从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter27

开发环境：
Xcode 5 +iOS 7

欢迎继续我们的学习。

让我们继续上一章的内容，在Add/Edit Item界面中添加"due date"和"should remind"两

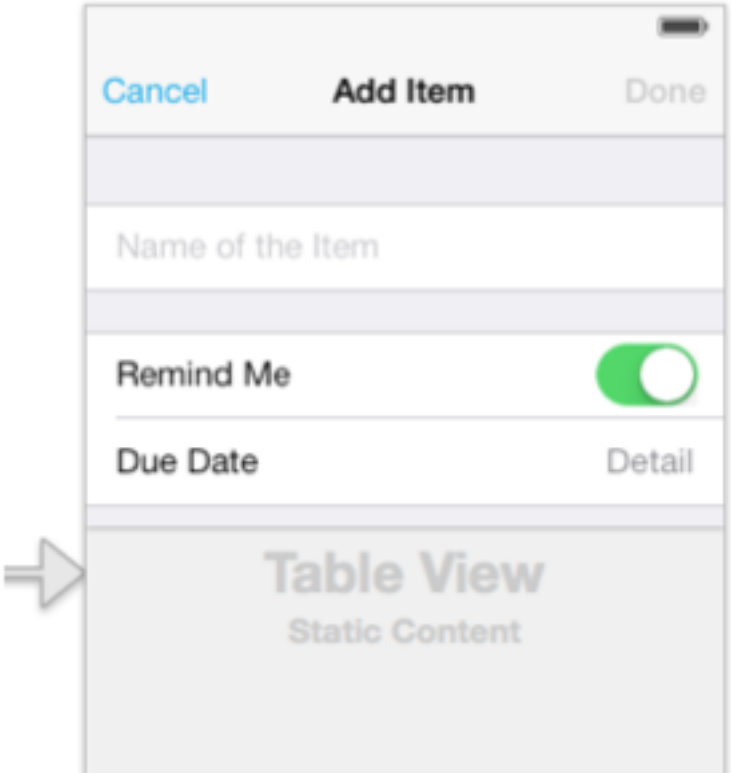
在Xcode中切换到ItemDetailViewController.h,然后添加以下outlet类型的属性变量声明：

```
@property(nonatomic,weak) IBOutlet UISwitch *switchControl;
@property(nonatomic,weak) IBOutlet UILabel *dueDateLabel;
```

接下来切换到storyboard，然后选中Item Detail View Controller（导航栏上是Add Item)里面的Table View。在表视图中添加一个新的section。还记得怎么添加吗？其实方法很简单，只需要在Xcode右侧的面板中切换到Attributes inspector，然后将Sections的值加1即可。通过这种方式可以很方便的复制添加一个section和其中的cell。

从新的cell中删除Text Field，从Object Library中拖一个新的Table View Cell，然后将其放到刚才的cell下面，这样这个section就有两行了。

最终的表视图会是类似下面的样子：



在第一个cell中添加一个Label,然后将其文本内容更改为Remind Me。
此外还需要拖一个Switch 控件到cell中。接下来将它和视图控制器中的switchControl outlet创建关联。（虽然我很想直接将这个outlet命名为switch，遗憾的是它是Objective-C语言中的保留关键字）。

第三个cell中有两个label: 左边是Due Date，而右侧的label将用来保存用户所选中的截止日期。其实我们无需手动添加这两个label，只需要将cell的Style属性设置为Right Detail就好了。

此时可以将右侧的label和视图控制器中的dueDateLabel outlet属性变量关联起来（选中这个表标签的最佳方式是从左侧的控件列表中选择）。

好了，接下来又是代码时间了。

在Xcode中切换到ItemDetailViewController.m，然后在@implementation这一行语句后面添加一个新的_dueDate实例变量。

```
@implementation ItemDetailViewController{
    NSDate *_dueDate;
}
```

