Android Studo使用总结文档

[一、开发环境 2](#_Toc477354807)

[二、安装步骤 2](#_Toc477354808)

[2.1 准备安装包 2](#_Toc477354809)

[2.2 运行安装包 2](#_Toc477354810)

[三、新建工程 2](#_Toc477354811)

[3.1 android studio的常用模块类型 2](#_Toc477354812)

[3.2 新建一个android studio 工程 3](#_Toc477354813)

[3.2.1 新建一个Java Library模块 3](#_Toc477354814)

[3.2.2 新建一个Android Library模块 3](#_Toc477354815)

[四 工程目录结构 4](#_Toc477354816)

[五 jar包和aar包 5](#_Toc477354817)

[5.1 导入jar包的方法 5](#_Toc477354818)

[5.1.1 导入本地jar包 5](#_Toc477354819)

[5.1.2 导入远程jar包 5](#_Toc477354820)

[5.2 生成和使用aar包 5](#_Toc477354821)

[5.2.1 什么是aar包 5](#_Toc477354822)

[5.2.2 生成aar包 6](#_Toc477354823)

[5.2.3 导入aar包 6](#_Toc477354824)

[5.2.4 导入aar包可能报出的问题 6](#_Toc477354825)

[5.3 jar包和aar包的区别 6](#_Toc477354826)

[六 ndk环境搭建与开发流程 7](#_Toc477354827)

[6.1 NDK环境搭建 7](#_Toc477354828)

[6.2 NDK开发流程 7](#_Toc477354829)

[6.3 NDK开发中命令与配置介绍 7](#_Toc477354830)

[七 eclipse项目与android studio项目互相导入 8](#_Toc477354831)

[7.1 eclipse项目导入到android studio中 8](#_Toc477354832)

[7.2 andorid studio项目导入到eclipse项目中 8](#_Toc477354833)

[八 android studio快捷功能 8](#_Toc477354834)

[8.1 内置模板 9](#_Toc477354835)

[8.2 常用的快捷键 9](#_Toc477354836)

[九 terminal 9](#_Toc477354837)

[十 build.gradle文件解释 10](#_Toc477354838)

[十一 断点调试 10](#_Toc477354839)

[十二 maven仓库 10](#_Toc477354840)

[12.1 jcenter 和 Maven Central 11](#_Toc477354841)

[12.2 使用自定义的Maven仓库服务器 11](#_Toc477354842)

[12.3 gradle获取library的方法 12](#_Toc477354843)

# 一、开发环境

Android studio的版本2.2

Android sdk tools 的版本 25.2.2

Android platform Version: API 24:Android 7.0(Nougat) revision 2

Android ndk revision: 13.1.3345770

# 二、安装步骤

## 2.1 准备安装包

android-studio-bundle\_2.2.0.0.exe

## 2.2 运行安装包

注意android studio 和sdk所在的路径，然后一直下一步即可，傻瓜式安装，简单。

# 三、新建工程

Android studio采用的是单工程模式，一个工程中又分为许多模块（module），与eclipse相比，一个Android studio工程相当于ecplise的WorkSpace，它的一个模块相当于eclipse的一个工程。你每新建一个工程，它就重新启动一个Android studio应用。

