昵称：Bugtags
园龄：2年1个月
粉丝：2
关注：0
+加关注

< 2017年12月 >

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

拥抱 Android Studio 之五：Gradle 插件开发

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

app 测试 bug管理 协同办公(1)
app测试 bug 管理(1)
Bug(1)
Bugtags(1)
Bug管理系统(1)
创业(1)
一周年(1)

随笔档案

2017年2月 (2)
2017年1月 (2)
2016年12月 (2)
2016年10月 (2)
2016年9月 (2)
2016年8月 (2)
2016年7月 (3)
2016年6月 (6)
2016年5月 (1)
2016年4月 (8)
2016年3月 (14)
2016年2月 (10)
2016年1月 (17)
2015年12月 (13)
2015年11月 (2)

阅读排行榜

1. NDK SO 库开发与使用中的 ABI 构架选择(1975)
2. Bugtags 创业一年总结(918)
3. Android Gradle 技巧之二： 最爱命令行(741)
4. 拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建(622)

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。

5. Android Gradle 技巧之一： Build Variant 相关(243)

推荐排行榜

1. 拥抱 Android Studio 之四：Maven 仓库使用与私有仓库搭建(2)
2. 69 个经典 Spring 面试题和答案(1)

Build script 插件

首先来直接在 build.gradle 中写一个 plugin：

```
class ApkDistPlugin implements Plugin<Project> {

    @Override
    void apply(Project project) {
        project.task('apkdist') << {
            println 'hello, world!'
        }
    }
}

apply plugin: ApkDistPlugin
```

命令行运行

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

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

同样，ApkDistPlugin 这个 plugin 也希望接受两个参数：

```
apkdistconf {
    nameMap { name ->
        println 'hello,' + name
        return name
    }
    destDir 'your-distribution-dir'
}
```

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：

```
class ApkDistExtension {
    Closure nameMap = null;
    String destDir = null;
}
```

接受参数

```
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，

```
def closure = project['apkdistconf'].nameMap;
closure('wow!');

println project['apkdistconf'].destDir
```

进化版本一：参数

```
class ApkDistExtension {
    Closure nameMap = null;
    String destDir = null;
}

class ApkDistPlugin implements Plugin<Project> {

    @Override
    void apply(Project project) {

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

        project.task('apkdist') << {
            println 'hello, world!'

            def closure = project['apkdistconf'].nameMap;
            closure('wow!');

            println project['apkdistconf'].destDir
        }
    }
}

apply plugin: ApkDistPlugin

apkdistconf {
    nameMap { name ->
        println 'hello, ' + name
        return name
    }
    destDir 'your-distribution-directory'
}
```

运行结果：

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

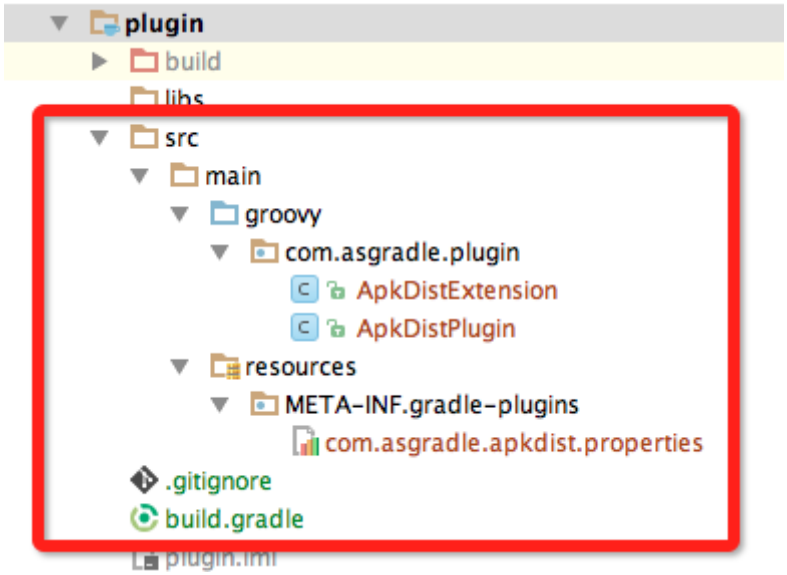
```
hello, world!
hello, wow!
your-distribution-directory
```

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。

