从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter18

欢迎继续我们的学习。

开发环境：
Xcode 5.0.2 +iOS 7.0.4
注明：在试用了几天Xcode5.1PR2+iOS7.1 beta之后发现有一些问题，为了不影响正常的项目开发和入门者可能遇到的问题，暂时切换回正式版的Xcode5.0.2+iOS7.0.4

为神马我们这里的保存方法会失效呢?

在此之前，每当用户对数据进行更改的时候都会保存数据：添加一个新项目，删除一个项目，或是切换勾选标志的显示状态。所有这些都是在Checklist View Controller上发生的。
但是现在我们已经将保存数据的逻辑处理放到了AllListsViewController里面。好吧，现在如何让ChecklistViewController里面所发生的变化得以保存?

或许你会想到一个办法，比如在AllListsViewController里面提供一个到ChecklistViewController的引用，然后当用户更改数据信息的时候让它调用saveChecklists方法。不过这样一来就产生了所谓的child-parent dependency（双重依赖）问题，而这正是我们一直极力想避免出现的情况。

那么好吧，这个时候我们是否可以考虑使用代理？当然，如果你这么想我会感到非常自豪，不过现在我们需要重新思考在应用中保存数据信息的策略。

有一点需要反思的是，是否需要在应用中随时保存信息？当应用还在运行的时候，数据模型一直在临时工作内存中，且始终是最新的状态。只有当应用启动的时候我们才需要从文件（也就是应用沙盒所在的长期内存）中加载数据信息，此后就再也不需要了。然后我们就可以在临时工作内存中对数据信息进行调整。但是当改动完成之后，之前的文件就过时了。此时我们需要考虑来保存这些变动-从而让沙盒中的文件和临时工作内存中的信息保持一致。

之所以我们需要将信息保存到文件中，是因为当应用被终止后我们仍然可以从长期内存中恢复所保存的数据模型。不过除此之外，保存在临时工作内存中的数据信息就足够我们在应用中使用了。因此，我们唯一需要明确的就是在应用被终止前将数据保存到沙盒的文件中。换句话说，只有当我们需要保证数据安全的时候才需要进行这种操作。
这样会让应用更加高效，特别是当应用中要处理大量的数据时更是如此。同时这种策略也会大大减少代码量。我们不再需要在用户对数据信息进行更改后随时保存数据，而只需要在应用被终止前进行一次这种操作。

那么什么情况下应用会被彻底终止呢？（不是关闭，而是终止）

1.当用户在运行应用的时候。
在这种情况下应用被终止非常罕见。早期的iOS版本并不支持多任务，比如当应用接听到电话时可能就会干掉当前正在运行的应用。在iOS4和之后的版本中，此类情况下应用只是悬停在后台。不过即便在最新的iOS操作系统中仍然可能发生这种情况，比如当应用始终没有响应，或是当系统内存不足的时候，很可能该应用就会成为牺牲品被追求用户体验的iOS系统干掉。

2.当应用悬停在后台的时候。
大多数时候iOS都会让应用在后台悬停。应用的相关数据被冻结在内存中，也不会发生什么计算。当用户打开一个被悬停的应用时，通常会从被中断的地方继续。

