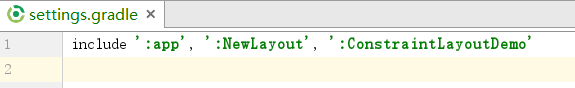
# 第一章 Android开发 Gradle基础

## 1.1 Android中的Gradle构建文件

创建一个新的Android项目时，会生成3个Gradle文件：settings.gradle、build.gradle、app/build.gradle

### settings.gradle

配置了项目中包含的模块，如下：



### 顶层的build.gradle

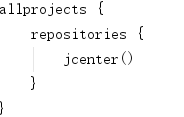


用于为所有的模块设置通用的配置

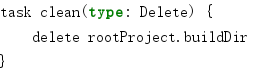


buildscript翻译过来为“构建脚本”，这个代码块就是用于配置从哪下载构建脚本插件，从图可知是从jcenter仓库中下载，下载的是gradle插件，这个gradle插件就是“构建脚本”，用于构建Android项目。

jcenter即Bintray Jcenter Artifactory库。也支持其他的库（特别是mavenCentral()，默认的Maven仓库），但是Jcenter现在已经成默认的了。Jcenter中的内容都使用HTTPS连接并通过CDN来提供，同时也更快速。



allprojects配置了所有模块都默认使用jcenter仓库来下载Java依赖库



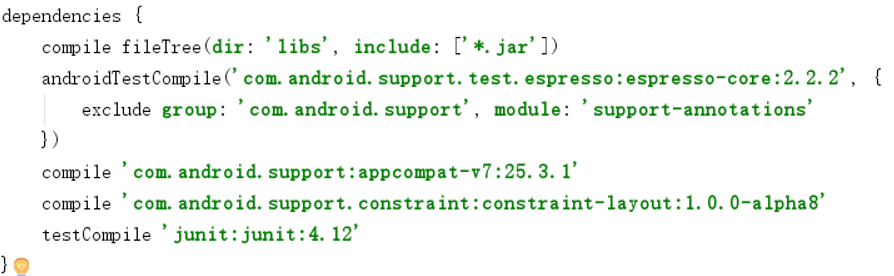
Gradle允许自定义task，上面代码就是一个自定义的task，type: Delete表示这个task是Gradle内置的Delete task的一个实例，在这个task 中，它会把项目根目录下的build目录删除。

### 模块里的build.gradle





通过Gradle的apply功能，将Android插件添加到构建系统中，从而能够使用Android的领域特定语言（Domain Specific Language, DSL）配置。



dependencies用于配置依赖，compile fileTree用于依赖一整个目录树里的文件，dir指定了依赖的目录，include为过滤器，表示只依赖\*.jar文件。

testComile 用于指定测试的依赖，也就是说这些依赖只能在src/AndroidTest/java或者src/test/java，这意味着打包apk时这个测试的依赖jar不会打包到apk中

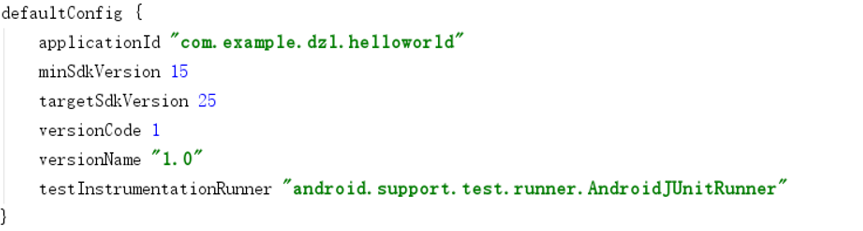
## 1.2 配置SDK版本以及其他的默认值

顶层的build.gradle通过buildscript块将Gradle的Android插件添加到了项目中，模块中的build.gradle通过apply应用了这个插件，因此可以将android块添加到Gradle的DSL中。在android块里面，你可以指定几个项目属性，如下：



android块是Android DSL的入口,compileSdkVersion和buildToolVersion指定编译SDK版本和构建工具版本，这些版本应该指定为最新的版本，因为它们是身后兼容的，并且包含了之前版本的漏洞修复。

