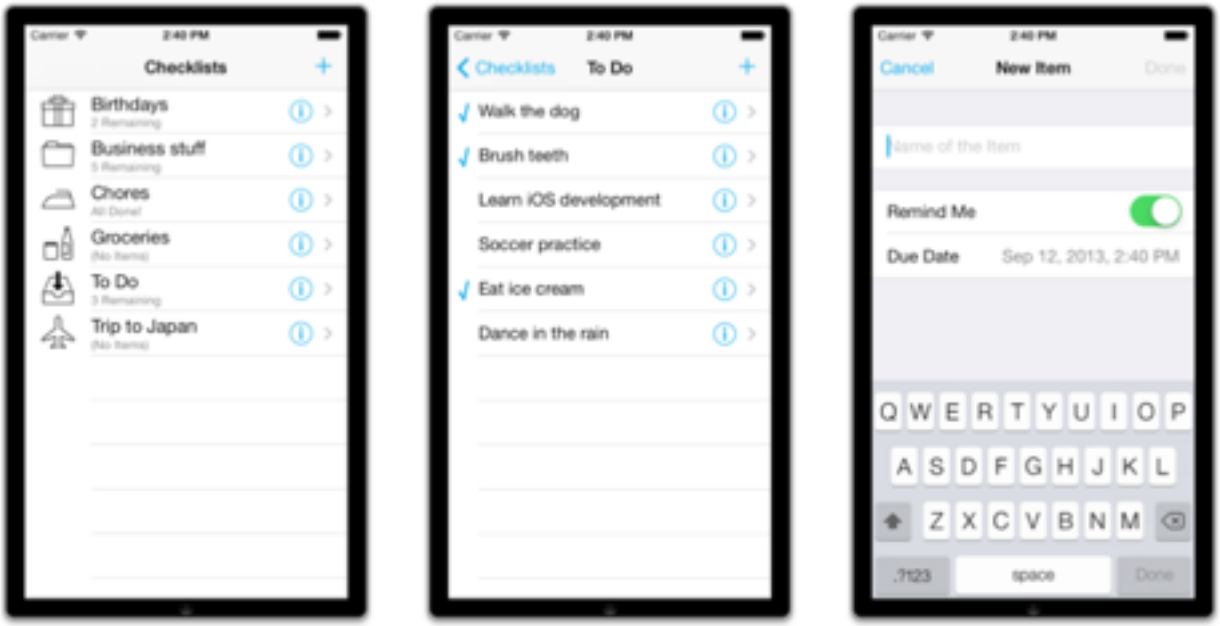
从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter1

原文及示例代码来自raywenderlich store中的iOS Apprentice 系列2教程，经过翻译和改编。

事务管理类应用（to-do list)是App Store中最火的项目类型之一，仅逊于打屁应用。对于iOS开发者来说，学习开发一款事务管理应用相当于通过iOS开发的成人礼。因此，我们这里也不能免俗。



最终的应用Checklists如下所示：

该应用可以让我们把自己的待办事务整理到一个列表中，然后当完成这些事项的时候会勾掉已完成事项。还可以对某个待办事项设定闹钟，这样在特定时刻iPhone会发出警告，提示用户要做这件事情，即便应用关闭的情况下都不受影响。
比如你要在1111光棍节疯狂shopping，那么最好把自己想要买的东西都放在待购清单里面，再设上闹钟，包你钱包里面的毛爷爷全部阵亡。

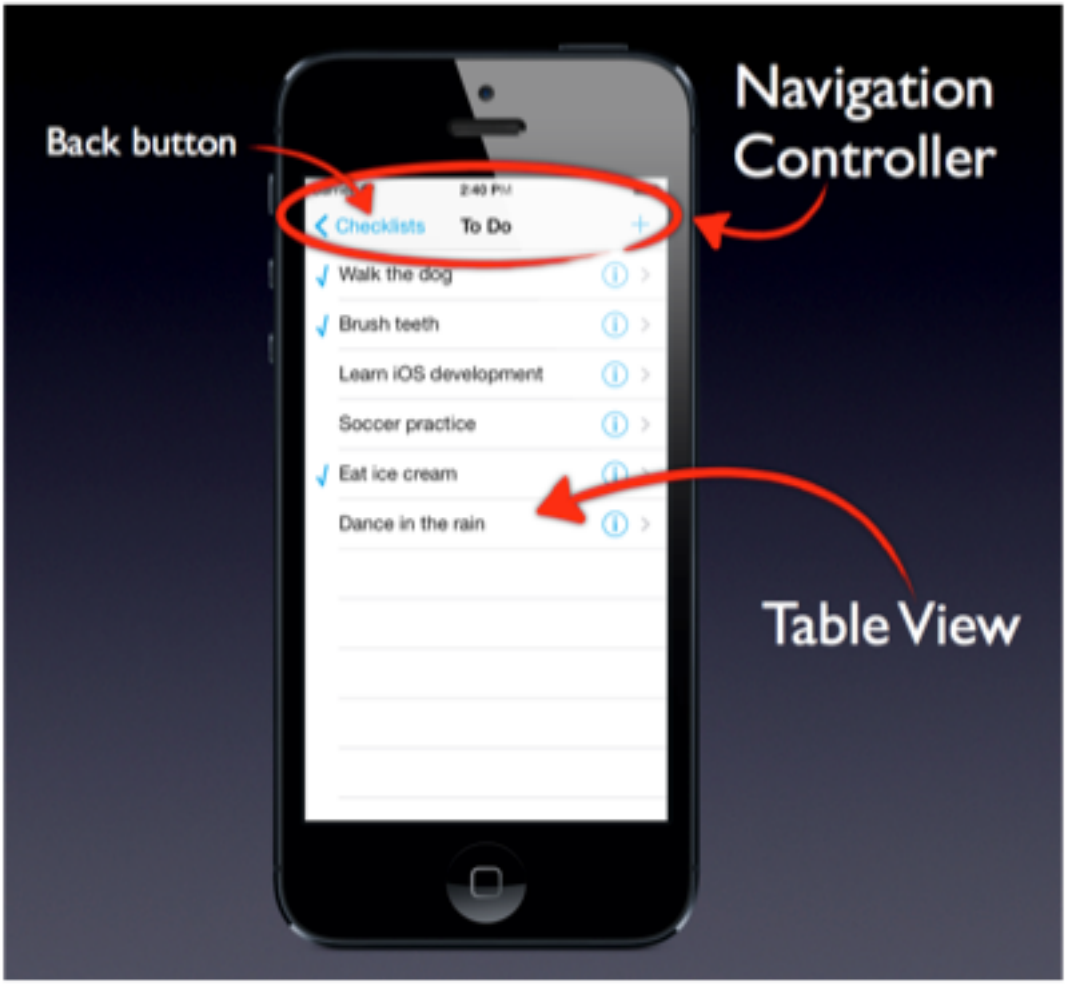
虽然Checklists这款应用的功能看似很简单，但麻雀虽小五脏俱全，里面包含了五个不同的界面，而每个界面的背后又有很多东西要去做。

Table views（表视图）和navigation controllers（导航控制器）

在这个系列的教程中，我们将学习iOS开发中最重要的两个UI界面元素-表视图和导航控制器。

表视图：虽然名为表视图，其实和我们平时接触的表不同。在iOS中表视图用来显示列表。在上面的三个图中，每个界面都用到了一个表视图（table view)。实际上，我们这款应用所有的界面都是由table view构成的。这个界面元素功能强大无比，也是iOS开发学习中需要掌握的最重要元素之一。

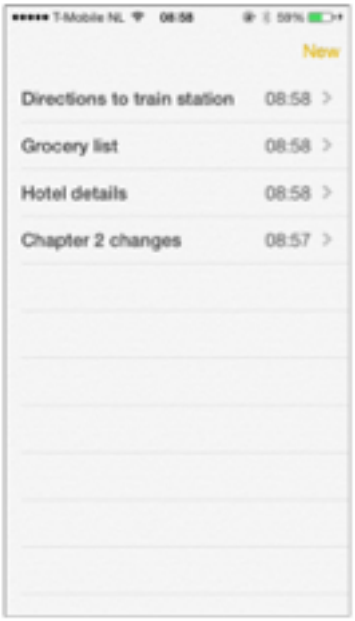
导航控制器(navigation controller)：使用导航控制器可以轻松构建一个界面层级体系，方便从一个界面跳转到另一个界面。导航控制器会在界面的顶部添加一个导航栏，里面包含一个标题和几个可选按钮。
在这个应用中，触碰checklist的名称，比如"To Do"，就会跳转到列出所有包含该列表对应的待办事

