从零开始学iOS7开发系列3-我的地盘我做主-Cha18

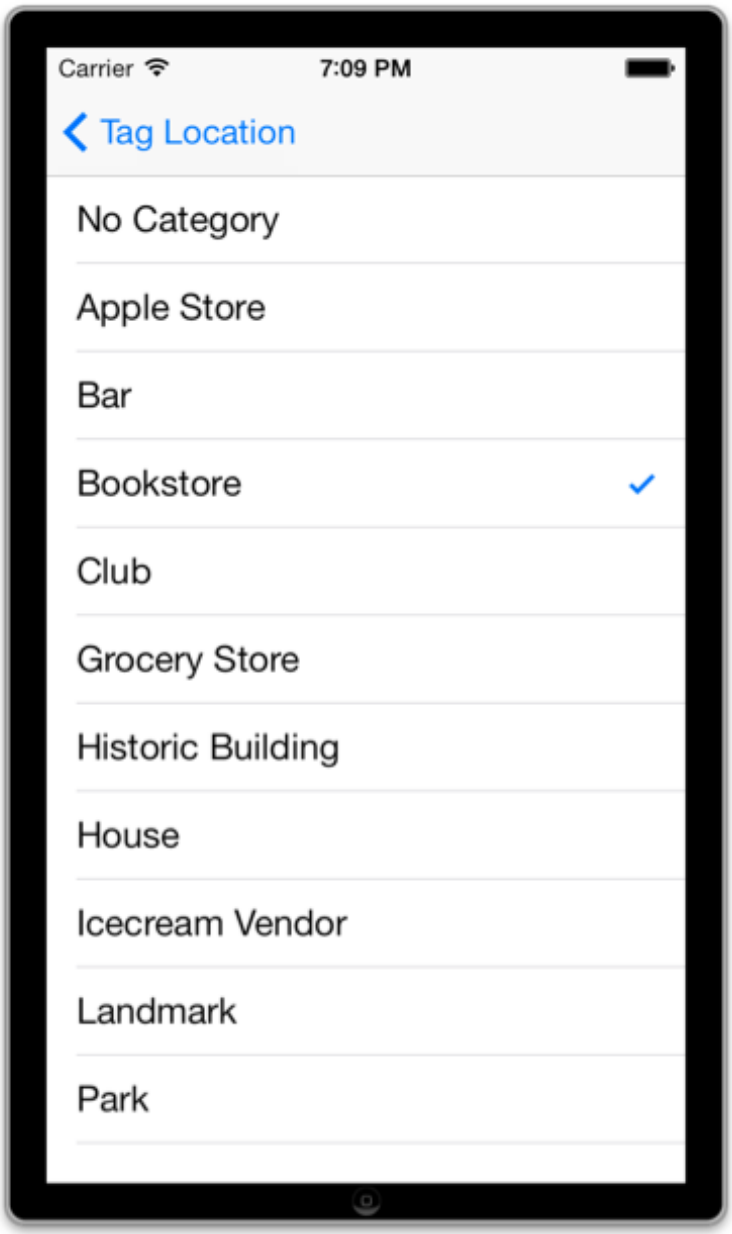
说明:

本系列文章的原文及示例代码来自raywenderlich store中的iOS Apprentice 系列3教程，经过翻译和改编。

欢迎继续我们的学习。

接下来我们将实现界面中的另外一个控件-类别选择器

当用户触碰Category这个cell的时候，应用会显示一个类别名称的列表以供选择：



当然，我们很容易看出来这是一个新的界面，因此需要一个新的视图控制器。它的工作方式和上一系列教程中的icon picker（图标选择器）很类似。因此这里我只会提供相关的源代码，并建立控件和代码之间的关联。

在Xcode中创建一个新的table view controller子类，将其命名为CategoryPickerViewController。

使用以下代码替换CategoryPickerViewController.h中的内容：

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

@interface CategoryPickerViewController : UITableViewController

@property(nonatomic,strong) NSString *selectedCategoryName;

@end
```

这里定义了一个表视图控制器，它的作用是显示category类别名称的列表。我们可以使用selectedCategoryName属性让某个类别作为初始选中的类别。对于选中的类别会在名称旁边放上一个勾选标志。

接下来使用下面的代码替代CategoryPickerViewController.m中的内容：

