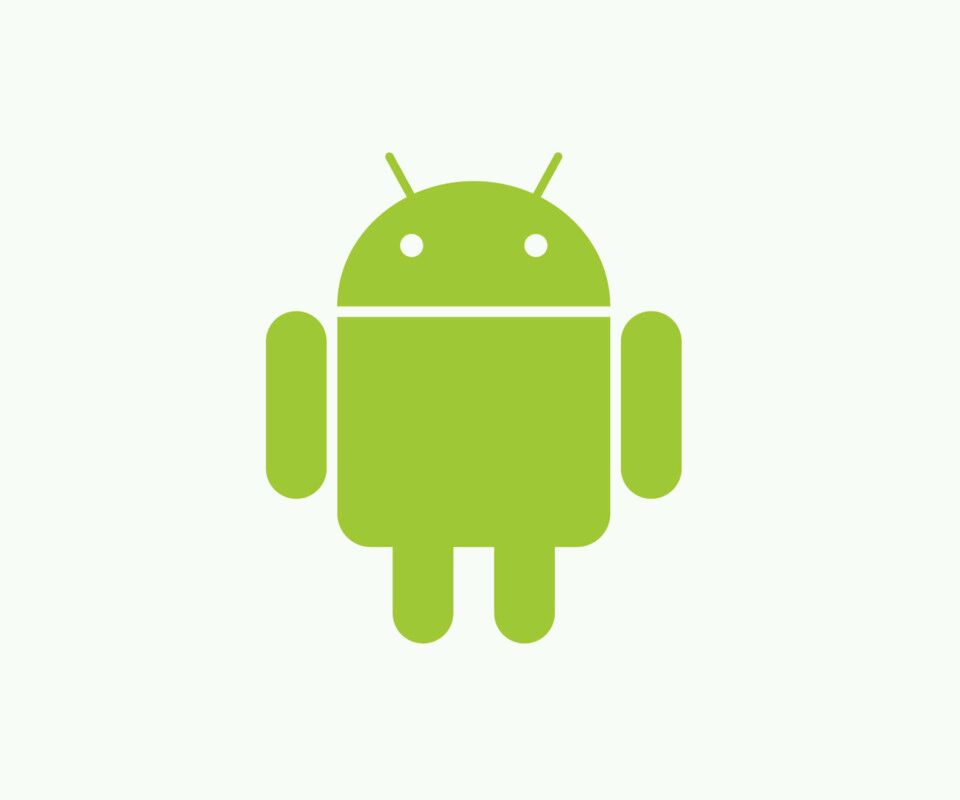
2016/7/5

|  |
| --- |
| 和而泰内部编码规范 | ~若相惜 |



|  |  |
| --- | --- |
| 和而泰家居在线 | Android编码规范 |

目录

[一、 前言 2](#_Toc456280459)

[二、 包 2](#_Toc456280460)

[三、 代码 4](#_Toc456280461)

[四、 资源 8](#_Toc456280462)

[五、 注释 10](#_Toc456280463)

[六、 附件 13](#_Toc456280464)

## 前言

1. 为什么需要开发规范

编码规范对于程序员而言尤为重要，有以下几个原因：

1. 一个软件的生命周期中，80%的时间花费在于维护
2. 几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护
3. 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新的代码
4. 如果你将源码作为产品发布，就需要确任它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品
5. 开发规范的作用
6. 减少维护花费
7. 提高可读性
8. 加快工作交接
9. 减少名字增生
10. 降低缺陷引入的机会

## 包

1. 包名全部采用小写，连续的单词只是简单地连接起来,不用下划线区分单词
2. 主包名采用反域名规则，一级包名为com/cn/org/edu/net，二级包名为xx(xx可以是公司或者个人的)，三级包名根据应用进行命名，四级包名为通用名称/模块名称/层级名称，前三级可以统称为[主包名]

