<http://www.cnblogs.com/sihaixuan/p/4852974.html>

[**Android studio Maven仓库使用**](http://www.cnblogs.com/sihaixuan/p/4852974.html)

原文：[How to distribute your own Android library through jCenter and Maven Central from Android Studio](http://inthecheesefactory.com/blog/how-to-upload-library-to-jcenter-maven-central-as-dependency/en)

转自：翻译 <http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/0623/3097.html>

如果你想在Android Studio中引入一个library到你的项目，你只需添加如下的一行代码到模块的build.gradle文件中。

dependencies {

compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3'

}

就是如此简单的一行代码，你就可以使用这个library了。

酷呆了。不过你可能很好奇Android Studio是从哪里得到这个library的。这篇文章将详细讲解这是怎么回事，包括如何把你的库发布出去分享给世界各地的其他开发者，这样不仅可以让世界更美好，还可以耍一次酷。

**Android studio 是从哪里得到库的？**

先从这个简单的问题开始，我相信不是每个人都完全明白Android studio 是从哪里得到这些library的。莫非就是Android studio 从google搜索然后下载了一个合适的给我们？

呵呵，没那么复杂。Android Studio是从build.gradle里面定义的Maven 仓库服务器上下载library的。Apache Maven是Apache开发的一个工具，提供了用于贡献library的文件服务器。总的来说，只有两个标准的Android library文件服务器：jcenter 和  Maven Central。

**jcenter**

jcenter是一个由 bintray.com维护的Maven仓库 。你可以在[这里](http://jcenter.bintray.com/)看到整个仓库的内容。

我们在项目的build.gradle 文件中如下定义仓库，就能使用jcenter了：

allprojects {

repositories {

jcenter()

}

}

**Maven Central**

Maven Central 则是由[sonatype.org](https://sonatype.org/)维护的Maven仓库。你可以在[这里](https://oss.sonatype.org/content/repositories/releases/)看到整个仓库。

注：不管是jcenter还是Maven Central ，两者都是Maven仓库

我们在项目的build.gradle 文件中如下定义仓库，就能使用Maven Central了：

allprojects {

repositories {

mavenCentral()

}

}

注意，虽然jcenter和Maven Central 都是标准的 android library仓库，但是它们维护在完全不同的服务器上，由不同的人提供内容，两者之间毫无关系。在jcenter上有的可能 Maven Central 上没有，反之亦然。

除了两个标准的服务器之外，如果我们使用的library的作者是把该library放在自己的服务器上，我们还可以自己定义特有的Maven仓库服务器。Twitter的Fabric.io 就是这种情况，它们在<https://maven.fabric.io/public>上维护了一个自己的Maven仓库。如果你想使用Fabric.io的library，你必须自己如下定义仓库的url。

repositories {

maven { url 'https://maven.fabric.io/public' }

}

然后在里面使用相同的方法获取一个library。

dependencies {

compile 'com.crashlytics.sdk.android:crashlytics:2.2.4@aar'

}

但是将library上传到标准的服务器与自建服务器，哪种方法更好呢？当然是前者。如果将我们的library公开，其他开发者除了一行定义依赖名的代码之外不需要定义任何东西。因此这篇文章中，我们将只关注对开发者更友好的jcenter 和 Maven Central 。

实际上可以在Android Studio上使用的除了Maven 仓库之外还有另外一种仓库：[Ivy 仓库](http://ant.apache.org/ivy/) 。但是根据我的经验来看，我还没看到任何人用过它，包括我，因此本文就直接忽略了。

**理解jcenter和Maven Central**

为何有两个标准的仓库？

事实上两个仓库都具有相同的使命：提供Java或者Android library服务。上传到哪个（或者都上传）取决于开发者。

起初，Android Studio 选择Maven Central作为默认仓库。如果你使用老版本的Android Studio创建一个新项目，mavenCentral()会自动的定义在build.gradle中。

但是Maven Central的最大问题是对开发者不够友好。上传library异常困难。上传上去的开发者都是某种程度的极客。同时还因为诸如安全方面的其他原因，Android Studio团队决定把默认的仓库替换成jcenter。正如你看到的，一旦使用最新版本的Android Studio创建一个项目，jcenter()自动被定义，而不是mavenCentral()。

有许多将Maven Central替换成jcenter的理由，下面是几个主要的原因。

- jcenter通过CDN发送library，开发者可以享受到更快的下载体验。

- jcenter是全世界最大的Java仓库，因此在Maven Central 上有的，在jcenter上也极有可能有。换句话说jcenter是Maven Central的超集。

- 上传library到仓库很简单，不需要像在 Maven Central上做很多复杂的事情。

- 友好的用户界面

- 如果你想把library上传到 Maven Central ，你可以在bintray网站上直接点击一个按钮就能实现。

基于上面的原因以及我自己的经验，可以说替换到jcenter是明智之举。

所以我们这篇文章将把重心放在jcenter，反正如果你能成功把library放在jcenter，转到 Maven Central 是非常容易的事情。

**gradle是如何从仓库上获取一个library的？**

在讨论如何上传library到jcenter之前，我们先看看gradle是如何从仓库获取library的。比如我们在 build.gradle输入如下代码的时候，这些库是如果奇迹般下载到我们的项目中的。

compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3'

一般来说，我们需要知道library的字符串形式，包含3部分

GROUP\_ID:ARTIFACT\_ID:VERSION

上面的例子中，GROUP\_ID是com.inthecheesefactory.thecheeselibrary ，ARTIFACT\_ID是fb-like，VERSION是0.9.3。

GROUP\_ID定义了library的group。有可能在同样的上下文中存在多个不同功能的library。如果library具有相同的group，那么它们将共享一个GROUP\_ID。通常我们以开发者包名紧跟着library的group名称来命名，比如com.squareup.picasso。然后ARTIFACT\_ID中是library的真实名称。至于VERSION，就是版本号而已，虽然可以是任意文字，但是我建议设置为x.y.z的形式，如果喜欢还可以加上beta这样的后缀。

下面是Square library的一个例子。你可以看到每个都可以很容易的分辨出library和开发者的名称。

dependencies {

compile 'com.squareup:otto:1.3.7'

compile 'com.squareup.picasso:picasso:2.5.2'

compile 'com.squareup.okhttp:okhttp:2.4.0'

compile 'com.squareup.retrofit:retrofit:1.9.0'

