# IDE

## 基本设置：

## Jdk

## Sdk：

## Android命名规范

包命名组织方式。

Impl 接口实现

Ible 接口

\*\*Provider

|  |  |
| --- | --- |
| com.example.app.activitys | 用来组织Activity类 |
| com.example.app.base | 基础共享的类，如多个Activity共享的 BaseActivity或整个应用共享的MyApplication类 |
| com.example.app.adapter | 项目中用到的适配器类 |
| com.example.app.view | 自定义的View，如常用的TitleBarView |
| com.example.app.util | 工具类，如HttpUtil，ImageUtil，FileUtil |
| com.example.app.db | 数据库类，如DataBaseHelper，MessageDB |
| com.example.app.service | 服务类，如GetMsgService |
| com.example.app.constant | 常量类 |
| com.example.app.domain/modle/entity | 元素实体类，如对应注册用户信息的User类， 对应聊天信息的TextMessage类 |
| com.example.app.broadcast | 广播服务类 |

**标识符命名法**  
标识符命名法最要有四种:  
  
1 驼峰(Camel)命名法:又称小驼峰命名法，除首单词外，其余所有单词的第一个字母大写。  
  
2 帕斯卡(pascal)命名法:又称大驼峰命名法，所有单词的第一个字母大写  
  
3 下划线命名法:单词与单词间用下划线做间隔。  
  
4 匈牙利命名法:广泛应用于微软编程环境中，在以Pascal命名法的变量前附加小写序列说明该变量的类型。 量的取名方式为：<scope\_> + <prefix\_> + <qualifier>范围前缀，类型前缀，限定词。  
  
个人觉得标识符命名原则:**尽可能的用最少的字符而又能完整的表达标识符的含义。**  
  
英文缩写原则：  
1 较短的单词可通过去掉“元音”形成缩写  
2 较长的单词可取单词的头几个字母形成缩写  
3 此外还有一些约定成俗的英文单词缩写.  
  
下面为常见的英文单词缩写:

|  |  |
| --- | --- |
| **名称** | **缩写** |
| icon | ic （主要用在app的图标） |
| color | cl（主要用于颜色值） |
| divider | di（主要用于分隔线，不仅包括Listview中的divider，还包括普通布局中的线） |
| selector | sl（主要用于某一view多种状态，不仅包括Listview中的selector，还包括按钮的selector） |
| average | avg |
| background | Bg（主要用于布局和子布局的背景） |
| buffer | buf |
| control | ctrl |
| delete | del |
| document | doc |
| error | err |
| escape | esc |
| increment | inc |
| infomation | info |
| initial | init |
| image | img |
| Internationalization | I18N |
| length | len |
| library | lib |
| message | msg |
| password- | pwd |
| position | pos |
| server | srv |
| string | str |
| temp | tmp |
| window | wnd(win) |

程序中使用单词缩写原则：**不要用缩写，除非该缩写是约定俗成的。**

命名规范：  
  
1  包（packages）: 采用反域名命名规则，全部使用小写字母。一级包名为com，二级包名为xx（可以是公司或则个人的随便），三级包名根据应用进行命名，四级包名为模块名或层级名

|  |  |
| --- | --- |
| **包名** | **此包中包含** |
| com.xx.应用名称缩写.activities | 页面用到的Activity类 (activities层级名用户界面层) |
| com.xx.应用名称缩写.base | 页面中每个Activity类共享的可以写成一个i额BaseActivity类 (基础共享的类) |
| com.xx.应用名称缩写.adapter | 页面用到的Adapter类 (适配器的类) |
| com.xx.应用名称缩写.tools | 此包中包含：公共工具方法类（tools模块名） |
| com.xx.应用名称缩写.bean  (或则 com.xx.应用名称缩写.unity ) | 此包中包含：元素类 |
| com.xx.应用名称缩写.db | 数据库操作类 |
| com.xx.应用名称缩写.view  (或则 com.xx.应用名称缩写.ui ) | 自定义的View类等 |
| com.xx.应用名称缩写.service | Service服务 |
| com.xx.应用名称缩写.broadcast | Broadcast服务 |

2  类（classes）：名词，采用大驼峰命名法，尽量避免缩写，除非该缩写是众所周知的，  比如HTML,URL，如果类名称中包含单词缩写，则单词缩写的每个字母均应大写。

|  |  |  |
| --- | --- | --- |
| **类** | **描述** | **例如** |
| activity 类 | Aty或者Activity为后缀标识 | 欢迎页面类WelcomeAty.或者WelcomeActivity |
| Adapter类 | Adp或者Adapte 为后缀标识 | 新闻详情适配器NewtDetailAdp或则直接  NewDetailAdapter |
| 解析类 | Hlr为后缀标识 | 首页解析类HomePosterHlr |
| 公共方法类 | Tools或Manager为后缀标识 | 线程池管理类：ThreadPoolManager             日志工具类：LogTools |
| 数据库类 | 以DBHelper后缀标识 | 新闻数据库：NewDBHelper |
| Service类 | 以Service为后缀标识 | 时间服务TimeService |
| BroadcastReceive类 | 以Broadcast为后缀标识 | 时间通知TimeBroadcast |
| ContentProvider | 以Provider为后缀标识 |  |
| 直接写的共享基础类 | 以Base开头 | BaseActivity,BaseFragment |

3  接口（interface）：命名规则与类一样采用大驼峰命名法，多以able或ible结尾，如interface Runna ble ;  
                     interface Accessible 。  
  
4  方法（methods）：动词或动名词，采用小驼峰命名法例如:onCreate(),run()

|  |  |
| --- | --- |
| **方法** | **说明** |
| initXX() | 初始化相关方法,使用init为前缀标识，如初始化布局initView() |
| isXX() | checkXX()方法返回值为boolean型的请使用is或check为前缀标识 |
| getXX() | 返回某个值的方法，使用get为前缀标识 |
| processXX() | 对数据进行处理的方法，尽量使用process为前缀标识 |
| displayXX() | 弹出提示框和提示信息，使用display为前缀标识 |
| saveXX() | 与保存数据相关的，使用sav为e前缀标识 |
| resetXX() | 对数据重组的，使用reset前缀标识 |
| clearXX() | 清除数据相关的 |
| removeXXX() | 清除数据相关的 |
| drawXXX() | 绘制数据或效果相关的，使用draw前缀标识 |

5  变量（variables）采用小驼峰命名法。类中控件名称必须与xml布局id保持一致。  
  
用统一的量词通过在结尾处放置一个量词，就可创建更加统一的变量，它们更容易理解，也更容易搜索。例如，请使用strCustomerFirst和strCustomerLast，而不要使用strFirstCustomer和strLastCustomer。  
量词列表：量词后缀说明  
First  一组变量中的第一个  
Last   一组变量中的最后一个  
Next   一组变量中的下一个变量  
Prev   一组变量中的上一个  
Cur    一组变量中的当前变量  
   
6  常量（Constants）全部大写,采用下划线命名法.例如：MIN\_WIDTH  
  
  
7  资源文件（图片drawable文件夹下）：全部小写，采用下划线命名法，加前缀区分

命名模式：activity名称\_逻辑名称/common\_逻辑名称

如果有多种形态如按钮等除外如**btn\_xx.xml（selector）**

|  |  |
| --- | --- |
| **名称** | **功能** |
| btn\_xx | 按钮图片使用btn\_整体效果**（selector）** |
| btn\_xx\_normal | 按钮图片使用btn\_正常情况效果 |
| btn\_xx\_press | 按钮图片使用btn\_点击时候效果 |
| bg\_head | 背景图片使用bg\_功能\_说明 |
| def\_search\_cell | 默认图片使用def\_功能\_说明 |
| icon\_more\_help | 图标图片使用icon\_功能\_说明 |
| seg\_list\_line | 具有分隔特征的图片使用seg\_功能\_说明 |
| sel\_ok | 选择图标使用sel\_功能\_说明 |

命名后缀：

|  |  |
| --- | --- |
| **后缀** | **说明** |
| unit | 在使用xml的tilemode来配图片时，element图片使用此后缀 |
| nor | 图片的状态，代表普通状态 |
| hl | 图片的状态，代表高亮状态 |
| press | 图片的状态，代表按下状态 |
| select | 图片的状态，代表其所占的view被选中 |
| unselect | 图片的状态，代表其所占的view没有被选中 |

8     资源布局文件（XML文件（layout布局文件））：

 全部小写，采用下划线命名法

1)．contentview命名, Activity默认布局，以去掉后缀的Activity类进行命名。不加后缀：

**功能模块.xml**

       例如：main.xml、more.xml、settings.xml

        或则：activity\_功能模块.xml

        例如：**activity\_main.xml、activity\_more.xml**

2)．Dialog命名：dialog\_描述.xml

       例如：**dlg\_hint.xml**

2)．PopupWindow命名：ppw\_描述.xml

       例如：**ppw \_info.xml**

3). 列表项命名listitem\_描述.xml

       例如：**listitem\_city.xml**

4)．包含项：include\_模块.xml

       例如：**include\_head.xml、include\_bottom.xml**

5)．adapter的子布局：**功能模块**\_item.xml

       例如：**main\_item.xml、**

9   动画文件（anim文件夹下）：全部小写，采用下划线命名法，加前缀区分。

//前面为动画的类型，后面为方向

|  |  |  |
| --- | --- | --- |
| **动画命名例子** | **规范写法** | **备注** |
| fade\_in | 淡入 |  |
| fade\_out | 淡出 |  |
| push\_down\_in | 从下方推入 |  |
| push\_down\_out | 从下方推出 |  |
| push\_left | 推像左方 |  |
| slide\_in\_from\_top | 从头部滑动进入 |  |
| zoom\_enter | 变形进入 |  |
| slide\_in | 滑动进入 |  |
| shrink\_to\_middle | 中间缩小 |  |

10    资源ID（resourcesid）：大小写规范与方法名一致，采用小驼峰命名法。命名规范为“资源控件的缩写  名”+“变量名”。注意：**页面控件名称应该和控件id名保持一致**

**strings.xml，colors.xml等中的id命名：**

命名模式：activity名称\_功能模块名称\_逻辑名称/activity名称\_逻辑名称/common\_逻辑名称

strings.xml中，使用activity名称注释，将文件内容区分开来

11   layout中的id命名

命名模式为：view缩写\_模块名称\_view的逻辑名称

view的缩写详情如下：

|  |  |
| --- | --- |
| **控件** | **缩写** |
| LayoutView | lv |
| RelativeView | rv |
| TextView | tv |
| Button | btn |
| ImageButton | imgBtn |
| ImageView | mgView 或则 iv |
| CheckBox | chk |
| RadioButton | rdoBtn |
| analogClock | anaClk |
| DigtalClock | dgtClk |
| DatePicker | dtPk |
| EditText | edtTxt |
| TimePicker | tmPk |
| toggleButton | tglBtn |
| ProgressBar | proBar |
| SeekBar | skBar |
| AutoCompleteTextView | autoTxt |
| ZoomControls | zmCtl |
| VideoView | vdoVi |
| WdbView | webVi |
| RantingBar | ratBar |
| Tab | tab |
| Spinner | spn |
| Chronometer | cmt |
| ScollView | sclVi |
| TextSwitch | txtSwt |
| ImageSwitch | imgSwt |
| listView | lVi 或则lv |
| ExpandableList | epdLt |
| MapView | mapVi |

12.activity中的view变量命名

命名模式为：逻辑名称+view缩写

建议：如果layout文件很复杂，建议将layout分成多个模块，每个模块定义一个moduleViewHolder，其成员变量包含所属view

13.styles.xml：将layout中不断重现的style提炼出通用的style通用组件，放到styles.xml中；

14.使用layer-list和selector

Android编码规范建议（别人弄的觉得蛮有道理）

1.java代码中不出现中文，最多注释中可以出现中文

2.局部变量命名、静态成员变量命名

只能包含字母，单词首字母除第一个外，都为大写，其他字母都为小写

3.常量命名

只能包含字母和\_，字母全部大写，单词之间用\_隔开

4.图片尽量分拆成多个可重用的图片

5.服务端可以实现的，就不要放在客户端

6.引用第三方库要慎重，避免应用大容量的第三方库，导致客户端包非常大

7.处理应用全局异常和错误，将错误以邮件的形式发送给服务端

8.图片的.9处理

9.使用静态变量方式实现界面间共享要慎重

10.Log(系统名称模块名称接口名称，详细描述)

