

从零开始学iOS7开发系列教程-事务管理软件开发实战-Chapter13

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>)。

欢迎继续我们的学习。

开发环境：

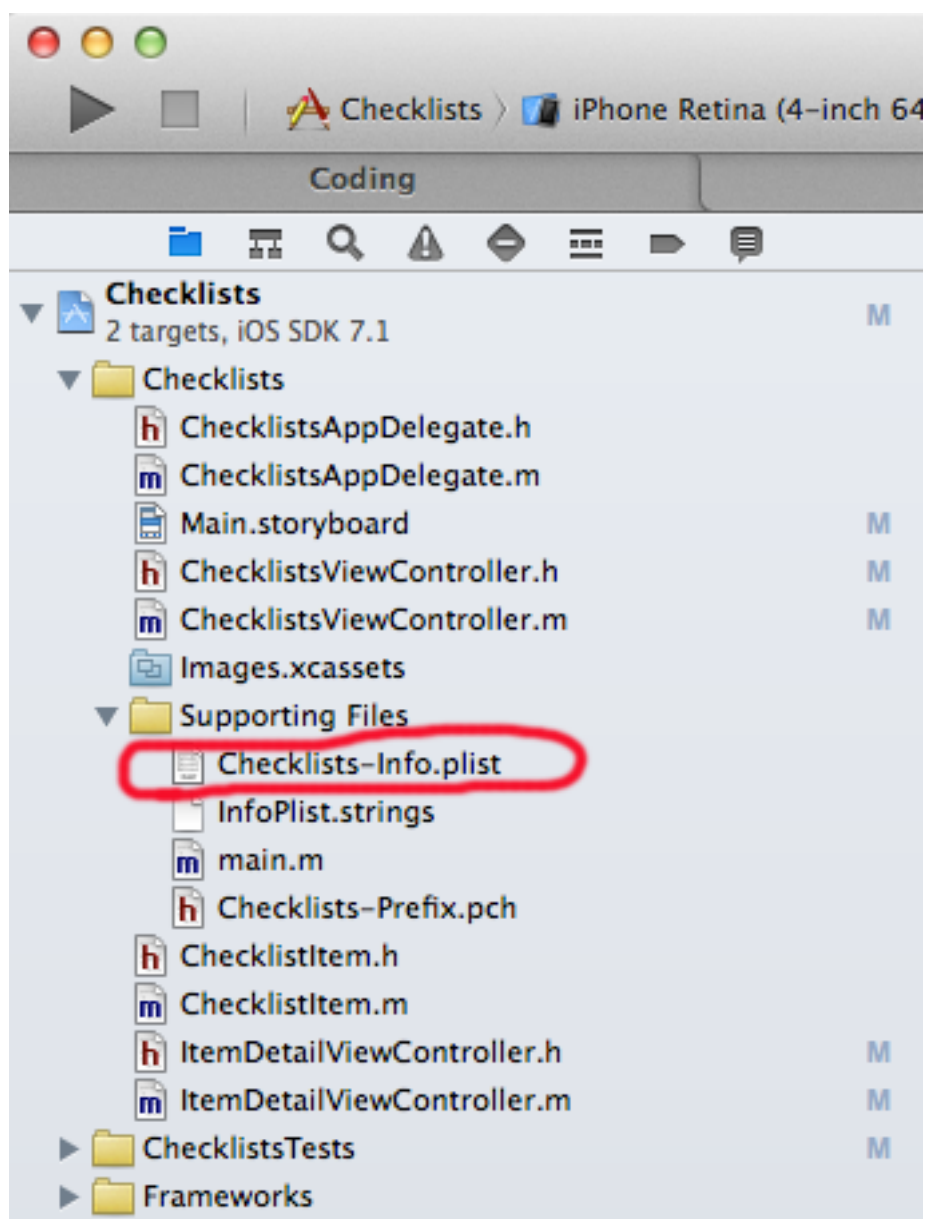
Xcode 5.1 DP 2 + iOS 7.1 beta2

在上一章的学习中，我们简单了解下iOS开发的沙盒机制。接下来我们需要真正的编写代码，从而当用户添加一个新的代办事务，或者编辑一个当前代办事务的时候，把代办事务清单保存到一个Checklists.plist文件中。此外，当我们完成了保存的任务后，当然还需要编写代码来加载这个清单。

