# Android代码规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 操作人 | 内容 | 创建日期 | 备注 |
| 1 | 李腾飞 | 创建文档 | 2016/3/17 |  |
| 2 | 李腾飞 | 完成初稿 | 2016/3/21 |  |
| 3 | 李腾飞 | 增加一些注意事项 | 2016/3/25 |  |
| 4 | 李腾飞 | 增加一些关于封装 | 2016/3/28 |  |
| 5 | 李腾飞 | 增加SVN分支和主干问题 | 2016/3/29 |  |
| 6 | 李腾飞 | 修改令人费解的句子 | 2016/3/31 |  |

**package，versionCode，versionName命名、**

示例：

package=*"com.xx.pos"*

android:versionCode=*"10"*

android:versionName=*"3.7"*

注1：**versionCode** 。

Android系统是以**versionCode** 判断是不是新版本，而不是**versionName**，所以我们判断版本更新的条件也应该是**versionCode** ，**versionName**只是一个展示的版本名称。

一般第一个版本**versionCode** 从整数型1开始（也可以以我们自定义的一个整数开始，但是这要求我们必须做好记录），每开发一个新版本一般**versionCode** 值+1（当然也可以+n）。

建议第一版versionCode 从整数型1，没开放一个新版本versionCode 值+1，以方便我们版本控制。

注2：**versionName**

**versionName** 的命名相对没有那么严格，我们一般取三位的0.0.0，也可以是大于三位的。

**建包，分包，包名**

示例：



如果，我们已在安装的清单文件命名了package=*"com.tianlian.pos"*，在pos下分包时，建议以模块的分包和建包，包名严格遵循java命名原则，全小写，不必有分隔符。

安卓四大组件中**Service** 和**BroadcasterReceiver**属于整个工程的。

因此，除了模块包需要建立，还需要建立：

1. **receiver**(广播接收者)；
2. **base**（因为我们会写一下基类或者全局变量访问的继承Application的类）；
3. **service**（服务）；
4. **main**(放置**SplashActivity**(欢迎页，程序的入口)，**GuideActivity**(引导页)，**MainActivity**（主页Activity)）；
5. **model**（存放网络请求数据的实体类），
6. **utils**(工具包)；
7. **constant**（存放接口常量，主机地址）。

在模块包建好以后，在模块包下分别建立

1. **act**(存放Activity)；
2. **fra**(存放Fragment)；
3. **adp**(存放适配器Adapter)；
4. **view**（存放自定义控件）。

**示例如下**



注3：

**在不影响原意的前提下，一些包名的单词，我取了缩写，以方便我们阅读。**

**类的命名**

1. **Activity**命名规范：以**Act**作为后缀。比如说**PersonAct**。
2. **Fragment**命名规范：以**Frg**作为后缀。比如说**PersonFrg**。
3. **Adapter**命名规范：以**Adp**作为后缀。比如说**PersonAdp**。
4. **Model**层实体类命名规范：可能有些PC开发喜欢以**Entity**命名，因为实体类在Java里面叫**JavaBean**,所以我建议以Bean作为后缀，比如说**PersonBean**。

其他包下的命名类同.

## 资源文件命名规范

**1. layout目录下的文件命名规范：**

* 页面布局文件。以**act\_**为前缀，以**Activity**的名称(去掉**Activity**后缀)作为后缀。注意都是小写。

**例：**对于**Person**这个模块下的**AddCustomerActivity**，它的layout文件就应该是：**act\_addcustomer.xml。**

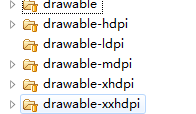
* **ListView**中的**item**布局文件。以**item\_**作为固定前缀，列表项的名称为后缀。注意都是小写。