```
#import "CategoryPickerViewController.h"


@implementation CategoryPickerViewController{

    NSArray *_categories;
    NSIndexPath *_selectedIndexPath;

}



- (void)viewDidLoad
{
    [super viewDidLoad];

    // Uncomment the following line to preserve selection between presentations.
    // self.clearsSelectionOnViewWillAppear = NO;

    // Uncomment the following line to display an Edit button in the navigation bar for this view
controller.
    // self.navigationItem.rightBarButtonItem = self.editButtonItem;

    _categories = @[
                    @"No Category",
                    @"Apple Store",
                    @"Bar",
                    @"BookStore",
                    @"Club",
                    @"Grocery Store",
```

```
                @"Historic Building",
                @"House",
                @"Icecream Vendor",
                @"Landmark",
                @"Park"
                ];
}

#pragma mark - UITableViewDataSource

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{

    return [_categories count];
}

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

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    NSString *categoryName = _categories[indexPath.row];

    cell.textLabel.text = categoryName;

    if([categoryName isEqualToString:self.selectedCategoryName]){
        cell.accessoryType = UITableViewCellAccessoryCheckmark;

    }else{
        cell.accessoryType = UITableViewCellAccessoryNone;
    }

    return cell;
}

#pragma mark - UITableViewDelegate

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

    if(indexPath.row != _selectedIndexPath.row){
        UITableViewCell *newCell = [tableView cellForRowAtIndexPath:indexPath];
        newCell.accessoryType = UITableViewCellAccessoryCheckmark;

        UITableViewCell *oldCell = [tableView cellForRowAtIndexPath:_selectedIndexPath];
        oldCell.accessoryType = UITableViewCellAccessoryNone;
        _selectedIndexPath = indexPath;
    }
}


