# 第三方库引入冲突

# 手动打印库依赖信息

使用以下命令打印依赖信息

gradlew app:dependencies

如果信息太长，控制台显示不下，可以把打印信息输出到一个文件中

gradlew app:dependencies > log.txt

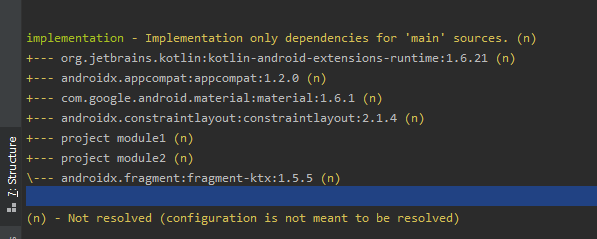
运行结束之后，会在当前工程目录下输出log.txt文件，查看该文件即可

还可以打印某个library的信息，例如libgoogle

gradlew libgoogle:dependencies  
gradlew libgoogle:dependencies > log.txt

直接打印某个模块的依赖，例如打印app模块的依赖

gradlew :app:dependencies --configuration implementation



这种只是打印出app模块的直接依赖，并没有打印出依赖树。

使用下面命令打印某种条件的依赖：

gradlew :app:dependencies --configuration releaseRuntimeClasspath

上述命令可以添加的条件有：

androidTestUtil

compile

publish

provided

api

implementation

runtimeOnly

compileOnly

archives

debugAndroidTestRuntimeClasspath

debugUnitTestRuntimeClasspath

debugRuntimeClasspath

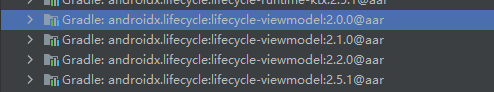
releaseRuntimeClasspath

releaseUnitTestRuntimeClasspath

# 控制依赖版本

## 问题引入

在项目中发现viewmodel库其实引入了4个版本



其中viewmodel:2.0.0由其中一个library引入（module2）

implementation 'androidx.legacy:legacy-support-v4:1.0.0'

viewmodel:2.1.0由module1引入

api 'com.google.android.gms:play-services-auth:18.0.0'

viewmodel:2.2.0由coommon引入

implementation 'com.google.android.material:material:1.6.1'

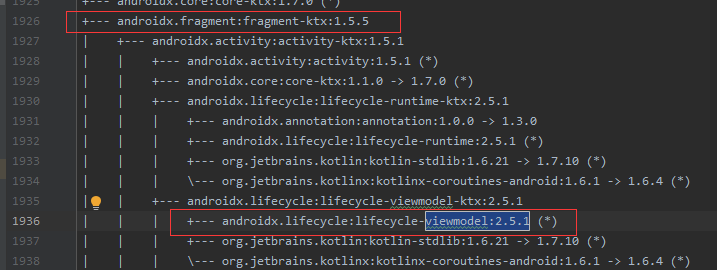
viewmodel2.5.1由app引入的

implementation 'androidx.fragment:fragment-ktx:1.5.5'

