**创建基于GitLab的Cocoapods私有库**

## 1、简介

Cocoapods是非常好用的一个ios依赖管理工具，使用它可以方便管理和更新项目中所用到的第三方库，以及自己的项目中的公共组件交由它去管理。例如：AFNetworking、SDWebImage等著名的第三方框架都是可以基于Github进行Cocoapods依赖的。我们这里主要介绍的是基于GitLab进行Cocoapods依赖。

## 2、创建并设置一个私有的Spec Repo

先来说第一步，什么是Spec Repo?它是所有的Pods的一个索引，就是一个容器，所有公开的Pods都在这个里面，它实际是一个Git仓库remote端在GitHub上，但是当你使用Cocoapods后它会被clone到本地的~/.cocoapods/repos目录下，使用公开的Pods的在repods目录下会有一个master目录，其实这个就是一个本地的仓库。可以进入到master目录下有一个Specs文件夹（这个就是官方的Spec Repo），这里面就是所有的公开组件的Spec。如图2.1：

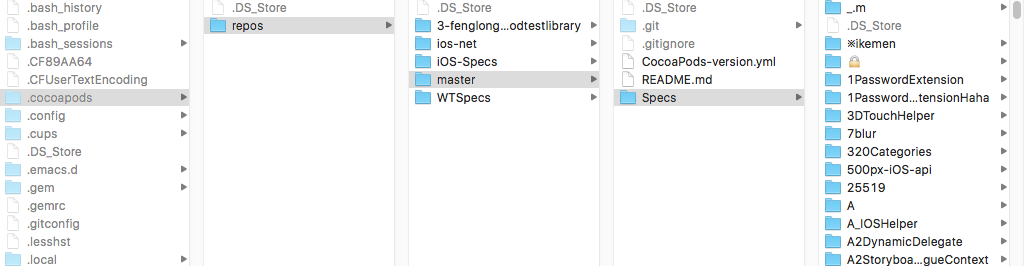


图2.1

因此我们需要创建一个类似于master的私有Spec Repo。所以我在自己的GitLab服务器上创建一个私有的Git仓库ios-Specs。如果2.2：

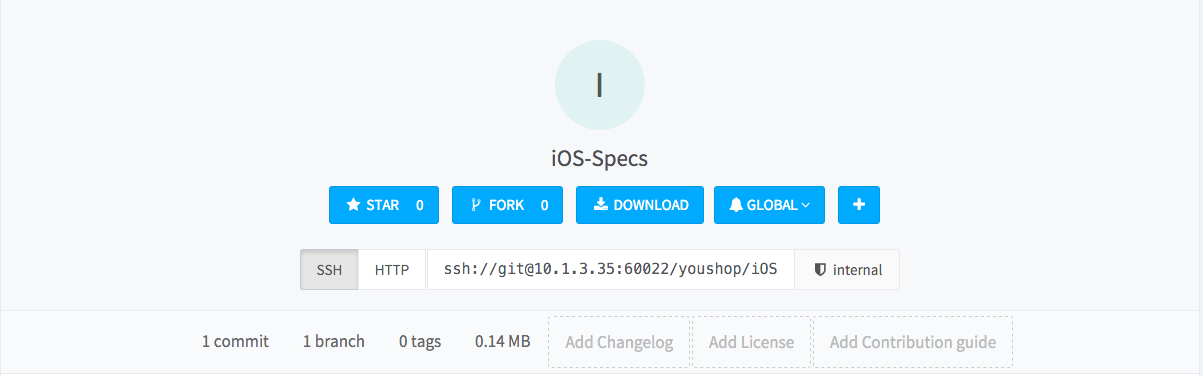


图2.2

创建完私有仓库ios-Specs之后，执行如下命令。如果2.3：

Macintosh HD:Users:fenglongchen:Desktop:F759EC5C-1ABA-480E-B55B-A4FA2DDFC162.png

图2.3

此时如果成功的话，进入到~/.cocoapods/repos目录下就可以看到ios-Specs这个目录了。至此第一个创建私有的Spec Repo完成了。如图2.3:



图2.3

PS:如果其合作人共同使用这个私有的Spec Repo的话，他在对应Git仓库的权限的前提下执行相同的命令添加这个Spec Repo即可。

## 3、创建Pod项目工程文件

这一步分为两种情况：1、有现有的组件项目，并且在Git的版本管理下。2、没有现成的组件项目，需要重头开始的。

### 3.1、没有现成的组件项目：

我们拿创建CXNetworking来举例子，先cd到要创建项目的文件夹下，然后执行命令。如图3.1.1：

Macintosh HD:Users:fenglongchen:Desktop:54447C11-7639-47E8-BF9C-2B036BAE0C13.png

图3.1.1

之后会有四个问题，1、开始语言是基于ObjC / Swift？2、是否要创建一个Demo？3、你需要哪个测试框架 Specta / Kiwi / None？4、类的前缀？我选择的是：1、ObjC；2、Yes；3、Specta；4、CX。如图3.1.2：

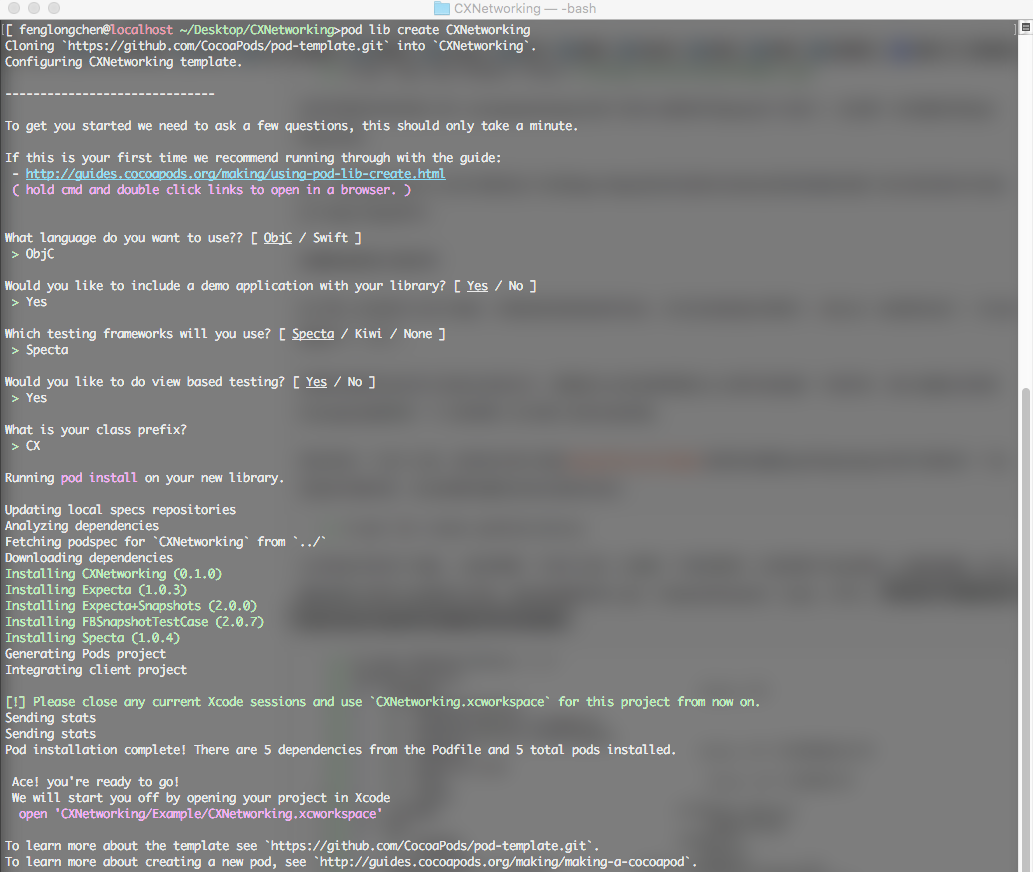