defaultConfig块：



applicationId 应用的唯一标识。应用一但上线，这个id就不能修改。一修改就相当于是另一个应用了。

minSdkVersion 应用程序支持的最小Android SDK版本，小于这个版本的设备在访问Google Play商店时将不会看到这个应用程序。

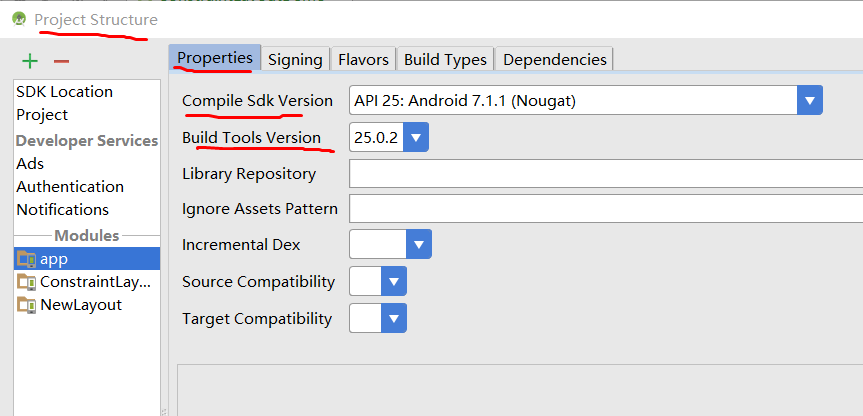
targetSdkVersion 应用程序的目标版本号。如果没有指定为最新版本，AS会给出警告，但是你可以作用任何的版本。

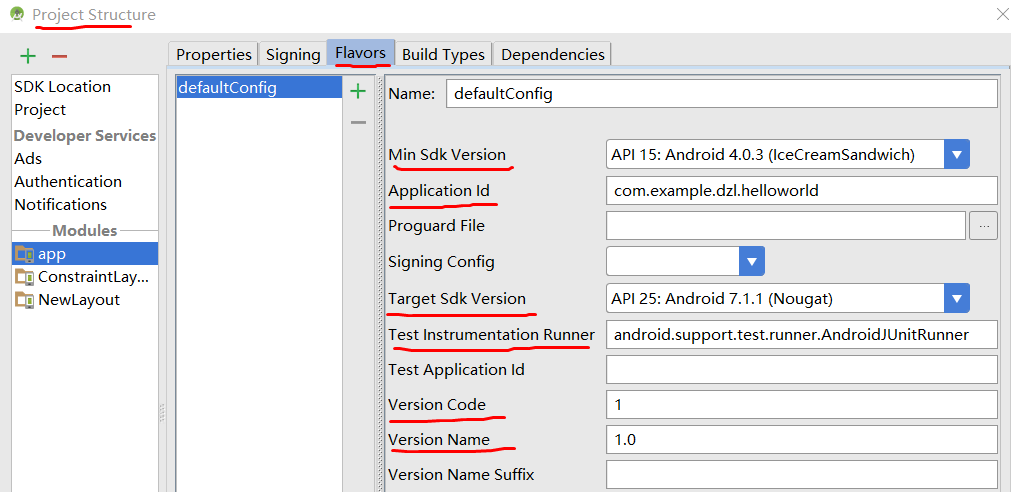
versionCode 应用程序的版本号，用于升级

versionName 版本号的名字，用于显示给用户看，通常的格式为：<major>.<minor>.<version>，就像大多数的项目一样。

testInstrumentationRunner

这些属性在AS的项目结构窗口中有图形化的显示，如下：





defaultConfig块的文档，以及其他元素的DSL，都可以在DSL参考中找到：<http://bit.ly/gradle-dsl> 或 <http://google.github.io/android-gradle-dsl/current/index.html>

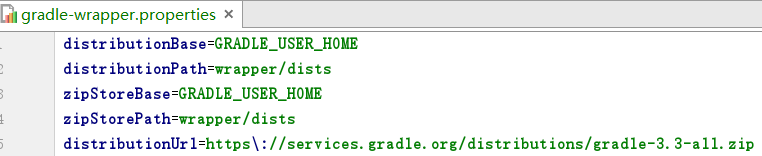
## 1.3 在命令行执行Gradle构建

在命令行执行Gradle构建，使用的是AS提供的Gradle wrapper或者安装Gradle直接运行。你不需要为了构建Android项目而安装Gradle，AS包含了一个Gradle的发行版本（以一个插件的形式）并包含了专门的功能来支持它。

