从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter7

欢迎继续我们的学习。

在上一章的学习中我们学会了使用"+"添加按钮来添加新的代办事务，虽然是完全相同的。
而在这一章，我们将要学习一个新东西，也就是用导航栏上的按钮来打开一个新的界面，让用户在里面添加新项目。

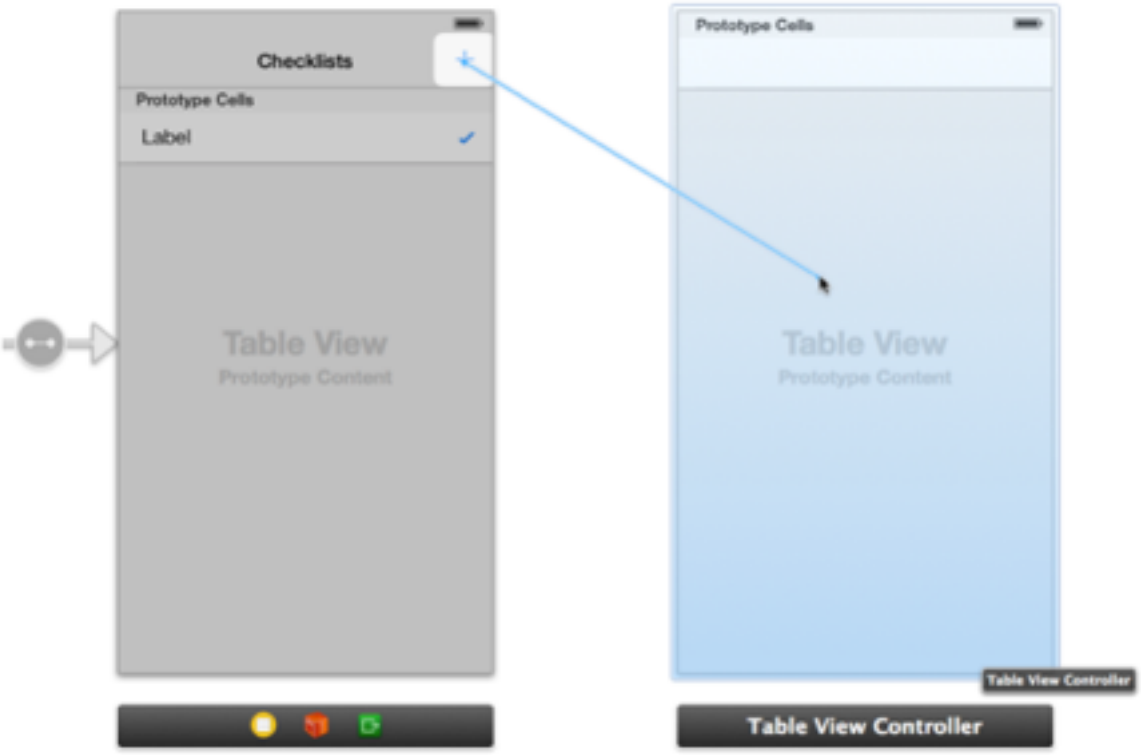
这一部分要学习的内容包括：
1.借用storyboarding的力量来创建一个Add Item界面
2.添加一个文本输入框，让用户可以使用内置键盘来输入信息
3.判断何时用户触碰Add Item界面上的Cancel或Done按钮
4.使用文本输入框里面所输入的文本来创建一个新的ChecklistItem
5.在主界面的表视图中添加新创建的ChecklistItem对象

新的界面意味着会有新的视图控制器，因此我们首先要做的事情是在storyboard中添加一个新的界面。

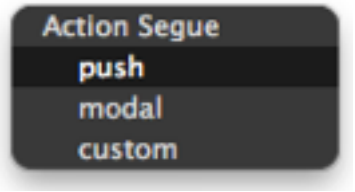
打开Xcode,切换到Main.storyboard。
在Xcode界面右侧的Objects Library中找到Table View Controller，然后把它拖到storyboard的画布上。

