

layout: post title: "Gradle使用方法与技巧「汇总」，未完...." subtitle: "主要介绍Gradle的使用方法" date: 2016-1-4 author: "JackSunny" header-img: "img/post-bg-rwd.jpg" tags: - Android

- Gradle

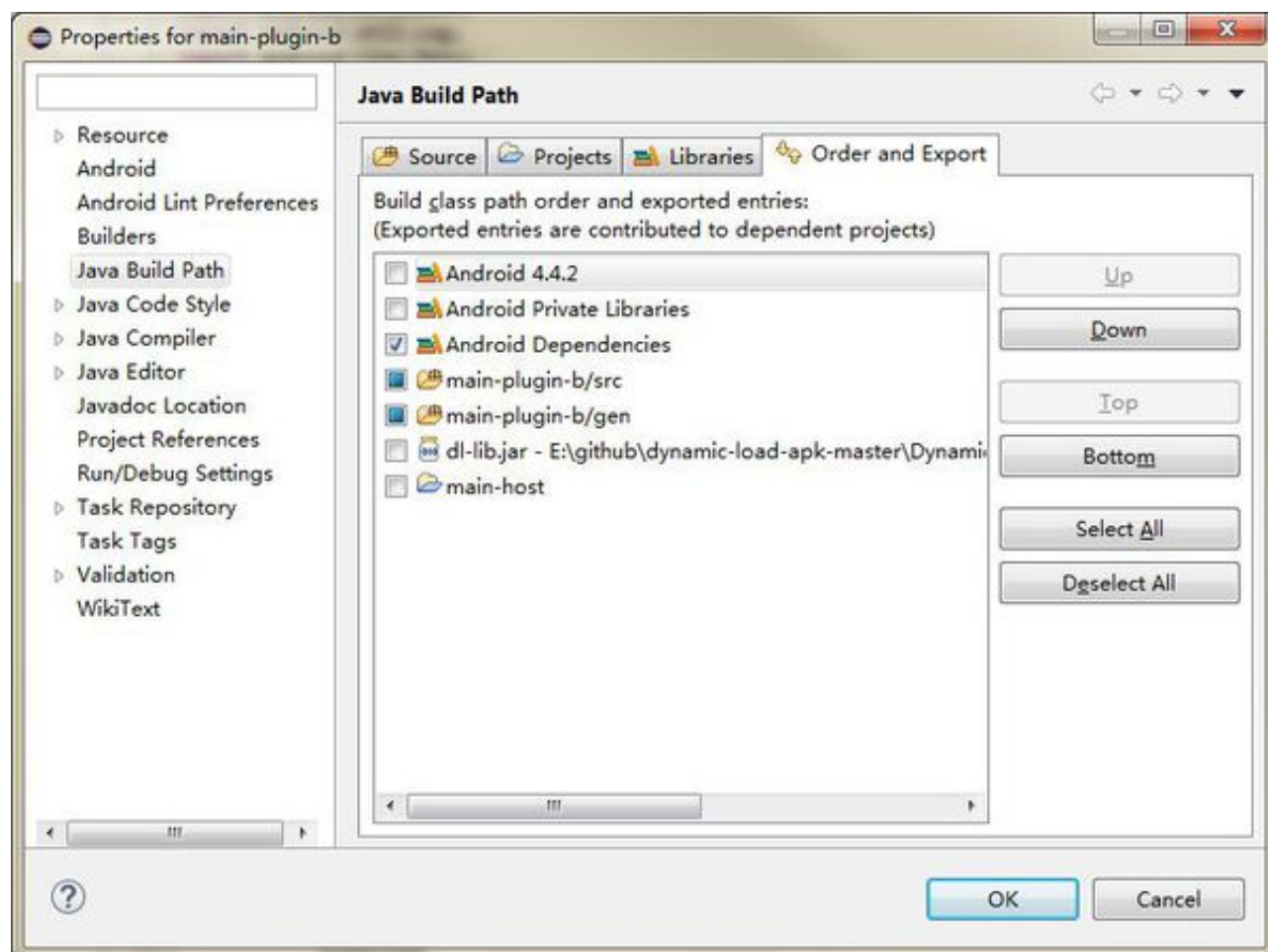
导语

本文你会看到

1. Gradle的一些使用方法和技巧

打包时不包含某些jar包

有的时候我们不需要打入某些jar包（如作为主工程插件，不能包含两份jar包），在eclipse中我们是这样操作的



在 AS 中，包含依赖包一般姿势是这样子的(以 v4 包为例)

```
dependencies {
    compile fileTree(dir: 'libs', include: '*.jar')
    compile files('lib/android-support-v4.jar')
}
```

