

Hello World!

一步一步学习 iOS 5 编程

EntLib.com Team 翻译整理

联系方式：EntLib@hotmail.com

官方网站：<http://www.EntLib.com> & <http://www.EntLib.net>



捐助 EntLib.com 团队！



<http://me.alipay.com/entlib>

EntLib.com 团队对您的支持表示衷心感谢，捐助资金将主要用于支付 EntLib.com 和 EntLib.net 服务器以及带宽的费用，我们将不断努力给您提供更好的免费产品和服务。



目 录

捐助 EntLib.com 团队！	2
--------------------------------	----------

第一部分：Hello World！创建你的第一个 iPhone App.....	7
---	----------

看看你的第一个应用程序	8
-------------------	---

开始编写代码！	8
---------------	---

熟悉 Xcode 工作环境.....	13
--------------------	----

第一次运行你的应用程序	14
-------------------	----

回来写代码吧！	16
---------------	----

编写 Hello World 按钮的代码	18
----------------------------	----

连接 Hello World 按钮和 Action 方法.....	20
-----------------------------------	----

测试你的应用程序.....	21
---------------	----

第二部分：iOS 编程基础：Hello World 应用程序如何工作的？ ...	24
---	-----------

Interface Builder、头文件和实现文件	25
----------------------------------	----

触摸和点击的背后机制.....	28
-----------------	----

Run 按钮幕后机制.....	30
-----------------	----

有任何问题么？	32
---------------	----

第三部分：iOS 编程向导：创建一个简单的表视图 (Table View) 应用程序	33
---	-----------

创建 SimpleTable 项目	34
-------------------------	----

设计视图.....	37
第一次运行你的应用程序	38
添加表数据	39
UITableViewDelegate 和 UITableViewDataSource	40
数组是什么？	42
连接数据源（DataSource）和委托（Delegate）	47
测试你的应用程序.....	49
在表视图中添加缩略图.....	50
接下来是什么？	53
<u>第四部分：定制 UITableView 表视图单元格</u>	<u>55</u>
显示不同的缩略图.....	55
[thumbnails objectAtIndex:indexPath.row] 有什么用途？	59
定制表视图单元格	60
设计单元格	61
为定制单元格创建类.....	68
Property 和 Outlet.....	70
@synthesize 指令.....	71
建立连接.....	71
更新 SimpleTableViewController	73
留给你的作业	77
接下来讲什么？	79

第五部分：如何处理 UITableView 中的行选择.....81

理解 UITableViewDelegate82

处理表视图行选择（ Table Row Selection ）83

开始编码吧！85

布置给你的作业88

接下来是什么？93

第六部分：应用 Property List 强化你的 Simple Table 应用程序 .95

为什么外部化表数据？95

Property List 是什么？97

这是存放表数据的最好方法吗？97

转换表数据为 Property List98

在 Objective-C 中加载 Property List..... 103

代码修改的解释 104

接下来介绍什么？ 106

第七部分：如何在 Xcode 4.* 添加定制字体到 iOS 应用程序中？**..... 108**

查找字体文件所在的位置 108

复制字体文件到应用程序包（ Application Bundle ） 109

编辑 App 中的 info.plist 文件 110

准备使用新添加的字体..... 111

第八部分：如何在 iOS App 中添加启动画面？ 113**什么是启动画面（ Splash Screen ）？ 113****在你的 App 中添加启动画面 114****准备你的启动画面 114****在 Xcode 中添加你的启动图像 116****开始测试！ 117****更进一步信息 118**

第一部分：Hello World！创建你的第一个 iPhone App

首先，你需要安装好 Xcode，配置好你的开发环境。如果你不知道如何安装，请参考文章 – [了解并准备 iOS 编程所需要的条件和环境](#)。

如果之前你看过任何编程书籍，你就应该听说过 Hello World 程序。它成为初学者创建第一个应用程序的惯例。Hello World 是一个非常简单的程序，通常用来输出 Hello World 在屏幕上。本文也将遵循这一编程惯例，使用 Xcode 创建 Hello World 应用程序。尽管比较简单，Hello World 程序实现如下几个目标：

- 让你更好地理解 iOS 的编程语言 Objective-C 的语法和结构；
- 让你初步了解 Xcode 编程环境，了解如何创建一个 Xcode 项目和使用内置的 interface builder 创建用户界面 (user interface)；
- 学习如何编译程序，构建 app 和在模拟器中测试；
- 最后，让你明白编程并不困难，不必害怕；

看看你的第一个应用程序

在开始编码之前，我们看看 Hello World 应用程序运行效果。最终效果如下图所示：

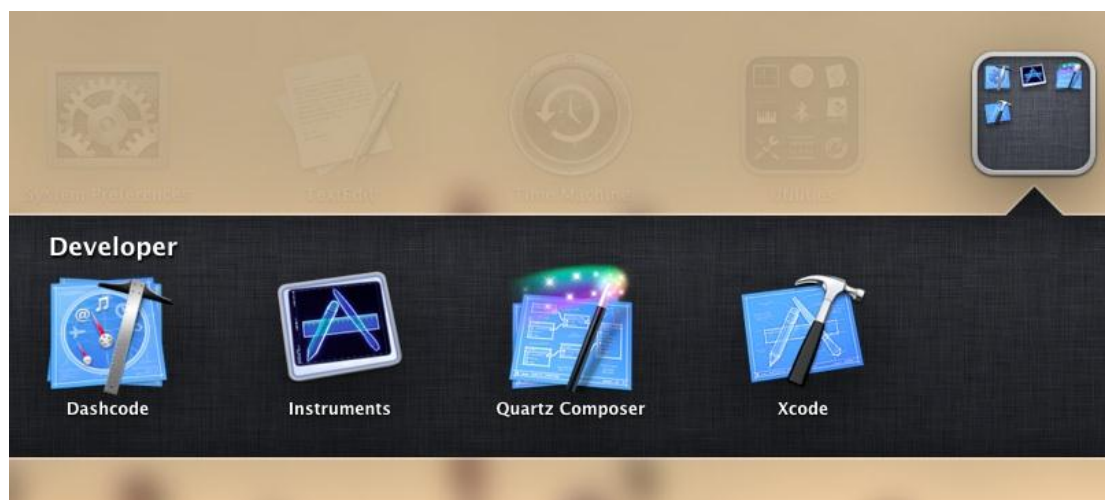


这个应用程序非常简单，仅仅显示一个 Hello World 按钮。当点击该按钮时，将弹出一个消息，就这些。并不复杂，但是它会帮助你开始你的第一个 iOS 编程旅程。

开始编写代码！

首先，启动 Xcode。如果你通过 Mac App Store 安装了 Xcode，你可以在

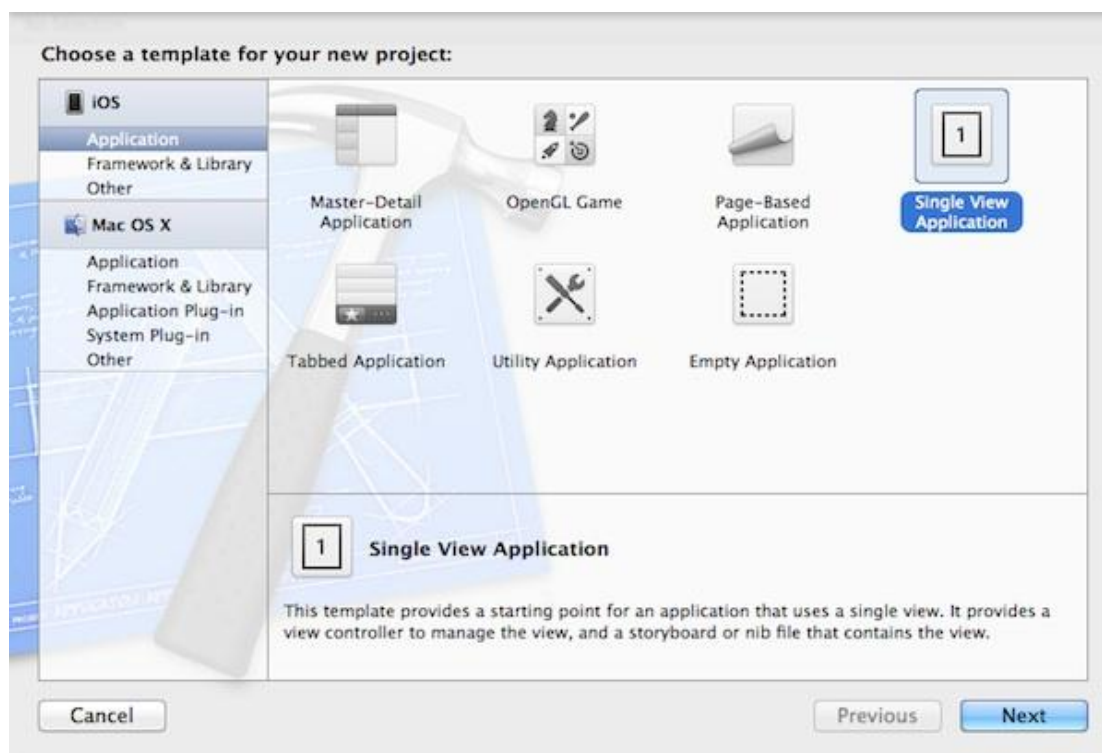
LaunchPad 中找到 Xcode , 点击 Xcode 图标就可以启动了。



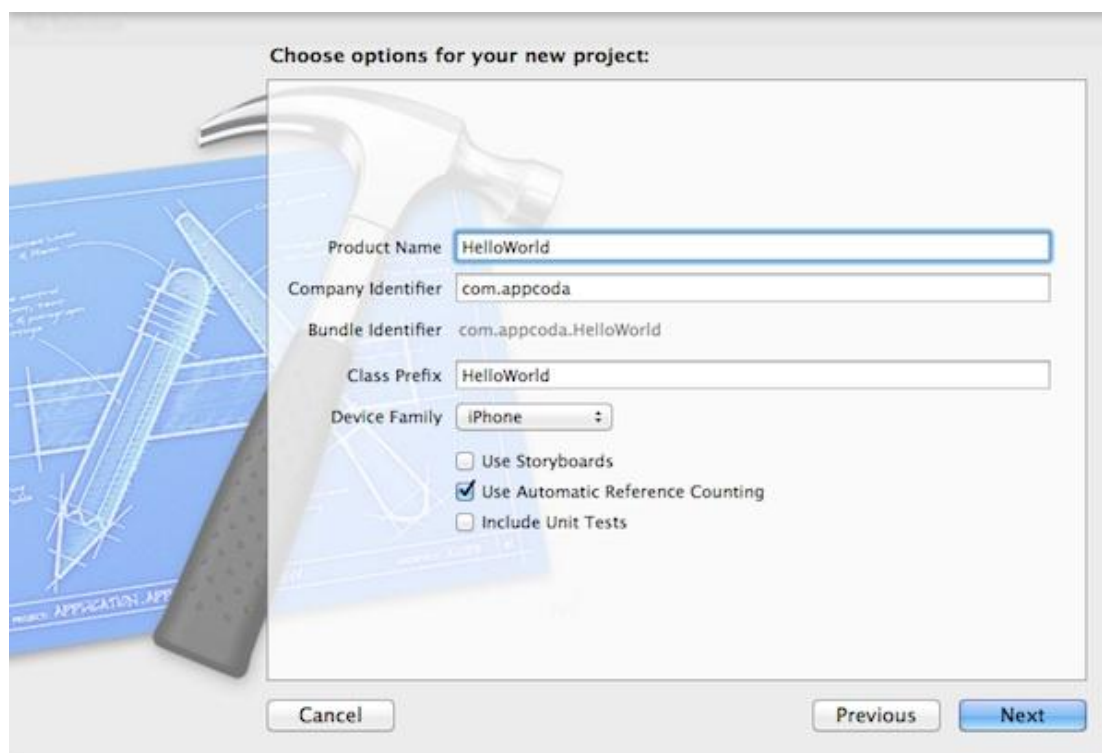
启动 Xcode 之后 ,Xcode 显示一个欢迎对话框。在该对话框中 ,选择 Create a new Xcode project 开始一个新的项目 :



Xcode 显示不同的项目模板供你选择。对于你的第一个应用程序 ,选择 Single View Application , 并点击 Next。



接下来弹出另一个窗口，输入项目的所有必要信息。

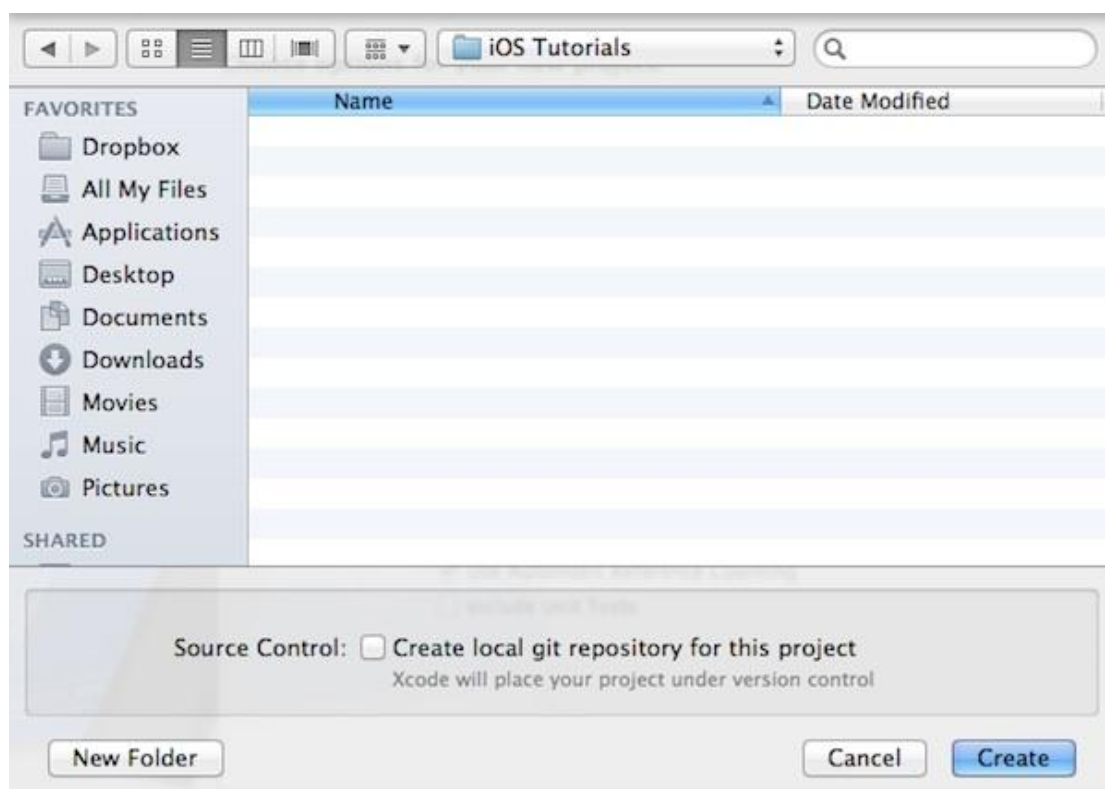


你可以简单输入如下选项：

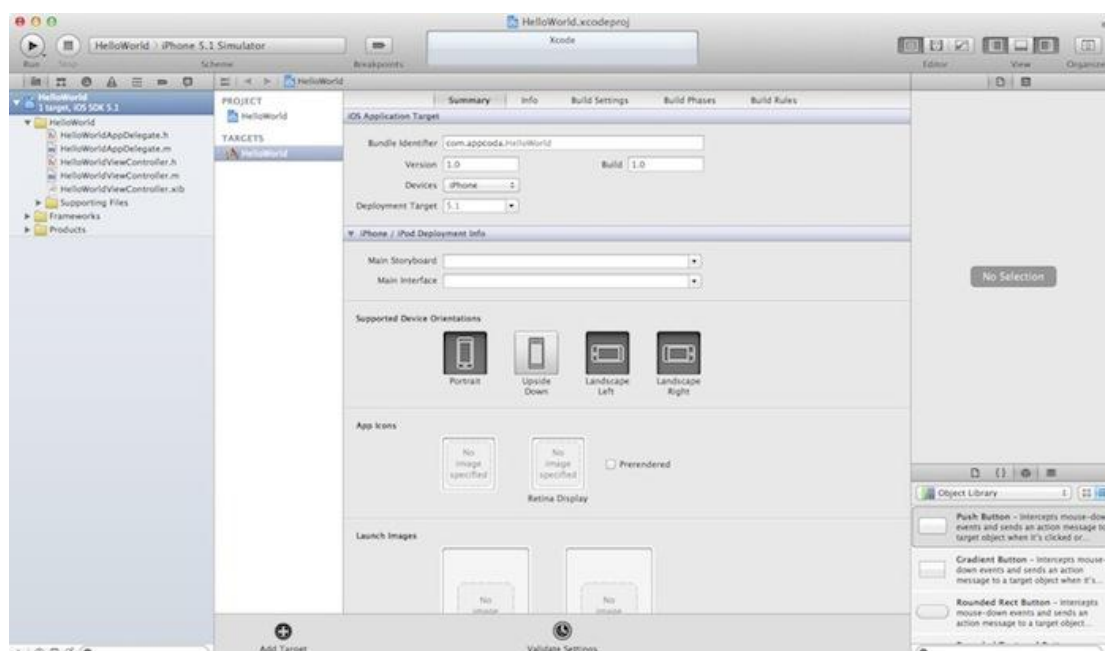
- Product Name : HelloWorld – 应用程序的名称；

- Company Identifier : com.appcoda – 通常将域名反写录入。如果你有自己的域名,可以使用你自己的域名。否则,你可以使用我们的域名或者仅仅输入 edu.self ;
- Class Prefix : HelloWorld – Xcode 会自动使用类前缀命名类。将来,你可以选择你自己的前缀名称,或者让它为空。但是在本教程中,为了简单,使用 HelloWorld。
- Device Family : iPhone – 针对本项目仅选择 iPhone ;
- Use Storyboards : **【不选择】** - 不必选择,对于这个简单的项目不必使用 Storyboards ;
- Use Automatic Reference Counting : **【选择】** - 默认情况下,启用这一选项。
- Include Unit Tests : **【不选择】** - 当前,不必使用单元测试类;

点击 Next 继续。Xcode 接着问你在哪里保存 HelloWorld 项目。选择 Mac 中任何目录(如 桌面)。你会注意到有一个选项为源码控制,不必选择。我们将在后面的教程中讨论。点击 Create 继续。



在你确认之后，Xcode 基于你的选择，自动创建 HelloWorld 项目。界面如下所示：

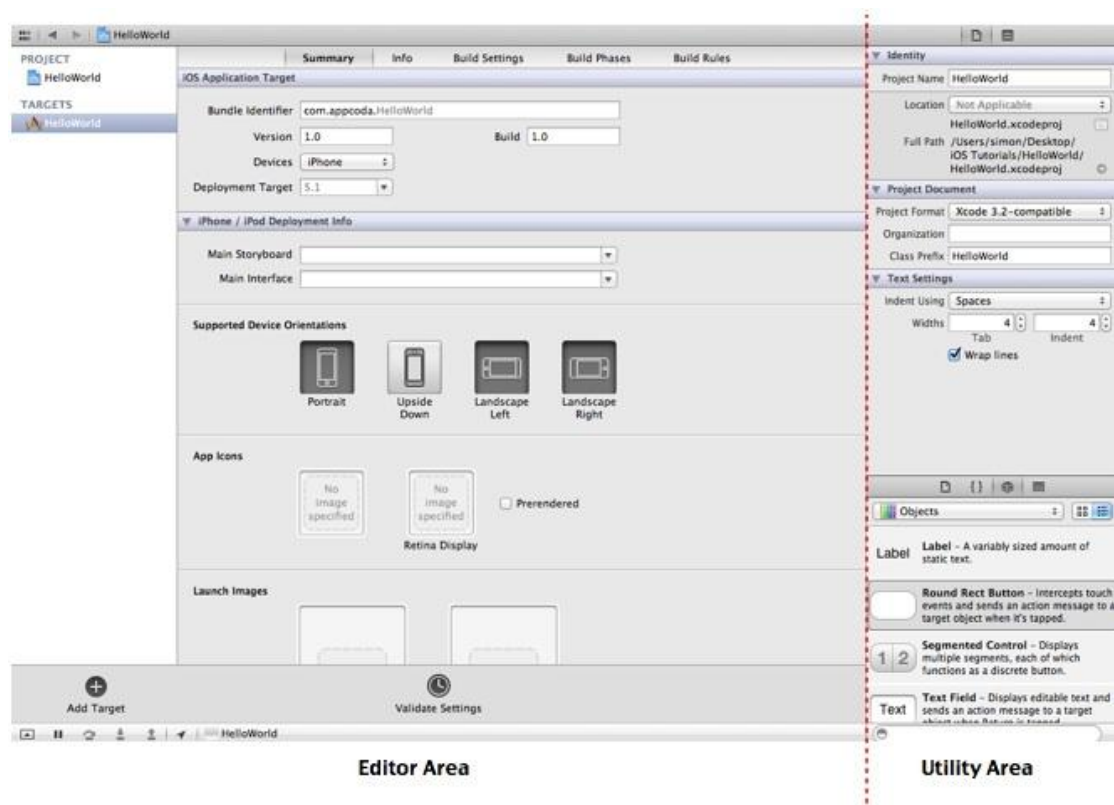


熟悉 Xcode 工作环境

在我们继续编写代码之前，让花几分钟快速看看 Xcode 的工作环境。在左侧面板，是项目导航栏。在这一区域，你可以发现所有项目文件。



Xcode 工作区域的中间部分是编辑区域。根据选择的文件类型，你在此区域进行所有的编辑工作（如编辑项目设置、类文件、用户界面等等）。



最右边的面板是工具区域 (Utility Area)。这一区域显示了文件属性，并允许你访问快速帮助 (Quick Help)。如果 Xcode 没有显示这一区域，你可以选择最右侧视图按钮 - 在工具栏上，启用这一区域。

最后，是工具条部分。工具条提供了不同的功能，让你运行你的应用程序，切换编辑器和工作区视图等等。



第一次运行你的应用程序

即使你没有编写任何代码，你可以在模拟器中运行你的应用程序。演示如何在

Xcode 中构建和测试你的应用程序。简单点击工具栏中的 Run 按钮。



Xcode 自动构建应用程序，并在模拟器中运行。下图是模拟器的运行界面：



一个灰色的界面，没有任何东西在里面？这个很正常。因为你的 App 还没有完成，模拟器仅仅显示一个空白的界面。终止应用程序，简单点击工具栏的 Stop 按钮。

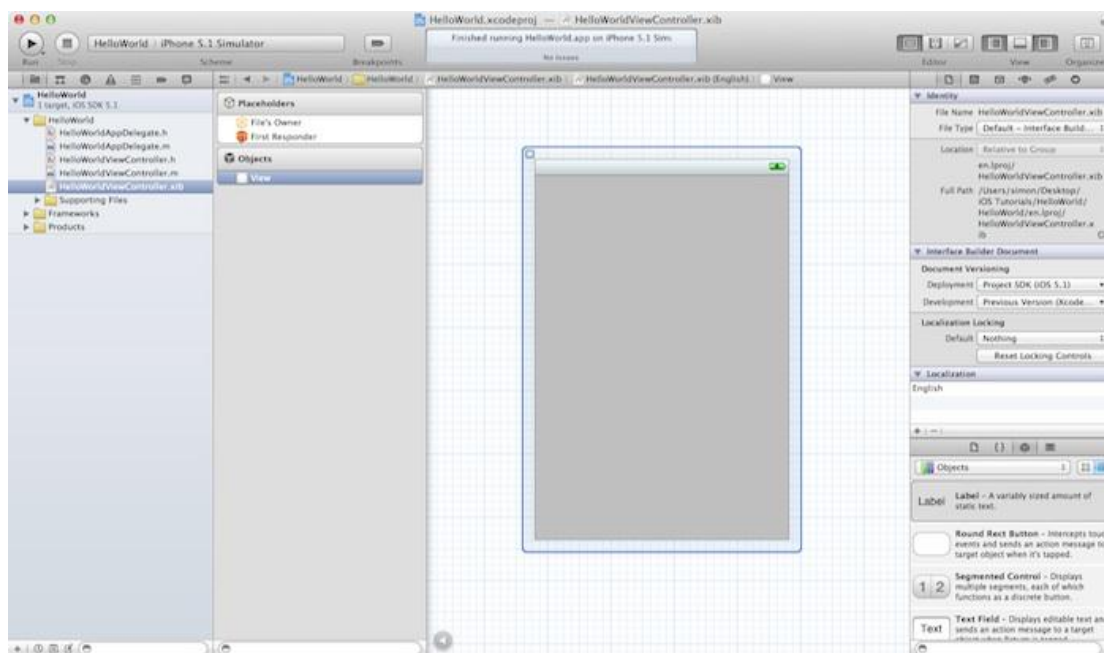


回来写代码吧！

OK. 现在继续，开始在应用程序中添加 Hello World 按钮。返回应用程序导航界面，选择 HelloWorldViewController.xib。

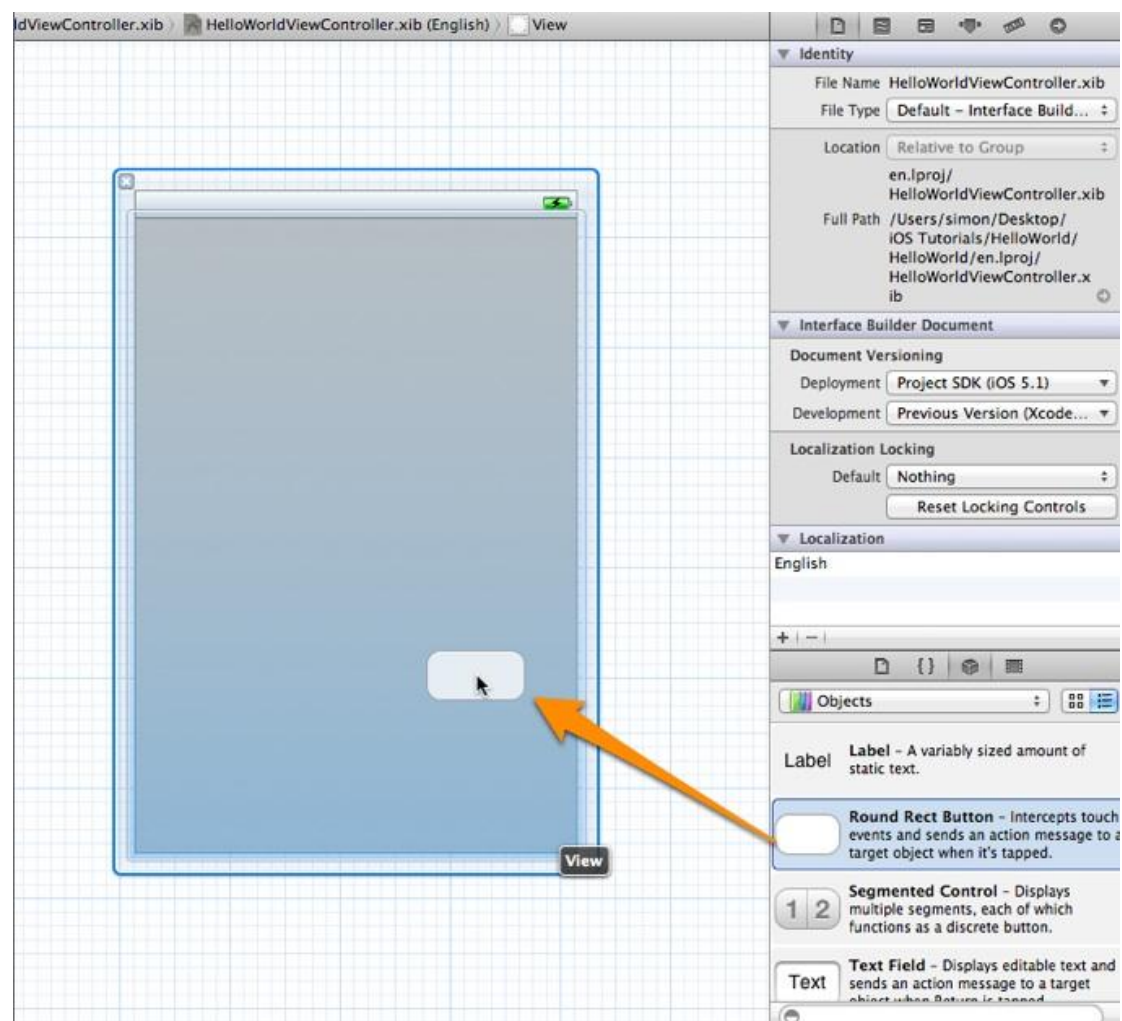


当你选择该文件时，编辑器改变为 Interface Builder，并显示应用程序的空白视图，如下图所示：



在工具区域 (Utility Area)，显示 Object library (对象库)。从这里，你可以选

择任何 UI 控件，拖曳到视图上。对于 Hello World 应用程序，我们选择 Round Rect Button，并拖曳到视图。将按钮放置到视图的中间位置。