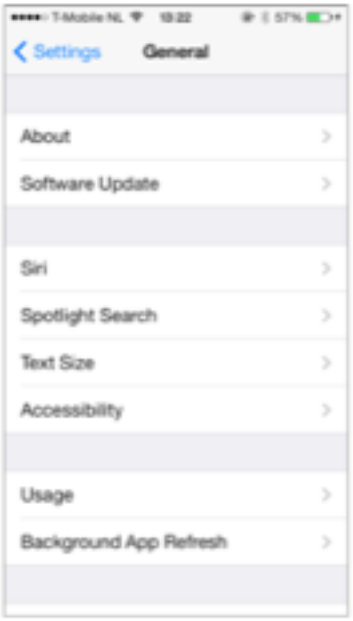


务的界面。触碰左上角的返回按钮可以跳转回上一个界面。这就是导航控制器的作用，毫无疑问我们都曾经在某个应用中碰到过。

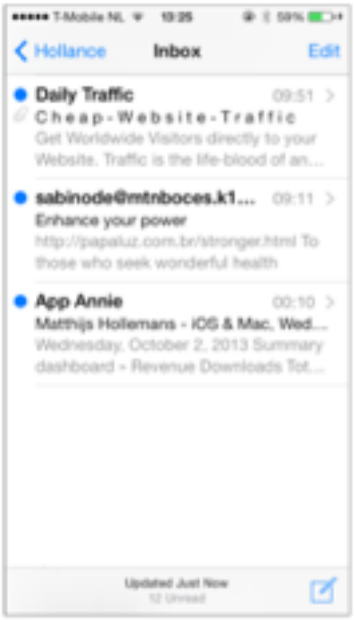
表视图和导航控制器是一对好基友，通常我们会一起使用它们：

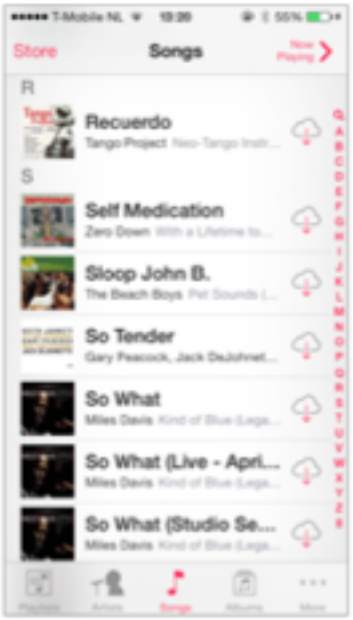
如果你不怕麻烦，可以看看自己iPhone上的官方应用和功能-Calendar,Notes,Contacts,Mail,Setting等等。你会发现尽管它们看上去有一些细微差别，所有这些应用的工作方式是类似的。原因就在于它们都用到了表视图和导航控制器。（Music应用中在底部还有一个tab bar，我们将在下一个系列的教程中来学习）。

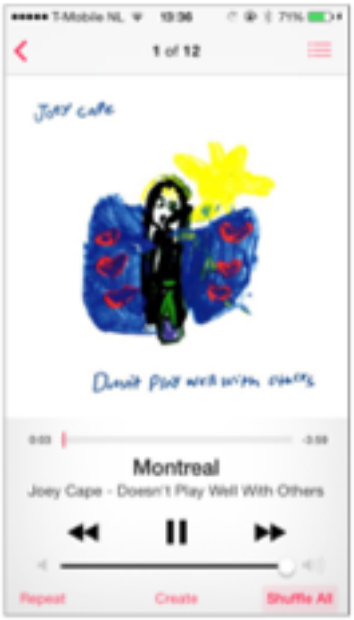

T-Mobile NL 08:58 59%
New
Directions to train station 08:58
Grocery list 08:58
Hotel details 08:58
Chapter 2 changes 08:57


T-Mobile NL 13:22 57%
Settings
General
About
Software Update
Siri
Spotlight Search
Text Size
Accessibility
Usage
Background App Refresh


T-Mobile NL 13:22 57%
oktober
M D W D V Z Z
30 1 2 3 4 5 6
vrijdag 4 oktober 2013
13:00
14:00
15:00
16:00
17:00
18:00
19:00
Today
Calendars
Inbox


T-Mobile NL 13:25 59%
Hollance
Inbox
Edit
Daily Traffic 09:51
Cheap-Website-Traffic
Get Worldwide Visitors directly to your Website. Traffic is the life-blood of an...
sabinode@mtnboces.k1... 09:11
Enhance your power
http://papaluz.com.br/stronger.html To those who seek wonderful health
App Annie 00:10
Matthijs Hollemans - iOS & Mac, Wed...
Wednesday, October 2, 2013 Summary dashboard - Revenue Downloads Tot...
Updated Just Now
12 Unread


T-Mobile NL 13:20 55%
Store
Songs
Now Playing
R
Recuerdo
Tango Project
S
Self Medication
Zero Down
Sloop John B.
The Beach Boys
So Tender
Gary Peacock, Jack DeJohnet...
So What
Miles Davis
So What (Live - Apri...
Miles Davis
So What (Studio Se...
Miles Davis


T-Mobile NL 13:36 71%
1 of 12
Montreal
Joey Cape - Doesn't Play Well With Others
Repeat
Create
Shuffle All

如果你想成为真正被人认可的iOS开发者，那么就必须掌握这两个基本的界面元素。此外，我们还将学习如何在不同的界面间传递数据，这个问题对大多数入门者来说都是个难题。

当我们学完这个系列的教程后，将会对view controller（视图控制器），table view（表视图）和delegate(代理）的概念轻车熟路，即便在梦中你也能开发出一款应用（好吧，哥知道你的梦中更多是土豪金或者苍老师）。

本系列教程会很长，里面包含了大量的源代码，最好可以专门抽一点时间来好好学一下。在学习的过程中，建议大家多改动代码，即便让应用崩溃了也无所谓。学习编程的最好方法就是多写多用多改代码。

在继续学习之前，我们还是稍微放松一下吧。

话说1111光棍节竟然是苍老师的生日，实在是。。。

All the screens of the Checklists app

顺便说一下，这个应用必然也是要用到table view的。

关于Checklists的设计

好了，现在让我们大概看下Checkists这款应用的工作原理：
其中首界面（1)显示了所有的事务清单。我们可以创建多个清单来管理自己的待办事项。一个事务清单包含了一个名称，一个图标，以及0或多个事项。我们可以通过界面（2）和（3）的Add/Edit Checklist来编辑事务清单的名称和图标。

我们可以触碰某个事务清单的名称来查看其中的待办事项-界面（4）。每个事项都有一个详细描述，以及用来标记该事项是否已经完成的一个标志，以及可选的截止日期。我们可以在界面（5）的Add/Edit Item界面中编辑事项。

这款应用还将使用本地推送来自动通知用户那些设置了“提醒我”选项的事项，即便该应用没有打开也不会受到影响-界面（6）。怎么样，这一点挺酷吧！

轻松玩转Table View（表视图）

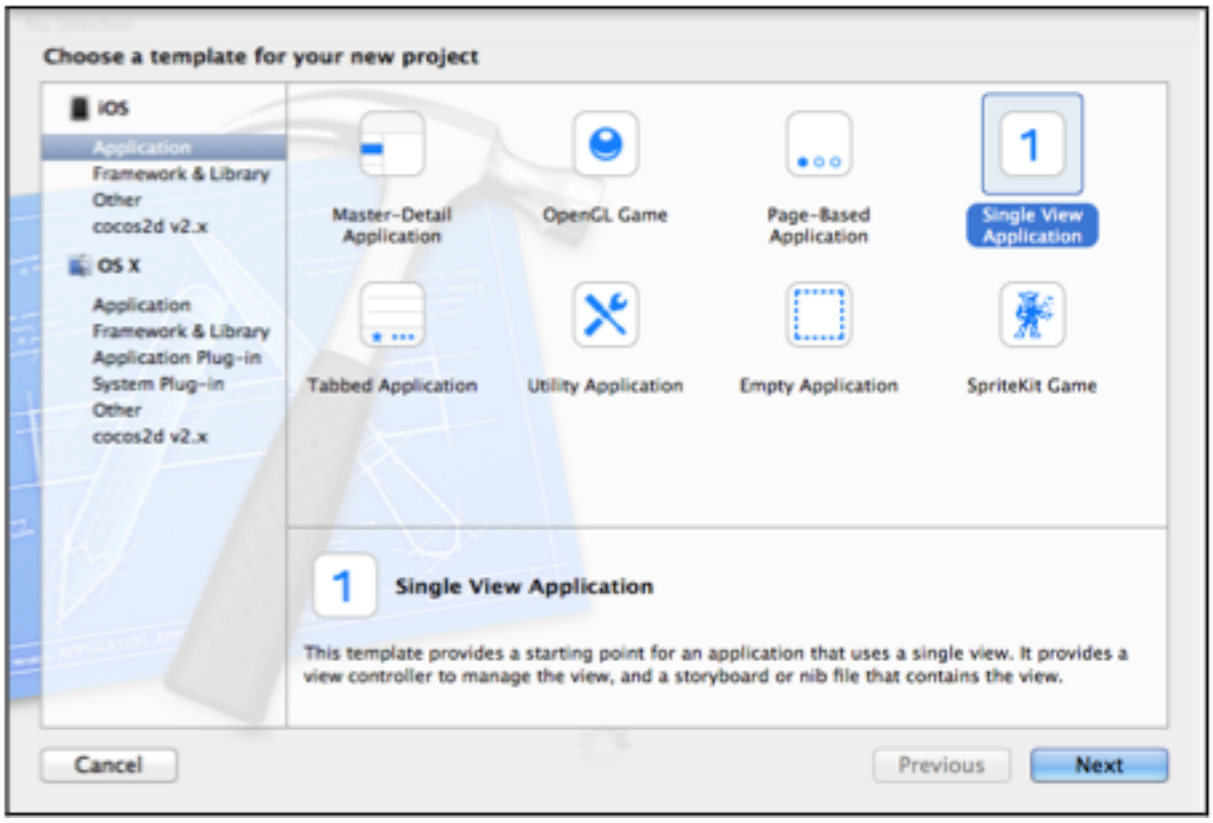
既然表视图对于iOS应用如此重要，显然小伙伴们已经迫不及待想了解怎么来用它了。和之前一样，我习惯于把工作分解成小的简单易操作的步骤，所以我们将会按照下面的步骤来逐一实现：

