

拥抱 Android Studio 之五：Gradle 插件开发

Bugtags 2016-03-28 原文



实践出真知

笔者有位朋友，每次新学一门语言，都会用来写一个贪吃蛇游戏，以此来检验自己学习的成果。笔者也有类似体会。所谓纸上得来终觉浅，绝知此事要躬行。这一章，笔者将以开发和发布一个 Gradle 插件作为目标，加深学习成果。

[官方文档](#)给出了比较详细的实现步骤，本文的脉络会跟官方文档差不了太多，额外增补实际例子和一些实践经验。文中的代码已经托管到了 [github 项目](#) 中。

需求

默认的 Android 打包插件会把 apk 命名成 module-productFlavor-buildType.apk，例如 app-official-debug.apk，并且会把包文件发布到固定的位置：
module/build/outputs/apk 有的时候，这个命名风格并不是你所要的，你也想讲 apk 输出到别的目录。咱们通过 gradle 插件来实现自定义。这个插件的需求

是：

- 输入一个名为 nameMap 的 Closure，用来修改 apk 名字
- 输入一个名为 destDir 的 String，用于输出位置

原理简述

插件之于 Gradle

根据官方文档定义，插件打包了可重用的构建逻辑，可以适用于不同的项目和构建过程。

Gradle 提供了很多官方插件，用于支持 Java、Groovy 等工程的构建和打包。同时也提供了自定义插件的机制，让每个人都可以通过插件来实现特定的构建逻辑，并可以把这些逻辑打包起来，分享给其他人。

插件的源码可以使用 Groovy、Scala、Java 三种语言，笔者不会 Scala，所以平时只是使用 Groovy 和 Java。前者用于实现与 Gradle 构建生命周期（如 task 的依赖）有关的逻辑，后者用于核心逻辑，表现为 Groovy 调用 Java 的代码。

另外，还有很多项目使用 Eclipse 或者 Maven 进行开发构建，用 Java 实现核心业务代码，将有利于实现快速迁移。

插件打包方式

Gradle 的插件有三种打包方式，主要是按照复杂程度和可见性来划分：

Build script

把插件写在 build.gradle 文件中，一般用于简单的逻辑，只在该 build.gradle 文件中可见，笔者常用来做原型调试，本文将简要介绍此类。

buildSrc 项目

将插件源代码放在 `rootProjectDir/buildSrc/src/main/groovy` 中，只对该项目中可见，适用于逻辑较为复杂，但又不需要外部可见的插件，本文不介绍，有兴趣可以参考[此处](#)。

独立项目

一个独立的 Groovy 和 Java 项目，可以把这个项目打包成 Jar 文件包，一个 Jar 文件包还可以包含多个插件入口，将文件包发布到托管平台上，供其他人使用。本文将着重介绍此类。

Build script 插件

首先来直接在 `build.gradle` 中写一个 plugin：

```
1.  class ApkDistPlugin implements Plugin<Project> {
2.
3.      @Override
4.      void apply(Project project) {
5.          project.task('apkdist') << {
6.              println 'hello, world!'
7.          }
8.      }
9.  }
10.
11.  apply plugin: ApkDistPlugin
```

命令行运行

```
1.  $ ./gradlew -p app/ apkdist
2.  :app:apkdist
3.  hello, world!
```

这个插件创建了一个名为 apkdist 的 task，并在 task 中打印。

插件是一个类，继承自 org.gradle.api.Plugin 接口，重载 void apply(Project project) 方法，这个方法将会传入使用这个插件的 project 的实例，这是一个重要的 context。

接受外部参数

通常情况下，插件使用方需要传入一些配置参数，如 bugtags 的 SDK 的插件需要接受两个参数：

```
1. bugtags {  
2.     appKey "APP_KEY" //这里是你的 appKey  
3.     appSecret "APP_SECRET" //这里是你的 appSecret，管理员在设置页可以查看  
4. }
```

同样，ApkDistPlugin 这个 plugin 也希望接受两个参数：

```
1. apkdistconf {  
2.     nameMap { name ->  
3.         println 'hello,' + name  
4.         return name  
5.     }  
6.     destDir 'your-distribution-dir'  
7. }
```

参数的内容后面继续完善。那这两个参数怎么传到插件内呢？

org.gradle.api.Project 有一个 ExtensionContainer getExtensions() 方法，可以用来实现这个传递。

声明参数类

声明一个 Groovy 类，有两个默认值为 null 的成员变量：

```
1. class ApkDistExtension {  
2.     Closure nameMap = null;  
3.     String destDir = null;  
4. }
```

