。iPhone自带的iPod App（音乐）的单元格就使用了这种风格。

就跟Contacts这个应用一样，虽然是用电话号码来对数据进行筛选，但用户更加在意的是联系人的名字。既然如此，我们只要使用默认风格的表视图就可以了。

如果希望在表视图中的单元格中显示更多数据，如联系人的地址，在这种情况下默认风格的表视图单元格就不行了，我们需要自定义一个单元格的类。

4.2.2 自定义单元格

自定义单元格是靠自定义一个新的view类来实现的。选择新建一个类，选择Objective-C Class选项并点击Next，然后将Subclass Of（父类）定义为UITableViewCell（图4.4）。

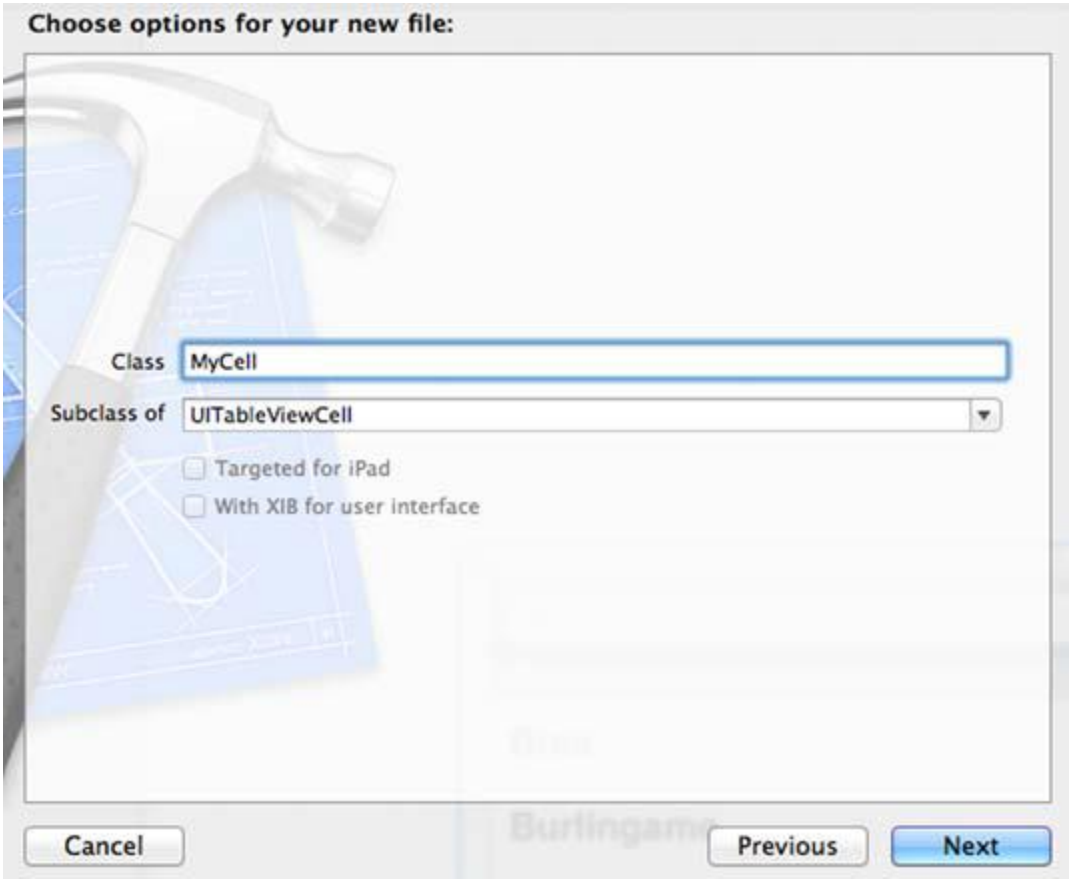


图4.4 创建一个为UITableViewCell子类的新的类

这样在工程中就会创建一个新的类，这个类接入了表视图需要的头文件。现在，在这个类的初始化方法中，就可以加入单元格需要的其他东西了。当然我们也可以创建方法来往单元格中加入新东西。

既然UITableViewCell是UIView的子类，所以我们也可以跟修改其他view的方法一样来修改它。比如加入文字显示区域、按钮、图片或者其他东西。

在这个自定义类被创建的时候，有一个setSelected方法覆盖了父类UITableViewCell的相同方法。在这可以对当用户选择这个单元格时对单元格的样子进行相应处理。不过这里并不是处理当单元格被点击时触发什么事件的地方，那是在delegate中处理的，在本次例子中，就是RootViewController。

注意如果使用Storyboard（故事板）来设计UI，表视图可以拥有一个原型单元格，而这个原型单元格是能够用Interface Builder来自定义的。

技巧14 处理被选中的表视图单元格

有些表格只是为了显示数据，但大部分表格都能让用户点击。比如点击以后会播放音乐、打开图片、跳转到某个网页等。

表视图的delegate能够对用户点击进行反应，并且用indexPath来表示被点击单元格。默认的这个方法里面什么都没有，但是注释建议了可以加入画面跳转的功能。

问题

当用户点击的时候，需要知道到底是哪行被点击了并且做出相应处理。

解决方案

使用UITableViewDelegate中的反应方法，当某行被点击时，就在那里会被探测到。而我们就可以在那个方法中处理用户点击。

讨论

除此之外还有一个方法可以使用，但在默认情况下这个方法是被注释掉的，而且方法内除了注释外什么都没有（请看下面的代码）。

在表视图delegate中默认的处理行被点击时的方法

```
// Override to support row selection in the table view.
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Navigation logic may go here.
```

```
        //                  Create and push another view controller.
        /*
<#DetailViewController#> *detailViewController =
            [[<#DetailViewController#> alloc]
                initWithNibName:@"<#Nib name#>" bundle:nil];
         // ...
         // Pass the selected object to the new view controller.
         [self.navigationController pushViewController:detailViewController
animated:YES];
         */
}
```

我们只需要做两件事情，一是取消对当前行的选择（外观变化），二是将行编号传给用来处理点击的方法（请看下面的代码）。

取消对当前行的选择，将行编号传给用来处理点击的方法

```
-(void)handleRowSelection:(int)rowIndex
{
}
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [[tableView cellForRowAtIndexPath:indexPath]
setSelected:NO animated:YES];
    [self handleRowSelection:indexPath.row];
}
```

注意handleRowSelection这个方法默认是空的，我们需要在之后会说的技巧20中写入相关内容，现在我们只需要考虑处理点击的问题。接下来我们需要从地址簿中获得要显示的数据。

4.3 使用地址簿

地址簿跟我们之前遇到的东西都不同，几乎所有地址簿中的数据都是CFTypeRef的子类，而且要使用这些数据的方法也和以前不同。

技巧14 从地址簿中获取数据

地址簿的框架提供了从其中获取数据的方法，但是要去的正确的数据还是要做点事情的。有些数据只有一个值，比如名字。但是有些时候，比如电话号码，就有可能有超过一个的值，比如家用电话、手机和公司电话等。在这些时候，获取数据的方式也会有些许差别，但这些差别直接影响到获取数据的准确性。

上面已经提到了有两种类型的数据需要获取，现在我们将一步一步实现它们，先从获取数据列表到数组中开始吧。

问题

需要从地址簿中获取数据。

解决方案

在地址簿框架中找到相应的类和方法，创建地址簿对象并从中获取联系人数据。

讨论

将地址簿的数据存进一个数组中是简单的。在获取数据并存进数组之后就可以将它存在myContacts这个对象里面（请看下面的代码）。在delegate中的numberOfSectionsInTableView方法中进行这项工作吧，这样就能保证地址簿仅在你的表视图读取完成之后才会读取。当然要注意，只有当数组为空的时候才读取，所以必须增加一个数组是否为nil的判定。

从地址簿中获取联系人数据并存入数组里

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView { if (nil == myContacts)
{
addressBook = ABAddressBookCreate();
if ([self checkAddressBookAuthorizationStatus:tableView])
myContacts = [NSArray arrayWithArray:(__bridge_transfer NSArray*)
ABAddressBookCopyArrayOfAllPeople(addressBook)];
} return 1;
}
```

首先我们必须用ABAddressBookCreate()创建一个地址簿的索引，然后检查用户是否允许App使用地址簿的内容，如果允许的话，就能够从地址簿中获取联系人数据到你的数组里面了。

用checkAddressBookAuthorizationStatus:这个方法就能够判定用户是否允许应用使用地址簿的内容。

```
-(bool)checkAddressBookAuthorizationStatus:(UITableView*)tableView; {
    ABAuthorizationStatus authStatus =
ABAddressBookGetAuthorizationStatus();
if (authStatus != kABAuthorizationStatusAuthorized)
    {
    ABAddressBookRequestAccessWithCompletion
        (addressBook, ^(bool granted, CFErrorRef error)
    {
        dispatch_async(dispatch_get_main_queue(), ^{
if (error)
NSLog(@"Error: %@", (__bridge NSError *)error);
else if (!granted) {
                    UIAlertView *av = [[UIAlertView alloc]
initWithTitle:@"Authorization Denied"
message:@"Set permissions in Settings>General>Privacy." delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"OK", nil];
[av show]; }
else {
} });
}); }
ABAddressBookRevert(addressBook);
myContacts = [NSArray arrayWithArray:
      (__bridge_transfer NSArray*)
ABAddressBookCopyArrayOfAllPeople(addressBook)]; [tableView reloadData];
return authStatus == kABAuthorizationStatusAuthorized;
}
```

首先检测当前的授权状况，如果返回值为kABAuthorizationStatusAuthorized就表示用户已经授权允许了。如果用户没有授权，那就会跳出相应的弹窗让用户授权，然后再次判断用户是否已经授权了。

如果用户没有授权 ,那么应用就会显示需要在Setting 应用里面进行授权。只有在用户授权后，地址簿中的联系人数据才会被存入数组。

注意对允许使用地址簿资料的授权和对允许使用相册、日历、提醒事项的授权都是在iOS6中才加入的，所以要注意向前和向后兼容性。

如果现在就进行编译，会发生错误。我们必须先在工程中加入AddressBook framework，并且在RootViewController的头文件中加入地址簿的头文件AddressBook/AddressBook.h。

现在就可以编译并运行了。但是你会发现还是空白一片！为什么？因为你还没有显示任何单元格。现在是时候把从地址簿中获取的数据显示出来了。

技巧14 获取地址簿中的图片

在地址簿中是允许使用图片的，比如每个联系人的照片。当然了，这也是可以获取到的数据。

我本人是一个喜欢给联系人加照片的人。虽然这不一定更好，但是他们加入到界面中总是更好的。地址簿中的大部分联系人都是带有照片的。Dial4当然必须要能够显示这些照片了，这样不仅看起来更好，而且还能让用户更容易分辨联系人。

问题

需要显示联系人的照片。

解决方案

在表视图创建单元格的方法中用获取到的图片数据创建单元格中的图片view。

讨论

我们需要为单元格设置三种属性，它们分别是名字、图片和装饰物（accessory type）。如果名字不存在就显示电话号码，如果图片不存在，那就直接不显示图片，而装饰物则与其他东西没有关系。

现在我们已经有了地址簿中的联系人数据，现在就用这些数据来设置单元格吧。首先可以将装饰物属性设置为UITableViewCellAccessoryNone，也就是空白。苹果提供了四种装饰物属性可供使用（图表4.2）。

图表4.2 苹果提供的装饰物属性（从苹果官方说明文档中摘抄）

UITableViewCellAccessoryNone	默认装饰物属性，就是空白。
UITableViewCellAccessoryDisclosureIndicator	单元格会有一个箭头状的装饰物，一般是用来提示用户还有扩展内容用的，无法点击。
UITableViewCellAccessoryDetailDisclosureButton	单元格会有一个蓝色箭头装饰物，一般是用来提示用户设定各种属性的，可以点击。
UITableViewCellAccessoryCheckmark	单元格的右侧会有一个不可点击的打勾符号。表视图的delegate能够在tableView:didSelectRowAtIndexPath:方法中对这个打勾符号进行管理。

设置单元格显示联络人的名字

```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath {
static NSString *CellIdentifier = @"Cell";
UITableViewCell *cell =
[tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (cell == nil) {
cell = [[UITableViewCell alloc]
        initWithStyle:UITableViewCellStyleDefault                    //只有一行的风格
reuseIdentifier:CellIdentifier];
[cell setAccessoryType:UITableViewCellAccessoryNone]; }
        // Configure the cell.
    [[cell textLabel]                                                //设置单元格的标
题文字
setText:[self personDisplayText:
                [myContact objectAtIndex:indexPath.row]]];
return cell;
}
```

接下来就是设置图片，设置名字的代码之后，读取图片数据，然后转换为UIImage类型，接着在单元格种默认的imageView中设置它（请看下面的代码）。

获取图片数据并在单元格中显示

```
NSData *d =
        (NSData*)ABPersonCopyImageData(
                [myContacts objectAtIndex:indexPath.row]);
if (nil != d) {
        UIImage *i = [UIImage imageWithData:d];
        [[cell imageView] setImage:i];
    }
else
        [[cell imageView] setImage:nil];
```

获取联系人的名字很简单，但是你还需要考虑到两件事情：如果这个联系人没有设定名字怎么办~~？~~如果没有电话号码怎么办？获取电话号码的方式是不同的，因为一个联系人可能有几个不同号码，是以组形式存在的。

技巧14 获取地址簿中以组形式存在的数据

像电话号码和电子邮件这种数据都是以组形式存在的，因为一个联系人通常都有超过一个电话号码或者电子邮箱。

我们必须以组的形式来获取这些数据。当获得了其中一组数据的时候，可以用编号来获取其中的任何一项数据。

问题

需要获取有用的数据来显示联系人。

解决方案

我们需要获取联系人的姓和名，并且根据用户的偏好来显示它。比如亚洲人的习惯是姓氏显示在名的前面，而西方人则相反。如果既没有姓也没有名，那就获取电话号码来表示这个联系人。

讨论

要获取这些以组形式存在的数据，首先要将他们复制到ABMultiValueRef中（请看下面的代码中的第三个注明）。当把组形式的数据复制到ABMultiValueRef后，就可以把其中任何一项数据用相应编号复制出来了。使用ABMultiValueGetCount方法可以知道有多少个项目在一个组里面。

用来决定用什么字来表现联系人的方法

```
-(NSString*)personDisplayText:(ABRecordRef)person {
    NSString *firstName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonFirstNameProperty);
    NSString *lastName = (__bridge_transfer NSString *)
ABRecordCopyValue(person, kABPersonLastNameProperty);
NSString *fullName = nil;
if (firstName || lastName)
  if (ABPersonGetCompositeNameFormat() ==
kABPersonCompositeNameFormatFirstNameFirst)      //根据用户的偏好排列姓和名的显示顺序
fullName = [NSString stringWithFormat:@"%@ %@",
          firstName, lastName];                          //当作姓和名都是存在的
else
fullName = [NSString stringWithFormat:@"%@, %@",
        lastName, firstName];
                                                  //没有名字的话就使用第一个电话号码
          ABMultiValueRef phoneNumbers = ABRecordCopyValue(person,
kABPersonPhoneProperty);
if (phoneNumbers &&
ABMultiValueGetCount(phoneNumbers) > 0)
          {
fullName = (__bridge_transfer NSString*)
ABMultiValueCopyValueAtIndex(phoneNumbers, 0);
CFRelease(phoneNumbers);
return fullName;
}
```

除了电话号码外，另外两个数据也是以组存在的，那就是地址和电子邮箱，它们对应的值分别是kABPersonEmailProperty和kABPersonAddressProperty。这些值也附带了相应标识，比如家庭、工作、或者其他什么的。如果需要其中特定的某一个，可以对ABMultiValueRef使用

ABMultiValueCopyLabelAtIndex方法，只需要传入编号，就会返回相应位置数据的标识。而这个标识可以用来和kABWorkLabel、kABHomeLabel以及kABOtherLabel进行比较（请看下面的代码）。当然这个例子的主要目的是为了教你如何从组数据中找出某项数据，在这个应用中用不着。

从几个不同的电子邮箱地址中找出工作的电子邮箱地址

```
NSString *retVal = nil;
    ABRecordRef person = [myContacts objectAtIndex:rowIndex];
    ABMultiValueRef vals =
ABRecordCopyValue(person, kABPersonEmailProperty);
if (ABMultiValueGetCount(vals) > 0)
    {
        CFIndex i;
for (i=0; i < ABMultiValueGetCount(vals); i++)
        {
CFStringRef label = ABMultiValueCopyLabelAtIndex(vals, i);
if (retVal == nil ||
CFStringCompare(label, kABWorkLabel, 0)
      == kCFCompareEqualTo)                    //考虑用户对姓和名顺序的偏好
 {
    } }
    CFStringRef val = ABMultiValueCopyValueAtIndex(vals, i);
retVal = (NSString *)val;
CFRelease(val);
 }
CFRelease(label);
CFRelease(vals);
return retVal;
```

现在我们有联系人的名字，图片和电话号码了，运行这个应用终于能看到些有意义的东西了（图4.5）。

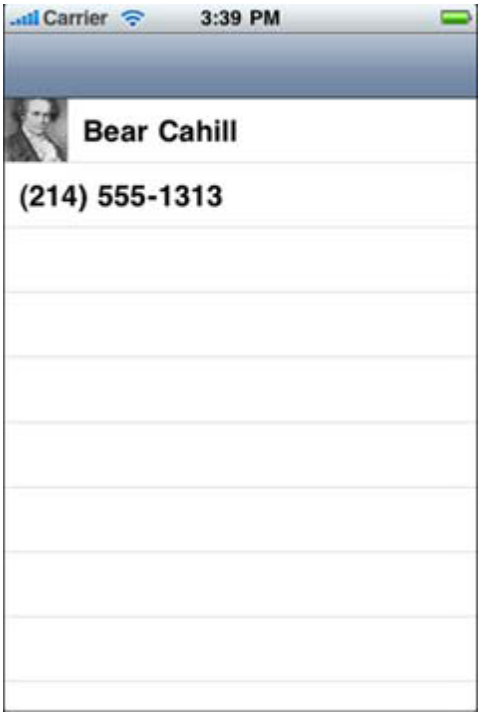


图4.5 运行Dial4可以看到联系人数据已经显示出来了

因为handleRowSelection方法还是空的，所以即使点击了也不会有什么反应。但那是必要功能，所以让我们继续吧。

技巧14 加入打电话功能

Dial4是一个主要在iPhone上使用的应用，肯定要有打电话的功能。很多同类的应用也都有这样的功能，比如联系售后服务或者订餐。而在这里，主要是从拨打电话给地址簿中的某个联系人。

从程序角度来说，用iPhone打电话是超级简单的事情。有好几个URL都是SDK自动支持的。比如tel：是用来打电话的，mailto：是用来发送邮件的，sms：是用来发送短信的，http：是用来访问网站的，map：是用来打开地图的。

问题

需要让用户在点击列表上的联系人后就打电话给该联系人。

解决方案

只要有用户点击了表视图中的单元格，handleRowSelection这个功能就会被自动调用。这个功能至今还是空的，现在就让我们在这里实现一些功能。当用户点击联系人的时候就打电话给该联系人，如果该联系人有多个号码，则让用户选择需要拨打的号码。

讨论

要使用打电话功能，需要创建一个带有电话号码的URL，并且跳转到这个URL。可以使用之前获得的电话号码并做成URL。

NSString *url = [NSString stringWithFormat:@"tel:%@", phoneNum]; [[UIApplication sharedApplication] openURL:[NSURL URLWithString:url]];

这样打电话的功能就实现了。但是当联系人拥有多个号码的时候该如何处理呢？这时候就必须让用户选择要拨打的电话。我们可以遍历数据中的电话号码，并且将它们存在数组里面，然后跳出警告窗口（alert view）显示这些号码，并且让用户选择拨打其中一个号码。

要实现这个功能就需要查看有几个可用的电话号码。如果只有一个，那就直接创建URL拨出电话。如果超过一个，那就创建警告窗口，把每个号码都做成按钮，然后让这个警告窗口跳出来。你还需要一个delegate来监测到底是哪个按钮被用户按下了（请看下面的代码）。

获取电话号码并拨出

```
-(void)callThisNumber:(NSString*)phoneNum
{
NSString *url = [NSString stringWithFormat:@"tel:%@", phoneNum]; [[UIApplication sharedApplication]
openURL:[NSURL URLWithString:url]];
}
-(void)handleRowSelection:(int)rowIndex                    //将自己设为delegate
{
otherButtonTitles:nil];                                    //将按钮文字设置为nil
for (int i=0; i < ABMultiValueGetCount(phoneNumbers); i++)
        [av addButtonWithTitle:(__bridge_transfer NSString*)
            ABMultiValueCopyValueAtIndex(phoneNumbers, i)];//将每个电话号码做成按钮

if (phoneNumbers)
CFRelease(phoneNumbers);
}
- (void)alertView:(UIAlertView *)alertView
```

```
clickedButtonAtIndex:(NSInteger)buttonIndex
{
}
    if (buttonIndex > 0)                                    //0就代表取消了
        [self callThisNumber:
[alertView buttonTitleAtIndex:buttonIndex]];                //使用按钮上的文字
```

现在就有让用户选择要拨的电话的功能了（图 4.6）。

图 4.6 让用户选择要拨打的电话

和往常一样，这看起来并不复杂的操作却需要经过好多困难的步骤，比如查找地址簿，查找要显示的名字和让用户选择需要拨打的号码。

技巧 14 显示地址簿的详细信息

我们应该允许用户查看地址簿中的详细数据。一般用户都认为当点击联系人的时候会显示联系人的详细信息，但是这个应用的目的是为了让用户快速拨打电话而不需要经过任何多余步骤，那这

该如何做到呢？

问题

我们要在表视图中显示联系人的详细信息。也就是说当用户点击某处的时候这行会变得有所不同。

解决方案

在屏幕顶部的导航栏加入一个切换按钮，让用户可以选择显示模式。当这个按钮被点击后，点击联系人就会出详细信息，再点一次该联系人才进入拨打电话的阶段。

讨论

让我们在导航栏的左上角加入一个按钮，只要这个按钮被按下，用户点击联系人就会出联系人的详细信息。我们需要创建一个按钮，以及其会触发的方法，和一个 view controller 来显示用户的详细资料。

用 Navigation Item 设定这么一个按钮以切换显示模式是很直观的。按钮需要用来切换显示模式，Display 和 Call（请看下面的代码）。在 viewDidLoad 方法中去掉（/*）和（*/）两个注释符号，然后加入 doCallDisplayBtn 方法。在这个方法中传入 nil，按钮的文字就会变为 Display。

按钮文字在 Call 和 Display 间切换

```
-(void)doCallDisplayBtn:(id)sender
{
    NSString *bbiTitle = @"Display";
if (nil != sender && [[sender title] compare:@"Display"]
                                                    == NSOrderedSame)
bbiTitle = @"Call";
UIBarButtonItem *bbi = [[UIBarButtonItem alloc] initWithTitle:bbiTitle
style:UIBarButtonItemStyleBordered target:self action:@selector(doCallDisplayBtn:)];
    self.navigationItem.leftBarButtonItem = bbi;
}
- (void)viewDidLoad {
    [super viewDidLoad];
    [self setTitle:@"Dial4"];
    self.navigationItem.rightBarButtonItem = self.editButtonItem;
```

```
    [self doCallDisplayBtn:nil];
}
```

在 viewDidLoad 方法中进行 self.navigationItem.rightBarButtonItem = self.editButtonItem 就允许了对表视图行的编辑功能。对 self 调用 setTitle 则可以让应用的名字显示在导航栏上。

运行应用，确认一下切换 Call 和 Display 按钮的功能。还可以按 Edit 按钮试试看。

现在我们需要一个 controller 来显示详细资料。在 Xcode 中右点击 Classes 选项，然后选择 Add>New File。在出现的窗口中将这个新建的类的 Subclass of 定为 UITableViewController（图 4.7），然后将这个类命名为 TVCDetails，再点击 Next 创建这个类和其相关文件。

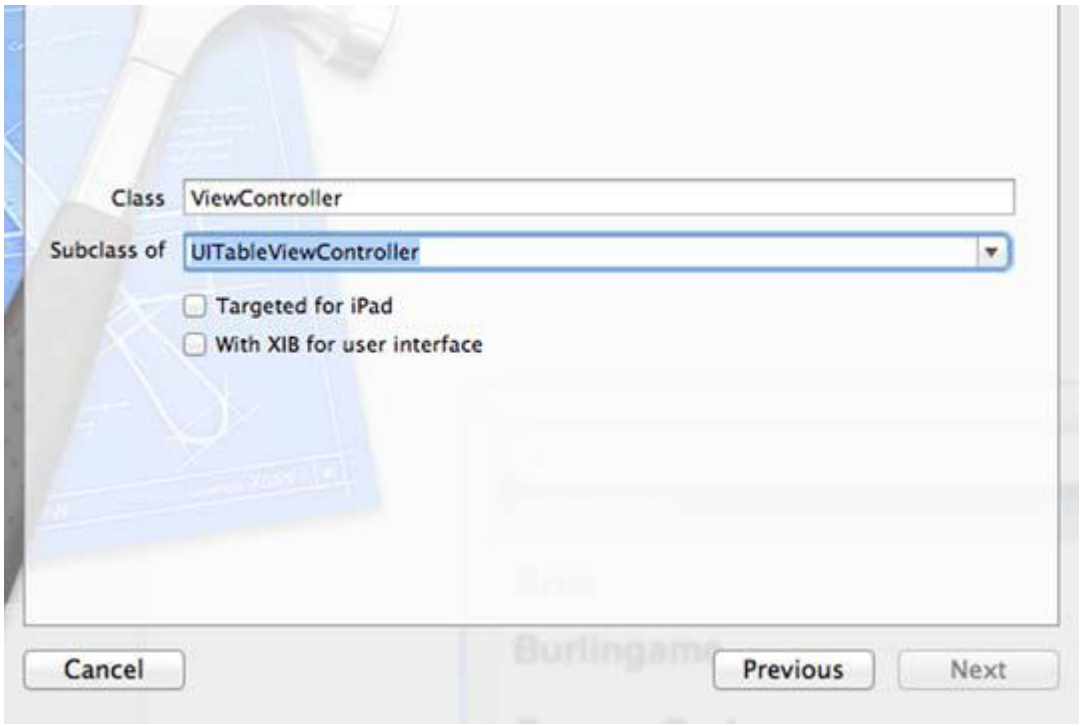


图 4.7 创建一个新的表视图 controller

在这个新建的类的头文件里面，定义一个类型为 ABRecordRef 的变量，并命名为 person（请看下面的代码）。当然，不要忘记在.m 文件中 synthesize 这个变量并用 CFRelease 来删除这个变量。

新表视图类的创建和设定

```
-(void)callThisNumber:(NSString*)phoneNum
{
NSString *url = [NSString stringWithFormat:@"tel:%@", phoneNum]; [[UIApplication
sharedApplication]
}
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
    ABMultiValueRef phoneNumbers = ABRecordCopyValue(person,
kABPersonPhoneProperty);
int retNum = ABMultiValueGetCount(phoneNumbers)+2;    被名字占用的两行也要算进去
CFRelease(phoneNumbers);
return retNum;
}
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"DetailsCell";
openURL:[NSURL URLWithString:url]];
UITableViewCell *cell =
[tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (cell == nil) {
cell = [[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleValue2
    reuseIdentifier:CellIdentifier];                    //Value2 风格在这里看起来会更舒
服
}
if (indexPath.row < 2)
[cell setSelectionStyle:UITableViewCellSelectionStyleNone];)        //名字不可被点击
else
[cell setSelectionStyle:UITableViewCellSelectionStyleBlue];
NSString *title;
NSString *text;
switch (indexPath.row)
```

```
{
case 0:
title = @"First Name";
text =              (__bridge_transfer NSString *)ABRecordCopyValue(person,
kABPersonFirstNameProperty);
break;
case 1:
title = @"Last Name";
text =
            (__bridge_transfer NSString *)ABRecordCopyValue(person,
kABPersonLastNameProperty);
break;
default:
title = @"Phone";
      ABMultiValueRef phoneNumbers = ABRecordCopyValue(person,
kABPersonPhoneProperty);
if (phoneNumbers &&
ABMultiValueGetCount(phoneNumbers) > 0)
{
text =
        (__bridge_transfer NSString*)ABMultiValueCopyValueAtIndex
                              (phoneNumbers, indexPath.row-2);       //记得减去被名字占用的两行
CFRelease(phoneNumbers);
} break;
      }
      [[cell textLabel] setText:title];
      [[cell detailTextLabel] setText:text];
return cell;
}
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
if (indexPath.row > 1)                                          //仅处理显示电话号码的行
 }
```

在 viewDidLoad 方法中设定名称的方法和之前是一样的。现在只需要在 RootViewController

中进行控制。在 RootViewController 的头文件中 import（导入）TVCDetails.h 文件。然后在 Navigation Item 按钮的文字上做一个简单的判定来告诉用户其作用（请看下面的代码）。

以导航栏按钮为基础在 RootViewController 中处理对行的点击

```
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    [[tableView cellForRowAtIndexPath:indexPath]
setSelected:NO animated:YES];
if ([self.navigationItem.leftBarButtonItem.title compare:@"Call"]!= NSOrderedSame)
        [self handleRowSelection:indexPath.row];
else
    {
        TVCDetails *tvc = [[TVCDetails alloc]
initWithStyle:UITableViewStyleGrouped];
        [tvc setPerson:objc_unretainedPointer(
                                [myContacts objectAtIndex:indexPath.row])];
        [self.navigationController
pushViewController:tvc animated:YES];
} }
```

注意我们需要用 UITableViewStyleGrouped 对 TVCDetails 进行初始化，这样就可以在行数不够多的时候避免多余的空白行。运行这个应用，点击 Display 按钮，然后再点击一个联系人，会发现显示出了这个联系人的详细信息（图 4.8）。

图 4.8 在表视图中显示地址簿联系人的详细信息

你可能会在想，在 UI 编辑器中何时能使用这个类。其实不需要这样，当创建 UITableViewController 的时候就会自动创建一个这个 controller 附带的表视图。而且这个表视图默认将这个类作为其 datasource 和 delegate。因为是 RootViewController 的实例，所以不需要为其连接任何 IB。同样的，由于你没有任何按钮或开关需要处理的，所以你不需要使用 IB（注意这不是你不能加按钮的意思）。

在表视图中显示数据是很好的展示方法，而且这也允许了用户一层层查看更加详细的数据。表视图本身也能够对数据进行管理，现在就试试看吧。

4.4 管理表格数据

表视图允许用户管理其表格上的数据。用户可以插入，改编排列顺序，以及删除行。删除是最常见的编辑功能，而你也会发现默认风格就有给你一个 Edit 按钮。

和之前说的一样，触摸屏是可以让数据被直接操作的。用户滑动屏幕就可以拖动表格而不需要用到什么别的方式。同样的，表视图本来就是能让用户在上面直接管理数据的。

技巧 14 删除和重新排列表视图的行

VCDetails 这个类用在行上本身就具有方法来监测编辑功能。搜索（Command + F）

commitEditingStyle 这个方法并且取消对其的注释。默认的编辑功能都是删除，也就是 UITableViewCellEditingStyleDelete,这就会删除相应 indexPath 的行。

运行这个应用，点击 Edit 按钮，试着删除一行。出错闪退！这是意料中的事情。查看 Debugger Console（Command + Shift + R），就会发现出错信息，在行被删除之后行数出错了。这是因为实际上并没有删除掉什么东西。

从表视图中删除一行只是表观上的删除，你还必须从数据上将其删除。大部分情况下，用来显示的数据都存在一个数组里面，我们需要从这个数组里面删除相应的数据。只要显示的顺序和数组里的一样，这个数据在一般在 indexPath.row 那里。

问题

需要从表视图中删除数据，而且必须是确实删除了。

解决方案

用监测表视图编辑的方法来处理删除数据时的行为。

讨论

要从 datasource 里面删除数据你就必须从你的 myContacts 数组里面将其删除掉。更重要的是，你需要将其从地址簿中删除掉。然后表视图将会调用相应方法刷新数据（请看下面的代码）。你将会设置 myContacts 数组为 nil ,这样当表视图中的内容发生改变的时候这个数组也会被更新。

在表视图和地址簿中删除数据

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath
{
if (editingStyle == UITableViewCellEditingStyleDelete) {
ABRecordRef person = objc_unretainedPointer([myContacts objectAtIndex:indexPath.row]);
CFErrorRef *error;
ABAddressBookRemoveRecord(addressBook, person, error);
ABAddressBookSave(addressBook, error);                    //保存更改
myContacts = nil;                                         //将数组设定为 nil 以刷新
[tableView deleteRowsAtIndexPaths:[NSArray
arrayWithObject:indexPath]
```

```
withRowAnimation:UITableViewRowAnimationFade];
}}
```

每种编辑类型都有其所述的作用，有三种编辑类型可供选择（图表 4.3）。

图表 4.3 表视图单元格可选择的编辑类型

UITableViewCellEditingStyleNone	这个单元格不能被编辑，这个是默认类型。
UITableViewCellEditingStyleDelete	这个单元格有删除选项，这是一个红色圆中间有个减号的标志。
UITableViewCellEditingStyleInsert	这个单元格有插入选项，这是一个绿色圈中间有个加号的标志。

重新排列单元格是用不同的方法来实现的。要让单元格允许被重新排列，我们需要 implement datasource 的相关方法，看看重新排列是否被允许和进行（请看下面的代码）。

在表视图 datasource 中默认的重新排列顺序的方法

```
- (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(NSIndexPath *)fromIndexPath
toIndexPath:(NSIndexPath *)toIndexPath {
}
- (BOOL)tableView:(UITableView *)tableView
canMoveRowAtIndexPath:(NSIndexPath *)indexPath
{
return YES;
}
```

当你运行应用并且点击 Edit 按钮的时候，你就会看到一个重新排列的标志甚至还可以使用其对行进行重新排列的操作（图 4.9）。由于你并没有在重新排列的方法中做什么（请看上面的代码），只要你退出应用，新的排列就会失效。

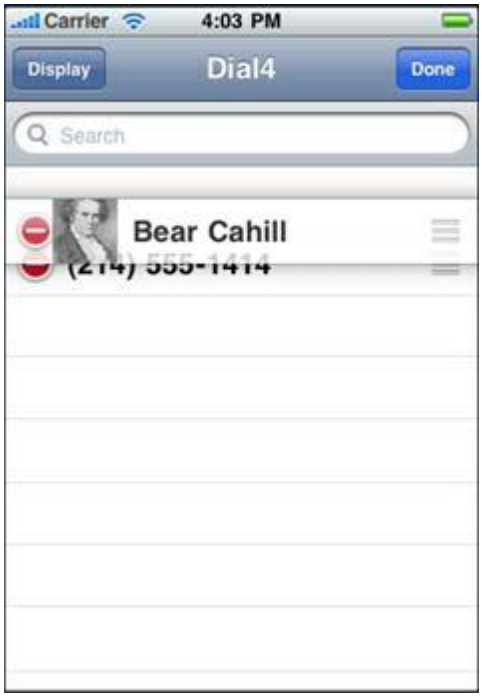


图 4.9 在行右侧的重新排列按钮让你可以重新排列表视图的行

将 tableView:canMoveRowAtIndexPath:方法的返回值设为 NO 就会禁用重新排列功能，其按钮也不会出现了。重新排列作为一个锦上添花的功能，它可以让用户更快找到需要的东西。还有另外一种方法就是搜索和筛选。

技巧 14 筛选表视图中的数据

这个应用的最初目的就是让用户只输入电话的四位尾数就能查找到相应号码。要实现这项功能，首先需要加一个输入框。

这个应用的目的就是让用户更容易找到号码，这时还需要用到筛选功能。用户在输入号码的时候筛选功能也同时工作，筛选功能并不只对 4 位尾数有效，对地区编码也适用。总而言之，就是在用户输入号码的同时筛选出符合特征的相应号码。

问题

让用户进行输入，并且根据输入显示相应数据。

解决方案

你需要为工程加入一个文本框，并根据用户的输入筛选出符合条件的联系人。

讨论

在 UI 编辑器中，在你的表视图顶部加入一个 Search Bar（搜索栏）和一个 Search Display

Controller。你会发现其 delegate 已经自动连接到了 File’ s Owner（图 4.10）。

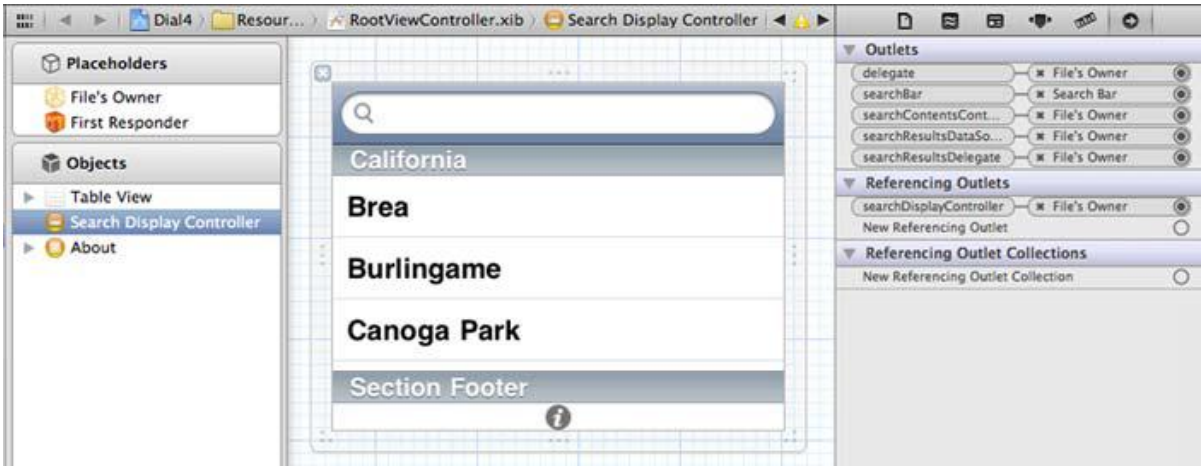


图 4.10 在加入 Search Bar 的时候 Search Bar 和 Search Display Controller 自动设定了连接

由于我们目标是筛选电话号码，所以要将键盘类型（Keyboard type）设置为 Number Pad（图 4.11）。

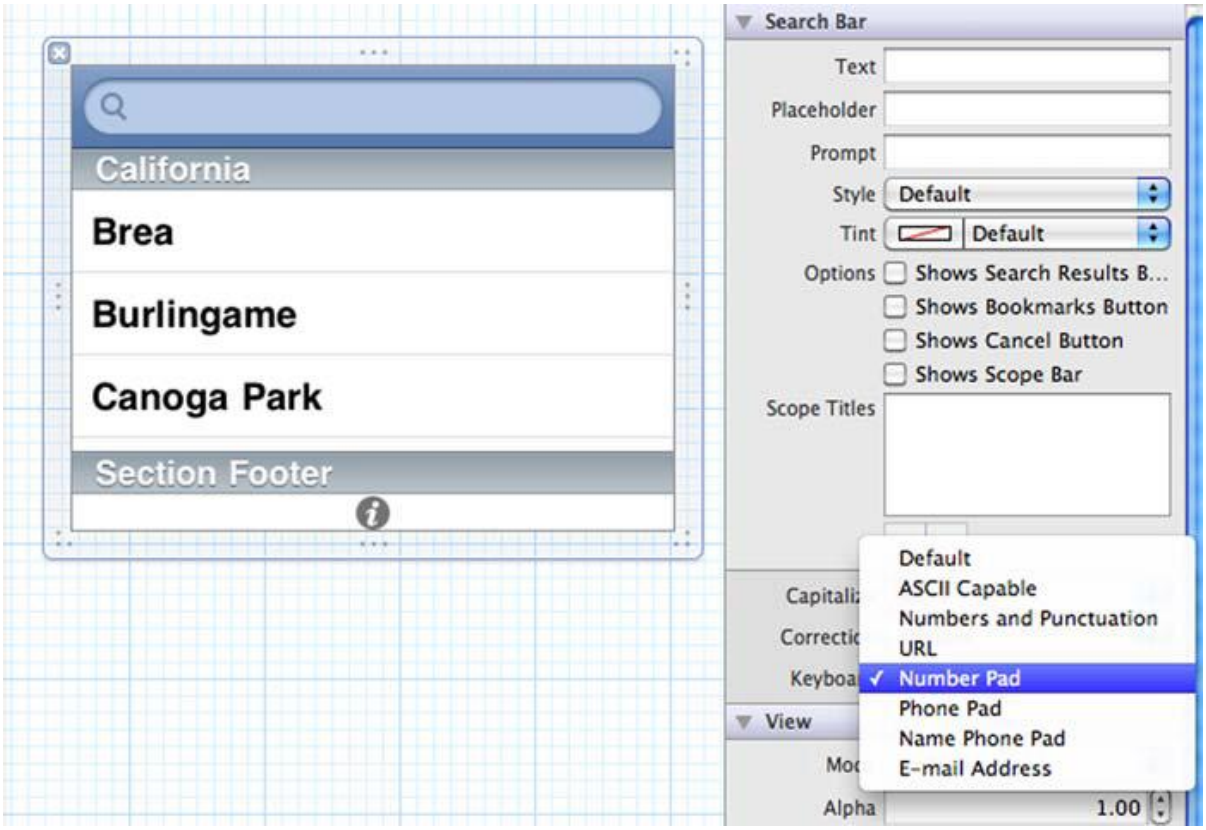


图 4.11 将 Search Bar 的键盘类型设置为 Number Pad。

做完了这步后，我们必须在 RootViewController 中加入两个方法。第一个方法用来在用户输入的时候筛选列表（请看下面的代码）。这创建了一个筛选数组并且只往里加入符合用户输入的地址簿数据。

根据用户输入对地址簿数据进行筛选

```
- (void)searchBar:(UISearchBar *)theSearchBar
textDidChange:(NSString *)searchText
{
if ([searchText length] < prevSearchTextLen &&
prevSearchTextLen != 0)                                    //重置数值以获得完整列表
        {
myContacts =
return; }
[NSArray arrayWithArray:(__bridge_transfer NSArray*)
ABAddressBookCopyArrayOfAllPeople(addressBook)];
prevSearchTextLen = [searchText length];                   //获得输入文本的长度
NSMutableArray *filteredContacts =
```

```
                                        [NSMutableArray arrayWithCapacity:10];
for (int i=0; i < [myContacts count]; i++)
{
    ABMultiValueRef phoneNumbers = ABRecordCopyValue(
        objc_unretainedPointer([myContacts objectAtIndex:i]),
kABPersonPhoneProperty);

if (phoneNumbers &&
{
for (int j=0;
ABMultiValueGetCount(phoneNumbers) > 0)
    j <ABMultiValueGetCount(phoneNumbers);
j++) {
        NSString *phNum = (__bridge_transfer NSString*)
ABMultiValueCopyValueAtIndex(phoneNumbers, j);
if ([phNum rangeOfString:searchText].location
                                                    != NSNotFound)
        {
            [filteredContacts addObject:[myContacts
objectAtIndex:i]];
j = ABMultiValueGetCount(phoneNumbers);                    //找到一个就将 j 设为 count
跳出循环
} }
CFRelease(phoneNumbers);
                    }
}
myContacts = [NSArray arrayWithArray:filteredContacts];
        }
```

将经筛选的数组复制到 myContacts 里面，这样表视图就会只显示你筛选过的数据（图 4.12）

图 4.12 在用户输入的同时动态更新联系人列表

现在这个应用的功能已经实现了。可以显示联系人列表，根据用户输入的号码筛选联系人，打电话。还可以使用切换按钮切换是否显示联系人详细信息的显示模式。这是 navigation controller 一个很好的运用范例。

制作这么一个应用涵盖了很多东西，还有足够数量的可动元素。当它们合在一起的时候就会很好的发挥效能。在这里并没有多少代码，所有困难的部分都由表视图、search controller、地址簿和其他 framework 做好了。

4.5 总结

你应该已经看过很多应用都使用了 navigation controller 和表视图来显示数据，而现在你知道他们是如何实现的了。主要的部分还是在表视图的 datasource 和 delegate 上。如果你将新创建的类定义为 UITableViewController 的子类，就会得到一个基础框架的代码。

在使用地址簿的时候，如果需要在单元格中列出联系人的更详细信息，比如住址什么的。这种情况下就可以把单元格的风格定义为 subtitle 来实现。如果需要的话，还可以显示地图来标明该联系人的地址。

在需要进行多层次浏览的时候(就是一大堆数据 ,然后可以一个个点开 ,越来越详细的那种。)，Navigation Controller 是再适合不过的了。其中每个 controller 都能推出下一个 controller，而 Back 按钮也已经自动准备好方便返回上一个 controller。只需要注意在 viewWillAppear 和 viewDidAppear 方法里面做了什么。不管是从上一个 controller 进来的还是从下一个 controller 退回来的，这两个方法在每个 view 出现的时候都会被调用。

在大部分应用中都可能需要用到 navigation controller 和表视图，所以如此重要的知识必须尽早掌握。当然你必须考虑好如何以不违反用户习惯的方式来使用它们。比如说，可以创建一个和屏幕等大的单元格，在里面显示若干图片，然后在表视图的 scroll view 属性中勾选 paging 功能，这样你就能创建出一本相册了，而且还会根据情况自动清理内存。

Navigation controller 和表视图将会成为你在 iOS 开发之路上的一个旅程碑。所以你现在就必须适应它们，并且以后熟练运用它们。而这部分功能也能更加开拓你的思维，让你更容易实现创意，应对以后开发中遇到的各种困难。

既然 view 和 view controller 的基础已经扎实了，那么我们就可以前往下一章，学习更加详细的 view 的运用知识。在第五单元中 ,你将会使用 MapKit 来实现地图相关功能 ,以及使用 CoreData 来获得用户的位置信息。SDK 已经提供了所需的基本工具，你可以自由使用这些工具来开发任何应用。

5 ——通过教你制作一个上架应用 WhereISMyCar 来学习 MapKit 和照相机功能

本章包括：

- Mapket 框架
- 处理地图以及用户位置
- 储存以及调用相机图片

我相信你会有这么一种体验，从商店出来，在停车场四处寻找，自己到底把车停到哪儿了。如果你的 iPhone 里有个应用能够给你展示一张地图以及照片，告诉你的车到底停到哪儿，这不是很棒吗？很多 App Store 里的应用都提供了这个服务，现在这个教程就教你如何创造一个自己的应用叫做：WhereAmIParked。

要创建 WhereAmIParked 这个应用，需要你添加一个地图到项目里，并且使用这个地图来查看用户的位置。CoreLocation 可以帮助你在没有地图的情况下查找到用户的位置，这需要使用 CoreLocation 框架。此外，在这个应用中还可以查找用户附近附近街区的地址，并且把这个地址存储下来。最后，我会教你让这个应用还有拍照功能。

从用户那儿收集信息，并且把这些信息展示出来需要很多的 UI 元素。首先让我们看看这个应用的 UI（见图 5.1）。

图 5.1 WhereAMIParked 的主界面

map view，展示了地图，地图锚点也是一个视图，就是从锚点上的那个弹出框。输入文本也需要一个视图，抓图和展示图片也都各需要一个视图。同时，还有一个记备忘的文本输入框，还有 Done 按钮也有一个视图。

WhereAMIParked，实际上是App Store上架应用。你可以点击下面链接下载：http://itunes.com/apps/WhereAm-IParked.

把所视图都放到一起是一件非常有意义的事情，这是所有 App 的核心。首先让我们从创建一个 map view 开始吧。

5.1.使用带用户位置的地图

一个 map view 可以被添加到一个 view 里，或者设置成一个 controller’ s view。默认设置是，map view 显示的是世界地图。通过各种设置你可以放大地图，将它定位到某个

特定坐标，得到用户的所在位置，然后展示其他用户可能感兴趣的地点。

最简单最快速的展示地图的方法是通过 Xcode 的 UI 编辑器来添加一个 map view 到你的工程中。但是在地图中展示用户的位置，且通过扩大地图找到位置，需要用到技巧 25——这个技巧会让你的应用更加好用。

技巧 24 添加一个 MapView

MapKit 框架拥有测绘功能和自己的 UI。像其他的控件一样，你需要在 Xcode 4 中把 MapView 添加到工程里，然后在代码里加一个引用。你还需要把 Mapkit 添加到你的工程里。

当需要添加某些特定控件到工程里，特别是那些不是你的工程里自带的默认控件时，添加一个和控件相关的框架是一种普遍的方式。只要你添加了 MapView，把他写入代码，你就可以准备添加真正的功能了。

问题

你需要添加一个 MapView 到 iPhone 应用。关于控制地图的事儿可以待会儿再看。现在首先要做的是开始一个工程并且添加一个地图。

解决方案

给你的工程添加一个基本的地图非常简单。你只要在 Xcode 4 里增加一个 MapView，然后添加 MapKit 框架，运行工程就可以了。

讨论

首先创建一个新的 Xcode 工程，选择一个地址和名字。我们给它命名为 WhereAmIParked。点击打开在工程的 Rescources group 里的 WhereAmIParkedViewController.xib 文件。打开之后，初始的 view 是空白的。

正如在第三章中提到的，你可以在 UI View 中添加任何形式的视图（例如：button，toolbar，table view）。点击 IB library 中的 Data View，把 MapView 拖拽到工程中的空白的 view 里。在 Attributes 窗口，检查一下展示用户位置的框（见图 5.2）。在 IB 中保存 UI，然后返回 Xcode。

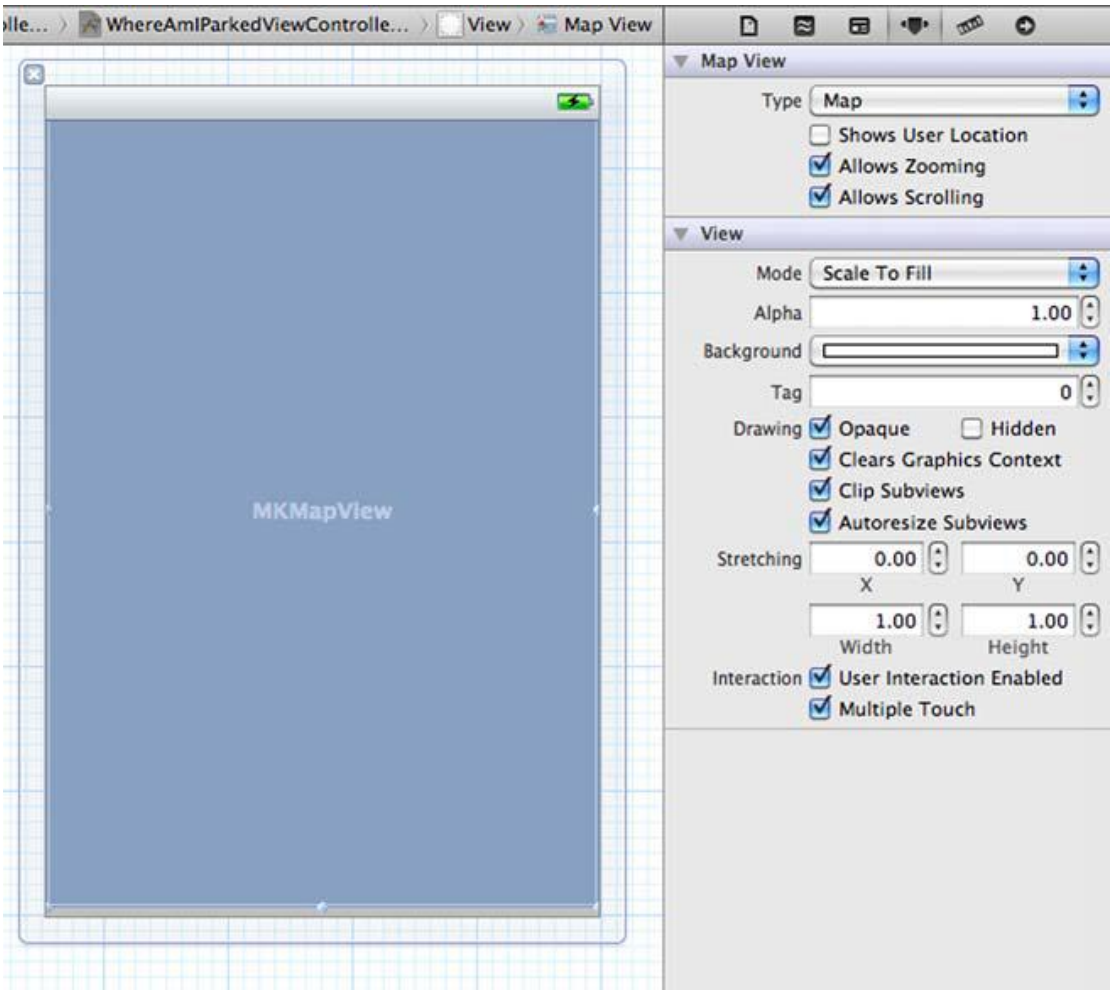


图 5.2 可以展示用户位置的 MapView 的设置

现在在模拟器中创建运行这个应用会生成一个错误，因为你还没有添加 MapKit 框架到工程中。下面告诉你接下来怎么做。在 Project Navigator 中点击工程，把选择 target 和 Build Phases 标签。使用 Libraries section 来扩展 Link Binary，再点击+按钮。从列表中，通过 Map 关键词来过滤，然后选择 MapKit. 框架（见图 5.3 ）。点击 Add 按钮，把 MapKit 框架添加到工程里。

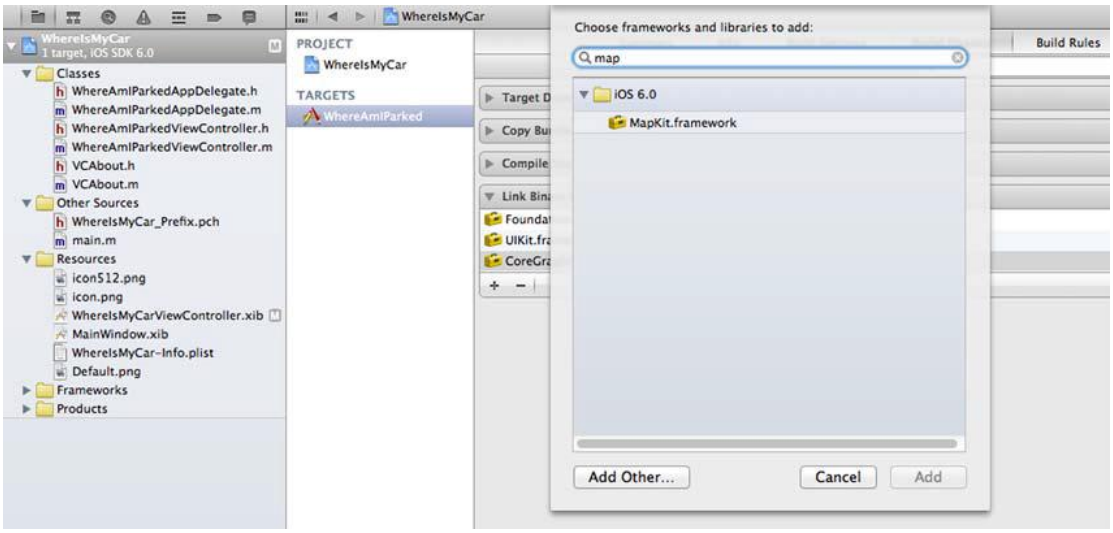


图 5.3 添加 Mapkit 框架

现在你可以在模拟器（Commard-r）中运行应用了，现在可以看到地图了。再等一会儿，你就可以在地图上找到你的位置。由于你现在在模拟器中运行程序，你需要使用底部的工具栏中提供的位置按钮（见图 5.4）。

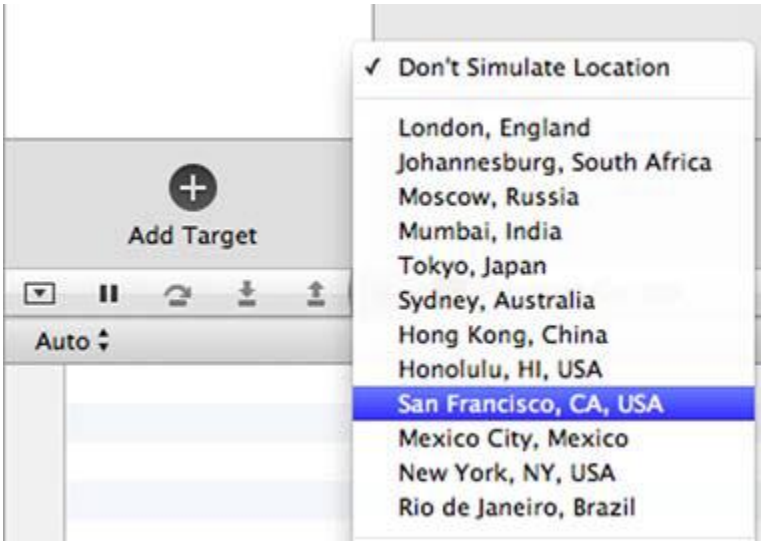


图 5.4 在模拟器中运行的显示用户位置的基本地图应用。

现在这个应用的功能还不完整，实际现在你并不能在地图上查找到你的位置。在做一些小的添加，你就可以扩大地图，找到你的所在位置。

技巧 24 放大缩小 Mapview

添加一个地图很简单，但是需要要让这个地图可操作，可以让用户找到特定的位置，这样才对用户真正有用。

正如你可能在其它应用里看到的一样，地图有很多可添加的功能，比如说变换位置，放大缩小，增加锚点，显示轮廓等。最基本的一个功能是显示你想要查找的位置。

问题

你需要设置地图展示的区域，包括设置 center coordinate 以及 zoom level。

解决方案

设置你的控件为 Map view 的 delegate，然后插入 mapViewDidFinishLoadingMap: 方法。在这个方法里，你需要从 Map View 中获取用户位置，然后在地图上找到想要的位

置，这么做的话，你需要确保 Map View 有时间来确定用户位置。同时，你需要把地图的范围——即展示的多大的区域设置到一个对一个停车应用来说比较合理的小范围。

讨论

正如像 UItaleView 里的其他的 classes 一样 ,map view 也有一个 delegate member。在 WhereAmIParkedViewController.h 里，声明你的 class 为 MKMapViewDelegate，（见以下代码）然后使用 IB 设置它为 map view 的 delegate。

将 MapKit.h 文件导入 WhereAmIParkedViewController

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>>
@interface WhereAmIParkedViewController : UIViewController
<MKMapViewDelegate> {
}
@end
```

设置 map view 的中心坐标以及 zoom level 的 setters 有很多种。中心坐标以及 zoom level 合起来可以称为一个 region（区域），这是我们需要设置的东西。

当地图加载完成后，我们要给委托接入第一个方法。这个时候，你希望定位用户的大体位置，然后可以通过放大地图，找到确切位置。

我们需要接入mapViewDidFinish-LoadingMap方法，你可以在 map view中找到它（见以下代码）。现在可以使用用户的位置了，可以看到一片区域，这个区域是根据经度纬度测量的。纬度通常都是一样的距离，约为69里。尽管纬度大约是赤道附近69度，随着它们离两级越来越近，它们会变得越来越紧密。要是设置为0.02的话大概代表1.38里。

当地图停止加载开始设置区域时调用MapView委托方法

```
- (void)mapViewDidFinishLoadingMap:(MKMapView *)mapView
{
MKCoordinateRegion region;
region.center = [mapView userLocation].location.coordinate;
                //以用户所在地为中心
region.span.latitudeDelta = 0.02;
region.span.longitudeDelta = 0.02;
                //缩小区域
[mapView setRegion:region animated:YES];
                //添加动画
}
```

在 UI 编辑器里，选择 Map View，然后查看它的连线。给 File’s Owner 的对象设置 delegate，即 WhereAmIParkedViewController 实例。

现在从 Xcode 的模拟器中运行应用还不会是你想要的结果。因为大部分地图的加载是在用户位置确定之前，region 的设置会使用经度 0.0000 和纬度 0.0000 来放大。

首先在 delegation 方法中检查一下用户位置，告诉 run loop 在一秒后再进行调用。如果还是找不到这个位置，一秒钟后返回，再进行上述步骤（见以下代码）。

MapView 委托方法重复检查用户所在位置

