



目录视图

摘要视图

RSS 订阅

个人资料



超越我来着

关注

发私信



访问：277808次

积分：2828

等级：BLOG > 5

排名：第14144名

原创：54篇

转载：87篇

图灵赠书——程序员11月书单 【思考】Python这么厉害的原因竟然是！ 感恩节赠书：《深度学习》等异步社区优秀图书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈

拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建

2016-03-02 15:39

7702人阅读

评论(1)

收

分类：

android studio (10)

目录(?)

[+]

关闭

使用、创造和分享

笔者曾经不量力的思考过『是什么推动了互联网技术的快速发展了摩尔定律之外，技术经验的快速积累和广泛分享，也是重要的

有人戏称，『写 Java，首先要学会选包』，在这里不好评论。一层意思：首先 Java 有大量的现成的依赖包，不必要自己造轮子中，集成方式也方便。

笔者从事 Android 和 Java 开发以来，经历了几个阶段：



魔方公寓



译文： 0篇

评论： 32条

文章搜索

文章分类

- android (99)
- android IPC (4)
- android activity (9)
- android view (12)
- android 自定义view (9)
- android 消息机制 (5)
- android handle (4)
- android 线程 (4)
- android 内存 (4)
- android 安全 (6)
- android JNI (6)
- android 性能优化 (5)
- android 数据库sql (8)
- android studio (11)
- android 测试 (3)
- java (21)
- c (3)
- html (2)
- git (3)
- eclipse (8)
- linux (3)

[闭门造轮子](#) > [使用别人的轮子](#) > [开门造轮子](#) > [分享轮子](#)

在使用、创造、分享轮子的过程中，maven 仓库的使用可谓必备技能。

相信各位使用 Android Studio，对于 `jcenter()`、`mavenCentral()` 等概念应该是司空见惯了。程序员要知其然，知其所以然。本篇将按照如下脉络介绍在 Android Studio 中 Maven 仓库相关的概念和应用。

Maven 包

Maven 仓库

发布包到本地仓库

发布包到 Bintray Jcenter 远程仓库

发布包到 Sonatype MavenCentral 远程仓库

搭建私有 Sonatype 仓库

搭建私有 Artifactory 仓库

Maven 包 (Package)

至于 Maven 是什么，请参考 [Apache Maven](#)。

对于 Android 开发者而言，只需要知道 Maven 是一种构建工具 POM (Project Object Model) 所定义的文件包格式即可。

Gradle 可以使用 Maven 包，而且大部分的 Android 能够使用

先来看一个托管在某仓库上的 Maven 包：Bugtags-Android-1

- 1 bugtags-lib-1.1.0-javadoc.jar//javadoc 文件
- 2 bugtags-lib-1.1.0-javadoc.jar.asc//javadoc 文件的签名
- 3 bugtags-lib-1.1.0-sources.jar//源码文件
- 4 bugtags-lib-1.1.0-sources.jar.asc//源码文件的签名

关闭



文章存档

2016年04月 (1)

2016年03月 (6)

2016年02月 (1)

2016年01月 (2)

2015年12月 (3)

展开

阅读排行

android 4.0 BLE开发官方文档... (41434)

Android 蓝牙4.0 BLE 理解 (29163)

android studio使用问题及说明 (28335)

android scroller overscroller用法 (9622)

android 手机获取外置SD卡路径 (9375)

拥抱 Android Studio 之四：Ma... (7700)

更新mac自带的python2.7到3.4... (7655)

Android 分析内存的使用情况 (6964)

android sql XUtils DB模块的性... (6662)

mac 使用adb (5854)

评论排行

android 4.0 BLE开发官方文档... (13)

Android 分析内存的使用情况 (2)

Android Eclipse Ant 编译打包 (2)

android scroller overscroller用法 (2)

- 5 bugtags-lib-1.1.0.aar//Android Library 的主文件包
- 6 bugtags-lib-1.1.0.aar.asc//主文件包的签名
- 7 bugtags-lib-1.1.0.pom//包描述文件
- 8 bugtags-lib-1.1.0.pom.asc//描述文件的签名

对于一个符合规范的 Maven Package , pom 文件、aar (或者 jar) 文件 是必须的。

而 javadoc 文件、源码文件、签名文件都不是必要的，但是 某些公开仓库 (如 mavenCentral) 有此要求。

使用这个包的方式，相信大家已经很熟悉了：

```
1 dependencies {
2     compile 'com.bugtags.library:bugtags-lib:1.1.0'
3 }
```

