从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter24

开发环境：
Xcode 5 +iOS 7

欢迎继续我们的学习。

在上一章的内容中，我们成功的在表视图中每个checklist的旁边添加了一个图标。
接下来我们将更改checklist的init方法，从而在默认情况下为每个checklist对象提供一个名为No Icon的图标。

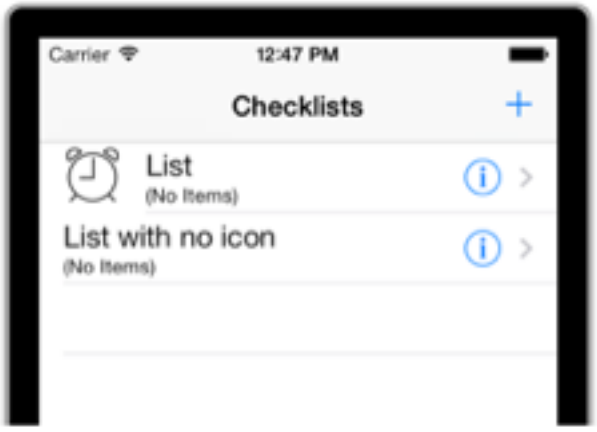
打开Xcode,切换到Checklist.m，然后更改init方法如下：

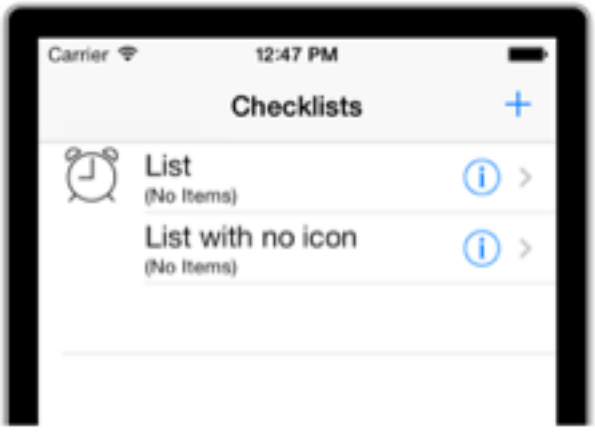
```
-(id)init{

    if((self = [super init])){
        self.items = [[NSMutableArray alloc]initWithCapacity:20];
        self.iconName = @"No Icon";
    }
    return self;
}
```

这里的No Icon图像是一个完全透明的PNG图片，它的尺寸大小和其它图标完全相同。使用一个透明图片可以让所有的checklist对齐，即便在默认没有icon的情况下。

ß如果我们把self.iconName设置成nil，那么table view cell中的image view不会有任何内容。此时文本区域会对齐屏幕的左侧。但在这种情况下就会看起来很糟糕：





接下来让我们创建icon picker界面。首先需要在项目中添加一个新文件，设置subclass of UITableViewController，将其命名为IconPickerViewController。

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

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

@class IconPickerViewController;

@protocol IconPickerViewControllerDelegate <NSObject>

-(void)iconPicker:(IconPickerViewController*)picker didPickIcon:(NSString*)iconName;

@end

@interface IconPickerViewController : UITableViewController

@property(nonatomic,weak)id <IconPickerViewControllerDelegate> delegate;

@end
```

接下来切换到IconPickerViewController.m，然后添加一个实例变量来保存图标数组：

```
@implementation IconPickerViewController{
   NSArray *_icons;
}
```

接着更改viewDidLoad方法如下：

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


   _icons = @[
            @"No Icon",
            @"Appointments",
            @"Birthdays",
            @"Chores",
            @"Drinks",
            @"Folder",
            @"Groceries",
            @"Inbox",
            @"Photos",
            @"Trips"
            ];
}
```

实例变量_icons是个NSArray类型的数组，其中包含了一个图标名称的列表。这些字符串既是你将在界面上显示的文本，同时也是asset catalog中的PNG图片名称。_icons数组其实就是这个表视图的数据模型。注意到它是一个non-mutalbe NSArray（而不是一个可变的NSMutableArray)，因为用户不能增加或删除图标。