}

那么在添加了上面的依赖之后会发生什么呢？简单。Gradle会询问Maven仓库服务器这个library是否存在，如果是，gradle会获得请求library的路径，一般这个路径都是这样的形式：GROUP\_ID/ARTIFACT\_ID/VERSION\_ID。比如可以在<http://jcenter.bintray.com/com/squareup/otto/1.3.7> 和  <https://oss.sonatype.org/content/repositories/releases/com/squareup/otto/1.3.7/>

下获得com.squareup:otto:1.3.7的library文件。

然后Android Studio 将下载这些文件到我们的电脑上，与我们的项目一起编译。整个过程就是这么简单，一点都不复杂。

我相信你应该清楚的知道从仓库上下载的library只是存储在仓库服务器上的jar 或者aar文件而已。有点类似于自己去下载这些文件，拷贝然后和项目一起编译。但是使用gradle依赖管理的最大好处是你除了添加几行文字之外啥也不做。library一下子就可以在项目中使用了。

**了解aar文件**

等等，我刚才说了仓库中存储的有两种类型的library：jar 和 aar。jar文件大家都知道，但是什么是aar文件呢？

aar文件时在jar文件之上开发的。之所以有它是因为有些Android Library需要植入一些安卓特有的文件，比如AndroidManifest.xml，资源文件，Assets或者JNI。这些都不是jar文件的标准。

因此aar文件就时发明出来包含所有这些东西的。总的来说它和jar一样只是普通的zip文件，不过具有不同的文件结构。jar文件以classes.jar的名字被嵌入到aar文件中。其余的文件罗列如下：

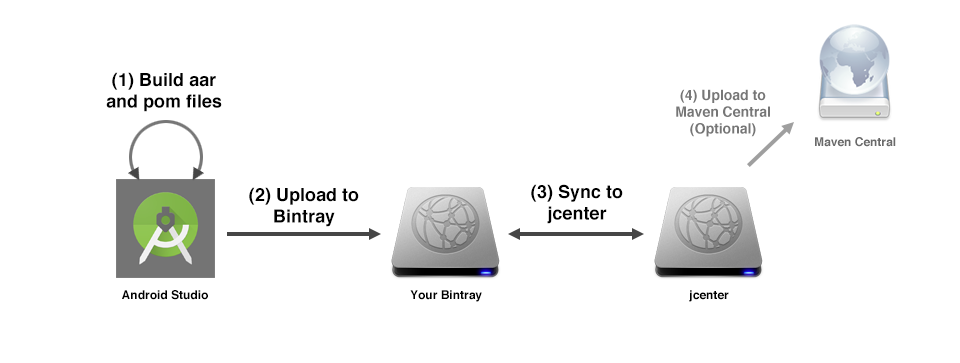
- /AndroidManifest.xml (mandatory)  
- /classes.jar (mandatory)  
- /res/ (mandatory)  
- /R.txt (mandatory)  
- /assets/ (optional)  
- /libs/\*.jar (optional)  
- /jni/<abi>/\*.so (optional)  
- /proguard.txt (optional)  
- /lint.jar (optional)

可以看到.aar文件是专门为安卓设计的。因此这篇文章将教你如何创建与上传一个aar形式的library。

**如何上传library到jcenter**

我相信你已经知道了仓库系统的大体工作原理。现在我们来开始最重要的部分：上传。这个任务和如何上传library文件到[http://jcenter.bintray.com](http://jcenter.bintray.com/)一样简单。如果做到，这个library就算发布了。好吧，有两个需要考虑：如何创建aar文件以及如何上传构建的文件到仓库。

虽然需要若干步骤，但是我还是想强调这事并不复杂，因为已经准备好了所有事情。整个过程如下图：



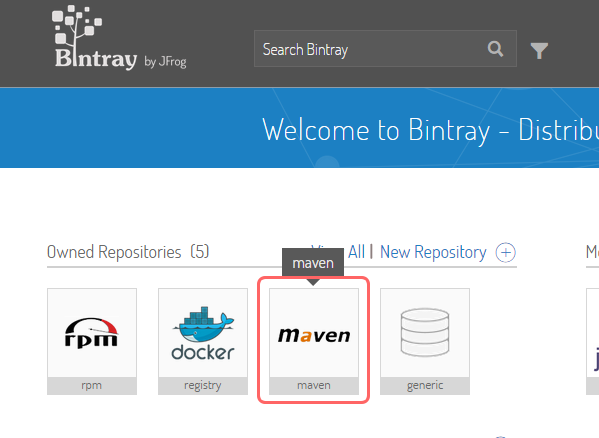
因为细节比较多，我分为7部分，一步一步的详细解释清楚。

**第一部分：在bintray上创建package**

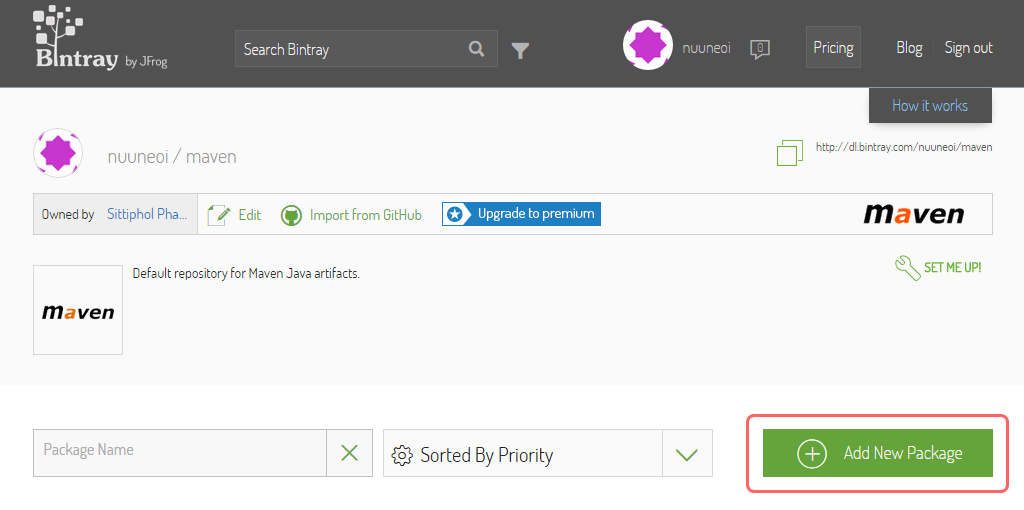
首先，你需要在bintray上创建一个package。为此，你需要一个bintray账号，并在网站上创建一个package。

