从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter16

欢迎继续我们的学习。

开发环境：
Xcode 5.1 DP 2 +iOS 7.1 beta2

在上一章的内容中我们成功添加了一个新的All Lists界面，现在数据模型中包含了来自AllListsViewController的_lists 数组，以及来自ChecklistViewController的_items数组，当然，还包含了数组中的Checklist和ChecklistItem对象。

当我们触碰某个checklist时，就会打开相应的Checklist界面，只不过目前其中的内容都是预设的相同内容。在实际使用中，每个checklist应该包含各自不同的代办事项。
我们在后面的学习中将对此进行一些修正，它将涉及到对数据模型的修改。

在此之前，先让我们完成其它的任务。

首先让我们来修改界面的标题，来反应所选中的checklist。
在Xcode中切换到ChecklistViewController.h，然后修改其中的代码如下：

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

@class Checklist;

@interface ChecklistViewController :
UITableViewController<ItemDetailViewControllerDelegate>

@property(nonatomic,strong)Checklist *checklist;

@end
```

通过以上代码，我们添加了一个Checklist类型的属性变量，名称就是checklist。别忘了在顶部添加@class Checklist，否则编译器会提示出错的。

接下来切换到ChecklistViewController.m，在顶部添加一行代码以导入Checklist对象的完整定义：

```
#import “Checklist.h"
```

接着更改viewDidLoad方法的代码如下：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
```

```
        // Do any additional setup after loading the view, typically from a nib.
    self.title = self.checklist.name;
}
```

我们添加了一行代码，从而当用户触碰不同的checklist时会显示不同的标题。这个标题会显示在导航栏中，其内容就是Checklist对象的名称。

接下来切换到AllListsViewController.m，更改didSelectRowAtIndexPath方法的内容如下：

```
-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{

    Checklist *checklist = _lists[indexPath.row];
    [self performSegueWithIdentifier:@"ShowChecklist" sender:checklist];
}
```

和往常一样，我们仍然使用performSegueWithIdentifier来启动segue。该方法有一个sender参数，之前设置为nil。现在我们将它设置为与用户所触碰的行相对应的Checklist对象。

事实上我们可以将任何东西放入sender中。如果segue是由storyboard触发（而非这里的手动触发），那么sender就会指向触发它的控件。比如+按钮的UIBarButtonItem对象，或者是表视图中行所对应的UITableViewCell对象。
因为这里是手动触发segue，所以可以方便的将任何对象放入sender中。

不过将Checklist对象放入sender参数并没有将该对象传递给ChecklistViewController。这件工作是由prepareForSegue方法完成的，因此我们需要继续提供相关的代码。

