由于国内Android市场众多渠道，为了统计每个渠道的下载及其它数据统计，就需要我们针对每个渠道单独打包，如果让你打几十个市场的包岂不烦死了，不过有了Gradle，这再也不是事了。

友盟多渠道打包

废话不多说，以友盟统计为例，在AndroidManifest.xml里面会有这么一段：

<meta-data

android:name="UMENG\_CHANNEL"

android:value="Channel\_ID" />

里面的Channel\_ID就是渠道标示。我们的目标就是在编译的时候这个值能够自动变化。

•第一步 在AndroidManifest.xml里配置PlaceHolder

<meta-data

android:name="UMENG\_CHANNEL"

android:value="${UMENG\_CHANNEL\_VALUE}" />

•第二步 在build.gradle设置productFlavors

android {

productFlavors {

xiaomi {

manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: "xiaomi"]

}

\_360 {

manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: "\_360"]

}

baidu {

manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: "baidu"]

}

wandoujia {

manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: "wandoujia"]

}

}

}

或者批量修改

android {

productFlavors {

xiaomi {}

\_360 {}

baidu {}

wandoujia {}

}

productFlavors.all {

flavor -> flavor.manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: name]

}

}

很简单清晰有没有？直接执行 ./gradlew assembleRelease ， 然后就可以静静的喝杯咖啡等待打包完成吧。

assemble结合Build Variants来创建task

上一篇博客介绍了 assemble 这个命令，会结合 Build Type 创建自己的task，如:

•./gradlew assembleDebug

•./gradlew assembleRelease

除此之外 assemble 还能和 Product Flavor 结合创建新的任务，其实 assemble 是和 Build Variants 一起结合使用的，而Build Variants = Build Type + Product Flavor ， 举个例子大家就明白了：

如果我们想打包wandoujia渠道的release版本，执行如下命令就好了：

•./gradlew assembleWandoujiaRelease

如果我们只打wandoujia渠道版本，则：

•./gradlew assembleWandoujia

此命令会生成wandoujia渠道的Release和Debug版本

同理我想打全部Release版本：

•./gradlew assembleRelease

这条命令会把Product Flavor下的所有渠道的Release版本都打出来。

总之，assemble 命令创建task有如下用法：

•\*\*assemble\*\*： 允许直接构建一个Variant版本，例如assembleFlavor1Debug。

•\*\*assemble\*\*： 允许构建指定Build Type的所有APK，例如assembleDebug将会构建Flavor1Debug和Flavor2Debug两个Variant版本。

•\*\*assemble\*\*： 允许构建指定flavor的所有APK，例如assembleFlavor1将会构建Flavor1Debug和Flavor1Release两个Variant版本。

完整的gradle脚本

最后福利大放送，来一份我在项目中使用的完整的gradle文件配置：

apply plugin: 'com.android.application'

def releaseTime() {

return new Date().format("yyyy-MM-dd", TimeZone.getTimeZone("UTC"))

}

android {

compileSdkVersion 21

buildToolsVersion '21.1.2'

defaultConfig {

applicationId "com.boohee.\*"

minSdkVersion 14

targetSdkVersion 21

versionCode 1

versionName "1.0"

// dex突破65535的限制

multiDexEnabled true

// 默认是umeng的渠道

manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: "umeng"]

}

lintOptions {

abortOnError false

}

signingConfigs {

debug {

// No debug config

}

release {

storeFile file("../yourapp.keystore")

storePassword "your password"

keyAlias "your alias"

keyPassword "your password"

}

}

buildTypes {

debug {

// 显示Log

buildConfigField "boolean", "LOG\_DEBUG", "true"

versionNameSuffix "-debug"

minifyEnabled false

zipAlignEnabled false

shrinkResources false

signingConfig signingConfigs.debug

}

release {

// 不显示Log

buildConfigField "boolean", "LOG\_DEBUG", "false"

minifyEnabled true

zipAlignEnabled true

// 移除无用的resource文件

shrinkResources true

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

signingConfig signingConfigs.release

applicationVariants.all { variant ->

variant.outputs.each { output ->

def outputFile = output.outputFile

if (outputFile != null && outputFile.name.endsWith('.apk')) {

// 输出apk名称为boohee\_v1.0\_2015-01-15\_wandoujia.apk

def fileName = "boohee\_v${defaultConfig.versionName}\_${releaseTime()}\_${variant.productFlavors[0].name}.apk"

output.outputFile = new File(outputFile.parent, fileName)

}

}

}

}

}

// 友盟多渠道打包

productFlavors {

wandoujia {}

\_360 {}

baidu {}

xiaomi {}

tencent {}

taobao {}

...

}

productFlavors.all { flavor ->

flavor.manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: name]

}

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

compile 'com.android.support:support-v4:21.0.3'

compile 'com.jakewharton:butterknife:6.0.0'

...

}