因为这个新的视图控制器是一个UITableViewController,所以我们需要为表视图实现数据源方法。

从IconPickerViewController.m中删除原有的数据源协议实现的方法，并用以下方法来替代：

```
#pragma mark - Table view data source

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return [_icons count];

}

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

    UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:@"IconCell"];
    NSString *icon = _icons[indexPath.row];

    cell.textLabel.text = icon;

    cell.imageView.image = [UIImage imageNamed:icon];
    return cell;

}
```

以上的方法我们应该已经相当熟悉了。这里我们获取了一个table view cell，然后给它提供了文本和图片。
我们很快就将在storyboard中设计这个cell，它将是一个prototype cell,其样式是Default（或者用Interface Builder的说法叫Basic）。此类样式的cell默认情况下含有一个文本标签和一个image view，使用起来非常方便。

继续之前，先来点理论知识充电。

convenience constructor(关于便利的构造函数)

之前我们在创建新对象的时候通常使用alloc和init的组合：

```
Checklist *checklist = [[Checklist alloc]init];
```

但这里我们使用了一个一种新的方法来创建一个UIImage对象：

```
cell.imageView.image = [UIImage imageNamed:icon];
```

这里没有用到alloc或init。你可能会想，通过这种方式来创建新对象真的靠谱吗？其实这种创建新对象的方式又被成为convenience constructor。

如果用之前那种形式，这行代码应该写成如下的格式：

image =[[UIImage alloc]initWithContentsOfFile:…];

其实这两种形式的作用和目的都是相似的，它们同样分配内存空间，并初始化了一个新的UIImage对象。

下面是另一个类似的例子：

NSString *string = [NSString stringWithFormat:@"I'm happy"];

它同样可以写成这种形式：

NSString *string = [NSString alloc]initWithFormat:@"I'm happy"];

那么为了达到同一个目的，为什么在Objective-C中提供了两种形式呢？好吧，其实是为了方便。通过使用imageNamed和stringWithFormat方法可以帮你省掉输入alloc的时间。当然，这样做还有一个历史原因，不过那要涉及到iOS 的历史变迁，以及Automatic Reference Counting(ARC)的革命。

好了，理论部分结束。

在Xcode中打开storyboard，然后从Object Library中拖出一个新的Table View Controller，并将它放在List Detail View Controller的旁边。

然后在Xcode右侧面板中切换到Identity inspector，然后将新的table view controller 的class 属性更改为IconPickerViewController。

接下来选中prototype cell,然后将其Style属性设置为Basic,将Identifier设置为IconCell。

这样我们就基本完成了icon picker的设计。现在我们需要选择从哪里调用它。为此，我们需要在Add/Edit Checklist界面中添加一个新的row。

1.在storyboard中选中List Detail View Controller，然后在table view中添加一个新的section。首先选中table view，然后在Xcode右侧面板中切换到Attributes inspector，然后将Sections部分从1更改为2。这样就完成了复制一个section的工作。

2.删掉cell中的Text Field，因为我们不需要。

3.然后拖出一个Label到cell中，将其命名为Icon。

4.选中cell,将Accessory属性更改为Disclosure Indicator。

5.在cell中的右侧添加一个Image View,设置其大小为36*36 points

6.打开Assistant Editor,然后为这个新添加的image view 关联一个outlet属性到ListDetailViewController.h，并将其命名为iconImageView。

好了，现在我们已经完成了两个界面的设计工作，接下来可以使用segue来连接它们。

在storyboard中按住ctrl键，从Icon所在的table view cell拖出一条线到Icon Picker View Controller，并添加一个segue(选择Selection Segue-push），然后将其identifier属性设置为PickIcon。