**例**：某个页面下有一个用户列表，控件名为**lvUserList**，那么**item**的**layout**就应该是：**item\_user\_list.xml。**

* **Dialog**布局文件

以**dlg\_**作为固定前缀，**Dialog**的功能名称为后缀。注意都是小写，例如：**dlg\_hint.xml**。

**2.drawable目录下文件命名规范。**



**drawable**文件夹下，尽量不要放置图片，而要存放如**selector**,**shape**等**xml**文件，以**selector,shape**为前缀，如：

**3.Java类中控件对象的命名规范。**

控件类型缩写+控件的逻辑名称(首字母大写)，比如登录按钮，就可以命名为btnLogin。

表列出了一些常用控件的缩写。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 控件 | 缩写 |  | 控件 | 缩写 |
| LayoutView | lv |  | EditText | et |
| RelativeView | rv |  | TimePicker | tp |
| TextView | tv |  | toggleButton | tb |
| Button | btn |  | ProgressBar | pb |
| ImageButton | imgbtn |  | webView | wv |
| CheckBox | chk |  | Tab | tab |
| RadioButton | rb |  | ListView | lv |
| DatePicker | dp |  | MapView | mv |

**4.Layout中控件对象的命名规范。**

个人建议，与**Activity**中相对应的控件名称保持一致。这样的好处是可以迅速**copy-paste**出以下代码而杜绝任何的潜在错误：

Button btnLogin = (Button)findViewById(R.id.btnLogin)；

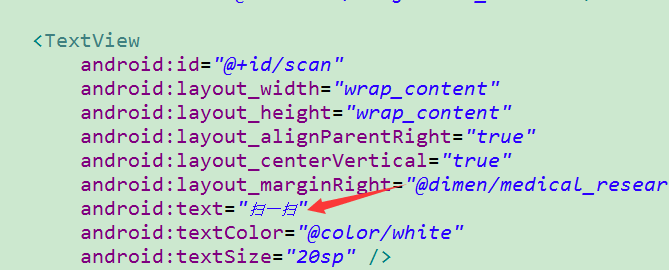
**5.strings.xml中常量的命名规范。**

因为这些值大多在Layout中的控件上使用，个人建议，与Activity中相对应的控件名称保持一致,例如btnLogin



注4：

**务必不出现直接在xml中写中文，如：**



**6.colors.xml注意的地方**

颜色统一写到colors.xml,在布局文件里引用，不能直接在布局文件里面，写16进制的色值。

**屏幕适配一些要注意的问题**

**1，切图**

应用图标，3套，主页如果采用底部按钮切换的话，底部按钮，也切三套

除了这两个之外，切一套图，1920\*1080分辨率，切多套的图的话，虽然UI会更完美，但是APK体积很大，这个分辨率，向下压缩图片不会丢失像素，向上的拉伸也不会丢失太多像素。

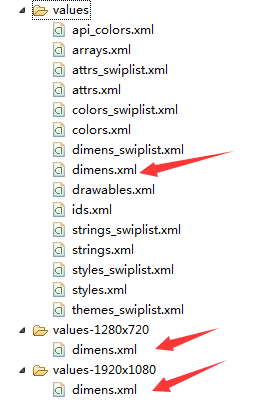
注5：

如果页面上是色块，可以用色值代替切图，减轻美工工作量，同时减少包的提交，提升APP的运行速度。

APP用到的图片格式多为**png**格式，谷歌最近推出**WebP**的图片格式 ,体积比png更小。

**2，控件的宽高**

如上图，如果页面上有等分的布局，一般采用线性布局权重去按比例分，其他情况，设置控件的高度使用**dp,**针对不同分辨率，创建多个**dimens.xml**如：



**代码规范**

**面向接口编程，项目里面不再采用基类继承的开发模式。**

**除非，多个页面有相同的地方。**

前言说明：所有的项目统一utf-8编码，如果从网上copy的代码，请修改编码后，再引入到项目里面

**1.规范代码**

一个页面显示出来，一般需要两步，一是初始化控件，二是加载数据。因此我们定一个接口**FromatCodeInterface**，所有的页面实现这个接口，并实现接口的两个方法：