```
- (void)mapViewDidFinishLoadingMap:(MKMapView *)mapView
{
if (0.00001 > [mapView userLocation].location.coordinate.latitude)
{
[self performSelector:@selector(mapViewDidFinishLoadingMap:)
```

```
withObject:mapView afterDelay:1.0];
return;
}
MKCoordinateRegion region = [mapView region];
region.center = [mapView userLocation].location.coordinate;
region.span.latitudeDelta = 0.02;
region.span.longitudeDelta = 0.02;
[mapView setRegion:region animated:YES];
}
```

现在可以在模拟器中运行应用，几秒过后，你可以扩大地图，查找到默认设置的位置，Cupertino 市的 Infinite Loop 路（见图 5.5）

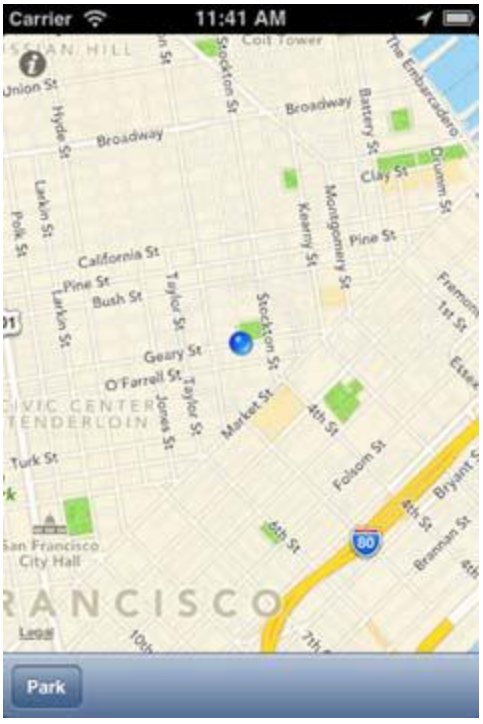


图 5.5 显示用户位置

技巧 24 通过 CoreLocation 查找用户位置

苹果提供了 CoreLocation 框架作为 SDK 的一部分，用它可以来查找设备的现在位置，也即是用户所在的位置。

用 CoreLocation 确定用户位置是一个异步过程。告诉 CoreLocation 来定位用户之后，会调用一个特定的回调，里面包含了用户位置的更新。我们的应用可以指定位置更新的精确度，当收到了需要的数据时可以停止这个过程。停止持续更新可以很好的保护电池寿命。

问题

在没有 Mapview 的情况下你需要找到用户位置的经度纬度。

解决方案

通过创建一个 CLLocationManager instance 你可以查找到用户坐标 ,调配这个实例，开始位置更新。

讨论

如果你想不使用地图，找到用户位置，CoreLocation 提供了另一种方式。创建一个 CLLocationManager instance，设置其为 delegate，调用 startUpdatingLocation。这个你设为 delegate 的 class 还需有接入一个 CLLocationManagerDelegate 协议（见以下代码）。

开始并处理 CoreLacation 更新

```
- (void)startUpdates
{
if (nil == locationManager)
locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.desiredAccuracy
= kCLLocationAccuracyKilometer;
                        //设置精确度
locationManager.distanceFilter = 10;
                        //更新阙值
```

```
[locationManager startUpdatingLocation];
                    //开始更新
}
- (void)locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *)oldLocation
                    //委托方法
{
NSDate* eventDate = newLocation.timestamp;
NSTimeInterval howRecent = [eventDate timeIntervalSinceNow];
if (abs(howRecent) < 120.0)
{
[manager stopUpdatingLocation];
theLocation = newLocation;
[theLocation retain];
}
}
```

根据对精确性和过滤器的设置，你可以控制更新的次数。同时，根据需要，也可以通过stopUpdatingLocation.来停止位置的更新。

一旦 startUpdatingLocation 调用了，CoreLocation 就会在一个单独的线程运行，这时你就可以继续应用的其他进程了。根据更新的信息你可以做很多其他操作。

使用 CLLocationManager 可以让你可以通过设定精确性和过滤器来控制地图和其他事情。使用 MapKit 有很多优点。最大的优点是地图可以给用户一个可视化的反馈。下面让我们来看看地图上的位置可以怎么展现给用户。

5.2 显示 MapView 位置的细节，储存用户备忘

大部分地图都用锚点或是其他图像来显示位置。这些元素被称为地图标注（annotation）。一般来说，当一个用户点击应用提供的标注时，应用会有所反应。大部分情况下，应用会显示一个弹出框，这个弹出框我们叫做 callout。

在接下来的教程里，你可以学到如何在一个给定的地图位置上显示一个锚点标注，储存用户位置（停车的地点）。为了让应用功能更多，可以反向地理编码（reverse geocode）用户位置来查找附近的街道地址，并且在 callout 弹出框里显示。同时，你还可以做备忘功能，让用户编辑停车地点的提醒信息。

技巧 24　显示 MapView 锚点

可以显示用户位置的基本地图非常有用，但是对我们的应用来说却远远不够。一个只能显示自己所在位置的地图的帮助性不是很大，也不独特。但是可以用锚点显示附近位置信息的地图就很好用了。

位置定位是手机应用的一个独特功能，你的所在位置对台式电脑来说没有什么意义，因为台式电脑不会移动。而不管是处于功能原因还是信息原因，定位对一个 iPhone 应用来说非常有用。在很多情况下，我们可以从服务器或者其他显示你感兴趣的地点（points of interest）的数据源获取地址。在我们的应用中，地点位置信息是由用户所在位置决定的，这个位置储存在 iOS 设备里。

问题

你需要添加一个锚点到地图中来显示用户位置。

解决方案

首先你需要找到一个方法来告诉这个应用你在哪儿停的车。最简单的一个办法是在用户界面增加一个按钮。

这个按钮需要调用 WhereAmIParkedViewController 实现文件中的一个方法，所以现

在给 doParkBtn:.增加一个 IBAction 到头文件里。为了可以给地图增加位置点，你还需要一个 map view 的 outlet。

WhereAmIParkedViewController 以及 Park 按钮的 IBAction

```
@interface WhereAmIParkedViewController : UIViewController
<MKMapViewDelegate> {
IBOutlet MKMapView *mapParking;
        //MapView的Outlet
}
-(IBAction)doParkBtn:(id)sender;
        //按钮的action
@end
```

或者，你可以等到创建一个对象的时候，在我们的例子里，也就是创建 map view 的时候，在 Assistant editor 里，按 ctrl 键把 mapview 拖拽到头文件里。

现在你已经给 map view 声明了一个 outlet，给按钮声明了一个调用 action。现在你需要创建一个按钮，同时接入 outlet 和 action。在 UI 编辑器里，变换 map view 的大小，让它在底部有一定的空间，从 Library 的 Windows&Bars group 放置一个 toolbar 到底部空间（见图 5.6）。

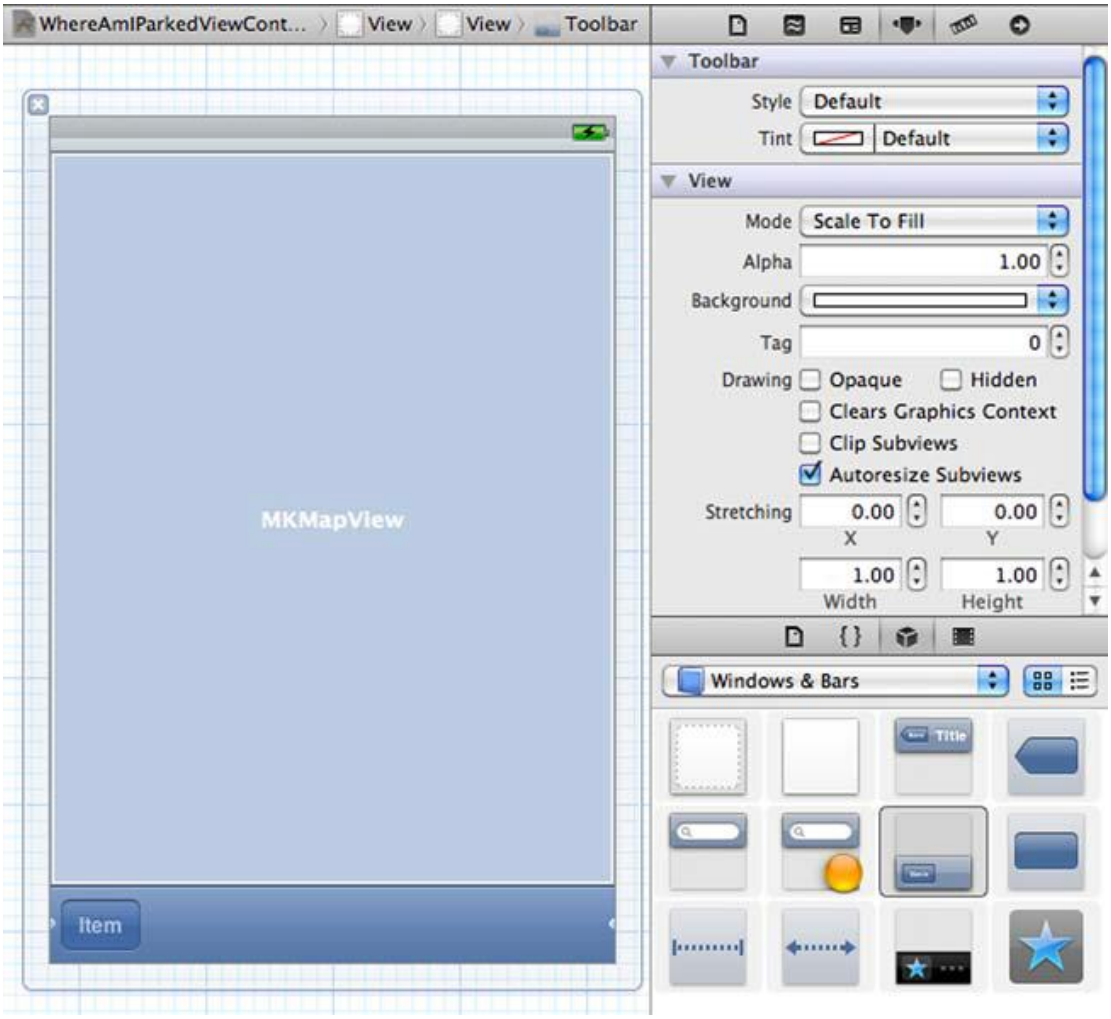


图 5.6 在 Interface Builder 添加一个 toolbar 到 map view 底部

在 File’ s Owner 的关联（connection）里，关联 mapParking outlet 到 mapview。双击在 toolbar 上默认的 Item 按钮，输入一个新名字:Park。在连线窗口，设置 Park 按钮的选择器（selector）为你刚刚在 WhereAmIParkedViewController 里定义过的 IBAction（命名为 doParkBtn）这个 park 按钮的选择器是在 Interface Builder 里的 File’ s Owner。

接入 doParkBtn，设置用户的位置为用户的停车地点。现在让我们来看看怎么做。

地图上的位置标注叫做 annotations。iOS 的 SDK 提供了一个 MKAnnotation 的原型，但这个原型不是默认就接入了的。你必须要创建一个 annotation class，这非常简单。你可以在 WhereAmIParkedViewController 文件中添加，如以下代码所示：

MKAnnotation 头文件的停车地点实现文件

```
@interface SpotAnnotation : NSObject <MKAnnotation>
{
CLLocationCoordinate2D coordinate;
NSString *title;
NSString *subtitle;
}
@property (nonatomic, assign) CLLocationCoordinate2D coordinate;
@property (nonatomic, retain) NSString *title;
@property (nonatomic, retain) NSString *subtitle;
@end
```

大部分的工作在声明变量和属性的过程中完成了。现在剩下的事情是把他们 synthesize 到执行文件中去，见以下代码：

MKannotation 实现文件的停车地点实现文件

```
@implementation SpotAnnotation
@synthesize title, subtitle, coordinate;
@end
```

doParkBtn 功能的实现，需要在地图上添加一个用户位置锚点标注的实例对象。见以下代码：

当用户点击 Park 按钮的时候增加一个 SpotAnnotation 到 map view

```
-(IBAction)doParkBtn:(id)sender;
{
SpotAnnotation *ann = [[SpotAnnotation alloc] init];
[ann setCoordinate:[mapParking userLocation].location.coordinate];
[mapParking addAnnotation:ann];
}
```

Map View 获取到标注之后，它会询问其 delegate 用什么图像来展示这个标注。你可

以使用它的默认锚点注释视图。你可以使用 delegation 方法来实现，和 UITableview 方法检索一个 cell 的过程类似。见以下代码：

设置 annotation view，让它在 map view 上显示

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
viewForAnnotation:(id <MKAnnotation>)annotation
{
if (![annotation isKindOfClass:[SpotAnnotation class]])
return nil;
        //不要替换掉用户位置视图
NSString *dqref = @"ParkingAnnon";
        //重新使用视图
id av = [mapView
dequeueReusableAnnotationViewWithIdentifier:dqref];
if (nil == av)
{
av = [[MKPinAnnotationView alloc] initWithAnnotation:annotation
reuseIdentifier:dqref];
[av setPinColor:MKPinAnnotationColorRed];
[av setAnimatesDrop:YES];
}
return av;
}
```

我们的应用现在已经的雏形已经做得差不多了，你可以在模拟器里运行看看。加载扩大地图，点击 Park 按钮放置一个注释锚点（见图 5. 7）。

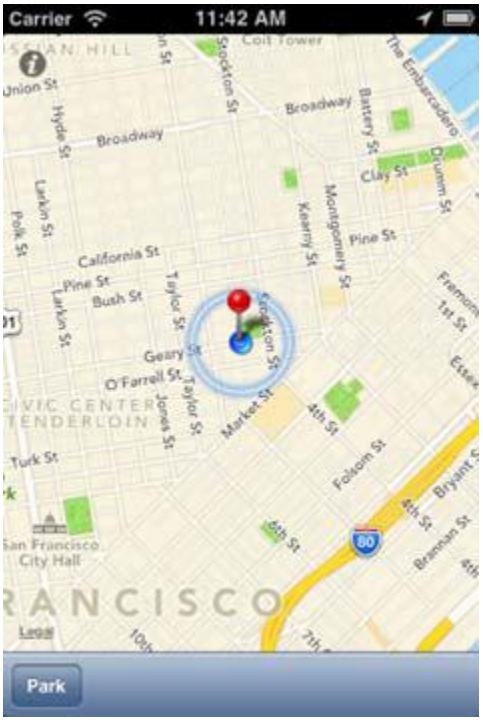


图 5.7 模拟器中的画面

你可能注意到了每次你点击 Park 按钮的时候，它都会在同一个位置增加一个新的锚点。假定我们有了一个新的位置，现在就需要在新的位置增加锚点，把原来老的位置上的锚点去掉。每次增加新锚点的时候我们需要检查在地图上是否有其他的锚点，如果有其他的锚点的话，就把之前的那个锚点去掉。见以下代码：

在增加新锚点之前从地图上移除除了第一个 annotation 之外的所有 annotation

```
-(IBAction)doParkBtn:(id)sender;
{
if ([[mapParking annotations] count] > 1)
[mapParking removeAnnotation:[[mapParking annotations]
objectAtIndex:1]];
SpotAnnotation *ann =
[[SpotAnnotation alloc] init];
[ann setCoordinate:[mapParking userLocation].location.coordinate];
[mapParking addAnnotation:ann];
```

}

通过以上代码我们可以实现增加新锚点的时候去除之前的锚点。

现在可以定位用户停车的位置了，但是这个信息只有在用户在找停车地点的时候才有用，为此，应用还必须有存储位置的功能。现在让我们看看怎么做。

技巧 24 储存并检索用户位置

只能实现在地图上添加一个锚点标注用户停车地点用处并不大，位置信息只有在用户需要找寻他们的车时才有用。要做到这点，应用必须能够储存该地点的经度纬度。

我们可以使用 iOS SDK 提供的 NSuserDefaults，它可以储存设备的信息。NSUserDefatults 可以轻易的储存多种类型的数据，如单精度浮点数，双精度浮点数 ，整数， boolean 值，还有包括 NSData, NSString, NSNumber, NSDate, NSArray,以及 NSDictionary 的多种对象。你可以使用 NSUserDefaults 来存储用户位置坐标，这个坐标可以在这个应用接下来的使用中存取使用。相同的对象在检索储存数据的存取方法是相似的。

问题

你需要存储加载位置数据，并在地图上展示出来。

解决方案

如果把在地图上添加锚点这个功能分离出来，那么，就可以在用户点击 Park 按钮时增加锚点，还可以应用加载后，显示之前的锚点位置了。可以使用 NSUserDefaults 来储存用户停车位置的经度纬度。当加载应用的时候，它会检查之前是否储存了之前的位置，如果储存了，就会在之前的位置添加一个锚点。

讨论

下一次应用加载的时候，它会检查之前储存的坐标，然后创建一个锚点(见以下代码)。创建锚点单独做一个方法，命名为dropPinAtCoord。这个新的方法被以及

doParkBtnmapViewDidFinishLoadingMap调用。

在用户的所在位置增加一个锚点，存储它

```
- (void) dropPinAtCoord: (CLLocationCoordinate2D) coord {
if ([[mapParking annotations] count] > 1)
[mapParking removeAnnotation:[[mapParking annotations]
objectAtIndex:1]];
SpotAnnotation *ann = [[SpotAnnotation alloc] init];
[ann setCoordinate:coord];
[mapParking addAnnotation:ann];
[self doRevGeocodeUsingLat:coord.latitude andLng:coord.longitude];
[[NSUserDefaults standardUserDefaults]
                //存储地点的坐标
setFloat:coord.latitude forKey:@"WIMCLat"];
[[NSUserDefaults standardUserDefaults]
setFloat:coord.longitude forKey:@"WIMCLng"];
}
-(IBAction)doParkBtn:(id)sender;
{
CLLocationCoordinate2D coord =
[mapParking userLocation].location.coordinate;
[self dropPinAtCoord: coord];
                //调用新方法
}
- (void)mapViewDidFinishLoadingMap:(MKMapView *)mapView
{
if (0.00001 >
[mapView userLocation].location.coordinate.latitude)
{
```

```
[self performSelector:@selector(mapViewDidFinishLoadingMap:)
withObject:mapView afterDelay:1.0];
return;
}
MKCoordinateRegion region = [mapView region];
region.center = [mapView userLocation].location.coordinate

region.span.latitudeDelta = 0.02;
region.span.longitudeDelta = 0.02;
[mapView setRegion:region animated:YES];
if ([[NSUserDefaults standardUserDefaults]
floatForKey:@"WIMCLat"] != 0.000)
                    //增加锚点
{
CLLocationCoordinate2D coord;
coord.latitude = [[NSUserDefaults standardUserDefaults]
floatForKey:@"WIMCLat"];
coord.longitude = [[NSUserDefaults standardUserDefaults]
floatForKey:@"WIMCLng"];
[self dropPinAtCoord:coord];
}
}
```

数据可以储存在 SQLite 或者服务器上。储存在哪取决于你的需要。

知道了用户位置之后，你就可以显示位置，给位置增加特定的信息，确定两个地点的位置距离等。这就是手机设备的优势之一。

技巧 24 反向地理编码

有时候我们需要寻找附近的街道位置。在我们的应用中，可以看到停车附近的街道地址

会很有帮助。

地理编码（geocoding）可以使用一个街道的地址来找到对应的地理经度纬度坐标。Mapkit 可以做相反的过程，叫做反向地理编码。

使用反向地理编码可以通过用户的所在经纬度数据，找到附近的街道地址。在我们的应用里，这个功能很有用。

问题

你需要找到用户位置附近的街道地址。

解决方案

把CoreLocation框架添加到target，（Target > BuildPhases > Link Libraries with Binary），将CoreLocation/CoreLocation.h文件导入到WhereAmIParkedViewController.h。然后你就可以创建CLGeocoder进行反向地理编码了。

讨论

反向地理编码完成后会取得completionhander以及CLLocation坐标，通过这两个CLGeocoder得以实例化。现在我们可以取得地址了。见以下代码。

通过反向地理编码来寻找所在位置的地址信息

```
-(void)doRevGeocodeUsingLat:(float)lat andLng:(float)lng;
{
CLLocation *c = [[CLLocation alloc] initWithLatitude:lat
longitude:lng];
                    //创建位置
CLGeocoder *revGeo = [[CLGeocoder alloc] init];
[revGeo reverseGeocodeLocation:c
```

```
                    //反向地理编码
completionHandler:^(NSArray *placemarks,
NSError *error) {
if (!error && [placemarks count] > 0)
{
NSDictionary *dict =
[[placemarks objectAtIndex:0] addressDictionary];
NSLog(@"street address: %@",
                    //记录地址
[dict objectForKey:@"Street"]);
}
else
{
NSLog(@"ERROR: %@", error);
}
}];
}
```

从代码中，你会发现地址在控制台存储了。

```
2012-09-18 12:44:06.773 WhereIsMyCar[18871:13d03] street address: 243–299
Geary St
```

在其它情况下，你可能会需要处理多个返回的标记。但是在我们的应用，只需要处理一个。

在这个应用里，地址可以在锚点callout上展示（锚点的callout是你点击锚点时弹出的视窗）。只需要再做一些工作，你就可以得到这个效果。

技巧24 添加一个锚点callout

地图上的锚点上有详细信息对用户来说很有用也很必要。如果你展示公园，商店，饭馆或者其他商业位置场所的名称，地址，离用户的距离提供给用户，那会非常棒。

点击一个锚点时，应用会弹出一个在锚点上方的小弹出框。 你可以在这个弹出框里展示一些东西，比如说是文本，命名为title（标题）或是subtitle（次标题）。就像table view cell一样，你可以在它的左边展示一副图，在右边展示一个配件（例如说是一个disclosure button）。

默认的标注并没有callout，但是只要把callout的值设为yes，就可以展示出来了。用户点击锚点，弹出callout是有OS控制的。有几件是app可以定制展示的，比如说是标题，次标题 ,图像 ,discloure button等。可以展示反向编码地理出的地址信息对应用来说非常好。

问题

你需要再地图锚点标注callout上显示一个标题。

点击锚点在callout上显示附近街道地址需要三个步骤。第一，你必须设置好点击锚点时即弹出callout；第二，你需要方向地理编码得到所在的地址信息；第三，把所得信息设置为标题。

讨论

首先，你必须要让标注可以显示callout。然后使用反向地理编码，设置标注的标题。见以下代码：

设置annotation，点击的时候显示callout弹出框

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
viewForAnnotation:(id <MKAnnotation>)annotation
{
if (![annotation isKindOfClass:[SpotAnnotation class]])
return nil;
```

```
NSString *dqref = @"ParkingAnnon";
id av = [mapView
dequeueReusableAnnotationViewWithIdentifier:dqref];
if (nil == av)
{
av = [[MKPinAnnotationView alloc]
initWithAnnotation:annotation
reuseIdentifier:dqref];
[av setPinColor:MKPinAnnotationColorRed];
[av setAnimatesDrop:YES];
        //被点击时弹出callout弹出框
[av setCanShowCallout:YES];
}
return av;
}
```

这样标注就可以展示callout弹出框了。现在你需要让标题显示为反向地理编码的地址。doParkBtn方法是用来创建标注的，在doParkBtn中调用doRevGeocodeUsingLat:andLng:方法。见以下代码:

当用户停车时开始反向定理编码

```
-(IBAction)doParkBtn:(id)sender;
{
if ([[mapParking annotations] count] > 1)
[mapParking removeAnnotation:[[mapParking annotations]
objectAtIndex:1]];
SpotAnnotation *ann = [[SpotAnnotation alloc] init];
[ann setCoordinate:[mapParking userLocation].location.coordinate];
[mapParking addAnnotation:ann];
CLLocationCoordinate2D coord =
```

```
        [mapParking userLocation].location.coordinate;
[self doRevGeocodeUsingLat:coord.latitude
        andLng:coord.longitude];
            //开始反向地理编码
 }
```

下一步，你需要设置反向地理编码得出的地址为标题。见以下代码：

设置街道地址作为标题

```
-(void)doRevGeocodeUsingLat:(float)lat andLng:(float)lng;
{
CLLocation *c = [[CLLocation alloc] initWithLatitude:lat
longitude:lng];
CLGeocoder *revGeo = [[CLGeocoder alloc] init];
[revGeo reverseGeocodeLocation:c
completionHandler:^(NSArray *placemarks,
NSError *error) {
if (!error && [placemarks count] > 0)
{
NSDictionary *dict =
[[placemarks objectAtIndex:0] addressDictionary];
NSLog(@"street address: %@",
[dict objectForKey:@"Street"]);
for (SpotAnnotation *ann in [mapParking annotations])
{
if ([ann isKindOfClass:[SpotAnnotation class]])
[ann setTitle:
[dict objectForKey:@"Street"]];
            //设置标题
}
```

```
}
else
{
NSLog(@"ERROR/No Results: %@", error);
        //处理错误
}
}];
}
```

reverseGeocodeLocation: completionHandler更新在标注进行循环，然后检索单个的SpotAnnotation实例，一旦检索到了结果，反向地理编码就会返回街道地址。

现在运行应用，就有我们想要的结果了（见图5.8）。

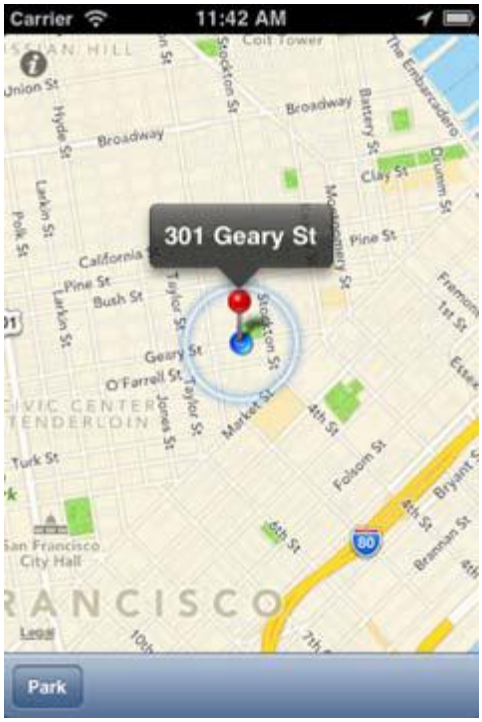


图5.8 锚点标注上显示地址信息

现在，你有了一个比较有用的应用了，但是还不是够好。让我们看看怎么提高它。

现在你已经学会了如何查找用户地址，在地图上展示地址的详细信息。同样的步骤和技术还可以在许多更复杂的应用上用到。你可以在地图上展示多个位置，使用多个标注和带有

更多详细信息的callouts。现在我们已经掌握了基础原理。

对我们这个应用来说，还有一些功能可以优化。

技巧24 输入储存用户备忘

你已经学会了如何使用NSUserDefaults来储存数据。包括储存数据以及之后获取NSString对象。使用NSString，应用可以储存一条文字信息。但是现在你还需要接入一个输入框让用户可以输入信息。

然后还需要再整个UI上加上这个输入框，当用户想要输入备忘的时候可以使用。

问题

为了让用户可以记住他们在哪停的车，你需要让用户可以输入一些评论或者笔记备忘，可以储存，然后可以调出展示。

解决方案

增加一个输入框来记录备忘，并且让这个数据可以储存，且可以调出。

讨论

用户输入备忘包含两个部分。输入，储存。这就意味着你需要增加一个输入框来让用户输入备忘，这个输入框之后还可以展示输入的备忘，还可以编辑备忘。

首先，你需要在代码里创建action和outlet。然后，你需要使用IB来创建UI，然后把这个UI元素在代码中和action和outlet连接起来。最后，你需要储存这个用户输入的文本，以后还可以把这个文本调出。

首先，创建WhereAmIParkedViewController类，把这个类添加到UI编辑器。你需要给备忘的文本输入框添加outlet和一个Done按钮 ,还需要给新的备忘和Done按钮添加两个action。见以下代码：

给Note和Done按钮在view controller的头部创建outlet和action

```
@interface WhereAmIParkedViewController : UIViewController
<MKMapViewDelegate, MKReverseGeocoderDelegate> {
IBOutlet MKMapView *mapParking;
        //Note按钮的文本框
IBOutlet UITextView *tvNote;
IBOutlet UIButton *btnDone;
        //备忘输入完成之后的Done按钮
}
-(IBAction)doParkBtn:(id)sender;
-(IBAction)doNoteBtn:(id)sender;
-(IBAction)doDoneBtn:(id)sender;
        //Done按钮的action
@end
```

现在回到UI，给toolbar增加一个bar按钮。然后在两个按钮中间增加一个Flexible Space。然后把定义好的action接入这个按钮。

在地图顶部增加一个文本输入框，用户可以在这儿输入信息（见图5.9）。然后在这个文本输入框上方增加一个按钮，设置其标题为Done。

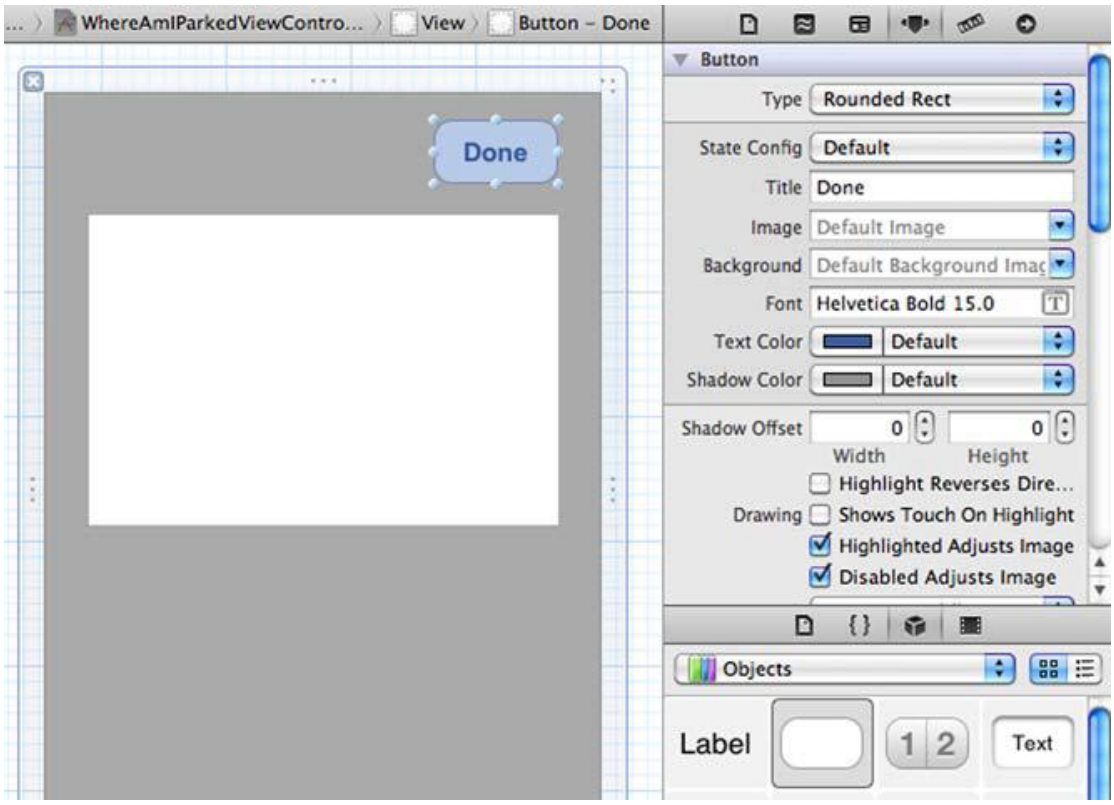


图5.9 增加文本输入框和Done按钮

使用IB，把之前在controller class中定义好的doNoteBtn：action接入None按钮。把doDoneBtn: action接入Done按钮。把action接入按钮还有另外一个方法，就是按住Ctrl键 ,拖拽UI上的button ,然后把他拖拽到class里 ,见图5.10。然后从弹出的目录里选择action。

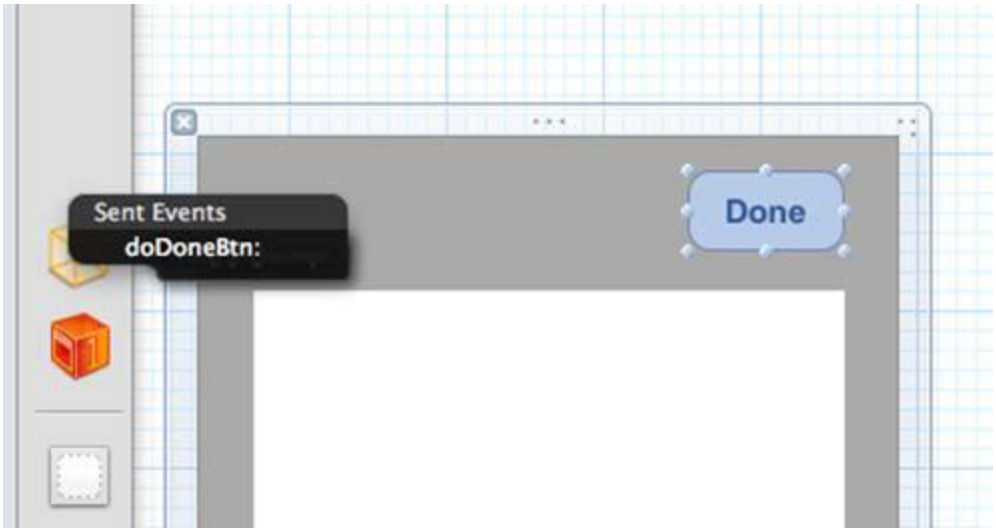


图5.10 实现Done按钮的action

然后，在控制器里把tvNote接入文本输入框。最后在controller class（即File’ s Owener）里写入btnDone outlet然后把它接入Done按钮。

应用启动时，这个文本输入框和Done按钮不能显示，所以你需要使用在IB里设置View setting为隐藏（见图5.11）。

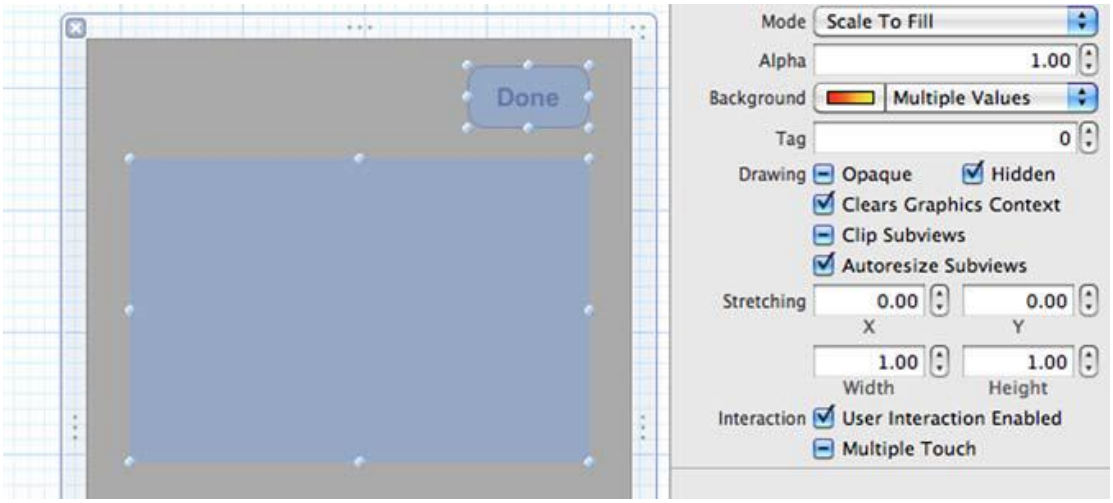


图5.11 设置文本输入框为隐藏

现在你需要接入doNotBtn。当用户点击按钮的时候，应用需要弹出文本输入框让用户可以输入备忘。同时你还需要一个Done按钮。最后，使用文本输入框的时候，需要弹出键盘供用户使用。见以下代码：

显示文本输入框以及显示键盘

```
-(IBAction)doNoteBtn:(id)sender;
{
[btnDone setHidden:NO];
[tvNote setHidden:NO];
[tvNote becomeFirstResponder];
```

```
        //显示键盘
}
```

现在ok了。还有一个问题，怎么让文本输入框和键盘消失呢？ 如何储存备忘呢。这就需要给Done按钮接入doDoneBtn:action。见以下代码。你可以把备忘在标准的NSUserDefaults里储存为一个对象。

移除文本输入框以及Done按钮，保存备忘

```
-(IBAction)doDoneBtn:(id)sender;
{
[btnDone setHidden:YES];
        //隐藏item
[tvNote setHidden:YES];
[tvNote resignFirstResponder];
[[NSUserDefaults standardUserDefaults]
        //保存备忘
setObject:[tvNote text] forKey:@"WIMCAppNote"];
}
```

备忘是只有在需要的时候展现才有用，现在如果在模拟器中运行应用，你可以添加备忘（见图5.12）。点击Done按钮是，文本输入框会消失。再次点击Note，添加的备忘会出现。

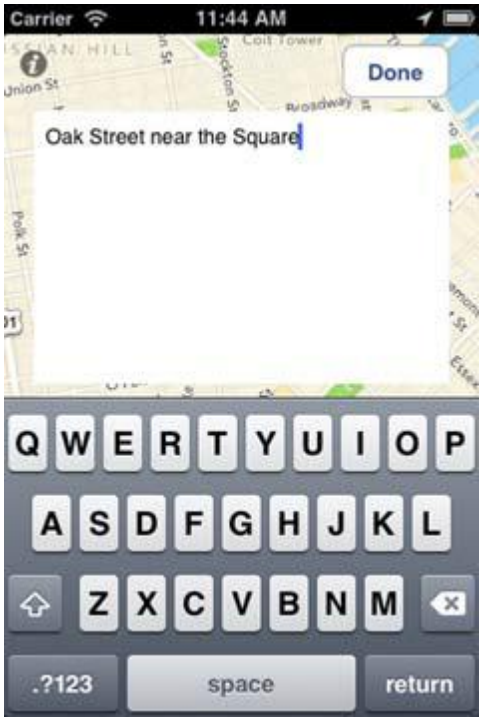


图5.12 编辑用户备忘的时候显示的文本输入框和Done按钮

但是如果你重启应用 ,点击Noete按钮 ,输入的备忘消失了 !真的消失了吗？实际没有，只是你没有让它展示出来，修正这个问题，你可以在doNoteBtn上再加几行代码。

检索备忘内容，让它在文本输入框中显示

```
-(IBAction)doNoteBtn:(id)sender;
{
[btnDone setHidden:NO];
[tvNote setHidden:NO];
[tvNote becomeFirstResponder];
[tvNote setText:[[NSUserDefaults standardUserDefaults]
objectForKey:@"WIMCAppNote"]];
}
```

如果你通过模拟器的Home按钮退出引用，信息还是会储存在NSUserDefaults里。但是如果通过Xcode关闭应用，信息不会储存。

现在你有了一个不错的应用了。还可以如何改进呢？

5.3 储存，检索展示照相机照片

如果用户可以拍一张停车地点的照片来记住他们在哪儿停的车，就再好不过了。

在应用里检索图片并不是很难。SDK可以解决大部分工作。和core location和反向地理编码类似，你可以启动照相机的image picker，接入delegate方法。

任何想要调用照相机或者相册的都需要一个image picker controller。首先让我们看看如何给工程添加一个image picker controller。然后再设置应用可以接收照相机拍下的照片然后在设备中储存这张照片。最后，在让这张照片之后可以检索到，然后展示给用户。

技巧24 添加一个camera control

SDK中是包含了Camera UI的，你可以利用它来让应用更有价值。Image picker让用户可以获取他们设备中的图片或者让用户可以使用内置照相机来拍照。

像之前见过的其他控制器一样（比如WebView），你可以通过UI编辑器来添加image picker。把它接入代码，让我们可以操控这个控制器，然后，可以选择相册照片或者拍照。

问题

给你的工程添加一个image picker controller。

解决方案

给你的类添加一个UIImagePickerController outlet。同时，你还需要给这个声明添加一个相关的action。在UI中，你需要添加image picker和按钮，然后把他们和类连线。

讨论

拍照功能的实现需要UIImagePickerController。UIImagePickerController可以设置为使用照相机或者是相册图片。你当然需要使用照相机，所以你需要新的按钮用来拍照，展示一张图片，删除图片。同时你还需要一个带有image view的view controller来展示图片，还有一个让view controller消失的按钮。

当然，你需要添加以上所有按钮的action。可以把以上全部添加到WhereAmIParkedViewController.h文件。见以下代码：

增加UIImagePickerController以及aciton到WhereAmIParkedViewController.h文件

```
IBOutlet UIImagePickerController *picController;
IBOutlet UIViewController *vcDisplayPic;
IBOutlet UIImageView *ivPic;
}
-(IBAction)doPicBtn:(id)sender;
-(IBAction)doShowPicBtn:(id)sender;
-(IBAction)doDonePicBtn:(id)sender;
```

现在让我们在UI中创建一个view controller。View controller中还需要一个view，这个view里需要一个image view和一个Done按钮（见图5.13）

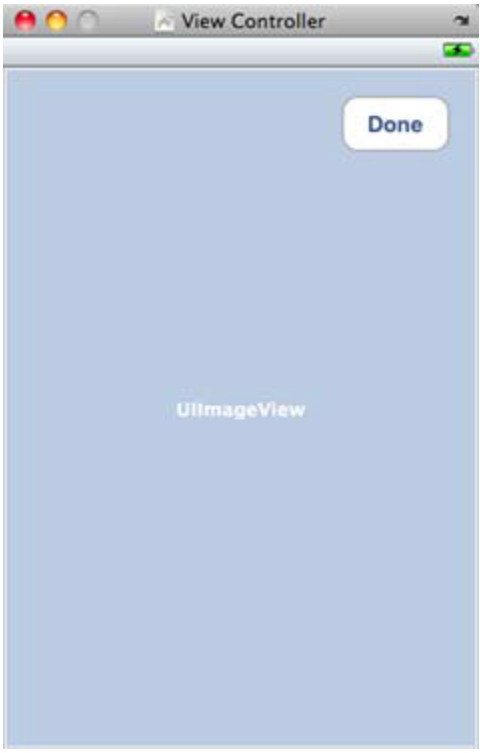


图5.13 View Controller以及可以展示图片的子view

你还需要添加一些按钮来照相，展示图片。（见图5.14）

图5.14 在toolbar上的 Show pic和Take Pic按钮

接下来你需要一个image picker。可以把它添加到XIB文件中（见图5.15）。设置它的delegate为WhereAmIParkedView-Controller (File’ s Owner)。你需要确定WhereAmIParkedViewController的头文件中实现了UIImagePickerControllerDelegate。

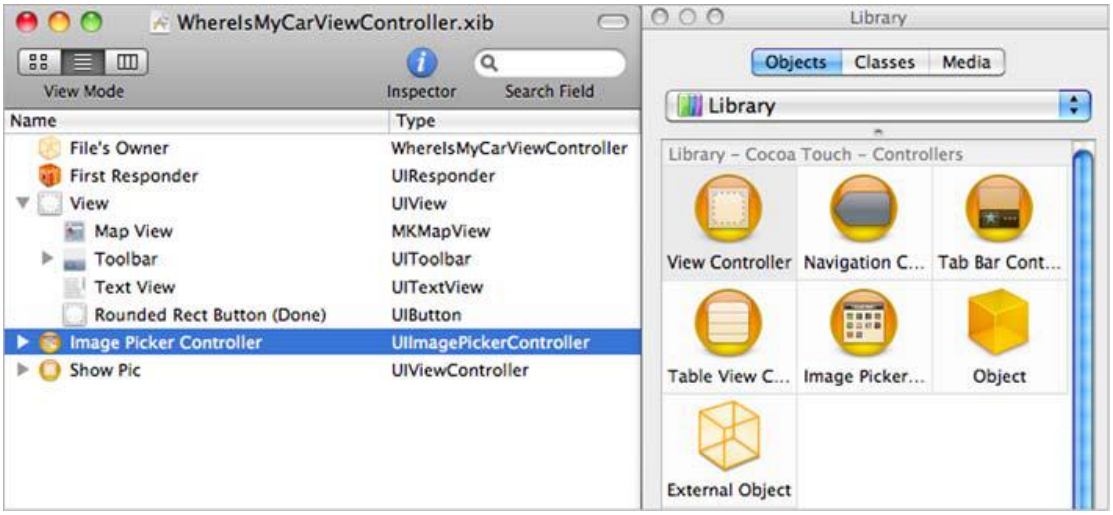


图5.15 给工程的UI添加一个image picker

现在需要把这些items实现oulets和actions,然后把他们添加到WhereAmIParkedViewController头文件。点击XIB中的File’ s Owner,然后查看所有连线。你可以看看到新的Image Picker Controller，View Controller，ImageView，以及图片的新的action。用IB连线。（见图5.16）

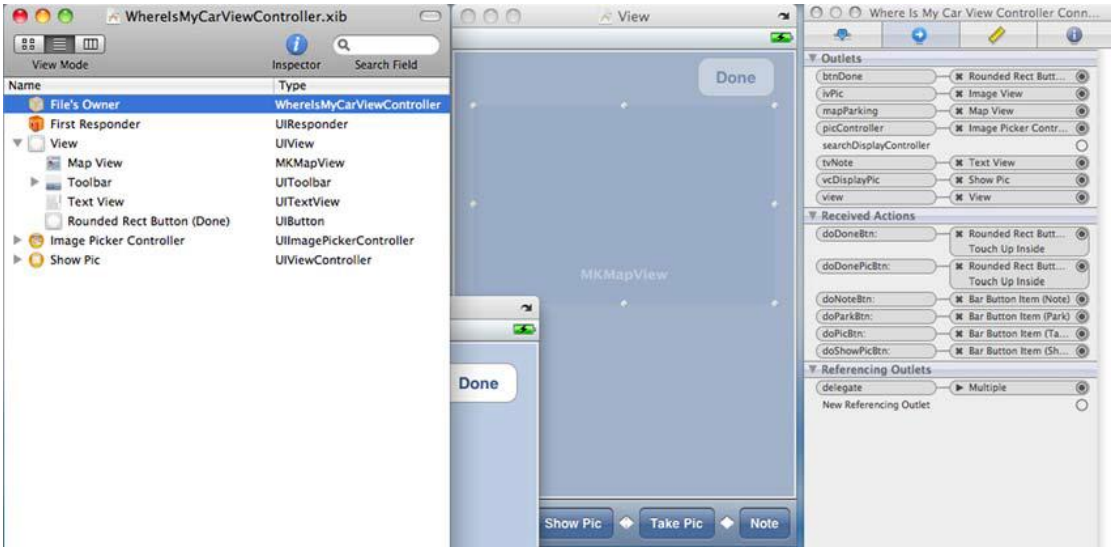


图5.16 使用Interface Builder连接图片相关的outlet以及action

技巧24 接收并储存图片

让用户可以使用照相机拍照很不错，但是如果拍了照之后就无法看到照片的话那就没什么用了。Image Picker Controller的类调用了两种方法。一个方法是用户点击取消，第二个方法是用户选择图片或者拍照。

问题

你需要从camera controller获取并且储存一张图片。

解决方案

实现这个功能需要几个步骤。当用户点击按钮的时候要调用照相机。这个类是camera controller的delegate，所以当用户拍照或者取消拍照的时候你还需要接入方法。然后，如果拍了照，你需要在设备中储存照片，供以后调用。

讨论

WhereAmIParkedViewController有一个叫做picController的image picker controller的实例。doPicBtn方法可以用它接入camera control。见以下代码：

在Pic按钮的action方法中接入照相机

```
-(IBAction)doPicBtn:(id)sender;
{
if ([[[UIDevice currentDevice] model]
rangeOfString:@"Sim"].location == NSNotFound)
    //在模拟器中使用相册
[picController setSourceType:
UIImagePickerControllerSourceTypeCamera];
[self presentModalViewController:picController animated:YES];
}
```

Camera control显示了之后，用户就可以用它来拍照了（见图5.17）：

图5.17 使用image picker controller camera拍的照片

用户的操作是通过Picker controller的 delegate WhereAmIParkedViewController来执行的。因为WhereAmIParkedViewController接入了UIImagePickerControllerDelegate，所有我们需要考虑两种方法：

imagePickerControllerDidCancel: 以及

imagePickerController:didFinishPickingMediaWithInfo:（见以下代码）。

Cancel方法告诉app用户点击cancel，需要把image view picker controller移除。另一个返回照片数据储存。储存图片还需要将图片缩小，然后再确定文件的储存位置。

使用照相机拍照之后获取图片，缩小图片并存储

```
+ (UIImage*)imageWithImage:(UIImage*)image
scaledToSize:(CGSize)newSize;
                    //缩小图片
{
UIGraphicsBeginImageContext( newSize );
[image drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
UIImage* newImage = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
return newImage;
}
-(NSString*)imagePath
{
NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
NSString *documentsDirectoryPath =
[paths objectAtIndex:0];
NSString *imgPath = [NSString stringWithFormat:@"%@/WIMCApp/",
documentsDirectoryPath];
                    //获取文件存储路径
NSError *err;
[[NSFileManager defaultManager]
createDirectoryAtPath:imgPath withIntermediateDirectories:YES
```

```
attributes:nil error:&err];
                //创建dir
return [NSString stringWithFormat:
@"%@WIMCpic.png", imgPath];
                //图片文件名
}
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
UIImage *pic = [info
objectForKey:UIImagePickerControllerOriginalImage];
pic = [WhereAmIParkedViewController imageWithImage:pic
scaledToSize:CGSizeMake(320.0, 460.0)];
[UIImagePNGRepresentation(pic)
writeToFile:[self imagePath] atomically:YES];    //append文件名
[self dismissModalViewControllerAnimated:YES];    //移除照相机
}
- (void)imagePickerControllerDidCancel:(UIImagePickerController
*)picker
{
[self dismissModalViewControllerAnimated:YES];    //移除照相机
}
```

你可以硬编码文件名，因为下一次用户停车的时候，他们就不再需要之前的图片了，可以覆盖之前的图片。但是如果用户下一次停车记录不拍照，图片会还是上一次拍照的停车地点。

现在已经可以拍照和储存照片了，让我们接入显示照片的功能。

技巧24 显示图片

显示出用户拍下的停车地点的图片更加方便用户寻找停车地点。在技巧23里，学会了储存用户拍摄的照片。把照片显示出来，需要从储存路径获得图片，然后在OS中创建可以展示图片的项目。你可以使用ImageView和之前添加到工程的按钮。

问题

你需要把储存的图片展示给用户。

解决方案

有了image view，按钮，使它们功能完备的outlet和action之后，展示一张存好的图片就非常简单了。你需要载入这张图片然后把这张图片置入image view显示。同时，你还需要一个当用户看完照片之后让图片消失的方法。

讨论

WhereAmIParkedViewController有个定义过了的outlet，这个outlet和用来展示图片的view controller连线好了，这个outlet的名字是vcDisplayPic。doShowPicBtn只需要加载图片然后让图片在controller中显示（见以下代码：）

加载储存的图片并显示图片

```
-(IBAction)doShowPicBtn:(id)sender;
{
UIImage *pic = [UIImage imageWithContentsOfFile:[self imagePath]];
if (nil == pic)    //如果没有图片则返回
return;
[ivPic setImage:pic];    //显示图片
[self presentModalViewController:vcDisplayPic animated:YES];
}
```

当用户点击按钮的时候，储存了的图片会显示出来（见图5.18）

图5.18 应用中显示储存好了的照片

让图片消失更加简单。你只需要让 这个controller以显示的方式让它消失就可以了，调用[self dismissModalViewController

Animated:YES]中的smissModalViewControllerAnimated:。

好，我们的应用完成了。功能完备非常有用。如果你没有做完，你可以在iTunes看看成品：http://itunes.com/apps/WhereAmIParked.

5.4 总结

创建一个有用的应用需要涉及 SDK 多方面的多系，包括地图，位置，拍照，存储图片，存储备忘，涉及 UI 以及完成 UI 的相关功能。这些功能在很多应用中都以各自不同的方式呈现出来，你可能也可以想到使用这些功能的其他方式。

由于 iPhone 是移动设备，许多应用都能从附加的地图上受益。在 MapKit 中提供了很多的功能。通过好的使用这些功能可以让你做出一个真正有意思的应用。

由于 iPhone 上有一个内置地图应用，我么可以基于这个地图应用制作其他很多的应用。类似的，其他内置 OS 系统的还有时钟应用。它包括的计数器，倒数计时器，闹钟以及其他功能，在下一章中，我们会探讨如何基于相似的概念，但是做一个在某些方面更加专业的应用。

6

——通过制作一个上架应用TimeDown教你设置，音频，以及晃动检测

本章包括：

- 用户设置
- 音频使用
- 设备运动检测

不同的应用 采取不同的方式与用户交互。最常见的方式是可视化交互。据我所知，没有一个应用是没有可见 UI 的，无论他是通过制作应用的选择框，重力感应，放置按钮，或者点击，滑过屏幕，应用可以使用很多交互方式。

许多时候，这些交互作用会影响应用的功能。有些设置可以改变应用的外观——点击按钮可以改变音量大小，晃动设备可以让一些游戏（例如拼图）变乱。同样，应用可以给用户反馈，就像来自用户的反馈形式多样话一样，应用的反馈也有多种形式，可视化的反馈可能是最常见的方式了，第二常见的是音频反馈。

在这一章中，我们会学习这些概念，剖析用户如何可以访问这些交互方式以及在应用中是如何使用它们的。接下来我们需要创建一个简单的应用，这个应用会使用 iPhone 自带的系统设置中的设定值。我们会使用 audio framework 来播放一个音频文件，以此作为音频反馈，还会使用加速计去检测设备的晃动，并且在 UI 中使用它们。

你可能对 iPhone 自带的系统设置不是很熟悉，我相信这是因为使用 iPhone 的系统设置来对应用进行设置并不好用，而且如果用户需要切换到系统设置去改变应用里面的东西的话，用户体验并不好。此外用户可能需要重启应用，才能启用新设置。正是由于这些原因，

大多数应用都会在本身的程序内进行设置。 但它是一个很重要的问题，因为它不容易，所以在本章中我们要好好讲讲这个问题。

播放音频文件，从另一方面来说也是比较常见的，也很容易实现。在很多应用中它是一个重要且常见的功能，而且它开发起来相较容易。

最后，我会说说圆角，它让你的 UI 更完美，创建一个试试吧，它让你和 iOS 的整体 UI 设计风格统一，特别是对输入框来说，输入框大多是圆角的。

图 6.1 TimeDown 在 iPhone 系统自带的设置应用中显示

让我们把这个应用命名为 TimeDown，它是一个倒数计时器，用于显示系统设置中计时器的时间剩余。

许多应用在程序内部会有设置的地方，”官方的”做法是在你的应用内部使用一个 settings bundle，这意味着将在你的应用内部进行设置。

设置/选项 (SETTINGS/OPTIONS) (引自苹果交互指南) 设置应该代表着应用的信息，比如说账户名，用户一旦设置了一次，就会很少改变。而一些面向应用的的设定用户要

在系统自带的设定应用中查看。用户可能会想要经常变换配置选项，如在列表中的类别，配置选项应该让用户可以在应用中进行变换。

图 6.2 TimeDown 的 UI 界面

TimeDown 的 UI 如图 6.2 所示，我们会添加一些额外的功能，让它更有用。在计时器运行以及晃动的时候（如果可用），我们将使用音频框架去播放一段短音频文件。同时，我们要使用加速计（accelerometer）去检测晃动，允许用户开始或者重置计时器。

让我们首先看一下如何添加使用项目中的设置。

6.1 iOS 项目中的 Settings bundle

项目的设置通过 Settings bundle 中的 plist 文件添加到项目中，这些设置允许用户通过这个文件去设置一些关联值。

苹果提供了这种机制去创建一个应用内的各种设置。使用 Settings bundle 的时候设定值以及条目是受限的，这些设置应该是系统设置可以理解的，并且可以展现给用户。

我们的应用设置应该包含一个默认的时间限制（按分钟显示），和一个自动开始的按钮

杆。我们需要一个使用数字键盘输入的数字和一个启动开关按钮。

技巧 35 给一个 Xcode 工程增加设置

我们已经讨论了我们项目应用要做什么，现在让我们开始创建项目，添加 Settings bundle。

打开 Xcode ,创建一个名字为 TimeDown 的项目(使用 View-Based Application)，创建完成后，我们就可以为项目添加 Settings bundle 了。

问题

需要为我们项目创建一个 Settings bundle 用于单独的用户设置，Settings bundle 包含一个 plist 文件，Xcode 允许编辑字段和值。设置中必须包含一个文本字段表示分钟，以及一个切换开关用于控制自动启动。

解决方案

右键项目名字，在项目中的 Groups & Files 选择添加-新建文件。在 Resource 界面中，选择 Settings Bundle 并且点击 OK（如图 6.3）

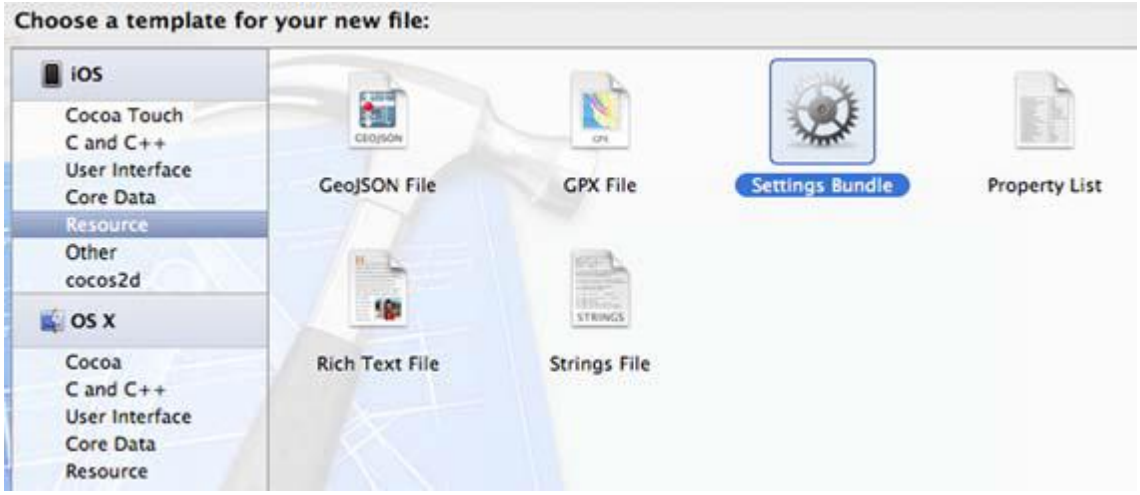


图 6.3 给 Xcode 工程添加 Settings bundle

讨论

可以给 Settings bundle 起任何你喜欢的名字,我使用的是系统的默认名字——

Settings（设置）。

在模拟器中运行应用，然后退出。 现在去系统自带的设置应用中查看，可以看到TimeDown 区域已经有了默认值了（如图 6.4）

图 6.4 TimeDown 的默认 Settings bundle 中的设定

让我们设置一些对我们的应用有意义的值。打开Setting.bundle选择Root.plist.。点击展开Preference Items 。

在 Preferens Items 的展开项中，有 4 个项目（见图 6.5）。第一个定义了 group.，接下来的三个是设定值，我们可能只需要 2 个（见图 6.6），所以可以把第三个删除。

Key	Type	Value
▼Preference Items	Array	(4 items)
▼Item 0 (Group - Group)	Diction...	(2 items)
Title	String	Group
Type	String	Group
▼Item 1 (Text Field - Name)	Diction...	(8 items)
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Default Value	String	
Text Field Is Secure	Boolean	NO
Identifier	String	name_preference
Keyboard Type	String	Alphabet
Title	String	Name
Type	String	Text Field
▼Item 2 (Toggle Switch - Enabled)	Diction...	(4 items)
Default Value	Boolean	YES
Identifier	String	enabled_preference
Title	String	Enabled
Type	String	Toggle Switch
▼Item 3 (Slider)	Diction...	(7 items)
Default Value	Number	0.5
Identifier	String	slider_preference
Maximum Value	Number	1
Max Value Image Filename	String	
Minimum Value	Number	0
Min Value Image Filename	String	
Type	String	Slider
Strings Filename	String	Root

图 6.5　默认的 Settings bundle 值

图 6.6 在系统设置中 TimeDown 的实际设定

现在需要对剩下的值做一些改动使得这些值适合我们的应用 TimeDown。我们需要把 Group Title 的值赋给 TimeDown。第一个设置元素将会有一个新的标题，标示符，默认值以及键盘类型。最后一项设置也需要有一个新的标题以及标示符，使设置符合如图 6.7 所示。

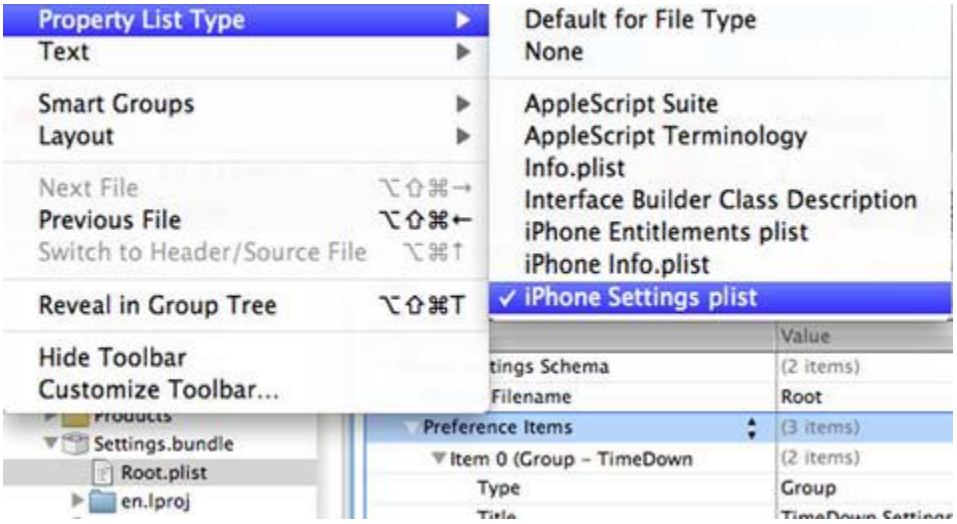


图 6.7 TimeDown 的设定 Root.plist 的值

现在可以运行应用程序，然后退出。打开系统设置，就可以看到项目的最新设置了。如果不运行系统应用设置，项目中使用的那些值不会生效。换句话说，如果系统设置没有访问 Settings bundle，则没有至被赋给应用的设定。当应用想要访问默认的计时器的值的时候，它不会是 5，因为系统设置从未对它进行设置。

为了以防用户不运行系统设置，这是很有可能出现的情况，我们需要检查丢失的设置。如果他们不存在，我们需要在应用内部设置一个默认值或者初始化设置，模拟系统设置。接下来让我们看看如何初始化默认设置，防止用户没有运行过我们的应用设置。

技巧 36 初始化 iOS 设置

如果我们在系统设置中打开过 TimeDown 的设置程序，那些值就会被自动的设置上（即使我们没有改变它们）。但是如果你（或是应用程序用户）没有打开过系统设置，那些值就不会被设置上。

通常情况下，当用户下载了一个应用，他们会先在应用的设置中查看可配置的东西，而

不是在系统设置中查看。用户很可能从来不运行系统设置来对我们的应用进行设置。而恰恰是这些在 Settings bundle 中的设置对我们的应用可以正确运行起了至关重要作用。如果用户没有设置他们，我们仍然需要那些值，在这种情况下，我们可以在应用内进行设置。我们不仅可以把他们设为默认值 ,还可以模拟使它们看起来好像系统设置已经对他们进行过设置了。

问题

需要初始化我们的应用需要的设定值，防止用户不在系统设置进行设置的情况。

解决方案

检查我们的应用设置情况，设置所有需要的值。

讨论

通过系统的 NSUserDefault 的实例访问那些设置的值。 NSUserDefault，跟 NSDictionary 一样，允许通过 key 来存贮数据。

如果通过 NSUserDefault 获取到的设置值为 nil ,那就意味着设置既没有在系统设置中存储也没有在我们的应用设置中存储。

为了设置需要的默认值，可以创建一个 NSDictionary（或者 NSMutalbleDictionary 存储可变的值），并且注册一些初始值。可以在 TimeDownAppDelegate.m 的类中的 applicationDidFinishLaunching:方法中添加如下代码：

在TimeDownAppDelegate中初始化应用的设定