POM 文件

一个 Maven Package ，最重要就是 POM (Project Object Model) 文件，这其实是一件，这里截取 Bugtags-Android-Lib POM 主要内容如下：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xsi:schemaLocation="http://maven.apache.org/P
3 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="h
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
5     <modelVersion>4.0.0</modelVersion>
6     <groupId>com.bugtags.library</groupId>
7     <artifactId>bugtags-lib</artifactId>
8     <version>1.1.0</version>
9     <packaging>aar</packaging>
10    <dependencies>
11        <dependency>
12            <groupId>com.android.support</groupId>
```

关闭



Android 蓝牙4.0 BLE 理解	(2)
mac 使用adb	(1)
Android任务和返回栈完全解析	(1)
android wearable 研究	(1)
AsyncTask 研究二 内部原理	(1)
android中gps的打开关闭	(1)

推荐文章

- * 【2017年11月27日】CSDN博客更新周报
- * 【CSDN】邀请您来GitChat赚钱啦！
- * 【GitChat】精选——JavaScript进阶指南
- * 改做人工智能之前，90%的人都没能给自己定位
- * TensorFlow 人脸识别网络与对抗网络搭建
- * Vue 移动端项目生产环境优化
- * 面试必考的计算机网络知识点梳理

最新评论

- android 4.0 BLE开发官方文档介绍
weixin_39237314：这个方法过时了。。
- android 4.0 BLE开发官方文档介绍
lovebkl：不懂就不要误人子弟
- android 4.0 BLE开发官方文档介绍
天二书生：iOS,Android蓝牙问题可加Q 3414 990373，免费分析交流
- android 4.0 BLE开发官方文档介绍
轩辕嘻嘻_飞：能不能赏赐小弟个源码,22741 77609@qq.com,谢谢了哈

```

13     <artifactId>support-v4</artifactId>
14     <version>19.0.0</version>
15     <scope>compile</scope>
16 </dependency>
17 </dependencies>
</project>

```

modelVersion: 从 mvn 2.x 起，这个数值都是4.0.0

packaging：打包方式，aar 是 Android Library 的打包方式，常用的还有 jar

dependency：声明依赖列表

包的唯一标示：

```

1  <!--包组 id，通常是发布者拥有的域名的反向，以免跟别人的重复-->
2  <groupId>com.bugtags.library</groupId>
3  <!--包 artifactId，不好意思我也不知道如何准确翻译，其实就是组以下应该有一个更!
4  <artifactId>bugtags-lib</artifactId>
5  <!--包版本-->
6  <version>1.1.0</version>

```

其中三个字段与 Gradle 的依赖格式 'com.bugtags.library:bugtags-lib:1.1.0' 冒号.....

一一对应。这就解释了所谓的 Gradle 兼容 Maven 包。

Maven 仓库

Maven 包集中存放的地方，就是 Maven 仓库。这些仓库，可程服务器上。可以是私有仓库，也可以是公开的。下面是笔

```

1  mavenCentral();
2  jcenter()
3  maven {
4      url 'file:///Users/my-user-name/Documents/Android/re
5  }

```



关闭

[android 4.0 BLE开发官方文档介绍](#)

轩辕嘻嘻_飞：小弟最近看ble的东西,能不能赏赐个源码哈 2274177609@qq.com,谢谢了哈

[android 4.0 BLE开发官方文档介绍](#)

lirui319：现在正在做AndroidBLE的开发，同时有两个特征接收通知，但是onCharacteristic C...

[android textview html font标签不好用](#)

冰点k：太好了，正好解决了我现在的问题，大赞！

[Android 蓝牙通信](#)

小生之独家记忆：您的文章还没有来得及看到了您的头像 同时星迷

[Android任务和返回栈完全解析](#)

zgengen：FLAG_ACTIVITY_CLEAR_TOP 这个flag 写错了，会重新创建新的activity

[android 4.0 BLE开发官方文档介绍](#)

帝都-辉：可以的。



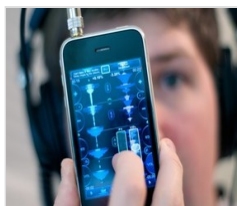
润滑剂怎么使用



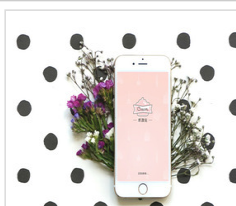
笔记本租赁



上海展台搭建



开发一个app多



app开发报价单



笔记本租赁