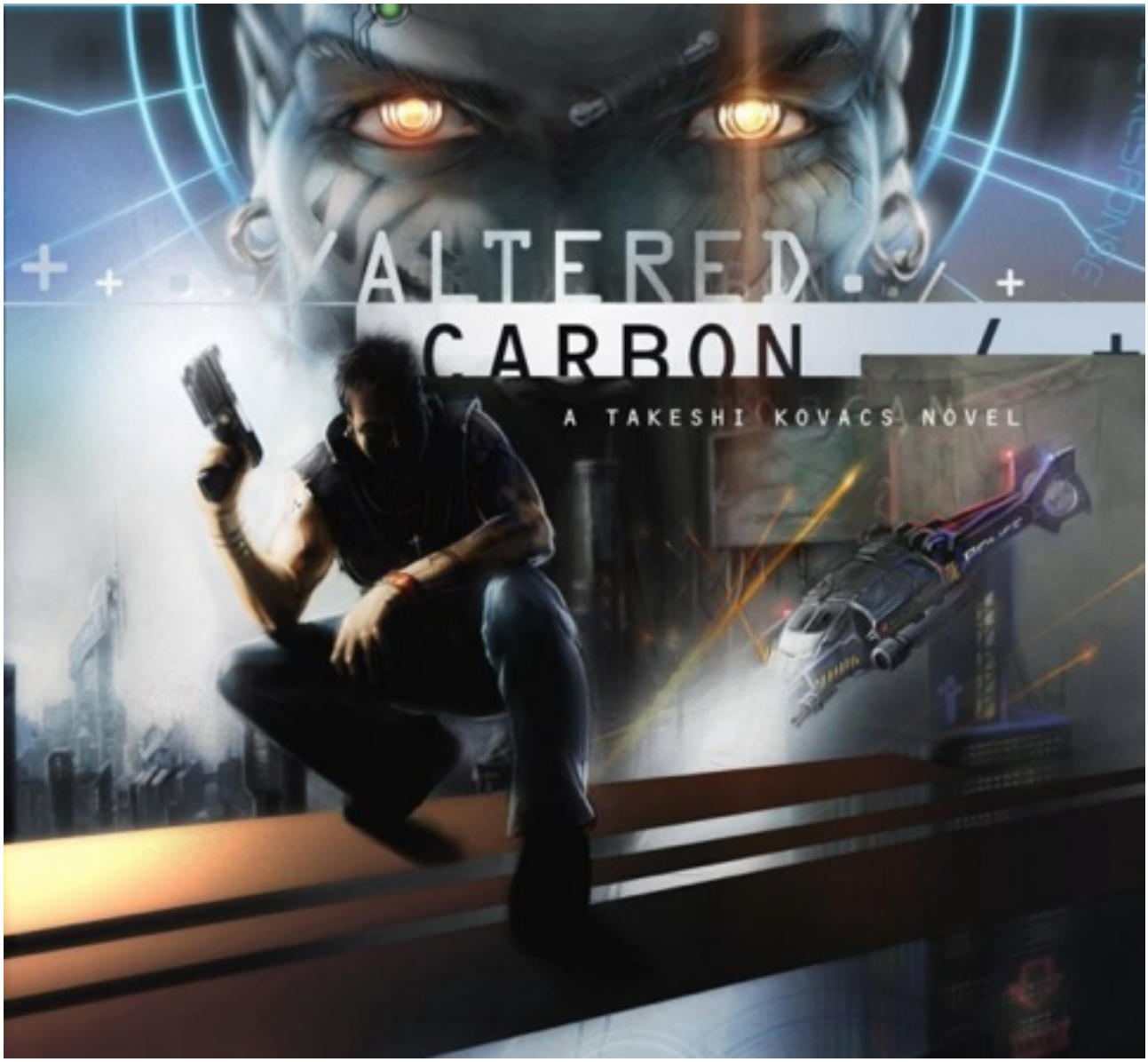
不过当iOS需要为某个占用大量工作内存的应用（通常是比较耗费资源的游戏）开路时，就会毫不犹豫的将某些悬停应用从内存中清除。

3.应用自己崩溃了

虽然有很多方式来监测应用的崩溃，但处理这些东西会非常的恼火。事实上有时候耗费大量时间精力来处理这些崩溃只会让事情更糟糕。避免应用崩溃的最好方式就是避免出现任何的编码错误。

不过iOS的开发者是幸福的，因为当出现以下变化时系统会主动通知应用：
比如当前应用将被终止时，以及当前应用将被转入后台悬停时。我们可以监听这些事件，然后在必要的时候保存数据。这样就可以保证我们的数据模型永远是最新的，直到应用被干掉的那一刻