接下来更改viewDidLoad方法的代码如下：

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

  if (self.itemToEdit != nil) {
    self.title = @"Edit Item";
    self.textField.text = self.itemToEdit.text;
    self.doneBarButton.enabled = YES;
      self.switchControl.on = self.itemToEdit.shouldRemind;
      _dueDate = self.itemToEdit.dueDate;
  }else{
      self.switchControl.on = NO;
      _dueDate = [NSDate date];

  }

    [self updateDueDateLabel];
}
```

如果我们已经有一个ChecklistItem对象了，那么可以根据对象的shouldRemind属性来设置switch的开关状态。如果用户在添加一个新的ChecklistItem对象，那么默认将switch设置为关闭状态。

对于新创建的代办事项来说，截止日期就是当前，或者说是[NSDate date]。当然，这个截止日期无关紧要。因为当用户填充完其它字段触碰Done时，截止日期早就过了。不够这里我们还是要提供一个设置。一个默认的替代值可以是明天的同一时间，或是从当前时间开始的10分钟后。但是大多数情况下用户还是希望可以选择自己定义的截止日期。

刚才的代码中调用了一个新的updateDueDateLabel方法，让我们在ItemDetailViewController.m中添加该方法的实现代码如下：

```
-(void)updateDueDateLabel{

    NSDateFormatter *formatter = [[NSDateFormatter alloc]init];
    [formatter setDateStyle:NSDateFormatterMediumStyle];
    [formatter setTimeStyle:NSDateFormatterShortStyle];
    self.dueDateLabel.text = [formatter stringFromDate:_dueDate];

}
```

为了将NSDate各式的值转换为文本，我们需要用到NSDateFormatter对象。它的工作原理很简单，我们需要为日期设置一种样式，然后为时间设置一种样式，最后用它来格式化NsDate对象。当然，只要你愿意，可以选择不同的显示样式，但由于label标签的空间有限，我们没办法把完整的月份放进去。NSDateFormatter更NB的一点在于，它会自动判断用户所在的地区，然后将各式转化成用户所习惯的样式。

好了，最后还需要更改done这个动作方法。在ItemDetailViewController.m中更改done方法如下：

```
- (IBAction)done
{
  if (self.itemToEdit == nil) {
    ChecklistItem *item = [[ChecklistItem alloc] init];
    item.text = self.textField.text;
    item.checked = NO;
    item.shouldRemind = self.switchControl.on;
    item.dueDate = _dueDate;

    [self.delegate itemDetailViewController:self didFinishAddingItem:item];

  } else {
    self.itemToEdit.text = self.textField.text;
    self.itemToEdit.shouldRemind = self.switchControl.on;
    self.itemToEdit.dueDate = _dueDate;
    [self.delegate itemDetailViewController:self didFinishEditingItem:self.itemToEdit];
  }
}
```

当用户触碰Done按钮时，我们会将switch控件和_dueDate实例变量的值保存到ChecklistItem对象的属性中。

注意：

你可能会想，为什么对_dueDate使用实例变量，但是shouldRemind却没用到。这是因为shouldRemind很容易获取switch控件的状态：只需要查看它的on属性就知道了。但是从

dueDateLabel中读取所选日期信息却不太方便，因为这个标签里面保存的是文本（NSString)各式，而不是NSDate。因此最好还是将所选日期放到一个NSDate实例变量中。

好了，现在编译运行应用，然后试着更改switch开关控件的状态。应用将会记住你在使用中的最后设置，即便你强关了应用。（记住在Simulator上测试时要先用Hardware-Home)

当然截止日期信息部分现在只是个摆设，为了让它正常工作，我们首先需要创建一个日期选择器。

添加日期选择器

日期选择器其实不是一个新的视图控制器。当用户触碰Due Date这一行的时候，会直接在表视图中插入一个新的UIDatePicker控件，就像一个内置的Calendar应用一样。

现在开始动手吧。
首先在ItemDetailViewController.m中添加一个新的实例变量，用来保存当前日期选择器是否可见的状态。