```
6 maven {
7     url 'http://192.168.99.100:8081/content/repositories/releases/'
8 }
9
10 maven {
11     url "https://jitpack.io"
12 }
```

Android Studio Gradle 主要支持两个 Maven 中央库：`mavenCentral` 和 `jcenter`。

`mavenCentral` 是最早的 maven 中央仓库

`jcenter` 是 Android Studio 0.8 版本起的默认 maven 中央仓库

第三个是笔者的本机的仓库

第四个是笔者部署在内网服务器的私有仓库

第五个是可以把 github 项目发不成 maven 库（jitpack 本身是一个很酷的想法）

读者可能会发现两个问题：

为什么有了 `mavenCentral`，谷歌还切换到了 `jcenter`？

`maven { url : xxx }`，这种格式可以配置任何一个存在的仓库？

解释如下：

jcenter VS. mavenCentral

根据[这篇博客](#)，`jcenter` 具有如下优胜特点，使得谷歌进行切换

`jcenter` 是一个 `mavenCentral` 的超集，`jcenter` 还包含了其他 maven 仓库

`jcenter` 具有更好的 cdn，默认支持 https，这个对于谷歌有巨大吸引力

bintray（`jcenter` 的服务提供商）表示 `jcenter` 具有更好的性能

有数据表明 bintray `jcenter` 占用更少的本地缓存空间



魔方公寓



关闭



更好的交互界面，可以从 jcenter 向 mavenCentral 同步包（下面两个平台的使用教程将会证实这一点）

笔者亲测，在 bintray 上发布包到 jcenter 在易用性上的确比在 sonatype 发布到 mavenCentral 要好得多。

使用符合规范的 maven 仓库

没错，你可以通过 `maven { url : xxx }` 使用任何一个符合 maven 规范的仓库。

存在本地的

```
1 maven {
2     url 'file:///Users/my-user-name/Documents/Android/repo/'
3 }
```

存在内网服务器的

```
1 maven {
2     url 'http://192.168.99.100:8081/content/repositories/releases/'
3 }
```

存在某个远程服务器的

```
1 maven {
2     url 'https://raw.githubusercontent.com/liaohuqiu/ume
3 }
```

此仓库由 [liaohuqiu](#) 同学为方便大家使用友盟开发者工具，把相应管在 github 项目中。

发布 Maven 包



关闭





使用 maven 包相信已经很清楚了，让我们更进一步。

当我们在日常开发实践中，积累了一些公共库，想要固定下来，被自己或者别人方便的使用，就需要发布 maven 包。

一个符合规范的 maven 包至少包含 pom 文件和主文件包。难道这些都要手动编写和创建么？

答案是：有了 gradle 插件，你只需要干很少的事儿。

全局设定

下面以发布这系列包为示例：

groupId: com.as-gradle.demo //改成你的 groupId

artifactId: x //artifactId 会有些变化，这里先用 x 代替，下面会修改。

version: 1.0.0

也就是 'com.as-gradle.demo:x:1.0.0'

读者要进行练习的时候，最好改一下你的 groupId，否则可能会发布失败

关闭

下面使用到的示例工程已经放在了 [github](#) 上。

为了后面使用方便，首先在工程的项目 gradle.properties 是用生成 POM 文件，将会在通篇文章中用到：

```
1 # 包信息
2 PROJ_GROUP=com.as-gradle.demo
3 PROJ_VERSION=1.0.0
4
5 # 项目的描述
6 PROJ_WEBSITEURL=https://bugtags.com
```