1.在应用的界面上放置一个表视图
2.在表视图里面填充数据
3.允许用户触碰某一行来打开或关闭选中标志。

当我们可以灵活自如的玩转以上三个步骤后，就可以自行添加一些新的功能，直到实现一个真正NB的应用。

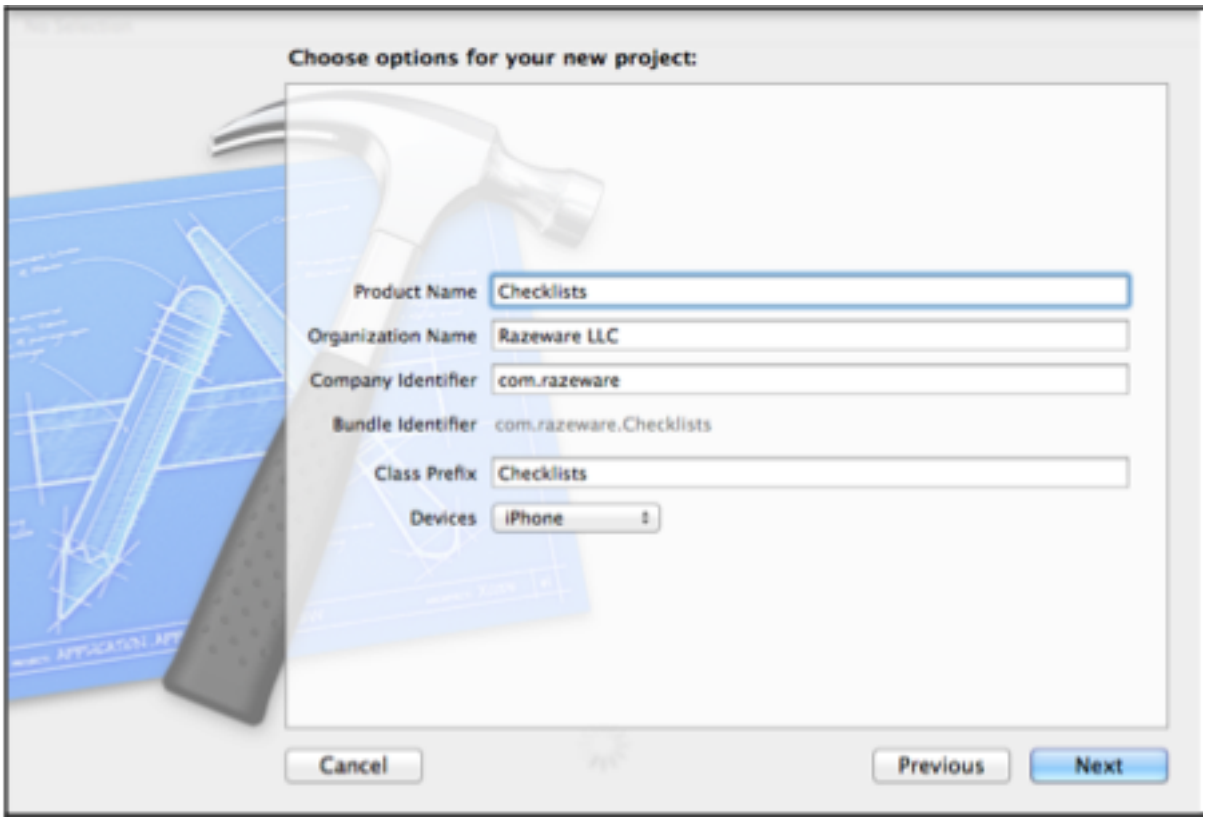
好了，废话不多说，让我们开干。

打开Xcode，创建一个新项目，



此时Xcode会让你填充一些信息：
Product Name: 这里填 Checklists
Oraganization Name:这里填你或者公司的名称，当然，一般我们都用英文的了
Company Identifier: 使用一个独特的标示，比如通常会用反转域名com.xxx
Class Prefix:这里填 Checklists
Devices:选择iPhone

然后点击next，选择项目的保存地址，就Ok了！

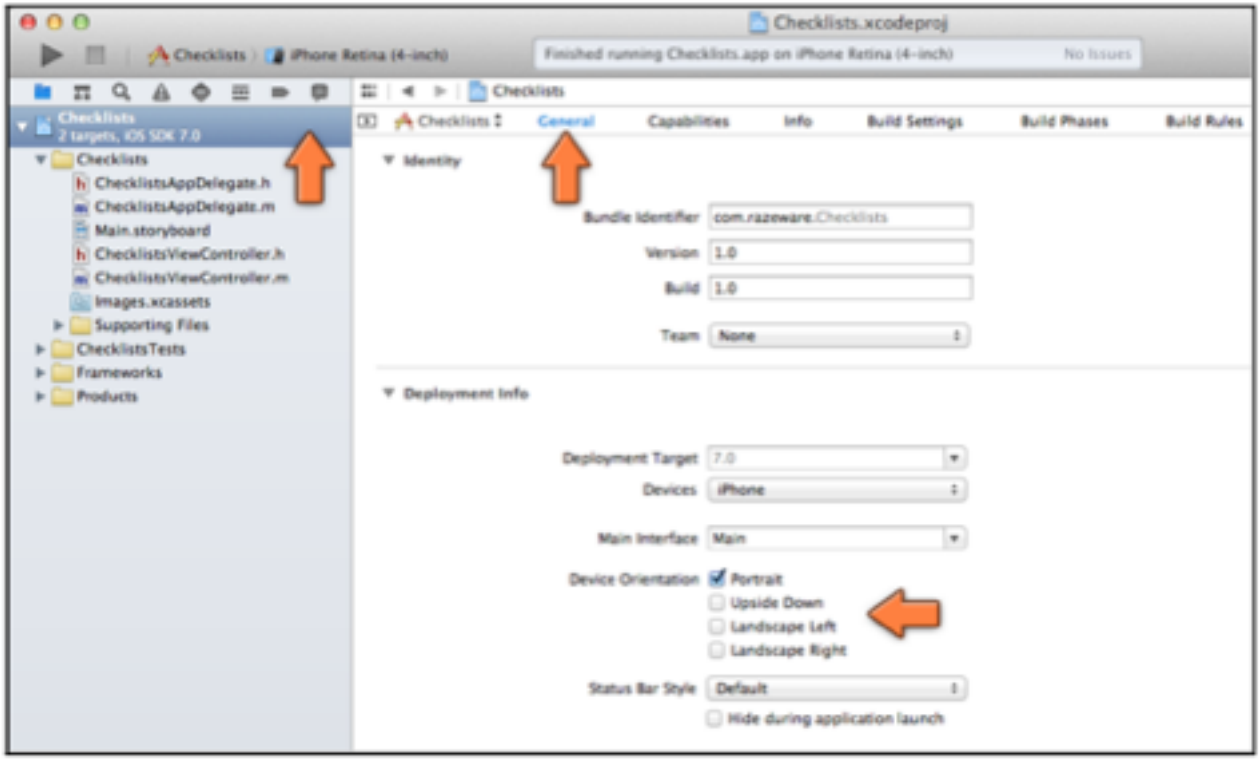


为了确保没有出错，建议按Command +R，或者点Xcode左上的运行按钮来试运行一下看看。什么也没有，只有蛋疼到无语的空白界面一张,说实话比之前版本的灰色背景还要苍白无力。把它插入到文档里面，你能看到的就只有顶部的状态栏。。。

无力吐槽，还是继续学习吧。

Checklists这款应用最终只会支持竖屏模式，但我们刚创建的项目还会支持横屏模式，这一点可以改一改。

点击项目导航器的Checklists项目名称，然后选中右侧的General选项卡。在Deployment Info的Device Orientation部分取消勾选Landscape Left和Landscape Right，只保留Portrait处于选中状态。



