从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter25

版权声明：
原文及示例代码来自raywenderlich store中的iOS Apprentice 系列2教程，经过翻译和改编。
版权归原作者所有，本系列教程仅供学习参考使用，感兴趣的朋友建议购买原教程(http://www.raywenderlich.com/store/ios-apprentice)。

开发环境：
Xcode 5 +iOS 7

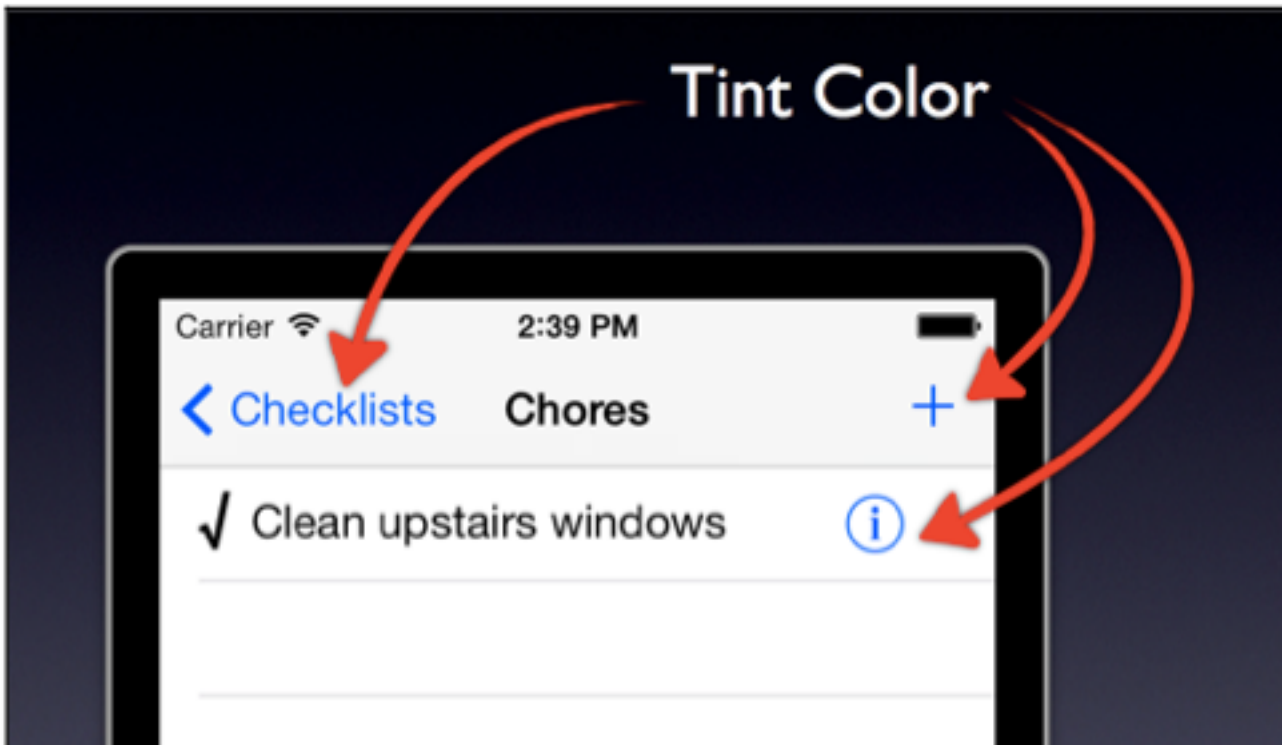