“Gradle wrapper”引用gradlew脚本和gradlew.bat脚本，bradlew给UNIX使用，gradlew.bat给Windows使用，这两个项目在项目的根目录中，结尾的“w”代表“wrapper”。

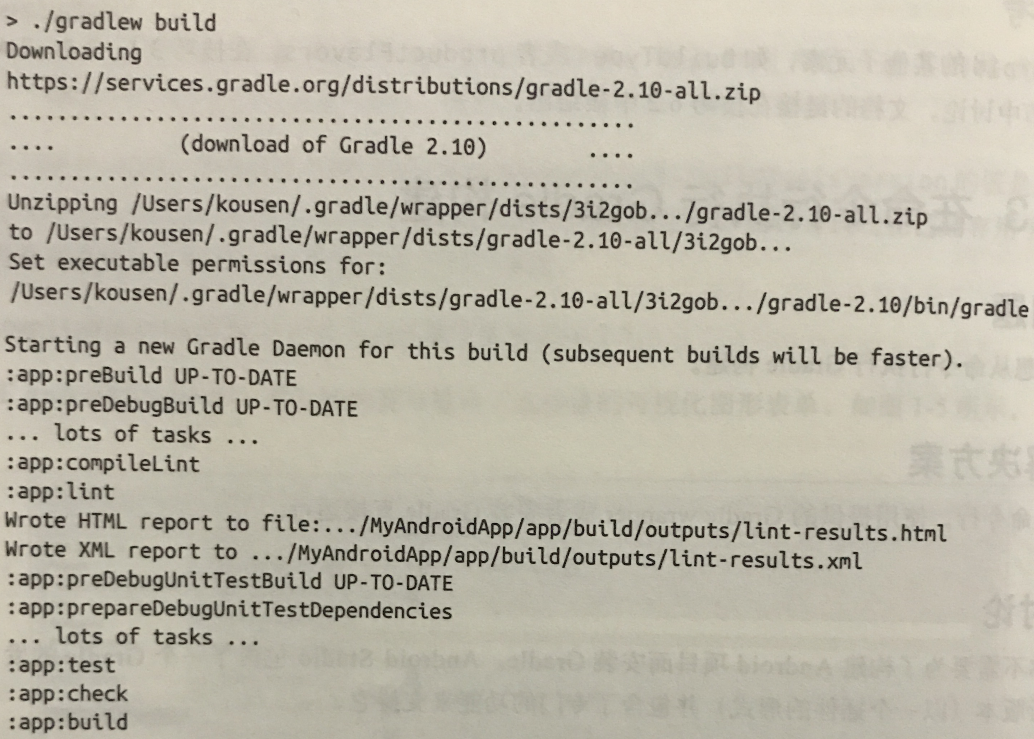
Gradle Wrapper的第一个目的就是允许客户端无须安装Gradle即可运行Gradle脚本（其实所谓的无需安装只是说不需要我们手动去安装，AS会根据wrapper中的配置去下载gradle并安装）。wrapper使用了项目根目录下的gradle/wrapper下的gradle-wrapper.jar和gradle.wrapper.properties文件来启动进程。

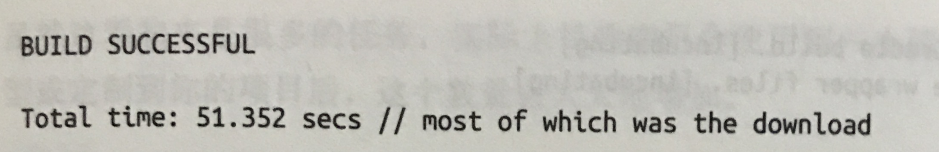
gradle.wrapper.properties文件内容如下：



distributionUrl显示了wrapper会下载并安装Gradle 3.3(可以修改为任意可用的版本),下载保存到zipStoreBase下的zipStorePath目录中，这样3.3版本只需要下载一次，以后如果是相同的版本就不需要再下载了。变量GRADLE\_USER\_HOME即电脑上的用户目录，如：C:\Users\EvenDai

假设把上面的版本修改为2.10，然后在命令行执行gradlew build，输出结果如下：





从上面截图可知，gradlew build是执行了一个脚本命令，并不是Gradle任务哦，这个脚本命令首先要下载gradle 2.10，然后解压，然后开启了一个Gradle的后台进程，然后开始执行Gradle tasks（只有开启了Gradle进程，才能执行Gradle tasks）。

