# Android代码规范

有的时候想看懂别人命名的那个控件什么是什么用的，要读好几遍才知道，在代码的审查和修改过程中就会浪费不少不必要的时间。如果就是一个人开发，一个人维护的话还好，可是如果一个项目是团队分工合作，这样让同事去看代码就更加吃力了，因为大家之间的编码方式不一样，所以，在开发过程中，命名规范统一尤为重要，最好是团队中统一好大家命名方法，这样对于日后的工作会轻松很多。

所以，下面我就总结下我从网上别人的经验和自己的感悟中收集总结的android开发命名规范，希望对大家有所帮助。

提醒：在命名规范的统一下，再**加上注释**，这样日后维护会高效快捷好多。

## **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** | **缩写与备注** |
| icon | icon （主要用在app的图标） |
| color | color（主要用于颜色值） |
| divider | div（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） |
| 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 |
| length | len |
| library | lib |
| message | msg |
| password- | pwd |
| position | pos |
| server | srv |
| string | str |
| temp | tmp |
| window | 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(m)，四级包名为模块名或层级名

|  |  |
| --- | --- |
| **包名** | **此包中包含** |
| com.xx.m.activitiy | 页面用到的Activity类 (activitiy层级名用户界面层) |
| com.xx.m.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.m.adapter | 页面用到的Adapter类 (适配器的类) |
| com.xx.m.tools | 此包中包含：公共工具方法类（tools模块名） |
| com.xx.m.bean  (或com.xx.m.enity ) | 此包中包含：元素类 |
| com.xx.m.db | 数据库操作类 |
| com.xx.m.views | 自定义的View类等 |
| com.xx.m.service | Service服务 |
| com.xx.m.broadcast | Broadcast服务 |
| com.xx.m.config | 里面包含项目的配置，常量类等 |
| com.xx.m.interfaces | 包含抽象类、回调、监听等类 |

### 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 | Activity为后缀标识 | 欢迎页面类SplashActivity |
| Adapter类 | Adapte 为后缀标识 | 新闻详情适配器NewDetailAdapter |
| 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 |  |
| 共享基础类 | 以Base开头 | BaseActivity,BaseFragment |
| Application |  | MyMyApplication |

### 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 |
| setXX() | 赋值使用的方法，用set为前缀 |
| doXX() | 对数据进行处理的方法，使用do为前缀标识 |
| displayXX()或dialogXX() | 弹出提示框和提示信息，使用display或dialog为前缀标识 |
| saveXX() | 与保存数据相关的，使用save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 资源文件名（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，如按钮背景 **btn\_xx.xml**

|  |  |
| --- | --- |
| **名称** | **功能** |
| btn\_xx | 按钮图片使用btn\_整体效果 |
| 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\_xx | 背景图片使用bg\_功能\_说明 |
| def\_xx | 默认图片使用def\_功能\_说明 |
| icon\_xx | 图标图片使用icon\_功能\_说明 |
| seg\_xx | 具有分隔特征的图片使用seg\_功能\_说明 |
| sel\_xx | 选择图标使用sel\_功能\_说明 |

### 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：activity\_功能模块.xml 例如：activity\_main.xml、activity\_more.xml

2)．Dialog命名：dialog\_描述.xml

       例如：dialog **\_hint.xml**

2)．PopupWindow命名：ppw\_描述.xml

       例如：**ppw \_xx.xml**

3). 列表项命名listview\_描述.xml

       例如：listview**\_city.xml**

4)．包含项：include\_模块.xml

       例如：**include\_xx.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** | **例如** |
| LayoutView | lv | lvMine |
| RelativeView | rv | rvMine |
| TextView | tv | tvName |
| Button | btn | btnSend |
| ImageButton | imgBtn | imgBtnDel |
| ImageView | img | imgPic |
| CheckBox | chk |  |
| RadioButton | rdoBtn |  |
| analogClock | anaClk |  |
| DigtalClock | dgtClk |  |
| DatePicker | dtPk |  |
| EditText | edit |  |
| 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 | lv |  |
| ExpandableList | epdList |  |
| 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

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图处理

**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.开发自测

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；

19.服务端响应数据解析:

个人觉得最好将数据格式规范化，固定的格式，少用嵌套。如

详情数据：HashMap<String, Objiect> mapData

列表数据：List< HashMap<String, Objiect>> listData

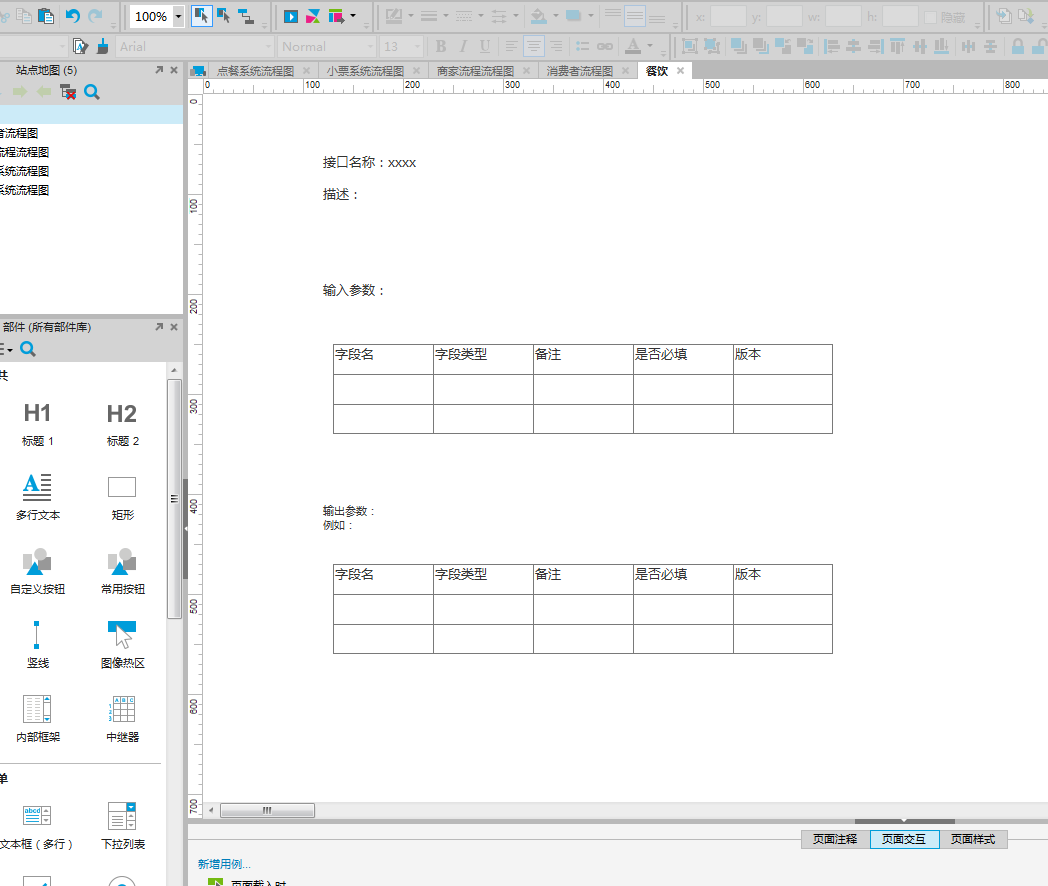
和ArrayList<HashMap<String, Objiect>> listData

少用实体类存储数据。

# 业务文档

## 接口文档

推荐使用Axure Rp工具，编写一目了然，可以创建SVN项目，多人修改。



## 原型文档

同样用Axure Rp工具编写。

