从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter11

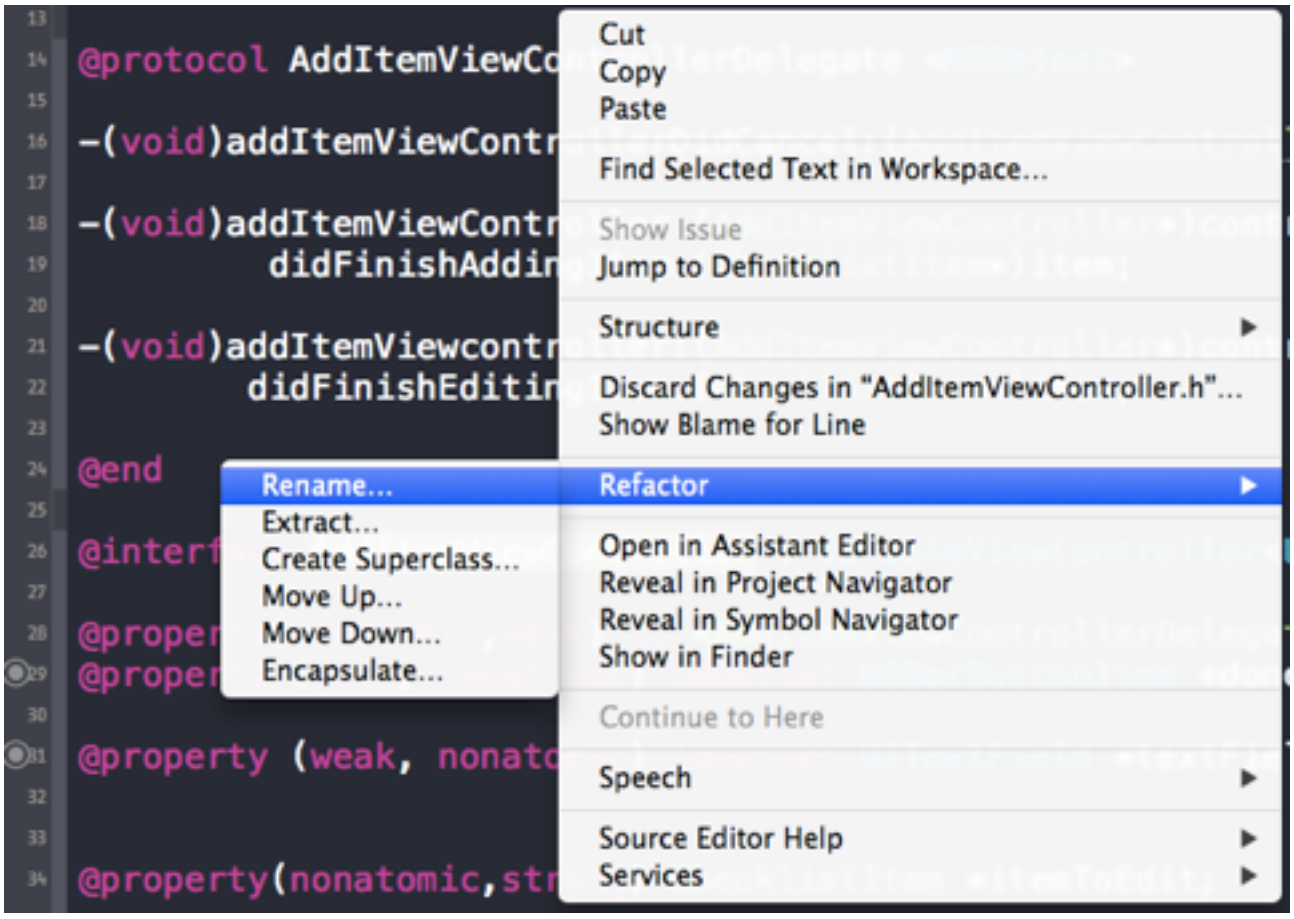
欢迎继续我们的学习。

在之前的几章，我们连续学习了不少新东西，比如代理协议这种看起来让人很头大的东东。当然收获也是很大的。
现在这个应用已经可以显示待办事项清单了，我们可以通过触碰某一行来开启或关闭勾选标志，我们可以添加新的事项，还可以编辑现有的事项，需要的时候也可以删除现有的事项。
看起来这个应用的雏形已经完成了。