接受参数

```
1. project.extensions.create('apkdistconf', ApkDistExtension);
```

要注意，create 方法的第一个参数就是你在 build.gradle 文件中的进行参数配置的 dsl 的名字，必须一致；第二个参数，就是参数类的名字。

获取和使用参数

在 create 了 extension 之后，如果传入了参数，则会携带在 project 实例中，

```
1. def closure = project['apkdistconf'].nameMap;  
2. closure('wow!');  
3.  
4. println project['apkdistconf'].destDir
```

进化版本一：参数

```
1. class ApkDistExtension {  
2.     Closure nameMap = null;  
3.     String destDir = null;  
4. }  
5.  
6. class ApkDistPlugin implements Plugin<Project> {  
7.
```

```
8.      @Override
9.      void apply(Project project) {
10.
11.          project.extensions.create('apkdistconf', ApkDistExtension);
12.
13.          project.task('apkdist') << {
14.              println 'hello, world!'
15.
16.              def closure = project['apkdistconf'].nameMap;
17.              closure('wow!');
18.
19.              println project['apkdistconf'].destDir
20.          }
21.      }
22.  }
23.
24.  apply plugin: ApkDistPlugin
25.
26.  apkdistconf {
27.      nameMap { name ->
28.          println 'hello, ' + name
29.          return name
30.      }
31.      destDir 'your-distribution-directory'
32.  }
```

运行结果：

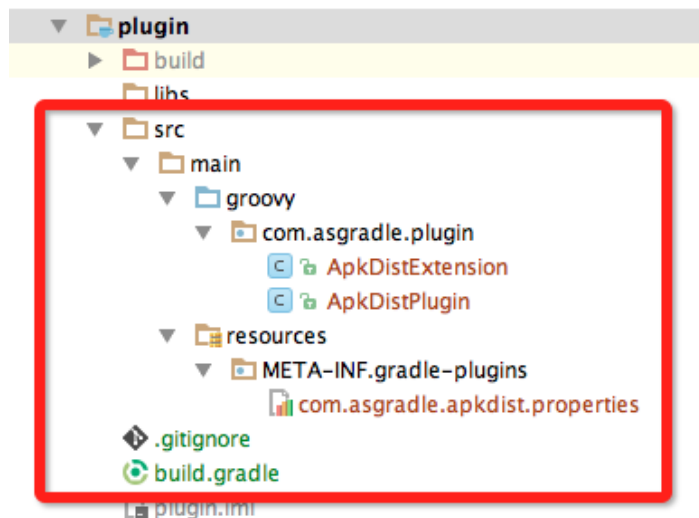
```
1.  $ ./gradlew -p app/ apkdist
2.  :app:apkdist
3.  hello, world!
4.  hello, wow!
5.  your-distribution-directory
```

独立项目插件

代码写到现在，已经不适合再放在一个 build.gradle 文件里面了，那也不是我们的目的。建立一个独立项目，把代码搬到对应的地方。

理论上，IntelliJ IDEA 开发插件要比 Android Studio 要方便一点点，因为有对应 Groovy module 的模板。但其实如果我们了解 IDEA 的项目文件结构，就不会受到这个局限，无非就是一个 build.gradle 构建文件加 src 源码文件夹。

最终项目的文件夹结构是这样：



下面我们来一步步讲解。

创建项目

在 Android Studio 中新建 Java Library module “plugin”。

修改 build.gradle 文件

添加 Groovy 插件和对应的两个依赖。

```
1. //removed java plugin
2. apply plugin: 'groovy'
3.
4. dependencies {
5.     compile gradleApi()//gradle sdk
6.     compile localGroovy()//groovy sdk
7.     compile fileTree(dir: 'libs', include: ['*.jar'])
8. }
```

修改项目文件夹

src/main 项目文件下：

- 移除 java 文件夹，因为在这个项目中用不到 java 代码
- 添加 groovy 文件夹，主要的代码文件放在这里
- 添加 resources 文件夹，存放用于标识 gradle 插件的 meta-data

建立对应文件

```
1. .
2. ├── build.gradle
3. ├── libs
4. ├── plugin.iml
5. └── src
6.     ├── main
7.     │   ├── groovy
8.     │   │   ├── com
9.     │   │   │   ├── asgradle
10.    │   │   │   │   ├── plugin
11.    │   │   │   │   │   ├── ApkDistExtension.groovy
12.    │   │   │   │   │   └── ApkDistPlugin.groovy
13.    │   └── resources
14.    │       └── META-INF
```



```
15. └─ gradle-plugins
16.    └─ com.asgradle.apkdist.properties
```

