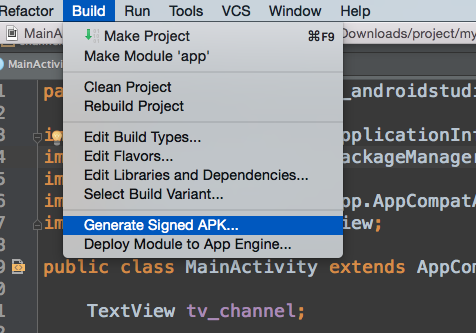
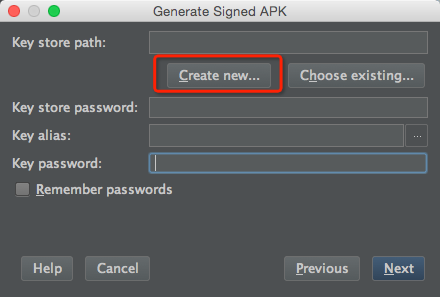
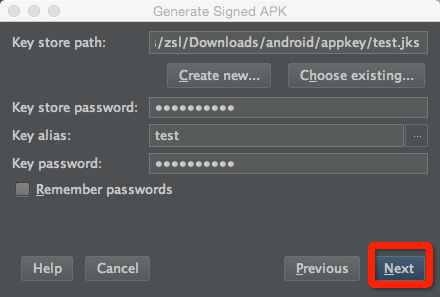
**Failed to resolve:com.android.support:appcompat-v7:XX**

**SDK版本，SDK-tools的版本不一致，就是你的Android Studio版本不同，编译工具版本不同**

**生成jks签名**

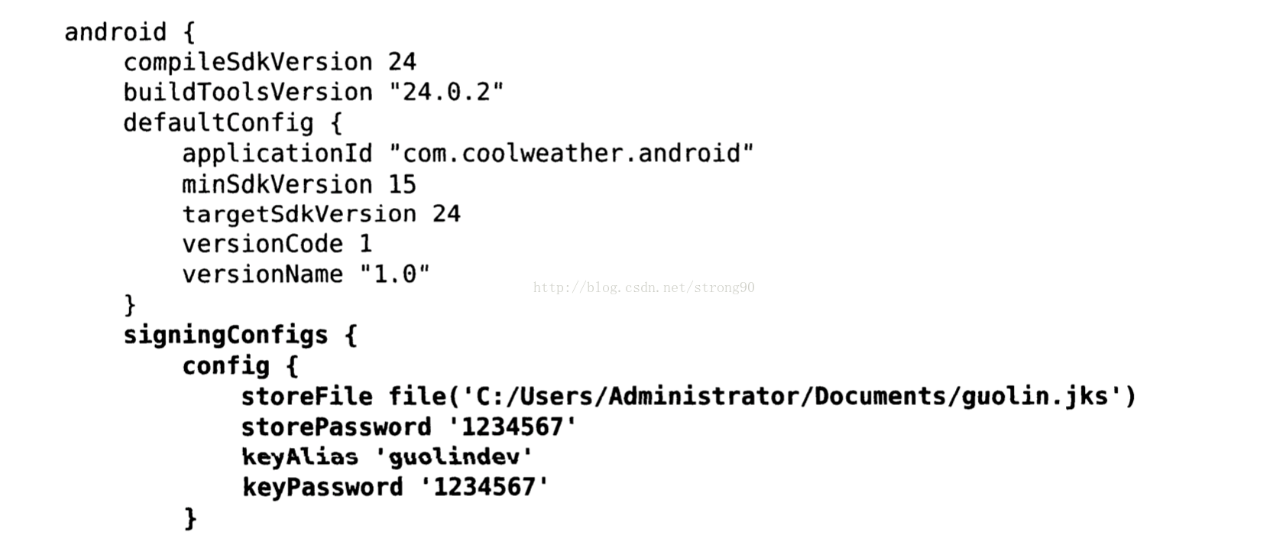






**使用gradle生成APK**

1、编辑 app/build.gradle 文件，在 android 闭包中添加如下内容



可以看到，在android闭包中添加了一个signingConfigs的闭包，在signingConfigs又添加了一个config的闭包。接着在config的闭包中配置keystore文件的各种信息