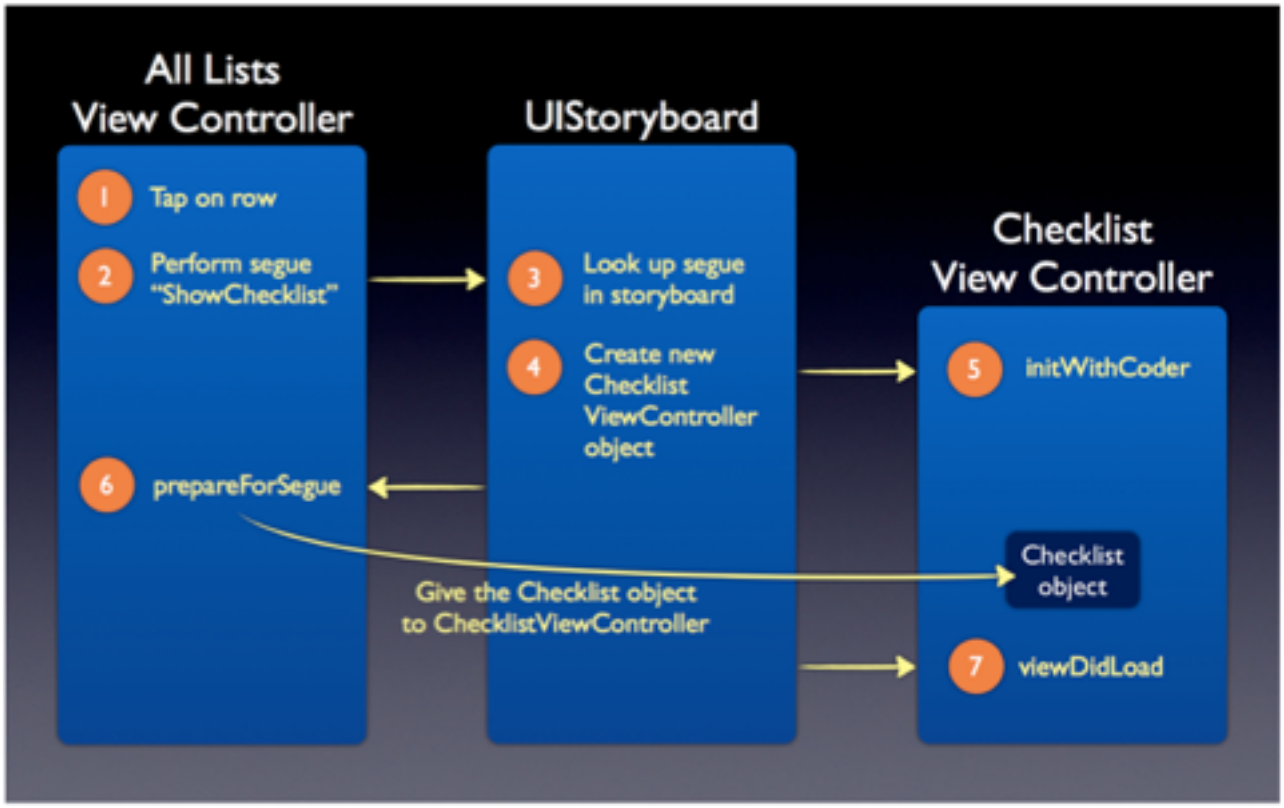
在刚才的didSelectRowAtIndexPath方法下面添加以下的方法：

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{

    if([segue.identifier isEqualToString:@"ShowChecklist"]){

        ChecklistViewController *controller = segue.destinationViewController;
        controller.checklist = sender;
    }
}
```

在之前的学习中我们接触过prepareForSegue方法，它会在触发segue时由storyboard来调用。在这个方法中我们可以在新的视图控制器可见前设置一些属性。

我们需要传递与用户所触碰的行相对应的Checklist对象。为此我们在刚才的方法中将该对象放入sender参数。你也可以选择把Checklist对象放到一个临时的实例变量中，不过通过sender参数传递更加简单。

所有这一切都在ChecklistViewController的视图被加载前完成，这样viewDidLoad方法就可以根据checklist的名称属性来设置界面的标题了。

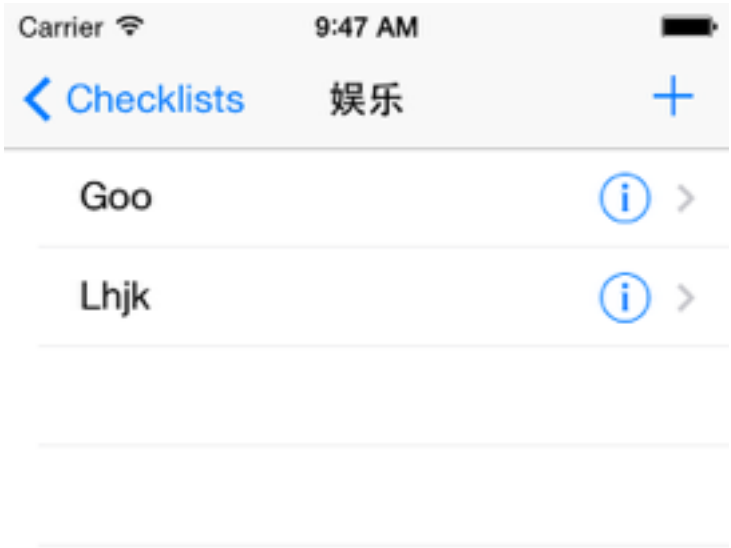


上图是当前应用中该阶段segue的执行流程：
1.用户触碰表视图中的某一行
2.触发ShowChecklist这个segue
3.在storyboard中查找对应的segue
4.创建新的Checklist View Controller对象
5.调用Checklist View Controller对象的initWithCoder方法进行初始化
6.调用All Lists View Controller的prepareForSegue方法
7.All List View Controller将与用户触碰的行相对应的checklist对象传递给ChecklistViewController
8.调用Checklist View Controller的viewDidLoad方法加载界面视图。

好了，最后别忘了在AllListsViewController.m的顶部添加一行代码：

