Gradle：现代高效的 java 构建工具

作者：jianzh5
原文链接：https://ld246.com/article/1602140358387
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

相信使用Java的同学都用过Maven，这是一个非常经典好用的项目构建工具。但是如果你经常使用Maven，可能会发现Maven有一些地方用的让人不太舒服：

- 一来Maven的配置文件是XML格式的，假如你的项目依赖的包比较多，那么XML文件就会变得非非常长；
- 二来XML文件不太灵活，假如你需要在构建过程中添加一些自定义逻辑，搞起来非常麻烦；
- 第三就是Maven非常的稳定，但是相对的就是对新版java支持不足，哪怕就是为了编译java11，也要更新内置的Maven插件。

如果你对Maven的这些缺点也有所感触，准备尝试其他的构建工具，那么你可以试试gradle，这是一全新的java构建工具，解决了Maven的一些痛点。

安装gradle

最传统的安装方法就是去gradle官网下载二进制包，解压，然后将路径添加到环境变量中。如果你没么其他需求，可以使用这种安装方式。但是，gradle是一个非常新潮的项目，每隔几个月就会发布一新版本，这种方式可能跟不上gradle的更新速度。

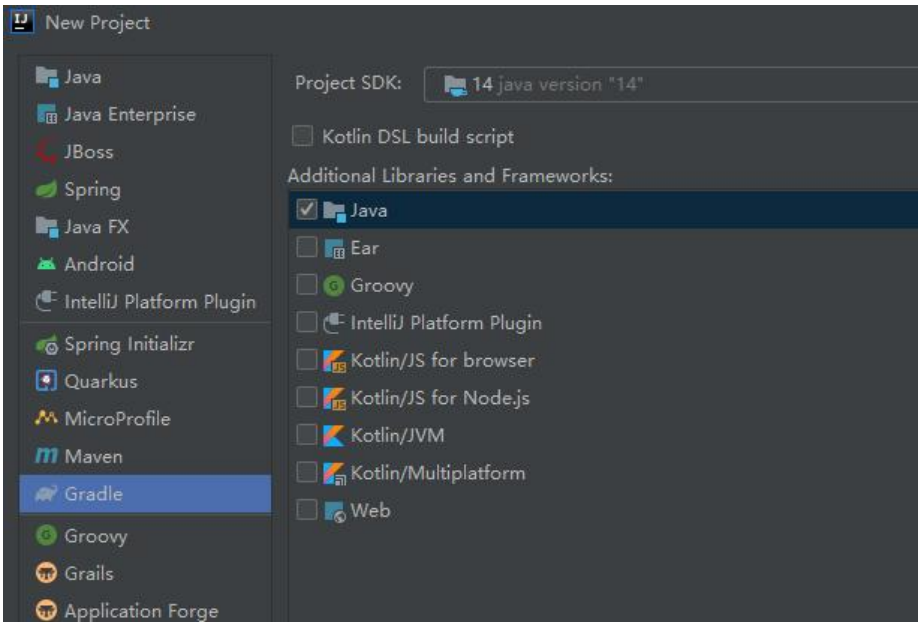
所以我更加推荐使用包管理器来安装gradle。如果你使用linux系统，那么不必多说。如果你使用Windows系统，我推荐使用scoop包管理器来安装gradle。它安装方便，而且使用SHIM目录来管理环境量，在各种工具中配置gradle也很方便。

当然，如果你完全不喜欢安装这么多乱七八糟的东西，那也可以使用gradle。gradle提供了一个名为gadle wrapper的工具，可以在没有安装gradle的情况下使用gradle。好吧，其实它就是个脚本文件，你运行wrapper脚本的时候，如果脚本发现你电脑里没有gradle，就会自动替你下载安装一个。现在至还出现了Maven wrapper，也是个脚本文件，可以自动安装Maven。