就这么简单~

现在新的视图控制器已经就位，选中Checklists View Controller上的添加"+"按钮，按住ctrl键，用鼠标拖出一条线到新的视图控制器。



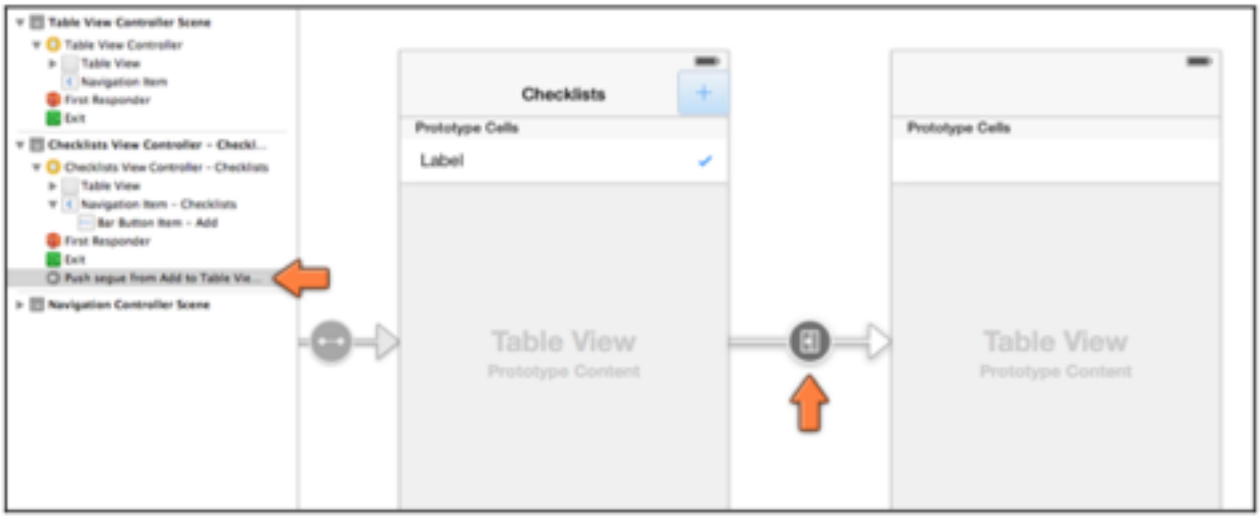
此时松开鼠标，会看到一个弹出菜单：



三个选项意味着从"+"按钮到新界面间的三种不同的关联方式。这里我们先选择push，至于为什么，后面会详细解释。

这种在storyboard中创建的关联叫做segue（如果你不知道怎么发音，可以想想被帮主痛骂过的seg-way，也就是下面这个家伙）。

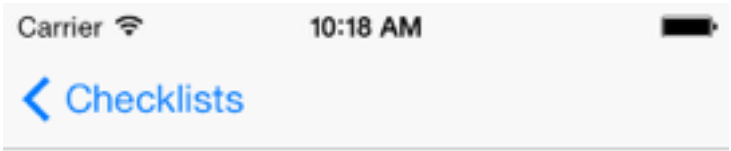
这个时候你回看到在我们之前的Checklists View Controller和后来新添加的视图控制器间多了一个看起来很奇怪的东东，这就是segue。



好了，到目前为止我们没有添加一行新代码。

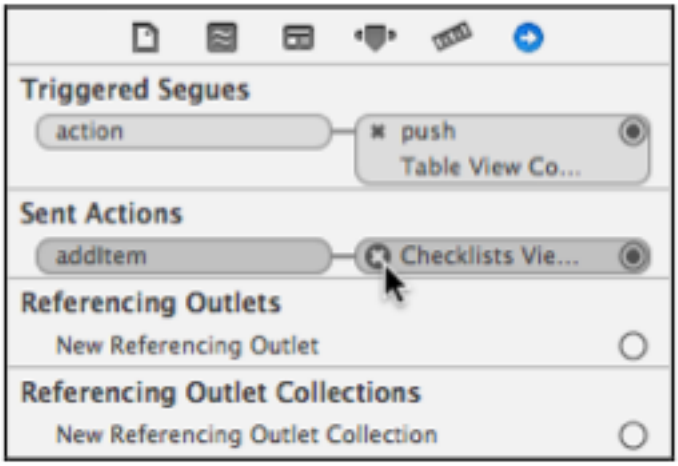
编译运行应用，看看发生了什么变化~

首界面没有任何变化，只是当我们触碰导航栏上的“+”按钮时，会从右侧滑入一个新的空白表视图。这时如果你触碰左上的返回按钮（文字部分是Checklists”，就会回到之前的界面。



