<http://blog.csdn.net/u011913612/article/details/51744302>

[详细配置android studio中的gradle](http://blog.csdn.net/u011913612/article/details/51744302)

标签： [android](http://www.csdn.net/tag/android)[studio](http://www.csdn.net/tag/studio)[gradle](http://www.csdn.net/tag/gradle)[配置](http://www.csdn.net/tag/%e9%85%8d%e7%bd%ae)[多渠道打包](http://www.csdn.net/tag/%e5%a4%9a%e6%b8%a0%e9%81%93%e6%89%93%e5%8c%85)

2016-06-24 17:44 6315人阅读 [评论](http://blog.csdn.net/u011913612/article/details/51744302#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u011913612/article/details/51744302#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

Android gradle（1） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/u011913612/article/details/51744302)

这篇博客着重点在详细的介绍如何配置gradle,实现诸如多渠道打包，一次生成不同UI的多渠道apk，自定义资源目录等。如果你对gradle还没有比较清晰的认识，那么推荐你在看这篇博客之前，先看下我的另一篇博客[理解与配置android studio的gradle](http://blog.csdn.net/u011913612/article/details/51732632)，这篇博客主要让你认识gradle。

**1.打包多渠道apk**

使用gradle进行多渠道打包是对gradle的一个简单的使用，gradle的功能非常强大，它还有很多值得深究的东西。目前，网络上已经有了大量关于多渠道打包的的资料可供参考，这里不做深究，主要展示一下多渠道打包的流程，如果你想深究，那么，认真看完这篇博客的第三章，你不禁能对多渠道打包有个较好的理解，还能使用gradle做更多很酷的事。

**1.1修改AndroidManifest**

渠道信息在AndroidManifest中一般使用如下方式标示：

<meta-data

android:name="UMENG\_CHANNEL"

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

这里面的value是个可变的值，它之所以可变，是因为在build.gradle中的productFlavors 这个script block下给每个渠道商下的UNMENG\_CHANNEL\_VALUE赋了不同的值。

**1.2修改module下的build.gradle**

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">productFlavors {
2. wandoujia {}
3. baidu {}
4. productFlavors.all {
5. flavor->flavor.manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: name]
6. }
7. }</span>

这段代码中，productFlavors是一个方法，{...}是一个闭包，闭包中自定义了两个方法，这两个方法的参数又都是闭包。android dsl使用了很多简写，比如baidu函数的定义，你可以这样写：baidu({...})这样看着就像是一个定义了一个方法，它的参数是一个closrue（闭包）。productFlavors的参数---闭包里面的productFlavors和是一个对象，她和它外面长得一样的那位没什么直接关系，这个对象有个方法叫all,调用它，然后又传入一个闭包。如果这里不明白，可以看一下我在文章开头推荐的那篇博客。它里面对android dsl有详细的介绍。总之这里就是给UMENG\_CHANNEL\_VALUE赋不同的值。

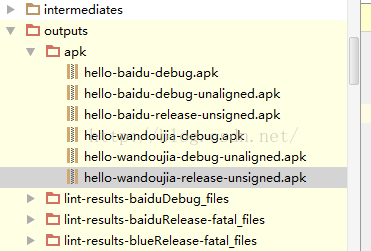
**1.3生成apk**

最好安装好gradle工具，配置好环境变量，然后在cmd或者android studio的控制台中输入以下命名：

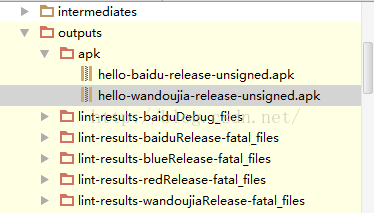
gradle assemble ，assemble是个task,此外，还有assembleRelease,assembleDebug两个task可用，它们的作用分别是全部编译、

只编译release版和只编译debug版。

gradle assemble 执行后生成的apk:



gradle assembleRelease执行后生成的apk:



**2. sourceSets,资源路径配置**

sourceSets是build.gradle中的一个script block，它的功能主要是配置工程下各个资源的路径信息，它可以像下面这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    sourceSets {
2. main {
4. }
5. }</span>

这里展示了一个空的sourceSets,它任然可以很好地工作，因为它对每一项属性都有一个默认值。如果使用默认的值的话，在build.gradle中是没有必要再配置一遍的。为了让自己的工程更加有特色一点，你可以修改他们中的任何一项。

**2.1manifest**

明确指定工程所用的manifest文件虽然并不常用，但它可以做到：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;"> sourceSets {
2. main {
3. manifest.srcFile('MineManifest.xml')
4. }
5. }</span>

我原以为做了以上修改不会有问题，但事实上出现了错误：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">Error:A problem was found with the configuration of task ':app:checkDebugManifest'.
2. > File 'E:\android\androidwork2.0\GradleTest2\app\MineManifest.xml' specified **for** property 'manifest' does not exist.</span>

说是找不到manifest，很郁闷，不知道怎么回事。我猜测可能是找不到MineManifest.xml文件，这个文件是我把AndroidManifest.xml改了个名字，那找不到怎么办呢？我尝试加上绝对路劲试试，于是修改为如下代码：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    sourceSets {
2. main {
3. manifest.srcFile('/src/main/MineManifest.xml')
4. }
5. }</span>

是的，这样就没问题了。具体为什么这么匪夷所思，我搞不懂，如果你也碰到了这样的问题，可以尝试下加上绝对路劲。

**2.2java,res,aidl等**

这些资源路劲也是可以修改的：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    sourceSets {
2. main {
3. manifest.srcFile('/src/main/MineManifest.xml')
4. java.srcDir('java')
5. res.srcDir('res')
6. aidl.srcDir('java')
7. }
8. }</span>

你可以修改他们默认的值，比如把java改成hello,这没有问题，但是如果你把res改成其他，比如改成haha,如下：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    sourceSets {
2. main {
3. manifest.srcFile('/src/main/MineManifest.xml')
4. java.srcDir('hello')
5. res.srcDir('haha')
6. aidl.srcDir('hello')
7. }
8. }</span>

就会报一下类似错误：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">Error:(13, 23) No resource found that matches the given name (at 'icon' with value '@mipmap/ic\_launcher').</span>

应该还是资源找不到，加上绝对路径试试？是的，就可以了，内部不知道是怎么实现的，但加上绝对路劲确实可以解决这种问题。修改以后变成这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">        main {
2. manifest.srcFile('/src/main/MineManifest.xml')
3. java.srcDir('/src/main/helo')
4. res.srcDir('/src/main/haha')
5. aidl.srcDir('/src/main/hello')
6. }</span>

这里关于这些资源目录的配置可能和你见到到不太一样，你见到的可能是这样的：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    sourceSets {
2. main {
3. manifest.srcFile 'AndroidManifest.xml'
4. java.srcDirs = ['src']
5. resources.srcDirs = ['src']
6. aidl.srcDirs = ['src']
7. renderscript.srcDirs = ['src']
8. res.srcDirs = ['res']
9. assets.srcDirs = ['assets']
10. }
12. instrumentTest.setRoot('tests')
13. }</span>

这两种写法是等价的，java,srcDir,res.srcDir等都是函数调用，只不过写法不太一样。既然是函数调用，我更加喜欢前面那种写法，它更好理解，后面的写法比较抽象。

**3.配置buildTypes（做到一次生成多个不同ui的apk）**

一次生成过个渠道发布的不同UI的apk,这看起来很酷。所以我觉得这一章节是全文最有意思的地方。

**3.1创建build 变量**

创建build变量是很有用的，比如说你有这样的需求：debug版打印Log,release版不打印Log,怎么实现呢？

可做如下修改：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">    buildTypes {
2. release {
3. minifyEnabled **false**
4. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
5. resValue("string","DEBUG","false")
6. }
7. debug {
8. minifyEnabled **false**
9. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
10. resValue("string","DEBUG","true")
11. }
12. }</span>

