

Android Jenkins自动化打包

前言

每个Android开发应该都有经历过正在码代码的时候突然被打断要求打个啥啥环境啥啥配置的安装包，然后就得暂存代码、切换分支、更改配置、等待build、balabala.....等等一些事宜，导致浪费了研发的时间效率。如果可以将打包操作交给产品/测试/运维/XX呢？就是Jenkins自动化打包的事情了。

准备工作

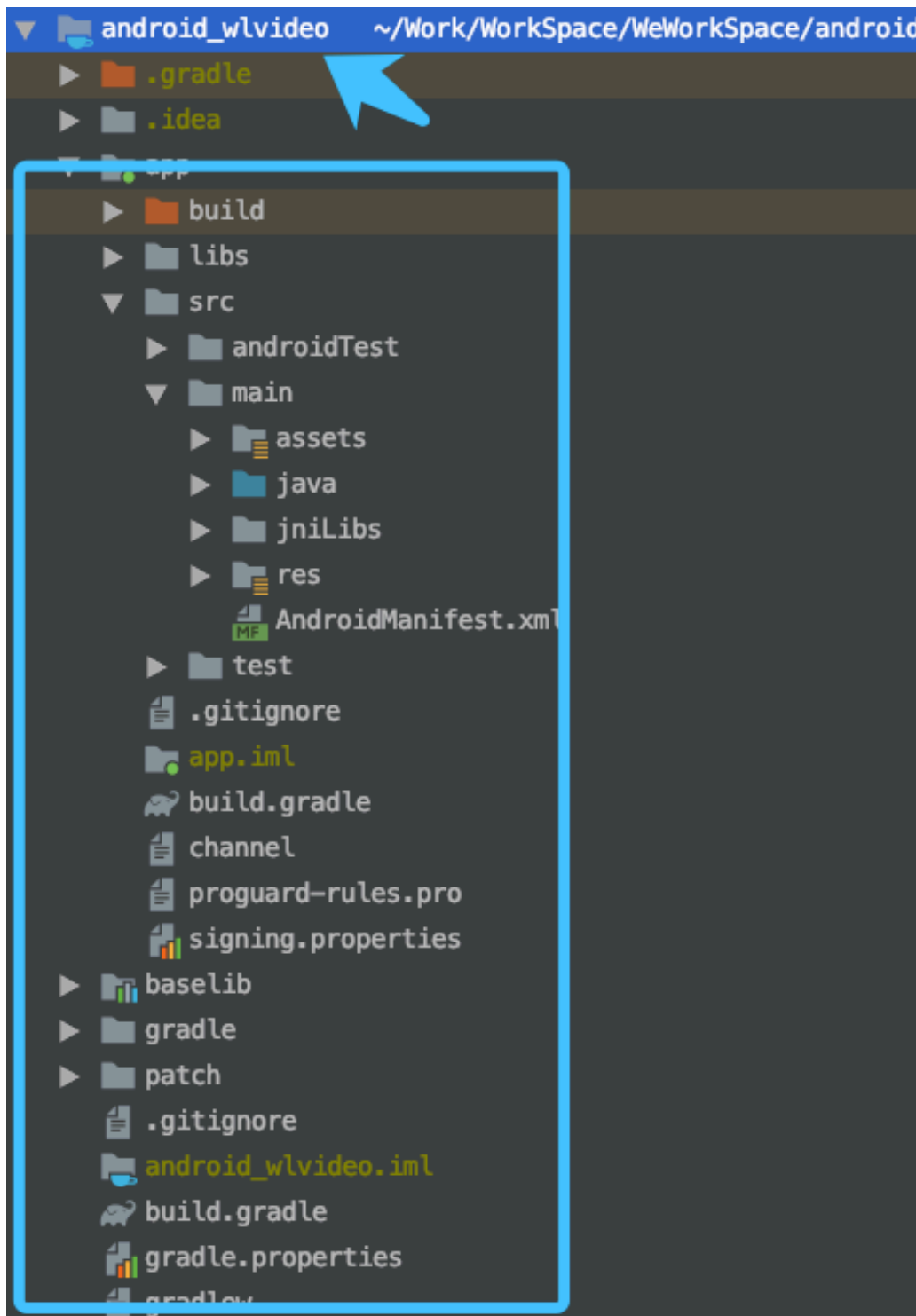
汇总

Jenkins自动化打包有一定的项目要求，也是优化项目的一种方式，以下四个点：

- 项目目录变更成直接可编译目录
- 环境不由某个变量发版的时候手动修改，也是为客户端统一配置打基础
 - 域名变量化
 - 从配置文件or常量读取
- Gradle配置动态属性控制环境变量
- Gradle配置组合编译执行命令