前段时间看过一篇科幻小说Altered Carbon，在小说的故事背景设定中，所有的人类都实现了永生。这种永生并非是指身体的永远年轻，而是对记忆和意识的完美复制。所有人在出生之后都在大脑中放入了一个高科技设备，这种设备在平时会定时（每天入睡的时候）对人的记忆和意识（情绪情感认知）的神经连接进行复制，并通过脑联网上传到一个中央节点上。当发生意外情况可能导致载体-承载当前思维的身体陨灭的时候（比如各种天灾人祸），设备会紧急对此前的记忆和意识进行完美复制，并实时上传。只需要自己在银行还有富余的重生点数，就可以让自己购买一部克隆体将思维拷贝到其中。通过这种方式，人类彻底摆脱了对死亡的恐惧。当然，准备的说是土豪们彻底摆脱了对死亡的恐惧。不过也因此引发了很多伦理道德问题。。。



想象应用就是身体，而数据模型就是记忆和意识。当应用因为意外被干掉的那一刻，我们幸运的通过iOS系统把记忆和意识复制并上传到沙盒中。假如我们需要且有足够的重生点数，就可以在某一刻重新找到一个载体，然后将之前的数据模型加载。
对于此类消息通知，一个最理想的场所莫过于application delegate。我们此前并没有深入了解过这个对象，但事实上它是iOS开发中值得高度重视的一环，因为任何一个iOS应用或游戏都有这样一个类。正如它的名字所暗示的，它是一个代理对象，其主要作用是对所有和应用整体相关的消息做出响应。正是在这里我们就收到"应用将被终止"以及"应用将被悬停"的通知。

好了，一大堆的理论知识和八卦之后，终于到了代码时间。

打开Xcode，切换到ChecklistsAppDelegate.m，可以看到里面有如下两个方法：

```
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
}
```

当然里面还有一大堆其它方法，不过这两个是我们目前最需要的。在里面的注释行说明中，我们也可以清楚的看到苹果工程师的友情提示，实在是业界良心~

问题来了，我们如何从这些代理方法中调用AllListsViewController的saveCheckList方法？目前app delegate对AllListsViewController还一无所知呢。这里我们需要用到一些小小的技巧。

首先在ChecklistsAppDelegate.m的顶部添加一行代码（记得用快捷键command+向上的箭头跳转到文件的顶部）：

```
#import "AllListsViewController.h"
```

在applicationDidEnterBackground方法上面添加以下的新方法：

```
-(void)saveData{

    UINavigationController *navigationController = (UINavigationController*)self.window.rootViewController;
    AllListsViewController *controller = navigationController.viewControllers[0];
    [controller saveChecklists];

}
```

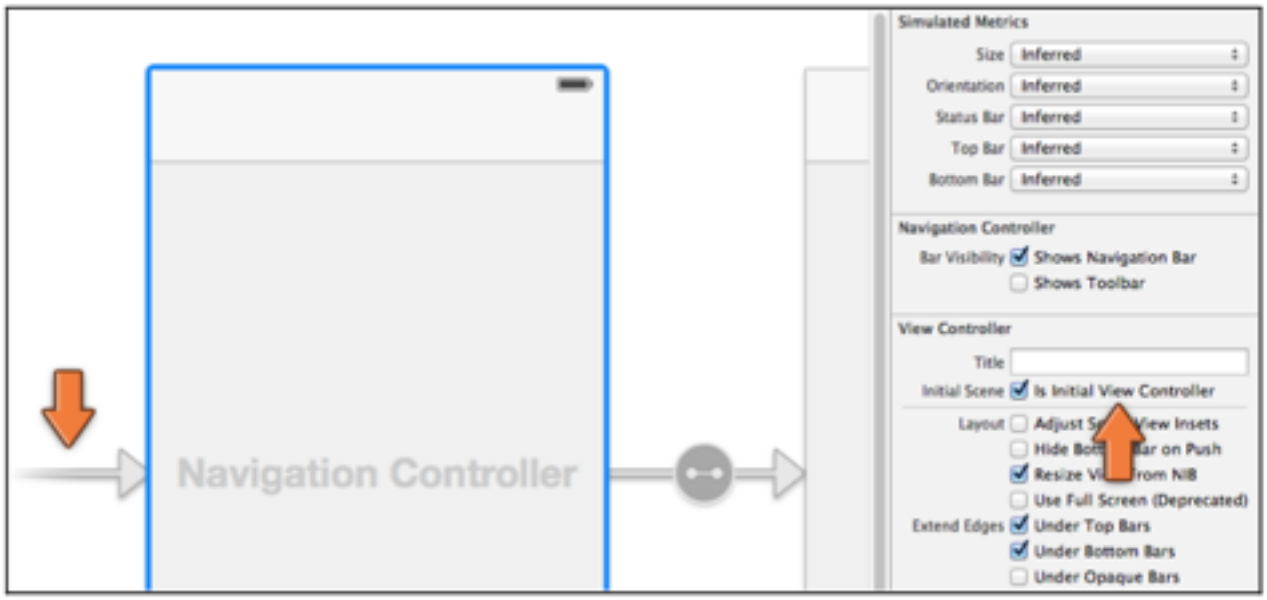
接下来更改applicationDidEnterBackground和applicationWillTerminate方法：

```
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    [self saveData];
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    [self saveData];
}
```