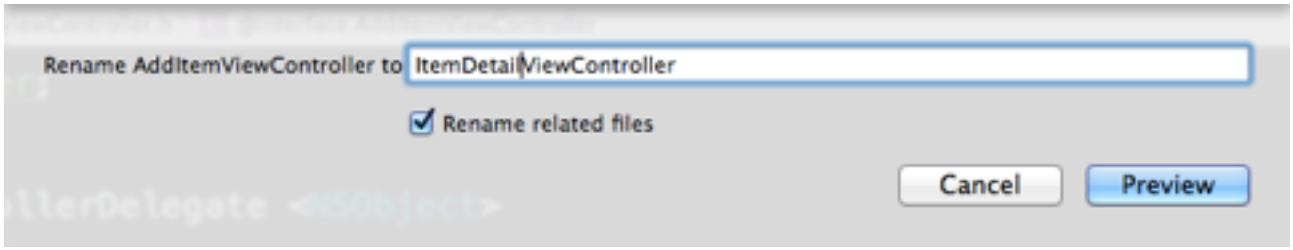
不过持续高强度的学习也会令人精疲力尽，因此本章我们可以稍微放松下，看看如何重构和调整一下已经完成的代码。

首先要做的事情就是更改AddItemViewController这个名称，因为这个界面现在身兼两个重任，分别是添加新事项和编辑现有事项。因此我们考虑将其更名为ItemDetailViewController。

在Xcode中切换到AddItemViewController.h，用鼠标点击@interface这行代码，让光标停留在AddItemViewController这个单词上。右键单击，从弹出菜单中选择Refactor-Rename



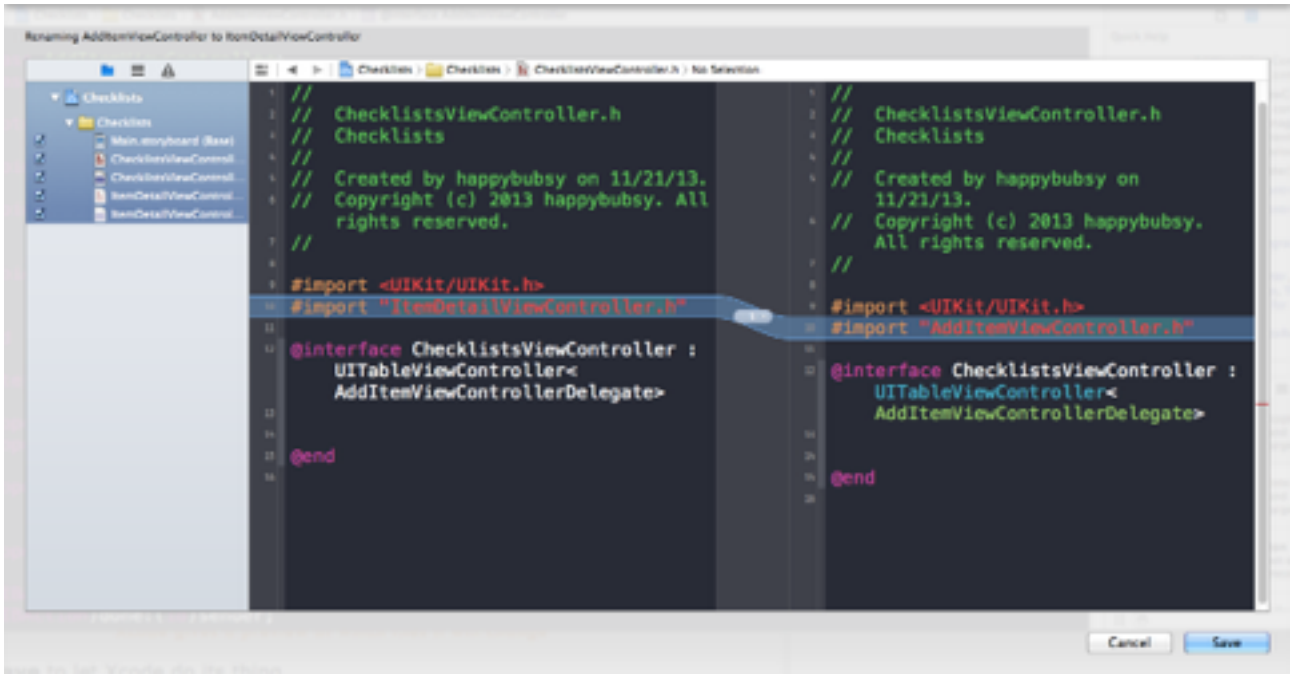
然后在弹出的对话框中输入新的类名ItemDetailViewController



