前段时间忽然发现自己对于Android studio的Gradle打包并不了解，这篇博客参考网上众多教程，为大家详细介绍Gradle。

Gradle简介

We would like to introduce Gradle to you, a build system that we think is a quantum leap for build technology in the Java (JVM) world. Gradle provides:

1. A very flexible general purpose build tool like Ant.
2. Switchable, build-by-convention frameworks a la Maven. But we never lock you in!
3. Very powerful support for multi-project builds.
4. Very powerful dependency management (based on Apache Ivy).
5. Full support for your existing Maven or Ivy repository infrastructure.
6. Support for transitive dependency management without the need for remote repositories or pom.xml and ivy.xml files.
7. Ant tasks and builds as first class citizens.
8. Groovy build scripts.
9. A rich domain model for describing your build。

上面这段是grade官方介绍的grade的特性：

- 一个像 Ant 一样的灵活的构建工具
- 一种可切换的, 像 maven 一样的基于合约构建的框架
- 支持强大的多工程构建
- 支持强大的依赖管理(基于 ApacheIvy)
- 支持已有的 maven 和 ivy 仓库
- 支持传递性依赖管理, 而不需要远程仓库或者 pom.xml 或者 ivy 配置文件
- 优先支持 Ant 式的任务和构建
- 基于 groovy 的构建脚本
- 有丰富的领域模型来描述你的构建

gradle属于任务驱动型构建工具，它的构建过程是基于Task的。Gradle是以 Groovy 语言为基础，面向Java应用为主。基于DSL（领域特定语言）语法的自动化构建工具。

Gradle安装与环境变量配置

网上有好多关于Windows，Linux，MAC的grade安装与配置的教程这里就不写了。

Gradle脚本构建（build.gradle）

Gradle中的所有东西都是围绕两个基本概念：项目和任务。

每个Gradle构建都是由一个或多个项目组成。一个项目代表什么，取决于你用Gradle正在做的。比如，一个项目可能代表一个库或一个网络应用。它可能代表一个由其他项目产生的一个或多个jar包打包d一个zip包。一个项目不需要代表一个事物而被构建。它可以代表一个事物而被做出来，比如部署你的应用到暂存区或产品环境。不要担心这个现在是否好像有一点含糊。Gradle的通过约定来构建的功能支持为一个项目添加一个更加具体的定义。

下面介绍一下build.gradle文件：

```
apply plugin: 'com.android.application'//说明module的类型，com.android.application为程序，com.android.library为库
```

```
android {
    compileSdkVersion 22//编译的SDK版本
    buildToolsVersion "22.0.1"//编译的Tools版本
    defaultConfig {
        //默认配置
        applicationId "com.nd.famlink"//应用程序的包名
        minSdkVersion 8//支持的最低版本
        targetSdkVersion 19//支持的目标版本
        versionCode 52//版本号
        versionName "3.0.1"//版本名
    }
    sourceSets {
        //目录指向配置
        main {
            manifest.srcFile 'AndroidManifest.xml'//指定AndroidManifest文件
            java.srcDirs = ['src']//指定source目录
            resources.srcDirs = ['src']//指定source目录
            aidl.srcDirs = ['src']//指定source目录
            renderscript.srcDirs = ['src']//指定source目录
            res.srcDirs = ['res']//指定资源目录
            assets.srcDirs = ['assets']//指定assets目录
            jniLibs.srcDirs = ['libs']//指定lib库目录
            }
            debug.setRoot('build-types/debug')//指定debug模式的路径
            release.setRoot('build-types/release')//指定release模式的路径
        }
        signingConfigs {
            //签名配置
            release {
                //发布版签名配置
                storeFile file("fk.keystore")//密钥文件路径
                storePassword "123"//密钥文件密码
                keyAlias "fk"//key别名
                keyPassword "123"//key密码
            }
            debug {
                //debug版签名配置
                storeFile file("fk.keystore")
                storePassword "123"
                keyAlias "fk"
                keyPassword "123"
            }
        }
        buildTypes {
            //build类型
            release {
                //发布
                minifyEnabled true//混淆开启
                proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-project.txt'//指定混淆规则文件
                signingConfig signingConfigs.release//设置签名信息
            }
            debug {
                //调试
                signingConfig signingConfigs.release
            }
        }
        packagingOptions {
```

```
            exclude 'META-INF/ASL2.0'//排除一些文件
            exclude 'META-INF/LICENSE'
            exclude 'META-INF/NOTICE'
            exclude 'META-INF/MANIFEST.MF'
        }
        lintOptions {
            abortOnError false//lint时候终止错误上报,防止编译的时候莫名的失败
        }
    }
    dependencies {
        compile fileTree(dir: 'libs', exclude: ['android-support*.jar'],
include: ['*.jar'])   //编译lib目录下的.jar文件
        compile project(':Easylink')//编译附加的项目
        compile 'com.nostra13.universalimageloader:universal-image-
loader:1.9.3'//编译来自Jcenter的第三方开源库
    }
```