编辑按钮的文本 - 双击该按钮，命名为 Hello World。



再次运行 App，你将看到如下图界面：



现在，如果你点击该按钮，没有任何效果。我们将编写代码，实现显示 Hello，World 消息。

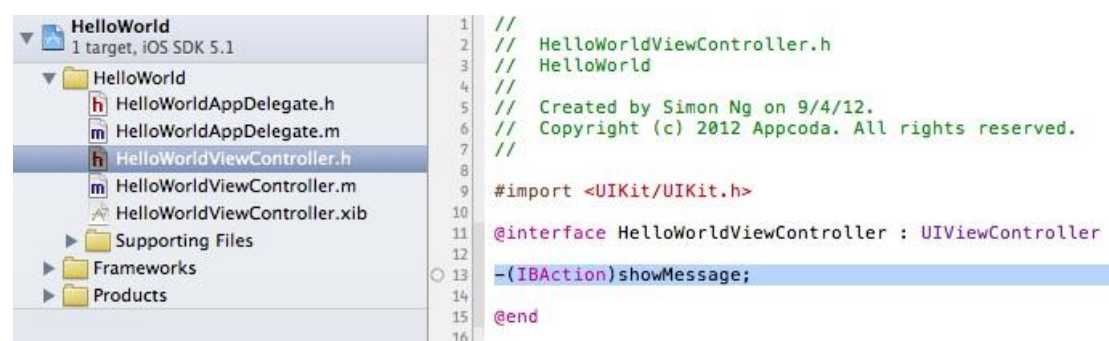
编写 Hello World 按钮的代码

在项目导航栏中，选择 HelloWorldViewController.h 文件，编辑器现在显示所

选文件的源代码。在@end 代码行前面添加如下一行代码：

```
-(IBAction)showMessage;
```

编辑完成之后，你的代码应该如下图所示：



接着，选择 HelloWorldViewController.m 文件，并在 @end 代码行前面，插入如下代码：

```
- (IBAction)showMessage
```

```
{
```

```
    UIAlertView *helloWorldAlert = [[UIAlertView alloc]
```

```
        initWithTitle:@"My First App" message:@"Hello, World!"
```

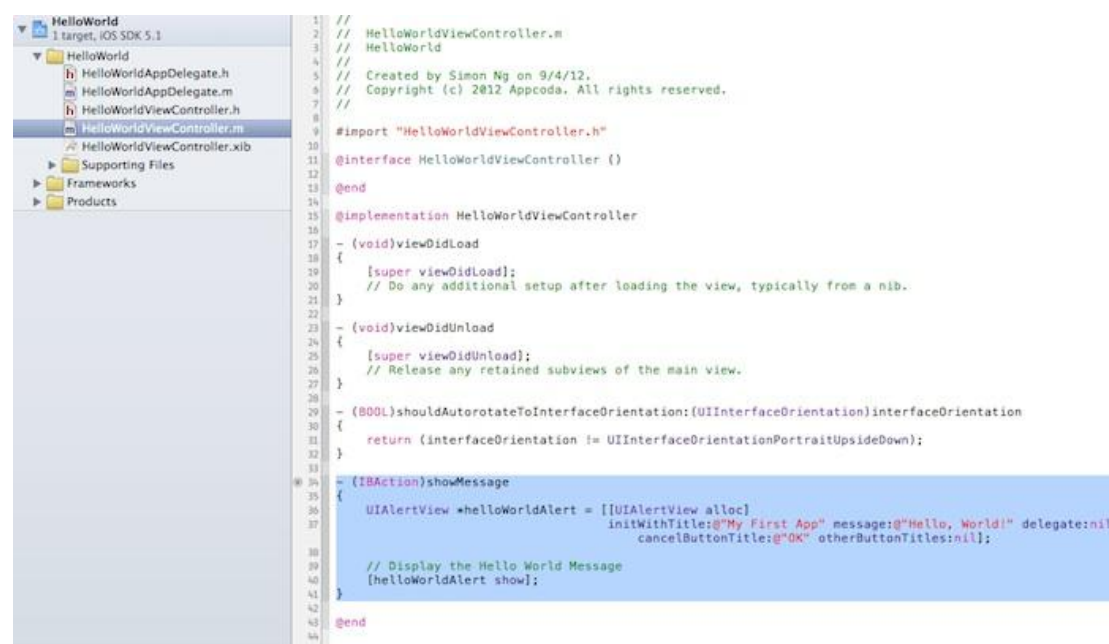
```
        delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
```

```
    // Display the Hello World Message
```

```
    [helloWorldAlert show];
```

```
}
```

编辑完成之后，代码如下图所示：

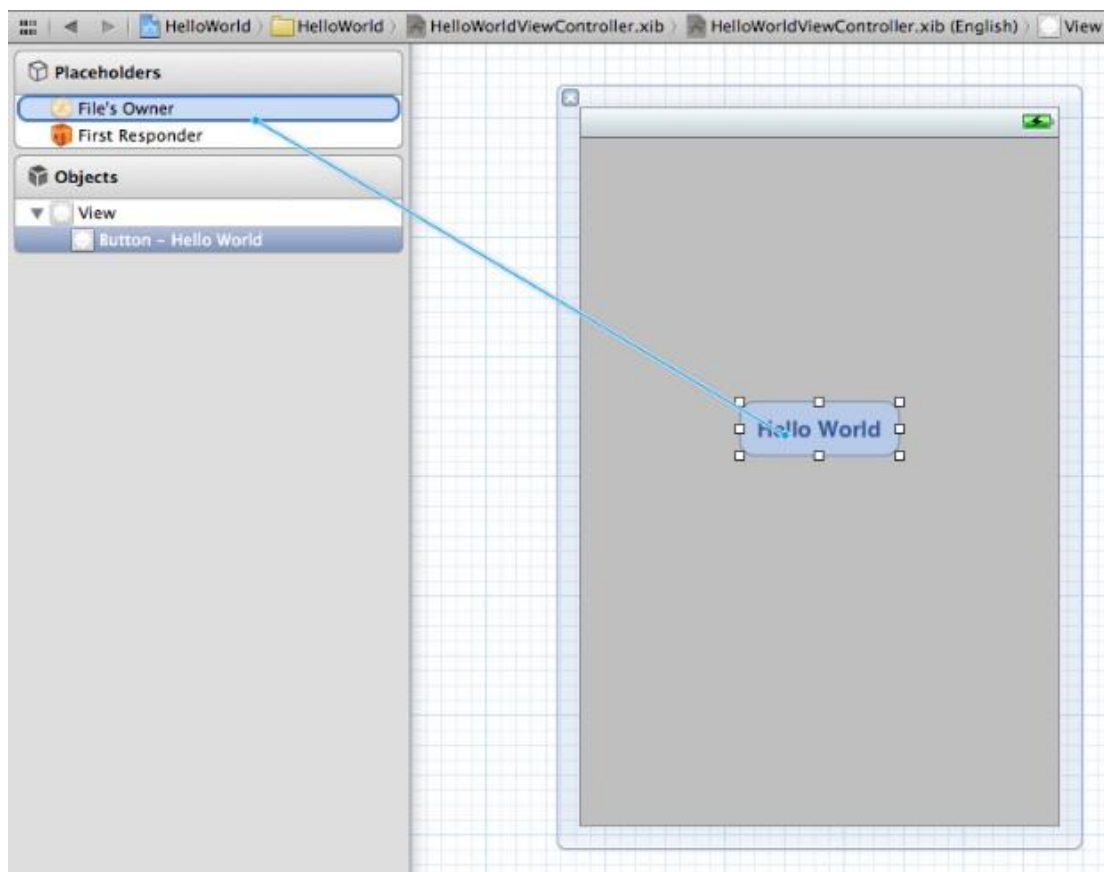


暂不必考虑上述 Objective-C 代码的意思 ,后面的文章会进行解释。现在开始 , 仅需考虑 showMessage 作为一个方法 , 该方法指示 iOS 显示 Hello World 消息在屏幕上。

连接 Hello World 按钮和 Action 方法

这里有个问题：当用户点击时 ,Hello World 按钮如何知道调用哪个 action 方法？

接下来 ,你需要建立 Hello World 按钮和刚刚加入的方法 – showMessage 之间的连接。选择 HelloWorldViewController.xib 文件 ,返回 Interface Builder 窗口。按下 Control 键 ,并点击 Hello World 按钮 ,拖曳到 File' s Owner 图标上。屏幕如下图所示：



释放按钮和键，弹出一个窗口，显示 `showMessage` 方法，选择该方法，建立按钮和 `showMessage` 方法之间的连接。



测试你的应用程序

好啦！你现在可以开始测试你的第一个 App 了。点击 Run 按钮，如果一切正

常，你的 App 将运行在模拟器中，如下图所示：



恭喜你！你完成了你的第一个 iPhone App。尽管这是一个简单的 App，但是我相信你已经更好地了解了 Xcode 和 App 的开发。

下一篇文章中，我将继续解释前面编写的 Objective-C 代码，以及 HelloWorld 应用程序是如何运行的。继续关注哦！

本文由 [EntLib.com Team](#) 翻译整理，如你在创建 App 中遇到问题，欢迎访问 [EntLib.net](#) 留言和提交你的问题。

英文原文连接：[Hello World! Build Your First iPhone App](#)

第二部分：iOS 编程基础：Hello World 应用程序如何工作的？

我希望你已经阅读了《[Hello World！创建你的第一个 iPhone App](#)》教程，并且已经完成了 Hello World 应用程序的创建。在开始下一篇向导和创建稍复杂的 App 之前，我们回头仔细看看 Hello World 应用程序。理解一些 Objective-C 语法和 App 的内部工作机制对你是有好处的。

目前位置，你已经一步一步创建了 Hello World 应用程序。可能你在完成向导的过程中，想到了如下这些问题：

- 哪些 .xib、.h 和 .m 文件是什么？
- 在 showMessage 方法里面的粗陋代码是什么？有什么用途？
- 在你点击 Hello World 按钮之后，实际发生了什么？该按钮如何触发 showMessage 方法的？
- Xcode 中 Run 按钮是如何工作的？

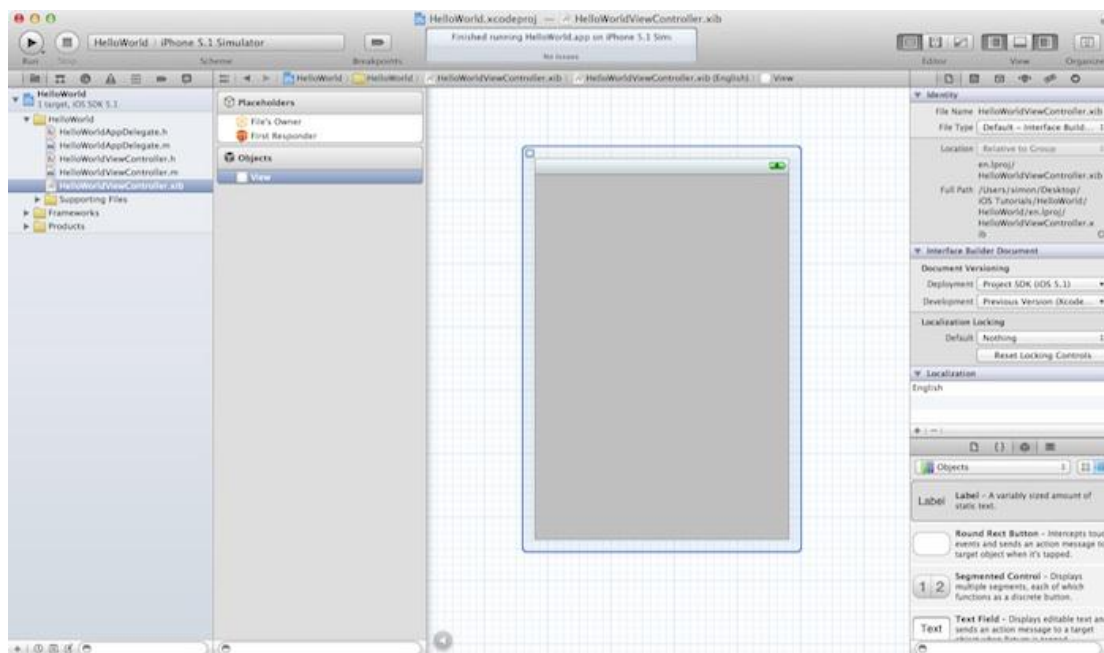
在前一篇向导中，我希望你专注于 Xcode 开发环境，因此没有解释任何这些问题。对于每一个开发人员而言，理解内部的细节和掌握 iOS 编程的基本概念是非常重要的。对于一些技术概念，如果你没有编程经历，可能有些难以理解。但是不必担心，这仅仅是一个开始而已。在你逐步深入，并编写更多的代码之后，你将对 iOS 编程有更好的理解。尽你最大的努力学习更多的知识。

Interface Builder、头文件和实现文件

首先,这些.xib、.h 和 .m 文件是什么?这是一些读者提出的一个很好的问题。

在项目导航栏下,可以发现主要有 3 类文件:.xib、.h 和 .m (如果你展开 Supporting Files 文件夹,将会发现其他文件类型,如 plist 和框架。但是现在我们忘掉这些吧,在后面会谈到的)。

.xib – 对于.xib 后缀文件,是 Interface Builder 文件,存放应用程序的用户界面 (UI)。当你点击 .xib 文件时,Xcode 自动切换到 Interface Builder 界面,你可以通过拖拉编辑应用程序的用户界面。



.h 和 .m - .h 后缀的文件指头文件,.m 后缀的文件是实现文件。和大多数编程语言一样,Objective-C 源代码文件分为 2 部分:接口文件和实现文件。

为了更好地理解这些术语,我们举一个例子。以电视机遥控器为例,它可以通过无线来方便控制电视机的音量。点击增加音量按钮,可以增加电视机的声音。你知道按住音量按钮时,里面发生了什么吗?可能不知道。我相信多数人都不知道电

视机和音量按钮的远程通信。我们仅仅知道这个按钮可以改变电视机的音量而已。在本示例中，和你交互的按钮是一个接口，隐藏在按钮内部的细节指的是具体实现。

现在你应该知道接口和实现的概念了。返回 Objective-C 代码，类的接口定义在 .h 文件中，使用 @interface 语法定义类的接口。看看 HelloWorldViewController.h 文件，这是一个头文件：

```
@interface HelloWorldViewController : UIViewController

-(IBAction)showMessage;

@end
```

和音量按钮一样，我们不知道 showMessage 方法是如何工作的，仅仅知道这个方法用来在屏幕上显示一条消息。具体的实现在 HelloWorldViewController.m 实现文件中，代码如下：

```
@implementation HelloWorldViewController

// I've removed other methods for better reading. Focus on the
showMessage method first.

- (IBAction)showMessage

{

    UIAlertView *helloWorldAlert = [[UIAlertView alloc]

initWithTitle:@"My First App" message:@"Hello, World!" delegate:nil
```

```
cancelButtonTitle:@"OK" otherButtonTitles:nil];
```

```
    // Display the Hello World Message
```

```
    [helloWorldAlert show];
```

```
}
```

```
@end
```

从上面的代码可以看出，使用@implementation 语法定义一个实现。在 showMessage 方法里面，是实际的代码，用来在屏幕上显示警告消息。你可能没有理解 showMessage 方法的每一行代码。简而言之，它创建一个 UIAlertView 视图，标题为 My First App，消息为 Hello，World。接着调用 show 方法，要求 iOS 在屏幕上显示弹出消息。



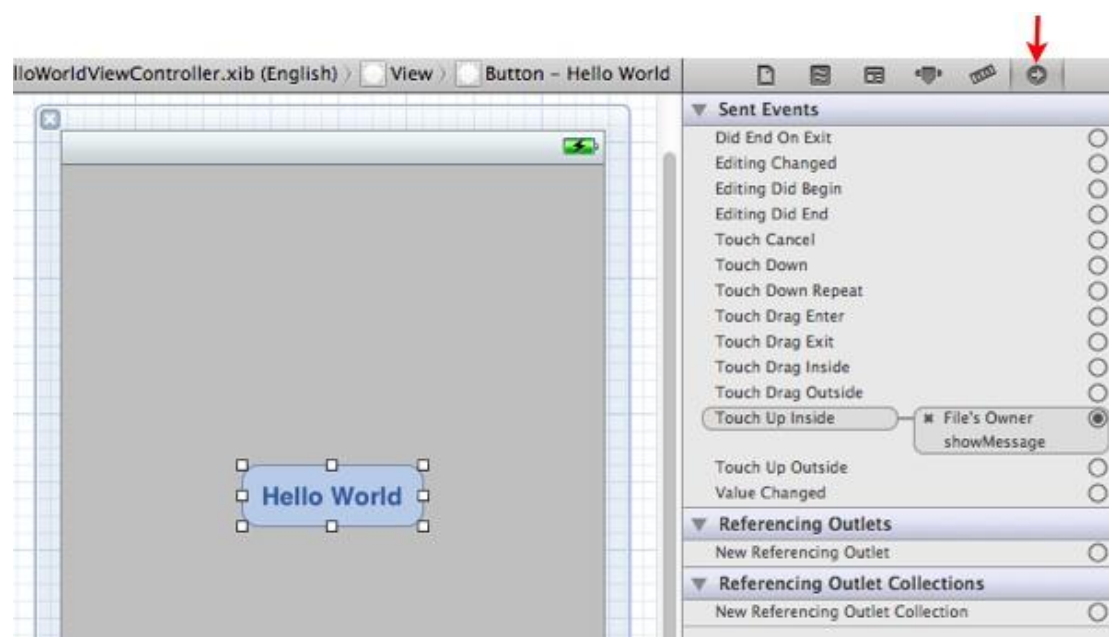
理解接口和实现的概念是非常重要的。如果你有任何问题 ,欢迎在后面留言提出。

触摸和点击的背后机制

在点击 Hello World 按钮之后 ,实际发生了什么 ? Hello World 按钮如何调用 showMessage 方法 ,显示 Hello World 消息 ?

回想起你在 Interface Builder 界面时 ,为 Hello World 按钮和 sendMessage 方法建立了一个连接。再次打开 HelloWorldViewController.xib 文件 ,选择

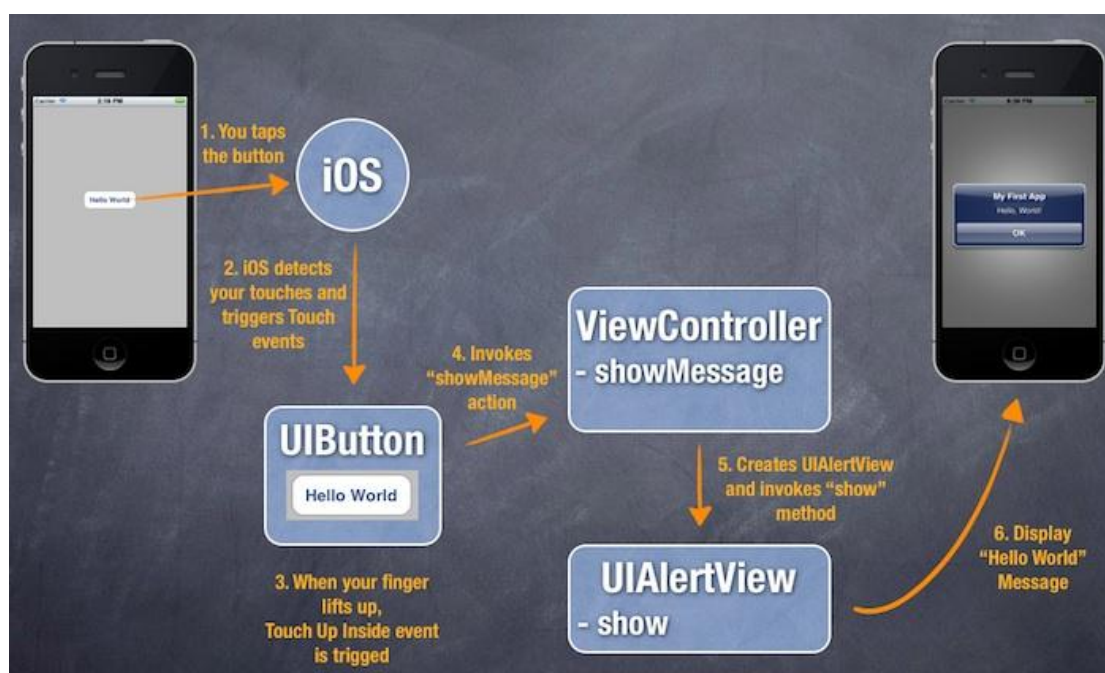
Hello World 按钮，然后点击工具区域 (Utility Area) 的 Send Events 按钮，打开 Send Events 窗口。



Send Events 窗口显示了事件和动作方法之间的所有连接。从上图中可以看到，Touch Up Inside 事件和 showMessage 方法之间建立了连接。在 iOS 中，应用程序是事件驱动的。控件或对象（如 UIButton）监听特定事件（如触摸和点击）。当事件触发时，对象会调用和事件关联的预先设置的方法。

在我们的 Hello World 应用程序中，当用户提起在按钮上的手指时，将触发 Touch Up Inside 事件。结果是调用 showMessage 方法，显示 Hello World 消息。

下图总结了整个事件流程和我前面描述的内容：



Run 按钮幕后机制

当你点击 Run 按钮时，Xcode 自动启动模拟器，并运行你的应用程序。但是幕后到底发生了什么呢？作为一名开发人员，你需要了解整个过程。



整个过程可分为 3 个阶段：编译、打包和运行。

编译 (Compile) - 你可以认为 iOS 理解 Objective-C 代码。实际上, iOS 仅仅理解机器码。Objective-C 代码是适合开发人员你来编写和阅读代码的。为了让 iOS 理解应用程序的原地吗, 我们需要通过转换过程将 Objective-C 代码翻译为机器码。这个过程称为编译。Xcode 已经内置了编译器来编译源代码。

另外, 关于 Xcode 应用程序的编译和构建阶段 (Build Phases), 建议你参考如下的文章：

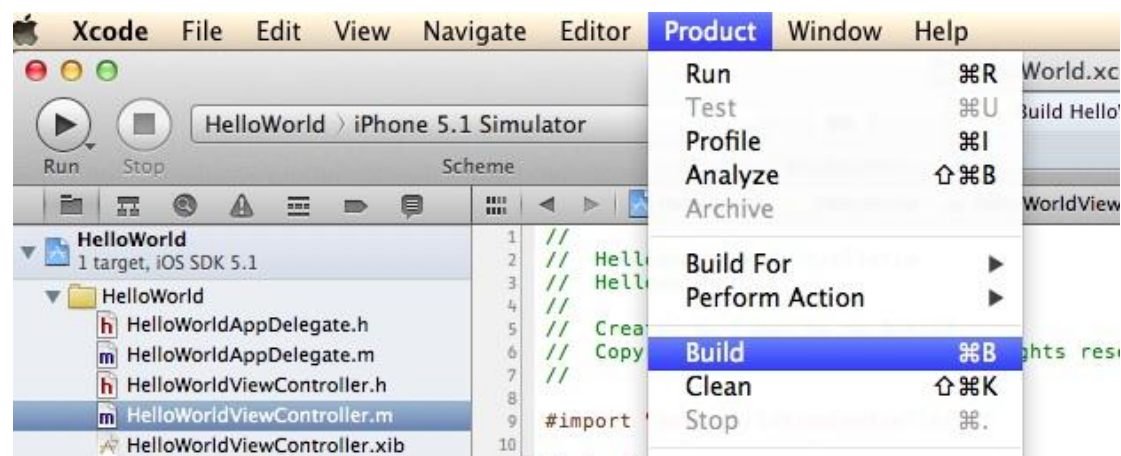
[Xcode 应用程序构建阶段 \(Build Phases \) 分析 \(1 \)](#)

[Xcode 应用程序构建阶段 \(Build Phases \) 分析 \(2 \)](#)

上述文章对 Xcode App 的构建阶段进行了深入描述和讲解。

打包 (Package) - 除了源代码之外, 应用程序通常还包含一些资源文件, 如图像文件、文本文件、xib 文件等等。所有这些资源文件打包, 创建一个最终的应用程序。

我们通常认为这两个过程为构建过程 (Build Process)。



运行 (Run) - 实际启动模拟器, 并装载你的应用程序。

有任何问题么？

我尽最大努力解释 Hello World 应用程序的实际工作机制。如果初学者没有之前的编程经验，并不容易理解我们讨论的所有概念。

本文由 [EntLib.com Team](#) 翻译整理，如你在创建 App 中遇到问题，欢迎访问 [EntLib.net](#) 留言和提交你的问题。

英文原文连接：[iOS Programming Basic: How Does the Hello World App Work?](#)

第三部分：iOS 编程向导：创建一个简单的表视图（Table View）应用程序

创建 [Hello World 应用程序](#) 有趣吗？在本教程中，我们继续学习稍复杂的内容，使用表视图（Table View）创建一个简单的 App。如果你还没有阅读 iOS 编程基础，建议你先阅读该文章。

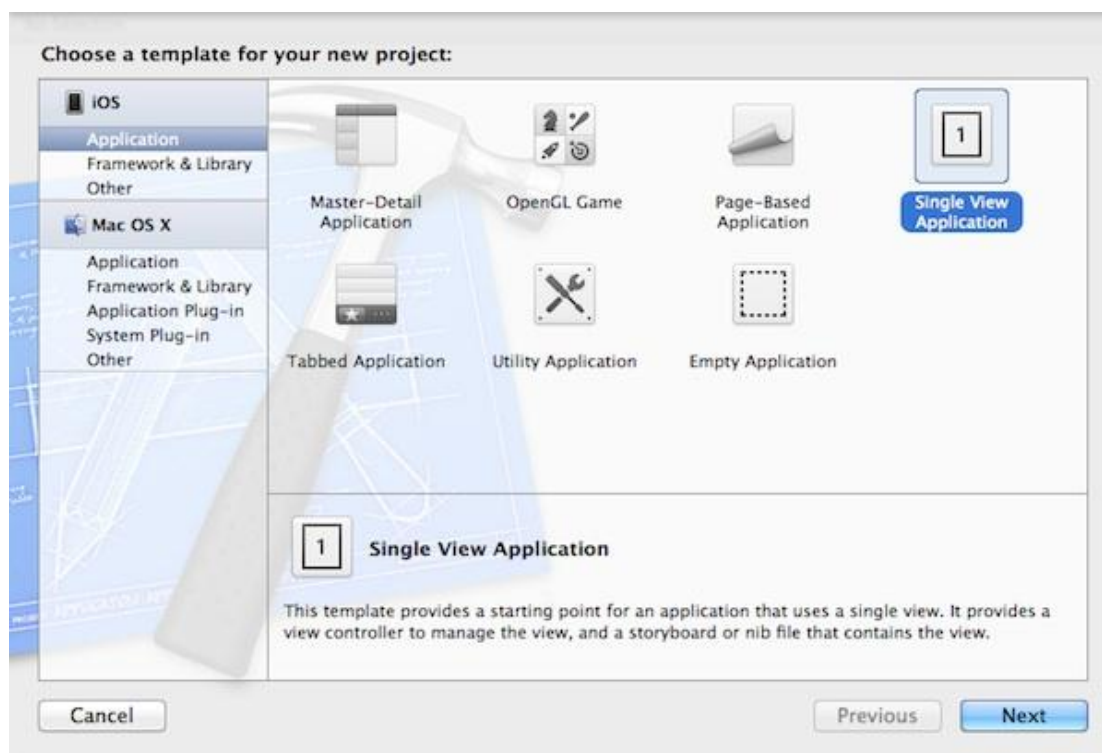
首先，在 iPhone App 中的表视图是什么？表视图（Table View）是 iOS Apps 中一个通用的 UI 元素。很多应用程序在一定程度上，都有使用表视图来显示数据列表。最好的例子是内置的 iPhone 应用程序。你的联系人显示在表视图中。另外一个例子是 Mail 应用程序，它使用表视图显示你的邮箱和邮件。不仅可以用来显示文本数据，表视图也可以呈现图像数据。内置的 Video 和 YouTube 应用程序是这一用法的例子。



创建 SimpleTable 项目

了解表视图之后，接着我们着手创建一个简单的 App。如果你想认真学习 iOS 编程，就不要浏览教程，而是打开你的 Xcode，动手编写代码。这是需要编程的最好方法。

启动 Xcode 之后，创建一个 Single View application 的新项目。



点击 Next 按钮继续。同样的，输入 Xcode 项目所有必须的选项：

Product Name: SimpleTable.

Company Identifier: com.appcod.

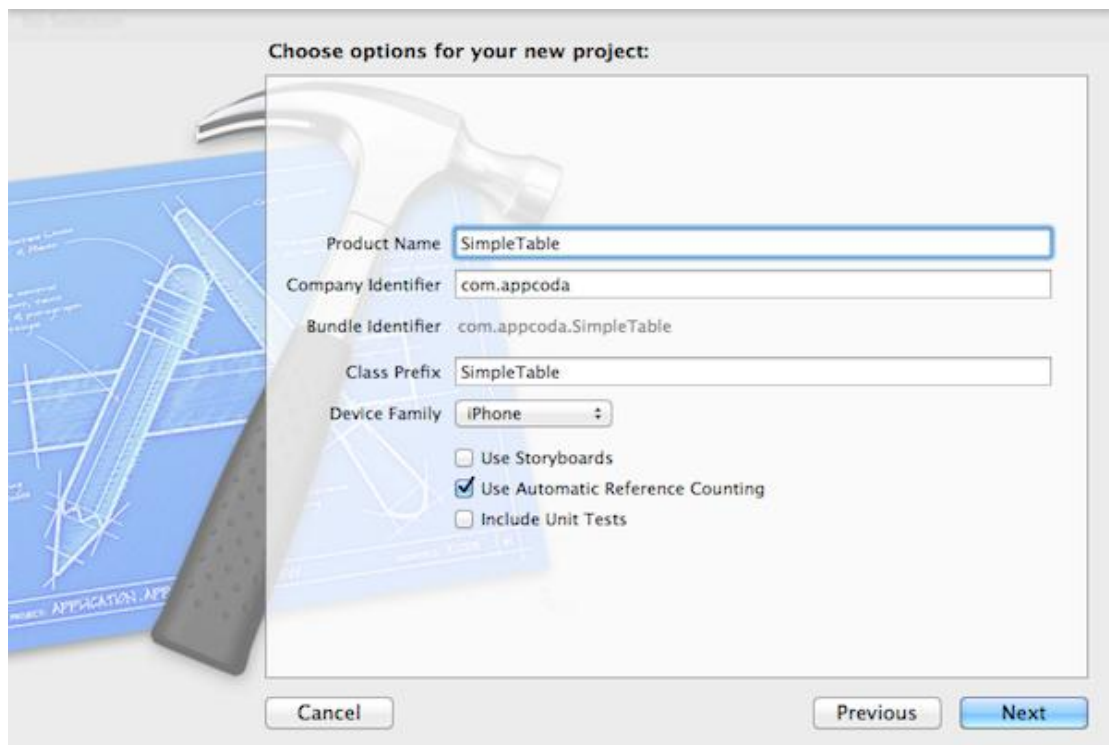
Class Prefix: SimpleTable.