第一步：在[bintray.com](https://bintray.com/)上注册一个账号。（注册过程很简单，自己完成）

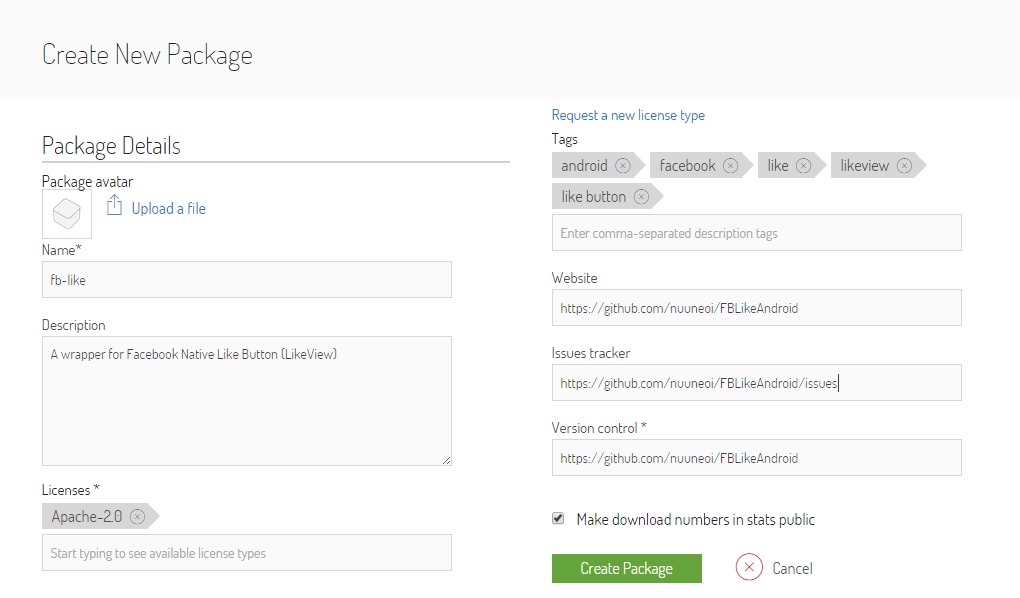
第二步：完成注册之后，登录网站，然后点击maven。



第三步：点击Add New Package，为我们的library创建一个新的package。



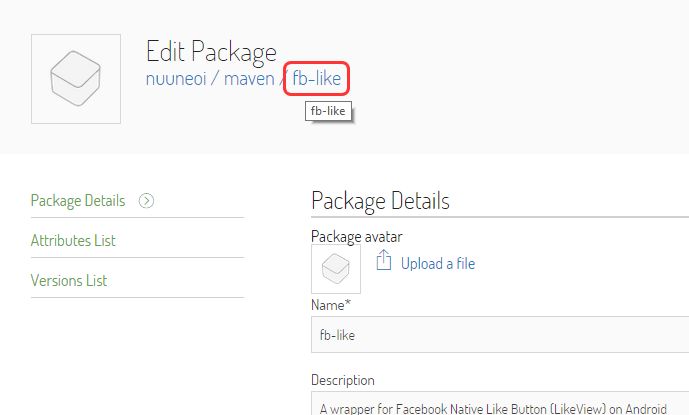
第四步：输入所有需要的信息



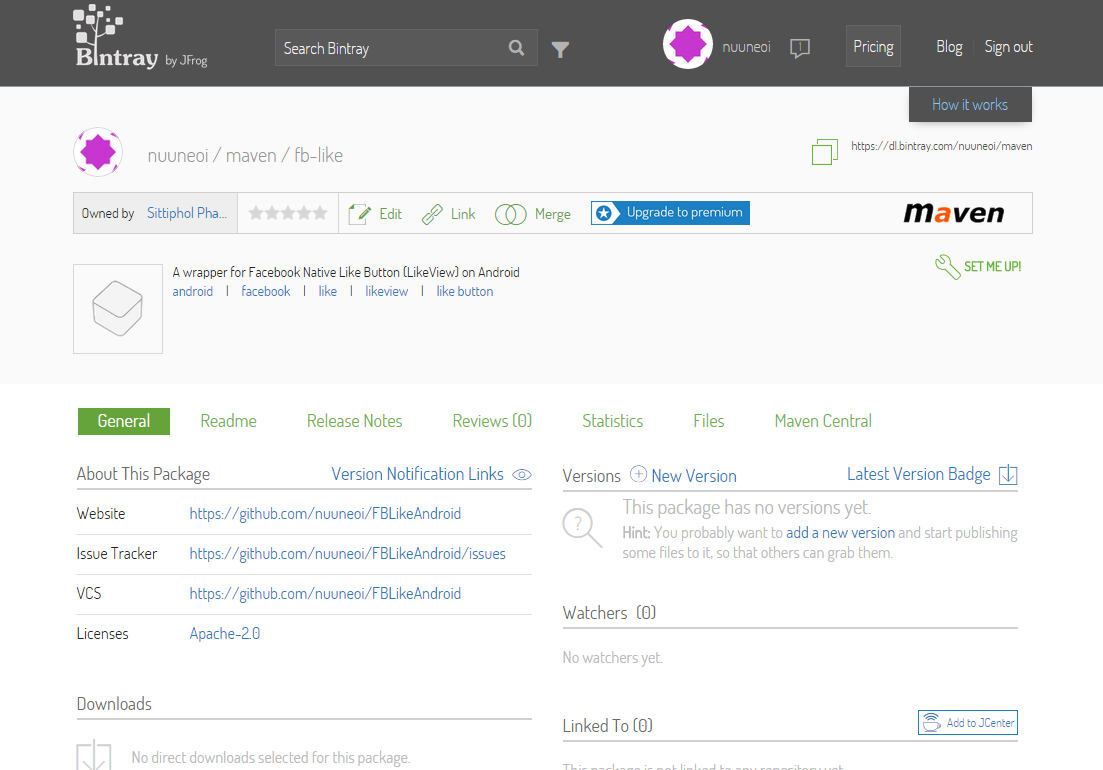
虽然如何命名包名没有什么限定，但是也有一定规范。所有字母应该为小写，单词之间用－分割，比如，fb-like。