好吧，什么是.plist文件。

在上一系列的教程中，我们曾经接触过Info.plist文件。实际上所有的iOS项目中都有这样一个plist文件，包括我们的这个Checklists应用。

在Xcode 中的项目导航部分，可以看到在Supporting Files群组中有一个Checklists-Info.plist



Plist其实是Property List（属性列表）的缩写，它的本质是一个XML文件，其中保存了结构化的数据，通常是以一系列的键对值的形式存在的。应用的xxx-Info.plist(xxx就是项目名称)文件中保存了该应用的一些重要设置信息，在上一系列的教程中有提到过，不过这里再次复习下吧。

科普：关于Info.plist文件，XML

在任何一个iOS应用或游戏中，如同AppDelegate类一样，都有一个以项目名称开头的Info.plist文件，比如这里的CrazyDrag-Info.plist文件。它的内容通常由三列组成，最左边是Information Property List（属性列表），中间是Type（属性值的类型），而最右边则是Value（属性值）。

Info.plist文件其实是一个XML文档。XML其实就是可扩展性标记语言(extensible markup language)，它并非iOS中所特有的，在几乎任何一种编程语言的使用过程中，我们都会碰到XML文档。XML是所谓标准通用标记语言(SGML)的子集，其作用是以规范的形式（成对出现的标记）来保存数据。XML与传统的Access,Oracle,SQL Server, MySQL数据库不同。传统的数据库功能强大，提供了强大的数据存储和分析能力，而XML仅仅用来存储数据，需要自行编写代码来进行数据的分析和处理。但XML的好处是它超级简单易用，可以在任何语言编写的任何应用程序中读写数据，已经成了网络数据交互的唯一公共语言。

你可能不知道XML文档，但或许多半听说过HTML文档吧。XML文档只不过是HTML文档的规范式表达。它们的区别在于，XML的核心是数据内容本身，而HTML的核心是如何显示数据。

plist文件的本质就是XML文档，只不过其中的内容都和iOS应用的相关设置有关。在Xcode中右键单击CrazyDrag-Info.plist,选中open as,选中source code，就可以看到下面的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
    <key>CFBundleDisplayName</key>
    <string>${PRODUCT_NAME}</string>
    <key>CFBundleExecutable</key>
    <string>${EXECUTABLE_NAME}</string>
    <key>CFBundleIdentifier</key>
    <string>com.ioslearning.${PRODUCT_NAME:rfc1034identifier}</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>${PRODUCT_NAME}</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>1.0</string>
    <key>CFBundleSignature</key>
    <string>????</string>
    <key>CFBundleVersion</key>
```