注意：

- groovy 文件夹中的类，一定要修改成 .groovy 后缀，IDE 才会正常识别。
- resources/META-INF/gradle-plugins 这个文件夹结构是强制要求的，否则不能识别成插件。

com.asgradle.apkdist.properties 文件

如果写过 Java 的同学会知道，这是一个 Java 的 properties 文件，是 key=value 的格式。这个文件内容如下：

```
1. implementation-class=com.asgradle.plugin.ApkDistPlugin
```

按其语义推断，是指定这个插件的入口类。

- 英文敏感的同学可能会问了，为什么这个文件的承载文件夹是叫做 gradle-plugins，使用复数？没错，这里可以指定多个 properties 文件，定义多个插件，扩展性一流，可以参考 linkedin 的插件的组织方式。
- 使用这个插件的时候，将会是这样：

```
1. apply plugin: 'com.asgradle.apkdist'
```

因此，com.asgradle.apkdist 这个字符串在这里，又称为这个插件的 id，不允许跟别的插件重复，取你拥有的域名的反向就不会错。

将 plugin module 传到本地 maven 仓库

参考上一篇：[拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建](#)，和对应的 [demo 项目](#)，将包传到本地仓库中进行测试。

添加 gradle.properties

```
1. PROJ_NAME=gradleplugin
2. PROJ_ARTIFACTID=gradleplugin
3. PROJ_POM_NAME=Local Repository
4.
5. LOCAL_REPO_URL=file:///Users/changbinhe/Documents/Android/repo/
6.
7. PROJ_GROUP=com.as-gradle.demo
8.
9. PROJ_VERSION=1.0.0
10. PROJ_VERSION_CODE=1
11.
12. PROJ_WEBSITEURL=http://kvh.io
13. PROJ_ISSUETRACKERURL=https://github.com/kevinho/Embrace-Android-Studio-Demo/issues
14. PROJ_VCSURL=https://github.com/kevinho/Embrace-Android-Studio-Demo.git
15. PROJ_DESCRIPTION=demo apps for embracing android studio
16.
17. PROJ_LICENCE_NAME=The Apache Software License, Version 2.0
18. PROJ_LICENCE_URL=http://www.apache.org/licenses/LICENSE-2.0.txt
19. PROJ_LICENCE_DEST=repo
20.
21. DEVELOPER_ID=your-dev-id
22. DEVELOPER_NAME=your-dev-name
23. DEVELOPER_EMAIL=your-email@your-mailbox.com
```

在 build.gradle 添加上传功能

```
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.     }  
10. }
```

上传可以通过运行：

```
1. $ ./gradlew -p plugin/ clean build uploadArchives
```

在 app module 中使用插件

在项目的 buildscript 添加插件作为 classpath

```
1. buildscript {  
2.     repositories {  
3.         maven{  
4.             url 'file:///Users/your-user-name/Documents/Android/repo/'  
5.         }  
6.         jcenter()  
7.     }  
8.     dependencies {  
9.         classpath 'com.android.tools.build:gradle:2.1.0-alpha3'  
10.        classpath 'com.as-gradle.demo:gradleplugin:1.0.0'  
11.     }  
12. }
```

在 app module 中使用插件：

```
1. apply plugin: 'com.asgradle.apkdist'
```

命令行运行：

```
1. $ ./gradlew -p app apkdist
```

```
2. :app:apkdist
3. hello, world!
4. hello, wow!
5. your-distribution-directory
```

可能会遇到问题

```
1. Error:(46, 0) Cause: com/asgradle/plugin/ApkDistPlugin : Unsupported major.minor version 52.0
2. <a href="openFile:/Users/your-user-name/Documents/git/opensource/embrace-android-studio-demo/s5-GradlePlugin/app/build.gradle">Open File</a>
```

应该是本机的 JDK 版本是1.8，默认将 plugin module 的 groovy 源码编译成了1.8版本的 class 文件，放在 Android 项目中，无法兼容。需要对 plugin module 的 build.gradle 文件添加两个参数：

```
1. sourceCompatibility = 1.6
2. targetCompatibility = 1.6
```