注意一定要勾选Rename related files。

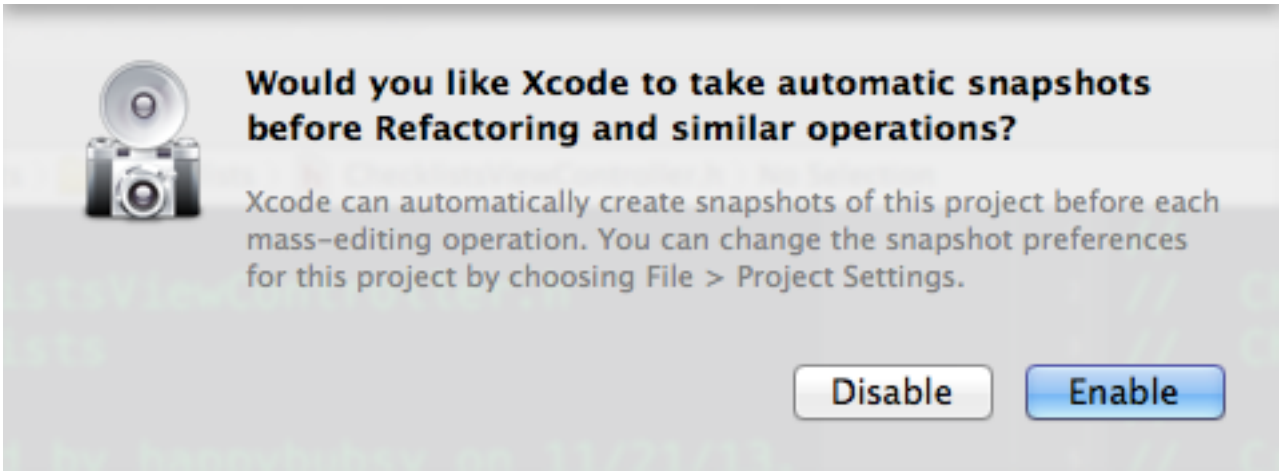
如果Xcode提示"Wait for indexing and try again"，那么记住先停止编译运行项目。