当然，你也会注意到"+"按钮之前的功能-向表中添加一行新数据-已经消失了。这是因为之前的关联被segue替代了。不过为了保险起见，我们还是手动来删除下这种关联吧。

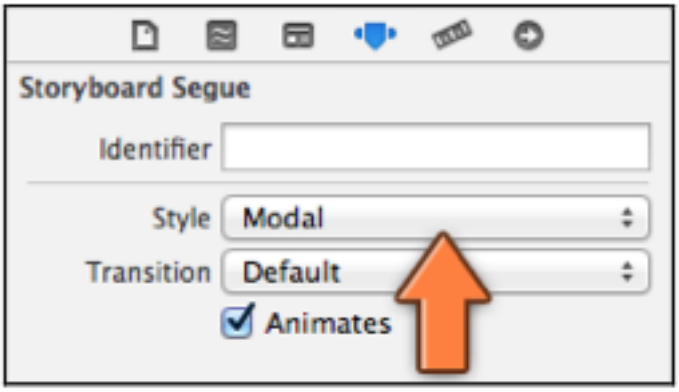
选中"+"按钮，然后在Xcode右侧的面板中切换到Connections inspector（也可以用右键单击该按钮的方式）。然后点掉Sent Actions下面的addItem关联的那个小叉子。



注意到这里也显示了一个Triggered Segues，就是我们刚刚所创建的segue。

通过这种方式，当我们触碰"+"添加按钮时，就会自动切换到新的表视图控制器。但为了在新的界面中可以添加新项目，我们最好换用modal类型的segue。

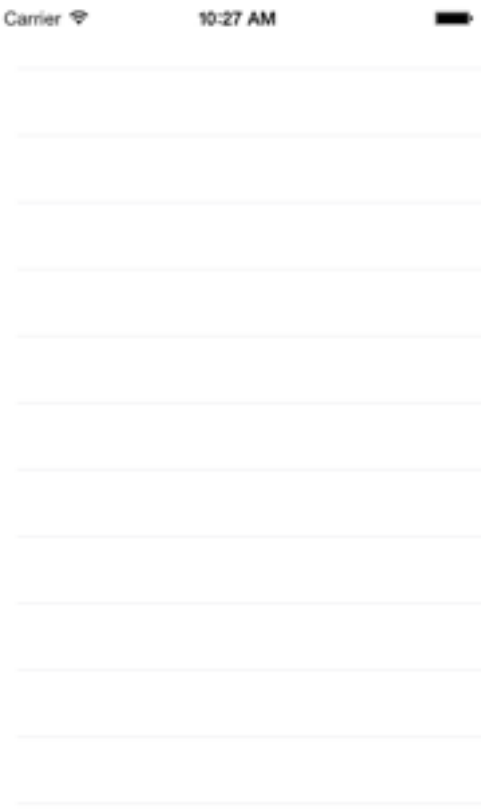
选中两个视图控制器之间的箭头，从而就选中了这个segue。注意，segue也是一个对象，因此我们也可以修改它的属性。在Xcode右侧面板中切换到Attributes inspector，在Style部分选中Modal。



注意到，新的视图控制器上的导航栏已经消失了。此时新的界面不再作为导航层级的一部分呈现，而是作为和之前界面独立的界面存在。

此时编译运行应用，感受下。

好吧，我们再也回不去了~



对于Modal类型的界面，其导航栏的左侧通常有一个Cancel按钮，右侧有一个Done按钮（在有些应用中右侧的按钮被命名为Save或Send）。触碰这些按钮将关闭当前界面，但惟有触碰Done才会保存所做的变动。