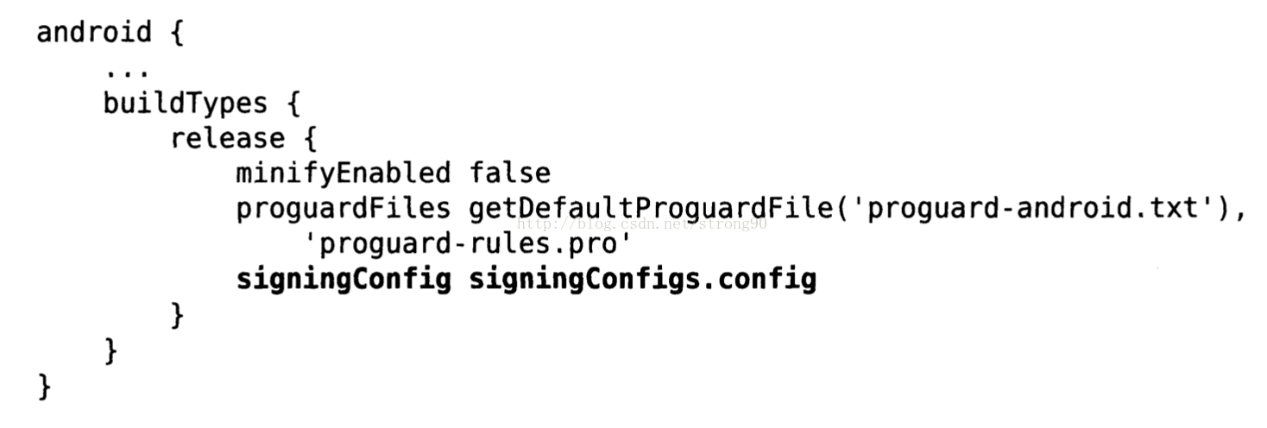
storeFile：指定keystore文件的位置

storePassword：指定密码

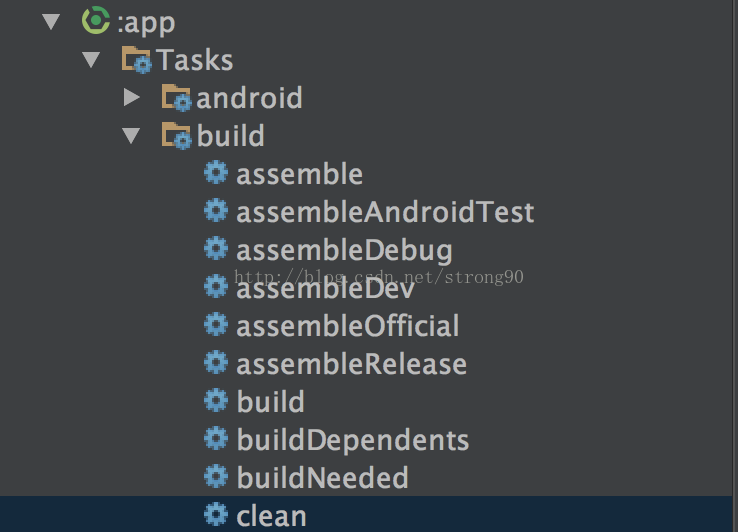
ketAlias：指定别名

keyPassword：指定别名密码

2、配置好签名信息后，接下来需要在生成正式版APK的时候去应用这个配置就可以了，如下图所示：



3、生成APK文件。点击右侧工具栏的Gradle--项目名--:app--Tasks--build，如图所示



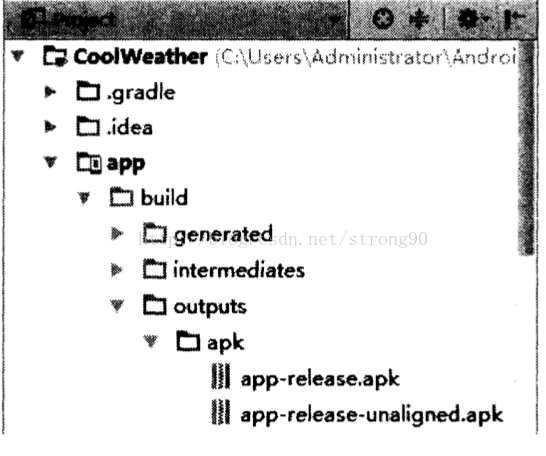
**assembleDebug：生成测试版的APK文件**

**assembleRelease：生成正式版的APK文件**

**assemble：同时生成测试版和正式版的APK文件**

**clean：在生成APK之前，先要双击clean来清理下当前项目，然后执行你需要生成的apk**

4、APK文件会自动生成在 app/build/outputs/apk 目录下

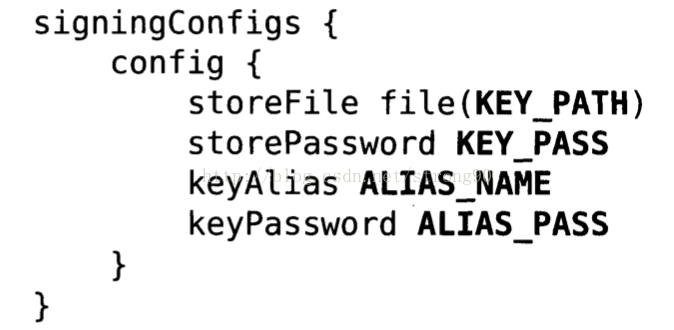


5、注意

上述方式的keystore文件所有信息都是以明文形式直接配置在build.gradle中，不太安全。android推荐的做法是将这类敏感数据配置在一个独立的文件里面，然后再在build.gradle中读取这些数据。

根目录下有一个gradle.properties文件，专门用来配置全局键值对数据的，我们在 gradle.properties 文件中添加如下内容：





**使用gradle生成多渠道APK**

1.背景：

apk可能要求有多种配置要求，比如访问的ip不同，配置文件资源不同等，这就要求可以一次性打包多个apk，避免手动多次打包；

2.原理：

a.在app的build.gradle文件种配置：

android {

....

sourceSets {

//设置各个渠道的目录文件

main {

jniLibs.srcDirs = ['libs']

}

supion {

java.srcDirs = ['src/supion', 'src/supion/java', 'src/supion/java/']

}

hopeland {

java.srcDirs = ['src/hopeland', 'src/hopeland/java', 'src/hopeland/java/']

}

}

productFlavors {//两个渠道

hopeland {

}

supion {

}

}

//遍历productFlavors打包输出文件名

applicationVariants.all { variant ->

variant.outputs.each { output ->

def appName = 'Ioter'

def oldFile = output.outputFile

def buildName

variant.productFlavors.each { product ->

buildName = product.name

}

output.outputFile = new File(

output.outputFile.parent + "/${variant.buildType.name}",

appName + "-" + getDate() + "-" + "${variant.buildType.name}-${variant.versionName}-" + buildName + ".apk".toLowerCase())

}

}

}

def getDate() {

def date = new Date()

def formattedDate = date.format('yyyyMMddHHmm')

return formattedDate

}

b.在代码目录下，创建各自渠道的文件目录（和main同级），

c.在各自渠道配置相同的文件DeviceUtil.java，在该文件种配置各自的资源。

注：在main目录种不能有DeviceUtil文件，原理是一个渠道中，main目录和渠道目录合并，如果相同就存在冲突了。