

Android Studio 兼容导入 android eclipse 工程

## 为 android studio 设置环境变量

sudo gedit /etc/profile 加入

```
export ANDROID_STUDIO=/home/benson/android-tool/android-studio
export PATH=${ANDROID_STUDIO}/bin:${ANDROID_STUDIO}/gradle/gradle-2.4/bin:$PATH
```

## Android Studio (AS) 工程和 eclipse 工程的区别

A. AS 构建项目使用的 gradle, eclipse 构建 android 项目用的是 ADT

B. AS 的 project 相当于 eclipse 的 workspace

C. AS 的 module 相当于 eclipse 的 project

因此最好是将 eclipse 的 project 当作一个个 AS 的 module 来处理, 我们知道如果用的 AS 建立的一个 App module, 它的目录结构和 eclipse 下的工程有很大差别的。如果要兼容的迁移 eclipse 工程到 AS, 则原有 eclipse 的工程目录结构是不能变的。

AS 使用的构建工具是 gradle, 使用 build.gradle 脚本来构建项目, 只需要在需要迁移的 eclipse 工程下增加一个 build.gradle 脚本文件, 通过编辑脚本内容来实现兼容。

下面以迁移 DvbService lib 工程和 StarryLauncher app 工程来说明迁移方法。

迁移 DvbService, 首先在原来 DvbService 工程增加一个 build.gradle 文件。



然后编辑文件内容如下:

```

*build.gradle x
apply plugin: 'com.android.library' //申明这是一个library module

//这里是配置sdk版本, 以及配置各个资源的路径, 目的是为了兼容原有的eclipse工程
android {
    compileSdkVersion 17
    buildToolsVersion "23.0.1"

    sourceSets {
        main {
            manifest.srcFile 'AndroidManifest.xml'
            java.srcDirs = ['src']
            resources.srcDirs = ['src']
            aidl.srcDirs = ['src']
            renderscript.srcDirs = ['src']
            res.srcDirs = ['res']
            assets.srcDirs = ['assets']
            jniLibs.srcDirs = ['libs']
        }
    }
}

//这里是配置依赖关系, compile的意思是编译并一起打包发布; provided的意思是只是编译不需要一起打包, 例如layoutlib.jar就是只需要编译不打包
//使用的时候注意修改一下路径
dependencies {
    compile fileTree(dir: 'libs', include: '*.jar')
    provided files(['/home/benson/android-tool/android-sdk_eng.benson_linux-x86/platforms/android-4.0.4/data/layoutlib.jar'])
}

```

只后将 build.gradle 上传 svn 就行了, 到时候别人更新只后稍微修改一下就可以使用了。文件我已经上传, 大家更新看一下就知道了。

迁移 StarryLauncher 工程, 同样在原来的工程下增加 build.gradle 脚本, 然后编辑内容。这里我就不在说了, 我已经编辑好了上传 svn 了, 下一步就是教大家如何利用 android studio checkout svn 上的工程。

## AS 中使用 SVN 导入工程

上面说了把 eclipse 的工程看做一个个 module, 所以现在需要建立一个工程来导入这些 module, 最好不要用 AS 菜单向导去创建一个 Project, 因为如果用 AS 创建, 首先它会要你填包名啊, app 名这些, 然后创建完之后, 就会默认创建了一个 app 的 module. 但我们不需要这个 module, 我们已经有 module 了, 都在 svn 上, 现在做的是把它导出来而已. 因此我们手动创建工程。

### 1. 找个地方创建一个放 Project 的文件夹, 例如在用户目录下创建 GatewayProject 的文件夹

```
mkdir ~/GatewayProject
```

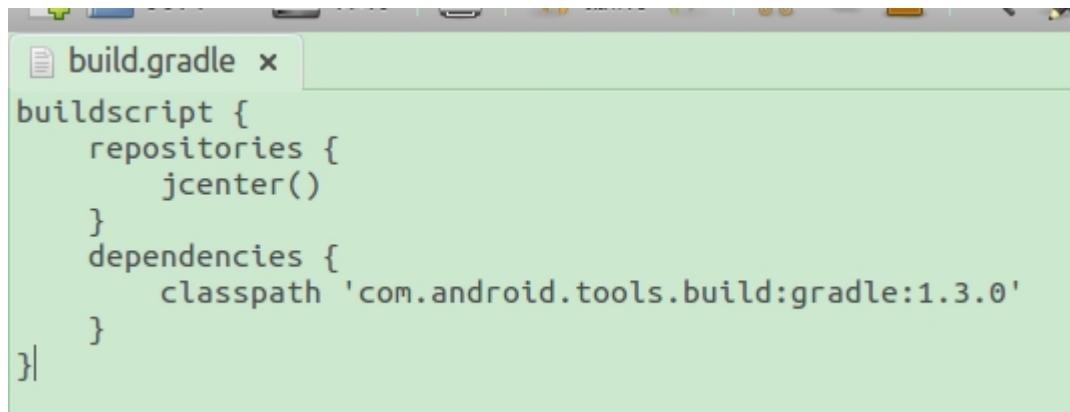
### 2. 进入文件夹, 创建一个 build.gradle 文件和一个 settings.gradle 文件

```
cd ~/GatewayProject
```

```
touch build.gradle settings.gradle
```

```
gedit build.gradle
```

编辑 build.gradle, 加上以下内容:

A screenshot of an IDE window titled 'build.gradle x'. The window shows the following Groovy code:

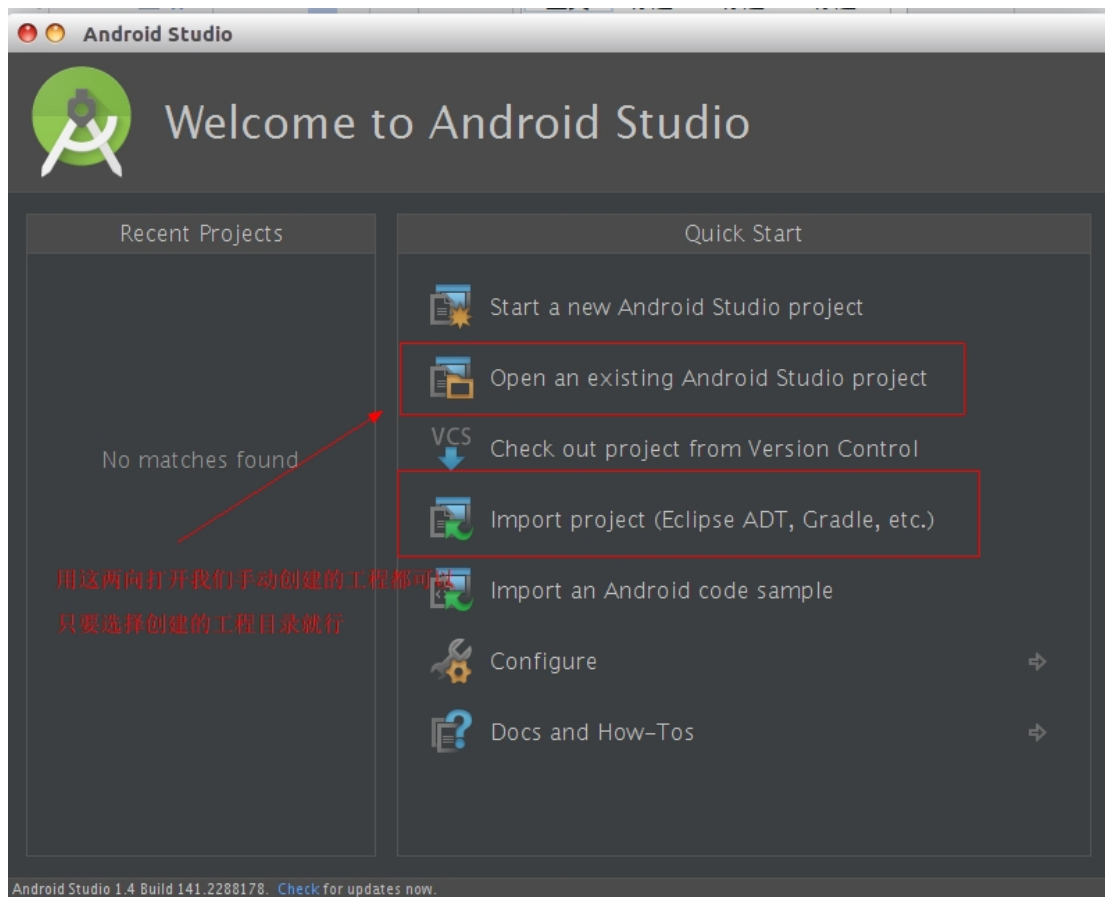
```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.3.0'
    }
}
```