## 使用命令查看依赖关系

app依赖关系

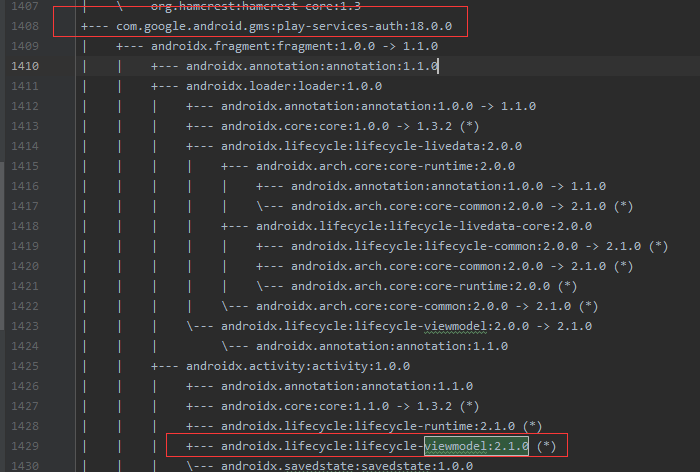
gradlew app:dependencies > log.txt



所以viewmodel:2.5.1版本就是fragment-ktx:1.5.5这里引入的

mdule1依赖关系

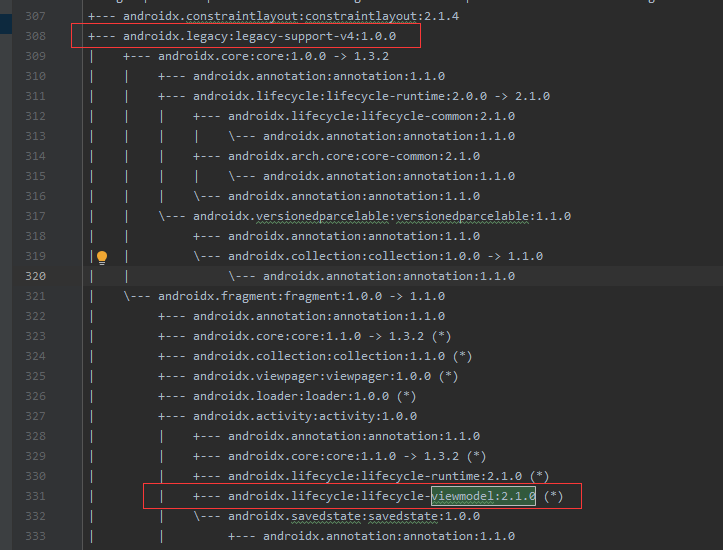
gradlew module1:dependencies > log.txt



所以viewmodel:2.1.0是module1引入的

mdule1依赖关系

gradlew module2:dependencies > log.txt

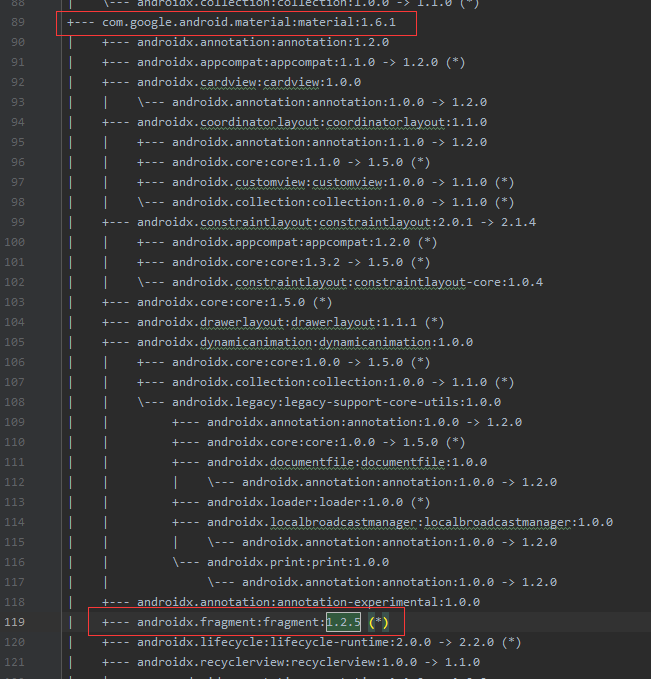


所以viewmodel:2.1.0是module2引入的

common依赖关系

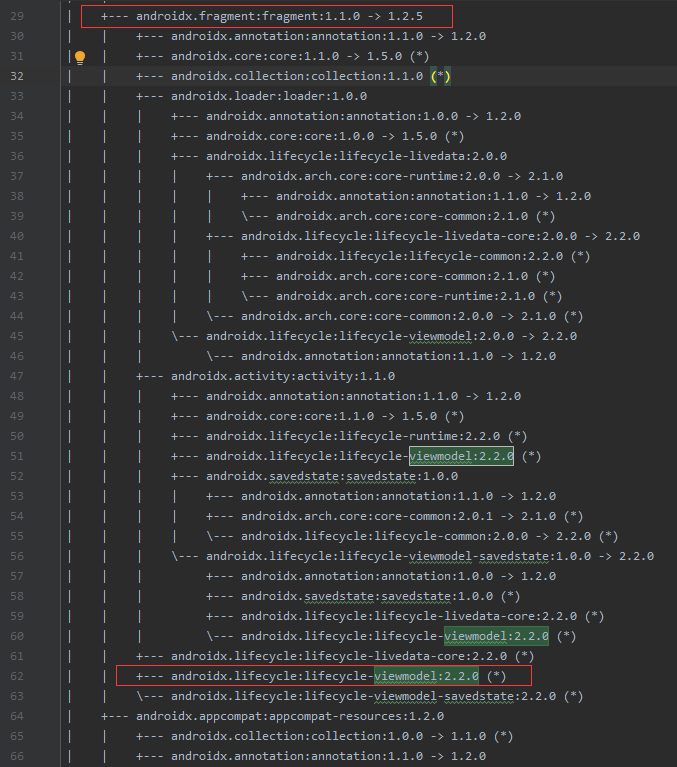
gradlew comomon:dependencies > log.txt

com.google.android.material:material:1.6.1引入

androidx.fragment:fragment:1.2.5 

androidx.fragment:fragment:1.2.5引入

androidx.lifecycle:lifecycle-viewmodel:2.2.0



# 控制依赖版本处理

如果同一个项目中对同一个模块有不同版本的两个依赖的话，默认情况下Gradle会在解析完DAG（有向无环图）之后，选择版本最高的那个依赖包。

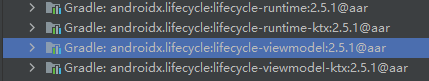
## 指定某个library的依赖版本号

例如module1现在引入的是viewmodel:2.1.0,因为是第三方库自动引入的，手动在build.gradle中添加依赖引入：

implementation 'androidx.lifecycle:lifecycle-viewmodel:2.5.1'

这样同步之后，External Libraries就没有了之前2.1.0的版本。

放到公共库common中，并使用api传递依赖，同步之后发现其他版本也没有，就只有最新的2.5.1版本



## 排除某个依赖

当我们引入的第三方库包含了某些不需要的依赖时，可以手动排除不需要的依赖。例如com.google.android.material:material:1.6.1引入了

androidx.cardview:cardview:1.0.0，当我们引入了material库时，是直接可以使用cardview的。当我们使用exclude排除cardview之后，代码里面就找不到cardview这个库了。

implementation ('com.google.android.material:material:1.6.1') **{** exclude group: 'androidx.cardview', module: 'cardview'  
**}**

## 强制依赖

有时候可能使用库的最新版本比我们想使用的版本高，这时需要强制使用某个库的版本，可以使用enforcedPlatform 指定版本号。

implementation enforcedPlatform('androidx.lifecycle:lifecycle-viewmodel:2.1.0')

api enforcedPlatform('androidx.lifecycle:lifecycle-viewmodel:2.1.0')

可以在公共模块使用api enforcedPlatform指定版本号，其他模块都引用公共模块，如果没有引用公共模块，那个模块使用的还是引用库的最新版本。