```

7  PROJ_ISSUETRACKERURL=https://github.com/bugtags/Bugtags-Android/issues
8  PROJ_VCSURL=https://github.com/bugtags/Bugtags-Android.git
9  PROJ_DESCRIPTION=Simple and effective bug & crash reporting tool for Android apps
10
11 # Licence信息
12 PROJ_LICENCE_NAME=The Apache Software License, Version 2.0
13 PROJ_LICENCE_URL=http://www.apache.org/licenses/LICENSE-2.0.txt
14 PROJ_LICENCE_DEST=repo
15
16 # Developer 信息
17 DEVELOPER_ID=your-dev-id
18 DEVELOPER_NAME=your-dev-name
19 DEVELOPER_EMAIL=your-email@your-mailbox.com

```

发布包到本地仓库

创建一个 module：localrepo

将本地某个路径设置为仓库根目录：

/Users/your-user-name/Documents/Android/repo/ (Mac)

这里使用了一个叫做 your-user-name 的用户下的某个目录，请

为了优雅，在 localrepo 这个 module 的 gradle.properties 定

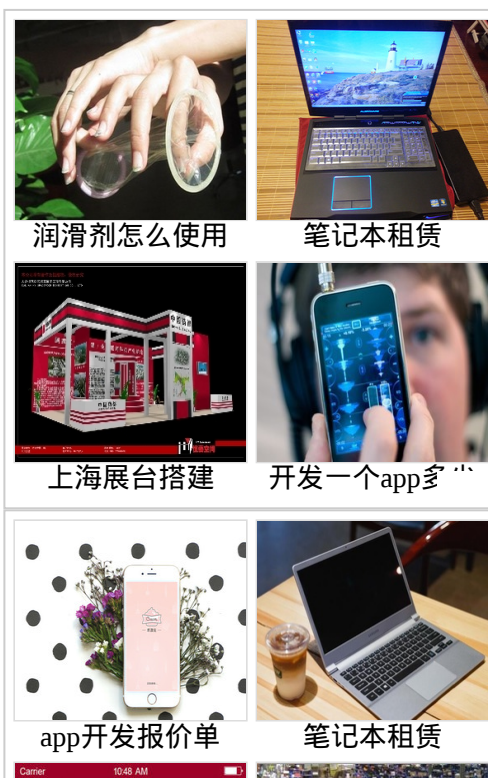
```

1  PROJ_NAME=localrepo
2  PROJ_ARTIFACTID=localrepo
3  PROJ_POM_NAME=Local Repository
4  LOCAL_REPO_URL=file:///Users/your-user-name/Documents/Android/repo/
5  #以上是 Mac 的本地路径，如果是 Windows，则是类似：
6  #LOCAL_REPO_URL=file:///C:/Users/cadmanager/Documents/Android/repo/

```

在 module 中应用和配置 maven plugin：





```

1  apply plugin: 'maven'
2
3  uploadArchives {
4      repositories.mavenDeployer {
5          repository(url: LOCAL_REPO_URL)
6          pom.groupId = PROJ_GROUP
7          pom.artifactId = PROJ_ARTIFACTID
8          pom.version = PROJ_VERSION    }
9  }

```

在控制台运行：

```
1  $ ./gradlew -p localrepo clean build uploadArchives --info
```

一切顺利的话，你的第一个本地包已经发布到设定的目录的本地仓库了：

```

1  |   | com
2  |   | as-gradle
3  |   |   | demo
4  |   |   | localrepo
5  |   |   |   | 1.0.0
6  |   |   |   |   | localrepo-1.0.0.aar
7  |   |   |   |   | localrepo-1.0.0.aar.md5
8  |   |   |   |   | localrepo-1.0.0.aar.sha1
9  |   |   |   |   | localrepo-1.0.0.pom
10 |   |   |   |   | localrepo-1.0.0.pom.md5
11 |   |   |   |   | localrepo-1.0.0.pom.sha1
12 |   |   |   | maven-metadata.xml
13 |   |   |   | maven-metadata.xml.md5
14 |   |   |   | maven-metadata.xml.sha1

```

使用本地包（两个疑问向读者请教）

关闭





要使用这个包，首先在项目的 build.gradle 中添加这个本地仓库：

```
1  allprojects {
2      repositories {
3          jcenter()
4      }
5      maven{
6          url 'file:///Users/your-user-name/Documents/Android/repo/'
7      }
8  }
9  }
```

在某个 module（如 demo 项目中的 app）的 build.gradle 中添加依赖：