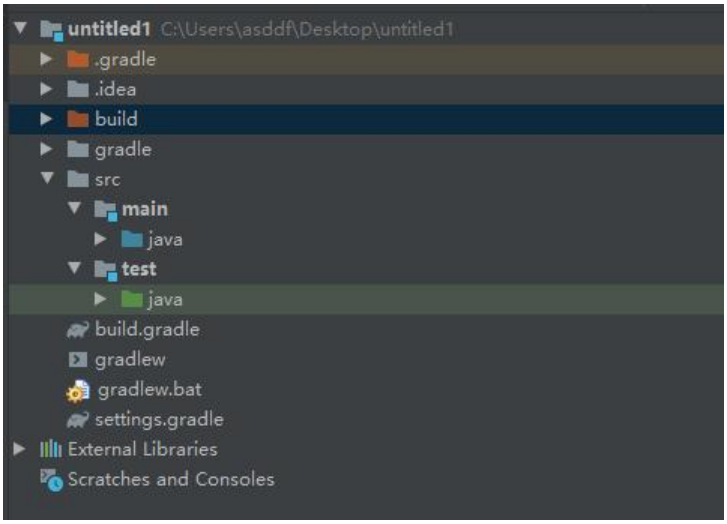
之前相信一些朋友听说过gradle，然后尝试使用它，结果因为速度太慢，最后放弃了。之前我也因为gadle的速度，放弃了它一段时间。不过现在使用gradle的话会方便很多。gradle官方在中国开设了，DN，使用gradle wrapper的时候下载速度非常快。可以说现在是一个学习使用gradle的好时候。

使用gradle wrapper

这里我使用的IDEA来创建和使用gradle项目。



IDEA默认就会使用gradle wrapper来创建项目，所以无需安装gradle也可以正常运行。这时候项目构应该类似下图所示，使用Maven的同学应该比较熟悉，因为这和Maven的项目结构几乎完全一致。radle文件夹和gradlew那几个文件就是gradle wrapper的文件，而.gradle后缀名的文件正是gradle配置文件，对应于Maven的pom.xml。



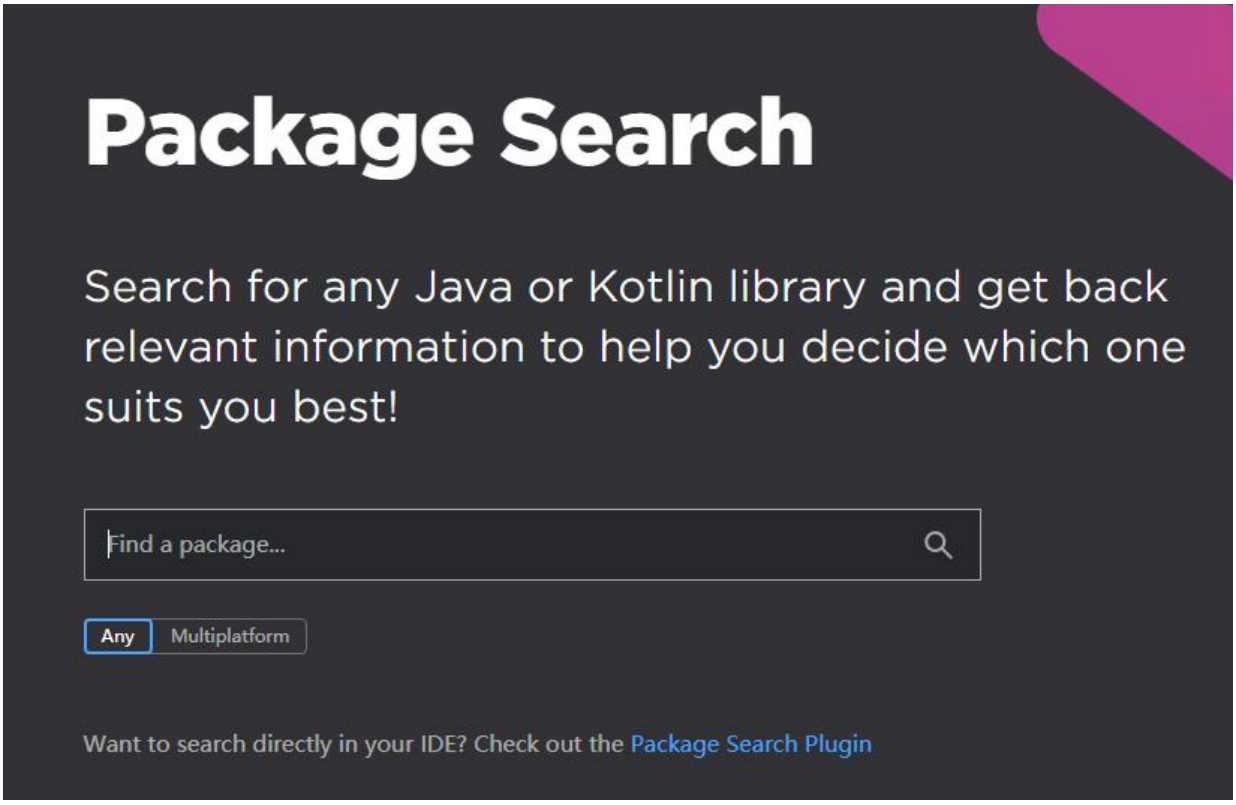
gradle wrapper的优点之一就是可以自定义下载的gradle的版本，如果是团队协作的话，这个功能就常方便，简单设置即可统一团队的构建工具版本。这里我就设定成目前最新的gradle 6.4.默认下载安的是bin版，仅包含二进制。如果你使用IDEA的话，它会推荐下载all版，包含源代码，这样IDEA就可分析源代码，提供更加精确的gradle脚本支持。