```
NSUserDefaults *settings = [NSUserDefaultsstandardUserDefaults];

NSObject *timeSettings = [settings objectForKey:@"timeSettings"];
if (nil == timeSettings)
{
NSDictionary *appDefaults =
```

```
[NSDictionarydictionaryWithObject:@"5"
forKey:@"timeSettings"];
[settingsregisterDefaults:appDefaults];
[settings synchronize];
}
```

如果没有设置自动启动，当访问的时候会返回 NO，也就是没有默认设置，这也是可以的。

在这个例子中，很容易进行测试。如果你的计时器的设定是 nil，那么就对它进行设置。在其他情况下，或许需要采用更有创意的方式，或者我们也可以用一 NSMutable-Dictionary 去存贮更多的数据。无论哪种方式，当所需值被设定后，就可以通过我们的应用去访问它们。接下来让我们看看是如何实现的。

技巧 37 在应用中访问设置的值

那些存储在 NSUserDefault 的实例中的值，会定期的写入磁盘，而且会持续存在于一个执行方法到下一个执行方法。这意味着这些我们设置的值稍后是可以访问的，并且这些值会影响到执行方法。

除了 Settings bundle，你或许希望 NSUserDefault 自动存贮一些 Number 类型的值，例如最后执行时间，GPS 坐标,以及其他的一些数值。我们的应用只存储了 2 中基本类型的数据，而 NSUserDefault 可以存储大量的数据。

NSUSERDEFAULTS (引自苹果官方文档)

NSUserDefault 给访问常见类型的数值提供了方便的方法，可以访问的数值类型包括单精度浮点数，双精度浮点数 ，整数， boolean 值，以及网址。默认的对象必须是一个属性列表，也就是说只能是如下几个类别的实例（集合类什么的）：NSData, NSString, NSNumber, NSDate, NSArray, NSDictionary。如果想要存贮其他类型的对象，通常需要

把他们转成 NSData。

就跟检查前文提到过的 Settings bundle 里的值一样，这些值可以直接访问。只需获取到 NSUserDefault 的实例，就像使用 NSDictionary 一样通过 key 取 value 那样使用就可以了。

问题

需要访问那些在 TimeDown 应用中使用的设置数据。

解决方案

通过对 NSUserDefault 的实例进行检索，可以获取到与你的 key 值对应的那个 NSDictionary。

讨论

通过 NSUserDefault 获取到 dictionary 后，可以通过设置在 Root.plist 中的标示符去获取想要的数据。

通过标示符 “time- Settings”来对定时器的时间长度（分钟）进行设置。通过标示符 “autoStart” 来对自动切换的开关进行设置。我们可以访问这些设置，在viewWillAppear中通过NSLog输出出来，如下所示：

访问并打印出设置的值

```
-(void)viewWillAppear:(BOOL)animated
{
NSUserDefaults *settings = [NSUserDefaultsstandardUserDefaults];
timeSettings =
[[settingsobjectForKey:@"timeSettings"] intValue];    //存储为String

autoStart = [[settings objectForKey:@"autoStart"] boolValue];
```

NSLog(@"timeSettings: %d autoStart: %d", timeSettings, autoStart);

}

现在运行应用，就可以看到控制台输出的值了。运行系统的应用程序设置，并且修改其中的值，当你再次运行应用的时候，控制台输出的就是你改变后的结果了（见图6.8）。

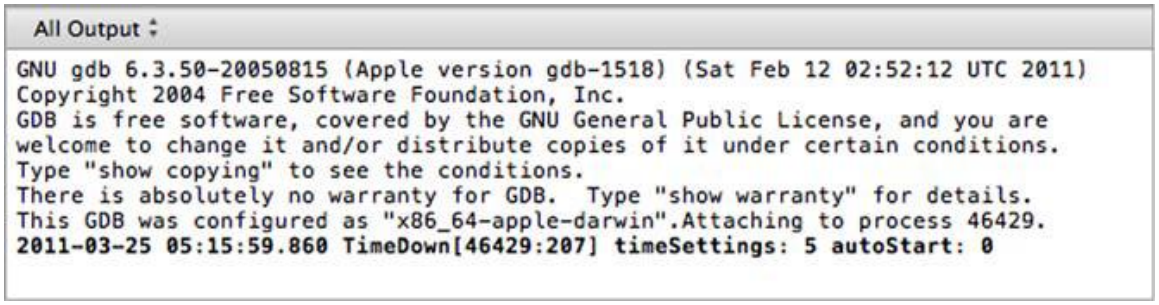


图 6.8 控制台输出的应用设置解决方案

在模拟器中识别TimeDown的设置项很简单，因为就那么几个。接下来我们会给这个设置添加一个图标，当运行在真实环境中的时候会更清晰的识别这个设置。

技巧38 设置应用的设定图标

苹果 在官方文档中明确的解释了哪些图标是必须创建并且使用的，详情请查看：http://developer.apple.com/library/ios/navigation/

在“Human Interface Guidelines”中”搜索“Custom Icon and Image Creation Guidelines”。

所有的应用在设备上都有一个图标，图标的大小在iPhone/iPod Touch 57*57,在iPad上是 72*72。还有其他的一些图标你可以在你的应用中使用，每种图标的大小，解析度以及命名规范都需要符合上文提及到的文档。

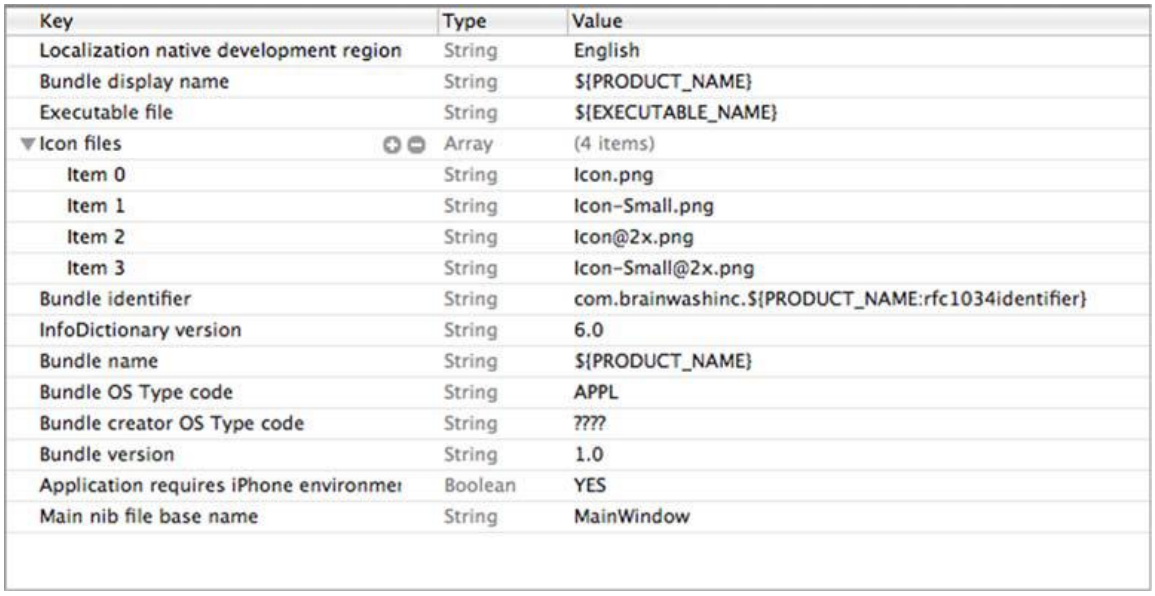
问题

如果不为应用设置添加图标，在系统的设置列表中只会显示应用名字。那不仅是显得不专业的问题，而且对于用户来说，在长长的列表中找到应用名字也不是一件容易的事。所以现在让我们给应用设置添加一个图标。

解决方案

系统设置中使用PNG作为图标，普通的是29*29，高分辨率的是58*58，我们需要创建这种规格的图标，把它添加到项目中去，设置CFBundleIconFiles，并且把它命名为Icon-Small.png。

把Icon- Small.png.拖进你的项目的Resources中，打开TimeDown-Info.plist 查看icon setting 设置。右键其中一个key，选择 Show Raw Keys/Values， 查看key的名字，在很多时候你需要添加@2x的图片去适应高分辨率的机器（如图6.9）



Key	Type	Value
Localization native development region	String	English
Bundle display name	String	${PRODUCT_NAME}
Executable file	String	${EXECUTABLE_NAME}
▼ Icon files	Array	(4 items)
Item 0	String	Icon.png
Item 1	String	Icon-Small.png
Item 2	String	Icon@2x.png
Item 3	String	Icon-Small@2x.png
Bundle identifier	String	com.brainwashinc.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	String	6.0
Bundle name	String	${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environmer	Boolean	YES
Main nib file base name	String	MainWindow

图6.9 TimeDown-Info.plist items 的 keys/values值

如果字段CFBundleIconFiles不存在，创建它，右键点击，并把它的数据类型设置为Array 。为应用添加图标（例如Icon.png），系统设置中使用的图标（Icon-Small.png）为你的应用创建一个57*57的图标并把它添加到项目中命名为icon.png。

在苹果 官方文档上明确的定义了这些图标的大小，分辨率，命名规则以及其他需要与支持的各种设备类型的图标。

现在我们已经为我们的应用设置了 Setting bundle ,并且在初始化代码中设定了缺省的默认值，通过使用应用的 plist 文件中的Array类型的入口，我们可以制定在设备上使用的各类图标。但是在应用内部，这些图片都是不可见的。

6.2 运行时以及以时间为基础的UI变化

有时候，通过Xcode中的Interface Builder (IB) 去设置UI是最好的方式，有时候直代码写UI更好。很多情况下，我们需要通过代码去更新UI，不是基于用户动作来更新，而是基于时间。

Interface Builder (IB) 提供了很多可视化修改UI设置的方法，但并不是所有方法。如果想操作一些底层的设置，则需要在代码中操作。圆角在iOS的应用中是很常见的（参考联系人列表中的grouped的列表） 。如果想在应用中运用这样的效果，我们可以使用图片。但是还有一种更好的方式，可以在代码中实现这样的效果。

有一些属性可以很简单的在Interface Builder (IB)中设置，还有一些需要依照应用的状况而定。TimeDown是一个计时器应用，所有的用户都期望直观的看到时间的减少，接下来我们会处理在Lable中显示剩余时间，做到每秒去更新Label一次。

技巧39 圆角的视图

就像前文提到过的，很多应用都使用圆角的视图。打开系统设置，它呈现了使用了圆角的分组列表。接下来我们就会把我们的页面做成图6.2所示的那样，使用圆角。

Interface Builder (IB) 并没有提供方法去访问UIVIew的属性CALayer，它恰恰是设置圆角的方法，我们可以通过代码的方式去设置它。接下来让我们看看如果把应用视图设置为圆角。

问题

需要在应用中使用圆角，既不使用图片，也不使用其他复杂的方式。

解决方案

通过调用 Calayer 底层的方法去设置圆角，这些底层方法无法通过UI编辑器访问。

讨论

首先先为我们的应用创建UI。 打开默认的xib文件 TimeDownViewController.xib。添加一个黑色背景的view。添加2个 label 并中心对齐 ,调整lable的大小 ,字体(如图6.10)

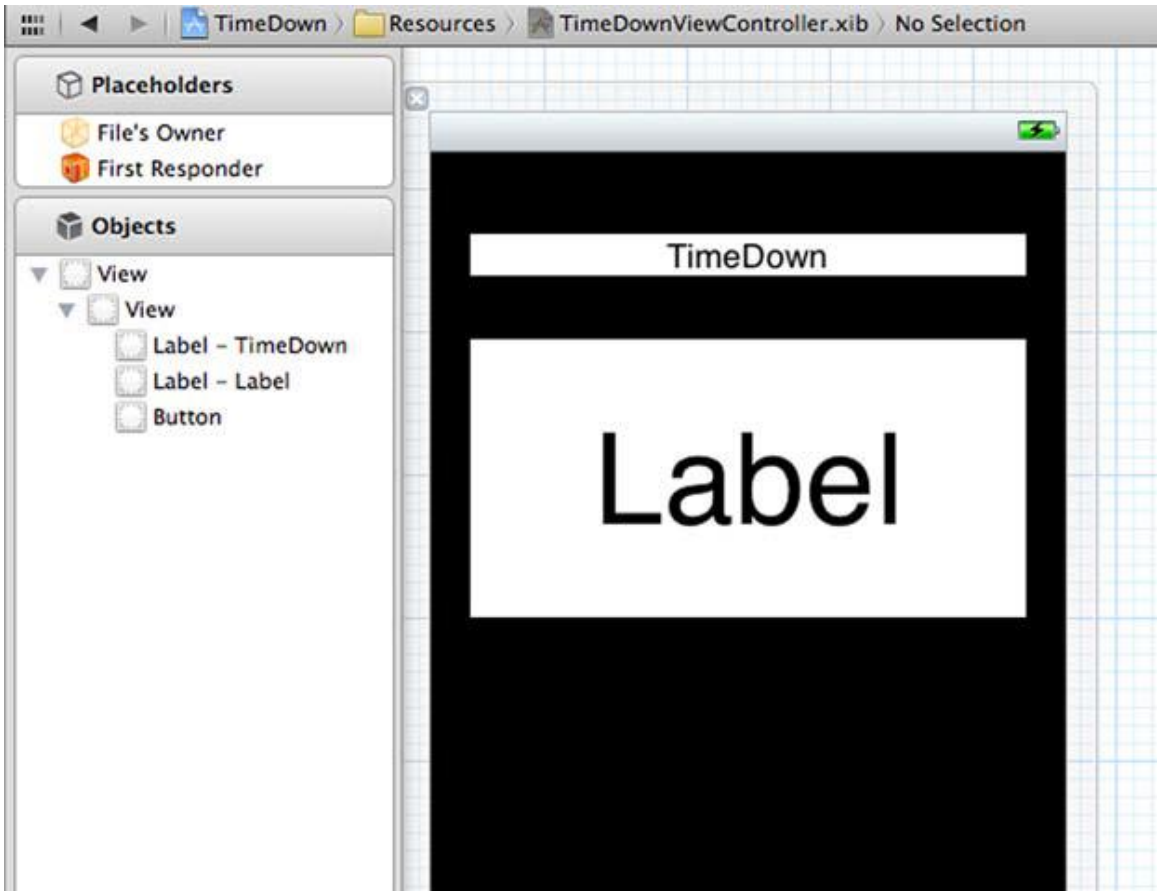


图6.10 TimeDown的UI设计

在项目中添加QuartzCore framework，在TimeDownViewController.h中导入。还是在头文件中声明UILabel outlets ，并且把它们与xib连接起来，如下：

导入QuartzCore.h文件，然后声明UILable outlet

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

#import <QuartzCore/QuartzCore.h>

@interface TimeDownViewController :UIViewController {

IBOutletUILabel *lblTimer, *lblTitle;

}
```

@end

由于我们并没有对这个Label的文本进行任何操作，应用看起来很土，也没有圆角。事实上它就长成6.10所示的那个样子。

我们可以通过使用QuartzCore framework 里面的setCornerRadius的方法来设置圆角。

需在视图显示之前添加如下的代码到需要调用的方法中，如viewWillAppear：方法。

通过QuartzCore把UIlabel变圆角

```
-(void)viewWillAppear:(BOOL)animated
{
    [superviewWillAppear:animated];
    [lblTimer.layer setCornerRadius:5.0];
    [lblTitle.layer setCornerRadius:5.0];
}
```

再次运行应用的时候就会发现圆角已经出现了。见图6.11

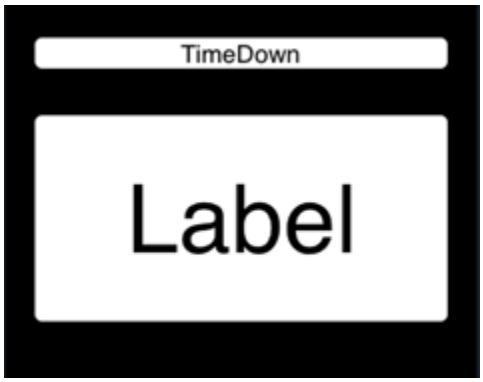


图6.11 UI没有更新时在模拟器中的TimeDown

你可以试试为圆角设置不同的值，以便得到最好的显示效果。在这个例子中修改圆角操作只需要做一次，有时候你会想多次修改这些值，接下来让我们看看如何通过计时器去修改。

技巧40　用重复的计时器来更新UI

计时器应用是一个按时间间隔修改UI的很好的例子，我们可以使用NSTimer类来帮助我们知道什么时候应该更新UI。

计时器可以在特定的时间段启动一次，或者重复的在一个时间段启动多次，那恰恰是我们的应用想要的效果。

问题

需要创建一个计时器，并且每秒调用一次更新UILabel的方法，显示剩下多少时间。

解决方案

创建一个NSTime的实例，选择一些设置来完成UI更新。将它设置为每秒重复一次，并调用oneSecond：方法更新UI。

讨论

首先，我们需要在头文件中创建一个NSTimer类型的成员变量。还需要在dealloc的方法中释放它。

让我们回到viewWIllAppear :方法 ,现在已经从系统设置中读取到了默认的timeStting了，它以分钟为单位，所以我们需要用它乘以60转换成秒。然后把计时器的Label 设置为空白。如果再把自动启动打开的话，计时器就创建好了，代码如下：

设置时间，清空label，如果autoStart打开的话创建计时器

```
timeSettings *= 60;
    [lblTimersetText:@""];
if (autoStart)
setTimer = [NSTimer
    scheduledTimerWithTimeInterval:1.0     //每秒执行一次
target:self
    selector:@selector(oneSecond:)        //调用oneSecond:
userInfo:nil
repeats:YES];                          //是否重复执行
```

我们的计时器将每秒会触发一次，调用oneSecond的方法。在oneSecond方法中，我们需要用剩余时间去更新Label。此外还需要判断剩余时间是否为0，如果是，就停止计时器，代码如下：

减少剩余时间，升级lable，完成之后停止计时器

```
-(void)oneSecond:(NSTimer*)timer;
{
timeSettings--;
if (timeSettings> -1)
        [lblTimersetText:
            [NSStringstringWithFormat:@"%d:%02d",
timeSettings/60, timeSettings%60]];
else {
[secTimer invalidate];                    //释放引用
secTimer = nil;
} }
```

如果剩余时间大于0，更新label，否则让计时器失效（并且释放它）并把member设置为nil。

打开系统设置，把time 设置为1，打开自动启动，运行应用。现在可以看到应用启动的时候，计时器启动了，下面的lable（如图6.12）的数字在不短的减少。

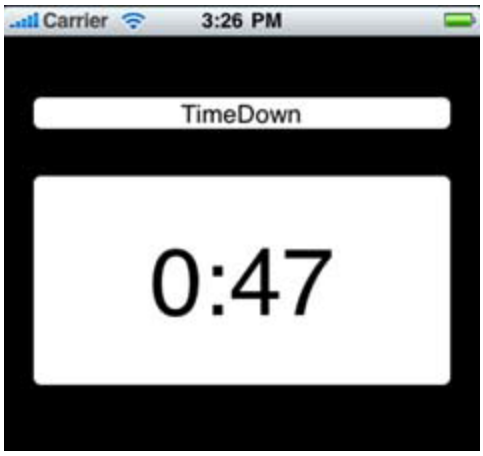


图6.12 auto-start启动时候的TimeDown

现在你知道了如何在应用中创建UI并且一次或者多次（以一定的时间规律）更新它。当计数器时间用尽的时候，尽管应用做了一些操作，但是它并没有通知用户。在下一节中，我们将看看如何通知用户时间用尽了，包括播放音频文件。

6.3 播放音频并设置振动

苹果 在过去几年中的成功，开始于 iPod 音频播放。iOS 提供了很多方式去播放音频，包括访问 iPod 的功能。下面我们来看看几个实现这个功能的基本方式。

在应用中使用音频文件就跟使用图片或者其他类型的文件一样简单---直接把它拖进项目的 Groups & Files 就可以。播放文件并不复杂。

AVFoundation framework 提供了 AVAudioPlayer 来播放音频，这个类库帮我们做了绝大多数的事。你只需要告诉它要播放哪一个文件，什么时候播放就好了。当然还有其他的一些音频播放的类库，都很简单。

问题

需要在计时器结束的时候播放音频文件，提醒用户计时器时间已经到了。

解决方案

使用 AVFoundation framework 的 AVAudioPlayer 类，创建一个实例，利用设备播放项目中的 MP3 文件。

讨论

想用 AVFoundation framework 的 AVAudioPlayer，首先需要把 AVFoundation framework 添加到项目中。然后还需要把 AVFoundation.h 导入到 TimeDownViewController 的头文件中，并且声明一个 AVAudioPlayer 的实例（例如 aAudioPlayer），当然不要忘记在 delloc 方法中释放它。

下一步需要把 MP3 文件添加到项目中去。这里我创建了一个 MP3 ,叫“beeps.mp3”。把它拖到项目中的 Resources 组里。

让我们写一个单独的方法参数为 filename（没有扩展名），创建一个 AVAudioPlayer 实例，并播放文件，代码如下所示：

Filename 方法，播放 MP3

```
-(void)playSound:(NSString*)soundFileName
{
NSString *aFilePath = [[NSBundlemainBundle]
pathForResource:soundFileNameofType:@"mp3"];
if (nil != aAudioPlayer)
    {
        [aAudioPlayer stop];     //在释放前需要先停止播放器
         [aAudioPlayer release];
    }
aAudioPlayer =[[AVAudioPlayeralloc]
initWithContentsOfURL:[NSURL fileURLWithPath:aFilePath]
error:NULL];                    //创建MP3文件
```

```
        [aAudioPlayer play];
    }
```

如果已经有了一个播放器的实例，停止播放，然后释放它。AVAudioPlayer 的初始化方法需要一个 NSURL 指向你的文件（在 NSBundle’ smainBundle 中），在创建完毕后，可以调用播放器的 play 方法播放。

当定时器跳出方法 onSecond：时 ，调用 playSound：方法（参数为 name）播放 beeps.mp3 文件。代码如下：

定时器时间用尽之后调用playsound：方法

```
-(void)oneSecond:(NSTimer*)timer;
{
timeSettings--;
if (timeSettings> -1)
            [lblTimersetText:
                [NSStringstringWithFormat:@"%d:%02d",
timeSettings/60, timeSettings%60]];
else {

[secTimer invalidate];
secTimer = nil;
[selfplaySound:@"beeps"];
} }
```

去通知用户一个事件，这是一种很漂亮的方式。尤其是定时器或者警报事件。通知用户的另一种方式是通过设备的振动。让我们看看怎样做到这一点。

现在绝大部分手机都提供振动功能。对于手机的静音模式来说，振动简直太棒了。而对于一些游戏或者应用中的警告以及反馈来说，振动也是非常棒的交互体验。接下来我们会使

用振动作为计时器结束时的响应。

并不是所有的 iOS 设备都支持振动的，当你想在 iPod，iPad 上执行振动指令的时候，什么都不会发生。

问题

TimeDown 计时器结束的时候我们希望 iPhone 能够振动。

解决方案

用 AudioToolbox framework 去播放系统声音并振动。

讨论

与播放 MP3 文件（技巧 41）类似，我们将创建一个单独的方法用于使设备震动。当想要设备振动的时候，就调用这个方法。

首先把 AudioToolbox framework 加入到项目中，并在 TimeDownViewController 头文件中引入 AudioTool- box 的头文件，然后创建一个调用 AudioServicesPlaySystemSound 的振动方法，指定系统声音 kSystemSoundID_Vibrate，代码如下：

振动设备的同时播放系统声音

```
-(void)vibrate;
{
AudioServicesPlaySystemSound (kSystemSoundID_Vibrate);
}
```

现在调用振动方法 playSound:当计时器结束的时候，应用就会播放 MP3 并且振动了。

看到了吧，播放音频文件并让设备振动是多么简单啊。但是，当用户移动设备的时候会发生什么呢？接下来我们会来看看如何使用加速计去检测设备晃动，并在我们的应用中使用它。

6.4 检测并处理设备的晃动，

我们应用的UI比较简单。应用的设置是通过系统设置的settings bundle来处理的。主要输出的是一个带有漂亮的圆角效果的UILabl。

让我们继续完善这个UI。我们将通过使用加速计来检测并处理晃动，而不是通过按钮触发。

技巧43 使用加速计（Accelerometer）来检测晃动

我们可以在应用程序中使用UIKit framework的加速计（Accelerometer）来检测设备的震动。

对于不同的应用来说，通过加速计检测运动是非常有效的。有许多应用，晃动允许用户取消之前的操作。 接下来让我们看看如何检测晃动。

问题

我们想要在应用中检测设备的晃动。

解决方案

首先需要在应用中检测设备的运动，然后确认它是晃动还是只是简单的移动。

讨论

在TimeDownViewController的头文件中让你的类实现UIAccelerometerDelegate这个接口：

```
@interface TimeDownViewController :UIViewController
```

<UIAccelerometerDelegate>

然后在viewWIllAppear：方法中，设置共享的加速计的 delegate 为你的TimeDownViewController:

if ([UIAccelerometersharedAccelerometer].delegate == nil)
[UIAccelerometersharedAccelerometer].delegate = self;

当把这些加到viewwillAppear：以后，每次视图显示的时候（在某些应用中）它会被执行。这很多余，在这里它只会执行一次。

在头文件中声明prevAcceleration 作为UIAcceleration的引用。声明它的属性，在 实例（.m）文件中synthesize它，并且在dealloc 方法中释放它。

接下来写一个方法，使用2个UIAcceleration类型的参数 ，prevAcceleration，一个double类型的shakeThreshold 结束晃动的临界值，代码如下：

使用两个UIAcceleration类型的参数一级一个临界值来检测晃动的临界值

```
-(BOOL)shakingEnoughFromPrev:(UIAcceleration*)prevShake
                    toThisShake:(UIAcceleration*) thisShake
withThisThreshold:(double) shakeThreshold;
{
double dX = fabs(prevShake.x - thisShake.x);     //计算x
doubledY = fabs(prevShake.y - thisShake.y);
doubledZ = fabs(prevShake.z - thisShake.z);
```

```
return (dX>shakeThreshold&&dY>shakeThreshold) ||
                (dX>shakeThreshold&&dZ>shakeThreshold) ||
                (dY>shakeThreshold&&dZ>shakeThreshold); //当晃动的时候返回true
}
```

在这个加速计的回调方法里，会检测晃动如果是previous的UIAcceleration，那么把标志位设置为yes，如果不是previous的，那么会跳过此方法的调用，并设置prevAcceleration 代码如下：

检测晃动是否是previous的UIAcceleration

```
- (void) accelerometer:(UIAccelerometer *)accelerometer
didAccelerate:(UIAcceleration *)acceleration {
if (self.prevAcceleration)
    {
if (!isShaking&&
shakingEnough(self.prevAcceleration, acceleration, 0.5))
isShaking = YES;                //当晃动的时候设置为yes
    }
self.prevAcceleration = acceleration;
}
```

我们只想要在晃动的标示符为no的时候检测晃动，如果已经检测过晃动了，那么就不需要再确认一次了。

现在我们知道了如何基于临界值去判断设备是否在晃动。你的临界值是0.5，这对于一个特定的应用来说，可能大了，也可能小了，这取决于本身。根据应用的不同，或许需要在设备晃动的时候做很多不同的事情。让我们看看在TimeDown中是如何处理晃动的。

技巧44： 使用action sheet响应一个晃动

快速的呈现一个选项列表，其中包含选择或者确认的动作，这是一种很常见的做法。使用action sheet，是很常见的技术，它呈现给用户各种允许的选项并且当点击按钮的时候执行回调方法。

我们的应用比较简单，那就让我们通过晃动设备去重置计时器。我们使用一个actionsheet来确认用户想要重置计时器。

问题

我们需要提示用户，当他们晃动设备的时候，可以开始或者重置计时器。

当检测到用户晃动设备的时候，通过UIActionSheet 提供的一些选项让提示用户开始计时器。

讨论

现在我们可以知道用户在晃动设备，现在需要添加一个action sheet给予用户选择，并且对用户的选择进行处理。

要想处理action sheet，需要让TimeDownViewController 实现UIActionSheetDelegate接口。在把晃动标志位设成yes的地方，可以创建并显示一个action sheet，提示用户是否开始计时器，代码如下：

使用一个action sheet提示用户开始计时

```
IActionSheet *as = [[UIActionSheetalloc]
initWithTitle:@"Start Timer"
delegate:self
        //把delegate设置为self
cancelButtonTitle:@"Cancel"
destructiveButtonTitle:nil
otherButtonTitles:@"Start", nil];
            [asshowInView:self.view];
```

因为这里把delegate设置为了self，以当用户在action sheet上做了选择之后可以接入回调。见以下代码。我们可以通过按钮index来确定用户按了哪个按钮。在我们的应用中，Start button的index为0。检测到了这个之后，我们就可以启动（或重启）计时器。

根据用户的action sheet的选择开始/重启计时器

```
- (void)actionSheet:(UIActionSheet *)actionSheet
clickedButtonAtIndex:(NSInteger)buttonIndex
{
if (0 == buttonIndex)
{
```

```
[lblTimersetText:@""];
NSUserDefaults *settings =
[NSUserDefaultsstandardUserDefaults];
timeSettings =
[[settingsobjectForKey:@"timeSettings"] intValue];
timeSettings *= 60;
        //重设时间
if (!secTimer)
secTimer = [NSTimer scheduledTimerWithTimeInterval:1.0
target:self
selector:@selector(oneSecond:)
userInfo:nil
repeats:YES];
            //如果返回的是nil则重启计时器
}
isShaking = NO;     //重设晃动fag
}
```

在程序开始的时候把Label与time值重置为默认值，这对于计时器来说是很必要的。如果还没有一个计时器，你需要采用同样的方式，在viewwIllAppear：的方法中创建它。你还需要重置晃动标志，这样就可以检测之后的晃动。

现在在设备上运行这个应用 ,测试设备是否检测到晃动。当你运行它的时候 ,晃动设备，这时候一个action sheet出现了 ,选择Start ,action sheet 消失的同时 ,计数器开始工作，如图6.13

图6.13 Action sheet提示用户开始计时器

现在，尽管你还没有真正的交互式UI，但你的应用，功能齐全。它是一个计时器，使用Settings bundle并且通过加速计检测晃动。

6.5 总结

通过开发TimeDown应用，我们学习到了如何使用Settings bundle对应用程序的值进行设置。还知道了如何初始化，设置以及访问这些值，还有如何为应用添加在系统设置中的图标。

我们还知道了如何更新UI，包括圆角以及通过一个repeating的timer去控制UI的更新。当timer结束的时候，知道了如何通过播放一个声音文件，并且震动（如果设备支持的话）去通知用户。最后，我们学到了如何做运动检测，例如晃动以及如何使用接口去处理这个检测。

尽管有些方面用起来很简单，你可以看到他们如何工作的基本知识，以及他们可以做什么。但是在其他方面，需要一些有进一步的开发去处理复杂的事情。在应用中播放一个单独的MP3文件是很简单的，但是如果通过iPod 去播放用户的音轨就有一些复杂了，下一章我们会学习如何做到这一点。

7 ——通过教你制作一个上架应用 Playlist 来学习 CoreData，获取本地音乐并播放

本章包括：

- CoreData 框架
- 数据库的操作
- 创建及使用播放列表

iPhone，iPod 以及 iPad 都是非常优秀的音乐播放设备。用户还可以自行创建播放列表。如果你想编写一个能创建音乐播放列表的应用，通过访问设备中 iPod 内的音乐或者应用程序自己的数据库，用户可选择自己的音乐列表路径并创建个人的播放列表。

虽然创建播放列表非常实用，但是如何播放它们是首要问题。一个有用的播放器应用是使用 iPod 选择音乐路径，通过 CoreData 存储选择的内容，再通过音乐播放器播放音乐。

也许部分用户仅仅是将编写上述程序当做一次练习，因为这并不能完全代替 iPod 应用。但是通过学习本章节的知识，你可以编写出一个具有功能性的应用，并为以后编写更多更复杂的应用打好基础。

和很多数据驱动的应用一样，你需要通过一种方式存储数据，CoreData 就是一种很好的载体。你需要在 Xcode 中创建一个数据库，并生成相关代码，这样就可以用 CoreData 来存储和检索播放列表对应的数据。一方面这是创建应用的基础，另一方面鉴于前文已对 UI 基础有了一定的阐述，本章将着重于对 CoreData 概念进行讲解。

在设计完数据库并生成相应代码以后，接下来用户在作出选择后即可进入音乐程序。这个应用的 UI 部分很大程度上受限于系统所支持的框架，但是这个应用设计的定位和功能的关键是执行用户的选择并将其存储至数据库。

最后，我们还将研究如何在一个既定的播放列表内播放由用户自行选择和存储的音乐。这包含怎样处理

当一个音乐播放结束后继续播放音乐列表下一首音乐的回调通知。

让我们先看下如何设置一个支持 table view navigation 的工程。

7.1 创建一个 table view 工程

所有应用都要处理某些形式的数据，其中很大一部分还需要以一定方式存储数据。常用的解决方案是创建一个包含表格、表格行以及介于它们之间关系的数据库。

其中，常用的方法就是在一个列表中加入很多行来向用户展示数据。

技巧 45 创建带导航列表（table navigation）的工程

数据驱动应用一般需要一个允许用户选择一些元素和它们细节的界面。当使用 CoreData 存储和检索被储存在数据库的数据时，用户通过 UI 见到的往往是一个展现数据列表。当用户点击列表特定行时候，他们希望能见到从右划入的详细内容，而这个详细内容是 navigation controller 的一个功能。**问题**

需要创建一个工程，一个 table view，和一个 navigation controller。

解决方案

可以在 Xcode 通过 Storyboard 和 CoreData 创建一个 Master-Detail 表单应用。然后再创建一个在 navigation controller 嵌入 table view 的 UI。

讨论

打开 Xcode，选择创建一个新工程。从弹出的选择界面上选择 Master-Detail 应用，并点击 Next（见图 7.1）。

图 7.1 创建一个 Master-Detail 应用

紧接着是设置项目工程名称，可以设置为 PlayMyLists。还需要设置公司标识符（见图 7.2）。特别要注意的是这个项目我们使用 Storyboard 和 CoreData。值得注意的是 Storyboard 需要 iOS5.0 以上版本，否则你不能选择使用 Storyboard。

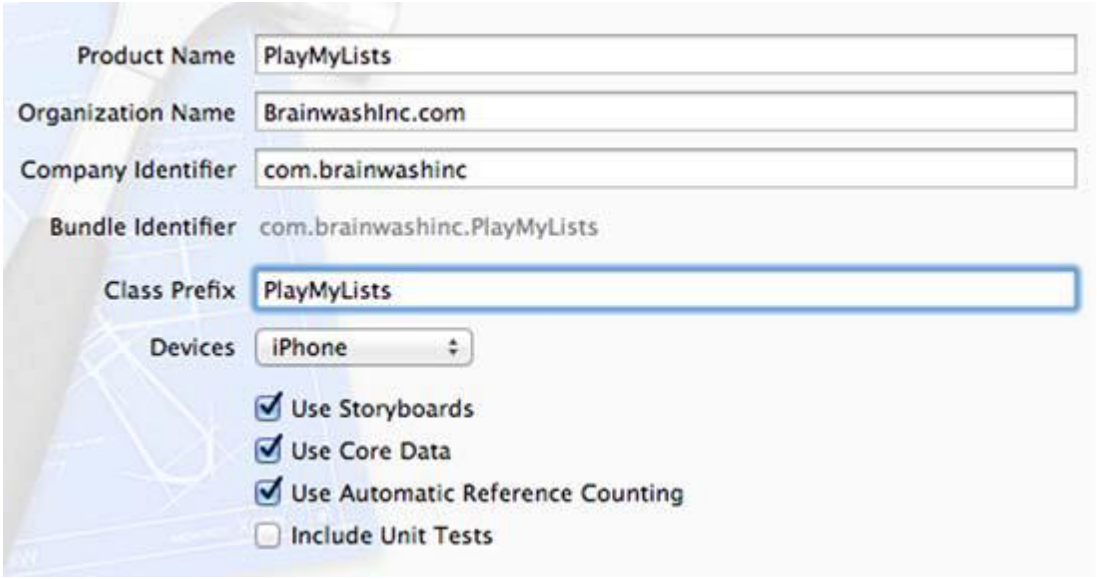


图 7.2 为工程选择 Storyboard 和 CoreData

接下来选择本地存储位置和点击 Create 按钮。这样就可以看到项目的概况了（见图 7.3）。

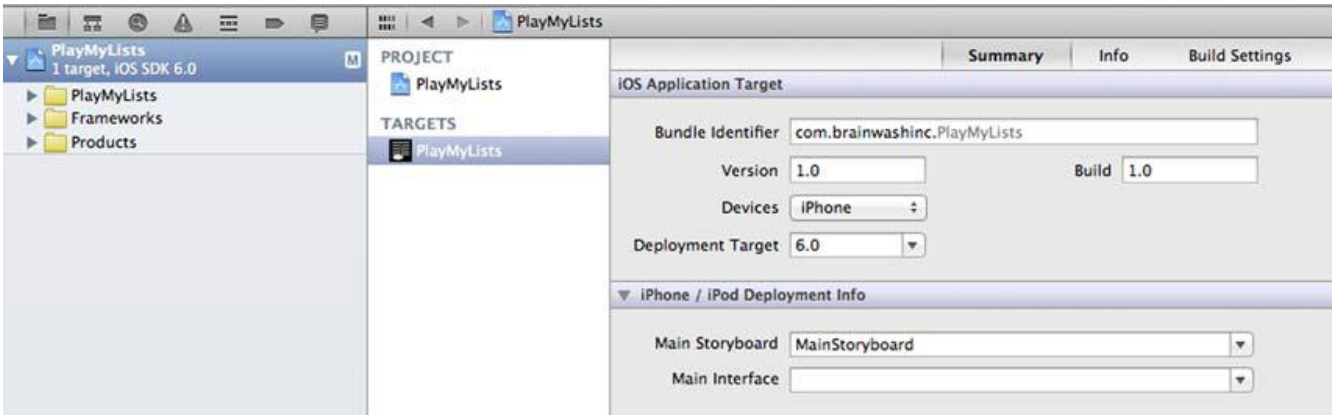


图 7.3 新创建的工程概况展示。

打开使用 navigator 的 Storyboard 文件。正如你所见，base view controller 是一个 navigation controller ,这是一个以 tableviewcontroller 为 root 的 navigationtroller ,当点击 table 的其中一个 cell 会在当前 navigationcontroller 转向一个 detailview controller（见图 7.4 ）。

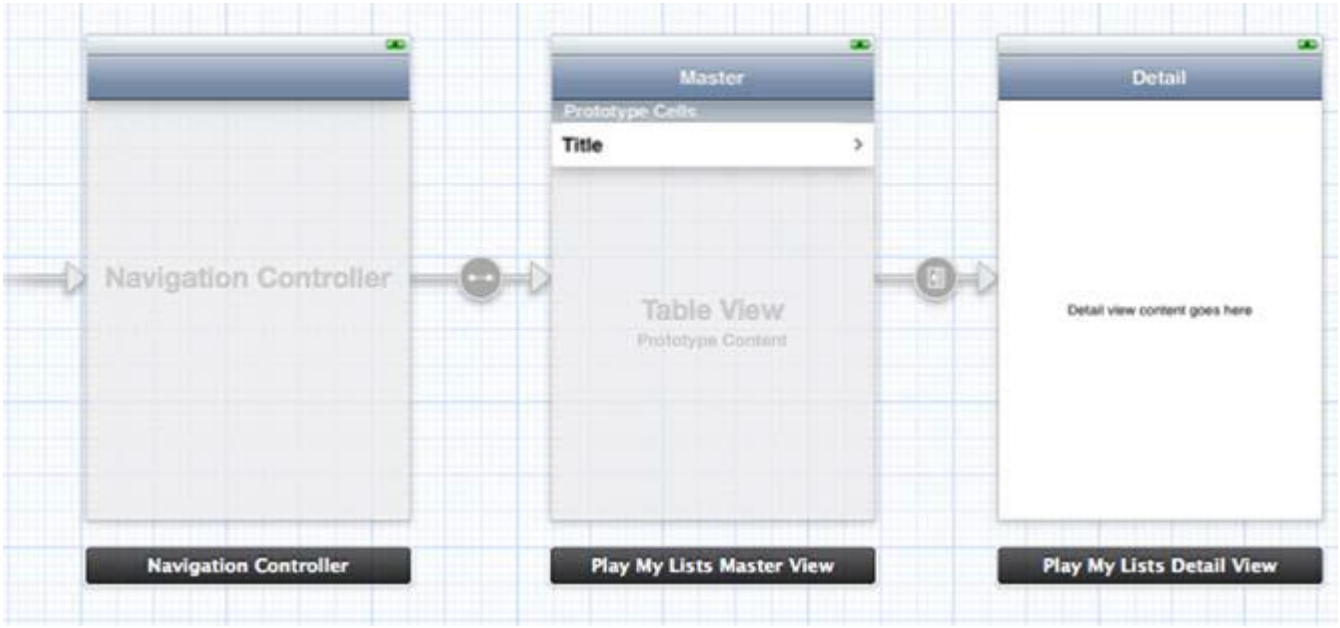


图 7.4 Master-Detail 应用的 Storyboard

可以确定的是这种分层数据展示和跳转到详细界面的设计是最常见的用户体验设计。

就此，Xcode已经为你处理了绝大多数设计，其中包括CoreData框架。它还创建了你应用的delegate，master view controller ,detail view controller classes。在Storyboard的顶部 ,你的master view controller是一个table view controller。最后，应用的delegate和master view controller有许多由CoreData生成的和一个普通tableview controller相关的代码。

如果现在用模拟器运行应用，就可以使用“+”这个按钮创建新的数据库数据，使用“Edit”按钮删除数据库数据，还可以通过单击任何一行浏览每行的详情（见图 7.5）。

图 7.5 在模拟器上执行模板代码

现在，你已经创建了 UI 代码，但是数据方面的代码呢？你需要配置项目工程，使其可以通过 CoreData 存储和访问数据。

技巧 45 在 CoreData 中定义实体

实体需要简单，但是更为复杂的实体是有可能被创建的。要为播放列表创建两个实体：播放列表和它的成员音轨。

播放列表将会包含音轨，所以必须要设置播放列表和音轨关系。在这个技巧中，你将会关注实体的创建和添加属性。

嵌入 Xcode 是开发数据库实体界面。一个列表和属性的整体详细规划是列表地域。你将会使用这种可以访问所有数据的界面。这些数据是你数据库中需要的数据。

问题

需要为你的应用创建两个CoreData实体，AppPlayList（应用播放列表）和PlayList-

Track(播放列表音轨)，其目的是为了规划你的数据库列表。播放列表需要有名字。音轨需要有继续的永久的 ID。通过这个 ID 可以关联到 iPod。

解决方案

使用 data model editor(数据模型编辑)时候，你将会定义你的实体和它们的属性。

讨论

在 Xcode 的左边，Groups & Files 区域选择 PlayMyLists.xcdatamodel。需要注意的是 Event 的模板实例是在 Entities 区域的。选择并且删除这个 item，然后我们就可以开始刷新。

现在我们的列表空了，点击 Add Entity 按钮。现在一个新的实体创建好了，需要命名（让我们把它命名为 AppPlayList。

Entity Naming Warning (实体命名警告)命名实体的时候需要注意。如果和 iOS 框架中已有的类名称重复的话，就有可能在编译的时候出现错误。

下一步，给实体添加一个属性。点击 Attributes 区域左下角的（+）按钮，输入 name（名称），然后设置 Type（类型）为 String（见图 7.6）

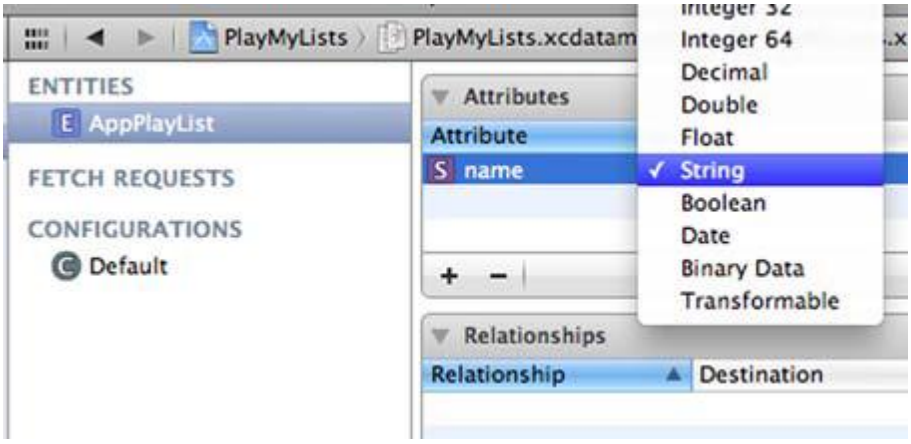


图 7.6 AppPlayList 实体，name 的属性的 type 为 String

现在点击框架底部的按钮创建一个新的实体，命名为 PlayListTrack。添加一个新的属性，命名为 persistentID，设置它的 type 为 String（见图 7.7）

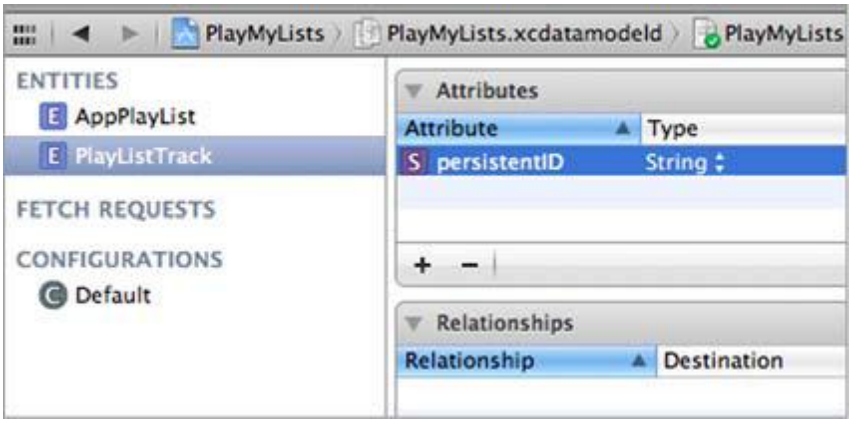


图 7.7 PlayListTrack 实体，persistentID 的 type 设为 String

现在有了两个实体。但是他们不相关联。我们的应用如何知道哪个 playlists 的路径。我们需要定义两个实体之间的关系，让我们现在开始。

技巧 45 在 Coredata 中创建关系

通常来说，数据库需要关系。播放表需要访问多重的路径，而音轨需要知道他们属于那个播放表。播放表和音轨的关系是一对多(to-many)的关系，一个播放列表可以指向多个音轨。

我们可以使用 model editor(模型编辑器)来创建关系，包括一对多，多对多的关系。一个关系可以用多种方式来处理对象删除，例如是否只删除这个对象，还是把在数据库中和这个对象相关的数据全部删除。

问题

需要定义播放列表和音轨的一对多关系。

解决方案

使用 CoreModel 编辑器，我们可以定义两个实体的关系。同时，也可是指定 AppPlayList 到 Play-ListTrack 的关系是一对多的关系。我们还可以定义 PlayListTrack 到 AppPlayList 的关系是可逆的。

讨论

从左边的面板选择 xcdatamodel。在 Definition 中选择 AppPlayList 实体。点击 然后点住 Add Attribute 按钮到下拉式目录中，选择 Add Relationship。

我们给 AppPlayList 实体定义了一个新的关系。给这个新的关系命名为 tracks。在它的右侧，设置 Destination 为 PlayListTrack。不要指定逆向。

给这个新的关系调出 Utilities 视图 ,在 Utilitie 视图的顶部 ,确保选择了 Data Model Inspector。在 Plural 区域勾选 To-Many Relationship（见图 7.8）。设置 Delete Rule 为 Cascade。这就意味着如果播放列表被删除了，路径的数据也会被删除。你不想要在数据库中单独保存路径的数据。

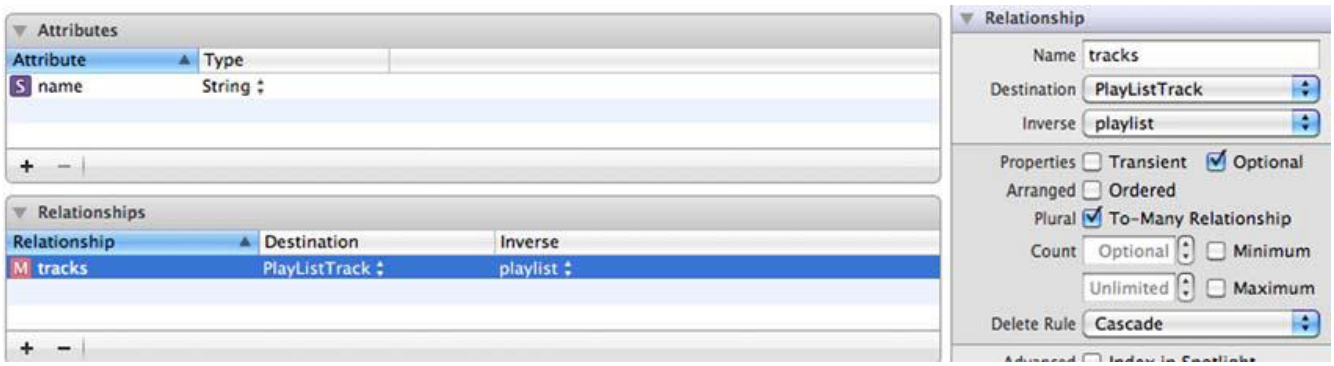


图 7.8 给 AppPlayList 和 PlayListTrack 创建一个一对多的关系

现在选择 PlayListTrack 实体然后从下拉目录中创建一个新的关系，给它命名为 playlist 这是因为它是指向给定路径的播放列表的。设置它的 destination 为 AppPlaylist。设置它的逆向到音轨，音轨是我们之前给 AppPlaylist 定义的关系（见图 7 .9）。这是同一关系的另一端。

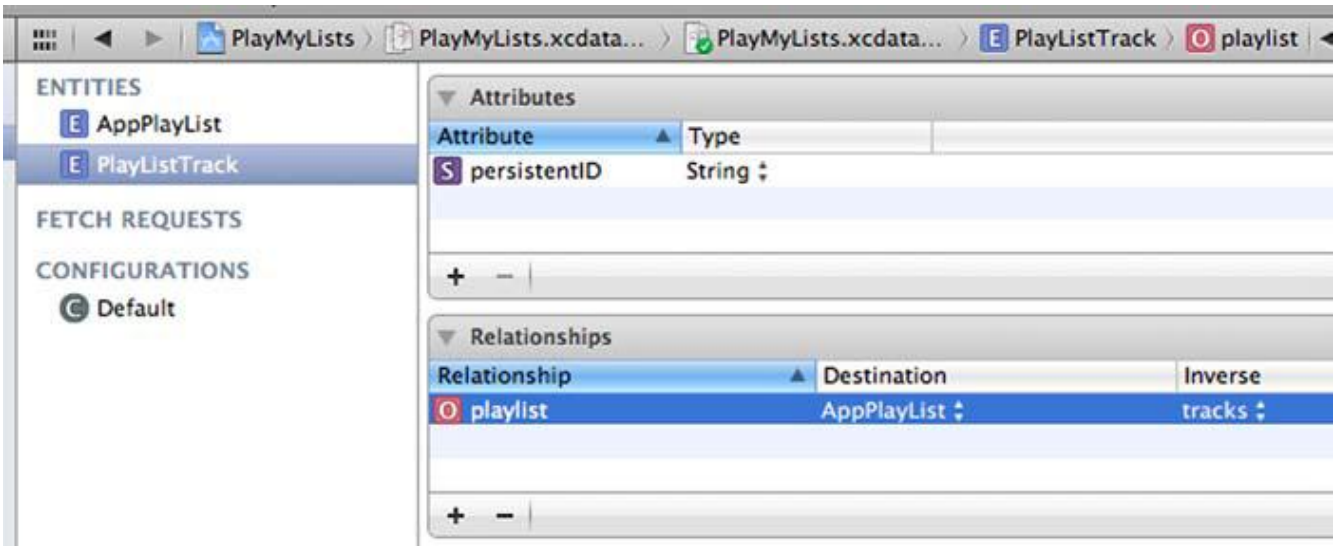


图 7 .9 创建一个从 PlayListTrack 到 AppPlaylist 的逆向关系

如果你不把它设置为逆向（或是完全不创建这个关系），那么播放列表就会和音轨有一个关系。但是音轨不会知道它是属于哪个播放列表的。根据应用的不同，这方面也许会有用。

现在我们已经创建了两个实体，以及两个实体之间的关系。现在需要了解更高层次的关于这两个实体的信息，在右侧，选择这两个实体（见图 7 .10）

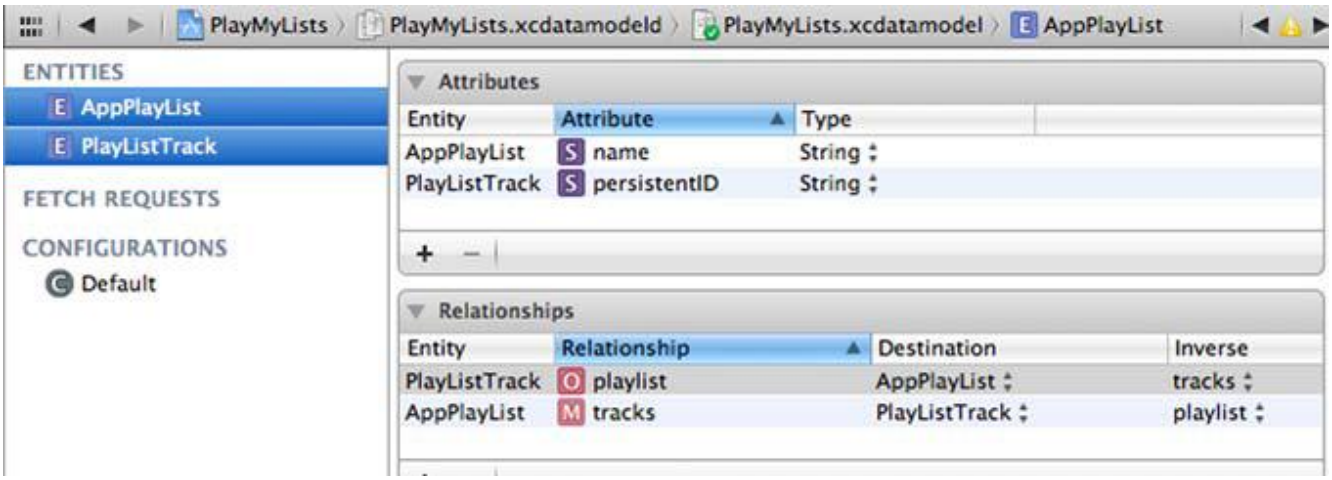


图 7 .10　产看所有定义的实体以及它们的属性和关系

点击右下角的 Table/Graph，可以看到由我们的实体定义的数据库的一个展示图。（见图 7.11）

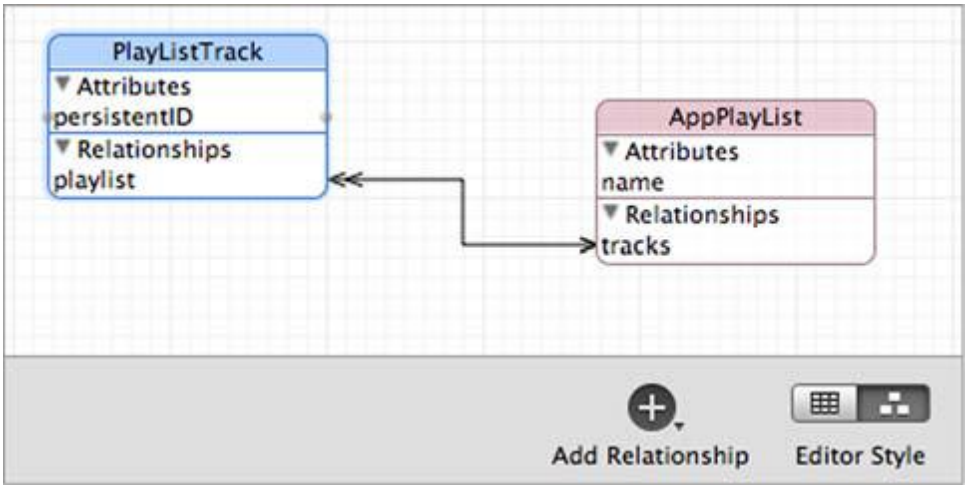


图 7.11 数据库的实体以及关系的展示图

技巧 45　插入以及删除 CoreData 对象

我们现在已经改变了数据库的定义，应用不再有效，它还在使用之前定义的实体和属性。但是只要做一

些小改变，CoreData 就可以让我们访问数据。

由于默认工程已经处理了一些 CoreData 的功能 ,包括创建新的对象 ,把他们在 table view 中排列出来，然后删除 ,我们可以重新使用之前的一些代码。但是我们需要更新代码来对应我们对 Event 实体所做的变化。