例如：和而泰C-Life美容：com.het.cbeauty

注：和而泰android项目全部以com.het.XX形式命名包括工程和库

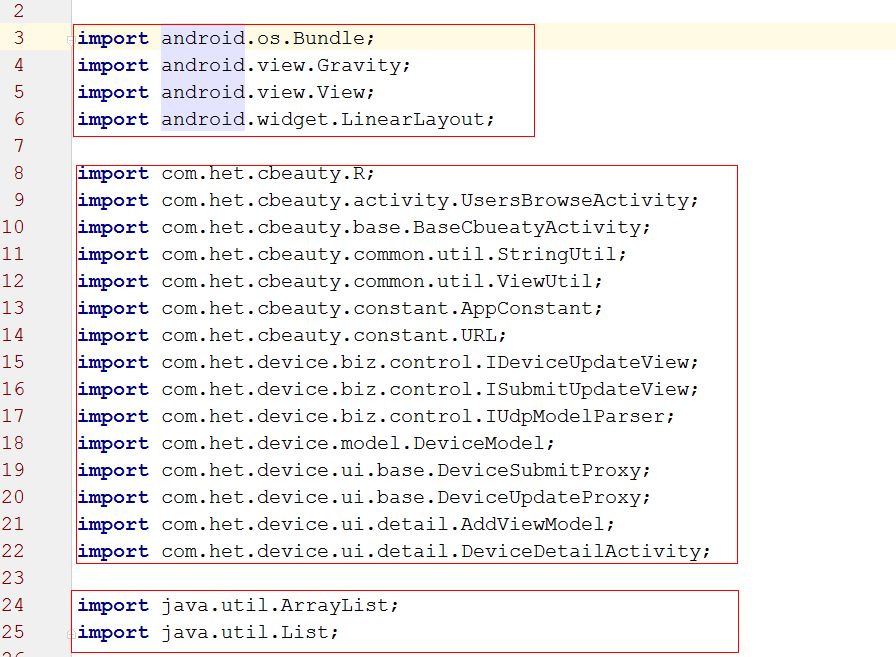
常见通用功能部分如下表：

|  |  |
| --- | --- |
| 包名 | 含义 |
| [主包名]. activity | 页面用到的Activity类 (activitie层级名用户界面层) |
| [主包名]. base | 基础共享的类 例如：BaseActivity、BaseFragment、BaseView等 |
| [主包名]. adapter | 页面用到的Adapter类 (适配器的类) |
| [主包名]. api | 网络接口请求相关 |
| [主包名]. constant | 常量配置 |
| [主包名].view/widget | 自定义View等 |
| [主包名].model/bean/entity | 实体类 |
| [主包名]. db | 数据库操作相关所在的包 |
| [主包名]. impl | 接口实现 |
| [主包名]. service | 服务相关 |
| [主包名]. manager | 管理类相关 |
| [主包名]. receiver | BroadcastReceiver服务 |
| [主包名]. util/tools | 公共工具方法类 |

1. 只需导入用到的类，不得用\*导入包下所有类

例如：import foo.\*; （×） import foo.Bar;（√）

1. 导入类时，系统类在上方，自定义类在下方



注：如果不想太麻烦：可以用快捷键：Ctrl+Shift+o 快捷某个类/某个包/整个工程src都可以

## 代码

1. 代码主要采用大/小驼峰命名法，即除首字母外，每个单词首字母大写，整体首字母大小根据其它规范决定
2. 类名、接口名、枚举名等首字母大写，若由多个单词组成，则其后每个单词首字母大写，简明扼要，望文知意原则；

例如：

class ConfigManager{}

1. 继承自安卓组件的类，采用父类名作为后缀

规则：

1. Acitivity类以Acitivity结尾；
2. Fragment类以Fragment结尾；
3. Service类以Service结尾；
4. BroadcastReceiver类以Receiver结尾；
5. ContentProvider类以Provider结尾；
6. Application类以Application结尾；
7. 自定义View类以Custom/功能\*\*View结尾；
8. 自定义Adapter类以Adapter结尾；
9. adapter中的ViewHolder以Holder结尾；
10. 实体Bean以Model结尾；

例如：

class LoginActivity extends Activity{}

1. 接口命名需要简单明了，长度不宜过长，建议在名称前面追加“I”

备注：

I\*\*Listener

I\*\*CallBack

I\*\*；

1. 自定义异常必须以Exception结尾
2. 除for循环变量外，一律不得使用i、j、k等单字符作为变量名

定义数组时方括号紧随在原始类型之后，数组名称一般使用复数形式

例如：

int[] arrays;