添加一个导航栏和两个按钮的最简单方式是把Add Item界面的视图控制器置入一个自己的导航控制器，其操作和之前类似：

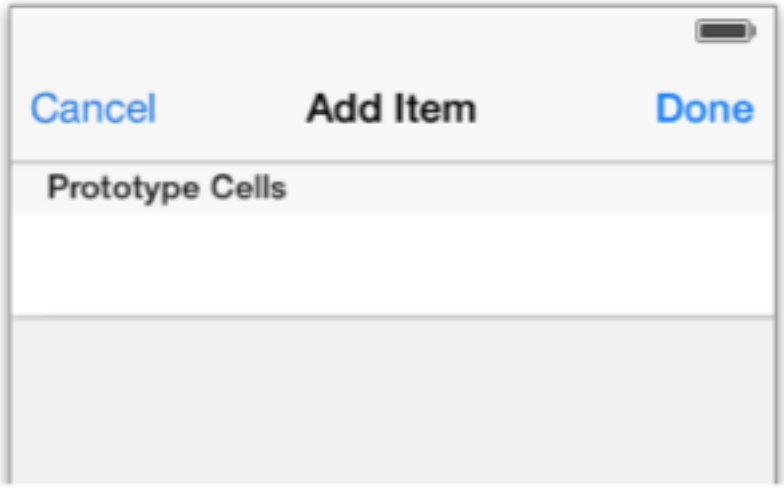
选中当前的table view controller（新创建的），然后从Xcode的菜单中选中Editor–Embed In–-Navigation Controller即可。

现在整个storyboard又变了：

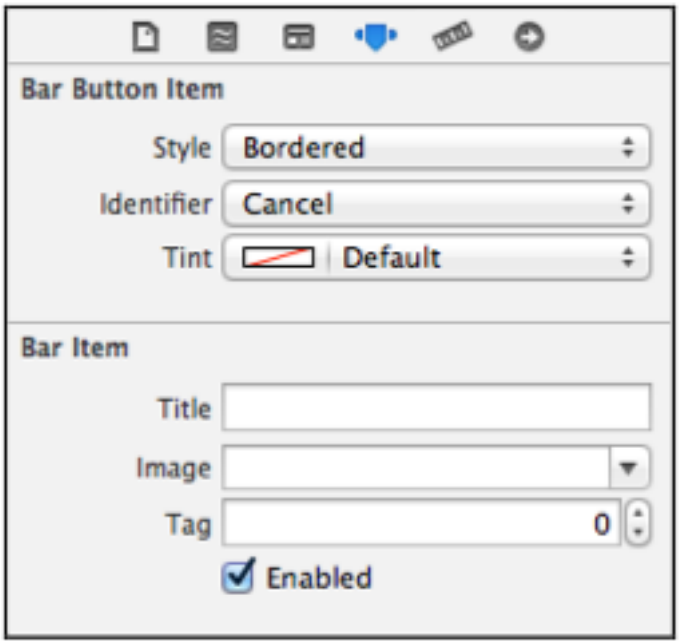
在两个表视图控制器之间插入了一个新的导航控制器，此时原来的"+"按钮的作用就是执行一个modal类型的segue从而切换到新的导航控制器。

双击最右边表视图控制器的导航栏，把标题改为Add Item。然后拖出两个Bar Button Items到导航栏的左侧和右侧。

分别选中两个bar button item,在Xcode右侧面板中切换到Attributes inspector，然后修改左侧的Identifier的属性为Cancel，把右侧那个bar button item的Identifier和Style都改为Done