项目目录

Jenkins自动化打包，要求git项目目录为直接可编译目录：



环境配置

环境不由某个变量发版的时候手动修改，手动修改难免会有一些失误，所以要配置完整，通过打包选择来控制所有打包属性。APP内部的环境主要还是由域名或者一些H5的地址控制。

- 将APP内部域名前缀以及H5地址做成变量，用一个类维护起来

```

/**
 * Http请求URL前缀
 */
public static String HTTP_URL_VIDEO = "";
public static String HTTP_URL_LI_YUE_AD = "";
public static String HTTP_URL_CONFIG = "";

/**
 * H5地址
 */
public static String H5_URL_USER_AGREEMENT = "";
public static String H5_URL_PRIVACY_POLICY = "";
public static String H5_URL_CANCEL_ACCOUNT = "";
public static String H5_URL_WITHDRAW_PROGRESS = "";
public static String H5_URL_BIND_INVITE = "";

```

- 将APP内部域名前缀以及H5地址统一配置在配置文件or常量类里

```

#测试环境
VIDEO_URL_TEST=http://test-wlvideo.weli010.cn/wlvideo/
LI_YUE_AD_URL_TEST=http://adx-test.weliads.com/adxv3/api/ad/get/v2
CONFIG_URL_TEST=http://test-tinker.weli010.cn/tinker/
#测试H5地址
USER_AGREEMENT_URL_TEST=http://test-wlvideo-h5.weli010.cn/wlvideo_h5/agreement.html
PRIVACY_POLICY_URL_TEST=http://test-wlvideo-h5.weli010.cn/wlvideo_h5/privacy.html
CANCEL_ACCOUNT_URL_TEST=http://test-wlvideo-h5.weli010.cn/wlvideo_h5/userCancel.html
WITHDRAW_PROGRESS_URL_TEST=http://test-wlvideo-h5.weli010.cn/wlvideo_h5/withdrawHistory.html
BIND_INVITE_URL_TEST=http://test-wlvideo-h5.weli010.cn/wlvideo_h5/inviteCode.html

#生产环境
VIDEO_URL_PRODUCT=https://wlvideo.weli010.cn/wlvideo/
LI_YUE_AD_URL_PRODUCT=https://ads.weilitoutiao.net/kuaima_ads/api/ad/get/v2
CONFIG_URL_PRODUCT=http://tinker.weli010.cn/tinker/
#生产H5地址
USER_AGREEMENT_URL_PRODUCT=https://wlvideo-h5.weli010.cn/wlvideo_h5/agreement.html
PRIVACY_POLICY_URL_PRODUCT=https://wlvideo-h5.weli010.cn/wlvideo_h5/privacy.html
CANCEL_ACCOUNT_URL_PRODUCT=https://wlvideo-h5.weli010.cn/wlvideo_h5/userCancel.html
WITHDRAW_PROGRESS_URL_PRODUCT=https://wlvideo-h5.weli010.cn/wlvideo_h5/withdrawHistory.html
BIND_INVITE_URL_PRODUCT=https://wlvideo-h5.weli010.cn/wlvideo_h5/inviteCode.html

```

- 在build.gradle的productFlavors或者buildTypes配置BuildConfig中的URL测试or正式

```

WlvideoTest {
    buildConfigField "boolean", "URL_DEBUG", "true"
}

WlvideoProduct {
    buildConfigField "boolean", "URL_DEBUG", "false"
}

```

- APP初始化的时候通过BuildConfig的URL属性读取配置文件来赋值环境变量

```

if (BuildConfig.URL_DEBUG) {
    HttpConstant.HTTP_URL_VIDEO = FileHelper.getUrls(props,
        HttpConstant.HTTP_URL_VIDEO_TEST);
    HttpConstant.HTTP_URL_LI_YUE_AD = FileHelper.getUrls(props,
        HttpConstant.HTTP_URL_LI_YUE_AD_TEST);
    HttpConstant.HTTP_URL_CONFIG = FileHelper.getUrls(props,
        HttpConstant.HTTP_URL_CONFIG_TEST);
}