11.单元测试（逻辑测试、界面测试）

12.不要重用父类的handler，对应一个类的handler也不应该让其子类用到，否则会导致message.what冲突

13.activity中在一个View.OnClickListener中处理所有的逻辑

14.strings.xml中使用%1$s实现字符串的通配

15.如果多个Activity中包含共同的UI处理，那么可以提炼一个CommonActivity，把通用部分叫由它来处理，其他activity只要继承它即可

16.使用button+activitgroup实现tab效果时，使用Button.setSelected(true)，确保按钮处于选择状态，并使activitygroup的当前activity与该button对应

17.如果所开发的为通用组件，为避免冲突，将drawable/layout/menu/values目录下的文件名增加前缀

18.数据一定要效验，例如

字符型转数字型，如果转换失败一定要有缺省值；

服务端响应数据是否有效判断

## android工程：

### 工程目录文件

.gradle

gradle项目产生文件夹（自动编译工具产生的文件）

gradle 运行时自动生成的目录，一般情况不做修改，不需要纳入项目源代 码管理中。

.idea

IDEA项目文件夹（开发工具产生的文件）

Intellij IDEA 运行时候生成的文件目录，一般情况不做修改，不需要纳入 项目源代码管理中。   
 Intellij IDEA是JetBrains公司推出的Java集成开发环境，Android Studio 是基于IDEA Commutity Edition开发的，Community 版本不仅是免费的而 且是开源的。

app

module模块

每一个module可看成在Eclipse中的一个Project，里面的文件结构与父类 差不多。里面也能包含build.gradle、gradle.properties、setting.gradle 等相关gradle文件，若没有定义，则在项目中使用父类的设置。更多的内 容，后面再详细介绍了。

build 构建时生成文件的地方 编译时产生文件，不需要修改，也不需要纳入项目 源代码管理中。

captures 捕获系统信息的日志目录 不需要纳入项目源代码管理中。

gradle gradle环境支持文件夹 wrapper目录下有两个文件，里面有一些项目对 gradle的配置信息

.gitignore git源码管理文件 **[Git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/wx198210/article/details/_blank)**对项目文件管理，可以在里面添加你不希望纳入 **[git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/wx198210/article/details/_blank)**管理的文件。

build.gradle

gradle项目自动编译的配置文件

gradle.properties gradle运行环境配置文件 配置gradle运行环境的文件，比如配置 gradle运行模式，运行时jvm虚拟机的大小

gradlew 自动完成 gradle 环境的linux mac 脚本，配合gradle 文件夹使用

gradlew.bat 自动完成 gradle 环境的windows 脚本，配合gradle 文件夹使用

local.properties Android SDK NDK 环境路径配置

setting.gradle gradle 项目的子项目包含文件

Test.iml IDEA 项目文件 iml文件是Android Studio识别项目的配置文件，跟 Eclipse里面的.project文件作用类似

### App下目录结构：

目录文件 作用

build module编译时所生成文件的目录

我们编译最终生成的apk就在build/outputs/apk目录下，里面包含了app-debug.apk, app-debug-unaligned.apk,app-release-unaligned.apk三种apk, 另外app-release.apk是生成在module的根目录下。

lib 第三方依赖库所在目录 第三方库存放目录，我们可以把需要的第三方库jar文 件放到这里，\*.so也同样放在这里。可以在Project Structure中管理它的 依赖关系，也可以在build.gradle中直接修改。

src module源码所在目录

src\andrroidTest android Studio生成的测试模块，可删除

src\main module代码目录，结构和Eclipse中的差不多了

src\test 单元测试模块，可删除

.gitignore module中的git管理文件

app.iml module中的IDEA 项目文件

proguard-rules.pro module代码混淆配置文件

module中的代码混淆配置文件

build.gradle module自动编译的配置文件 整个项目最主要的gradle配置文件

// 这句是gradle版本的写法，声明是这是一个Android程序。

apply plugin: 'com.android.application'

android {

compileSdkVersion 23 // 编译android的sdk版本 也就是API Level

例如API-19、API-20、API-21等等。

buildToolsVersion "23.0.2" //构建工具的版本 build tools的版本

C:\Users\Administrator\AppData\Local\Android\sdk\build-tools

API20对应的build-tool的版本就是20.0.0

defaultConfig {

applicationId "eric.test" // 应用的包名

minSdkVersion 17 // 允许的最少版本

targetSdkVersion 23 // 目标版本

versionCode 1 // apk的版本

versionName "1.0" // apk的显示版本

}

//自定义签名配置

signingConfigs {

config {

}

}

//构建类型，分为release和debug两种

buildTypes {

// 表明是在release版本中使用的配置

release {

minifyEnabled false // 是否运行混淆

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

// 混淆的配置文件路径，默认给了我们两个配置文件。

// proguard-android.txt在<sdk目录>/tools/proguard下面，里面已经包含了基本的混淆声明，有兴趣的可以去看下。

// proguard-rules.pro是在module根目录下面，我们可以根据项目去添加。

}

debug {

}

}

}

//DEX文件构建属性配置（加快构建速度）

dexOptions {

preDexLibraries true

maxProcessCount 8

javaMaxHeapSize "2048m"

}

//将一个apk拆分成多个相关配置（拆分依据：屏幕密度、系统架构）

splits {

density {

enable true

exclude "xxxhdpi"

reset()

include "ldpi", "xxhdpi"

compatibleScreens 'small', 'normal', 'large', 'xlarge'

}

abi {

enable true

reset()

include "x86", "armeabi-v7a", "mips"

universalApk false

}

}

//依赖的配置：

dependencies {

//1、**模块依赖项**，2、**本地二进制依赖项**，3、**本地二进制依赖项**，4、**远程二进制依赖项**，5、**远程二进制依赖项**

// 表明依赖libs目录下的所有jar包

compile fileTree(dir: 'libs', include: ['\*.jar'])

// 表明在编译项目的测试代码时依赖

testCompile 'junit:junit:4.12'

compile 'com.android.support:appcompat-v7:23.1.1'

compile 'com.android.support:design:23.1.1'

}

### res资源文件夹介绍：

//1.图片资源：

drawable：存放各种位图文件，(.png，.jpg，.9png，.gif等)除此之外可能是一些其他的drawable类型的XML文件

mipmap-hdpi：高分辨率，一般我们把图片丢这里

mipmap-mdpi：中等分辨率，很少，除非兼容的的手机很旧

mipmap-xhdpi：超高分辨率，手机屏幕材质越来越好，以后估计会慢慢往这里过渡

mipmap-xxhdpi：超超高分辨率，这个在高端机上有所体现//2.布局资源：

layout：该目录下存放的就是我们的布局文件，另外在一些特定的机型上，我们做屏幕适配，比如480\*320这样的手机，我们会另外创建一套布局，就行：layout-480x320这样的文件夹！//3.菜单资源：

menu：在以前有物理菜单按钮，即menu键的手机上，用的较多，现在用的并不多，菜单项相关的资源xml可在这里编写，不知道谷歌会不会出新的东西来替代菜单了~//4.values目录

demens.xml：定义尺寸资源string.xml：定义字符串资源

styles.xml：定义样式资源

colors.xml：定义颜色资源

arrays.xml：定义数组资源

attrs.xml：自定义控件时用的较多，自定义控件的属性！

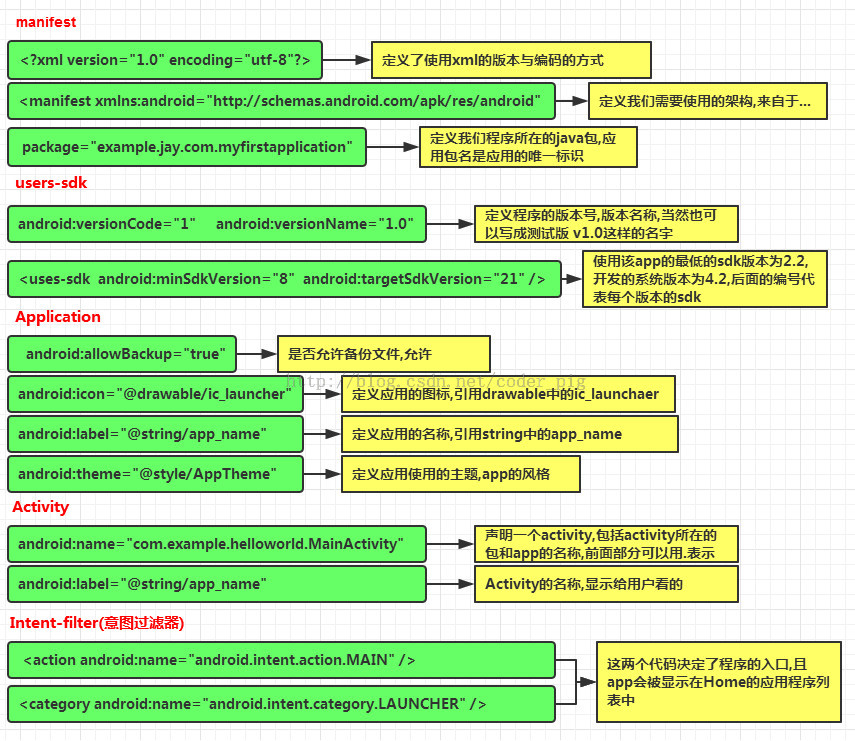
theme主题文件，和styles很相似，但是会对整个应用中的Actvitiy或指定Activity起作用，一般是改变窗口外观的！可在Java代码中通过setTheme使用，或者在Androidmanifest.xml中为<application...>添加theme的属性！ PS:你可能看到过这样的values目录：values-w820dp，values-v11等，前者w代表平板设备，820dp代表屏幕宽度；而v11这样代表在API(11)，即android 3.0后才会用到的！//5.在接着说下raw目录：

用于存放各种原生资源(音频，视频，一些XML文件等)，我们可以通过openRawResource(int id)来获得资源的二进制流！其实和Assets差不多，不过这里面的资源会在R文件那里生成一个资源id而已//6.最后还有个动画的:

动画有两种：属性动画和补间动画：

animator：存放属性动画的XML文件

anim：存放补间动画的XML文件



## 视图：

## 主题：

## 字体：

## 颜色：

## 自动导入:

## 两个gradle

1:作为插件的gradle

2:gradle的包

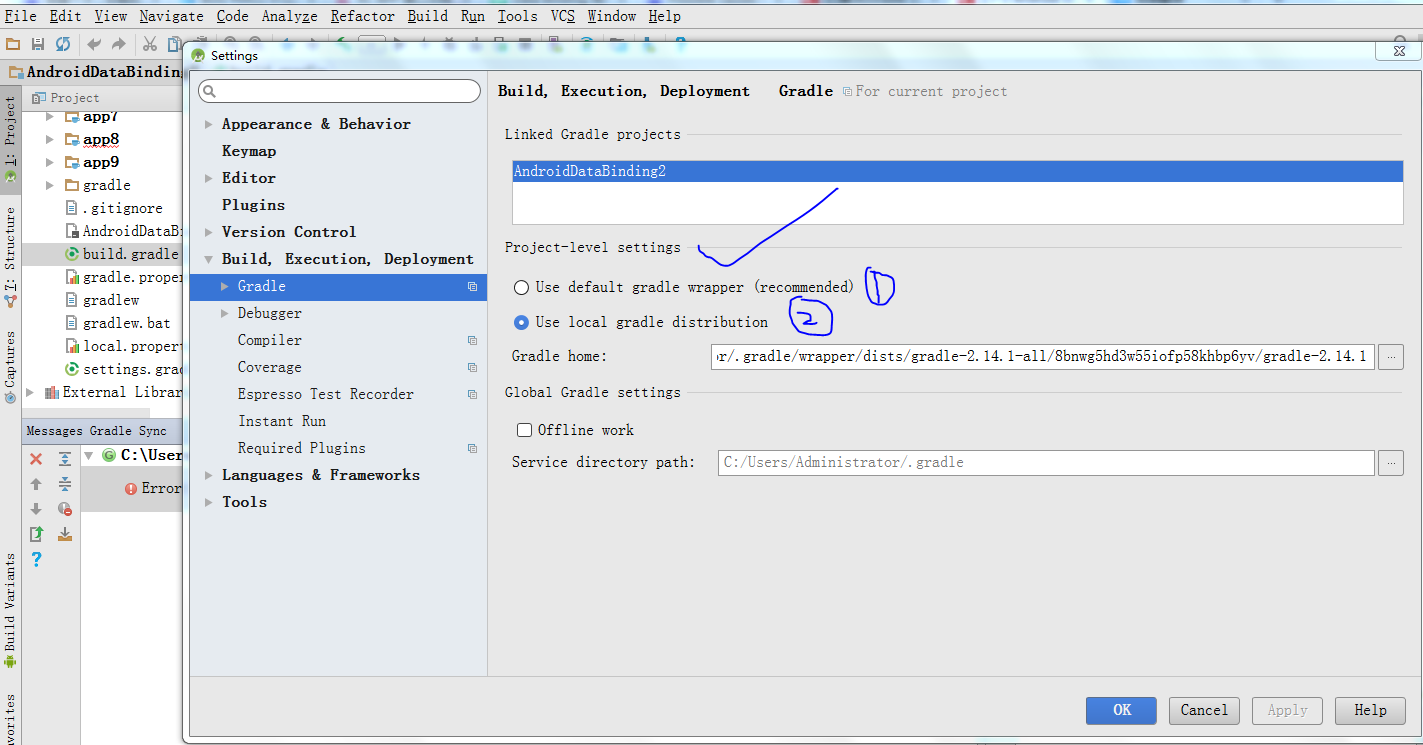
1: 作为插件的gradle 是在

Gradle 文件里面下面的这个是gradle给android studio的插件。

build.gradle(project) 里面的classpath 'com.android.tools.build:gradle:1.5.0'