编译通过但是打包不包含通过 provided 即可

```
dependencies {
    compile fileTree(dir: 'libs', include: '*.jar')
    provided files('lib/android-support-v4.jar')
}
```

项目与 build 的特定全局变量

用 gradle 可以自动生成 BuildConfig 类，其中能够生成附加字段。这对配置服务器 URL 之类的工作十分有用，使用它也能轻松开启或关闭功能。

```
defaultConfig {
    buildConfigField "String", "TWITTER_TOKEN", "'SDASJHDKAJSK'"
}
buildTypes {
    debug {
        buildConfigField "String", "API_URL", "'http://api.dev.com/'"
        buildConfigField "boolean", "REPORT_CRASHES", "true"
    }
    release {
        buildConfigField "String", "API_URL", "'http://api.prod.com/'"
        buildConfigField "boolean", "REPORT_CRASHES", "false"
    }
}
```

可以从 BuildConfig 的 final 类 BuildConfig.TWITTER_TOKEN, BuildConfig.REPORTCRASHES 与 BuildConfig.API_URL 进行访问，后两个根据所在的 build 类不同也会有差异。

每个 buildtype 的名字、版本与 app id 都不同

这样一来使用者就能同时安装发布版与 debug 版了（切记，在安卓系统中无法安装同名的不同应用）。用户可以在崩溃报告工具中以不同的版本名筛选问题与崩溃。通过查看应用名很容易找到目前所运行的版本。

```

android {
    buildTypes {
        debug {
            applicationIdSuffix ".debug"
            versionNameSuffix "-debug"
            resValue "string", "app_name", "CoolApp (debug)"
            signingConfig signingConfigs.debug
        }
        release {
            resValue "string", "app_name", "CoolApp"
            signingConfig signingConfigs.release
        }
    }
}

```

隐私信息

在Android系统中，所有应用都必须经过证书数字签名才能安装，以便系统能够识别应用的作者。而其中有些属于敏感信息，不应被别人看到。

使用者永远不该将这类信息check in到源代码管理工具中。

有些人主张，每个人都应当有自己的本地配置文件，甚至用全局的`~/.gradle/build.gradle`，不过如果你要执行持续集成（CI）或部署，特别是没有自己CI服务器的情况下，不应在CVS系统里存放任何类型的纯文本凭证。

```

signingConfigs {
    release {
        storeFile      "${System.env.COOL_APP_PRIVATE_KEY}"
        keyAlias       "${System.env.COOL_APP_ALIAS}"
        storePassword  "${System.env.COOL_APP_STORE_PW}"
        keyPassword    "${System.env.COOL_APP_PW}"
    }
}

```

因此，可以通过环境变量将敏感信息提供给自己的持续集成服务器，而无需担心将任何“危险”信息check in到公司了。

自动生成版本名称（versionName）与版本号（versionCode）

将你的版本拆分成逻辑组件，分别管理。不用再考虑版本号修改的是否正确，也不用担心版本名更新的是否合适了。

```

def versionMajor = 1
def versionMinor = 0
def versionPatch = 0
android {
    defaultConfig {
        versionCode versionMajor * 10000 + versionMinor * 100 + versionPatch
        versionName "${versionMajor}.${versionMinor}.${versionPatch}"
    }
}

```

给BuildConfig增加git hash与build时间

```

def gitSha = 'git rev-parse --short HEAD'.execute([], project.rootDir).text.trim()
def buildTime = new Date().format("yyyy-MM-dd'T'HH:mm:ss'Z'", TimeZone.getTimeZone('UTC'))
android {
    defaultConfig {
        buildConfigField "String", "GIT_SHA", "\"${gitSha}\""
        buildConfigField "String", "BUILD_TIME", "\"${buildTime}\""
    }
}