```

Gradle配置动态属性

在build.gradle的productFlavors或者buildTypes配置动态属性

```

buildTypes {
    debug {
        buildConfigField "boolean", "LOG_DEBUG", "true"
        buildConfigField "boolean", "STATISTICS_DEBUG", "false"
        signingConfig signingConfigs.debug
    }
    release {
        buildConfigField "boolean", "LOG_DEBUG", "false"
        buildConfigField "boolean", "STATISTICS_DEBUG", "false"
        //混淆开关
        minifyEnabled true
        //是否zip对齐
        zipAlignEnabled true
        //是否打开debuggable开关
        debuggable false
        //是否打开jniDebuggable开关
        jniDebuggable false
        // 移除无用的resource文件
        shrinkResources true
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        signingConfig signingConfigs.release
    }
}
}

```

```

productFlavors {
    WlvideoTest {
        buildConfigField "boolean", "URL_DEBUG", "true"
        manifestPlaceholders = [
            APP_KEY          : project.APP_KEY,
            APP_SECRET       : project.APP_SECRET,
            PACKAGE_NAME     : project.PACKAGE_NAME,
            CHANNEL_NAME     : project.CHANNEL_NAME,
            GAO_DE_APP_KEY   : project.GAO_DE_APP_KEY_TEST,
            UCLOUCD_APP_KEY  : project.UCLOUCD_APP_KEY,
            LINKEDME_KEY     : project.LINKEDME_KEY,
            UM_APP_ID        : project.UM_APP_ID_TEST,
            GETUI_APP_ID     : project.GETUI_APP_ID_TEST,
            GETUI_APP_KEY    : project.GETUI_APP_KEY_TEST,
            GETUI_APP_SECRET : project.GETUI_APP_SECRET_TEST,
            USER_PRIVACY     : project.USER_PRIVACY_LIST,
            WX_APP_ID        : project.WX_APP_ID,
            WX_APP_SECRET    : project.WX_APP_SECRET,
            QQ_APP_ID        : project.QQ_APP_ID,
            QQ_APP_KEY       : project.QQ_APP_KEY
        ]
    }

    WlvideoProduct {
        buildConfigField "boolean", "URL_DEBUG", "false"
        manifestPlaceholders = [
            APP_KEY          : project.APP_KEY,
            APP_SECRET       : project.APP_SECRET,
            PACKAGE_NAME     : project.PACKAGE_NAME,
            CHANNEL_NAME     : project.CHANNEL_NAME,
            GAO_DE_APP_KEY   : project.GAO_DE_APP_KEY_RELEASE,
            UCLOUCD_APP_KEY  : project.UCLOUCD_APP_KEY,
            LINKEDME_KEY     : project.LINKEDME_KEY
        ]
    }
}

