刘坤的技术博客

- BlueBox
- 所有文章

嗨，我是刘坤，一名来自中国的 IOS 开发者，现就职于杭州阿里，花名‘念纪’，沉淀技术，寻求创意

GitHub RSS

友情链接

- Casatwy Taloyum
- gf&zjの盗梦空间
- 明弈

Cocoapods代码管理

只要是做代码开发，就会遇到代码管理的问题，IOS也不例外。比如你写了一个用于网络图片加载的公共组件EGOImageLoading. 然后这个组件在10个project中被使用。某一天你对这个组件做了一些优化或bug修复，怎么把代码同步到所有工程呢。普通程序员：手动一个一个拷贝到每个工程。文艺程序员：`pod update`

cocoapods是什么

作为一个objc代码的程序员，应该都听过cocoapods,如果你还没听过，那可以说out了。

每种语言发展到一个阶段，就会出现相应的依赖管理工具, 或者是中央代码仓库。比如

```
Java: maven, Ivy
Ruby: gems
Python: pip, easy_install
Nodejs: npm
```

那么cocoapods就是objc代码的依赖管理工具。cocoapods非常适合代码的集中管理，和工程依赖的管理。比如以前我要添加ASIHttpRequest到工程中，我需要：

1、将ASIHttpRequest下载到本地 2、解压后把对应的代码拖到工程中去 3、在target中的build settings添加对应的framework: CFNetwork, SystemConfiguration, MobileCoreServices, CoreGraphics, zlib.

这些体力活虽然简单，但毫无技术含量并且浪费时间，而且很容易漏加framework，导致报错排查。使用cocoapods之后，我们就可以在配置文件中加一行 `pod 'ASIHTTPRequest', '~> 1.8.2'`, 然后`pod install`就行了啊，如果ASI有新版本1.8.3发布，那么我就改成`pod 'ASIHTTPRequest', '~> 1.8.3'`，然后`pod update`就行了。

cocoapods安装和使用简介

cocopods安装非常简单，本来想略去这一部分的，但还是贴一下命令吧，如果不明白网络上搜索一搜一大把。

```
sudo gem install cocoapods
pod setup
```

使用也比较简单，cd到你的工程的根目录$projectPath

```
cd $projectPath
pod init
```

这个命令就会再工程根目录下面生成一个Podfile文件,podfile用文本编辑就行了，我给出一个例子

```
1 platform :ios, "5.0"
2
3 target "TestPod" do
4 pod 'ASIHTTPRequest', '~> 1.8.2'
5 pod 'CocoaLumberjack', '~> 1.8.1'
6 pod 'libextobjc', '~> 0.4'
```

```
7 pod 'FMDB', '~> 2.2'
8 end
```

然后再使用pod install就可以把这些开源控件安装了。安装完，这些控件都放在一个Pods的工程中，然后会用xcode的workspace来管理这个工程和你自己的工程。以后就打开yourproject.xcworkspace就行了。

搜索公共组件：

```
$ pod search ASIHttpRequest

-> ASIHTTPRequest (1.8.2)
Easy to use CFNetwork wrapper for HTTP requests, Objective-C, Mac OS X and iPhone.
pod 'ASIHTTPRequest', '~> 1.8.2'
- Homepage: http://allseeing-i.com/ASIHTTPRequest
- Source:   https://github.com/pokeb/asi-http-request.git
- Versions: 1.8.2, 1.8.1 [master repo]
- Sub specs:
    - ASIHTTPRequest/Core (1.8.2)
    - ASIHTTPRequest/ASIWebPageRequest (1.8.2)
    - ASIHTTPRequest/CloudFiles (1.8.2)
    - ASIHTTPRequest/S3 (1.8.2)
```

cocoapods的工作流程简介

1.cocoapods代码库列表

上面讲了pod可以搜索对应的开源组件库，是如何搜索的呢，难道是联网在线搜索的?

当然不是的，cocoapods有一个git仓库专门保存了github上一些有名的开源组件的podspec文件。cocoapods会把它clone到你本地的~/.cocoapods/repo/master路径下。当你pod search 一个内容时，就在这个本地目录下进行搜索。

这里我们就洞悉了cocoapods是如何管理中心仓库的了，就放在你的~/.cocoapods/repo目录下啊，了解这个就便于后面我们创建自己的私有的代码中心仓库，在后面会介绍如何创建自己的私有库。

2.pod如何安装代码

把pod install改为pod install --verbose之后就可以看到这个过程cocoapods做了什么

2.1首先会更新你的本地中心仓库

```
Updating spec repositories
```

这个过程可能会很慢，当然我们可以忽略这个过程。

```
pod install --no-repo-update
```

2.2然后下载源代码到本地

cocoapods会预先从github上把代码拉下来放到缓存里，缓存的目录~/Library/Caches/Cocoapods，知道这个是有作用的，方便清理缓存。