在saveData方法中，会根据self.window属性找到包含了storyboard的UIWindow对象。UIWindow是应用所有视图的总容器。每个应用中只有一个UIWindow对象（和桌面应用不同，通常有多个窗口对象，iOS应用中有且只有一个UIWindow对象！！！）

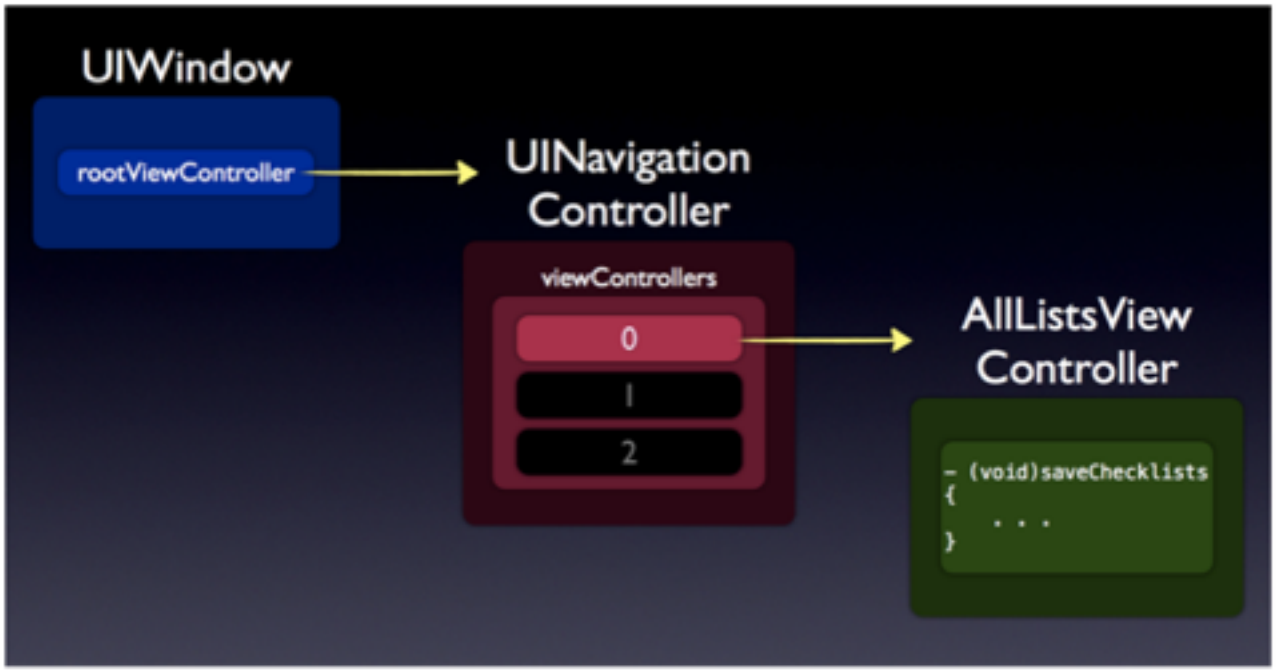
通常情况下我们不需要对UIWindow进行任何处理，不过这里我们需要获取它的rootViewController



