<https://www.jb51.net/article/256293.htm>

<https://blog.csdn.net/Jason_HD/article/details/128789601>

<https://blog.csdn.net/xinpings/article/details/58688361?spm=1001.2014.3001.5506>

目前看这三个就够了

安卓配置多渠道包error：All flavors must now belong to a named flavor dimension

需要配置 flavorDimensions "version"

productFlavors {

flavorDimensions "minSdkVersion"//这行放外面好像也行，别的帖子看见过，没测试

// Define separate dev and prod product flavors.

juliang {

minSdkVersion 21

}

common {

minSdkVersion 21

}

}

android多渠道打包使用不同的资源、依赖、代码

android多渠道打包使用不同的资源、依赖、代码1android多渠道打包使用不同的资源、依赖、代码2android多渠道打包使用不同的资源、依赖、代码3android多渠道打包使用不同的资源、依赖、代码4

需求是这样来的，在发布我的app时，在国内的其他市场发布时都能正常通过，但到了百度市场时被拒绝了，给出的理由是个人合作者的app不能包含有支付功能，app里包含了支付的依赖。我就郁闷了，事实上我的app是没有支付功能的，只有分享功能，分享到微信及QQ的功能，所包含的依赖也是为实现此功能的。顺便也吐槽一下百度。。

吐槽归吐槽，app还是得发布的，那只能打包一个不包含分享功能的app了。

这就要用到多渠道打包了，不想用新建多个module的方式来做，就使用gradle配置来实现，实施的时候走了不少的弯路和花了不少时间，在此做记录及分享给有需要的人。

遇到的主要问题是：选择不同的渠道时需要加载不同的依赖并使用不同的代码，这二个地方都有一些坑。

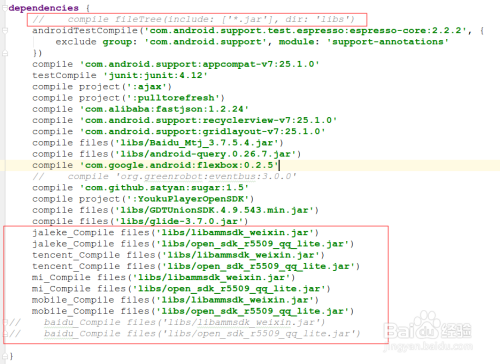
工具/原料

android studio 2.3

gradle

方法/步骤

建立不同的渠道，主要通过设置productFlavors来实现的，如图设置了tencent\_、baudu\_等几个渠道，记住已经设置好的渠道名，下面的步骤需要用到。



android多渠道打包使用不同的资源、依赖、代码

编辑build.gradle的依赖（dependencies），这一步要实现不同的渠道加载不同的依赖。

这里有个需要注意的地方，一般都有这么一句：

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

这句的作用是自动加载libs里的所有的.jar文件，这一句一定要去掉，否则实现不了不同渠道加载不同的jar的目的。去掉这句之后其他的jar只能通过compile 'com.github.satyan:sugar:1.5'的方式加载。

不同渠道加载不同的依赖就要在compile上做文章了，格式是：渠道名+Compile，这样就是渠道独有的，如tencent\_Compile就是tencent\_渠道独有的加载。

如图中的这二句：

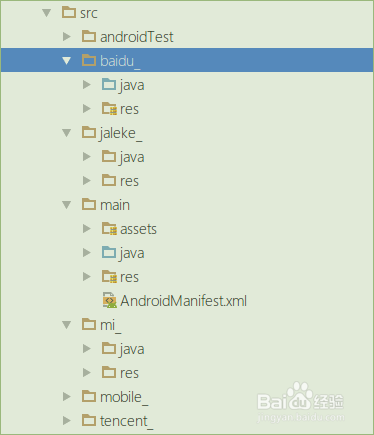
// baidu\_Compile files('libs/libammsdk\_weixin.jar')

// baidu\_Compile files('libs/open\_sdk\_r5509\_qq\_lite.jar')

这二句被注释了，所以baidu\_渠道不会加载这二个jar了。

android多渠道打包使用不同的资源、依赖、代码

在项目（module）的src新建已渠道名称命名的文件夹，名字一定要跟渠道名一样，级别跟main文件夹是一样的，里面独有文件独有资源的文件结构要跟main一致。（不要在AS里创建，会自动给加个main文件夹，导致文件找不到，直接在文件系统中创建就行，创建好如果不刷新，就多等一会，早晚会刷新的）



这里有二种文件，一种是需要用到的资源文件，如各种xml文件或图片等，一种是java代码。这里我遇到了一个坑。一开始我以为渠道文件夹里的文件在编译时会自动替换掉main文件夹里的相应文件，xml文件和图片等资源确实是这样的，没有问题；但到了java文件时，文件就出现了，如图中ShareActivity.java文件，在各个渠道文件夹里有这个文件，在main里也有这个文件，但在编译时就会提示类重复，编译不通过。哈哈。。说好的自动替换呢，怎么还编译不了呢，郁闷了。后来经过多次实验得出结论，只要把main里的java文件删除就能正常编译了。这是遇到的最大的坑，花了不少时间。后来才得知，各个渠道文件夹和main文件夹之间的关系不是单纯的替换，而是优先使用渠道的东西，编译时相当于是合并，xml等文件相当于直接替换，但java却不是这样的。我只能只要理解了，哈哈

android多渠道打包使用不同的资源、依赖、代码

**Android studio gradle中分渠道加载res、libraries及Class**

前段时间遇到这样的需求，在App里使用了地图功能，对应海外版使用Android内置google的地图服务。而对于国内版（大家都知道的），我们使用了高德地图。而在导入高德地图的时候需要加入对应的sdk及.so文件，这样最后打包的大小会增加2M左右。但海外版并不需要这部分资源，故需要根据不同的渠道加载不同的jar、res、及不同的类，有助于保持代码的干净，控制包的大小。

不多说了，直接上步骤！

1）首先要获创建不同的productFlavors，便于分渠道打包和调试；

productFlavors {

// Define separate dev and prod product flavors.

dev {

minSdkVersion 21

...