@end
```

实际上这部分也没有什么新的内容，不过还是要稍微解释下。该界面是一个表视图控制器，因此它又一个数据源和delegate代理。数据源（data source)将从_categories实例变量中获取行，这个实例变量是一个NSArray对象，其内容将在viewDidLoad方法中。

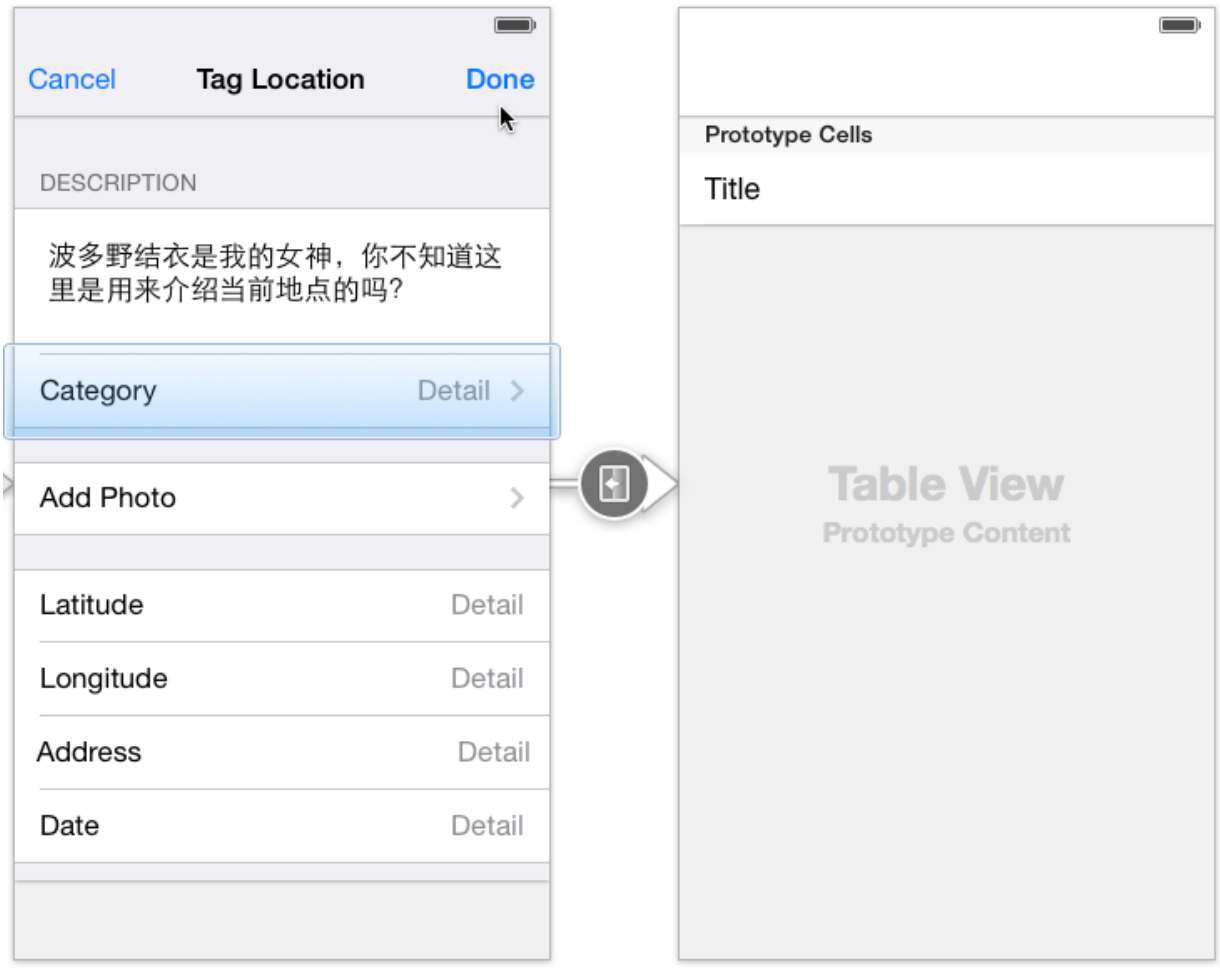
唯一特殊的一点是_selectedIndexPath这个实例变量。当界面打开时会在当前所选中的类别旁显示一个勾选标志。当用户触碰另外一个行时，我们希望可以取消之前所选中cell的勾选标志，转而让一个新的cell显示。为了实现这一点，我们需要知道哪一行是当前所选中的。

调用该界面的对象会赋予selectedCategoryName属性一个值，但它是一个字符串，而不是行编号。因此，在cellForRowAtIndexPath方法中，我们比较行中的标签内容和self.selectedCategoryName属性值。如果二者相匹配，那么就将行的index-path保存在_selectedIndexPath变量中。这样我们就可以随后在didSelectRowAtIndexPath方法中删除当前行的勾选标志。虽然这样一个小功能看起来浪费了不少精力，不过对于一款优秀的应用来说，细节决定成败。

打开storyboard，然后拖出一个新的Table View Controller到画布上。在Identity inspector面板中设置其Class为CategoryPickerViewController。
选中这个新table view controller里面的prototype cell，将其Style设置为Basic，将identifier命名为Cell。
按住Ctrl键，从Location Details View Controller里面的Category cell拖曳一条线到新的view controller,添加一个Selection Segue- push segue。

选中这个segue，将其identifier命名为PickCategory。

现在Category Picker View Controller在顶部有了一个导航栏。我们可以更改其title为'Choose Category',不过苹果官方建议对于在iOS7开发中对于有明显用途的视图控制器，没有必要提供title。这样可以让导航栏看起来更清爽。
好了，storyboard部分基本上就搞定了，剩下的事情就是如何处理这个segue。



在LocationDetailsViewController.m的顶部添加一行代码：

```
#import "CategoryPickerViewController.h"
```

接下来我们需要添加一个新的名为_categoryName的实例变量。我们可以用它来临时保存所选中的category类别。

此时实例变量声明部分的代码如下：

```
@implementation LocationDetailsViewController

{
    NSString *_descriptionText;
    NSString *_categoryName;

}
```

更改initWithCoder:方法的代码为：

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

    if((self = [super initWithCoder:aDecoder])){
        _descriptionText = @"";
        _categoryName = @"No Category";
    }
    return self;
}
```