```

现在有两个可用变量：BuildConfig.GIT_SHA和BuildConfig.BUILD_TIME，用来结合日志与提交信息或者build时间再好不过。

扣紧安全带

想要快速完成部署，只需创建dev类型，将minSdkVersion设定为21。注意：这样做的话，就无法获得针对真实minSdk的合适linting了。因此很明显只能用在日常工作中，而不能用在发布时。这样一来，安卓gradle插件可以将应用程序的每个模块构建为不同的dex文件（pre-dex），并生成可以在Android Lollipop及以上系统中测试的APK包，而无需再耗费大量时间进行dex合并进程。

```

android {
    productFlavors {
        dev {
            minSdkVersion 21
        }
        prod {
            // The actual minSdkVersion for the application.
            minSdkVersion 14
        }
    }
}

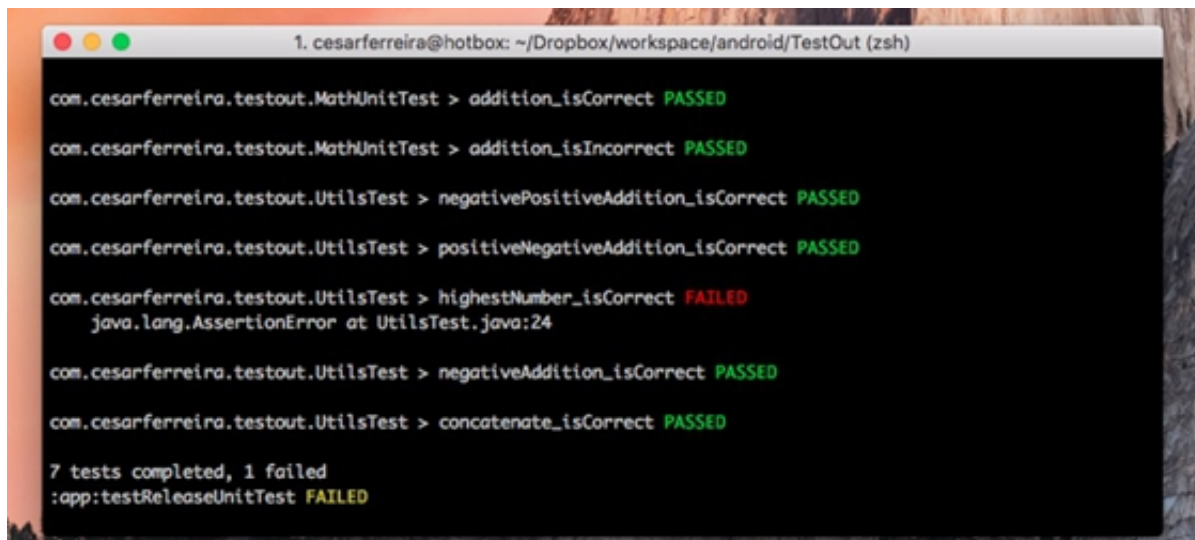
```

直接将单元测试结果输出到console中

使用这个小技巧，我们可以实时看到安卓单元测试的记录结果。

```
android {  
    ...  
  
    testOptions.unitTests.all {  
        testLogging {  
            events 'passed', 'skipped', 'failed', 'standardOut', 'standardError'  
            outputs.upToDateWhen { false }  
            showStandardStreams = true  
        }  
    }  
}
```

现在运行测试时，输出结果如下：



```
1. cesarferreira@hotbox: ~/Dropbox/workspace/android/TestOut (zsh)  
com.cesarferreira.testout.MathUnitTest > addition_isCorrect PASSED  
com.cesarferreira.testout.MathUnitTest > addition_isIncorrect PASSED  
com.cesarferreira.testout.UtilsTest > negativePositiveAddition_isCorrect PASSED  
com.cesarferreira.testout.UtilsTest > positiveNegativeAddition_isCorrect PASSED  
com.cesarferreira.testout.UtilsTest > highestNumber_isCorrect FAILED  
    java.lang.AssertionError at UtilsTest.java:24  
com.cesarferreira.testout.UtilsTest > negativeAddition_isCorrect PASSED  
com.cesarferreira.testout.UtilsTest > concatenate_isCorrect PASSED  
7 tests completed, 1 failed  
:app:testReleaseUnitTest FAILED
```

Gradle, tell me I'm pretty

全部放在一起的话，顺序如下：

```
android {  
    ...  
    buildTypes {  
        def BOOLEAN = "boolean"  
        def TRUE = "true"  
        def FALSE = "false"  
        def LOG_HTTP_REQUESTS = "LOG_HTTP_REQUESTS"  
        def REPORT_CRASHES = "REPORT_CRASHES"  
        def DEBUG_IMAGES = "DEBUG_IMAGES"  
  
        debug {  
            ...  
            buildConfigField BOOLEAN, LOG_HTTP_REQUESTS, TRUE  
            buildConfigField BOOLEAN, REPORT_CRASHES, FALSE  
            buildConfigField BOOLEAN, DEBUG_IMAGES, TRUE  
        }  
  
        release {  
            ...  
            buildConfigField BOOLEAN, LOG_HTTP_REQUESTS, FALSE  
            buildConfigField BOOLEAN, REPORT_CRASHES, TRUE  
            buildConfigField BOOLEAN, DEBUG_IMAGES, FALSE  
        }  
    }  
}
```

感谢

- [受用不尽的Gradle使用方法与技巧](#)