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.3.0'
    }
}
```

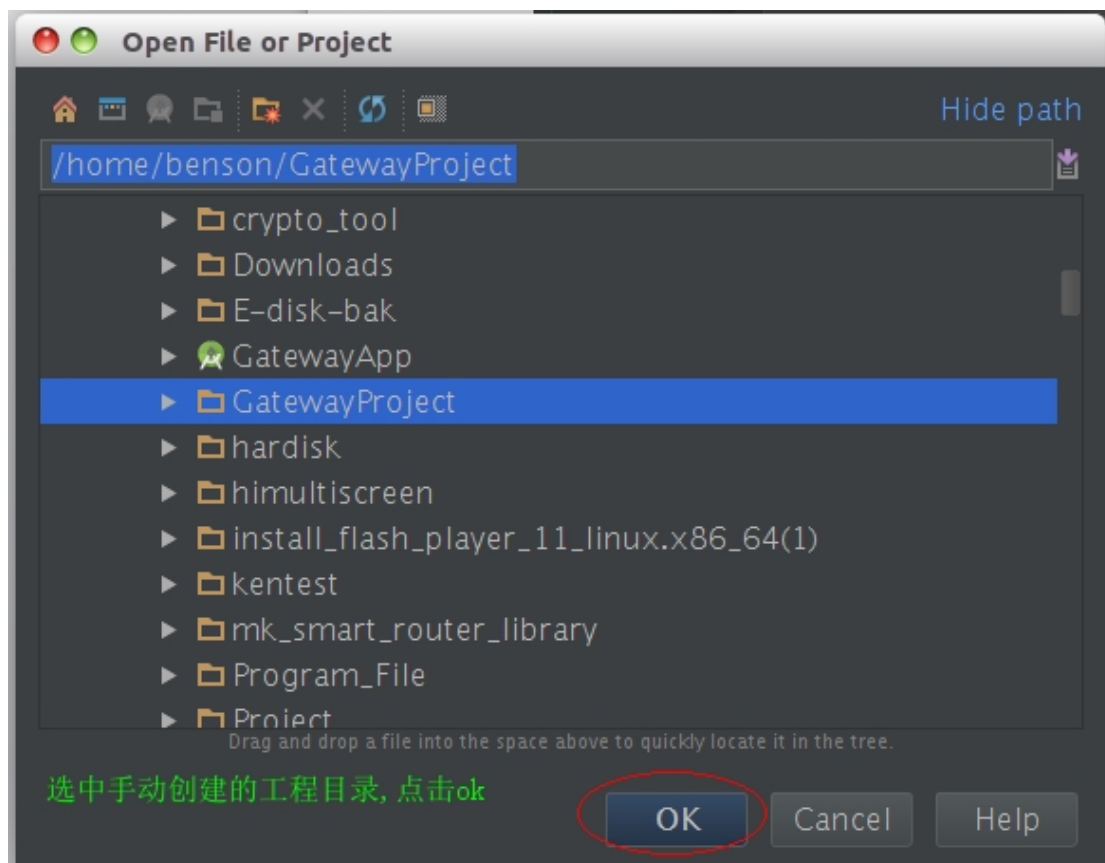
3.一开始就教大家设置了环境变量,因此可以使用 **gradle** 命令, 在工程目录下执行初始化 **gradle** 的命令, 这样我们的工程基本就建立好了, 下一步就是用 **AS** 来打开这个工程.