这样一来，无论如何旋转设备，应用始终停留在竖屏模式。

补充一下Upside Down选项

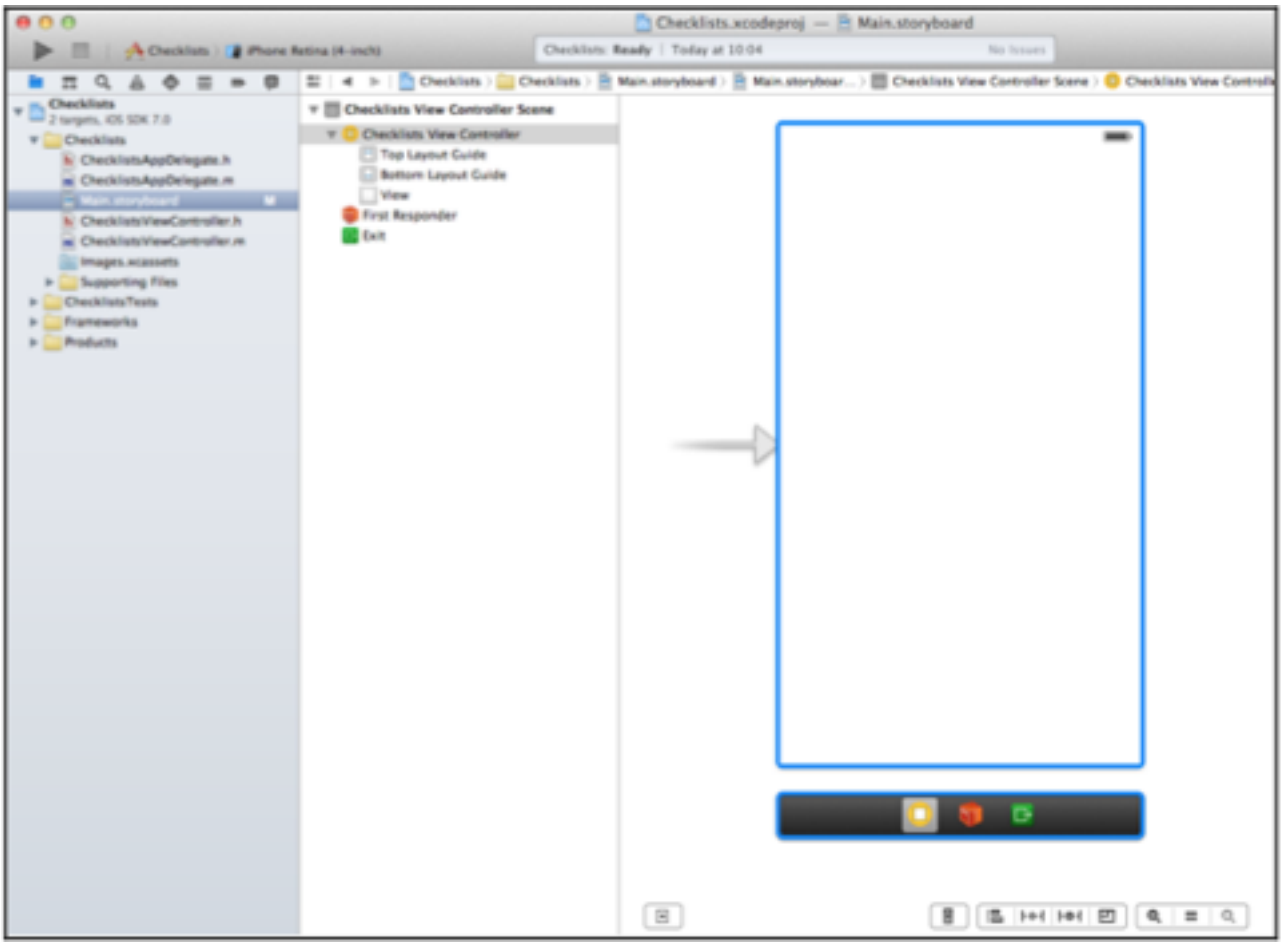
刚才的设备朝向那里还有一个Upside Down选项，但通常我们不会选它。因为如果选中它的话，玩家就可能旋转自己的iPhone手机，让Home键处于屏幕的顶部。这样当用户接到电话的时候就会不知所措。当然，与之相反的是，苹果建议iPad应用支持四种不同的朝向。

下一站：Storyboards故事板

对于这款应用，我们将使用storyboarding，也就是iOS5引进的所谓"故事板"技术。在iOS5之前，当我们设计应用的时候，必须为每个界面单独提供一个xib文件，但使用故事板之后，可以把所有的视图控制器界面设计整合到一个单独的storyboard文件中。我们在Bull's Eye中已经体会到Storyboard技术的妙处了，而在这个应用中它的作用会体现的更为明显。

现在让我们来看看Xcode左侧的项目导航器，在创建新项目的时候Xcode自动为我们生成了ChecklistViewController.h和.m文件，其中包含了主界面的视图控制器源代码。还记得吗？每个视图控制器都代表应用中的一个界面。

点击Main.storyboard可以在Xcode中打开内置的Interface Builder界面编辑器。



在storyboard的术语体系中，每个视图控制器都被命名为一个scene（场景）。

接下来选中Checklists View Controller，然后按Delete键将其从storyboard中删除，此时左侧的outline面板中会显示“No Scenes”。

注意：
outline 面板中会显示storyboard中所有场景的视图层级。如果你看不到这个面板，可以点击Interface Builder窗口底部的箭头来打开它。

之所以我们要删除这个场景，是因为我们不想用通用的视图控制器，而想用传说中NB无比的table view controller（表视图控制器）。使用这种特殊类型的视图控制器，可以让我们更轻松自如的处理表视图。

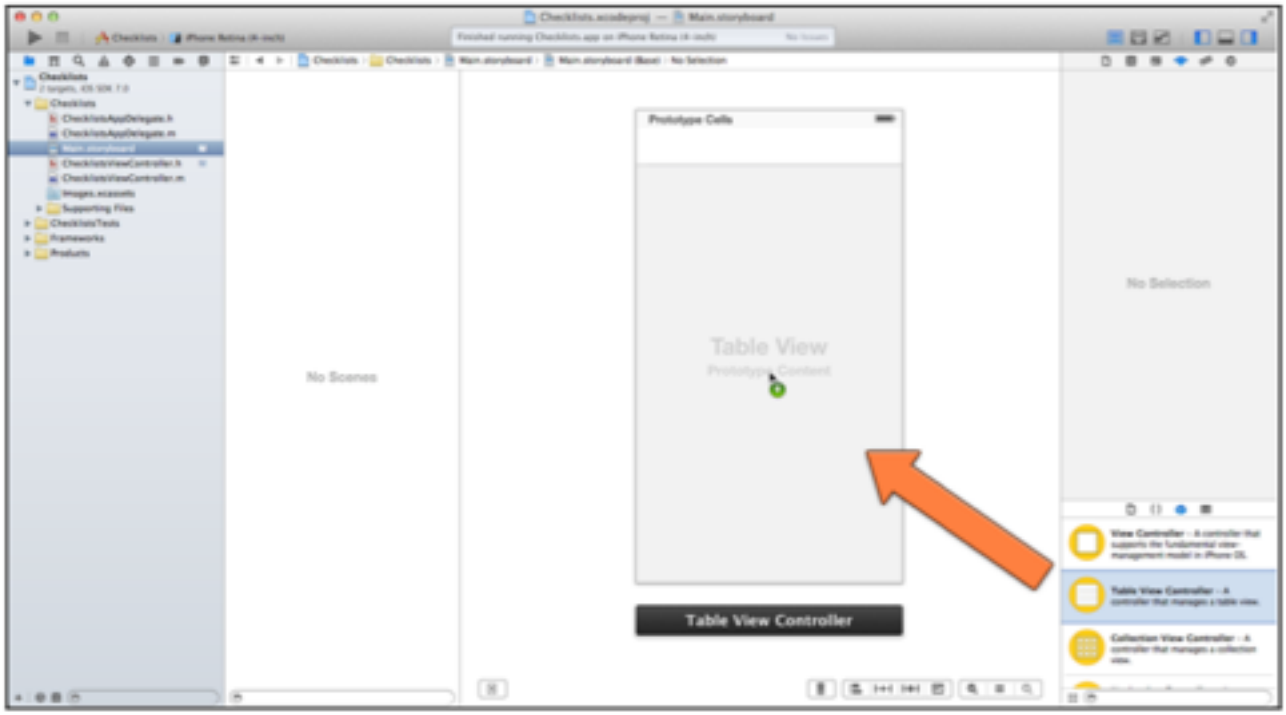
当然，要想把ChecklistViewController这个视图控制器的类型修改为表视图控制器，还需要编辑.h文件。

在Xcode中切换到ChecklistViewController.h，然后修改其中的代码，将@interface这一行代码
从：
@interface ChecklistsViewController : UIViewController

修改为：
@interface ChecklistsViewController : UITableViewController

通过这里的代码修改，Objective-c编译器就会知道该视图控制器现在是一个UITableViewController，而不是一个一般的UIViewController。

现在返回storyboard，然后从Object Library(右下角）中拖曳出一个Table View Controller到画布上。



就这么简单，我们已经在storyboard里面添加了一个表视图控制器场景。

接下来切换到Identity inspector（Xcode右侧的inspector面板的第三个选项卡），在Custom Class下面将Class类型手动更改为ChecklistsViewController。

此时场景列表中的场景名称就会自动更改为"Checklists View Controller Scene"。
需要注意的是，当你在进行这一步操作的时候，要确保自己选中的是Table View Congtroller，而不是Table View。

就这样，我们已经把ChecklistsViewController从一个普通的视图控制器更改为一个表视图控制器。

顾名思义，一个表视图控制器里面包含了一个Table View对象。很快我们会复习下视图控制器和视图之间的差异，不过现在我们只需要记住视图控制器代表着整个的界面，而表视图只是其中的一个视觉元素。