```

<string>1.0</string>
<key>LSRequiresiPhoneOS</key>
<true/>
<key>UIMainStoryboardFile</key>
<string>Main</string>
<key>UIRequiredDeviceCapabilities</key>
<array>
    <string>armv7</string>
</array>
<key>UIStatusBarHidden</key>
<true/>
<key>UISupportedInterfaceOrientations</key>
<array>
    <string>UIInterfaceOrientationLandscapeLeft</string>
    <string>UIInterfaceOrientationLandscapeRight</string>
</array>
<key>UIViewControllerBasedStatusBarAppearance</key>
<false/>
</dict>
</plist>

```

很显然，plist文档其实就是满足苹果DTD标准的XML文档。

那么，Info.plist文件中这些键值的作用是什么呢？

这里大概说明了一下，更详细的可以参考苹果的官方文档，也可以参考（<http://www.biancheng521.com/article/enauola/7853130.html>）。

Localization native development region --- CFBundleDevelopmentRegion 本地化相关，如果用户所在地没有相应的语言资源，则用这个key的value来作为默认。

Bundle display name --- CFBundleDisplayName 设置程序安装后显示的名称。应用程序名称限制在10- 12个字符，如果超出，将被显示缩写名称。

Executable file -- CFBundleExecutable 程序安装包的名称

Bundle identifier --- CFBundleIdentifier 该束的唯一标识字符串，该字符串的格式类似com.yourcompany.yourapp，如果使用模拟器跑你的应用，这个字段没有用处，如果你需要把你的应用部署到设备上，你必须生成一个证书，而在生成证书的时候，在apple的网站上需要增加相应的app IDs.这里有一个字段Bundle identifier，如果这个Bundle identifier是一个完整字符串，那么文件中的这个字段必须和后者完全相同，如果app IDs中的字段含有通配符*，那么文件中的字符串必须符合后者的描述。

InfoDictionary version --- CFBundleInfoDictionaryVersion Info.plist格式的版本信息

Bundle name ---CFBundleName产品名称

Bundle OS Type code -- CFBundlePackageType: 用来标识束类型的四个字母长的代码，

Bundle versions string, short --- CFBundleShortVersionString 面向用户市场的束的版本字符串

Bundle creator OS Type code --- CFBundleSignature: 用来标识创建者的四个字母长的代码

Bundle version --- CFBundleVersion 应用程序版本号，每次部署应用程序的一个新版本时，将会增加这个编号，在app store上用的。

Application require iPhone environment -- LSRequiresiPhoneOS:用于指示程序包是否只能运

行在iPhone OS 系统上。Xcode自动加入这个键，并将它的值设置为true。您不应该改变这个键的值。

Main storyboard file base name -- UIMainStoryboardFile 这是一个字符串，指定应用程序主nib文件的名称。

supported interface orientations -- UIInterfaceOrientationOrientations 程序默认支持的方向。

Plist文件的作用不仅仅限于此，在iOS开发的过程中，我们还经常使用它来保存一些重要的数据信息，比如在当前的这个应用中我们将用它来保存代办事务清单。有的游戏项目会使用plist文件存放关卡设置。相比其它数据存储形式（比如sql数据库），plist使用起来非常方便，对于一些简单的数据信息用它就足够了。

为了保存Checklist 中的代办事项，我们需要用到NSCoder系统，使用它可以让对象用结构化的文件格式来保存数据。就目前来说，我们还没有必要过多关注文件格式的问题。这里我们只需要知道将使用.plist文件来保存数据，而这些数据将保存在应用的沙盒Documents文件夹中。至于其它的技术细节，NSCoder将帮你来搞定。

NSCoder究竟是神马东东？

其实我们在不知不觉中已经和NSCoder打过交道了，因为storyboards的工作就是基于NSCoder的。当我们往storyboard里面添加一个视图控制器的时候，Xcode会使用NSCoder系统将这个对象写入到文件中（encoding编码）。而当应用启动的时候，它会再次使用NSCoder从storyboard文件中读取对象（decoding解码）。这个将对象写入到文件中再转换回来的过程被称之为“序列化”(serialization)。

好吧，这样说来有点太抽象了，还是来打个比方吧。

你可以把这个过程看做用冰柜冷冻食品。为了保证食品的新鲜，我们会把食品放入冰柜，在那里它们被冷冻起来，冰柜就好比文件。当我们需要的时候可以打开冰柜（读取文件），然后解冻这些食品，就可以再次使用了。

理论知识充电到此结束，该是动手的时候了。

在Xcode中切换到ChecklistsViewController.m，然后在dataFilePath: 方法之后添加以下方法：

```
-(void)saveChecklistItems{  
  
    NSMutableData *data = [[NSMutableData alloc] init];  
    NSKeyedArchiver *archiver = [[NSKeyedArchiver  
alloc] initWithMutableData:data];  
    [archiver encodeObject:_items forKey:@"ChecklistItems"];  
    [archiver finishEncoding];  
    [data writeToFile:[self dataFilePath] atomically:YES];  
  
}
```

看到以上代码，似乎令人望而生畏，毕竟一行都没碰到过。

其实这个方法的作用很简单，它会获取_items 数组中的内容，然后分两步将它转换为二进制数据块，然后把数据写入到文件中：

1.NSKeyedArchiver，是用于创建plist文件的NSCoder的一种形式，它可以对数组进行编码，然后将所有的ChecklistItems写入到某种二进制数据格式中。

2.数据会被保存在一个NSMutableData对象中，它可以将自身写入到通过dataFilePath所获取的完整路径的文件中。

如果你想了解NSKeyedArchiver的内部工作原理，目前还稍微有点早。现在我们需要知道的是，它可以允许我们将对象保存到一个文件中，然后在需要的时候可以读取出来。

每当我们修改了待办事项清单时d0ou需要调用这个新的saveChecklistItems方法，实际上也就是在ItemDetailViewControllerDelegate的协议方法中。

在Xcode中切换到ChecklistsViewController.m，然后更改以下方法的代码（主要是添加对saveChecklistItems方法的调用）：

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

```
    NSInteger newRowIndex = [_items count];  
    [_items addObject:item];
```

```
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:newRowIndex inSection:0];
```

```
    NSArray *indexPaths = @[indexPath];  
    [self.tableView insertRowsAtIndexPaths:indexPaths  
withRowAnimation:UITableViewRowAnimationAutomatic];  
    [self saveChecklistItems];
```