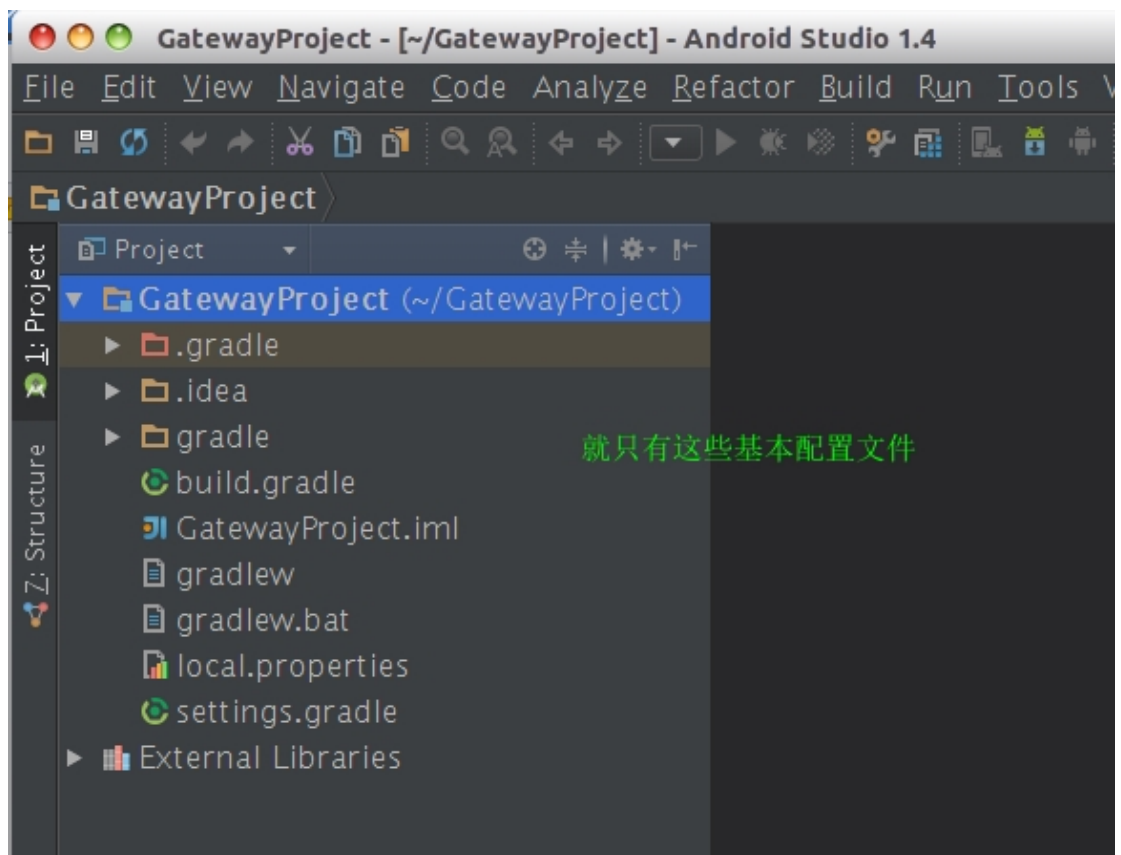
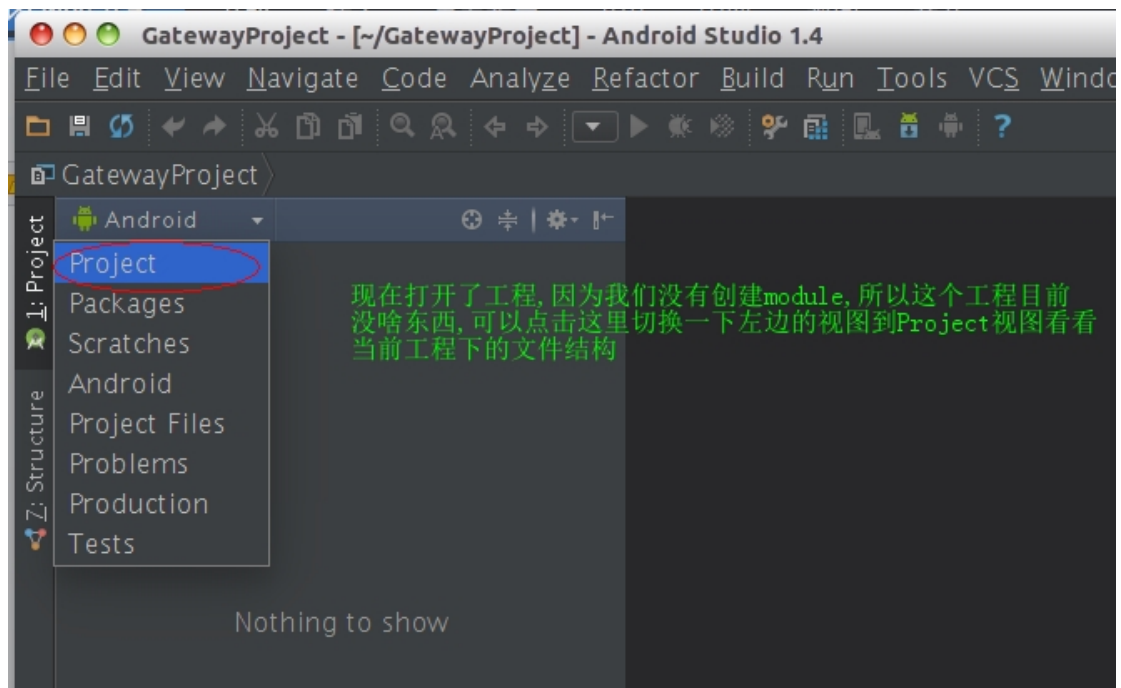
```
gradle init
gradle wrapper
```

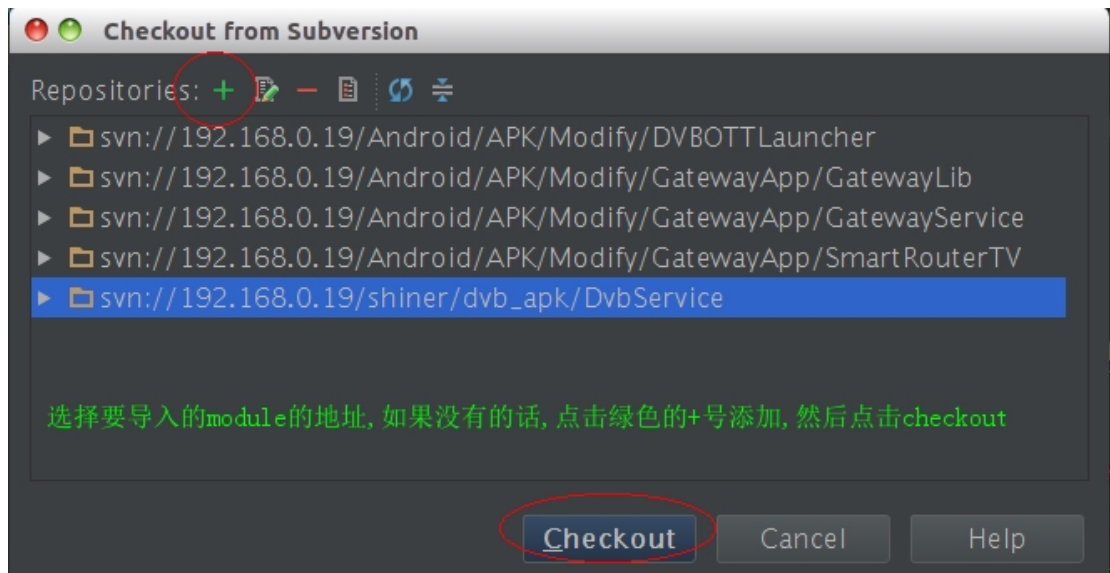