这里主要使用resValue这个函数创建了一个build变量，这个变量会被写到在BuildConfig类中，如下代码所示：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">**public** **final** **class** BuildConfig {
2. **public** **static** **final** **boolean** DEBUG = Boolean.parseBoolean("true");
3. </span>

所以在代码中使用的时候，只需要这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">        **if**(BuildConfig.DEBUG){
2. Log.d("hello","debug");
3. }</span>

这样就实现了debug版本打印Log,而release版本不打印Log。

**3.2自定义build type**

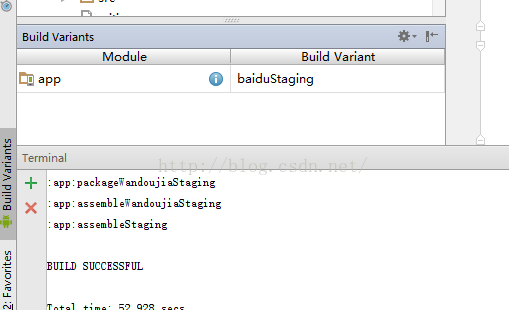
[**Android**](http://lib.csdn.net/base/android) studio中默认的build type 有两种：debug和release,想像3.1展示的那样，如果这些默认的build type满足不了项目的需求的时候，你可以自定一个build type,所有你需要做的就是新建一个type,想下面这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">staging {
2. applicationIdSuffix ".staging"
3. versionNameSuffix "-staging"
4. buildConfigField "String", "API\_URL",
5. "\"http://staging.example.com/api\""
6. }</span>

把它添加到buildTypes { }中，添加好以后，你可以在android studio左下的build Variants菜单下选择对应的staing,然后就可以编译出使用staging作为build type的apk,如图：

  
  
图中，Build Variant是可选则的，你点击它看一下就会明白一切。此外，你还可以使用命令生成apk：gradle assemblestaging

执行这条命令的前提是你的系统中已经安装了gradle工具，我还是推荐你安装一个gradle,通过敲命令你更容易理解gradle工作的过程。关于为什么可以执行这条命令，这里还想啰嗦一下，staging毕竟使我们自定义的，gradle怎么会知道它存在呢？是的这就是自动创建task,gradle会根据配置文件自动给你生成需要的task,你如你有一个staging的build type,gradle就会生成一个assemblestaging的task,如果同时，你有一个flavor叫做baidu,gradle还会给你生成一个assedblebaidu的task,同时你还可以组合他们，比如：assemblebaidustaging,这里是不是开始领悟gradle的强大了？

此外，自定义build type时，你可以充分利用已有的版本，然后在它的基础上做一些修改，就像下面这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

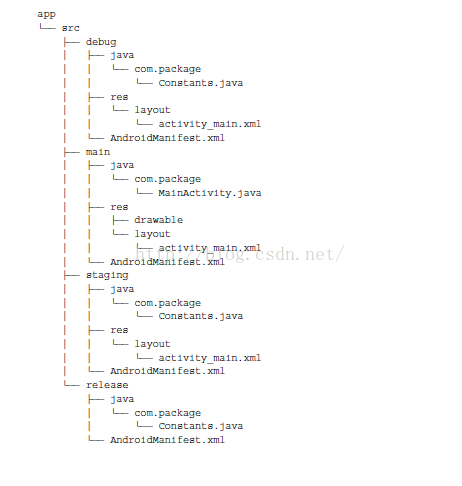
[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">android {
2. buildTypes {
3. staging.initWith(buildTypes.debug)
4. staging {
5. applicationIdSuffix ".staging"
6. versionNameSuffix "-staging"
7. debuggable = **false**
8. }
9. }
10. }</span>

