刘春桂的博客

CocoaPods创建私有Pods

Oct 19, 2015 | ios | 1161 Hits

创建Pod私有源步骤：

1、创建两个git仓库，一个用来做私有的Spec Repo，一个是我们自己的公共组件

2、添加私有的repo到CocoaPods

3、制作Podspec，并且推送到你创建的私有repo

4、使用Pod，在Podfile添加私有源来搭建项目

一、创建两个git仓库，一个用来做私有的Spec Repo，一个是我们自己的公共组件

在这一步当中需要git服务器用来创建仓库，我这里只是作为一个例子，所以直接使用github来当做我的服务器，我创建的这些项目都是public的。你如果是练手的话，也可以仿照我一样在github上面创建仓库。如果是为公司建立私有的公共组件库，你可以在github等平台上面创建私有仓库，或者直接搭建一个git服务器来创建仓库。搭建服务器可以参考Git简单总结的第一部分，里面有Mac搭建也有Linux搭建的链接。

在这里，我们需要搭建两个git仓库。

第一个仓库名称是first，专门用来做私有Spec Repo。

第二个仓库名称是BGNetwork，这个就是我们公共组件库。

在这里，我们来说一下什么是Spec Repo?

> 他是所有的Pods的一个索引，就是一个容器，所有公开的Pods都在这个里面，他实际是一个Git仓库remote端。
> 在GitHub上，当你使用了Cocoapods后他会被clone到本地的~/.cocoapods/repos目录下，可以进入到这个目录看到master文件夹就是这个官方的Spec Repo了。

这里引用了使用Cocoapods创建私有podspec的内容。

二、添加私有的repo到CocoaPods

主要命令是 `pod repo add REPO_NAME SOURCE_URL` 。其中，REPO_NAME是私有repo的名字，取一个容易记住的名字，后面还会用到，以后公司内部的组件对应的podspec都可以推送到这个repo中；SOURCE_URL就是刚刚我们创建的 `first` 仓库链接。

```
$ pod repo add liuchungui https://github.com/liuchungui/first.git
$ ls ~/.cocoapods/repos
liuchungui      master
```

这时，你会发现有两个文件夹 `liuchungui` 和 `master` ，master是Cocoapods官方的repo，而liuchungui就是我刚刚创建的。进入liuchungui文件夹查看，你会发现它是clone了一份 `https://github.com/liuchungui/first.git` 。
在这里，我们是一个空的仓库，可以不检查，但是你的仓库如果有什么其他东西的话，可以检查一下。

```
$ cd ~/.cocoapods/repos/liuchungui
$ pod repo lint .
```

三、制作Podspec，并且推送到你创建的私有repo

1、将我们前面创建的BGNetwork项目克隆到本地

`git clone https://github.com/liuchungui/BGNetwork.git`

2、在本地我们使用了xcode创建了项目，并且写了一个网络框架，运行没有问题，我们准备提交到github，并打上版本号。

```
git add .
git commit -m 'add file'
git push origin master
git tag -m 'add tag' '0.1.1'
git push --tags
```

3、我们开始制作Podspec文件。

`BGNetwork` 是一个基于 `AFNetworking` 而封装的网络框架，它主要的源文件都在 `BGNetwork/BGNetwork` 路径下。我们将它放在CocoaPods给第三方使用，主要是将这个文件夹下的源文件加载到第三方的项目中以供使用。
下面是供第三方使用的源文件结构，具体可以下载BGNetwork代码查看

```
____BGNetwork
| |____BGAFHTTPClient.h
| |____BGAFHTTPClient.m
```

```
| |____BGAFRequestSerializer.h
| |____BGAFRequestSerializer.m
| |____BGAFResponseSerializer.h
| |____BGAFResponseSerializer.m
| |____BGNetworkCache.h
| |____BGNetworkCache.m
| |____BGNetworkConfiguration.h
| |____BGNetworkConfiguration.m
| |____BGNetworkConnector.h
| |____BGNetworkConnector.m
| |____BGNetworkManager.h
| |____BGNetworkManager.m
| |____BGNetworkRequest.h
| |____BGNetworkRequest.m
| |____BGNetworkUtil.h
| |____BGNetworkUtil.m
```

在BGNetwork项目的根目录下创建一个BGNetwork.podspec文件，对应上面的需求，我们的podspec可以这么写

```
Pod::Spec.new do |spec|
  #项目名称
  spec.name         = 'BGNetwork'
  #版本号
  spec.version      = '0.1.1'
  #开源协议
  spec.license      = 'MIT'
  #对开源项目的描述
  spec.summary      = 'BGNetwork is a request util based on AFNetworking'
  #开源项目的首页
  spec.homepage     = 'https://github.com/chunguiLiu/BGNetwork'
  #作者信息
  spec.author       = {'chunguiLiu' => 'chunguiLiu@126.com'}
  #项目的源和版本号
  spec.source       = { :git => 'https://github.com/chunguiLiu/BGNetwork.git', :tag
  #源文件，这个就是供第三方使用的源文件
  spec.source_files = "BGNetwork/*"
  #适用于ios7及以上版本
  spec.platform     = :ios, '7.0'
  #使用的是ARC
  spec.requires_arc = true
  #依赖AFNetworking2.0
  spec.dependency 'AFNetworking', '~> 2.0'
end
```