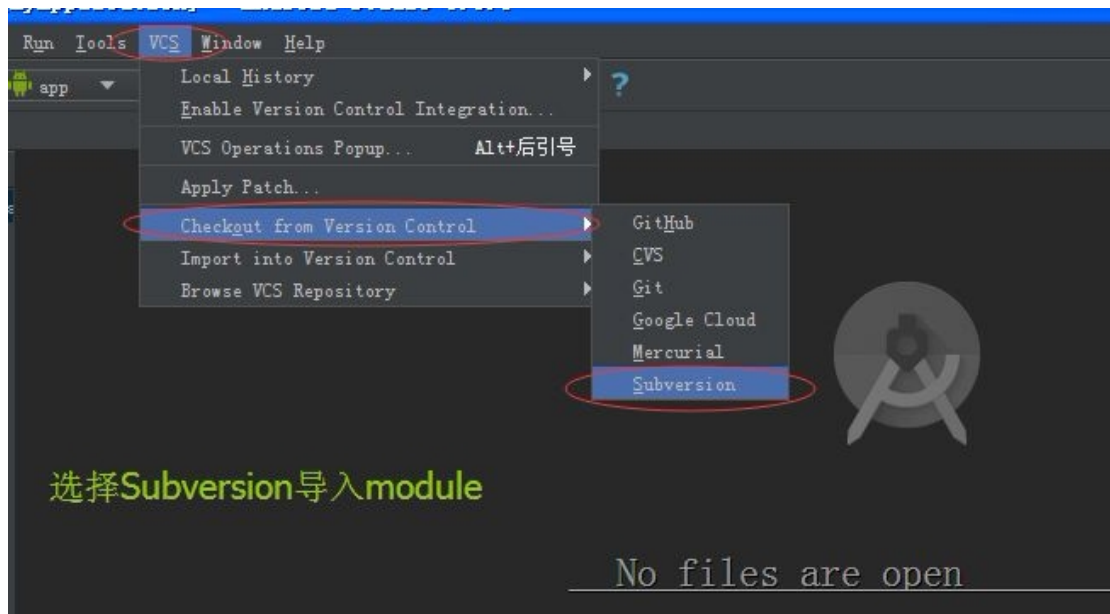
4.打开 **AS**, 然后通过 **AS** 自带的 **svn** 工具来导入 **module**



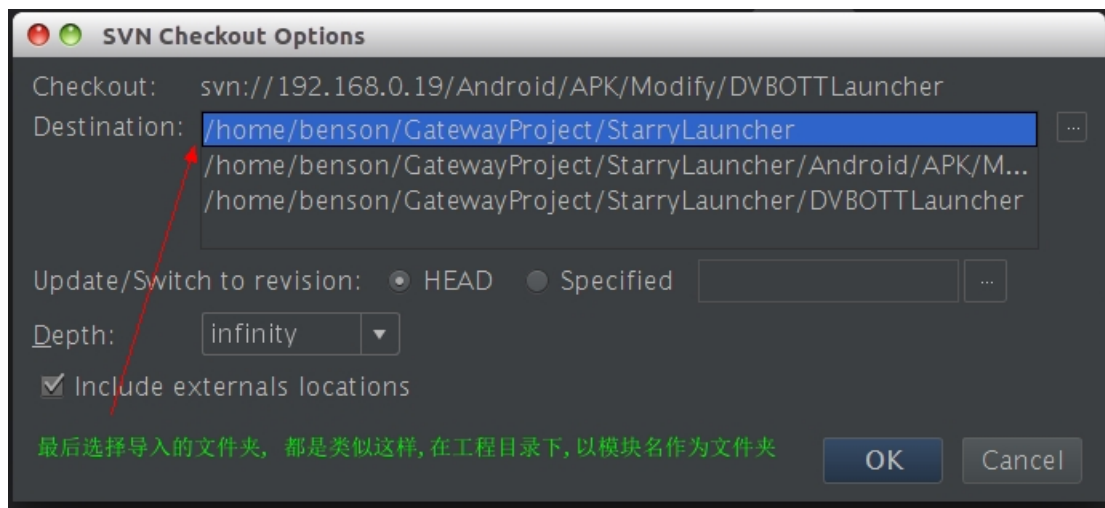
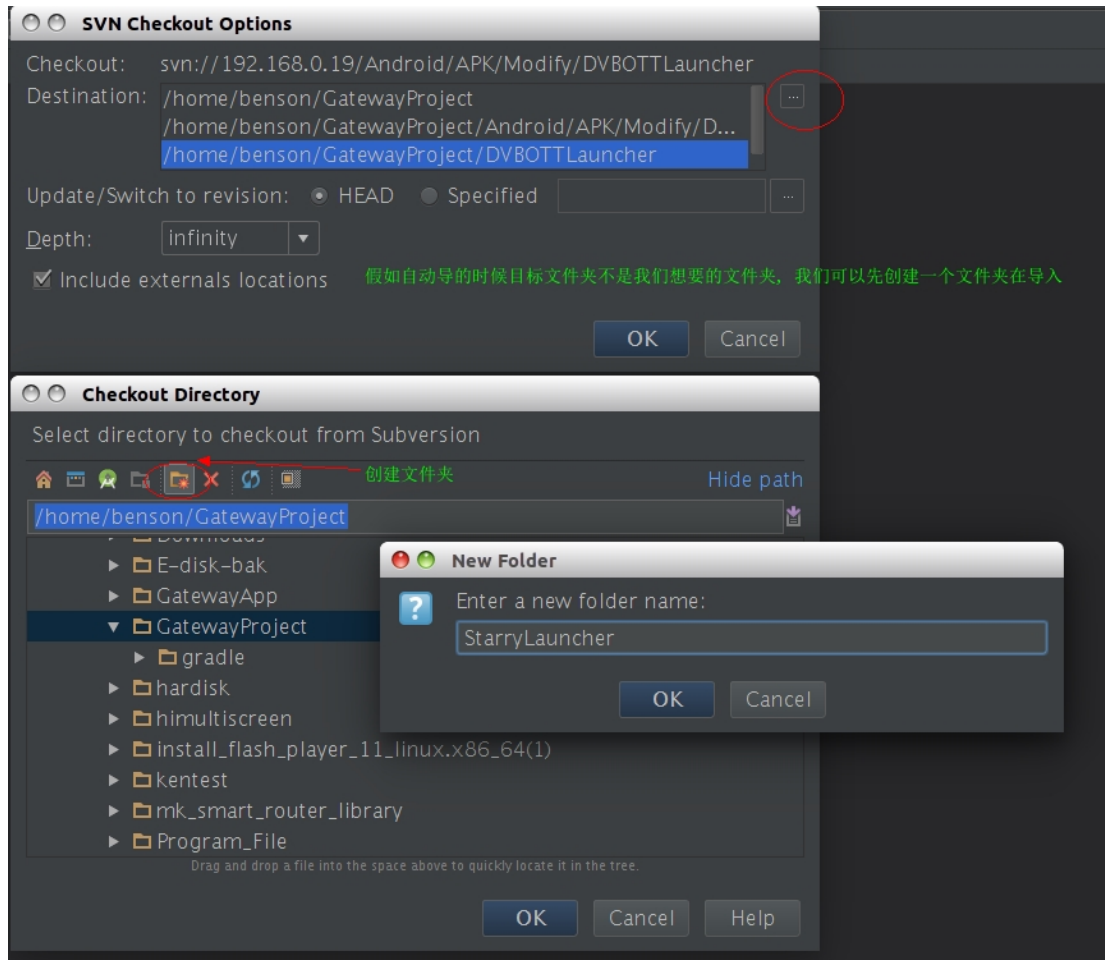
用这两项打开我们手动创建的工程都可以  
只要选择创建的工程目录就行