图3.1.2

问完这四个问题之后他会自动执行pod install命令创建项目并生成依赖。如图3.1.3：



图3.1.3

接下来向Pod文件夹中添加库文件和资源，并配置podspec文件，我把共有的组件放入Pod/Classes中，然后进入Example文件中执行pod update命令，在打开项目工程可以看到，刚刚添加的组件的文件已经在Pods子工程下Development Pods/CXNetworking中了，然后可以编辑Demo工程，测试组件。如图3.1.4：

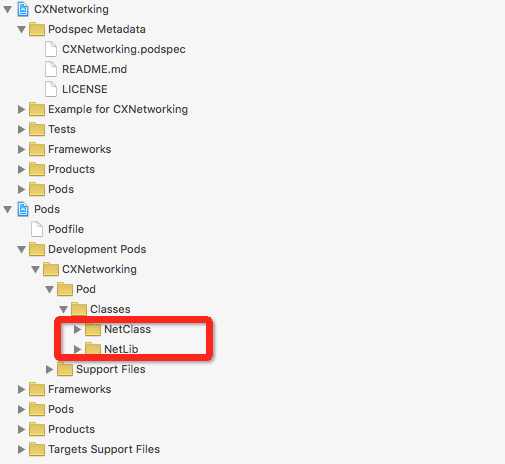


图3.1.4

PS:这里需要注意每当你想Pod中添加了新的文件或者以后更新了podspec的版本都需要重新执行一遍pod update命令。

通过Ccocoapods创建出来的目录本身就在本地的Git管理下，我们把本地仓库关联到远端仓库，同样需要在GitLab上创建一个私有的仓库iOS-net。如图：3.1.5：