可以看到staging使用debug类型初始化自己，然后在此基础上做了些定制。

**3.3build type与 source set**

当你创建一个build type时，gradle会为你生成一个source set,默认的，它的目录和你build type的名字相同，但是它不会自动为你创建目录，你需要自己创建目录，你只需要在和main同级的目录下创建和build type名字相同的目录即可，这个目录结构和你原来的那个目录很相似，下图是添加staging后的目录结构示意图：



gradle这种强大的功能为你打开了一个新的世界，这使得你为不同的构建类型添加不同的代码，添加不同的资源成为可能。

**3.3.1资源的合并**

资源的处理比较特殊，当你使用不同的source set时，Drawable和layout文件将会完全的覆盖main source set下的同名文件，但是value下的文件，比如string.xml就不是这样的，gradle合并的是内容而不是文件。一定要注意它合并的是内容，这很重要。比如，某个build type 的source set中有这样的string.xml:

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;"><resources>
2. <string name="app\_name">TypesAndFlavors STAGING</string>
3. </resources></span>

main source set下的string.xml是这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;"><resources>
2. <string name="app\_name">TypesAndFlavors</string>
3. <string name="hello\_world">Hello world!</string>
4. </resources></span>

那么使用main source set下的string.xml合并build type下的string.xml的内容，合并结果是这样的：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;"><resources>
2. <string name="app\_name">TypesAndFlavors STAGING</string>
3. <string name="hello\_world">Hello world!</string>
4. </resources></span>

**3.3.2给build type添加类文件**

给build type添加类文件时，要注意，这个过程是互斥的，就是说main source set下的[**Java**](http://lib.csdn.net/base/javase)文件不能很buile type下的source set下的java文件相同，比如，如果staging 的source set和main source set下都有一个A.java,那么会暴错，gradle会认为你的一个类定义了两次。

**3.3.3manifest文件的合并**

manifest文件的合并和string.xml文件的合并类似，是文件内容的合并。因此，build type中的manifest文件不需要完全复制main下的manifest文件，只需要写出不同于main source set下的manifest的部分。

**3.3.4再谈productFlavors**

除了一次性生成多个productFlavors，和build type 类似，productFlavors也能生成自己的source set目录，这个目录也需要你自己区创建，名字和productFlavors声明的flavor名字相同即可。更进一步，这个目录和buile type的目录名字可以组合使用。比如，我在productFlovors下定义了一个名为baidu的flavor，定义了一个staging的build type,然后目录名就可以为baidustaging。

**3.3.5一个实现生成多apk,不同UI,的工程实例**

第一步：定义build type 和 product flavors 以及他们的sourceSets

这我们已经做过了，现在 直接把build.gradle贴出来：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

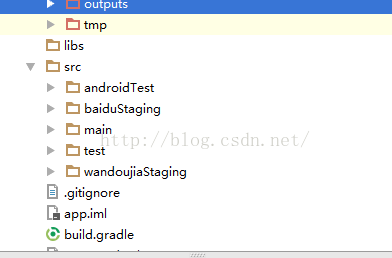
[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">apply plugin: 'com.android.application'
3. android {
4. compileSdkVersion 23
5. buildToolsVersion "23.0.3"
7. defaultConfig {
8. applicationId "com.konka.gradletest2"
9. minSdkVersion 16
10. targetSdkVersion 23
11. versionCode 1
12. versionName "1.0"
13. }
14. buildTypes {
15. release {
16. minifyEnabled **false**
17. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
18. resValue("string","DEBUG","false")
19. }
20. debug {
21. minifyEnabled **false**
22. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
23. resValue("string","DEBUG","true")
24. dependencies {
26. }
27. }
28. staging {
29. applicationIdSuffix ".staging"
30. versionNameSuffix "-staging"
31. buildConfigField "String", "API\_URL",
32. "\"http://staging.example.com/api\""
33. }
34. }
35. sourceSets {
36. main {
37. manifest.srcFile('/src/main/MineManifest.xml')
38. java.srcDir('/src/main/helo')
39. res.srcDir('/src/main/haha')
40. aidl.srcDir('/src/main/hello')
41. }
42. baiduStaging {
43. manifest.srcFile('/src/main/MineManifest.xml')
44. java.srcDir('/src/main/helo')
45. res.srcDir('/src/baiduStaging/haha')
46. aidl.srcDir('/src/main/hello')
47. }
48. wandoujiaStaging {
49. manifest.srcFile('/src/main/MineManifest.xml')
50. java.srcDir('/src/main/helo')
51. res.srcDir('/src/wandoujiaStaging/haha')
52. aidl.srcDir('/src/main/hello')
53. }
54. }
55. productFlavors {
56. wandoujia {
57. }
58. baidu {
59. }
60. productFlavors.all {
61. flavor->flavor.manifestPlaceholders = [UMENG\_CHANNEL\_VALUE: name]
62. }
63. }
64. }
66. dependencies {
67. compile fileTree(dir: 'libs', include: ['\*.jar'])
68. testCompile 'junit:junit:4.12'
69. compile 'com.android.support:appcompat-v7:23.3.0'
70. }</span>