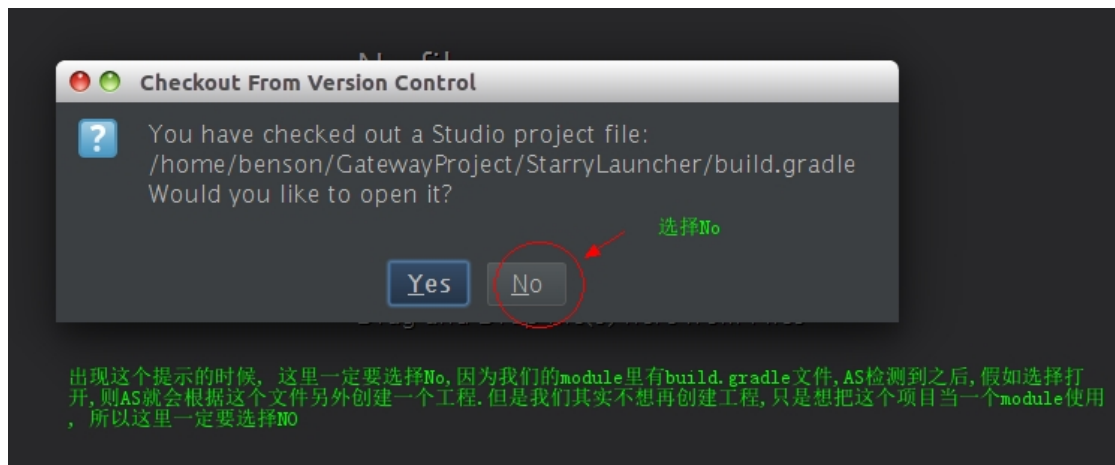
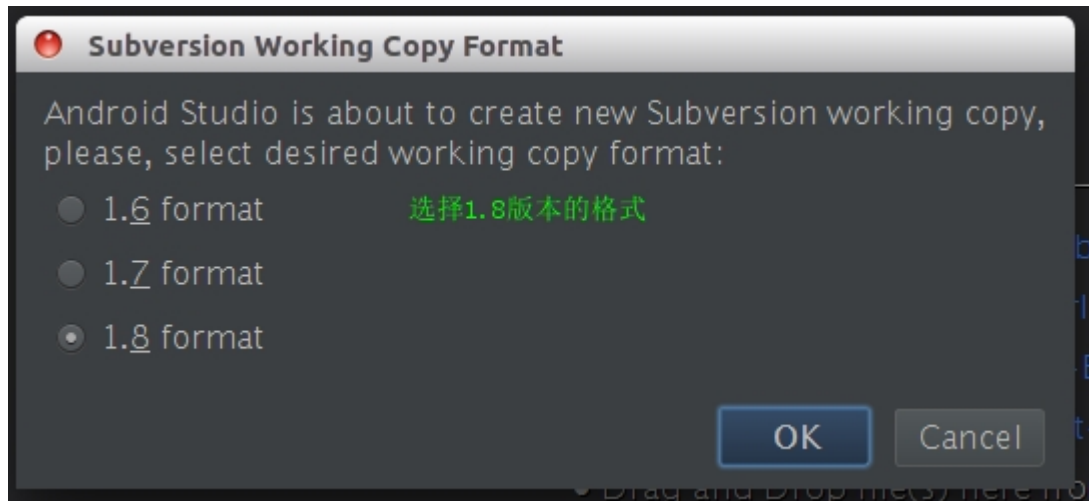




因为我们 StarryLacuner 依赖 Dvb service 和 GatewayLib, 所以我们先把这 3 个工程导下来.

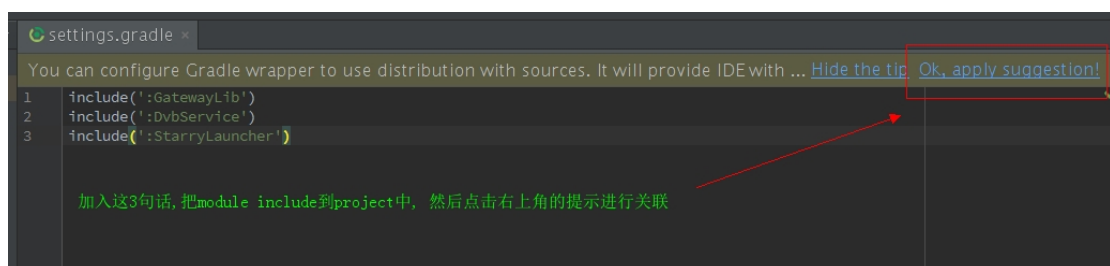