初始情况下我们将category的名称设置为”No Category”。它同时也是类别选择器列表中最上面的那个选项。

更改viewDidLoad方法的代码如下：

```
-(void)viewDidLoad{

    [super viewDidLoad];

    self.descriptionTextView.text = _descriptionText;
    self.categoryLabel.text = _categoryName;

    self.latitudeLabel.text = [NSString stringWithFormat:@"%.8f",self.coordinate.latitude];
    self.longitudeLabel.text = [NSString stringWithFormat:@"%.8f",self.coordinate.longitude];

    if(self.placemark !=nil){

        self.addressLabel.text = [self stringFromPlacemark:self.placemark];
    }else{
        self.addressLabel.text = @"No Address Found";
    }
    self.dateLabel.text = [self formatDate:[NSDate date]];


}
```

最后，在cancel:方法的后面添加一个prepareForSegue方法：

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

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

        CategoryPickerViewController *controller = segue.destinationViewController;
        controller.selectedCategoryName = _categoryName;

    }

}
```

这个方法的内容就是设置类别选择器的slectedCategoryName属性值。

好了，现在这个应用就可以选择类别了。

编译运行应用，然后看看类别选择器的功能是否正常。

好吧，现在的类别选择器其实是个花架子。我们可以选择某个类别，但界面不会自动关闭。同时如果我们触碰Back返回按钮，那么所选择的类别也并不会显示在界面中。

思考：看看怎么解决这个问题

答案：CategoryPickerViewController当前不能反向和LocationDetailsViewController进行沟通。

好吧，可能此时你立马会想到一种方法，那就是用代理协议的方式。
代理协议的确是一种非常不错的解决方法，不过这里我们将介绍一种新的storyboard特性，可以同样完美解决这个问题：unwind segue。它在哪里呢？其实就是storyboard立马的绿色”Exit”图标。

常规的segue用于打开一个新的界面，而unwind segue的作用则是关闭一个当前的活跃界面。听起来很简单，不过用起来就没那么简单了~

绿色的Exit图标看起来似乎毫无用处。按住Ctrl键，从prototype cell拖曳一条线到Exit图标，这样并不会帮你创建一个关联。我们需要首先在代码中添加一种特殊的动作方法。

切换到LocationDetailsViewController.m，添加以下方法：

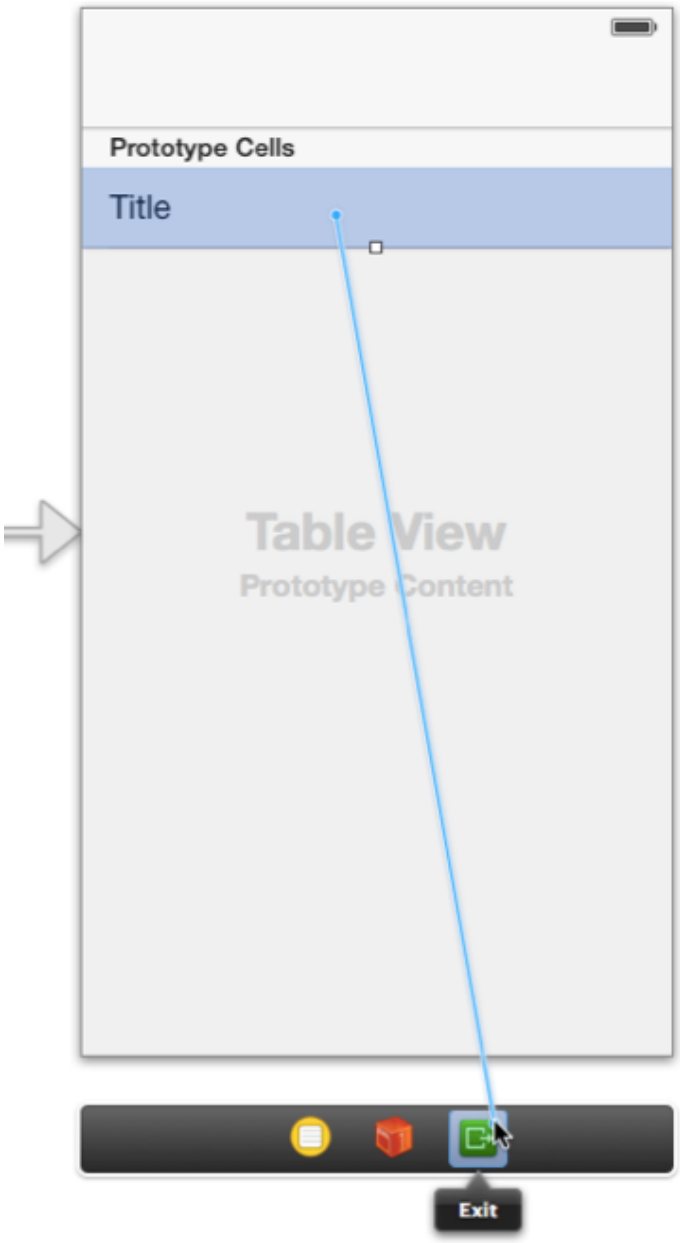
```
-(IBAction)categoryPickerDidPickCategory:(UIStoryboardSegue *)segue{