问题

需要从代码访问数据库 AppPlayList 实体，展示它的属性。

解决方案

可以在代码中改变应用，使用改变过后的定义。

讨论

由于默认的 CoreData 代码是使用 key-value 编码的，我们不需要去概念类的名称。同时，由于默认的实体和我们的 AppPlayList 实体只有一个属性，所以我们只需要改变这唯一一个属性的指引。

应用需要知道如何处理实体（比如说创建，还是访问，或是删除）。对创建或者返回对象来抓取数据的方法，我们需要告诉这个方法要使用新的实体名称。

找到 fetchedResultsController 方法。这个方法有一行代码是指定它是处理哪些实体的。找到这行代码，让后把 Event 变为 AppPlayList。

```
NSEntityDescription *entity = [NSEntityDescription
        entityForName:@"AppPlayList"
        inManagedObjectContext:self.managedObjectContext];
```

这是唯一一个使用实体名称的地方。现在 NSFetchedResultsController 实例可以访问实体了。但是你也需要告诉它属性的名称。

fetchedResultsController 实例需要知道如何整理从数据库获取的对象。整理是通过为 fetchedResultsController 创建的 NSSortDescriptor 实例。在同样的方法 fetchedResultsController 中，把 timeStamp 换为 name，创建整理标示符。

```
NSSortDescriptor *sortDescriptor = [[NSSortDescriptoralloc]
```

```
initWithKey:@"name" ascending:NO];
```

之前的实体有一个叫做 timeStamp 的属性。如果把代码中的所有指引都指向这个新的属性，那就能正确的获取到数据。

在配置表单元的方法中，置顶在表单中显示的属性。把 timeStamp 变换为 name，如以下代码所示：

在表单单元格中显示 name 属性

```
- (void)configureCell:(UITableViewCell *)cell
atIndexPath:(NSIndexPath *)indexPath {
NSManagedObject *managedObject =
[self.fetchedResultsController
objectAtIndexPath:indexPath];
cell.textLabel.text = [[managedObject
valueForKey:@"name"] description];
}
```

现在我们已经获取到了正确的属性，但是应用仍需要设置正确的属性。在 insertNewObject 方法中仍把 timeStamp 变换为 name。把值的默认名称设为 New List，这么做对应用来说更有意义。

```
[newManagedObjectsetValue:@"New List" forKey:@"name"];
```

现在可以运行应用，然后创建数据库的 item——AppPlayList 了。当然，我们也可以删除它。但是现在这些播放列表是数据库中唯一的数据，名字都相同，而且没有音轨（见图 7.12）。确保要删除应用的老版本，避免出现数据库内容混淆的情况。

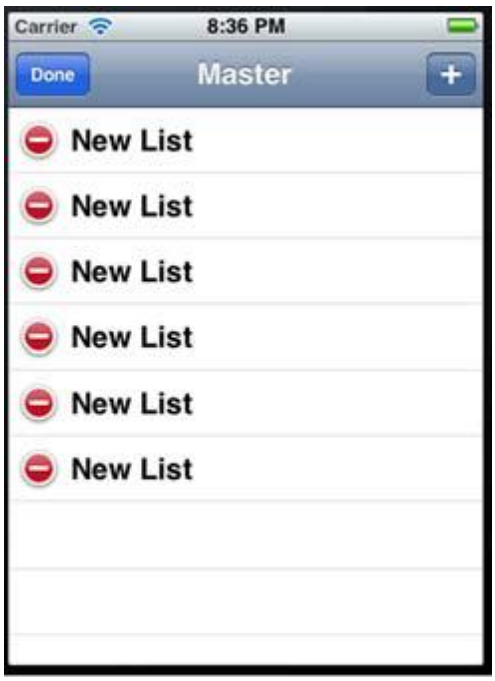


图 7.12 PlayMyLists 列出了新创建的 AppPlayList 实体

要把音轨添加到播放列表，我们可以使用相同的 key-value 编码方式，或者我们可以给实体生成代码，然后通过 member items 来访问。现在让我们看看如何从 CoreData 实体生成 Objective-C 代码。

技巧 45 给 CoreData 实体创建类

基于 CoreData 实体来创建类对访问数据来说很有用。但是你可能在创建之后不想再编辑这段代码。如果之后你需要改变实体定义，重新生成代码的话，那这些改变就没有用了。为了避免这样，我们可以取而代之的创建辅助类或者子类。

辅助类可以是继承了我们生成的 CoreData 类的类，也可是完全单独的类。辅助类可能会过滤你的对象，整理对象，然后转变对象，或者可以做到你其他需要的功能。

问题

需要基于 CoreData 实体定义来生成代码。

解决方案

可以使用 Xcode 的新文件机制来给工程生成代码。

讨论

再次打开 data model，选择一个或多个实体。可以右击实体的名称，然后可以得到相关的 CoreData 操作说明。

创建子类，可以选择你想要生成源代码的一个或多个实体。然后从 Editor 目录中，选择 Create NSManagedObject Subclass 选项（见图 7.13）。

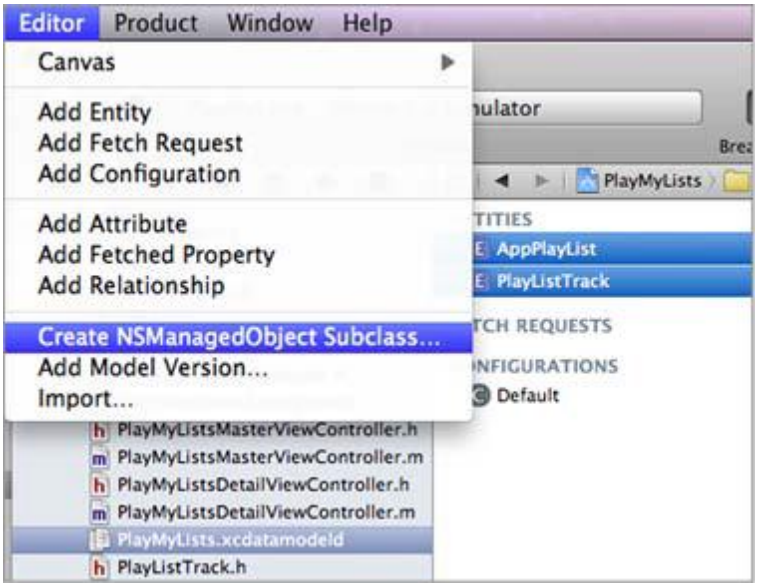


图 7.13 选择需要生成 Objective—C 代码的实体

在下一个表单中，选择 location and target（见图 7.14）

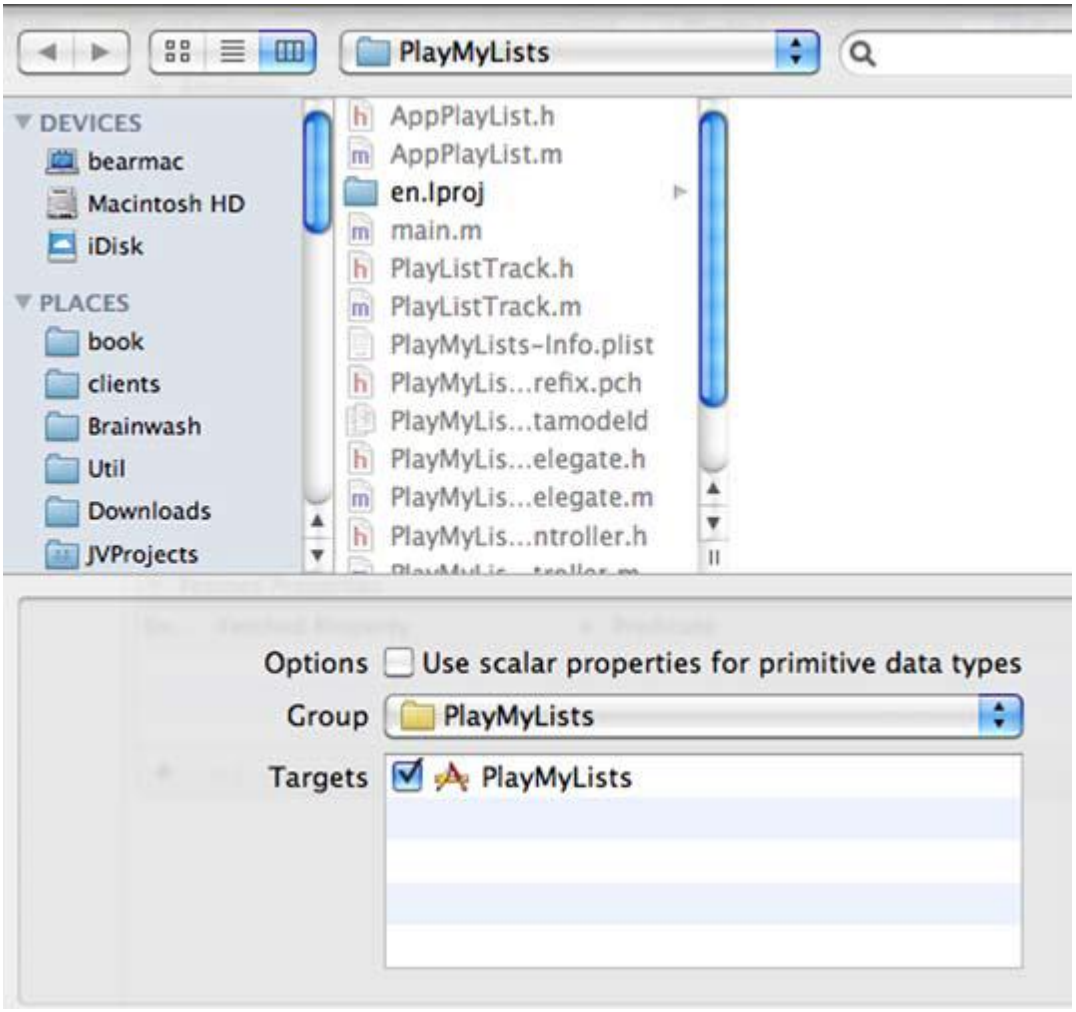


图 7.14 给新的 NSManagedClass 文件制定 directory，group，以及 target

现在基于选择的实体应该生成了 2-4 个新的类——头文件，执行文件各一个。这些文件的名字和创建的实体的名字相同。但是，同样的，如果实体的名字有和其他的文件重复，那就可能出现问题。

通过研究这些执行文件，我们可以发现类不是很多。头文件则更加有趣，因为它们能够告诉你访问了哪些方法（见以下代码）。要注意属性和关系如何 转化为 items 的。

AppPlaylist的头文件

```
#import <CoreData/CoreData.h>
@interface AppPlayList :NSManagedObject
{
}
@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSSet* tracks;
```

```
@end
@interface AppPlayList (CoreDataGeneratedAccessors)
- (void)addTracksObject:(NSManagedObject *)value;
- (void)removeTracksObject:(NSManagedObject *)value;
- (void)addTracks:(NSSet *)value;
- (void)removeTracks:(NSSet *)value;
@end
```

CoreData 对象为一对多关系使用 NSSet 实例。NSSet 和 NSArray 类似，但是并不规则。在我们的例子里，访问路径返回了一批 playlist 的路径数据，这些事一个或者多个 PlayListTrack 实例。也可以生成方法来添加或者移除一个或多个路径数据。

AppPlaylist的头文件

```
#import <CoreData/CoreData.h>
@class AppPlayList;
@interface PlayListTrack :NSManagedObject
{
}
@property (nonatomic, retain) NSString * persistentID;
@property (nonatomic, retain) AppPlayList * playList;
@end
```

由于播放列表和音轨是一对多的关系，存取器(accessor)需要支持它们。它使用 NSSet 来返回路径，也有可以添加附加路径的方法。

这就是给 CoreData 实体生成代码的方法。你可以为数据库的数据来使用这些类，然后获取属性作为数据。让我们看看怎么做。

现在我们知道了如何使用 CoreData 来创建功能。还了解到了如何定义实体，实体的属性以及关系，更进一步的，我们还学了如何用一个可以理解的方式为用户显示这些数据。

7.2 在数据驱动的应用中呈现 CoreData

在上节中，你看到了怎么利用 CoreData 创建一个数据驱动的工程，也学会了在数据库和源代码创建的时

候怎么去处理数据。

我们最想做的是允许用户通过 UI 来创建和操作这些数据。你需要给用户展示数据，允许他们浏览数据，增加或者删除数据库中的条目（UI 就是用来干这个的）。让我们来看看怎么做。

技巧 45 显示选择的 item 的详细信息

默认的工程代码有一个 table view controller 来显示数据库中默认的实体数据。这个默认的 table view 会显示一列播放列表。因为播放列表包含了一个音轨的集合或者列表，所以我们继续进行用另外一个 table view 来显示每一个列表中的内容。

你可以最大限度的利用已有的代码，只是要使用不同的实体名称和属性。这将保证你用同样的方法创建和访问播放列表中的音轨和播放列表。

问题

需要创建一个 table view controller 来显示一个选中的播放列表中的音轨。

解决方案

可以基于工程默认创建的 table view controller 创建一个新的 table view controller 类。这样就可以用新建的 table view 来显示播放列表传入类中的音轨。

讨论

在类列表区域邮件单击选择 New File。选择 NSObject 为基类，但是这没关系，因为你要替换里面的内容。新建的类名名为 VCPlayListItems。

复制 PlayMyListsMasterViewController.h 中的内容到新建类中的头文件。同样复制 PlayMyListsMasterViewController.m 到 VCPlayListItems.m 文件。记住要保留类的名字和重要声明。

跟修改 Event 实体为 AppPlaylist 一样，我们也需要将 AppPlayList 修改成 PlayListTrack。在这个新的类里，需要替换 name 为 persistentID。

在 insertNewObject 方法中，需要把实体的值设置为正确的属性:

```
[newManagedObjectsetValue:@"New Track" forKey:@"persistentID"];
```

在 fetchResultsController 中，修改实体的名称为 PlayListTrack:

```
NSEntityDescription *entity =
[NSEntityDescriptionentityForName:@"PlayListTrack" inManagedObjectContext:self.managedObjectContext];
```

在 configureCell 方法的实现中不需要访问 name 属性，因为在 PlayListTrack 中没这个属性。你可以修改它来使用 persistentID:

```
- (void)configureCell:(UITableViewCell *)cell
atIndexPath:(NSIndexPath *)indexPath {

NSManagedObject *managedObject = [[[playList tracks] allObjects]
objectAtIndex:indexPath.row];
// [self.fetchedResultsController
objectAtIndexPath:indexPath];

cell.textLabel.text=
[[managedObjectvalueForKey:@"persistentID"] description];

}
```

在 tableView:cellForRowAtIndexPath:方法中调用了 configureCell 方法。有几种不同办法来显示一个播放列表的音轨。最开始，你需要一个选中的播放列表。

Master-Detail 应用程序模版创建了一个 master view controller ,它传入 selectedObject 到 prepare-ForSegue 方法中的 detail view controller。可以用这个方法来获取选中的播放列表，传给新建的 table view controller。

技巧 45 给 Detail View controller 传递一个播放列表

Master view controller 已经通过合成的 detailItem 成员传递了 selectedObject 到 detail view controller。可以增加同样的代码到新建的 VCPlayerListItems table view controller。然后，就可以访问播放列表的数据然后展示给用户。

问题

你需要 VCPlayListItemstable view controller 来访问在 master view controller 选中的播放列表 。如果有了这个引用，就可以在 table view 中列出音轨。

解决方案

可以从 detail view controller 拷贝与 detailItem 相关的代码到 VCPlayListItems 类，并且通过引用使用。

讨论

你需要增加两行代码到你的 VCPlayListItems 类：一行到头文件，一行到类的实现。头文件需要声明成员变量：

```
@property (strong, nonatomic) id detailItem
```

实现中需要合成成员变量：

```
@synthesize detailItem = _detailItem
```

注意 或者，如果想要修改成员变量类型、引用以及 master view controller 对他的调用方法 setDetailItem 名称，可以修改 detailItem 成员变量为 playList。

现在，当播放列表的数据传过来以后，你可以访问访问音轨并且在 table view 中列出来。播放列表的音轨列表将只有一节（section），所以可以在 numberOfRowsInSection 方法中设置。在这里进行一些初始化也不错，所以我们调用 fetchedResultsController 存取器（accessor）。因为你忽略了返回值，所以将在以后使用的时候设置这个实例：

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView { [self fetchedResultsController];
return 1;
}
```

对于一节中音轨的数量，可以通过返回播放列表中的音轨数量来得到。注意下面的代码会产生一个错误，因为 _detailItem 声明为 ID 类型而不是一个 AppPlayList 实例。可以通过引入 AppPlayList.h 头文件和强制类型转换 _detailItem 为 AppPlayList 类别修复这个错误：

```
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section {
return [[_detailItem tracks] count];

}
```

现在就可以用新建的 VCPlayListItemstable view 来显示播放列表的音轨，可以用新的 detail view controller 来替换以前的 detail view controller 。

技巧 45 替代 detail view controller

默认的 detail view controller 是一个简单的 view controller，当任何一个选中的 item 传入，只使用一个 label 来显示其基本的数据。在我们的应用中 ,我们更了解这个对象 ,并且想显示与之相关的更合适的信息。

除此之外，也想让用户在选中的播放列表中增加和删除音轨。

问题

我们想用刚才新建的 VCPlayListItemstable view controller 替换项目中 Storyboard 文件中默认的 detail view controller 。

解决方案

在 Storyboard 文件中，先删除 detail view controller，拖入一个新的 table view controller ，指定它的类为 VCPlayListItems。

讨论

打开 Storyboard 文件，选中 Detail View Controller，删除它。拖入一个新的 table view controller 到以前旧的所在的地方，指定它的类为 VCPlayListItems。

点击新建的 tableview 的单元格 ,在 Attributes Inspector 中设置它的标识符(identifier) 为 TrackCell。这样以后作为重复用的标识符，CellIdentifier 在代码中使用。可以现在就在在 cellForRowAtIndexPath:方法中设置它。

按住 Control 键 ,然后从 master view controller 中 table view 的一个 cell 拖向新建的 VCPlayListItems，

设定 segue 值为 Push (见图 7.15)。选择新建的连线（segue），在 Utilities 的 Attributes Inspector 中设定它的标示符为 showDetail 见图 7.16)。

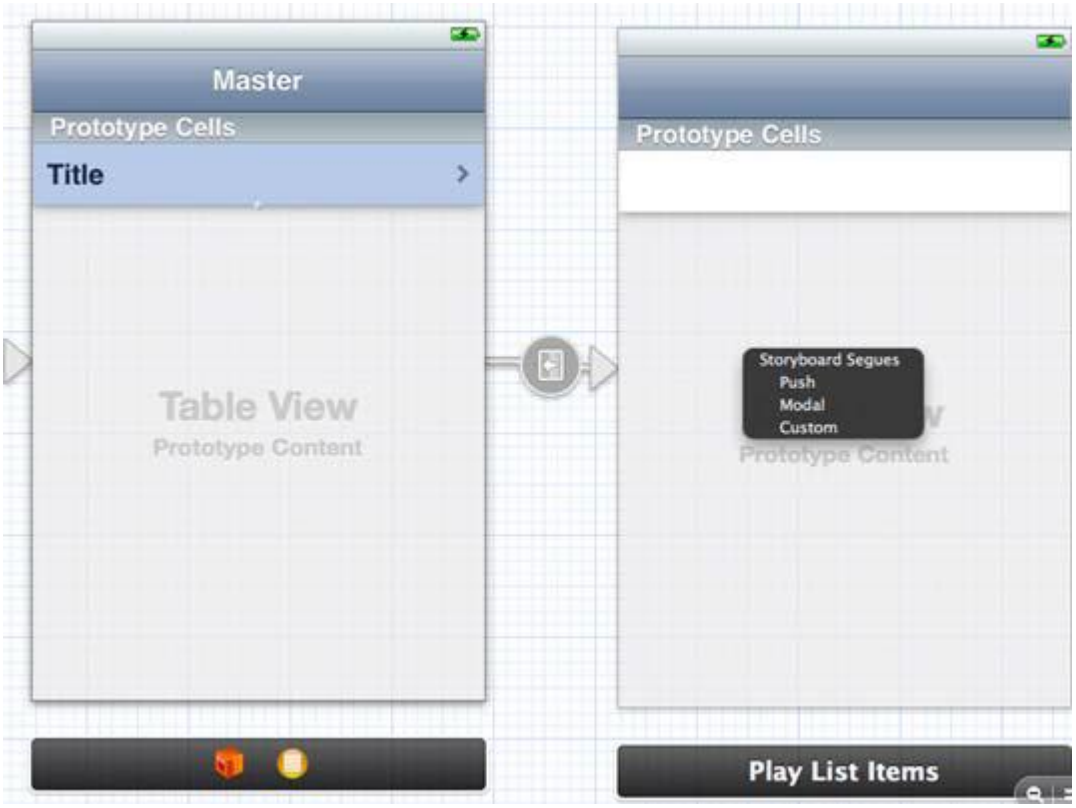


图 7.15 通过 master view controller 连线（segue）链接新的 detail view

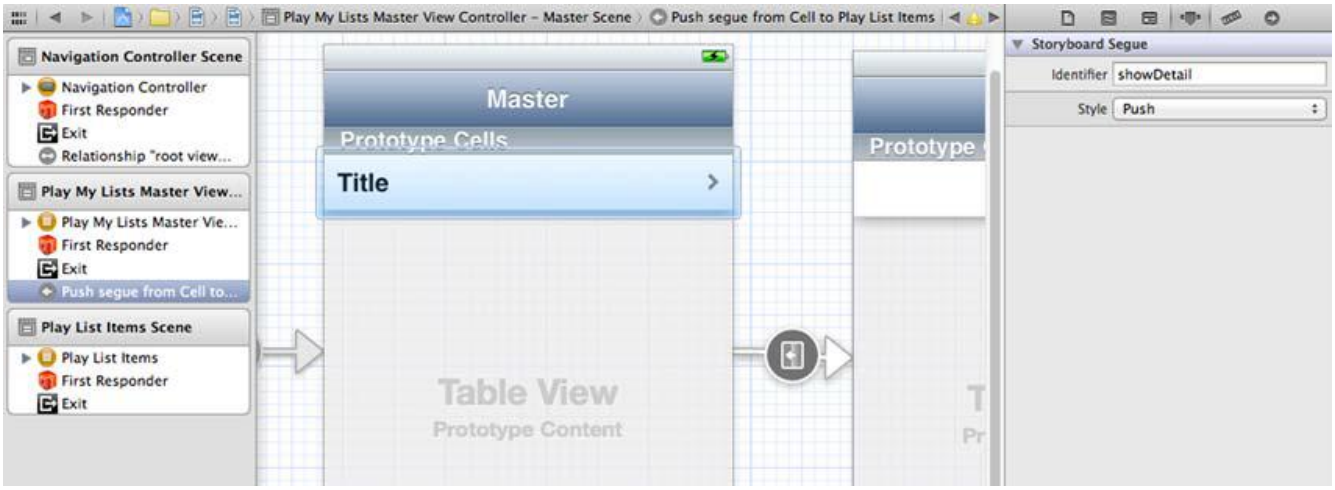


图 7.16 从 master view controller 新建的 detail view controller 和连线（segue）

现在当用户点击 table view 中的一个播放列表 ,连线(segue)的方法就被会调用。master view controller 会传入选中的对象，同时 VCPlayListItems 实例会从右边滑入。

你已经设置好了 VCPlayListItems 类来列出播放列表的音轨 ,但是没有提供用户往播放列表中增加音轨。下面我们就来做这个。

技巧 45 在选中的播放列表中管理音轨

你基本上完成好了新建的 detail view controller ，VCPlayListItems，但是现在你需要实现增加音轨到播放列表的功能。

问题

当用户增加一个音轨到播放列表的时候这个 table view 没有被更新 ,用户也就不知道音轨是否已经添加。同样，删除的时候也是。

解决方案

当一个音轨增加到数据库的时候，应用也同时需要增加音轨到选中的播放列表。应用的 delegate 回调函数会让 table view 更新新的数据。同样的，table view 的回调函数会删除一个音轨后更新数据库，也更新这个 table view。

讨论

你想通过(+)按钮添加一个新的音轨后更新你的 table view。你知道增加一个后（在 insertNewObject 方法中增加)，需要更新这个 table view。因为你的 table 是基于播放列表中的音轨而不是数据库中的音轨，所以需要增加新的音轨到播放列表 ,对应的 NSFetched- ResultsControllerDelegate 回调函数会更新 table：

```
[_detailItemaddTracksObject:newManagedObject];
```

同样 ,删除的方法是 tableView:commitEditingStyle:forRowAtIndexPath: 。确保删除音轨的时候已经先从播放列表中删除了，如下所示，删除一个已经删除的音轨会导致错误。

把音轨从 playlist 移除，删除然后重新加载

```
NSManagedObjectContext *context =
[self.fetchedResultsControllermanagedObjectContext];

[playListremoveTracksObject:
[self.fetchedResultsController

objectAtIndexPath:indexPath]];
[contextdeleteObject:

[self.fetchedResultsController
objectAtIndexPath:indexPath]];
```

一些不错的点击设计是 Edit 按钮不会遮住 Back 浏览按钮，并且设置标题为 Tracks。这两件事情可以通过 viewDidLoad 方法来实现：

```
self.navigationItem.leftItemsSupplementBackButton = YES; self.title = @"Tracks";
```

另外一个办法是把传入的播放列表的名字作为播放列表的抬头，是否采用这个方法取决于你，实现也自己试试吧。

现在你知道了播放列表和音轨的整体结构，我们来看看访问用户的音乐来创建一个实际的播放列表。

7.3 访问 iPod 上的 音乐

每一个 iOS 设备都有一个音乐应用来播放音乐。对于日常使用这个播放器很不错了，但是如果你需要一些特殊功能，比如允许用户玩游戏的过程中播放一首歌，这个播放器是不提供这些功能的，需要你自己的游戏来自己实现这个功能。

iOS SDK 提供了直接访问音乐的方法，并且提供了给用户选择音频使用的 UI。你的应用需要允许用户选择添加到播放列表的音轨，然后你可以使用 API 直接获取选择项目的标题。

技巧 45 使用 media picker 访问音乐

跟地址簿和 image picker 一样，iOS 框架也提供了访问设备上音乐的接口。你不但可以获取音频数据，

也提供给你一个漂亮的、统一的用户界面来选择音乐。

同样，跟访问地址簿和图片选择器的场景一样，当用户从音乐库选择了音乐后，你通过回调 media picker 类的方法来提示你的应用。

如果你要创建一个播放列表，你可以使用这个用户界面来让用户选择音轨，同时提示应用程序用户做的选择。

问题

你需要允许用户从他们的设备选择音轨添加到播放列表。

解决方案

可以使用 MediaPlayer 框架的 MPMediaPlayerController 来显示一个选择音轨的 UI 界面。然后应用从音轨数据获取 persistent ID，在数据库创建一条新条目关联到正在编辑的特定播放列表。

讨论

处理类似使用 media picker controller 来选择 items 需要四步—你需要先做一些基础工作，创建和显示 controller，用户选择后获取数据时处理回调，同时需要处理获取的数据。在这里，处理获取的数据意思是把播放列表的音轨数据存储到数据库。

要想使用 media picker，需要添加 MediaPlayer.framework 到工程。同时，需要导入 MediaPlayer/MediaPlayer.h 文件到 VCPlayListItems 头文件。最后一项基础工作是把 addButton 按钮的目标动作改为一个新方法 showMediaPicker，这个方法你马上就会实现，见以下代码。

让 addButton 调用 showMediaPicker

```
UIBarButtonItem *addButton = [[UIBarButtonItemalloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemAdd
target:self
action:@selector(showMediaPicker)];
```

第二步是创建并显示一个 controller，在 showMediaPicker 中来处理。在最后一步你会实例化 media

picker，设置一些值并且来展示这些值，详见下面的代码。

创建并显示 media picker

```
-(void)showMediaPicker
{

MPMediaPickerController *picker =
[[MPMediaPickerControlleralloc] init];

[pickersetDelegate: self];

[pickersetAllowsPickingMultipleItems: YES];

picker.prompt = @"Pick Tracks";

[selfpresentModalViewController: picker animated: YES];
}
```

对于 iPhone 和 iPod，这没问题。但是对于模拟器，没有 iPod，将会使应用奔溃，看起来好像出现个大问题。为了防止这个问题，需要在应用最开始的时候做一个检查:

```
if (NSNotFound != [[[UIDevicecurrentDevice] model]
rangeOfString:@"Simulator"].location)

return;
```

现在，你可以运行这个应用了，选择一个播放列表，点击 Add 按钮就能够看到 media picker (见图 7.17).

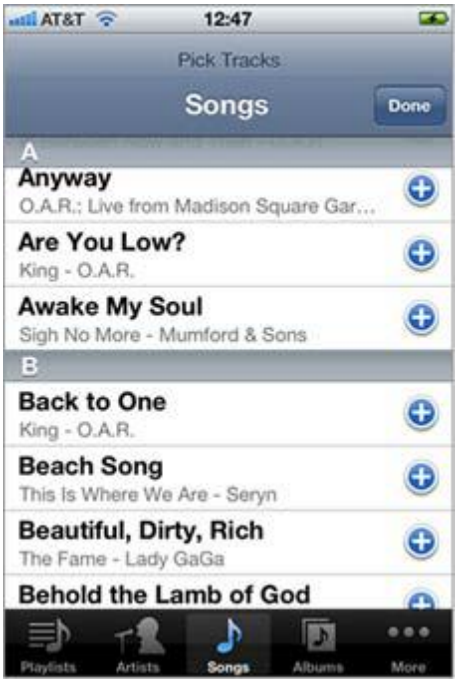


图 7.17 显示 media picker

但是现在还没有处理回调，所以不但不能选择音轨，从屏幕上也不能移除 media picker。下面我们来完善一下。

第一步，也是最容易的，是处理 Cancel 回调。当用户点击 Cancel，会简单的收起 media picker:

```
- (void)mediaPickerDidCancel:
(MPMediaPickerController *)mediaPicker

{
[selfdismissModalViewControllerAnimated:YES];

}
```

另外一个回调是当用户点击 Done 按钮，你会检查一下他们是否选择了音乐，创建新的数据库条目，收起选择器，详见下面的代码。

传递 media item 到 insetNewObject，重新加载表单，让 media picker 消失

```
- (void)mediaPicker: (MPMediaPickerController *)mediaPicker
didPickMediaItems:(MPMediaItemCollection *)mediaItemCollection

{
```

```
for (MPMediaItem *mi in [mediaItemCollection items])

[selfinsertNewObject:mi];

[self.tableViewreloadData];

[selfdismissModalViewControllerAnimated:YES];
}
```

注意，添加好所有新的 item 后会刷新 table。可以移除同一个对 insertNewObject 方法的调用来确保它只被调用一次。你会看见 insertNewObject 方法被传递给了 MPMediaItem。我们来更新一下这个方法。

之前我们输入了 New Track 来作为 persistent ID。这样不是很好。需要获取 media item 的真实 ID 存储在数据库。Media item 的属性可以通过 valueForProperty: 方法来访问，这有点类似于处理 NSManagedObject 实例，详见下面的代码。

抓取 persistent ID，把它存储到实体对象中

```
- (void)insertNewObject:(MPMediaItem*)mediaItem {

NSString *pID = [NSStringstringWithFormat:@"%@",
[mediaItemvalueForProperty:MPMediaItemPropertyPersistentID]];
if ([[[detailItem tracks] allObjects] containsObject:pID])
return;

...
[newManagedObjectsetValue:pIDforKey:@"persistentID"];
```

来检查一下音轨是否已经在列表里面，如果是返回真。同时，可以移除 reloadData 的调用，因为在 media picker 中调用了 reloadData。

现在当你在设备上运行应用并选择音乐，不但可以取消，还可以添加选择的音乐到播放列表（见图 7.18)。

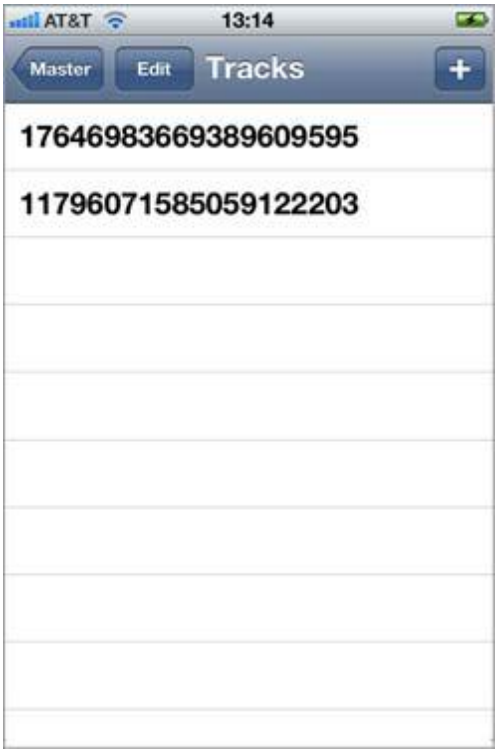


图 7.18 选择播放列表上列出的 persistent ID

它使用 persistent ID 来显示，作为开发者你可以很兴奋了！接下来我们来看一下怎么用音轨的标题来替代显示的 persistent ID。

技巧 45 从 iPod 中查找媒体信息

iPod 给这个设备上的每一个音轨指定了在这个设备上唯一的 persistent ID。在这个设备上它不会改变，但是在其他人的 iPod 数据库里面 ID 不会是一样的。

因为存储了 persistent ID ,所以可以通过 MPMediaQuery 类查找选择的音乐。当你知道媒体条目的 ID，就可以查询其他的信息，包括标题。从某种程度来说，在应用中显示标题对用户来说更加有用。

问题

需要通过 persistent ID 查找选定的音乐，获取标题，然后显示在 table 中。

解决方案

通过 MPMediaQuery 类，可以获取到符合条件的音乐（包括 Persistent ID）。 媒体查询类使用

MPMediaPropertyPredicate 类来指定查询条件。你可以指定 persistentID 来查询条目。

讨论

从 configureCell 方法，已经知道从 indexPath.row 选择的索引值。有这些信息，你可以获得, managedObject 和 persistentID。通过创建一个 MPMediaPropertyPredicate，你可以过滤 MPMediaQuery 中的条目来找到你要的，可以参考下面的代码。

通过persistent ID检索media item，把单元格中的文本为设置为它的标题

```
- (void)configureCell:(UITableViewCell *)cell
atIndexPath:(NSIndexPath *)indexPath {
NSManagedObject *managedObject = [[[detailItem tracks] allObjects]
objectAtIndex:indexPath.row];
NSString *pID = [managedObjectvalueForKey:@"persistentID"];
NSNumber *longID =
[NSNumbernumberWithLongLong:[pIDlongLongValue]];
MPMediaPropertyPredicate *pred = [MPMediaPropertyPredicate
predicateWithValue:longID
forProperty:MPMediaItemPropertyPersistentID];
MPMediaQuery *mpQ = [MPMediaQuerysongsQuery];
[mpQaddFilterPredicate:pred];
NSArray *items = [mpQ items];
if ([items count] > 0)
cell.textLabel.text = [[items objectAtIndex:0]
valueForProperty:MPMediaItemPropertyTitle];
else
cell.textLabel.text = @"Unknown Title";
}
```

可以通过 property predicate 过滤音乐，推荐去阅读 property predicate 的更多功能。但是你有了基本的概念：基于指定条件来过滤音乐。

现在如果在设备上运行应用，可以看到音乐都有标题了 (见图 7.19)。

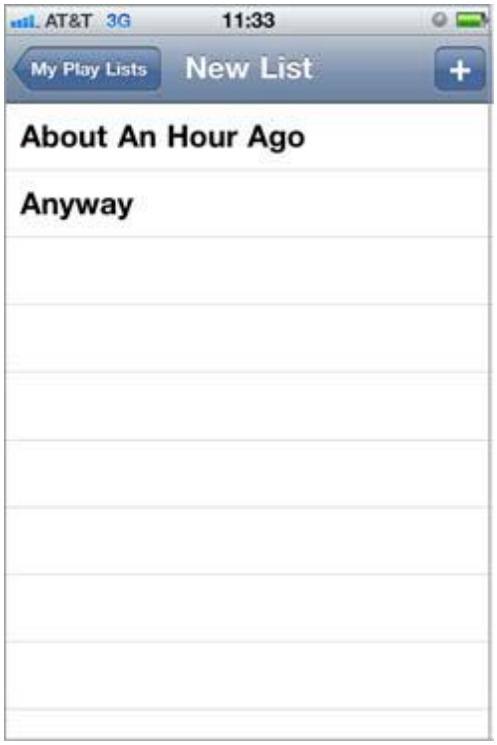


图 7.19 带有标题的播放列表

在很多情况下，最好的做法是把音轨标题和 persistentID 同时存储到数据库，还有其他一些频繁访问的数据。但是需要看实际情况，在有些情况下使用最新的信息更好（比如，方案中的数据需要经常改变，但是音轨的标题不是这种情况）。

不管你存储了多少数据，一定依靠 persistentID 来播放音乐。下面我们来看看怎么播放音轨。

7.4 用 iOS 播放音乐

苹果发布的移动设备系列中数量最多的是 iPod，现在仍然是很棒的设备，所有的 iOS 设备都从中收益很多。它的功能仍旧是 iPhone，iPod，iPad 的核心功能，用户也使用最多，在应用中播放音乐并不复杂。

现在已经存储了所有选择音轨的 persistent ID，所以可以访问设备上的音乐。这样，就可以播放这些音轨了。因为我们要处理播放列表。所以需要处理什么时候结束一个音轨以及什么时候播放下一个音轨。我们知道播放他们的顺序，所以当音轨播放结束了就知道可以换一首了。

技巧 45 播放播放列表

你已经知道了播放列表，也知道怎么去访问所有音轨，用 MPMediaQuery 去查找他们。现在你需要了解怎么去播放他们。

SDK 提供了一个播放音乐的 controller。跟其他框架的 controller 不一样，这个 controller 没有用户界面。但是需要实现一个 delegate 来接受更新的信息。

问题

需要来访问播放列表并且来播放音乐。

解决方案

创建一个 MPMusicPlayerController，获取所有的 MPMediaItems，并且在音乐播放器里面播放。

讨论

为了使得 UI 交互简单，当用户手指点击一首歌曲的时候我们让播放器开始播放。也就是说我们要在 didSelectRowAtIndexPath: 方法里面启动播放。第一件事是检查它是否在模拟器里运行程序。

因为已经在工程中添加了 MediaPlayer 框架，所以可以在头文件中声明一个 MPMusicPlayerController 成员 musicPlayer。 在 didSelectRowAtIndexPath: 方法中，如果它是 nil，你会创建一个（比如，连续在一行点击不会创建新的实例），详见下面代码。

如有需要的话创建一个音乐播放器，或者已经如果有一个播放器的话，则让它停止播放

```
if (nil == musicPlayer)
    {

musicPlayer = [MPMusicPlayerControllerapplicationMusicPlayer];

[musicPlayersetRepeatMode:MPMusicRepeatModeAll]; }

else
        [musicPlayer stop];
```

下一步，你可以创建一个在播放器播放的列表。跟前面做的一样。可以用 MPMediaQuery 类读取播放列表中的每一个音轨。如果把他们添加到一个可变数组，则可以创建一个 MPMediaItemCollection，然后播

放他们，详见下面的代码：

创建一个 MPMediaItemCollection 播放 playlist 实例上的音轨

```
NSMutableArray *tracksToPlay = [NSMutableArrayarrayWithCapacity:
        [[[playList tracks] allObjects] count]];

for (NSManagedObject *managedObject in [detailItem tracks])
{

NSString *pID = [managedObjectvalueForKey:@"persistentID"];
NSNumber *longID =

            [NSNumbernumberWithLongLong:[pIDlongLongValue]];
MPMediaPropertyPredicate *pred = [MPMediaPropertyPredicate

predicateWithValue:longID

forProperty:MPMediaItemPropertyPersistentID]; MPMediaQuery *mpQ = [MPMediaQuerysongsQuery];

[mpQaddFilterPredicate:pred];

NSArray *items = [mpQ items];

            //查找 media

if ([items count] > 0)
        [tracksToPlayaddObjectsFromArray:items];

}

MPMediaItemCollection *collection = [MPMediaItemCollection
collectionWithItems:tracksToPlay];

            //创建 collection

    [musicPlayersetQueueWithItemCollection:collection];
    [musicPlayer play];

            //添加并播放
```

}

如果现在运行程序，可以发现所有的音轨（或者歌曲）都是乱序的。如果尝试去播放音轨，他们不会按照逻辑顺序播放。当播放完一首，将会选择播放列表任意一行来播放——而不是按照用户期望的顺序！

为了确定顺序，可以添加一个 Order 属性到数据模型的 Track 实体，重新创建一个代码。当每一个音轨添加到播放列表，你可以设定它当时在播放列表的序号为顺序。现在当播放列表的音轨迭代添加到 tracksToPlay 数组，它们会首先通过调用

[[detailItem tracks] sortedArrayUsingDescriptors:[NSArrayarrayWithObject:[NSSortDescriptorsortDescriptorWithKey:@"order" ascending:YES]]]进行排序。

技巧 45 处理音乐播放器的更新

音乐播放器会连续播放，因为你设定了循环模式为重复所有（见“**如有需要的话创建一个音乐播放器，或者已经如果有一个播放器的话，则让它停止播放**”代码段）。但是当音轨变化或者新的音轨添加进来你会收到提示。跟许多其他音乐播放器一样，播放歌曲的时候可以显示歌曲名字很好。让我们把这个功能添加进来。

这个音乐播放器可以设置为发送通知。当状态变化（比如播放，暂停或者停止）或者当播放的一首歌曲发生变化（比如切换到下一个音轨）的时候会发出通知。这样程序会知道什么时候去显示音轨名称。

问题

需要在播放器播放的时候显示音轨名称。

解决方案

当变化发生的时候，你会指定播放器发出通知。同时你会设定你的类为这些通知的观察者（模式），当播放新的歌曲的时候来显示它的名字。

讨论

回到刚才创建音乐播放器的时候，你设定了重复模式，让它发送通知、设定类为观察者：

```
[musicPlayerbeginGeneratingPlaybackNotifications]; [[NSNotificationCenterdefaultCenter]
addObserver:self
selector:@selector(handleNowPlayingChange:)
```

现在需要实现 handleNowPlayingChange: 方法来显示歌曲名称。音乐播放器能够返回当前播放的音轨到你的方法。你已经知道了怎么去获得标题等属性，你要做的是创建一个 label，设定显示的文字，显示 label 到界面上，见以下代码：

开始处理音乐播放器的通知

```
-(void)handleNowPlayingChange:(NSNotification*)notification {
MPMediaItem *nowPlaying = [musicPlayernowPlayingItem];

NSString *title =
[nowPlayingvalueForProperty:MPMediaItemPropertyTitle];
if (nil == lblTitle)
    {

CGRect r = self.view.frame;
r.size.height = 44;
lblTitle = [[UILabelalloc] initWithFrame:r]; [lblTitlesetTextAlignment:UITextAlignmentCenter];
[lblTitlesetTextColor:[UIColorblueColor]]; [lblTitlesetFont:[UIFont boldSystemFontOfSize:22]];
[lblTitlesetBackgroundColor:[UIColorblackColor]]; [lblTitle setAlpha:0.8];
                    //半透明
        [self.viewaddSubview:lblTitle];

    }
    [lblTitlesetText:title];

    [lblTitlesetHidden:NO];
    [lblTitleperformSelector:@selector(setHidden:)
                    //两秒之后隐藏

withObject:[NSNumbernumberWithBool:YES] afterDelay:2.0];
```

}

跟你看到的一样，使用好几种方法分割显示音轨，包括文字颜色，背景颜色，字体。同时，2 秒钟后，调用 setHidden: 方法来移除这个 label。在设备上运行程序，选择一首歌来播放，你可以观察下 lable 的改变（见图 7.20）label 会显示在你们内容上面，但是只显示两秒钟。

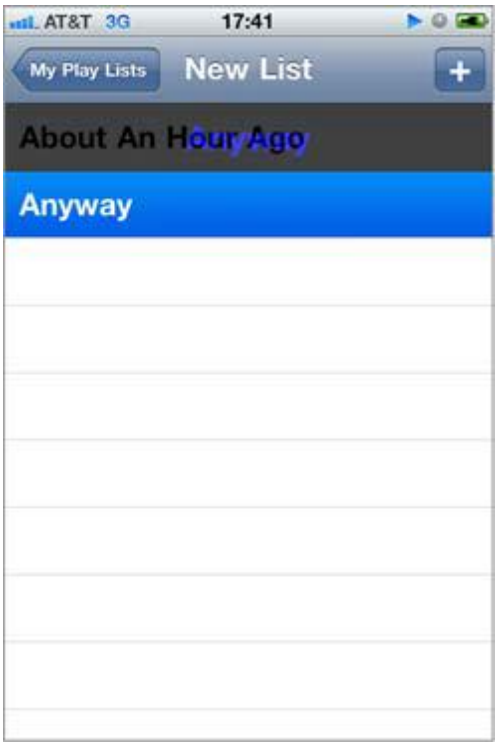


图 7.20 音乐播放器音轨改变通知时显示音轨的标题

音乐播放器其他的通知包括调整音量，状态变化（比如暂停播放）。是否处理这些通知取决于你的播放器应用。

这里也有一些其他后台来的通知，允许你的程序来响应锁屏时候的音频控制，甚至改变锁屏的背景图片，参见 http://wp.me/puRFY-hX。

播放音乐或者选择某一首歌来播放也都不是很难，对于一个其他应用，这是很棒的一个插件功能 - 用户来选择音轨。不管它是一个游戏，或者一个练手的应用或者其他，增加音乐播放功能都可以作为提高用户体验的一个选择。

7.5 总结

播放音乐或许适用于你开发的应用的一个功能，但是 CoreData 和数据库在开发很多应用的时候都会用到。有很多方法来存储数据 - 包括用户的默认值，设置，普通文件等等。但是数据库让你快速、可靠访问或者存储数据，也提供通过关系关联数据的简单方法。

虽然某些概念以及去实现更为复杂的 CoreData 功能也许令人感到非常困难 ,但我鼓励你去练习和体验其他例子和教程。不久你就会适应在你的程序中使用它，CoreData 是很自然的、功能强大的工具。

如果你的应用有一个服务器端的组件和一个在线数据库，CoreData 会帮助你来设计一个同样的数据库。在下一章节，你会设计一个应用来和用户的设备通过互联网来通信。你将会看到在设备和服务器同时设计数据库，同时拥有数据的好处。

你看到了怎么去设计 CoreData 实体的属性、关系以及通过代码来管理数据。使用 media picker 界面和回调，你允许用户定义播放列表。通过没有用户界面的音乐播放器，来播放音乐，同时响应通知来改变标题。

这里示范了几种 iOS SDK 和框架提供的访问设备内特征和功能的方法。这些功能和用户界面可以让你组合一些更强大设备功能。在下一章，将会看到一些苹果提供给你的更强的服务器功能，包括推送通知和内置应用购买。

8 ——通过教你制作一个上架应用石头，剪子，布来学习推送通知和应用内置购买

本章包括：

- 通过 UrbanAirship 发送推送通知
- 应用内置购买
- 制作一个简单的游戏

推送通知是服务器给设备发送消息的一种方式，消息的类型包括文本，一个指定程序图标的徽标数字，一段可以在程序内播放的音频。

如果应用当前没有运行，用户将会看到通知，并且可以打开应用。如果应用正在运行，应用里面设计来处理详细通知信息的方法将会被调用。

应用会促进应用内置购买，但是付款处理要通过服务器支持。在应用内可以购买的物品都被存储在服务器。被存储的信息只是一个 ID，描述和价格。应用可以从服务器查询可售物品，并展示给用户，然后开始销售。

苹果的服务器提醒用户输入 iTunes 帐户的密码来处理交易。销售处理就像买一个应用一样，并且用户将会邮件收到一个收据。应用将会被告知销售的结果（比如成功或者失败），然后提供用户买的东西。

推送发起于服务器端，但是应用内置购买由应用的用户发起。用户在应用中买某些东西（比如一个游戏的新等级或者额外的特色功能）。苹果服务器提示用户登录他们的 iTunes 帐户，结果会让程序来处理错误，使新的功能生效，或者使用户买的任何东西生效。

在这章 ,我们将会看一看这两种技术如何在一个应用程序中工作。你将会学习到大致的，并且配置推送和程序内购买，然后你将可以设置 Rock,Paper,Scissors（石头，剪子，布）。

当创建完推送和应用内置购买，我们将会将他们添加到你的项目中去。对石头，剪子，布游戏，你将允许用户通过购买选项来扭转他们在游戏中失利的局面。当用户扭转局面的时候，应用将会发出一个通知告诉其他人他们做了什么。

这经常在一些形式的项目中看到新的概念 ,所以增加这些特色到你的应用将会是一个很好的学习经验。你也将会在下一章设计 Game Center 特色的时候用到相同的项目。

8.1 使用苹果推送通知

推送通知是一个应用未运行时仍能发挥作用的方法。许多应用都有一个服务器端，这通常是用来维护应用用的。服务器端可以查询数据库，存储数据，或者下载内容。

即使应用没有在运行，推送通知也允许应用（例如服务器）工作。提醒，升级和信息这些都是应用在没有运行的时候起作用的例子。

通过苹果服务器给设备推送的必要功能有很多种开发方式。UrbanAirship 有一种简便的方法来推送通知，你将在这个技巧中使用它。

通过这个技巧，你将看到如何给应用在苹果网站和 UrbanAirship 网址配置推送，并且可以了解他们两者间的关系。然后，设置几个必要的配置，在应用里面编写发送和处理推送的代码。和配置相比，编写应用内推送的代码是相对无关紧要的工作。

技巧 58 配置推送

在少数情况下下，在 Provisioning Portal 中的配置需要和默认的设置不同。推送和应用内置购买（IAP）就是两个少数情况。游戏中心是第三个，我们会在下一章当中涉及到。

大部分的配置方法是相同的，但是这里需要将为推送创建的证书导出到服务器。

问题

需要在 Provisioning Portal 中创建一个配置好并且可以推送的新 app ID。

解决方案

使用 Provisioning Portal，你可以创建一个新的程序 ID。你将使用 com.brainwashinc.RPS.

讨论

因为之前已经设置过证书，你需要到 Provisioning Portal 中登录（如果需要），并且点击 app ID。然后，点击新的 app ID，并且填上必要的信息（见图 8.1）。

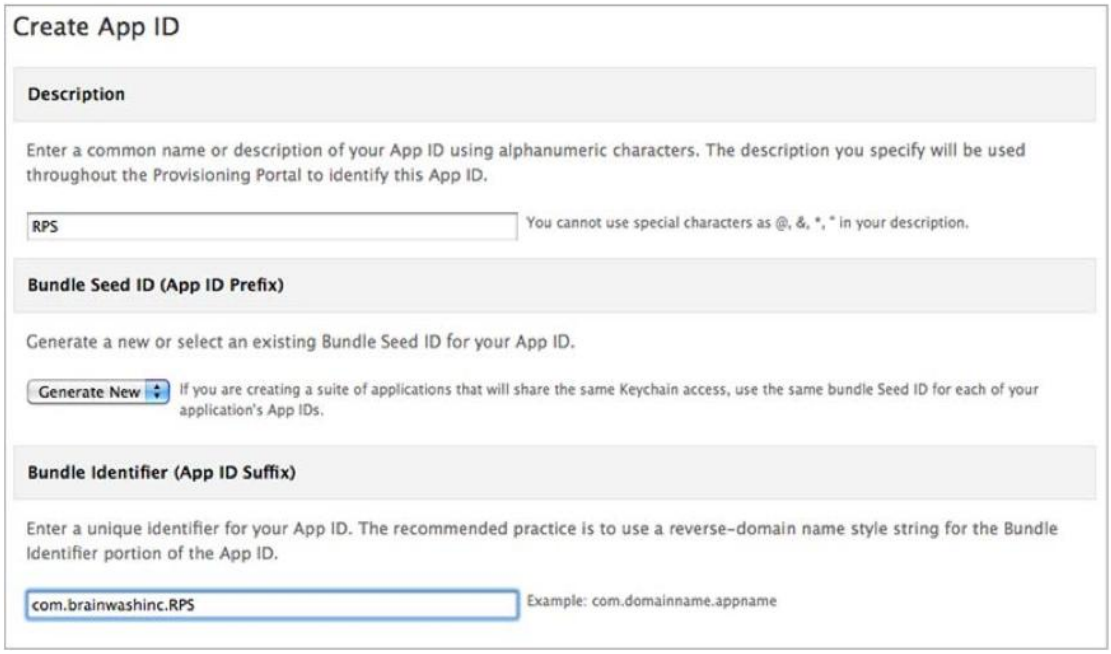


图 8.1 在 Provisioning Portal 中为石头，剪子，布创建一个新的应用 ID

点击 Submit 创建一个新的 app ID，注意在列表中的新的 app ID（见图 8.2）。还要注意打开 Game Center 和应用内置购买的功能 把推送通知的功能设置成可 Configurable（可调配）。左边栏是（Development）开发的配置，右边栏是（Production）产品的配置。

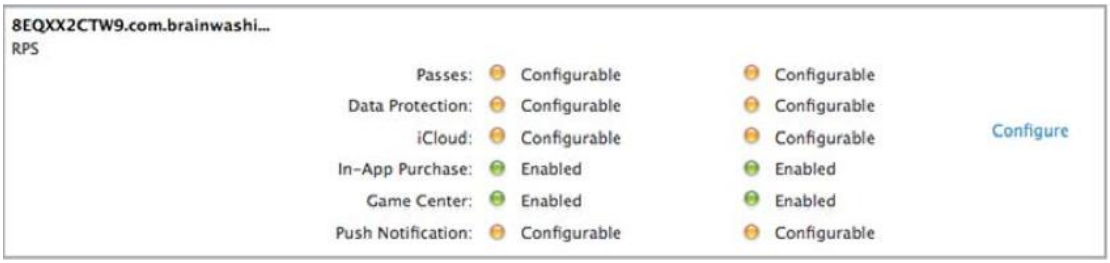


图 8.2 新建一个拥有游戏中心，程序内购买,和推送设置的新程序 ID

现在新的 app ID 创建好了，需要配置推送。点击离远处右边的 configure，并且检查苹果推送服务是否能够使用。点击行中开发那一列发送 SSL 证书的配置并且确认证书的签名和在 Portal 中注册的一样。在指引列表中查看详细。

证书准备好后 ,点击 Download 按钮并且保存文件。在 Finder 中双击保存的文件安装。现在你的新 appID 已经配置好了苹果的推送服务（APN）（见图 8.3）。

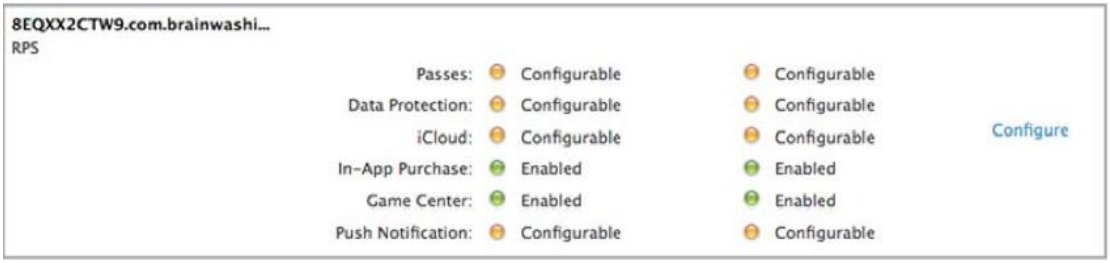


图 8.3 为开发配置好推送的新程序 ID

需要在 Provisioning 标签中用这个程序 ID 创建一个新的 provisioning profiles。请确认给 profiles 一个恰当的名字，并且选择新的 app ID—RPS。一旦创建完，下载这些 profiles，在 Finder 中双击安装。

不管在什么服务器上使用推送，都需要新的推送证书。打开 Keychain，查找新创建的证书，并且右击选择 export 导出证书（见图 8.4）。

图 8.4 从 keychain 中导出推送证书

当出现提示框，输入任何你喜欢的文件名并且点击保存。然后可以选择输入一个密码或者让它为空。不用仔细检查服务器端推送开发的各种选项，取而代之的，可以为应用使用 UrbanAirship。在这种情况下，最好采纳网站的建议，不要使用密码。

这是使用 APN 的基础，你需要为应用创建一个新的 app ID，并且在 Xcode 中配置新的推送证书和新的 provisioning profiles。现在来配置服务器端。

技巧 58 为 UrbanAirship 配置 APN

正如我在上一个技巧中提到的，有很多种方法在服务器端开发推送。取决于你在服务器端开发水平，你可能选择了一个又一个。

UrbanAirship 提供了一个已经成熟的，有很好 API 的解决方案，除此之外还是免费的（取决于产生的流量或者选择的套餐）。这样你可以将集中在 iOS 开发上，将服务器端开发放到一边。让我们开始使用 UrbanAirship！

幸运的是，UrbanAirship 在网络上为他们的服务和接口做的非常好。即使你是第一次使用它，在线的流程都非常有用且迅速。

问题

你拥有一个配置好推送的 app ID，但是需要配置服务器端。之前已经决定使用 UrbanAirship，所以将按照他们的流程来配置程序到他们的服务器。

解决方案

首先，需要创建一个 UrbanAirship 帐户并且在这个帐户中创建一个应用。然后可以尝试测试一下根据你创建的应用的信息生成的测试应用。

讨论

在 http://urbanairship.com 上创建一个 UrbanAirship 帐户。创建并且验证后，在帐

户里创建一个新的程序。（见图 8.5）

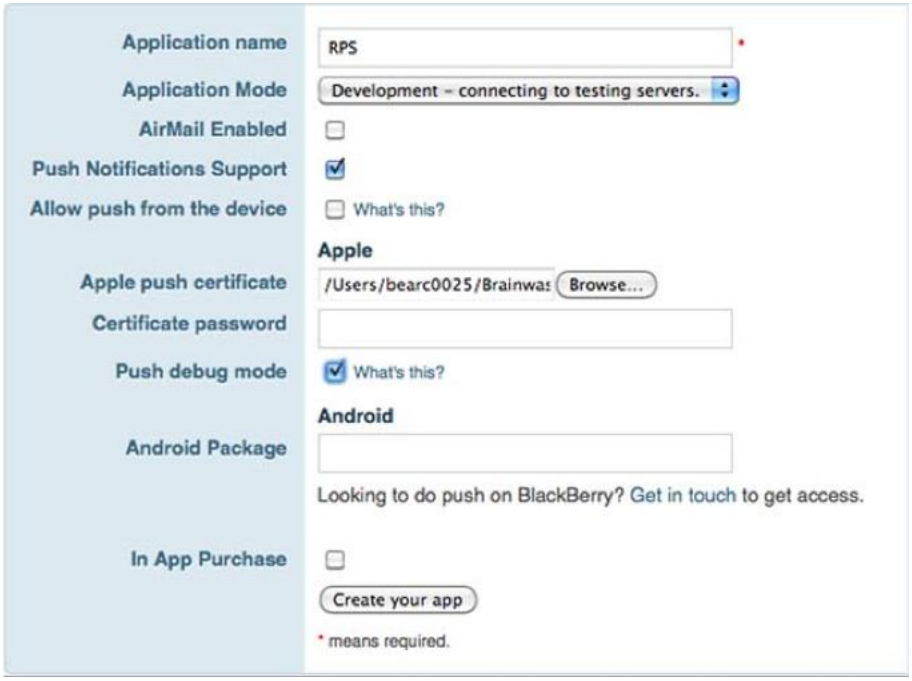


图 8.5 在 UrbanAirship 帐户中创建一个新的程序

当应用创建完成后，可以看见应用的详细信息，包含 app key，app secret，和 app master secret（见图 8.6）。

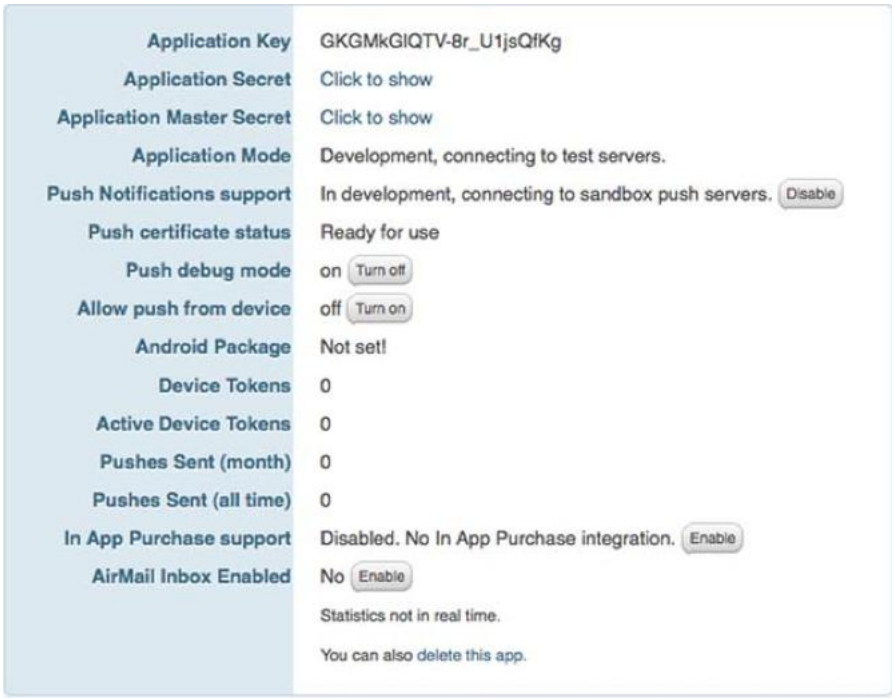


图 8.6 在 UrbanAirship 程序详情中包含程序 key 和 secrets

记住 key 和 secrets，你将在编码应用使用 UrbanAirship API 时需要用到这些。

UrbanAirship 文档可以在网站里找到，两个关键的页面 http://urbanairship.com/docs/push_index.html 和 https://docs.urbanairship.com /display/DOCS/Getting+Started%3A+iOS%3A+Push。可以在里面看到详细说明----特别有趣的是注册 device token 和发送一个推送请求。

注册你的 device token 是通过发送一个 HTTP PUT 到这个 URL：/api/device_tokens/<device_token>,当在苹果服务器注册完后，返回到你的设备。你可以在技巧 60 中了解到更多相关内容。

正如我说的，推送是在网络上发起的。至于 UrbanAirship，他们的服务器发送一个一个请求到 APN 服务器，请求给特定的设备发送信息。设备是通过 token 值来识别的。

但是需要发送一些信息告诉 UrbanAirship 来发送推送。可能是从网站发送，从服务器或者像这种情况，由应用来发送请求。请求的格式可以在 UrbanAirship 网站中的文档中找到。

让我们用UrbanAirship的测试应用来进行测试。在UrbanAirship的服务器里找到推送的测试客户端（Push NotificationTest Client）（也可叫做APNS例子），下载，在Xcode中打开。

图 8.7 通过 http://urbanairship.com 测试发送推送

使用之前给新应用创建的开发者 profile，设置 bundle identifier 和 Provisioning Profile 匹配。让这个测试应用看起来像是你的应用，和 Apple 和 UrbanAirship 匹配。在 APNSAppDelegate.m 中使用从 UrbanAirship 中得到的 key 和 secrets。

在设备上运行程序并且打开控制台查看日志。当设备 的 token 打印出来后，拷贝一下。在 http://urbanairship.com，打开推送标签，选择测试推送。粘贴设备 token，其他的设置可以随便设置，如下（见图 8.7）

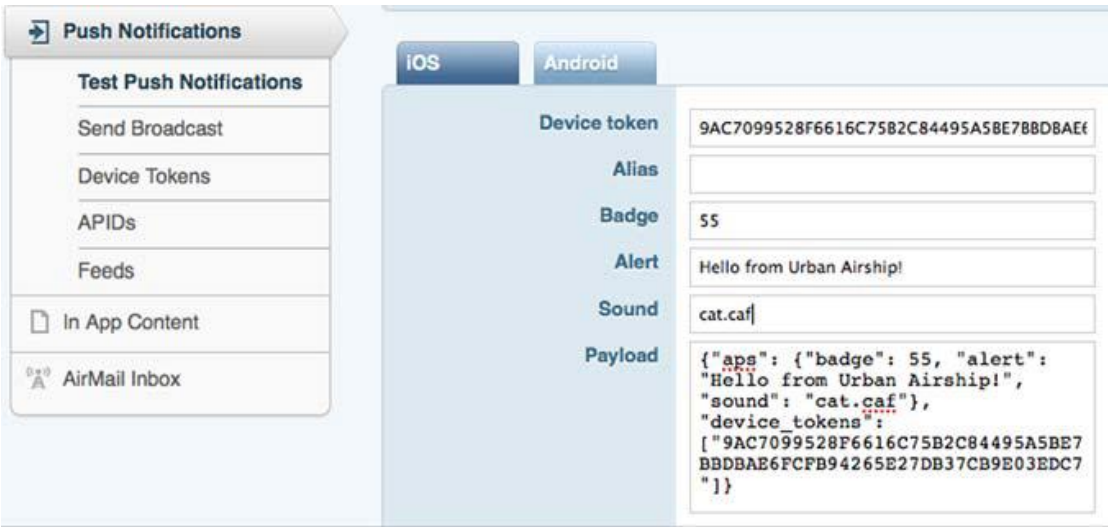


图 8.7 通过 http://urbanairship.com 测试发送一个推送到一台设备

应用运行时和没有运行时都发送推送，要注意两者的区别。（见图 8.8，8.9）注意 badge

数量设置，和选中的音频播放的时间。通知出现的形式可能是不同的，这取决于在设置中用户通知中心的设置。

图 8.8 当 demo 应用运行时进行的 UrbanAirship 推送通知

图 8.9 当 demo 应用没有运行时进行的 UrbanAirship 推送

当测试完成后，很有可能需要从设备上删除这个测试程序。不然，当用相同的 bundle ID 创建真正的应用时，设备将会不知如何是好。

此时，除了看 iOS 如何处理它本分之内的事 ，你已经基本上完成了所有的操作。你已经从开始到结束的方法都使用了，除此之外你还写了代码。服务器是巧妙的，所以，让我们开始着手 iOS 代码。

技巧 58 为应用注册推送

为了让推送服务分辨出设备，设备需要被注册。当成功注册完成，设备接受到一个服务器使用的 token 值。UrbanAirship 需要得到这个 token 值来知道让苹果给哪个设备发送通知。

苹果在这儿可以照顾到主要的工作，但是你首先需要要求苹果做这个工作，并且处理苹果的响应。让我们来看一下当程序运行的时候，如何来处理通知。

问题

你需要注册设备到APN服务器上并且处理返回的token值。你也需要处理当程序运行时收到一个通知的情况。在这种情况下，你需要显示notification payload。

解决方案

首先，创建一个新的项目，并且创建 app ID 的时候使用决定好的配置。然后，需要为程序注册推送服务和处理响应。同样的，需要通过实现适当的回调方法处理通知。

讨论

像之前做的那样，在 Xcode 中创建一个单视图的项目，并且使用 app ID 在 portal 中设置（见图 8.10）。

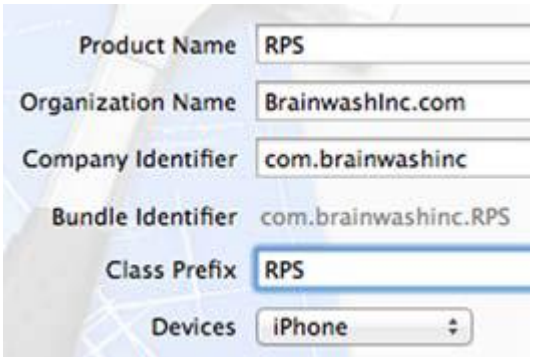


图 8.10 基于新创建的 app ID 创建一个新的 Xcode 工程

在 app delegate 类中，添加注册 APN 服务的代码，如下所示：

注册 APN 服务