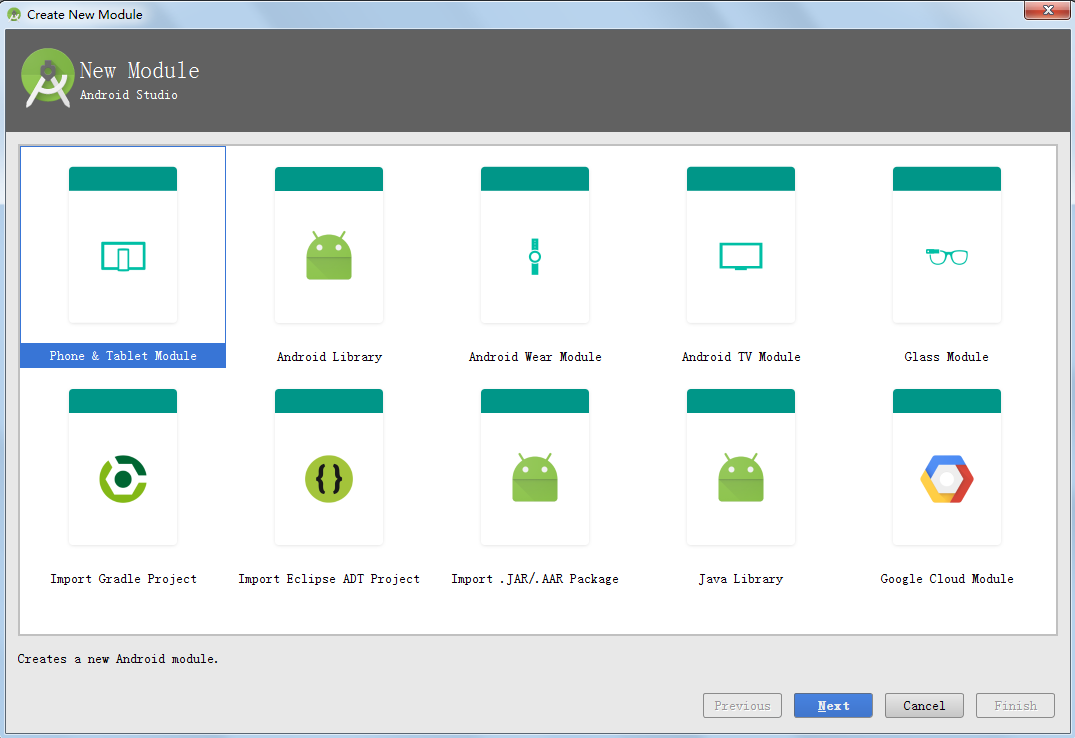
## 3.1 android studio的常用模块类型

模块类型包含有：Android Library，Java Library，Phone & Tablet Module等，如图【1】

Android Library模块：这个模块在当工程构建的时候，会生成相应的aar文件

Java Library模块：这个模块在当工程构建的时候，会生成相应的jar文件，当然它也可以单独运行，后面会讲到。

Phone &Tablet Module模块：这个模块在当工程构建的时候，会生成相应的apk文件



图【1】 module类型

## 3.2 新建一个android studio 工程

当新建一个工程，系统会默认选择Phone&Tablet Module模块，选择android sdk版本时，会有一个人性化的推荐，点击”Help me choose”,会弹出一页显示用户选择版本的分布图，我们可以根据来分析常用的sdk版本是哪一个并选择它，其余的操作和eclipse几乎一致，下一步就可以了。

### 3.2.1 新建一个Java Library模块

新建一个java模块比较简单，你可以通过点击File – new – New Module...或者右击项目根目录点击New ---- Module两种方式创建一个新的模块，接着界面会弹出所有模块的类型，这里我们选择Java Library，然后取名下一步，完成。

假设你java模块main方法所在的类名为MyClass，则你要运行这个main方法，只需要右击这个类文件，选择Run ‘MyClass.main()’即可，相应的系统也会自动的在build/libs/目录下创建相应的jar包。

### 新建一个Android Library模块

新建一个andorid library模块开始也应该创建一个Module，然后类型选择Android Library即可。在整个工程构建的时候，在build目录下会自动生成一个aar文件，这就是我们的目标文件。