图3.1.5

然后需要把本地仓库与刚才创建的远程仓库进行关联，并且把本地的内容Push到远端。

因为podspec文件中的获取Git版本控制的项目还需要tag号，所以我们要打上一个tag。如图：3.1.6：

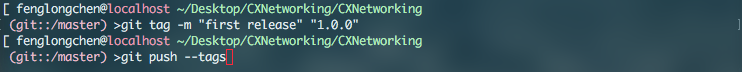


图3.1.6

做完这些就可以开始编辑podspec文件了，它是一个Ruby的文件，把编辑器的格式修改为Ruby就能看到语法高亮。如图3.1.7：

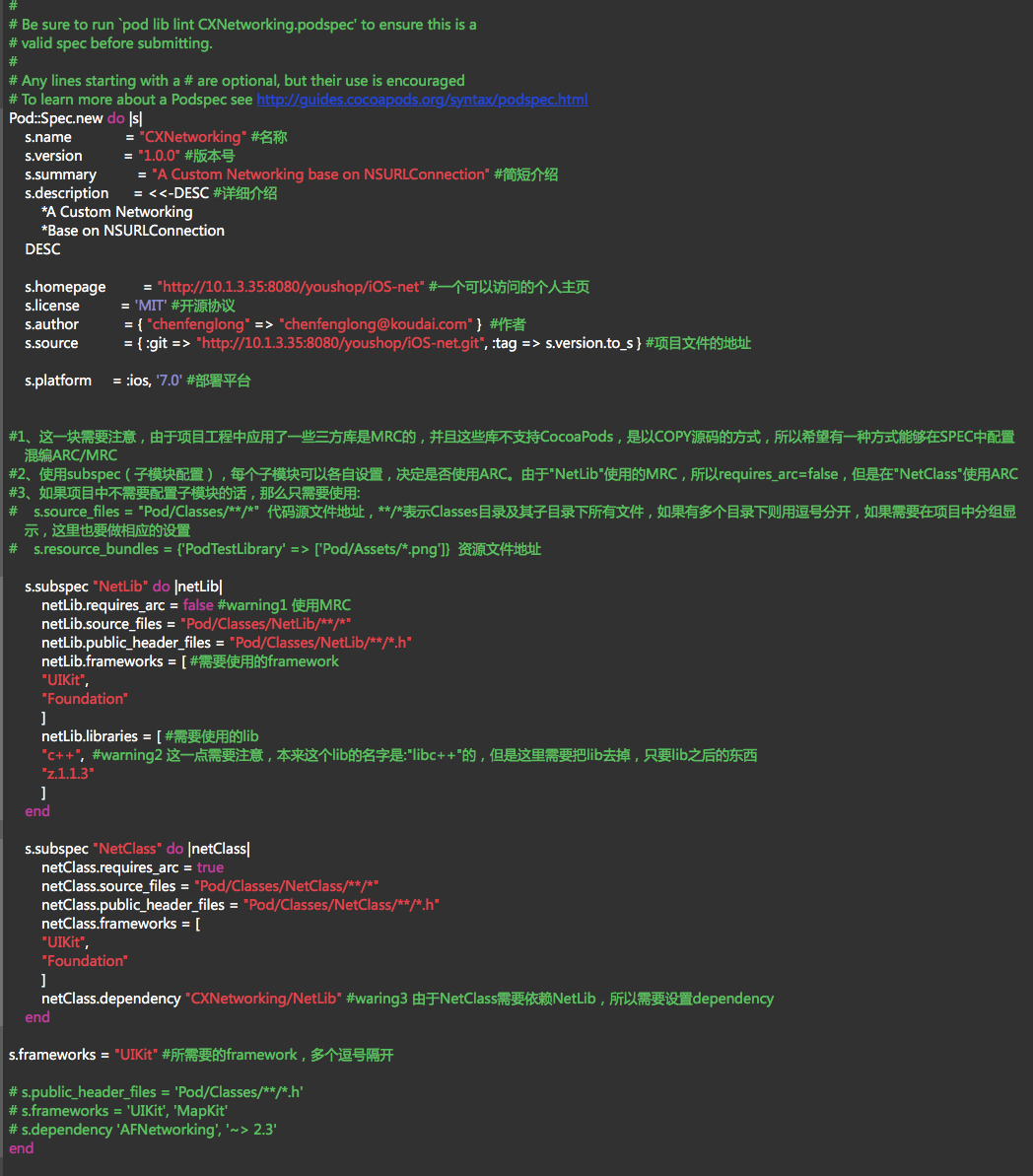


图3.1.7

PS：这个配置文件中需要特别的注意#warning[1、2、3]，还要需要自己知道自己的项目是否需要“子模块”配置，根据不同的配置来编写合适的配置。具体可以参考图3.1.7中的注释。

编辑完podspec文件后，需要验证一下文件是否可用，如果有任何的waring或者error都不可以被添加到Spec Repo中，不过xcode的warning是可以存在的，验证需要执行一下命令。如图3.1.8：



图3.1.8

当你看到：->CXNetworking(1.0.0)

CXNetworking passed validation.