选择模拟器为iPhone Retina(4-inch)或者iPhone Retina(4-inch 64-bit），然后试运行一下。

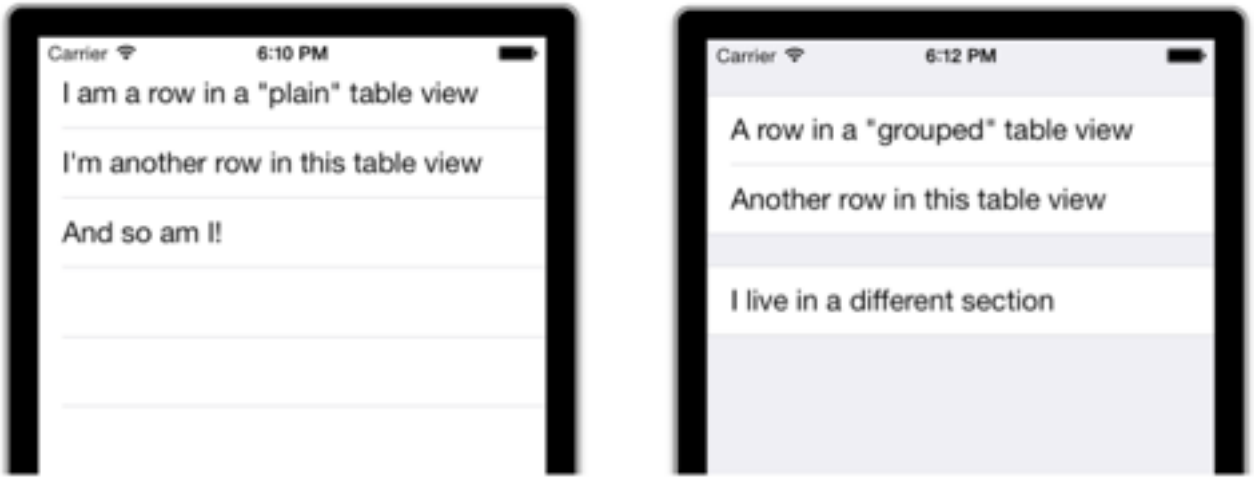
好吧，现在终于不是一片惨白了。

现在界面中显示的是一个空白列表，这就是传说中神秘而又强大的表视图（table view)。我们可以随意拖曳这个列表，但里面还没用什么数据。

table view表视图的详解

在继续操作之前，还是让我们更进一步的了解一下表视图。UITableView对象用来显示一个列表。和我们经常提到的表不同，table view并不会有多行和多列，因为UITableView实际上只有行的概念，而没有列的概念。与其说它是一个表格，不如说它是一个列表。
不知道苹果的工程师当年为何这样来命名table view，但既然已经这样了，还能怎样呢?
接受这个事实吧。

在iOS中，存在两种类型的表：plain和grounped。它们的使用方法差不多，不过有些小小的差别。从视觉呈现的角度来看，grouped表的各行会被整合到一起，然后会有浅灰色的背景。
我们还是看图说话吧。



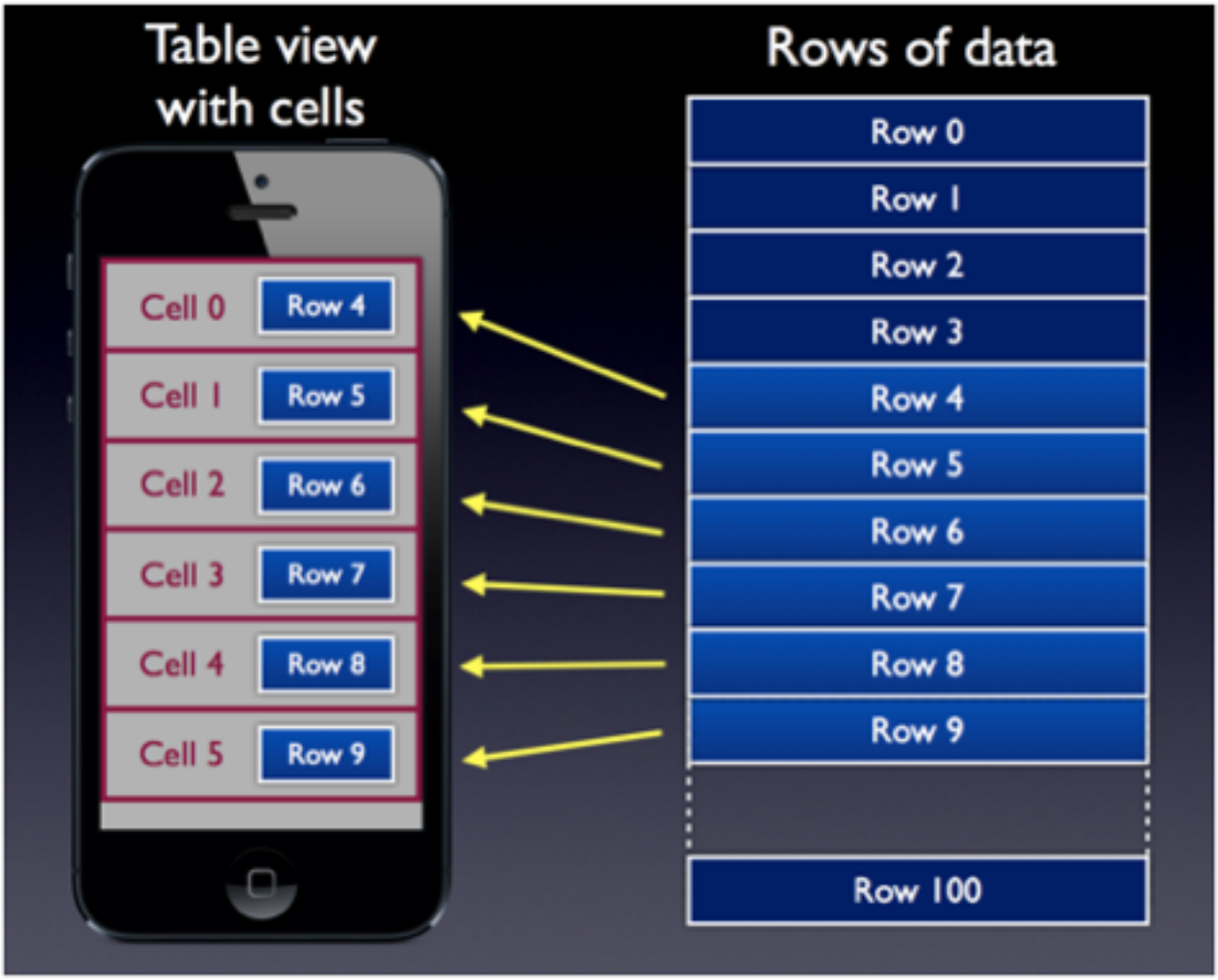
A plain-style table (left) and a grouped table (right)

如果各行的内容形式比较相似，我们会采用plain样式的表，比如地址栏中的联系方式，其中每一行都会包含一个人名。而如果各行的内容形式有所差异，我们就会采用grouped样式的表。，比如联系方式中的不同属性。比如一个姓名行，一个地址行，一个电话号码行，等等。在Checklists这款应用中，我们会用到这两种不同形式的table view。

表的数据来自于rows。在Checklists的首个版本中，每一行都会对应一个待办事项。虽然我们可以让它显示很多行（比如成千上万行），但这种设计方式是比较反人类的。想想看，如果你要滚动表格上万行才能找到自己要的东西，恐怕你会被唾沫淹死的。

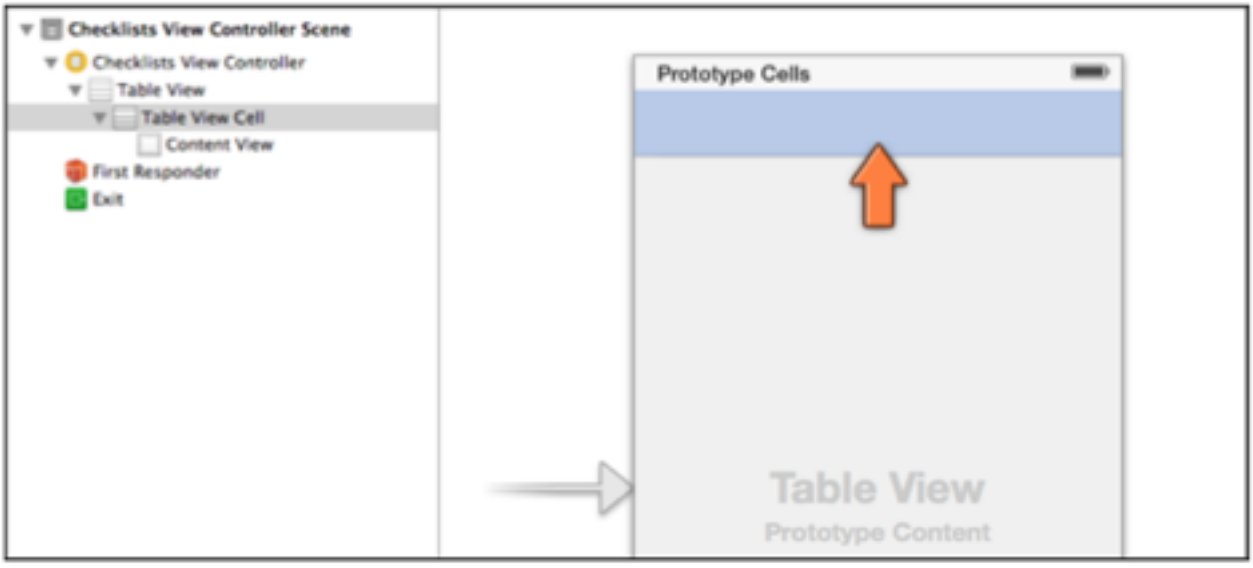
表的数据是用cells的形式来呈现的。一个cell和一个row会对应起来，但两者是不同的概念。
可能初次接触table view的时候，大家对cell和row的区别都有点搞不懂。没关系，我们先大概了解下，后面在实际操作的过程中就会逐渐熟悉起来的。
一个cell实际上是一个视觉元素，其中包含了当前时刻要显示的一行数据。如果你的表需要在同一时间显示10行数据，那么它只会有10个cells，虽然你可能会有上千rows的真实数据。
如果某一行数据被滚动到界面之外变得不可见，它所占用的cell就会被用于显示一个新的row。