当每项都填完之后，点击Create Package。

第五步：网页将引导你到 Package编辑页面。点击 Edit Package文字下的Package名字，进入Package详情界面。



完工！现在你有了自己在Bintray上的Maven仓库，可以准备上传library到上面了。



Bintray账户的注册就完成了。下一步是Sonatype，Maven Central 的提供者。

**第二部分：为Maven Central创建个Sonatype帐号**

注：如果你不打算把library上传到Maven Central，可以跳过第二和第三部分。不过我建议你不要跳过，因为仍然有许多开发者在使用这个仓库。

和jcenter一样，如果你想通过Maven Central,贡献自己的library，你需要在提供者的网站Sonatype上注册一个帐号。

你需要知道的就是这个帐号，你需要在Sonatype网站上创建一个IRA Issue Tracker 帐号。请到[Sonatype Dashboard](https://issues.sonatype.org/secure/Dashboard.jspa) 注册这个帐号。

完成之后。你需要请求得到贡献library到Maven Central的权限。不过这个过程对我来说有点无厘头，因为你需要做的就是在JIRA中创建一个issue，让它们允许你上传匹配Maven Central提供的GROUP\_ID的library。

要创建上述所讲到的issue，访问[Sonatype Dashboard](https://issues.sonatype.org/secure/Dashboard.jspa)，用创建的帐号登录。然后点击顶部菜单的Create。

填写如下信息：

Project: Community Support - Open Source Project Repository Hosting

Issue Type: New Project

Summary: 你的 library名称的概要，比如The Cheese Library。

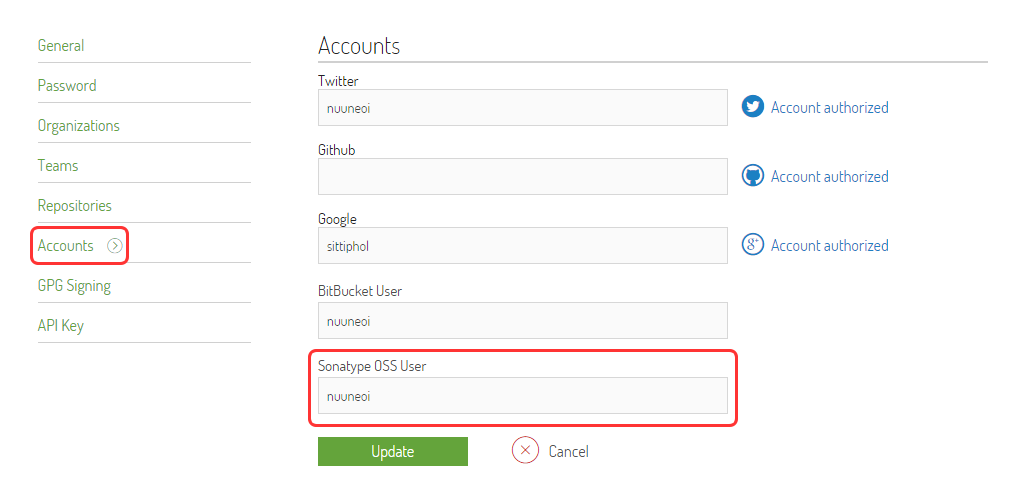
Group Id: 输入根GROUP\_ID，比如，com.inthecheeselibrary 。一旦批准之后，每个以com.inthecheeselibrary开始的library都允许被上传到仓库，比如com.inthecheeselibrary.somelib。

Project URL: 输入任意一个你想贡献的library的URL，比如， <https://github.com/nuuneoi/FBLikeAndroid>。

SCM URL: 版本控制的URL，比如 <https://github.com/nuuneoi/FBLikeAndroid.git>。

其余的不用管，然后点击Create。现在是最难的部分...耐心等待...平均大概1周左右，你会获准把自己的library分享到 Maven Central。

最后一件事是在[Bintray Profile](https://bintray.com/profile/edit)的帐户选项中填写自己的Sonatype OSS用户名。



点击Update，完成。

**第三部分：启用bintray里的自动注册**

就如我上面提到的，我们可以通过jcenter上传library到Maven Central ，不过我们需要先注册这个library。bintray提供了通过用户界面让library一旦上传后自动注册的机制。

第一步是使用下面的命令行产生一个key。（如果你用的是windows，请在[cygwin](https://www.cygwin.com/)下做这件事情）

gpg --gen-key

有几个必填项。部分可以采用默认值，但是某些项需要你自己输入恰当的内容，比如，你的真实名字，密码 等等。

创建了key之后，调用如下的命令查看被创建key的信息。

gpg --list-keys

如果没没问题的话，可以看到下面的信息：

pub 2048R/01ABCDEF 2015-03-07

uid Sittiphol Phanvilai <yourmail@email.com>

sub 2048R/98765432 2015-03-07

现在你需要把key上传到keyserver让它发挥作用。为此，请调用如下的命令并且将其中的PUBLIC\_KEY\_ID替换成上面pub一行中2048R/ 后面的 8位16进制值，譬如本例是01ABCDEF。

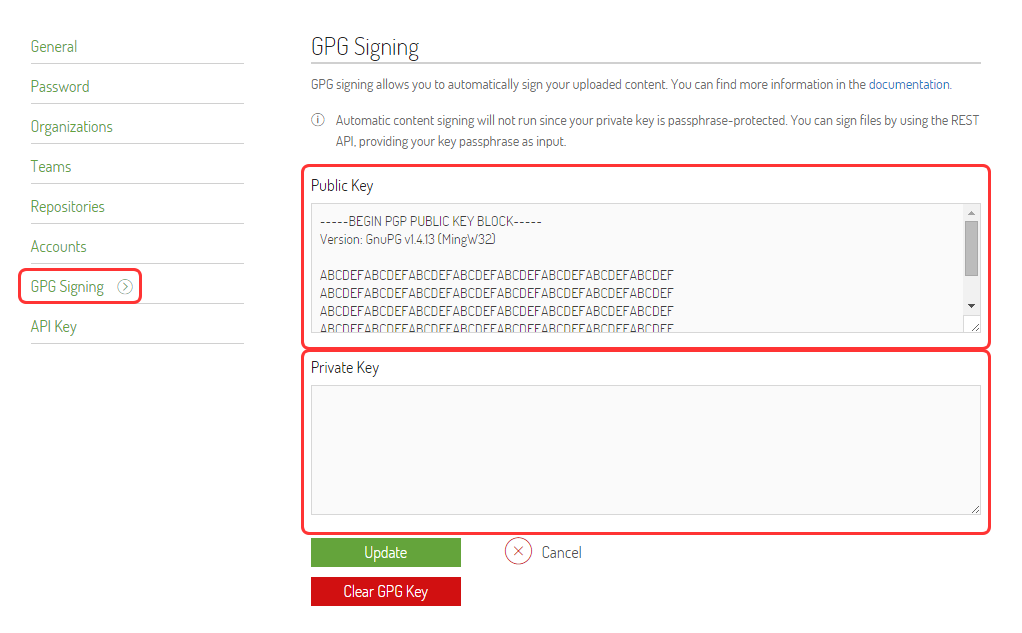
gpg --keyserver hkp://pool.sks-keyservers.net --send-keys PUBLIC\_KEY\_ID