欢迎继续我们的学习。

开始最后几章的终极挑战吧。

首先让我们美化下这个应用。

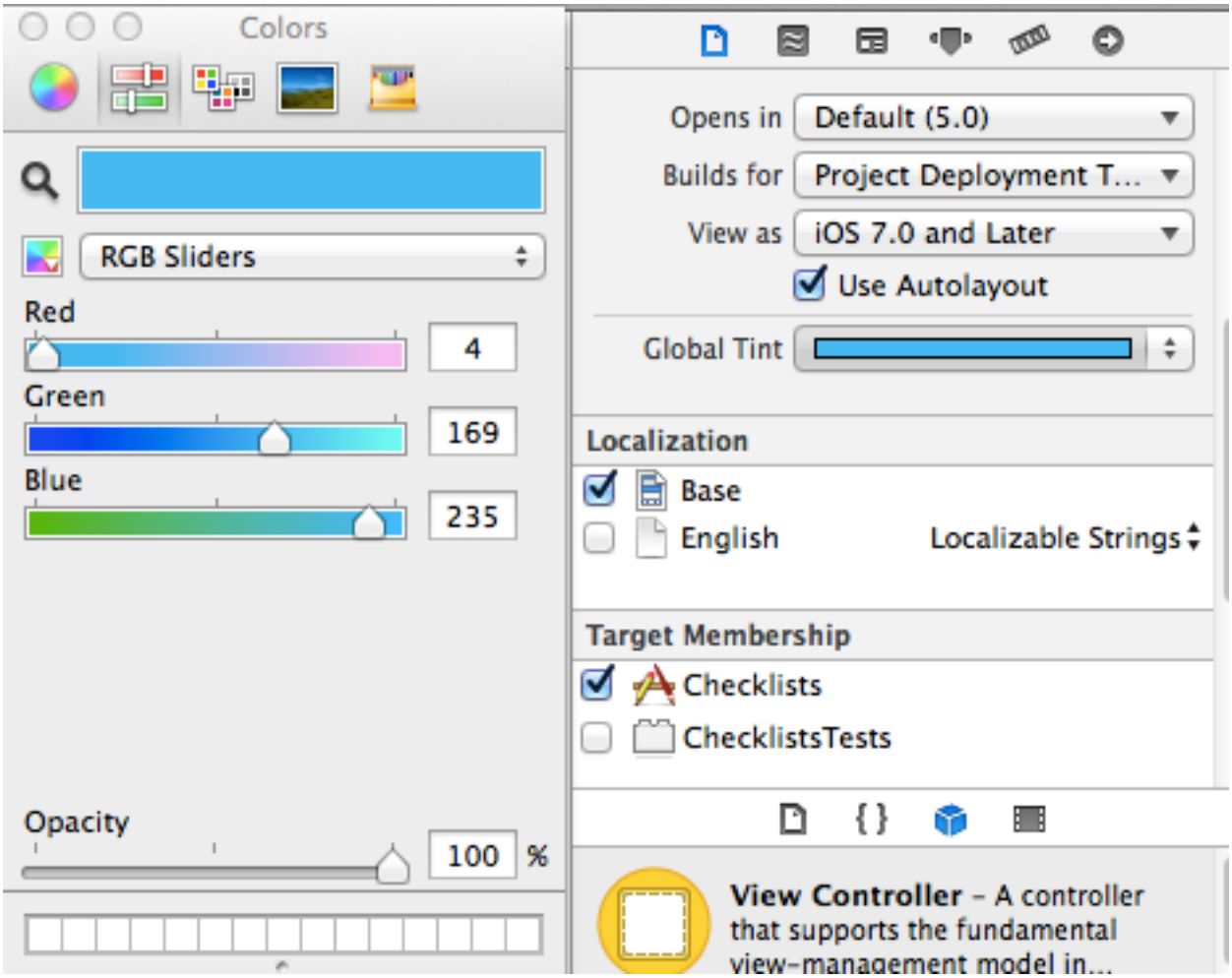
考虑到教程的复杂度，本篇教程依然会采用系统默认的标准样式，比如导航控制器，表视图，等等。当然，或许对于稍有审美情趣的人都会觉得这种样式有点苍白无力。

不过即便是采用系统默认的标准样式，仍然可以让界面富有个性：比如改变tint color，也就是着色。UIKit使用着色来指明那些用于交互的视觉元素，比如按钮。默认的着色是中蓝色。