```

Gradle配置组合编译执行命令



在build.gradle的productFlavors和buildTypes配置组合编译命令

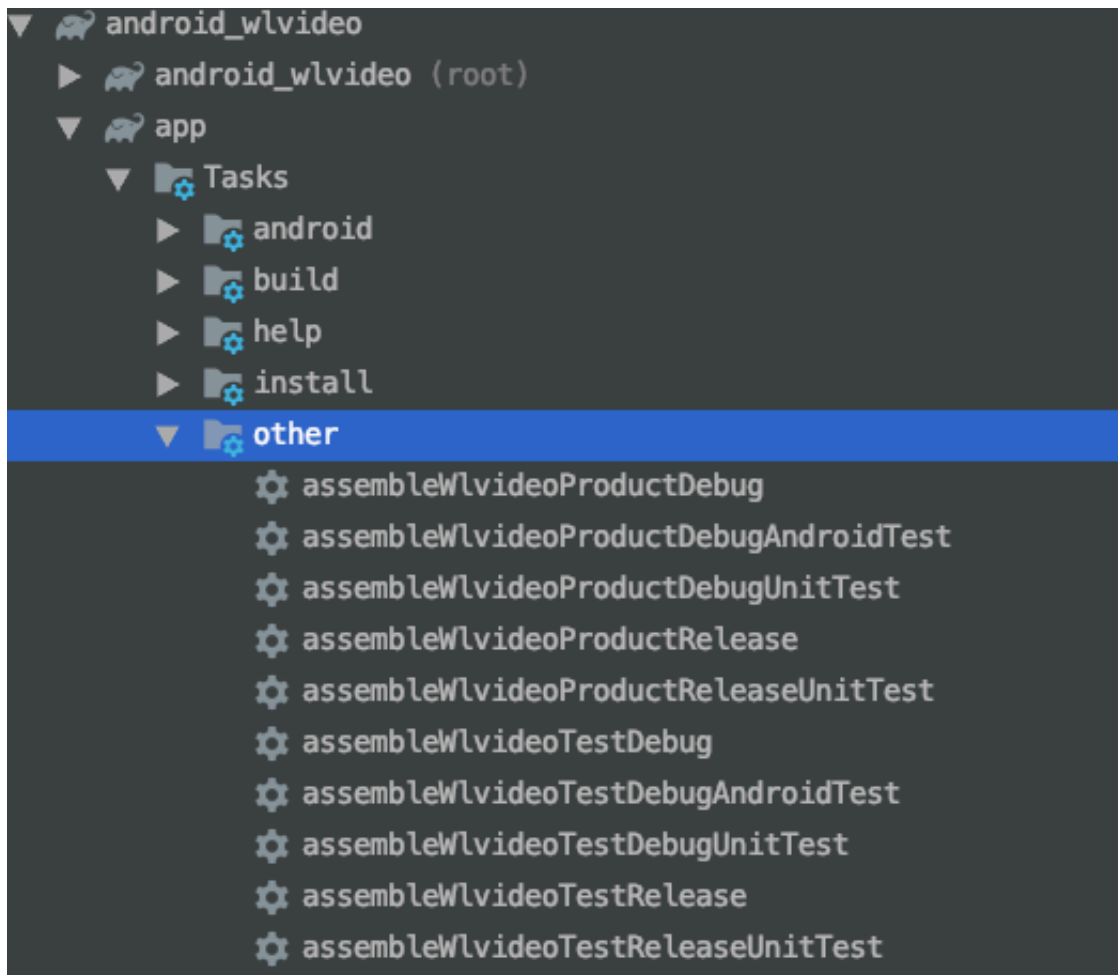
```

productFlavors {
    WlvideoTest {
        buildConfigField "boolean", "URL_DEBUG", "true"
        manifestPlaceholders = [
            APP_KEY          : project.APP_KEY,
            APP_SECRET       : project.APP_SECRET,
            PACKAGE_NAME     : project.PACKAGE_NAME,
            CHANNEL_NAME     : project.CHANNEL_NAME,
            GAO_DE_APP_KEY   : project.GAO_DE_APP_KEY_TEST,
            UCLOUCD_APP_KEY  : project.UCLOUCD_APP_KEY,
            LINKEDME_KEY     : project.LINKEDME_KEY,
            UM_APP_ID        : project.UM_APP_ID_TEST,
            GETUI_APP_ID     : project.GETUI_APP_ID_TEST,
            GETUI_APP_KEY    : project.GETUI_APP_KEY_TEST,
            GETUI_APP_SECRET : project.GETUI_APP_SECRET_TEST,
            USER_PRIVACY     : project.USER_PRIVACY_LIST,
            WX_APP_ID        : project.WX_APP_ID,
            WX_APP_SECRET    : project.WX_APP_SECRET,
            QQ_APP_ID        : project.QQ_APP_ID,
            QQ_APP_KEY       : project.QQ_APP_KEY
        ]
    }
}

WlvideoProduct {
    buildConfigField "boolean", "URL_DEBUG", "false"
    manifestPlaceholders = [
        APP_KEY          : project.APP_KEY,
        APP_SECRET       : project.APP_SECRET,
        PACKAGE_NAME     : project.PACKAGE_NAME,
        CHANNEL_NAME     : project.CHANNEL_NAME,
        GAO_DE_APP_KEY   : project.GAO_DE_APP_KEY_RELEASE,
        UCLOUCD_APP_KEY  : project.UCLOUCD_APP_KEY,
        LINKEDME_KEY     : project.LINKEDME_KEY
    ]
}

```

Build Variants			
	Module		Active Build Variant
Structure	app		WlvideoTestDebug
	baselib		WlvideoProductDebug
			WlvideoProductRelease
			WlvideoTestDebug
			WlvideoTestRelease



至此，Jenkins自动化打包项目准备工作已完成。

Jenkins自动化打包配置

这里先配置一个简单的打包配置，后期再同步复杂build模式给大家。

步骤一：新建项目

在Jenkins主页面，找到Android客户端的单独Jenkins编译项目目录：android_project，点击新建，输入项目名称，选择构建一个自由风格的项目。

Item名称

android_rose_test

构建一个自由风格的软件项目

这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.

构建一个maven项目

构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置.