然后，使用如下的命令以ASCII形式导出公共和私有的key，请将yourmail@email.com替换成你前面用于创建key的email。

gpg -a --export yourmail@email.com > public\_key\_sender.asc

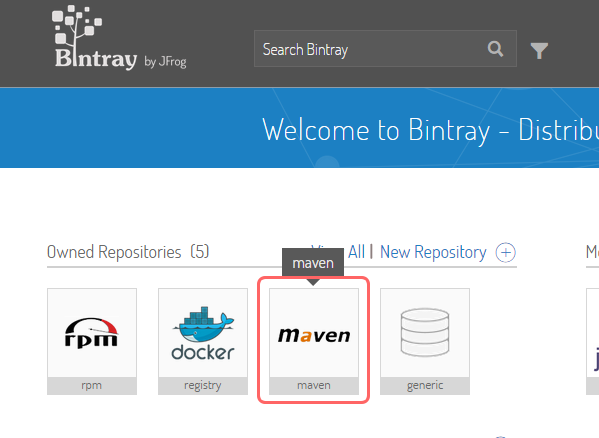
gpg -a --export-secret-key yourmail@email.com > private\_key\_sender.asc

打开Bintray的[Edit Profile](https://bintray.com/profile/edit)页面点击GPG 注册。分别在Public Key和 Private Key中填入上一步导出的public\_key\_sender.asc和 private\_key\_sender.asc文件中的内容。



点击Update保存这些key。

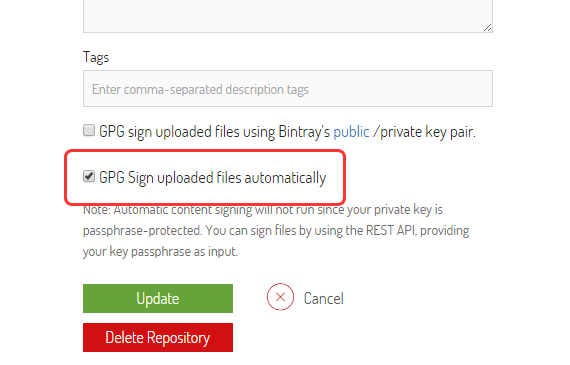
最后一步就是启用自动注册。到[Bintray](https://bintray.com/)的主页点击maven。



点击编辑



勾选中GPG Sign uploaed files automatically以启用自动注册。



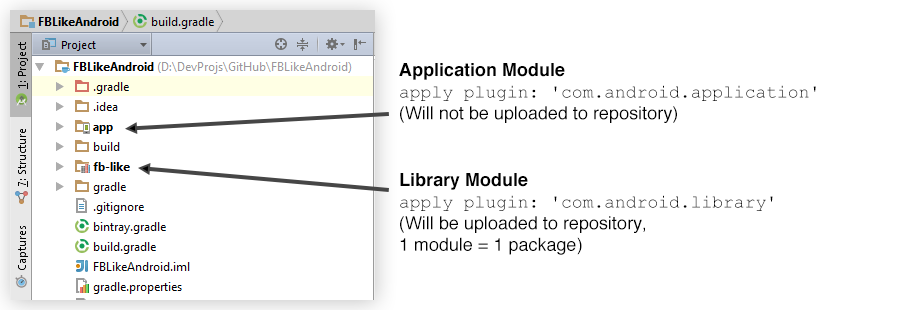
点击Update保存这些步骤。完成。现在只需点击一下，每个上传到我们Maven仓库的东西都会自动注册并做好转向Maven Central 。

请注意这是一次性的操作，以后创建的每一个library都要应用此操作。

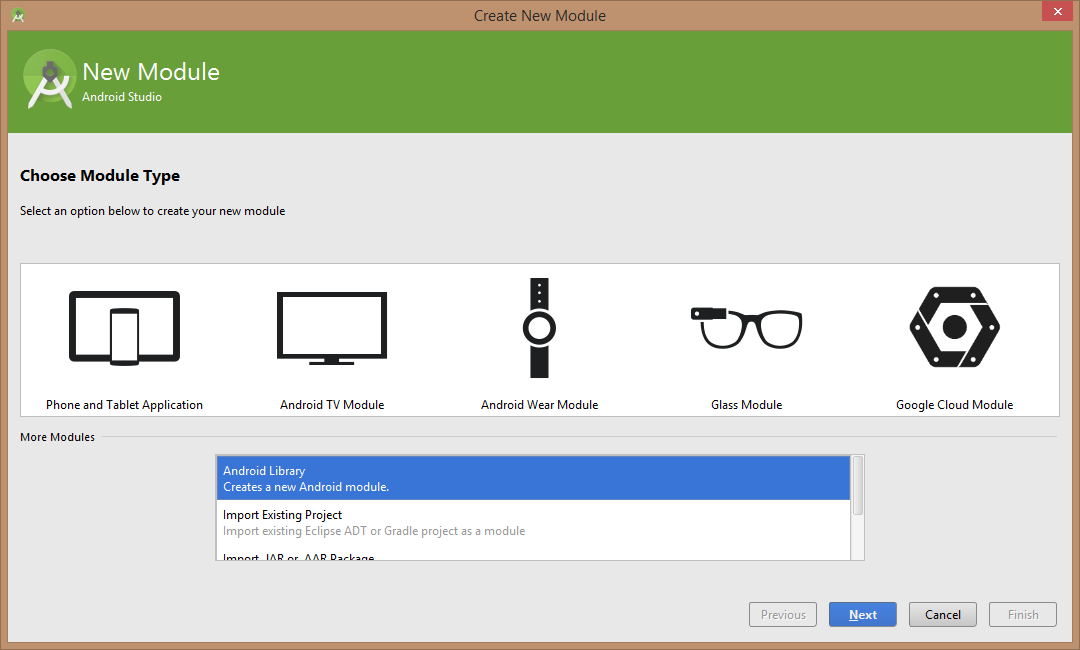
Bintray和Maven Central 已经准备好了。现在转到Android Studio部分。