# 四 工程目录结构

Android studio为我们提供了多种项目结构，不过用的最多的是Project结构类型和Android结构类型。

Android结构类型如图【2】：

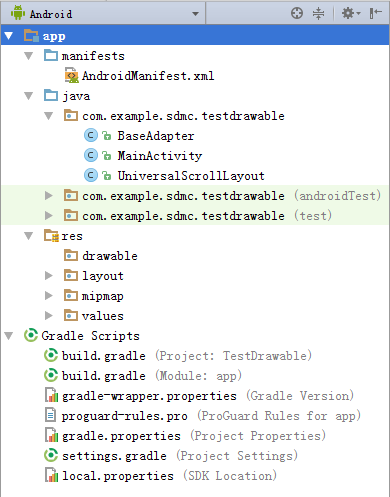


图 【2】 Android 结构类型的工程目录

Project结构类型及说明如图【3】：

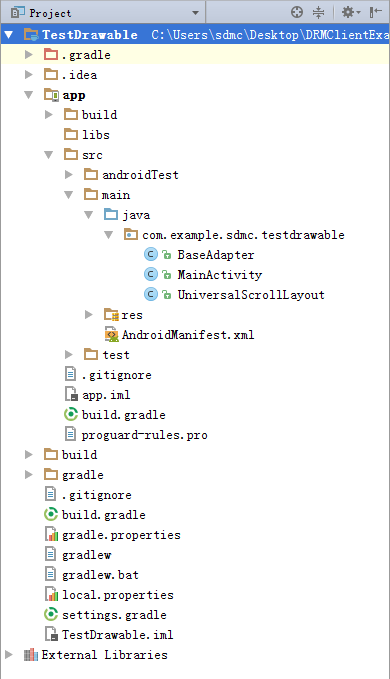


图 【3】 Project结构类型的工程目录及说明

# 五 jar包和aar包

## 5.1 导入jar包的方法

### 5.1.1 导入本地jar包

先将本地jar包放入该模块下的libs目录下，然后如下操作即可：

右键项目工程根目录 ---- 点击open module settings ----- 切换到dependences ----- 点击右上角的加号并选择File dependency ------- 选择相应的jar包

或者如果已知jar名，可以通过修改app目录下的build.gradle中的属性导入

### 5.1.2 导入远程jar包

这里以maven为例，当然也可以使用其他类型的仓库，例如Ivy。，如下：

complie ‘com.alibaba:fastjson:lastest.integration’

或者是按照导入本地jar包的方法，但是在点击右上角的加号后应该选择Library dependency，然后输入关键字过滤出相应的远程jar包，导入即可，然后在扩展包下就可以看见相应的jar包了。

详情见示例： [示例\导入远程jar包.flv](示例/导入远程jar包.flv)

## 5.2 生成和使用aar包

### 5.2.1 什么是aar包

aar（Android Archive）包是一个Android库项目的二进制归档文件。

文件扩展名是.aar，但文件本身是具有以下条目的一个简单zip文件：

/AndroidManifest.xml (强制)

/classes.jar (强制)

/res/ (强制)

/R.txt (强制)

/assets/ (可选)

/libs/\*.jar (可选)

/jni/<abi>/\*.so (可选)

/proguard.txt (可选)

/lint.jar (可选)

### 5.2.2 生成aar包

在一个工程中新建一个模块（Module），模块类型为Android Library，然后将点击build菜单下的Rebuid Project重新构建工程，在新建模块的build/outputs/arr目录下就是所生成的aar文件。

### 5.2.3 导入aar包

①当aar包位于本地目录：

将aar文件放到要使用该文件的模块的libs目录下，然后再该模块的build.gradle文件中加入以下配置：

在android节点中加入：

repositories { flatDir { dirs. ‘libs’ } }

在dependences属性框中加入：

complie（name：’myapplib-debug‘，ext：‘aar‘）

注：这里name后面跟的是aar包的名字，这里仅作参考。

②当aar位于远程仓库时

这里以maven为例，当然也可以使用其他类型的仓库，例如Ivy。只需要在jar包引用方式后面添加一个@aar就可以了，如下：

complie ‘com.alibaba:fastjson:lastest.integration@aar’

### 5.2.4 导入aar包可能报出的问题

若你的项目在导入aar包报出一下异常：

Java.lang.RuntimeException:Manifest merger failed:uses-sdk:minSdkVersion 1 cannot be smaller than version 4 declared in library XXX

这是因为引入的aar包的build.gradle文件中的defaultConfig节点中的信息与导入该包的build.gradle文件的defaultConfig节点中的信息不一致所致，因此有两种解决方法：