2: 作为gradle包的gradle 是在这里

File->setting->build->gradle

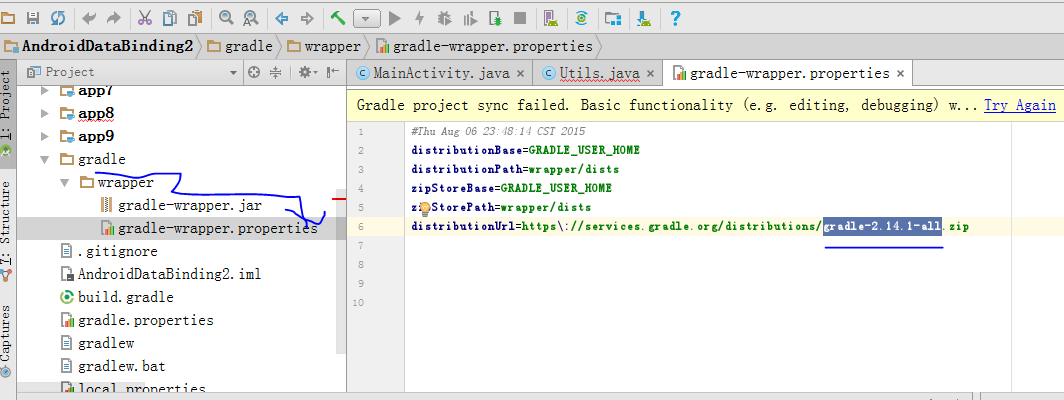


**第一种是用wrapper的方法，也是android studio 推荐的方法，它会自动的去下载**

**第二种是手动下载后指定位置**

**更改gradle版本**

**项目的根目录下面的gradle文件夹--》wrapper--》gradle-wrapper.properties文件：**

最后一行：

**distributionUrl**=**https\://services.gradle.org/distributions/gradle-2.14.1-all.zip**

**gradle-2.14.1-all 这里修改对应的版本**

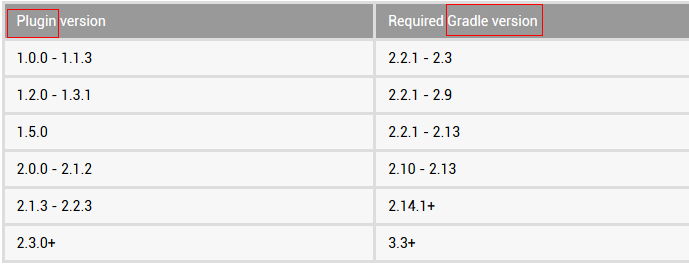
**手动下载gradle 方法，最好下载 \*\*.Src类型，这个下载的不是Android Gradle Plugin）：**

下载地址就是这个：[https://services.gradle.org/distributions/](https://services.gradle.org/distributions/" \t "http://blog.csdn.net/rodulf/article/details/_blank)

**Gradle Plugin 和 Gradle的版本对比**

**如果plugin是2.3.0的那么Gradle 就至少是3.3的**

参考官方网站，选择合适的[https://developer.android.com/studio/releases/gradle-plugin.html#updating-plugin](https://developer.android.com/studio/releases/gradle-plugin.html" \l "updating-plugin" \t "http://blog.csdn.net/rodulf/article/details/_blank)



相关知识：

依赖管理由两方面构成。

一方面，Gradle需要知道在构建您的项目时所需要构建或者运行的东西以便Gradle找到它们，我们将这些被导入的文件称作项目的依赖。

另一方面，Gradle需要需要构建或者上传您的项目产出的东西，我们将这些由您的项目产出的文件称作项目的出版物

<http://ask.android-studio.org/?/article/7>

[http://ask.android-studio.org/?/article/10](http://ask.android-studio.org/?/article/10" \t "http://blog.csdn.net/rodulf/article/details/_blank)

[https://developer.android.com/studio/releases/gradle-plugin.html](https://developer.android.com/studio/releases/gradle-plugin.html" \t "http://blog.csdn.net/rodulf/article/details/_blank)

[http://gradle.android-studio.org/](http://gradle.android-studio.org/" \t "http://blog.csdn.net/rodulf/article/details/_blank)

如果 Module 用到的 **buildToolsVersion** 高于 **22.0.1**，比如 **23 rc1**，那 com.android.databinding:dataBinder 的版本要改为 **1.3.0-beta1**，否则会出现如下错误：

**gradle插件版本配置位置：**

project对应的build.gradle文件中

buildscript {

repositories {

jcenter()

