Gradle开发入门

任玉刚

关于我

任玉刚（singwhatiwanna），滴滴出行Android技术专家，《Android开发艺术探索》作者，热爱开源和分享，活跃在Github和CSDN，曾先发主导开发了插件化框架dynamic-load-apk和VirtualAPK，目前在滴滴出行从事APP架构相关的开发工作。

微博：@任玉刚Coder　　微信公众号：

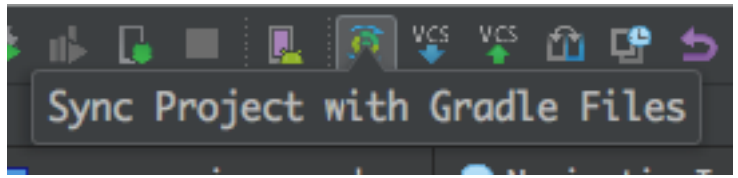
关于 Gradle

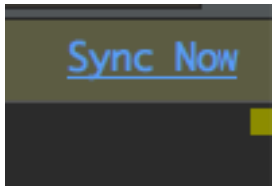
- 一个像 Ant 一样的非常灵活的通用构建工具
- 一种可切换的, 像 maven 一样的基于合约构建的框架
- 支持强大的多工程构建
- 支持强大的依赖管理(基于 Apachelvy)
- 支持已有的 maven 和 ivy 仓库
- 支持传递性依赖管理, 而不需要远程仓库或者 pom.xml 或者 ivy 配置文件
- 优先支持 Ant 式的任务和构建
- 基于 groovy 的构建脚本
- 有丰富的领域模型来描述你的构建

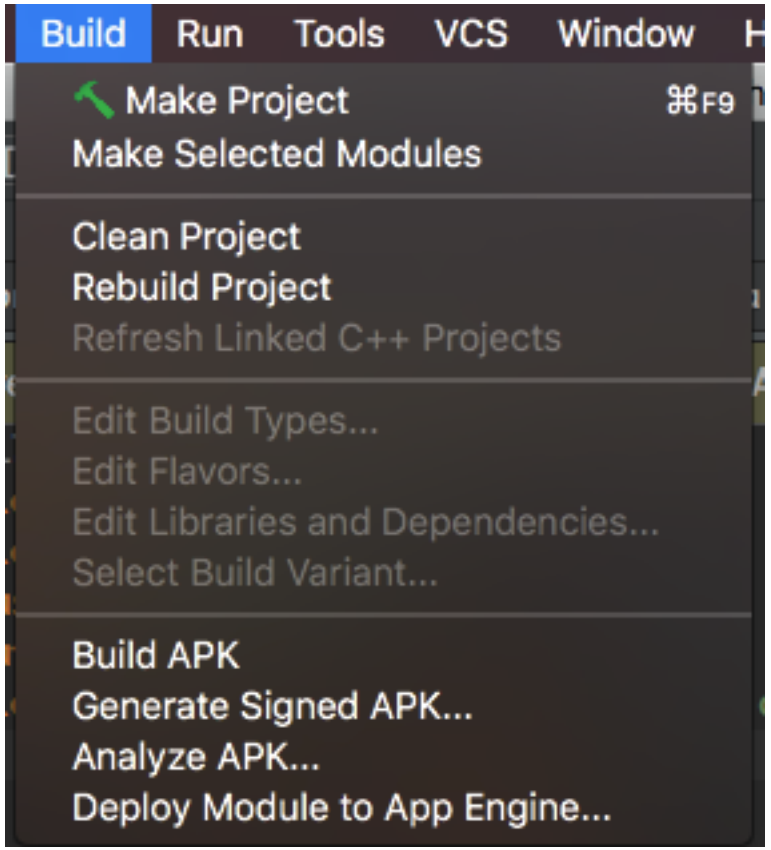
如何学习 Gradle

- 学习 Groovy（http://docs.groovy-lang.org/）
- 学习 Gradle DSL（https://docs.gradle.org/current/javadoc/org/gradle/api/Project.html）
- 学习 Android DSL和Task（http://google.github.io/android-gradle-dsl/current/index.html）