好了，因为有segue的友情协助，现在新的视图控制器有了一个导航栏。双击导航栏，并将其title更改为Choose Icon。

此时storyboard的这部分将会如下所示：

现在回到代码时段~

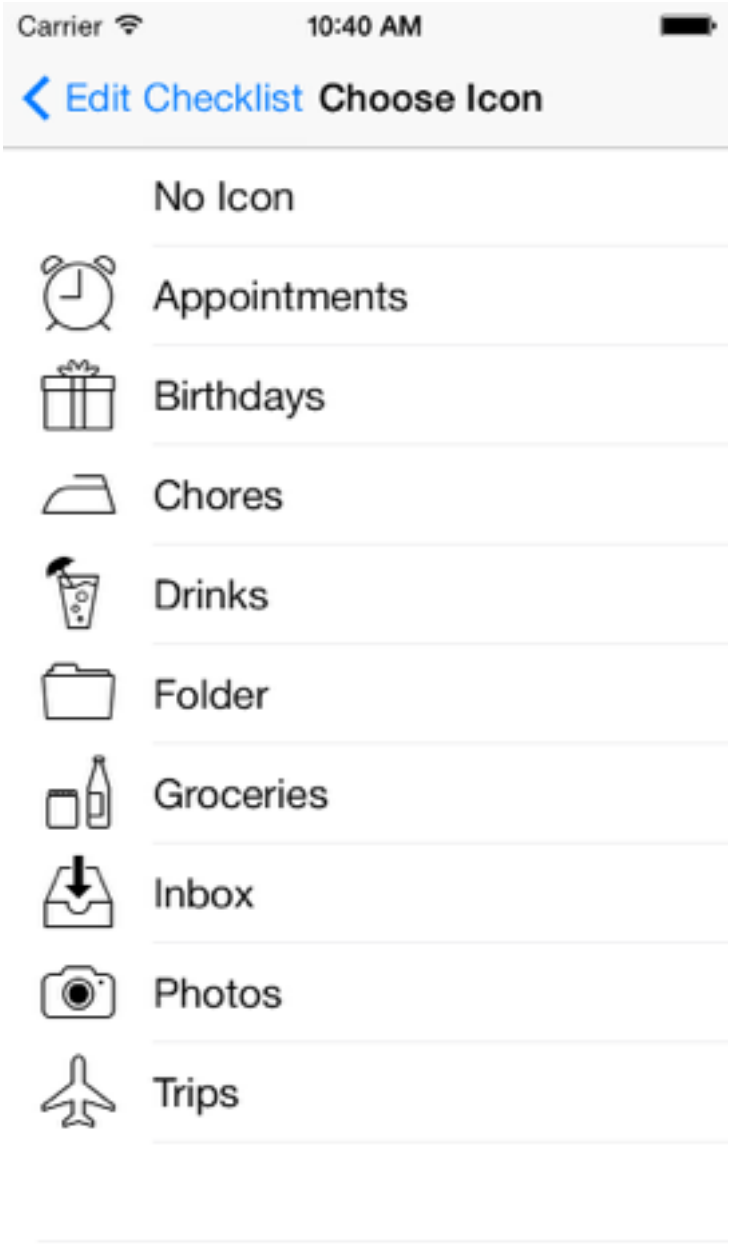
切换到ListDetailViewController.m，更改willSelectRorAtIndexPath方法的代码如下：

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

    if(indexPath.section ==1){
        return indexPath;
    }else{
  return nil;
    }
}
```

更改这个方法的内容十分有必要，否则用户在触碰cell时讲无法触发segue。之前该方法始终返回nil,也就意味着触碰行是无效的。但现在不同，我们允许用户触碰Icon所在的cell，并在该方法中返回cell的index-path。因为Icon cell是第二个section中的唯一行，我们只需要检查indexPath.section就可以了。不过用户仍然不能选择文本字段部分（section为0）。

好了，此时可以编译运行应用，看看Add/Edit Checklist界面是否出现了一个Icon行。试着触碰这一行，看能否打开Choose Icon界面。此时icon picker界面应该会显示一个图标的列表。
我们可以触碰返回按钮回到之前的界面。



提醒：

如果触碰Icon行不能切换到Icon picker界面，那么你需要检查table view的Selection属性是否被设置为No Selection。这里它应该被设置为Single Selection。

虽然我们可以显示图标列表了，但选中一个图标不会有任何反应我们需要通过代理协议将icon picker界面和Add/Edit Checklist界面关联起来。

在Xcode中切换到ListDetailViewController.h，在文件顶部导入一个头文件，然后添加一个协议遵循声明（黄色高亮部分代码）：

```
#import “IconPickerViewController.h"

@interface ListDetailViewController : UITableViewController
<UITextFieldDelegate,IconPickerViewControllerDelegate>
```

接着在ListDetailViewControllerm中添加一个实例变量声明：