真正的实现插件需求

读者可能会观察到，到目前为止，插件只是跑通了流程，并没有实现本文提出的两个需求，

那接下来就具体实现一下。

```
1. class ApkDistPlugin implements Plugin<Project> {
2.
3.     @Override
4.     void apply(Project project) {
5.
6.         project.extensions.create('apkdistconf', ApkDistExtension);
7.
8.         project.afterEvaluate {
9.
```

```
10. //只可以在 android application 或者 android lib 项目中使用
11. if (!project.android) {
12.     throw new IllegalStateException('Must apply \'com.android.application\' or
    \'com.android.library\' first!')
13. }
14.
15. //配置不能为空
16. if (project.apkdistconf.nameMap == null || project.apkdistconf.destDir == null) {
17.     project.logger.info('Apkdist conf should be set!')
18.     return
19. }
20.
21. Closure nameMap = project['apkdistconf'].nameMap
22. String destDir = project['apkdistconf'].destDir
23.
24. //枚举每一个 build variant
25. project.android.applicationVariants.all { variant ->
26.     variant.outputs.each { output ->
27.         File file = output.outputFile
28.         output.outputFile = new File(destDir, nameMap(file.getName()))
29.     }
30. }
31. }
32. }
33. }
```

必须指出，本文插件实现的需求，其实可以直接在 app module 的 build.gradle 中写脚本就可以实现。这里做成插件，只是为了做示范。

上传到 bintray 的过程，就不再赘述了，可以参考[拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建](#)。

后记

至此，这系列开篇的时候挖下的坑，终于填完了。很多人借助这系列的讲解，真正理解了 Android Studio 和它背后的 Gradle、Groovy，笔者十分高兴。笔者也得到了很多读者的鼓励和支持，心中十分感激。

写博客真的是一个很讲究执行力和耐力的事情，但既然挖下了坑，就得填上，对吧？

这半年来，个人在 Android 和 Java 平台上也做了更多的事情，也有了更多的体会。

AS 系列，打算扩充几个主题：

- Proguard 混淆
- Java & Android Testing
- Maven 私有仓库深入
- 持续集成
-待发掘

记得有人说，只懂 Android 不懂 Java，是很可怕的。在这半年以来，笔者在工作中使用 Java 实现了一些后端服务，也认真学习了 JVM 字节码相关的知识并把它使用到了工作中。在这个过程中，真的很为 Java 平台的活力、丰富的库资源、几乎无止境的可能性所折服。接下来，会写一些跟有关的学习体会，例如：

- Java 多线程与锁
- JVM 部分原理
- 字节码操作
- Java 8部分特性
-待学习

随着笔者工作的进展，我也有机会学习使用了别的语言，例如 Node.js，并实现了一些后端服务。这个语言的活力很强，一些比 Java 现代的地方，很吸引人。有精力会写一写。

因为业务所需，笔者所经历的系统，正处于像面向服务的演化过程中，我们期望建立统一的通讯平台和规范，抽象系统的资源，拆分业务，容器化。这是一个很有趣的过程，也是对我们的挑战。笔者也希望有机会与读者分享。

一不小心又挖下了好多明坑和无数暗坑，只是为了激励自己不断往前。在探索事物本质的旅途中，必然十分艰险，又十分有趣，沿途一定风光绚丽，让我们共勉。

参考文献

[官方文档](#)

系列导读

本文是笔者《拥抱 Android Studio》系列第四篇，其他篇请点击：

[拥抱 Android Studio 之一：从 ADT 到 Android Studio](#)

[拥抱 Android Studio 之二：Android Studio 与 Gradle 深入](#)

[拥抱 Android Studio 之三：溯源，Groovy 与 Gradle 基础](#)

[拥抱 Android Studio 之四：Maven 公共仓库使用与私有仓库搭建](#)

[拥抱 Android Studio 之五：Gradle 插件使用与开发](#)

有问题？在文章下留言或者加 qq 群：453503476，希望能帮到你。

番外

笔者 [kvh](#) 在开发和运营 [bugtags.com](#)，这是一款移动时代首选的 bug 管理系统，能够极大的提升 app 开发者的测试效率，欢迎使用、转发推荐。

笔者目前关注点在于移动 SDK 研发，后端服务设计和实现。

我们团队长期求 PHP 后端研发，有兴趣请加下面公众号勾搭：



拥抱 Android Studio 之五：Gradle 插件开发的更多相关文章

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

使用.创造和分享 笔者曾经不思量力的思考过『是什么推动了互联网技术的快速发展?』这种伟大的命题.结论是,除了摩尔定律之外,技术经验的快速积累和广泛分享,也是重要的原因. 有人戏称,『写 Java,首先 ...