在 Android Studio 中使用 Gradle







Gradle 常用命令和参数

-v, --version	*# 查看当前 Gradle 版本信息*
-h	*# 查看帮助信息*
-s, --stacktrace / -S, --full-stacktrace	*# 如果发生异常则输出异常栈信息*
-i, --info / -d, --debug	*# 控制日志级别*
--stop	*# 停止正在运行的 Daemon*
-D, --system-prop / -P, --project-prop	*# 传入特定参数*
-m, --dry-run / -x, --exclude-task	*# 运行构建，但不执行任务的实际操作*
--offline / --refresh-dependencies	*# 离线模式构建，强制刷新依赖构建*
-b, --build-file / -c, --settings-file	*# 指定 build 脚本，指定 settings 脚本*
Ctrl + c	*# 终止当前运行*
./gradlew aFR > log.txt 2>&1	*# 输出所有内容到 log.txt*

推荐使用 Gradle Wrapper

- 构建时使用一致的 Gradle 版本(可通过 git 等版本管理工具配置)
- 使用 ./gradlew (类 Unix) 或 gradlew (Windows)
- 在 gradle/gradle-wrapper.properties 中配置 Gradle 版本

```
#Mon Dec 28 10:00:20 PST 2015
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-3.3-all.zip
```

Gradle 常用 Task

- 缩写 ——只需提供足够的可以唯一区分出该任务的字符即可
  - ./gradlew assembleFullRelease
  - ./gradlew assFRel
  - ./gradlew aFR

- tasks [--all] # 列出所有可运行的 task
- help --task <task> # 查看指定 task 的帮助信息
- androidDependencies # 查看当前 project 的 android 依赖树
- dependencies --configuration <config> # 查看 configuration 的 依赖树
- clean # 清除构建产出物
- assemble, assemble{BuildVariant} # 构建 apk，构建指定 Variant 的 apk
- install , install{BuildVariant} # 构建 apk 并安装
- uploadArchives # 构建 aar 并上传到指定依赖管理服务器