}

dependencies {

classpath **'com.android.tools.build:gradle:1.2.3'** }

}

**gradle版本配置位置：**

gradle-wrapper.properties 中

**distributionUrl**=**https\://services.gradle.org/distributions/gradle-2.2-all.zip**

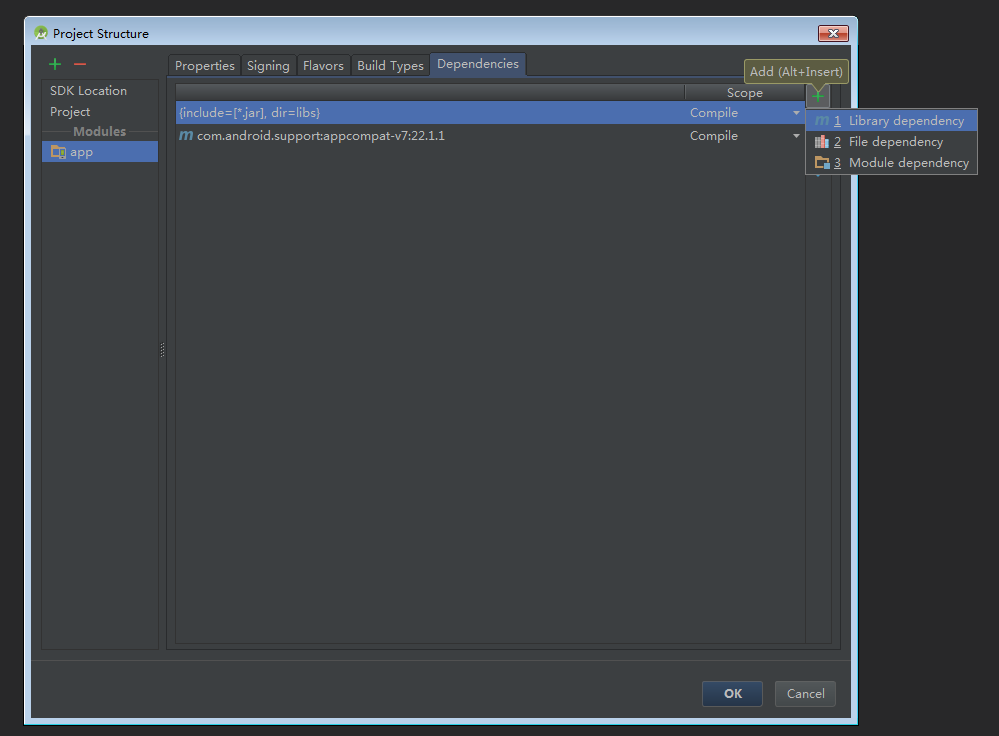
查看gradle 版本和Android 插件的版本

File->Project Structure->Project

## 添加依赖：

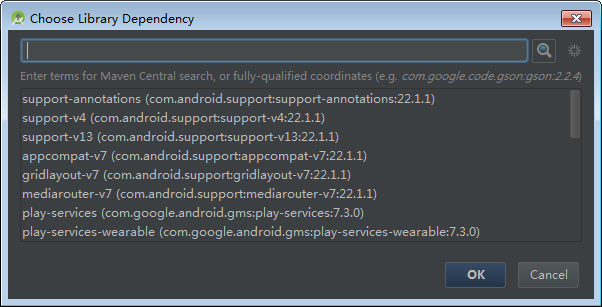
## 一 JAR包的依赖配置

首先使用快捷键**Ctrl+Alt+Shift+S**打开当前项目的配置，如下图所示：



### 1.1 直接搜索法

点击 +->Library dependency，如下图所示搜索选择需要添加的库。



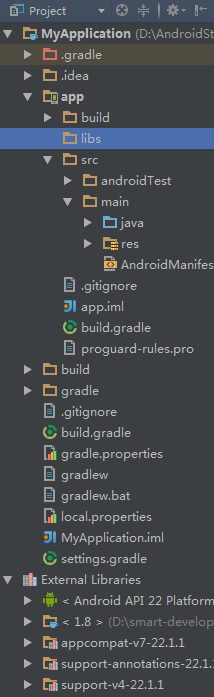
直接在Module上右键Open Module Settings

选中Modules，打开Dependencies选项卡，点击右边的‘+’号，选第一个Library Dependency

选中appcompat-v7即可

### 1.2 libs添加法

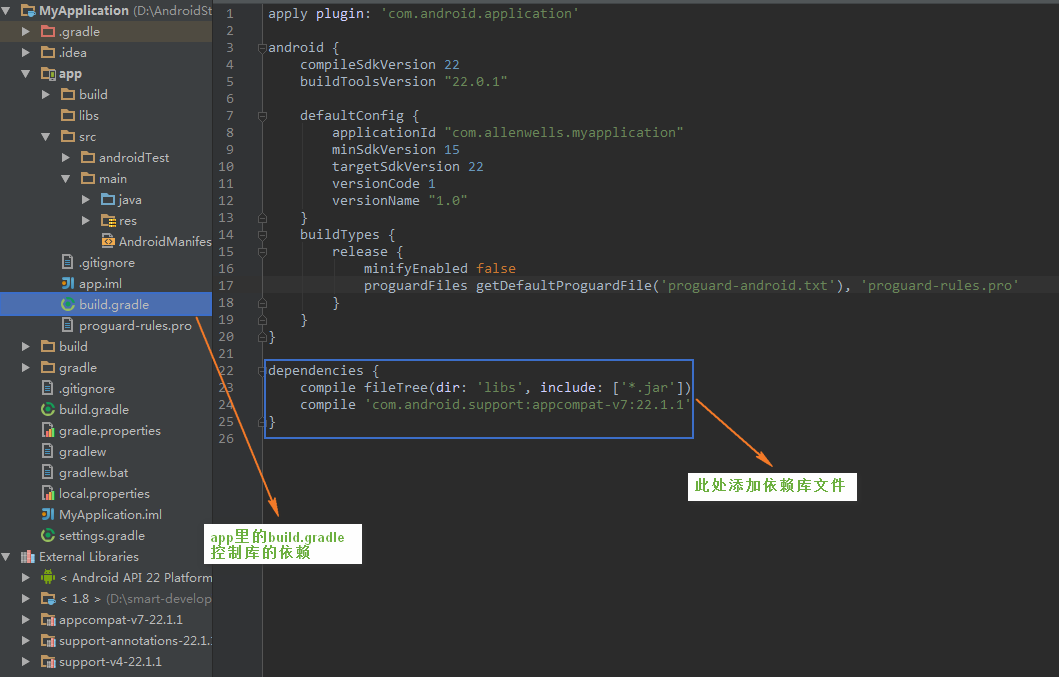
将库文件直接复制到libs目录下，如果没有可以新建一个，如下图所示：



**注意：Eclipse导入的工程libs目录在根目录下。**

### 1.3 gradle添加法

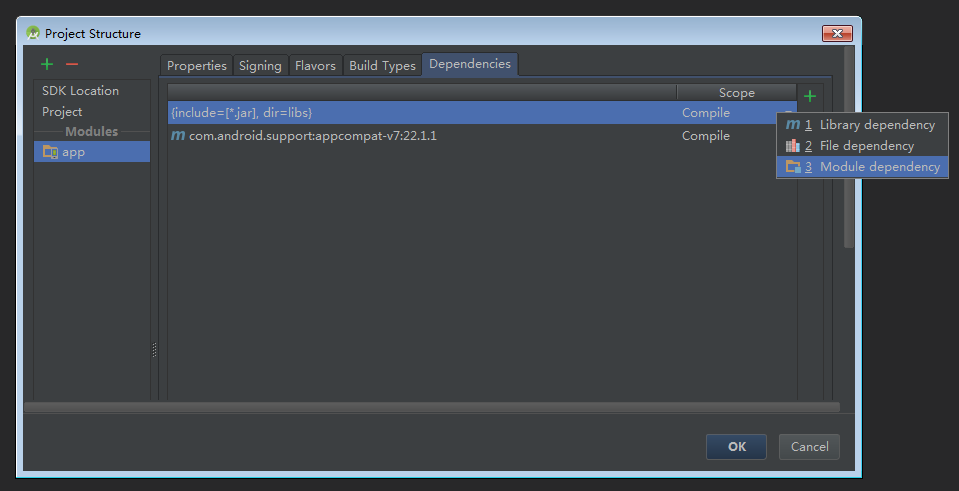
我们同样也可以直接在模块里的build.gradle配置库的依赖，如下图所示：



compile ‘com.Android.support:appcompat-v7:+’，然后rebuild就可以

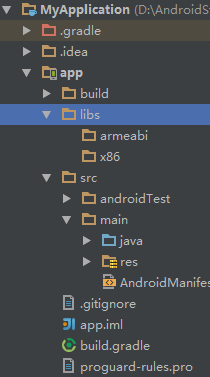
## 二 Module的依赖配置

添加工程依赖的Module，如下图所示：



## 三 SO库的依赖配置

按照CPU类型创建相应目录，放入SO库，再将该目录复制到libs目录下，如下图所示：



然后在Module中的build.gradle添加以下内容，如下图所示：



SO库链接成功后，src/main目录下回生成一个jniLibs文件夹。

## 编译：

## 版本控制

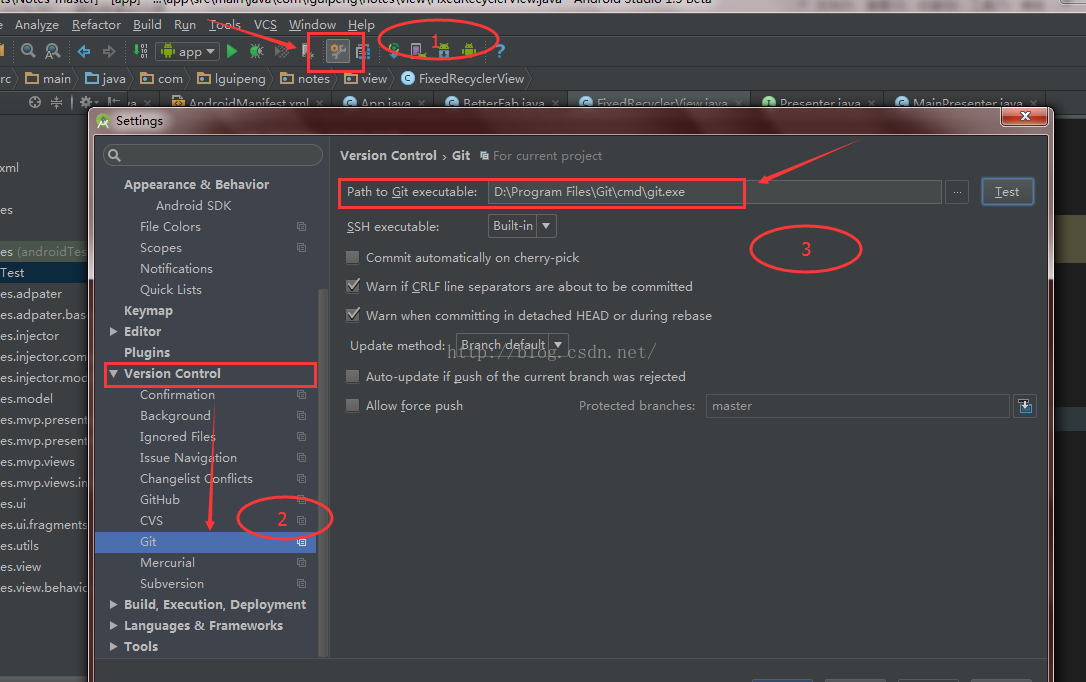
### Git命令

<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/001373962845513aefd77a99f4145f0a2c7a7ca057e7570000>

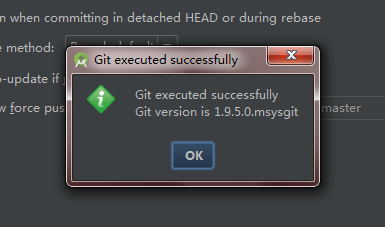
[http://www.runoob.com/git/git-tutorial.html](http://www.runoob.com/git/git-tutorial.html" \t "http://blog.csdn.net/gao_chun/article/details/_blank)

### Git命令端

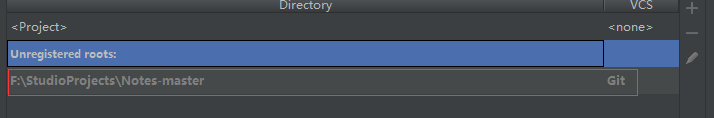
1. 下载git
2. 创建本地仓库并提交代码
3. Git init 来初始化一个仓库 该仓库的根目录下生成一个.git的目录
4. Studio配置
5. Settings中选择Version Control 并配置Git



配置完后我们点下路径后的Test按钮，若出现该Success提示框则表明配置成功



若发现该路径为灰色，需要选中，点击"+"



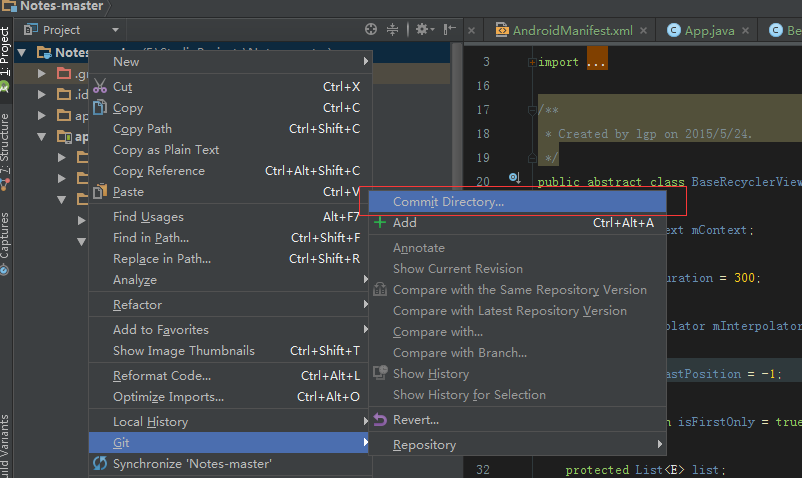
工程项目中：

[白色：正常文件的颜色]

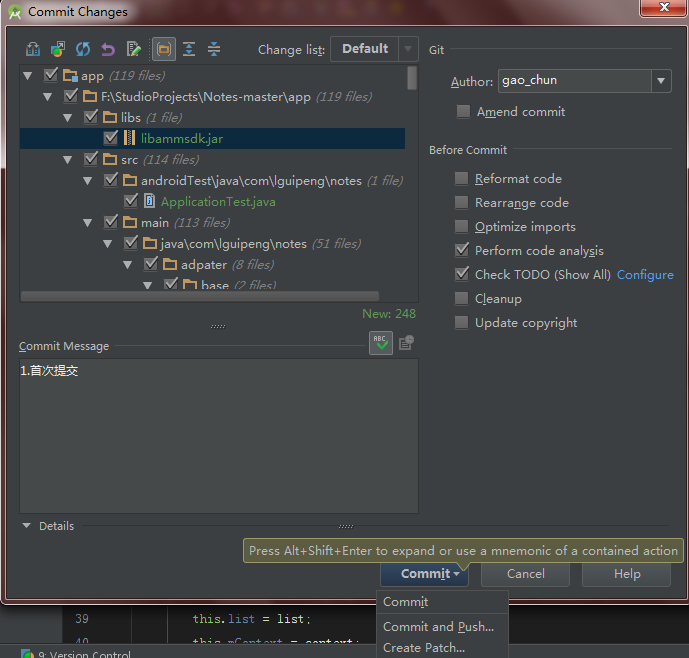
红色：指定仓库路径后，未Add的文件]

绿色：已Add，但未commit的文件]

Commit文件可通过 项目右键--> Git --> Commit Directory...  也可点击 工具栏上的两个按钮。

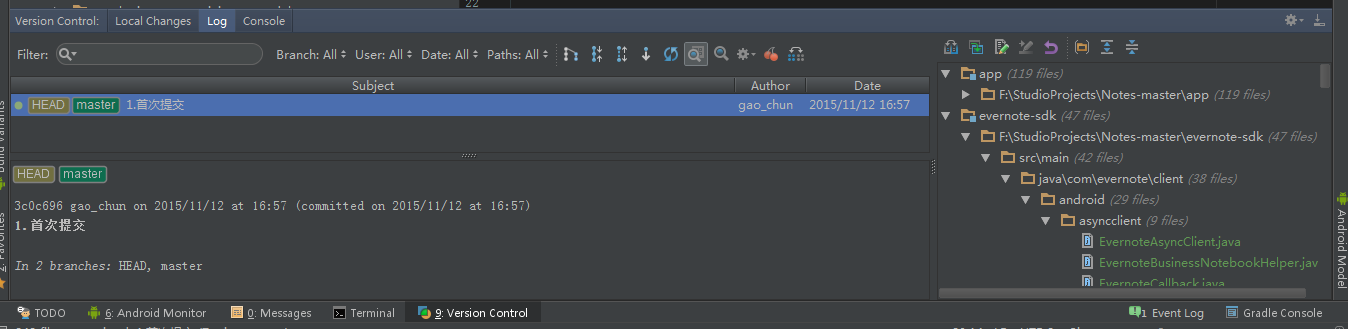


填写提交信息，然后可先Commit 然后再 Push 或者直接选择 Commit And Push ...

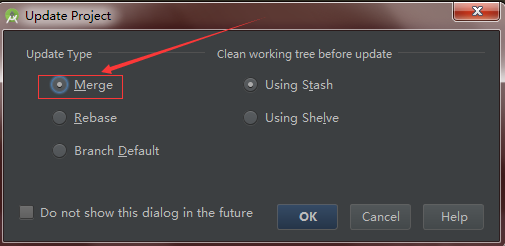


最后在提交完成时，由于项目中可能存在一些警告信息，提交后会出现提示框，选择提交便可。

⑤当我们对某个类进行修改后，会发现文件颜色又变了 [墨蓝色：提交成功后修改过的文件]，到此就将项目上传至仓库了，可以通过Studio中的Version Control一目了然的查看提交Log。

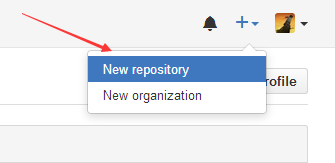


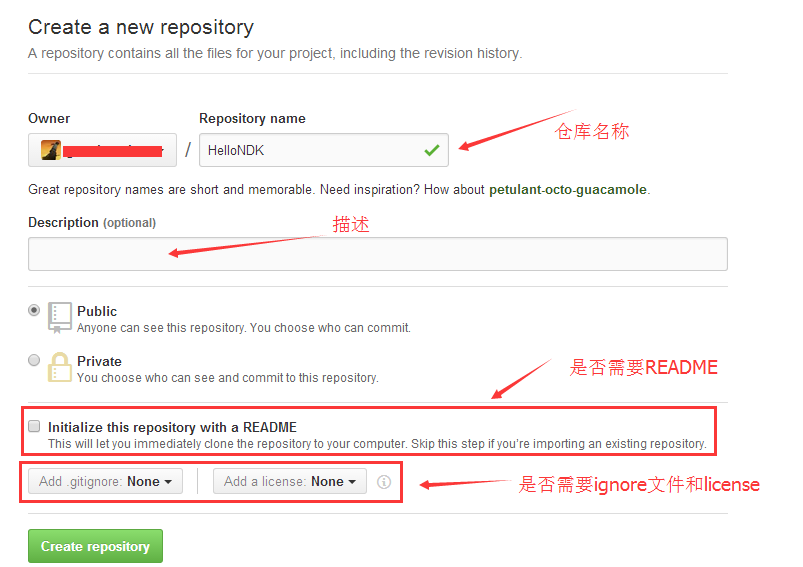
⑥关于如何下拉代码，我们可以点击工具栏上的 按钮，然后在在弹出框中选择 Merge 合并。