这么说可能大家还是有点糊涂，还是看图更明白点。



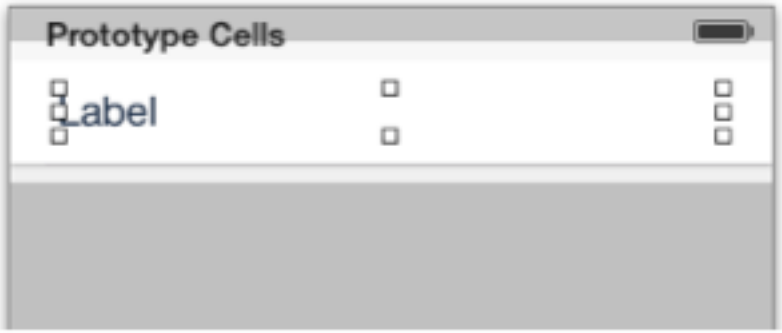
在iOS5之前，要创建一个table的cell是非常麻烦的事情，但是此后Xcode提供了一个非常方便的功能，叫prototype cells，这样我们就可以在Interface builder中可视化设计自己的cells。

好了，让我们继续来动手操作下。
在Xcode中打开storyboard，然后选择空白的cell，此时它会变成蓝色：



Selecting the prototype cell

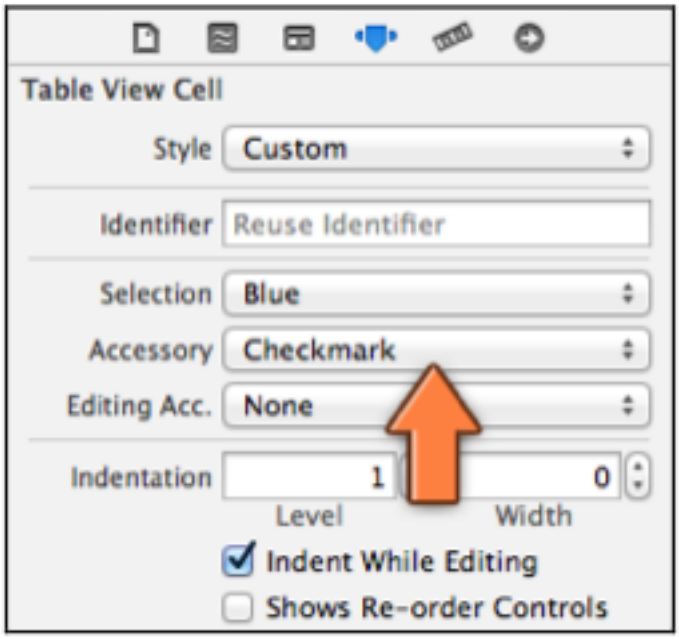
从Object Library中拖曳出一个Label放到这个cell上面。将其命名为:Checklist Item。注意让这个标签占据整个cell的宽度（两边各留少量的边缘）。



Adding the label to the prototype cell

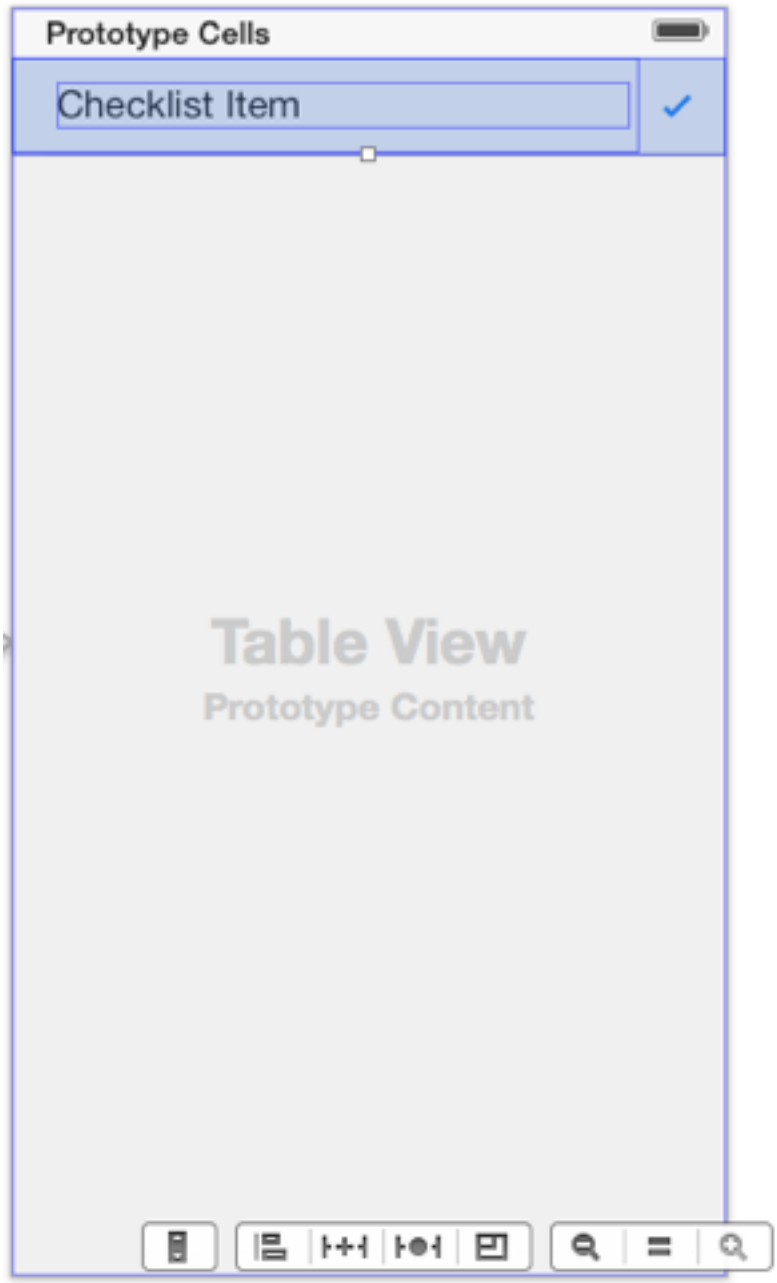
在标签旁边添加一个checkmark。checkmark是cell的accessory，也就是cell右侧的附属子视图。我们可以选择标准的附属控件，也可以使用自己设计的样式。

首先还是选中Table View Cell。然后在右侧切换到Attributes Inspector，然后将Accessory类型更改为Checkmark：



Changing the accessory to get a checkmark

完成后，表视图的设计类似下面：

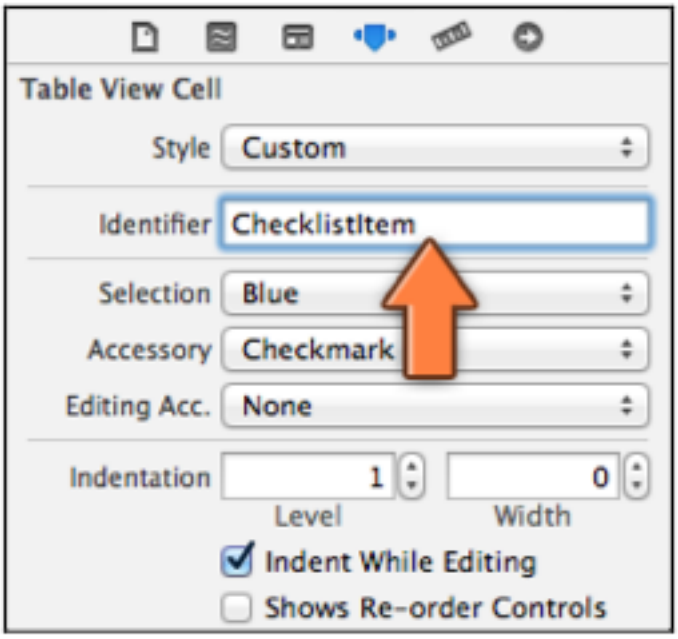


接下来我们还需要设置cell的reuse identifier。这个古怪的名词是table view表视图的内部术语，用于找到那些可以重用的cell。（比如当之前的rows滚动到界面之外，而新的rows需要显示的时候）。
此时table view需要将cell分配给那些需要显示的新rows，显然重用之前已存在的cells比创建新的cells要高效。正是因为使用了这种技术，我们的表视图滚动才会显得非常流畅（比起android感觉好很多）。

如果我们想在同一个表中显示不同类型的cells，也需要用到reuse identifier。比如，某个类型的cell中会显示一个图片和一个标签，而另一个类型的cell中会显示一个标签和一个按钮。我们需要给每个cell类型提供一个独特的identifier标识符，这样表视图才能将正确的cell分配给所需显示的cell。