```
#import "ChecklistViewController.h"
```

这就搞定了。
编译运行应用，当你触碰首界面上的某个checklist时，会进入相应的界面，而导航栏上的标题会显示checklist的名称。



需要提醒的是，当我们把Checklist对象传递给ChecklistViewController的时候，并不是让它获得了该对象的一个备份。这里仅仅是传递给视图控制器一个引用（或者有一个更高深的名词pointer指针）。这样用户对它的任何更改也会在AllListsViewController中体现出来。两个视图控制器都可以访问同一个Checklist对象。

继续下一步，添加和编辑checklist

现在让我们快速添加Add Checklist/Edit Checklist界面。显然它仍然是一个UITAbleViewController，这次仍然是static cell，而我们将从AllListsViewController以模态（modally)的形式切换到该界面。

首先在项目中添加一个新的文件，设置为subclass of UITableViewController，命名为ListDetailViewController。具体如何创建这里就不再重复了。

接下来切换到ListDetailViewController.h，更改其中的代码如下：

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

@class ListDetailViewController;
@class Checklist;

@protocol ListDetailViewControllerDelegate <NSObject>

-(void)listDetailViewControllerDidCancel:(ListDetailViewController*)controller;
-(void)listDetailViewController:(ListDetailViewController*)controller
      didFinishAddingChecklist:(Checklist*)checklist;
-(void)listDetailViewController:(ListDetailViewController*)controller
     didFinishEditingChecklist:(Checklist*)checklist;

@end

@interface ListDetailViewController : UITableViewController<UITextFieldDelegate>

@property(nonatomic,weak)IBOutlet UITextField *textField;
@property(nonatomic,weak)IBOutlet UIBarButtonItem *doneBarButton;
@property(nonatomic,weak) id <ListDetailViewControllerDelegate> delegate;

@property(nonatomic,strong) Checklist *checklistToEdit;

-(IBAction)cancel:(id)sender;
-(IBAction)done:(id)sender;

@end
```

以上代码的内容看起来非常熟悉，实际上我就是从ItemDetailViewController.h中拷贝粘贴过来，然后更改了名称而已。同样的，现在的属性变量是Checklist类型，而不是ChecklistItem

切换到ListDetailViewController.m，在文件顶部添加一行代码：
#import "Checklist.h"
然后更改viewDidLoad方法的代码如下：
```
- (void)viewDidLoad
{
    [super viewDidLoad];

    if(self.checklistToEdit !=nil){
        self.title = @"Edit Checklist";
        self.textField.text = self.checklistToEdit.name;
        self.doneBarButton.enabled = YES;
    }
}
```
接着添加viewWillApprear方法的代码如下：
```
-(void)viewWillAppear:(BOOL)animated{

    [super viewWillAppear:animated];
    [self.textField becomeFirstResponder];
}
```