```
1  compile 'com.as-gradle.demo:localrepo:1.0.0@aar'
```

这里有两个奇怪的地方，笔者也没有深入研究，初步猜测是 Android Studio 的 B 答案的读者请到我博客文章下留言赐教：

* 依赖末尾一般都需要加一个 '@aar'，在某些版本的 Android Studio 中，这并不重要，这是为什么？
* 另外，如果本地包本身使用了远程的依赖，也需要在使用本地包时加上 '@aar'，这又是为什么？

想要让更多的人使用到你的劳动成果，你就需要把 Maven 仓库上，而不是本机，接下来要介绍发布 Maven 到 jcenter 因为前者的使用简单，本着『先易后难，快速出成效』的

发布包到 Bintray jcenter 远程仓库

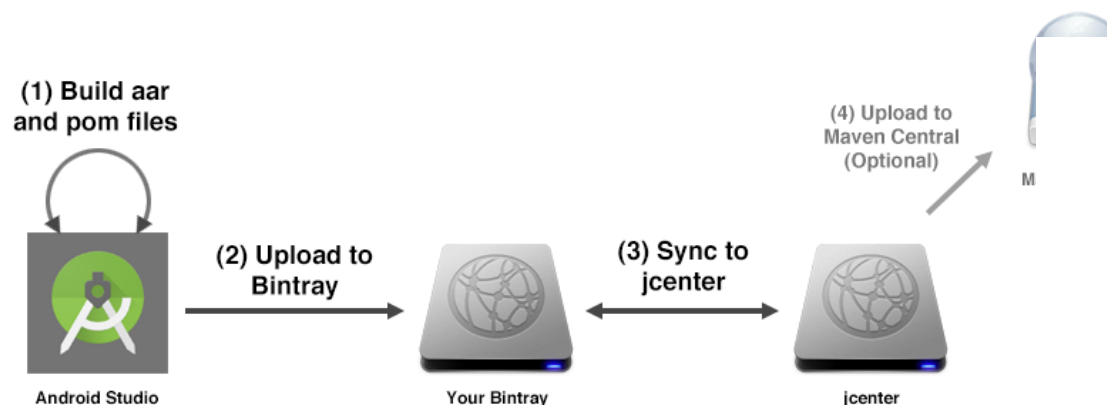




简介

jcenter 是由 [bintray](#) 提供的 maven 中央库托管服务，bintray 又是 [jfrog](#) 公司的一款产品。jfrog 是一个商业公司，通过提供高级服务盈利，又为普通开发者提供了足够用的免费基础功能（截止至 2016-01-24），笔者较为推崇这种开发者服务的商业模式。

引用一张图来表述 bintray 的工作方式



关闭

图片来源，<http://inthecheesefactory.com/>

使用 jcenter 需要在 bintray 上注册账号，在本地进行加密签名

1. 注册账号

登陆 jcenter [首页](#)

signin -> signup，填写表单，注意 username 这一栏是后面的 bintray 用户名

2. 创建 GPG 签名





前方高能预警：比较繁琐，切勿半途放弃

前面介绍过，可以把 bintray 的包同步到 mavenCentral，而后者需要对包文件进行签名，签名和验证过程需要用到一个叫做 GPG 的工具产生的公钥和私钥。[这里有适合多个平台的 GPG 程序](#)，下面只介绍 OSX 平台。

这种工具大概的意义是产生公钥和私钥，把公钥发送到公开的服务器，私钥用来产生包文件签名。包的使用者在拿到包文件之后，通过公钥来验证文件的有效性，运行具体原理参考[这里](#)。

下载 [gpgtool](#)，安装

检测安装成功，在命令行运行

```
1 $ gpg --version
2 gpg (GnuPG/MacGPG2) 2.0.28
3 libgcrypt 1.6.3
```

有类似的输出，就是正常安装了

产生证书，运行命令，按照提示输入

```
1 $ gpg --gen-key
```

检查结果

```
1 $ gpg --list-keys
```

找到刚才创建证书的时候，输入的相关信息那三行，例如：

```
1 pub 2048R/2E686B39 2015-06-02
2 uid [ultimate] Your Name <your-email@your-mailbox.co
3 sub 2048R/your-sub-key-id 2015-06-02
```

关闭





上传公钥到服务器，找到你的 pub 的那一行，2048R/后的那一串八位字符串，如上面的：2E686B39，就是公钥 ID

```
1 $ gpg --keyserver hkp://pool.sks-keyservers.net --send-keys your-public-key-id
```

输出公钥和私钥成文件

```
1 $ gpg -a --export your-email@your-mailbox.com > public_key_sender.asc
2 $ gpg -a --export-secret-key your-email@your-mailbox.com > private_key_sender.asc
```

配置本地 gradle 运行环境的属性，位于 `~/.gradle/gradle.properties`，添加内容：