    CategoryPickerViewController *viewController = segue.sourceViewController;
```

```
    _categoryName = viewController.selectedCategoryName;
    self.categoryLabel.text = _categoryName;
}
```

这是一个动作方法，因为它的返回类型是IBAction。但是和常规动作方法不同的是它的参数，一个UIStoryboardSegue对象。通常情况下动作方法的参数是触发该动作的控件，比如一个按钮。但是为了创建一个unwind segue，我们需要定义一个动作方法，使用UIStoryboardSegue作为参数。

这个方法的作用很简单。我们会查看发送该segue的视图控制器，这里当然就是CategoryPickerViewController，然后读取它的selectedCategoryName属性。假定该属性包含了用户所选择的类别。

打开storyboard，按住Ctrl键，从prototype cell拖曳一条线到绿色的Exit按钮。此时我们就可以创建一个关联了：



从弹出菜单中选择Selection Segue-categoryPickerDidPickCategory:，也就是我们刚刚添加的unwind动作方法。

此时当我们触碰类别选择器界面中的某个cell时，当前界面会关闭，这个新的方法会被调用。

试一下看看，是不是很神奇？！不幸的是，我们所选中的类别仍然被无视。这是因为categoryPickerDidPickCategory:方法会查看selectedCategoryName属性，但该属性还没有在任何地方被赋值。

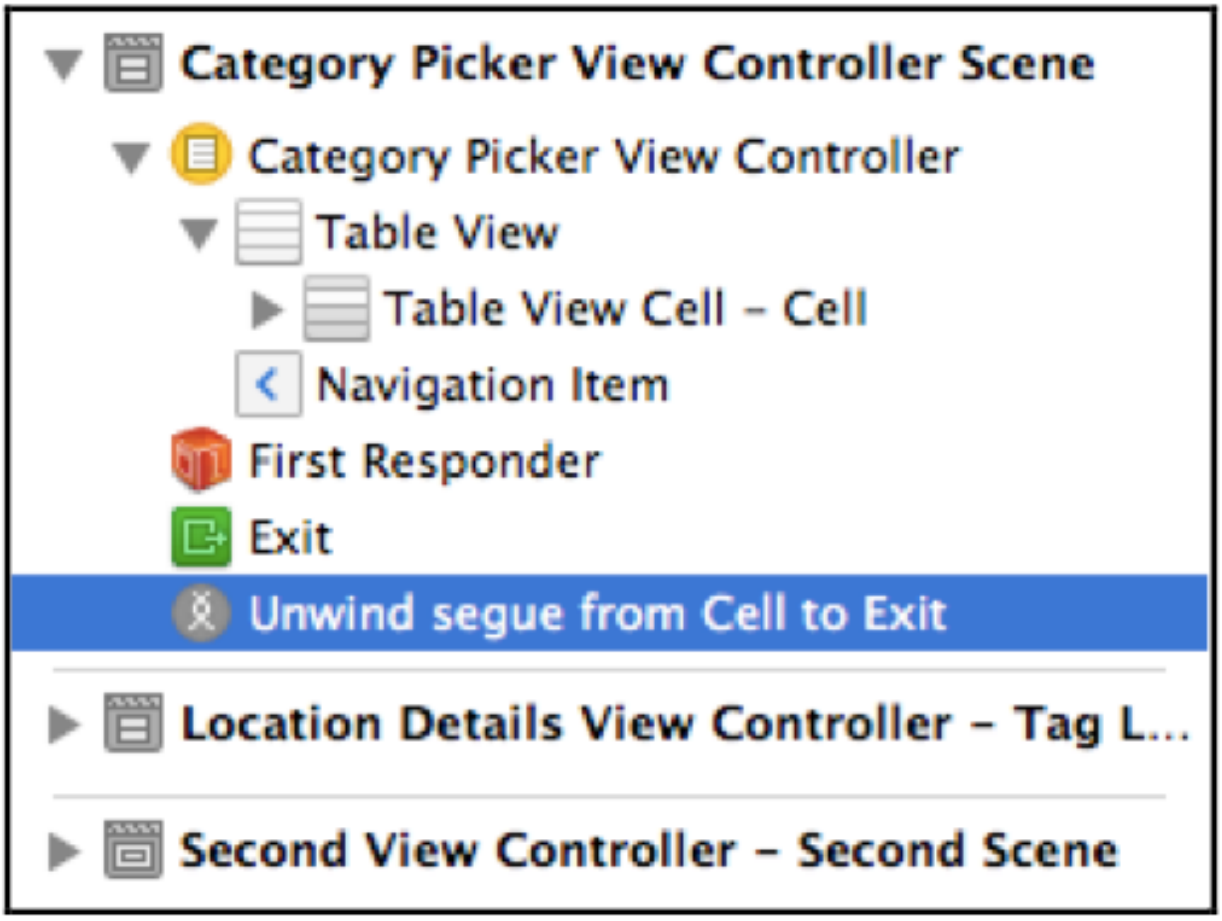
我们需要建立一种机制，当触发unwind segue时根据所选择的行获得selectedCategoryName属性。那么这种机制的名称是什么？当然是prepareForSegue。这个方法是可以双向使用的。

在CategoryPickerViewController.m中添加下面的方法：

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

    if([segue.identifier isEqualToString:@"PickedCategory"]){
        UITableViewCell *cell = sender;
        NSIndexPath *indexPath = [self.tableView indexPathForCell:cell];
        self.selectedCategoryName = _categories[indexPath.row];
    }
}
```