更改着色非常简单。

首先打开Xcode，切换到storyboard,然后在右侧面板切换到File inspector（第一个选项卡）。
点击Global Tint，选择最下面的Others，打开色彩选择器，选择Red:4,Green:169,Blue:235，这样可以让着色变成浅一点的蓝色（有点天蓝色的赶脚）。当然，你也可以使用自己的配色，只要视觉效果好就行。



接着我们可以调整下勾选标志的颜色，毕竟黑色看起来有点太单调了。

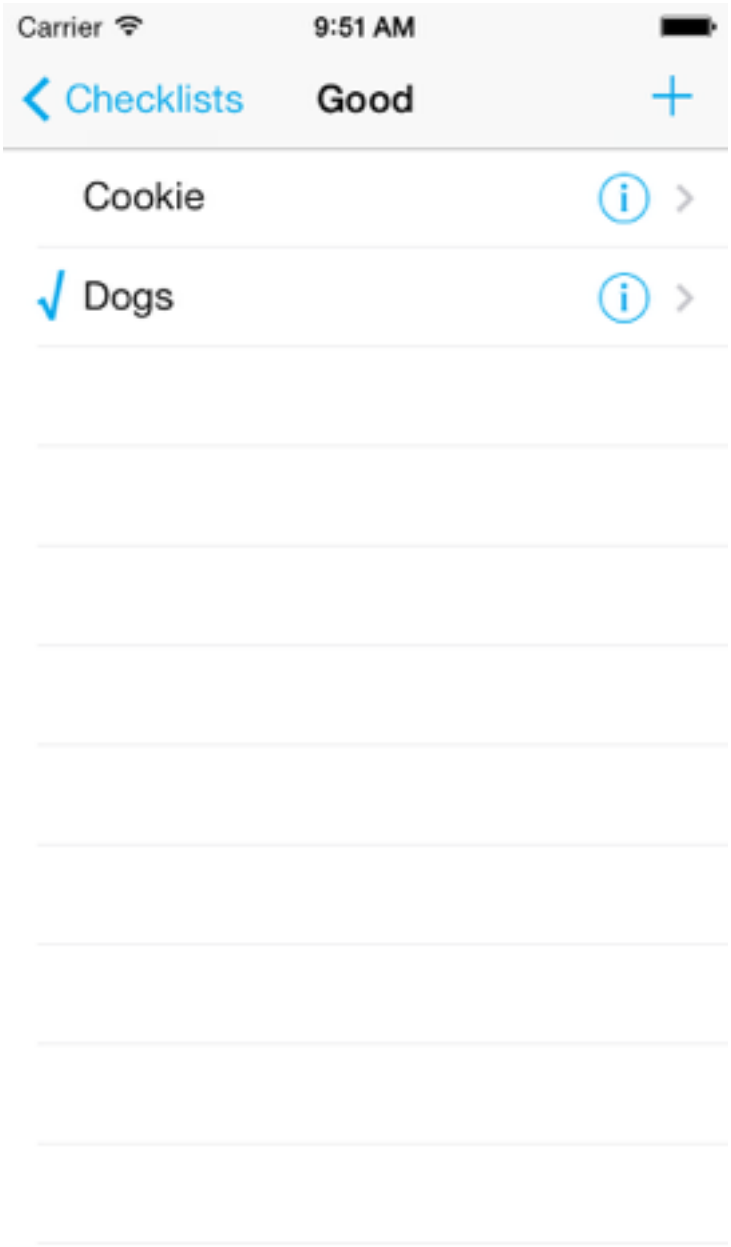
在Xcode中切换到ChecklistViewController.m，然后在configureCheckmarkForCell方法中添加一行代码：

```
- (void)configureCheckmarkForCell:(UITableViewCell *)cell withChecklistItem:(ChecklistItem *)item
{
  UILabel *label = (UILabel *)[cell viewWithTag:1001];

  if (item.checked) {
    label.text = @"√";
  } else {
    label.text = @"";
  }

    label.textColor = self.view.tintColor;
}
```