现在点击Preview，Xcode会打开一个界面，上面显示了将被更改的文件。点击某个文件名，可以看看其中可能哪些内容可能被更改。



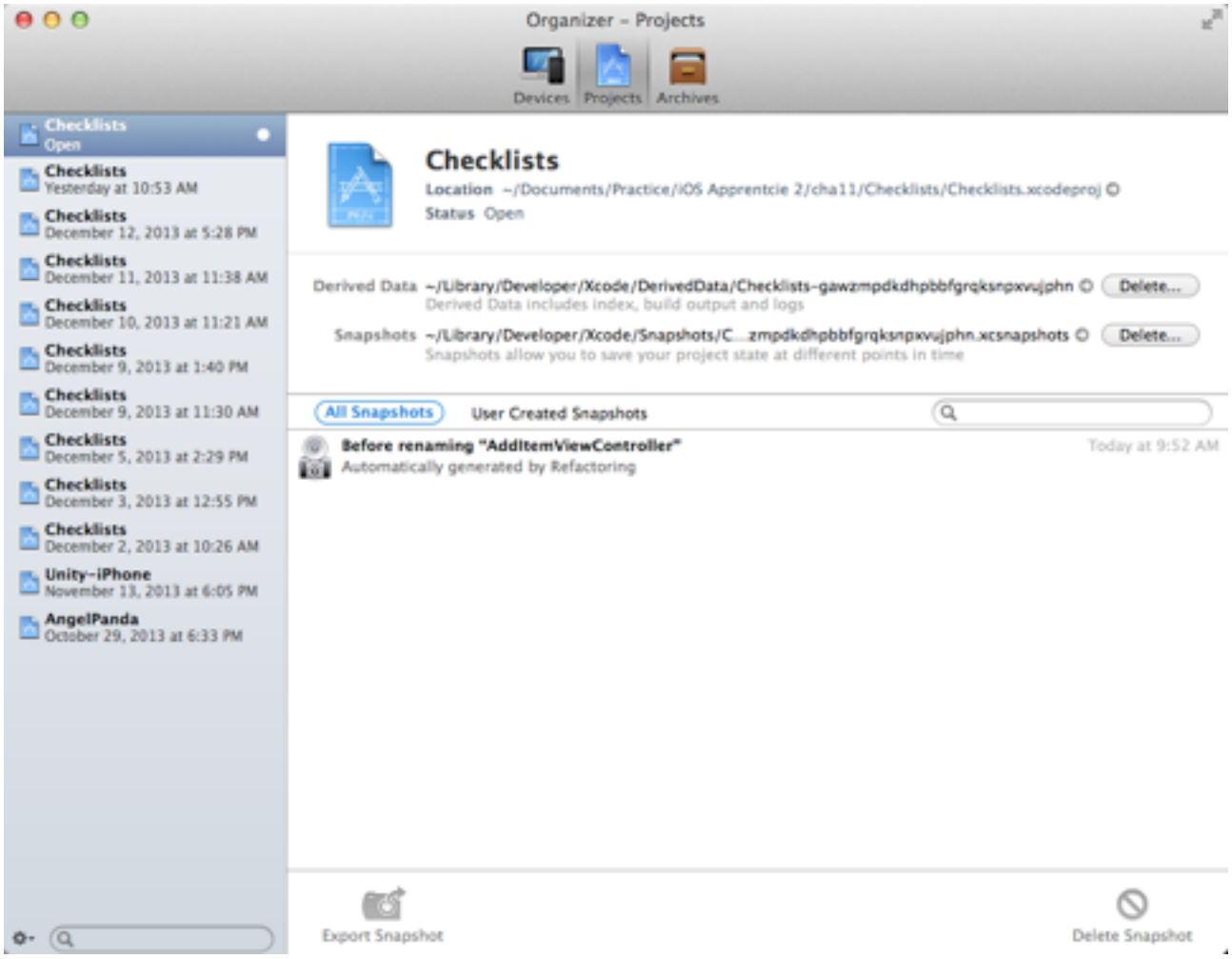
当然，其实Xcode只是把所有的AddItemViewController都替换成了ItemDetailViewController，不过在作出改变之前检查一下还是有必要的。
现在点击Save让Xcode完成这个工作。

Xcode是个很负责任的智能工具，它会再次给你一个提示，问你是否要启用automatic snapshots，也就是自动创建一个项目的快照。快照是当前整个项目的备份，为了安全起见，我们推荐你保存一个快照。这样如果哪里修改错了，我们可以直接使用快照回到修改之前的地方。