Gradle常用命令

```
gradle --help
gradle tasks  //列出task列表
gradle asD (gradle assembleDebug) //编译debug打包
gradle asR (gradle assembleRelease) //编译release打包
gradle asD --refresh-dependencies  //强制刷新依赖
gradle asD --parallel //并行编译
gradle asD --parallel-threads 3
gradle build gradle clean
```


直接执行gradle build会生成debug包和release包如果不想要debug包可以使用gradleasR命令。

上面大家接触了一些命令如 **./gradlew -v ./gradlew clean ./gradlew build**, 这里注意是**./gradlew**, **./**代表当前目录，**gradlew**代表 gradlewrapper，意思是gradle的一层包装，大家可以理解为在这个项目本地就封装了gradle，即gradle wrapper，在**9GAG/gradle/wrapper/gralde-wrapper.properties**文件中声明了它指向的目录和版本。只要下载成功即可用grdlew wrapper的命令代替全局的gradle命令。z理解了gradlewrapper的概念，下面一些常用命令也就容易理解了。

- ./gradlew-v版本号
- ./gradlewclean清除9GAG/app目录下的build文件夹
- ./gradlewbuild检查依赖并编译打包

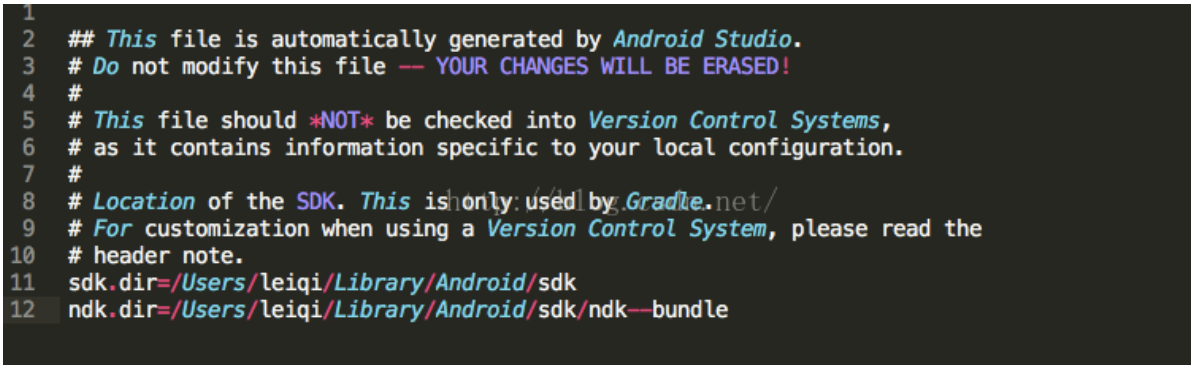
这里注意的是 **./gradlew build** 命令把debug、release环境的包都打出来，如果正式发布只需要打Release的包，该怎么办呢，下面介绍一个很有用的命令 **assemble**,如