1. 常量、枚举等均采用大写形式，用下划线区分各单词

例如：

static final String DEVICE\_ID = "device\_id"

enum CounterType {NUMBER,DECIMAL,BOTH}

1. 全局变量添加所有者前缀：实例成员变量前缀m（表示member），静态字段命名以s开头（表示static）

例如：

实例变量mRun

静态变量sRun

1. 控件变量添加组件前缀，顺序在所有者前缀之后

例如：

全局名称：mBtnNext

局部名称：btnNext

常见控件前缀见附件UI控件表

|  |  |  |
| --- | --- | --- |
| 缩写 | 全称 | 含义 |
| alc | AnalogClock | 模拟时钟 |
| btn | Button | 按钮 |
| cal | CalendarView | 日历 |
| chb | CheckBox | 复选框 |
| chm | Chronometer | 秒表 |
| dgc | DigitalClock | 数字时钟 |
| dbk | DatePicker | 日期选择框 |
| et | EditText | 编辑框 |
| elv | ExpandableListView | 可扩展列表 |
| glr | Gallery | 画廊 |
| gv | GridView | 网格布局 |
| iv | ImageView | 图片控件 |
| lv | ListView | 列表控件 |
| mcr | MediaController | 多媒体控制 |
| npk | NumberPicker | 数字选择器 |
| rdg | RadioGroup | 单选按钮 |
| rl | RelativeLayout | 相对布局 |
| tl | TableLayout | 表格布局 |
| rdo | RadioButton | 单选按钮 |
| rtb | RatingBar | 评分控件 |
| scr | ScrollView | 滚动控件 |
| sdr | SlidingDrawer | 滑动抽屉 |
| sfc | SurfaceView | 渲染视图 |
| skb | SeekBar | 进度条 |
| spn | Spinner | 下拉框 |
| swh | Switch | 开关 |
| th | TabHost | 标签页 |
| tw | TabWidget | 标签按钮 |
| Tb | ToggleButton | 切换按钮 |
| tp | TimePicker | 时间选择器 |
| tv | TextView | 文本框 |
| vdo | VideoView | 视频 |
| web | WebView | 网页控件 |
| ll | LinearLayout | 线性布局 |
| fl | FrameLayout | 帧布局 |
| ibtn | ImageButton | 图片按钮 |

1. 常量一般使用final static修饰，根据需要使用可见性修饰符

例如：

public static final int VISIBLE = 0x00000000;

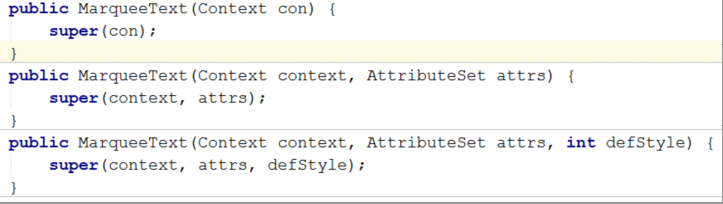
1. 一般方法名首字母小写，若由多个单词组成，则其后每个单词首字母大写，方法名通常是动词或动词短语。

例如：

|  |  |
| --- | --- |
| 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 |
| handleXX() | 对数据进行处理的方法，尽量使用handle为前缀标识 |
| displayXX()/showXX() | 弹出提示框和提示信息，使用display/show为前缀标识 |
| saveXX() | 与保存数据相关的，使用save为前缀标识 |
| 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 |

1. 构造方法采用递增方式（参数多的写在后面）

例如：



1. 实体类中不得随意修改的成员变量可添加下划线前缀以作区别

例如：

class User{public int \_id;}

1. 项目中一般不使用System.out输出，而是使用Log中的方法
2. 使用BuildConfig.DEBUG标记对Log进行封装，只在调试时输出重要信息，正式版不输出
3. 一般try……catch只捕获需要的异常，至少应当将异常信息输出
4. 代码中不出现中文，最多注释中可以出现中文，应用中的字符串统一在strings.xml中定义，然后在代码和布局文件中引用。
5. 编码方式统一用UTF-8 .Android Studio 默认已经是UTF-8,只要不去修改就可以了
6. 一个方法最多不要超过40行代码，源文件的行数不能大于2K行，过多的话可以考虑拆分功能，拆分函数等，尽量避免一行的长度超过 100 个字符。
7. 带有生命周期的组件一定要注意书写顺序