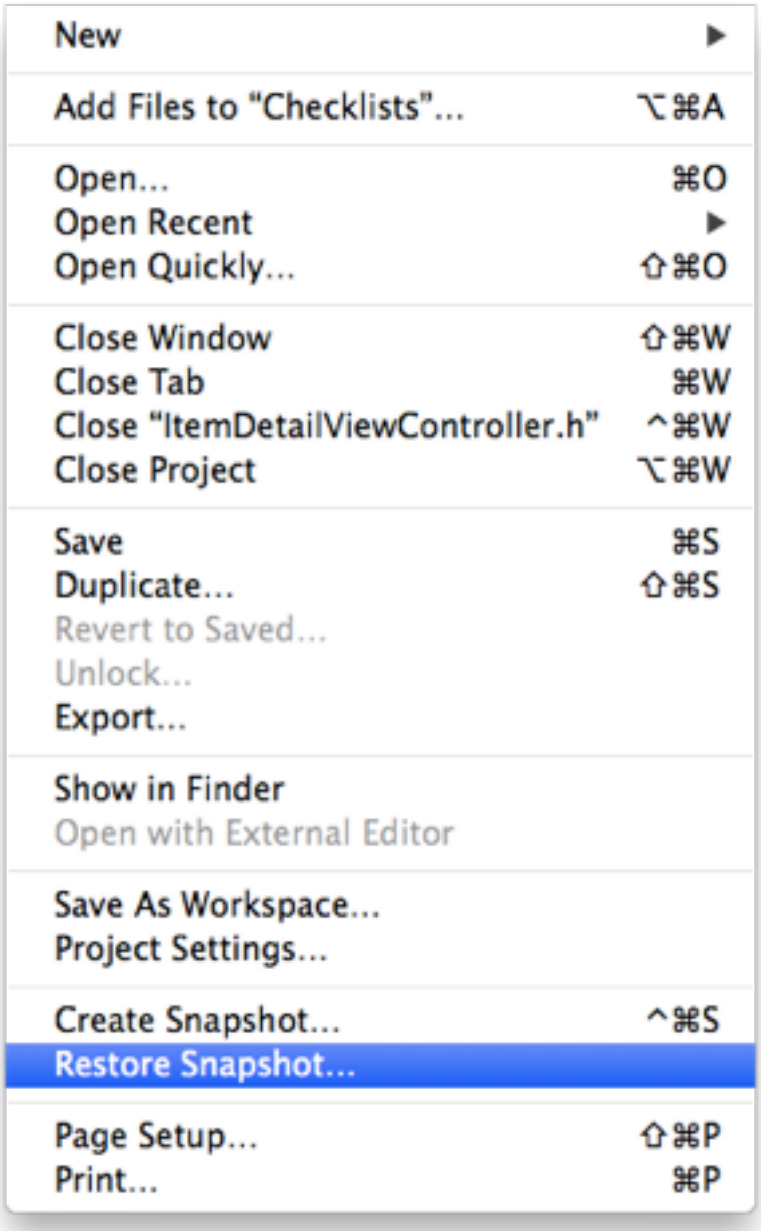


这里点击Enable，然后等Xcode花几秒钟的时间来完成这个操作。

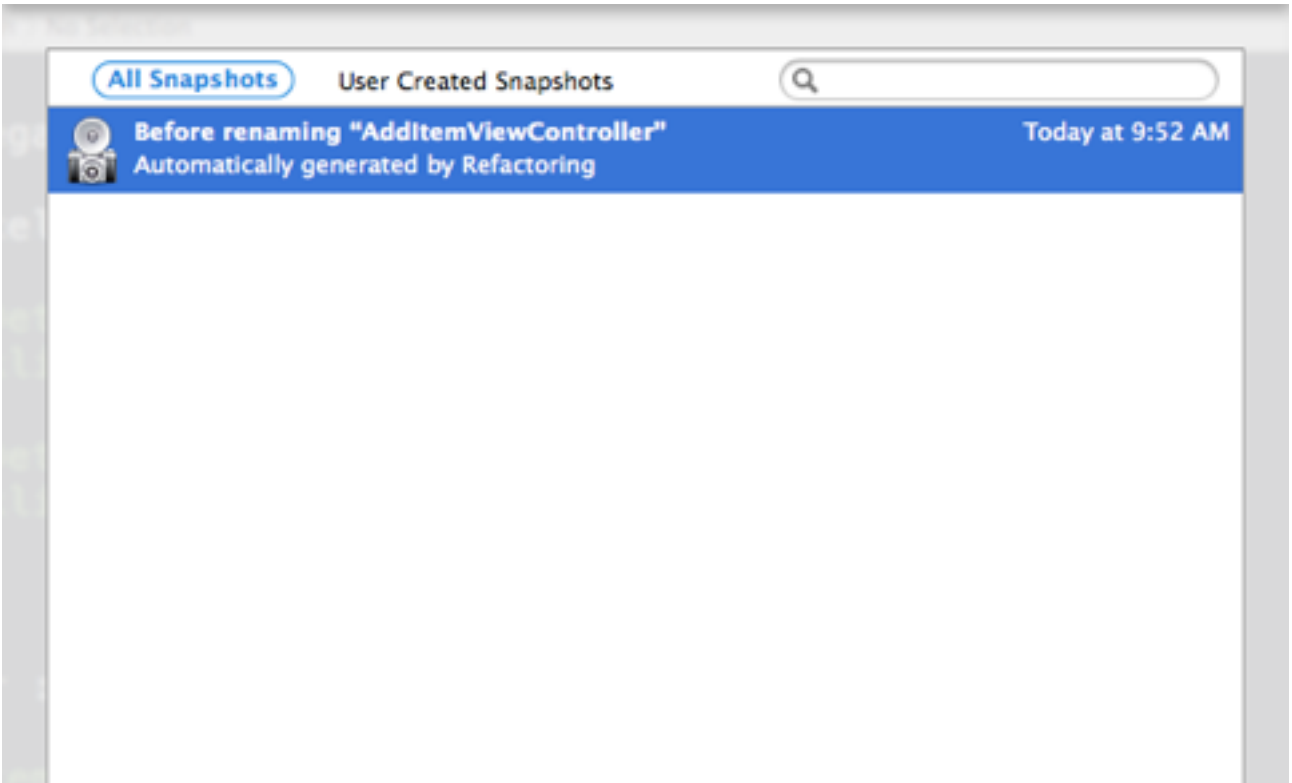
假如你想在之后找到这些快照，可以从Xcode的菜单中打开Organizer窗口(Window-Organizer)，然后切换dao Projects选项卡，就可以看到之前所保存的快照：