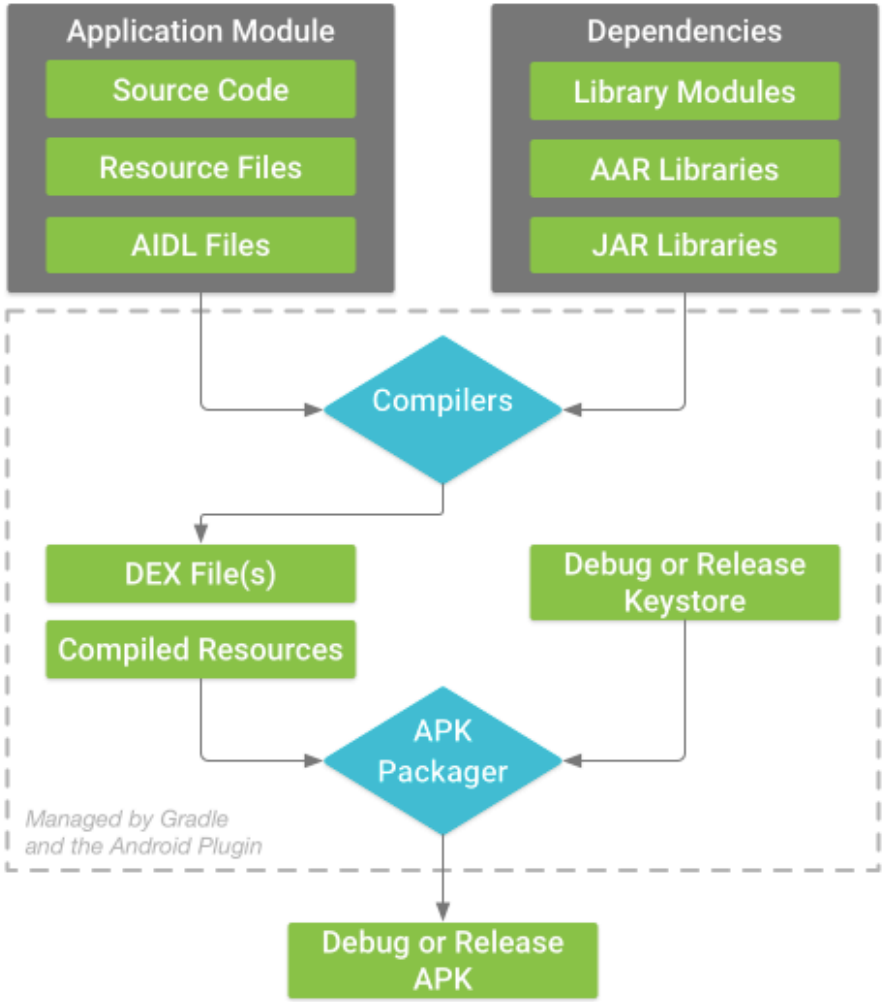
Android 常用 Task

- 一次构建过程中，assembleRelease依赖了许多task，如下是一些比较关键的task：

• compileReleaseJavaWithJavac	# 编译Java文件
• mergeReleaseAssets	# 收集所有的assets
• mergeReleaseResources	# 收集所有的resources
• processReleaseManifest	# 生成最终的AndroidManif.xml文件
• transformClassesAndResourcesWithProguardForRelease	# 混淆
• transformClassesWithDexForRelease	# 生成dex
• packageRelease	# 打包生成apk

Gradle 构建流程

- 生命周期
  - Initialization
  - Configuration
  - Execution



Gradle 依赖管理

- 依赖类型
  - External module dependency
    - compile group: 'org.springframework', name: 'spring-core', version: '2.5'
    - compile 'org.springframework:spring-core:2.5'
  - Project dependency
    - compile project(':shared')
  - File dependency
    - compile files('libs/a.jar', 'libs/b.jar')
    - compile fileTree(dir: 'libs', include: '*.jar')

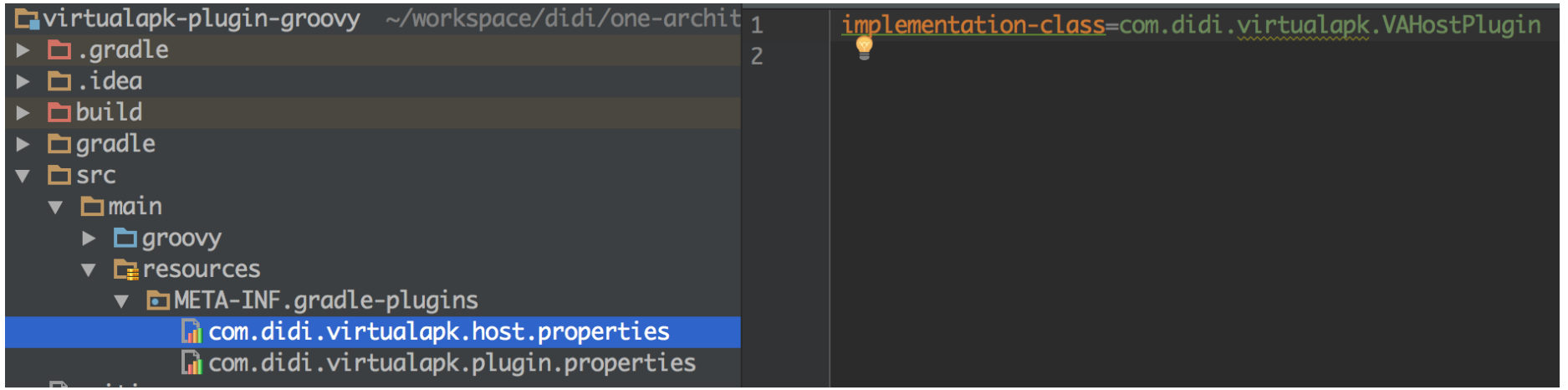
Gradle 自定义插件

- build.gradle

```
apply plugin: 'java'
apply plugin: 'groovy'
apply plugin: 'maven'

dependencies {
    compile gradleApi()
    compile 'com.android.tools.build:gradle:2.3.3'
}
```

- 声明插件



Gradle 构建源码查看

- 通过依赖相关库查看源码

```
apply plugin: 'java'
apply plugin: 'groovy'
apply plugin: 'maven'

dependencies {
    compile gradleApi()
    compile 'com.android.tools.build:gradle:2.3.3'
}
```

- 注意，不是所有的 gradle plugin版本都附带有源码的jar。如果遇到了一些没有源码的，即打开后看到的内容是反编译的class或者是没有javadoc的内容，最好换一个版本。