下载到缓存目录后把代码拷贝的你工程目录下的对应目录下：如$projectPath/Pods/ASIHTTPRequest

2.3Generating Pods project

创建或更新Pods工程，cocoapods会创建一个Pods的project来管理这些代码，Pods工程会把公共组件代码编译为静态库文件.a给你的工程使用。

2.4配置workspace

创建一个workspace管理Pods工程和你原本的工程。并通过xcconfig配置文件配置编译参数到你的原本工程中去。

cocoapods私有库管理

1.配置podspec

cocoapods是如何知道一个开源组件需要引入哪些代码，需要添加哪些framework和lib，需要哪些资源文件的呢。

这些信息都配置在podspec文件中。如果你打开cocoapods的本地中心仓库文件来看的话，会发现保存的都是podspec文件。首先生成一个podspec文件，在工程目录下：

```
pod spec create [spec_name]
```

关于podspec怎么写，cocoapods官网上的guide写的非常详细啦，当你要写的时候参考这写就行了：Podspec Syntax Reference这里给出一个我

自己写的示例,我写的组件叫BBFoundation:

```
Pod::Spec.new do |s|
  s.name         = "BBFoundation"   #名称
  s.version      = "0.0.4"          #版本号
  s.summary      = "有大量的类别和UI组件方便构建工程"  #描述
  s.homepage     = "http://example.com/projects/BBFoundation" #描述页面
  s.license      = 'MIT'    #版权声明
  s.author       = { "liukun" => "765409243@qq.com" }  #作者
  s.platform     = :ios, '5.0'    #支持的系统
  s.source       = { :git => "git@example.com:BBFoundation.git", :tag => "0.0.4" }   #源码地址
  s.source_files  = 'BBFoundation/**/*.{h,m}'    #源码
  s.frameworks = 'CoreGraphics', 'CoreFoundation'  #framework
  s.libraries = 'z', 'bz2.1.0' #lib
  s.requires_arc = true       #是否需要arc
end
```

2.测试pod是否能成功

有了上面的podspec，我们就可以测试这个配置是否可以正确使用了啊。正好可以使用cocoapods的本地pod功能。

比如我的BBFoundation的代码放在~/Code/BBFoundation路径下，podspec文件是BBFoundation.podspec.首先我们吧source路径改成本地的，就是把上面一段代码中的s.source改下:

```
s.source       = { :git => "$HOME/Code/BBFoundation", :tag => "0.0.4" }
```

然后在测试的工程的Podfile中加入

```
pod 'BBFoundation', :podspec => '$HOME/Code/BBFoundation/BBFoundation.podspec'
```

然后 pod install --no-repo-update测试下安装. 如果能够安装成功，并且在你的测试工程中能够成功引用代码并编译成功，那就没问题啦。

3.添加私有中心仓库

上面一步已经完成了自己创建组件并本地使用pod进行依赖管理的功能了。但是用起来似乎还是不够完美。我们想和github上的第三方库同样的使用方法啊，像这样简单的:

```
pod 'BBFoundation' '~> 0.0.4'
```

我们就需要添加自己的中心仓库了。添加过程也比较简单的:

3.1创建一个git仓库作为中心仓库

直接在~/.cocoapods/repo目录下面创建一个目录MyPrivatePods,放自己的podspec

```
cd ~/.cocoapods/repo
mkdir MyPrivatePods
cd MyPrivatePods
mkdir BBFoundation
cd BBFoundation
mkdir 0.0.4
cd 0.0.4
cp ~/Code/BBFoundation/BBFoundation.podspec .
```

上面的命令创建了3个文件夹，是因为cocoapods规定了中心代码库的文件结构树是这样的:

```
repo
|—— Spec repo name  //就是中心仓库的名称，如MyPrivatePods
    |—— [SPEC_NAME]  //就是组件的名称
        |—— [SPEC_VERSION]  //组件的版本号
            |—— [SPEC_NAME].podspec    //组件的spec配置文件
```

这个时候试试pod search BBFoundation就会发现，可以搜索到自己建立的私有控件了。

下面为了保持这个仓库可以共享，我们就把它用git托管起来，最好是放在git server上，那么别人可以添加你的私有中心库了哈:

```
cd ~/.cocoapods/repo/MyPrivatePods
git init
git add .
```

```
git commit -am 'init commit'
git remote add origin git@example.com:MyPrivatePods.git
git push origin master
```

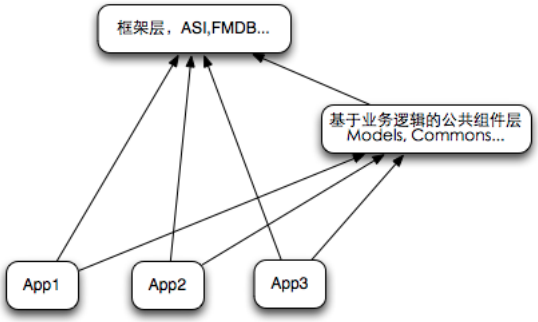
这个时候就可以共享你自己的私有中心仓库了，其他人可以通过以下命令把这个仓库添加到他的本地：

```
pod repo add reponame git@example.com:MyPrivatePods.git
```

cocoapods代码托管的使用意义

讲了这么多，好像一直在讲这个怎么用，怎么设置，但是没有讲它到底能够给我们带来多少好处。使用过java的maven的可能可以理解它的好处。

如果你像我们一样拥有多个app, 这其中每个app都要依赖于某个基于业务逻辑的公共模块。就可以像如下图处理工程之间的依赖：



Comments