- ./gradlewassembleDebug编译并打Debug包
- ./gradlewassembleRelease编译并打Release的包

除此之外，assemble还可以和productFlavors结合使用，具体在下一篇多渠道打包进一步解释。

- ./gradlewinstallRelease Release模式打包并安装
- ./gradlewuninstallRelease卸载Release模式包

local.properties文件：



这个文件主要是配置sdk、 ndk路径，格式必须符合要求。

settings.gradle文件

```
include ':app'
```

该文件中就仅仅只包含了一句话，在你的项目中如果有多个Model存在的时候，就可以选择包含哪些进行编译。

这个文件是全局的项目配置文件，里面主要声明一些需要加入gradle的module，如：

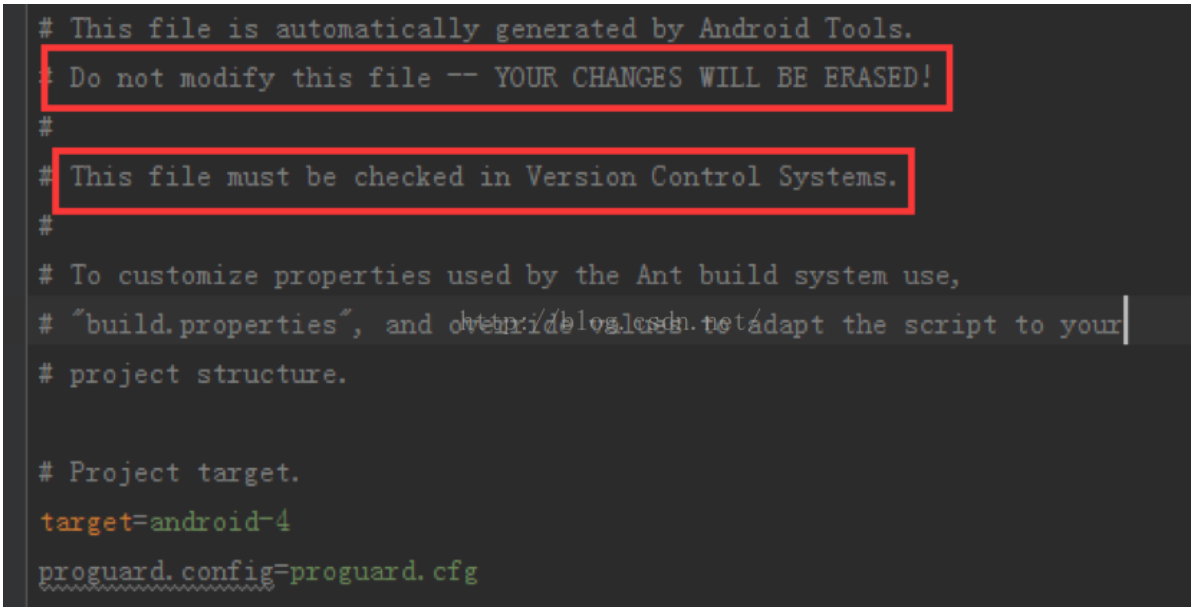
```
include':app', ':extras:ShimmerAndroid'
```

文件中的 **app**, **extras:ShimmerAndroid** 都是module，如果还有其他module都需要按照如上格式加进去。

gradle-wrapper.properties 文件

可以看到里面声明了gradle的目录与下载路径以及当前项目使用的gradle版本，这些默认的路径我们一般不会更改的，这个文件里指明的gradle版本不对也是很多导包不成功的原因之一。

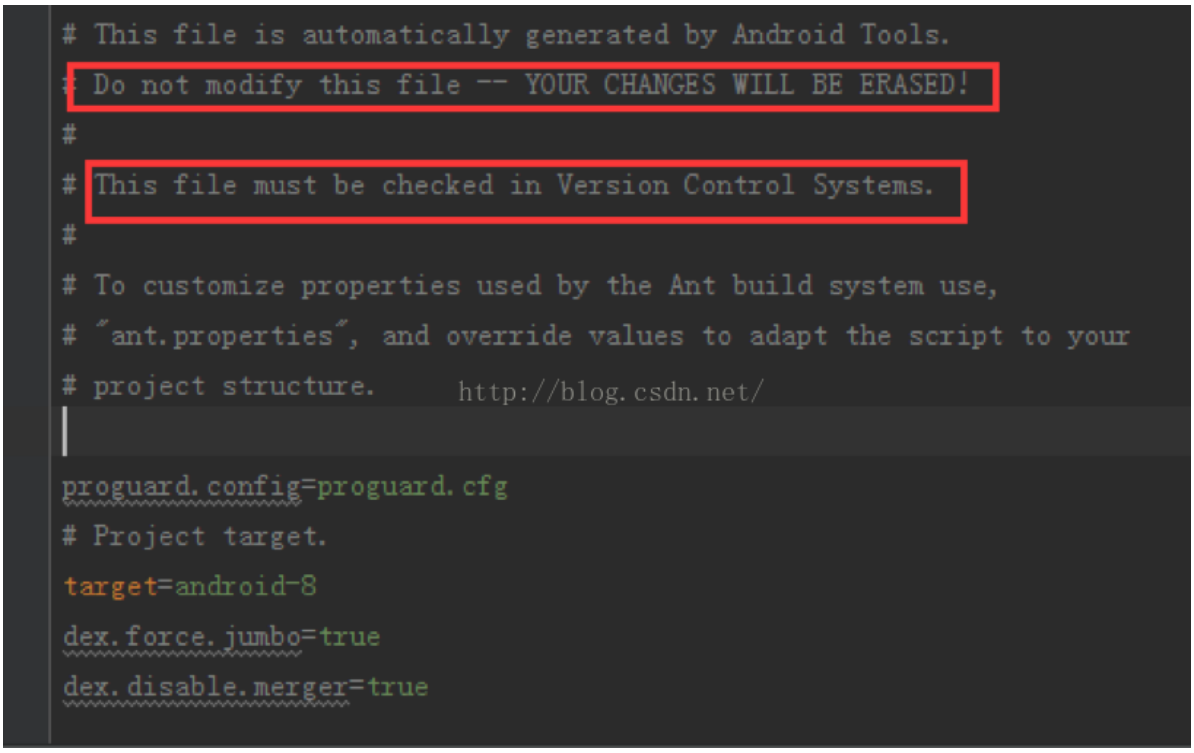
default.properties文件



我们看红色框中圈的两句，第一句说：不要修改这个文件，您的修改将被清除。第二句说：这个文件必须被版本控制系统检查。

意思就说这个文件我们不能动。这个文件的内容是配置混淆文件及android系统版本。

project.properties文件



这个文件和上个文件一样，我们不能修改。