Device Family: iPhone.

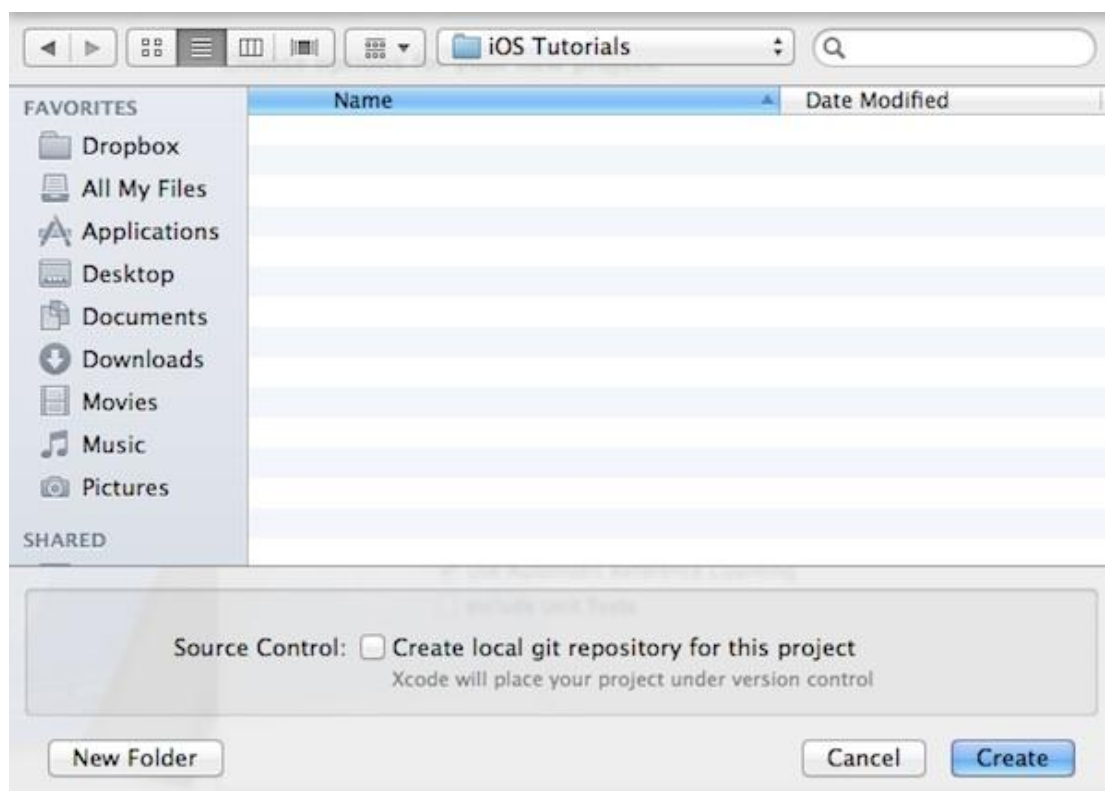
Use Storyboards: [不选择].

Use Automatic Reference Counting: [选择].

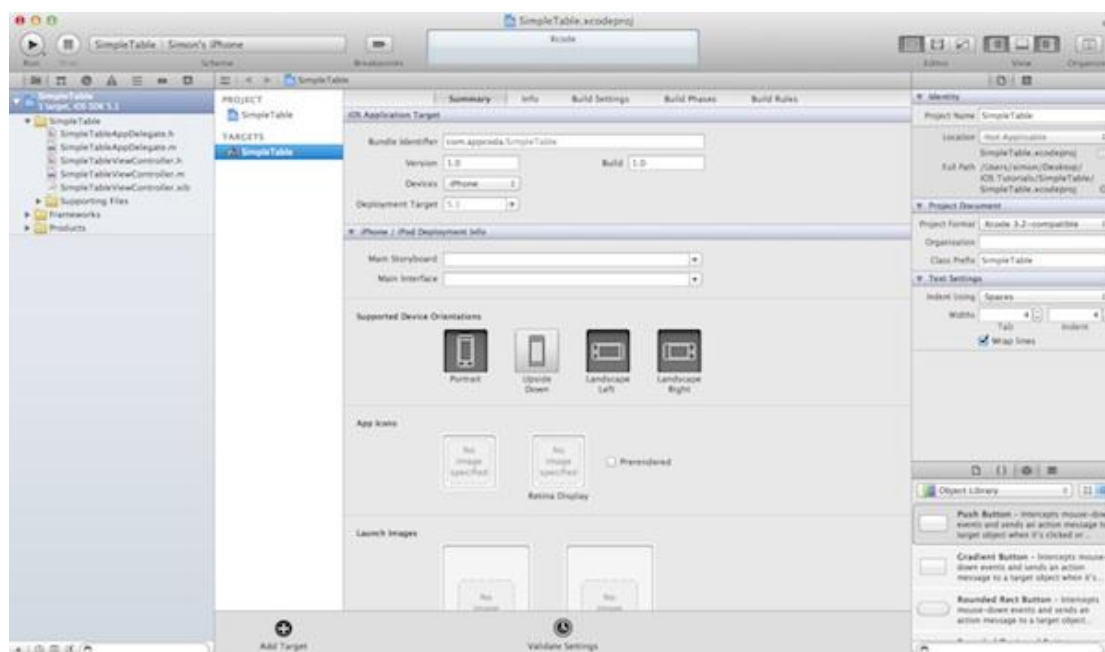
Include Unit Tests: [不选择].



点击 Next 按钮继续。Xcode 接着问你在哪里保存 SimpleTable 项目。选择任何目录(如 Desktop 桌面)来保存项目。和之前一样,不必选择源码控制(Source Control)。点击 Create 按钮继续。

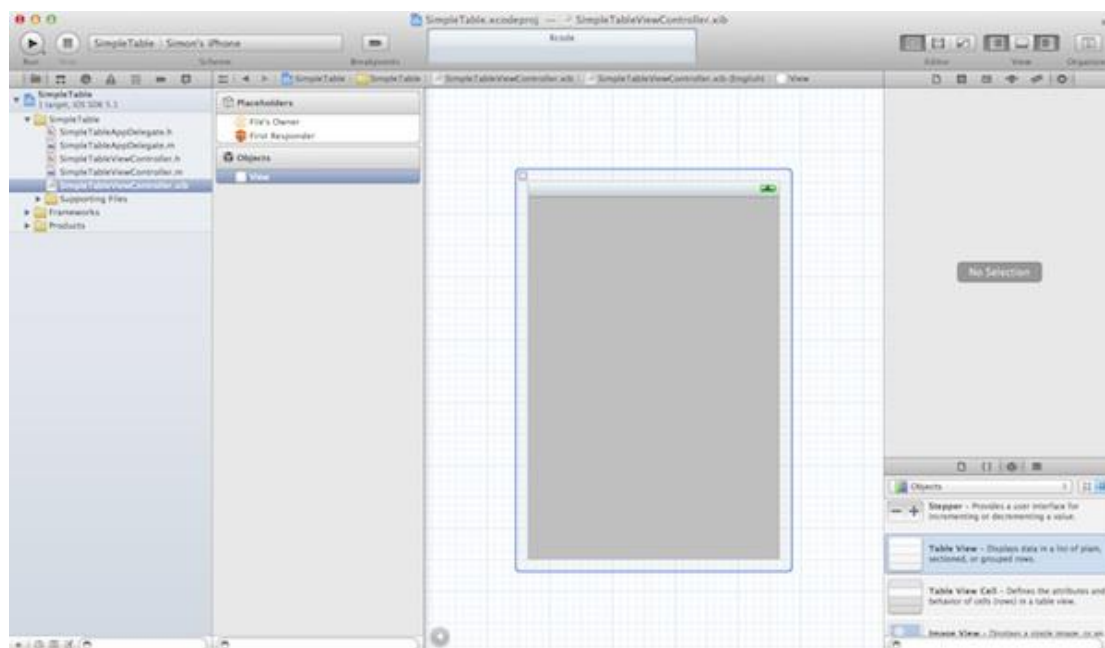


随着你的确认，Xcode 根据你提供的选项，自从创建 SimpleTable 项目。结果如下图所示：

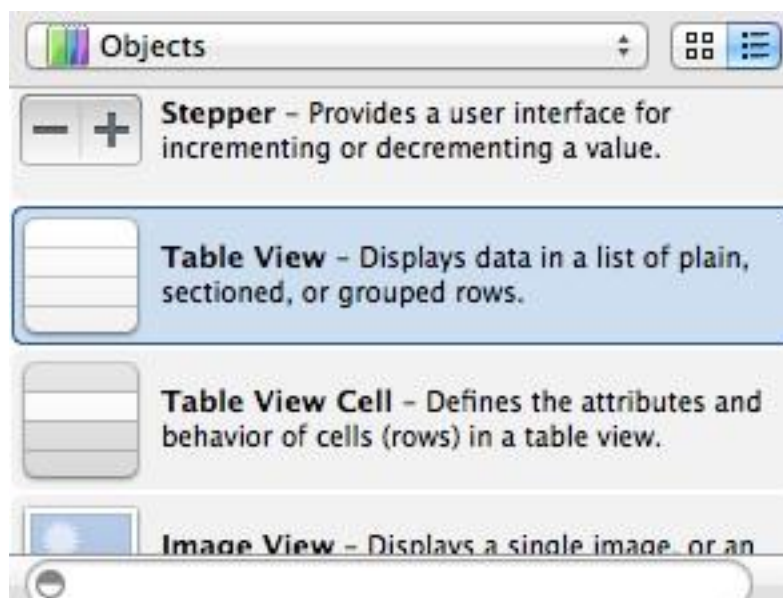


设计视图

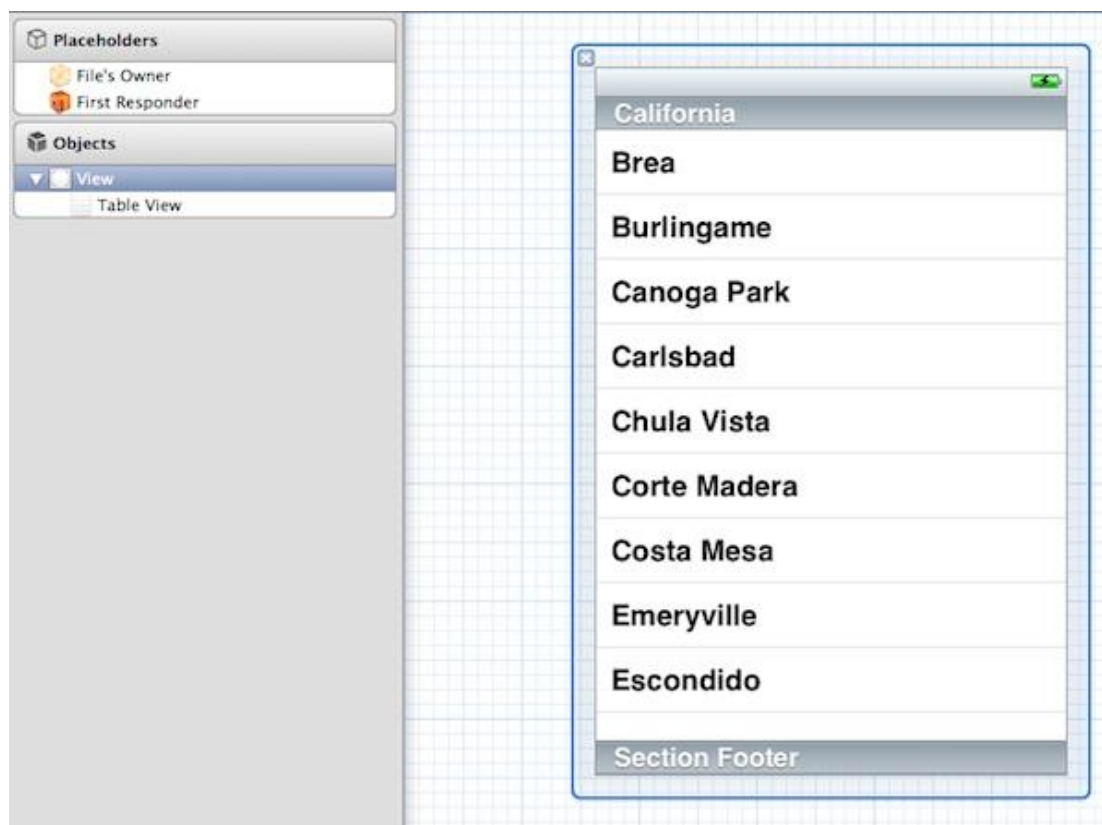
首先，我们将创建用户界面，并添加表视图。选择 SimpleTableViewController.xib 文件，切换到 Interface Builder 界面。



在对象库 (Object Library) 中，选择 Table View 对象，并拖曳到视图中。



添加 Table View 之后，你的屏幕将如下所示：



第一次运行你的应用程序

在继续之前，尝试使用模拟器运行你的应用程序。点击 Run 按钮构建你的 App 并进行测试。



模拟器屏幕如下图所示：



很简单，对吧？你已经在你的 App 中设计了表视图。但是，现在它没有包含任何数据。接着，我们将编写代码，添加表数据。

添加表数据

返回项目导航栏，选择 SimpleTableViewController.h 文件。在 UIViewController 之后，添加 `<UITableViewDelegate, UITableViewDataSource>`。完成后代码如下所示：


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

```
@interface SimpleTableViewController : UIViewController  
  
<UITableViewDelegate, UITableViewDataSource>  
  
@end
```

在 Objective-C 中，UITableViewDelegate 和 UITableViewDataSource 称为协议。基本上，为了在表视图中显示数据，我们必须遵守定义在协议中的要求，并实现所有要求的方法。

UITableViewDelegate 和 UITableViewDataSource

前面，我们在头文件中添加了 UITableViewDelegate 和 UITableViewDataSource 协议。你可以困惑，它们是什么？

UITableView 是表视图幕后的实际类，用来灵活处理不同的数据类型。你可以显示国家列表或者联系人姓名。或者像本示例一样，我们将使用表视图程序菜谱列表。因此，你如何告诉 UITableView 需要显示的数据列表呢？

UITableViewDataSource 是答案，它用来连接你的数据和表视图。

UITableViewDataSource 协议定了 2 个要求实现的方法

(tableView:cellForRowAtIndexPath 和 tableView:numberOfRowsInSection)。通过实现这些方法，你告诉表视图显示多少行数据和每一行中的数据。

另一方面，UITableViewDelegate 负责处理 UITableView 的表现。协议中的可选方法让你管理表行的高度，配置节点头部和底部，对表单元重新排序等等。在本示例中，我们不改变这些方法，在后面的教程中会提到。

接着，选择 SimpleTableViewController.m 文件，定义一个实例变量，存放表数据。

```
@implementation SimpleTableViewController
```

```
{  
  
    NSArray *tableData;  
  
}
```

在 viewDidLoad 方法中 (Called after the controller' s view is loaded into memory – 在控制器的视图装载到内存中完成之后，调用该方法)，添加如下代码实例化 tableData 数组。我们初始化数组为菜谱列表：

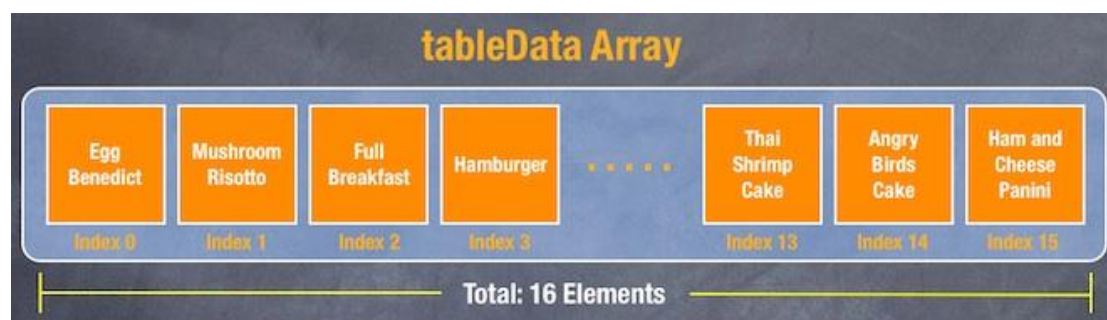
```
- (void)viewDidLoad  
{  
  
    [super viewDidLoad];  
  
    // Initialize table data  
  
    tableData = [NSArray arrayWithObjects:@"Egg Benedict",  
    @"Mushroom Risotto", @"Full Breakfast", @"Hamburger", @"Ham and  
Egg Sandwich", @"Creme Brelee", @"White Chocolate Donut",  
    @"Starbucks Coffee", @"Vegetable Curry", @"Instant Noodle with Egg",  
    @"Noodle with BBQ Pork", @"Japanese Noodle with Pork", @"Green
```

```
Tea", @"Thai Shrimp Cake", @"Angry Birds Cake", @"Ham and Cheese  
Panini", nil];  
}
```

数组是什么？

数组在计算机编程中，是一个基础的数据结构。你可将数组认为数据元素集合。

分析上述代码中的 `tableData` 数组，它表示一个文本元素集合。数组的可视化描述如下所示：



每一个数组元素都通过 `index` 来识别或访问。包含 10 个元素的数组的 `index` 为 0-9。这表示 `tableData[0]` 返回 `tableData` 数组的第一个元素。

在 Objective-C 中，`NSArray` 是创建和管理数组的类。你可以使用 `NSArray` 创建静态数组，其容量是固定的。如果你需要动态数组，则使用 `NSMutableArray` 代替。

`NSArray` 提供了一组工厂方法来创建数组对象。在我们的代码中，我们使用 `arrayWithObjects` 来实例化一个 `NSArray` 对象，并预先填充特定的元素（如 `Hamburger`）。

你也可使用其他内置方法来查询和管理数组。随后，我们将调用 `count` 方法查询数组中的数据元素个数。你可以查询 [Apple 的官方文档](#)，学习更多关于 `NSArray` 的用法。

或者参考文章 – [Xcode 使用帮助文档\(Documentation and API Reference\)](#)。

最后，我们需要添加 2 个数据源方法：`tableView:numberOfRowsInSection` 和 `tableView:cellForRowAtIndexPath`。这两个方法是 `UITableViewDataSource` 协议的一部分。在配置 `UITableView` 时，需要强制实现这两个方法。

Tasks

Configuring a Table View

- `tableView:cellForRowAtIndexPath:` *required method*
- `numberOfSectionsInTableView:`
- `tableView:numberOfRowsInSection:` *required method*
- `sectionIndexTitlesForTableView:`
- `tableView:sectionForSectionIndexTitle:atIndex:`
- `tableView:titleForHeaderInSection:`
- `tableView:titleForFooterInSection:`

Inserting or Deleting Table Rows

- `tableView:commitEditingStyle:forRowAtIndexPath:`
- `tableView:canEditRowAtIndexPath:`

Reordering Table Rows

- `tableView:canMoveRowAtIndexPath:`
- `tableView:moveRowAtIndexPath:toIndexPath:`

第一个方法用来通知表视图选择了多少条数据行，因此添加如下代码。`count` 方法简单返回 `tableData` 数组中元素个数。

```
- (NSInteger)tableView:(UITableView *)tableView
```

```
numberOfRowsInSection:(NSInteger)section
```

```
{  
  
    return [tableData count];  
  
}
```

接着，我们实现另外一个需要实现的方法：

```
- (UITableViewCell *)tableView:(UITableView *)tableView
```

```
cellForRowAtIndexPath:(NSIndexPath *)indexPath
```

```
{  
  
    static NSString *simpleTableIdentifier = @"SimpleTableItem";
```

```
    UITableViewCell *cell = [tableView  
    dequeueReusableCellWithIdentifier:simpleTableIdentifier];
```

```
    if (cell == nil) {  
  
        cell = [[UITableViewCell alloc]  
        initWithStyle:UITableViewCellStyleDefault  
        reuseIdentifier:simpleTableIdentifier];  
  
    }
```

```
    cell.textLabel.text = [tableData objectAtIndex:indexPath.row];  
  
    return cell;
```

```
}

```

每一次数据行显示的时候，都会调用 `cellForRowAtIndexPath` 方法。下图让你更好地理解 `UITableView` 和 `UITableViewDataSource` 是如何工作的。



下图是 `tableView:cellForRowAtIndexPath` 的帮助文档：

`tableView:cellForRowAtIndexPath:`

Asks the data source for a cell to insert in a particular location of the table view. (required)

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
```

Parameters

tableView

A table-view object requesting the cell.

indexPath

An index path locating a row in *tableView*.

Return Value

An object inheriting from `UITableViewCell` that the table view can use for the specified row. An assertion is raised if you return `nil`.

Discussion

The returned `UITableViewCell` object is frequently one that the application reuses for performance reasons. You should fetch a previously created cell object that is marked for reuse by sending a `dequeueReusableCellWithIdentifier:` message to *tableView*. The identifier for a reusable cell object is assigned when the delegate initializes the cell object by calling the `initWithStyle:reuseIdentifier:` method of `UITableViewCell`. Various attributes of a table cell are set automatically based on whether the cell is a separator and on information the data source provides, such as for accessory views and editing controls.

Availability

Available in iOS 2.0 and later.

Declared In

`UITableView.h`

请求数据源，在表视图的特定位置插入一个单元格。表视图中可见的每一行都会触发该事件。事件中包含的参数之一是 NSIndexPath 类型。NSIndexPath 类表示数组集合中的某个特定项的路径。要知道当前填充的是哪一行，只需要调用 NSIndexPath 对象 (indexPath) 的 row 属性，然后使用行号来引用 tableData 数组中的元素即可。得到的值被用来设置表视图中该行的文本值。

需要注意的是，这里使用 UITableView 类的 dequeueReusableCellWithIdentifier: 方法获取 UITableViewCell 类的一个实例。

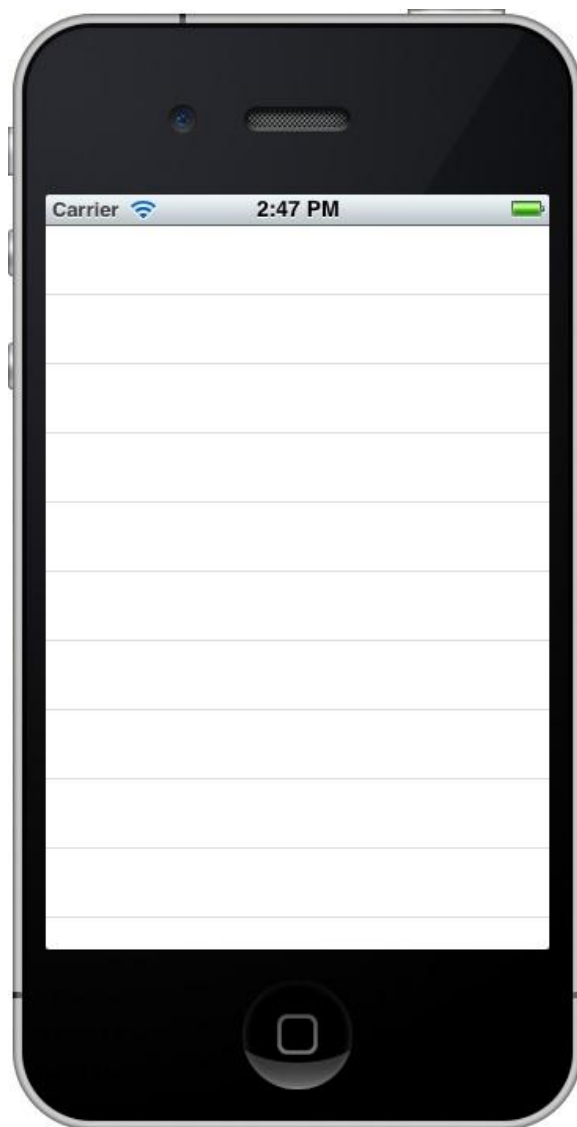
```
UITableViewCell *cell = [tableView
```