public class ClassName {

//1.成员变量集合

//2.回调方法集合

若该类为activity，则：onCreate、\*\*、onDestory；

若该类为Fragment、则：onCreateView、\*\*、onDestory；

//3.其他方法集合

}

## 资源

1. 资源命名全部采用小写，各单词间以下划线区分
2. 布局文件采用[前缀]\_[功能模块].xml的命名方式

例如：

MainActivity的布局activity\_main.xml

常见前缀如下表：

|  |  |
| --- | --- |
| 前缀名称 | 含义 |
| activity\_xx.xml | Activity对应的布局 |
| dialog\_xx.xml | 自定义对话框的布局 |
| item\_xx.xml | 适配器视图中每个项目的布局 |
| popup\_xx.xml | 弹出框的布局 |
| window\_xx.xml | 悬浮窗的布局 |
| fragment\_xx.xml | Fragment对应的布局 |
| 包含项命名：模块\_(位置)描述.xml (include) | 例如：activity\_main\_head.xml、activity\_main\_bottom.xml  注意：通用的包含项命名采用：项目名称缩写\_描述.xml  xxxx\_title.xml |

1. layout中的id命名

命名模式为：view缩写\_模块名称\_view的逻辑名称

1. 资源文件（图片drawable/mipmap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：可加后缀 \_small 表示小图, \_big 表示大图，逻辑名称可由多个单词加下划线组成，采用以下规则：  
用途\_模块名\_逻辑名称  
用途\_模块名\_颜色  
用途\_逻辑名称  
用途\_颜色  
说明：用途也指控件类型

例如：

btn\_main\_home.png 按键

divider\_maket\_white.png 分割线

ic\_edit.png 图标

bg\_main.png 背景

btn\_red.png 红色按键

btn\_red\_big.png 红色大按键

ic\_head\_small.png 小头像

bg\_input.png 输入框背景

divider\_white.png 白色分割线

常见属性后缀如下表：

|  |  |
| --- | --- |
| 后缀名称 | 含义 |
| btn\_xx\_selected | state\_selected选中效果 |
| btn\_xx\_checked | state\_checked选中效果 |
| btn\_xx\_focused | state\_focused聚焦效果 |
| btn\_xx\_pressed | state\_pressed按下状态时的图片 |
| btn\_xx\_disabled | 禁用状态下的图片state\_enabled (false)不可用效果 |
| btn\_xx\_normal | 常规状态下的图片 |
| btn\_xx\_hovered | state\_hovered悬停效果 |
| btn\_xx\_checkable | state\_checkable可选效果 |
| btn\_xx\_activated | state\_activated激活的 |
|  |  |
|  |  |
| btn\_xx\_selector | 包含所有状态的选择器 |
| 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 |
| ic\_more\_help | 图标图片使用ic\_功能\_说明 |
| di\_list\_line | 具有分隔特征的图片使用seg\_功能\_说明 (di: divider) |
| sel\_ok | 选择图标使用sel\_功能\_说明 |

注意：

使用AndroidStudio的插件SelectorChapek可以快速生成selector，前提是命名要规范。

1. 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。  
   具体动画采用以下规则：  
   模块名\_逻辑名称  
   逻辑名称  
   refresh\_progress.xml  
   market\_cart\_add.xml  
   market\_cart\_remove.xml  
   普通的tween动画采用如下表格中的命名方式  
   // 前面为动画的类型，后面为方向

例如：

fade\_in ：淡入 fade\_out： 淡出

1. values中name命名

|  |  |  |
| --- | --- | --- |
| 类别 | 命名 | 示例 |
| strings | strings的name命名使用下划线命名法，采用以下规则：  模块名+逻辑名称 | main\_menu\_about 主菜单按键文字  friend\_title 好友模块标题栏  friend\_dialog\_del 好友删除提示  login\_check\_email 登录验证  bind\_dialog\_title 绑定弹出框标题 |
| colors | colors的name命名使用下划线命名法，采用以下规则：  模块名+逻辑名称 颜色 | friend\_info\_bg  course\_bottom\_bar\_back  analysis\_skin\_value |
| styles | styles的name命名使用驼峰命名法，采用以下规则：  模块名+逻辑名称 | main\_tabBottom |

## 注释

1. 文件注释

/\*  
\* -----------------------------------------------------------------  
\* Copyright (C) 2007-2016, by het, Shenzhen, All rights reserved.  
\* -----------------------------------------------------------------

\*  
\*<p>描述：(这里用一句话描述这个类的作用)</p>  
\*名称: ${file\_name} <br>

\*作者: ~若相惜<br>  
\*版本: 1.0<br>  
\*日期: ${DATE} ${TIME}<br>  
\*/

1. 类定义一般需要写类注释，接口一般需要写接口注释，如果没有文件注释，则需要在类注释和接口注释中标出作者，包括类、接口的目的、作用、功能、继承于何种父类，实现的接口、实现的算法、使用方法、示例程序等例如：