①最直接的，将两个build.gradle文件中的defaultConfig框中的信息改成一致

②将出问题的库的Mainfest文件中加入：

<uses-sdk tools：overrideLibrary=“该库的包名”>

以上两种方法均可解决这种问题，改完后记得重新构建，要不是不起作用的！

## 5.3 jar包和aar包的区别

jar包只包含class文件与清单文件，不包含资源文件，如图片等所有res中的文件

aar包包含jar包和资源文件。

# 六 ndk环境搭建与开发流程

## 6.1 NDK环境搭建

打开SDK Manager ，把TAB页切换到SDK Tools页，选择NDK，然后点击Apply，开始下载ndk相关的文件，这样就可以了。

## 6.2 NDK开发流程

先在某个类中声明本地方法，然后在终端先进入到项目的根目录下，使用javah命令生成头文件，自己实现源文件，然后编写一份Android.mk文件，最后在项目的跟木下运行ndk-build命令即可。使用的时候，把生成的so库放入到该模块下的libs文件夹下，要在该模块下的build.gradle文件中进行相应的配置，在代码的静态代码块中调用System的loadLibrary方法，最后编译运行整个程序，完毕！

## 6.3 NDK开发中命令与配置介绍

**Javah命令：**

javah –d 文件输入的目录 –classpath 要生成的头文件所在类的class文件的目录

本地方法所在类的全类名

**可能出现的问题：**

无法访问android.app.Activity，找不到android.app.Activity的类文件

这是因为找不到adnroid sdk中的activity类，因此要在-classpath后面加入android.jar的全路径名，比如F:/AS/sdk/platforms/android-24/android.jar;.

提供一个完整的javah命令：

javah -d jni -classpath F:/AS/sdk/platforms/android-24/android.jar;

./app/build/intermediates/classes/debug com.example.sdmc.ndk\_test.MainActivity

具体的javah参数，可以自己去了解，这里只讲用到的。

**Android.mk文件编写：**

Android.mk文件名字必须这样写，大小写也不能错，空格也不能错，把该文件放在生成的头文件所在的目录下，该文件的内容如下：

LOCAL\_PATH := $(call my-dir)   
include $(CLEAR\_VARS)  
LOCAL\_MODULE := demo //要生成的so库名字，实际上为libdemo.so  
LOCAL\_SRC\_FILES := demo.c //要使用的源文件  
include $(BUILD\_SHARED\_LIBRARY)

在build.gradle中的配置：

在andorid节点中加入：

SourceSets { main { jniLibs.srcDirs = [‘libs’] } }

# 七 eclipse项目与android studio项目互相导入

## 7.1 eclipse项目导入到android studio中

ecplise项目导入到andorid studio中比较简单，应为android studio本身就兼容ecplise的项目结构，可以直接导入，导入后工程的结构将被转化成android studio的项目结构，因此导入前最好做好备份工作。

## 7.2 andorid studio项目导入到eclipse项目中

android studio项目导入到ecplise中就相对比较复杂，因为eclipse没有andorid studio强大，它并不能兼容android studio的项目结构，因此要将项目导入到eclipse中，就必须将android

studio的项目结构转换成ecplise的项目结构，上面的项目结构分析中，源码基本都在模块下src/main目录下，根据这一点就可以将相应的源码重新组合起来，构建成一个ecplise项目结构，当然相应的参照，也必须要我们自己手动进行导入，构建完之后，然后按照正常地导入ecplise工程导入即可，具体问题还是要具体去修改！

具体做法可参考一下两篇文章：

<http://jcodecraeer.com/a/anzhuokaifa/androidkaifa/2014/0918/1692.html> <http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/0110/2294.html#commettop>。

# 八 android studio快捷功能

Android studio里面的快捷键适配了其他开发环境的快捷键，在File菜单的Settings的Keymap中的Keymaps下拉框中可以选择其他开发环境的快捷键方式，其中就包含了Eclipse。因此之前使用Eclipse的小伙伴们就不必因为快捷键不适应而放弃学习androdi studio了。

关于快捷键介绍，可阅读如下文章：

<http://www.open-open.com/lib/view/open1483064098297.html>

## 8.1 内置模板

如果在以往的程序中敲入以下代码：

logt，logm，logi，loge，psm，gone，sout等

你的开发工具肯定会报错，然而在android studio中它们都会对应下面的代码：

logt：快速生成TAG，即private static final String TAG = “类名”

logm：快速生成Log.d(TAG,”方法名() call with:方法参数”)

logi: 快速生成Log.i(TAG,”方法名:’自己输入’”);

loge: 同logi；

psm：这个是快速生成main方法private static main（…）{}

gone：.setVisibility(View.GONE)

sout：System.out.println();

可以根据这样的规律，快速编写代码!