```
dequeueReusableCellWithIdentifier:simpleTableIdentifier];
```

dequeueReusableCellWithIdentifier: 方法返回的是一个可重用的表视图单元格对象。因为如果表非常大，为每一行都创建一个单独的 UITableViewCell 对象会产生严重的性能问题，并占用大量的内存。

此外，由于表视图在某一个时刻只会显示固定数量的行，因此重用已经滚动到屏幕外面的那些单元格将非常有意义。这正是 dequeueReusableCellWithIdentifier: 方法将要完成的事情。比如，如果表视图显示了 10 行，那么只会创建 10 个 UITableViewCell 对象 --- 当用户滚动表视图时，总是会重用这 10 个 UITableViewCell 对象。

OK。现在点击 Run 按钮，运行你最终的 App！如果你正确编写代码，你的 App 在模拟器中的效果如下所示：



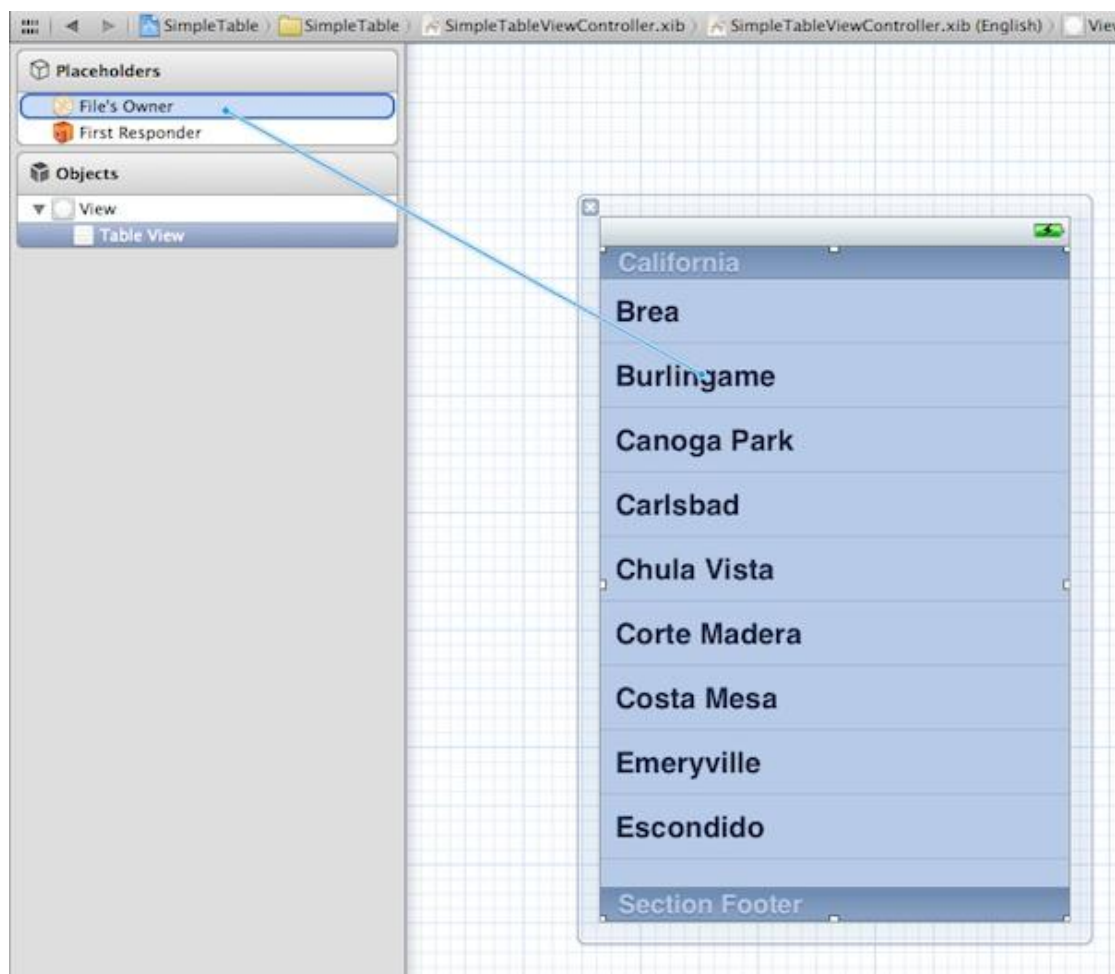
为什么仍然是空白的呢？我们已经编写了代码，生成了表数据，并且实现了要求的方法。但是为什么表视图没有如期望显示数据出来呢？

仍然还有事情没有完成！

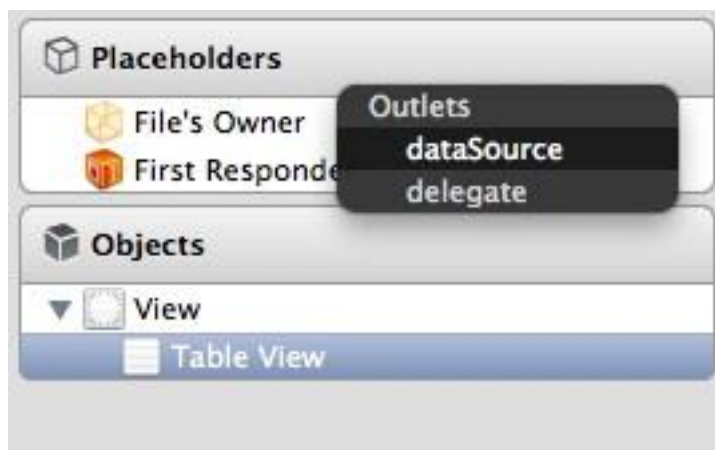
连接数据源（DataSource）和委托（Delegate）

像第一个 Hello World 应用程序一样，我们需要在表视图和创建的 2 个方法之间建立连接。

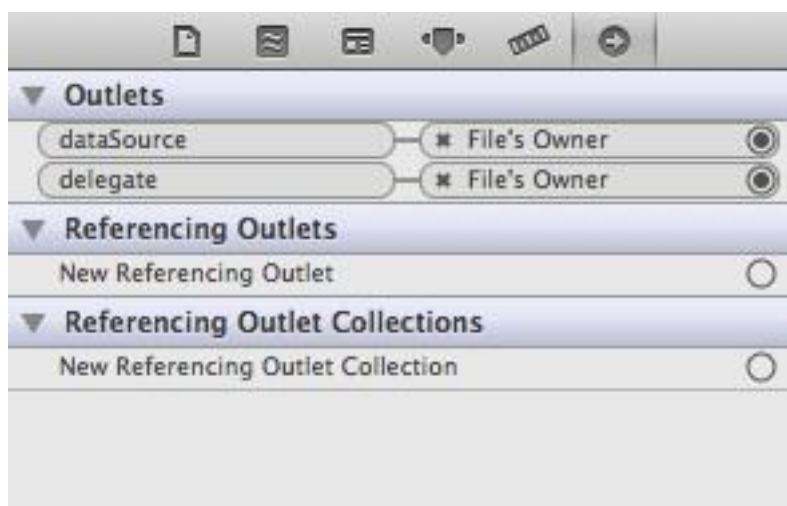
返回 SimpleTableViewController.xib 文件，点击并按住 Control 键，选择表视图，并拖拉到 File's Owner 图上，屏幕如下图所示：



释放按钮，弹出 dataSource 和 delegate 窗口。选择 dataSource，在表视图和它的数据源之间建立连接。重复上述操作，在委托（delegate）上也建立连接。

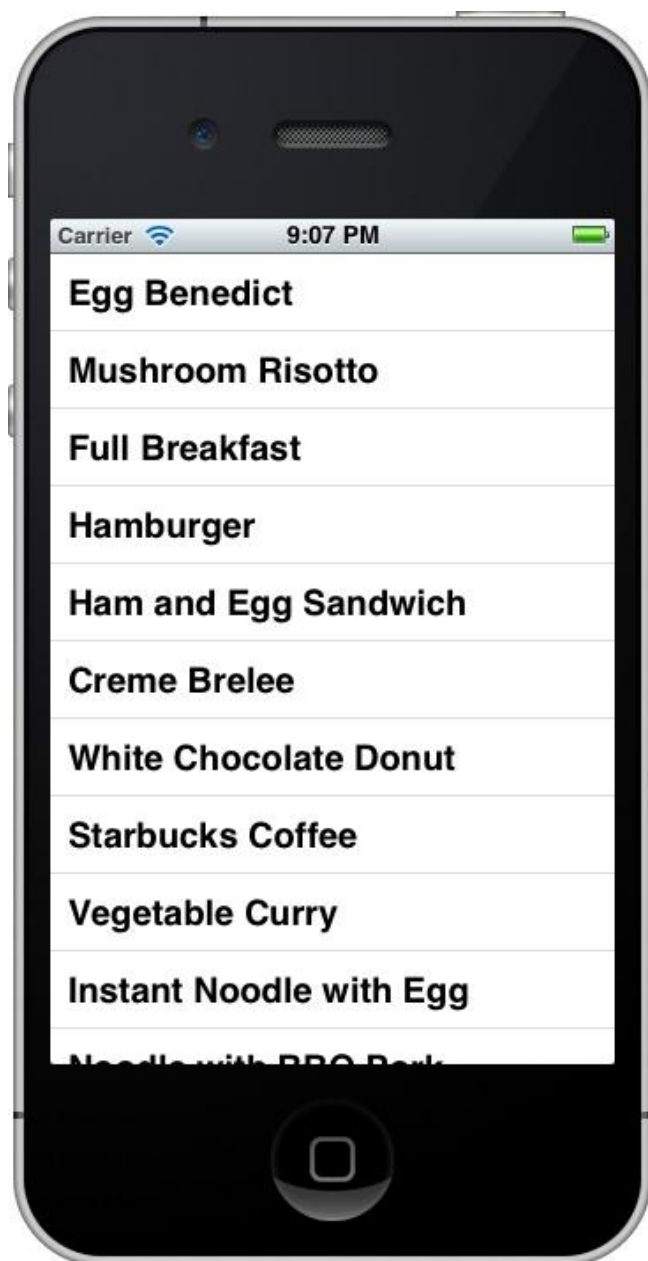


就这些了。为了确保正确建立了连接，你可以再次选择表视图。在工具区域 (Utility Area) 的上部，点击 Connection Inspector 显示所有现存的连接 (最右侧的标签页)。



测试你的应用程序

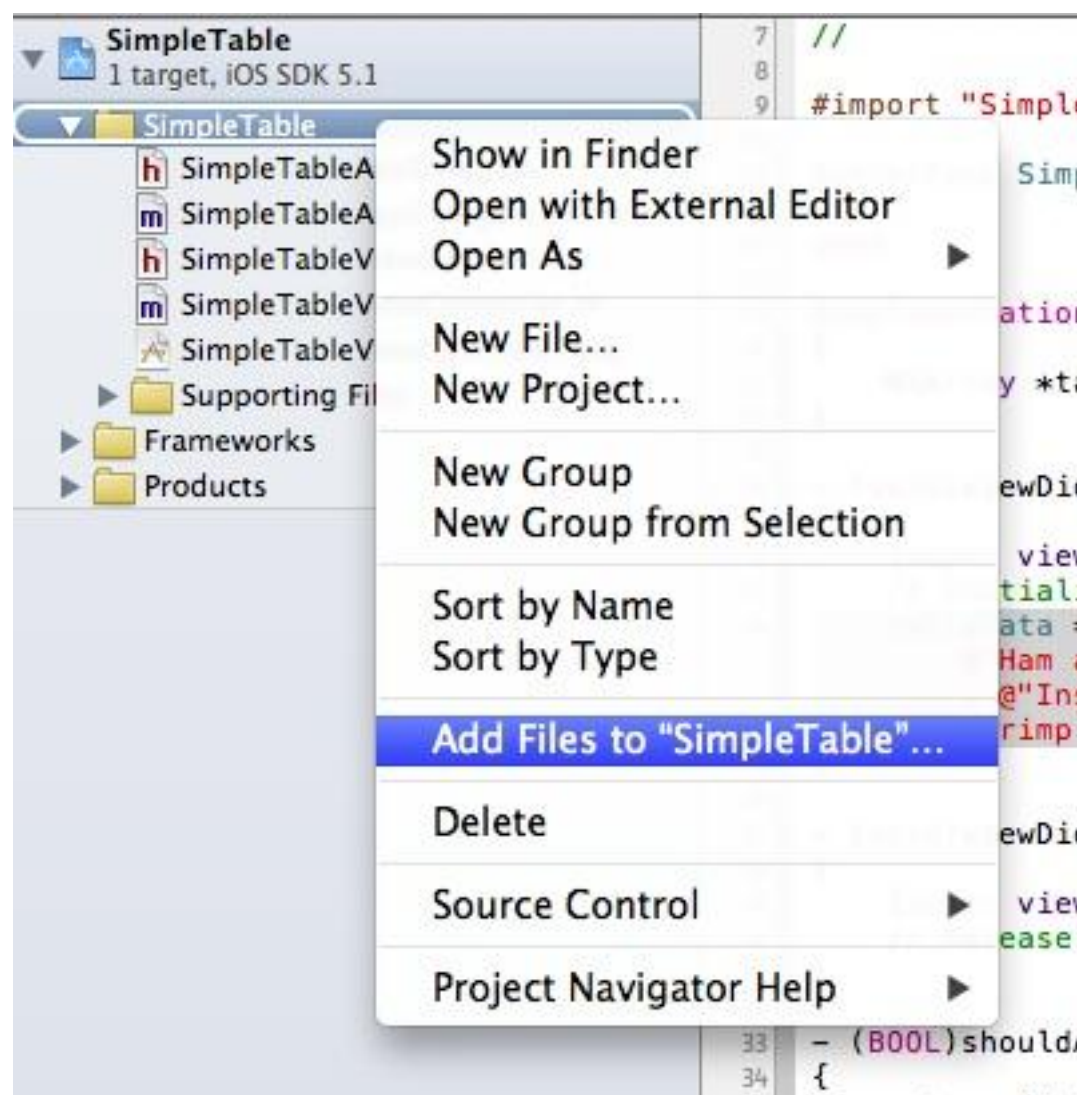
最后，准备测试你的应用程序了。点击 Run 按钮，让模拟器装载你的 App：



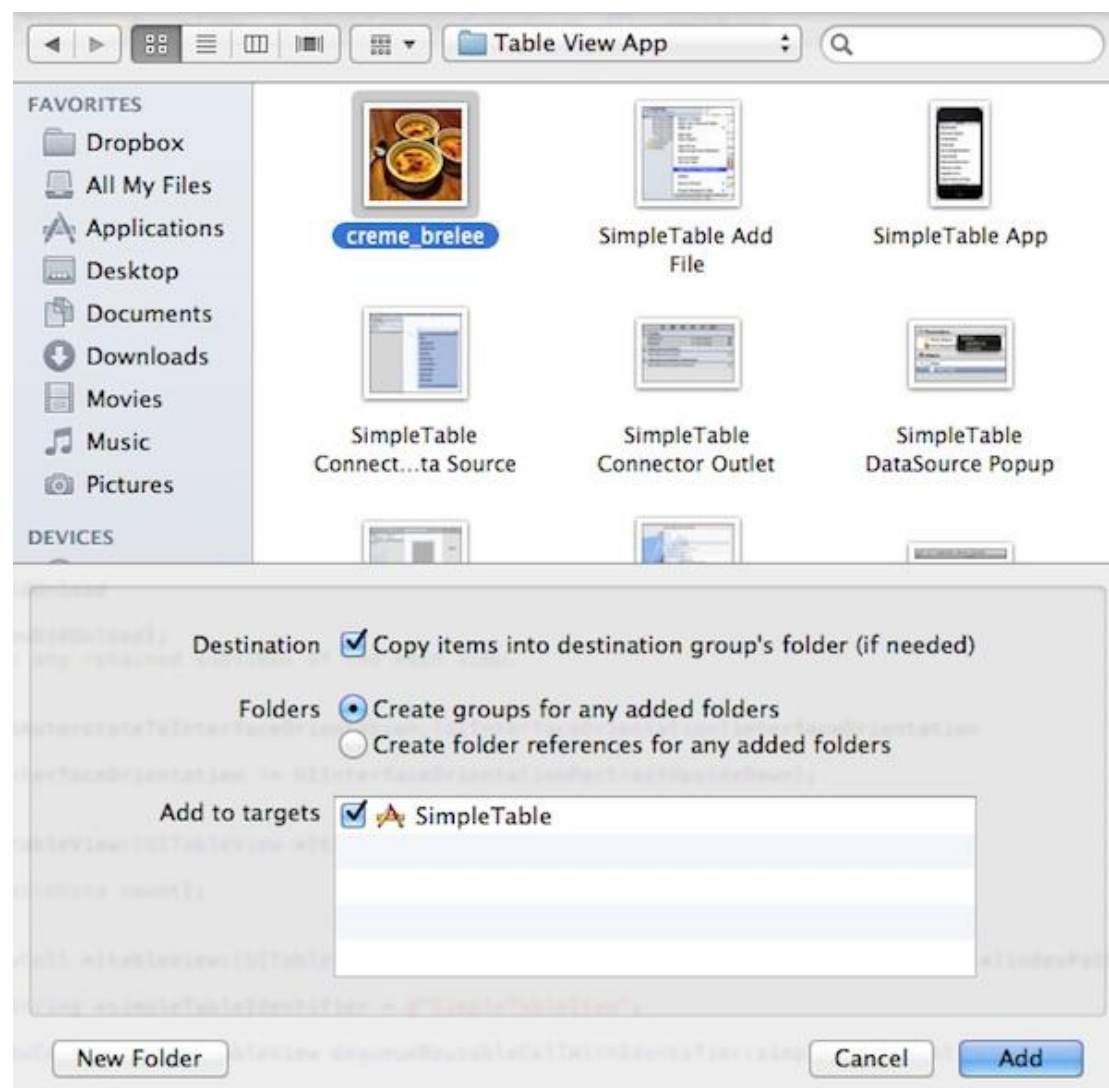
在表视图中添加缩略图

表视图相当朴素，对吧？每一行添加一个图像文件如何？iOS SDK 可以方便实现这一点。你仅需要添加一行代码，实现为每一行添加缩略图。

使用你自己喜欢的图片，确保图片文件为 `crème_brulee.jpg`。在项目导航栏中，右击 `SimplyTable` 文件夹，选择 `Add Files to SimpleTable...`



选择一个图像文件，同时选中 Copy Items to destination group' s folder 复选框。点击 OK 按钮添加该文件。



现在编辑 SimpleTableViewController.m 文件，添加如下代码行到 tableView:cellForRowAtIndexPath 方法中：

cell.imageView.image = [UIImage imageNamed:@"creme_brelee.jpg"];

```
43 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
44 {
45     static NSString *simpleTableIdentifier = @"SimpleTableItem";
46     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:simpleTableIdentifier];
47     if (cell == nil) {
48         cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:simpleTableIdentifier];
49     }
50     cell.textLabel.text = [tableData objectAtIndex:indexPath.row];
51     cell.imageView.image = [UIImage imageNamed:@"creme_brelee.jpg"];
52     return cell;
53 }
```

添加的代码行装载图像文件，并保存在表单元格的图像区域。现在，再次点击 Run 按钮，你的应用程序应该在每一行显示图像了：



接下来是什么？

创建一个表视图很简单，对吧？表视图在 iOS 编程中是一个最常用的元素之一。

如果你跟着教程完成了该应用程序，你应该对如何创建表视图有了一些基本的想法。在本教程中，我尽量保持示例的简单性。实际上，表数据不在代码中直接指定。通常，表数据将从文件、数据库或者其他地方加载。

本文由 [EntLib.com Team](http://EntLib.com) 翻译整理，如你在创建 App 中遇到问题，欢迎访问

[EntLib.net](#) 留言和提交你的问题。

英文原文连接：[iOS Programming Tutorial: Create a Simple Table View App](#)

第四部分 :定制 UITableView 表视图单元格


之前，我们已经创建了一个[简单的表视图 App](#)，用来显示菜单列表和图片。在本教程中，我们将继续改进该 App，使其效果更佳。

- 1) 实现不同的行显示不同的图片 – 之前所有的菜单行显示相同的缩略图，每一个菜单行显示自己的缩略图不是更好么？
- 2) 定制表视图单元格 – 而不是使用表视图单元格的默认样式，我们将演示如何创建自己的样式风格。

显示不同的缩略图

在修改代码之前，我们回顾一下在数据行上显示缩略图的代码：

```
43 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
44 {
45     static NSString *simpleTableIdentifier = @"SimpleTableItem";
46
47     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:simpleTableIdentifier];
48
49     if (cell == nil) {
50         cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:
51             simpleTableIdentifier];
52     }
53
54     cell.textLabel.text = [tableData objectAtIndex:indexPath.row];
55     cell.imageView.image = [UIImage imageNamed:@"creme_brelee.jpg"];
56
57     return cell;
58 }
```



上图，我们添加了代码行，指示 UITableView 在每一行显示 creme_brelee.jpg 缩略图。显然，为了显示不同的图片，我们需要更改这一行代码。之前解释过，在每一行数据显示之前，iOS 自动调用一次 cellForRowAtIndexPath 方法。

```
- (UITableViewCell *)tableView:(UITableView *)tableView
```

```
cellForRowAtIndexPath:(NSIndexPath *)indexPath
```

查看该方法的参数，在调用的时候，传入 indexPath 参数。indexPath 参数包含了数据行的行编号（也就是节点编号）。你可以使用 indexPath.row 属性获知当前指向的行。和数组一样，数据行从 0 开始。也就是说，indexPath.row 属性针对表的第一行返回 0。

为了显示不同的缩略图，我们将添加一个新的数组（如 thumbnails），存放缩略图的文件名称：

```
@implementation SimpleTableViewController
```

```
{
```

```
    NSArray *tableData;
```

```
    NSArray *thumbnails;
```

```
}
```

```
- (void)viewDidLoad
```

```
{
```

```
    [super viewDidLoad];
```

```
    // Initialize table data
```

```
    tableData = [NSArray arrayWithObjects:@"Egg Benedict",
```

```
    @"Mushroom Risotto", @"Full Breakfast", @"Hamburger", @"Ham and
```

```
    Egg Sandwich", @"Creme Brelee", @"White Chocolate Donut",
```



```
@ "Starbucks Coffee", @ "Vegetable Curry", @ "Instant Noodle with Egg",  
@ "Noodle with BBQ Pork", @ "Japanese Noodle with Pork", @ "Green  
Tea", @ "Thai Shrimp Cake", @ "Angry Birds Cake", @ "Ham and Cheese  
Panini", nil];
```

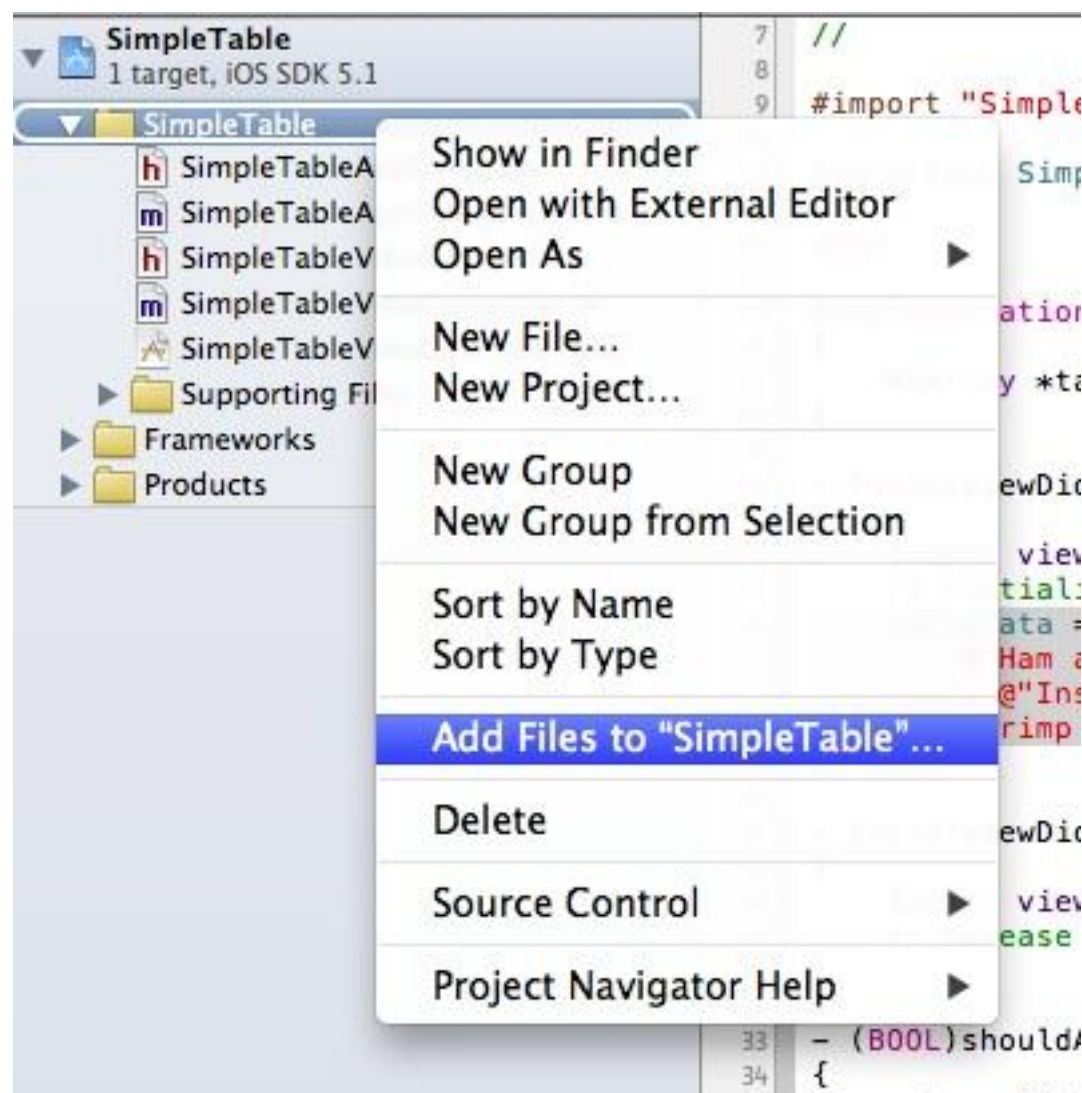
```
// Initialize thumbnails  
  
thumbnails = [NSArray arrayWithObjects:@ "egg_benedict.jpg",  
@ "mushroom_risotto.jpg", @ "full_breakfast.jpg", @ "hamburger.jpg",  
@ "ham_and_egg_sandwich.jpg", @ "creme_brelee.jpg",  
@ "white_chocolate_donut.jpg", @ "starbucks_coffee.jpg",  
@ "vegetable_curry.jpg", @ "instant_noodle_with_egg.jpg",  
@ "noodle_with_bbq_pork.jpg", @ "japanese_noodle_with_pork.jpg",  
@ "green_tea.jpg", @ "thai_shrimp_cake.jpg", @ "angry_birds_cake.jpg",  
@ "ham_and_cheese_panini.jpg", nil];  
}
```

从上面的代码可以看出 我们初始化 thumbnails 数组为缩略图文件名称列表。

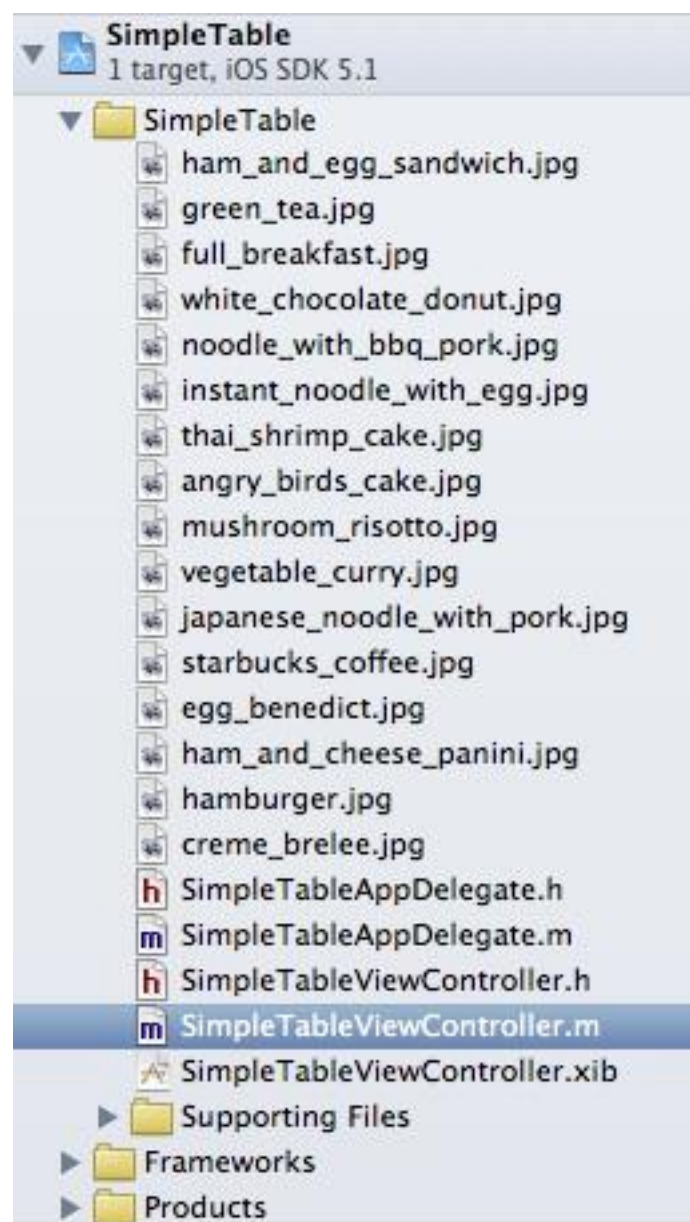
图片的顺序和 tableData 数组记录保持一致。

为了你方便使用缩略图，你可以到[这里下载图片包](#)，并添加到你的项目中。确保

Copy items into destination group' s folder 选中。



在添加这些图片文件之后，你可以在项目导航栏看到这些文件，如下图所示：



最后，更改 `cellForRowAtIndexPath` 方法的代码行：