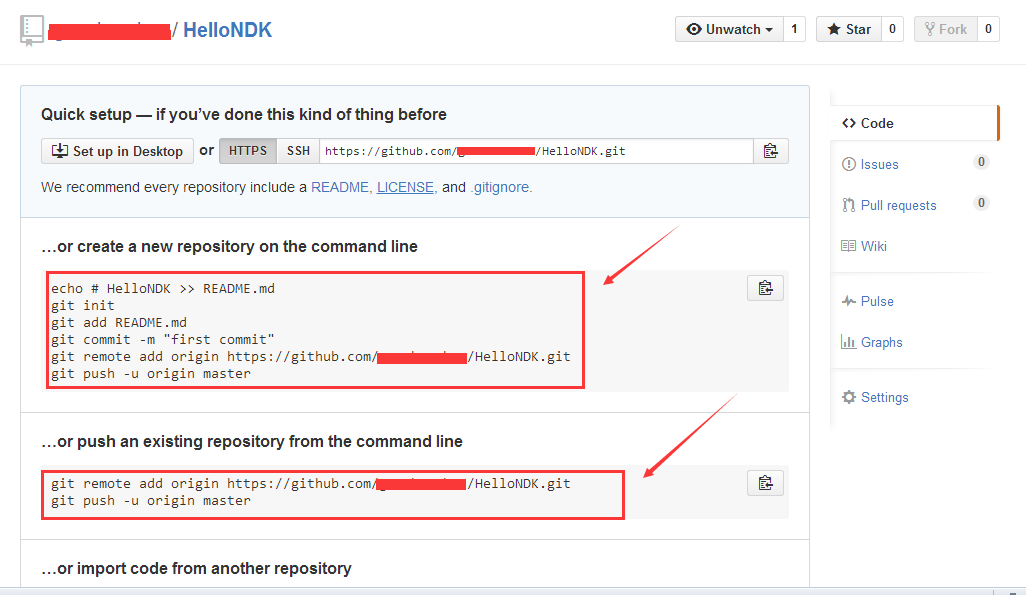
### Studio上传项目至GitHub

1. GitHub账号
2. new 一个仓库





1. Create成功后



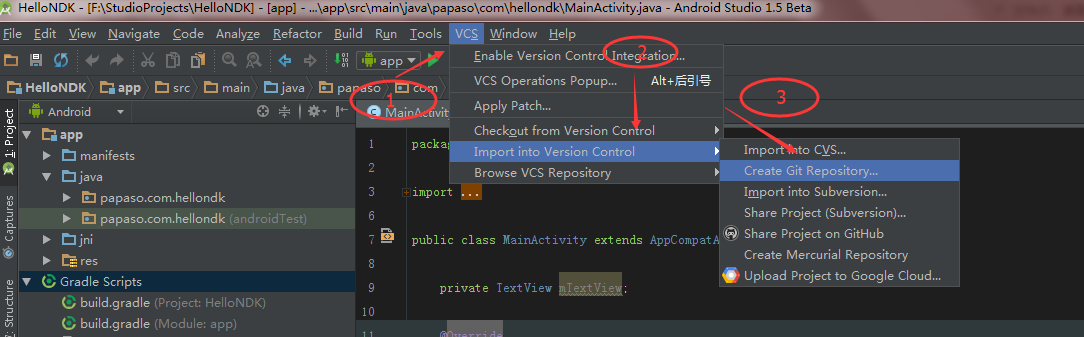
1. 新建项目的话

在Project location指定好项目路径 和之前上传本地仓库相同

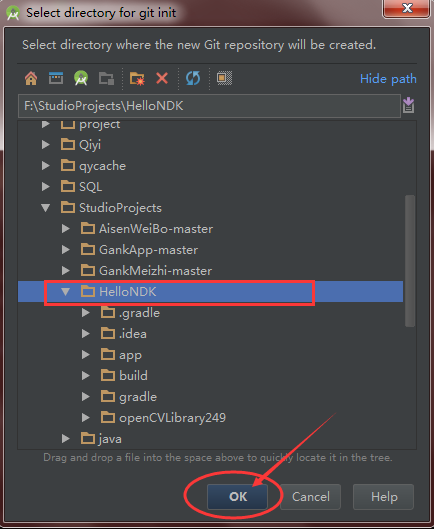
1. 已有项目

远程仓库建好了，本地项目也准备好了

需要为本项目创建Git



选择根目录

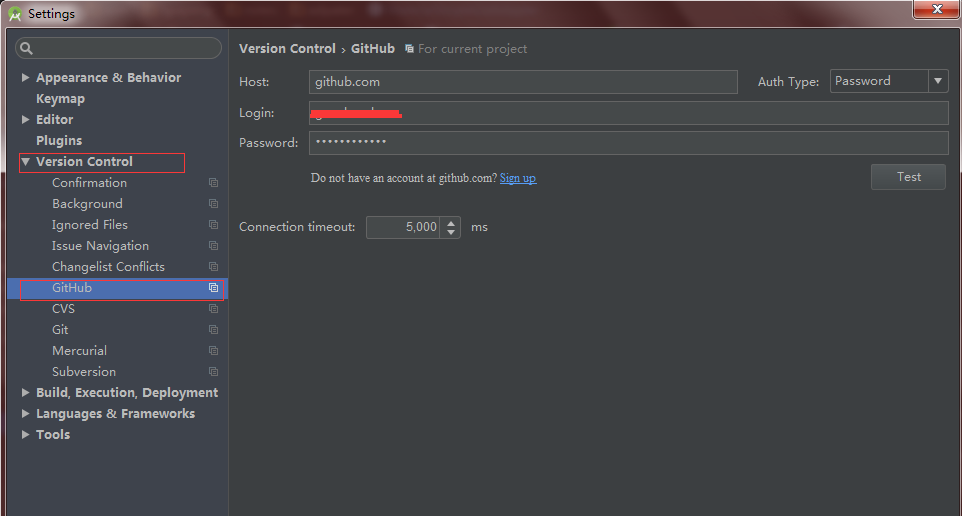


已经全部变成了红色

### Studio中配置GitHub并下拉代码

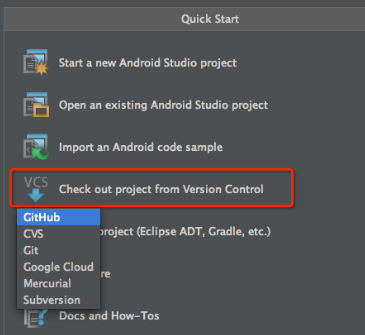
GitHub上将项目使用Studio下载到本地

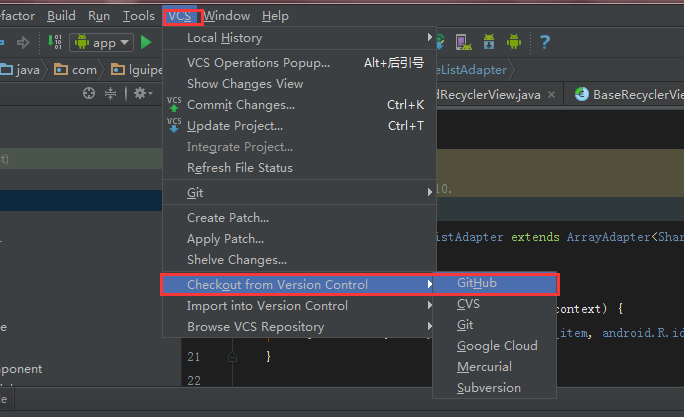
首先来配置Studio，打开 Settings 中 Version Control 下的GitHub，输入账号密码



点击Test，可能需要再次确认密码，之后点击OK

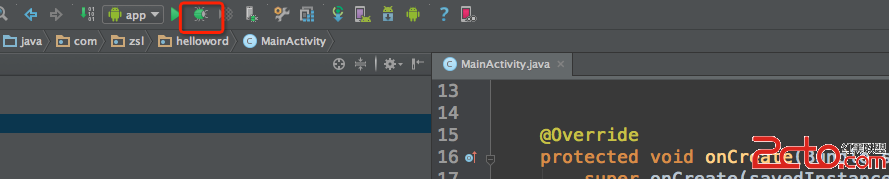
从首界面Check也好，从当前项目Check也好.