编译运行应用，虽然只是小小的改动，或许也会有种耳目一新的赶脚~



任何一款iOS应用都不能没有图标和载入画面。之前也谈到过，如果要想让你的应用在App Store中脱颖而出，有几点元素是至关重要的。
1.图标和载入画面、截图、描述、名称
2.启动速度和交互爽快感
3.恰到好处的动态效果
4.简约不简单的功能设计
5.清爽可口的界面设计
6.社交元素
7.针对目标客户的运营（渠道，LBS，等等）

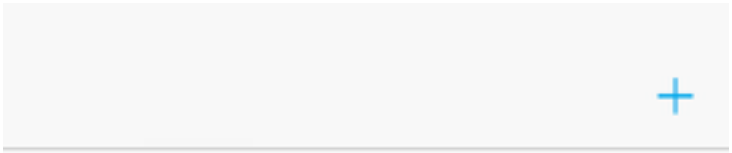
接下来让我们给这个应用添加图标和载入画面。

在本章的Resources文件夹中包含了一个名为Icon的子文件夹，如果你足够细心，会发现它的色彩和刚才的着色是相同的浅蓝色。

使用和之前类似的方法将图标文件添加到asset catalog（Images.xcassets)中。记得要把图标添加到AppIcon部分。添加的方法比你想象的还要简单，只需要从Finder中对应的文件夹中把这些文件拖进这部分就行。(比如把Icon-29.png拖到最左的1x里面，以此类推）

接下来将Resources文件夹里面LaunchImage子文件夹中的文件拖到Images.xcassets里面。（把Default-568h@2x.png拖到R4那一格，把Default@2x.png拖到2x那一格。注意到现在针对iPhone的应用只需要提供retina分辨率的就可以了（960*480和1136*480)

这里的启动加载画面很简单，只不过是一个导航栏带一个添加按钮，然后是一个空的表视图。通过这样的加载画面，可以给用户造成一个错觉，让他们觉得应用的UI已经加载成功，只是数据还没有填充进来。



生成这种启动加载画面很简单，只需要在Simulator中打开应用，然后选择File-Save Screen Shot就可以了。通过这种方式会直接在Desktop生成一个PNG文件。然后我们可以在Photoshop中打开这个文件，根据需要简单的裁剪或调整即可。比如这里我把截图中的状态栏部分剪切掉了，因为在实际运行的时候iPhone会提供自己的状态栏。

此时编译运行应用，不会有黑色画面的过渡，而是会立即看到刚才的加载画面，让人感觉是光速秒进的。

注意：
目前我们主要是用4-inch Simulator来测试的，不过即便在3.5-inch的设备上也是同样的效果。因为这个应用只用到了表视图控制器。表视图控制器会根据屏幕大小自动调整自己的尺寸。不过对于其它类型的视图控制器就未必如此了。

在下一系列的教程中，我们将学习如何根据iPhone设备的不同尺寸来处理这些微妙的差别。

其实到目前为止，这款应用已经算是完成了。

在这二十几章的内容中（看似很多，每章不超过45分钟），我们了解了iOS开发的诸多核心概念，比如表视图控制器，导航控制器，storyboard,segue，表视图和cell单元格（我个人很讨厌这种翻译），数据模型，代理协议。。。

或许有很多概念即便看到现在依然是云里雾里。但是,Don't Panic! 对于看不懂的地方不要钻牛角尖儿，现在看不懂很正常，因为你是在被动的接受知识。而学习一种技能或知识的最佳途径是实战，在挑战实际项目的过程中你还会接触到类似的概念。通过看官方文档，去stackoverflow,github,苹果开发者官方论坛和国内的一些论坛去请教高手或自己主动学习，你会学到更多的知识，而且理解也会更加深刻。

过一段时间再回过头来看这些教程，虽然是入门级别，但仍然会对你有所启示。

好了，今天的学习暂时到此结束。

2013年的最后一天，发送福利的时刻到了。

特别说明的是，虽然这一系列的教程仍然是翻译改编，但不会是最终版本。根据大家的提问和反馈，我会将部分内容进一步优化，同时会专门加一部分内容把大家所提的问题和解决方法整理出来，方便更好的理解教程内容。