## [Gradle使用手册（一）：为什么要用Gradle？](http://www.android-studio.org/index.php/docs/guide/134-gradle)

原文地址：http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Using-sourceCompatibility-1.7  
翻译：Android Studio 中文组（[大锤译](http://www.android-studio.org/index.php/team/130-team-dachui)）

鉴于Android Studio使用Gradle构建项目，在IDE的使用过程中可能会遇到很多令人头疼的问题。大锤也是第一次接触，为有个基本的认识，特找来Android官方手册，和大家一起学习学习。翻译不当之处，望指出。

**为什么要用Gradle？**

Gradle是比较先进的构建系统，也是一个很好的构建工具，允许通过插件自定义构建逻辑

以下是为什么Android Studio选择Gradle的主要原因：

        使用领域专用语言（Domain Specific Language）来描述和处理构建逻辑。（以下简称DSL）

        基于Groovy。DSL可以混合各种声明元素，用代码操控这些DSL元素达到逻辑自定义。

        支持已有的Maven或者Ivy仓库基础建设

        非常灵活，允许使用best practices，并不强制让你遵照它的原则来。

        其它插件时可以暴露自己的DSL和API来让Gradle构建文件使用。

        允许IDE集成，是很好的API工具 ****需要准备：****

       Gradle 1.6 or 1.7

       SDK with Build Tools 17.0.0 (released 5/16/2013)

****Basic Project****

      在Gradle项目的根目录下，有个叫build.gradle的文件，它描述了这个项目的整体构建基础。

****build文件****

      最基本的java程序，它的build.gradle文件就一句话：

apply plugin: 'java'

最基本的Android项目，它的build.gradle如下：

buildscript {  
    repositories {  
          mavenCentral()  
     }  
  
    dependencies {  
          classpath 'com.android.tools.build:gradle:0.5.6'  
     }  
}  
apply plugin: 'android'  
android {  
       compileSdkVersion 17  
}

我们一步步来分析一下上面三部分的内容。

1. buildscript{...} 配置了驱动build的代码，它声明将在Maven中央仓库，取一个classpath dependency，也就是Android plugin for Gradle v0.5.6
2. apply plugin     指明了用到的plugin是android，就像前面java程序中，用的plugin是java一样
3. android{...}     中配置了所有android构建的参数，这里也就是Android DSL的入口点。

默认的，只有目标编译环境是必要的，也就是compileSdkVersion这个属性。这和以前在project.properties中的target属性类似。

值得注意的是，如果你在Android项目中写 apply plugin:java 而不是apply plugin:android的话，将会build失败。

## [如何将Eclipse中的项目迁移到Android Studio 中](http://www.android-studio.org/index.php/docs/guide/129-eclipse-to-androidstdio)

英文地址：http://developer.android.com/sdk/installing/migrate.html  
翻译：Android Studio 中文组（[大锤译](http://www.android-studio.org/index.php/team/130-team-dachui)）

如果你之前有用Eclipse做过安卓开发，现在想要把Eclipse中的项目导入到Android Studio的环境中，那么首先要做的是生成Build Gradle的文件。因为Android Studio 是用Gradle来管理项目的，具体操作步骤如下：

**从Eclipse中导出**  
        1.将你的ADT插件版本升级到22.0以上。  
        2.在Eclipse中，选择File-->Export。  
        3.在弹出的导出窗口中，打开Android的文件夹，选择“Generate Gradle Build Files”。  
        4.选中你想要导入到Android Studio中的项目，Finish。  
PS:导出的项目将会和原来的项目在同一目录，覆盖原来的同时，会新增一个叫build.gradle的文件，导入Android Studio时将首先读取这个文件。  
**导入到Android Studio**  
        1.在Android Studio 中，首先关掉你当前的打开的项目。  
        2.在欢迎界面，点击Import Project（注：也是可以直接在菜单选择Import project的）  
        3.选中你在Eclipse中导出的项目，展开目录，点击build.gradle文件，然后OK  
        4.在之后的弹出对话框中，会要求你选择Gradle的配置，选中Use gradle wrapper.(注：也可以自定义你本机装的Gradle)  
PS：如果没有Grade build文件，也是可以将普通的安卓项目导入到Android Studio中，它会用现有的Ant build.但为了更好地使用之后的功能和充分使用构建变量，还是强烈地建议先从ADT插件中生成Gradle文件再导入Android Studio

## [Gradle使用手册（二）：项目结构](http://www.android-studio.org/index.php/docs/guide/135-gradle-2)

原文地址：http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Using-sourceCompatibility-1.7  
翻译：Android Studio 中文组（[大锤译](http://www.android-studio.org/index.php/team/130-team-dachui" \t "http://www.android-studio.org/index.php/docs/guide/_blank)）

