

## Ant自动编译打包&发布 android项目

Eclipse用起来虽然方便，但是编译打包android项目还是比较慢，尤其将应用打包发布到各个渠道时，用Eclipse手动打包各种渠道包就有点不切实际了，这时候我们用到Ant帮我们自动编译打包了。

### 1 Ant自动编译打包android项目

#### 1.1 Ant安装

ant的安装比较简单，下载ant压缩包 <http://ant.apache.org> (最新的为1.9.3版本)，下载之后将其解压到某个目录(本人解压到E:\Program Files\apache-ant-1.9.3)，然后配置环境变量(新建 ANT\_HOME环境变量，值为ant所在的目录，然后将ANT\_HOME/bin添加到path中)，如图：



打开命令行工具，输入 `ant -version`，如果出现如下结果，说明ant 安装成功。

#### 1.2 为Android项目生成Ant配置build.xml

打开命令行工具，切换路径到项目所在的目录；输入命令

```
android update project --name <project_name> --target <target_ID>--path  
<path_to_your_project>
```

<http://developer.android.com/tools/projects/projects-cmdline.html>

项目根目录下多了build.xml，以及local.properties两个文件。

其中local.properties写明了我们的android SDK的目录(其实是环境变量ANDROID\_HOME的值，所以如果环境变量中没有这个的，请增加)。

build.xml则是ant构建的最重要脚本，打开一看，发现里面其实大部分都是写注释，有用的没几

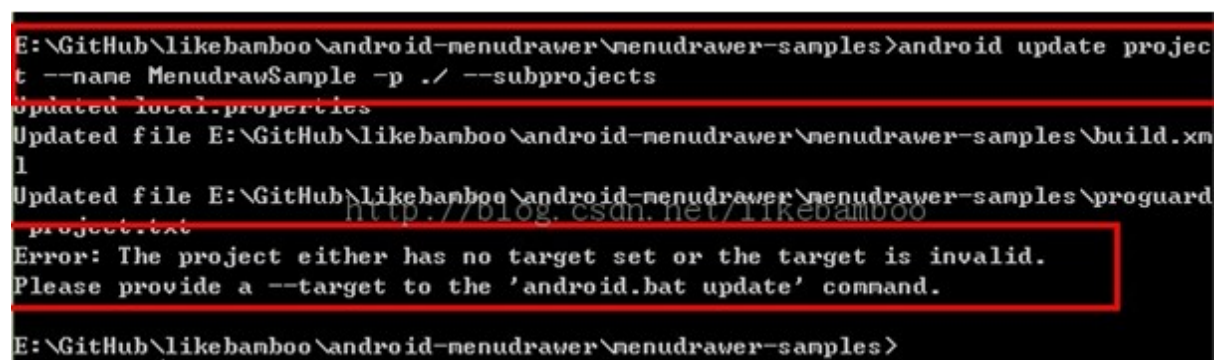
行，这是因为生成的这个build.xml引用了android SDK自带的构建脚本，具体目录是{sdk目录}/tools/ant/build.xml。

这样，项目就支持ant编译打包了，但是，有人可能会说了，我的项目有引用第三方的jar包，而且还引用了其他的android library 项目，该怎么办呢？

## 1.3 android使用ant打包时，添加第三方jar包

### 1.3.1 libs 库中的第三方jar包

如果项目只是引用了第三方jar包，只要将jar包放在libs文件夹下就ok了，ant会在编译打包过程中会自动将第三方jar加进去的。但是当我们的android 项目参考了其他library项目，这时候我们最初在输入android update 命令的时候应该多一个参数 --subprojects：



```
E:\GitHub\likebamboo\android-menudrawer\menudrawer-samples>android update project --name MenudrawSample -p ./ --subprojects
Updated local.properties
Updated file E:\GitHub\likebamboo\android-menudrawer\menudrawer-samples\build.xml
Updated file E:\GitHub\likebamboo\android-menudrawer\menudrawer-samples\proguard-project.txt
Error: The project either has no target set or the target is invalid.
Please provide a --target to the 'android.bat update' command.
E:\GitHub\likebamboo\android-menudrawer\menudrawer-samples>
```