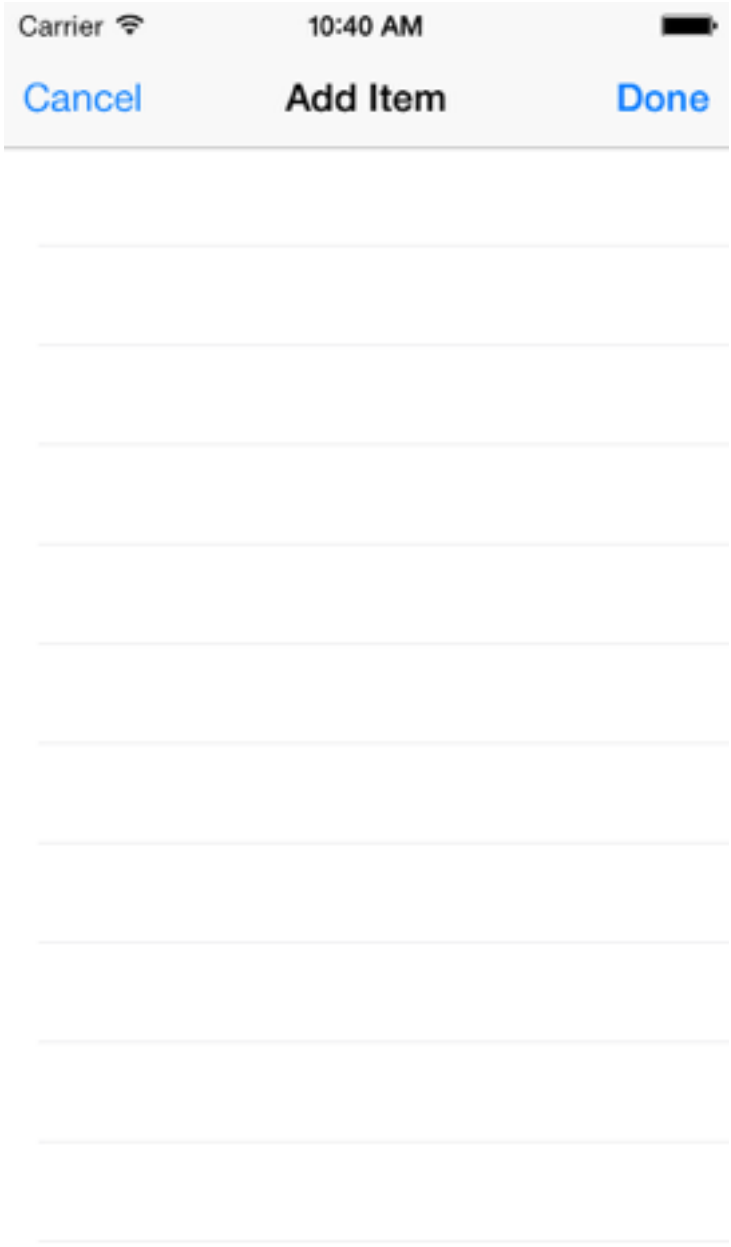


注意我们不需要手动输入这两个按钮的文字标题，Cancel和Done按钮是iOS系统内置的按钮类型，用就是了。



当然，你也可以把Identifier处设置为Custom，然后输入自己喜欢的文字，随你。
不过用系统预定义按钮的好处是，如果你的应用运行在默认语言为非英语国家的iPhone设备上，预定义的标准按钮会自动帮你翻译成用户的语言。

