android gradle依赖：implementation 和compile的区别

沐风雨木 关注

7 2018.02.27 20:05:14 字数 965 阅读 116,084

> 2017 年google 后，Android studio版本更新至3.0，更新中，连带着com.android.tools.build:gradle 工具也升级到了3.0.0，在3.0.0中使用了最新的Gralde 4.0 里程碑版本作为gradle的编译版本，该版本gradle编译速度有所加速，更加欣喜的是，完全支持Java8。
> 当然，对于Kotlin的支持，在这个版本也有所体现，Kotlin插件默认是安装的。

我们来看看新建一个项目在 `Module` 中的 `dependencies` 中的变化。

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

下面我们来看看他们之前的差异：

首先是2.x版本的依赖方式

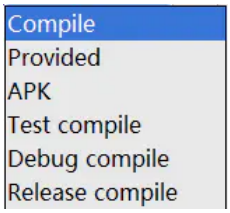

image.png

再来看看3.0的

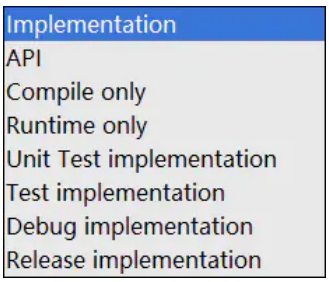

image.png

可以看到在 `Android studio3.0` 中， `compile` 依赖关系已被弃用，被 `implementation` 和 `api` 替代， `provided` 被 `compile only` 替代， `apk` 被 `runtime only` 替代。

我们先来看看 `implementation` 和 `api` 的区别：

api：跟 2.x 版本的 `compile` 完全相同

implementation：使用了该命令编译的依赖，它仅仅对当前的 `Module` 提供接口。例如我们当前项目结构如下

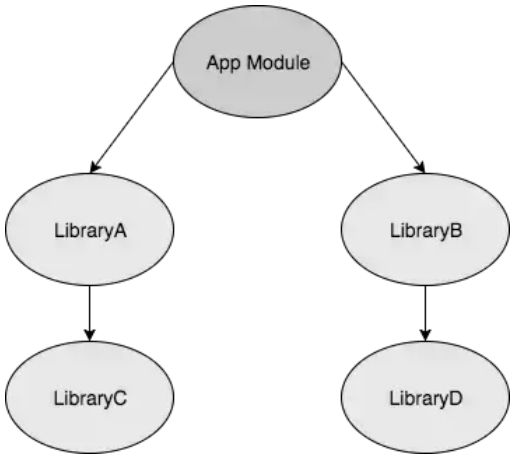

image.png

`LibraryA` 中引用了 `LibraryC` 的库，如果对 `LibraryC` 的依赖用的是 `implementation` 关键字。如下：

```
dependencies {
    . . . .
    implementation project (path: ': libraryC')
}
```

那么 `LibraryC` 中的接口，仅仅只能给 `LibraryA` 使用，而我们的 `App Module` 是无法访问到 `LibraryC` 提供的接口的，也就是将该依赖隐藏在内部，而不对外部公开。这就是 `implementation` 关键字的作用。

建议

在 `Google IO` 相关话题的中提到了一个建议，就是依赖首先应该设置为 `implement` 的，如果没有错，那就用 `implement` ，如果有错，那么使用 `api` 指令，这样会使编译速度有所增快。

那为什么要这么做呢?

答案是： 1. 加快编译速度。2. 隐藏对外不必要的接口。

为什么能加快编译速度呢?

这对于大型项目含有多个 `Module` 模块的， 以上图为例，比如我们改动 `LibraryC` 接口的相关代码，这时候编译只需要单独编译 `LibraryA` 模块就行， 如果使用的是 `api` 或者旧时代的 `compile` ， 由于 `App Module` 也可以访问到 `LibraryC` ,所以 `App Module` 部分也需要重新编译。当然这是在全编的情况下。

还不熟悉 2.x 版本依赖的可以看看下面的说明，括号里对应的是 3.0 版本的依赖方式。

compile（api）

这种是我们最常用的方式，使用该方式依赖的库将会**参与编译和打包**。

当我们依赖一些第三方的库时，可能会遇到 `com.android.support` 冲突的问题，就是因为开发者使用的 `compile` 依赖的 `com.android.support` 包，而他所依赖的包与我们本地所依赖的 `com.android.support` 包版本不一样，所以就会报 `All com.android.support libraries must use the exact same version specification (mixing versions can lead to runtime crashes` 这个错误。

解决办法可以看这篇博客：com.android.support冲突的解决办法

provided（compileOnly）

只在编译时有效，不会参与打包
可以在自己的 `module` 中使用该方式依赖一些比如 `com.android.support`， `gson` 这些使用者常用的库，避免冲突。

apk（runtimeOnly）

只在生成 `apk` 的时候参与打包，编译时不会参与，很少用。

testCompile（testImplementation）

`testCompile` 只在单元测试代码的编译以及最终打包测试apk时有效。

debugCompile（debugImplementation）

`debugCompile` 只在 **debug** 模式的编译和最终的 **debug apk** 打包时有效

releaseCompile（releaseImplementation）

`Release compile` 仅仅针对 **Release** 模式的编译和最终的 **Release apk** 打包。

参考链接：
Android Studio3.x新的依赖方式（implementation、api、compileOnly）
还再用compile依赖？那你就落后啦
android gradle tools 3.X 中依赖，implement、api 指令