实现目的：

根据命令行传入的参数，如：app 名称、app 图标、主页的网址等，自动带证书打包出一个 web 应用 app。

实现原理和过程：

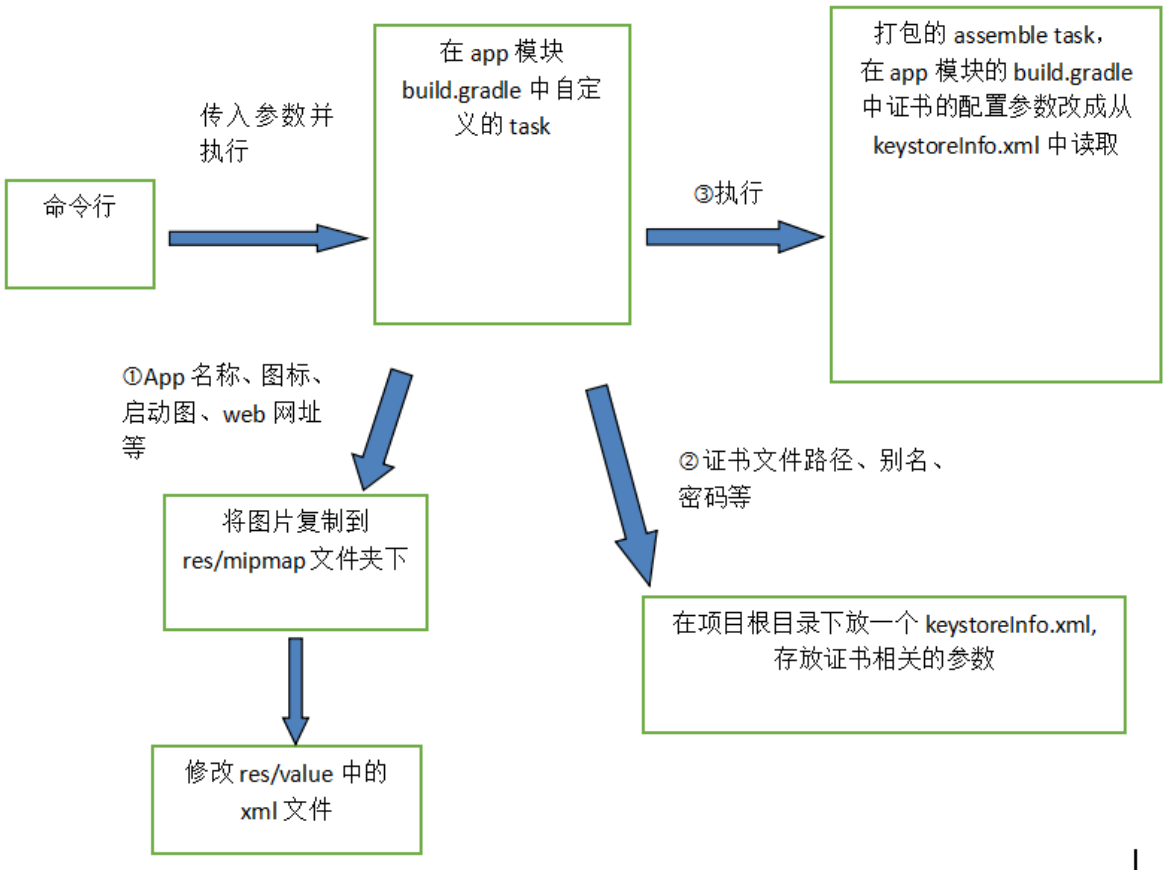
app 编译和打包的过程，是调用相关的 task 进行的，如打包是调用 assemble。task 定义在 build.gradle 中，可从命令行调起，并可以传入参数。

首先需要一个已经完成的 Web 应用 app 项目来作为模板项目，然后在 app 模块的 build.gradle 中自定义一个 task。

该 task 接收命令行传入的参数，参数包含 app 名称、图标路径、web 网址、证书路径、证书别名和密码等，task 先将图片资源复制进项目里存放图片的文件夹如 mipmap,然后修改 res/values 文件中代表 app 名称、图标等的值，即可达到根据参数修改 resValues 的功能。

参数中跟证书相关的，则放到自己创建的 keystoreInfo.xml 中，打包时证书的配置参数写成从 keystoreInfo.xml 中读取，则达到了根据参数配置证书的功能。