```
[[UIApplication sharedApplication]
        registerForRemoteNotificationTypes: (UIRemoteNotificationTypeBadge
|UIRemoteNotificationTypeSound |
UIRemoteNotificationTypeAlert)];
```

有两个回调函数需要实现。一个是为成功接受 token 值情况准备的，还有一个是为注册失败准备的，见以下代码：

推送服务注册回调方法

```
- (void)application:(UIApplication *)app
didRegisterForRemoteNotificationsWithDeviceToken:(NSData

{
    NSLog(@"deviceToken: %@",deviceToken);
}

- (void)application:(UIApplication *)app
didFailToRegisterForRemoteNotificationsWithError:(NSError *)err

{
NSLog(@"APN Registration Error: %@", err);
}
```

在上面的代码中，打印出可用的信息。在第一个方法中，需要发送 token 到自己的服务器（或者 UrbanAirship，根据情况而定）。可以查看 token 和最后一次是否不同，如果没有改变可以不更新。

可以用一个类似的方法用来处理在程序执行期间接受到通知。同样的，根据情况，你可以记录并展示出信息（见以下代码）。根据应用的不同，可以做更多的有用的处理。

当应用运行时接受到通知，记录并且显示出来。

```
- (void)application:(UIApplication *)application
      didReceiveRemoteNotification:(NSDictionary *)userInfo
{
      NSLog(@"APN: %@", [userInfo description]);
      UIAlertView *alert = [[UIAlertView alloc]
                        initWithTitle:@"APN"
                            message:[userInfo description]
                           delegate:nil
                  cancelButtonTitle:@"OK"
                  otherButtonTitles:nil];
      [alert show];
                  //显示APN
}
```

这是应用准备接收推送通知的核心。苹果对这个设计很满意，取决于你和服务器的解决方案来决定需要做什么和怎么做。以现在的情况，需要当用户需要反败为胜的时候发送推送。让我们看看服务器解决方案。

技巧 58 在 UrbanAirship 上注册应用

现在，应用已经创建了，在苹果和 UrbanAirship 服务器上都做了配置。也已经使用 APN 服务器注册了设备。苹果现在知道如何发送你的通知通知，但是 UrbanAirship 还不

知道如何告诉苹果去发送推送。

技巧 60 会告诉你如何在 APN 服务器上注册设备，但是当 token 返回到应用时不知道如何处理。所以，需要在 UrbanAirship 上注册这个 token，这样它就会知道怎样告诉苹果给哪些设备发送推送。

问题

UrbanAirship 应用配置已经准备好了接收命令来发送通知给 APN 服务器，但是它现在还没有一个设备的 token。应用在 APN 服务器上注册了之后会得到一个 token，我们需要在 UrbanAirship 服务器上注册这个 token。

解决方案

通过UrbanAirship API，我们知道需要发送设备的token到/api/device_tokens/<device_token>。你还需要使用从UrbanAirship 应用配置得到的app key和app secret来发送请求，这个请求是作为一个PUT和完全授权。

讨论

让HTTP请求注册token有三件事情很重要：token的格式化，HTTP方法以及授权。

Token中需要没有空格，移除（<）和（>）符号。然后附在API里的URL尾端（见以下代码）。这让接受token的方法有了变化，现在它会重新格式化token然后在控制器里打印。

格式化APN token，记录到控制器

```
- (void)application:(UIApplication *)app
didRegisterForRemoteNotificationsWithDeviceToken:(NSData
*)deviceToken
{
NSString *stringToken = [[[[deviceToken description]
stringByReplacingOccurrencesOfString: @"<"
withString: @""]
```

```
stringByReplacingOccurrencesOfString: @">"
withString: @""]
stringByReplacingOccurrencesOfString: @" "
withString: @""];
NSLog(@"stringToken: %@",stringToken);
```

HTTP方法是PUT，这个方法会调用URL请求。让我们使用token创建API里的URL，然后把这个方法为设置为PUT，见以下代码：

给UrbanAirship API创建一个URL请求，设置这个方法为PUT

```
NSString *UAServer =
@"https://go.urbanairship.com/api/device_tokens/";
NSString *urlString = [NSString stringWithFormat:@"%@%@/",
UAServer, stringToken];
NSURL *url = [NSURL URLWithString:urlString];
NSMutableURLRequest *urlRequest = [[NSMutableURLRequest alloc]
initWithURL:url];
[urlRequest setHTTPMethod:@"PUT"];
```

而授权更加复杂，但是总体也可以接受。授权可以使用app key和secret来给请求增加一个头信息（header field）（见以下代码）。使用app key 和secret——两者之间 用（ ：）分离——创建一个string，然后把它转化为base 64（参考UrbanAirship的demo 应用看看可以如何转化为base 64）。然后在头信息授权的URL请求上把它作为一个value添加上。

在请求上创建授权的头信息，然后发送

```
NSString *keySecret = [NSString stringWithFormat:@"%@:%@",
appKey,
appSecret];
NSString *base64KeySecret = [RPSAppDelegate base64forData:
[keySecret dataUsingEncoding:NSUTF8StringEncoding]];
[urlRequest addValue:[NSString stringWithFormat:@"Basic %@",
```

```
base64KeySecret]
forHTTPHeaderField:@"Authorization"];
[[NSURLConnection connectionWithRequest:urlRequest
delegate:self]
start];
```

由于NSURLConnection的delegate是其本身，所以需要为回复以及失败处理回调方法（见下列代码）。把这些回复注销，现在还没有什么确实在工作。

处理为服务器请求回复的NSURLConnection回调

```
- (void)connection:(NSURLConnection *)theConnection
didReceiveResponse:(NSURLResponse *)response
{
NSLog(@"Server Response: %@\nStatus Code: %d",
[(NSHTTPURLResponse *)response allHeaderFields],
[(NSHTTPURLResponse *)response statusCode]);
}
- (void)connection:(NSURLConnection *)connection
didFailWithError:(NSError *)error
{
NSLog(@"Error Server Response: %@",
[error userInfo]);
}
```

现在为UrbanAirship接受命令来发送通知的准备工作已经完成了。苹果服务器已经准备好了，如果告诉苹果服务器发送通知，它就会发送，而UrbanAirship也知道如何告诉苹果服务器让它发送通知，去发送给哪些设备。现在，就让我们看看最终如何发送推送。

技巧58 发送一批推送通知

我们可以发送很多种通知，而发送它们的方法也有很多。使用UrbanAirship API，可

以了解如何发送各种类型的推送。

一个常见的方式是给单一的用户发送通知。另一种方式是发送一个广播信息，这意味着安装app的全部用户都会接收到这个通知（除非他们在系统设置中禁止了接收通知）。

问题

你需要发送一个广播信息给所有使用应用的用户。你需要格式化有效载荷（payload），然后使用合适的URL把它发送到UrbanAirship服务器

解决方案

通过UrbanAirship API，你知道需要在JavaScript Object Notation (JSON)中格式化有效载荷（payload）。也知道URL是/api/push/broadcast/。

通过设定app key以及master secret，请求通过服务器得到授权，然后发送给注册的token/设备。一般，master secret不会再app中存储，但是如果需要的话也可以在app中存储master secret。

讨论

创建一个APNSender的类来管理推送通知的功能。APNSender基于NSObject，也拥有相当一部分HTTPPost方法的属性来处理给UrbanAirship发送的不同的推送请求。

对广播式推送来说，则需要制定一个URL以及通知的payload（见以下代码）。需要包括一段文本信息，但是没有badge号或者需要播放的音频文件。

准备发送给UrbanAirship的指定推送信息的方法

```
+(void)sendAPNWithMsg:(NSString*)msg;
{
NSString *url = @"https://go.urbanairship.com/api/push/broadcast/";
NSString *payload = [NSString
stringWithFormat:@"{\"aps\":{\"alert\":\"%@\"}}",
```

```
    msg];
    NSString *resp = [APNSender HTTPPost:url body:payload];
    NSLog(@"Response: %@", resp);
    }
```

注意这个方法是静态的，因为并不真正需要一个实例来保存信息。也许你觉得采用另一种方法会更好。

可以写类似的方法，然后发送其他种类的通知。或者，你可以使用使用相同的代码来发送HTTP请求，见以下代码。

使用key和master secret来发送HTTP Post的方法

```
+(NSString*)HTTPPost:(NSString *)urlString body:(NSString*)body;
{
NSData *postData = [body dataUsingEncoding:NSASCIIStringEncoding
allowLossyConversion:YES];
NSURL *url = [NSURL URLWithString:urlString];
NSMutableURLRequest *urlRequest = [NSMutableURLRequest
    requestWithURL:url
cachePolicy:NSURLRequestReturnCacheDataElseLoad
timeoutInterval:10.0];
[urlRequest setHTTPMethod:@"POST"];
[urlRequest setValue:@"application/json; charset=utf-8"
forHTTPHeaderField:@"Content-Type"];
                        //JSON内容
[urlRequest setHTTPBody:postData];
NSString *keySecret = [NSString stringWithFormat:@"%@:%@",
appKey,
masterSecret];
```

```
NSString *base64KeySecret = [RPSAppDelegate base64forData:
[keySecret dataUsingEncoding:NSUTF8StringEncoding]];
[urlRequest addValue:[NSString stringWithFormat:@"Basic %@",
base64KeySecret] forHTTPHeaderField:@"Authorization"];
NSData *urlData;
NSURLResponse *response;
NSError *error;
                    //等待响应
urlData = [NSURLConnection sendSynchronousRequest:urlRequest
returningResponse:&response
error:&error];
if (!urlData)
return @"";
NSString *dataString = [[NSString alloc]
initWithData:urlData
                    //转化为string
encoding:NSASCIIStringEncoding];
return dataString;
    }
```

由于在UrbanAirship服务器上注册了token，这个请求设置了方法，但是这次才POST出来。这个请求包括了app key以及master secret，使用base 64来为授权的头信息编码。

然而，这次，没有使用delegate来接收回复。取而代之的，使用一个同步的请求call来等待。这个回复然后会转化为一个string，然后返回给调用者来处理。

增加一个按钮到UI，调用一个doBtnPush方法，用这个方法调用APNSender方法，这样操作来测试这个功能（见图8.11）。

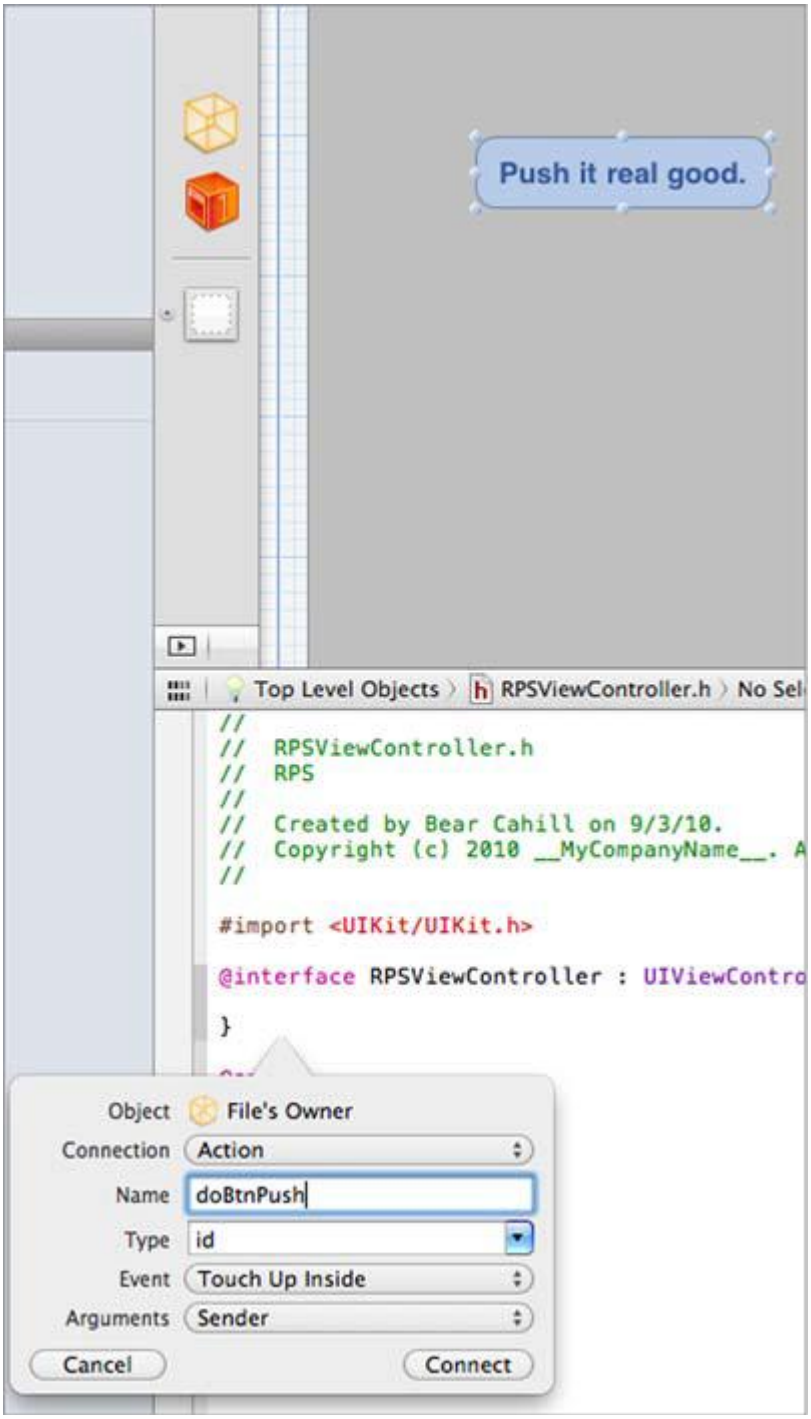


图8.11 给按钮添加doBtnPush：action来测试推送

现在有了doBtnPush:方法调用APNSender’ s sendAPNWithMsg:，并且发送测试string，比如说@"Test"作为信息。不要忘记在view controller导入APNSender类。

在设备上运行应用，然后触摸新的按钮来测试一下（见图8.12）。要确保在安装运行应用之前先移除UrbanAirship上的demo 应用。

图8.12 应用在运行的时候推送通知显示的形式

根据UrbanAirship API文档，我们了解到可以发送的通知形式和参数有多种。试着发送其他种类的信息，当应用没运行时发送，看看会如何显示。

现在你已经学会了APN的基础知识。现在剩下的大部分内容就只是主题了——发送通知到其他 设备，发送其他种类的信息，用不同的方式处理信息。根据需要发送的文本内容不同，可以选择不同的发送方式。

应用有一个服务器端就可以发送离线通知。应用允许用户设置用户名，用户之间可以发送消息。推送通知可以发送这些内容，比如说：提示，更新以及状态变化——比如说一个玩家在游戏中反败为胜。

幸运的是，苹果处理了大部分推送功能的基础工作。剩下的时间的可以来想想其他的功能，然后接入这些功能让其运作。现在我们游戏是需要宣布用户反败为胜。但是首先必须要让用户可以购买道具，使用道具来反败为胜。让我们看看如何增加这个功能。

8.2应用内置付费（IAP）

游戏有一个免费版本和付费版本非常好，但是如果用户升级到付费版本需要放弃他们在免费游戏中的数据，成就以及图片会怎么样呢？

如果当你给游戏增加新的关卡，然后希望发布一个免费的版本，但所有的用户都可以免费的开启这些新增的关卡，那会怎么样呢？

给应用增加应用内置付费需要几个步骤，包括：配置iTunesConnect，创建一个测试账户，把测试账户增加到代码里。让我们来具体的看看这几个步骤。

技巧58 给应用内置付费设置iTunesConnect

就像推送通知一样，苹果也需要知道应用想要增加应用内置付费。在技巧58中，我们创建新的app ID的时候，打开了应用内置付费功能。这样设置好了Provisioning Portal。你仍然需要设置iTunesConnect来处理购买的细节问题。

iTunesConnect陈列了用户所有可以购买的道具。每个道具都有一个指定的product ID，应用用product ID来判定用户想要买什么。在运行的时候服务器也会询问这些product ID。

问题

需要在iTunesConnect里创建一个新的道具，为应用内置付费做准备。你也可以在iTunesConnect中创建多个道具贩卖，但是现在可以先看看创建一个道具怎么做，之后就可以以此为参照了。

解决方案

首先现在iTunesConnect中创建一个新的应用，然后再增加可以让用户购买的付费道具。

讨论

登陆http://iTunesConnect.apple.com ，然后点击 Manage Your Application。点击Add New Application然后设置app 的名称，SKU，以及bundle ID（见图8.13）。

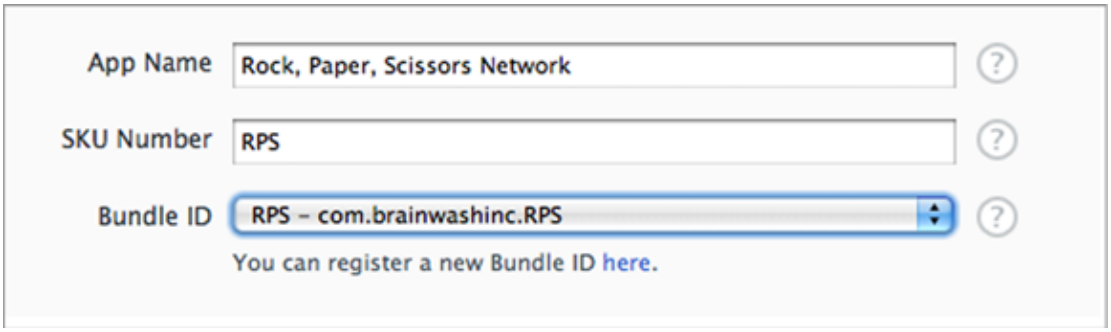


图8.13 在iTunesConnect里面创建一个新app

点击Continue，然后设置数据，价格，描述以及其他设定，包括图标以及至少一张的截图。这些值以后可以更改，所以现在可以任意的使用placeholder类型的值。点击Save，然后就可以看到这个新创建的应用了（见图8.14）。我使用了一个白色的图片作为我的placeholder，所以这块地方现在没什么好看的。点击Done回到应用的区域，然后点击Go Back回到iTunesConnect的首页。

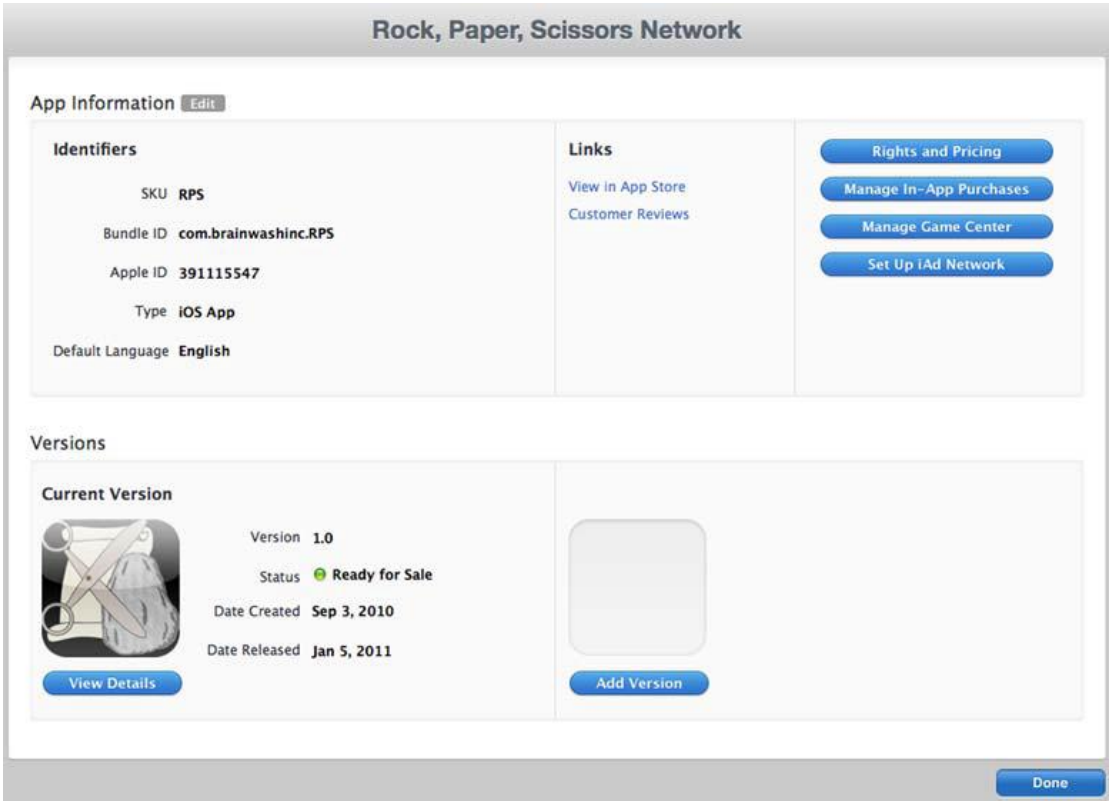


图8.14 在iTunes里创建了一个新应用

下一步，我们可以给应用增加IAP的道具。点击Manage Your In-App-Purchases，点击Create New，然后再单击刚刚创建的应用。

通过使用IAP我们解决了很多潜在的难题，包括订阅类的付费。用户可以通过IAP购买三类产品：消耗类，非消耗类以及订阅类的产品。

非消耗类的产品是指用户购买一次就能够一直拥有。比如说新的游戏关卡，媒体内容以及应用的高级功能。

消耗类产品是指用户购买一次，使用之后就会消失的物品。比如说在游戏中给汽车购买燃料，给一个角色购买一个暂时的能力，或者进入一次性的场景。

订阅类产品是两者的结合。用户可以重复的购买产品，但是购买的内容不会消失。比如说是每月获取报告或者像杂志类的媒体内容。

在这个应用中我们会销售一个“反败为胜”的消耗性道具。这个道具可以让用户在玩游戏的时候反败为胜。只能使用一次，所以是消耗性的道具，用户可以无限次的购买。

设置一个指引名称，product ID，类型，以及价格（见图8.15）。这些值除了价格用户都看不到。Product ID会由代码发送，可以是描述性的，也可以是非描述性的。比如说，我使用一个数字来显示它没有定义好的而是。反向域名解析（Reverse DNS）很好，有真正的价值，因为可能需要买几个应用的几种产品。

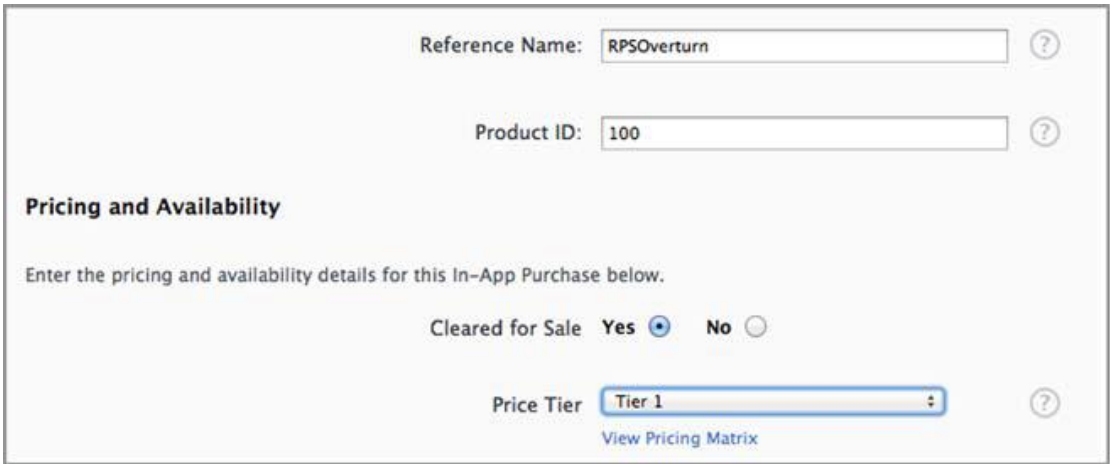


图8.15 设置增加一个IAP

你需要至少给这个产品选择一种语言（见图8.16）。我增加的语言是英语，然后给这个产品一个有意义的名字。显示的名称和描述需要给用户解释他们买的到底是个什么东西。

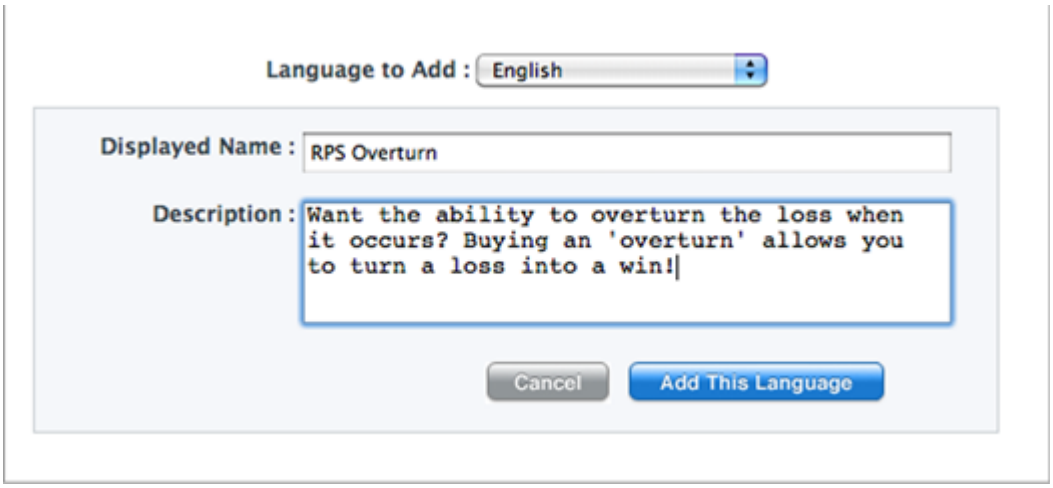


图8.16 给一个IAP产品增加一种语言和描述

完成后，点击Save，然后就可以看到这个新产品显示出来了（见图8.17）。

图8.17 在iTunesConnect中显示的新的IAP产品

请注意status要设置为Pending Developer Approval。点击产品，给这个产品增加一个截图，然后点击Approve。

请注意在文本框中显示的备注：“如果需要这个IAP和您的这个应用版本一同审核，请访问app’s Version Details页面，然后选择这个IAP。“如果不这样做的话，这个IAP项目需要进行单独的审核。这样的规则，让我们也可以增加IAP项目，但是不升级应用。

现在，我们可以测试IAP了。现在在沙盒的测试环境可以进行销售。你可能还没有沙盒环境下的账号。所以先让我们创建一个账号。

技巧58 创建一个in-app测试账号

在沙盒环境下测试IAP可以让你使用苹果服务器来进行购买和处理虚拟的产品订单。同时，也可以让你无需花钱就可以进行购买。不幸的是，只能在沙盒环境下这样做。

到目前为止，你做的所有工作都和在沙盒环境下进行测试是一致的。产品版本方面需要进行的步骤也是相同的。不同的是，当应用上线了的话，真正的用户可以访问应用。现在，我们需要一个用来访问的特殊账号来测试购买。iTunesConnect提供了创建测试账号的方法。

问题

你需要一种可以测试IAP的方法。苹果允许所有的产品都可以在沙盒环境下购买，所以

我们可以在苹果的服务器上进行测试。但是需要一个测试账号。

解决方案

在iTunesConnect中，创建一个测试账号来进行购买测试。

讨论

登陆http://iTunesConnect.apple.com，点击Manage Users。选择In App Purchase Test User然后店家Add New User。

使用真实方便记忆的信息来填写账号信息。不需要对邮箱进行验证或是需要进行信息验证，但是使用以后可以记住的信息总是好的（见图8.18）。

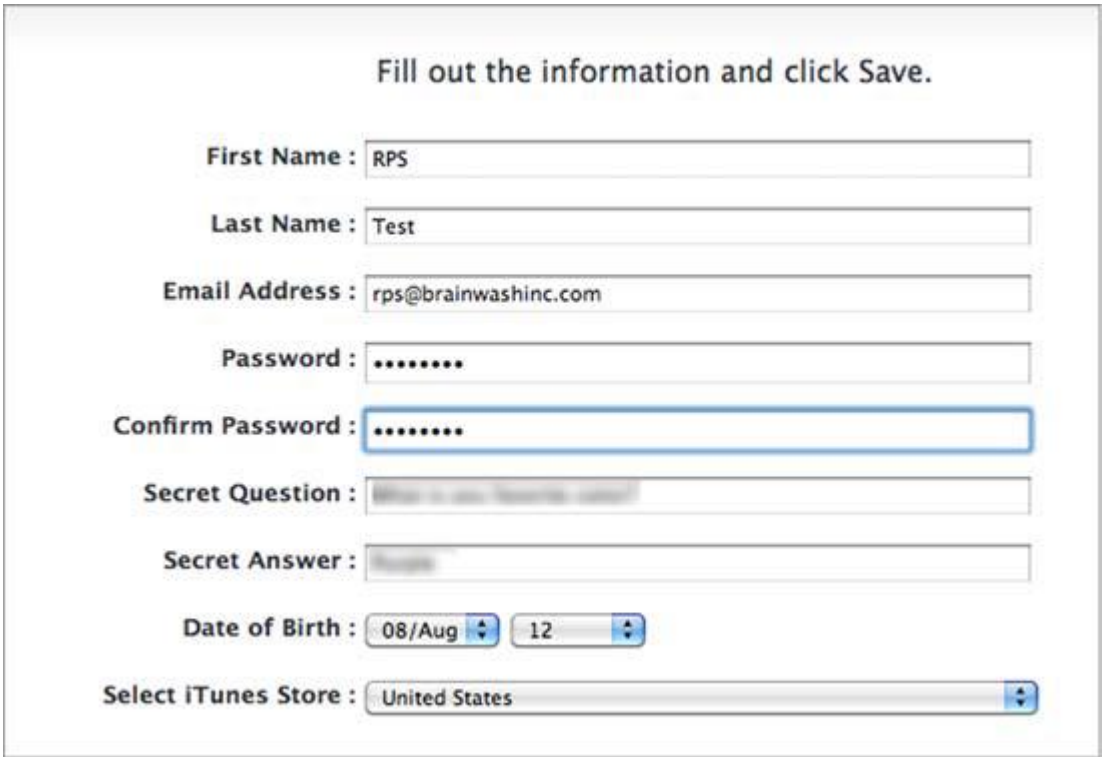


图8.18 在iTunesConnect中创建一个in-app测试账号

点击Save，账号就创建好了。现在可以在沙盒环境下测试IAP购买。

我们已经创建了用于贩卖的IAP产品，也有一个可以用于购买的账户。让我们现在来编码增加功能，进行测试吧。

技巧58 给项目增加IAP

IAP功能我们已经完成了2/3。现在已经创建了用于贩卖的产品以及用于购买的产品，现在我们需要增加代码。

苹果处理了大部分的难点。用户使用他们的iTunes账号来进行购买，苹果也为我们处理了登陆的过程。基本上，我们需要当用户要购买的时候来初始化购买然后处理订单。

使用StoreKit类，你可以创建一个订单，然后把订单请求加入苹果服务器处理的队列。通过设置类到observer，你可以接收交易的结果，然后对交易进行处理。

问题

你需要给工程增加一个IAP，在交易开始的时候进行处理，然后处理订单。

解决方案

使用StoreKit框架来初始化IAP产品的订单。可以设置类为observer来回调基于购买结果调用的方法。你需要处理成功购买的订单，失败的订单，以及重复的订单。

讨论

给工程添加StoreKit框架，把它导入到RPSViewController头文件。你还需要给工程添加一个Buy按钮，调用一个新的action叫做doBtnBuy:。

doBtnBuy:方法需要初始化购买，同时还需要检查这个账户是否可以进行购买。因为有些账户没有对支付进行配置。例如，父母为他们的孩子把支付功能关闭了。

如果账户可以进行支付，则创建SKPayment实例，把它添加到默认的支付队列中，设置Self作为observer（见以下代码）。RPSOverturn需要作为你的IAP产品ID进行定义。以这个例子来看，是100。

如果账户通过了验证则初始化购买

```
- (IBAction)doBtnBuy:(id)sender;
```

```
{
if ([SKPaymentQueue canMakePayments])
{
SKPayment *payment = [SKPayment
paymentWithProductIdentifier:RPSOverturn];
                    //item ID
[[SKPaymentQueue defaultQueue]
addPayment:payment];        //增加到购买队列
[[SKPaymentQueue defaultQueue]
addTransactionObserver:self];        //接收结果
}
else
{
UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Not Authorized"
message:@"Not authorized to purchase."
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
}
}
```

你需要接入处理订单更新的方法。交易返回的时候是一串数列，这是因为可能有好几个订单。根据这种状况，你可以调用适合的方法（见以下代码）。由于我们只贩卖一种产品，即使用户在你收到回复之前开始下了多次购买，你也同样可以处理这些订单。

基于交易状态，调用合适的方法

```
- (void)paymentQueue:(SKPaymentQueue *)queue
```

```
updatedTransactions:(NSArray *)transactions
{
for (SKPaymentTransaction *transaction in transactions)
{
switch (transaction.transactionState)
{
case SKPaymentTransactionStatePurchased:
[self completeTransaction:transaction];
break;
case SKPaymentTransactionStateFailed:
[self failedTransaction:transaction];
break;
case SKPaymentTransactionStateRestored:
[self restoreTransaction:transaction];
default:
break;
}
}
}
```

下一步，你需要接入这三个方法。在每笔交易中，都通过在默认的支付列表上调用finishTransaction:方法来结束交易。

交易失败，则提供给用户一些信息来帮助用户了解情况。交易失败的号码也可以包含一些失败的原因已经建议，见以下代码。

给用户显示交易失败的原因

```
- (void) failedTransaction: (SKPaymentTransaction *)transaction
{
if (transaction.error.code != SKErrorPaymentCancelled)
{
```

```
NSString *messageToBeShown = [NSString
stringWithFormat:@"Reason: %@, You can try: %@",
[transaction.error localizedFailureReason],
[transaction.error localizedRecoverySuggestion]];
UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Unable to complete your purchase"
message:messageToBeShown
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
}
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
}
```

还原状态说明用户之前购买过这个物品，这个物品可以被还原。这是给非消耗性产品服务的，比如说用户购买了一个新的游戏关卡。如果用户丢失了某些包括购买的数据的时候，这样的情况会发生。这次获得产品不用付费，应用要还原之前用户购买的产品。以我们的情况来说，这种情况不会发生，因为我们只贩卖消耗性产品，但是以后我们总会需要处理这种情况，完成交易，见以下代码。

还原状态下完成交易

```
- (void) restoreTransaction: (SKPaymentTransaction *)transaction
{
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
}
```

最后，如果购买成功，增加“反败为胜”的数量，把此次购买保存到设备。最后注销，来看看发生了什么，见以下代码：

成功购买后增加“反败为胜”的数量并储存

```
- (void) completeTransaction: (SKPaymentTransaction *)transaction
{
int overturns = [[NSUserDefaults standardUserDefaults]
integerForKey:RPSOverturnCount];
overturns++;
NSLog(@"number of overturns: %d", overturns);
[[NSUserDefaults standardUserDefaults]
setInteger:overturns
forKey:RPSOverturnCount];
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
}
```

现在可以准备测试购买了。在运行应用之前，打开iTunes，找到Featured标签，然后找到页面的底部。点击你的账户名称，注销。如果跳过这步的话，当场试购买的时候，就会使用你现在的iTunes account。

在设备上运行应用，点击Buy按钮（见图8.19）。需要登录iTunes账号，这次确保使用IAP测试账户。

图8.19 使用IAP测试账户登录来进行购买

请放心的在代码中增加其他的log，查看log控制器了解进程情况。应该可以看到“反败为胜”的数量增加了，每次购买完成后都会显示“感谢，此次购买成功”的信息。

现在我们只贩卖一种产品，但是以后你也许贩卖多种产品。这样的话，你也许不希望增加所有的ID到代码中。StoreKit框架有其他支持更多购买相关功能的方法，比如说是根据一个给定的产品标识来询问产品的详细信息。

恭喜！到现在我们已经完成了不少功能。我们现在本章中接触的这些概念都有着坚实的基础。这些低级的功能对支持应用的功能性和实用性来说非常棒，但是它们它们本身并不足以创作一个应用。

现在我们有了支持应用的产品，让我们把它们写出来吧。

8.3 石头，剪子，布

通过在线配置应用以及为推送通知和IAP编写支持方法，现在我们已经完成了不少功

能。但是你需要一个游戏来让这些功能可以有用起来。

我们使用简单但经典的游戏石头，剪子，布。用户可以做选择，然后一个随机数字会决定电脑对手的选择，然后确定输赢。

我们还将处理用户想要反败为胜的情况。如果用户购买了反败为胜道具，那么他们就可以赢得游戏。否则，你要让用户了解他们需要购买反败为胜的道具。

技巧58 设计游戏

需要编码然后给游戏增加一些界面元素，同时还需要处理游戏的玩法。游戏非常简单，但是对大部分人来说非常经典，有重复玩的价值。

UI基本上就是三个按钮提供三种选择，还有一个按钮提供反败为胜的功能。你的UI需要像其他的按钮一样处理这些action。但是你还需要根据游戏的逻辑来处理用户的选择。

问题

需要用三个选择键，一个反败为胜键以及一个购买按钮来给石头，剪子，布游戏来设计UI。

解决方案

使用IB以及Xcode的UI editor来创建界面，outlet以及action。

讨论

打开RPSViewController.xib文件，给按钮，action，outlet创建UI layout(见图8.20)。

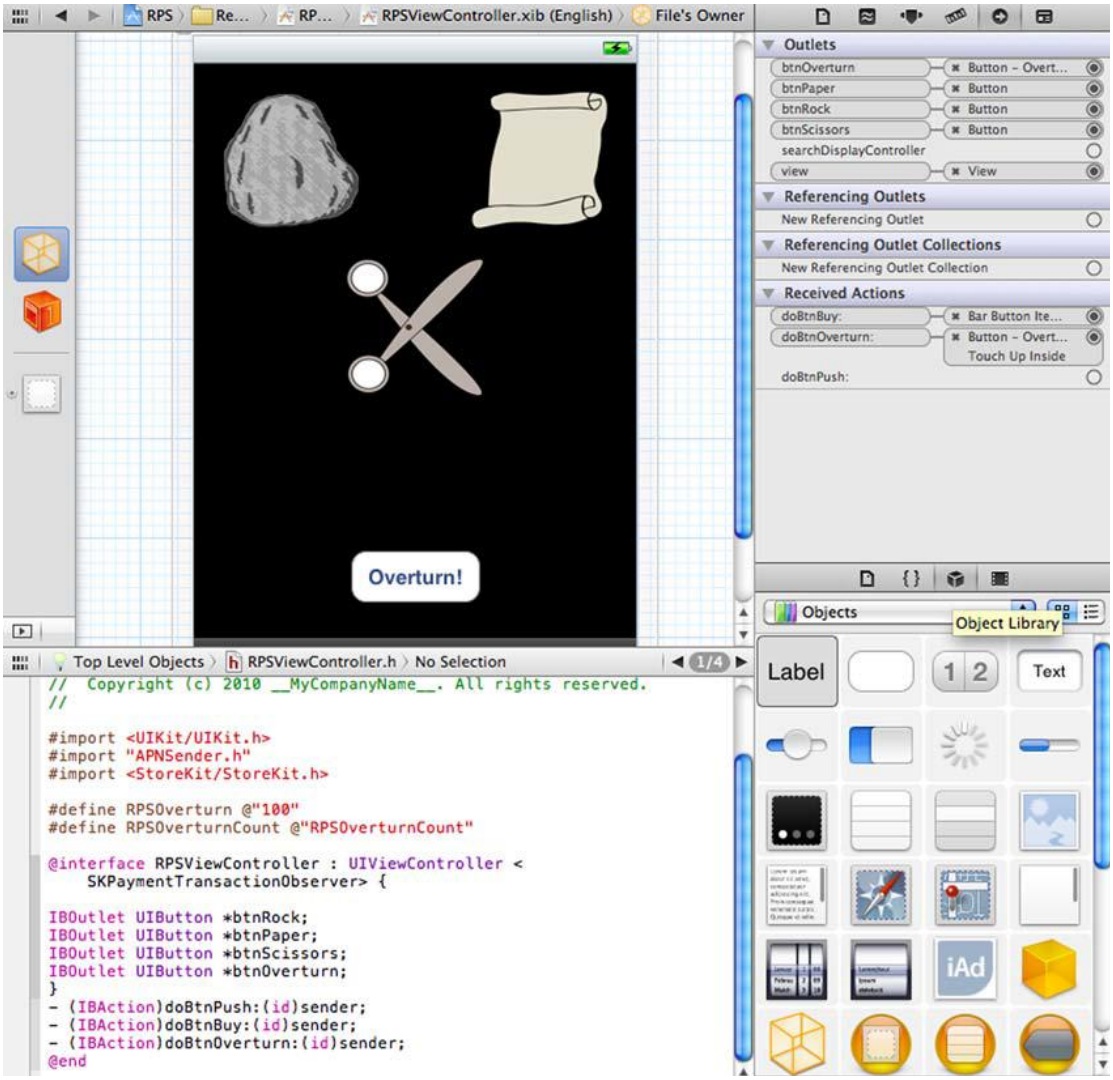


图8.20 给石头，剪子，布游戏设计UI

对Buy按钮来说，在底部的工具栏是最佳选择，因为它与其说是游戏的一部分，不如说是应用的一个功能。

在头文件中，需要定义一些用于玩家选择的值，见以下代码。

在头文件中定义的用户选择

```
#define ROCK 0
#define PAPER 1
#define SCISSORS 2
```

在游戏中，你还需要存储一些值，比如说用户的选择，这回合游戏的输赢，本轮游戏的总分，见以下代码。

在游戏中储存游戏信息的数据成员

```
int myChoice;
bool isWin;
int gamePoints;
```

下一步，需要处理用户做选择时候按钮的action，见以下代码。

处理用户选择的方法

```
- (IBAction)doBtnRock:(id)sender;
{
myChoice = ROCK;
[self waitForOtherPlayer];
}
- (IBAction)doBtnPaper:(id)sender;
{
myChoice = PAPER;
[self waitForOtherPlayer];
}
- (IBAction)doBtnScissors:(id)sender;
{
myChoice = SCISSORS;
[self waitForOtherPlayer];
}
```

在我们的游戏中，电脑扮演了另一个玩家，然后生成一个随机的数字来进行选择（见以下代码）。电脑选择完之后，就来处理游戏的结果。

电脑角色的选择以及调用方法来处理结果

```
-(void)waitForOtherPlayer;
{
int otherPlayerChoice = rand()%3;
```

```
[self processOtherPlayersChoice:otherPlayerChoice];
}
```

处理游戏结果包括了对两个选择进行比较，设置胜利的标识，给用户增加分数，见以下代码。

加工结果来决定双方输赢以及分数

```
-(void)processOtherPlayersChoice:(int)otherPlayerChoice;
{
isWin = NO;
NSLog(@"My Choice: %d Other Player's Choice: %d",
myChoice, otherPlayerChoice);
if (otherPlayerChoice != myChoice)
{
int diff = myChoice - otherPlayerChoice;
if (diff == 1 || diff == -2)
{
gamePoints++;
isWin = YES;
}
}
NSLog(@"Win: %@ Points: %d", isWin ? @"Yes" : @"No", gamePoints);
}
```

由于选择基本是数字上的，所以只要进行基本的运算看它们是否相等或是其他。如果不相等的话，1 比 0,2 比 1，或是 0 比 2 就是赢。

打印出选择的 log，分数以及赢的信息，如果可以显示用户的总分就再好不过了。可以给 UI 增加一个标签，这个标签做为处理方法中的类的一个 outlet，然后对它进行更新，见以下代码。

更新标签，显示给用户总分结果

```
[lblPoints setText:[NSString stringWithFormat:@"Points: %d",
gamePoints]];
```

现在有了 log 和标签来显示分数数据（见图 8.21）

图 8.21 Xcode 控制器和模拟器运行应用，显示分数数据

玩游戏会有随机的输赢。输了很讨厌，所以给用户一个可以反败为胜的机会，而你也可以从中获得盈利，这是双赢。

技巧 58 反败为胜

反败为胜很简单，至少在这个游戏中。你需要个反败为胜(Overturn)按钮接入 action。doBtnOverturn:action 需要检查反败为胜的数量，如果有的话就使用一个。

如果用户没有反败为胜道具的话，可以让他们知道，建议他们购买更多的道具。不让用户选择而自动进行购买不是个好主意。这样会让用户感到沮丧，造成流失。直到收到账单的

时候用户才会知道他们进行了购买。这样非常不好。

问题

需要给用户反败为胜的选择机会。应用需要检查反败为胜道具的数量，然后根据数量进行合适的处理。

解决方案

如果用户购买了反败为胜道具，这些道具会被储藏在设备中，如果没有购买，则其值为 0，这样的工作方式非常完美。我们可以检查值然后反败为胜。

讨论

我们可以获取到设备上反败为胜道具的数量，如果道具数量为 0，则提示用户。见以下代码。

检查反败为胜的道具数量，如果数量为 0，则提示用户

```
-(IBAction)doBtnOverturn:(id)sender;
{
int overturns = [[NSUserDefaults standardUserDefaults]
integerForKey:RPSOverturnCount];
if (overturns == 0)
{
UIAlertView *av = [[UIAlertView alloc]
initWithTitle:@"No Overturns"
message:@"You do not currently have any overturns.\n\nTap
'Buy' below to buy overturn credits."
delegate:nil cancelButtonTitle:@"OK"
otherButtonTitles:nil];
[av show];
return;
}
```

如果有反败为胜的道具，但是游戏结果是赢，则可以告诉用户他们不需要反败为胜，见以下代码。

通知胜利的玩家他们不需要使用 反败为胜道具

```
if (isWin)
{
UIAlertView *av = [[UIAlertView alloc]
initWithTitle:@"No Overturns"
message:@"Ummm, but you won."
delegate:nil cancelButtonTitle:@"Oops"
otherButtonTitles:nil];
[av show];
return;
}
```

最后，经过了这些步骤后，我们至少还拥有一个反败为胜的道具，而本轮游戏输了。则减少反败为胜道具的数量，储存数据，然后重新设置赢的标识，增加比分，见以下代码。

重新设置反败为胜道具的数量，重设标识，以及比分

```
overturns--;
[[NSUserDefaults standardUserDefaults]
setInteger:overturns
forKey:RPSOverturnCount];
isWin = YES;
gamePoints++;
[lblPoints setText:[NSString
stringWithFormat:@"Points: %d", gamePoints]];
[APNSender sendAPNWithMsg:@"Loss overturned!"];
```

请注意你在标签中设置了分数，然后发送推送（给所有人）告诉他们反败为胜的消息。在现实世界，这个主意不是太好，但是现在是为了好玩，而且你现在是唯一一个看到消息的

人！

8.4 总结

我们已经学习了很多关于 Provisioning Portal，iTunesConnect，UrbanAirship 以及 Xcode 的知识。这些知识都需要非常熟悉，在需要它们的时候可以把它们增加到应用中去。

使用 iTunesConnect，可以让应用使用推送以及 IAP 功能。然后可以在服务器导出从 KeyChain 获得的证书来进行推送。在 iTunesConnect 中，可以创建管理应用可以贩卖的产品，然后创建一个测试账户在沙盒中来测试购买。

这些都是重量级的概念，尽管我们的游戏不是重量级的。在下一章中，我们会继续开发我们的游戏，看看怎么样会让这个游戏变的更好。增加一个功能可以允许用户联网和其他用户进行游戏。这样的话游戏就会变得有趣了，反败为胜是有意义的！

——给石头，剪子，布游戏添加 Game Center 排行榜以及成就

9

本章包括：

- 接入 GameCenter
- 给游戏增加排行榜
- 和其他玩家一起玩游戏，聊天

GameCenter 可以帮助你给你的游戏增加更多的社交方面的内容。它包括了排行榜，成就，以及在应用内聊天。在本章中，你会给第八章中制作的石头，剪子，布游戏增加 GameCenter 功能。

使用 GameCenter，玩家可以查看上次游戏的得分和成就，也可以看到朋友的得分和成就。他们可以通过 Gamecenter 邀请朋友来玩游戏，匹配设备之后就可以和朋友一起玩游戏了。

在本章中，你将学会如何验证 GameCenter 的玩家，以及使用排行榜和成就。排行榜会让应用存储玩家的分数，然后以一个 view 的形式显示排行榜。成就和排行榜相同，但是是以百分数的形式储存（例如，100%代表玩家完成了这个目标）。

最后会学习如何匹配玩家，使用 GameCenter 让玩家在应用中聊天。

9.1 Game Center 验证和排行榜

GameCenter 要求用户创建一个用户名和密码来储存和显示分数和成就数据。很多工作和 UI，GameKit 已经做好了，但是关于应用和 iTunesConnect 我们还有一些工作要做。

在本小节中，你将会学到如何给在石头，剪子，布游戏中增加 GameCenter 的用户验

证，给应用在 iTunesConnect 中配置排行榜，存储并显示用户的数据。

技巧 68 验证用户

问题

如果用户没有进行账户验证的话就不能参与 GameCenter 的互动，不能储存和显示分数。GameKit 做了大部分工作，但是应用必须要告诉 GameKit 来进行验证。

解决方案

石头，剪子，布应用需要使用 GameKit 来获取 GKlocalPlayer 实例，开始验证过程。如果需要的话会显示 UI，否则验证会自动进行。

讨论

GameKit 框架处理了大部分关于 GameCenter 的较困难的工作。打开石头，剪子，布工程，给工程增加 GameKit 框架。基于 NSObject 创建一个新的类，给它命名为 UtilGameCenter。在这里你可以把关于 GameCenter 的代码单独分离出来，在以后的工程可以进行复用。

在新的头文件 UtilGameCenter.h 中，导入 GameKit/GameKit.h，声明一个 bool 标识（flag），命名为 gameCenterFeaturesEnabled（见以下代码）。如果用户通过了验证，这个标识会设置为 Yes，就可以使用 GameCenter 的功能。

UtilGameCenter 头文件导入 GameKit 和标识

```
#import <Foundation/Foundation.h>
#import <GameKit/GameKit.h>
@interface UtilGameCenter :NSObject {
boolgameCenterFeaturesEnabled;
}
@property (nonatomic, assign) boolgameCenterFeaturesEnabled;
@end
```

请确定在实现文件中 synthesize gameCenterFeaturesEnabled。

在头文件中声明一个验证方法然后把它接入到.m 文件中来获取本地玩家，然后通过 GameCenter 验证玩家，如以下代码所示。

获取 GKLocalPlayer，通过 GameCenter 验证

```
-(void)authenticate;
{
GKLocalPlayer *locPlayer = [GKLocalPlayerlocalPlayer];
[locPlayerauthenticateWithCompletionHandler:^(NSError *error)
{
if (!error)
self.gameCenterFeaturesEnabled = YES;
else
self.gameCenterFeaturesEnabled = NO;
        //设置标识
}];
}
```

验证方法可以可能在任何时候被调用 ,所以当 UtilGameCenter 初始化的时候调用它，见以下代码。

UtilGameCenter 初始化方法调用 GameCenter 验证

```
- (id)init {
if ((self = [super init])) {
[self authenticate];
}
return self;
}
```

返回 RPSViewController，你需要开始 GameCenter 的整个进程。在 RPSViewController 头部导入 UtilGameCenter.h 文件，声明一个 UtilGameCenter 的 GameCenter 变量。在 viewDidLoad 方法中，创建 UtilGameCenter，用它来做验证工作，见以下代码。

RPSViewControllerviewDidLoad 开始 GameCenter 验证