模板：

/\*\*

\* <p>描述：(这里用一句话描述这个类的作用)</p>

\* 作者： ~若相惜<br>

\* 日期： ${DATE} ${TIME}<br>

\* 版本： v2.0<br>

\*

\*/

1. 成员变量、静态变量、常量等添加属性注释，例如：

/\*\* This view is invisible. \*/

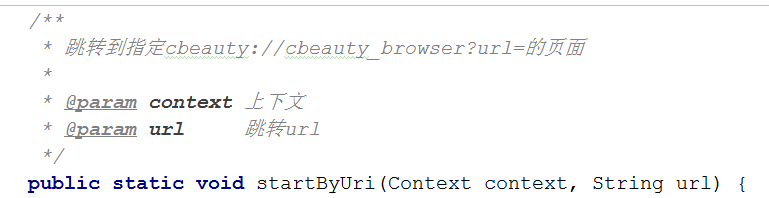
public static final int INVISIBLE = 0x00000004;

1. 关联性较大的多个成员变量等可以共用同一条注释，例如：

/\*\* The width and height of View. \*/

private int mWidth, mHeight;

1. public和protected方法必须添加方法注释，default和private方法建议添加方法注释，例如：



模板：

/\*\*

\* <p>描述:${todo}(这里用一句话描述这个方法的作用)</p>

\* ${tags} 设定文件

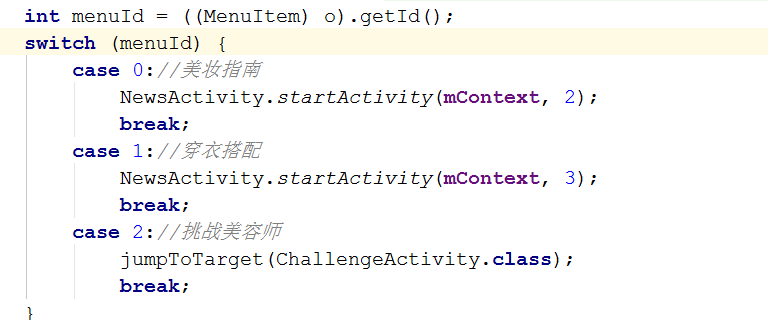
\*/

1. 若覆盖基类的方法，则可以不写方法注释，但必须用@Override标出，例如：

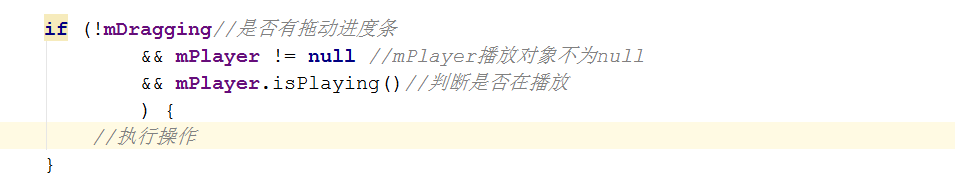
@Override

protected void onCreate(Bundle savedInstanceState) {}

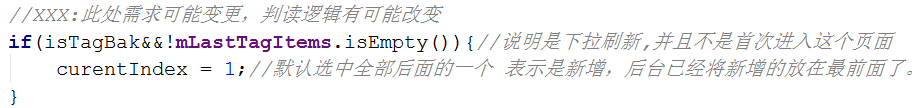
1. switch……case的每个条件一般添加简短说明，例如：



1. 如果if的条件大于2个，则必须写注释，例如：



1. 若功能已完成，但存在效率等潜在问题时，使用XXX加以标记，例如：



1. 规范的注释方法
2. 一个工程应有一个统一的头文件注释，以说明整个工程的信息、创建日期、版本等等
3. 对重要的程序必须加注释进行说明
4. 修改代码或删除时，将原代码用注释的方法屏蔽，同时要加开发者自身对修改操作的注释
5. 移除所有临时或无关的注释，以避免在日后的维护工作中产生混乱
6. 注释应对代码进行准确的说明，不应存在歧义
7. 在整个应用程序中，使用具有一致的标点和结构的统一样式来构造注释

## 附件

1. 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 | sel（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） |
| 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 |
| infomation | info |
| initial | init |
| image | img |
| library | lib |
| message | msg |
| string | str |
| temp | tmp |