```
cell.imageView.image = [UIImage imageNamed:[thumbnails  
objectAtIndex:indexPath.row]];
```

[thumbnails objectAtIndex:indexPath.row] 有什么用途？

该代码行负责为特定行检索缩略图的文件名。对第一行，`indexPath.row` 属性

返回 0，我们使用 `objectAtIndex` 方法，从 `thumbnails` 数组获得第一张图片（如 `egg_benedict.jpg`）

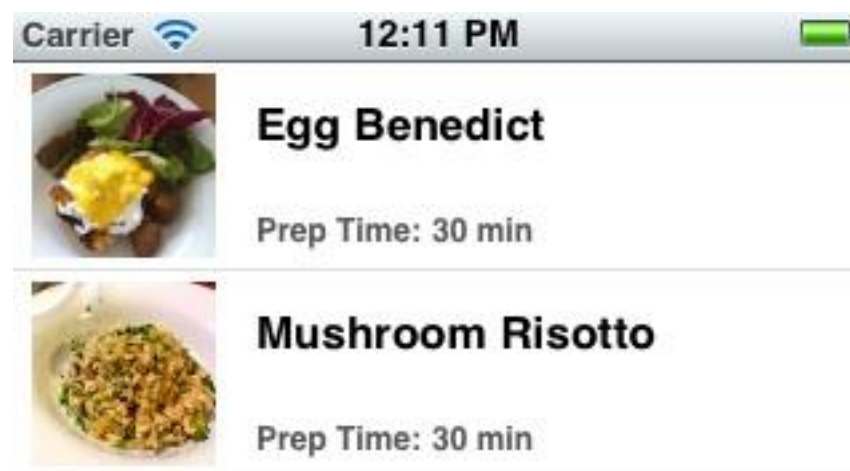
在你保存所有的更改之后，再次运行你的 App。现在，应该每一数据行显示不同的缩略图了。



定制表视图单元格

现在 App 界面好看一点了吧？我们将通过定制表单元格让界面更好看一点。目

前，我们使用了表视图的默认样式，缩略图的位置和大小是固定的。何不让缩略图更大一点，同时显示每一个菜需要准备的时间呢？如下图所示：



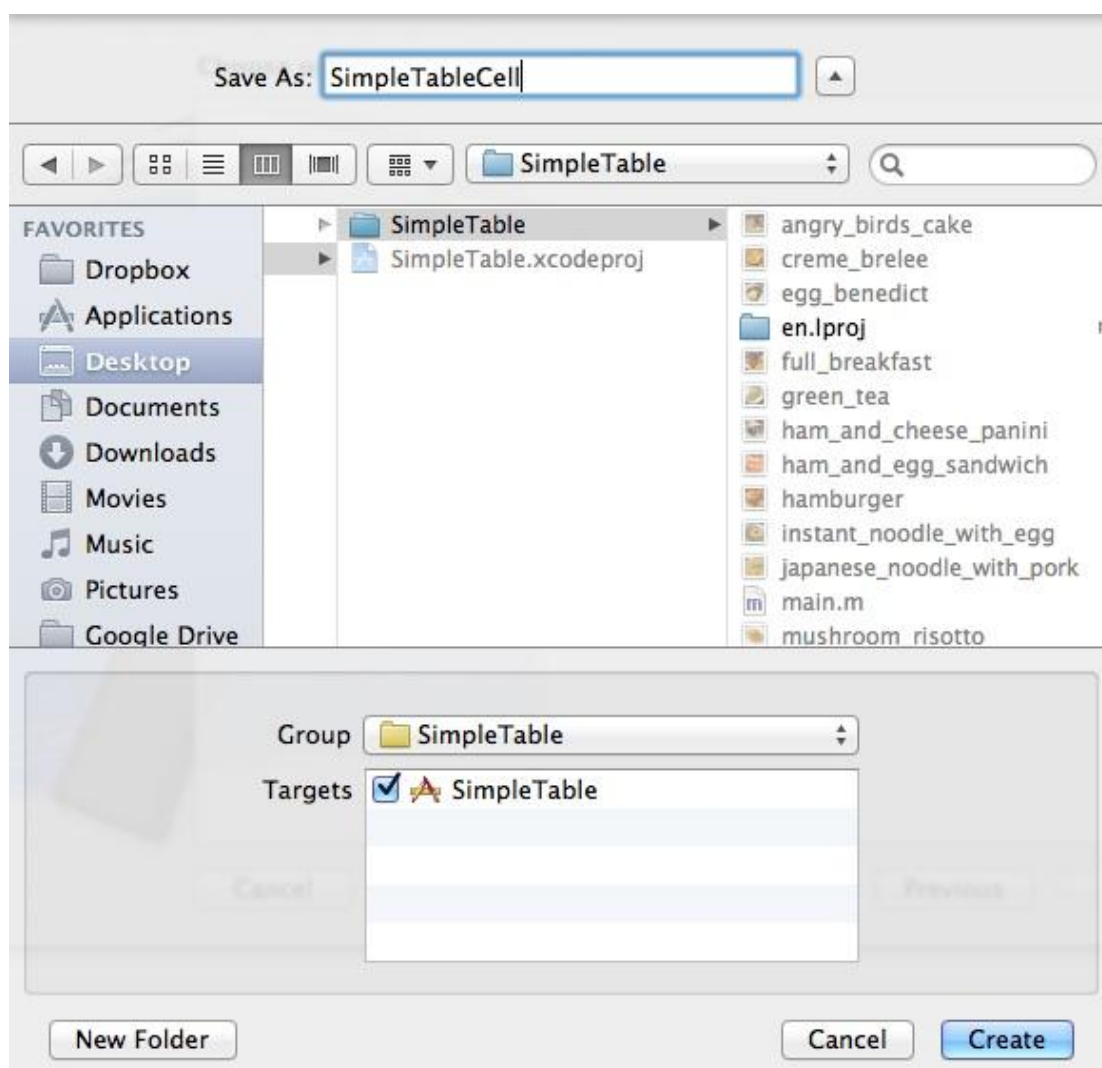
设计单元格

在这种情况下，需要创建和设计你自己的表单元格。返回 Xcode，在项目导航栏中，右击 SimpleTable 文件夹，并选择 New File ...。

为了设计我们自己的表单元格，我们需要为单元格创建一个新的 Interface Builder 文件。针对这个例子，我们仅需要从一个 Empty 的用户界面开始，点击 Next 按钮继续。

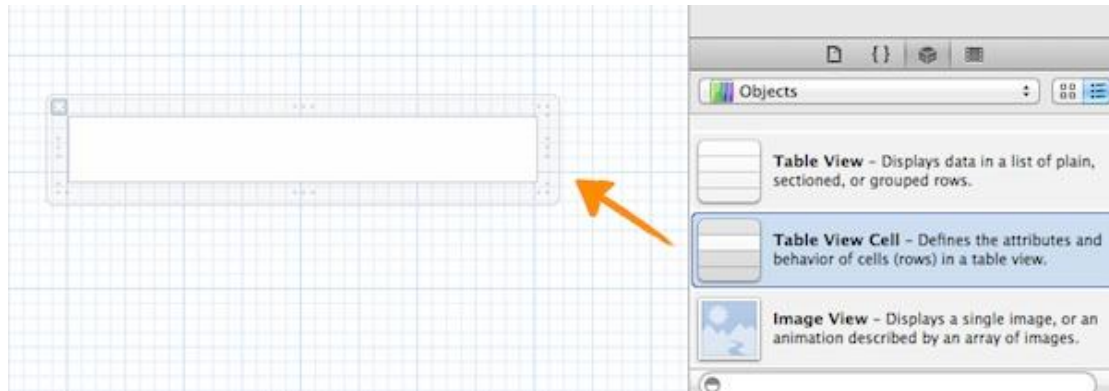


在弹出选择 device family 窗口时，选择 iPhone，并点击 Next 按钮继续。保存文件为 SimpleTableViewCell。

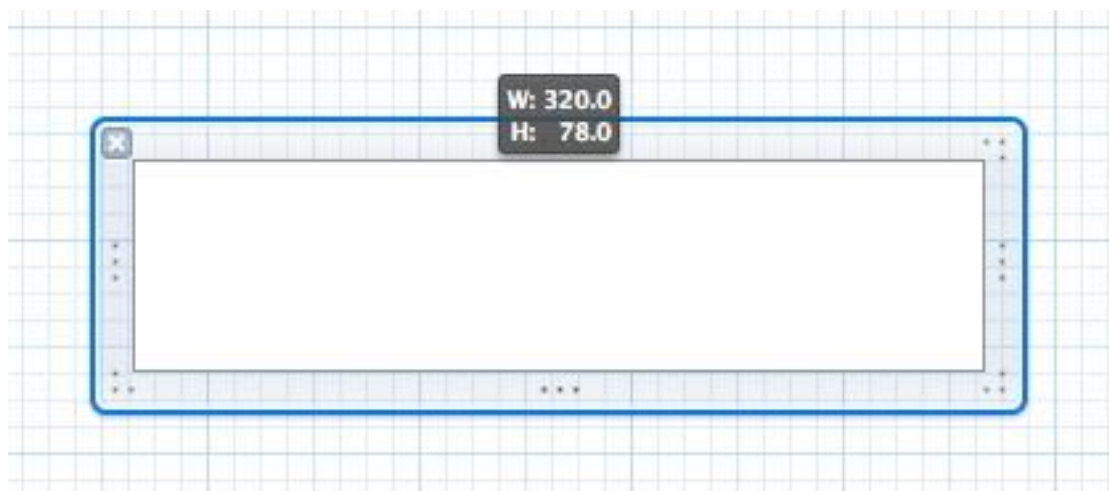


在创建文件之后，可以在项目导航栏可以看到这个文件。选择 SimpleTableCell.xib 文件切换到 Interface Builder 窗口。下面，我们将设计定制单元格的用户界面。

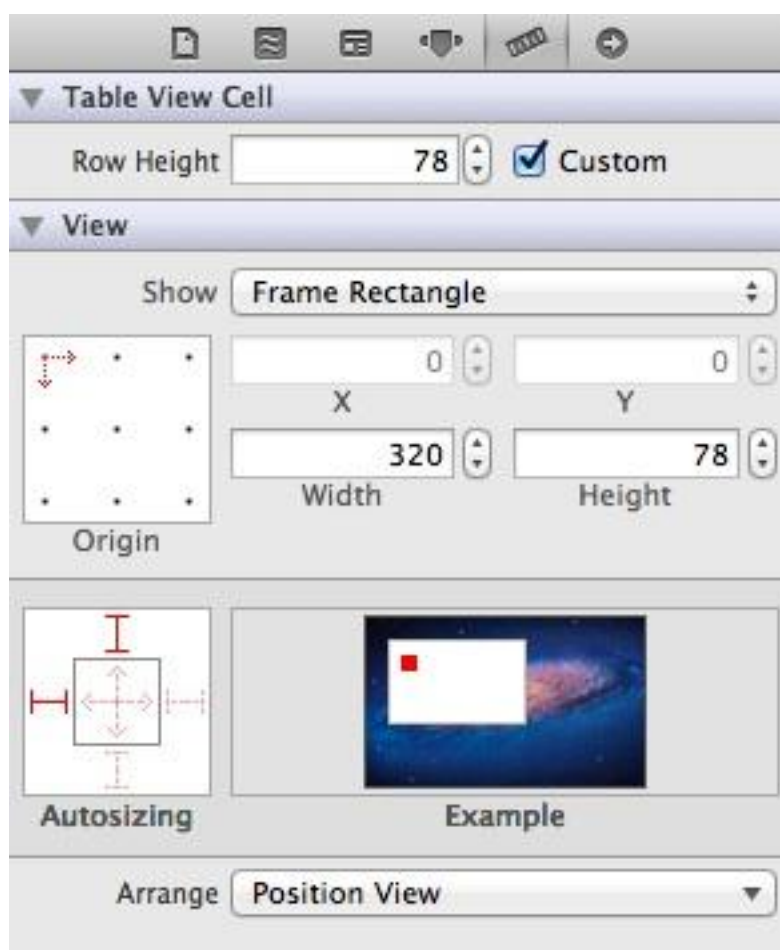
在对象库(Object Library)中，选择 Table View Cell 控件，并拖拉到 Interface Builder 窗口的设计区域。



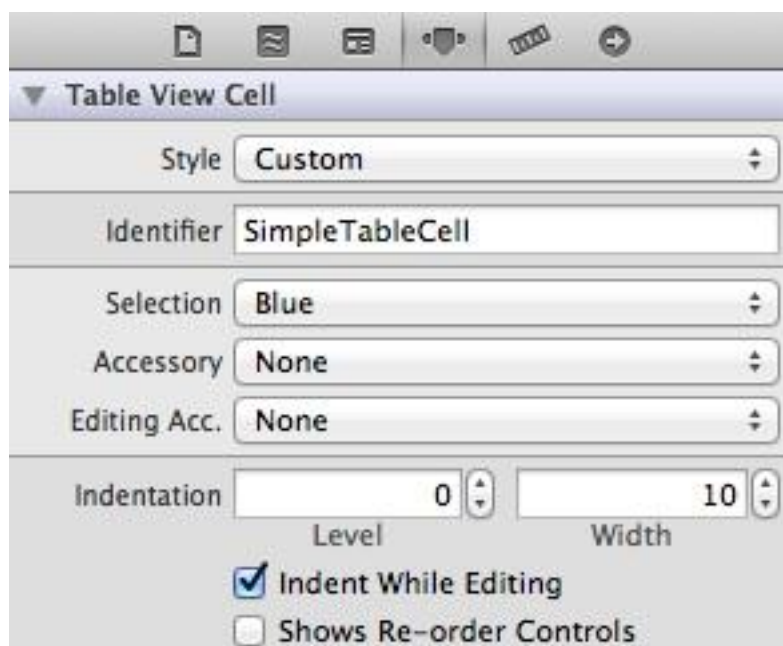
为了容纳大尺寸的缩略图，我们需要更改单元格的高度。按住单元格的底部或顶部边缘，拖拉到 78 高度。



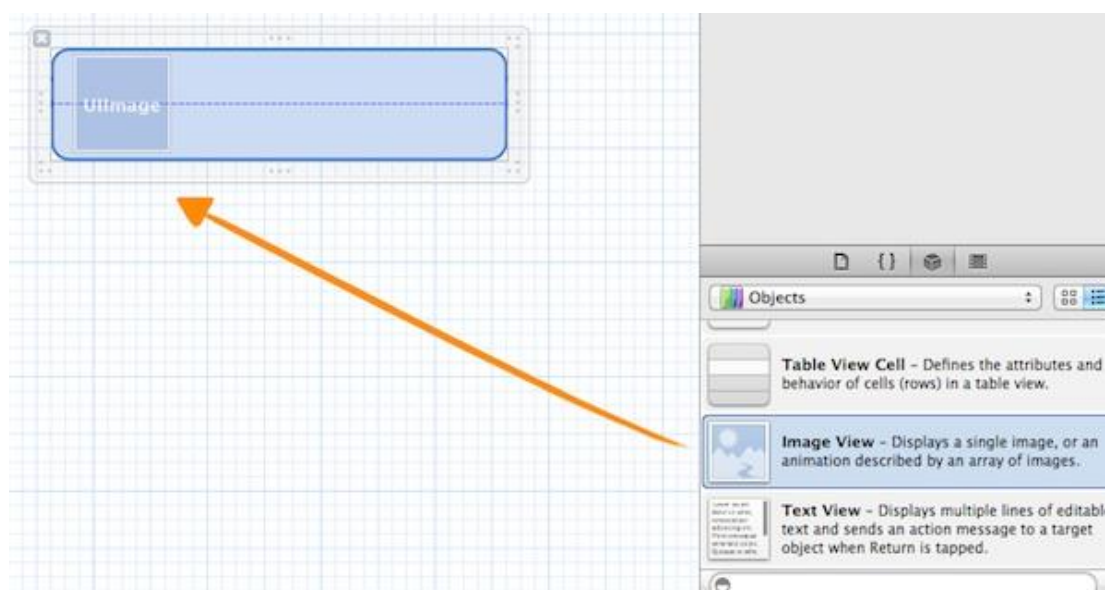
另外一个方法是，你可以使用 Size Inspector 窗口来更改高度：



接着，选择工具区域（Utility Area）上部分的 Attribute Inspector 窗口，设置定制单元格的 Identifier 属性值为：SimpleTableCell。这个 Identifier 将在后面的代码会用到。



在配置好 Table Cell View 之后 ,我们将放置其他控件在里面。选择 Image View 控件 ,并拖曳到 Table Cell View 里面。



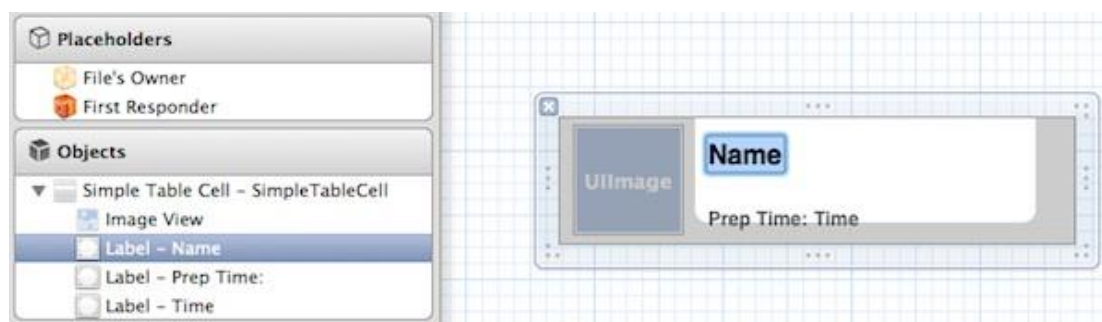
图像视图控件用来显示缩略图。你可以改变该控件的尺寸 ,以适合单元格的大小。

你可设置图像控件的高度和宽度为 69 像素 ,供你参考。

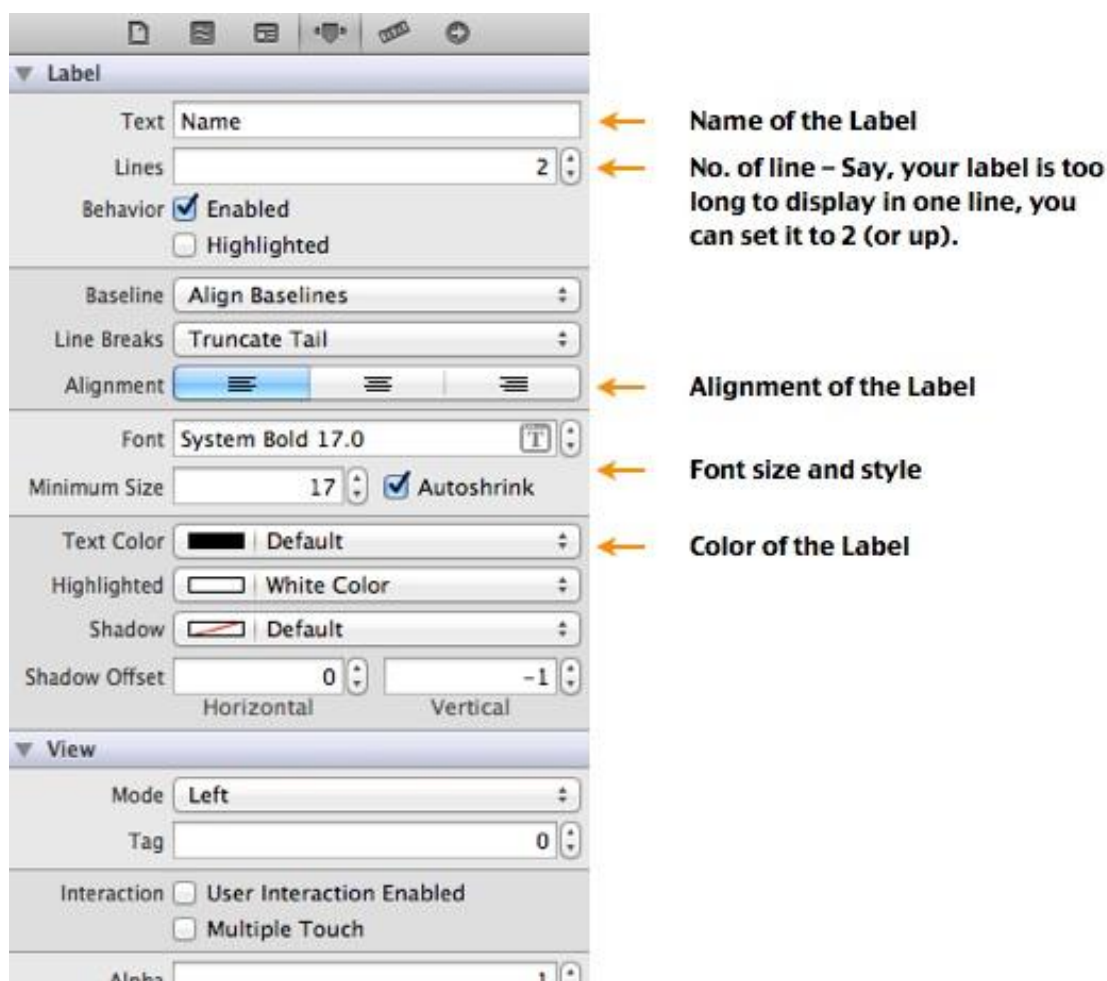
接着 ,我们添加 3 个标签 Labels 控件 : Name , Prep Time 和 Time。Name 标签显示菜谱名称 ; Prep Time 是一个静态标签 ,显示文本 Prep Time (准备时间); 最后 , Time 标签是一个动态标签 ,显示特定菜谱的实际准备时间。

添加 Label 控件 ,从对象库(Object Library)选择 Label 控件 ,拖曳到单元格。

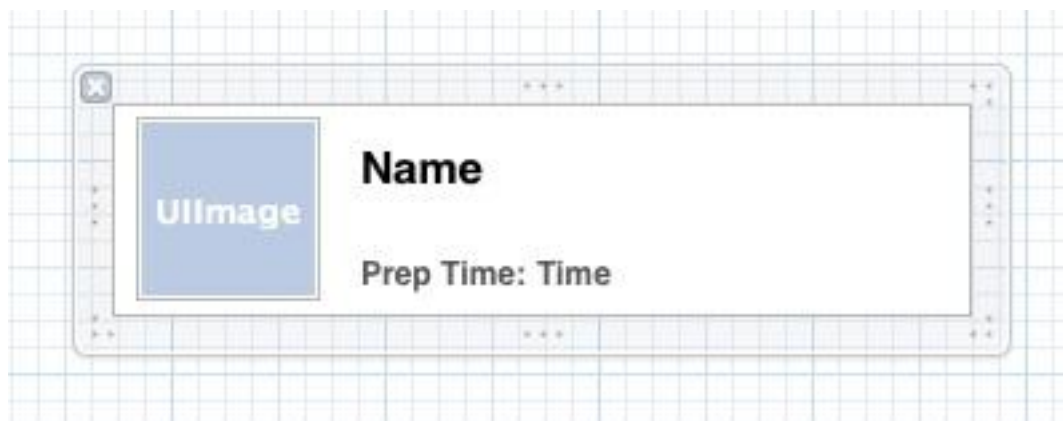
你可以双击 Label 更改其名称。



你可能注意到字体大小和样式和上图不一样。可选择 Label ,然后点击 Attribute Inspector 窗口。在该窗口 , 你可以修改 Font 设置和最小化字体大小 ; 同时 , 还可以更改字体颜色和对方方式。



最终设计的 UI 如下图所示 :

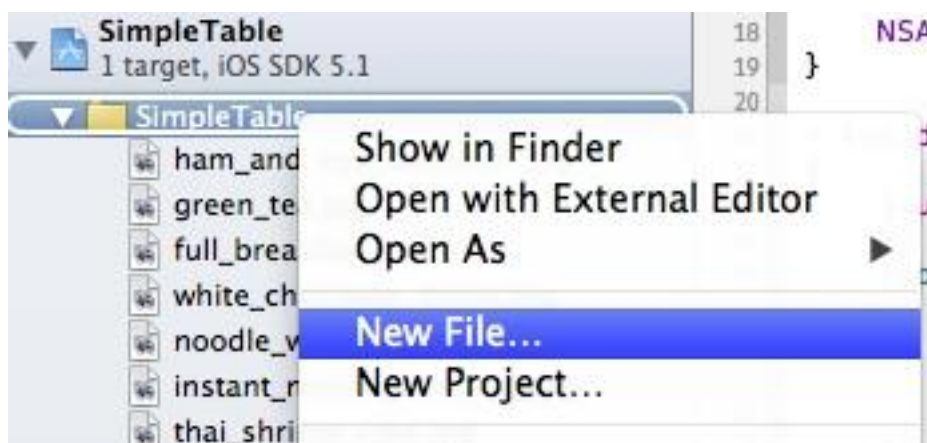


为定制单元格创建类

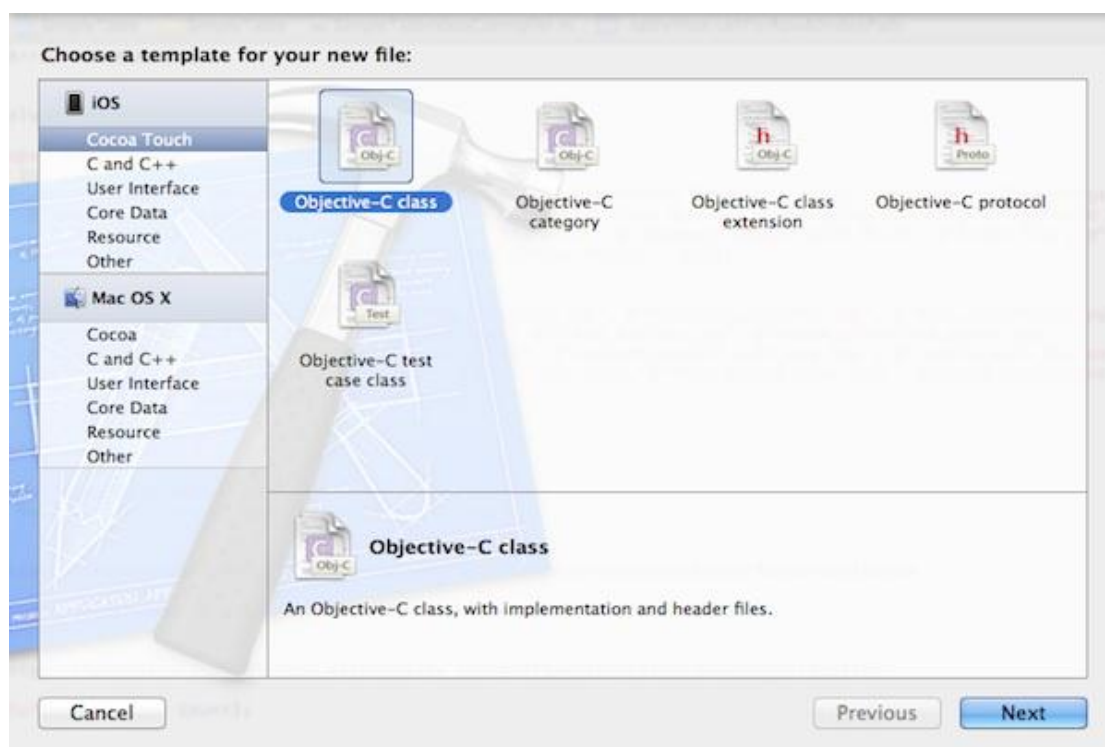
目前为止，我们已经设计好了表单元格。但是如何改变定制单元格的标签值呢？

我们将为定制的表视图单元格创建一个新类。这个类表示定制单元格的底层数据模型。

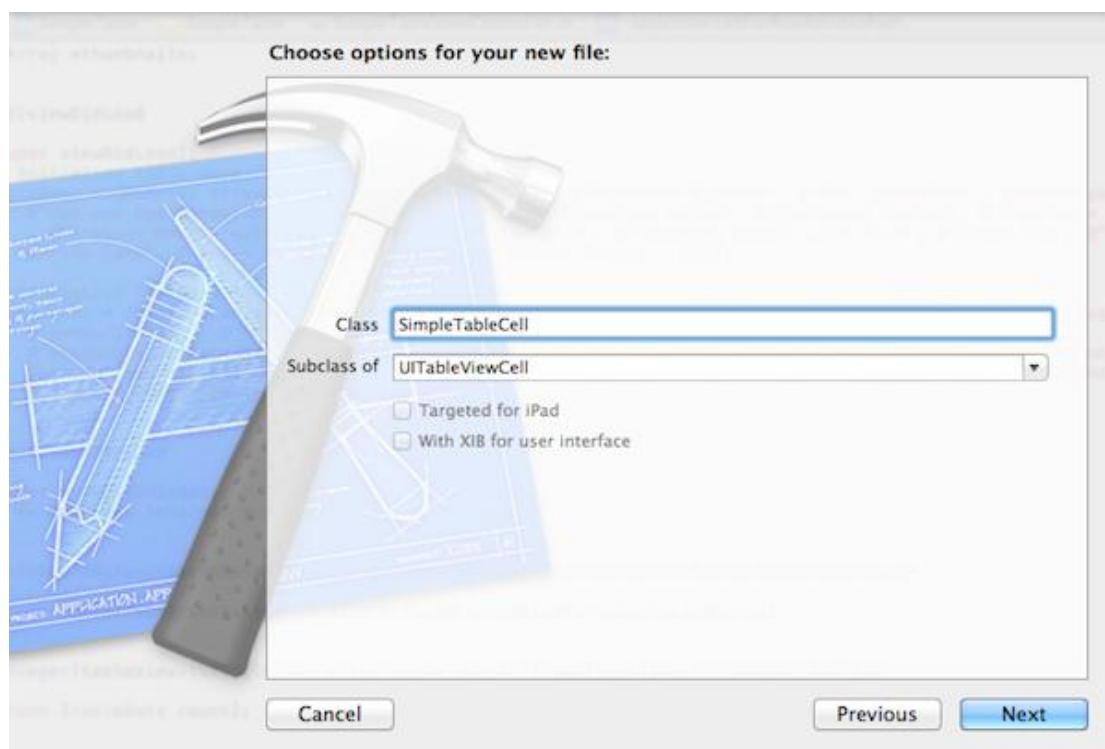
和之前一样，右击项目导航栏中 SimpleTable 文件夹，选择 New File ...



接着 Xcode 弹出一个窗口，让你选择模板。因为我们需要为定制表视图单元格创建一个新类，因此选择 Cocoa Touch 下面的 Objective-C class，然后点击 Next 按钮。



输入 SimpleTableCell 作为类名，并选择 UITableViewCell 作为 Subclass of 下拉选项。



点击 Next 按钮，保存文件在 SimpleTable 项目文件夹中，并点击 Create 按

钮继续。Xcode 应该在项目导航栏中创建了 2 个文件：SimpleTableCell.h 和 SimpleTableCell.m。

前面讲过，SimpleTableCell 类作为定制单元格的数据模型。在单元格中，有 3 个值是需要变更的：缩略图、菜谱名称标签和时间标签。在类中，我们将添加 3 个属性，分别表示这些动态值。

打开 SimpleTableCell.h 文件，在 @end 代码行前面添加如下属性：

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;  
  
@property (nonatomic, weak) IBOutlet UILabel *prepTimeLabel;  
  
@property (nonatomic, weak) IBOutlet UIImageView  
*thumbnailImageView;
```

Property 和 Outlet

上述代码定义了 3 个实例变量，下面会用来和 Interface Builder 中的表视图单元格进行关联。关键字 @property 用来在类中定义一个属性：

```
@property (attributes) type name;
```

关于前面的代码，weak 和 nonatomic 是 property 的特性。UILabel 和 UIImageView 是类型，nameLabel、prepTimeLabel 和 thumbnailImageView 是变量名称。

IBOutlet 是什么？你可将 IBOutlet 理解为一个指示符（indicator）。为了关

联实例变量和表视图单元格（如 SimpleTableCell.xib）中的元素，我们使用 IBOutlet 关键字，让 Interface Builder 知道它们允许建立连接。随后，你将看到如何在这些 Outlets 和 Interface Builder 中的对象（或控件）建立连接。

现在，打开 SimpleTableCell.m 文件，跟随在 @implementation SimpleTableCell 代码行下面，添加如下代码：

```
@synthesize nameLabel = _nameLabel;

@synthesize prepTimeLabel = _prepTimeLabel;

@synthesize thumbnailImageView = _thumbnailImageView;
```

@synthesize 指令

@synthesize 关键字告诉编译器自动生成代码，用来访问前面定义的属性。如果你忘记包含这些指令，Xcode 将抛出如下异常：



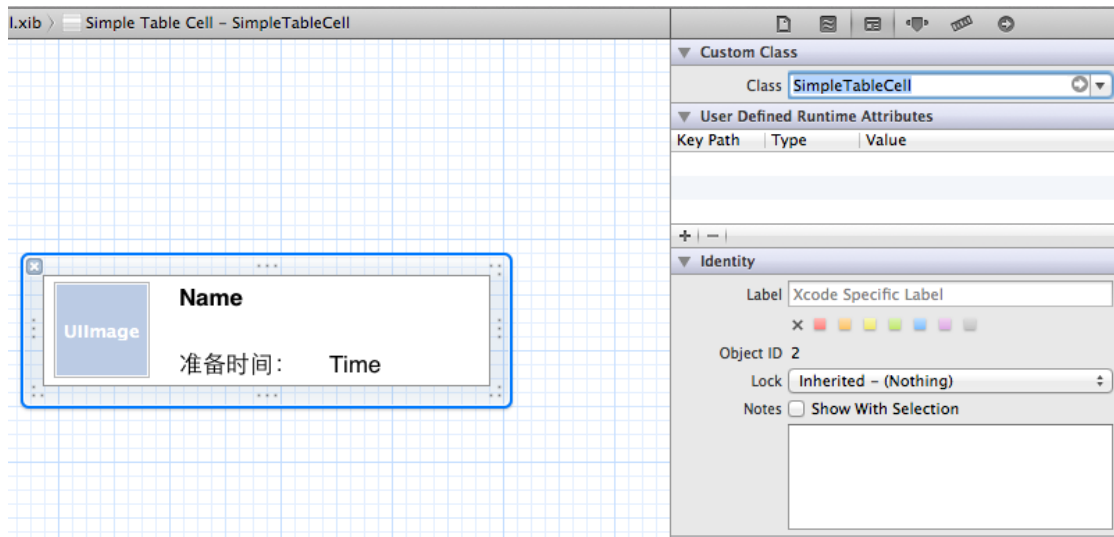
```
9  #import "SimpleTableCell.h"
10
11  @implementation SimpleTableCell  ⚠ Property 'nameLabel' requires method 'nameLabel' to be defined - use @synthesize.
12  @synthesize prepTimeLabel = _prepTimeLabel;
13  @synthesize thumbnailImageView = _thumbnailImageView;
```