```
1 signing.keyId=your-public-key-id
2 signing.password=your-gpg-password
3 signing.secretKeyRingFile=/Users/your-user-name/.gnupg/secring.gpg
```

bintray 本身可以通过在 [profile->GPG Sining](#) 中配置 public key 和 private key 来自动对上传的文在下图中，对应填入 `public_key_sender.asc` 与 `private_key_sender.asc` 的内容




关闭






Edit Your Profile


efaces (Kevin He)


- General
- Password
- Organizations
- Teams
- Repositories
- Accounts
- GPG Signing 
- API Key

GPG Signing

GPG signing allows you to automatically sign your uploaded content. You can find more information in the [documentation](#).

 Public Key

 Private Key

 Update Cancel

Clear GPG Key

设置 bintray maven 包自动签名

选取 maven 仓库首页，进入 edit：



关闭



Tags

☐ GPG sign uploaded files using Bintray's public/private key pair.

☒ GPG Sign uploaded files automatically

Note: Automatic content signing will not run since your private key is passphrase-protected. You can sign files by using the REST API, providing your key passphrase as input.

Update



Cancel

Delete Repository

最下面有两个选项：

- 1 GPG sign uploaded files using Bintray's public/private key pair.
- 2 GPG Sign uploaded files automatically

因为咱们是希望使用自己的 key，所以勾选第二个。

3. 创建 bintray 项目

首页-> maven -> add new package，填入对应的信息，其中上传的时候，使用的项目名称，例如：bintrayaar，这是要上传包的方式有 点点不一样，下面有介绍。

Package avatar



Upload a file

Name*

bintrayaar



关闭



Description

demo aar package for bintray

Licenses *

Apache-2.0 (x)

Start typing to see available license types

Request a new OSS license type

Tags

Enter comma-separated description tags

Website

http://kvh.io

Issues tracker

https://github.com/kevinho/Embrace-Android-Studio-Demo

Version control *

https://github.com/kevinho/Embrace-Android-Studio-Demo

☐ Make download numbers in stats public

Create Package

(x) Cancel

4. 配置插件

关闭





bintray 官方在 github 上托管了 [bintray-examples](#)，方便用户使用 gradle 上传包。

因为这里要上传的是 aar 格式的包，所以，具体是参考 [gradle-aar-example](#) 例子，然而例子有一些地方没有更新，请注意下面的描述。

在项目中创建 local.properties 来配置 bintray 登陆信息以及 gpg 证书密码

- 1 bintray.user=your-bintray-user
- 2 bintray.apikey=your-bintray-apikey
- 3 bintray.gpg.password=your-gpg-password

其中 your-bintray-user 就是 bintray 右上角显示的用户名称，your-bintray-apikey 在 profile->API Key 可以找到，your-gpg-password 则是创建 gpg 证书的时候的密码

在项目的 build.gradle 配置 buildscript 的 classpath

```

1  buildscript {
2      repositories {
3          jcenter()
4      }
5      dependencies {
6          classpath 'com.android.tools.build:gradle:1.3.0'
7          //下面两个包是用于上传的插件
8          classpath 'com.jfrog.bintray.gradle:gradle-bintray-plugin:1.0'
9          classpath 'com.github.dcendents:android-maven-gradle-plugin:1.5'
10     }
11 }
```

在 module 的 gradle.properties 文件中定义属性

- 1 PROJ_NAME=bintrayaar
- 2 PROJ_ARTIFACTID=bintrayaar
- 3 PROJ_POM_NAME=Bintray Aar Repository

关闭





在 module 的 build.gradle 中使用插件

```
1 apply plugin: 'com.github.dcendents.android-maven'
2 apply plugin: 'com.jfrog.bintray'
```

为了build.gradle 文件干净，笔者创建了一个名为 bintray.gradle 的文件配置插件信息，请参考[这个文件](#)。

关键点：

```
1 artifacts {
2     archives javadocJar
3     archives sourcesJar
4 }
```

是为了同时生成 javadoc.jar 和 sources.jar 文件

build，上传

```
1 $ ./gradlew -p bintrayrepo/ clean build bintrayUpload --info
```

如果一切顺利，你会在控制台看到多个文件上传成功的标输出

踩坑实录

HTTP/1.1 401 Unauthorized

apikey 或者 user 填写错误

HTTP/1.1 409 Conflict

包的该版本已经存在，需要在 bintray 管理界面上删除该版本后才

想让 sources.jar 或者 javadoc.jar 为空

```
1 task sourcesJar(type: Jar) {
2     classifier = 'sources'
3     from sourceSets.main.java.srcDirs
```

关闭





```
4         exclude '**'
5     }
```

5. 上传 Jar 包

在上传 jar 的时候，使用的插件有些区别

- 1 `apply plugin: 'maven-publish'`
- 2 `apply plugin: 'com.jfrog.bintray'`

在生成符合规定的 pom 文件的时候，要调用 groovy 的 API，具体请参考[这个文件](#)

6. 通过私有仓库的方式引用

至此，刚才上传的两个库，已经可以通过如下方式引用了