**第四部分：准备一个Android Studio项目**

很多情况下，我们需要同时上传一个以上的library到仓库，也可能不需要上传东西。因此我建议最好将每部分分成一个Module。最好分成两个module，一个Application Module一个Library Module。Application Module用于展示库的用法，Library Module是library的源代码。如果你的项目有一个以上的library，尽管创建另外的module：1个 module对应1 个library。



我相信大家知道如何创建一个新的module，因此就不会深入讲解这个问题了。其实很简单，基本就是选择creating an Android Library module ，然后就完了。



下一步是把bintray插件应用在项目中。我们需要修改项目的build.gradle文件中的依赖部分，如下：

dependencies {

classpath 'com.android.tools.build:gradle:1.2.3'

classpath 'com.jfrog.bintray.gradle:gradle-bintray-plugin:1.2'

classpath 'com.github.dcendents:android-maven-plugin:1.2'

}

有一点非常重要，那就是gradle build tools的版本设置成1.1.2以上，因为以前的版本有严重的bug，我们将使用的是最新的版本1.2.3。

接下来我们将修改local.properties。在里面定义api key的用户名以及被创建key的密码，用于bintray的认证。之所以要把这些东西放在这个文件是因为这些信息时比较敏感的，不应该到处分享，包括版本控制里面。幸运的是在创建项目的时候local.properties文件就已经被添加到.gitignore了。因此这些敏感数据不会被误传到git服务器。

下面是要添加的三行代码：

bintray.user=YOUR\_BINTRAY\_USERNAME

bintray.apikey=YOUR\_BINTRAY\_API\_KEY

bintray.gpg.password=YOUR\_GPG\_PASSWORD

bintray username 放在第一行， API Key放在第二行， API Key可以在[Edit Profile](https://bintray.com/profile/edit)页面的API Key 选项卡中找到。

最后一行是创建 GPG key的密码。保存并关闭这个文件。

最后要修改的是module的build.gradle文件。注意前面修改的是项目的build.gradle文件。打开它，在apply plugin: 'com.android.library'之后添加这几行，如下：

[复制代码](javascript:void(0);)

pply plugin: 'com.android.library'

ext {

bintrayRepo = 'maven'

bintrayName = 'fb-like'

publishedGroupId = 'com.inthecheesefactory.thecheeselibrary'

libraryName = 'FBLike'

artifact = 'fb-like'

libraryDescription = 'A wrapper for Facebook Native Like Button (LikeView) on Android'

siteUrl = 'https://github.com/nuuneoi/FBLikeAndroid'

gitUrl = 'https://github.com/nuuneoi/FBLikeAndroid.git'

libraryVersion = '0.9.3'

developerId = 'nuuneoi'

developerName = 'Sittiphol Phanvilai'

developerEmail = 'sittiphol@gmail.com'

licenseName = 'The Apache Software License, Version 2.0'

licenseUrl = 'http://www.apache.org/licenses/LICENSE-2.0.txt'

allLicenses = ["Apache-2.0"]

}

[复制代码](javascript:void(0);)

bintrayRepo使用默认的，即maven。bintrayName修改成你上面创建的 package name。其余的项也修改成和你library信息相匹配的值。有了上面的脚本，每个人都能通过下面的一行gradle脚本使用这个library。

compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3‘

最后在文件的后面追加两行如下的代码来应用两个脚本，用于构建library文件和上传文件到bintray（为了方便，我直接使用了github上连接到相关文件的链接）：

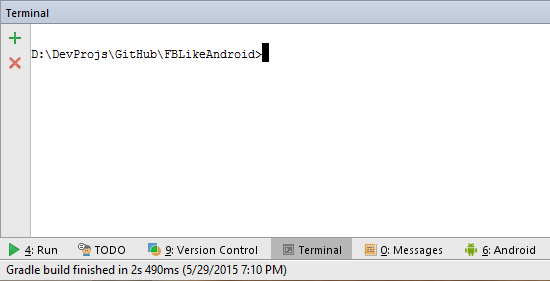
apply from: 'https://raw.githubusercontent.com/nuuneoi/JCenter/master/installv1.gradle'

apply from: 'https://raw.githubusercontent.com/nuuneoi/JCenter/master/bintrayv1.gradle'

完成！你的项目现在设置好了，准备上传到bintray吧！

**第五部分：把library上传到你的bintray空间**

现在是上传library到你自己的bintray仓库上的时候了。请到Android Studio的终端（Terminal）选项卡。



第一步是检查代码的正确性，以及编译library文件（aar，pom等等），输入下面的命令：

gradlew install

如果没有什么问题，会显示：

BUILD SUCCESSFUL

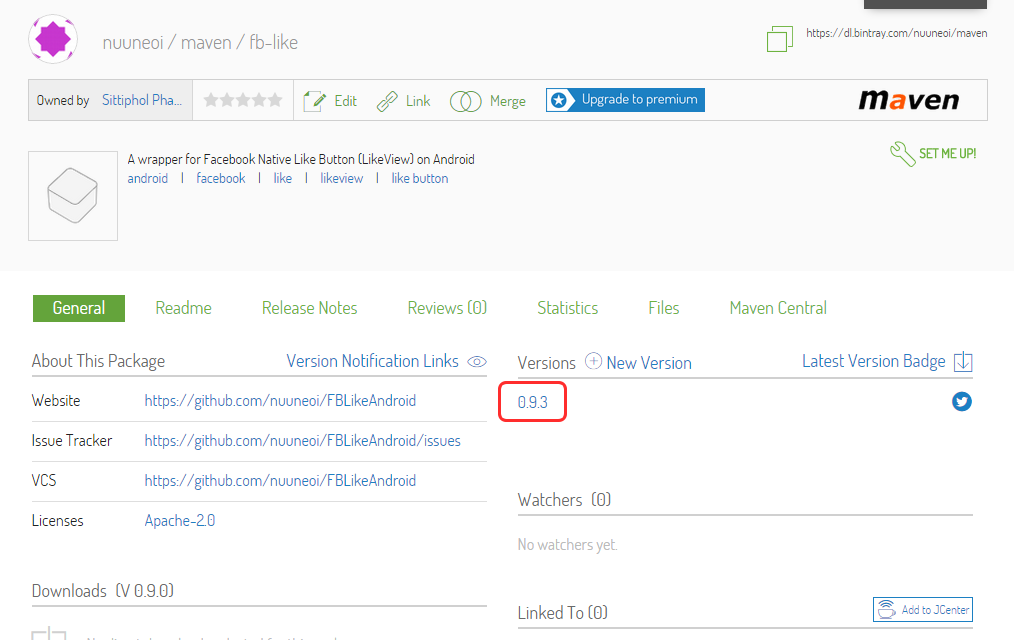
现在我们已经成功一半了。下一步是上传编译的文件到bintray，使用如下的命令：