- 参考资料：https://fucknmb.com/2017/06/01/Android-Gradle-Plugin%E6%BA%90%E7%A0%81%E9%98%85%E8%AF%BB%E4%B8%8E%E7%BC%96%E8%AF%91/

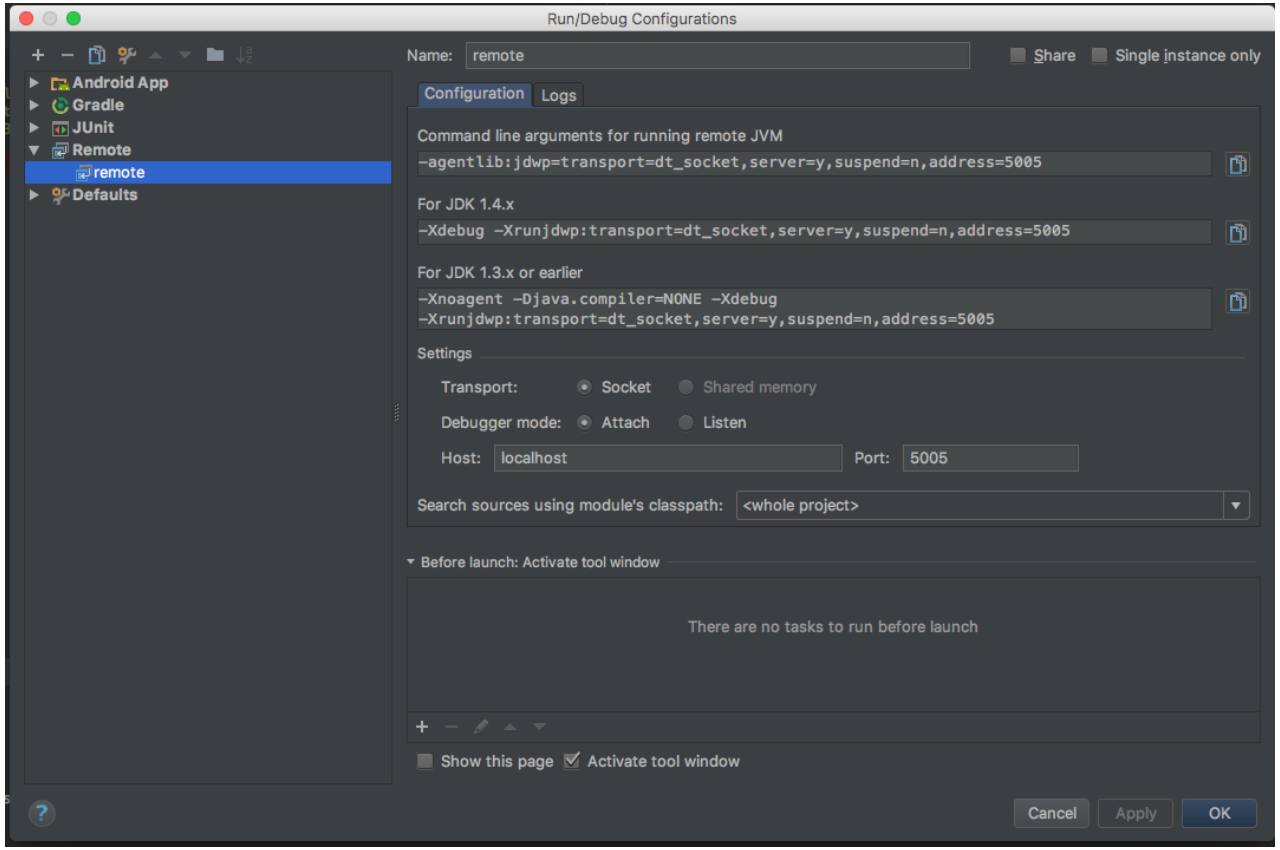
Android 自定义插件的运行

- build.gradle
- buildSrc
  - ln -s <plugin-dir> buildSrc
- 发布远程资源

- 参考资料： https://docs.gradle.org/current/userguide/custom_plugins.html

Android 自定义插件的调试方法

- debug 模式运行 gradle 脚本
  - gradle :app:clean -Dorg.gradle.debug=true --no-daemon
- 声明环境变量 GRADLE_OPTS
  - export GRADLE_OPTS="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=5005"
- attach Debugger