依赖管理

下面来看看gradle的依赖管理功能，这也算是我们使用构建工具的主要目的之一了。这点也是gradle较maven的优势之一了。相较于maven一大串的XML配置，gradle的依赖项仅需一行。

```
dependencies {
    testImplementation 'junit:junit:4.13'
    implementation 'com.google.code.gson:gson:2.8.6'
}
```

这里推荐一下Jetbrains的package search网站，是寻找maven和gradle依赖包的最佳网站，可以非轻松的搜索和使用依赖项。



gradle依赖的粒度控制相较于Maven也更加精细，maven只有compile、provided、test、runtime种scope，而gradle有以下几种scope：

- implementation，默认的scope。implementation的作用域会让依赖在编译和运行时均包含在内但是不会暴露在类库使用者的编译时。举例，如果我们的类库包含了gson，那么其他人使用我们的类时，编译时不会出现gson的依赖。
- api，和implementation类似，都是编译和运行时都可见的依赖。但是api允许我们将自己类库的赖暴露给我们类库的使用者。
- compileOnly和runtimeOnly，这两种顾名思义，一种只在编译时可见，一种只在运行时可见。而rntimeOnly和Maven的provided比较接近。
- testImplementation，这种依赖在测试编译时和运行时可见，类似于Maven的test作用域。
- testCompileOnly和testRuntimeOnly，这两种类似于compileOnly和runtimeOnly，但是作用于试编译时和运行时。

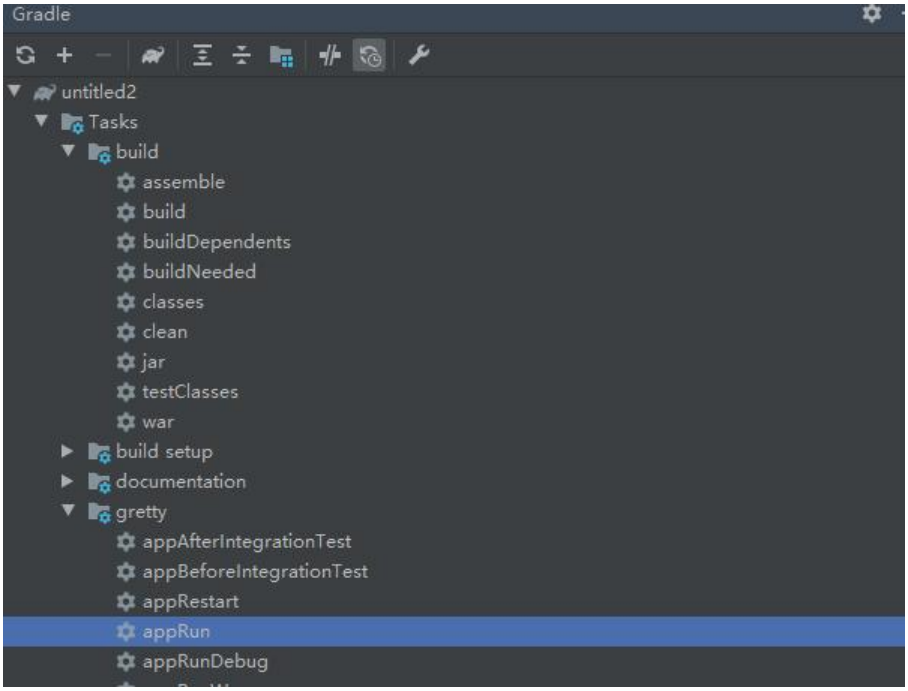
通过简短精悍的依赖配置和多种多样的作用与选择，Gradle可以为我们提供比Maven更加优秀的依管理功能。

gradle的任务和插件

gradle的配置文件是一个groovy脚本文件，在其中我们可以以编程方式自定义一些构建任务。因为用了编程方式，所以这带给了我们极大的灵活性和便捷性。打个比方，现在有个需求，要在打包出jar时候顺便看看jar文件的大小。在gradle中仅需在构建脚本中编写几行代码即可。而在Maven中则需编写Maven插件，复杂程度完全不在一个水平。