如果我们想恢复到某个快照，可以在Xcode的菜单中选择File-Restore Snapshot即可。

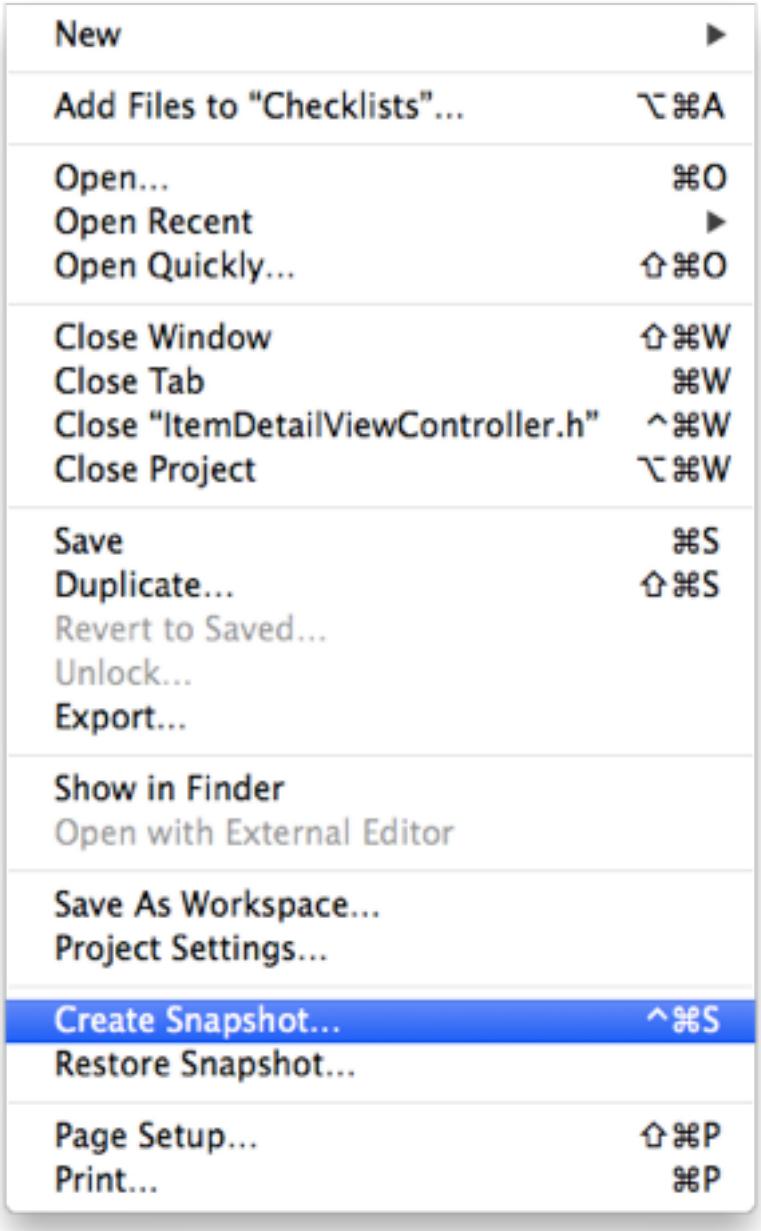


此时可以看到之前所创建的snapshot，比如：

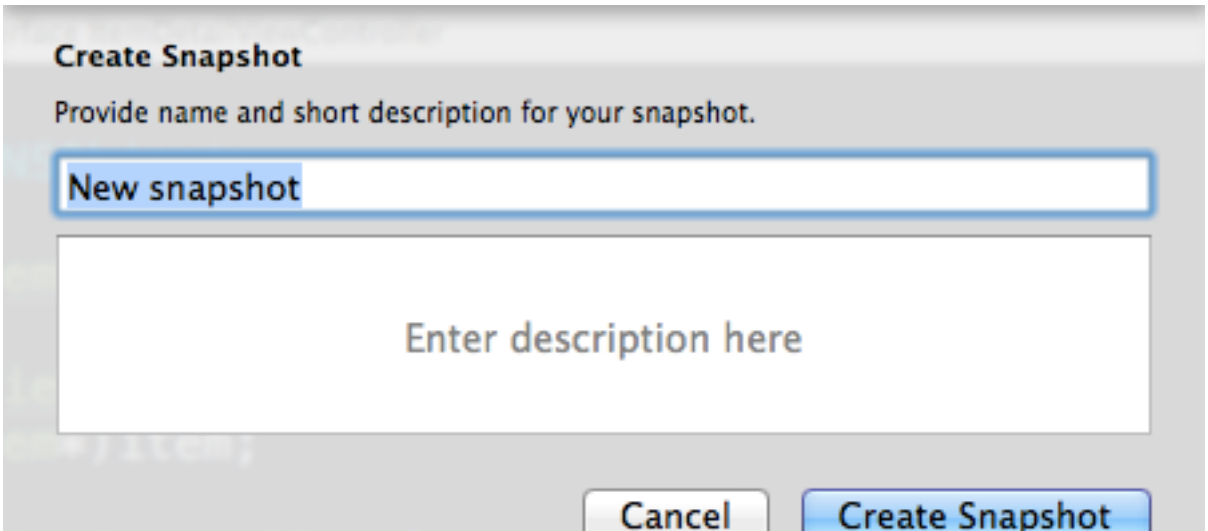


当然，这里我们不需要恢复，所以点击cancel就好了。

好吧，既然说到了restore，那么也顺便提一提如何手动创建一个snapshot。