```
@implementation ListDetailViewController{

    NSString *_iconName;
}
```

我们使用该变量来保持所选中的图表名称。即便checklist对象已经有一个iconName属性了，但我们仍然无法保存checklist对象的所选中图标。原因很简单，因为首先我们不一定有一个checklist对象，比如当用户添加新的checklist时。因此我们需要将图标名称保存在一个临时变量中，然后在需要的时候将其赋予checklist的iconName属性。

当然，我们需要用一些有意义的东西来初始化iconName变量。这里我们用folder图标来初始化。为此，我们需要添加initWithCoder方法，因为当使用storyboard来加载一个视图控制器的时候，通常会选择这个初始化方法。

在Xcode中切换到ListDetailViewController.m，然后添加一个新的initWithCoder方法：

```
-(id)initWithCoder:(NSCoder *)aDecoder{

    if((self = [super initWithCoder:aDecoder])){

        _iconName = @"Folder";
    }
    return self;
}
```

这样我们就将_iconName变量的值设置为@“Folder”。当然，仅对新的checklist才有必要这么做，此时默认情况下会获得一个Folder图标。

此时我们可以大胆删除之前的initWithStyle方法了，因为它已经没有存在的必要了。

接下来更新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;
    _iconName = self.checklistToEdit.iconName;
  }
    self.iconImageView.image = [UIImage imageNamed:_iconName];
}
```

这里我们添加了两行新代码：如果checklistToEdit的属性不是nil,那么就将checklist对象的图标名称赋予_iconName这个实例变量。我们可以将图标放入一个新的UIImage对象，然后当Icon行显示时将其设置为iconImageView中的图片。

此前我们已经将Add/Edit Checklist界面和IconPickerViewController界面使用一个名为PickIcon的push segue关联在一起了。为了通知IconPickerViewController该界面将成为它的代理对象，我们需要实现prepareForSegue方法。

在ListDetailViewController.m的底部添加以下方法：

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


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

        IconPickerViewController *controller = segue.destinationViewController;

        controller.delegate = self;
    }
}
```

上面方法中的代码你也看过很多次了，这里不再详细解释。如果仍然看不懂，可以回过头看看之前教程中所涉及到的部分。

最后，我们还需要实现代理协议的回调方法，让程序记住所选中图标的名称。

在ListDetailViewController.m的底部添加以下方法：

```
-(void)iconPicker:(IconPickerViewController *)picker didPickIcon:(NSString *)iconName{


    _iconName = iconName;

    self.iconImageView.image = [UIImage imageNamed:_iconName];

    [self.navigationController popViewControllerAnimated:YES];
}
```

通过上面的代理协议方法，我们将所选中的图标名称保存到_iconName变量中，并使用新的图片更新了image view视图。

之所以这里没有调用dismissViewController，而是popViewControllerAnimated，是因为Icon Picker现在位于导航视图的堆栈上（segue类型是push而不是modal）。

接下来更改done这个动作方法，当用户关闭当前界面时，让它将所选中的图标名称放到checklist对象的iconName属性中。

```
- (IBAction)done
{
  if (self.checklistToEdit == nil) {
    Checklist *checklist = [[Checklist alloc] init];
    checklist.name = self.textField.text;
     checklist.iconName = _iconName;

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

  } else {
    self.checklistToEdit.name = self.textField.text;
     self.checklistToEdit.iconName = _iconName;
    [self.delegate listDetailViewController:self didFinishEditingChecklist:self.checklistToEdit];
  }
}
```

最后，我们必须更改IconPickerViewController中的方法，确保当用户触碰某一行时会调用代理方法.

切换到IconPickerViewController.m，然后在底部添加以下方法：

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

    NSString *iconName = _icons[indexPath.row];
    [self.delegate iconPicker:self didPickIcon:iconName];
}
```

好了，一番辛苦之后，终于可以搞定了！

现在我们设置Checklist对象的图标了。试试看！

在这一章的内容中，我们添加了一个新的视图控制器对象，在storyboard中设计了它的用户界面，使用segue和delegate将其关联到Add/Edit Checklist界面。
实际上以上就是我们在应用中创建任何新界面所需要的基本步骤。

休息一下，准备迎接最后的三章内容吧。当你完全搞定这最后三章后，就可以自豪的宣布自己在iOS开发上算是基本入门了！

福利时间到，美图送上

宜家的可爱狗狗

清新自然的妹子