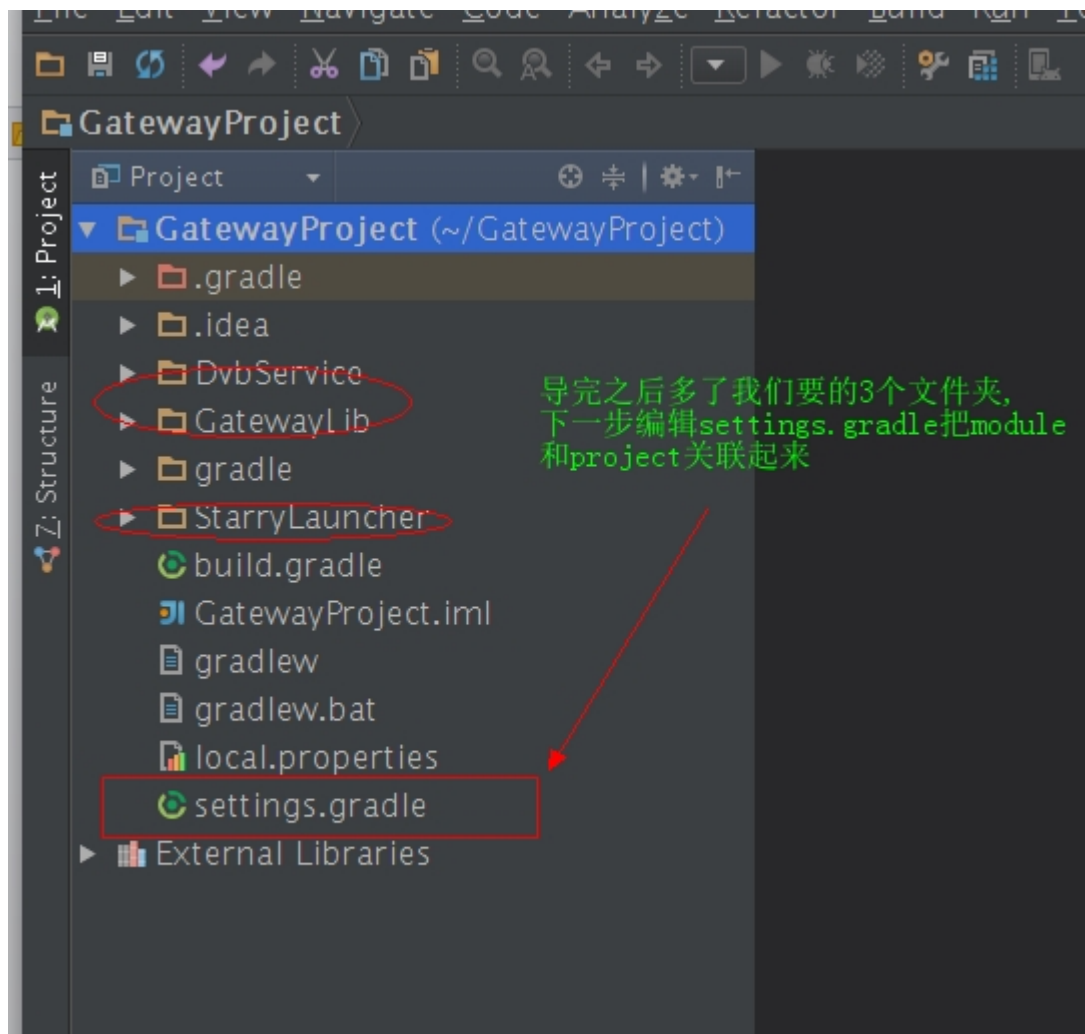


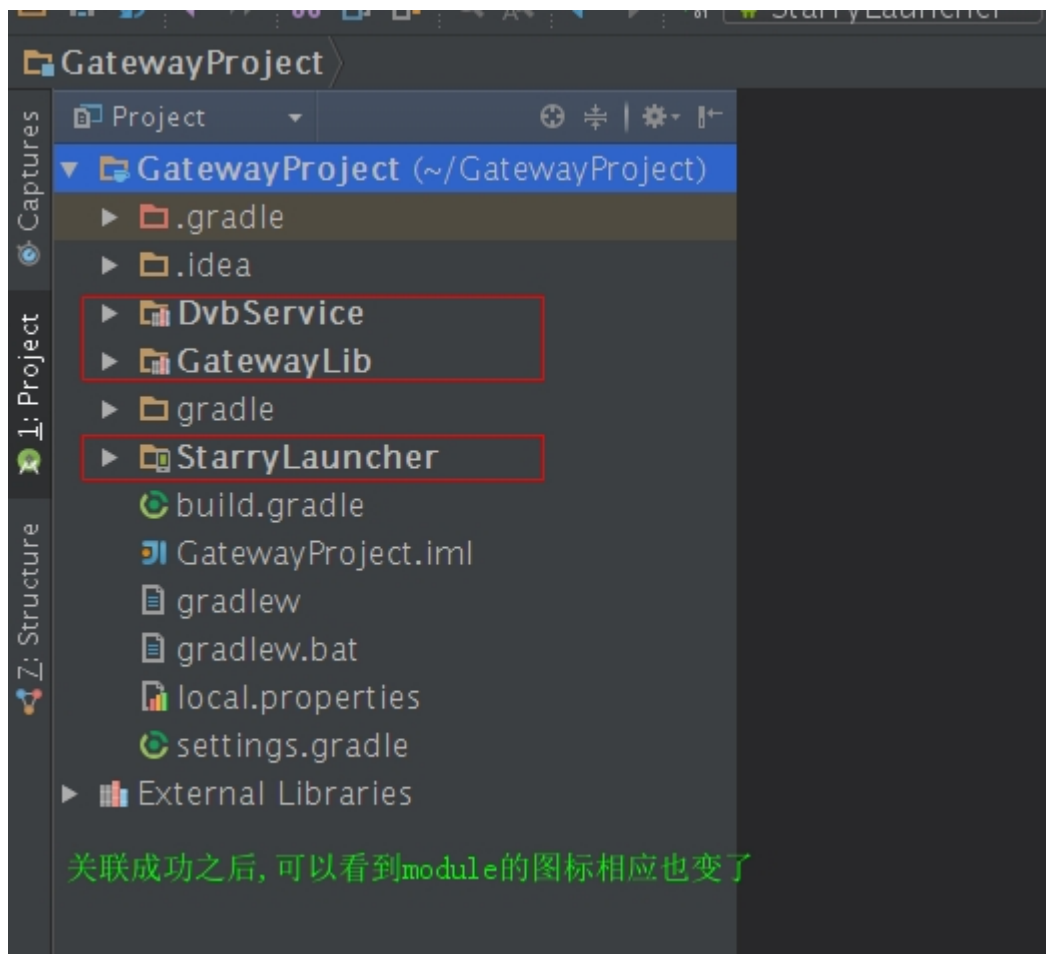




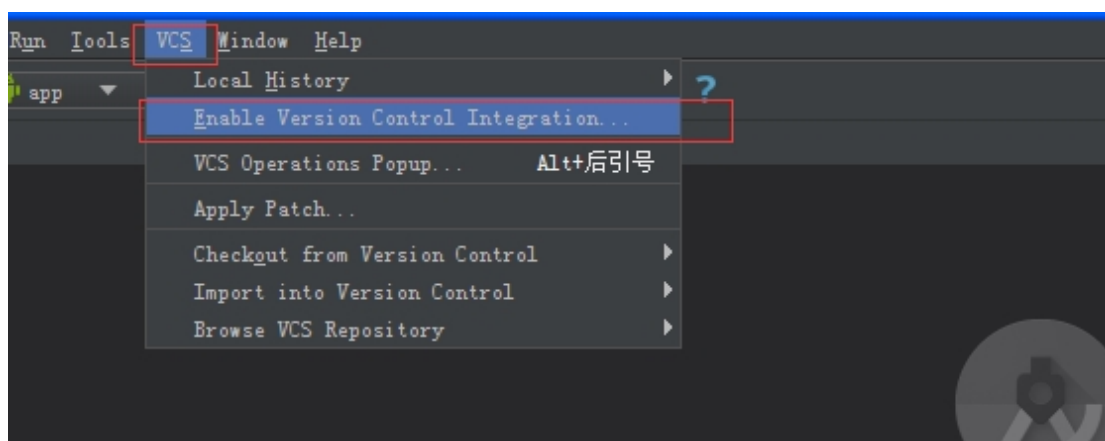
这样就完成了一个 module 的导入,大家按照导入步骤,把 GatewayLib 和 Dvbservice 都导下来.