很简单，从Xcode的菜单中选择File-Create Snapshots即可



此时会看到一个弹出对话框，输入相关信息就好了。



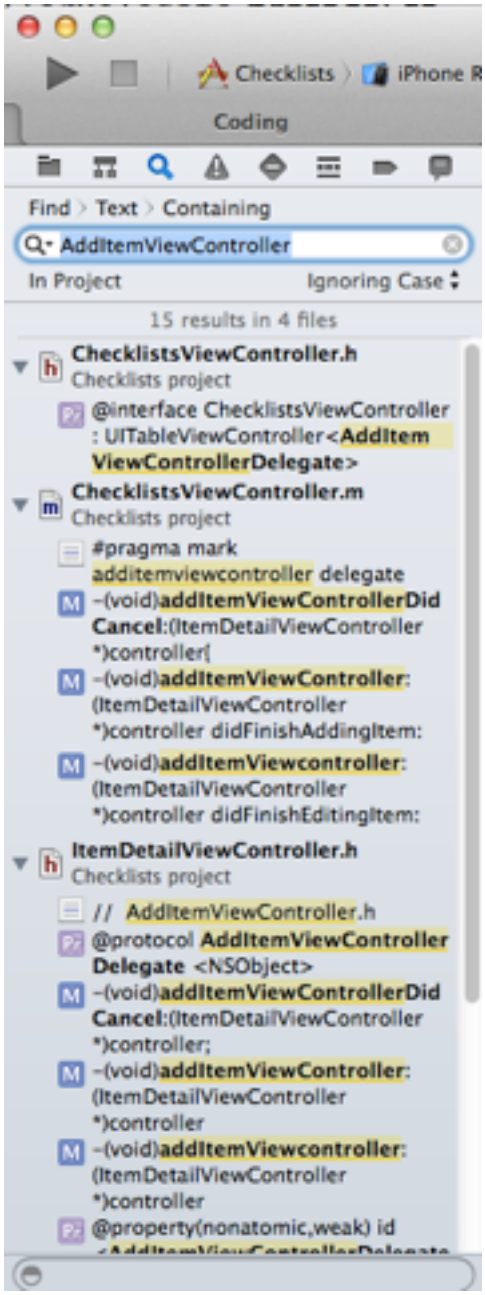
在必要的地方创建快照有时候是有必要的，不过这个功能更多是在单枪匹马开发时候用的比较多。如果是团队合作，或者是需要频繁修改的情况下，我们更多会用到版本控制(source control)。目前在iOS中主要使用git版本控制方式，关于这一点我们会在随后的教程中详细介绍。