时，说明验证通过了，不过这只是这个podspec是合格的，不一定说明这个Pod是可以用的，这个需要在接下来进行验证。

### 3.2、已有现成的组件项目：

如果你已有现成的项目组件，那么需要把这个项目创建一个podspec文件，创建它需要执行另外一个命令。如图3.2.1



图3.2.1

执行完之后，就创建了一个podspec文件，他其中会包含很多内容，可以按照3.1之前介绍的进行编辑，编辑完成之后使用验证命令验证一下。如图3.2.2：



图3.2.2

验证无误就可以进行下一步了。

## 4、向Spec Repo提交podspec

向Spec Repo提交podspec前提是一个podspec必须通过验证无误。向我们私有的Spec Repo提交podspec只需要一个命令。如图4.1：



图4.1

完成之后这个组件就添加到我们私有的Spec Repo中，可以进入到~/.cocoapods/repos/CXNetworking目录下查看。如图4.2：

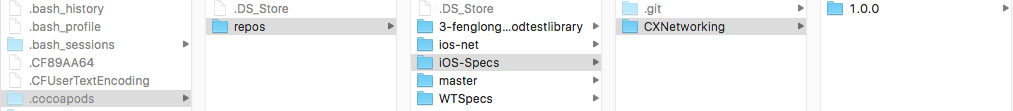


图4.2

再去看我们的Spec Repo远端仓库，也有一次提交，这个podspec也被push上去了。

到现在为止，我们的这个组件就已制作完成，使用pod search命令就可以查看我们的库了。如图4.3：

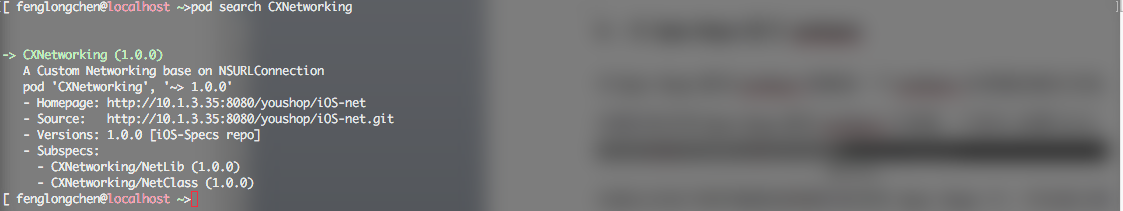


图4.3

## 5、更新维护podspec

编辑之前的项目文件之后，在测试项目中里用pod update一下，编辑的文件就会自动更新到项目工程中，写代码无误之后，就可以 将这个工程push到远端仓库，并打上新的tag->1.0.1

再次使用pod lib lint验证编辑好的podspec文件，没有自身的waring或者error之后，就可以再次提交到Spec Repo中了，命令和之前还是一样的，然后再次到~/.cocoapods/repos/iOS-Specs/CXNetworking就可以看到新的版本1.0.1了。如图5.1：

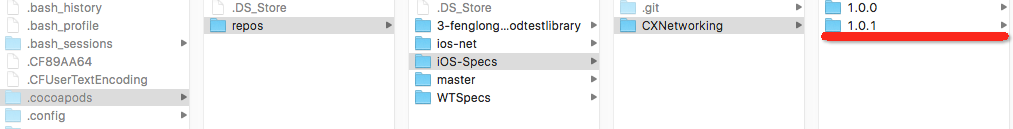


图5.1

删除一个私有的Spec Repo，只需要一条命令即可。如图5.2：



图5.2

## 6、使用私有的podspec

使用私有的podspec和使用公共的podspec用法是一样的，但是需要注意在Podfile文件把自己私有的Pod Repos的地址添加上，要不然的话在pod install会出现找不到组件的情况。如图6.1：

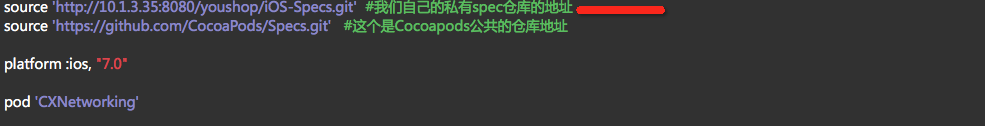


图6.1

使用pod install命令就会把自己的组件关联过来。如图6.2：

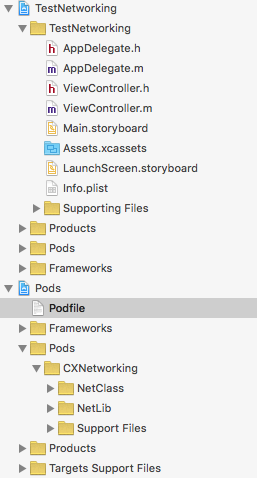


图6.2

Create by:ChenFengLong

Date:2015/11/2