## 8.2 常用的快捷键

Ctrl+X，Ctrl+D：快速剪切行和快速删除行

ALT+上键，ALT+下键，Ctrl+ALT+上键，Ctrl+ALT+下键：

将一行代码上移，下移，往上一行复制，往下一行复制。

ALT+/：快速弹出智能提示框

ALT+ENTER：作用就很多了，可用于导包，类型转换，添加抽象方法，快速生成变量等在程序出错的时候，我就会先敲一下这个快捷键，一般都能够解决

Ctrl+加号或减号：快速展开和折叠代码块

Ctil+Shift+i：可用于预览方法体，不会跳入相应的类中

右击项目标签，选择split vertical\horizontal,可以拆分窗口

Ctrl+o：查看大纲

Ctrl+Shift+M：将一块代码包装成一个方法

Ctrl+alt+T：将一块代码用条件、循环、捕捉异常，Runnable等包裹。

请看示例视频：[示例\快捷键演示.flv](示例/快捷键演示.flv)

# 九 terminal

Android studio内置的终端与windows的cmd终端功能差不多，我们可以通过android studio的终端执行adb命令，这样不用通过串口也能够向机顶盒抛apk文件，还可以查看logcat，不过不能够删除system/app目录下的文件，显示没有权限,不过执行一下su root命令，就可以了。

adb的命令大全可参考下面的文章：

<http://blog.csdn.net/janronehoo/article/details/6863772>

adb连接同一网络下的android设备，可参考下面的文章：

<http://blog.csdn.net/lb377463323/article/details/47334289>

# 十 build.gradle文件解释

见示例文件build.gradle

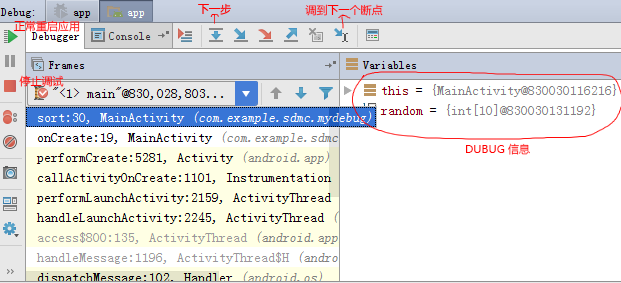
详细可参考如下文章：

<http://blog.csdn.net/u012246458/article/details/51722624>

# 十一 断点调试

断点调试是指自己在程序的某一行设置一个断点，调试时，程序运行到这一行就会停住，然后你可以一步一步往下调试，调试过程中可以看各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下。

断点调试的控制栏如图【4】



图【4】 调试控制栏

具体操作，请看示例视频：[示例\断点调试.flv](示例/断点调试.flv)

# 十二 maven仓库

详情可阅读以下文章：

<http://www.cnblogs.com/sihaixuan/p/4852974.html>

在build.gradle的里面写一条配置语句，一个library包就可以正常使用了，那么Android studio是从哪里得到这些library的？

Android Studio是从build.gradle里面定义的Maven 仓库服务器上下载library的。Apache Maven是Apache开发的一个工具，提供了用于贡献library的文件服务器。总的来说，只有两个标准的Android library文件服务器：jcenter 和  Maven Central。

## 12.1 jcenter 和 Maven Central

jcenter是一个由 bintray.com维护的Maven仓库 。你可以在[这里](http://jcenter.bintray.com/)看到整个仓库的内容；Maven Central 则是由[sonatype.org](https://sonatype.org/)维护的Maven仓库。你可以在[这里](https://oss.sonatype.org/content/repositories/releases/)看到整个仓库。

我们在项目的build.gradle 文件中如下定义仓库，就能使用jcenter/mavenCentral：

allprojects { repositories { jcenter()或mavenCentral } }

注意，虽然jcenter和Maven Central 都是标准的 android library仓库，但是它们维护在完全不同的服务器上，由不同的人提供内容，两者之间毫无关系。在jcenter上有的可能 Maven Central 上没有，反之亦然。