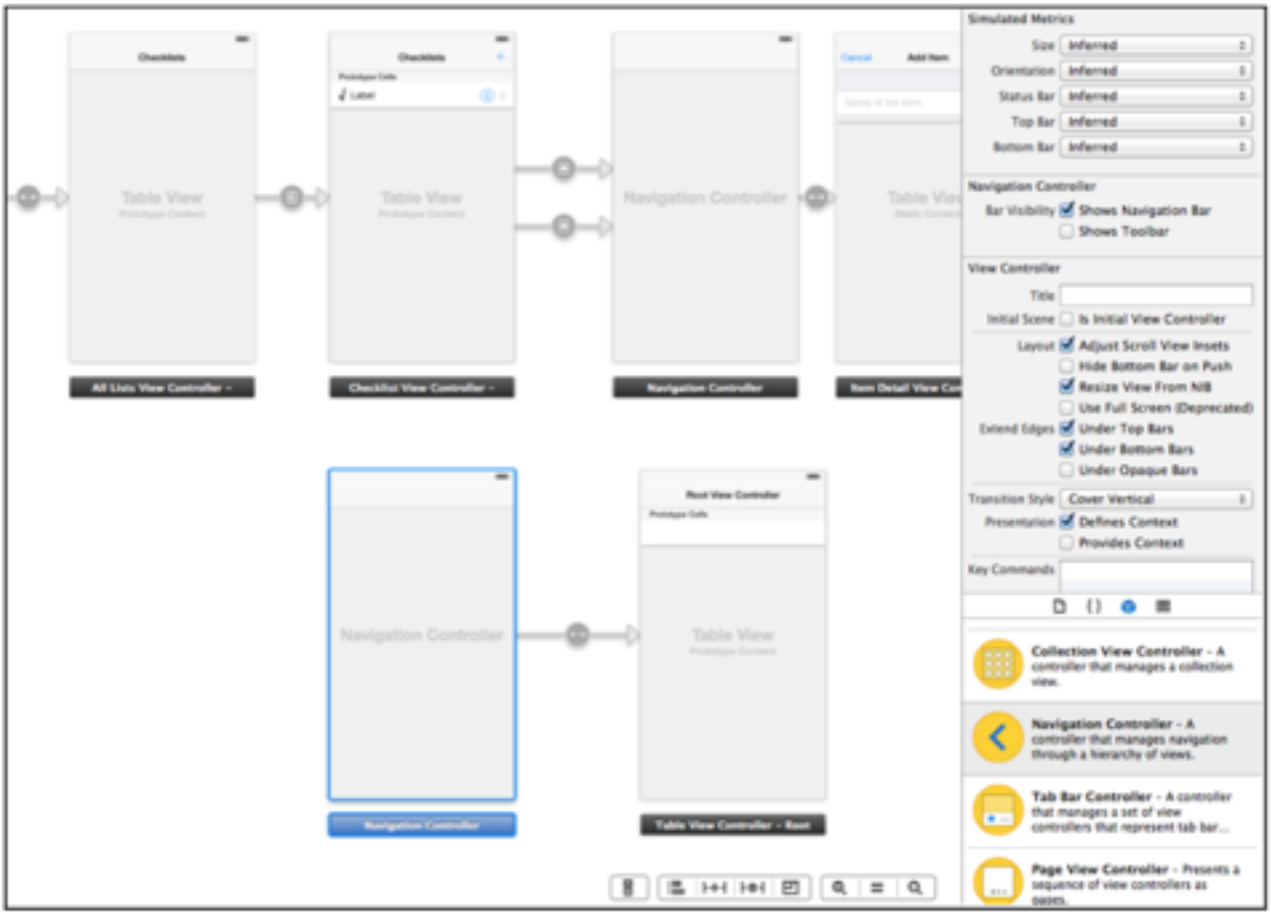
删除#pragma mark- Table view data source这一行代码之下到@end之间的所有代码。
然后添加以下方法：

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

    [self.delegate listDetailViewControllerDidCancel:self];

}

-(IBAction)done:(id)sender{

    if(self.checklistToEdit ==nil){
        Checklist *checklist = [[Checklist alloc]init];
        checklist.name = self.textField.text;

        [self.delegate listDetailViewController:self didFinishAddingChecklist:checklist];


    }else{

        self.checklistToEdit.name = self.textField.text;
        [self.delegate listDetailViewController:self didFinishEditingChecklist:self.checklistToEdit];
    }
}

-(NSIndexPath *)tableView:(UITableView *)tableView willSelectRowAtIndexPath:(NSIndexPath
*)indexPath{
```

```
    return nil;
}

-(BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range
replacementString:(NSString *)string{