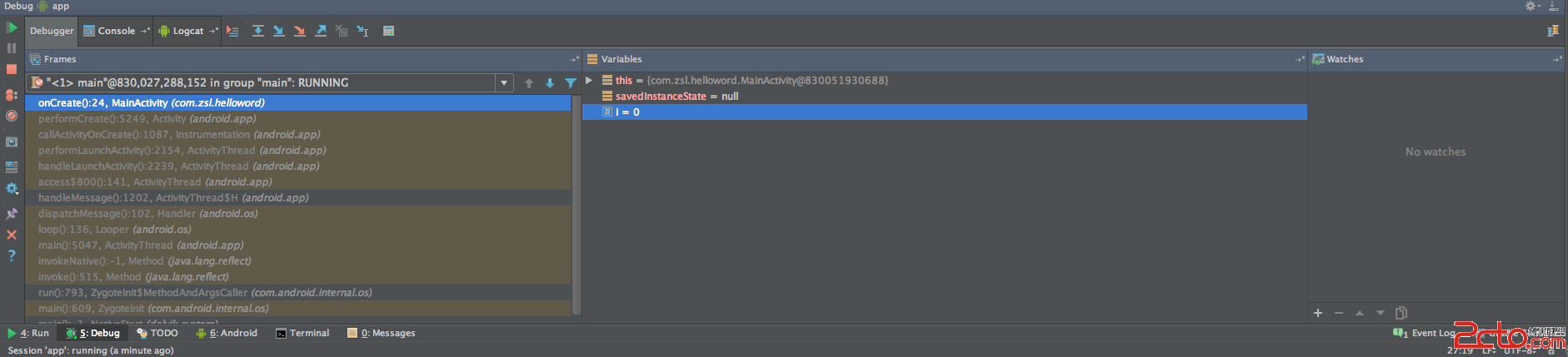


## 断点调试和高级调试

1. 设置断点（点击红点位置添加或取消断点）
2. 点击debug模式运行

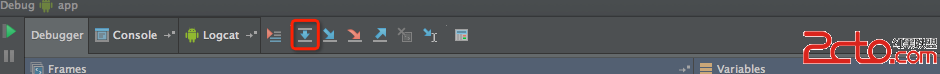


1. 查看调试面板

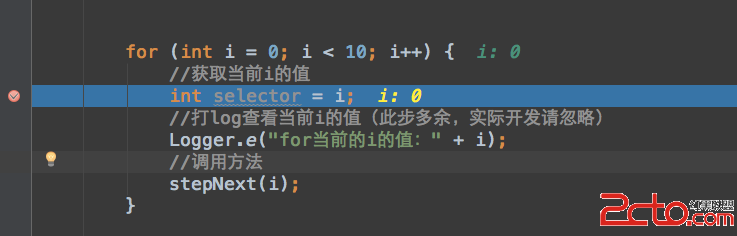


## 简单调试

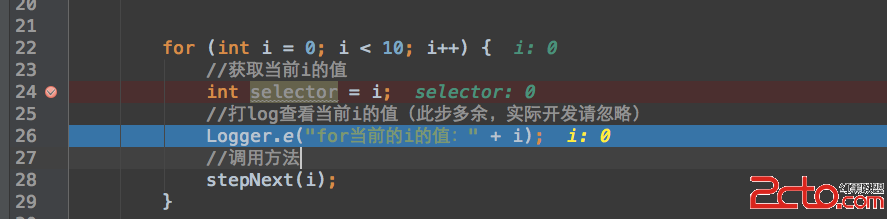
### step over：一步步往下走



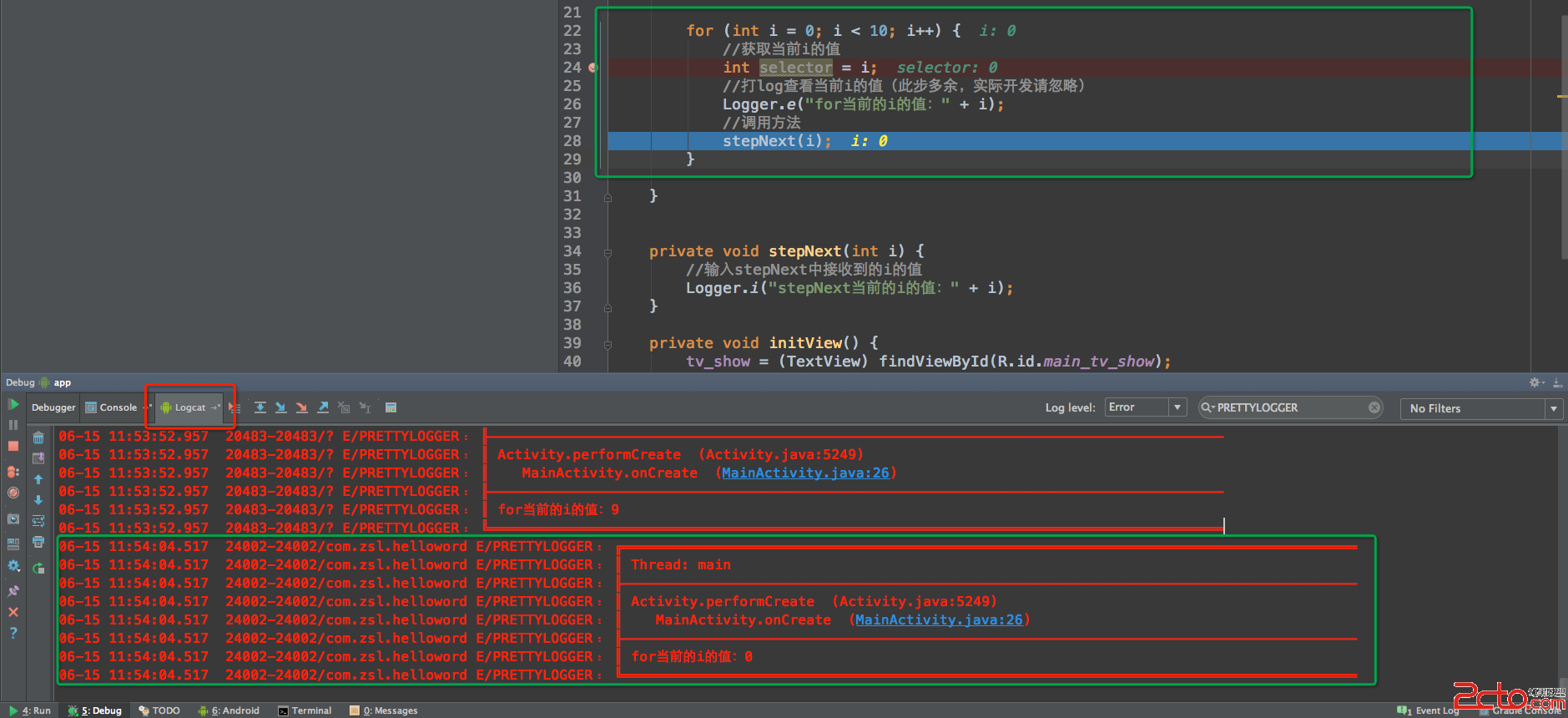
当前程序运行的位置，我们看到i的值已经在程序代码中展示出来了，黄色的代码处，这个是AS的功能，对于我们调试来讲，这简直是非常大的福利了。



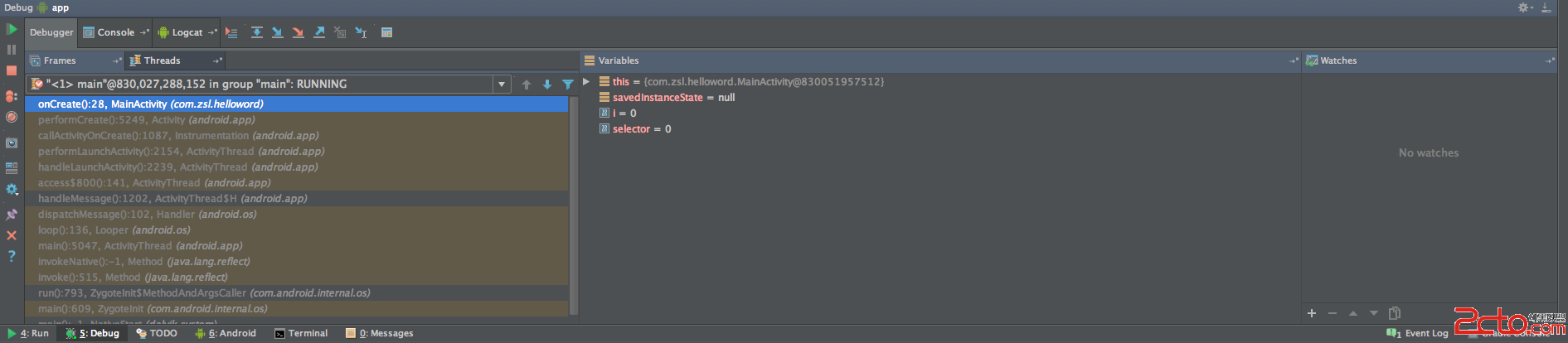
点击单步调试按钮或按快捷键F8，看看效果。这里我们看到selector变量的值已经出来了selector：0,我们在看看黄色位置i的当前值是0。



这时我们继续F8，我们切换到logcat查看日志，我打印出的i的值是0，

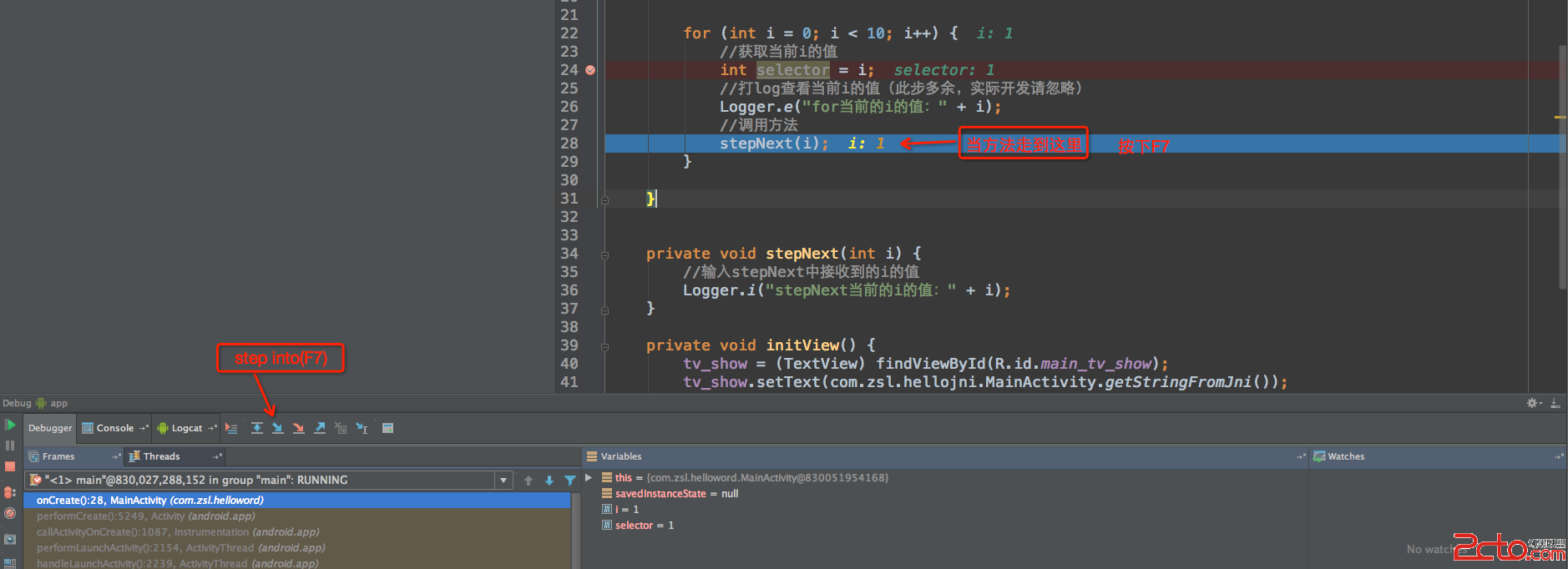


我们在切回道Debugger面板，可以看到Variables显示面板中，有i的值是0，selector的值是0。以及我们可以看到Frames控制面板中可以显示出当前程序的位置在：onCreate():28,第28行。



### 2. step into：看到方法往里走

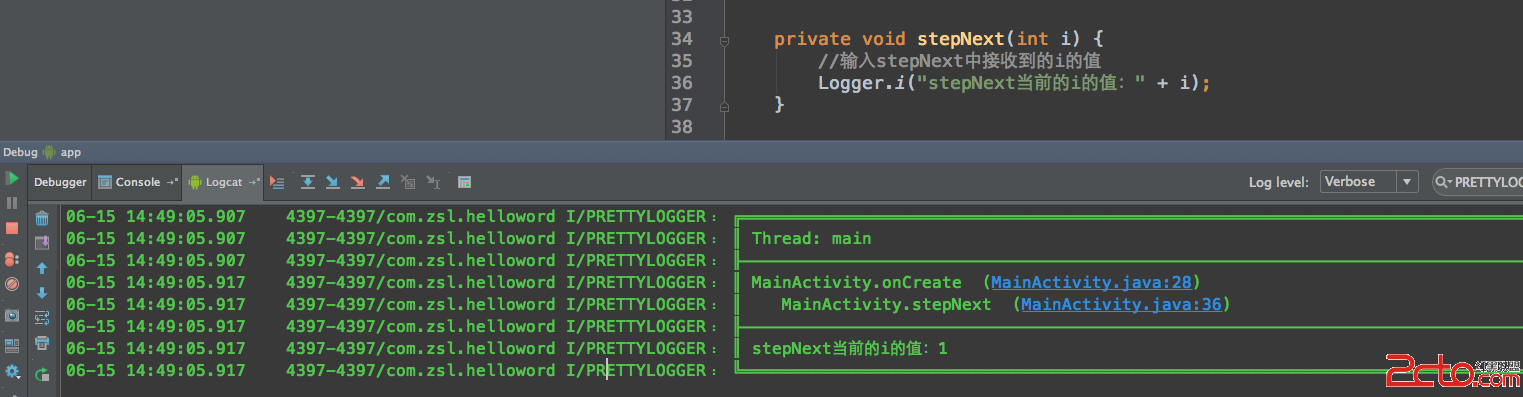
比如我们的for循环当中调用了一个stepNext(int i)方法，当我们走到这里想看看这个方法里面的运行过程的时候我们可以这样，当走到这个方法的时候我们可以按下F7,或者如下图的图标。



这时就走到了stepNext方法当中。

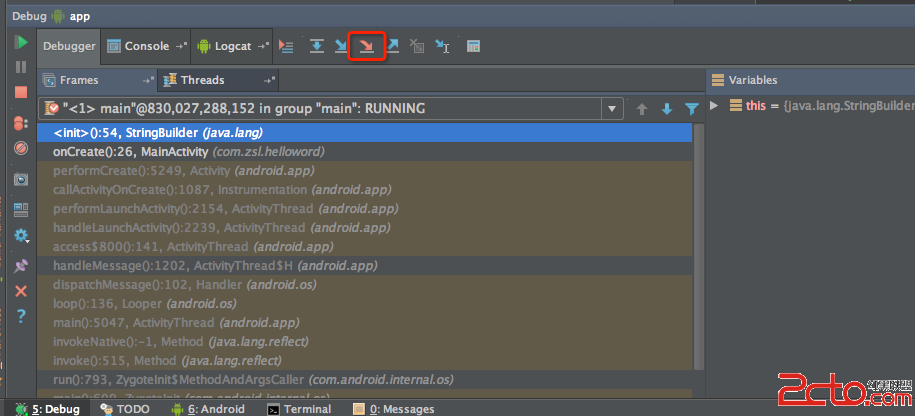


在这里打印了一个log，我们再按一下F8我们来看看Logcat, 这里我打印的log都是为了做教程用，调试我们就不用打log了直接看显示面板就OK了



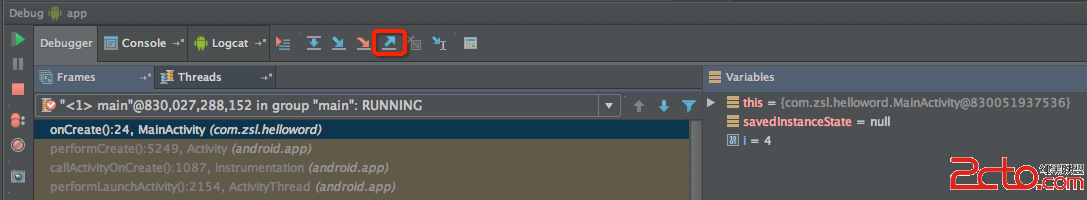
### 3. force step into :所有方法看完整

这个是可以看到你所调用的所有方法的实现会让你跟着它走一遍，研究[源码](http://www.2cto.com/ym" \t "http://www.cnblogs.com/mmcc/p/_blank)使用非常方便