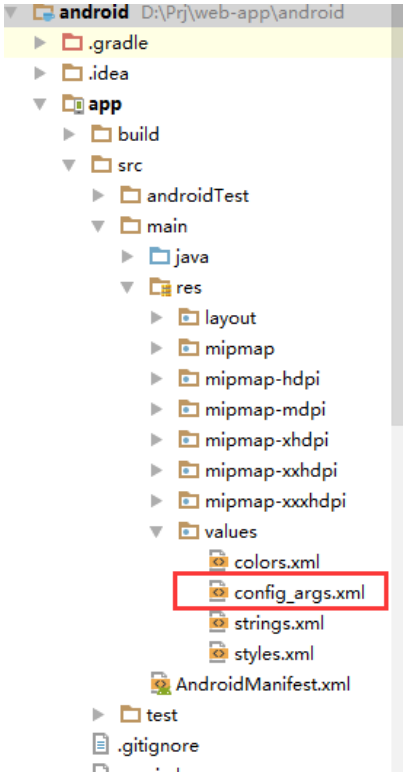
task 完成上面两个操作后，通过命令行调起 assemble task 即可。



1. 模板项目

这个模板项目是个简单的单 WebView 应用，打开后显示启动图片 n 秒，然后就是一个 Webview。

项目中的 app 名字、图标、启动图、web 网址等 value 写在 res/value 文件中，比如自己创建的一个 config_args.xml




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">mAppName</string>
    <string name="web_home_url">http://www.sina.com</string>
    <drawable name="app_launcher">@mipmap/m_app_launcher</drawable>
    <drawable name="splash_img">@mipmap/m_splash_img</drawable>
</resources>
```

在 app module 的 build.gradle 中跟证书有关的参数写成从 keystore_info.xml 文件中读取，这个 keystore_info .xml 文件 是自己创建的,里面保存了跟证书有关的信息。

```
//从 keystore_info.xml 读取签名文件的信息
def keystoreFile = rootProject.file("keystore_info.xml")
def signInfo = new XmlParser().parse(keystoreFile)
```

```
android {
   ...

    //    签名
    signingConfigs {
        release {
//          证书文件
            storeFile file( signInfo.storeFile.text() )
//          证书库的密码
            storePassword signInfo.storePassword.text()
//          证书密码
            keyPassword signInfo.keyPassword.text()
//          别名
            keyAlias signInfo.keyAlias.text()
        }
```

```
    }

    ...
}
```

keystore_info .xml

```
<?xml version="1.0" encoding="utf-8"?>
<signInfo>
    <storeFile>F:/keystore/demo.keystore</storeFile>
    <storePassword>123456</storePassword>
    <keyPassword>123456</keyPassword>
    <keyAlias>mKeyStore</keyAlias>
</signInfo>
```

2. 在 gradle 中定义一个 task

android 中项目的编译打包基于项目模型，gradle 提供了构建项目的框架，项目构建的具体内容由插件来完成。

gradle 是一种领域相关语言（Domain Specific Language）,相当于某个行业的“行话”，所以有很多特定的术语，同时基于 groovy，所以需要了解 groovy 的语法。

gradle 中的两个主要对象是 project 和 task，project 即 android 项目中的一个 module，为 task 提供执行的上下文。一个 task 表示一个逻辑上较为独立的执行过程，比如编译 android 源码，打包等。

编译过程分为三个阶段：

- **初始化阶段：**创建 Project 对象，如果有多个 build.gradle，也会创建多个 project.
- **配置阶段：**在这个阶段，会执行所有的编译脚本，同时还会创建 project 的所有的 task，为后一个阶段做准备。
- **执行阶段：**在这个阶段，gradle 会根据传入的参数决定如何执行这些 task,真正 action 的执行代码就在这里

task 在 module 的 build.gradle 中定义，如定义一个简单的 task

```
task helloWorld << {
   println "Hello World!"
}
```

<< 表示执行阶段

由于我们要编译出的 apk 是 app module 中的，所以自定义的 task 写在 app module 的 build.gradle 中

3. utils.gradle

为了方便开发，将一些常用的“方法”放在了项目根目录下的 utils.gradle 中。

utils.gradle 中的“方法”其实是闭包，通过在闭包名前加 ext 使其成为成员变量供其他代码使用。

如，在 utils.gradle 中定义 myHello 的闭包：

```
ext.myHello = { a ->
    println "hello $a"
}
```

a 表示传入的参数，$a 表示参数的值

在 app module 的 build.gradle 中引入

```
apply from: "../utils.gradle"
```

然后即可像方法一样使用：

```
myHello("world")
```

则会打印出 hello world

utils.gradle 中封装了一些常用功能，如：执行命令行 等等

```
/*
* 执行命令行
*/
ext.exeCmd = { cmd ->
    Process p = cmd.execute()
    p.consumeProcessOutput( System.out ,System.err )
    p.waitFor()
//    println "${p.text}"
    return !p.exitValue()
}