```
- (void)viewDidLoad {
[superviewDidLoad];
gameCenter = [[UtilGameCenteralloc] init];
}
```

现在，运行应用时，GameCenter 也会加入，你可以设置账户，然后就登陆了 GameCenter 账户了。下一次再次运行应用的时候，验证会自动进行（见图 9.1）。

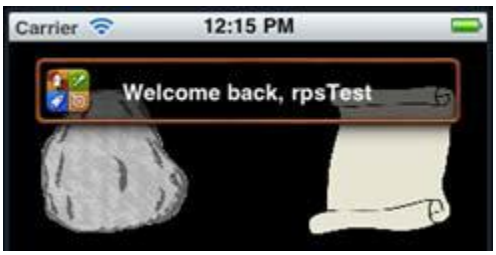


图 9.1 在应用启动时，GameCenter 通过验证后欢迎用户返回游戏

现在应用可以验证 GameCenter 用户了，但是还做不了其他什么事。GameCenter 的一个主要功能就是储存用户的分数，让我们看看如何设置 iTunesConnect，让应用可以使用 GameCenter 排行榜。

技巧 68 在 iTunesConnect 中设置排行榜

在 iTunesConnect 中配置排行榜非常简单。iTunesConnect 需要知道这个给定的应用有一个排行榜，还需要知道一些细节例如说数据的格式以及排行榜的名称。

问题

你需要给石头，剪子，布游戏在 iTunesConnect 中建立一个排行榜。这样应用可以储

存用户的分数。

解决方案

iTunesConnect 提供了一个在线接口可以给应用创建并配置排行榜。接口的形式非常清楚，用户登陆站点之后，只需很少的步骤就可以创建排行榜。

讨论

首先登陆 iTunesConnect (http://itunesconnect.apple.com)。 点击 Manage Your Applications，点击你的 app icon，然后点击 Manage GameCenter。点击 Enable，然后，在排行榜（Leaderboard）区域，点击 Add Leaderboard，然后选择 Choose under Single Leaderboard（见图 9.2）。

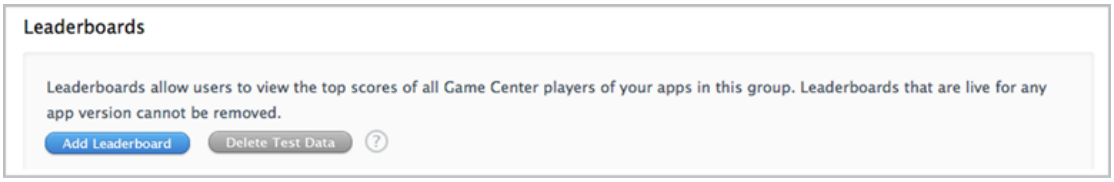


图 9.2 在 iTunesConnect 中给应用创建 leaderboard

给新的排行榜输入值，包括 reference name，leaderboard ID，format type，以及 sort order（见图 9.3 和 9.4）

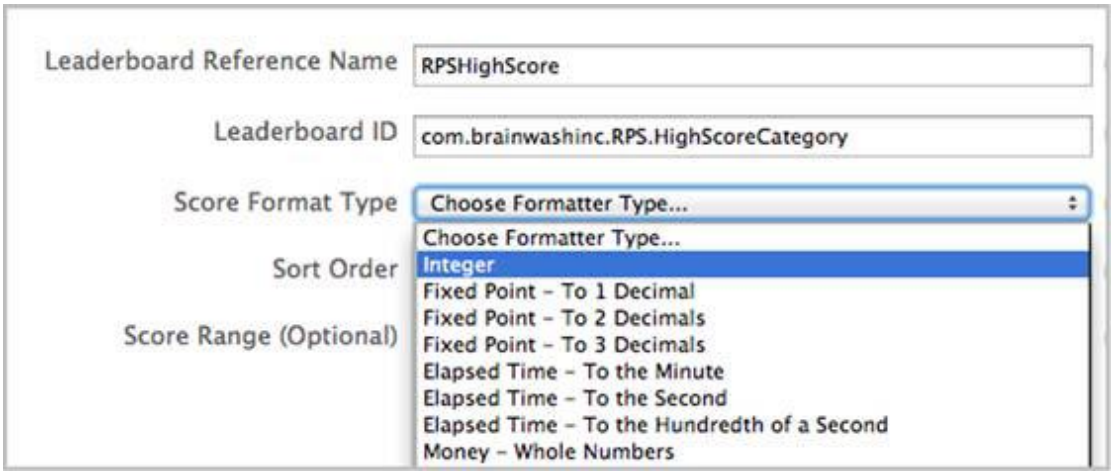


图 9.3 给排行榜配置 sort order 和 format type

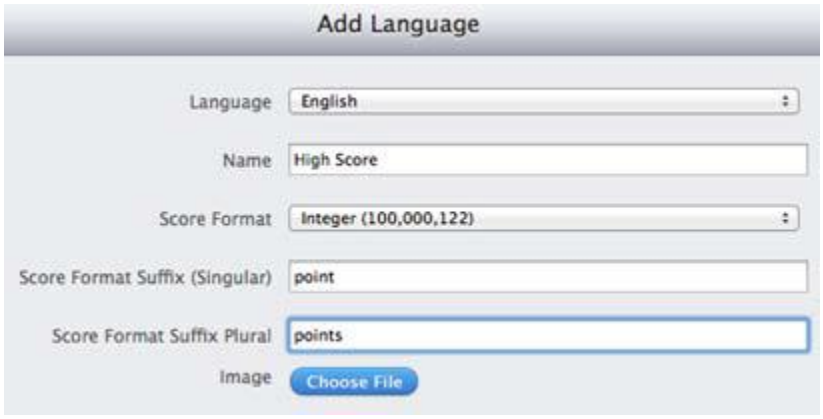


图 9.4 给排行榜增加一种语言

最后一步是增加至少一种语言。这会确定排行榜的数据格式，分数的后缀，以及用给定的语言显示名称。点击 Add Language，然后基于分数的类型做选择（见图 9.5）。Leaderboard ID 会在技巧 71 中使用，Leaderboard ID 用来指定在应用中显示哪个排行榜。

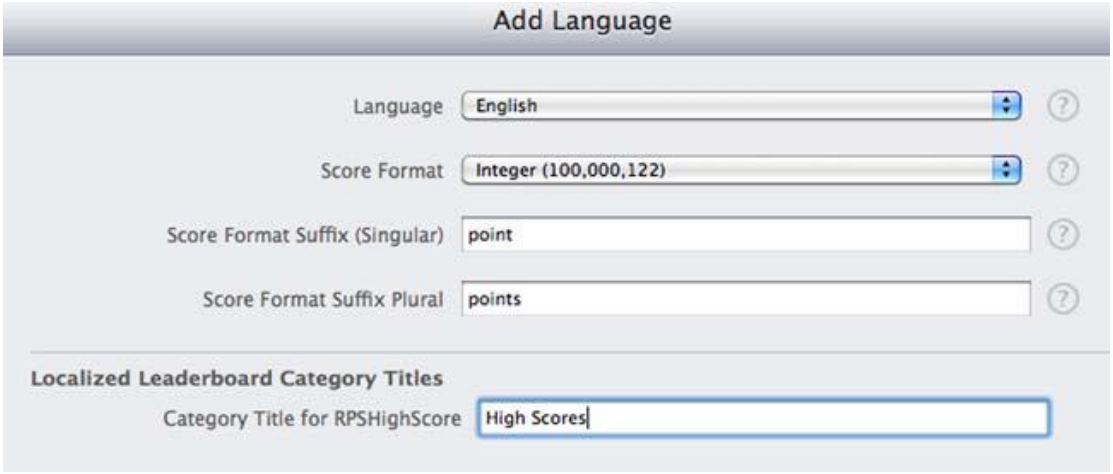


图 9.5 给排行榜增加一种语言然后设置显示数据

现在排行榜上线了，可以报告分数了。在下一个技巧中，你会看到如何让应用上传分数到排行榜。

问题

现在你有一个上线了的排行榜和一个通过验证的用户，你需要一种机制来上传分数。

解决方案

你可以使用一个内置的 GameKit 类和方法来 发送分数到排行榜。如果上传过程出错

了，那么要有一个方法可以过会儿再上传分数。

讨论

和使用 GKPlayer 类验证用户类似，上传分数可以使用 GKScore 类。你可以创建一个方法来接受分数的值。

在 UtilGameCenter 中，创建一个 GKScore 实例，设置分数，然后通过调用方法 reportScoreWithCompletionHandler 告诉排行榜这个分数，见以下代码。

创建一个 GKScore 实例，设置分数的值，然后报告分数

```
-(void)reportScore:(int)score;
{
GKScore *theScore = [[GKScorealloc] init];
theScore.value = score;
[theScorereportScoreWithCompletionHandler:^(NSError *error)
{
if (error)
{
NSData *archivedScore = [NSKeyedArchiver
archivedDataWithRootObject:theScore];
[selfarchiveScoreToSendLater:archivedScore];
        //过会儿再报告分数
}
}];
}
```

根据上面的 reportScore 方法，分数要么显示出来，要么调用 archiveScoreToSendLater，过会儿再发送数据。你需要一个有意义的 archiveScoreToSendLater 方法来储存数据。

在头部声明一个调用 archivedScoresToSendLater 的 NSMutableArray。在 archivedScoresToSendLater 方法中 ,你可以实例化数列(如果需要的话),然后添加分数，过会儿再上传显示。

储存一个让分数过会儿再上传显示的方法

```
-(void)archiveScoreToSendLater:(NSData*)score;
{
if (nil == archivedScoresToSendLater)
archivedScoresToSendLater = [NSMutableArray
arrayWithCapacity:5];
[archivedScoresToSendLateraddObject:score];
}
```

基础的工作做好之后，你需要一个方法开始报告数据。为什么不使用一个按钮呢？以我们的情况，这会是一个非常好的解决方式，因为你可以人工添加一个按钮用来测试和观察。

在 UI 界面上添加一个按钮（比如说 Save 按钮），然后在 RPSViewController 中把按钮和一个 action 连线（例如 doBtnSave：）。这个 action 需要检查你的标识，确定 GameCenter 功能是打开的，然后再调用 reportScore 方法，见以下代码。

UI 的 action 方法来检测 GameCenter 标识，然后报告玩家的分数

```
-(IBAction)doBtnSave:(id)sender;
{
if (gameCenter.gameCenterFeaturesEnabled)
[gameCenterreportScore:gamePoints];
}
```

启动应用，可以看到，你收到了 Welcome Back（欢迎回来）的验证信息，玩一会儿游戏得到一个分数。点击 Save，然后看分数是否显示了，或者有错误。如果有错误的话，应用会给用户显示可能的错误信息，例如说“没有网络连接”。

但是你真正想看到的是什么呢？你希望看到在排行榜上有一个分数，然后分数旁边显示你的名字。这会发生在排行榜显示的时候，让我们现在给你的应用添加显示排行榜的功能。

技巧 68 显示排行榜

GameKit 再一次做好了很多工作。显示排行榜需要应用的框架以及服务器端提供支持。

GKPayer 验证了用户，GKScore 报告分数，GKLeaderboardViewController 可以让排行榜在应用中显示。

问题

给游戏报告了分数，但是现在看不到分数。你需要在应用中显示排行榜。

解决方案

GameKit 又一次来帮忙了，它提供了显示排行榜的解决方法和 UI。你可以创建一个 leaderboard view controller，使用合适的设置来制定排行榜显示。

讨论

作为一个 modal view controller 来显示排行榜，你需要一个 view controller 来调用现有的方法。同时还需要指定显示的 category（见技巧 69）。

注意：leaderboard ID 即是 category 的值： com.brainwashinc.RPS.High-ScoreCategory。

我们可以在 UtilGameCenter 类中再写一个方法加载 view controller 来显示排行榜以及排行榜的 category。然后，它会创建所需的 view controller，指定 category，显示排行榜，见以下代码。

指定 category，显示模态的排行榜

```
-(void)showLeaderboardToVC:(UIViewController*)displayWithVC
forCategory:(NSString*)cat;
{
```

```
GKLeaderboardViewController *vcLeaderboard =
[[GKLeaderboardViewControlleralloc] init];
vcLeaderboard.timeScope = GKLeaderboardTimeScopeAllTime;
vcLeaderboard.leaderboardDelegate = self;
vcLeaderboard.category = cat;
[displayWithVC
presentModalViewController:vcLeaderboardanimated:YES];
}
```

还有关于时间范围的设置：GKLeaderboardTimeScopeToday 以及 GKLeaderboardTimeScopeWeek。当排行榜显示的时候，用户可以选择查看这些时间段内的排行（见图 9.6）。

图 9.6 显示排行榜，排行榜上显示分数和可供用户选择的选项

由于 UtilGameCenter 被设为了 leaderboard view controller 的 delegate，你需要实现这个协议，这需要一个方法：leaderboardViewControllerDidFinish:（见以下代码）。

当完成后需要让 leaderboard view controller 消失。

当完成后 GKLeaderboardViewController 回调方法

```
- (void)leaderboardViewControllerDidFinish:
(GKLeaderboardViewController *)viewController
{
[viewController.parentViewController
dismissModalViewControllerAnimated:YES];
}
```

至于储存分数，你需要有一个 action 来让这个储存的分数在排行榜上显示。使用一个按钮如何呢？在 UI 上增加一个按钮（比如说分数），然后在 RPSViewController 中把这个按钮和一个 action（比如说 doBtnScores:）连线。这个 action 可以检查 GameCenter 是否是打开状态的标识，然后在 UtilGameCenter 中调用 showLeaderboardToVC 方法，见以下代码。

Scores（分数）按钮的 action 检查 GameCenter 标识，显示排行榜

```
- (IBAction)doBtnScores:(id)sender;
{
if (gameCenter.gameCenterFeaturesEnabled)
[gameCentershowLeaderboardToVC:self
forCategory:@"com.brainwashinc.RPS.HighScoreCategory"];
}
```

现在当这个按钮被点击的时候，排行榜就会显示（见图 9.6），你可以在排行榜上看到你的名字以及分数，甚至可以把这个截图和朋友分享。当然，如果你的朋友在 GameCenter 也是你的好友，他们就会见到你的分数。同时他们还会看到你其他游戏的分数以及成就。说到成就...

9.2 GameCenter 成就系统

成就系统是分数的一种延伸，它很有趣。通过给应用提供长期目标可以让玩家的粘性更高。有些成就看起来很傻或是莫名其妙，但就是它们让用户返回游戏。

给你的游戏增加成就很简单，但是它可以提高应用的多次玩的价值，增加吸引力。

在本小节中，你会学到如何在 iTunesConnect 增加成就系统，报告用户一项成就的进程，显示成就面板。

技巧 68 在 iTunesConnect 中增加成就系统

在 iTunesConnect 中增加成就系统和增加排行榜类似，但可能更加简单。过程主要包括设置成就名称，ID，以及成就需要的分数值。这个分数值是评估值的一种基本方法（例如说，完成了 60%）。

问题

你需要在 iTunesConnect 中给应用增加一个成就，这样应用就可以用存储下来用户的分数除上成就的所需分数得到一个百分比。

解决方案

在 iTunesConnect 中创建成就以及必须的值，这样可以在线显示成就的完成度。

讨论

登陆 iTunesConnect ,找到你的应用 ,选择 Manage GameCenter ,在 Achievements（成就）区域点击 Edit。

点击 Add New Achievement(增加新成就),然后可以看到一个设置成就值的面板(见图 9.7)。关于石头，剪子，布的成就，你可以设置 reference name 为 100 分，ID 为 com.brainwashinc.RPS.HundredPoints，point value 为 100。

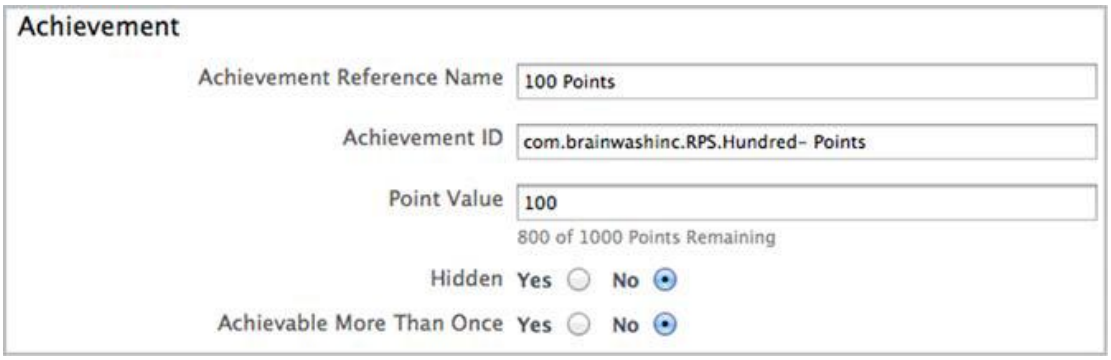


图 9.7 在 iTunesConnect 中设置一个新成就

和配置排行榜一样，可以添加一种语言。点击 Add Language，指定 Title，Pre-earned Description，根据 Pre-earned Description 给用户显示成就的完成度（见图 9.8）。同时还可以上传一副图来显示这个成就。

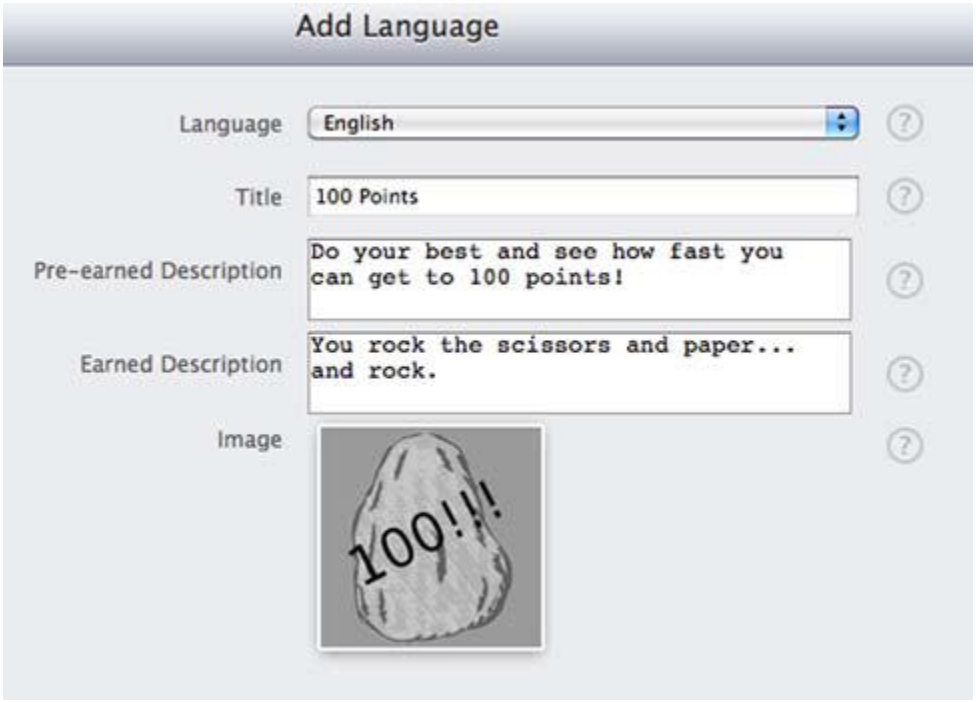


图 9.8 成就的 add language 设定

点击 Save Changes，成就就创建好了。现在要开始准备把成就的完成度显示给用户了。应用需要使用之前的成就 ID（com.brainwashinc.RPS.HundredPoints），然后 以百分比的形式给用户报告成就的完成度。让我们对应用来做一些设置。

技巧 68 报告成就的完成度

再一次，GameKit 可以帮助你。就像 GKPlayer, GKScore, 以及 GKLeaderboardViewController 处理相关的 UI 和功能 猜猜什么可以报告成就的完成度？如果你说是 GKAchievement，那么你猜对了。

问题

你需要给成就设置报告完成度。只要知道 achievement ID，Gamekit 可以帮助你做剩下的工作。

解决方案

可以使用 GKAchievement，以百分比的形式来报告给定 achievement ID 的成就的完成度。如果存在错误的话，你希望把这个进度存储起来，以后可以使用，见技巧 70。

讨论

在 UtilGameCenter 中使用 GKAchievement 类，你可以设置和 iTunesConnect 中相同的 achievement ID（com.brainwashinc.RPS.HundredPoints），以及给定用户的成就完成度，然后报给给服务器。如果有错误的话，则储存改数据，稍后把数据报给服务器，见以下代码。

报告指定成就完成百分比的方法

```
-(void)reportPercentage:(int)percentage
ofAchievement:(NSString*)achievementID;
{
GKAchievement *theAch = [[GKAchievementalloc]
initWithIdentifier:achievementID];
theAch.percentComplete = percentage;
[theAchreportAchievementWithCompletionHandler:^(NSError *error) {
if (error)
{
```

```
NSData *archiveAch = [NSKeyedArchiver
archivedDataWithRootObject:theAch];
[selfarchiveAchievementToSendLater:archiveAch];
}
}];
}
```

如果有错误的话，你可以创建一个数据对象，调用 archiveAchievementToSendLater: 来处理数据。让我们创建这个方法（见以下代码）。把 rchivedAchievementsToSendLater 作为一个 NSMutableArray，在头部进行申明。如果调用这个方法的时候返回值是 nil，则创建这个数据对象，储存数据。

增加成就数据，稍后报告给服务器

```
-(void)archiveAchievementToSendLater:(NSData*)achievement;
{
if (nil == archivedAchievementsToSendLater)
archivedAchievementsToSendLater =
[NSMutableArray arrayWithCapacity:5];
[archivedAchievementsToSendLateraddObject:achievement];
}
```

有没有不增加按钮但是把这些都放到 action 中的方法呢？你已经使用 Save 按钮储存了分数，所以让我们在 RPSViewController 中，把报告成就完成度增加到相同的 action。在 UtilGameCenter 实例中改变 doBtnSave:来调用 reportPercentage 方法（见以下代码）。传递分数以及 achievement ID 到 reportPercentage 方法，剩下的就让 GameKit 来处理吧。

Save 按钮的 action，储存分数以及成就的完成度

```
-(IBAction)doBtnSave:(id)sender;
{
```

```
if (gameCenter.gameCenterFeaturesEnabled)
{
[gameCenterreportScore:gamePoints];
if (gamePoints< 101)
[gameCenterreportPercentage:gamePoints
ofAchievement:@"com.brainwashinc.RPS.HundredPoints"];
}
}
```

当报告成就的完成度的时候，是报告完成的百分数。所以如果达到成就需要的总分数是 100 分，而用户拥有 90 分，那就说这个成就完成了 90%。然而，如果成就的总分要求是 1000 分，而只完成了 90 分，那么就是完成了 9%。请注意在代码中应用不会报告完成超过 100 分（这是为了节省带宽）。然而，这同时也意味着如果分数没有恰好达到 100 分，则不会收到这个成就。

同样的，如果过一段时间你再打开应用，得了 20 分，点击 Save（保存），排行榜分数则会下降到 20 分，成就完成度则也会下降到 20%。

再一次，应用现在会报告成就，但用户看不到。如何显示成就完成度呢？

技巧 68 显示成就榜

和排行榜一样，你也希望在应用中可以看到成就榜。同时也和排行榜一样，UI 是由 Gamekit 处理的。

问题

你需要可以在应用中显示成就榜。

解决方案

可以使用 GameKit 提供的类，以 view controller 的形式来显示成就榜。

讨论

就像在技巧 71 中处理排行榜时一样，可以使用一个 GameKit 提供的 view controller 来显示成就榜。GKAchievementViewController 只需要一个 delegate 调用 finish 回调函数。现在让我们在 UtilGameCenter 类中写一个显示成就方法（见以下代码）。finish 回调函数会让 achievement view controller 消失。

Achievement view controller 和 finish 回调函数让 Achievement view controller 消失

```
-(void)showAchievementsToVC:(UIViewController*)displayWithVC;
{
GKAchievementViewController *vcAchievements =
[[GKAchievementViewControlleralloc] init];
vcAchievements.achievementDelegate = self;
[displayWithVCpresentModalViewController:vcAchievements
animated:YES];
}
- (void)achievementViewControllerDidFinish:
(GKAchievementViewController *)viewController
{
[viewController.parentViewController
dismissModalViewControllerAnimated:YES];
}
```

通过增加一个按钮和增加一个 action 到 RPSViewController，你可以和你的 RPSViewController 调用 showAchievementToVC:方法，显示成就。

成就有趣的一点就是对抗。不管是和自己还是和其他人对抗，都很有趣。但是玩剪刀，石头，布，和其他人对抗更加有意思。在下一节中，你可以知道 GameCenter 怎么样允许玩家之前进行对抗赛。

9.3 通过 GameCenter 匹配玩家

关于 GameCenter，苹果提供了令人惊叹的一系列功能给开发者，移除了联网游戏的壁垒。在此之前，你必须要邀请和同一服务器端的朋友组建联网对战，开发 UI，发送消息功能，等等等等。现在 GameCenter 为你做了所有这些，而且所有的游戏都有一个共同的朋友网络。

和他人一起游戏，GameCenter 允许你匹配随机的玩家，邀请朋友来进行游戏，来回发送消息，甚至是在应用中进行语音交谈。你的应用还可以不是一个游戏。你可以使用 GameCenter 来获取朋友网络，互动，以及进行语音聊天。

匹配玩家允许你的应用找到其他想玩同一个游戏的玩家。使用 GameKit 框架的类，你可以指定匹配玩家的规则，比如说多少玩家参赛。和其他玩家一起玩游戏还可以通过邀请一个特定的玩家。

GameCenter 不仅可以匹配玩家，还可以为玩家在游戏中提供语音聊天的支持。让我们看看如何匹配玩家，增加聊天功能。

技巧 68 匹配玩家

匹配玩家的基础非常简单。GameKit 再次完成了最难的部分。你只需要告诉它你希望玩什么游戏，希望和谁一起玩游戏，匹配的时候显示什么场景，然后处理匹配完成之后的事情就可以了。

问题

你需要使用 Gamekit 找到匹配玩家一起玩石头，剪子，布。

解决方案

使用 GKMatchRequest，你可以指定参数告诉它去寻找匹配玩家。

讨论

UtilGameCenter 类可以再次处理开始过程。传递一个 delegate 开处理各种寻找匹配玩家的回调函数。而这个 delegate 就是 RPSViewController，但是首先让我们看看方法，见以下代码。

GKmatchrequest 开始匹配请求搜索玩家

```
staticNSObject *matchDel;
-(void)fetchMatchWithMatchDelegate:(id)del;
{
GKMatchRequest *req = [[GKMatchRequestalloc]
init];
req.minPlayers = 2;
req.maxPlayers = 2;
        //两个玩家
req.playerGroup = 1;          //配对
matchDel = del;
GKMatchmaker *mmaker = [[GKMatchmakeralloc]
init];
          //储存 delegate
[mmakerfindMatchForRequest:req
withCompletionHandler:^(GKMatch *match,
NSError *error)
{
if (!error)
          //设置 delegate
{
gameMatch = match;
```

```
[gameMatchsetDelegate:(id)matchDel];
}
}];
}
```

而对匹配的请求，你指定需要最少一个最多两个玩家。同时，设置 playerGroup 为 1。playerGroup 的值可以是任意的 int 类型值，它用来限制匹配玩家的数量。你也许希望通过等级、地点或者其他条件来匹配玩家。

会有一个静态的 NSObject（matchDel）传递给 delegate，储存这个 delegate。在头部声明一个 GameMatch 的成员变量，如果没有任何错误的话，设置它为返回的匹配。设置这个匹配的 delegate 为你储存的 matchDel 静态对象。现在无论传递任何 delegate 过来都会收到一个匹配相关的回调函数。让我们看看它。

在 RPSViewController 的头部，你需要为游戏创建一些变量。你需要一个标识（flag），inAGame，用它来获取玩家是否在游戏中的信息。可以使用一个 int 类型的变量（opponentChoice）来获取其他玩家选择了什么的信息，然后为一个 label（lblMessage）创建一个 outlet 来给用户显示信息。

让我们在 doBtnPlay：方法中给新的成员变量增加一些初始化设置，然后在 UtilGameCenter 实例中调用获取匹配方法，见以下代码。

当 Play 按钮被点击的时候初始化游戏相关的成员变量

```
- (IBAction)doBtnPlay:(id)sender;
{
inAGame = NO;
myChoice = -1;
opponentChoice = -1;
if (gameCenter.gameCenterFeaturesEnabled)
[gameCenterfetchMatchWithMatchDelegate:self];
```

```
}
```

如果用户通过了验证，抓取这个匹配用户，传入它作为 delegate。现在你需要处理 GKMatchDelegate 方法。

当找到一个匹配用户的时候回调函数传入匹配实例：playID 以及匹配用户的联网状态（如果处于非联网状态的时候仍然发送信息。

你想要保存这个匹配用户，使用它来使设备之间进行交流（见以下代码）。如果联网断了，需要通知用户。

处理匹配回调，得到 playerID 和联网状态

```
(void)match:(GKMatch *)match
player:(NSString *)playerID
didChangeState:(GKPlayerConnectionState)state
{
if (state == GKPlayerStateConnected)
{
inAGame = YES;
gameMatch = match;
        //储存 GKMatch
}
else if (state == GKPlayerStateDisconnected)
{
UIAlertView *alert = [[UIAlertViewalloc]
initWithTitle:@"Disconnected"
message:@"The other player disconnected."
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
```

```
}
}
```

现在你有了一个随机匹配的对手。这个匹配的另一个回调是要得到对手的动作。信息发送的回调是通过 GKmatch 类发送匹配，一个 NSData 实例以及 playerID。对你的游戏而言，已经有了匹配了，所以可以忽略它。由于现在只有一个其他玩家，所以 playerID 现在不是很重要。但是数据类会包含他们选择剪刀，石头，布的信息。你可以看到当用户之后做了选择后数据时如何传递的。首先需要储存玩家的选择，见下列代码。

处理 GameCenter 玩家匹配

```
- (void)match:(GKMatch *)match
didReceiveData:(NSData *)data
fromPlayer:(NSString *)playerID
{
NSString *otherPlayersChoice =
[[NSStringalloc] initWithData:data
encoding:NSUTF8StringEncoding];
opponentChoice = [otherPlayersChoiceintValue];
                    //储存玩家选择
if (myChoice != -1)
                    //检查我的选择

{
[lblMessagesetText:@"Player selected."];
[selfprocessOtherPlayersChoice:opponentChoice];
}
else
```

```
[lblMessagesetText:@"Other player selected. Waiting for you..."];
}
```

储存传递给你的用户选择，然后如果你做了选择的话（myChoice != -1），就处理其他玩家的选择。这里假定两件事——通过匹配对手发送的数据是包含他们选择内容的字符串，当你做出你的选择时，它会储存在 myChoice 里。doBtnPaper:, doBtnRock:, 以及 doBtnScissors:方法需要处理这两个数据，见以下代码。

加工选择的方法储存选择，然后等待，或者发送数据进行处理

```
- (IBAction)doBtnRock:(id)sender;
{
myChoice = ROCK;
if (opponentChoice == -1)
[selfwaitForOtherPlayer];
else
{
[selfsendMyChoice];
[selfprocessOtherPlayersChoice:opponentChoice];
}
}
```

储存这个选择，如果 opponentChoice 没做选择(== -1)，则等待其他玩家（调用 waitForOtherPlayer）。如果其他玩家已经做出了选择，则发送你的选择，然后处理这两个数据。

等待其他玩家的时候，可以发送你的选择，如果你现在不在游戏中了，可以告诉另一个用户这个消息，见以下代码。

waitForOtherPlayer 更新，发送选择

```
-(void)waitForOtherPlayer;
{
```

```
if (inAGame)
{
[lblMessagesetText:@"Waiting for other player..."];
[selfsendMyChoice];
}
else
{
UIAlertView *alert = [[UIAlertViewalloc]
initWithTitle:@"Not Playing"
message:@"Please press play to start a game."
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
}
}
```

或者，你也可以发送你的选择给其他用户。接受数据的方法可以储存数据，只有当你已经做出选择的时候才处理数据，所有可以发送多次。

那么 sendMyChoice 做什么呢？怎么发送选择，你把游戏的选择转化为了一个 NSString，然后再转化为 NSData，把它传递给 GameMatch 实例方法，见以下代码。

把选择转化为数据，然后通过 GameMatch 发送

```
-(void)sendMyChoice;
{
NSString *choice = [NSStringstringWithFormat:
@"%d", myChoice];
NSData *data = [choice dataUsingEncoding:
NSUTF8StringEncoding];
NSError *error;
```

```
[gameMatchsendDataToAllPlayers:data
withDataMode:GKMatchSendDataReliable error:&error];
}
```

以上的这些是从一个用户的角度来考虑的，但是两个用户我们都可以这样处理。如果你做了一个选择，你把它发送给另一个用户。这个用户的设备收到了你的选择数据，储存起来，等到这个用户也做了选择的时候，就处理两人的数据来看谁是胜利者，奖励分数等等。

如果另一个用户收到了你的选择，但是他还没有做出选择话，游戏就等待他做出选择。当另一个用户做出选择了，游戏就有两人的选择数据了，可以对它们进行处理。同时，其他用户发送给你他们的选择，而你有自己的选择，游戏可以在你的设备上处理数据。

由于匹配可能是随机的也可能是定向的（playGroup），你可以看看如何把用户连接在一起比较简单。然后，发送数据也会比较简单，把内容转化为 NSData 传递。这对对战类游戏的应用非常明显，但是这种方式对所有应用都适用。

和一个随机的玩家一起游戏很好，但是如果想和你的一个朋友一起玩呢？让我们看看如何通过 GameCenter 邀请特定的好友来进行游戏。

技巧 68　邀请朋友

在 GameCenter 有一个内置的朋友网络，你希望可以邀请这些朋友来进行游戏。GameCenter 和 GameKit 帮你处理了很多 UI 和功能上的东西。你可以邀请朋友，启动 qq，然后进行游戏。

问题

邀请朋友玩游戏有两个方面。你希望可以通过 GameCenter 邀请朋友。但是邀请成功之后还要处理玩家匹配。

解决方式

GameCenter 自动处理了这部分。它知道什么应用激活了 GameCenter 功能，可以发

送邀请给你的朋友。

应用给朋友发送邀请，还需要使用一些 GameKit 类。还有相关的 delegate 回调方法来处理这个过程。你还需要输入邀请者的 inviteHandler 代码。

讨论

启动 GameCenter 之后，可以选择一个游戏，选择一个朋友，输入一个邀请信息或使用默认的邀请信息，然后发送（见图 9.9）。

图 9.9 GameCenter 发送邀请

当收到来自朋友的邀请之后，你还需要在 GameCenter 中查看邀请信息（见图 9.10）

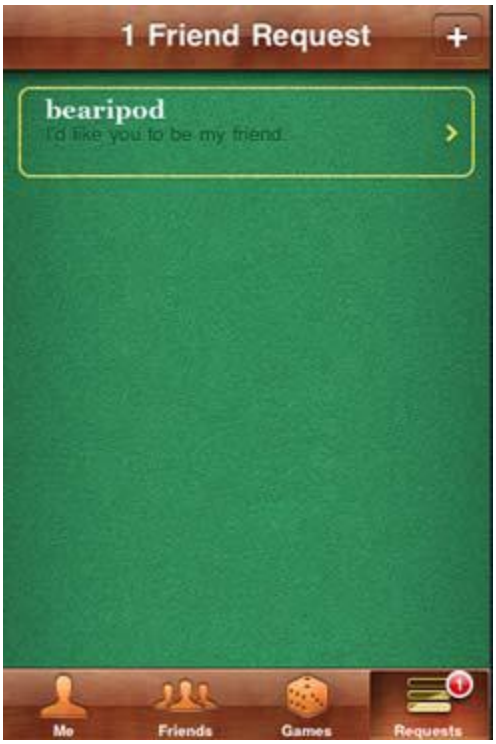


图 9.10 GameCenter 请求列表

点击一个请求会弹出提示选项，接受，忽略，或是报道此请求有问题（见图 9.11）。

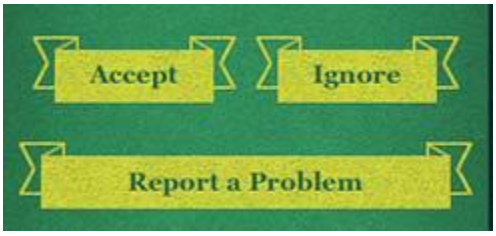


图 9.11 GameCenter 邀请的选项

要使用应用中的相关 UI，你需要使用一个 GameKit 类叫做 GKMatchmakerViewController。邀请配对的启动和随机匹配类似，你可以设置玩家的最低人数和最高人数，以及 GKMatchRequest 上的 playergroup。然后传递匹配请求给 GKMatchmakerViewController，设置 delegate，然后显示 view controller。

你可以通过接受 matchmakerviewcontroller 的一个 delegate 来在 UtilGameCenter 中创建一个方法来完成这个工作（配置游戏指定的值）。见以下代码。

根据请求显示 GameCenter 的 matchmaker UI

```
-(void)inviteMatchWithMatchDelegate:(id)del;
{
GKMatchRequest *req = [[GKMatchRequestalloc]
init];
req.minPlayers = 2;
req.maxPlayers = 2;
req.playerGroup = 1;
GKMatchmakerViewController *vcMMaker =
[[GKMatchmakerViewControlleralloc]
initWithMatchRequest:req];
[vcMMakersetMatchmakerDelegate:del];
[delpresentModalViewController:vcMMakeranimated:YES];
}
```

过一会儿你需要给 matchmaker view controller 实现 delegate 方法。但是现在，让我们来看看如何让 matchmaker 启动。

现在可以让 Play 按钮处显示一个提示视图，让用户选择是邀请朋友还是匹配随机玩家（见图 9.12 ）。

图 9.12 提示用户是邀请朋友还是匹配随机对手

在提示视图的回调函数中，可以调用一个新的方法 inviteFriend，如果用选择了邀请朋友就调用这个新方法，如果他们选择了随机匹配，则调用另一个新方法 autoMatch。

AutoMatch 方法可以是以前的 doBtnPlay:方法，只是对它进行重命名，见以下代码。

询问用户他们希望和谁一起进行游戏，是邀请朋友还是随机匹配

```
- (IBAction)doBtnPlay:(id)sender;
{
UIAlertView *av = [[UIAlertViewalloc]
initWithTitle:@"Play"
message:@"Would you like to invite a friend or...
play a random player?"
delegate:self
cancelButtonTitle:@"Cancel"
otherButtonTitles:@"Invite", @"Random", nil];
[av show];
}
```

```
- (void)alertView:(UIAlertView *)alertView
clickedButtonAtIndex:(NSInteger)buttonIndex
{
if (buttonIndex == 1)
[selfinviteFriend];
else if (buttonIndex == 2)
[selfautoMatch];
}
```

这个新的 inviteFriend 方法和 autoMatch 方法（重命名的 doBtnPlay:方法）类似。它会初始化游戏的变量，检查 GameCenter 功能是否激活的标识，如果设备没有激活 GameCenter 功能则告诉用户他们没有通过验证，这个方法还会调用 UtilGameCenter 中的 matchmaker 相关的方法，和随机匹配的方法相对，见以下代码。

开始创建一个 GameCenter 玩家配对

```
-(void)inviteFriend;
{
inAGame = NO;
myChoice = -1;
opponentChoice = -1;
if (gameCenter.gameCenterFeaturesEnabled)
{
[lblMessagesetText:@"Inviting friend..."];
[gameCenterinviteMatchWithMatchDelegate:self];
                    //matchmaker 方法
}
else
{
UIAlertView *alert = [[UIAlertViewalloc]
```

```
initWithTitle:@"Authenticate"
message:@"You have not been authenticated into GameCenter yet."
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
}
}
```

Matchmaker 的 UI 和 GameCenter 的 UI 类似，有一个 Invite Friend 的按钮（见图 9.13）。

图 9.13 Matchmaker 邀请朋友的 UI

你传递了 self 参数给 matchmaker 方法，所以需要实现相关的回调。现在有好几个回调函数，有一些只有在邀请取消或是失败的时候才调用，然后你可以以让 matchmaker view controller 消失（见以下代码）。仍然还有一些必须的方法需要实现，即使这些方法现在是空的。

处理 GameCenter matchmaker delegate 调用

```
- (void)matchmakerViewControllerWasCancelled:
(GKMatchmakerViewController *)viewController
{
[selfdismissModalViewControllerAnimated:YES];
[lblMessagesetText:@"Nevermind."];
}
- (void)matchmakerViewController:
```

```
(GKMatchmakerViewController *)viewController
didFailWithError:(NSError *)error
{
[selfdismissModalViewControllerAnimated:YES];
[lblMessagesetText:@"Invite failed."];
}
```

有一个有意思的回调是在找到匹配玩家的时候调用的（比如说，朋友接受邀请了）。以你的情况，可以设置 inAGame 标识为 Yes，然后让 matchmaker view controller 消失，见以下代码。

接受邀请了/找到匹配玩家了——设置标识，让 matchmaker view controller 消失

```
- (void)matchmakerViewController:
(GKMatchmakerViewController *)viewControllerdidFindMatch:
(GKMatch *)match
{
inAGame = YES;
gameMatch = match;
[lblMessagesetText:@"Match Found. Play!"];
[selfdismissModalViewControllerAnimated:YES];
}
```

最后，你需要增加下列 block 来处理游戏请求。

在 viewDidLoad 中用来处理邀请的 block

```
[GKMatchmakersharedMatchmaker].inviteHandler =
^(GKInvite *invite, NSArray *playersToInvite) {
GKMatchmakerViewController *controller =
[[GKMatchmakerViewControlleralloc]
initWithInvite:invite];
controller.delegate = self;
```

```
[selfpresentModalViewController:controller
animated:YES];          //显示 matchmaker
```

尽管通过 GameCenter 和 GameKit 邀请其他用户一起玩游戏有很多的步骤 ,但是我希望你能够赞同那并不复杂。如果你有一个游戏的话 ,我建议你接入 GameCenter 的排行榜，成就，以及朋友邀请，语言聊天等内容。让我们现在给你的应用增加语言聊天的功能。

技巧 68 通过 GameCenter 进行语言聊天

当玩家匹配成功之后，GameCenter 允许进行语音聊天。语言聊天很容易建立，只需要少数几行代码，它只需要一个 GameMatch 实例。这就意味着你必须和至少一位玩家建立网络连接，如果无网络连接的话，很明显的，聊天不能进行。

问题

需要使用 GameCenter 给应用增加语言聊天功能。

解决方案

两个玩家之间进行了匹配后，可以开始聊天会话了。

讨论

可以在界面上建立一个 Chat 按钮，把在这个按钮和 RPSviewController 中的 doBtnChat :方法连线起来。它可以检查来确保你有一个 GameMatch 实例 ,如果确实有，可以用一个指定的语言通道开始聊天。

一个匹配可以有多个语言通道，而玩家可以同时加入多个语言通道。

开始一个聊天，需要确保你有一个 GameMatch 实例（如果没有的话要提示用户），然后使用 voiceChatWithName:方法来创建聊天，传入语言通道的名称。存储之后返回给 GKVoiceChat 一个成员变量，见以下代码。

如果有 GameMatch 实例的话则创建一个叫做 RPS 的语言聊天通道

```
-(IBAction)doBtnChat:(id)sender;
{
if (nil == gameMatch)
{
UIAlertView *alert = [[UIAlertViewalloc]
initWithTitle:@"No Game"
message:@"You are not currently in a game."
delegate:self
cancelButtonTitle:@"OK"
otherButtonTitles: nil];
[alert show];
inAGame = NO;
return;
}
chat = [gameMatchvoiceChatWithName:@"RPS"];
[chat start];
}
```

正如你所见，给你的应用增加语言聊天功能只需要为数不多的代码，但是他可以极大的丰富你的应用。

9.4 总结

GameCenter 提供了很多功能，只需要付出一点努力就可以获得很大的回报。他可以为你的应用剪刀，石头，布提供值得信赖的配对以及游戏功能。同时，还可以增加报告分数的功能，查看游戏排行榜。如果没有苹果提供的成员网络的整合工作，很多功能无法完成。

有了 GameCenter 提供的功能，很多类型的应用（尤其是游戏）可以通过允许用户和他们的朋友一起参与，以一种新的方式来体验应用，在应用中争取更多的成就和更高的分数

让应用提升一个高度。

10

——通过教你制作一个上架应用 MusicSearch 来学习 iTunes API，iPad 适配，以及 iAd

本章包括：

- 连接到 iTunes
- 把你的应用适配 iPad
- 给应用添加广告

大体来说，整合各种系统似乎还是在科技领域有待成长的一部分，特别对手机世界来说是这样。越来越多的应用允许你在 twitter 发微博，Facebook 应用可以同步联系人，ShareKit 提供了多种在线选择整合服务。

还有一个可以整合和优质应用就是 iTunes。iTunes API 是一栋的 JSON（JavaScript Object Notation）文本，可以解析清除有意思的数据。

还有两个发展成长的领域是 iPad 应用和 iAds。它们发布时宣传造势足，现在也在持续成长。iPad 设备数量有很多，而我认为广告以后也不会消失。

在本章中我们会一起了解 iTunes API，iPad/iPod,以及 iAds，学习如何将他们整合到应用中去。你可以创造一个应用可以通过 iTunes API 来搜索音乐，显示结果，试听音乐。然后可以将应用转为通用二进制代码（Universal binary），适配 iPhone/iPod 以及 iPad。然后给应用添加 iAds，让应用可以展示苹果相关的内容。

10.1 使用 iTunes API 搜索音乐

在线搜索 iTunes API 可以得到很多的搜索结果。让我们找到真正的来自苹果的链接，然后可以看到很多关于 API 的信息。

我们将会创建一个应用——MusicSearch，这个应用可以允许用户搜索歌手，歌曲，专辑等，然后在一个表中显示包括曲名，专辑名，专辑封面的信息（见图 10.1）。

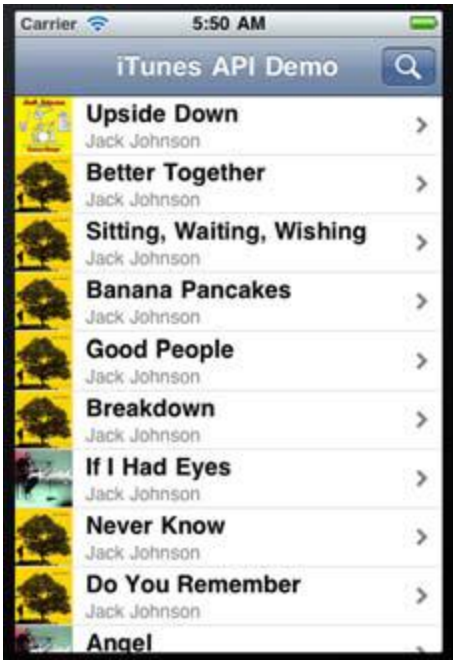


图 10.1 搜索 Jack Jason 的结果

点击歌曲可以让用户该音乐，加载 iTunes 播放，查看详细信息（比如说放大版的专辑封面）。

技巧 78 询问 iTunes API

对外发布的 iTunes 搜索 API 可以通过一个简单的 URL 来进行搜索。结果会以 JSON 形式返回，解析方便。我们可以专注于搜索用户输入文本相关的音乐。

问题

我们需要根据用户输入的文本内容来搜索 iTunes 的在线数据。

解决方案

使用 iTunes API，可以搜索影月的相关信息，包括文本信息，视觉信息（图片），以及

音频信息（试听音乐）。

讨论

在线定义 API 很容易识别。URL 地址是：

http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/wa/wsSearch。

如果增加参数?term=<text>&entity=<entity>，参数中的 text 是搜索输入的文本，entity 的媒体类型（比如说 tvshow（电视剧）或者 movie（电影）,我们可以查看详细信息。可以使用 musicTrack，让读者输入要搜索的文本。

首先，需要创建一个新的基于视图的工程，命名为 MusicSearch。在 UI 编辑器中，把默认的 view 替换成一个 table view。在 table view 中显示搜索结果。你需要在代码中把 RootViewController 换成一个 UITableViewController，然后连线这个 table view。

下一步，需要用 IB 制作一个简单的搜索输入框的视图控制器（search input view controller）。（见图 10.2）

图 10.2 搜索输入框的视图控制器

RootViewController 需要一个 UIViewController 的 outlet ,还有一个 Cancel 按钮的 action ,例如(doBtnCancel:)同时 ,给 RootViewController 一个 UITextFieldDelegate。连线 RootViewController 和搜索视图控制器，把 RootViewController 作为输入框的 delegate。

在 RootViewController 的 viewDidLoad 方法中，你可以添加一个 Search 按钮，然后设置 navigation 标题（见以下代码）。

设置标题，给 navigation item 创建一个 Search 按钮

```
- (void)viewDidLoad {
[superviewDidLoad];
self.title = @"iTunes API Demo";
UIBarButtonItem *bbi = [[UIBarButtonItemalloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemSearch
target:self
action:@selector(doBtnSearch:)];
[self.navigationItemsetRightBarButtonItem:bbi];
}
```

doBtnSearch:方法需要显示你的搜索视图控制器（见以下代码）。

Search 按钮被点击时显示 search view controller

```
-(void)doBtnSearch:(id)sender;
{
[selfpresentModalViewController:vcSearchanimated:YES];
}
```

由于 RootViewController 是搜索文本框的 delegate，你可以在 textFieldShouldReturn 被调用的时候（当用户点击 Return 键的时候在输入框中进行编辑）进行搜索。你需要一个地方来储存结果，所以来声明一个 NSArray，在头部把它命名为

results，然后在回调方法中使用它（见以下代码）。

用输入的文本来询问 iTunes，然后检索结果

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
[selfdismissModalViewControllerAnimated:YES];
[textFieldresignFirstResponder];
NSString *searchText = textField.text;
searchText = [searchTextstringByReplacingOccurrencesOfString:@" "
withString:@"+"];
NSString *url =
[NSStringstringWithFormat:@"http://ax.itunes.apple.com/WebObjects/...
MZStoreServices.woa/wa/wsSearch?term=%@&entity=musicTrack",
searchText];
NSError *error;
NSString *search = [NSStringstringWithContentsOfURL:
[NSURL URLWithString:url]
encoding:NSUTF8StringEncoding error:&error];
return NO;
}
```

结果的例子如以下代码所示。

从 iTunes API 搜索返回的部分结果例子

```
{
"resultCount":50,
"results": [
{"wrapperType":"track", "kind":"song", "artistId":909253,
"collectionId":120954021, "trackId":120954025,
"artistName":"Jack Johnson", "collectionName":"Sing-a-Longs and
Lullabies for the Film Curious George", "trackName":"Upside Down",
```

"collectionCensoredName":"Sing-a-Longs and Lullabies for the Film Curious George", "trackCensoredName":"Upside Down", "artistViewUrl": "http://itunes.apple.com/us/artist/jack-johnson/id909253?uo=4",...

结果是以一种叫 JSON 的形式显示的，它易读而且容易解析成可用的形式，例如说数组或是 Dictionary。让我们看看如何处理从 iTunes 返回的 JSON 数据。

技巧 78 显示 JSON 结果

你需要把 JSON 数据解析出来，让它在代码中可用。这儿有一个非常棒的 JSON 框架：https://github.com/stig/json-framework/ 。把它添加到工程很简单，它可以解析 NSArrays 和 NSDictionary 数据，也可以把代码生成 NSArrays 和 NSDictionary 数据。

问题

你需要解析 JSON 使他成为 dictionary 数列，一列是一个查询结果，可以把这种数据显示给用户看。

解决方案

可以使用 JSON 框架来解析和返回数据。Dictionary 数据包含了以各种 item 的 key 或是 value，比如说 artistName item, trackNameitem,previewUrl item, 以及 rtwork item。

讨论

把框架添加到工程之后，就可以管理从 iTunes 返回的数据了。你可以在收到回复之后添加代码来消除新行，把 JSON 数据解析成 ditionary（见以下代码）。

准备并且把解析的 JSON 数据加到结果成员变量数列中

```
search = [search stringByReplacingOccurrencesOfString:
@"\\n" withString:@""];
NSDictionary *dict = [search JSONValue];
results = [dictobjectForKey:@"results"];
```

```
[self.tableViewreloadData];
```

现在你的数据是以数列的形式显示的，你知道查询返回的数据量。有了数据量，就可以指定在 table 中有多少行（见以下代码）。

返回的搜索结果行的数量

```
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section {
return [results count];
}
```

每个结果数列的 item 都是行的内容。你可以在顶部显示 trackName（曲名），在底部显示 artistName（艺术家名），在左边显示图片。如果想要作为一个附件，指定一个展开指示器(Disclosure indicator)，让用户知道他们可以通过点击获得更多信息(见以下代码)。

基于询问结果设置 table cell 中的数据

```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath {
staticNSString *CellIdentifier = @"Cell";
UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
if (cell == nil) {
cell = [[UITableViewCellalloc]
initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:CellIdentifier];
cell.accessoryType =
UITableViewCellAccessoryDisclosureIndicator;
}
NSDictionary *trackDict = [results objectAtIndex:indexPath.row];
cell.textLabel.text =
[trackDictobjectForKey:@"trackName"];
```

```
cell.detailTextLabel.text =
[trackDictobjectForKey:@"artistName"];
NSString *imageURL =
[trackDictobjectForKey:@"artworkUrl60"];
cell.imageView.image = [UIImageimageWithData:
[NSDatadataWithContentsOfURL:[NSURL URLWithString:imageURL]]];
return cell;
}
```

运行应用，搜索艺术家或是专辑，查看结果（见图 10.1）。现在我们在 table 中显示了结果，但是我们还想要在点击一个指定的行的时候发生一些更令人值得期待的有意义的东西。让我们看看在搜索结果中使用 previewUrl 来播放 iTunes 的歌曲进行试听。

技巧 78 播放歌曲试听

URL 可用的时候有几种方式可以播放歌曲，最简单的方法是使用分享的组件来播放 URL。另一种方法是在应用内部的 web view 中加载 URL。最利落的方法是下载歌曲，在内置的媒体播放器中播放。让我们来看看这几种技巧。

问题

给搜索结果提供一个 previewUrl，你需要播放歌曲的试听文件。让我们尝试两种解决方式——播放 URL，下载音频文件。

解决方案

UIApplication 类有一个方法可以播放一个给定的 URL。它使用的 OS 来确定使用什么应用，如何更好的处理数据。这在你不知道链接类型的时候特别有用。

我们会研究这个方法，因为是在处理音频数据，所以需要研究下载文件的方法，让它在本地播放。

讨论

当用户点击一行的时候，可以根据他们点击的行的数字来得到 dictionary 数据。然后，可以得到一个给 previewUrl key 提供的值。简单的调用一个在 UIApplication 实例的 openURL 会启动这个 URL，播放音频（见以下代码）。

基于行指数，给给定的数据 dictionary 加载 previewUrl

```
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
NSDictionary *dict = [results objectAtIndex:indexPath.row];
[[UIApplicationsharedApplication]
openURL:[NSURL URLWithString:[dict
objectForKey:@"previewUrl"]]];
                        //打开previewUrl
}
```

还有一个更好的方法，特别是知道你处理什么样的数据形式的情况下，那就是是把数据下载，在本地播放。首先，汇总服务器里抓取的数据，然后转化为一个 NSdata 对象。然后，可以把这个文件储存到本地，使用 AVFoundation 框架提供的内置音频播放器播放（要确定你把 AVFoundation 框架添加到了工程）。

还需要一个方法来得到储存文件的路径，以及一个单独的方法来播放文件。首先，让我们专注于从服务器获取数据（见以下代码）。

给出试听的 URL，抓取音频文件，在本地储存，然后播放

```
NSData *fileData = [NSDatadataWithContentsOfURL:
[NSURL URLWithString:
[dictobjectForKey:@"previewUrl"]]];
NSString *path = [NSString
stringWithFormat:@"%@/soundTrack.mp4",
[selfgetFilePath]];
[fileDatawriteToFile:pathatomically:YES];
```

[selfplaySound:path];

这个逻辑非常清楚——抓取数据，储存数据，播放数据。抓取数据由调用 dataWithContentsOfURL 的 NSData 处理。那么如何得到储存文件的路径呢？因为你只需要暂时的使用他们，所以可以给所有的文件都使用同样的名字：soundTrack.mp4 。

基于文件的目录和执行文件的名称创建一个存储目录

```
-(NSString*)getFilePath;
{
NSArray *paths = NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectoryPath = [paths objectAtIndex:0];
NSString *exeName = [[NSBundlemainBundle]
objectForInfoDictionaryKey:@"CFBundleName"];
NSString *path = [documentsDirectoryPath
stringByAppendingPathComponent:exeName];
NSError *error;
[[NSFileManagerdefaultManager]
createDirectoryAtPath:path
withIntermediateDirectories:YES
attributes:nil
error:&error];
return path;
}
```

最后，可以播放文件。通过 AVFoundation.框架，

在头部声明一个 AVAudioPlayer。然后，在 playSound 方法中，你可以对它进行初始化，使用从储存路径获得的数据，播放歌曲（见以下代码）。

使用音频文件的路径创建一个AVAudioPlayer，然后播放

```
-(void)playSound:(NSString*)aFilePath
```

```
{
if (nil != aAudioPlayer)
{
[aAudioPlayer stop];
        //停止播放
}
aAudioPlayer =[[AVAudioPlayeralloc]
initWithContentsOfURL:
[NSURL fileURLWithPath:aFilePath] error:NULL];
[aAudioPlayer stop];
[aAudioPlayer play];
}
```

在搜索结果中会有很多其他的数据和 URL。其中一个 URL 是 trackViewUrl，它会加载一个歌曲数据的视图。还有一些会把 URL 中的 http 替换成 itms，然后在 iTunes 中加载，用户可以通过它来购买歌曲（见以下代码）。

加载 trackViewUrl 到 iTunes

```
NSString *url = [dictobjectForKey:@"trackViewUrl"];
url = [urlstringByReplacingOccurrencesOfString:@"http"
withString:@"itms"];
[[UIApplicationsharedApplication]
openURL:[NSURL URLWithString:url]];
```

如果从控制器中注销这些数据，你可以查询一些其他想在应用中使用的 item。当然，这仅仅只是音乐搜索。但是你需要开发一些搜索其他类型媒体的东西。

像其他 iPhone 应用一样，这个应用会适配 iPad，使用一套为 iPad 设计的 UI。通过创建一个通用二进制（universal binary）,工程可以在 iPhone 和 iPad 上运行，显示合适的 UI。让我们看看如何适配 iPad。

10.2 应用适配 iPad

尽管人们第一个想到的还是 iPhone，iPad 也很流行，而且，在某些方面 ，iPad 使用起来比 iPhone 更有效率。就算其他方面的东西不算，iPad 有一个更大的显示屏，视觉媒体方面可以做的更好。

把为 iPhone/iPod 开发的应用搬到 iPad 上 ,或是吧一个用通用二进制编码的可以适配所有设备的应用适配 iPad 非常常见。你也许想要制作一个适配所有设备的应用，但是最好还是先开发 iPhone 版本，再添加 iPad 方面的东西。

在本小节，你可以把 MusicSearch 应用转变成一个通用二进制型的应用。你将使用 Xcode 的内置功能来转化工程。Xcode 有把 UI 适配 iPad 的功能。Xcode 可以为你适配 iPad，把适配工作作为工程升级的一部分，同时也可以单独为 iPad 开发。

你可以使用一个 iPad 特定的控制器叫做 split view controller。这个控制器在系统的邮件应用和系统设置应用中使用（见图 10.3 ）。

图 10.3 在 iPad 上使用 split view controller 的系统设置应用

通常来说，split view controller 有两个内置的 view controllers。左边的是 masterview controllers，右边的是 detailview controllers。在 masterview controllers 中，显示用户导航，内容则在 detailview controllers 中显示。

