**一、Podfile.lock文件**

上文讲过，在开始使用CocoaPods，执行完pod install之后，会生成一个Podfile.lock文件。这个文件看起来跟我们关系不大，实际上绝对不应该忽略它。

该文件用于保存已经安装的Pods依赖库的版本，通过CocoaPods安装了SBJson、AFNetworking、Reachability三个POds依赖库以后对应的Podfile.lock文件内容为：

PODS:

- AFNetworking (2.3.1):

- AFNetworking/NSURLConnection

- AFNetworking/NSURLSession

- AFNetworking/Reachability

- AFNetworking/Security

- AFNetworking/Serialization

- AFNetworking/UIKit

- AFNetworking/NSURLConnection (2.3.1):

- AFNetworking/Reachability

- AFNetworking/Security

- AFNetworking/Serialization

- AFNetworking/NSURLSession (2.3.1):

- AFNetworking/Reachability

- AFNetworking/Security

- AFNetworking/Serialization

- AFNetworking/Reachability (2.3.1)

- AFNetworking/Security (2.3.1)

- AFNetworking/Serialization (2.3.1)

- AFNetworking/UIKit (2.3.1):

- AFNetworking/NSURLConnection

- AFNetworking/NSURLSession

- FXReachability (1.1.1)

- GCJSONKit (1.5.0)

DEPENDENCIES:

- AFNetworking (~> 2.3.1)

- FXReachability (~> 1.1.1)

- GCJSONKit (~> 1.5.0)

SPEC CHECKSUMS:

AFNetworking: 6d7b76aa5d04c8c37daad3eef4b7e3f2a7620da3

FXReachability: 34c68085c9a12c7fe4f791bdb54e99f14d90b5ae

GCJSONKit: 2e66dba7183b1557f01bcbb3021746073c6fa523

COCOAPODS: 0.33.1

Podfile.lock文件最大得用处在于多人开发。对于没有在Podfile中指定Pods依赖库版本的写法，如下：   
pod JSONKIT   
  
该句话用于获取当前SBJson这个Pods依赖库的最新版本。   
当团队中的某个人执行完pod install命令后，生成的Podfile.lock文件就记录下了当时最新Pods依赖库的版本，这时团队中的其它人check下来这份包含 Podfile.lock文件的工程以后，再去执行pod install命令时，获取下来的Pods依赖库的版本就和最开始用户获取到的版本一致。如果没有Podfile.lock文件，后续所有用户执行pod install命令都会获取最新版本的SBJson，这就有可能造成同一个团队使用的依赖库版本不一致， 这对团队协作来说绝对是个灾难！

在这种情况下，如果团队想使用当前最新版本的JSONKIT依赖库，有两种方案：

* 更改Podfile，使其指向最新版本的JSONKIT依赖库；
* 执行pod update命令；

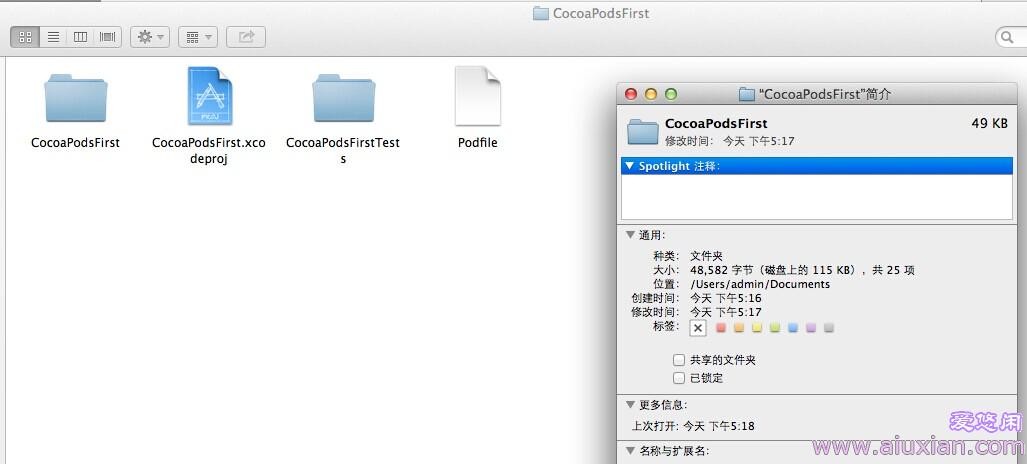
鉴于Podfile.lock文件对团队协作如此重要，我们需要将它添加到版本管理中。

**二、Podfile文件**

对于普通用户来说，使用CocoaPods我们打交道最多的就是Podfile文件。CocoaPods是用ruby实现的，因此Podfile文件的语法就是ruby的语法。接着从以下几个方面来介绍Podfile:

1、Podfile文件存放位置

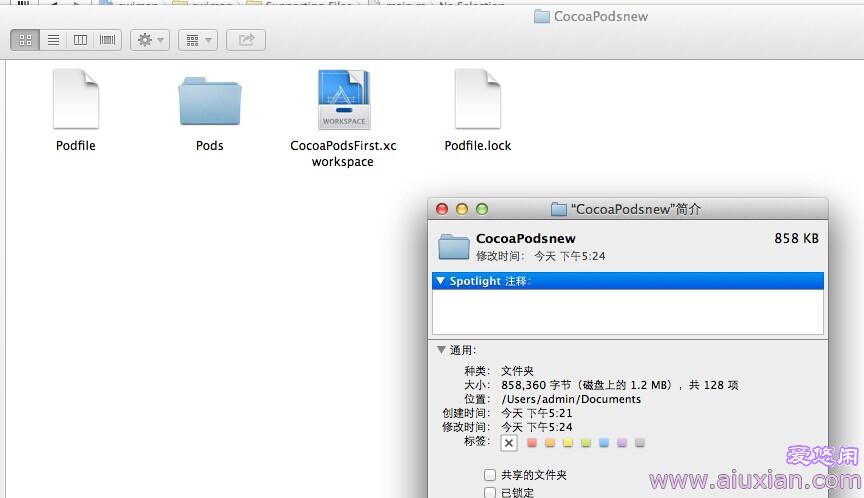
这是在上篇文章中，遗留的一个问题。通常情况下我们都推荐Podfile文件都放在工程根目录，如下图所示：



事实上Podfile文件可以放在任意一个目录下，需要做的是在Podfile中指定工程的路径，和原来相比，Podfile文件就在最开始的位置增加了一行，具体内容如下：

xcodeproj "/Users/admin/Documents/CocoaPodsFirst/CocoaPodsFirst.xcodeproj"   
  
platform :ios  
pod 'FXReachability', '~> 1.1.1'  
pod 'GCJSONKit', '~> 1.5.0'  
  
platform :ios ,'7.0'  
pod 'AFNetworking', '~> 2.3.1'

指定路径使用的是xcodeproj关键字。此后，进入Podfile文件所在路径，执行pod install命令就会和之前一样下载这些Pods依赖库，而且生成的相关文件都放在了Podfile所在目录下面，如下图：

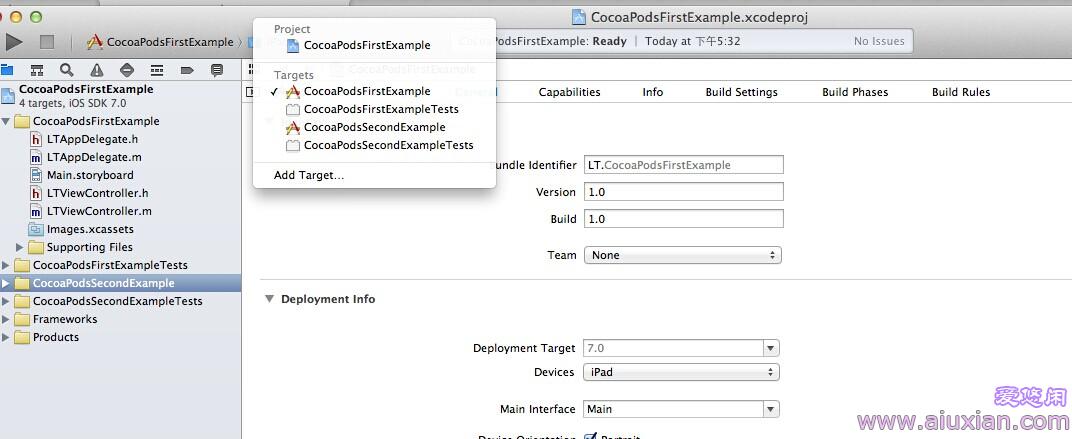


和之前一样，我们仍然需要使用这里生成的workspace文件打开工程。

**2、Podfile和target**

Podfile本质上是用来描述Xcode工程中的targets用的。如果我们不显式指定Podfile对应的target，CocoaPods会创建一个名称为default的隐式target，会和我们工程中的第一个target相对应。换句话说，如果在Podfile中没有指定target，那么只有工程里的第一个target能够使用Podfile中描述的Pods依赖库。

如果想在一个Podfile中同时 描述project中的多个target，根据需求的不同，可以有不同的实现方式。为了说明问题，在原来的工程中再创建一个名称为 CocoaPodsSecondExample的target，现在的project中包含的target有：



①多个target中使用相同的Pods依赖库

比如，名称为CocoaPodsTest的target和Second的target都需要使用Reachability、SBJson、AFNetworking三个Pods依赖库，可以使用link\_with关键字来实现，将Podfile写成如下方式：   
  
link\_with 'CocoaPodsFirstExample','CocoaPodsSecondExample'    
  
platform :ios   
pod 'FXReachability', '~> 1.1.1'   
pod 'GCJSONKit', '~> 1.5.0'   
  
platform :ios ,'7.0'   
pod 'AFNetworking', '~> 2.3.1'   
  