```
    [self dismissViewControllerAnimated:YES completion:nil];  
}
```

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

```
    NSInteger index = [_items indexOfObject:item];
```

```
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:index inSection:0];
```

```
    UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
```

```
    [self configureTextForCell:cell withChecklistItem:item];  
    [self saveChecklistItems];  
    [self dismissViewControllerAnimated:YES completion:nil];
```

```
}
```

别忘了还要在swipe-to-delete删除当前事项的方法中添加对该方法的调用：

```
-(void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath{

    [_items removeObjectAtIndex:indexPath.row];
    [self saveChecklistItems];

    NSArray *indexPaths = @[indexPath];

    [tableView deleteRowsAtIndexPaths:indexPaths
    withRowAnimation:UITableViewRowAnimationAutomatic];

}
```

还有开启关闭勾选标志的方法中也需要：

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

    UITableViewCell *cell =[tableView cellForRowAtIndexPath:indexPath];
    ChecklistItem *item = _items[indexPath.row];
    [item toggleChecked];

    [self configureCheckmarkForCell:cell withChecklistItem:item];
    [self saveChecklistItems];

    [tableView deselectRowAtIndexPath:indexPath animated:YES];
}
```

就这么简单？！

答案是：当然不是，仅仅对_items数组调用NSKeyedArchiver方法还不够。假如我们现在编译运行应用，然后试着触碰某一行来切换勾选状态，oops，应用会崩溃的，然后会提示以下信息：

```
*** Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '-
[ChecklistItem encodeWithCoder:]: unrecognized selector sent to instance 0x6a26810'
```

此时Xcode的debugger会指向出错的那行代码：

```
[archiver encodeObject:_items forKey:@"ChecklistItems"];
```

之前我们接触过“unrecognized selector”这条错误信息。它意味着我们忘了实现某个特定方

法。而在这里根据错误信息提示，我们似乎忘了实现ChecklistItem对象上的encodeWithCoder方法。

到底是神马情况呢？

答案是：这里我们要求NSKeyedArchiver对数组中的代办事项编码，因此它不仅需要知道如何对数组本身编码，还需要知道如何对每个ChecklistItem对象进行编码。遗憾的是，目前NSKeyedArchiver只知道如何对一个NSMutableArray对象编码，而对ChecklistItem一无所知。肿么办？当然是伟大万能的程序猿出马的时刻到了。

在Xcode中切换到ChecklistItem.h，然后更改@interface这行代码；

```
@interface ChecklistItem : NSObject <NSCoding>
```

记住<>标示该对象遵从某个特定的协议。这里我们让ChecklistItem对象遵从NSCoding协议。

接下来当然是要实现协议的某个方法了。

切换到ChecklistItem.m，然后添加下面的方法：

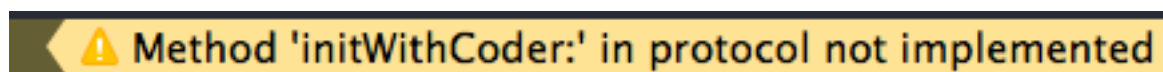
```
-(void)encodeWithCoder:(NSCoder *)aCoder{  
  
    [aCoder encodeObject:self.text forKey:@"Text"];  
    [aCoder encodeBool:self.checked forKey:@"Checked"];  
  
}
```

这个方法就是unrecognized selector error信息中所缺失的方法。当NSKeyedArchiver尝试对某个ChecklistItem对象编码时，就会发送encodeWithCoder消息。

通过以上方法，我们就会告诉别人：一个ChecklistItem有一个名为“Text”的对象，其中包含了self.text的属性，还有一个名为“Checked”的布尔值，其中包含了self.checked的属性。这两行代码就足以让coder系统恢复正常了。

再次编译运行应用，尝试触碰某个行来开启关闭勾选标志。看看情况肿么样？Oh yeah,不再崩溃了！！

注意：此时Xcode还是给了你一条警告信息，

A yellow warning banner from Xcode with a yellow triangle icon on the left. The text inside the banner reads: "Method 'initWithCoder:' in protocol not implemented".