让我们先从创建一个通用二进制应用开始。

技巧 78 在 Xcode 中转换工程

Xcode 有把存在的应用升级适配 iPad 的功能。我们可以通过升级开始，然后指定需要的代码，开始 UI 的编辑。

问题

你需要吧 iPhone/iPod 应用转化为通用二进制，使得应用可以在所以的 iOS 设备上使用。

解决方案

首先可以通过使用内置的 Xcode 功能来升级工程，然后相关 xib 文件。这样技术上就

会帮你实现你需要的效果。

讨论

选择想要升级的目标。在 Summary 标签下 ,选择 Universal forDevices(见图 10.4)。

然后会出现一个提示框，询问您是否想要将 UI 适配 iPad，选择 YES。

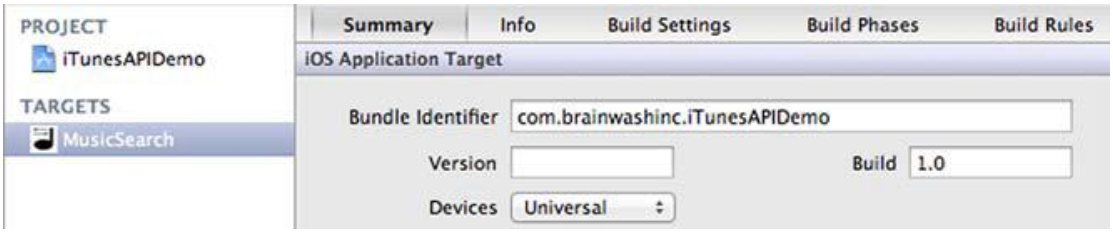


图 10.4 Xcode 的内置功能，可将 iPhone 应用适配 iPad

现在需要获取 iPad 特定的设置，如方向，应用 icon，启动画面等等（见图 10.5)。

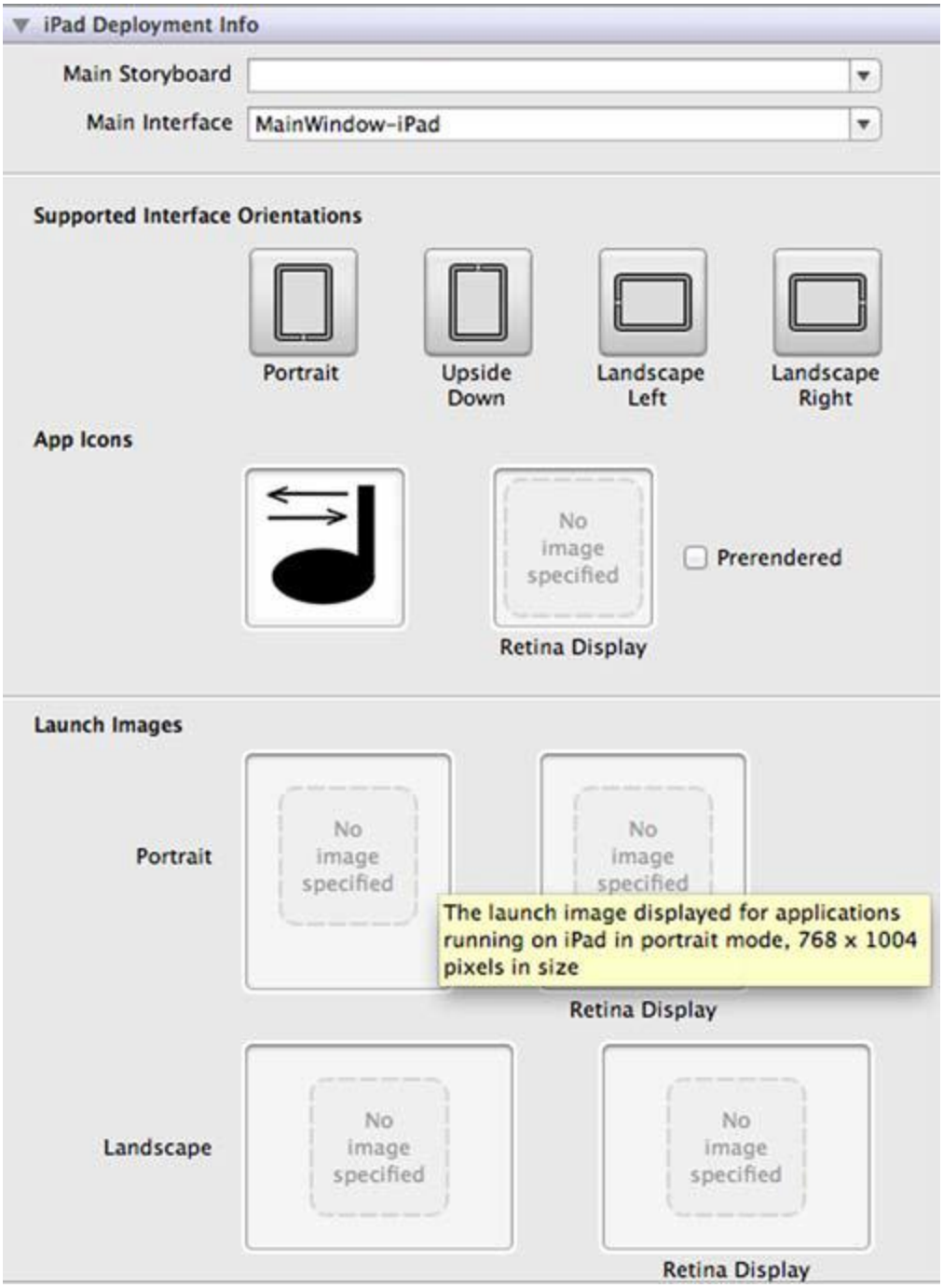


图 10.5 Xcode 允许设置 iPad 专用的图片

作为升级的一部分，Xcode 创建了一个新的资源组叫做 Resources-iPad。在这个新组中有一个 MainWindow.xib 文件的升级版本，叫做 MainWindow-iPad .xib（见图 10.6）。

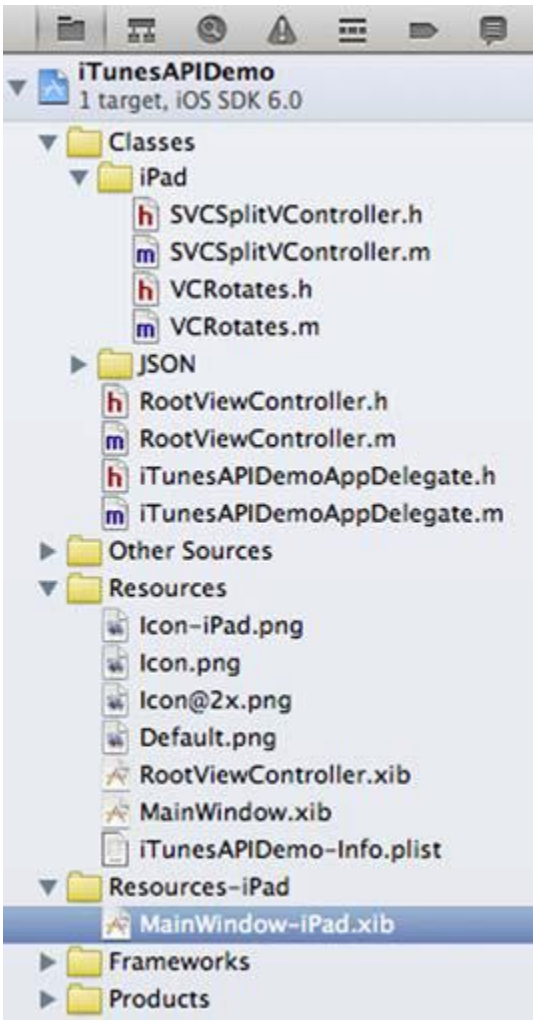


图 10.6 Xcode 创建一个新的资源组，里面有一个升级的 MainWindow-iPad.xib 文件

你可以在 iPad 模拟器中运行应用了，在运行应用之前确定选择的是 iPad 模拟器（见图 10.7）。

图 10.7 Xcode 现在有一个运行应用的选项是使用 iPad 模拟器

可以注意到现在运行应用显示了一个空白的表单（见图 10.8）。

图 10.8 在 iPad 模拟器中运行升级了通用二进制的应用

旋转模拟器到水平方向没有任何变化（见图 10.9）。

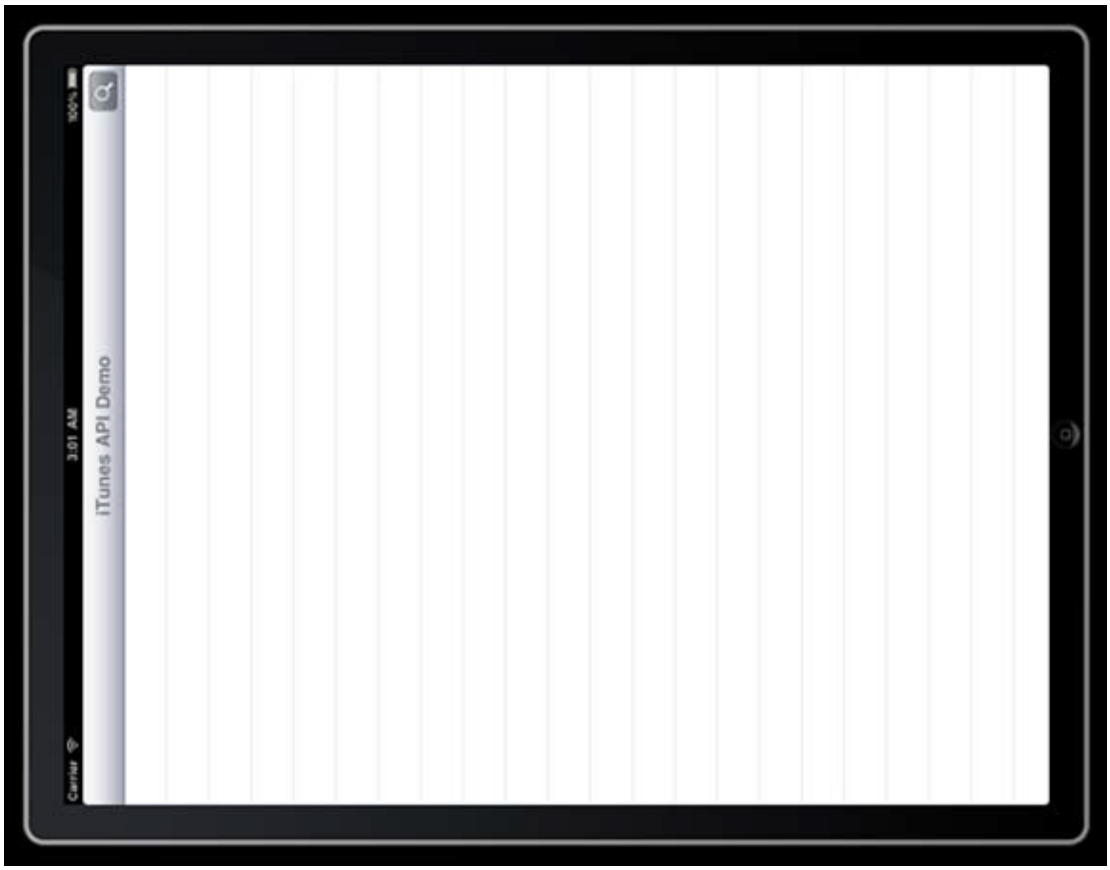


图 10.9 iPad 水平放置后应用在 iPad 模拟器中的运行情况

给 RootViewController 增加一个方法可以让应用跟随 iPad 旋转到水平方向（见以下代码）。

通过回调来确定 view controller 是否需要旋转

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation {
return YES;
}
```

去查看其它的应用，你可以看到有同样的屏幕情况。搜索工具栏显示不正常（见图 10.10）。

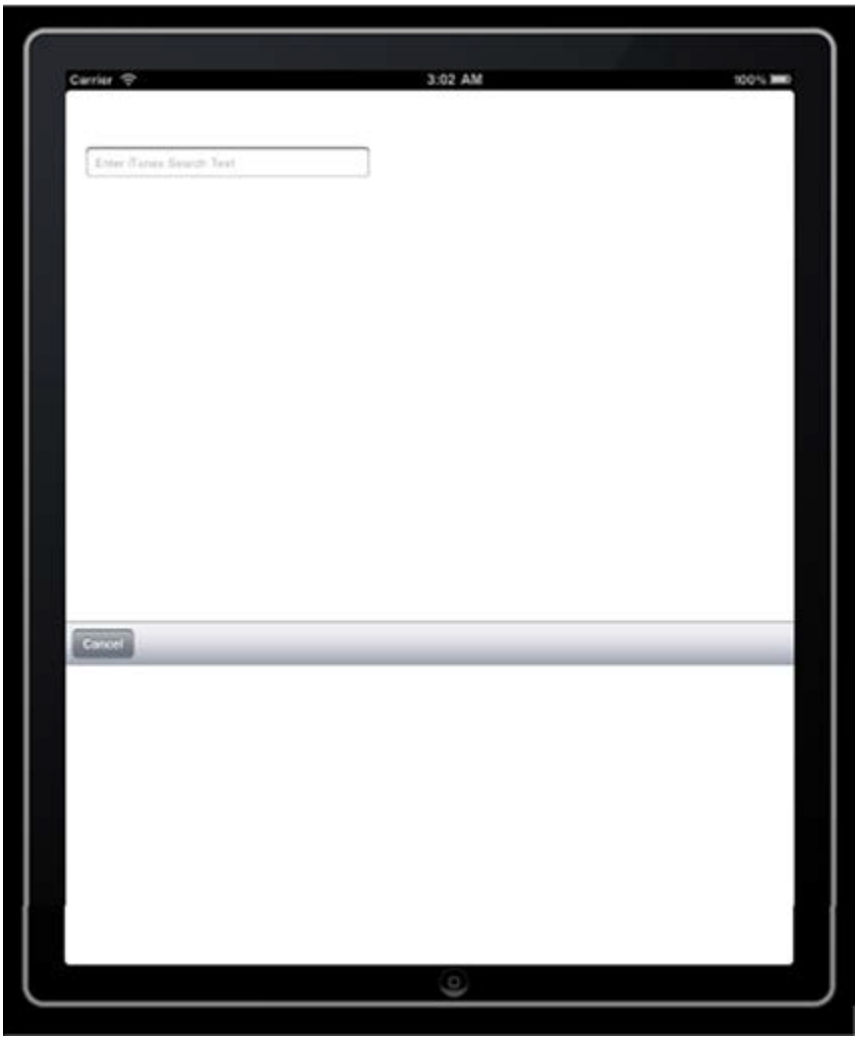


图 10.10 Xcode 升级到 iPad 之后，搜索视图的 layout 出错了

但是功能没有什么大问题。你可以在输入框中输入文本，点击 Send，在第一个 table view 中看到搜索结果（见图 10.11）。

技术上的升级完成之后，UI 的 layout 却不怎么对。看起来像在一个巨大的 iPhone/iPod 上运行一样，但是文字变小了。对一些应用来说，这样可能没有问题，但是对 table view 来说，看上去有点糟，导航给人感觉有些尴尬。使用 UISplitViewController 会更好。

技巧 78 给应用增加一个 split view

split view controller 的一个常见用法是使用 master view 来做导航，显示层级信息，右边的 detail view 可以用来展示左边选择的相关信息。

问题

你想要使用一个 split view controller 来显示左边的 table view，detail view 来显示相关的专辑图片。

解决方案

可以把一个 split view 添加到工程的 UI 中，让显示数据的方式更优雅。

讨论

打开 MainWindow-iPad.xib 文件，把 UISplitViewController 拖拽到文件中（见图 10.12）。

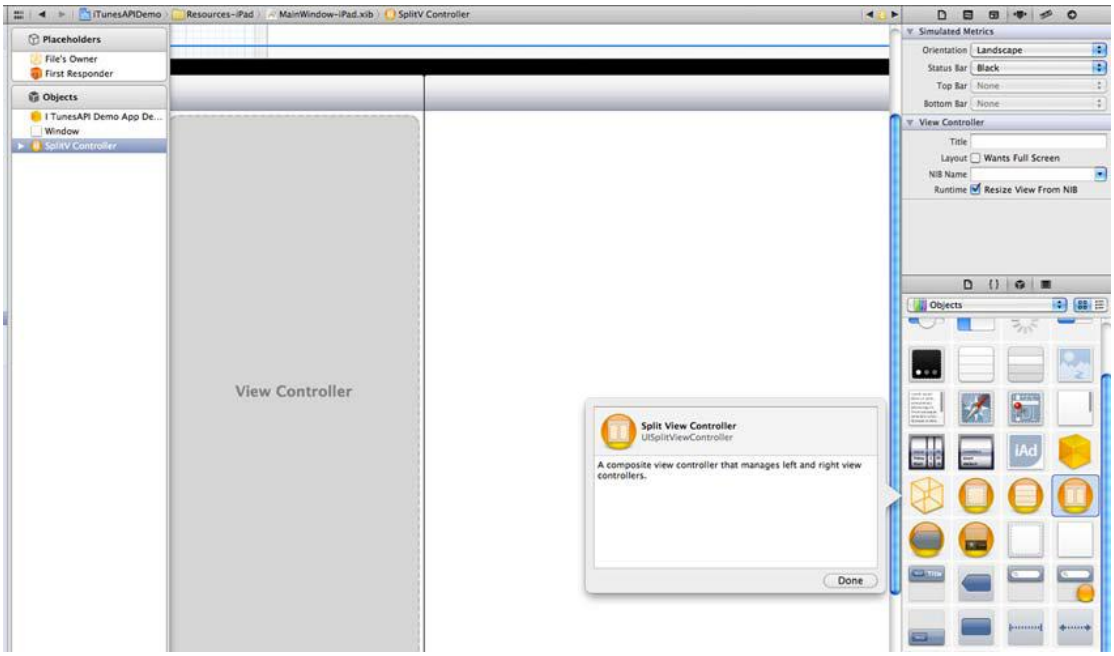


图 10.12 添加一个 split view 到 MainWindow-iPad.xib 文件

展开 split view controller 的层级，第一个子 view controller 是是一个 navigation controller。因为你已经有一个 navigation toolbar，所以把它拖到 split view controller 中进行替换（见图 10.13）。

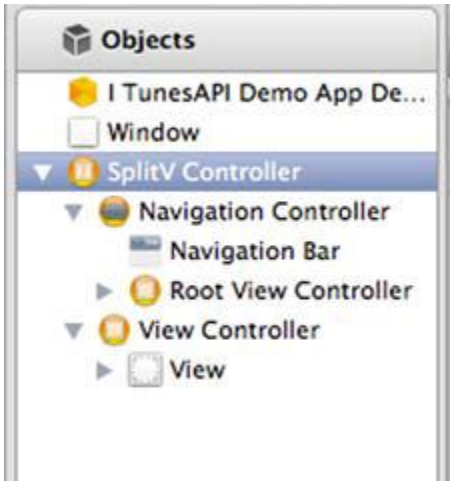


图 10.13 在 split view controller 中替代 navigation controller

在工程中创建一个新的类，命名为 SVCSplitController，这个类拓展了 UISplitViewController。在 IB 中，把 split view controller 的 class 类型变为新的 SVCSplitController 类。同时，给 SVCSplitController 的执行文件增加和 RootViewController 相同的旋转方法(参考代码：回调一个 view controller 来确定是否 view controller 需要旋转)。

下一步，你需要根据应用在什么设备上运行来显示相关的 main controller。要做到这样，可以把 MusicSearchAppDelegate 中的 base view controller 提取出来。现在是 UINavigationController ,但是对于 iPad 来说 ,你是把 navigation controller 放在了 split view controller 的 master view 里。

如果你把 navigation controller 留下作为主视图，那么它就会一直这么显示。还记得那个巨大的 table view 吗？在 MusicSearchAppDelegate 中，把 navigationController 的类型改为 UIViewController。然后，右击 navigationController，选择 Refactor（要确保所有的 Xcode 运行项目都停止了——如果是运行状态的时候，点击 icon 看起来会像一个停止符号）。选择 Refactor，输入新名字 mainController。

对于老的 MainWindow.xib，navigation controller 仍然是 mainController。而对

iPad 来说，把 split view controller 和 mianController 的成员连线（见图 10.14）。

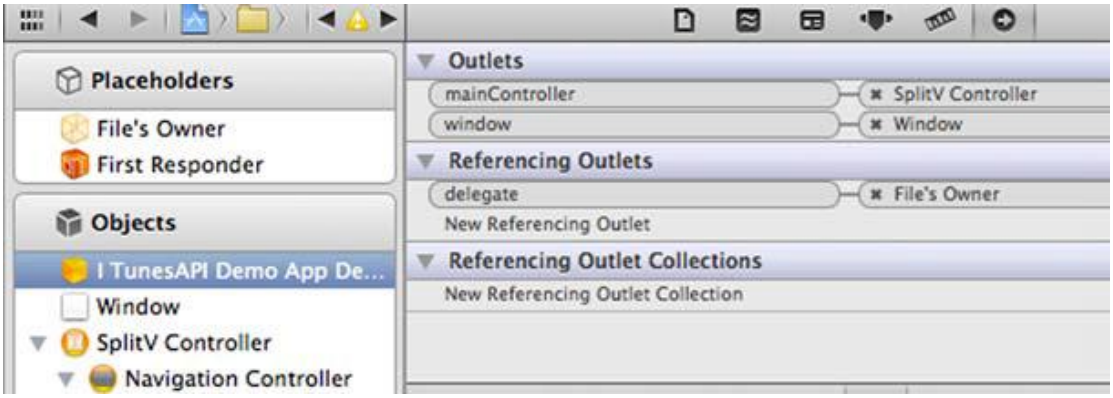


图 10.14 在 app delegate 中连线 split controller 和 mainController

现在运行应用，你可以看到一个空白的黑屏（见图 10.15），这是因为我们 detail view 中什么也没有。

图 10.15 带有 split view controller 的 iPad 在 iPad 模拟器中运行

如果你旋转到水平位置，可以看到 navigation controller 和 master view 中的 table view（见图 10.16）。

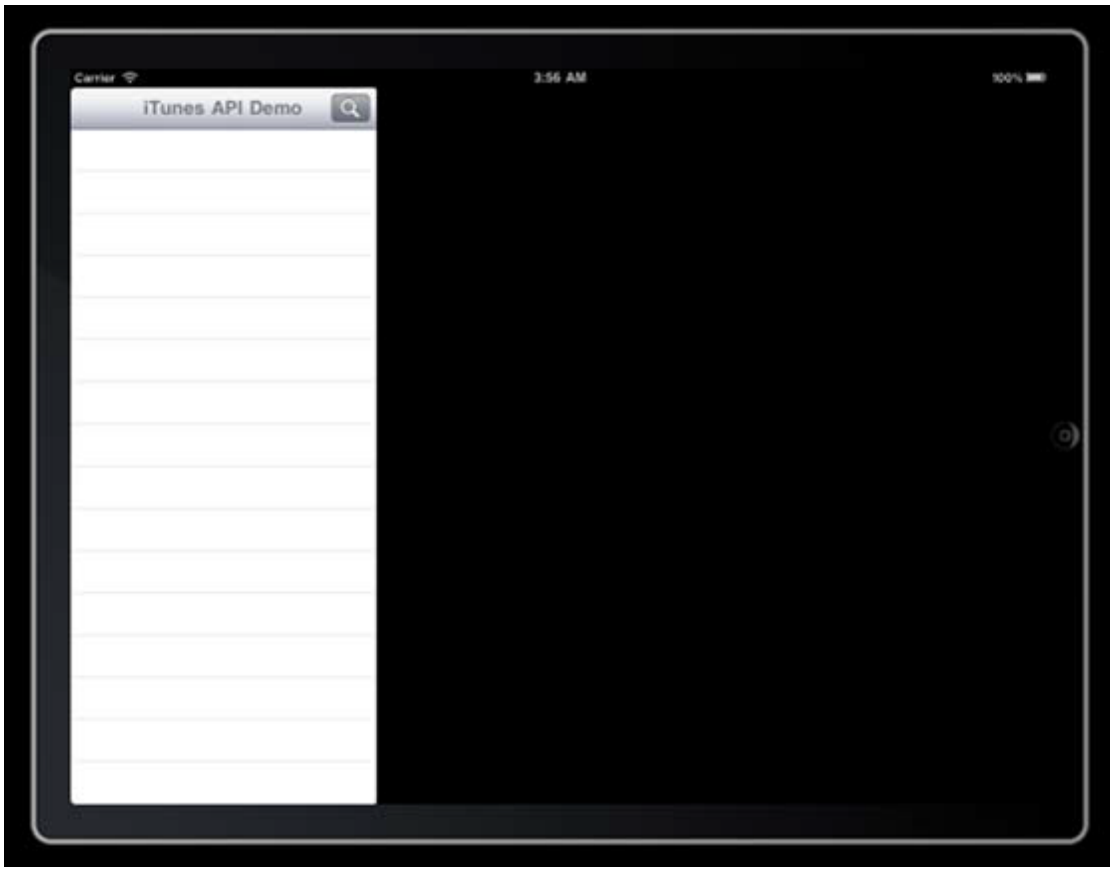


图 10.16 带有 split view controller 的 iPad 水平方向在 iPad 模拟器中运行

把一个普通的 UIView 拖拽到 detail 区域中，然后将一个 toolbar 放在顶部，然后再拖拽一个 UIView 让它占据其他的位置（见图 10.17）。

图 10.17 detailview 中一个 toolbar 和两个 view 的布局

回到 SVCSplitController，在头部声明一个 UIToolbar，然后在 IB 中把它和在 detail view 中顶部的 toolbar 连接起来。

如果现在运行应用，toolbar 会消失，因为它不知道如何处理多种方向。这是因为自动调整大小（Auto-sizing）设置不正确（见图 10.18）

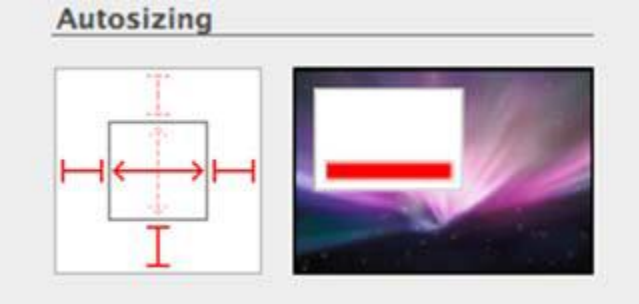


图 10.18 toolbar 的 Auto-sizing 导致这个控件在屏幕之外

按照图 10.19 设置 auto-sizing，让不管 iPad 的方向如何，toolbar 保持在顶部。

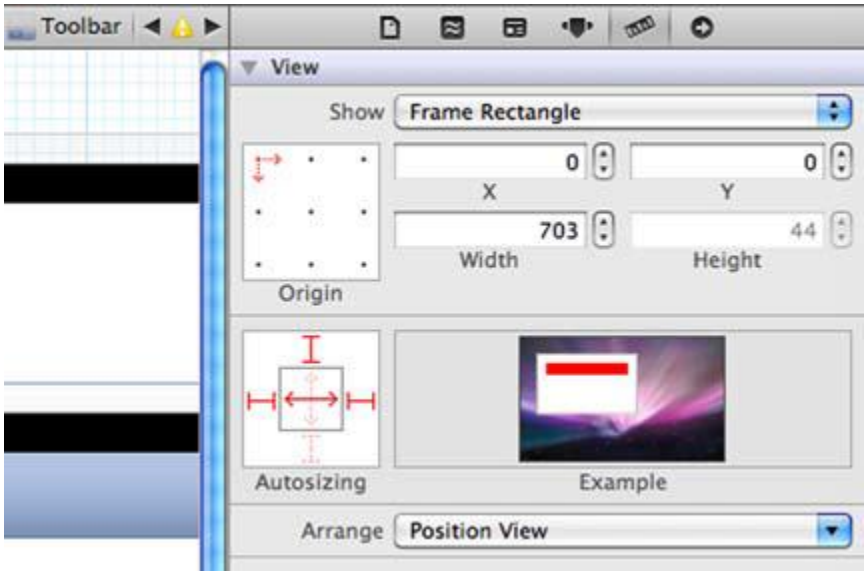


图 10.19 设置 auto-sizing 让 toolbar 保持在顶部

现在不管 iPad 方向如何，detail view 上都有一个一直保持在顶部的 toolbar 了。

当 split view 是竖直方向的时候，master view 不会显示，但是有回调调用 split controller 的 delegate。

让我们在 split view controller 的类本身中处理这些回调，意思就是它本身的 delegate。在头部声明接入 UISplitViewControllerDelegate 然后设置本身作为 viewDidLoad 中的 delegate（见以下代码）。

设置本身作 SVCSplitControllerviewDidLoad 中的 delegate

```
- (void)viewDidLoad {
[superviewDidLoad];
```

```
self.delegate = self;
}
```

当 split view controller 旋转至水平方向时，一个回调会指定让 master view controller 隐藏起来，同时一个和 master view 相关的 UIButtonItem 会传递过来。你可以增加这个 Bar 按钮到 detail 区域的 toolbar 中（见以下代码）。

当 split controller 在水平方向时进行回调并且隐藏 master

```
- (void)splitViewController:(UISplitViewController*)svc
willHideViewController:(UIViewController *)aViewController
withBarButtonItem:(UIBarButtonItem*)barButtonItem
forPopoverController:(UIPopoverController*)pc
{
[barButtonItemsetTitle:@"Menu"];
[tbTopsetItems:
[NSArrayarrayWithObject:barButtonItem]
animated:YES];
}
```

同理，当 split view controller 到水平方向把按钮作废的时候会有一个回调，然后显示 masterview controller（见以下代码）。

作废目录按钮

```
- (void)splitViewController:(UISplitViewController*)svc
willShowViewController:(UIViewController *)aViewController
invalidatingBarButtonItem:(UIBarButtonItem *)button
{
[tbTopsetItems:nilanimated:NO];
}
```

现在当用户在 iPad 模拟器中把应用调到水平方向的时候，Menu 按钮会出现。当这个按钮被点击的时候，带着 table view 的 master view 会显示（见图 10.20）。

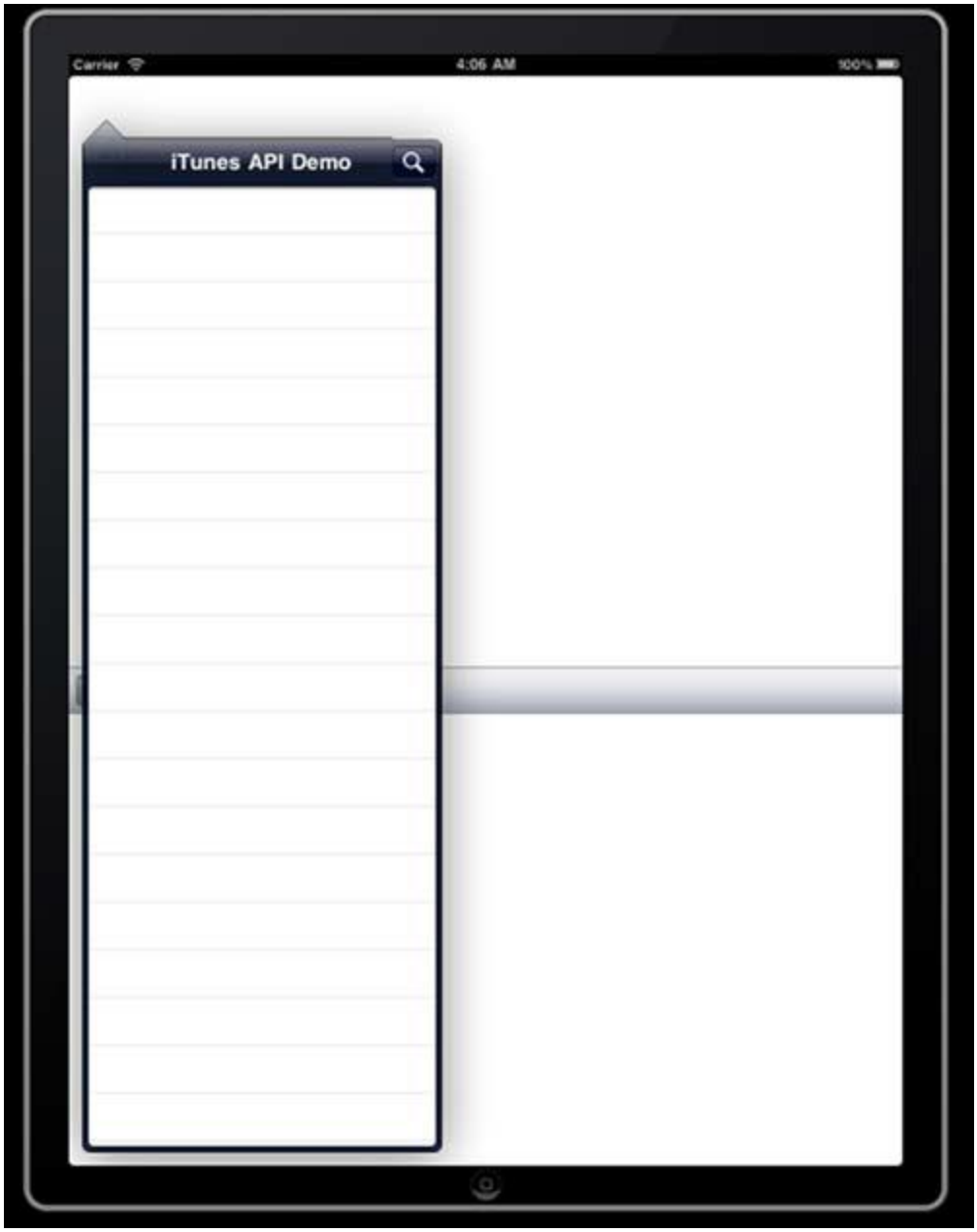


图 10.20 iPad 应用在竖直方向运行的时候 master view 会以一个弹出框的形式出现

就像 Search view toolbar 需要有一个固定的 auto-sizing 设置，web view controller 也一样（见图 10.21）

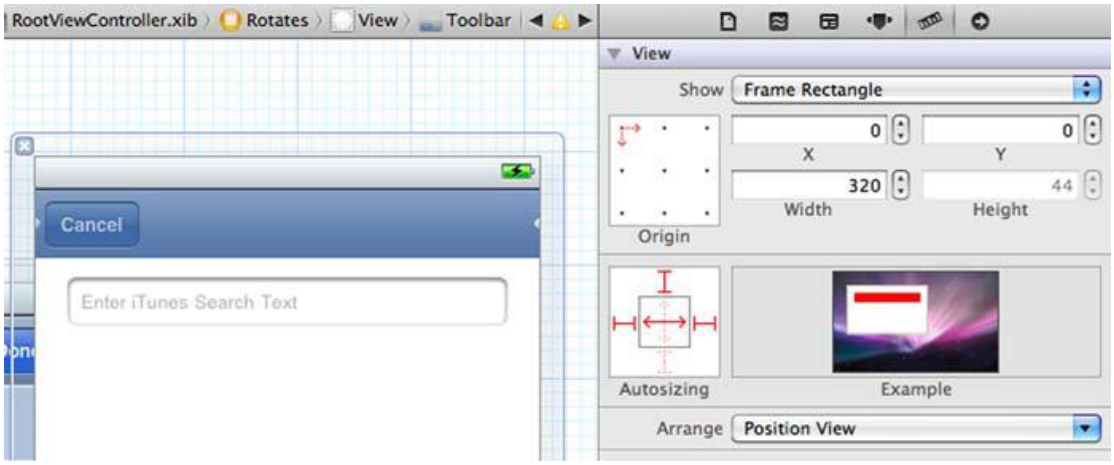


图 10.21 在新设备上固定 toolbar，让其方向正确

基于控件的不同，你可能会希望设置不同的 auto-sizing，这样让控件可以在中心，或是左边，右边，放大缩小显示。

让不同的控制器在 view 上正确的显示需要多少工作量，这取决于应用的不同。同时，如果你需要在不同大小的设备上测试 view 的话，需要确保让他保持在一个位置，不要和其他的重合。

Master view 经常需要处理 navigation，但是以某种程度来说，你会希望在 detail view 上显示一些相关数据（比如说邮件内容），让我们看看再通用二进制的情况下怎么做。

技巧 78 在 detail view 中显示 item

正如早前说的，在 iPad 应用中使用 split view controller 一般都倾向于这么一种模式：master view 处理 navigation，detail view 处理细节信息。

在 MusicSearch 应用中我们也会采取同样的模式。应用中没有很多 navigation 的内容，但是 master view 中有 table view，用户在 table view 里选择查询结果。用户选择了查询结果之后，可以在 detail view 中显示图片。

但是由于这是通用二进制，如果不是 iPad 设备运行应用就不会使用 split view controller。那么如何处理设备不同造成的 UI 上功能的不同呢？

问题

关于专辑图片的显示 ,在 iPhone 上你想要在一个模态显示的 view controller 中显示，而在 iPad 上，则在 detail view 中显示。

解决方案

可以使用多种方法来试探应用在什么设备上运行。[[UIDevicecurrentDevice] model]返回了 model，你可以用这个来测试者是否是 iPad 还是其他型号。UI_USER_INTERFACE_IDIOM()返回 UIUserInterfaceIdiomPhone（如果是 iPhone 或是 iPod），或是 UIUserInterfaceIdiomPad（如果是 iPad）。

你也可以在 main navigation controller（iPhone/iPod）中设置变量，在 SVCSplitController 也做相似的办法。根据设备型号的不同，mainController 的值也不同（iPhone/iPod 是 UINavigationController，iPad 是 SVCSplitController）。这个变量可以用来确定设备的型号。基于设备分辨率的不同，也可以使用相似的变量，但是长期来说可能会存在不一致的情况。

我们可以使用 UI_USER_INTERFACE_IDIOM()路径，但是如果是 iPad，需要用 NSNotificationCenter 来发送一个通知，由 SVCSplitController 注册并监听它。. 需要显示的 view 会在通知中包括起来，SVCSplitController 会在 detail view 中显示这个 view。

讨论

为了显示 detail view ,SVCSplitController 需要注册一个通知来接受传过来的视图(见以下代码)，这可以在 viewDidLoad 方法中完成。

SVCSplitControler 为 DisplayDetailsView 注册通知

```
[[NSNotificationCenterdefaultCenter] addObserver:self
selector:@selector(displayInDetails:)
name:@"DisplayDetailsView" object:nil];
```

当这个命名为 DisplayDetailsView 通知发送出去的时候，SVCSplitController 会收到

通知然后传送给 NSNotification 实例。NSNotification 实例会包含要显示在 detail view 中的 view（见以下代码）。

加工通知来显示细节

```
-(void)displayInDetails:(NSNotification*)notif;
{
UIView *view = [notif object];
for (UIView *v in [detailsViewsubviews])
[vremoveFromSuperview];
[detailsViewaddSubview:view];
}
```

现在无论何时，一个类希望在 detail view 中显示一些东西的时候，它可以发送一个通知，这个通知里面包含需要显示的 view(见以下代码)。

iPhone/iPod 设备显示模态的 detail view，iPad 则显示带 view 的通知

```
if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPhone)
[selfpresentModalViewController:vcWebViewanimated:YES];
else
{
NSString *artistViewUrl =
[dictobjectForKey:@"artworkUrl100"];
[webViewloadRequest:[NSURLRequest
requestWithURL:[NSURL URLWithString:artistViewUrl]]];
[[NSNotificationCenterdefaultCenter]
postNotificationName:@"DisplayDetailsView"
object:webView];
}
```

苹果的 iPad 确实非常伟大。概念简单，执行起来完成的产品很棒。iPad 拥有优秀的手机体验，但是又稍有不同，可能更好。苹果做的另一个令人激动的东西就是广告。我第一次

见到 iAds 的时候就觉得非常有趣。让我们看看如何把它加入到应用中。

10.3 给应用增加 iAds

打广告并不是一种新的盈利方式。就算对手机应用来说都不新鲜。但是苹果的方式不同，由于你应用的分发商同时也是广告商（从某种程度来说，苹果自己本身虽然不做广告，但是他们提供广告，并且为交易提供保障）。

以我们的应用来说，实际不好做销售，因为它和 iTunes 差不多。而且我们也有没什么 IAP 的内容提供给用户。所以就让我们来添加广告看看是否能用这种方式盈利。

你的应用可能买不了 0.99 美元，但是你可以每次从一个用户那里得到 0.1 美元，这样有了 20 次的话，实际赚的更多。让我们看看怎么给应用增加 iAds。

技巧 78 给 iAds 配置 iTunes

像其他的给应用添加的一些特殊功能一样（比如说 Game Center，IAP，APN），iAds 也需要在 iTunesConnect 进行在线配置，但是工作量不大。

问题

你需要在 iTunesConnect 中配置应用，让应用增加 iAds。

解决方案

你只需要登录 iTunesConnect 然后为应用打开 iAds 功能。

讨论

工作量不大，但是很重要。如果没有给应用打开 iAds 功能，就不会有任何广告。

在 Provisioning Portal，创建一个新的 app ID。然后登录 iTunesConnect，点击你的应用进行管理。点击 Set Up iAd Network（见图 10.22）。

你可能需要在 iTunesConnect 中检查一下合同，确认所有的条例。周期性的这样确认

是个好习惯。有的时候客户会要问我为什么他们的应用不在 AppStore 中了。我第一件事就是让他们登录 iTunesConnect，看看有没有什么新的或者更新的合同条例需要他们同意。

现在你已经在 iTunesConnect 中增加了 iAds 了，现在让我们给应用加上。

技巧 78 给应用增加 iAds

给应用增加 iAds 不想是增加一个常用的 view。在 IB 中，我们可以拖拽一个 ADBannerView 到工程中。

问题

你需要给工程增加 iAds，然后依次处理它。

解决方案

在 UI 编辑器中，添加一个 ADBannerView 到 UI，然后，在编码中，当 ad 没有加载的时候，处理回调方法来隐藏广告的 view。

讨论

打开 RootViewController.xib 文件。由于专辑图片很小，所以 UIWebView 不需要太大。让我们用 UIWebView 空出的空间作为放置广告的区域。

把顶部的 web view 往下拖一点，留出一些空间，然后把一个 ADBannerView 放在那儿（见图 10.24）。

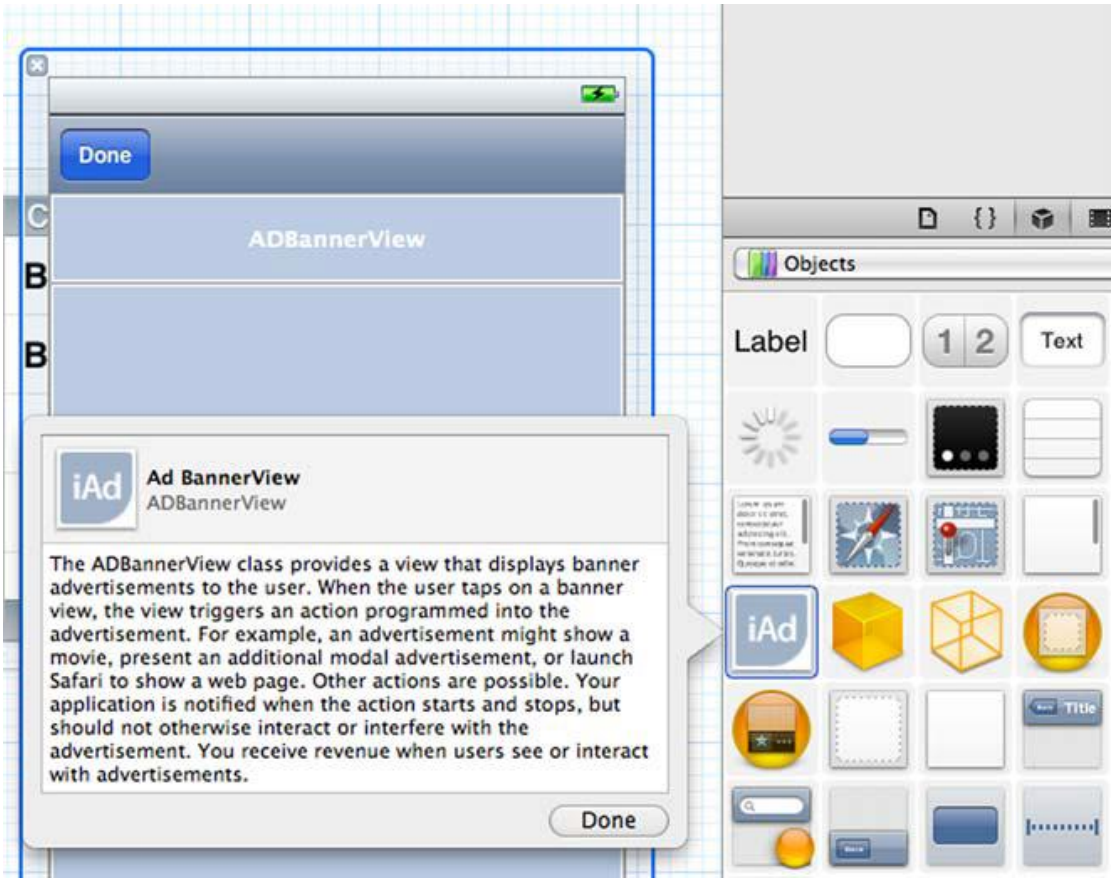


图 10.24 拖拽一个 ADBannerView control 到 Web view 的 UI 中

在头部声明 RootViewController 接入 ADBannerViewDelegate。同时声明一个 bool flag 命名为 hidingAdBanner。

广告不显示的时候，你希望可以隐藏 ADBannerView control。基于什么会进入以及现在的状态的方法可以帮你隐藏或者展示广告 banner view（见以下代码）。

隐藏AdBanner的方法

```
-(void)hideAdBanner:(bool)hideIt;
    {
    if ((hideIt&&hidingAdBanner)
    || (!hideIt&& !hidingAdBanner))
    return;
    hidingAdBanner = hideIt;
    [UIViewbeginAnimations:nilcontext:nil];
    intadHeight = adBanner.frame.size.height;
```

```
CGRect r = adBanner.frame;
r.origin.y -= adHeight;
adBanner.frame = r;
r = webView.frame;
r.origin.y -= adHeight;
r.size.height += adHeight;
webView.frame = r;
[UIViewcommitAnimations];
}
```

你可以给 ADBannerView 使用不同的 delegate 方法。使用 IB，把 ADBannerView 的 delegate 设置为 RootViewController。你对两个回调方法感兴趣——广告加载时以及广告加载失败时。

应用启动时，可以调用 hideAdBanner，参数为 YES，这样 banner 就会消失因为没有广告加载进来。当广告加载之后，再调用它，参数为 NO。如果一个广告加载失败，再次调用这个方法，参数为 YES（见以下代码）。

广告加载失败调用 hideAdBanner，参数为 YES，广告加载成功，参数为 NO

```
- (void)bannerView:(ADBannerView *)banner
didFailToReceiveAdWithError:(NSError *)error
{
[selfhideAdBanner:YES];
}
- (void)bannerViewDidLoadAd:(ADBannerView *)banner
{
[selfhideAdBanner:NO];
}
```

在模拟器中运行，会加载来自苹果服务器的测试广告。这样测试很好，可以很好的监测出代码是否有效。有的时候它们会加载失败，这对测试来说也有帮助（也许苹果是故意这么

做的）。应用在 App Store 发布之后，会加载真正的广告。

10.4 总结

iTunesConnect 可以给你生成报告，报告会显示应用目前加载显示广告的情况，点击率，付费率。

在 Provisioning Portal，苹果开发者站，iOS 论坛，还有很多其他地方会有一些很棒的信息。我建议你去访问这些网站，看看其他开发者是如何解决问题分享解决方案的，尽可能的多去学些东西。

不要害怕问问题，我认为 iOS 开发者都是乐于助人的！

11 ——通过制作一个上架应用 MeetSocial 学习集合视图，社交功能，提醒事项以及应用状态存储

本章包括：

- 使用集合视图
- 集成 Social Framework
- 提醒用户日程安排
- 存储和恢复应用程序状态

iOS 6 添加和修改了大量框架、类以及协议，使得它们的概念和功能都发生了变化。本章将带本章将通过教您制作一款上架应用来学习许多 iOS 6 中出现的新技术。。

如果用户需要参加某项会议，可以使用 MeetSocial 来搜索 meetup.com 上的会议信息，可以根据会议的关键词或会议地点来查找。会议的搜索结果会显示在一个集合视图（UICollectionView）中。集合视图和列表视图（UITableView）很像，但是集合视图可以改变单元格的宽度大小，每一行可能有不同数量的单元格；与之相反，列表视图的宽度与单元格宽度相同，因此每一行只有一个单元格。

用户还可以将 MeetSocial 的搜索结果分享给好友，为实现这个功能我们可以使用 iOS6 中全新的社交框架（Social Framework），它包括 Twitter 和 Facebook 等流行社交元素。在 MeetSocial 项目中你将学习如何使用框架在应用中集成分享功能以及将分享界面展示给用户。

一项会议可以看作是一个具有日期和时间的事件（event）。用户可以使用 MeetSocial 创建一个提醒事项（reminder），它可以在会议召开前夕提醒用户。提醒是 EventKit 框架在 iOS 6 中添加的一项新功能。iOS 6 添加的另一项新功能是应用状态恢复，例如，对于

UIKit，此功能可以让应用程序存储其用户界面会话状态，以便在用户下一次启动应用时快速恢复，让用户产生没有关闭过应用程序的错觉。

MeetSocial 涵盖了以上所描述的所有新功能，当然也包括一些您之前学过的功能。虽然本项目中即将学习的新技术你没有接触过，但是使用它们将让应用的用户体验大大增强，因此你应该努力学习，了解如何将新功能集成进自己的项目。

11.1 使用集合视图展示数据

集合视图的许多核心概念都源自列表视图，你在第四章的 Dial4 项目中曾学过列表视图，它们非常相像，无论是数据源（datasource）还是委托（delegate）的使用方法都是一样的，也都包含块（section）、行（row）以及单元格（cell）。

技巧 86 创建一个使用集合视图的项目

你将创建一个基于“Master-Detail Application”模版的 iOS 项目，但是默认它使用的是列表视图，所以还需将其修改为使用集合视图。然后可以给项目添加 UI 和功能，让用户可以使用我们的软件搜索 meetup.com 上的会议信息并将搜索结果显示在集合视图上。

问题

你需要创建一个包含集合视图的新项目，以便展示搜索结果。

解决方案

打开 Xcode，选择 Create a new Xcode project，再选择 Master-Detail Application 模版，点击 Next 后将项目命名为 MeetSocial，在填完其他需要的信息后选择项目保存路径，单击 create 完成项目的创建。

讨论

目前你已经创建了一个项目，似乎在重复着和前面所有章节一样的乏味过程，再没有什么新鲜感了，但是我必须指出一些你可能忽略的注意事项。首先，本项目使用了 Storyboards

（见图 11.1），但是不使用 CoreData。

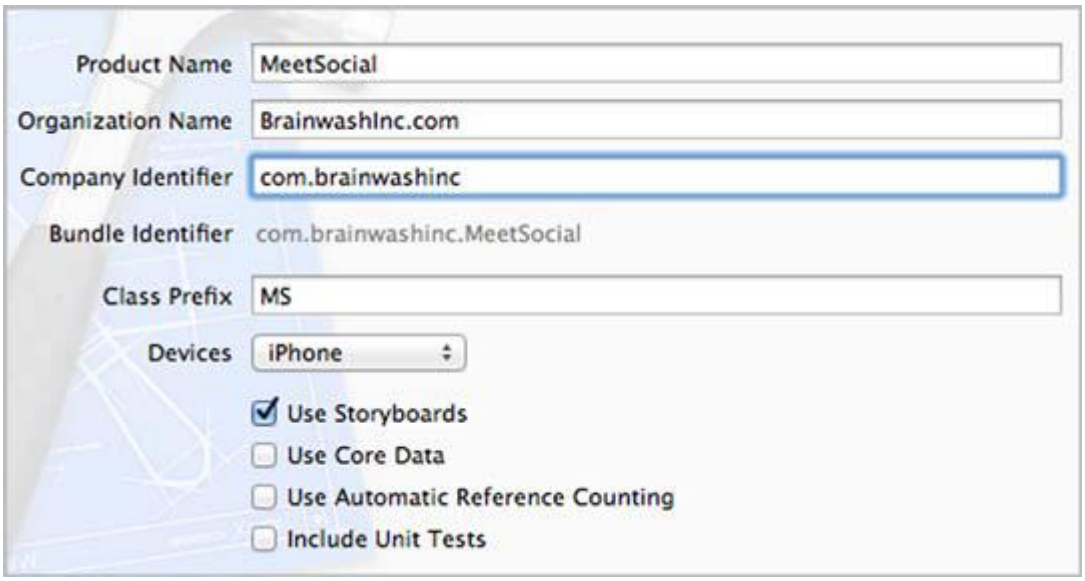


图 11.1　创建使用 Storyboard 的项目——MeetSocial

其次，本项目不使用自动引用计数（ARC），这是因为截至本书写作期间，SBJSON 框架还不支持 ARC，我本人对是否使用 ARC 并不感冒……我已经记不清曾经在哪个项目中用过 ARC 了。

为了让标准的 Master-Detail Application 模版更符合本项目的需求，我们需要做一些额外工作。首先，在 navigator 栏中选择 MSMasterViewController.h 文件，在代码编辑区中你可以看到它目前继承自 UITableViewController，我们需要将其父类改为 UIView - Controller。接下来再打开 Storyboard 文件，删除系统为我们生成的主视图控制器和详细视图控制器。

拖一个新的 UIViewController 到 UI 编辑区，并在 Identity Inspector 中将其类名设置为 MSMasterViewController。然后将其与 UINavigationController 中的 root view controller 相连（见图 11.2）。

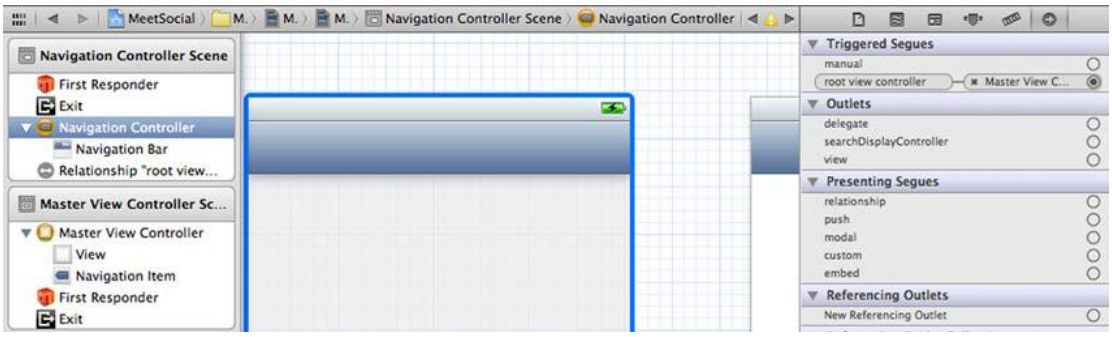


图 11.2　删除 Xcode 自动创建的列表视图

再拖一个 UICollectionViewController，选中其中包含的 Collection View Cell，模仿图 11.3 为其绘制 UI。您还需将单元格的 identifier 设置为 ResultsCell，之后我们通过这个标识符来创建和重用此单元格。

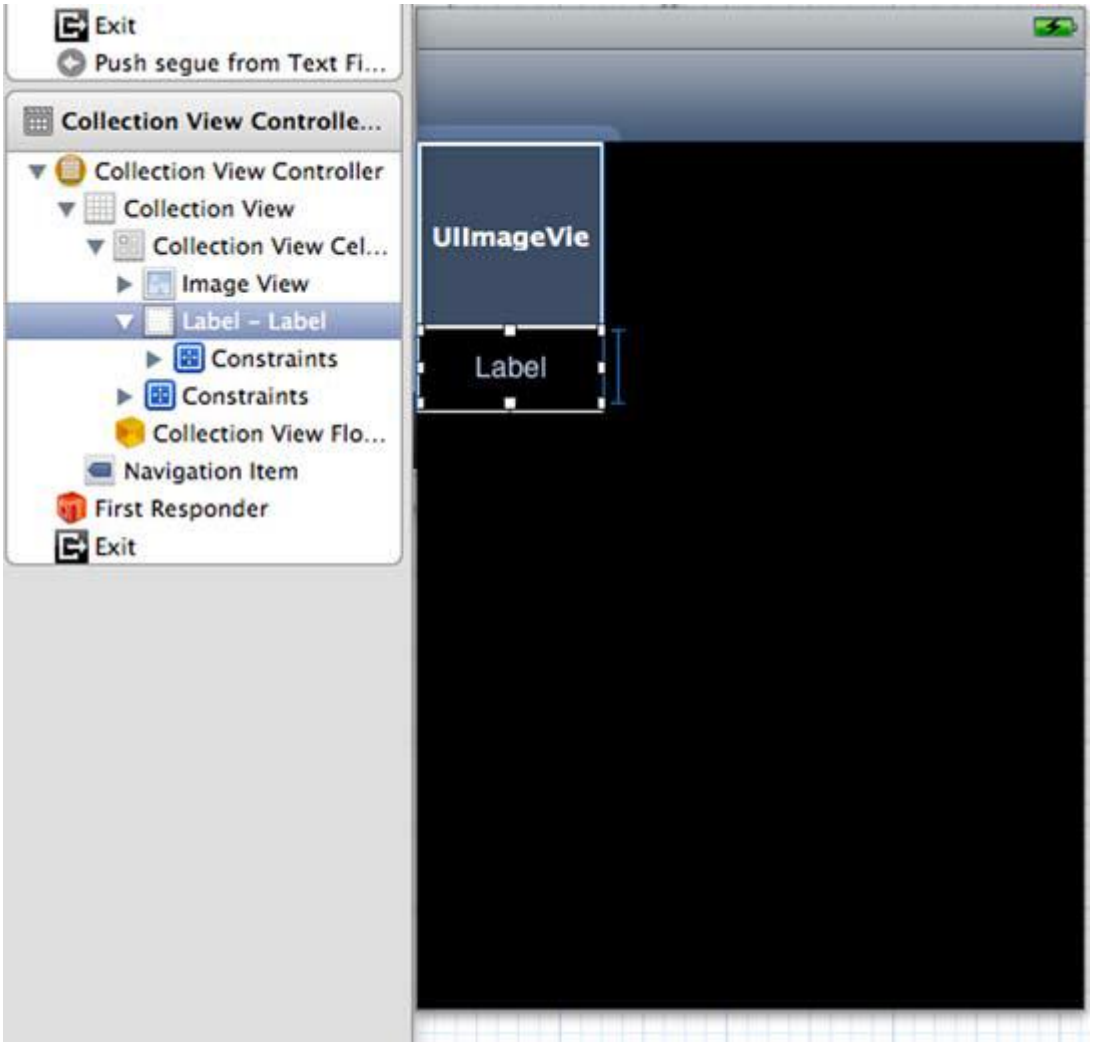


图 11.3　添加 UICollectionViewController 并绘制 Collection View Cell

在开发本应用的过程中，你随时可以调整 cell 的大小以符合 UI 需要。接下来请将 MSDetailViewController 的父类由 UITableViewController（译者：实际上是 UIViewController）改为 UICollectionViewController，同时在 IB 中将之前创建的 UICollectionViewController 的类名设置为 MSDetailViewController。之后再删除 configureView 方法，这个方法是 Xcode 自动创建的，本项目中我们不需要这个方法。

现在请模仿图 11.4 为主视图绘制 UI，并在 Assistant editor 中将这些 UI 元素自动转化为头文件中的 IBOutlet 属性。需要注意对象的命名原则——有意义并符合规范。

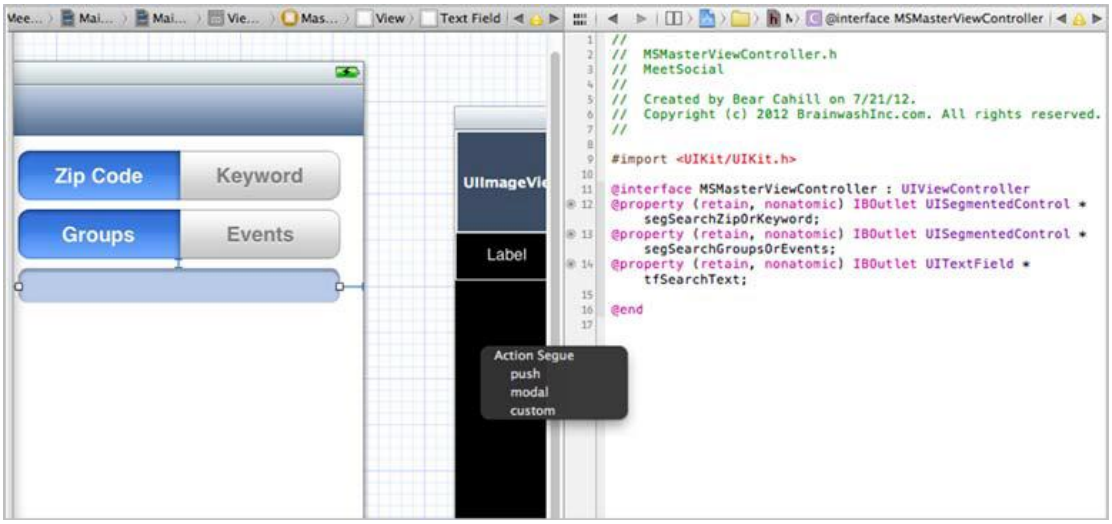


图 11.4 绘制主视图 UI 并设置页面流

接下来请按住 control 键，将主界面中的 UITextField（对象名：tfSearchText）拖到 UICollectionView 上，在弹出菜单中选择 Push，这样您就创建了一个从输入搜索关键词到显示搜索结果的页面流。同时请您设置此文本框的委托，将其 delegate 拖动至 MSMasterViewController，并在 Attributes Inspector 中将文本框的 Return key 设置为 Search。按照我个人习惯，我通常还将 Clear button 设置为 appear while editing 并勾选 Auto-enable Return Key。

现在请返回 MSMasterViewController.m 文件。需要删除此行代码：