大家注意flavor和build type组合的方式：flavors后面跟build type,builed type首字母大写,比如这里的：wandoujiaStaging,baiduStaging

第二部：修改目录

看一下图就明白了：



第三步：修改资源

现在只把引用程序的图标和名字给替换掉，这里不多说，我想大家都会弄。其中标题在string.xml下，图标在mipmap-xxhdpi目录下，大家注意他们的合并方式，前面都讲过了。

第四步：生成apk

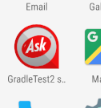
直接运行gradle assemblestaging

第五步： 签名并安装

第六步：成果展示：

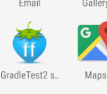
baidu:

图标                                      界面



wandoujia:

图标                                  界面

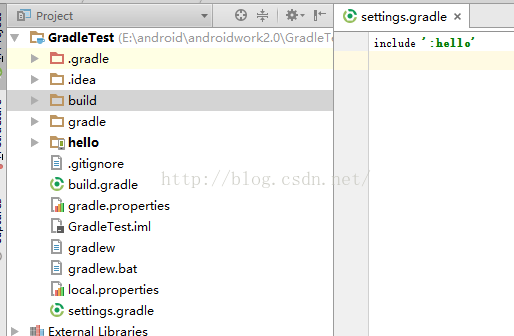


**4 多个模块**

**4.1添加一个android app**

多模块的工程就是在项目的根目录下有多个module,在android studio下这很简单：

比如就你现在有一个这样的工程：



你只需要做一件事：：在你的工程上右键，选择新建mudole。

是的就这么简单，现在看看工程的样子：



是的，这个时候，settings.gradle中多了一项，他就是我们新加的module的名字，它其实就是工程顶层目录下的一个目录的名字。这个名字你可以随便改，module你也可以随便加。

**4.2添加一个java library 及其他**

添加一个[**Java**](http://lib.csdn.net/base/java)library时，需要修改module下的build.gradle,修改后看起来应该这样：

**[java]** [view plain](http://blog.csdn.net/u011913612/article/details/51744302) [copy](http://blog.csdn.net/u011913612/article/details/51744302)

[print?](http://blog.csdn.net/u011913612/article/details/51744302)[在CODE上查看代码片](https://code.csdn.net/snippets/1728780)

1. <span style="font-size:14px;">apply plugin: 'java'
2. dependencies {
3. compile fileTree(dir: 'libs', include: ['\*.jar'])
4. }</span>

在android studio中，这些都不用你来做，和创建一个app一样，你只需要在项目名字上右键，选件new,然后接着选择java library就可以了，其他的创建方式都类似，你只需要知道他们的build.gradle是不同的，然后在必要的时候自定义的去配置他们。