可以在命令行运行任何支持的Gradle tasks，包括你自定义的tasks。编译的代码在app/build目录下，生成的apk(Android package)在app/build/apk目录下。

gradlew tasks 命令显示了构建中可以使用的任务，这个输出很长，可查看“可用任务.txt”

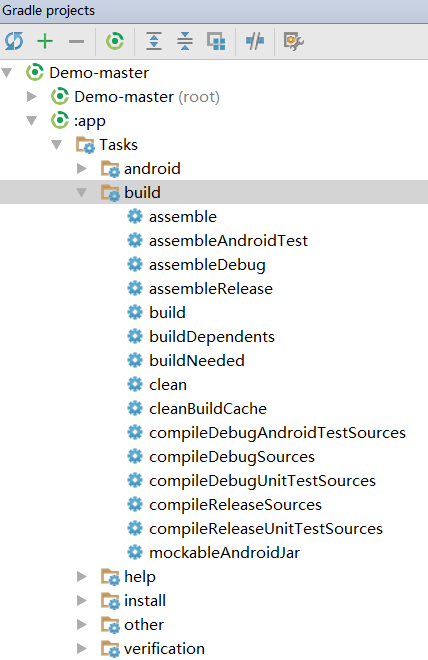
gradlew tasks –all 作了显示可用任务外，也显示每个任务的依赖。

可以使用任务的名字的缩写，只要这个缩写在所有任务中是唯一的即可，如：gradlew anDep表示gradlew androidDependencies。

如果你的构建文件不叫build.gradle，可以使用 –b 标志来指定一个构建文件名：gradlew –b app.gradle

## 1.4 在AndroidStudio中执行Gradle构建

当你创建一个Android项目时，AS产生了用于多项目构建的Gradle构建文件，IDE还提供了一个Gradle视图，其中整理了所有任务，如下：



这里的gradle任务被分类，双击任务即可执行。

运行项目，从Gradle console中查看可知，其实运行本质上只是Gradle的一个前端。任何执行，从构建到测试，再到部署，实际上就是从命令运行Gradle任务。

## 1.5 添加Java库的依赖

Gradle支持几种不同的方式来定义依赖。最常用的方式就是使用引号，并通过冒号分隔开组名、名字和版本号。（Gradle使用Groovy，其即支持单引号字符串，也支持双引号字符串。双引号允许修改，或者变量替换，其他方面是完全一样的。

每个依赖都关联了一个configuration。Android项目包含compile、runtime、testCompile、testRuntime等配置。插件可以添加额外的配置，你也可以定义你自己的。

依赖的完整语法：

testCompile group: ‘junit’, name: ‘junit’, version: ‘4.12’

依赖快捷语法：

testCompile ‘junit:junit:4.12’ 推荐

testCompile ‘junit:junit:4.+’ 这是合法的，但是不推荐使用，指定一个带加号的版本号，这是告诉Gradle任何高于4.0版本的Junit在编译项目的测试是是必须的。虽然这是可以工作的，但是这让构建的可确定性降低，并因此变得更不可重现。显示的版本号也在以后的版本中特定的API的变化中起到保护作用。

如果你想要添加一系列的文件配置中，可以使用files或者fileTree语法，如下：

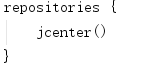
dependencies {

compile files(‘libs/a.jar’, ‘libs/b.jar’)

compile fileTree(dir: ‘libs’, include: ‘\*.jar’)

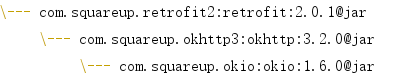
}

compile 'com.android.support:design:25.0.1' ，像这样的依赖，Gradle需要知道去哪里搜索并解析依赖，这通过一个repositories块实现，在项目根目录的build.gradle中指定了这个仓库，如下：



**传递依赖**

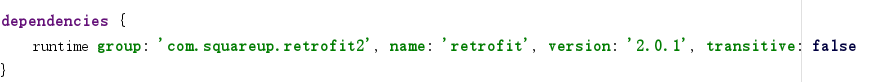
传递依赖，即一个库本身又依赖于另外的库（可能另外的库又依赖另外的库）。比如依赖compile 'com.squareup.retrofit2:retrofit:2.0.1'，这个库会以传递依赖的形式添加okhttp和okio两个库，这两个库在dependencies中是看不到的。如果想查看传递依赖有哪些库，可执行gradlew androidDependencies（注：双击对应的命令看不到相同的结果）：