    NSString *newText = [textField.text stringByReplacingCharactersInRange:range
withString:string];
    self.doneBarButton.enabled =([newText length]>0);
    return YES;
}
```

好吧，看起来内容很多，其实也是从ItemDetailViewController.m里面拷贝粘贴过来的，只不过用checklist对象取代了checklistitem对象。

现在代码部分有了，接下来让我们在Interface Builder中为新的视图控制器创建用户界面。
打开storyboard，从Object Library中拖曳出一个Navigation Controller对象到画布上，然后将它放置在其它视图控制器的下面。



删除和新的导航控制器相关联的Root View Controller。当我们添加一个新的导航控制器的时候，Interface Builder通常会自动添加一个视图控制器关联在该导航控制器上，这里显然我们不需要。

选中当前的Item Detail View Controller,按住command +D,创建一个该视图控制器的备份。然后把这个备份拖到新的导航控制器旁边。当然，你回看到上面没有cancel 和done按钮，不过这不是问题。

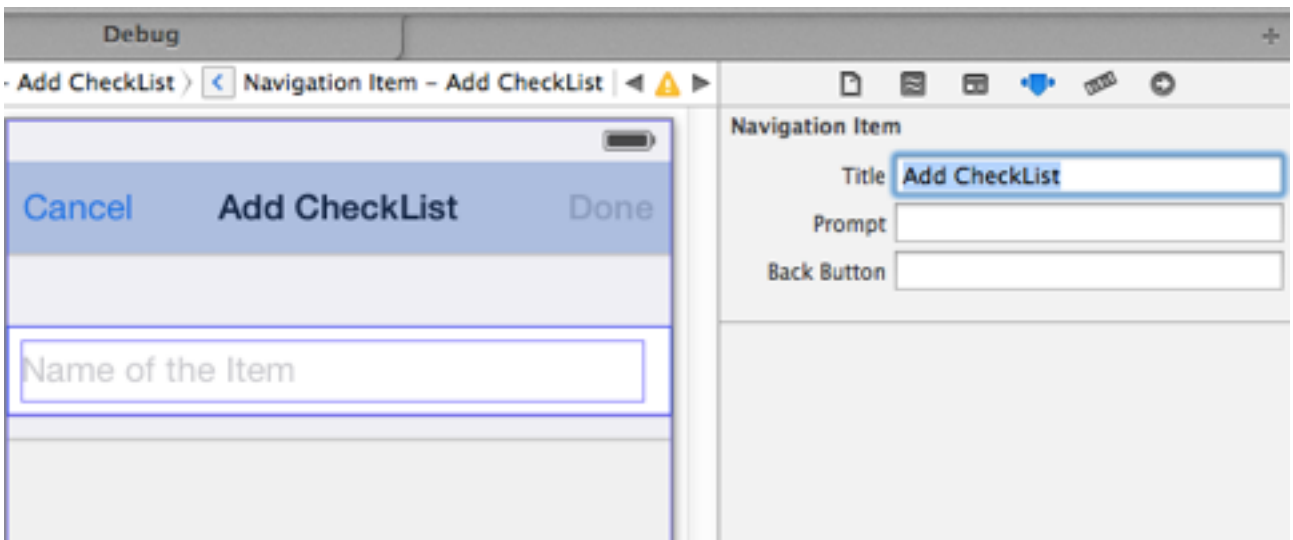
按住Ctrl键，从新的导航控制器拖出一条线到刚才的备份视图空气上，选择Relationship Segue-root view。好吧，现在导航栏上的cancel/done按钮竟然神奇的再次出现了！

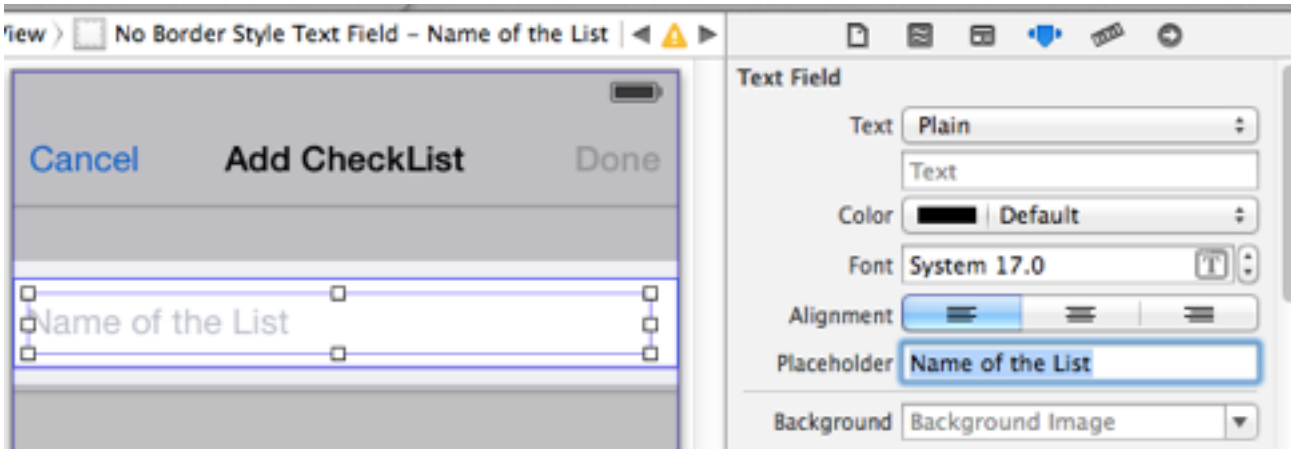
这种心情，就好比在恒大0：3输给拜仁后觉得世俱杯不关我什么事了。突然发现小罗领衔的米内罗竞技队竟然1：3输给了卡萨布兰卡，这就是命啊。
当年小罗的牛尾巴动作让我奉若神明，在很长一段时间里他就是球场上最闪耀的精灵，和齐达内一起在西甲赛场堪称绝代双骄。