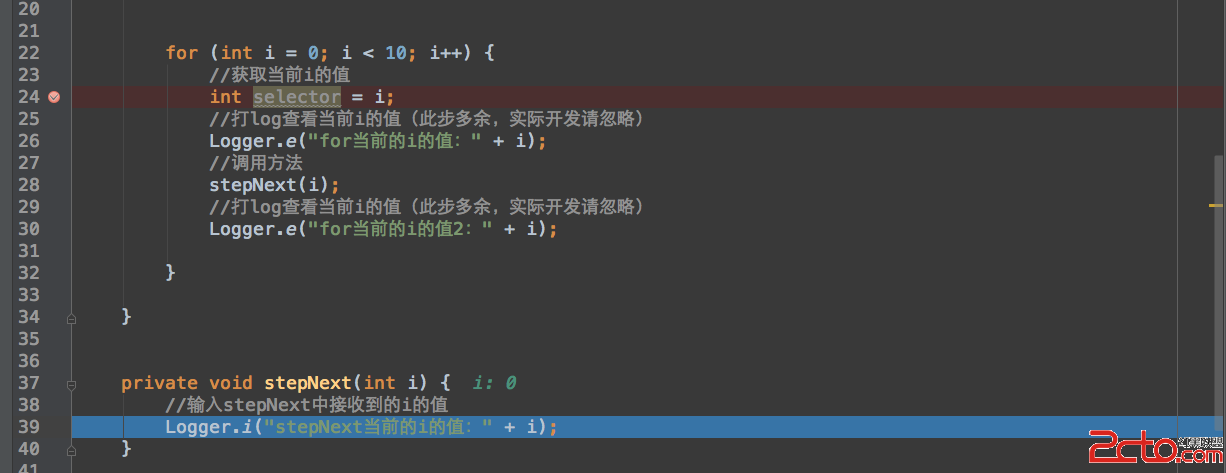


### 4. step out ：有断点下一个，走完断点继续走

这里如果我们的一个流程当中，包括调用的方法，如果有断点走到下一个断点，如果没有断点，而是在一个调用的方法当中，会跳出这个方法，继续走。



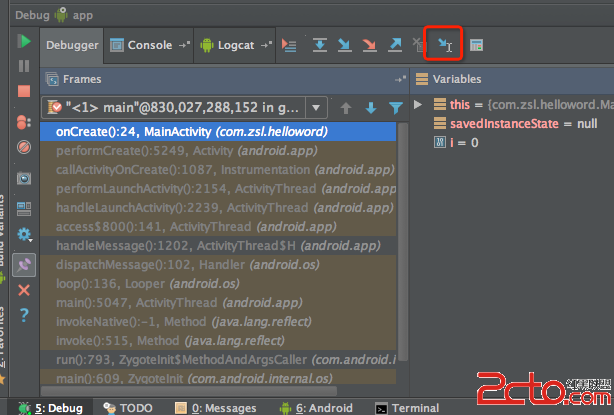
(上图)我现在程序位置在第一个断点位置（24行），我调用的stepNext方法中也有一个断点，此时我按下step out按钮会走到stepNext中的断点处（39行）我此时如果再按一下step out 会走到stepNext方法的调用出的下一个可执行代码（30行）



(上图)如果我现在程序位置在stepNext的方法中，如果我此时按下step out，会走到stepNext方法的调用出的下一个可执行代码（30行）

### 5. run to Cursor ：下个断点我们见

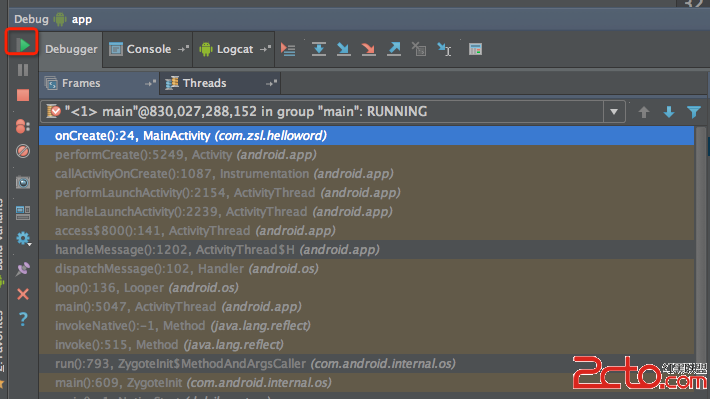
这里的意思就是说，会很快执行到下一个断点的位置，而且可以静如任何调用的方法



## 高级调试

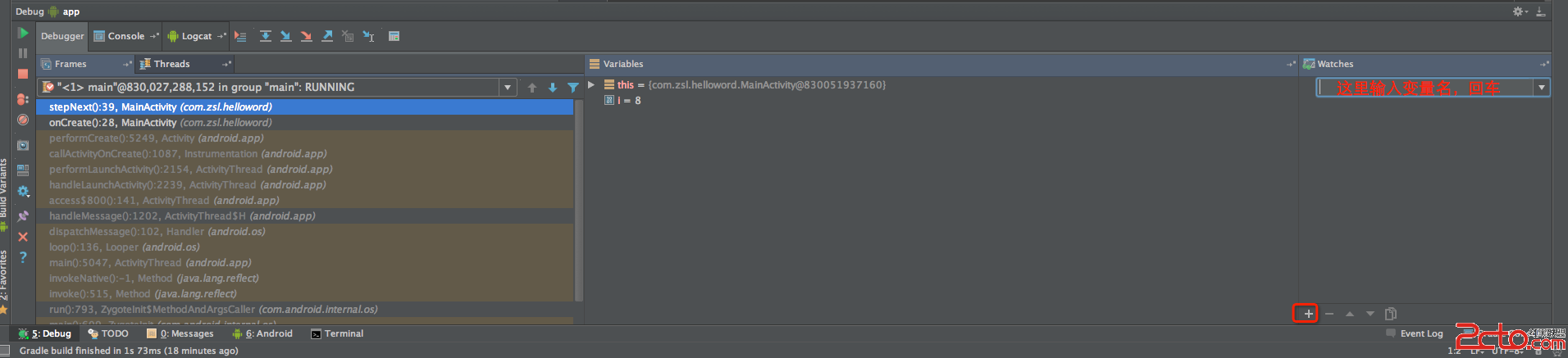
### 1. 跨断点调试

如果我们设置了多个断点，现在我们需要直接跳转到下一个断点，那么直接点击下图就可以了



### 2.观察变量

如果我们想观察1个或者几个变量的值的变化，如果我们在Variables显示面版中观察如果我这里有太多太多的自定义变量和[系统](http://www.2cto.com/os/" \t "http://www.cnblogs.com/mmcc/p/_blank)变量了，那么就难观察了，我们可以做如下操作：  
点击Watches,点击＋号，然后输入变量的名称回车就OK了，而且会有历史记录哦

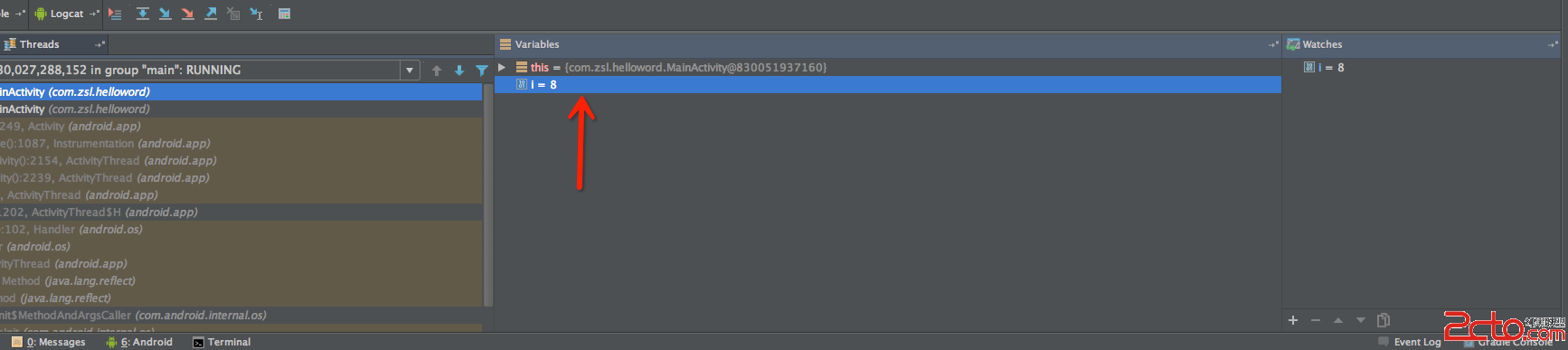


如果变量名比较长我们可以这样：  
选择［Variables］中的变量名然后点击［右键］，选择［Add to Watches],然后Watches面板中就有了

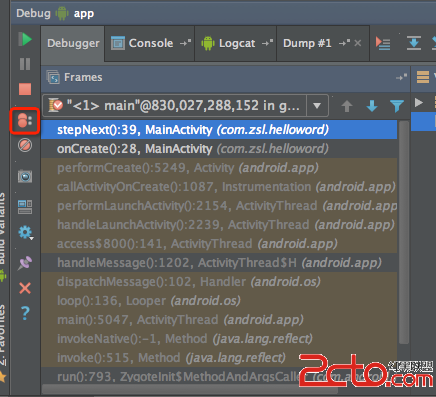


### 3.设置变量的值

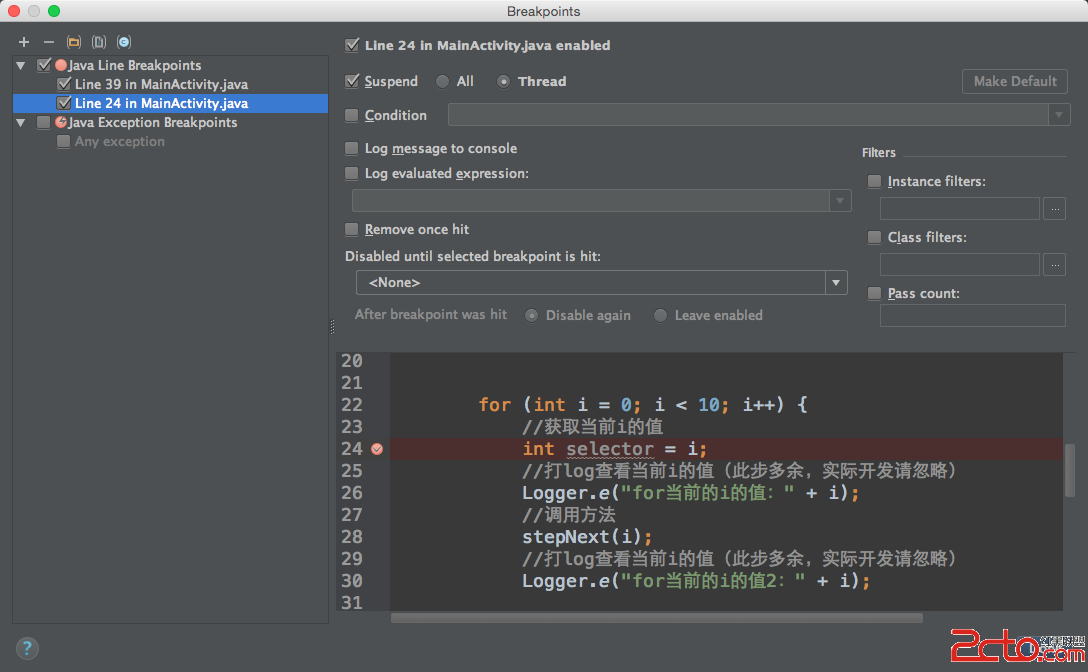
在程序中有很多的条件语句和循环语句，调试也是比较耗时的，我们可以通过快速设置变量的值来加快调试速度，我们可以做如下操作：  
选择［Variables］中的变量名然后点击［右键］，选择［Set Value..]或者选择之后直接F2(如上图)（下图为Variables面板）