在以上方法中，我们获得了所选择的index-path，然后把所对应的类别名称保存到selectedCategoryName属性中。该逻辑假定segue的名称是”PickedCategory”，因此我们需要为刚才的unwind segue设置一个identifier。

不幸的是，对于storyboard中的unwind segue并没有视觉化的标志符号。为了选中一个unwind segue，我们需要使用左侧的树状面板：



在这里选中unwind segue，然后在右侧切换到Attributes inspector，将identifier的名称更改为PickedCategory。

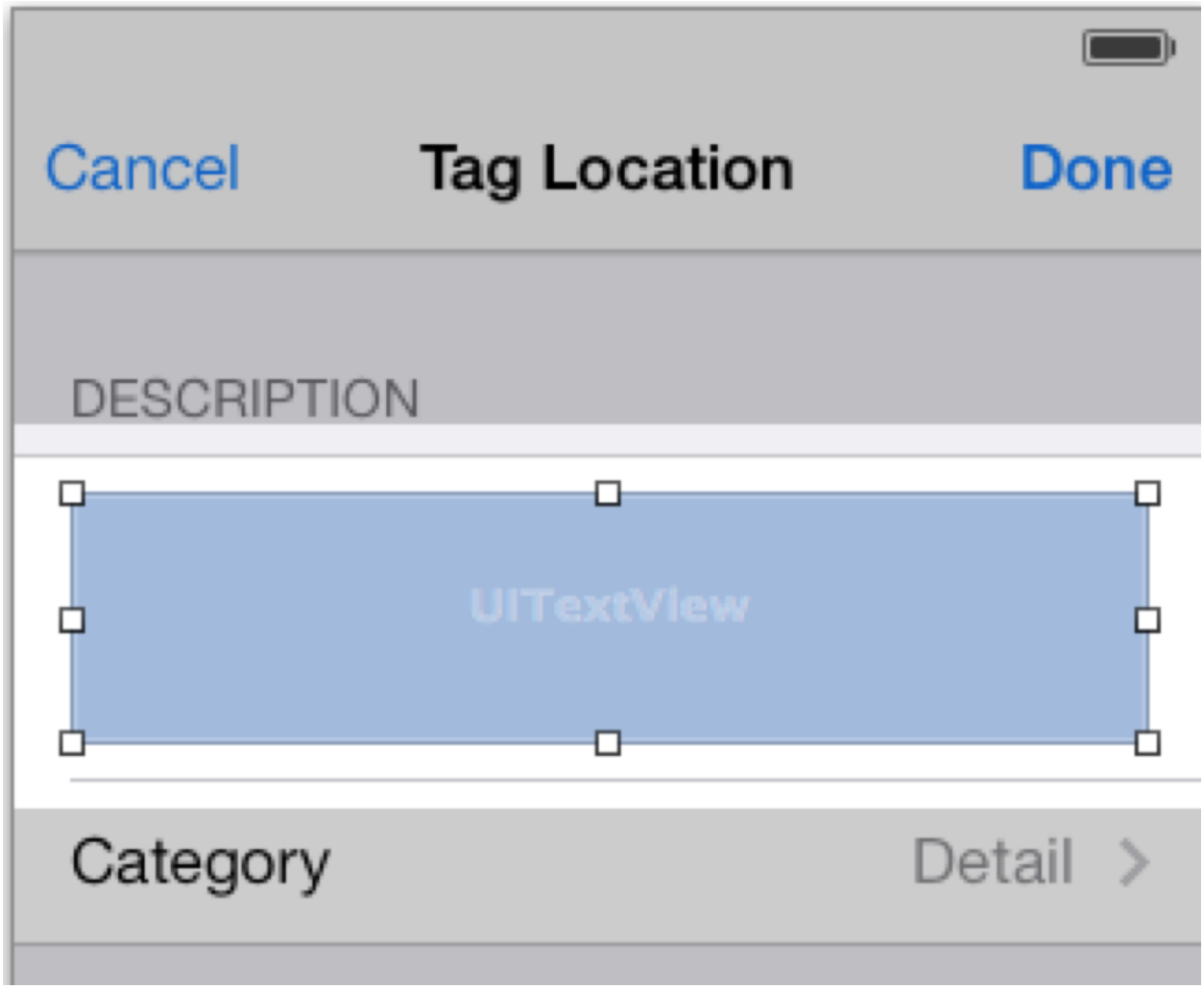
编译运行应用。现在类别选择器应该可以正常工作了。只要我们选中某个类别，界面就会自动关闭，同时在Tag Location界面中会显示所选中的类别名称。

相比使用delegate protocol（代理协议）来说，unwind segue简单易用，特别对于这种简单的选择界面非常适用。

在结束本课内容之前，先让我们看看接下来要完成的下一步工作-改善用户体验

现在Tag Location界面的基本功能算是实现了，但还需要进行一些优化。通过对一些细节进行调整，可以让用户更加喜欢你的应用，同时也让你的应用脱颖而出。

先看看Description text view的cell设计：



在text view和cell边缘之间有10-point的间距，不过因为cell的背景和text view的背景都是白色的，因此用户不知道text view从哪里开始。

很有可能用户触碰的位置在cell之中但却在text view之外。当我们想要在里面输入一些内容的时候这一点让人很恼火。用户以为已经点到text view里面了，但虚拟键盘却没有出现。没有任何反馈告诉用户他是否已经点进了text view，因此用户甚至会以为应用崩溃了。

因此我们需要对此做一些改进。当用户触碰第一个cell的任何位置时，应用应自动激活text view，即便触碰的对象不是text view本身。

在LocationDetailsViewController.m中的UITableViewDelegate部分添加两个新方法：

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

    if(indexPath.section ==0 || indexPath.section == 1){
        return indexPath;
    }else{
        return nil;
    }
}