别害怕，现在还别管它，很快我们就会解决这个问题。

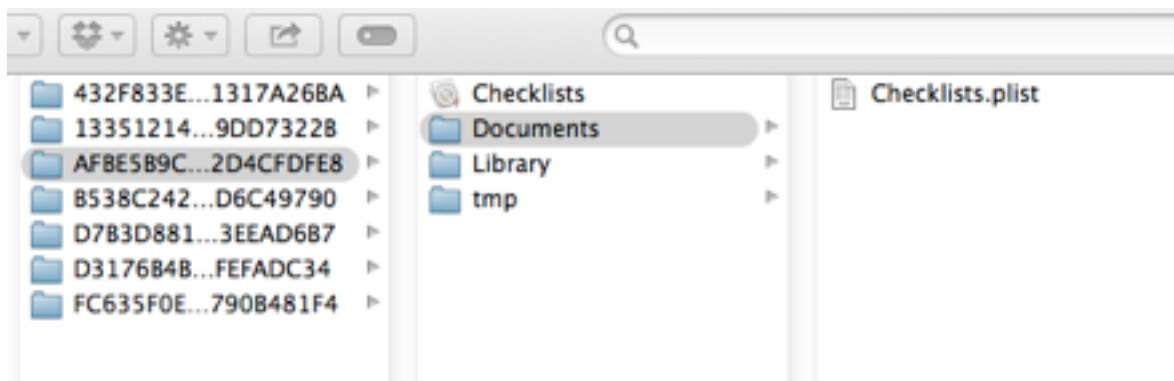
好了，在Mac系统中打开Finder，然后跳转到应用的Documents所在的目录。神马？忘了在哪里？？！！好吧，你击败了我！

这个时候console里面的信息应该还在，比如：

2013-12-16 10:39:55.706 Checklists[2133:70b] 文件夹的目录是： /Users/happybubsy/Library/Application Support/iPhone Simulator/7.1-64/Applications/B157795E-E647-4F48-8F12-E46A8E737385/Documents

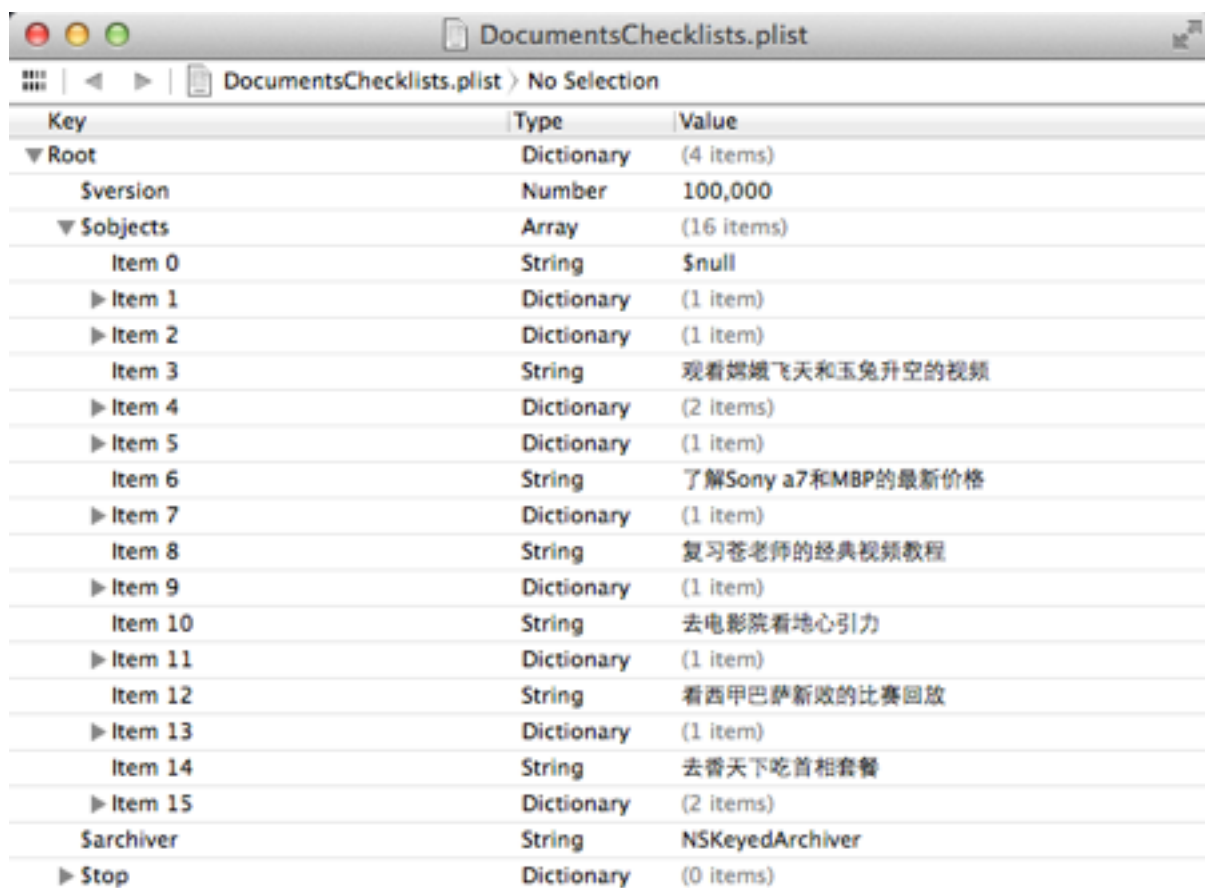
2013-12-16 10:39:55.709 Checklists[2133:70b] 数据文件的最终路径是： /Users/happybubsy/Library/Application Support/iPhone Simulator/7.1-64/Applications/B157795E-E647-4F48-8F12-E46A8E737385/DocumentsChecklists.plist