属性。在storyboard中是第一个视图控制器，也就是最左边的导航控制器。在Interface Builder中我们可以看得一清二楚，因为这个导航控制器的Is Initial View Controller属性总是处于勾选状态，而且在最左侧有个大大的箭头指向它。

当我们获取了到首个导航控制器的引用，就有望找到AllListsViewController，然后调用它的saveChecklists方法。不幸的是，UINavigationController没有一个名为rootViewController的属性，因此我们需要在其viewController数组中找到最开始的那个。

UINavigationController其实还有一个topViewController属性，但这里却没有什么用。因为"top"view controller是当前正在显示的界面，而这个界面却很有可能是ChecklistViewController。显然我们不能将saveChecklists消息发送到那个界面，因为它没有方法处理这个消息，最终只会导致应用崩溃。



通过上图可以看到，在UINavigationController的viewControllers数组中的第一个就是AllListsViewController，也正是我们想要的。

```
#pragma mark save data
-(void)saveData{

    UINavigationController *navigationController = (UINavigationController*)self.window.
        rootViewController;
    AllListsViewController *controller = navigationController.viewControllers[0];
    [controller saveChecklists];
```

No visible @interface for 'AllListsViewController' declares the selector 'saveChecklists'

不过现在还是有个问题，Xcode给了个错误提示：“No visible @interface for “AllListsViewController” declares the selector ‘saveChecklists’”。只不太科学啊，我们在AllListsViewController对象中不是已经添加了saveChecklists方法了吗?

好吧，之前我们并没有讨论过interface和implementation的区别，但实际上一个对象在外的表现和它在家的表现是有所区别的。毕竟，对象和人一样，都是爱面子的。通常对外所展示的都是自己最好的一面，家丑不可外扬是所有生命体的共性。
实际上一个对象的内部实现细节对外部使用者来说并不重要，而且我们通常尽可能的隐藏对象的细节，对外只提供一些接口。

例如，实例变量最好在.m的@implementation部分声明，因为我们不希望让外部的使用者也了解这些信息。
简单来说，.h文件就是一个对象的接口，而.m则是具体的实现细节。

为了让其它对象可以使用saveChecklists方法，我们需要在.h中添加一个声明。

在Xcode中切换到AllListsViewController.h，然后添加一行代码：
-(void)saveChecklists;

此时AllListsViewController.h中的代码内容如下：

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

@interface AllListsViewController : UITableViewController<ListDetailViewControllerDelegate>


-(void)saveChecklists;

@end
```

因为我们已经将saveChecklists方法添加到了对象的@interface部分，其它对象现在就可以使用它了。

编译运行应用，添加一些checklist,然后往其中添加一些事项，再设置一下勾选标志。在simulator中按Home键，让应用回到后台。

使用finder查看应用的沙盒文件夹，现在里面有一个新的.plist文件在里面了。

点击Xcode的Stop来终止应用。再次编译运行应用，好了，这下所有的数据都还在里面。
搞定收工！

关于Xcode的Stop按钮

特别注意：当你按下Xcode上的Stop按钮时，application delegate并不会收到applicationWillTerminate的消息通知。Xcode会毫不犹豫的斩杀这个应用。因此，为了测试保存功能，首先应该按下simulator上的Home键，然后再按Stop。如果不是这样，数据就会丢失。

到目前为止，代码基本可以正常工作，但我们还可以做得更好。我们已经创建了Checklist和ChecklistItem的数据模型对象，不过在AllListsViewController中仍然有加载和保存到.plist文件的代码。实际上这也是数据模型的职责。

好了，今天的学习到此结束了。

老习惯，福利送上~今天是可爱的动物，天天看美女也会烦的。