```
@implementation ItemDetailViewController{
    NSDate *_dueDate;
    BOOL _datePickerVisible;
}
```

接着添加showDatePicker方法如下：

```
-(void)showDatePicker{

    _datePickerVisible = YES;

    NSIndexPath *indexPathDatePicker = [NSIndexPath indexPathForRow:2 inSection:1];
    [self.tableView insertRowsAtIndexPaths:@[indexPathDatePicker]
withRowAnimation:UITableViewRowAnimationAutomatic];
}
```

在上面的方法中，我们首先设置新创建的实例变量值为YES，然后告诉表视图在Due Date cell的下面添加一个新的行。这个新的行里面将包含UIDatePicker控件。

问题来了：

这个新行的cell从何处获取呢？通常情况下我们会实现cellForRowAtIndexPath方法，不过在这个界面中表视图是使用static cell创建的。此类表视图没有特定的数据源，因此无法使用cellForRowAtIndexPath方法。不信你可以在ItemDetailViewController.m中找找看。

不过别担心。只需要换一种思路，我们就可以用自己的定制方法来覆盖静态表视图的数据源。

在ItemDetailViewController.m中添加cellForRowAtIndexPath方法如下：

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

   //1
   if(indexPath.section ==1 &&indexPath.row ==2){

      //2
      UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"DatePickerCell"];

      if(cell == nil){

         cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"DatePickerCell"];

         cell.selectionStyle = UITableViewCellSelectionStyleNone;

         //3

         UIDatePicker *datePicker = [[UIDatePicker alloc]initWithFrame:CGRectMake(0.0f, 0.0f,
320.0f,216.0f)];
         datePicker.tag =100;
         [cell.contentView addSubview:datePicker];

         //4
         [datePicker addTarget:self action:@selector(dateChanged:)
forControlEvents:UIControlEventValueChanged];


      }
      return cell;

      //5
   }else{
      return [super tableView:tableView cellForRowAtIndexPath:indexPath];
   }
```

}

让我们来仔细看看上面这段代码做了些什么：

1.检查当前是否存在日期选择器所在行对应的index-path，如果没有，跳转到第5步

2.询问表视图它是否已经有了日期选择器的cell。如果没有就创建一个新的。selection style（选择样式）是none，因为我们不希望在用户触碰它的时候显示一个已选中的状态。

3.创建一个新的UIDatePicker控件。将其tag值设置为100，以便后续使用。

4.告诉日期选择器，每当用户更改了日期的时候调用dateChanged:方法。此前我们已经知道如果从Interface Builder关联动作方法了。而这里则演示了如何从代码中关联动作方法。UIDatePicker的Value Changed方法将会触发dateChanged方法。当然，此时我们还没有定义该方法，Xcode会给出警告提示。

5.对于任何非日期选择器cell对应的index-paths，直接调用super（也就是表视图控制器）。这样之前的static cell不会受到任何影响。

接下来还需要覆添加一个numberOfRowsInSection方法：

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

    if(section ==1 && _datePickerVisible){
        return 3;
    }else{
        return [super tableView:tableView numberOfRowsInSection:section];
    }
}
```

如果日期选择器可见，就说明当前的section1有3行，反之我们只需要调用super，利用初始数据源获取相关信息。

然后还需要添加一个heightForRowAtIndexPath方法：

```
-(CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath{

    if(indexPath.section ==1 &&indexPath.row ==2){
        return 217.0f;
    }else{
        return [super tableView:tableView heightForRowAtIndexPath:indexPath];
    }


}
```

到目前为止，表视图中的cell高度是相同的44points，不过这不是必须的。通过heightForRowAtIndexPath方法，我们可以为每个cell指定不同的高度。UIDatePicker控件的高是216points，再加上1point的分割线，所以总共占了217points。

只有当用户触碰了Due Date cell之后才会显示日期选择器，而这个事件是在didSelectRowAtIndexPath中进行处理的：

添加该方法的实现代码：