而如今，恒大队竟然有机会和小罗在正式比赛中对战，实在是一种莫名的幸福。话说，这关我什么事？哎！

好了，言归正传。

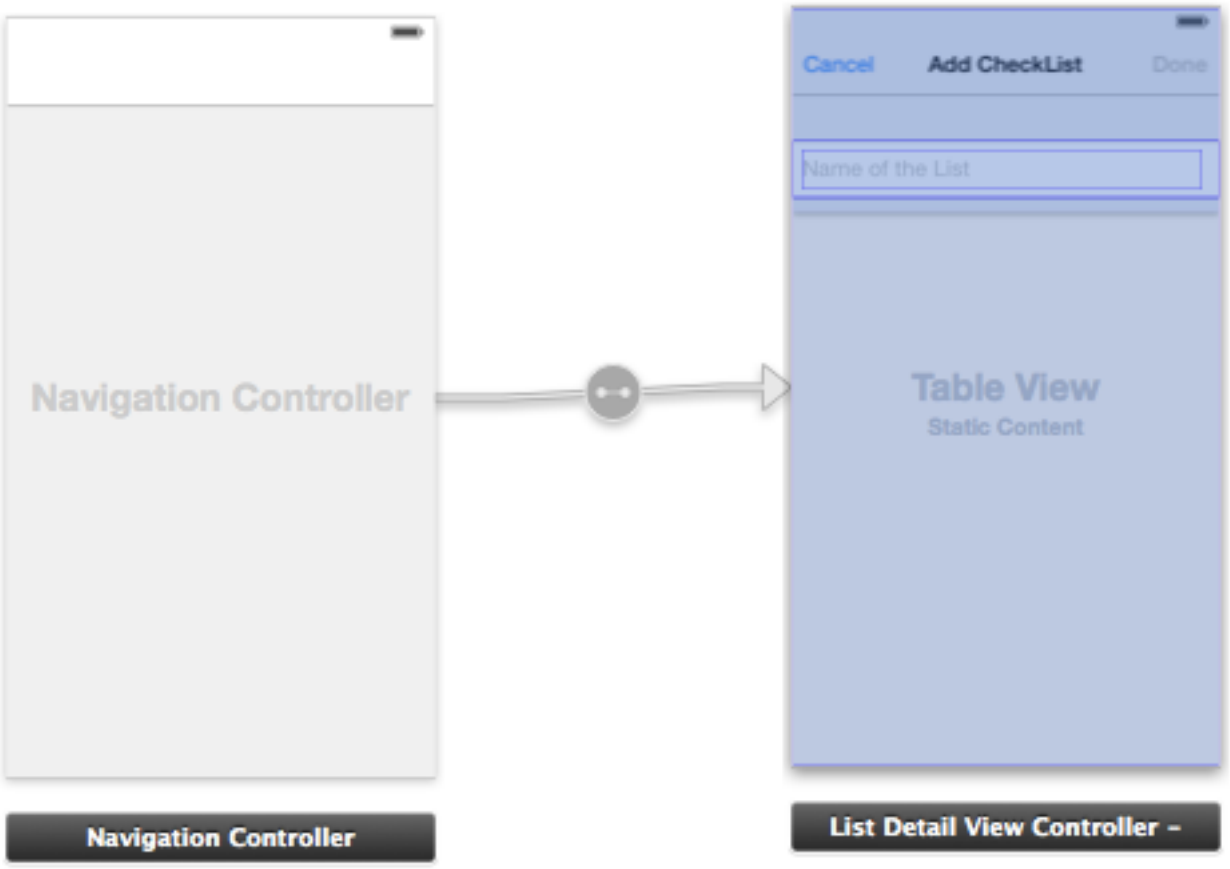
选择刚才的克隆人Item Detail View Controller，在Xcode右侧的面板中切换到Identity inspector，然后将其class设置为ListDetailViewController。

接下来更改导航栏上的标题为Add Checklist，将文本域中的占位字符设置为Name of the List。





好了，现在这个新添加的视图控制器将作为Add/Edit Checklist界面来使用了。



在storyboard中选中All Lists View Controller，从Objects Library中拖出一个Bar Button Item到导航栏上，将其更改为Add添加按钮。按住Ctrl键，从该按钮上拖出一条线到新的导航控制器，然后添加一个modal类型的segue。

选中这个新的segue，将其命名为AddChecklist
接下来将Checklist View Controller的标题从"Checklists"更改为Checklist Name

此时storyboard中的布局如下：

好了，界面部分基本上完成了。当然，我们还要让AllListsViewController成为ListDetailViewController的代理。这一步之前我们也做过，所以不必害怕。

在Xcode中切换到AllListsViewController.h，更改其中的代码如下：

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

@interface AllListsViewController : UITableViewController<ListDetailViewControllerDelegate>

@end
```