事实上两个仓库都具有相同的使命：提供Java或者Android library服务。上传到哪个（或者都上传）取决于开发者。

起初，Android Studio 选择Maven Central作为默认仓库。如果你使用老版本的Android Studio创建一个新项目，mavenCentral()会自动的定义在build.gradle中。

但是Maven Central的最大问题是对开发者不够友好。上传library异常困难。上传上去的开发者都是某种程度的极客。同时还因为诸如安全方面的其他原因，Android Studio团队决定把默认的仓库替换成jcenter。正如你看到的，一旦使用最新版本的Android Studio创建一个项目，jcenter()自动被定义，而不是mavenCentral()。

有许多将Maven Central替换成jcenter的理由，下面是几个主要的原因。

- jcenter通过CDN发送library，开发者可以享受到更快的下载体验。

- jcenter是全世界最大的Java仓库，因此在Maven Central 上有的，在jcenter上也极有可能有。换句话说jcenter是Maven Central的超集。

- 上传library到仓库很简单，不需要像在 Maven Central上做很多复杂的事情。

- 友好的用户界面

- 如果你想把library上传到 Maven Central ，你可以在bintray网站上直接点击一个按钮就能实现。

## 12.2 使用自定义的Maven仓库服务器

除了两个标准的服务器之外，如果我们使用的library的作者是把该library放在自己的服务器上，我们还可以自己定义特有的Maven仓库服务器。Twitter的Fabric.io 就是这种情况，它们在<https://maven.fabric.io/public>上维护了一个自己的Maven仓库。如果你想使用Fabric.io的library，你必须自己如下定义仓库的url。

repositories {

maven { url 'https://maven.fabric.io/public' }

}

然后在里面使用相同的方法获取一个library。

dependencies {

compile 'com.crashlytics.sdk.android:crashlytics:2.2.4@aar'

}

## 12.3 gradle获取library的方法

在讨论如何上传library到jcenter之前，我们先看看gradle是如何从仓库获取library的。比如我们在 build.gradle输入如下代码的时候，这些库是如果奇迹般下载到我们的项目中的。

compile 'com.inthecheesefactory.thecheeselibrary:fb-like:0.9.3'

一般来说，我们需要知道library的字符串形式，包含3部分

GROUP\_ID:ARTIFACT\_ID:VERSION

上面的例子中，GROUP\_ID是com.inthecheesefactory.thecheeselibrary ，ARTIFACT\_ID是fb-like，VERSION是0.9.3。

GROUP\_ID定义了library的group。有可能在同样的上下文中存在多个不同功能的library。如果library具有相同的group，那么它们将共享一个GROUP\_ID。通常我们以开发者包名紧跟着library的group名称来命名，比如com.squareup.picasso。然后ARTIFACT\_ID中是library的真实名称。至于VERSION，就是版本号而已，虽然可以是任意文字，但是我建议设置为x.y.z的形式，如果喜欢还可以加上beta这样的后缀。

下面是Square library的一个例子。你可以看到每个都可以很容易的分辨出library和开发者的名称。

dependencies {

compile 'com.squareup:otto:1.3.7'

compile 'com.squareup.picasso:picasso:2.5.2'

compile 'com.squareup.okhttp:okhttp:2.4.0'

compile 'com.squareup.retrofit:retrofit:1.9.0'

}

那么在添加了上面的依赖之后会发生什么呢？简单。Gradle会询问Maven仓库服务器这个library是否存在，如果是，gradle会获得请求library的路径，一般这个路径都是这样的形式：GROUP\_ID/ARTIFACT\_ID/VERSION\_ID。

比如可以在<http://jcenter.bintray.com/com/squareup/otto/1.3.7> 和

<https://oss.sonatype.org/content/repositories/releases/com/squareup/otto/1.3.7/>

下获得com.squareup:otto:1.3.7的library文件。

然后Android Studio 将下载这些文件到我们的电脑上，与我们的项目一起编译。整个过程就是这么简单，一点都不复杂。

我相信你应该清楚的知道从仓库上下载的library只是存储在仓库服务器上的jar 或者aar文件而已。有点类似于自己去下载这些文件，拷贝然后和项目一起编译。但是使用gradle依赖管理的最大好处是你除了添加几行文字之外啥也不做。library一下子就可以在项目中使用了。