你发现报错了，不要着急，这是因为那个library 还不支持ant自动编译，我们需要先让它也支持。

进入到library项目所在的目录，输入命令 android update lib-project -p ./ （注意是 lib-project）；

再回到原项目,输入命令” android update project --name MenudrawSample -p ./ --subprojects”，这下就OK了。

### 1.3.2 user library库中第三方jar包

在Android开发中，除了通常在Eclipse中的编译方法之外，有的时候为了进行持续集成，可能还需要用ant进行自动化编译。Android SDK本身已经提供了默认的ant编译脚本，就在每个工程下的build.xml中，其中引用了SDK的编译脚本\${sdk\_dir}/tools/ant/build.xml。

通常情况下，在工程根目录下直接执行 ant debug 即可进行一次正常的build。默认的classpath会包括libs目录下的所有jar文件。但是如果工程中使用了USER LIBRARY，或者引用了外部的jar文件，那么在编译中就可能会遇到问题，因为USER LIBRARY等这些jar文件不会被自动包含在classpath中，这时就要扩展ant的path变量，把自己的jar文件加入到classpath中。

通过察看sdk提供的build.xml编译脚本，可以发现javac使用的classpath定义如下：



```
<path id="project.javac.classpath">
```

```

    <path refid="project.all.jars.path"></path>
    <path refid="tested.project.classpath"></path>
</path>
<javac encoding="${java.encoding}"
    source="${java.source}" target="${java.target}"
    debug="true" extdirs="" includeantruntime="false"
    destdir="${out.classes.absolute.dir}"
    bootclasspathref="project.target.class.path"
    verbose="${verbose}"
    classpathref="project.javac.classpath"
    fork="${need.javac.fork}">
    <src path="${source.absolute.dir}"></src>
    <src path="${gen.absolute.dir}"></src>
    <compilerarg line="${java.compilerargs}"></compilerarg>
</javac>

```



其中 project.all.jars.path 包含了所有的jar文件，我们可以通过在工程目录下的build.xml中重新定义这个变量来引入其他的jar文件。

例如在我的工程中，引用了ormlite这个ORM库，为了能够在开发中使用“attach source”察看源码，该jar文件不能放在libs目录中，因为Eclipse不允许对libs目录中的jar文件“attach source”。

因此我将此文件放到了libs/ormlite目录中，为了能够将这两个jar文件加入到classpath中，就要重新定义 project.all.jars.path 这个元素。

基本思路是，重新定义 -pre-compile这个target，在其中重新定义 project.all.jars.path 的值。

这段代码写在哪里呢？在每个项目的build.xml中引用了当前目录下的custom\_rules.xml，那么我们就在项目根目录下创建一个custom\_rules.xml，内容如下：



```

<?xml version="1.0" encoding="UTF-8"?>
<project name="custom_rules" default="release">
<target name="-pre-compile">
<echo message="JARPATH=${toString:project.all.jars.path}"></echo>
<echo message="JARPATH=${jar.libs.dir}"></echo>
    <property name="ormlite.dir" value="${jar.libs.dir}/ormlite"> </property>
    <path id="ormlite.lib">
        <path path="${toString:project.all.jars.path}"></path>
        <pathelement location="${ormlite.dir}/ormlite-android-4.41.jar">
</pathelement>
        <pathelement location="${ormlite.dir}/ormlite-core-4.41.jar">
</pathelement>
    </path>

```

```
<path id="project.all.jars.path">
    <path refid="ormlite.lib"></path>
</path>
<echo message="JARPATH=${toString:project.all.jars.path}"></echo>
</target>
</project>
```