gradlew bintrayUpload

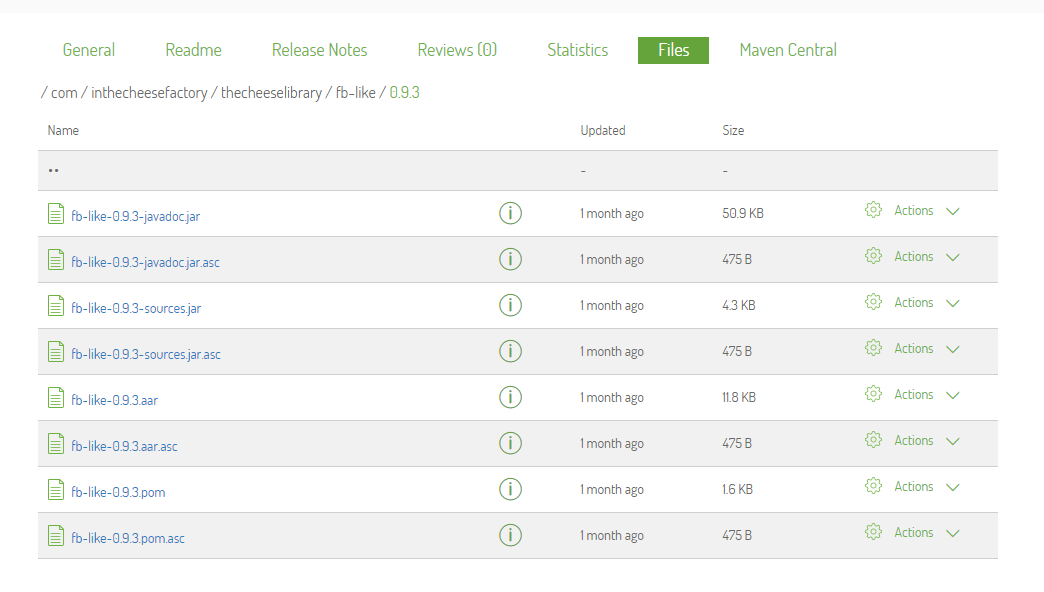
如果显示如下你就大喊一声eureka吧！

SUCCESSFUL

在bintray的网页上检查一下你的package。你会发现在版本区域的变化。



点击进去，进入Files选项卡，你会看见那里有我们所上传的library文件。



恭喜，你的library终于放在了互联网上，任何人都可以使用了！

不过也别高兴过头，library现在仍然只是在你自己的Maven仓库，而不是在jcenter上。如果有人想使用你的library，他必须定义仓库的url，如下：

[复制代码](javascript:void(0);)

repositories {

maven {

url 'https://dl.bintray.com/nuuneoi/maven/'

}

}

...

dependencies {

compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3'

}

[复制代码](javascript:void(0);)

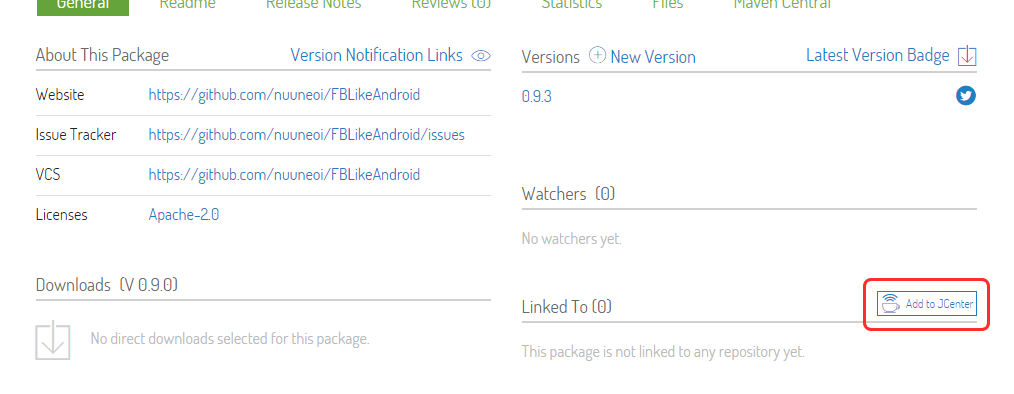
译者注：前面都没怎么看懂，看到上面的代码之后一下子全懂了，呵呵。

你可以在bintray的web界面找到自己Maven仓库的url，或者直接吧nuuneoi替换成你的bintray用户名（因为前面部分其实都是一样的）。我还建议你直接访问那个链接，看看里面到底是什么。

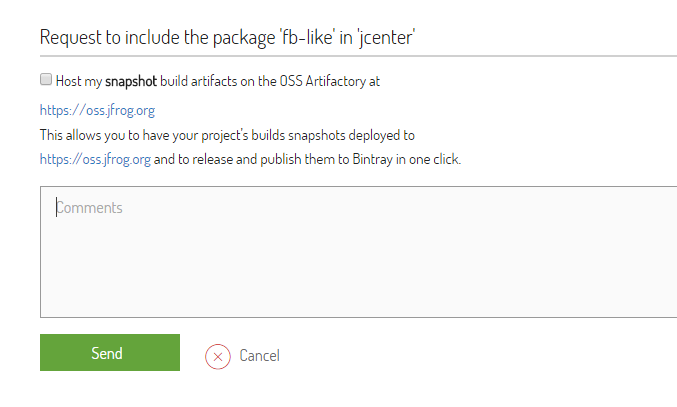
但是，就如我们前面所讲的那样，让开发者去定义url这种复杂的事情并不是分享library的最佳方式。想象一下，使用10个library不得添加10个url？所以为了更好的体验，我们把library从自己的仓库传到jcenter上。