继续回到我们的代码重构环节。

刚才我们只是替换了AddItemViewController这个类，还有一个AddItemViewControllerDelegate协议的名称也需要修改为ItemDetailViewControllerDelegate。当然，这不是必须的，只是这样做会让你的代码结构更加清晰易懂易维护。

不过遗憾的是Xcode并不能像刚才修改类名一样自动重命名代理协议的方法名称，我们需要手动完成这项工作。

在Xcode左侧的面板中切换到Search navigator,然后输入：addItemViewController，此时Xcode将自动搜索整个项目中包含该文本的内容：



我们需要做以下更改：
把addItemViewControllerDidCancel: 更改为itemDetailViewControllerDidCancel:
把addItemViewController:更改为itemDetailViewContyroller:

在做完这些更改后，ItemDetailViewController.h中的@protocol协议将拥有以下方法：

```
@protocol ItemDetailViewControllerDelegate <NSObject>
 -(void)itemDetailViewControllerDidCancel:(ItemDetailViewController *)controller;

- (void)itemDetailViewController:(ItemDetailViewController *)controller
        didFinishAddingItem:(ChecklistItem *)item;

- (void)itemDetailViewController:(ItemDetailViewController *)controller
        didFinishEditingItem:(ChecklistItem *)item;

@end
```

与此同时，我们还需要更改ItemDetailViewController.m和ChecklistsViewController.m中的方法名称。
在更改的过程中，最好重复这个search搜索过程，以确保没有遗漏。

全部更改完成后，记住使用command+B来编译，确保没有问题。

此外，因为我们在这个过程中做了很多改动，最好进行一次clean清理，确保Xcode完成了所有的更改，并且找到可能的警告或编译错误。
从Xcode的菜单中找到Product- Clean，完成Clean后再次使用Product -Build(或者command+B)来编译。如果一切无误，再次按command+R编译运行，确保一切都可以正常工作！

注意：
如果你在更名的过程中出现了错误，很有可能是因为大小写错误引起的。

好了，这一章的内容就这么多，很轻松吧~

在结束这一章的学习之前，我们来了解一个概念-迭代开发

如果你是个产品经理，那么对所谓迭代开发的概念肯定不会陌生。在互联网和移动互联网产品的开发中，经常用到迭代开发的概念，更确切的说是快速迭代开发。

简单来说，在产品经理+码农+UI/UX的共同努力下，用尽可能短的时间出一个产品原型，然后根据用户的反馈来不断完善，这就是快速迭代开发。

在软件开发领域，有两种开发模式，分别是自顶向下的规划模式和与它相反的迭代开发模式。如果只看概念你可能不好理解，我们还是来打个比方。

自顶向下的规划模式类似于我朝20多年前的计划经济，一切都在蓝图上规划好，事无巨细，等到一切都计划完美了才投入开发。而迭代开发模式更类似于市场经济，大家都是摸着石头过河，每出一个版本都要经过市场和用户的检验。

这两种模式孰优孰劣？其实没有那么绝对。通常来说，对于企业和机构市场这种2B的商务软件，采用自顶向下的设计和开发模式更常见。因为企业和机构市场所用的软件更看重功能和架构的稳定性，不一定单纯追求效率和用户体验。而如果是面向普通消费者开发软件，那么快速迭代开发是比较适合的，因为消费者的心理需求很难把握，而且是个动态变化的过程，唯有不断完善改进。

此外，在实际的开发过程中，通常是计划和快速迭代并举的方式。

简单规划-快速迭代开发–根据反馈修正规划–再次迭代开发–。。。

这是个永不停止的过程。

好了，今天的内容相对比较少，大家可以喘口气，等待下一章的继续爆发。

福利送上，休息一下吧。