```
-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    [self.tableView deselectRowAtIndexPath:indexPath animated:YES];

    [self.textField resignFirstResponder];

    if(indexPath.section ==1 &&indexPath.row ==1){
        [self showDatePicker];
    }

}
```

当我们根据index-path信息判断出Due Date行被触碰时，就可以调用showDatePicker方法了。同时我们还将隐藏虚拟键盘。

虽然大多数的准备工作已经就绪，但现在Due Date 行还不能被触碰。这是ItemDetailViewController.m已经有一个willSelectRowAtIndexPath方法，但其中返回的却是nil。

更改willSelectRowAtIndexPath方法如下：

```
- (NSIndexPath *)tableView:(UITableView *)tableView willSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    if(indexPath.section ==1 &&indexPath.row ==1){
        return indexPath;
    }else{
        return nil;
    }
}
```

现在Due Date这一行可以对用户的触碰操作有所响应了，但其它行则不会。

编译运行应用，添加一个新的ChecklistItem对象，然后触碰Due Date行试试看。

然后就没有然后了。

因为程序崩溃了。原因何在？当我们覆盖了静态table view cell的数据源方法时，还应提供一个代理方法indentationalLevelForRowAtIndexPath。通常情况下我们用到该方法的机会不多，但因为这里我们混合使用了static table view的数据源和自定义的数据源方法，就必须提供该方法。

在ItemDetailViewController.m中添加indentationLevelForRowAtIndexPath方法：

```
-(NSInteger)tableView:(UITableView *)tableView indentationLevelForRowAtIndexPath:
(NSIndexPath *)indexPath{

    if(indexPath.section ==1 &&indexPath.row ==2){
        NSIndexPath *newIndexPath = [NSIndexPath indexPathForRow:0
inSection:indexPath.section];
        return [super tableView:tableView indentationLevelForRowAtIndexPath:newIndexPath];
    }else{
        return [super tableView:tableView indentationLevelForRowAtIndexPath:indexPath];
    }

}
```

应用之前会崩溃，是因为标准数据源方法对section1的第2行的cell一无所知（也就是日期选择器的cell），因为它不是storyboard中表视图设计的一部分。因此在插入新的日期选择器cell后，数数据源就被搞糊涂了。
这里我们通过这个方法让数据源相信它们在这个section的确拥有三行。
再次运行应用，此时会出现日期选择器的cell。

可惜好事多磨，如果你手痒尝试着转一转日期选择器，oops,程序再次崩溃了！

这是因为UIDatePicker会在用户更改了日期信息时尝试调用dateChanged方法，可惜我们还根本没添加这个方法。

在ItemDetailViewController.m中添加dateChanged方法：

```
-(void)dateChanged:(UIDatePicker*)datePicker{

    _dueDate = datePicker.date;
    [self updateDueDateLabel];
}
```

这个方法的内容很简单，首先会将新选择的日期保存到_dueDate这个实例变量中，然后调用updateDueDateLabel方法来更新Due Date标签上的文本。

再次编译运行。当你试着拨动日期选择器时，Due Date行的文本内容也会随之改变。

NB！

不过还没完，当我们编辑一个现有的待办事项时，日期选择器并不会显示该项目所设置的日期，而是直接显示当前时间。

让我们来修复这个小问题。

更改showDatePicker方法如下：

```
-(void)showDatePicker{

    _datePickerVisible = YES;

    NSIndexPath *indexPathDatePicker = [NSIndexPath indexPathForRow:2 inSection:1];
    [self.tableView insertRowsAtIndexPaths:@[indexPathDatePicker]
withRowAnimation:UITableViewRowAnimationAutomatic];

    UITableViewCell *datePickerCell = [self.tableView
cellForRowAtIndexPath:indexPathDatePicker];