## 传递依赖（后面需要研究）

如果我们项目中依赖了某个module，而这个module又依赖了另外的module，我们叫做传递依赖。在这种情况下，如果我们希望控制传递依赖的版本，比如说将传递依赖的版本升级为一个新的版本，那么可以使用dependency constraints：

上面讲的是传递依赖的版本升级。同样是传递依赖，如果本项目也需要使用到这个传递依赖的module，但是需要使用到更低的版本（因为默认gradle会使用最新的版本），就需要用到版本降级了。strictly

我们可以在implementation中指定特定的version即可。

strictly表示的是强制匹配特定的版本号，除了strictly之外，还有require，表示需要的版本号大于等于给定的版本号。prefer，如果没有指定其他的版本号，那么就使用prefer这个。reject，拒绝使用这个版本。

除此之外，你还可以使用Java Platform Plugin来指定特定的platform，从而限制版本号。

## 强制统一使用库版本

在app模块下添加下面脚本，发现没有任何效果

android **{**

configurations.all **{** resolutionStrategy.force 'androidx.lifecycle:lifecycle-viewmodel:2.5.1'  
 **}**

**}**

在common模块模块添加也没有任何效果。

产生上面的问题是因为上述脚本单独放到了一个子 模块下，需要下面对多有模块配置才可以：

configurations.all **{** configurations.all **{** resolutionStrategy.force 'androidx.lifecycle:lifecycle-viewmodel:2.3.1'  
 **}**// resolutionStrategy.eachDependency { DependencyResolveDetails details ->  
// def requested = details.requested  
// if (requested.group == 'androidx.lifecycle') {  
// if (requested.name.startsWith("lifecycle-viewmodel")) {  
// details.useVersion '2.5.1'  
// }  
// }  
// }  
**}**

configurations.all 和resolutionStrategy.eachDependency 2中方式都是可以的。

# 参考文档

<https://www.lmlphp.com/user/151865/article/item/3180179>

<https://blog.csdn.net/angcyo/article/details/102582771>

<https://www.jianshu.com/p/9470e9ffa749>

<https://www.pudn.com/news/62e6ab21864d5c73ac347b54.html>

<https://www.dandelioncloud.cn/article/details/1531937108822618113>

<https://www.jianshu.com/p/f4fe8e1a268f>

<https://www.jianshu.com/p/ac2e1efb2b85>

<https://www.javaclub.cn/tool/24281.html>

<https://blog.csdn.net/sinat_31057219/article/details/105493617>

式样

微软雅黑 Light 小四

段落 固定值 23磅

# 目录1

## 目录2

### 目录3

#### 目录4

##### 目录5