### 4.查看断点

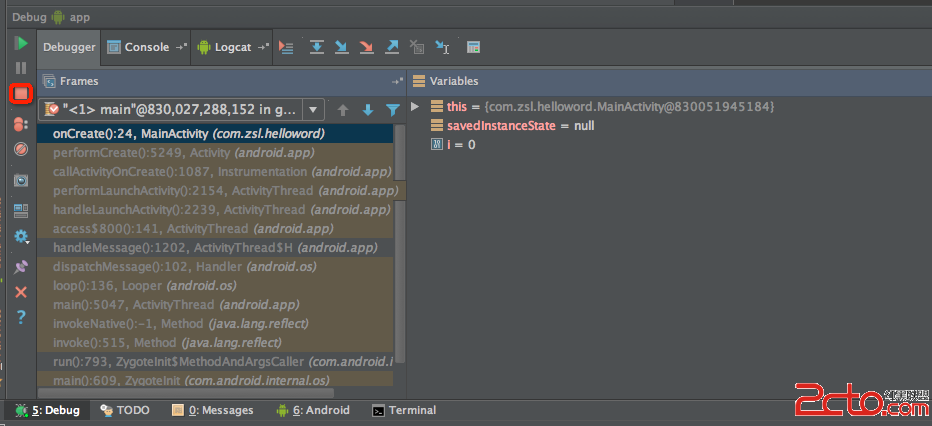


点击之后我们可以看到所有的断点，以及位置代码,也可以设置一些属性



### 5.停止调试

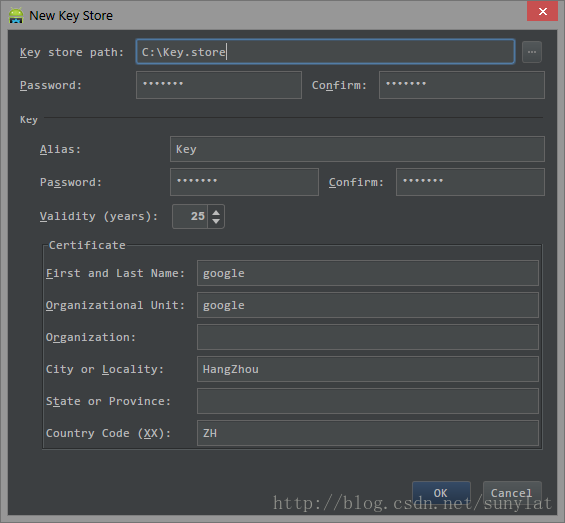
要注意的是这里的［停止调试］不是让程序停止，而是跳过所有调试



## 打包

1. **[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/sunylat/article/details/9239595/_blank)** Studio菜单Build->Generate Signed APK
2. 创建keystore或者使用现有的

Key store path：密钥库文件的地址   
        Password/Confirm：密钥库的密码   
        Key：   
            Alias：密钥名称   
            Password/Confirm：密钥密码   
            Validity(years)：密钥有效时间   
            First and Last Name：密钥颁发者姓名   
            Organizational Unit：密钥颁发组织   
            City or Locality：城市   
            Country Code(XX)：国家

1. 
2. 选择已存在密钥库及密钥（在（3）中创建密钥库后跳过此步骤）   
       点击“Choose existing...”按钮找到密钥库文件   
       Key store password输入已选择的密钥库文件的密码   
       点击Key alias后的“...”按钮，选择或者创建一个密钥

## 导入项目问题

build.gradle

dependencies {

classpath 'com.android.tools.build:gradle:2.2.2'

｝

gradle\wrapper\gradle-wrapper.properties

distributionUrl=https\://services.gradle.org/distributions/gradle-2.14.1-all.zip

local.properties文件

Sdk：

sdk.dir=C\:\\Users\\Administrator\\AppData\\Local\\Android\\Sdk

项目src同级目录下build.gradle：

compileSdkVersion 24

buildToolsVersion "24.0.3"

defaultConfig {

applicationId "com.gogobeauty.client"

minSdkVersion 9

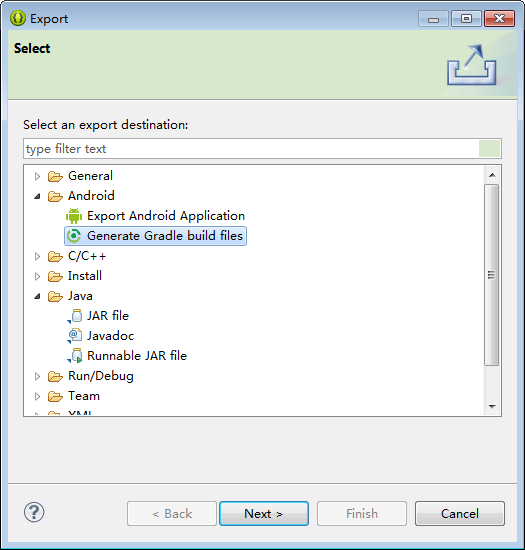
targetSdkVersion 24

}

## Eclipse项目迁移到**[android](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/allenwells/article/details/_blank)** Studio

从Eclipse导出

如下图所示：

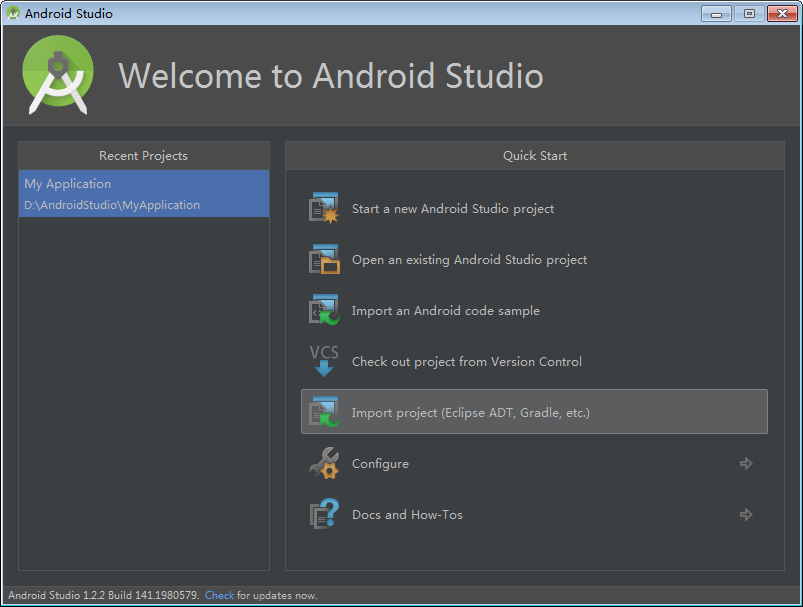


1. 将你的ADT插件版本升级到22.0以上。
2. 在Eclipse中，选择File–>Export。
3. 在弹出的导出窗口中，打开Android的文件夹，选择“Generate Gradle Build Files”。
4. 选中你想要导入到Android Studio中的项目，点击finish。

导出的项目将会和原来的项目在同一目录，覆盖原来的同时，会新增一个叫build.gradle的文件，导入Android Studio时将首先读取这个文件。

导入到Android Studio

如下图所示：



1. 在Android Studio 中，首先关掉你当前的打开的项目。
2. 在欢迎界面，点击Import Project（注：也是可以直接在菜单选择Import project的）
3. 选中你在Eclipse中导出的项目，展开目录，点击build.gradle文件，然后OK
4. 在之后的弹出对话框中，会要求你选择Gradle的配置，选中Use gradle wrapper.(注：也可以自定义你本机装的Gradle)

如果没有Grade build文件，也是可以将普通的**[安卓](http://lib.csdn.net/base/android" \o "Android知识库" \t "http://blog.csdn.net/allenwells/article/details/_blank)**项目导入到Android Studio中，它会用现有的Ant build.但为了更好地使用之后的功能和充分使用构建变量，还是强烈地建议先从ADT插件中生成Gradle文件再导入Android Studio。

# AS快捷键

## **导入**

Alt+回车 导入包,自动修正

Ctrl+Alt+O 优化导入的类和包

## 查找

Ctrl+N   查找类  
Ctrl+Shift+N 全局查找文件，文件名 方法名 类名

Ctrl+R 替换文本  
Ctrl+F 查找文本  
Ctrl+Shift+Alt+N 查找类中的方法或变量

Alt+F1 查找代码所在位置

代码标签输入完成后，按Tab，生成代码。

选中文本，按Ctrl+Shift+F7 ，高亮显示所有该文本，按Esc高亮消失。

Ctrl+W 选中代码，连续按会有其他效果

选中文本，按Alt+F3 ，逐个往下查找相同文本，并高亮显示。

Ctrl+Up/Down 光标跳转到第一行或最后一行下

Ctrl+B 快速打开光标处的类或方法

Ctrl＋[或]可以跳到大括号的开头结尾

Ctrl＋Alt＋B可以跳转到抽象方法的实现

Ctrl＋Shift＋F7可以高亮当前元素在当前文件中的使用

Ctrl＋Shift＋Backspace可以跳转到上次编辑的地方

Ctrl＋F7可以查询当前元素在当前文件中的引用，然后按F3可以选择

Ctrl＋N，可以快速打开类

Ctrl＋Shift＋N，可以快速打开文件

Alt＋Q可以看到当前方法的声明

Ctrl＋Alt＋T可以把代码包在一块内，例如try/catch

Alt＋Up and Alt＋Down可在方法间快速移动

## 代码设置

Ctrl+Alt+L  格式化代码  
Alt+Insert 生成代码(如get,set方法,构造函数等)  
Ctrl+E或者Alt+Shift+C  最近更改的代码

Ctrl+Y 删除行（ctrl+x不是删除行，是剪切。如果不选中，则为剪切当行。ths for 貌似掉线）  
Ctrl+D 复制行  
Ctrl+/ 或 Ctrl+Shift+/  注释（// 或者/\*...\*/ ）

Ctrl+Shift+Space 自动补全代码

Alt+ Up/Down 在方法间快速移动定位  
Ctrl+Shift+Up/Down 代码向上/下移动。

Shift+F6  重构-重命名

## 提示

Ctrl+空格 代码提示

Ctrl+Alt+Space 类名或接口名提示

Ctrl+P 方法参数提示

Alt+Shift+C 对比最近修改的代码

Ctrl+Shift+先上键

Ctrl+J  自动代码

Ctrl+E 最近打开的文件

## 显示

Ctrl＋F12，可以显示当前文件的结构

Ctrl+H 显示类结构图

Ctrl+Q 显示注释文档

Alt+1 快速打开或隐藏工程面板

Ctrl+Alt+ left/right 返回至上次浏览的位置

Alt+ left/right 切换代码视图

F2 或Shift+F2 高亮错误或警告快速定位

Ctrl＋P，可以显示参数信息

Ctrl＋Q可以看JavaDoc

Ctrl+Shift++/--

# AS问题

## 导入问题：

android studio中import所有的类文件都是红色的，不可用

答案：

File -> Invalidate Cashes / Restart

Error:(2, 0) Unable to load class 'org.apache.commons.lang3.StringUtils'.

Possible causes for this unexpected error include:<ul><li>Gradle's dependency cache may be corrupt (this sometimes occurs after a network connection timeout.)

<a href="syncProject">Re-download dependencies and sync project (requires network)</a></li><li>The state of a Gradle build process (daemon) may be corrupt. Stopping all Gradle daemons may solve this problem.

<a href="stopGradleDaemons">Stop Gradle build processes (requires restart)</a></li><li>Your project may be using a third-party plugin which is not compatible with the other plugins in the project or the version of Gradle requested by the project.</li></ul>In the case of corrupt Gradle processes, you can also try closing the IDE and then killing all Java processes.

Error:Could not find com.android.databinding:dataBinder:1.3.0-beta1.

Searched in the following locations:

file:/C:/Program Files/Android/Android Studio2/gradle/m2repository/com/android/databinding/dataBinder/1.3.0-beta1/dataBinder-1.3.0-beta1.pom

file:/C:/Program Files/Android/Android Studio2/gradle/m2repository/com/android/databinding/dataBinder/1.3.0-beta1/dataBinder-1.3.0-beta1.jar

https://jcenter.bintray.com/com/android/databinding/dataBinder/1.3.0-beta1/dataBinder-1.3.0-beta1.pom

https://jcenter.bintray.com/com/android/databinding/dataBinder/1.3.0-beta1/dataBinder-1.3.0-beta1.jar

Required by:

:AndroidDataBinding2:unspecified

Error:A problem occurred configuring project ':app'.> failed to find target with hash string 'android-22' in: C:\Users\Administrator\AppData\Local\Android\Sdk

1. Error:(20, 0) Could not find method android() for arguments [build\_7siny3fvhyb0dd53m3oxzypa2$\_run\_closure2@4aaa8785] on root project 'AndroidDataBinding2'.<a href="openFile:C:\Users\Administrator\Desktop\10pro\AndroidDataBinding2\build.gradle">Open File</a>

Gradle sync failed: Could not find method buildToolsVersion() for arguments [23.0.3] on root project 'AndroidDataBinding2'.

Consult IDE log for more details (Help | Show Log)

Error:The SDK Build Tools revision (23.0.3) is too low for project ':app'. Minimum required is 25.0.0

<a href="install.build.tools">Install Build Tools 25.0.0, update version in build file and sync project</a><br><a href="openFile:C:/Users/Administrator/Desktop/CC/app/build.gradle">Open File</a>