续[Gradle（一）](http://www.android-studio.org/index.php/docs/guide/134-gradle" \t "http://www.android-studio.org/index.php/docs/guide/_blank)

##### ****项目结构****

两大组件（source sets）： main source code 和 test code. 它们分别在以下两个目录中：

src/main/  
src/instrumentTest/

这两个目录里面，又分别有各自的代码源文件和资源文件。

java/  
resources/

对于 Android plugin, 又有以下额外的目录：

AndroidManifest.xml  
res/  
assets/  
aidl/  
rs/  
jni/

##### ****配置项目结构****

根据Gradle文档说明，可以通过以下两个方法来重新配置项目结构。

sourceSets {  
    main {  
        java {  
            srcDir 'src/java'  
        }  
  
        resources {  
            srcDir 'src/resources'  
        }  
    }  
}

或者：

sourceSets {  
    main.java.srcDirs = ['src/java']  
    main.resources.srcDirs = ['src/resources']  
}

而Android的项目也类似，如下列所示：

android {  
    sourceSets {  
        main {  
            manifest.srcFile 'AndroidManifest.xml'  
            java.srcDirs = ['src']  
            resources.srcDirs = ['src']  
            aidl.srcDirs = ['src']  
            renderscript.srcDirs = ['src']  
            res.srcDirs = ['res']  
            assets.srcDirs = ['assets']  
        }  
        instrumentTest.setRoot('tests')  
    }  
}

注：setRoot这个方法将所有src/instrumentTest目录下的文件及文件夹移到了tests/目录下。

## [引入第三方jar包（以google play service为例）](http://www.android-studio.org/index.php/docs/guide/132-androidstudio-inport-jar)

原文地址：http://forum.xda-developers.com/showthread.php?t=2421186

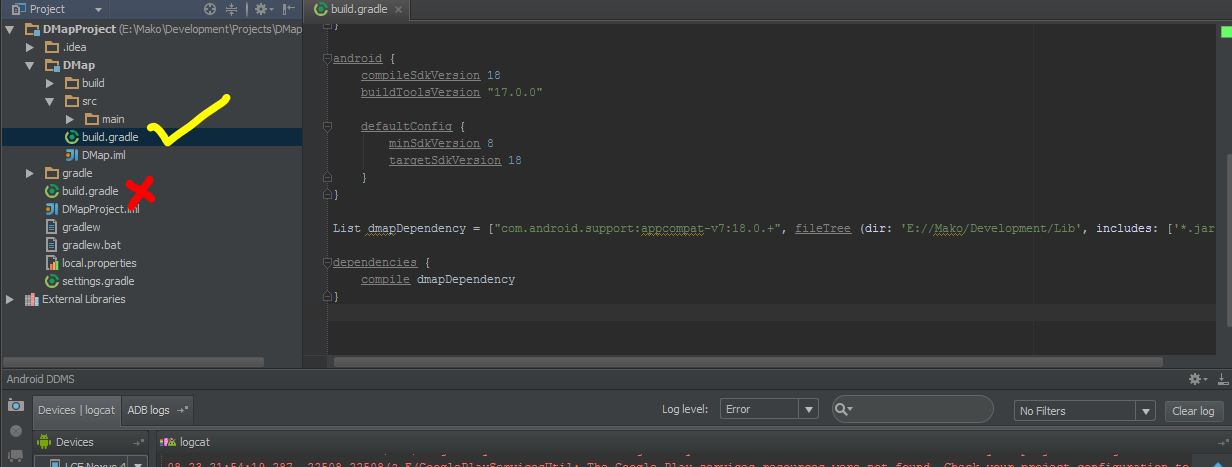
翻译：Android Studio 中文组（[大锤译](http://www.android-studio.org/index.php/team/130-team-dachui)）

在Android Studio（AS）的开发中，遇到引入外部类库或第三方jar包的情况，确实是件很恼人的事情。因为AS是基于Gradle构建，所以和传统的导入方式有些不一样。

笔者也曾为此头疼过好一阵，现在终于有了解决办法，拿出来和大家分享一下。

这里以导入Google Services library为例，具体操作步骤如下。

1. 打开你的buid.gradle文件。应该用哪一个呢？Project中的，还是Module中的？（如图所示，Module中的）

[](http://www.android-studio.org/images/guide/which-build-gradle.jpg)

2 . 将google-play-services.jar这个jar包拷到指定的位置。原来可能是在这个目录下：$ANDROID-SDK/extras/google/google\_play\_services/libproject/google-play-services\_lib/libs ，比如现在我把它放到了$GOOGLE\_PLAY\_LIB目录下。

3 . 如果你已经有了别的dependencies(如Android Studio Library),我们把它加到list当中去,也就是将你需要编译的部分，从dependencies中移到list里。