}

Chinese\_dev {

minSdkVersion 21

...

}

1

2

3

4

5

6

7

8

9

10

2）要获取当前Flavor()，用于选择相应的资源：

def getCurrentFlavor() {

Gradle gradle = getGradle()

String taskStr = gradle.getStartParameter().getTaskRequests().toString()

println("taskStr:" + taskStr)

println()

Pattern pattern;

if (taskStr.contains("assemble")) {

pattern = Pattern.compile("assemble(\\w+)(Release|Debug)")

} else {

pattern = Pattern.compile("generate(\\w+)(Release|Debug)")

}

Matcher matcher = pattern.matcher(taskStr)

if (matcher.find()) {

returnmatcher.group(1);

}

return ""

}

3）判断是否是自己需要的Flavor,并选择libs的路径（这里需要创建不同名字的libs包，如这里可以把高德的相关jar放入cn\_libs目录下）：

def buildLibs() {

String currentFlavor = getCurrentFlavor()

println("current Flavor:" + currentFlavor)

booleanisChinese = "Chinese\_dev".equalsIgnoreCase(currentFlavor))

println("--- is Chinese ---" + isChinese)

if (isChinese) {

return'cn\_libs'

}

return 'libs'

}

4）在dependencies中加载时调用buildLibs方法来取出要加载的资源包：

dependencies {

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

5）为了完整的让不同的资源分开，还需要把AndroidManifest.xml及resource文件都分开。在main目录的同级目录下可创建两个单独的目录，如分别是cn及en，并放入对应的代码及资源。

对于AndroidManifest.xml，关键资源使用 tools:node=“merge”：

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools">

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" tools:node="merge"/>

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" tools:node="merge"/>

<application

tools:node="merge">

<meta-data

android:name="com.amap.api.v2.apikey"

android:value="key"/>

<service android:name="com.amap.api.location.APSService"/>

</application>

</manifest>

2.对于代码class的替换，若要根据flavor来切换代码，则需要不同的flavor的java包，并在其下创建下创建相同的类名，并且在main下不能再有该类，则在打包时只打对应的class文件，其他的则丢弃。如：

| Flavor A

| ——-> com.fan.util

| ————> class A

| main

| ——>com.fan.util

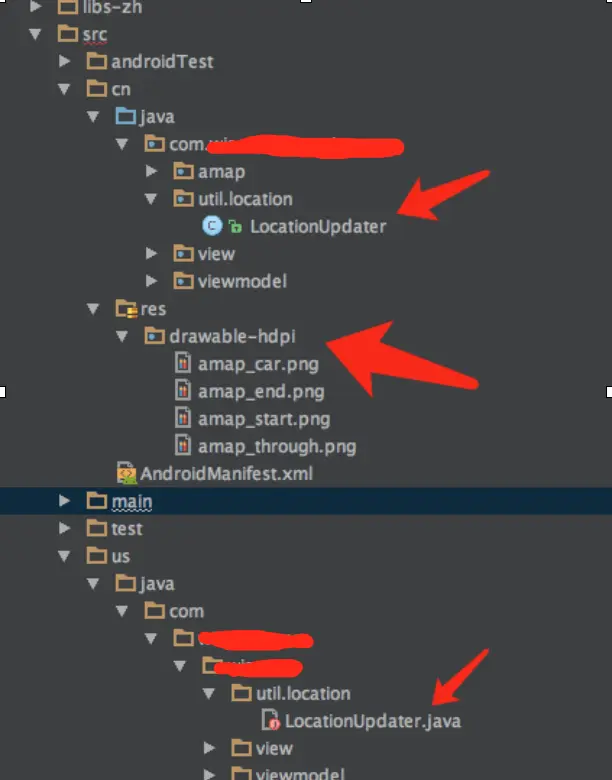
| ————>不能再有class A

| Flavor B

| ——>com.fan.util

| ————->class A

创建好后的目录大概长这样：



3.在gradle文件中配置java.srcDirs,AndroidManifest.xml及resource的位置:

sourceSets {

main {

jniLibs.srcDirs = ['libs']

java.srcDirs = ['src/main/java']

res.srcDirs = ['src/main/res']

}

dev {

java.srcDirs = ['src/us/java/']

}

Chinese\_dev {

jniLibs.srcDirs = ['libs-zh']

manifest.srcFile 'src/cn/AndroidManifest.xml'

java.srcDirs = ['src/cn/java/']

res.srcDirs = ['src/cn/res']

}

}

然后就可以通过Build Variants愉快的玩耍了！！！

————————————————

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。