注意：spec.source源是 `BGNetwork` 的git仓库，版本号是我们上一步打上的版本号0.1.1。

4、验证并推送到服务器

在推送前，我们先验证Podspec，验证的时候是验证BGNetwork.podspec文件，所以我们需要保证进入的目录和BGNetwork.podspec同级的，验证命令如下：

```
$ pod lib lint
```

注意：验证的时候，会获取 `BGNetwork.podspec` 文件中的 `spec.source` 来获取git服务器上面对应版本的代码，然后再找到 `spec.source_files` 中的源代码，通过xcode命令行工具建立工程并且进行编译。所以这一步的过程会比较久，如果编译没有错误，就验证通过。

如果没有错误和警告我们就可以推送到服务器了，推送使用的命令如下：

```
$ pod repo push REPO_NAME SPEC_NAME.podspec
```

它也会先验证，然后再推送。我这里推送BGNetwork命令是：

```
$ pod repo push liuchungui BGNetwork.podspec
```

如果没有错误，但是有警告，我们就将警告解决，也可以加 `--allow-warnings` 来提交

```
$ pod repo push liuchungui BGNetwork.podspec --allow-warnings
```

如果有错误，我们可以去查看错误信息对应下的Note信息并解决。在这错误当中，常常会遇到找不到对应文件的错误，这个时候你需要查看 `BGNetwork.podspec` 文件中 `spec.source` 下git仓库链接是否没问题，git仓库下对应的tag版本中 `spec.source_files` 路径下是否正确。

如果查看Note信息看不出什么问题，可以加上 `verbose` 参数进行更详细的查看。

```
$ pod repo push liuchungui BGNetwork.podspec --allow-warnings --verbose
```

注意事项：碰到本地使用 `pod lib lint` 验证通过，但是push到服务器却失败了，这个时候很可能就是服务器tag版本不对，使用 `--verbose` 能查看详细的错误信息。

5、搜索我们的框架

到这一步，我们就可以通过 `pod search BGNetwork` 来搜索了，搜索到了说明我们私有源建立成功。

```
$ pod search BGNetwork
-> BGNetwork (0.1.2)
   BGNetwork is a request util based on AFNetworking
   pod 'BGNetwork', '~> 0.1.2'
   Homepage: https://github.com/chunguiLiu/BGNetwork
   Source:   https://github.com/chunguiLiu/BGNetwork.git
   Versions: 0.1.1, 0.1.0 [liuchungui repo] - 0.1.2, 0.1.1 [master repo]
```

由上面的搜索知道，BGNetwork在liuchungui这个私有repo中存在0.1.1和0.1.0版本，在master中存在0.1.2和0.1.1版本。
搜索成功之后，我们将BGNetwork.podspec也推送到远程服务器。

四、使用Pod，在Podfile添加私有源来搭建项目

使用时，在Podfile文件中添加**本地私有源和官方源**。如果没有添加本地私有源，它默认是用官方的repo，这样找不到本地的Pod；如果只是设置了本地私有源，就不会再去官方源中查找。
下面是Podfile内容

```
#官方Cocoapods的源
source 'https://github.com/CocoaPods/Specs.git'
#本地私有源
source 'https://github.com/liuchungui/first.git'
platform :ios, '7.0'
pod 'BGNetwork', '~> 0.1.1'
```

注意

1、途中遇到了几次问题，就是 `pod repo push` 不上去，显示没有找到对应文件，后来发现是版本的问题，没有打上版本号或者 `Podspec` 中版本错了。所以我们在维护一个框架时，修改框架之后，push到git服务器之后先打上tag，然后再修改podspec文件中的版本，最后push到对应的pod repo中。如果你遇到这个问题，可以详细查看**第三步中的验证并推送到服务器**这一节。

2、若是在框架当中，存在不同的文件夹，请使用 `subspec` 。如果不同文件夹之间的文件有相互导入的情况，请将被导入的头文件设置为 `public_header_files` ，并且通过 `dependency` 设置依赖，具体可以参考AFNetworking的podspec文件。

3、若是需要提交给官方，请使用

```
pod trunk register youremail
查看信息
pod trunk me
将对应的pod推送到服务器
pod trunk push
```

4、使用 `pod install` 时，它首先会更新整个官方的源，而Cocoapods每天都有很多人提交，所以更新比较慢。所以，建议每过一段时间更新一下官方库，平常的时候，咱们可以在 `install` 或 `update` 加一个参数 `--no-repo-update` 让它不用更新。

```
$ pod install --verbose --no-repo-update
```

```
$ pod update --verbose --no-repo-update
```

参考

Private Pods
使用Cocoapods创建私有podspec
CocoaPods详解之一–使用篇

修改记录

- 2015.10.19 发布，中途修改过几次
- 2016.4.11 修改了第三部中的验证并推送到服务器内容

cocoapods private pods cocoapods私有源 分享到

Mac环境下Nginx实现反向代理 iOS9之适配ATS