    UIDatePicker *datePicker = (UIDatePicker*)[datePickerCell viewWithTag:100];
    [datePicker setDate:_dueDate animated:NO];
}
```

上面的黄色高亮部分就是增加的代码。
通过这两行代码，我们获取了UIDatePicker所在的cell,然后让它上面的日期选择器显示正确的日期。

好了，现在可以编译运行应用，验证下刚才工作的成效。

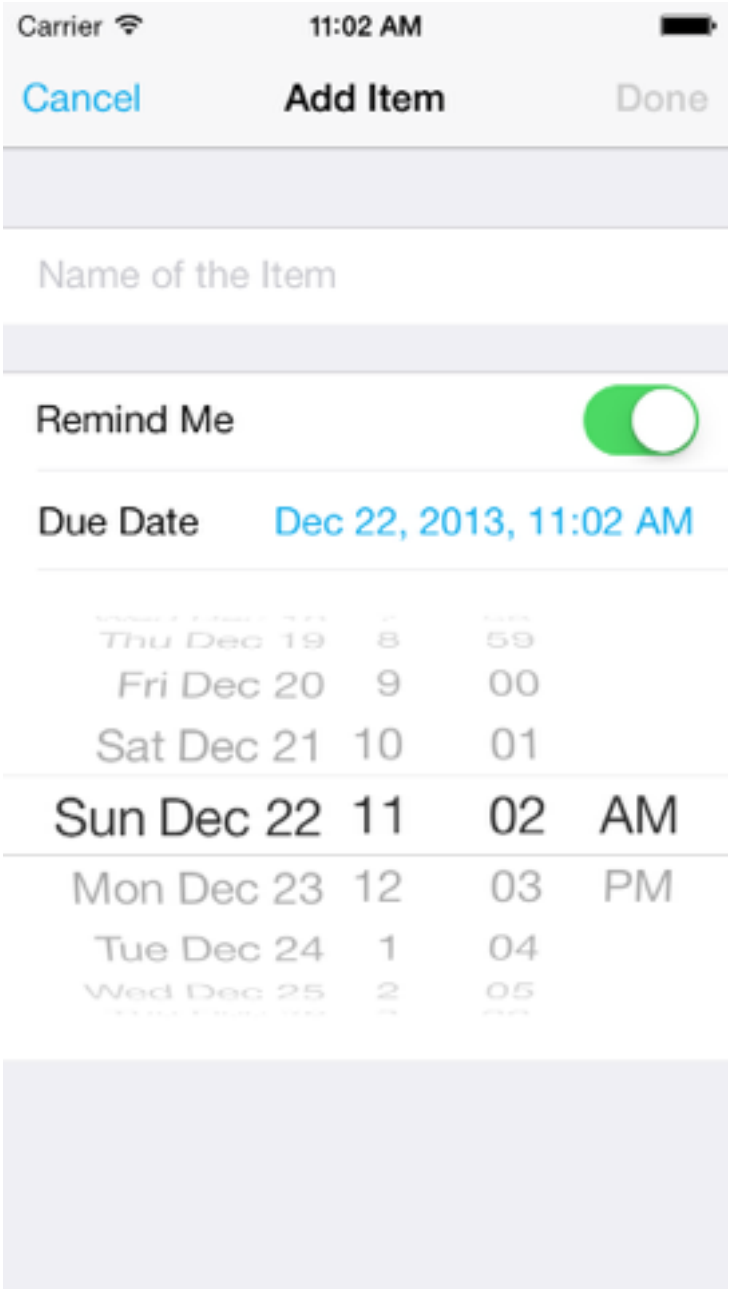
说到日期标签，我们可以考虑当它处于活跃状态时将其高亮显示。一个很简单的方法是使用tint color（着色），也就是苹果官方的Calendar应用采用的方式。

再次更改showDatePicker方法：

```
-(void)showDatePicker{

    _datePickerVisible = YES;