```
1 allprojects {
2     repositories {
3         maven {
4             url 'https://dl.bintray.com/freeraces/maven'//这个地址在包的页面的右方
5         }
6     }
7 }
1 compile 'com.as-gradle.demo:bintrayaar:1.0.0'
2 compile 'com.as-gradle.demo:bintrayjar:1.0.0'
```

但是你也发现了，包的用户需要添加一个额外的 maven 仓库发者，那当然不希望用户麻烦的。那就需要把这个私有的库，

7. 推送到 jcenter

在 bintray 的 maven 库的界面，有 `add to jcenter`



关闭



Linked to (0)

[Add to JCenter](#)

This package is not linked to any repository yet.

点击之后，会进入一个消息页面，写或者不写都可以。提交，等待回复即可。

记住,包必须满足如下条件：

包含 sources.jar 和 javadoc.jar 文件

必须是 maven 包

bintray 的消息系统有些 bug，假设你的包提交申请被驳回，你修改之后再提交申请没有人回复你。请不要傻等。直接找页面右侧的 Feedback，这个他们是很快有，

成功了之后，会出现如下的标记：

Linked to (1)

[Stage snapshots on oss.jfrog.org](#)


你可以在 jcenter [服务器](#)上看到你的包了

8. 推送到 mavenCentral

关闭





在包管理页面，可以找到推送到 mavenCentral 功能，

General Readme Release Notes Reviews (0) Statistics Files **Maven Central**

① To have your package artifacts synced to the Maven Central repository, your package must comply with the Maven Central [requirements](#)..

User token key:

User token password:

① We do not save this password.

☒ Close and release repository when done.

Sync

Sync Status

Last Synced: Never

Last Sync Status: N/A

Last Sync Errors: N/A

Note: The responsiveness of the Maven Central gateway at Sonatype fluctuates. This operation may take several minutes depending on the weather.

一个包要从 bintray 推送到 jcenter，有几个前提：

已经推送到了 jcenter[已完成]

每个文件都有对应的签名[已完成]

有 sonatype 账户[未完成]

maven 仓库经过审核[未完成]

点击 Sync 之后，一段时间之后，右边的 Sync Status 会反馈

当然了，现在咱还干不了这个，因为还有两个条件没准备好。件准备。

发布包到 Sonatype MavenCentral 远程仓库

1. 注册 sonatype 账户

进入 [sonatype issue](#) 页面，注册账号。



关闭





2. 创建 issue

登陆之后，顶部有按钮，[Created](#)，下面是关键的条目

Project: [Community Support - Open Source Project Repository Hosting \(OSSRH\)](#)

Issue Type: New Project

Group Id：就是你的包的 groupId

其他的都认真填写。确认之后，大概两个工作日，Issue 会变成 resolved 状态，就可以发布你的包了。有了这两部，其实就可以从 [bintray](#) 上直接反向推到 [mavenCentral](#)，而不需要走下面的步骤了，但是我还是简略介绍一下下面的步骤。如果很感兴趣详情，可以参考 [trinea](#) 的

3. 上传包

也有方便的 gradle 插件帮助我们进行传送，可以参考 [chrisbanes/gradle-mvn-push](#) 插件，上传。

4. 发布包

登陆 [oss.sonatype](#)，登陆，选择左边栏里的 [Staging Repos](#)，sonatype 会做响应的 validation，通过的话，就可以点 Release，先 Drop，并重新做 Staging 发布。

5. 检查包

在 <https://oss.sonatype.org/content/repositories/releases> 可以

6. 为何如此简略

关闭





因为这个过程真的很繁琐，ui 也不友好，在体验了 bintray 的便捷和友好，并发现 bintray 可以反向推送到 mavenCentral 之后，我就再也不想使用 sonatype 了。无奈，贪嗔痴是人类天性。

搭建私服

由于“你懂得”的原因，在国内访问 jcenter，总是偶尔不稳定，经常会出现诸如 `peer not found` 这种错误。为了保证用户的稳定使用库，那就要考虑搭建放在自己服务器上的私有仓库。

Sonatype 和 bintray 都提供了可供自己部署的 maven 库管理软件。Sonatype 提供了免费的 [sonatype/nexus](#)，bintray 提供了免费的 [artifactory-oss](#)。

为了部署简便，笔者使用了 [docker](#) 进行这两个私服搭建。对于 docker 是什么，怎么系列文章的重点。有兴趣可以自行学习。入门文档[在此](#)。

搭建私有 Sonatype 仓库

下载安装 docker 镜像