接下来切换到AllListsViewController.m，更改prepareForSegue方法的代码如下：

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{

    if([segue.identifier isEqualToString:@"ShowChecklist"]){

        ChecklistViewController *controller = segue.destinationViewController;
```

```
        controller.checklist = sender;
    }
    else if([segue.identifier isEqualToString:@"AddChecklist"]){
        UINavigationController *navigationController = segue.destinationViewController;

        ListDetailViewController *controller =
(ListDetailViewController*)navigationController.topViewController;

        controller.delegate = self;
        controller.checklistToEdit = nil;

    }
}
```

这里我们只是添加了else if的部分。当segue的identifier等于AddChecklist时，我们会进行相应的处理。

在AllListsViewController.m的最后，实现以下代理方法：

```
-(void)listDetailViewControllerDidCancel:(ListDetailViewController *)controller{

    [self dismissViewControllerAnimated:YES completion:nil];

}


-(void)listDetailViewController:(ListDetailViewController *)controller didFinishAddingChecklist:
(Checklist *)checklist{

    NSInteger newRowIndex = [_lists count];
    [_lists addObject:checklist];

    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:newRowIndex inSection:0];

    NSArray *indexPaths = @[indexPath];
    [self.tableView insertRowsAtIndexPaths:indexPaths
withRowAnimation:UITableViewRowAnimationAutomatic];
    [self dismissViewControllerAnimated:YES completion:nil];
}


-(void)listDetailViewController:(ListDetailViewController *)controller didFinishEditingChecklist:
(Checklist *)checklist{

    NSInteger index = [_lists indexOfObject:checklist];

    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:index inSection:0];
```

```
    UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
    cell.textLabel.text = checklist.name;
    [self dismissViewControllerAnimated:YES completion:nil];

}
```

别看这里有一大堆代码，其实哥都是拷贝粘贴的，然后把里面的名称替换为ListDetailViewController和Checklist对象。当用户触碰Add/Edit Checklist界面中的cancel或done按钮时，就会调用这些代理方法。

别忘了给表视图添加data source(数据源）代理方法，这样用户才可以删除checklist:

```
-(void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath{

    [_lists removeObjectAtIndex:indexPath.row];

    NSArray *indexPaths = @[indexPath];
    [tableView deleteRowsAtIndexPaths:indexPaths
withRowAnimation:UITableViewRowAnimationAutomatic];
}
```

编译运行应用，现在我们可以添加新的checklist,并删除之前的checklist，但还不能编辑现有的checklist名称。

为了打开Edit Checklist界面，用户需要触碰蓝色的accessory附属按钮。在之前的ChecklistViewController中，我们通过触发一个segue来实现。这里当然你还是可以用同样的方法，不过既然是教程，这里还是尝试一种不同的方式。在实际开发的过程中，你可以根据自己的喜好来选择不同的方式。
这里我们将从storyboard手动加载一个新的视图控制器哦。

在AllListsViewController.m中添加一个accessoryButtonTappedForRowWithIndexPath方法。该方法来自表视图的代理协议，看这个名字你差不多就知道它的作用了。

所以，不怕方法名字长，就怕你e文太乱看不懂。

```
-(void)tableView:(UITableView *)tableView accessoryButtonTappedForRowWithIndexPath:
(NSIndexPath *)indexPath{

    UINavigationController *navigationController = [self.storyboard
instantiateViewControllerWithIdentifier:@"ListNavigationController"];
    ListDetailViewController *controller =
(ListDetailViewController*)navigationController.topViewController;
    controller.delegate = self;
    Checklist *checklist = _lists[indexPath.row];
    controller.checklistToEdit = checklist;