**第六部分：同步bintray用户仓库到jcenter**

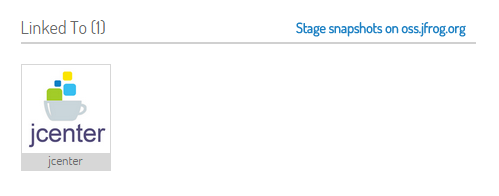
把library同步到jcenter非常容易。只需访问网页并点击Add to JCenter



什么也不做直接点击Send。



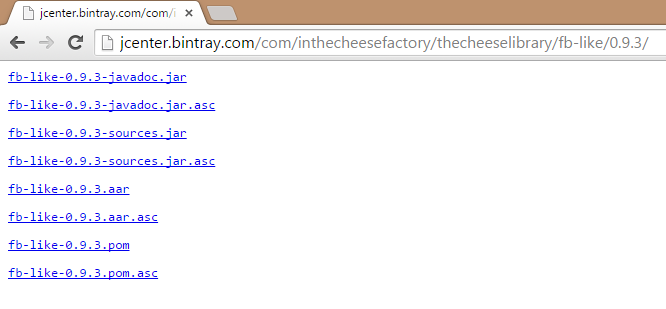
现在我们所能做的就是等待bintray团队审核我们的请求，大概2-3个小时。一旦同步的请求审核通过，你会收到一封确认此更改的邮件。现在我们去网页上确认，你会在 Linked To 部分看到一些变化。



从此之后，任何开发者都可以使用jcenter() repository 外加一行gradle脚本来使用我们的library了

|  |  |
| --- | --- |
| 1 | compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3' |

想检查一下自己的library在jcenter上是否存在？你可以直接访问[http://jcenter.bintray.com](http://jcenter.bintray.com/)，然后进入和你library的group id 以及artifact id匹配的目录。在本例中就是com -> inthecheesefactory -> thecheeselibrary -> fb-like -> 0.9.3。



请注意链接到jcenter是一个只需做一次的操作。如果你对你的package做了任何修改，比如上传了一个新版本的binary，删除了旧版本的binary等等，这些改变也会影响到jcenter。不过毕竟你自己的仓库和jcenter在不同的地方，所以需要等待2－3分钟让jcenter同步这些修改。

同时注意，如果你决定删除整个package，放在jcenter仓库上的library不会被删除。它们会像僵尸一样的存在，没有人再能删除它了。因此我建议，如果你想删除整个package，请在移除package之前先在网页上删除每一个版本。

**第七部分：上传library到Maven Central**

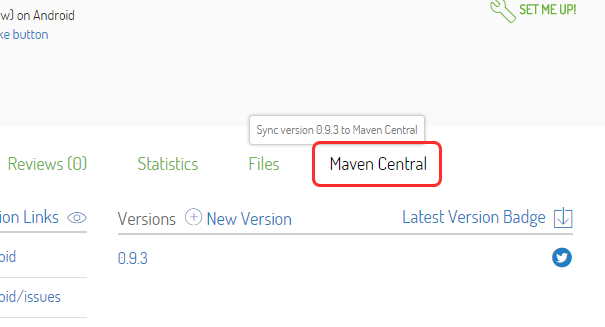
并不是每个安卓开发者都使用jcenter。仍然有部分开发者还在使用mavenCentral() ，因此让我们也把library上传到Maven Central 吧。

要从jcenter到Maven Central，首先需要完成两个任务：

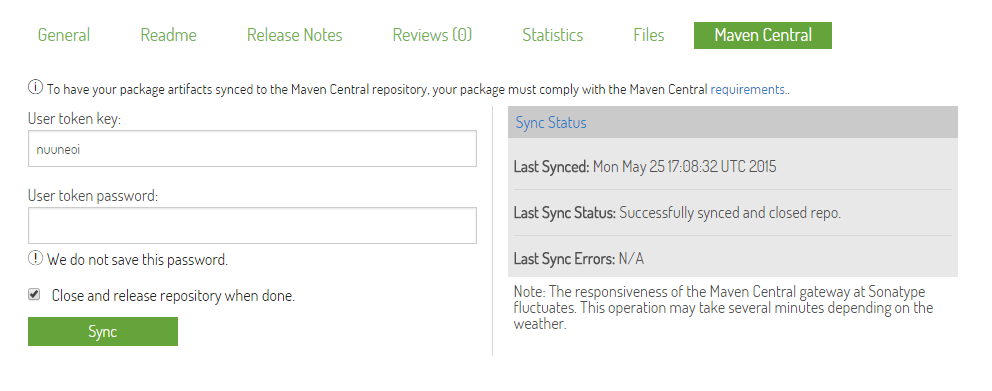
1) Bintray package 已经连接到jcenter。

2) Maven Central上的仓库已经认证通过

如果你已经通过了这些授权，上传library package到Maven Central就异常简单了，只需在package的详情页面点击Maven Central 的链接。



输入你的Sonatype用户名和密码并点击Sync。



如果成功，在Last Sync Status中会显示Successfully synced and closed repo（见图），但是如果遇到任何问题，则会在Last Sync Errors显示出来。你需要根据情况修复问题，能上传到Maven Central 的library的条件是相当严格的，比如+ 号是不能在ibrary版本的依赖定义中使用的。

完成之后，你可以在  [Maven Central Repository](https://oss.sonatype.org/content/repositories/releases/) 上找到你的library。在那些匹配你ibrary的group id以及artifact id的目录中。比如本例中就是com -> inthecheesefactory -> thecheeselibrary -> fb-like -> 0.9.3。

恭喜！虽然需要许多步骤，但是每一步都很简单。而且大部分操作都是一劳永逸的。

如此长篇的文章！希望对你有所帮助。我的英语也许有点晦涩，不过希望至少内容是可以理解的。

期待能在上面看到你的library大作！

分类: [android](http://www.cnblogs.com/sihaixuan/category/636724.html)

标签: [Android maven仓库](http://www.cnblogs.com/sihaixuan/tag/Android%20maven%E4%BB%93%E5%BA%93/)