

AndroidStudio使用教程(第五弹)

本文为转载文章：原文地址：<https://github.com/CharonChui/AndroidNote>

Create and Build an Android Studio Project

接下来是以下这四个部分：

- Create projects and modules. - Work with the project structure. - Edit build files to configure the build process. - Build and run your app.

关于如何创建 Project 这里就不说了，默认创建的 Project 中有一个 app 的 Module。

Add a library module

接下来的部分说一下如何在 Project 中创建一个 library module 并且把该 library 变成程序的一个依赖 module。

Create a new library module

- 点击 File 菜单后选择 New Module 或在 Project 上右键选 New Module。
- 展开页面下方的 More Modules 选择 Android Library 后 Next。
- 输入名字这里为了演示方便名字叫做 mylibrary 后一直 Next 即可。
完成之后打开该 Module 中的 build.gradle 你会看到 apply plugin: 'com.android.library' 说明这是一个 library。


Add a dependency on a library module

上一步我们创建了 mylibrary module，现在 we 想让 app module 依赖与 mylibrary module，但是构建系统还不知道，我们需要修改 app module 下的 build.gradle 文件添加 mylibrary module 就可以了。

```
...
dependencies {
    ...
    compile project(":mylibrary")
}
```

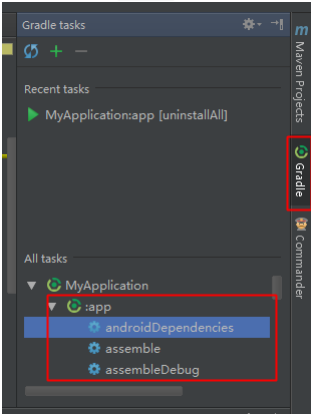
Build the project in Android Studio

Android Studio 中 build project 点击上方导航栏中的 Build 菜单然后选择 Make Project，这时窗口底部的状态栏就会显示 build 的进度。

点击窗口右边底部的  图标来显示 Gradle Console。



在窗口右边栏点击 Gradle 窗口可以看到当前所有可用的 build tasks，双击里面的 task 即可执行。



Build a release version

点击 Gradle tasks 页面，展开 app 中的 task 然后双击 assembleRelease 即可。

Configure the Build

接下来以 MyApplication Project 说明以下几个部分：

- Use the syntax from the Android plugin for Gradle in build files. - Declare dependencies. - Configure ProGuard settings. - Configure signing settings. - Work with build variants.

Build file basics

Android Studio projects 中包含一个 build file，每个 module 中也有一个 build file 名字为 build.gradle。

下面是 Project 中 app module 的 build.gradle 文件

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 19
    buildToolsVersion "21.1.1"

    defaultConfig {
        applicationId "com.charon.myapplication"
        minSdkVersion 15
        targetSdkVersion 19
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            runProguard false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile project(":mylibrary")
}
```

apply plugin: 'com.android.application' 声明了 Gradle 的类型为 Android 应用程序，这样会在最高级的 build tasks 中添加一个 Android 程序特有的 build 任务并且创建 android {} 来声明 Android 程序特殊的 build 配置。

android {} 部分配置了所有 Android 程序的 build 配置：

- compileSdkVersion 表明了编译的目标 SDK 版本。
- buildToolsVersion 声明了当前 build 的版本，可以使用 SDK Manager 来下载多个 build 版本。**注意：**最好使用高版本的 build 工具或者是和编译是目标 SDK 版本对应的 build 版本。
- defaultConfig 配置了 AndroidManifest.xml 中的重要设置。
- buildTypes 部分控制着如何去 build 和打包你的程序，默认时会定义两种 build 类型：debug 和 release。debug 类型带有默认的 debugging 标示，用 debug key 进行签名，release 版本默认时没有签名，上面的配置中 release 时没有使用 ProGuard。

dependencies 部分是在 android 之外，该部分声明了依赖的 module。

注意：当修改项目中得 build files 时，Android Studio 需要进行项目同步来导入相应的 build 配置变化， 点击 Android Studio 中黄色通知部分的 Sync Now 来进行变化的导入。

Declare dependencies

```
dependencies {  
    // Module dependency  
    compile project(":lib")  
  
    // Remote binary dependency  
    compile "com.android.support:appcompat-v7:19.0.1"  
  
    // Local binary dependency  
    compile fileTree(dir: "libs", include: ["*.jar"])  
}
```

- **Module dependency**
为本地依赖的 Module。
- **Remote binary dependency**
为远程依赖的二进制文件， 例子中为 Android SDK 仓库中所有的 support v7 包。
- **Local binary dependency**
为本地项目中依赖的 jar 包，这些 jar 包是在项目中的 libs 目录中。

Run ProGuard

构建过程中可以使用 ProGuard 进行代码混淆， 修改 build 文件中的 runProGuard 选项为 true 即可。

```
...  
android {  
    ...  
    buildTypes {  
        release {  
            runProGuard true  
            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"  
        }  
    }  
}  
...  
}
```

getDefaultProguardFile('proguard-android.txt') 包含了 Android SDK 安装时默认的 ProGuard 设置。Android Studio 在 module 的根目录中 添加了 proguard-rules.pro 文件，可以在这里配置相应的 ProGuard 规则。

Configure signing settings

debug 版本和 release 版本的应用区别在于应用程序能不能在一些稳定的设备上进行 debug 和 APK 是怎样进行签名的。构建系统对 debug 版本使用默认的签名并且使用已知的 证书以便在构建过程中不会进行代码提示。如果你不指定签名配置时构建系统不会对 release 版本进行签名。

下面是如何让程序在 release 版本进行签名

- 拷贝签名文件到 app module 的根目录。

这样就可以保证当你在其他机器上构建项目的时候可以找到你的签名文件， 如果你没有签名文件，可以先创建一个。

- 在 app module 中的 build file 中配置签名选项。

```
java ... android { ... defaultConfig { ... } signingConfigs { release { storeFile file("myreleasekey.keystore") storePassword "password" keyAlias "MyReleaseKey" keyPassword "
```

- 在 Android Studio 中的 build task 页面运行 assembleRelease。在 app/build/apk/app-release.apk 下的包现在就是使用签名文件签过名的了。
注意：把签名密码等写到 build 文件中不是很安全， 可以把密码配置到环境变量中或者是让其在构建的过程中提示输入密码。 这里我们就先不介绍如何配置了，可以自己搜索下。

至于开始所说利用 Gradle 可以很简单的进行多渠道打包会在以后专门讲解， 这里先到此为止了。

- 邮箱：charon.chui@gmail.com
- Good Luck!