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

    if(indexPath.section ==0 && indexPath.row ==0){
        [self.descriptionTextView becomeFirstResponder];
    }
}
```

willSelectRowAtIndexPath方法限制了用户的触碰操作只能是第一个或第二个section。第三个section中都是只读的标签，因此不允许触碰操作。
didSelectRowAtIndexPath方法用来处理对行的真实触碰操作。我们无需对Category或Add Photo行的触碰进行处理，因为这些cell和segue关联在一起。但是当用户触碰了第一个section的第一行时（也就是描述文本区），就会让输入的焦点自动切换到text view。

编译运行应用试一下，试着在第一个cell的边缘点点看。现在妈妈不再担心虚拟键盘不自动出现了~

如果可以让用户界面更加友好，我们都值得投入一定的努力。

说到text view，一旦我们激活了它，就没法自动消除键盘了。因为键盘占据了界面一面的大小，因此这一点让人很不爽。当我们触碰屏幕上的其它区域时，虚拟键盘应该可以自动消失。这一点很容易实现。

在viewDidLoad方法的最后添加以下代码(黄色高亮部分）：

```
-(void)viewDidLoad{

    [super viewDidLoad];

    self.descriptionTextView.text = _descriptionText;
    self.categoryLabel.text = _categoryName;

    self.latitudeLabel.text = [NSString stringWithFormat:@"%.8f",self.coordinate.latitude];
    self.longitudeLabel.text = [NSString stringWithFormat:@"%.8f",self.coordinate.longitude];

    if(self.placemark !=nil){
```

```
        self.addressLabel.text = [self stringFromPlacemark:self.placemark];
    }else{
        self.addressLabel.text = @"No Address Found";
    }
    self.dateLabel.text = [self formatDate:[NSDate date]];

    UITapGestureRecognizer *gestureRecognizer = [[UITapGestureRecognizer
alloc]initWithTarget:self action:@selector(hideKeyboard:)];
    gestureRecognizer.cancelsTouchesInView = NO;
    [self.tableView addGestureRecognizer:gestureRecognizer];

}
```

gesture recognizer（手势识别）是iOS开发中非常拥有的对象，我们可以使用它来识别触摸和其它手指运动。在上面的代码中，我们创建了一个gesture recognizer对象，然后当某种特殊的手势被捕捉到时调用一个特定的方法，最后将这个手势识别对象添加到表视图中。这里我们创建的是一个UITapGestureRecognizer，它只用于识别简单的触摸操作。当然在iOS中还有其它的手势操作，比如swipe滑动，pan（平移），pinches（捏合）等等。

注意到这里的@selector关键字：

. . . initWithTarget:self action:@selector(hideKeyboard:) . . .
通过使用这个语法，我们通知UITapGestureRecognizer调用@selector()中所指定的方法。这个模式就是所谓的target-action模式。当我们将UIButton,UIBarButtonItem和其它控件关联到动作方法上时，其实就是用的这种模式。target就是消息的接收方，通常是self,而action就是要发送的消息内容。

这里我们选择了在表视图中识别到触摸操作时发送hideKeyboard:消息，接下来需要实现这个方法：

在Locations Details View Controller中添加hideKeyboard:方法，我们选择在viewDidLoad方法下面添加它：

```
-(void)hideKeyboard:(UIGestureRecognizer*)gestureRecognizer{

    CGPoint point =[gestureRecognizer locationInView:self.tableView];
    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:point];
    if(indexPath !=nil && indexPath.section ==0 &&indexPath.row ==0){
        return;
    }

    [self.descriptionTextView resignFirstResponder];
}
```

CGPoint是UIKit中另一种常用的结构体，它包含了两个字段x和y,用来描述屏幕上的位置。这里我们使用这个CGPoint类型的变量来获取所在位置的index-path。这一点很重要，因为我们不希望用户触摸描述文本区的时候隐藏键盘！
编译运行应用，试试看是不是实现我们所想要的效果了？

在iOS7中，当用户开始滚动的时候table view可以自动关闭虚拟键盘，这一点可以在storyboard中启用。

打开storyboard，选择带static cells的table view，在右侧面板中切换到Attributes inspector,然后将Keyboard的选项从Do not dismiss更改为Dismiss on Drag。现在滚动操作也会自动隐藏虚拟键盘了。

好了，这一课的内容就到此结束吧。

5天过去了，马航还是一点消息都没有，难道黑匣子里面没有GPS装置?

继续祈福吧。