<http://my.oschina.net/yunfound/blog/169288>

## 1.4 编译打包项目

ant debug: 生成一个测试版apk，默认使用 debug key 进行签名，生成的apk (your\_project\_name-debug.apk) 在bin目录下。

ant release: 生成一个未签名和未aligned的apk包， project\_name-release-unsigned.ap和 project\_name-release-unaligned.apk 在bin目录下。

---

## 2 签名与渠道包

基于ant自动编译打包现有的android项目，可以在ant打包应用的时候加入签名信息以及自动打包渠道包。

### 2.1 加入签名信息

在项目的根目录下建一个ant.properties文件，输入如下内容，其中keystore密码和alias密码可以不指定（防泄漏），那么在命令执行的过程中会要求你输入。



```
#keystore的路径，必须使用正斜杠
key.store= "E:/wp_android_sample/me.key"
#keystore的密码
#key.store.password=*****
#alias名
key.alias=me
#alias密码
#key.alias.password=*****
```



在项目根目录下运行 ant release 命令就会帮你生成一个经过签名和aligned的apk，生成的apk (your\_project\_name-release.apk) 在bin目录下

## 2.2 自动打包渠道包

实现批量循环打包需要一个类似于for循环的功能，在Ant的核心包里没有相关的For循环的Task，即不支持for循环，但是ant支持第三方扩展包，以支持更多的其他功能。

于是我们要下载相应的支持for循环的扩展包。可以使用开源的Ant-contrib包。下载地址：<http://ant-contrib.sourceforge.net/>。

下载后的解压得到的jar文件放到ant的lib目录。接下来我们就可以打包渠道包了，具体做法是：

(1) 首先在ant.properties文件中增加属性 market\_channels (渠道列表,以逗号分割),version(应用程序版本名)

```
#渠道市场列表
market_channels=91,360,wandoujia,baidu
#版本号
version=1.2.1
```

(2) 在我们项目的build.xml中加入如下代码:



```
<!-- 渠道包打包脚本  ant deploy-->
<taskdef resource="net/sf/antcontrib/antcontrib.properties">
    <classpath>
        <pathelement location="lib/ant-contrib-1.0b3.jar"/>
    </classpath>
</taskdef>

<target name="deploy">
    <foreach target="modify_manifest" list="${market_channels}" param="channel"
    delimiter=",">
    </foreach>
</target>

<target name="modify_manifest">
    <replaceregexp flags="g" byline="false">
        <!-- 匹配的内容是 android:value="*****" android:name="UMENG_CHANNEL" -->
        <regexp pattern='android:value="(.)" android:name="UMENG_CHANNEL"' />
        <!-- 匹配之后将其替换为 android:value="渠道名" android:name="UMENG_CHANNEL" -->
        <substitution expression='android:value="${channel}"
        android:name="UMENG_CHANNEL"' />
        <!-- 正则表达式需要匹配的文件为AndroidManifest.xml -->
        <fileset dir="" includes="AndroidManifest.xml" />
    </replaceregexp>
```

```
<property name="out.release.file"
location="${out.absolute.dir}/${ant.project.name}_${channel}.apk" />
<!--包 -->
<antcall target="release" />
<!--输出渠道包到bin/out目录下 -->
<copy tofile="${out.absolute.dir}/out/${ant.project.name}v${version}-${
${channel}.apk" file="bin/${ant.project.name}-release.apk"/>
</target>
```



在项目根目录下运行 `ant deploy` 命令就会帮你各个渠道的签名包了(为了全程可以自动执行, `ant.properties` 文件中的 `keystore` 的密码可以指定, 这样在执行过程中就不需要手动输入密码了), 在 `bin` 目录的 `out` 目录下。

参考:

<http://blog.csdn.net/likebamboo/article/details/17888563>

<http://blog.csdn.net/likebamboo/article/details/17953259>

<http://my.oschina.net/yunfound/blog/169288>

<http://developer.android.com/tools/projects/projects-cmdline.html>