现在再来编译运行应用，感受下新的界面。



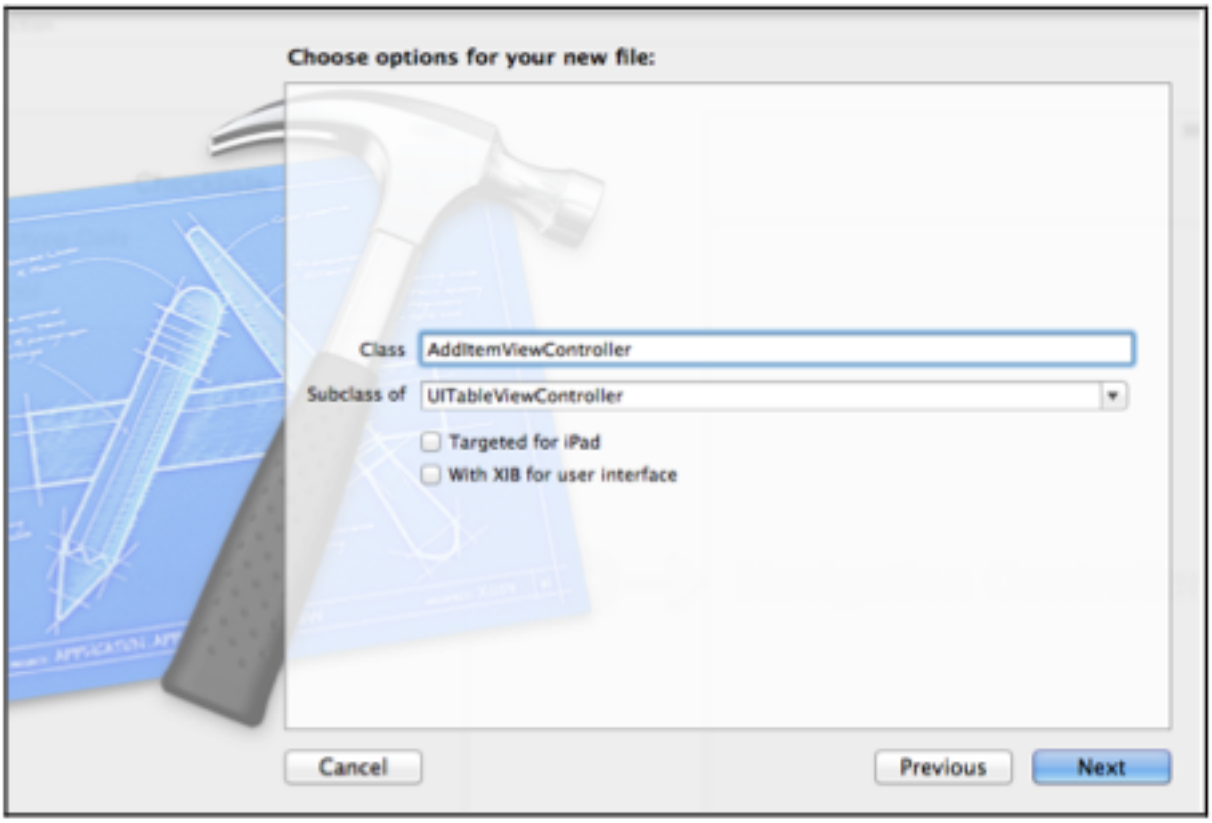
现在界面的导航栏上出现了两个按钮，可以触碰它们暂时没有任何反应。

在下一篇教程中我们将学习如何在storyboard直接实现一个所谓的”反向“segue，不过这里我们需要用代码来解决。换句话说，我们需要把这些按钮和action方法关联起来。

好吧，我们应该在那里放这些action动作方法呢？显然不是ChecklistsViewController，因为它不是主管当前界面的视图控制器。
肿么办？我们都知道一个界面对应一个视图控制器。现在storyboard里面已经有了一个新的界面，接下来我们需要手动创建一个新的视图控制器来对应它。