尽管Checklists中只会用到一种类型的cell,但我们仍然需要给它指定一个identifier。

继续保持选中Table View Cell，然后在右侧设置Accessory类型上方的Identifier中输入ChecklistItem



Giving the table view cell a reuse identifier

好了，现在你可以试着运行下应用。然后发现。。。啥都没变。表依然在，依然没有内容。

为神马？哥裤子都脱了就为了看这个？

当然不是。

刚才我们只是给表添加了cell的设计，但并没有提供数据源rows。记住一点，cell只是表中cell的视觉呈现元素，而并非真实的数据。为了向表中添加数据，我们必须写一些代码。

添加表的数据源

在Xcode中切换到ChecklistsViewController.m，然后在@end前添加以下的代码。

```
-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return 1;
}

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

    UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:@"ChecklistItem"];
    return cell;
}
```

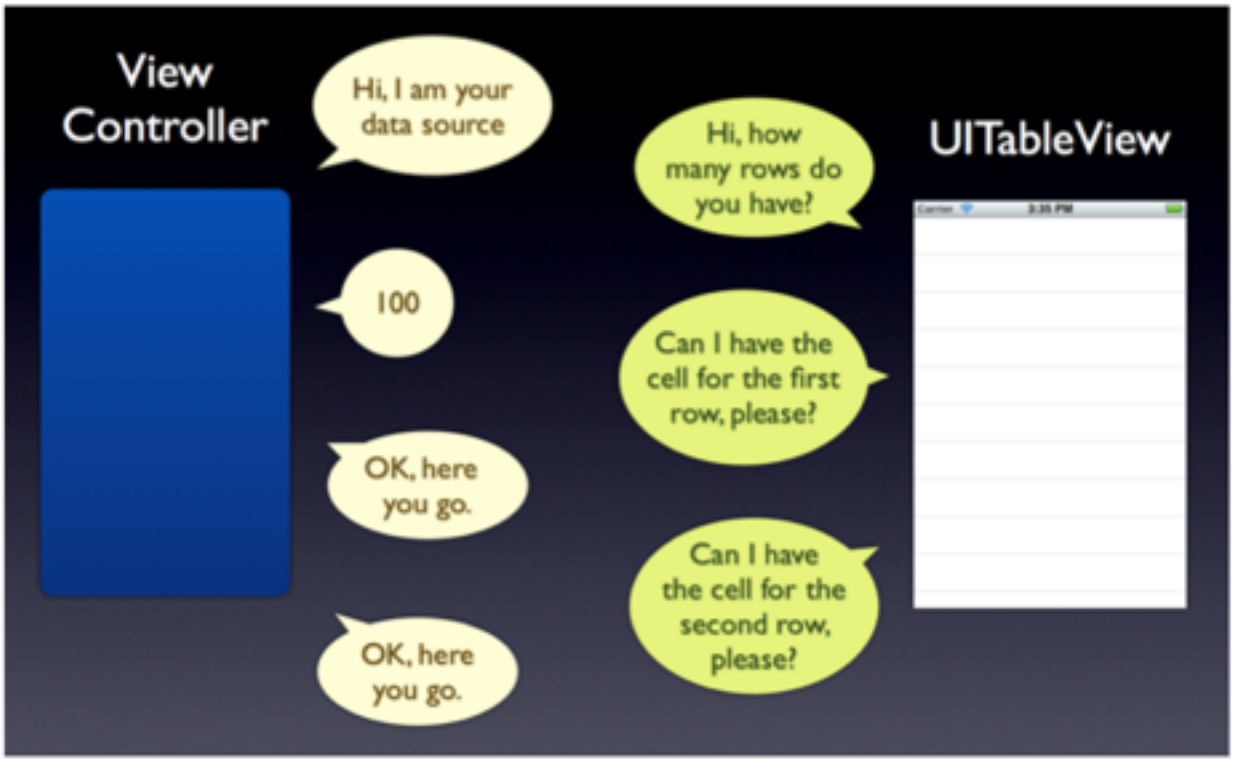
以上两个方法其实是UITableView数据源协议的一部分。所谓的数据源就是表视图（视觉元素）和数据之间的一种关联。通常视图控制器会扮演数据源的角色，并在其代码中实现相应的方法。

表视图需要知道表中会有多少航数据，同时还需要知道该如何显示每一行数据。

懒人可能会想，我们把一堆数据倒在表视图里面，然后就Ok了。好吧，你可以念出一段咒语，“啊，伟大的表视图之神，这里有子民提供的100行神圣数据，请将它们显示在界面上吧。”
表视图不是神，也听不懂人话，实际上它的智商比不上你家中的小狗。我们需要这样来和它沟通：“乖小花，这个视图控制器现在就是你的数据源。如果你需要显示数据，可以向它提一些问题。”

一旦一个表视图和数据源关联起来，当它需要知道要显示多少行数据的时候，就会发送一条numberOfRowsInSection消息。
而当它想知道如何在cell里面显示某一个具体的行数据时，就会发送一条cellForRowAtIndexPath消息给数据源。

在iOS开发中，我们会经常遇到这种模式：
某一个对象代表另一个对象来完成某些任务。这里，ChecklistsViewController向表视图提供数据，不过仅当表视图请求的时候才会这样做。



The dating ritual of a data source and a table view

你可能要问了，何必这么麻烦呢？干脆让表视图自己提供自己的数据来源不更简单，为神马还要多此一举让视图控制器帮它提供数据？

原因是，iOS应用的开发遵循非常严格的MVC（Model-View-Controller)模式。表视图只不过是一种视觉元素，它不能也不应该和具体的数据扯在一起。我们只有通过controller视图控制器这个桥梁将视图和数据模型关联起来。

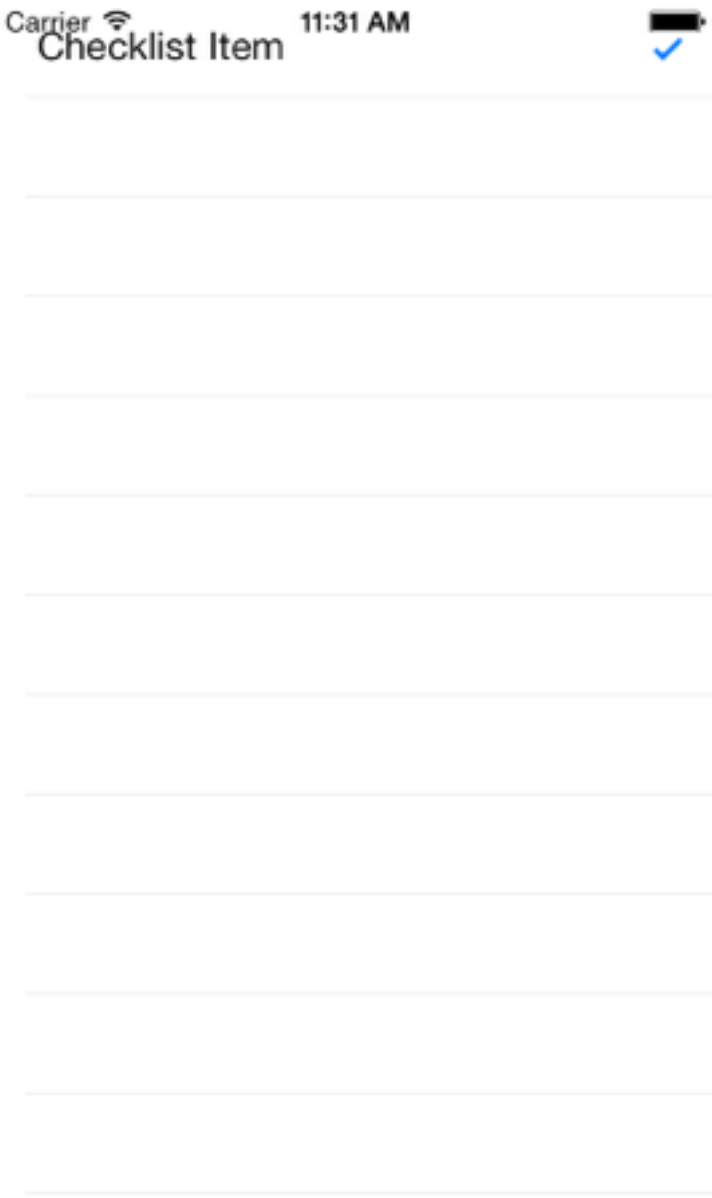
如果你学过其它编程语言和框架，会发现iOS应用的开发模式是最符合MVC的，逻辑上也是最清晰的。

好了，如果MVC暂时让你头疼，先不管它，回到我们这个具体的例子。

我们在numberOfRowsInSection方法中返回的是数字1，其实就是告诉表视图我们只需要显示1行数据。如果要显示10行呢？只需要更改这个数字就可以了。你懂的。

接着表视图会调用cellForRowAtIndexPath方法为某一行数据获取一个cell。
在该方法中，我们简单获取了prototype cell（原型表格）的copy，然后将其返回给表视图。通常我们会在这个方法中将行数据放到cell中，不过到目前为止这个应用还没有任何行数据。所以我们就用最简单的prototype cell。