建立连接

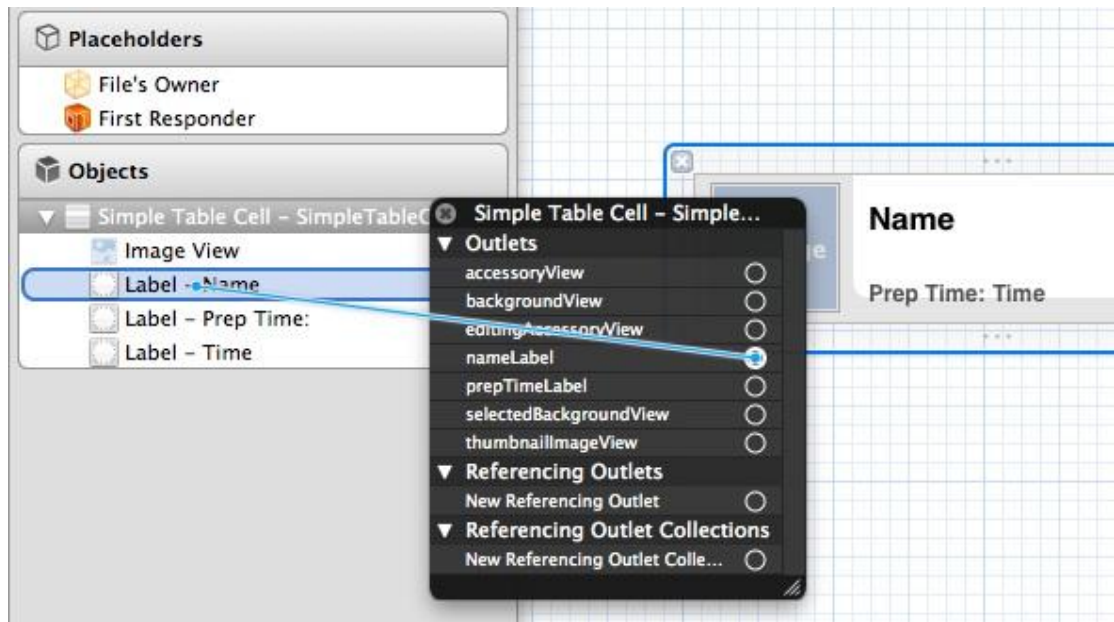
保存上述更新的代码，然后选择 SimpleTableCell.xib 文件，返回 Interface Builder 界面。现在，我们将在类的属性（或实例变量）和界面上的 Label、ImageView 控件上建立连接。

首先, 选择单元格, 并在 Identity Inspector 窗口中, 选择类 SimpleTableCell。

这里将关联单元格视图 (cell view) 和前面创建的定制类。



现在, 我们将和属性建立连接。右击 Objects 下面的 SimpleTableCell, 显示 Outlets 监视器 (inspector)。点击并按住 nameLabel 右侧的圆形图标, 拖拉到 Label – Name 对象上。Xcode 自动建立它们之间的连接。

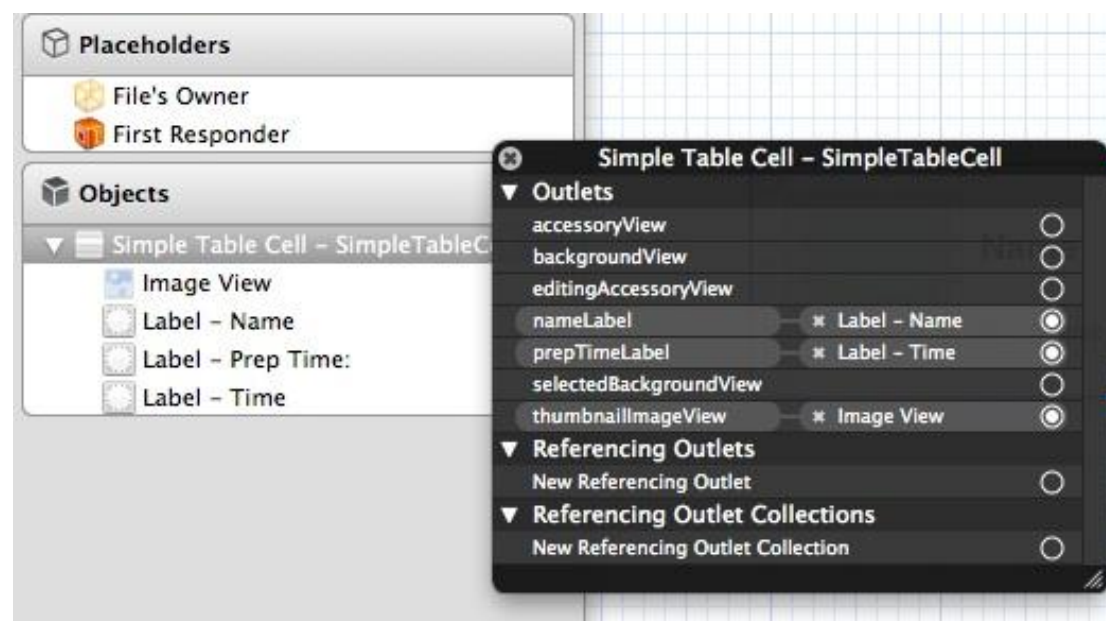


重复上面的步骤, 为 prepTimeLabel 和 thumbnailImageView 建立连接。

(1) 连接 prepTimeLabel 和 Label-Time 对象;

(2) 连接 thumbnailImageView 和 UIImageView 对象；

在你完成所有连接之后，界面如下所示：



更新 SimpleTableViewController

我们已经完成了定制表视图单元格的设计和代码编写工作。最后，我们最最后一部分的更新 – 在 SimpleTableViewController 中使用定制单元格。

再次访问 SimpleTableViewController.m 中用来创建表视图行的代码：

```
- (UITableViewCell *)tableView:(UITableView *)tableView  
cellForRowAtIndexPath:(NSIndexPath *)indexPath  
{  
  
    static NSString *simpleTableIdentifier = @"SimpleTableItem";  
  
    UITableViewCell *cell = [tableView
```

```
dequeueReusableCellWithIdentifier:simpleTableIdentifier];
```

```
    if (cell == nil) {  
        cell = [[UITableViewCell alloc]  
initWithStyle:UITableViewCellStyleDefault  
reuseIdentifier:simpleTableIdentifier];  
    }  
  
    cell.textLabel.text = [tableData objectAtIndex:indexPath.row];  
    cell.imageView.image = [UIImage imageNamed:[thumbnails  
objectAtIndex:indexPath.row]];  
  
    return cell;  
}
```

我们使用默认的表视图单元格（如 UITableViewCell）来显示表元素。为了使用我们之前创建的定制单元格，我们需要修改 SimpleTableView.m 中的这部分代码，如下所示：

```
- (UITableViewCell *)tableView:(UITableView *)tableView  
cellForRowAtIndex:(NSIndexPath *)indexPath  
{
```

```
static NSString *simpleTableIdentifier = @"SimpleTableCell";

SimpleTableCell *cell = (SimpleTableCell *)[tableView
dequeueReusableCellWithIdentifier:simpleTableIdentifier];

if (cell == nil)
{
    NSArray *nib = [[NSBundle mainBundle]
loadNibNamed:@"SimpleTableCell" owner:self options:nil];

    cell = [nib objectAtIndex:0];
}

cell.nameLabel.text = [tableData objectAtIndex:indexPath.row];

cell.thumbnailImageView.image = [UIImage
imageName:[thumbnails objectAtIndex:indexPath.row]];

cell.prepTimeLabel.text = [prepTime objectAtIndex:indexPath.row];

return cell;
}
```


然而,在你更新好代码之后,Xcode 检测到一些错误,并显示在源码编辑器中:

```
59 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
60 {
61     static NSString *simpleTableIdentifier = @"SimpleTableCell";
62
63     UITableViewCell *cell = [UITableViewCell *)[tableView dequeueReusableCellWithIdentifier:simpleTableIdentifier];
64     if (cell == nil)
65     {
66         NSArray *nib = [[NSBundle mainBundle] loadNibNamed:@"SimpleTableCell" owner:self options:nil];
67         cell = [nib objectAtIndex:0];
68     }
69
70     cell.textLabel.text = [tableData objectAtIndex:indexPath.row];
71     cell.thumbnailImageView.image = [UIImage imageNamed:[thumbnails objectAtIndex:indexPath.row]];
72
73     return cell;
74 }
```

这是什么问题呢？刚刚更新的代码要求 SimpleTableViewController 使用 SimpleTableCell 类作为表单元格。然而，SimpleTableViewController 并不知道这个类。这就是 Xcode 显示错误的原因。

在 [前面的教程](#) 中解释过，头文件定义了类的接口。为了让 SimpleTableViewController 知道 SimpleTableCell，我们必须在 SimpleTableViewController.m 文件中引入 SimpleTableCell.h 头文件。

```
8
9 #import "SimpleTableViewController.h"
10 #import "SimpleTableCell.h"
11
```

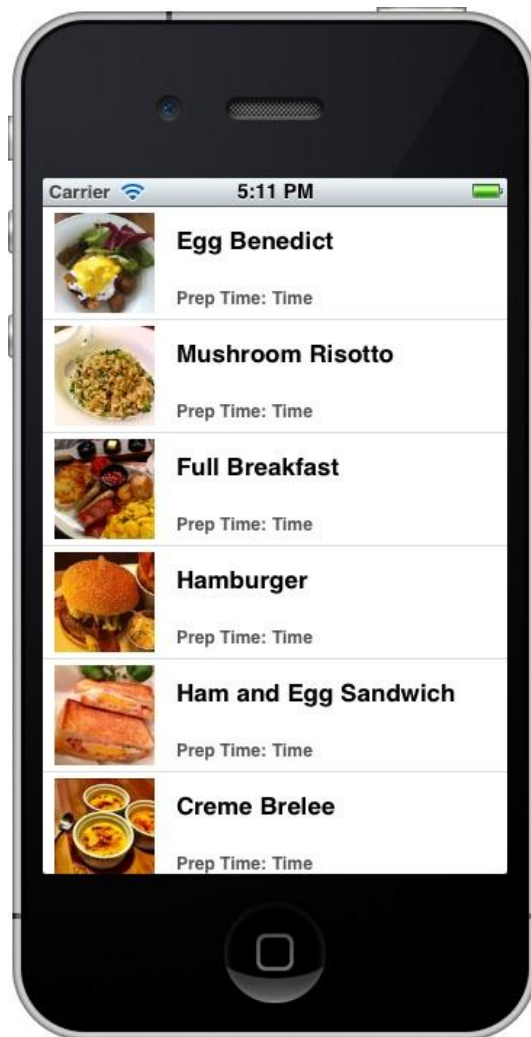


通过引入 SimpleTableCell.h 头文件，SimpleTableViewController 才知道它是什么，就可以使用它了。

最后，因为表单元格的高度更改为 78，因此在 @end 代码行之前添加如下代码：

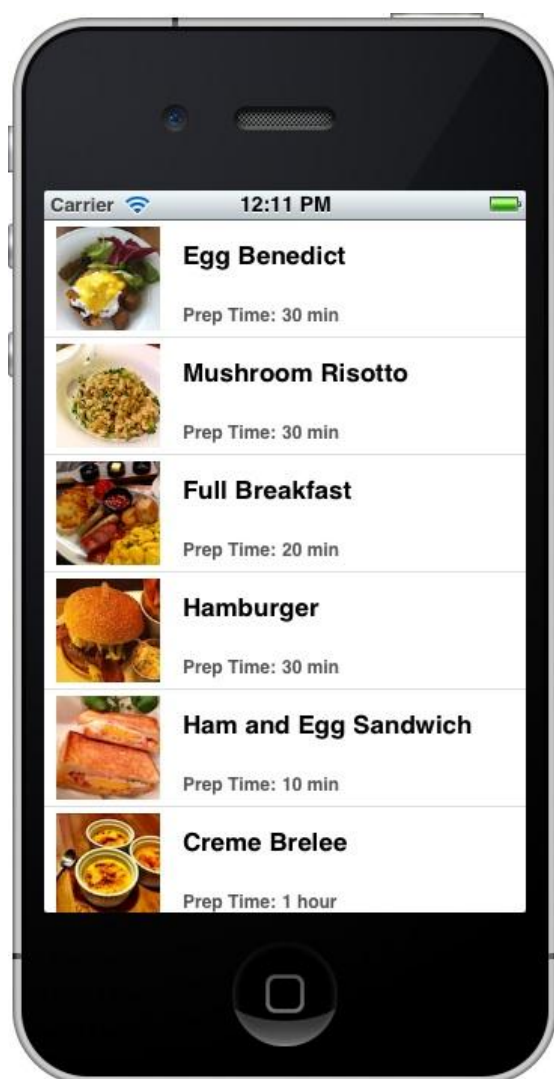
```
- (CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 78;
}
```

现在，点击 Run 按钮，你的 SimpleTable App 运行界面如下图所示：



留给你的作业

你可能注意到 App 显示 Prep Time : 为 Time，将这个作业留给你了。请你尝试修改代码，并更新准备时间。你最终的 App 运行如下图所示：



你可以更进一步，对上述 App 进行改进：

- (1) 显示中文菜单名称；
- (2) 设置中文字体 - 微软雅黑；
- (3) 设置字体的颜色；



接下来讲什么？

学习完本文之后，你应该对 UITableView 了解了，并且知道如何创建自定义的表视图单元格。接下来，我们将进一步讨论如何处理行选择。

本文由 [EntLib.com Team](#) 翻译整理，如你在创建 App 中遇到问题，欢迎访问 [EntLib.net](#) 留言和提交你的问题。

英文原文连接：[Customize Table View Cells for UITableView](#)

第五部分 :如何处理 UITableView 中的行选择

如果你跟着[我们的 iOS 编程教程](#)学习 ,我相信你已经完成了[简单表视图 App](#)和[定制表视图单元格 App](#)项目。目前为止 ,我们关注在表视图上显示数据 ,但是如何知道用户点击了表单元格呢 ?这就是我们在这篇教程中说明的 ,并演示如何处理行选择 (row selection)。

首先 ,重新访问我们的 App ,了解我们将增加什么功能。



在本教程中 ,我们将尝试如下一些变化 :

- (1) 当用户点击一行时 ,显示一个提醒消息 ;
- (2) 当用户选择了一行时 ,显示一个选中的标志 ;

理解 UITableViewDelegate

在我们创建 Simple Table View 应用程序时，我们在 SimpleTableController.h 中定义了 2 个委托 (UITableViewDelegate 和 UITableViewDataSource)：

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

```
@interface SimpleTableViewController : UIViewController  
<UITableViewDelegate, UITableViewDataSource>
```

```
@end
```

前面的教程已经解释过，2 个委托在 Objective-C 中称为协议 (protocol)。为了创建 UITableView，你必须遵守这些协议中定义的要求。

在 iOS 编程中，遇到不同的委托是非常常见的。每一个委托负责特定的角色或任务，保持系统的简洁。任何时候在一个对象需要完成特定任务时，它依赖于另外一个对象去负责处理。这个在系统设计领域通常称为**关注点分离 (Separation of Concern)**。

当你分析 UITableView 类时，它也应用了这一设计概念。这 2 个委托迎合了不同的目标。我们实现的 UITableViewDataSource 委托定义了方法，用来显示表数据；另一方面，UITableViewDelegate 委托则负责处理 UITableView 界面和行选择。

显然，我们将使用 UITableViewDelegate 委托，来实现处理行选择的方法。

处理表视图行选择 (Table Row Selection)

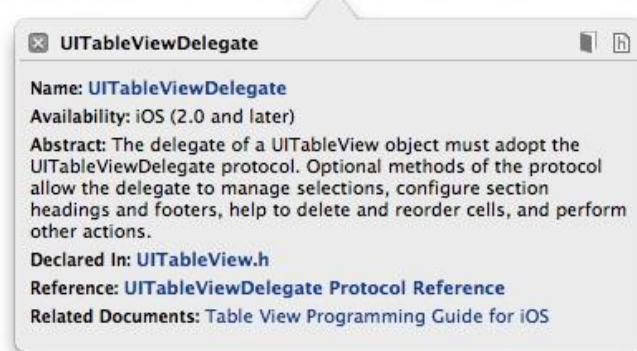
在更改代码之前，我们需要知道：

我们怎么知道 UITableViewDelegate 中的哪一个方法需要实现？

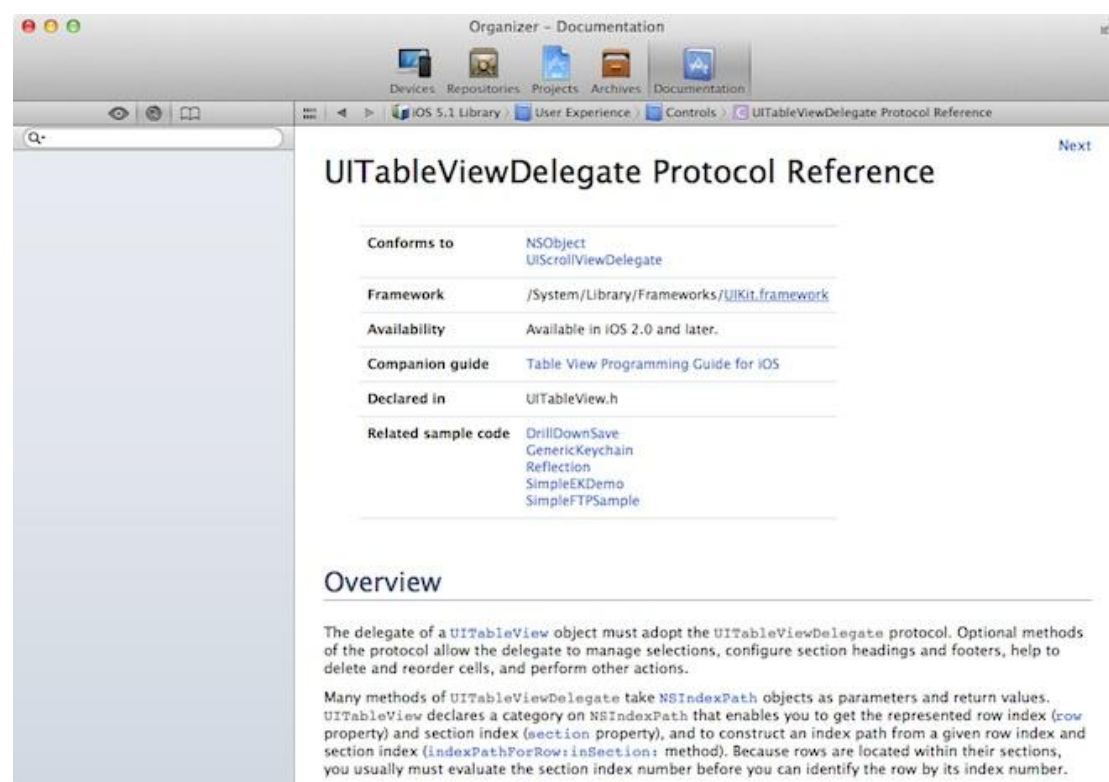
你需要参考 Apple 的 iOS programming reference 文档，有 2 种方法可以访问该文档。你可以方法 Apple 网站的 API 文档，或者直接在 Xcode 中查询。

操作过程：选择类的名称，如 UITableViewDelegate，点击 control-command-？，将弹出 UITableViewDelegate 的 API 文档，如下图所示：

```
@interface SimpleTableViewController : UIViewController <UITableViewDelegate, UITableViewDataSource>
```



点击上图中的 UITableViewDelegate Protocol Reference，显示 API 文档：



如果读完文档，你将发现如下方法用来管理行选择：

- tableView:willSelectRowAtIndexPath:
- tableView:didSelectRowAtIndexPath:

2 个方法都用于行选择。唯一的区别是：在行将要选择的时候，调用 willSelectRowAtIndexPath 方法。通常，你可以使用这个方法阻止选择特定的行。一般情况下，你使用 didSelectRowAtIndexPath 方法，在用户选择一行时，调用这个方法去负责行选择。在这个方法里面，添加代码来指定具体业务行为，在行选择的时候调用。在本教程中，我们将添加一些动作来处理行选择：

- 1) 显示警告消息；
- 2) 显示一个勾选标志，表示该行已经选择；

开始编码吧！

现在我们解释足够清楚了，接下来开始进入有趣的部分 – code, code, code！

在 Xcode 中，打开 SimpleTableViewController.m 文件，在 @end 指令之前

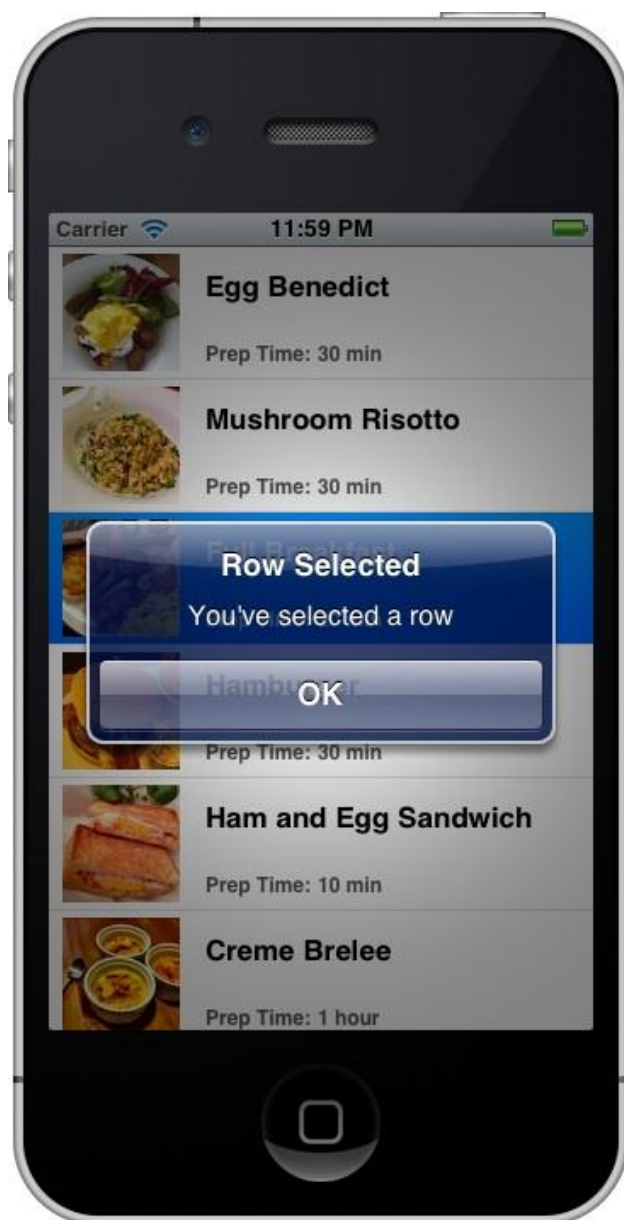
添加如下方法：

```
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    UIAlertView *messageAlert = [[UIAlertView alloc]
                                   initWithTitle:@"行选择"
                                   message:@"你已经选择了一行！" delegate:nil cancelButtonTitle:@"确定"
                                   otherButtonTitles:nil];

    // Display Alert Message

    [messageAlert show];
}
```

代码非常容易理解。在选择一行时，App 创建一个 UIAlertView 对象，并弹出一个警告消息。再次运行 App，在你轻拍一行时，App 显示如下图所示：

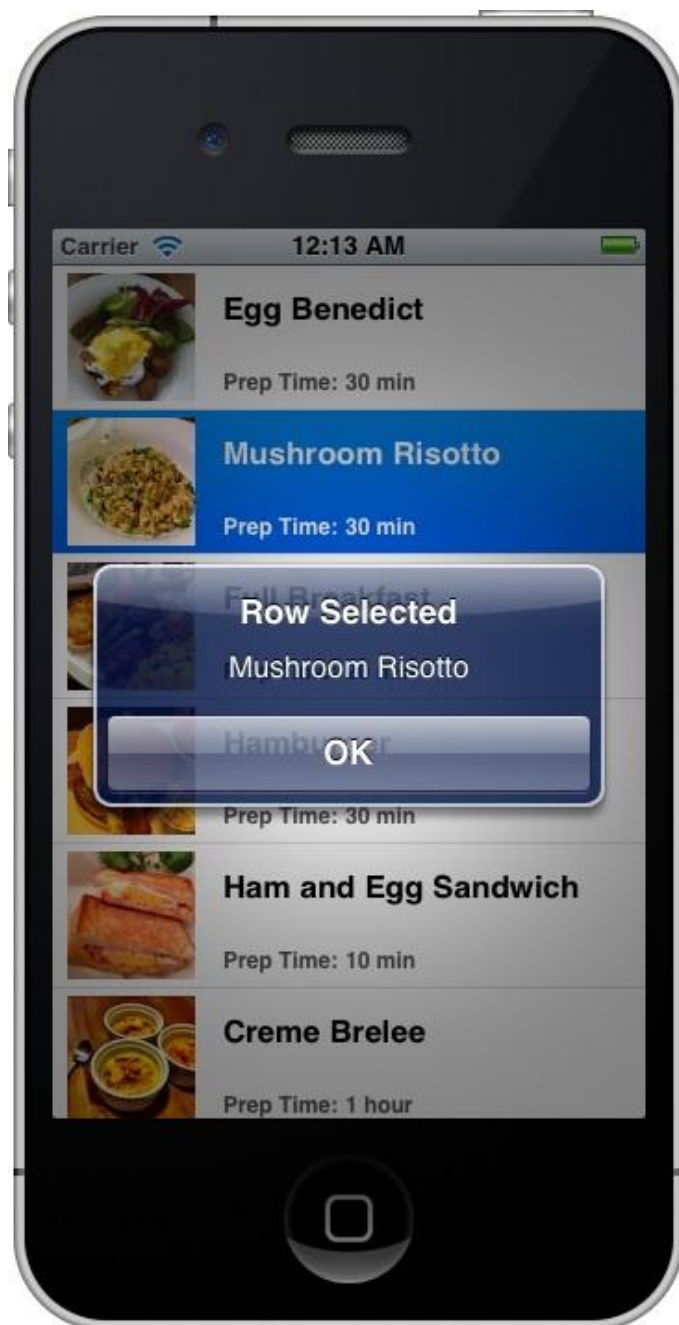


下图是改进之后的中文菜单显示效果图：



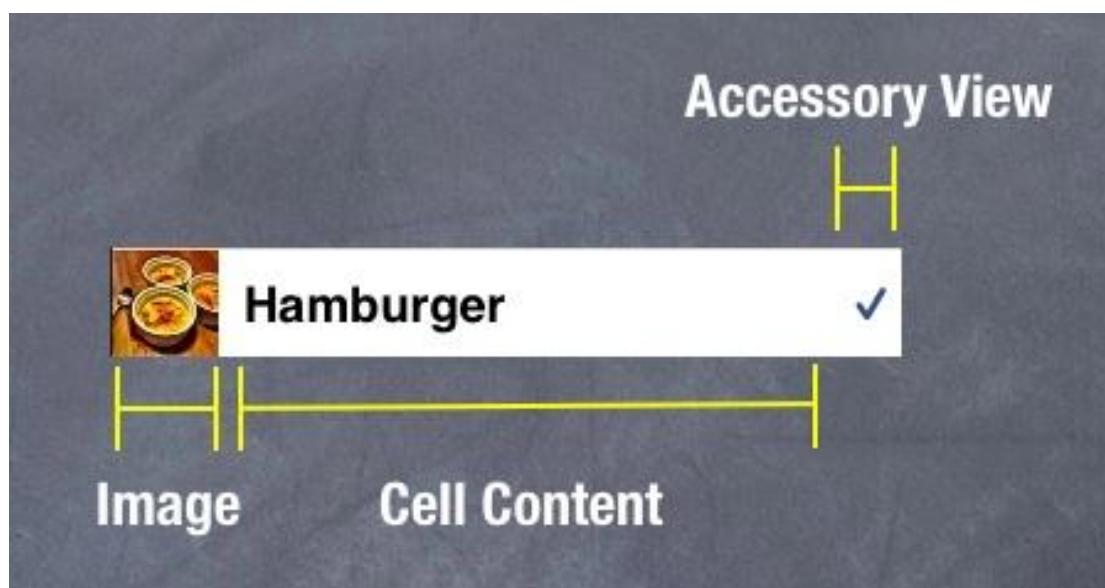
布置给你的作业

现在，在行选择的时候，我们仅仅显示一个通用的消息。最好是显示一条有意义的消息，如下所示：



考虑如何修改代码（提示：`indexPath` 参数包含了所选择行的行号），并显示如上图所示的消息。如果你学习了前面的教程，这个并不困难实现。

很简单，对吧？通过使用委托，可以非常检测到所选择的行。接着，我们将添加一些代码，显示所选择行的标志。在开始之前，让我们看看一个表格单元格的默认内容：

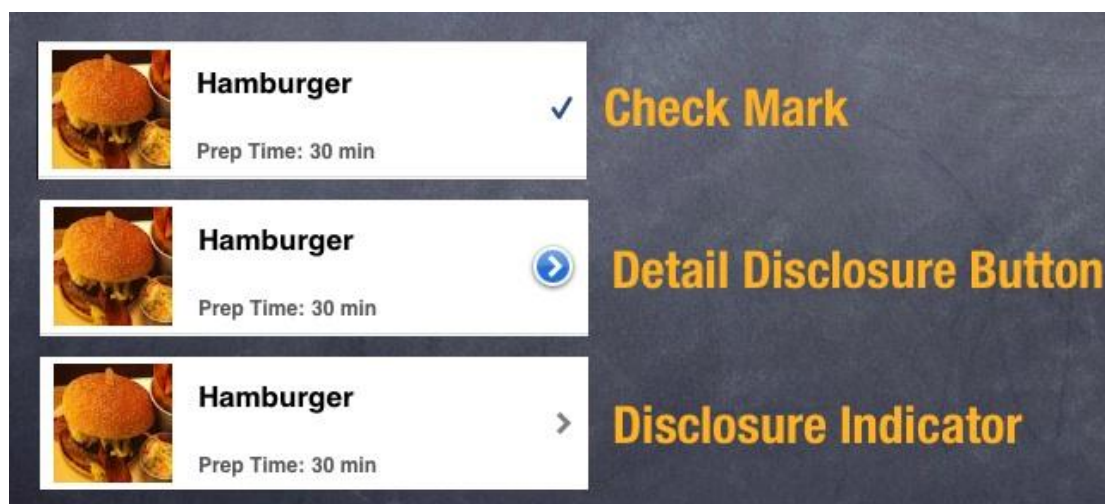


一个单元格分为 3 部分：

图像(缩略图) - 左边部分保留显示缩略图，就像我们之前在 Simple Table 应用程序中实现的一样；

内容 - 主要部分用来显示文本标签和相信内容；

附件视图 (Accessory View) - 右侧部分保留为附件视图。有 3 类默认的附件视图，包括 Disclosure 标示符、Detail Disclosure 标示符和 Check Mark (选中标志)，下图显示了这些标示符的效果：