    [self presentViewController:navigationController animated:YES completion:nil];
```

}

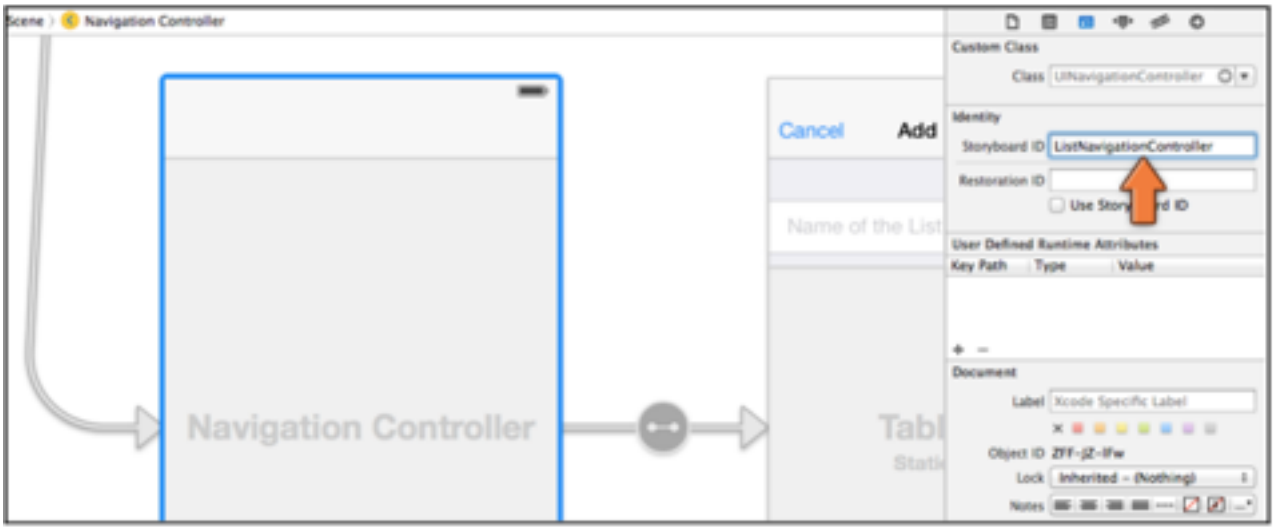
在上面的方法中，我们为Add/Edit Checklist界面创建了视图控制器对象，并将其显示（present）在屏幕上。其实这里所做的事情就是segue在幕后所进行的工作。视图控制器内置在storyboard中，因此这里需要请求storyboard对象来加载该视图控制器。

那么如何获得storyboard对象呢？实际上每个视图控制器都有一个self.storyboard属性，指向加载该视图控制器的storyboard。我们可以利用该属性让storyboard完成很多工作，比如初始化其它的视图控制器。

这里我们调用了instantiateViewControllerWithIdentifier方法，并使用了一个identifier字符串参数@"ListNavigationController"。在这里，它指的就是包含了ListDetailViewController的导航控制器。我们当然也可以直接初始化ListDetailViewController，但因为它被内置在一个导航控制器中。如果直接初始化，就没法看到标题栏或者cancel和done按钮了。

显然这里我们需要手动设置导航控制器的identifier，否则storyboard根本不知道如何找到它。

打开storyboard，选择新添加的导航控制器，切换到Identity inspector，然后在Storyboard ID部分输入ListNavigationController。
注意不要犯低级错误（大小写或者单词拼写错误）。



好了，这样问题应该就解决了。

编译运行应用，然后触碰细节显示按钮看看效果。

小练习：
将ListNavigationController设置在List Detail View Controller上面，看看会发生什么。

好了，今天的学习中其实更多是在复习之前的内容，很多代码都是重用的，只做了小小的修改。

该是福利时刻了。

首先期待小罗能够重新找回自己，有机会参加明年的巴西世界杯。因为有小罗在，我希望米内罗大胜恒大，小罗重施神技，给他重回国家队增加一点筹码，哪怕是微小的筹码。而对于孔卡来说，也希望他可以来个完美收官战，进一球吧。
当年的小罗- 五英战吕布

最后还是福利，超级养眼的车展外国mm

原图更震撼，来自色影无忌：http://forum.xitek.com/forum-viewthread-tid-1240093-highlight-%CA%B7%C9%CF%D7%EE%C7%E5%CE%FA%B5%C4.html