External Job

这个类型的任务允许你记录执行在外部Jenkins的任务, 任务甚至运行在远程机器上.这可以让Jenkins作为你所有自动构建系统的控制面板.参阅 [这个文档查看详细内容](#).

构建一个多配置项目

适用于多配置项目,例如多环境测试,平台指定构建,等等.

复制已有的Item

要复制的任务名称

OK

步骤二：设置项目信息

在项目配置页面，输入当前项目的信息，并且设置打包机（运维只在这台机子上配置了Android Sdk环境）。

项目名称

android_wlvideo_test

描述

鲤刷刷测试环境

[Plain text] [预览](#)

GitBucket

URL

Enable hyperlink to the issue

丢弃旧的构建

参数化构建过程

关闭构建 (重新开启构建前不允许进行新的构建)

在必要的时候并发构建

Restrict where this project can be run

Label Expression

slave4-android-s4.all.jenkins.freed.so

[Label](#) is serviced by 1 node

步骤三：代码配置

在项目配置里，选择git拉取代码，输入git地址，设置git账号密码，并选择当前打包分支。

源码管理

☐ None

☒ Git

Repositories

Repository URL

Credentials

 Add

Branches to build

Branch Specifier (blank for 'any')

源码库浏览器

(自动)

Additional Behaviours

Add ▼

☐ Subversion

步骤四：设置触发器

在项目配置里，选择日程表编译，设置编译时间，示图是设置的每天的凌晨3点打包。

构建触发器

☐ 触发远程构建 (例如,使用脚本)

☐ Build after other projects are built

☒ Build periodically

日程表

0 3 * * *

 Spread load evenly by using 'H 3 * * *' rather than '0 3 * * *'

Would last have run at Tuesday, February 11, 2020 3:00:28 AM CST; would next run at Wednesday, February 12, 2020 3:00:28 AM CST.

☐ Build when a change is pushed to GitBucket

☐ Poll SCM

步骤五：设置编译配置

在项目配置里，选择gradle编译版本，设置编译脚本task。

Invoke Gradle script

Invoke Gradle

Gradle Version

Gradle4.4

Use Gradle Wrapper

Build step description

Switches

Tasks

clean assembleWlvideoTestRelease --stacktrace --debug -g gradle-user-home

Root Build script

Build File

Specify Gradle build file to run. Also, [some environment variables are available to the build script](#)

Force GRADLE_USER_HOME to use workspace

步骤六：设置上传脚本

在项目配置里选择Execute Shell，配置上传脚本信息，上传的apk名称以及ftp相对目录

Execute shell

Command

```
sh /data/jenkins/workspace/apk_upload.sh 'wlvideo_test.apk' 'wlvideo/android/'
```

See [the list of available environment variables](#)

步骤七：构建后操作

在项目配置里设置构建后识别的apk文件

构建后操作

Archive the artifacts

用于存档的文件

**/*.apk

Did not manage to validate **/*.apk (may be too slow)

点击保存。

步骤八：构建

在项目页或者项目列表页点击构建，成功后会自动上传到ftp对应目录，下载二维码会更新。

 返回面板

 状态

 修改记录

 工作空间

 立即构建

 删除 Project

 配置

Project android_wlvideo_test

鲤刷刷测试环境

 工作区

 最后一次成功的构建结果
 [WlvideoTest_v1.0.3_release.apk](#) 7.76 MB  [view](#)

 最新修改记录

 Build History

[构建历史](#)

S	W	Name ↓	上次成功	上次失败	上次持续时间	
		android_wlbox_product	14 小时 - #42	无	1 分 20 秒	
		android_wlbox_test	12 小时 - #57	28 days - #14	1 分 18 秒	
		android_wlnovel_product	8 小时 20 分 - #5	无	4 分 9 秒	
		android_wlnovel_test	8 小时 20 分 - #5	无	4 分 7 秒	
		android_wlreader_product	8 小时 20 分 - #25	无	3 分 25 秒	
		android_wlreader_test	8 小时 20 分 - #28	24 days - #1	3 分 31 秒	
		android_wlvideo_product	3 小时 57 分 - #38	1 月 0 days - #1	1 分 22 秒	
		android_wlvideo_test	11 小时 - #78	1 月 0 days - #33	2 分 2 秒	
		android_zhwnl_product	4 小时 41 分 - #36	5 days 10 小时 - #29	3 分 14 秒	
		android_zhwnl_test	4 小时 41 分 - #296	1 月 0 days - #256	3 分 12 秒	