/*
* 将打包前要改变的参数放到 resource value 文件中
*/
ext.writeArgToResourceValue = { appName, webHomeUrl, appLauncher, splashImg ->
    String xml =
"""<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">$appName</string>
    <string name="web_home_url">$webHomeUrl</string>
    <drawable name="app_launcher">$appLauncher</drawable>
    <drawable name="splash_img">$splashImg</drawable>
</resources>
"""
    File file = new File("app/src/main/res/values/config_args.xml")
    if (!file.exists()) { file.createNewFile() }
    file.write(xml)
}
```

```
/*
* 将证书相关的信息写入文件 keystore_info.xml
* */
ext.writeSignInfo = { storeFile, storePassword, keyPassword, keyAlias ->
    String xml =
"""<?xml version="1.0" encoding="utf-8"?>
<signInfo>
    <storeFile>$storeFile</storeFile>
    <storePassword>$storePassword</storePassword>
    <keyPassword>$keyPassword</keyPassword>
    <keyAlias>$keyAlias</keyAlias>
</signInfo>
"""
    File file = new File("keystore_info.xml")
    if (!file.exists()) { file.createNewFile() }
    file.write(xml)
}

/**
* 复制文件
* */
ext.copyFile = { File inputFile, File outputFile, boolean toOverwrite = false ->
    if (!inputFile.exists()) {
        throw new FileNotFoundException("Template file does not exist:
${inputFile.getAbsolutePath()}")
    }
    outputFile.getParentFile().mkdirs()
    if (toOverwrite || !outputFile.exists()) {
        outputFile.bytes = inputFile.readBytes()
    }
}

/**
*  删除文件或文件夹
*  */
ext.removeFile = { File f ->
    if (!f.exists()){
        throw new FileNotFoundException("file does not exist: ${f.getAbsolutePath()}")
    }
    if (f.isDirectory()) { f.listFiles().each { removeFile(it) } }
    f.delete()
}
```

4. 从命令行执行 task 并传入参数的方法

在 app module 的 build.gradle 中定义一个 task,将传入的参数显示出来，只需要这样子写：

```
task show << {
      println a
   }
```

命令行输入为

```
gradlew -P a=hello show app:show
```

app 表示 app module，show 表示要执行的 task

-P 表示后面为一个参数，task 会自动根据参数名来匹配参数，如会将命令行中的 a=hello 传入给 task 中的 a

若要传入多个参数，则每个参数前都要加-P,如

```
 gradlew -P a=hello -P b=world show app:show
```

5. 在 app module 的 build.gradle 中自定义一个 task 来根据参数打包 apk

```
task inputArgsToPackage << {
    println "input args:"
    println "appName=$appName"
    println "webHomeUrl=$webHomeUrl"
    println "appLauncherImgPath=$appLauncherImgPath"
    println "splashImgPath=$splashImgPath"
    println "storeFile=$storeFile"
    println "storePassword=$storePassword"
    println "keyPassword=$keyPassword"
    println "keyAlias=$keyAlias"

//    复制 app 图标到 mipmap 文件夹
    def appLauncherFileName = "m_app_launcher"
    println "copy app launcher file..."
    def sourceFile = new File(appLauncherImgPath)
    def desFile = new File("app/src/main/res/mipmap/${appLauncherFileName}.png")
    copyFile(sourceFile, desFile, true)
    println "copy succeed."

// 复制启动图片到 mipmap 文件夹
    def splashImgFileName = "m_splash_img"
    println "copy splash img..."
    copyFile(new File(splashImgPath), new
File("app/src/main/res/mipmap/${splashImgFileName}.png"), true)
    println "copy succeed."

//    将参数 appName,webHomeUrl,appLauncher,splashImg 写入 resource value 文件
    println "write args to resource value..."
    writeArgToResourceValue(appName, webHomeUrl, "@mipmap/${appLauncherFileName}",
"@mipmap/${splashImgFileName}")
    println "write succeed."

// 将签名相关的参数写入 keystore.xml
    println "write sign info to keystore.xml"
    writeSignInfo(storeFile, storePassword, keyPassword, keyAlias)
    println "write succeed."