回到Xcode，右键单击项目导航器中的Checklists群组，选择New File…，然后选择Objective-C class模板，在下一步中选择以下选项：

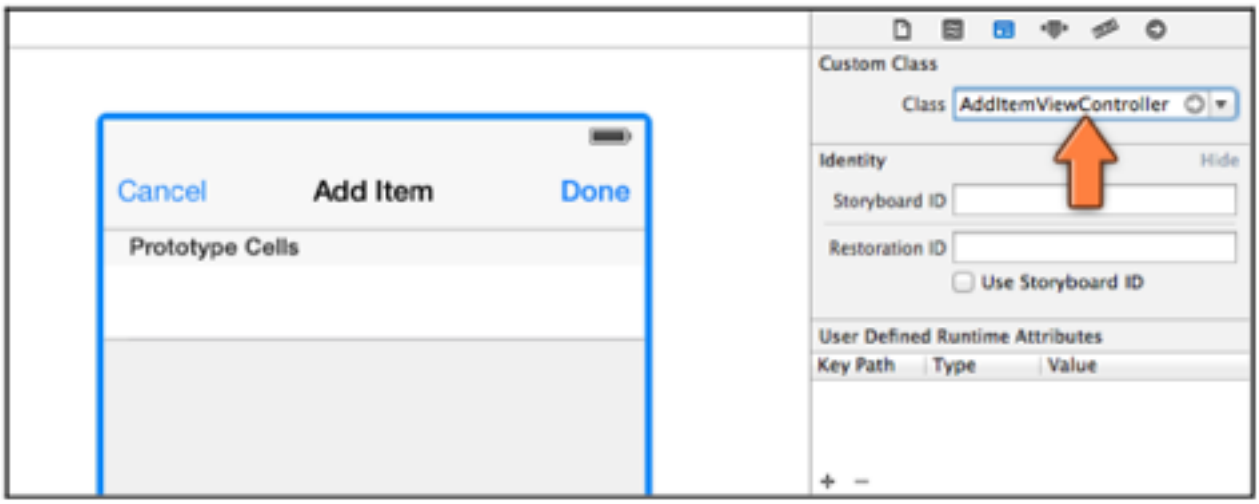
Class: AddItemViewController
Subclass of: UITableViewController
Targeted for iPad: 不要选中
With XIB for user interface: 不要选中



一定要注意Subclass of要选中UITableViewController，而不是一般的UIViewController，这一点极为重要！！！

此时我们会看到项目中多了两个文件，AddItemViewController.h和.m。不过目前它们和storyboard的界面元素没有任何关联。

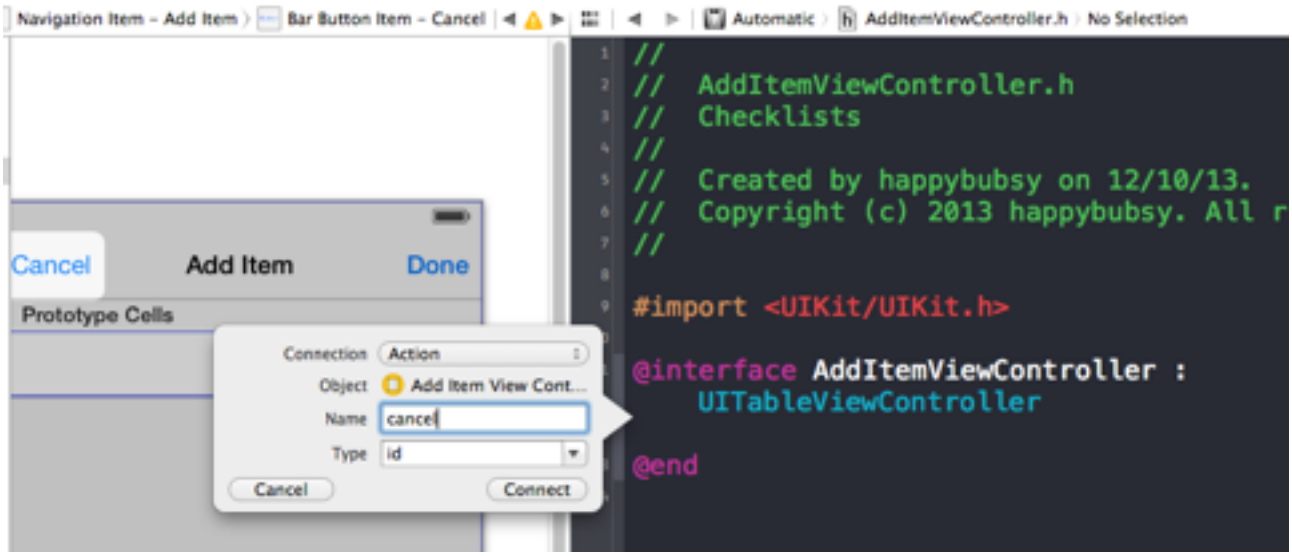
让我们回到storyboard，选中新添加的表视图控制器，然后在Xcode右侧面板中切换到Identifity inspector，在Custom Class下面输入AddItemViewController。