```
NSMutableArray *_objects;
```

它位于文件起始处的匿名类别声明中，我们不需要使用这个 Xcode 自动创建的变量。

现在你会看到大量的编译错误（见图 11.5——为突出错误代码所在行，这里只摘录了部分代码）。为了解决这些错误，您需要定位到每一处使用了_objects 的代码。这很简单：首先删除方法 insertNewObject:（在 viewDidLoad:委托中创建 addButton 时调用了此方法），其次将 tableView:numberOfRowsInSection:委托方法的实现代码改为 return 0;。接下来在另一个委托方法 tableView:cellForRowAtIndexPath:中，删除

```
NSDate *object = _objects[indexPath.row];
cell.textLabel.text = [object description];
```

两行代码。按照同样的方式，请您删除以下方法体中包含了_objects 的代码：commitEditingStyle, didSelectRowAtIndexPath 和 prepareForSegue。

然后还需要在头文件中添加一个协议声明：UITextFieldDelegate。再次编译，你将高兴的看到项目已经顺利通过编译了。

现在有一个万事俱备的空项目等着你添加以下功能了：得到用户搜索关键词，联网搜索结果并显示在集合视图上。下一步你将实现这些功能。

```
- (void)insertNewObject:(id)sender
{
    if (!_objects) {
        _objects = [[NSMutableArray alloc] init];
    }
    [_objects insertObject:[NSDate date] atIndex:0];
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSection:0];
    [self.tableView insertRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:
        UITableViewRowAnimationAutomatic];
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return _objects.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    NSDate *object = [_objects objectAtIndex:indexPath.row];
    cell.textLabel.text = [object description];
    return cell;
}

- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    forRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [_objects removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:
            UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table
            view.
    }
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    if ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPad) {
        NSDate *object = [_objects objectAtIndex:indexPath.row];
        self.detailViewController.detailItem = object;
    }
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([[segue identifier] isEqualToString:@"showDetail"]) {
        NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
        NSDate *object = [_objects objectAtIndex:indexPath.row];
        [[segue destinationViewController] setDetailItem:object];
    }
}
```

图 11.5　删除 NSMutableArray *_objects;后的编译错误

技巧 86　从 Meetup.com 获取搜索结果

我开发过的大多数应用都有服务器端，服务器端可能是项目的一部分（项目自身包含服务器端和客户端），也可能是为一个已经存在的服务器开发客户端。无论哪种情况，您都需要了解服务器提供的 API，并根据 API 设计应用的界面和功能。

问题

现在已经完成了界面设计，接下来你将为程序添加功能。需要让应用取得用户在搜索框输入的关键词，并根据关键词从 Meetup.com 获取搜索结果。

由于本应用功能比较简单，因此我们只实现了 meetup.com 所提供接口的很小一部分——应用可以按会议日期和会议组织者两大类搜索，并且分别提供邮政编码（会议举办地

点）和关键词两个搜索选项。

解决方案

MeetSocial 获取用户输入的关键词后，向 meetupcom 发送查询请求。注意你需要在 meetupcom 上注册以获得 API 使用许可密钥。请点击这里获取该密钥：http://www.meetup.com/meetup_api/key/。

如果想参考 meetup.com 的 API 文档请点击这里：http://www.meetup.com/meetup_api/，这里详细描述了 API 的使用方法，为了在本项目中使用 API，只需要使用两个附加了相应 GET 请求参数的 URL 即可。

讨论

当应用启动后，应该做一些准备工作，在 viewDidLoad 方法中，删除[super viewDidLoad];下方的几行代码，并增加：

```
[self setTitle:@"Search"];
```

类似的，添加 viewWillAppear 委托方法，并增加：

```
[tfSearchText becomeFirstResponder];
```

现在，如果运行应用，界面应该与图 11.6 类似，我们已经为下一步工作做好了准备。

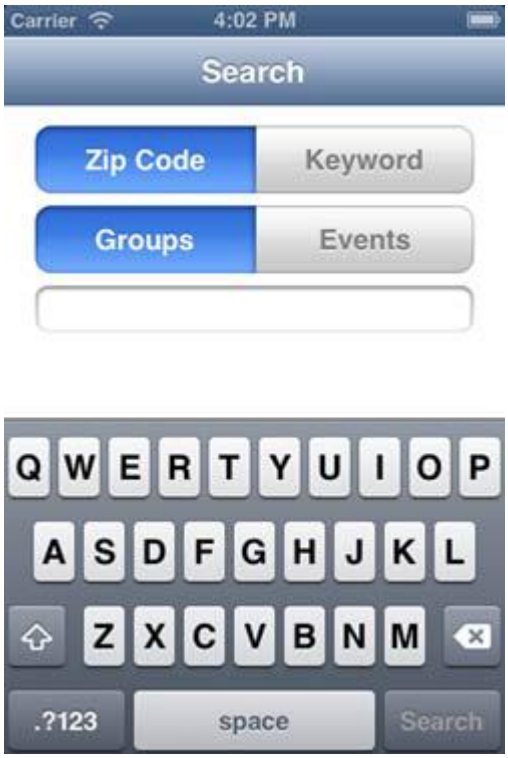


图 11.6 应用在模拟器中运行

现在可以选择搜索选项，也可以在搜索框中输入文本，但是当你点击 Search 时程序没有反应。所以请在 MSMasterViewController.m 中加入下面的代码：

```
- (BOOL)textFieldShouldReturn:              (UITextField *)textField;
{
    [textField resignFirstResponder];
    return NO;
}
```

现在再次运行项目，你会看到一个集合视图，虽然还没有任何数据显示（见图 11.7）。

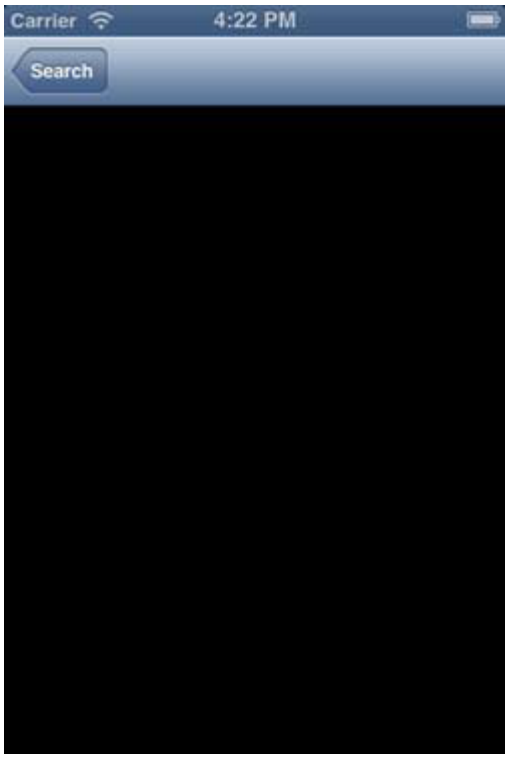


图 11.7 因为没有实现搜索方法，所以集合视图没有数据

那么现在就让我们根据用户提交的信息实现搜索方法。您将获得三个方面的信息：搜索选项（邮政编码/关键词）、搜索内容（会议日程/会议组织者）以及搜索框中的文本。

根据这些信息，您可以构造相应的 URL 访问服务器并取得搜索结果（请看下列代码）。

构造 URL 访问服务器并取得搜索结果

```
-(void)doSearch;
{
    //查询内容
    NSString *groupsOrEvents =
        segSearchGroupsOrEvents.selectedSegmentIndex == 0
            ? @"groups" : @"2/open_events";
    //查询选项
    NSString *zipOrKeyword =
        segSearchZipOrKeyword.selectedSegmentIndex == 0
            ? @"zip" : @"topic";
    //填写您的 API 密钥
```

```
        NSString *apiKey = @"...";
        NSString *query = [NSString stringWithFormat:
            @"https://api.meetup.com/%@?key=%@&sign=true&%@=%@",
    groupsOrEvents,    apiKey,          zipOrKeyword,
            tfSearchText.text];
        //查询服务器
        NSError *error = nil;
        NSURL *url = [NSURL URLWithString:query];
        NSHTTPURLResponse* retResp=nil;
        NSData *respData = [NSURLConnection sendSynchronousRequest:
            [NSURLRequest requestWithURL:url]
            returningResponse:&retResp error:&error];
        if (error)
            NSLog(@"ERROR: %@", error);
        else
        {
            //写入文件
            NSString *resultsJSON = [[[NSString alloc]
                initWithData:respData
                    encoding:NSASCIIStringEncoding] autorelease];
            [self writeToResultsFile:resultsJSON];
        }
    }
```

这里我想指出两点需要注意的地方。首先，请在上述代码中标示的位置填写你自己的 API 密钥，将其赋给 apikey 变量。另一方面，我们在代码中将搜索结果写入了一个本地文件，集合视图将使用此文件展示搜索结果，相对于直接将搜索结果保存在实例变量中，这样做可能会损失一些性能，但是在稍后我们学习恢复用户界面会话状态时，直接从本地文件读

取搜索结果而不是再次联网搜索将大大提高用户体验。

为了编写相关代码，你需要使用一个 NSString 的方法，请看下列代码（代码为简略起见只是将捕获到的错误打印出来，你可以为错误添加用户体验更好的处理方式）。

将搜索结果写入文件

```
-(void)writeToResultsFile:(NSString*)stringToStore;
{
    //构造写入文件的存放路径
    NSArray *paths = NSSearchPathForDirectoriesInDomains
        (NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectoryPath = [paths objectAtIndex:0];
    NSString *path = documentsDirectoryPath;

    //根据构造的路径创建目录
    NSError *error = nil;
    [[NSFileManager defaultManager] createDirectoryAtPath:path
        withIntermediateDirectories:YES
            attributes:nil error:&error];
    if (error)
        NSLog(@"Dir Error: %@", error);
    //写入文件
    path = [NSString stringWithFormat:@"%@/%@",
        path, @"results.json"];
    [stringToStore writeToFile:path atomically:YES
        encoding:NSUTF8StringEncoding error:&error];
    if (error)
        NSLog(@"Write Error: %@", error);
}
```

NSData 也有一个 writeOut:方法，您可以使用它来存储数据，当然也可以在集合视图控制器（对本项目来说是 MSDetailViewController）中将数据转化为您需要的形式。但是

我在这里使用了 NSString 的方法是因为我想在之后调试程序时，将结果在命令行中打印出来，以便检查是否得到了我们想要的结果。

现在请您在 textFieldShouldReturn:委托方法中增加[self doSearch];调用，至此，您已经得到了搜索结果，主视图控制器完成了自己的任务，该轮到集合视图控制器来展示搜索结果了。

技巧 86 在集合视图中展示数据

集合视图与列表视图很像，都使用数据源管理要显示的内容，也都用委托处理用户的各种操作，还都使用单元格。但是集合视图更加灵活，可以通过调整单元格的大小满足各类 UI 设计需求（如瀑布流，照片墙等）。

问题

你现在有一个存储了搜索结果的文件，现在需要在集合视图中展示。为了使界面更美观，你可能希望以这样一种方式来展示搜索结果：上方是一张会议相关图片，下面是会议名称和简要介绍。

解决方案

首先，MeetSocial 从文件中读取搜索结果，此时数据是 JSON 格式，因此需要将其转化为一个由 NSDictionary 对象组成的数组，然后集合视图将根据此数组中的元素展示搜索结果。

你目前有一个集合视图控制器，那么必然有一个集合视图与其关联，同时集合视图控制器需要实现 UICollectionViewDataSource 和 UICollectionViewDelegate 协议以便为集合视图提供数据和响应用户操作。现在，请实现协议中定义的方法。

对于 UICollectionViewDataSource，你需要实现两个方法：第一个是 section 中的单元格数量，第二个是根据 indexPath 返回指定的单元格：

```
-(NSInteger)collectionView:(UICollectionView *)collectionView
    numberOfItemsInSection:(NSInteger)section;
-(UICollectionViewCell*)collectionView:
    (UICollectionView *)collectionView
        cellForItemAtIndexPath:(NSIndexPath *)indexPath;
```

我们没有实现 numberOfSectionsInCollectionView:方法，因为本项目只需要一个 section，而此方法的默认返回值就是 1。

在用户选择单元格时，会调用 UICollectionViewDelegate 中的方法，你需要在后面的技巧 89 中实现其中的方法。

讨论

首先让我们从文件中读取之前存储的搜索结果。

从文件中读取存储的搜索结果

```
-(NSString*) readResultsFile;
{
    NSArray *paths = NSSearchPathForDirectoriesInDomains
        (NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSString *fileName =
        [NSString stringWithFormat:@"%@/results.json",
            documentsDirectory];
    NSError *error = nil;
    //从文件中读取
```

```
    NSString *content = [[[NSString alloc]
        initWithContentsOfFile:fileName
            usedEncoding:nil error:&error] autorelease];
    if (error) {
        NSLog(@"read error: %@", error);
            return nil;
    }
    return content;
}
```

现在你可以在展示搜索结果前调用 readResultsFile 方法来获取数据。那么就让我们在 numberOfItemsInSection 方法中调用它，我们需要使用 SBJSON 框架，可以从下面的地址下载：http://stig.github.com/json-framework/。请确保将 JSON.h 文件在您的头文件中引入。类似的，请在头文件中声明一个 NSArray 对象，名为 displayItems，同样，务必记住在 dealloc 方法中释放它。

计算并返回 section 中的单元格数量

```
-(NSInteger)collectionView:(UICollectionView *)collectionView
        numberOfItemsInSection:(NSInteger)section;
{
    NSString *resultsJSON = [self readResultsFile];
    //转化为对象
    NSDictionary* resultsDict =
        [resultsJSON JSONValue];
```

```
    if (resultsDict)
    {
        //取得数据
        [displayItems release];
        NSString *key = @"results";
        displayItems = [[resultsDict objectForKey:key] retain];
    }
    return [displayItems count];
}
```

现在你已经读取了之前存储的搜索结果，并将其转化为一个 NSDictionary 对象——resultsDict，同时，你将 resultsDict 中键名为 results 的值赋给了 displayItems 数组，那么 displayItems 中就存储了搜索结果，之后返回此数组的元素数量表示单元格的数量。现在请创建单元格以显示搜索结果。

collectionView:cellForItemAtIndexPath:委托方法将使用之前在 IB 中绘制的 UICollectionViewCell。单元格界面很简单，只有一幅图片和一个文本标签，分别用来显示会议相关图片和会议名称。听上去似乎不难，但开发过程并不像想像中的那么简单。

首先，搜索结果中会包括会议日程（搜索 events 的结果）或会议组织者（搜索 group 的结果），两者的数据格式并不相同。例如，group 有一个图片链接，但是 events 没有。类似的，events 有对应的日期（对于会议日程安排来说非常重要），而 group 没有。那么，就让我们在搜索结果为 group 时显示会议相关图片，events 时显示会议日期。

之前您已经创建好了默认 cell，现在您可以通过单元格标识符将其放入队列中重用。如果队列中一个可以重用的 cell 都没有，系统会为我们创建一个。然后，如果您获得了图片

URL（键名为 photo_url 的值），就显示此网络图片。

自定义单元格

```
//单元格标识符
static NSString *CellIdentifier = @"ResultsCell";
-(UICollectionViewCell *)collectionView:
    (UICollectionView *)collectionView
        cellForItemAtIndexPath:(NSIndexPath *)indexPath;
{
    UICollectionViewCell *cell = [collectionView
        dequeueReusableCellWithReuseIdentifier:CellIdentifier
            forIndexPath:indexPath];
    cell.contentView.backgroundColor = [UIColor darkGrayColor];
    NSDictionary *item =
        [displayItems objectAtIndex:indexPath.row];
    NSString *photoURL = [item objectForKey:@"photo_url"];
    UIImageView *iv = nil;
    UILabel *lbl = nil;

    //遍历 UI 元素
    for (UIView *v in cell.contentView.subviews)
    {
        if ([v isKindOfClass:[UIImageView class]])
```

```
                iv = (UIImageView*)v;
            else if ([v isKindOfClass:[UILabel class]])
            {
                lbl = (UILabel*)v;
            }
        }
        [iv setImage:nil];
        [lbl setText:[item objectForKey:@"name"]];
        if (photoURL)
        {
            if ([photoURL length] > 0)
                [iv setImage:[UIImage imageWithData:
                    [NSData dataWithContentsOfURL:
                        [NSURL URLWithString:photoURL]]]];
        }
        return cell;
    }
```

现在你得到了系统新建或重用队列中的单元格以及需要显示的数据，又取得了键名为 photo_url 的值（暂时不考虑 events），现在可以在单元格中显示按 group 搜索的结果了。

无论是 events 还是 group 都需要在文本标签中设置名称，但对于 group，还需要根据 photoURL 设置图片。现在如果选择搜索 group，你将看到我们想要的搜索结果（见图 11.8）。

图 11.8　搜索 group，显示图片

搜索 events 相比搜索 group 的区别是，显示的不是图片，而是日期。由于日期对于会议日程安排来说非常重要，所以它需要突出显示以引起用户注意，所以我们将把日期显示在图 11.8 中显示图片的位置。

这要求我们做两项额外工作：动态改变单元格的显示内容以及格式化日期。请您创建一个 UILabel 以显示会议日期，并将其作为子视图添加到 UIImageView 上，您还可以根据需要在显示此 UILabel 前对其做一些调整。

现在，如果是一个按 events 搜索的结果，您可以用底部的文本标签显示会议名称（与 group 一样），同时，名称上面的文本标签则显示会议日期，请您通过键名为 time 的值取得会议日期并将其用 NSDateFormatter 类格式化为用户满意的形式。你还可以设置一些 UILabel 的属性，使其看起来更美观。

设置单元格的图像和文本

```
-(UICollectionViewCell *)collectionView:
    (UICollectionView *)collectionView
        cellForItemAtIndexPath:(NSIndexPath *)indexPath;
{
    UICollectionViewCell *cell = [collectionView
        dequeueReusableCellWithReuseIdentifier:CellIdentifier
            forIndexPath:indexPath];
    cell.contentView.backgroundColor = [UIColor darkGrayColor];
    NSDictionary *item =
        [displayItems objectAtIndex:indexPath.row];
    NSString *photoURL = [item objectForKey:@"photo_url"];
    //取得会议时间
    NSDecimalNumber *time = [item objectForKey:@"time"];
    UIImageView *iv = nil;
    UILabel *lbl = nil;

    for (UIView *v in cell.contentView.subviews)
    {
        if ([v isKindOfClass:[UIImageView class]])
        {
            iv = (UIImageView*)v;
            //删除之前的文本
```

```
            [iv.subviews makeObjectsPerformSelector:
                @selector(removeFromSuperview)];
        }
        else if ([v isKindOfClass:[UILabel class]])
        {
            lbl = (UILabel*)v;
        }
    }
    [iv setImage:nil];
    [lbl setText:[item objectForKey:@"name"]];
    if (photoURL)
    {
        if ([photoURL length] > 0)
            [iv setImage:[UIImage imageWithData:
                [NSData dataWithContentsOfURL:
                    [NSURL URLWithString:photoURL]]]];
    }
    else if (time)
    {
        if ([time floatValue] > 0)
        {
            //创建 NSDateFormatter 对象
```

```
if (!dateFormatter)
{
    dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter
        setTimeStyle:NSDateFormatterNoStyle];
    [dateFormatter setDateFormat:@"M/d/YY H:MM"];
}
NSDate *eventDate =
    [NSDate dateWithTimeIntervalSince1970:
        [time floatValue]/1000];
NSString *dateStr =
    [dateFormatter stringFromDate:eventDate];
//设置文本内容
UILabel *lblDate = [[[UILabel alloc]
    initWithFrame:CGRectMake(0, 0, 90, 90)]
        autorelease];
[lblDate setNumberOfLines:2];
[lblDate
    setFont:[UIFont boldSystemFontOfSize:22]];          [lblDate
    setLineBreakMode:NSLineBreakByWordWrapping];
[lblDate
    setTextAlignment:NSTextAlignmentCenter];            [lblDate
```

```
setTextColor:[UIColor whiteColor]];
                    [lblDate setBackgroundColor:[UIColor blackColor]];
                    [lblDate setText:dateStr];
                    //将文本添加到图片上
                    [iv addSubview:lblDate];
                }
            }
            return cell;
        }
```

这里需要注意的是，首先，我们在重用 cell 前需要从表示会议图片的 UIImageView 中删除旧的文本标签，以便显示新的会议日期。其次，dateFormatter 是一个实例变量，请将其在接口中声明并在 dealloc 方法中释放。最后，您还可以查阅 UILabel 的其他属性，将界面调整为您满意的显示效果（见图 11.9）。

图 11.9 搜索 events，显示日期

那么应用基本的功能我们已经做好了，应用首先通过用户输入的搜索信息来查找会议日程和会议组织者，然后将搜索结果存入本地文件并在集合视图上显示，应用可以根据不同情况显示会议相关图片或会议日期。那么，如果现在用户点击屏幕上显示的一项会议，我们该如何做呢？

11.2 通过 Social Framework 分享

地球人都知道现在是社交化时代，所有人都热衷于更新自己的衣食住行轨迹，随时随地都准备发一条 tweets 或者微博。MeetSocial 可以通过 iOS 原生的社交框架——Social Framework 实现将用户的搜索结果分享给好友的功能。

Social Framework 集成了社交网络的账户设置和在应用内访问的功能。如果用户配置好了自己的账户（对于 Twitter 来说，包括用户名和密码），那么应用就可以使用这些信息

帮助用户访问相应的社交网络。这意味着用户不必在应用中再次输入已经配置好的账户信息了，应用通过 Social Framework 帮助用户完成了所有登录过程。在 Social Framework 推出之前，这个过程相当复杂，无论对开发人员还是用户来说，都很不方便。

Social Framework 还为应用提供了访问社交网络的用户界面，由于 Social Framework 帮助我们处理了这么多复杂工作，因此 MeetSocial 应用自身不需要做太多工作，只要指定我们需要分享的内容，并使用 UIKit 框架中的 UIActivityViewController 类将各种分享途径展现给用户，其他的事情就交给 Social Framework 自动去完成吧。

技巧 86　设置分享内容

设置想要分享的内容是很容易的事情，你可以分享会议的名称，举办日期，会议相关图片等，上一节我们已经获取了这些内容。

问题

你想要设置需要分享的内容，主要包括上一节用户搜索结果中的会议日程（events）或会议组织者（group）。

解决方案

在之前创建的集合视图中，我们根据搜索结果的类型展示了不同的数据，因此也应该相应地分享不同的内容，例如，如果搜索结果是 group，那么我们就分享会议图片，如果是 events，则分享会议日期。因此，可以将这些需要分享的内容存入一个数组。

UIActivityViewController 有一个初始化方法，需要用刚才提到的数组作为参数，所以，只需要实现 collectionView:didSelectItemAtIndexPath:委托方法，取得用户的选择项，得到需要分享的内容，再存入数组。至于如何展示分享界面，我们将在下一个技巧中学习。

讨论

正如集合视图中定义单元格的委托方法一样，对于用户选择的单元格，您可以通过查找

键名为“link”的值来判断用户选择的是一个 event 还是一个 group。当然还有其他方法，例如 ,您可以在用户搜索时设置一个标志 ,或者在方法中设置一个值来表示搜索结果的类型，但是这里最简单的方法还是检查键名为“link”的值。

我们的主要目标是通过 Social Framework 分享信息，但是实际上，您在这里能做的并不只有这些，那么，让我们再为用户提供一些其他功能。例如，用户可以打开网页查看更详细的会议信息。好的，我相信您知道该如何实现——group 有一个键名为“link”的 URL，event 有一个键名为“event_url”的 URL，只要对两者区别对待即可（译者：这里根据“点击 UIAlertView 中不同的按钮执行相应的逻辑”这段代码意译，另外，谁能告诉我作者在这里写一个“Busted.”干嘛……看来他自己也被 group，event 绕晕了），这将在下一段代码中再进行介绍。

总结一下，collectionView:didSelectItemAtIndexPath:委托方法可以用来取得用户选择的单元格，并向用户展示可以对所选的会议进行哪些操作。

当用户选择单元格时，展示操作选项

```
-(void)collectionView:(UICollectionView *)collectionView
    didSelectItemAtIndexPath:(NSIndexPath *)indexPath;
{
    [selectedItem release];
    //将用户选择的条目存储到 selectedItem 成员变量中
    selectedItem = [[displayItems objectAtIndex:indexPath.row]
        retain];
    av = [[[UIAlertView alloc] initWithTitle:@"Details"
        message:@"Would you like to..."
```

```
            delegate:self cancelButtonTitle:@"Cancel"
            otherButtonTitles:@"See Website", @"Share", nil]
            autorelease];
    [av show];
}
```

在以上代码中，您取得了用户选择的条目，并将其保存在成员变量 selectedItem 中。请您务必在头文件或.m 文件的匿名类别中声明它，同样，您也需要在 dealloc 方法中释放它，因为本项目并未使用 ARC。另外，您还需要在头文件中添加一个协议声明：UIAlertViewDelegate。

那么，现在用户可以选择上网查看或者分享了，为了区分这两种情况，您将在以下方法中响应用户所点击的不同按钮。

点击 UIAlertView 中不同的按钮执行相应的逻辑

```
-(void)alertView:(UIAlertView *)alertView
    clickedButtonAtIndex:(NSInteger)buttonIndex;
{
    //取消按钮
    if (buttonIndex == 0)
        return;
    //分享按钮
    if (buttonIndex == 2)
        [self share];
    else {
```

```
            NSString *url = [selectedItem objectForKey:@"link"];
            //显示所选会议的网站
            if (url)
                [[UIApplication sharedApplication]
                    openURL:[NSURL URLWithString:url]];
            else {
                NSString *url = [selectedItem
                    objectForKey:@"event_url"];
                [[UIApplication sharedApplication]
                    openURL:[NSURL URLWithString:url]];
            }
        }
    }
```

如果用户点击取消按钮，你只需简单地跳出方法。如果用户点击了分享按钮，则需要在另一个方法中响应该操作（将在技巧 90 中介绍）（译者：其实还是技巧 86……）。如果用户点击了“See Website”按钮，您可以获取相应的 URL 并在浏览器中打开。接下来让我们一起实现分享功能。

使用活动视图控制器分享会议信息

```
-(void)share;
{
    NSString *url = [selectedItem objectForKey:@"link"];
    NSString *textToShare = [selectedItem objectForKey:@"name"];
```

```
UIImage *imageToShare = nil;
NSArray *activityItems = nil;
if (url)
{
    NSString *photoURL = [selectedItem
        objectForKey:@"photo_url"];
    imageToShare = [UIImage imageWithData:
        [NSData dataWithContentsOfURL:
            [NSURL URLWithString:photoURL]]];
    //分享图片
    activityItems = @[textToShare, url, imageToShare];
}
else
{
    url = [selectedItem objectForKey:@"event_url"];
    NSDecimalNumber *time =
        [selectedItem objectForKey:@"time"];
    NSDate *eventDate =
        [[NSDate dateWithTimeIntervalSince1970:
            [time floatValue]/1000] autorelease];
    NSString *dateStr =
        [dateFormatter stringFromDate:eventDate];
```

```
        //分享日期
        activityItems = @[textToShare, url, dateStr];
    }
    //传入数组作为初始化参数
    UIActivityViewController *activityVC =
        [[[UIActivityViewController alloc]
            initWithActivityItems:activityItems
                applicationActivities:nil] autorelease];
    [self presentViewController:activityVC
        animated:YES completion:nil];
}
```

就像 collectionView:didSelectItemAtIndexPath:委托方法一样，您通过检查键名为“link”的值来判断选择项是一个 group 还是 event，在这两种情况下，您构建了不同的 activityItems 数组。那么现在就让我们向用户展示分享界面。

技巧 86　使用活动视图控制器展示分享界面

在技巧 89 中（译者：请读者自己根据目录对技巧进行排序吧……）你确定了需要分享的内容，现在可以开发分享功能了。Social Framework 提供了基本的功能框架，UIKit 提供了分享界面视图，我们只需要做一点额外的数据工作就可以了。

问题

你需要在用户选择分享会议信息后展示应用提供的分享途径，理想状态下，用户应该无需繁琐步骤就能通过多种途径分享内容。

解决方案

你将使用 activityItems 数组来创建活动视图控制器。它将把需要分享的内容传递给 UIActivityViewController。对于其他操作就全部交给 Social Framework 吧。

讨论

使用 activityItems 数组实例化了 UIActivityViewController 并将其推入主屏幕（见图 11.10）。注意，这里我们简单地为 completion:参数传入 nil，你可以定制自己的 block 以便在其离开屏幕时执行需要的操作。

图 11.10 分享界面

用户将在分享界面看到自己待分享的内容，注意，这里由于分享内容包括一张图片，所以视图中有一个操作选项是将其存入图片库。接下来，如果我们点击 Facebook 按钮，那么您会看到有一个分享框弹出来，用户可以在这里将图片分享到 Facebook（见图 11.11）。

图 11.11　发送 Facebook 动态

既然你获取到了会议日期 ,那么我们可以根据此日期向日历应用中插入一条日程安排并在提醒事项应用中提醒用户。那么下面我们就来看看该如何实现吧。

11.3　在内置的日历应用中创建日程安排

我们可以通过 EventKit 框架轻松创建日程安排，或对现有日程安排进行搜索、编辑、展示等操作。EventKit 甚至提供了创建流畅用户体验所需要的用户界面，无论对开发者还是用户来说都十分方便。

对于 MeetSocial 来说 ,每个 event 都带有日期属性 ,因此让我们为应用开发一项功能，以便用户可以根据会议日期在内置的日历应用中创建日程安排。

技巧 86　创建 EKEventStore 对象并插入一条日程安排

EventKit 框架提供了需要创建和管理日程安排的大量功能，你可以轻松地将它集成进

应用程序中，那么现在就让我们一起学习如何它。

问题

你想让用户在选择一个 event 后，可以创建一条日程安排。

解决方案

您将在应用中集成 EventKit，并取得 event 的会议日期和时间，之后您可以使用 EKEventStore 创建一条日程安排。

对日程表进行某些操作可能会增加程序复杂度，例如计算日期，重复事件等，我们在这里只涉及到一些基础功能，如果你打算开发日程功能更复杂的程序，你将在下面的章节中学到更多的关于 EventKit 的技巧，这些技巧是很好的入门基础。

讨论

首先您需要将 EventKit 和 EventKitUI 加到项目的 Build Phases 中（见图 11.12）。接下来在使用了 EventKit 框架的头文件中引入 EventKit/EventKit.h 和 EventKitUI/EventKitUI.h。

图 11.12 添加 EventKit 框架

现在您可以向用户展示新功能了。

在 UIAlertView 按钮中增加有关 EventKit 的功能

```
-(void)collectionView:(UICollectionView *)collectionView
    didSelectItemAtIndexPath:(NSIndexPath *)indexPath;
{
    [selectedItem release];
    selectedItem = [[displayItems
        objectAtIndex:indexPath.row] retain];
    NSString *url = [selectedItem objectForKey:@"link"];
    UIAlertView *av = nil;
    if (url)
        av = [[[UIAlertView alloc] initWithTitle:@"Details"
            message:@"Would you like to..."
            delegate:self
            cancelButtonTitle:@"Cancel"
            otherButtonTitles:@"See Group Page",
            @"Share", nil] autorelease];
    else
        //添加了新按钮
        av = [[[UIAlertView alloc] initWithTitle:@"Details"
            message:@"Would you like to..."
```

```
                delegate:self
                cancelButtonTitle:@"Cancel"
                otherButtonTitles:@"See Event Page",
                @"Share",
                @"Create Cal Item",
                @"Create Reminder", nil] autorelease];
        [av show];
    }
```

现在用户有了两个新功能：一个是创建日程安排，一个是创建提醒事项（见下一个技巧）。您需要修改 alertView:clickedButtonAtIndex:方法，当前方法中的 else 假定用户想浏览关于该 event 的网页，现在请将其改成检查用户点击的按钮并作出相应处理。

在此我们为 else 语句添加了新内容，用来处理新按钮的逻辑。注意，我们将之前所有的 else 代码移动到了 if (buttonIndex == 1)中。剩下的模块改为处理日程安排和提醒事项。首先让我们看看如何处理日程安排。

选择打开 URL 或创建日程安排

```
    if (buttonIndex == 1)
    {
        NSString *url = [selectedItem objectForKey:@"event_url"];
        [[UIApplication sharedApplication] openURL:
            [NSURL URLWithString:url]];
    }
    else
```

```
{
    NSDecimalNumber *time = [selectedItem objectForKey:@"time"];
    NSDate *eventDate = [[NSDate dateWithTimeIntervalSince1970:
        [time floatValue]/1000] autorelease];
    //创建 EKEventStore
    EKEventStore *eStore = [[[EKEventStore alloc]
        initWithAccessToEntityTypes:
            EKEntityMaskReminder | EKEntityMaskEvent]
                autorelease];
    NSError *error = nil;
    if (buttonIndex == 3)
    {
        //如果得到了用户的许可，就创建此项日程安排
        if ([self isAuthorizedForEntityType:EKEntityTypeEvent])
            [self createCalItem];
    }
}
```

在这个方法中你可以看到，创建一个日程安排并不需要很多的代码。为了创建日程安排，首先检查应用是否已经被授权访问用户日历，如果是，就调用 createCalItem 方法创建它。让我们一起看看请求访问用户日历权限的代码。

检查是否允许访问用户日历

```
-(bool)isAuthorizedForEntityType:(EKEntityType)type;
```

```
{
    //检查是否允许访问用户日历
    EKAuthorizationStatus authStatus =
        [EKEventStore authorizationStatusForEntityType:type];
    if (authStatus != EKAuthorizationStatusAuthorized)
    {
        [[EKEventStore alloc]
            requestAccessToEntityType:EKEntityTypeEvent
                completion:^(BOOL granted, NSError *error)
                {
                    //如果得到了授权，就创建此项日程安排
                    if (granted)
                    {
                        [self createCalItem];
                    }
                    else
                    {
                        //如果没有得到授权，就提示用户给应用授权
                        UIAlertView *av = [[[UIAlertView alloc]
                            initWithTitle:@"Permissions"
                                message:
                                    @"If you deny the app permissions,
```

```
                                                you can not \
                                                create calendar events.\n
                                                \nYou can \
                                                change your permissiongs in the
                                                Settings \
                                                app under Privacy."
                                            delegate:nil
                                            cancelButtonTitle:nil
                                            otherButtonTitles:@"OK", nil]
                                                autorelease];
                                    [av show];
                                }
                            }];
                    return NO;
                }
                else
                    //已经得到了授权
                    return YES;
            }
```

此方法接受一个 EKEntityType 类型的对象作为参数，并检查系统偏好设置中是否允许应用程序访问用户日历。在应用第一次检查授权状态时，EKAuthorizationStatus 对象的值是 EKAuthorizationStatusNotDetermined——这是一个枚举值。

如果用户在启动应用前就进入系统偏好设置中对应用授权，那么方法直接进入 else 语句，并返回 YES。如果用户允许应用访问日历，就调用 createCalItem 方法。如果 EKAuthorizationStatus 对象的值不是 EKAuthorizationStatusAuthorized，则需要请求用户授权。

如果用户允许应用访问日历，就调用 createCalItem 方法创建一条日程安排。否则，显示一条信息提示用户。让我们看看 createCalItem 方法是如何创建一条日程安排的。

创建一条日程安排并展示其编辑界面

```
-(void)createCalItem;
{
    EKEventStore *eStore = [[[EKEventStore alloc] init]
        autorelease];
    //创建一个事件
    EKEvent *event =
        [EKEvent eventWithEventStore:eStore];
    [event setCalendar:[eStore defaultCalendarForNewEvents]];
    [event setTitle:[selectedItem objectForKey:@"name"]];
    NSDecimalNumber *time = [selectedItem objectForKey:@"time"];
    NSDate *eventDate = [NSDate dateWithTimeIntervalSince1970:
        [time floatValue]/1000];
    [event setStartDate:eventDate];
    [event setEndDate:eventDate];
    NSError *error = nil;
```

```
        //存储事件
        [eStore saveEvent:event span:EKSpanThisEvent
            commit:YES error:&error];
        if (error)
            NSLog(@"Saving createCalItem: %@", error);
        EKEventEditViewController *editEvent =
            [[[EKEventEditViewController alloc] init] autorelease];
        [editEvent setEditViewDelegate:self];
        [editEvent setEvent:event];
        //显示事件并让用户进一步编辑
        [self presentViewController:editEvent
            animated:YES completion:nil];
    }
```

在 createCalItem 方法中，你创建了一个 EKEventStore 对象用来访问用户日历。之后创建了一条日程安排，并设置了它的日程，标题，起止日期。然后我们保存了此日程安排。最后，我们展示了一个内建视图，让用户对此日程安排做进一步修改。

你或许认为这里不需要向用户展示修改界面，但是我们为什么要在此让用户修改呢，因为他们可能想增加更多细节信息。下面让我们学习如何创建一条提醒事项。

技巧 86　创建一条提醒事项

在上一个技巧中你学习了如何在 UIAlertView 按钮中增加有关 EventKit 的功能，同时，还学习了如何创建 EKEventStore 对象并插入一条日程安排。

我们在 UIAlertView 按钮中增加了创建提醒事项的功能，可是还没有实现它。下面就

让我们看看如何创建一条提醒事项。

问题

你想让用户在选择一个 event 后，可以创建一条提醒事项，就像上一个技巧中创建日程安排一样。

解决方案

同样需要使用 EKEventStore，但是这里是创建一条提醒事项。与创建日程安排不同，这里我们不需要展示修改提醒事项的界面，只需要提示用户创建完成即可。

讨论

上一个技巧中的最后一段代码展示了点击序号为 3 的按钮（Create Cal Item）时所发生的情况，它将调用 createCalItem 方法创建一条日程安排，现在让我们为其添加 else 语句，用来创建一条提醒事项。

增加 else 语句，检查授权并创建提醒事项

```
if (buttonIndex == 3)
{
    if ([self isAuthorizedForEntityType:EKEntityTypeEvent])
        [self createCalItem];
}
else
{
    if ([self isAuthorizedForEntityType:EKEntityTypeReminder])
        //创建一个新提醒
        [self createReminderItem];
```

```
}
```

就像前一个技巧一样，需要检查用户是否允许程序创建提醒事项。如果用户允许，我们就调用 createReminderItem 方法创建一条新提醒事项。

由于 isAuthorizedForEntityType 已经写好了，现在只需要调用它并修改传入的参数就可以判断用户是否允许程序创建提醒事项。

根据 EKEntityType 来创建日程安排或提醒事项

```
if (granted)
{
    if (type == EKEntityTypeEvent)
        [self createCalItem];
    else
        //已经得到授权
        [self createReminderItem];
}
```

如果用户已经授权，应用就根据 EKEntityType 来创建日程安排或提醒事项。如果 EKEntityType 的值不是 EKEntityTypeEvent，就调用名为 createReminderItem 的新方法来创建一条提醒事项。

创建一条提醒事项

```
-(void)createReminderItem;
{
    EKEventStore *eStore = [[[EKEventStore alloc] init]
        autorelease];
```

```
NSDecimalNumber *time = [selectedItem objectForKey:@"time"];
NSDate *eventDate = [NSDate dateWithTimeIntervalSince1970:
    [time floatValue]/1000];
//创建提醒事项
EKReminder *reminder =
    [EKReminder reminderWithEventStore:eStore];
NSCalendar *gregorian = [[[NSCalendar alloc]
    initWithCalendarIdentifier:
        NSGregorianCalendar] retain];
NSDateComponents *comps = [gregorian components:
    (NSDayCalendarUnit | NSMonthCalendarUnit
        | NSYearCalendarUnit)
            fromDate:eventDate];
[comps setDay:[comps day]];
[comps setMonth:[comps month]];
[comps setYear:[comps year]];
//设置提醒事项信息
[reminder setCalendar:
    [eStore defaultCalendarForNewReminders]];
[reminder setTitle:[selectedItem objectForKey:@"name"]];
[reminder setDueDateComponents:comps];
NSError *error = nil;
```

```
        //存储提醒事项
        [eStore saveReminder:reminder
              commit:YES error:&error];
        if (error)
              NSLog(@"Saving createCalItem: %@", error);
        UIAlertView *av = [[[UIAlertView alloc]
              initWithTitle:@"Reminder"
                    message:@"Reminder created!"
                    delegate:nil
                    cancelButtonTitle:nil
                    otherButtonTitles:@"OK", nil]
                          autorelease];
        [av show];
    }
```

与创建日程安排类似，你需要创建一个 EKEventStore 对象，再创建提醒事项，设置相关信息，最后通过 EKEventStore 对象存储到提醒事项应用中。但是这里并没有像创建日程安排一样，向用户展示一个编辑界面，我们只是向用户显示一条信息，告诉用户提醒事项已经创建完毕。

我们处理了一大堆针对用户界面的琐事，包括集合视图和社交框架。你现在可能正在使用应用浏览大量的会议信息或者一遍一遍地搜索自己感兴趣的内容。那么，我们如何在退出应用后，下一次打开应用时，还能看到之前浏览的内容呢？

11.4 保存和恢复应用状态

保存和恢复应用状态是 iOS 6 引入的一项新功能。您可以自己处理有关保存和恢复的细节，但是现在使用系统自带的 API 可以更加简单地实现这项功能，甚至大部分操作是全自动的，减少了开发人员的很多工作。

对于 MeetSocial 来说，您在服务器上查找感兴趣的内容并将其存入本地文件，但是，如果在查询到结果后您终止了应用或者打开一个其他应用 ,甚至是在不知道的情况下被系统从内存中清理出去，你必须重新选择搜索内容，输入搜索关键词，消耗流量连接服务器，这极大的降低了用户体验。为了给用户提供一个仿佛不会中断的工作流，你可以实现 UIStateRestoration 协议中的方法 ,它包含在 UIKit 框架中。至于其他细节工作 ,就交给 iOS 帮你完成吧。

技巧 86　为应用添加保存/恢复状态的功能

首先需要告诉 iOS，MeetSocial 应用需要保存自己的状态，并在用户重新打开或返回应用时恢复之前的状态，对此只需要实现两个简单的方法。

同时，由于我们已经告诉 iOS 本应用具有恢复功能，那么 iOS 将在尝试恢复应用状态时调用一个方法，因此你也需要实现它。

问题

你想让 MeetSocial 在用户重新打开或返回应用时恢复之前的状态，那么，你需要在应用程序委托中声明两个方法，用来添加上述功能。

解决方案

在 MSAppDelegate（译者：原文是 MESAppDelegate，后文中包括像 MENavCon，MESearchInput 等，我都根据作者所附源码依次修正了过来，如果读者拷贝英文原版的代码运行不通过，请直接下载源码学习）中，你需要实现 UIStateRestoration 协议中的方法，

为应用添加保存/恢复状态的功能。

同时，你还需要实现 application:viewControllerWithRestoration-IdentifierPath:coder:方法，以便在系统尝试恢复应用状态时返回带有恢复功能的视图控制器。

讨论

你需要实现两个方法，方法体很简单，只需要返回 YES 即可。

为应用添加保存/恢复状态的功能

```
-(BOOL) application:(UIApplication *)application
    shouldSaveApplicationState:(NSCoder *)coder
{
    return YES;
}
-(BOOL) application:(UIApplication *)application
    shouldRestoreApplicationState:(NSCoder *)coder
{
    return YES;
}
```

方法很简单，不过它们确实也没有做太多的事情，它们只是告诉 iOS 在需要的时候调用相应的方法去保存或恢复应用状态，至于这些方法有没有实现，它们就不知道了。下面我们就来讨论其中一个方法，它在系统尝试恢复应用状态时返回各个视图控制器的实例。

根据 Restoration ID 恢复视图控制器的状态

```
-(UIViewController *) application:(UIApplication *)application
```

```
    viewControllerWithRestorationIdentifierPath:
        (NSArray *)identifierComponents
            coder:(NSCoder *)coder {
    UIViewController* theController = nil;
    NSString* lastID = [identifierComponents lastObject];
    if (!lastID) return nil;
    if ([lastID isEqualToString:@"MENavCon"])
        theController = _window.rootViewController;
    else if ([lastID isEqualToString:@"MESearchInput"])
        theController = [[(UINavigationController*)
    _window.rootViewController
        viewControllers] objectAtIndex:0];
    return theController;
}
```

这里您需要注意 identifierComponents 数组。它表示一个恢复树，包含根级的父视图控制器和叶子级的子视图控制器。因此，当您的应用中有一个复杂的导航控制器时，仅仅知道 restoration ID 是不够的，还需要知道此恢复树的层级结构，以便得到正确的视图控制器并得到它的对象。

同时还需要注意 restoration ID，我们从哪里得到它？实际上每个视图控制器的 restoration ID 是由你来决定的。MSNavCon 是应用的导航控制器，MSSearchInput 是得到用户输入的主视图控制器。由于这两个视图控制器在应用启动时便需要加载，因此你在 MSAppDelegate.m 中就需要实现这个方法。但是如何让 iOS 知道这些 restoration ID 分

别对应哪一个视图控制器呢？让我们一起看看如何在 Storyboard 中进行设置。

技巧 86　设置 restoration ID

设置 restoration ID 可以让 iOS 在尝试恢复应用状态时找到正确的对象（例如视图控制器），这样应用就可以正确地恢复成用户之前看到的样子。

问题

您需要为应用中的视图控制器设置 restoration ID，以便系统可以正确地找到相应的视图控制器。

解决方案

您可以在 UI 编辑区的 Attributes Inspector 中设置应用的导航控制器和主视图控制器所对应的 restoration ID（见图 11.13）。

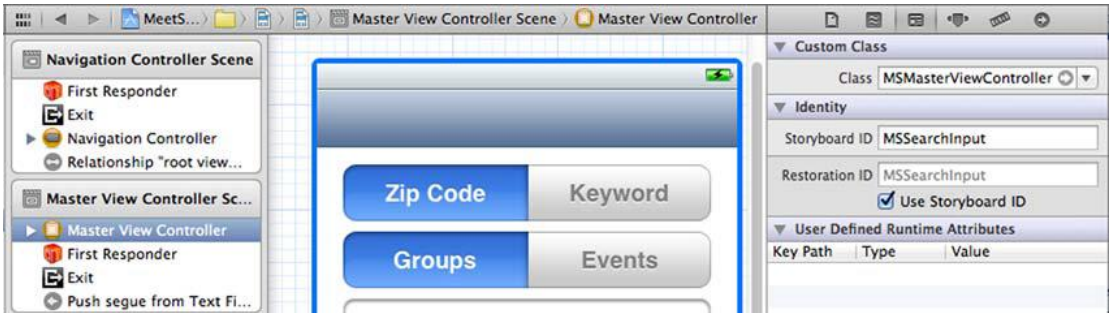


图 11.13　设置 restoration ID

讨论

请设置导航控制器的 restoration ID 为 MSNavCon，主视图控制器的 restoration ID 为 MSSearchInput。你可以使用 Storyboard ID 作为 restoration ID，也可以分别指定两者，在这里我想向你展示如果将两者设置成相同的，程序该如何编写。

对于详细视图控制器，请您设置 Storyboard ID 为 MSResults，并勾选 Use Storyboard ID。之后您会看到这么做的好处，与前两个视图控制器不同，它并不是在应用启动时就加

载的，如果将两个 ID 设置成相同的，系统在尝试恢复应用状态时可以直接传入 ID 在 Storyboard 中找到正确的待恢复对象，相对来说更加方便。

技巧 86 指定恢复类

对于像详细视图控制器这类不是在应用启动时就加载的对象，您需要指定它们的恢复类（restoration classes）。恢复类在系统尝试恢复应用状态时负责创建需要恢复的对象。

视图控制器的恢复类需要实现 UIViewControllerRestoration 协议，它有一个方法，返回一个待恢复视图控制器。

问题

你需要为详细视图控制器指定恢复类，因为它不会在应用启动时就加载。这个恢复类需要创建并返回一个详细视图控制器的对象，它的状态应该和上一次应用中断时的状态保持一致。

解决方案

首先我们让详细视图控制器实现 UIViewControllerRestoration 协议，并指定它的恢复类就是它自己。接下来请在 viewDidLoad:委托中添加如下代码：

```
self.restorationClass = [self class];
```

同时您还需要添加 UIViewControllerRestoration 协议的方法：

```
+(UIViewController *)
    viewControllerWithRestorationIdentifierPath:
        (NSArray *)identifierComponents coder:(NSCoder *)coder;
```

讨论

请注意传入的两个参数：identifierComponents 和 coder。我们可以通过 identifierComponents 找到对应的视图控制器。还记得之前我们将详细视图控制器的

Storyboard ID 设置为 MSResults 了吗？同时我们也勾选了 Use Storyboard ID，也就是说 restoration ID 也是 MSResults。那么现在我们可以使用 MSResults 找到待恢复的视图控制器，并在 Storyboard 中实例化相应的对象。

因此，你现在只需实例化详细视图控制器并返回它即可，由于您想从 Storyboard 加载它，因此我们可以从 coder 参数中获取 Storyboard 文件并根据 Storyboard ID 找到相应的对象。

从 Storyboard 中加载待恢复的视图控制器

```
+(UIViewController*) viewControllerWithRestorationIdentifierPath:
        (NSArray *)identifierComponents coder:(NSCoder *)coder;
{
        UIStoryboard* sb = [coder decodeObjectForKey:
        UIStateRestorationViewControllerStoryboardKey];
        NSString* lastID = [identifierComponents lastObject];
        if (sb)
            return (MSDetailViewController*)
        [sb instantiateViewControllerWithIdentifier:lastID];
        return nil;
}
```

在系统尝试恢复应用状态时，我们通常假定返回 nil 是安全的。在某些情况下我们需要返回 nil，例如，数据可能因为某种原因过期而必须重新加载，因此系统无需恢复到这种状态。

现在你已经创建了所有需要恢复的视图控制器，但是还没有恢复到应用中断时的状态。

还没有设置详细视图控制器中 UIKit 组件的状态值，那么下面就让我们一起将这些组件也恢复到上次用户操作的状态。

技巧 86 归档/解档 UIKit 组件的状态值

现在你已经恢复了视图控制器，但是并没有恢复到应用中断时的状态。你需要在用户退出应用或应用偶然被系统终止的情况下恢复应用界面数据，让用户觉得好像什么都没发生一样。如果用户选择了搜索内容或得到了搜索结果，你应该让用户回到相应的界面。

问题

你想把界面所有组件都恢复到应用中断时的状态，而不仅仅是创建视图控制器对象，因此你需要存储所有界面组件的状态值，以便在系统尝试恢复应用状态时，将这些状态值再还原到界面组件上。

解决方案

系统在尝试恢复应用状态时会调用 UIViewController 的两个方法，现在你已经声明过本应用带有恢复功能，因此也需要实现这两个方法。在系统存储应用状态时，会调用 encodeRestorableStateWithCoder:方法，它接受一个 NSCoder 类型的参数，用来存储状态值，类似的，在系统恢复应用状态时，会调用 decodeRestorableState-WithCoder:方法，同样接受一个 NSCoder 类型的参数，用来恢复状态值。

讨论

为了恢复上一次应用中断时用户在搜索框中填写的关键词，让我们一起来实现上述的两个方法。

将待恢复的状态值存入 NSCoder

```
#define kSearchGorE @"SearchGorE"

#define kSearchZorK @"SearchZorK"
```

```
#define kSearchText @"SearchText"

-(void)encodeRestorableStateWithCoder:(NSCoder *)coder

{

    [super encodeRestorableStateWithCoder:coder];

    [coder encodeInt:segSearchGroupsOrEvents.selectedSegmentIndex

        forKey:kSearchGorE];

    [coder encodeInt:segSearchZipOrKeyword.selectedSegmentIndex

        forKey:kSearchZorK];

    [coder encodeObject:tfSearchText.text

        forKey:kSearchText];

}
```

对于主视图控制器来说，你希望恢复用户输入的搜索关键词和两个搜索选项。

通过 NSCoder 恢复状态值

```
-(void)decodeRestorableStateWithCoder:(NSCoder *)coder

{

    [super decodeRestorableStateWithCoder:coder];

    [segSearchGroupsOrEvents setSelectedSegmentIndex:

        [coder decodeIntegerForKey:kSearchGorE]];

    [segSearchZipOrKeyword setSelectedSegmentIndex:

        [coder decodeIntegerForKey:kSearchZorK]];

    [tfSearchText setText:

        [coder decodeObjectForKey:kSearchText]];
```

}

无论哪种情况，都必须调用父类的对应方法，这样 iOS 就可以首先恢复父类，为正确地恢复子类做好铺垫。

那么，为什么我们没有对详细视图控制器也这么做呢？这是因为它的界面组件没有用户输入的部分（但是，如果用户在应用中断时正在分享会议信息，就可能会出现用户正在输入数据的情况，不过对本应用而言，我们不会考虑的如此复杂）。同时请你务必记住，详细视图控制器从一个文件中加载搜索结果，即使应用中断，文件却是不会被删除的，因此详细视图控制器将再次读取文件并显示搜索结果，之前我们特意将其实现为无需联网就可以显示搜索结果。

但是如果应用中断时用户正在浏览集合视图该如何做呢？用户可能正在上下滑动集合视图，理想的做法是让用户再次返回应用时，还能看见与上一次相同的数据。

技巧 86　恢复集合视图数据源

恢复应用状态时，您一定希望整个界面都恢复成用户上一次看到的样子，有时，我们需要为此付出一些额外努力。滑动视图可以记住自己上次被用户所滑动到的位置，但是它的子类：列表视图和集合视图，在应用恢复时，用户在上一次滑动到的位置并不一定能看到与上次相同的数据。

也就是说，虽然您想在应用恢复时，让集合视图显示与用户上一次浏览时相同的数据，但实际上，系统无法保证每一条数据的 NSIndexPath 都与之前相同，这意味着每一条数据的位置都可能与之前不一样。为此 iOS 提供了 UIDataSourceModelAssociation 协议帮助您解决这个问题，它可以存储或恢复单元格的 NSIndexPath。如果您想要正确地恢复表视图或集合视图，务必记住实现这个协议。

问题

当用户再次返回应用时，你想让集合视图显示的数据和用户上一次浏览时相同。

解决方案

你需要实现 UIDataSourceModelAssociation 协议，标识每一条用户上一次看到的数据。

在恢复应用时，系统根据恢复标识找到每一条数据在上一次应用退出前的 NSIndexPath，由此将对应的数据显示在正确的单元格中。

讨论

UIDataSourceModelAssociation 协议需要实现两个方法，第一个方法是根据 NSIndexPath 为用户上一次看到的数据定义恢复标识。

根据 NSIndexPath 为每一条数据定义一个恢复标识

```
-(NSString *) modelIdentifierForElementAtIndexPath:
    (NSIndexPath*)idx inView:(UIView *)view;
{
    NSDictionary *item = [displayItems objectAtIndex:idx.row];
    return [NSString stringWithFormat:@"%@",
        [item objectForKey:@"id"]];
}
```

对于传入的 NSIndexPath，你可以取得与其关联的单元格（其中的数据包括 event 或 group），再使用 ID 标识并存储。这样，就可以在恢复应用时，根据 ID 找到对应的单元格并正确地显示在集合视图上。

返回上一次应用退出前各个单元格的 NSIndexPath

```
-(NSIndexPath *) indexPathForElementWithModelIdentifier:
```

```
        (NSString *)identifier
              inView:(UIView *)view;
{
        int cnt = 0;
        for (NSDictionary *item in displayItems)
        {
              if ([[item objectForKey:@"id"]
                    isEqualToString:identifier])
                          return [NSIndexPath indexPathForRow:cnt
                                inSection:0];
              cnt++;
        }
        return nil;
}
```

因为我们的单元格只有一个 section，所以我们为 inSection:参数传入 0。同时，row 相对应于 displayItems 数组的下标。这样，您就可以遍历数组，根据前一段代码设置的 ID 查找对应的单元格，并返回该单元格在上一次应用退出前的 NSIndexPath。

现在系统可以根据您返回的 NSIndexPath 值找到正确的单元格并显示在集合视图中了。

如果单元格中的数据只是一个简单的字符串，那么您可以直接在 UIDataSource-ModelAssociation 协议第一个方法中返回该字符串，第二个方法中返回该字符串所在数组的下标。根据您需要显示的数据，我们将选择不同的方式去恢复。但是，如

果我们的数据内容从一个版本变动到另一个版本 ,我们应当如何存储并恢复正确的版本呢？

技巧 86 存储和恢复应用数据版本信息

有时，对于一些全局性的数据，可能并不直接关联到一个特定的视图或视图控制器，您可能想要储存一些值，但这些需要存储的数据可能会涉及到应用的多个模块，你希望让它们处于保护和恢复期时也可以访问。

UIApplicationDelegate 协议中包含两个方法，可以帮助您完成这个任务。

问题

你希望在保存应用状态时存储一些重要数据，以便之后恢复应用时使用。

解决方案

和视图控制器类似，应用程序委托也有一对归档/解档方法，系统在尝试保存或恢复应用状态时将调用这两个方法。同样，这两个方法也带有一个 NSCoder 类型的参数，数据通过它正确地实现存储或恢复。

讨论

这两个方法实现起来非常简单，让我们一起来看一个示例实现。

归档/解档在应用状态恢复后需要访问的数据

```
-(void)application:(UIApplication *)application
      willEncodeRestorableStateWithCoder:(NSCoder *)coder
{
      [coder encodeFloat:1.0 forKey:@"MSVersion"];
}


-(void)application:(UIApplication *)application
```

```
    didDecodeRestorableStateWithCoder:(NSCoder *)coder
{
    float curVer = [[NSUserDefaults standardUserDefaults]
    floatForKey:@"MSVersion"];
    float lastVer = [coder decodeFloatForKey:@"MSVersion"];
}
```

在归档方法中，你归档了现有数据，它将作为该数据的一个版本保存起来。通常情况下，之后将在应用其他模块中访问该数据。这个方法将在其他类的恢复工作进行前调用，因此你可以在这里设置一些重要数据或者为恢复其他类做一些准备性工作。

相反，在解档方法中，这个方法将在其他恢复工作完成后调用，因此，您可以在这里保存恢复后的数据或者用户当前偏好设置。在这个例子中，我尝试保存和恢复一条假设的数据 MSVersion，在归档方法中，该数据被保存为恢复前版本，在解档方法中，该数据的恢复前版本被取出，并将其与当前版本（恢复后版本）比较。之后可以在应用中根据需要对数据版本信息进行回滚和升级。

11.5 总结

在本章中，我们学习了如何开发一个 meetup.com 客户端，它可以根据用户需要搜索相应的会议信息，还可以将搜索结果保存到本地。在集合视图中展示搜索结果，使用 Social Framework 分享感兴趣的会议信息，使用 EventKit 创建日程安排和提醒事项，最后还包括保存和恢复应用状态。

这些技巧可以在大部分项目中使用，它们可以帮助你创建富有吸引力的应用。