// 清除上一次打包的结果
//  println "cleaning the last package"
//  exeCmd("gradlew.bat clean")
//  println "clean done"

// 执行打包命令
    println "packaging..."
//    String result = exeCmd("gradlew.bat packageDebug")
    def isExeSucceed = exeCmd("gradlew.bat assembleRelease")
    if( isExeSucceed ){
        println "package succeed !"
    }else{
        println "package fail !"
    }
}
```

命令行执行这个 task

```
gradlew -q -P appName=mAppName -P webHomeUrl=http://www.sina.com -P
appLauncherImgPath=F:/img/mAppLauncher.png -P splashImgPath=F:/img/mSplashImg.png -P
storeFile=F:/keystore/demo.keystore -P storePassword=123456 -P keyPassword=123456 -P
keyAlias=mKeyStore app:inputArgsToPackage
```

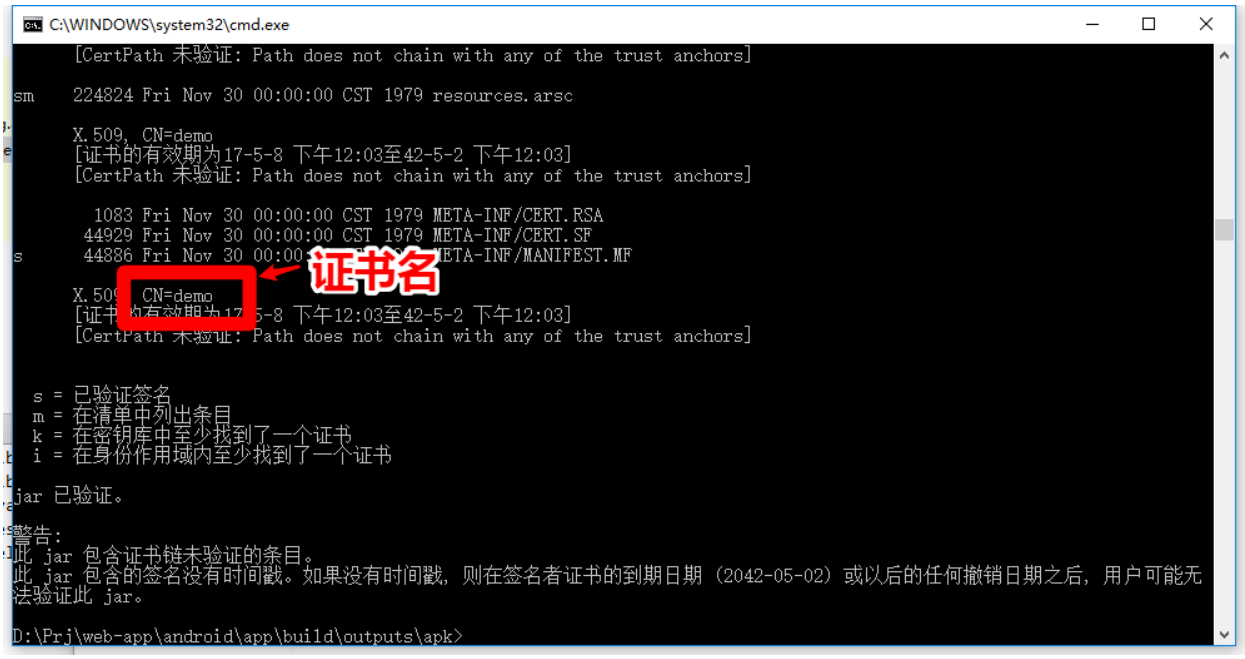
脚本执行完毕后即自动生成了 apk

6. 查看签名信息的方法

//查看签名信息

```
jarsigner -verbose -verify -certs app-release.apk
```

如下则表示有证书



7. 自动生成证书

命令行自动生成证书的命令行：

```
keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -
validity 10000 -dname "cn=Mark Jones, ou=JavaSoft, o=Sun, c=US" -storepass 123456 -keypass 123456
```

参数说明：

-keystore 证书文件名

-alias 别名
-keyalg 加密算法
-keysize 密钥长度
-validity 证书有效期
-dname 用户信息（cn 名字与姓氏，ou 组织单位名称，o 组织名称，c 国家地区代码）
-storepass 证书库密码
-keypass 证书密码

生成证书的 task

```
//gradlew -q -P keystore=mKeystore.keystore -P alias=minstone -P username=minstone -P
company=minstone -P organization=minstone -P countrycode=cn -P storepass=123456 -P keypass=123456
app:genKeystore
task genKeystore << {
    String cmd = "keytool -genkey -v -keystore $keystore -alias $alias -keyalg RSA -keysize 2048 -
validity 10000 -dname \"cn=$username, ou=$company, o=$organization, c=$countrycode\" -storepass
$storepass -keypass $keypass"
    def isExeSucceed = exeCmd(cmd)
    if (isExeSucceed) {
        println "package succeed !"
    } else {
        println "package fail !"
    }
}
```

在执行自动打包的 task 时，先执行生成证书的 task，再把自动打包过程的证书的信息改成生成的证书的信息，即可自动生成证书并打包。

8. 项目地址

http://codesync.cn/mobile/app/web-app