通过这种方式，我们通知了storyboard该界面的视图控制器实际上就是刚刚新创建的AddItemViewController

千万别忘了这一步！如果没有设置这个Class，Add Item界面就完全不会工作。当然，你还得确保在storyboard中选择了正确的视图控制器。一个常见的错误是选中了表视图而不是表视图控制器。

接下来就是分别创建两个按钮的action动作方法了，你应该已经很熟悉了，如果忘了可以参考下面的图。



两个关联中，其中一个Name是cancel，另一个是done,其它设置完全相同。

最后当然是要在AddItemViewController.m中具体实现动作方法了。

不过在此之前，我们首先注意到Xcode在创建.h和.m文件的时候已经加了一大堆不需要的代码。因此先让我们删掉一些没用的东西。
切换到AddItemViewController.m，删掉@end之前，#pragma mark - Table view data source 之后的所有代码。

我们刚刚所删除的代码中包含numberOfRowsInSection,cellForRowAtIndexPath和didSelectRowAtIndexPath等几个方法。它们是表视图的数据源和代理方法，不过对于我们这个特殊的视图控制器，暂时还用不到。

注意：以上步骤一定不能忽略，否则该界面不会正常工作的。

最后在@end前添加cancel和done两个动作方法的实现代码：

```
- (IBAction)cancel:(id)sender {

    [self.presentingViewController dismissViewControllerAnimated:YES completion:nil];
}

- (IBAction)done:(id)sender {

    [self.presentingViewController dismissViewControllerAnimated:YES completion:nil];
}
```

以上代码中，我们让"presenting view controller"，也就是呈现该modal界面的视图控制器来关闭当前界面。
注意千万不要用presentedViewController，否则你会郁闷死的~

那么在我们的四个视图控制器中，哪个才是presenting view controller呢？猜猜看？

答案是- 包含了ChecklistsViewController的UINavigationController。
猜对了吗？iOS开发的视图和视图控制器层级有时候会让你糊涂，不过-Don't panic，哥与其讲一大堆理论知识让你更糊涂，不如用实际的项目来引导你一步步认识这些东西。

编译运行应用，现在Cancel和Done按钮可以安全带你回家了。

那么，当你dismiss（打发掉）AddItemViewController对象的时候究竟发生了什么？当该视图控制器从界面中消失后，它的对象会被销毁，所使用的内存会被系统回收，也就是说，它魂归天国了。
每次用户打开Add Item界面时，我们看到的是一个新的实例。
这就意味着一个视图控制器对象的生命仅在用户和它交互的时候存在，此后它会彻底消失。

理论脑补- Container view controller 视图控制器容器？

之前我们说过，一个视图控制器用来呈现一个界面，但这里却看到一个界面对应了两个视图控制器，这不科学啊。
比如应用的主界面包含了一个内置在导航控制器中的ChecklistsViewController表视图控制器，而Add Item界面又由内置在其自身的导航控制器中的AddItemViewController表视图控制器组成。

导航控制器其实是一类非常特殊的视图控制器，它的作用不是直接管理一个界面，而是作为其它视图控制器的容器存在。它带有一个导航栏，并可以轻松从一个界面跳转到另一个界面。作为容器的导航控制器只是管理其它视图控制器，真正的界面"内容"还是由其它视图控制器来呈现。

另一个常用的视图控制器容器是Tab Bar Controller,在下一篇教程中我们将会学习到。
在iPad应用开发中，视图控制器容器的使用就更普遍了，比如popover或者split-view等等。

好了，今天的学习先到此结束，又到了发送福利时间。

神马？！美女对你的刺激已经不够了，还可以看看Obama呼吁每个美国人都学习编程的视频
http://v.youku.com/v_show/id_XNjQ2MTU1NzM2.html

If we want America to stay on the cutting edge, we need young Americans like you to master the tools and technology that will change the way we do just about everything.

Don’t just buy a new video game, make one.
Don’t just download the latest app, help design it.
Don’t just play on your phone, program it.