好了，现在可以试运行应用，看看效果如何。



当然，这个样子依然难看，但至少多了一行显示的内容。iPhone的状态栏和表视图的显示内容部分重叠了。这是iOS7出现的怪现状。
我会告诉你我对iOS7的各种UI设计仍然是很不感冒吗。
关于这个小问题，后续我们会将其修复，现在先不管它。可以透露下，只需要在表视图顶部放一个导航栏就可以了。

Objective-C充电- 有多个参数的方法

在之前的学习中，我们只接触过带有单个参数的方法，或者完全没有参数的方法。但在表视图数据源方法中我们碰到了带两个参数的方法。

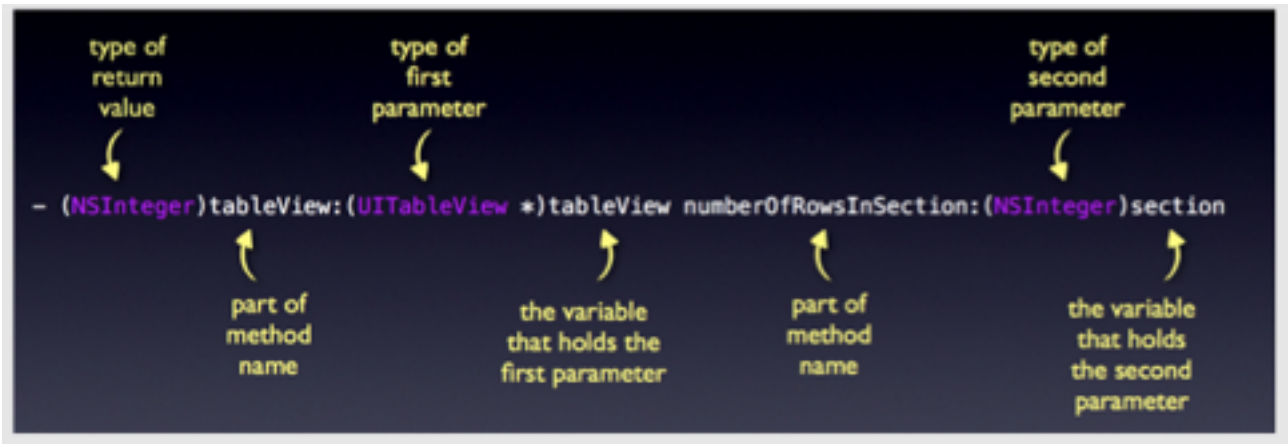
```
- (NSInteger)tableView:(UITableView *)tableView // parameter 1 numberOfRowsInSection:
(NSInteger)section // parameter 2
{
...
}
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath
{
...
}
// 1 // 2
```

在两个方法中，第一个参数都是UITableView对象（触发这些方法的表视图），而前一个方法的第二个参数是section 编号，后一个方法的第二个参数是所谓的indexPath。
如果你想知道NSIndexPath是个什么东西，可以按住option，鼠标左键点击它调出相关的文档来看。

在其它的编程语言中（比如C,C++），带有多个参数的方法同时用下面的形式来展示：

```
void numberOfRowsInSection(UITableView *tableView,
NSInteger section)
{
...
}
```

但在Objective-C中却并非如此。最开始接触Objective-C的时候，我被这种奇怪的多参数显示方式吓到了。但过了一段时间之后，我就发现这种显示方法更加具有可读性。
现在简单总结下一个方法的不同部分。



以上方法的官方正式命名应该是tableView:numberOfRowsInSection:（包含冒号）。如果你大声读出来，就会发现这种命名方式是有意义的。该方法的作用就是获取某个特定表视图中某个特定section的行数量。

小练习：修改代码，让表视图中显示五行信息。

这个应该很简单吧。

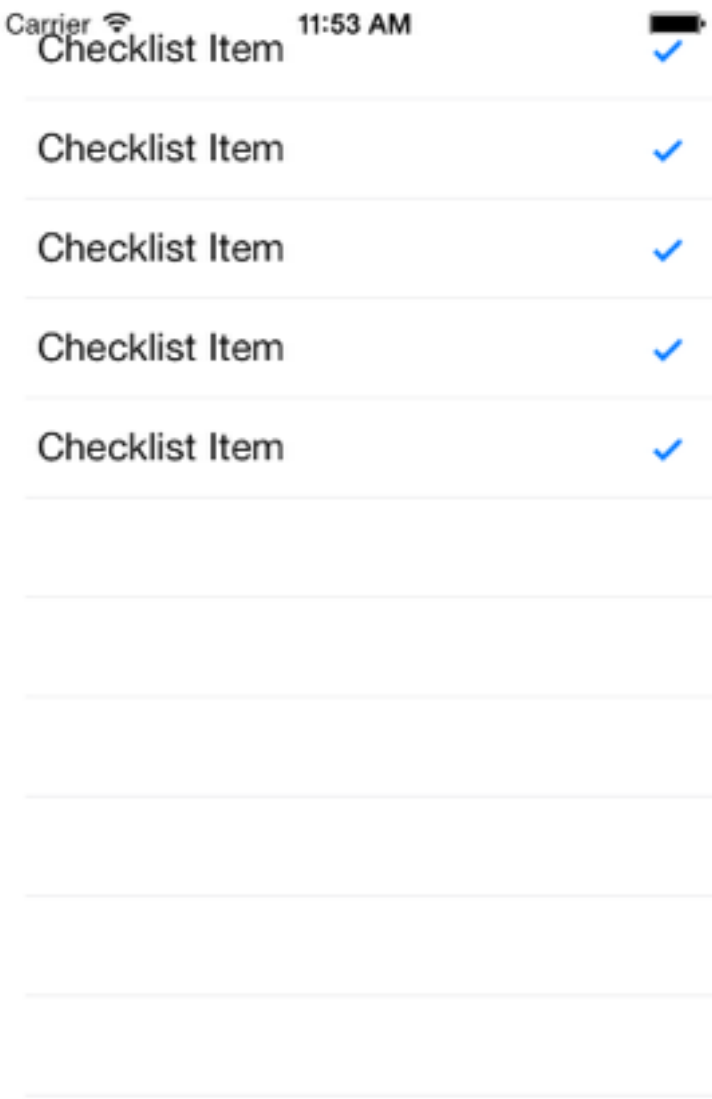
```
-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return 5;
}
```

只需要把数字改成5就Ok了。

但有些哥们不喜欢代码，更喜欢可视化操作。于是他们跑到storyboard里面，然后把prototype cell复制粘贴了5次，好吧，这下其实弄巧成拙了。

当我们在以上方法中将返回数值设置为5的时候，其实就是告诉表视图让它显示5行数据。此时表视图会调用cellForRowAtIndexPath5次，也就是每一行数据调用1次该方法。

因为cellForRowAtIndexPath方法中只需要返回一个prototype cell的copy,所以表视图就会显示5个相同的行。



我们可以使用不同的方法在cellForRowAtIndexPath方法中为行数据创建所需的cell，但目前来说最简单的方法是：在storyboard的表视图中添加一个prototype cell，然后调用[tableView dequeueReusableCellWithIdentifier]方法。这个听起来有点可怕，其实只不过是在需要的时候创建了prototype cell的一个副本，或者是重复利用了一个已经存在的cell。

一旦我们有了所需的cell后，就可以从相应行中获取数据来填充，并让其显示在表视图中。在下一部分的内容我们就会学习如何具体来实现。

关于Index path的脑补

刚才提到了在Xcode中按住option点击NSIndexPath可以查看相关的文档，现在让我们来了解下这个究竟是个什么玩意吧

在刚才的cellForRowAtIndexPath方法中，表视图使用该方法为某个cell请求数据源。
但index-path究竟是啥?

简单来说，NSIndexPath是指向表中某个特定行的对象。它由一个行编号和一个section编号共同组成，就这么简单。当表视图为cell请求数据源的时候，程序会查看indexPath.row的行编号属性，从而确定该cell需要哪一行数据。

在使用表视图的时候，我们还有可能把不同的行放进section里面。比如在地址薄应用中，我们会使用姓来查找联系人。所有姓以A开头的联系人会被分到同一个section组中，而所有姓以B开头的联系人会被放到另一个section组中，以此类推。
为了查找某一行所属的section，我们需要使用indexPath.section属性。因为Checklists应用不会用到这种分组方式，所以可以暂时忽略NSIndexPath的section属性。

最后要注意的是:

在程序世界里，万物从0开始。所有如果你的列表中有四个事项，那么它们的编号是0,1,2和3。当然这个有点反人类，毕竟我们习惯了数自己的手指1，2，3，4。不过在程序世界里面就是这样的。因此，对于第一个section中的第一行，indexPath.row的值是0，而indexPath.section的值也是0.第二行的编号是1，第三行的编号是2，以此类推。

其实万物从0开始也是符合阴阳太极理论的。老子曰过：“道生一，一生二，二生三，三生万物。”
这里的道可以看做0？万物负阴而抱阳，冲气以为和。0和1构成了变幻无穷的程序世界，岂不对应了阴阳?
当然，以上只是瞎掰，不要当真。

好了，第一章的学习到此结束，让我们来送上点福利吧。

新出的保卫萝卜2有米有玩过啊？强烈推荐啊。


保卫萝卜2
CARROT FANTASY
Copyright © 2013 LUOBO. All rights reserved.

然后是美女