这种写法就实现了CocoaPodsFirstExample和CocoaPodsSecondExample两个target共用相同的Pods依赖库。

②不同的target使用完全不同的Pods依赖库

CocoaPodsFirstExample这个target使用的是Reachability、SBJson、AFNetworking三个依赖库，但CocoaPodsSecondExample这个target只需要使用OpenUDID这一个依赖库，这时可以使用target关键字，Podfile的描述方式如下：(注意target后面的:不能有空格)   
  
target :'CocoaPodsFirstExample' do   
platform :ios   
pod 'FXReachability', '~> 1.1.1'   
pod 'GCJSONKit', '~> 1.5.0'   
  
platform :ios ,'7.0'   
pod 'AFNetworking', '~> 2.3.1'   
end   
  
target :'CocoaPodsSecondExample' do   
platform :ios   
pod 'OpenUDID', '~> 1.0.0'   
end   
其中，do/end作为开始和结束标识符。

**3、使用Podfile管理Pods依赖库版本**

再引入依赖库时，需要显示或隐式注明引用的依赖库版本，具体写法和表示含义如下：

|  |  |  |
| --- | --- | --- |
| 1 | pod 'AFNetworking'      //不显式指定依赖库版本，表示每次都获取最新版本 | |
| 2 | pod 'AFNetworking', '2.0'     //只使用2.0版本 |

|  |  |
| --- | --- |
| 3 | pod 'AFNetworking', '> 2.0'     //使用高于2.0的版本 |
| 4 | pod 'AFNetworking', '>= 2.0'     //使用大于或等于2.0的版本 | |

|  |  |
| --- | --- |
| 5 | pod 'AFNetworking', '< 2.0'     //使用小于2.0的版本 |
| 6 | pod 'AFNetworking', '<= 2.0'     //使用小于或等于2.0的版本 | |

|  |  |
| --- | --- |
| 7 | pod 'AFNetworking', '~> 0.1.2'     //使用大于等于0.1.2但小于0.2的版本 |
| 8 | pod 'AFNetworking', '~>0.1'     //使用大于等于0.1但小于1.0的版本 |

|  |  |
| --- | --- |
| 9 | pod 'AFNetworking', '~>0'     //高于0的版本，写这个限制和什么都不写是一个效果，都表示使用最新版本 |

**三、CocoaPods常用命令**

**1、pod install**

根据Podfile文件指定的内容，安装依赖库，如果有Podfile.lock文件而且对应的Podfile文件未被修改，则会根据Podfile.lock文件指定的版本安装。

每次更新了Podfile文件时，都需要重新执行该命令，以便重新安装Pods依赖库。

**2、pod update**

若果Podfile中指定的依赖库版本不是写死的，当对应的依赖库有了更新，无论有没有Podfile.lock文件都会去获取Podfile文件描述的允许获取到的最新依赖库版本。

**3、pod search**

命令格式为：$ pod search OpenUDID    
后面的OpenUDID为参数。从命令的名称不难看出，该命令是用来按名称搜索可用的Pods依赖库，执行结果如下：

|  |  |
| --- | --- |
| 1 | -> OpenUDID (1.0.0) |
| 2 | Open source initiative for a universal and persistent UDID solution for iOS. | |

|  |  |
| --- | --- |
| 3 | pod 'OpenUDID', '~> 1.0.0' |
| 4 | - Homepage: http://OpenUDID.org | |

|  |  |  |
| --- | --- | --- |
| 5 | - Source:   [https://github.com/ylechelle/OpenUDID.git](https://github.com/ylechelle/OpenUDID.git%C2%A0) | |
| 6 | - Versions: 1.0.0 [master repo] |

pod 'OpenUDID', '~> 1.0.0'   
不难看出，这是我们需要添加到Podfile文件中的。

有了这条命令，就可以方便、迅速地找到需要的Pods依赖库。

**4、pod setup**

命令格式为：$ pod setup   
执行完了以后会打印：

|  |  |  |
| --- | --- | --- |
| 1 | Setting up CocoaPods master repo | |
| 2 | Updating 7cd4668..f3d3ced |

|  |  |
| --- | --- |
| 3 |  |
| 4 | Fast-forward | |

接下来还会打印很多更新信息。这条命令用于跟新本地电脑上的保存的Pods依赖库tree。由于每天有很多人会创建或者更新Pods依赖库，这条命令执行 的时候会相当慢，还请耐心等待。我们需要经常执行这条命令，否则有新的Pods依赖库的时候执行pod search命令是搜不出来的。