2. Android studio下gradle Robolectric单元测试配置

android studio下gradle Robolectric单元测试配置 1.Robolectric Robolectric是一个基于junit之上的单元测试框架.它并不依赖于Android提供 ...

3. android studio 使用gradle 导出jar包，并打包assets目录

警告:本文年久失修. 随着android studio的升级,gradle的升级,严格按照本文的代码去做可能不会成功,希望依然可以作为解决问题的思路. 最近项目在做一个sdk,供别的开发者使用,所以 ...

4. Android Studio：Android Studio 与 Gradle 深入【二】

转载:<http://www.apkbus.com/forum.php?mod=viewthread&tid=255063&extra=page%3D2%26filter%3Dautho> ...

5. [转]--android studio 使用gradle 导出jar包，并打包assets目录

转自: <http://www.cnblogs.com/wuya/p/android-studio-gradle-export-jar-assets.html> 最近项目在做一个sdk,供别的开发者使 ...

6. Android 项目利用 Android Studio 和 Gradle 打包多版本APK

在项目开发过程中,经常会有需要打包不同版本的 APK 的需求. 比如 debug版,release版,dev版等等. 有时候不同的版本中使用到的不同的服务端api 域名也不相同. 比如 debug_ap ...

7. 利用 Android Studio 和 Gradle 打包多版本APK

在项目开发过程中,经常会有需要打包不同版本的 APK 的需求. 比如 debug版,release版,dev版等等. 有时候不同的版本中使用到的不同的服务端api 域名也不相同. 比如 debug_ap ...

8. 快速掌握 Android Studio 中 Gradle 的使用方法

快速掌握 Android Studio 中 Gradle 的使用方法 Gradle是可以用于Android开发的新一代的 Build System, 也是 Android Studio默认的build ...

9. Android studio 使用Gradle发布Android开源项目到JCenter 总结

1.注册账号 先到<https://bintray.com>注册一个账号. 这个网站支持 github 账户直接登录的 2.获取 bintray.user 和 bintray.apikey ...

随机推荐

1. An entity object cannot be referenced by multiple instances of IEntityChangeTracker 的解决方案

使用EF对建立了关系的表新增记录时出现: An entity object cannot be referenced by multiple instances of IEntityChangeTra ...

2. 自己开发一个 vsts agent 的 task

vsts 中支持自定义Build/Release的过程Task 目标:做一个可以读取 Xamarin.Android 所生成的 APK 的基本信息的 task ,包括 package(包名) / a ...

3. day4总结

函数是什么? 函数一词来源于数学,但编程中的「函数」概念,与数学中的函数是有很大的不同的,具体区别,我们后面会讲,编程中的函数在英文中也有很多不同的叫法.在BASIC中叫做subroutine(子过程或 ...

4. 简单的dp

有趣的数:(动态规划,状态转移) #include<stdio.h> []; int main() { int n,i; ; i<; i++) dp[i][]=; while(~s ...

5. 【代码笔记】iOS-判断有无网络

一,工程图. 二,代码. RootViewController.h #import <UIKit/UIKit.h> @interface RootViewController : UIVI ...

6. Java多线程与并发库高级应用-线程池

线程池 线程池的思想 线程池的概念与Executors类的应用 > 创建固定大小的线程池 > 创建缓存线程池 > 创建单一线程池(如何实现线程死掉后重新启动?) 关闭线程池 > ...

7. Spring MVC学习笔记——Welcome

参考: <http://blog.csdn.net/hehexiaoyou/article/details/23747617> <http://www.codingyun.com/article/47.ht> ...

8. leetcode 100. Same Tree

Given two binary trees, write a function to check if they are equal or not. Two binary trees are con ...

9. C# 窗体 (登录界面)

首先拖动一个 lable(写用户名) 后面 跟一个Textbox 再lable(写密码) 后面 跟一个Textbox(需设置一下属性—行为—useSystemPasswordChar(默认输入的密 ...

10. 读书笔记——Windows环境下32位汇编语言程序设计 (13) 关于EXCEPTION_DEBUG_INFO结构体

在动手自己尝试编写书上第13章的例子Patch3时,遇到了一个结构体EXCEPTION_DEBUG_INFO. 这个结构体在MASM的windows.inc中的定义和MSDN中的定义不一样. (我使用 ...