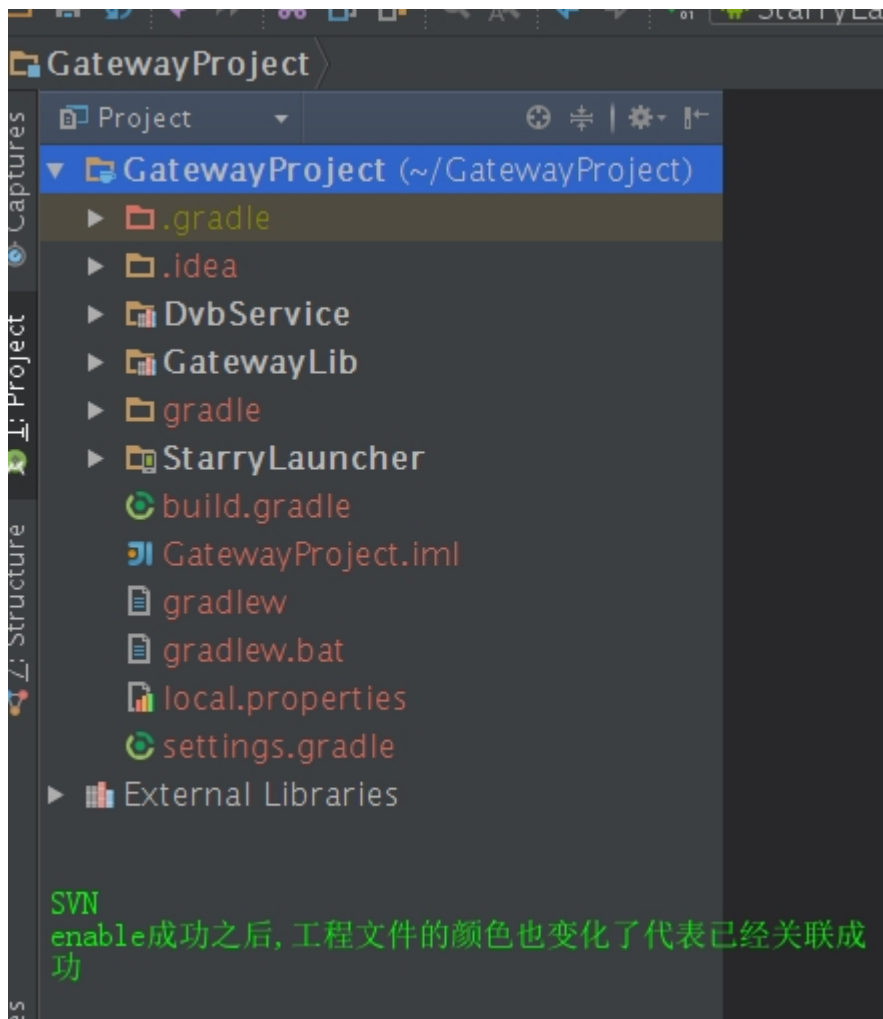
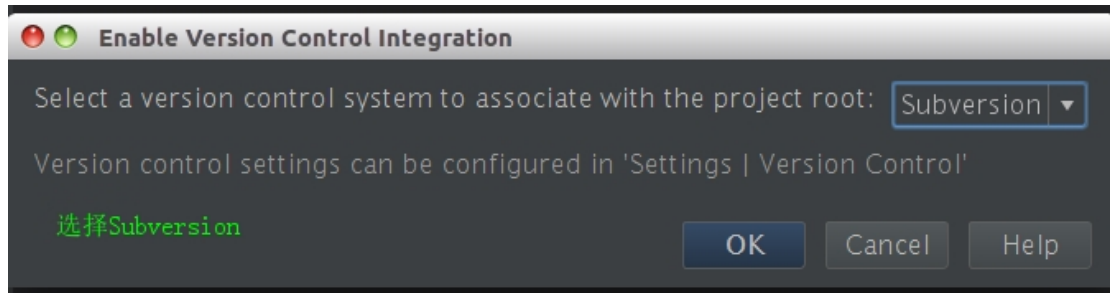






关联完之后, 基本就可以编译 apk 了. 以后要提交代码怎么办了? 所以要设置一下 SVN





后续 svn 的使用和 eclipse 里的差不多,右键文件夹选择 subversion 一栏就可以看到熟悉选项了.