打开Finder，点击Go-Go to Finder,输入类似的信息（每个电脑上都不一样!!!）：
/Users/happybubsy/Library/Application Support/iPhone Simulator/7.1-64/Applications/B157795E-E647-4F48-8F12-E46A8E737385/
此时你会看到类似下面的：



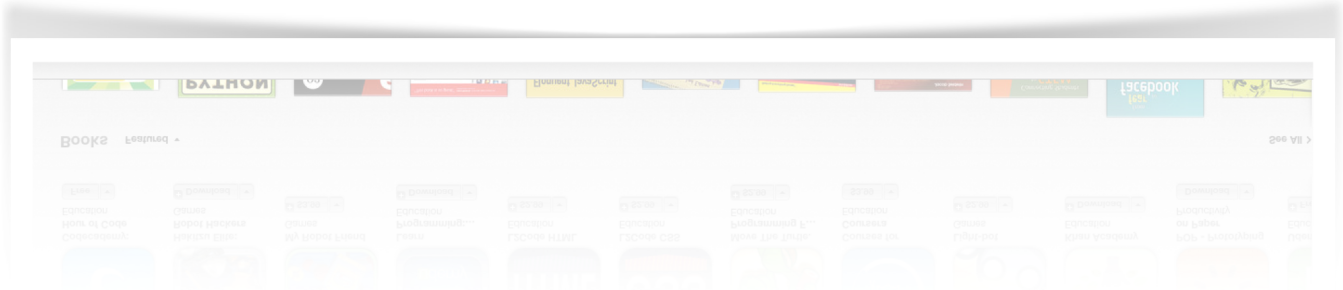
可以看到这里生成了一个名为DocumentsChecklists.plist的文件。
用文本编辑器打开这个文件的内容，可以看到其中的内容，好吧，惨不忍睹，根本就不是给人看的！
是的，虽然它是XML格式的文件，但也不是你我小小人类可以轻易读懂的，它本来就是给
NSKeyedArchiver系统来看的。

当然，除了用文本编辑器，我们也可以右键单击这个文件，然后选择Open with -Xcode
这样看起来稍微有点人性，没那么变态了。



Key	Type	Value
▼ Root	Dictionary	(4 items)
\$version	Number	100,000
▼ Subjects	Array	(16 items)
Item 0	String	\$null
▶ Item 1	Dictionary	(1 item)
▶ Item 2	Dictionary	(1 item)
Item 3	String	观看嫦娥飞天和玉兔升空的视频
▶ Item 4	Dictionary	(2 items)
▶ Item 5	Dictionary	(1 item)
Item 6	String	了解Sony a7和MBP的最新价格
▶ Item 7	Dictionary	(1 item)
Item 8	String	复习苍老师的经典视频教程
▶ Item 9	Dictionary	(1 item)
Item 10	String	去电影院看地心引力
▶ Item 11	Dictionary	(1 item)
Item 12	String	看西甲巴萨新败的比赛回放
▶ Item 13	Dictionary	(1 item)
Item 14	String	去香天下吃首相套餐
▶ Item 15	Dictionary	(2 items)
\$archiver	String	NSKeyedArchiver
▶ Stop	Dictionary	(0 items)

老美最近在搞The Hour of Code 2013,别忘了去下载这些应用哦!





美女福利也不可少，嫦娥和玉兔都回到月球了，元帅也出现了