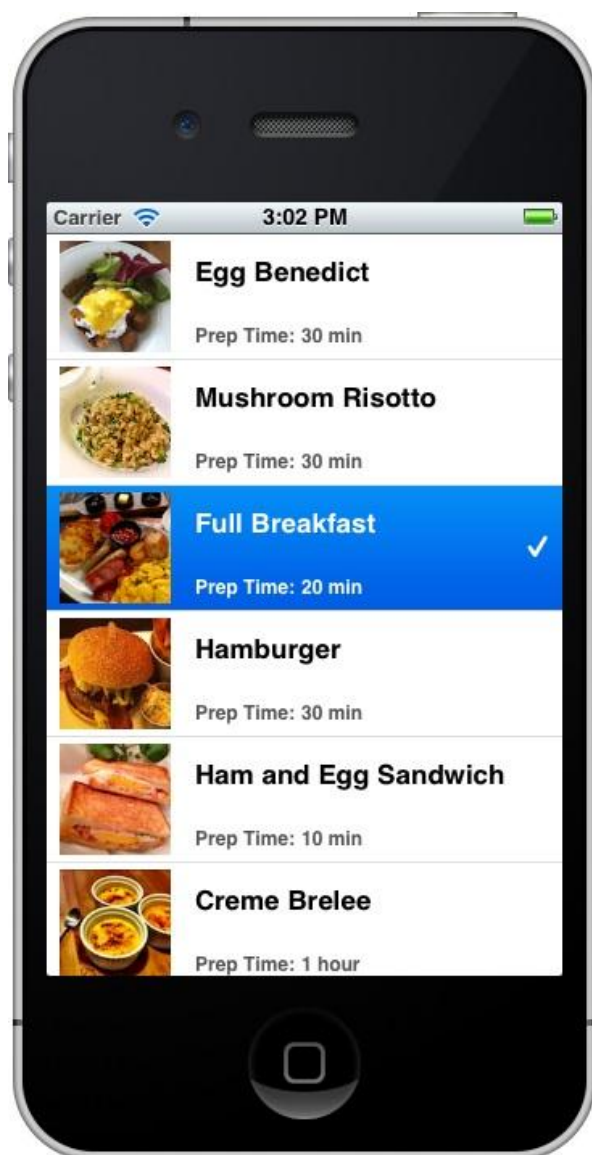


在行选择之后，为了显示选中标志，仅需要在[messageAlert show]代码之后，添加如下 2 行代码：

```
UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexPath];  
cell.accessoryType = UITableViewCellAccessoryCheckmark;
```

第一行通过使用 indexPath 获取选中表的单元格；第二行更新选中单元格的附件视图为选中标志（Check Mark）。

编译运行 App，在你轻拍一行后，将显示一个选择标志（Check Mark）。



中文化菜单名称，更新之后的效果如下图所示：



现在，当你选择一行时，将以蓝色突出显示该行。如果你不喜欢，可以添加如下

代码取消这一效果：

```
[tableView deselectRowAtIndexPath:indexPath animated:YES];
```



接下来是什么？

你觉得一系列的表视图 (Table View) 教程如何？我希望你学到了很多知识。现在，你应该掌握了如何创建表视图、定制表视图单元格和处理表视图行选择了。

确信你已经理解了本教程的内容,因为 UITableView 是 iOS 中常用的 UI 控件。

如有你有任何问题,欢迎留言。

在 iOS 开发中,仍然有大量的内容需要学习。在接下来的教程中,我们将演示如何替换菜谱数组为资源文件,和研究导航控制器 (Navigation Controller)。

同时,还包括使用 Storyboard 创建更复杂的应用程序。

本文由 [EntLib.com Team](http://EntLib.com) 翻译整理,如你在创建 App 中遇到问题,欢迎访问 EntLib.net 留言和提交你的问题。

英文原文连接: [How To Handle Row Selection in UITableView](#)

第六部分：应用 Property List 强化你的 Simple Table 应用程序

前面我们已经创建[一个简单的表视图应用程序](#)，显示菜谱列表。分析一下代码，你会发现所有的菜谱都硬编码在源代码中。之前，我们只考虑让事情变得简单，并着重演示如何创建一个 UITableView 应用程序。然而，将所有元素硬编码在代码中并不是推荐的方法。在真实的 App 开发中，我们常常将这些静态元素存放在外部（如菜谱列表）文件或数据库或其他地方。在 iOS 编程中，有一种类型的文件，称为 Property List。这一类型的文件通常在 Mac OS 和 iOS 中发现，用来存放简单的结构数据（如应用程序的设置）。在本教程中，我们将更新之前的 Simple Table 应用程序，使用 Property List。

简单而言，我们将讨论如下一些主题：

- 1) 转化静态数组中的表数据为 Property List；
- 2) 如何读取 Property List；

为什么外部化表数据？

将静态数据从代码中分离出来是一个很好的做法。为什么呢？将表数据置入外部数据源的优点是什么呢？假设让你添加 50 多个菜谱到 Simple Table 应用程序中，你需要返回源代码，将新的菜谱添加到初始化代码中。


```

1 // Initialize table data
2 tableData = [NSArray arrayWithObjects:@"Egg Benedict", @"Mushroom Risotto", @"Full Breakfast", nil];
3
4 // Initialize thumbnails
5 thumbnails = [NSArray arrayWithObjects:@"egg_benedict.jpg", @"mushroom_risotto.jpg", @"full_breakfast.jpg", nil];
6
7 // Initialize Preparation Time
8 prepTime = [NSArray arrayWithObjects:@"30 min", @"30 min", @"20 min", @"30 min", @"10 min", nil];

```

上面代码并没有任何错误，但是你看看代码，编辑这些代码并不简单，并且必须要严格遵守 Objective-C 语法。更改代码可能会引入其他错误，这并不是我们期望的。显然，最好是将数据和代码逻辑分离，像下面这一将表数据存放起来不是更好么？

Key	Type	Value
▼ RecipeName	Array	(16 items)
Item 0	String	Egg Benedict
Item 1	String	Mushroom Risotto
Item 2	String	Full Breakfast
Item 3	String	Hamburger
Item 4	String	Ham and Egg Sandwich
Item 5	String	Creme Brelee
Item 6	String	White Chocolate Donut
Item 7	String	Starbucks Coffee
Item 8	String	Vegetable Curry
Item 9	String	Instant Noodle with Egg
Item 10	String	Noodle with BBQ Pork
Item 11	String	Japanese Noodle with Pork
Item 12	String	Green Tea
Item 13	String	Thai Shrimp Cake
Item 14	String	Angry Birds Cake
Item 15	String	Ham and Cheese Panini
► Thumbnail	Array	(16 items)
► PrepTime	Array	(16 items)

Sample Property List

实际上，可能并不是你提供表数据（本例子是菜谱列表数据），而是其他并不懂 iOS 编程经验的人提供这些数据。当我们将数据放置在外部文件，这样更容易读写，且更容易理解。

随着你更深入学习，你将了解如何将数据存放在 Server 端（或者所谓的云）。

App 中的所有数据将根据需要从 Server 端读取。目前,这一提供了很大的好处,因为对数据的任何更新都需要你重新构建 App,并提交给 Apple 审批。通过将数据分离并存放在云(Cloud)中,你可以随时更新数据,而不必更新你的 App。今天,我们暂不讨论 Cloud。返回项目中,看看如何将所有的菜谱列表存放在 Property List 中。

Property List 是什么？

Property List 提供了一个方便的方法来存放简单的结构数据,通常为 XML 格式。如果你之前在 Mac 或 iPhone 编辑过配置文件,你可能看到过.plist 后缀的文件,它们就是 Property List 的应用例子。

你不能使用 Property List 存放所有类型的数据,Property List 中存放的数据类型是有限制的,包括数组 (Array)、字典 (Dictionary)、字符串 (String) 等等。关于支持类型的更详细信息,可参考 [Property List 官方文档](#)。

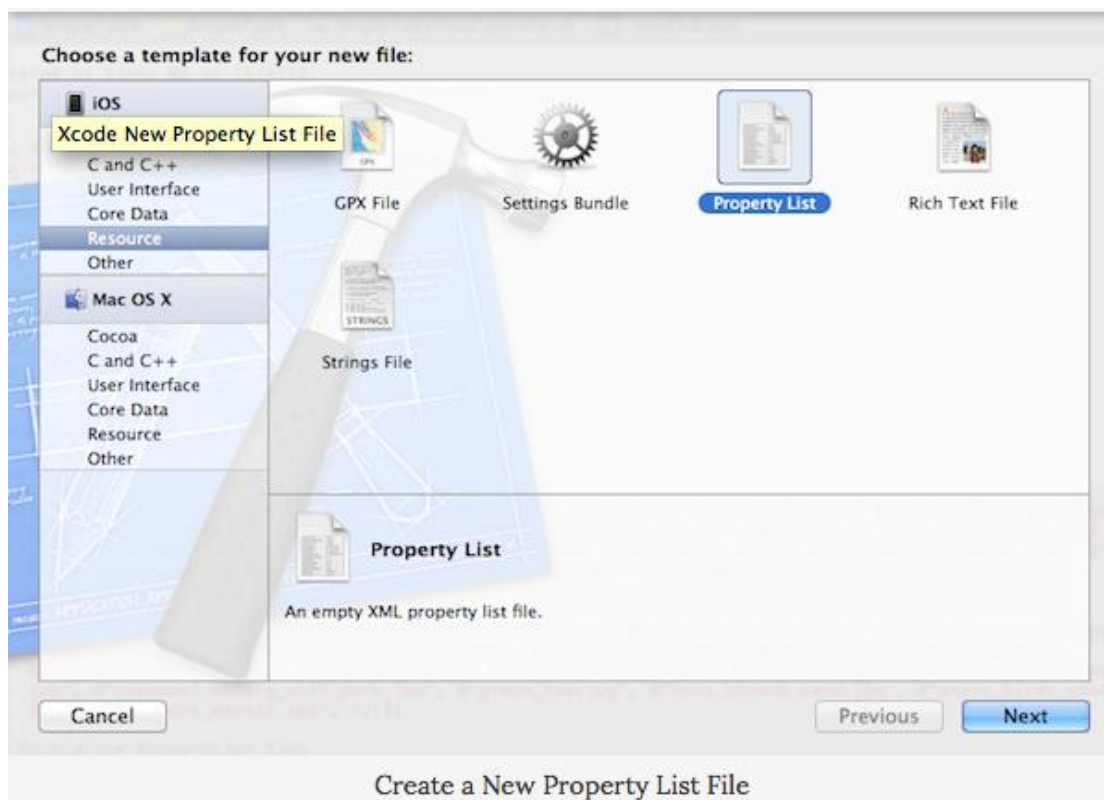
在 iOS App 开发中,Property List 经常用来存放应用程序的设置。这并不意味着你可以将它作为其他用途使用,它用来存放少量的数据。

这是存放表数据的最好方法吗？

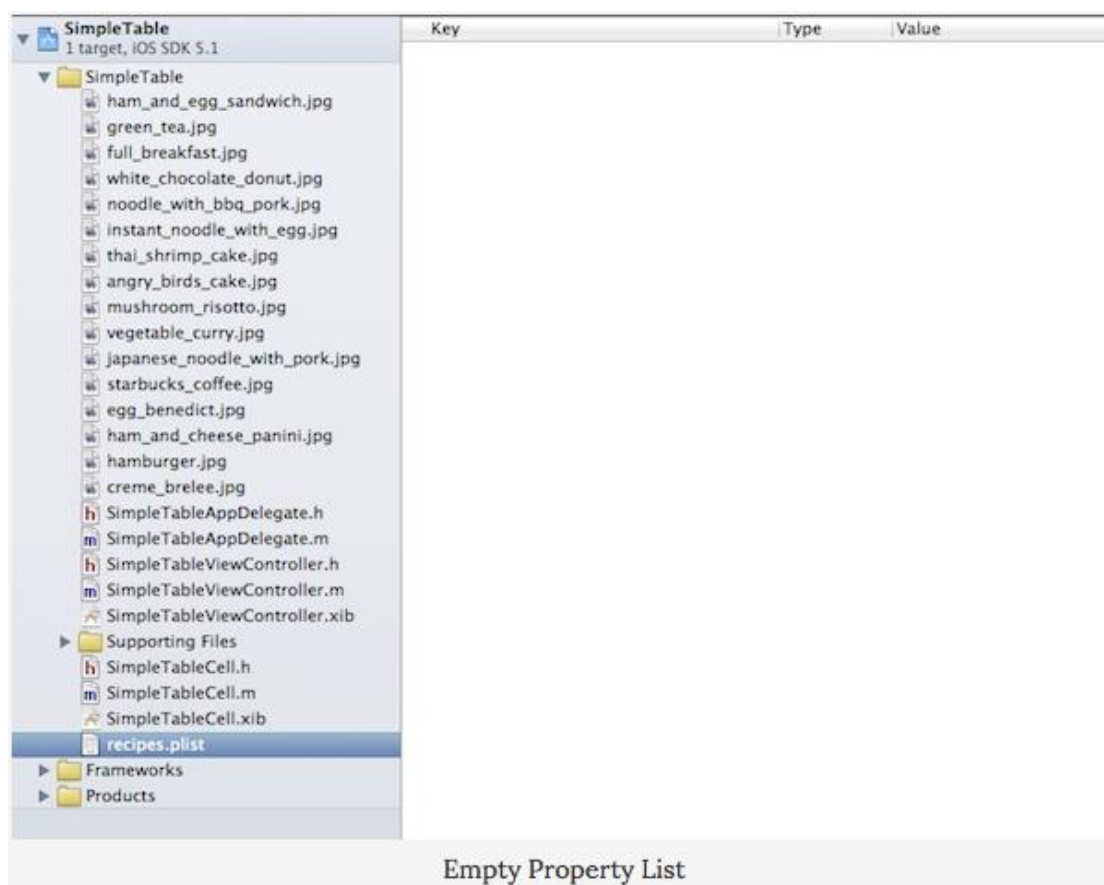
不是,绝对不是。我们使用 Property List 来演示如何在外部文件存放表数据,这仅仅只是一个例子。随着你学习更多经验,你将知道其它存放数据的方法。

转换表数据为 Property List

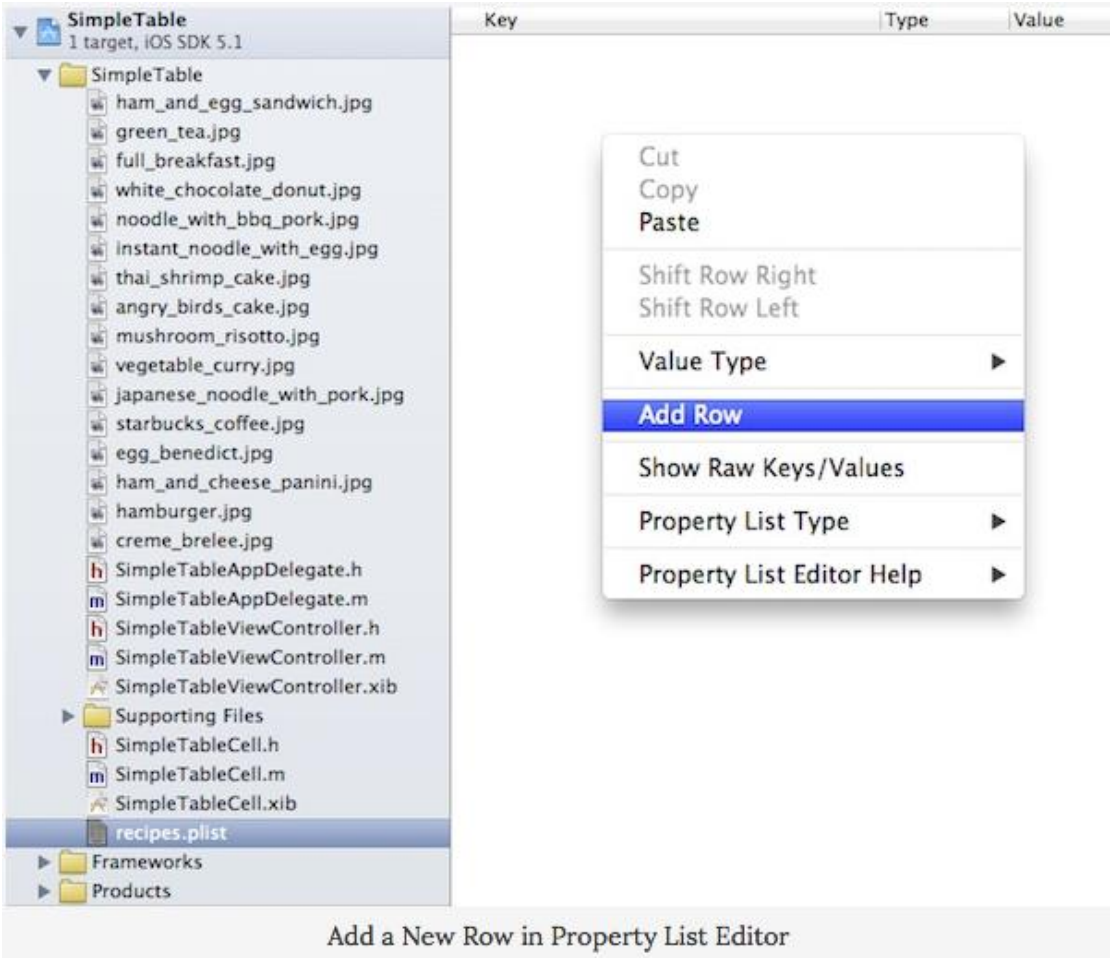
现在已经了解了 Property List。下面我们开始转换数据为 Property List。首先，在 Xcode 中打开 Simple Table 项目，右击 SimpleTable 文件夹，选择 New File...，接着选择 iOS 模板下面的 Resource，最后选择 Property List，并点击 Next 按钮继续。



在弹出窗口中，使用 recipes 作为文件名。确认之后，Xcode 将自动为你创建 Property List 文件。默认情况下，Property List 为空：



有 2 种方法来编辑 Property List。你可以右击编辑区域，并选择 Add Row 菜单添加一个新值。

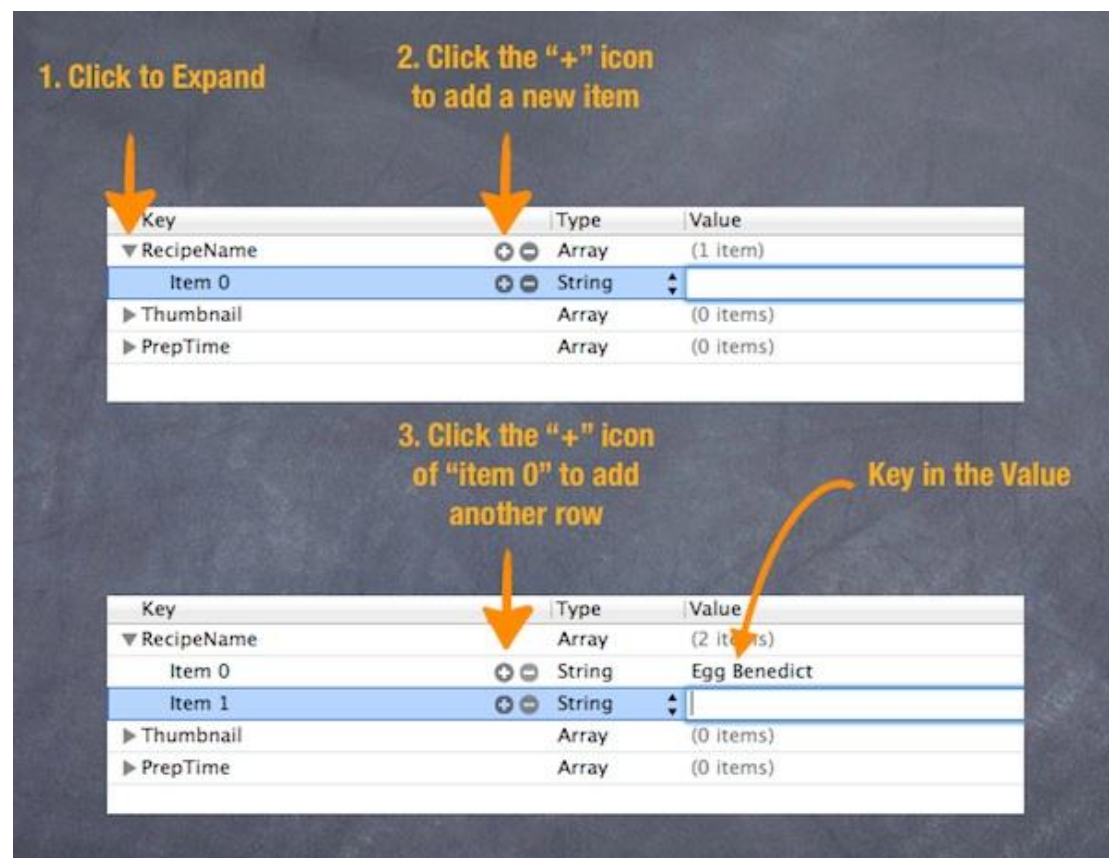


因为需要在 Property List 中添加 3 个数据数组 ,我们在 Property List 中添加 3 行 类型为 Array。分别命名 Key 值为 :RecipeName ,Thumbnail 和 PrepTime。Key 值作为识别码 , 在后面的代码中会用来获取相应的数组。

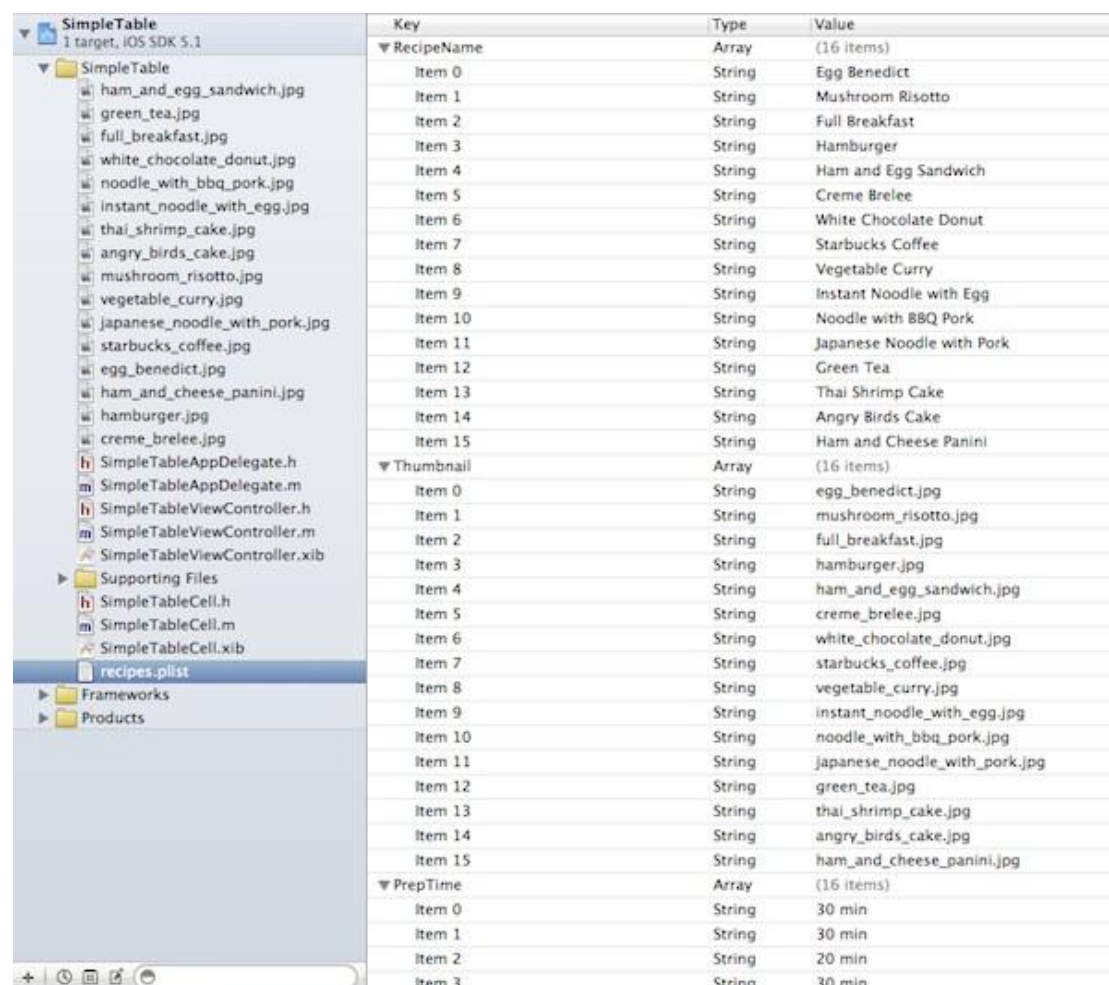
Key	Type	Value
▶ RecipeName	Array	(0 items)
▶ Thumbnail	Array	(0 items)
▶ PrepTime	Array	(0 items)

Define Three Arrays in Property List

点击展开图标 , 并点击 + 图标 , 添加新的 item , 实现向数组中添加数据。如果你不清楚 , 可以按照下图中的步骤 :



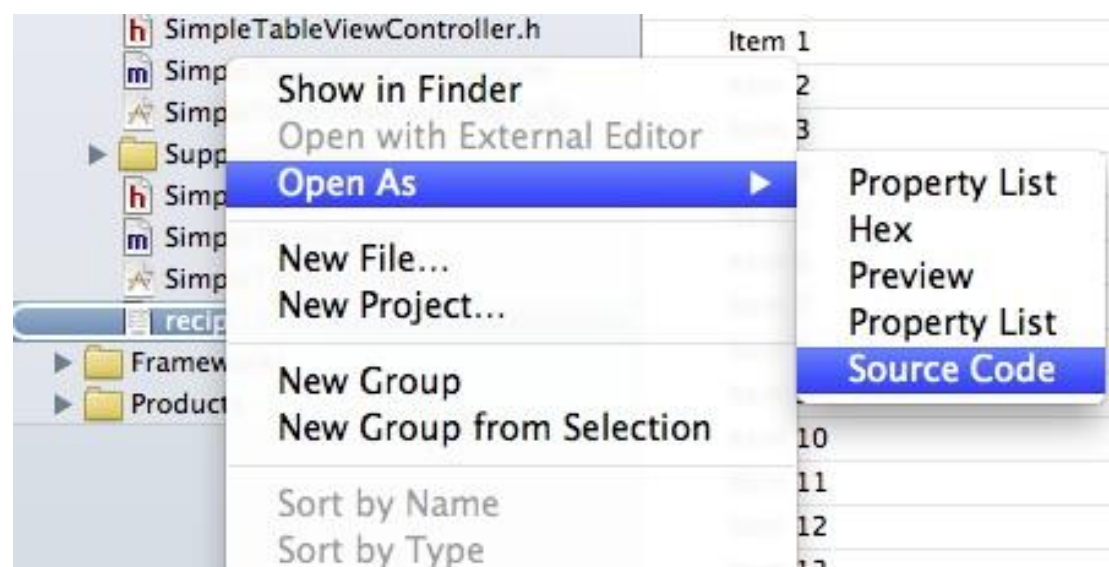
重复上面的操作步骤，直到你添加完成了所有数组元素。完成之后的 Property List 如下图所示：



Key	Type	Value
RecipeName	Array	(16 items)
Item 0	String	Egg Benedict
Item 1	String	Mushroom Risotto
Item 2	String	Full Breakfast
Item 3	String	Hamburger
Item 4	String	Ham and Egg Sandwich
Item 5	String	Creme Brelee
Item 6	String	White Chocolate Donut
Item 7	String	Starbucks Coffee
Item 8	String	Vegetable Curry
Item 9	String	Instant Noodle with Egg
Item 10	String	Noodle with BBQ Pork
Item 11	String	Japanese Noodle with Pork
Item 12	String	Green Tea
Item 13	String	Thai Shrimp Cake
Item 14	String	Angry Birds Cake
Item 15	String	Ham and Cheese Panini
Thumbnail	Array	(16 items)
Item 0	String	egg_benedict.jpg
Item 1	String	mushroom_risotto.jpg
Item 2	String	full_breakfast.jpg
Item 3	String	hamburger.jpg
Item 4	String	ham_and_egg_sandwich.jpg
Item 5	String	creme_brelee.jpg
Item 6	String	white_chocolate_donut.jpg
Item 7	String	starbucks_coffee.jpg
Item 8	String	vegetable_curry.jpg
Item 9	String	instant_noodle_with_egg.jpg
Item 10	String	noodle_with_bbq_pork.jpg
Item 11	String	japanese_noodle_with_pork.jpg
Item 12	String	green_tea.jpg
Item 13	String	thai_shrimp_cake.jpg
Item 14	String	angry_birds_cake.jpg
Item 15	String	ham_and_cheese_panini.jpg
PrepTime	Array	(16 items)
Item 0	String	30 min
Item 1	String	30 min
Item 2	String	20 min
Item 3	String	30 min