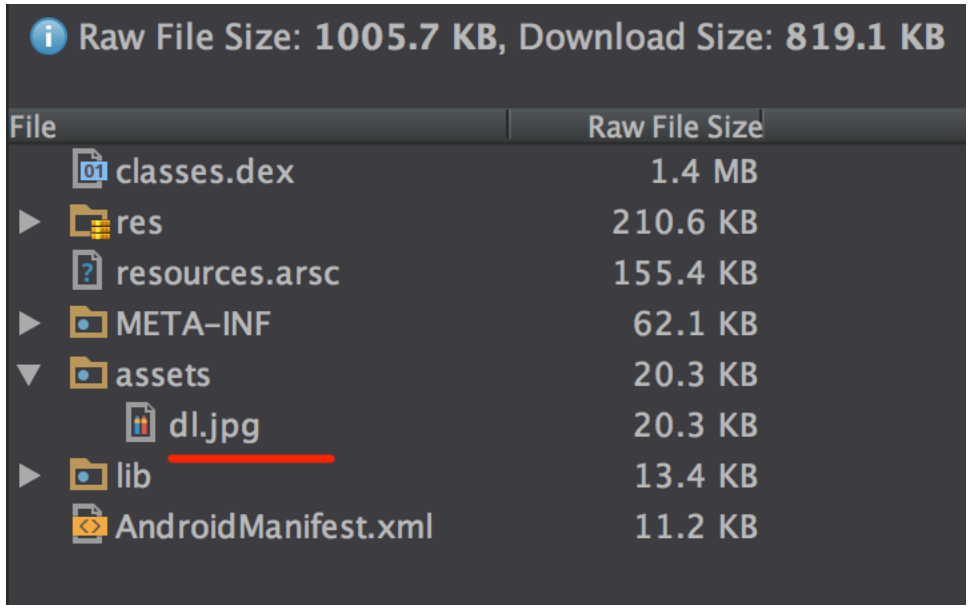


- 参考资料： https://fucknmb.com/2017/07/05/%E5%8F%88%E6%8E%8C%E6%8F%A1%E4%BA%86%E4%B8%80%E9%A1%B9%E6%96%B0%E6%8A%80%E8%83%BD-%E6%96%AD%E7%82%B9%E8%B0%83%E8%AF%95Gradle%E6%8F%92%E4%BB%B6/

Android 构建示例

问题：在构建过程中向apk的assets中添加一个文件，但是不允许在工程中直接添加这个文件，也不允许对apk进行二次打包。

```
project.afterEvaluate {
    project.android.applicationVariants.each { ApplicationVariant variant ->
        String variantName = variant.name.capitalize()
        def packageTask = project.tasks.getByName("package${variantName}")
        println "packageTask = ${packageTask.getClass()}"
        packageTask.doFirst {
            project.copy { param ->
                from "/Users/didi/Pictures/dl.jpg"
                into "${it.assets}"
            }
        }
    }
}
```



Android 构建示例

VirtualAPK renyugang$ ./gradlew clean assembleRelease
packageTask = class
com.android.build.gradle.tasks.PackageApplication_Decorated

```
 * Abstract task to package an Android artifact.
 */
public abstract class PackageAndroidArtifact extends IncrementalTask implements FileSupplier {

    public static final String INSTANT_RUN_PACKAGES_PREFIX = "instant-run";

    // ----- PUBLIC TASK API -----

    @InputFile
    public File getResourceFile() { return resourceFile; }

    public void setResourceFile(File resourceFile) { this.resourceFile = resourceFile; }

    @OutputFile
    public File getOutputFile() { return outputFile; }

    public void setOutputFile(File outputFile) { this.outputFile = outputFile; }

    @Input
    public Set<String> getAbiFilters() { return abiFilters; }

    public void setAbiFilters(Set<String> abiFilters) { this.abiFilters = abiFilters; }

    // ----- PRIVATE TASK API -----

    @InputFiles
    @Optional
    public Collection<File> getJavaResourceFiles() { return javaResourceFiles; }

    @InputFiles
    @Optional
    public Collection<File> getJniFolders() { return jniFolders; }

    private File resourceFile;

    private Set<File> dexFolders;

    private File assets;
```

Q & A

更多信息请访问

微博：@任玉刚Coder

微信公众号：