```
//removed java plugin
apply plugin: 'groovy'

dependencies {
    compile gradleApi()//gradle sdk
    compile localGroovy()//groovy sdk
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

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

```
.
├── build.gradle
├── libs
├── plugin.iml
└── src
    └── main
        ├── groovy
        │   └── com
        │       └── asgradle
        │           └── plugin
        │               ├── ApkDistExtension.groovy
        │               └── ApkDistPlugin.groovy
```

```
        └── resources
            └── META-INF
                └── gradle-plugins
                    └── 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：

```
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：

  ```
  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

```
PROJ_NAME=gradleplugin
PROJ_ARTIFACTID=gradleplugin
PROJ_POM_NAME=Local Repository
```

```
LOCAL_REPO_URL=file:///Users/changbinhe/Documents/Android/repo/

PROJ_GROUP=com.as-gradle.demo

PROJ_VERSION=1.0.0
PROJ_VERSION_CODE=1

PROJ_WEBSITEURL=http://kvh.io
PROJ_ISSUETRACKERURL=https://github.com/kevinho/Embrace-Android-Studio-Demo/issues
PROJ_VCSURL=https://github.com/kevinho/Embrace-Android-Studio-Demo.git
PROJ_DESCRIPTION=demo apps for embracing android studio

PROJ_LICENCE_NAME=The Apache Software License, Version 2.0
PROJ_LICENCE_URL=http://www.apache.org/licenses/LICENSE-2.0.txt
PROJ_LICENCE_DEST=repo

DEVELOPER_ID=your-dev-id
DEVELOPER_NAME=your-dev-name
DEVELOPER_EMAIL=your-email@your-mailbox.com
```

在 build.gradle 添加上传功能

```
apply plugin: 'maven'

uploadArchives {
    repositories.mavenDeployer {
        repository(url: LOCAL_REPO_URL)
        pom.groupId = PROJ_GROUP
        pom.artifactId = PROJ_ARTIFACTID
        pom.version = PROJ_VERSION
    }
}
```

上传可以通过运行：

```
$ ./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

```
buildscript {
    repositories {
        maven{
            url 'file:///Users/your-user-name/Documents/Android/repo/'
        }
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0-alpha3'
        classpath 'com.as-gradle.demo:gradleplugin:1.0.0'
    }
}
```

在 app module 中使用插件：

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

命令行运行：

```
$ ./gradlew -p app apkdist
:app:apkdist
hello, world!
hello, wow!
your-distribution-directory
```

可能会遇到问题

```
Error:(46, 0) Cause: com/asgradle/plugin/ApkDistPlugin : Unsupported major.minor version 52.0
<a href="openFile:/Users/your-user-name/Documents/git/opensource/embrace-android-studio-demo/s5-Gra
```

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：

```
sourceCompatibility = 1.6
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。

```
class ApkDistPlugin implements Plugin<Project> {

    @Override
    void apply(Project project) {

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

        project.afterEvaluate {

            //只可以在 android application 或者 android lib 项目中使用
            if (!project.android) {
                throw new IllegalStateException('Must apply \'com.android.application\' or \'com.ar
            }

            //配置不能为空
            if (project.apkdistconf.nameMap == null || project.apkdistconf.destDir == null) {
                project.logger.info('Apkdist conf should be set!')
                return
            }

            Closure nameMap = project['apkdistconf'].nameMap
            String destDir = project['apkdistconf'].destDir

            //枚举每一个 build variant
            project.android.applicationVariants.all { variant ->
```

```
                variant.outputs.each { output ->
                    File file = output.outputFile
                    output.outputFile = new File(destDir, nameMap(file.getName()))
                }
            }
        }
    }
}
```

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，正在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：

« 上一篇：和免费比起来，收费的才是捷径
» 下一篇：打造高性能Java应用需掌握的5大知识

posted on 2016-03-28 19:11 Bugtags 阅读(151) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

最新知识库文章:

· 步入云计算
· 以操作系统的角度述说线程与进程
· 软件测试转型之路
· 门内门外看招聘
· 大道至简，职场上做人做事做管理
» 更多知识库文章...