从这个图中知，retrofit依赖于okhttp，okhttp又依赖于okio。通过查看完整控制台可知：v7依赖于v4和support-vector-drawable和animated-vector-drawable

传递依赖的好处是Gradle很擅长解决依赖之间的版本问题。同时，Gradle也提供了语法来包含或者不包含指定的库。

Gradle默认遵循传递依赖，如果你想要针对一个特定的库关掉这个功能，使用transitive标志，如下：



transitive为false表示阻止传递依赖，则你自己需要手动添加任何需要的传递依赖。

如果你想要依赖一个模块的jar，不引入任何额外的依赖，可以这样：

dependencies {

compile ‘org.codehaus.groovy:groovy-all:2.4.4@jar’ (快捷语法， @符号即表示 ext: ‘jar’)

compile group: ‘ogr.codehaus.groovy’, name: ‘groovy-all’, version: ‘2.4.4’, ext: ‘jar’ (完整语法)

}

你也可以从dependencies块中排除一个传递依赖，如下：

dependencies {

androidTestCompile(‘org.spockframework:spock-core:1.0-groovy-2.4’) {

exclude group: ‘org.codehaus.groovy’

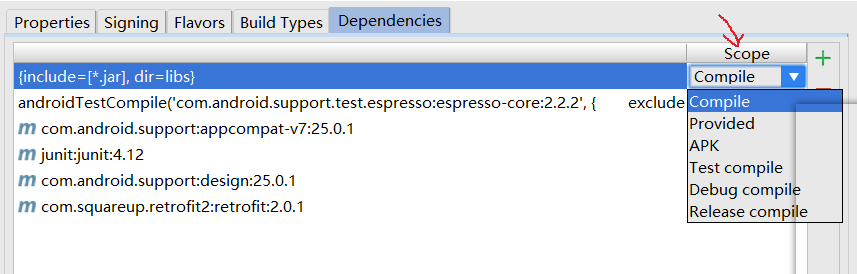
exclude group: ‘junit’

}

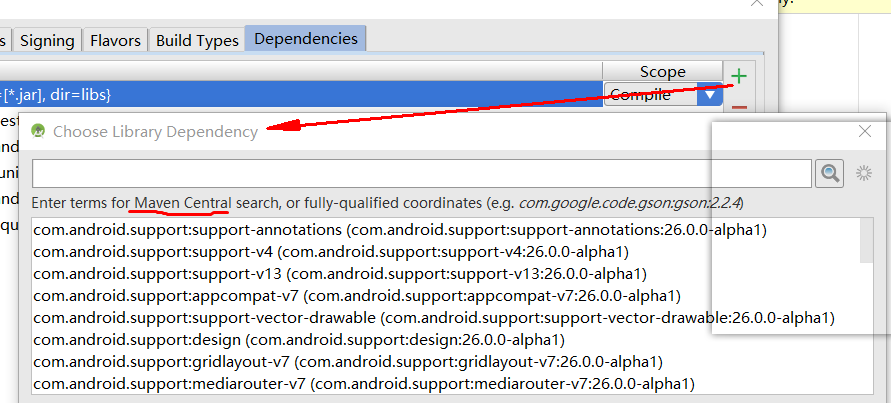
}

这个例子中，spock-core项目并没有包含Groovy依赖和Junit库，须要用其他的方式引入。

## 1.6 使用AS添加库的依赖



Scope列允许你指定哪处配置对依赖有需要。



这里是在Maven中央仓库中搜索，默认显示了所有可选的支持库和Google Play服务。

## 1.7 配置仓库

jcenter() 代表Bintray JCenter仓库，位于：<https://jcenter.bintray.com>

mavenCentral() 代表公共的Maven中央仓库，位于：<http://repo1.maven.org/maven2>

mavenLocal() 代表引用本地的Maven缓存

Maven仓库不止一个，任何的一个Maven仓库都可以使用maven并且跟随一个url块来添加到默认的仓库列表中去，如下：

repositories {

maven {

url ‘http://repo.spring.io/milestone’

}

}

通过密码保护的仓库可以使用一个credentials块，如：

repositories {

maven {

credentials {

username ‘username’

password ‘password’

}

url ‘http://repo.mycompany.com/maven2’

}

}