gradlew.properties 中的参数:
# 为 gradle 守护进程配置 jvmargs，可以用于调整 gradle 构建项目时的可用内存大小等，这里我们设置了 java 堆大小为 2G，当你在 build 的时候 IDE 报了类似的错误。...aapt.exe'' finished with non-zero exit value 1，你也可以使用这个参数去调整 java 堆大小，去避免这个错误。
org.gradle.jvmargs=-Xmx2048m

# 让 Gradle 以并行模式去执行 task (当你稍微了解 Gradle 之后，你就会知道 Gradle 脚本是由任务(task)组成的。而我们使用 Gradle 去构建项目的时候呢，就相当于要求 Gradle 去执行若干个构建任务)
org.gradle.parallel=true

# 激活新的 incubating mode。让 Gradle 在 build 大型多项目的时候更快!
org.gradle.configureondemand=true

# 当设置了它的时候，你的 gradle 将会以进程守护的方式运行（简而言之就是一直在后台运行着），这就意味着，gradle 进程的资源将会一直存在，随时待命你的 build task！我们都知道，创建进程是一个很耗时的过程。
setting.org.gradle.daemon=true

当你配置了这个参数的参数的时候，Gradle 会以相应数目的线程去构建项目，当然速度会嗖嗖的！但这个参数我没有用到，因为电脑实在太渣了！并不敢弄多线程!
org.gradle.workers.max=workerCount

未优化的 command line
(生成 build report，在 pathToProject/build/report/profile 路径下可查看)
gradlew app:assembleDebug --profile

优化的 command line
实际上就是将 gradle.properties 中的优化选项作为参数写进了 command line 中。可我们可以看出，在 console 中去 build，无论如何都比在 IDE 中 build 快。 而且不只一点点。
gradlew app:assembleDebug -x lint --configure-on-demand --daemon --parallel --profile

gradle 不去编译项目依赖
gradlew app:assembleDebug --daemon --parallel --no-rebuild --profile