```
1 $ docker pull sonatype/nexus
```

运行镜像

```
1 $ docker run -d -p 8081:8081 --name nexus sonatype/nexus
```

访问服务器

因为的 docker-machine ip 是：`192.168.99.100`，于是可以通过

`http://192.168.99.100:8081/` 这个 URL 来访问 sonatype 私有仓库

你会发现这个界面跟你在 `https://oss.sonatype.org/` 看到的界面是一样的

默认账户密码是：

关闭





- 1 admin
- 2 admin123

设置仓库

点击左侧 repository，会出现 repository 的列表，把其中的 Releases 的 Configuration->Access Setting-> Deploy Policy 设置成 Allow Redeploy 使得可以重复发布包。

配置和使用插件

我还是使用了 [chrisbanes/gradle-mvn-push](#) 插件，稍微改动了一下字段的值，主要改动是环境配置，如账号密码，repository URL 等，具体请参考这里 [mvn-push](#)。

关键设置

要在 `gradle.properties` 中设置：

- 1 PROJ_NAME=sonatyaar
- 2 PROJ_ARTIFACTID=sonatyaar
- 3 PROJ_POM_NAME=Sonatye Aar Repository
- 4 POM_PACKAGING=aar
- 5 RELEASE_REPOSITORY_URL=http://192.168.99.100:8081/content/repositories/releases
- 6 SNAPSHOT_REPOSITORY_URL=http://192.168.99.100:8081/

查看

上传成功之后，就可以在浏览器的

`http://192.168.99.100:8081/content/repositories/re`

错误

上传的时候，返回400，可能是 Configuration->Access Setti

401，可能是账号密码错误。

关闭





搭建私有 Artifactory 仓库

bintray 其实提供了多个私有部署仓库的版本，分别是：

- 1 Artifactory OSS
- 2 Artifactory Pro
- 3 Artifactory Pro Registry

按名字来看，后两者可能是收费的，这里就只介绍 `Artifactory OSS`，依然是使用 docker 进行部署运行。更详细的使用手册，参考 [Running with Docker](#)。

下载镜像（截止至2016-01-27，最新版本是4.4.1）

- 1 `$ docker pull jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss:4.4.1`

运行镜像

- 1 `$ docker run -d -p 8080:8081 jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss:4.4.1`

在浏览器中访问 `http://192.168.99.100:8080/`，默认账号密码是：

- 1 `admin`
- 2 `password`

笔者写到这，发现这个篇幅已经太长了。现在的读者，其实也考虑将更详细的私服的部署，放在一篇独立的博客中讲解。

kevinho/gradle-maven-complete

为了方便读者使用 gradle 将 aar、jar包推送到 jcenter 和 maven 的 sample 项目，分离出一个独立的 github 项目：[kevinho/gradle-maven-complete](#)
下范例：

localrepo：本地仓库推送





bintrayaar : bintray 的 aar 包推送
bintrayjar : bintray 的 jar 包推送
sonatypearr : mavenCentral 的 aar 包推送

基本上覆盖到了主流的场景了，希望我这个小轮子，也能帮助大家，喜欢记得 star 哦！

总结

这一篇，笔者结合实例讲解了 maven 仓库相关的知识，以及将 maven 包通过 gradle 插件上传本地仓库、bintray jcenter、sonatype mavenCentral，还简要介绍了 sonatype 和 artifactory 私服的 docker 搭建。或许你已经蠢蠢欲动了，那就赶紧打开你的电脑，把你的轮子，用 maven 来吧！下一篇会介绍 gradle 插件的编写以及发布使用，敬请期待！

顶 踩
1 0

关闭

- 上一篇 拥抱 Android Studio 之三：溯源，Groovy 与 Gradle 基础
- 下一篇 android studio 学习笔记

相关文章推荐

- 拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建
- 使用 Android Studio 搭建私有仓库
- 腾讯云容器服务架构实现介绍--董晓杰
- 用 Word2Vec 模型进行词向量表示





- 拥抱 Android Studio 之四：Maven 仓库使用与私有...
- 微博热点事件背后的数据库运维心得--张冬洪
- Android的Nexus搭建Maven私有仓库与使用
- JDK9新特性--Array
- Android业务组件化之Gradle和Sonatype Nexus搭建...
- Kubernetes容器云平台实践--李志伟
- Android业务组件化之Gradle和Sonatype Nexus搭建...
- Java之优雅编程之道
- 使用Archiva搭建Maven私有仓库
- 使用Gradle和Sonatype Nexus 搭建私有maven仓库
- 使用Nexus搭建Maven私有仓库
- 【Maven】Nexus3搭建Maven私有仓库及使用



查看评论



donnykeon

1楼 2016-03-28 12:01

您好？请问您弄成功了吗？

发表评论

用户名：

评论内容：





提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知网络技术有限公司
京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved



关闭