当然，Maven发展到现在，已经存在了大量的插件，提供了各式各样的功能可以使用。但是在灵活性面还是无法和Gradle相比。而且Gradle也有插件功能，现在发展也十分迅猛，存在了大量非常好用的件，例如gretty插件。gretty原来是社区插件，后来被官方吸收为官方插件，可以在Tomcat和jetty务器上运行web项目，比Maven的相关插件功能都强大。

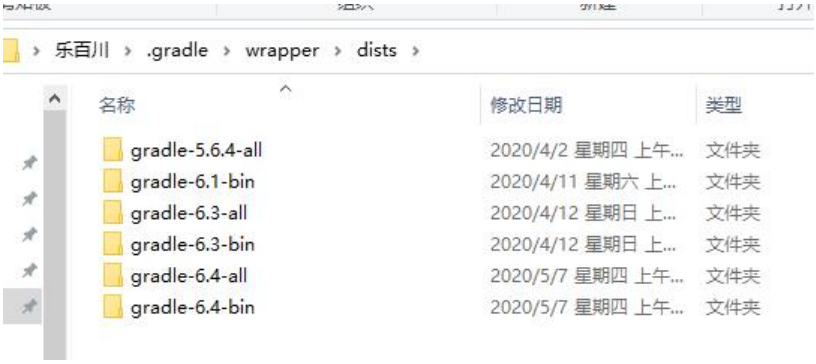
虽然gradle可以非常灵活的编写自定义脚本任务，但是其实一般情况下我们不需要编写构建脚本，利现有的插件和任务即可完成相关功能。在IDEA里，也可以轻松的查看当前gradle项目中有多少任务，本任务如build、test等Maven和Gradle都是相通的。



配置镜像

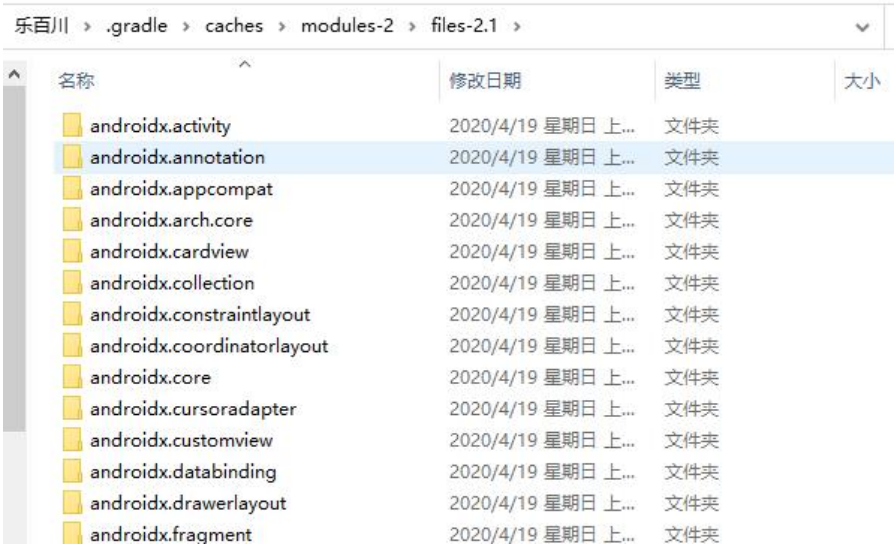
Maven官方仓库的下载速度非常慢，所以一般我们要配置国内的镜像源。gradle在这方面和Maven全兼容，因此只需稍微配置一下镜像源，即可使用Maven的镜像。如果你用gradle构建过项目，应该可以在用户目录的.gradle文件夹下看到gradle的相关配置和缓存。

之前wrapper下载的gradle也存放在该文件夹下，位置是wrapper/dists。



而依赖的本地缓存在caches\modules-2\files-2.1文件夹下。目录结构和Maven的本地缓存类似，都

包名+版本号的方式，但是gradle的目录结构最后一层和Maven不同，这导致它们无法共用本地缓存。



言归正传，在gradle中配置下载镜像需要在.gradle文件夹中直接新建一个init.gradle初始化脚本，脚文件内容如下。这样一来，gradle下载镜像的时候就会使用这里配置的镜像源下载，速度会快很多。加上gradle wrapper在中国设置了CDN，现在使用gradle的速度应该会很快。

```
allprojects {
  repositories {
      maven {
         url "https://maven.aliyun.com/repository/public"
      }     maven {         url "https://maven.aliyun.com/repository/jcenter"
      }     maven {         url "https://maven.aliyun.com/repository/spring"
      }     maven {         url "https://maven.aliyun.com/repository/spring-plugin"
      }     maven {         url "https://maven.aliyun.com/repository/gradle-plugin"
      }     maven {         url "https://maven.aliyun.com/repository/google"
      }     maven {         url "https://maven.aliyun.com/repository/grails-core"
      }     maven {         url "https://maven.aliyun.com/repository/apache-snapshots"
      }  }}
```