例如先前长这样：

List myDependencies = ["com.android.support:appcompat-v7:18.0.+"]  
dependencies {  
        //moved into List above, so there'll be nothing here.  
}

4.  加入Google Play Library之后：

List myDependencies = ["com.android.support:appcompat-v7:18.0.+",   
fileTree (dir: '$GOOGLE\_PLAY\_LIB', includes: ['\*.jar'])]

PS:不要忘了把$GOOGLE\_PLAY\_LIB 改成你本地的目录，如 E://Android/Development/Lib

5.  修改dependencies里面的内容：

List myDependencies = ["com.android.support:appcompat-v7:18.0.+",   
fileTree (dir: 'E://Android/Development/Lib', includes: ['\*.jar'])]  
dependencies {  
    compile myDependencies  
}

6.  然后就是耐心等待项目compile和make了。^\_^

笔者成功的测试过Google Android Map v2,如果你有遇到问题，欢迎留言。

=======以下是另一个网友留言，有更简便一些，不过得注意路径的问题========

只需要从SDK中下载 Google Repository & Google Play Services，然后在build.gradle文件中申明，如下所示：

dependencies {  
compile('com.android.support:appcompat-v7:18.0.+')  
compile('com.android.support:support-v4:13.0.0')  
compile ('com.google.android.gms:play-services:3.2.25')  
}

再或者，直接：compile files('libs/google-play-services.jar')

## [Gradle使用手册（三）：构建任务](http://www.android-studio.org/index.php/docs/guide/136-gradle-3)

原文地址：http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Using-sourceCompatibility-1.7  
翻译：Android Studio 中文组（[大锤译](http://www.android-studio.org/index.php/team/130-team-dachui" \t "http://www.android-studio.org/index.php/docs/guide/_blank)）

接续： [Gradle（一）](http://www.android-studio.org/index.php/docs/guide/134-gradle" \t "http://www.android-studio.org/index.php/docs/guide/_blank) | [Gradle（二）](http://www.android-studio.org/index.php/docs/guide/135-gradle-2" \t "http://www.android-studio.org/index.php/docs/guide/_blank)

##### ****通用任务****

        将一个plugin运用到build file中时，会自动创建一系列的构建任务（build task）去运行。Java plugin和Android Plugin也都会如此。

我们对于任务的约定有以下四个：

        assemble任务，汇集所有项目输出

        check任务，运行所有校验

        build任务，既汇集又校验

        clean任务，清除所有项目输出

        assemble, check and build任务自己本身不做任何事情，它们只是plugin锚点，真正任务的是由plugin来添加执行。

这样做的好处是，不管你在什么项目中，你都可以调用同样的命令来执行。

        通过命令行，你可以得到更高级别的任务，命令如下：

gradle tasks

列出当前运行的所有任务，以及查看他们之间的依赖关系：

gradle tasks --all

注: Gradle会自动地检测一个任务中申明的输入和输出。当重复执行两次build任务时，Gradle会报告当前所有任务是UP-TO-DATE的状态。

##### ****Java项目的任务****

        Java plugin会创建两个任务，分别挂到锚任务中，如下：

        assemble  
                jar  This task creates the output.  
        check  
                test This task runs the tests.

jar任务是编译执行Java源代码。

test任务是运行unit test

通常，java项目中的任务只会用到assemble和check这两个,更多的其他task详见此处。

##### ****Android 任务****

   Android的任务比通用的四大任务多了“connectedCheck”和“deviceCheck”，这是想要让项目忽视设备是否连接，正常执行check任务。

        assemble任务，  汇集所有项目输出

        check任务，运行所有校验

        connectedCheck任务，运行所有需要链接设备或模拟器的校验, 并行运行

        deviceCheck任务，运行调用远程设备的校验，运用于CI Servers

        build任务，既汇集又校验

        clean任务，清除所有项目输出

注：build任务不依赖与deviceCheck或connectedCheck

一个安卓的项目至少有两个输出，一是debug apk，二是release apk.这两个输出都有自己对应的锚任务，来实现它们各自的构建调用assemble任务时会同时调用assembleDebug和assembleRelease来保证有两个输出。

assemble  
        - assembleDebug  
        - assemblRelease

        Tip: Gradle 支持Camel命名方式的简写，比如在输命令行时，可以用aR代替assembleRelease，如果没有其他别的任务也是aR简写的话：

gradle  aR =  gradle assembleRelease

Check任务也有它们自己的依赖关系：

        check

                ---lint (目前还没实现，汗一个)

        connectedCheck

                ---connectedInstrumentTest  
                ---connectedUiAutomatorTest (这个也还没实现……)

        deviceCheck

                依赖于任务创建时，其他插件实现测试的扩展点

最后, 为了能够安装卸载，Android plugin 为所有的build类型（debug,release,test）都创建了install/uninstall 任务，但需要signing。