    NSIndexPath *indexPathDateRow = [NSIndexPath indexPathForRow:1 inSection:1];

    NSIndexPath *indexPathDatePicker = [NSIndexPath indexPathForRow:2 inSection:1];

    UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPathDateRow];
    cell.detailTextLabel.textColor = cell.detailTextLabel.tintColor;

    [self.tableView beginUpdates];

   [self.tableView insertRowsAtIndexPaths:@[indexPathDatePicker]
withRowAnimation:UITableViewRowAnimationFade];

    [self.tableView reloadRowsAtIndexPaths:@[indexPathDateRow]
withRowAnimation:UITableViewRowAnimationNone];


    [self.tableView endUpdates];

    UITableViewCell *datePickerCell = [self.tableView
cellForRowAtIndexPath:indexPathDatePicker];

    UIDatePicker *datePicker = (UIDatePicker*)[datePickerCell viewWithTag:100];
    [datePicker setDate:_dueDate animated:NO];
}
```

通过以上的黄色高亮代码，我们将detailTextLabel的textColor社设置为tint color。同时我们还通知表视图需要重新加载Due Date。
因为这里我们同时对表视图进行了两次操作，分别是插入新的一行，以及重新加载另一行，因此需要把这种操作放在beginUpdates和endUpdates之间。这样表视图就可以同时进行两种动画了。

编译运行应用，现在日期会用蓝色显示。



看起来有点高大上的赶脚了。

可惜的是，如果此时你再次触碰Due Date这一行，日期选择器会消失，但应用也会崩溃，显然这个应用很有可能无法通过苹果审核。即便运气逆天没被审查人员发现，也会在用户和boss的怒骂声中度过悲催的一晚。

在ItemDetailViewController.m中添加一个新的hideDatePicker方法：

```
-(void)hideDatePicker{

    if(_datePickerVisible){
        _datePickerVisible = NO;

        NSIndexPath *indexPathDateRow = [NSIndexPath indexPathForRow:1 inSection:1];
        NSIndexPath *indexPathDatePicker = [NSIndexPath indexPathForRow:2 inSection:1];
```

```
        UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPathDateRow];
        cell.detailTextLabel.textColor = [UIColor colorWithWhite:0.0f alpha:0.5f];

        [self.tableView beginUpdates];

        [self.tableView reloadRowsAtIndexPaths:@[indexPathDateRow]
withRowAnimation:UITableViewRowAnimationNone];

        [self.tableView deleteRowsAtIndexPaths:@[indexPathDatePicker]
withRowAnimation:UITableViewRowAnimationFade];

        [self.tableView endUpdates];

    }

}
```

以上方法做了和showDatePicker方法相反的工作：它从表视图中删除了日期选择器对应的cell,并重新将日期标签的色彩恢复到中度灰。

接下来更改didSelectRowAtIndexPath方法切换日期选择器的显示和隐藏状态：

```
-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    [self.tableView deselectRowAtIndexPath:indexPath animated:YES];

    [self.textField resignFirstResponder];

    if(indexPath.section ==1 &&indexPath.row ==1){

        if(!_datePickerVisible){
            [self showDatePicker];
        }else{
            [self hideDatePicker];
        }
    }

}
```

除此之外，还有一种情况下需要隐藏日期选择器：当用户触碰到文本区里面的时候。如果虚拟键盘和日期选择器重叠在一起，肯定会让用户感到不爽的，因此此时我们最好隐藏它。不过考虑到视图控制器已经是文本域的代理了，因此实现这一点相当简单：

在ItemDetailViewController.m中添加一个textFieldDidBeginEditing方法：

```
-(void)textFieldDidBeginEditing:(UITextField *)textField{

    [self hideDatePicker];
}
```

好了，只需这一行代码，又提升了用户体验。

2014年已经到了，这个系列的教程马上就要结束了。
不过考虑到篇幅，还是把最后的高潮留给下一章吧，让我们共同期待。

送上福利mm照