当然，如果你有代理的话，其实我推荐你直接为gradle设置全局代理。因为gradle脚本实在是太灵活，有些脚本中可能依赖了github或者其他地方的远程脚本。这时候上面设置的下载镜像源就不管用了。

所以有条件还是干脆直接使用全局代理比较好。设置方式很简单，在.gradle文件夹中新建gradle.properties文件，内容如下。中间几行即是设置代理的配置项。当然其他几行我也建议你设置一下，把gradle运行时的文件编码设置为UTF8，增加跨平台兼容性。

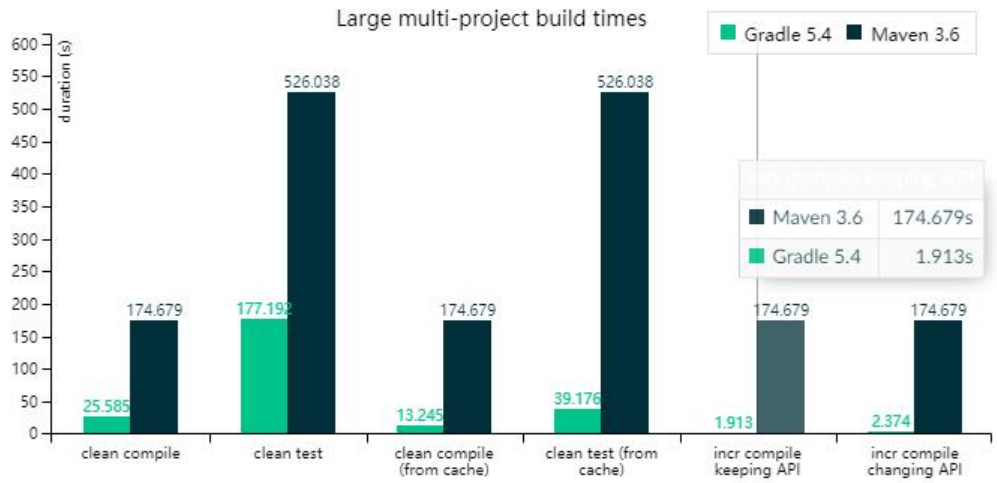
```
org.gradle.jvmargs=-Xmx4g -XX:MaxPermSize=512m -XX:+HeapDumpOnOutOfMemoryError
-Dfile.encoding=UTF-8
systemProp.http.proxyHost=127.0.0.1
systemProp.http.proxyPort=10800
systemProp.https.proxyHost=127.0.0.1
systemProp.https.proxyPort=10800
systemProp.file.encoding=UTF-8
org.gradle.warning.mode=all
```

为什么使用gradle?

看到这里，你应该对gradle有了基本的了解， 也可以将其用于你的项目之中。但是如果你Maven已

非常熟悉了，可能不太愿意使用gradle，因为貌似没有必要。但是既然gradle出现了，就说明有很多对Maven还是有一定的意见。因此在这里我来总结一下gradle相比maven的优势。

首先第一点也就是最重要的一点就是**速度**。gradle使用构建缓存、守护进程等方式提高编译速度。结就是gradle的编译速度要远超maven，平均编译速度比Maven快好几倍，而且项目越大，这个差距越明显。



第二点就是灵活性，gradle要比Maven灵活太多，虽然有时候灵活并不是一件好事情。但是大部分情下，灵活一点可以极大的方便我们。Maven死板的XML文件方式做起事情来非常麻烦。很多Maven目都通过执行外部脚本的方式来完成一些需要灵活性的工作。而在gradle中配置文件就是构建脚本，建脚本就是编程语言（groovy编程语言），完全可以自给自足，无需外部脚本。

第三点就是gradle DSL带来的简洁性。完成同样的功能，gradle脚本的长度要远远短于maven配置件的长度。虽然很多人都说XML维护起来不麻烦，但是我觉得，维护一个光是依赖就有几百行的XML件，不见得就比gradle脚本简单。

也许是因为我上面说的原因，也许有其他原因，不得不承认的一件事情就是gradle作为一个新兴的工已经有了广泛的应用。spring等项目已经从Maven切换到了gradle。开发安卓程序也只支持gradle了因此不管是否现在需要将项目从maven切换到gradle，但是至少学习gradle是一件必要的事情。