为了你操作方便,你也可以直接[下载 recipes.plist 文件](#),并添加到你的项目中。

前面提到过,Property List 通常以 XML 格式存放文件。可以右击并选择 Open as Source Code 来查看 Property List 的源代码。



recipes.plist 文件的源代码如下图所示：

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3  <plist version="1.0">
4  <dict>
5      <key>RecipeName</key>
6      <array>
7          <string>Egg Benedict</string>
8          <string>Mushroom Risotto</string>
9          <string>Full Breakfast</string>
10         <string>Hamburger</string>
11         <string>Ham and Egg Sandwich</string>
12         <string>Creme Brelee</string>
13         <string>White Chocolate Donut</string>
14         <string>Starbucks Coffee</string>
15         <string>Vegetable Curry</string>
16         <string>Instant Noodle with Egg</string>
17         <string>Noodle with BBQ Pork</string>
18         <string>Japanese Noodle with Pork</string>
19         <string>Green Tea</string>
20         <string>Thai Shrimp Cake</string>
21         <string>Angry Birds Cake</string>
22         <string>Ham and Cheese Panini</string>
23     </array>
24     <key>Thumbnail</key>
25     <array>
26         <string>egg_benedict.jpg</string>
27         <string>mushroom_risotto.jpg</string>
28         <string>full_breakfast.jpg</string>
29         <string>hamburger.jpg</string>
30         <string>ham_and_egg_sandwich.jpg</string>
31         <string>creme_brelee.jpg</string>
32         <string>white_chocolate_donut.jpg</string>
33         <string>starbucks_coffee.jpg</string>
34         <string>vegetable_curry.jpg</string>
35         <string>instant_noodle_with_egg.jpg</string>
36         <string>noodle_with_bbq_pork.jpg</string>
37         <string>japanese_noodle_with_pork.jpg</string>
38         <string>green_tea.jpg</string>
39         <string>thai_shrimp_cake.jpg</string>
40         <string>angry_birds_cake.jpg</string>
41         <string>ham_and_cheese_panini.jpg</string>
42     </array>
43     <key>PrepTime</key>
44     <array>
45         <string>30 min</string>
46         <string>30 min</string>
47         <string>20 min</string>
48         <string>30 min</string>
```

在 Objective-C 中加载 Property List

接着，我们更新代码，从上一步创建的 Property List 文件中加载菜谱列表，从 Property List 中读取内容相当简单。iOS SDK 已经有内置的方法来处理该文件的读写操作。

替换如下的代码：

```
1 // Initialize table data
2 tableData = [NSArray arrayWithObjects:@"Egg Benedict", @"Mushroom Risotto", @"Full Bread", nil];
3
4 // Initialize thumbnails
5 thumbnails = [NSArray arrayWithObjects:@"egg_benedict.jpg", @"mushroom_risotto.jpg", @"full_bread.jpg", nil];
6
7 // Initialize Preparation Time
8 prepTime = [NSArray arrayWithObjects:@"30 min", @"30 min", @"20 min", @"30 min", @"10 min", nil];
```

更新之后的代码：

```
// Find out the path of recipes.plist
```

```
NSString *path = [[NSBundle mainBundle]
```

```
pathForResource:@"recipes" ofType:@"plist"];
```

```
// Load the file content and read the data into arrays
```

```
NSDictionary *dict = [[NSDictionary alloc]
```

```
initWithContentsOfFile:path];
```

```
tableData = [dict objectForKey:@"RecipeName"];
```

```
thumbnails = [dict objectForKey:@"Thumbnail"];
```

```
prepTime = [dict objectForKey:@"PrepTime"];
```

代码修改的解释

第 2 行代码 – 在读取 recipes.plist 文件之前 ,你需要先获取资源文件的全路径。

第 5 行代码 – 你在 Property List 中定义了 3 个 Keys ,分别为 :RecipeName ,

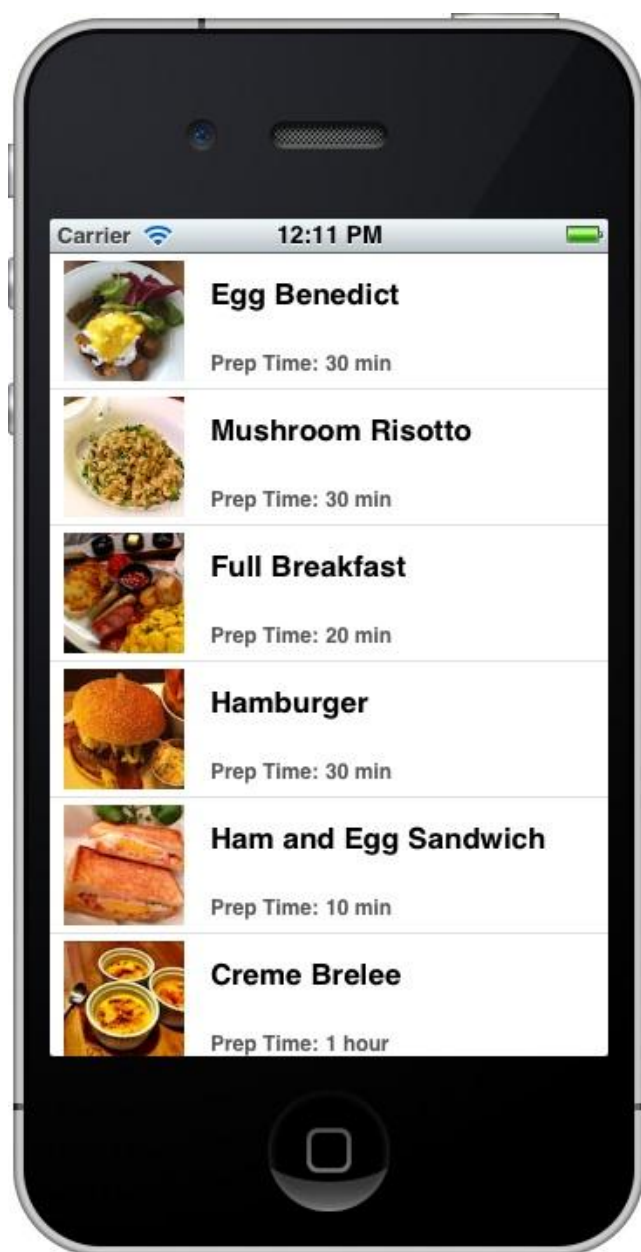
Thumbnail 和 PrepTime。在这个例子中 ,每一个 Key 和一个特定的数组关联 ,

提供数组值。在 iOS 编程中 ,我们使用 dictionary 术语指向 key-value 对组

合。NSDictionary 类提供了必要的方法来关联 dictionary。这里，我们使用 NSDictionary 类的 initWithContentsOfFile 方法来读取 Property List 文件中的 key-value 组合。

第 6-8 行代码 – 这些代码根据之前定义的 Key，负责分别检索相应的数组。

完成上述代码更新之后，再次运行 App。App 运行界面和之前的一样，但是，菜谱列表是从 Property List 中加载的。



接下来介绍什么？

我希望你通过这篇教程学到很多。现在，你应该对 Property List 理解了，并知道如何使用它来存放少量数据。接着，我们将学习 Storyboard 和 Navigation Controller。

本文由 [EntLib.com Team](#) 翻译整理，如你在创建 App 中遇到问题，欢迎访问 [EntLib.net](#) 留言和提交你的问题。

英文原文连接：[Enhance Your Simple Table App With Property List](#)

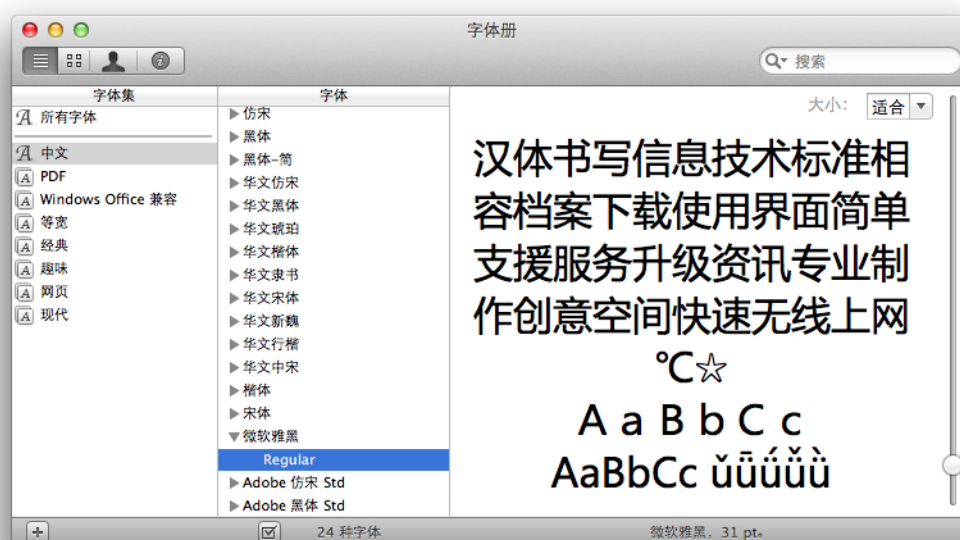
第七部分：如何在 Xcode 4.* 添加定制字体到 iOS 应用程序中？

在 iOS App 开发过程中,尤其是中文的 App,如果使用默认字体,会比较难看。

下面介绍如何使用中文字体。

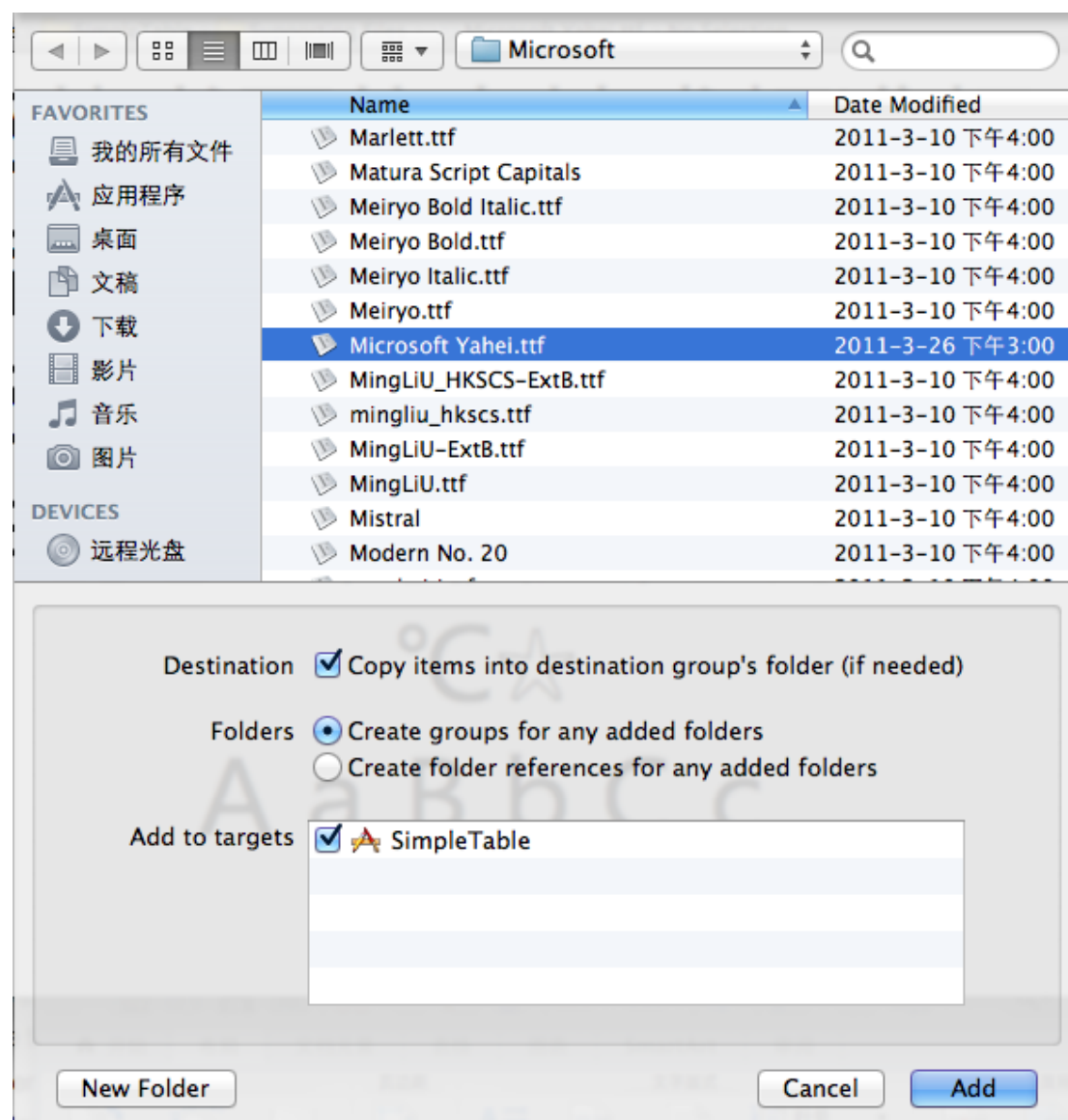
查找字体文件所在的位置

点击字体册,选中相应的中文字体文件位置,如下图所示:

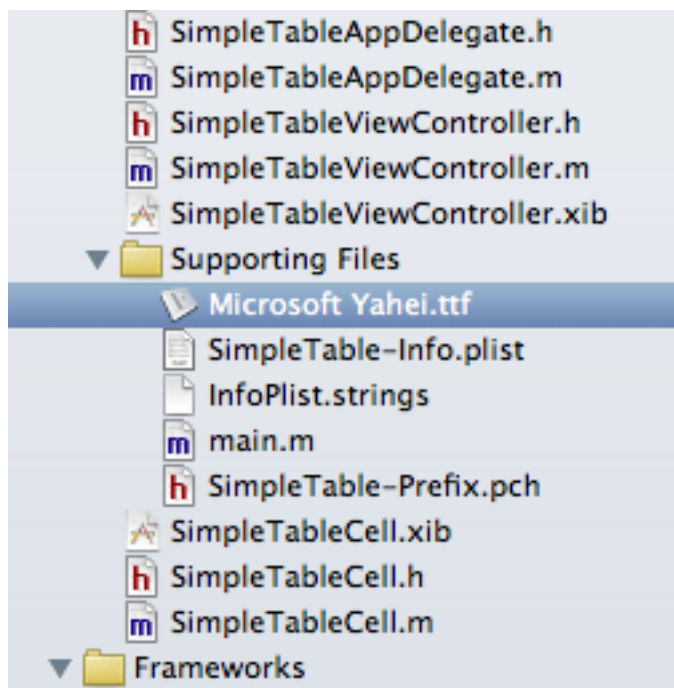


复制字体文件到应用程序包 (Application Bundle)

将特定的字体文件 (.ttf) 复制到应用程序包，如 Supporting Files 文件夹。注意：需要选中 Copy items into destination group's folder 选项。



上面选中微软雅黑字体文件 – Microsoft Yahei.ttf，点击 Add 按钮，添加到 App 项目中。下图是添加的字体文件，在 Supporting Files 目录下：



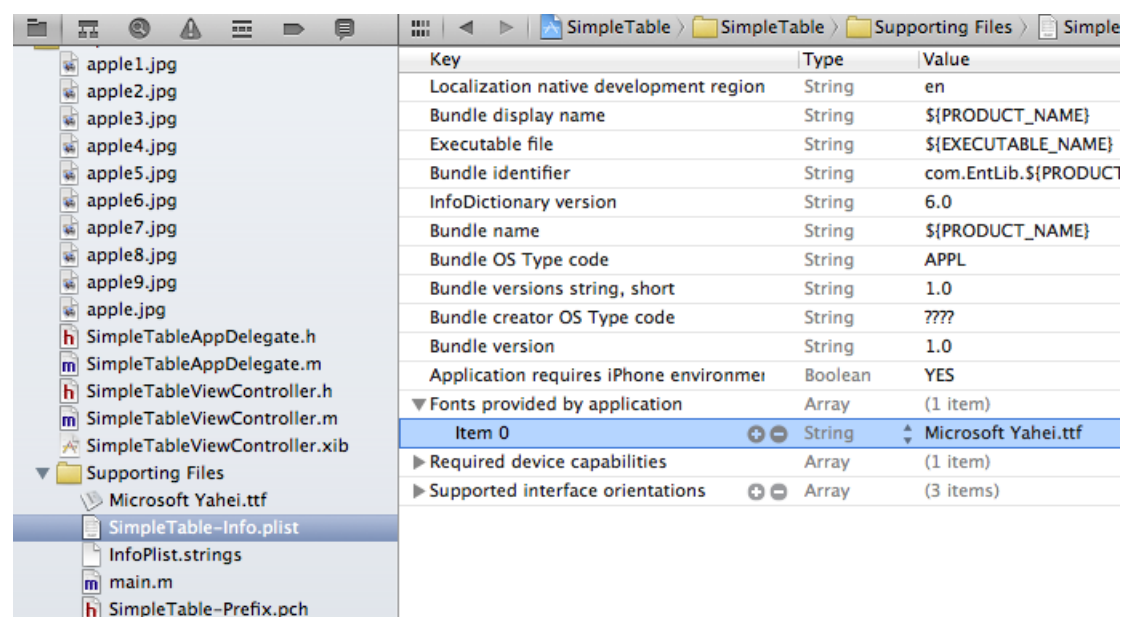
编辑 App 中的 info.plist 文件

接下来打开 App 中的 info.plist 文件，该文件应该在 Supporting Files 文件夹下。添加一个新的 Key，通过右键点击 或者 点击任何 Key 附近的+ 图标。

将 Key 命名为：Fonts provided by application (提醒：Xcode 会自动完成 Key 值的输入)。

同时，确认 type 设置为数组 (array)，这是一个 parent key。接下来，你需要添加不同的字体，作为该 key 的子元素。

通过点击你刚刚创建的 Key 左侧的三角符号，显示该数组包含的所有成员。下面，你就可以逐个添加字体文件了。需要注意的是：需要输入字体文件的名称，包括后缀名。如下图所示，输入 Microsoft Yahei.ttf：



准备使用新添加的字体

在下面的是例子中，我们已经定义了相应的标签，名称为：nameLabel 和 prepTimeLabel。

需要注意：在 Interface Builder 界面，你无法使用自己添加到 Xcode 中的字体，不得不在代码中设置和使用自定义字体。

在视图控制器(view controller)的 viewDidLoad 方法 ,我们就可以设置 label 使用这个字体了。示例代码如下所示：

```
self.nameLabel.font = [UIFont fontWithName:@"Microsoft Yahei"
size:20];

self.prepTimeLabel.font = [UIFont fontWithName:@"Microsoft Yahei"
size:14];
```

在上面代码中设置的字体名称时 ,只需要输入字体文件的名称 ,不必输入后缀名。

好啦！再次运行 App，看看界面效果如下所示：



本教程由 [EntLib.com Team](http://EntLib.com) 编译整理。

第八部分：如何在 iOS App 中添加启动画面？

你可以认为你需要为启动画面编写代码，然而 Apple 让你可以非常简单地在 Xcode 中完成。不需要编写代码，你仅需要在 Xcode 中进行一些配置。

什么是启动画面（ Splash Screen ）？

对于一些新的开发人员，可能没有听说过启动画面，让我简单解释一下。启动画面在 iOS Apps 中比较常见，也包括一些桌面应用程序。就是你启动 App 的时候看到的第一个界面。通常，启动画面是覆盖整个屏幕的一张图片，在主屏幕装载完成后隐藏。下图显示了一些启动画面的示例：



启动画面的主要目的是让用户知道你的 App 正在装载，并且对用户介绍你的品牌。启动画面对哪些需要较长时间启动的 Apps 尤其重要。一般而言，启动画面

是为了给用户更好的体验。

在你的 App 中添加启动画面

前面提到过，显示启动画面并不需要编写任何代码。iOS 提供了一个内置的功能 – 启动图像。在用户打开 App 时自动显示该图像，在 App 启动完成之后该图像自动消失。你可以在 Xcode 中进行简单设置启动图像，Xcode 就可以负责后续事情了。

准备你的启动画面

我们知道 iPhone 4/4S 支持更高分辨率（也就是所谓的视网膜显示器-Retina Display）。为了同时支持 2 中分辨率，我们需要准备 2 个版本的启动画面：

320 x 480 (for iPhone 2G / 3G / 3GS)

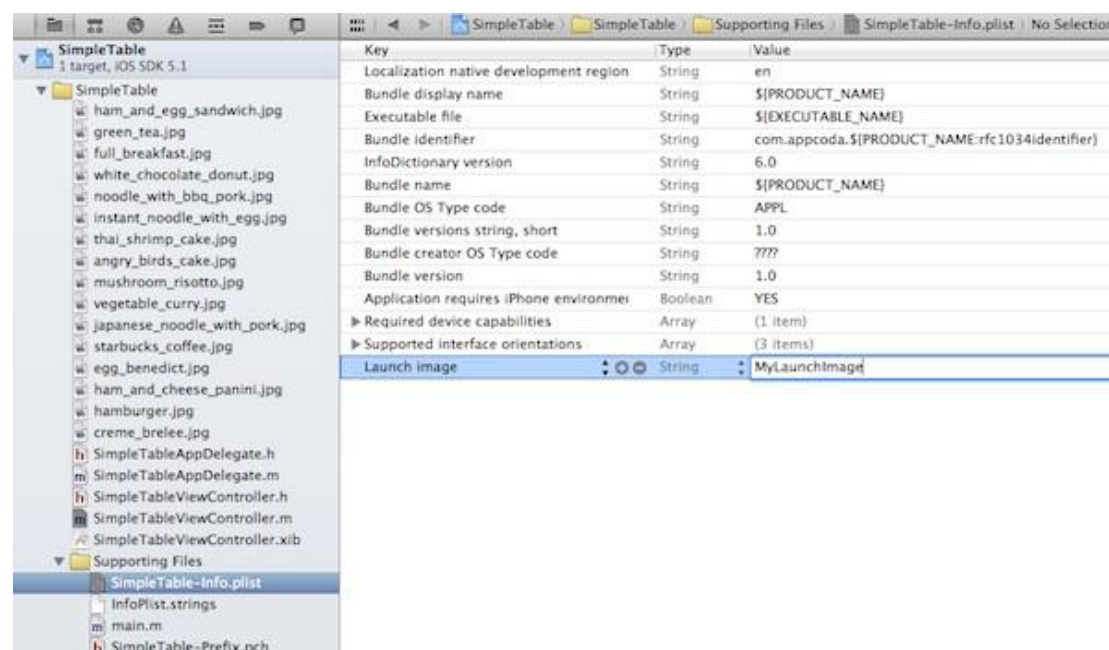
640 x 960 (for iPhone 4 / 4S)

为了简单一点，我这里演示如何为 iPhone App 添加启动画面，针对 iPad App 的操作，请参考 [Apple 的 iOS Human Interface Guideline](#) 文档 – 关于启动图像大小和命名规范。

启动图像必须为 PNG 格式。默认情况下，你可将低分辨率图像文件命名为 Default.png，对高分辨率的图像，用于 Retina 屏幕的（640*960 分辨率）图像文件命名为 [Default@2x.png](#)，@2x 是 iOS 中一种标准的分辨率修饰符。所

有用于在 Retina 屏幕显示的图像都应采用 @2x 字符串。

你也可以不是有 Default 作为启动图像文件名，使用其他你喜欢的文件名。启动图像文件定义在 App 的 Info.plist 文件中。以我们创建的 Simple Table 应用程序为例，在 SimpleTable-Info.plist 文件中，添加一个新的属性命名为 - Launch image，并指定你偏爱的文件名（如 MyLaunchImage）。



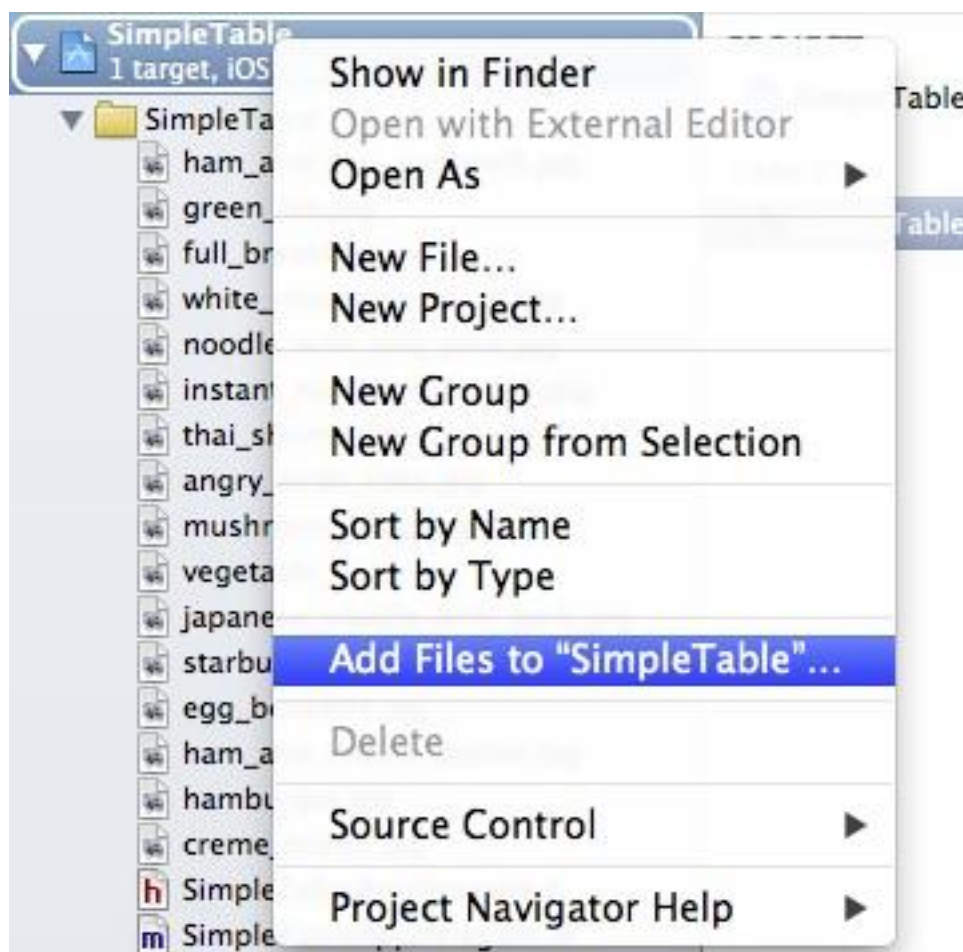
上述操作指示 iOS 使用 `MyLaunchImage.png` 和 `MyLaunchImage@2x.png` 文件作为启动画面。

你可以设计你自己的启动画面。基于测试的需要，你可以[到这里下载启动画面](#)。

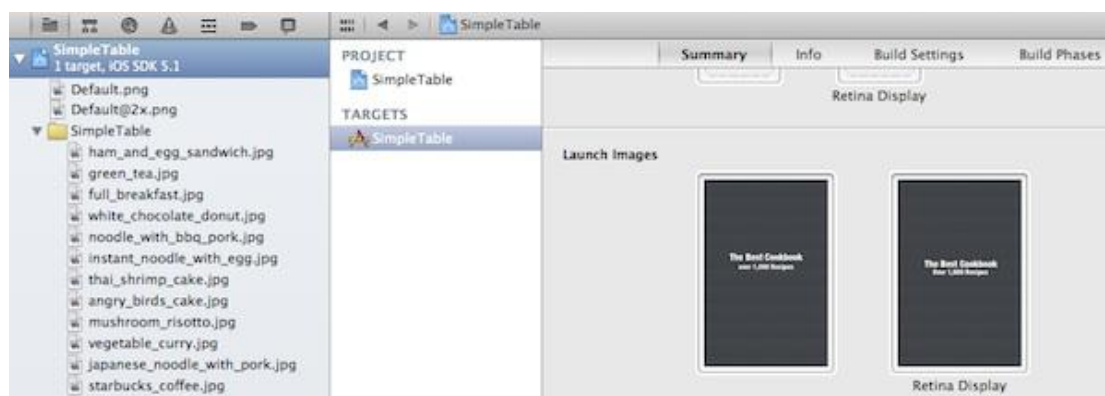


在 Xcode 中添加你的启动图像

在准备好启动图像之后，回到 Xcode，打开你的 Xcode 项目，我们继续使用 [Simple Table 项目](#)。右击 SimpleTable 项目，选择 Add Files to SimpleTable，同时添加 Default.png 和 [Default@2x.png](#) 到项目中。



在你完成上述操作之后，你会在项目中看到这两个文件，同时 Xcode 自动识别文件作为启动图像。



开始测试！

再次运行 App，这次在 App 运行的时候，你将会看到一个启动画面立即显示。

因为在 App 启动的时候没有太多东西加载,因此启动画面仅显示 1 秒,并消失。



更进一步信息

在本文简短教程中,我们演示了如何在 iPhone App 中添加一个简单的启动画面,其中我们使用的是纵向的图像。如何在一个横向启动的 App 中设计启动画面呢?如何对启动图像命名?我建议你查看 Apple 的官网文档 – [Apple' s programming guideline for App Launch Image](#) 获取更多信息。iOS 支持不同的启动图像以不同命名规范命名,阅读文档了解更多细节。

本文由 [EntLib.com Team](#) 翻译整理，如你在创建 App 中遇到问题，欢迎访问 [EntLib.net](#) 留言和提交你的问题。

英文原文连接：[How to Add Splash Screen in Your iOS App](#)