* **initViews**：加载**layout**布局文件，初始化控件。
* **loadData**：加载数据。

注6：

**不采用基类继承的方式，这是因为我们采用面向接口编程的思想。**

**2.统一Json解析工具：fastjson.jar**

**3.全局访问的变量或者常量，**

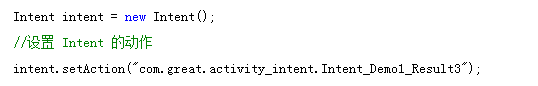
可放置在**Application**的继承类，严禁在**Activity**或者**Fragment**里设置静态引用对象。（**基本类型的静态变量可以适当使用，但不宜过多**）

1. **严禁在Layout的控件**

声明中直接声明点击事件方法，请不要使用以下代码：

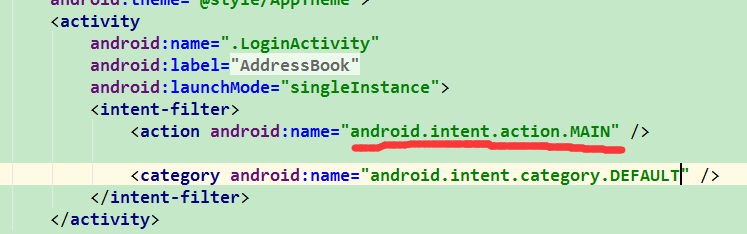
<Button android:onClick="gotoLogin" …… /> ，此处与使用注解式框架不同

**5.建议Activity的跳转采用隐式跳转，**



action可以自定义

只需在清单文件里面配置intent-filter如：



注7：

**所用动作都保存在Application的继承类，方便全局访问**

**6.数据的存储**

如果需要用到**HashMap**,使用谷歌推荐的**SparseArray**。

**7.图片缓存和网络请求框架（待定）**

统一采用OkHttp和Facebook开源的库Fresco来进行异步加载。

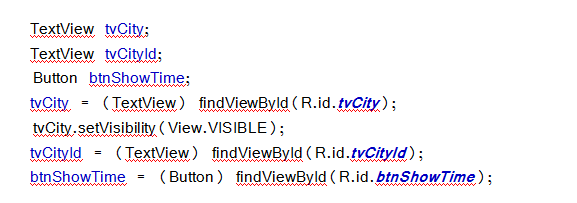
关于如何使用，请点击该链接：<http://fresco-cn.org/>

注8：

OkHttp是网络层的框架，Facebook底层使用的它，建议配合使用，使用图片加载的库Fresco的还有一个好处就是可以加载webp格式的图片，这点非常之好。

**该框架学习成本非常低，5分钟便可掌握自如，且有诸多好处，个人强烈推荐。**

## 要怎么使用findViewById语句



上图所示，这是我们经常会做的操作。

其实，可不必对每个控件都需要给定一个变量名，如果我们只是需要使用这个控件的点击事件，我们只需要

findViewById(R.id.***tvCity***)，

然后再

onClick处理R.id.***tvCity。***

**同样的，可不必把每个控件都制成成员变量。**

**若只是在方法里面使用，只需置成局部变量即可。**

**9.关于注解式框架的使用**

注解式框架的使用时为了节省代码量，让代码更整洁，可以采用。

反射式的注解尽量不要使用，影响性能，如xutils,final,但是编译时注解可以使用，对性能没有影响，好处是代码整洁，方便阅读，如butternife,drugger(谷歌出品)

**10.Dialog封装和使用**

**Dialog**经常在项目中用到，为了代码复用，以前我们可能考虑去封装一个Dialog，现在谷歌推荐我们使用**DialogFragment**去替代**AlertDialog**和**Dialog**原因有以下几点：

使用**DialogFragment**来管理对话框，当旋转屏幕和按下后退键时可以更好的管理其声明周期，它和Fragment有着基本一致的声明周期。且**DialogFragment**也允许开发者把**Dialog**作为内嵌的组件进行重用，类似**Fragment**（可以在大屏幕和小屏幕显示出不同的效果）。上面会通过例子展示这些好处~

使用**DialogFragment**至少需要实现**onCreateView**或者**onCreateDIalog**方法。**onCreateView**即使用定义的**xml**布局文件展示Dialog。**onCreateDialog**即利用**AlertDialog**或者**Dialog**创建出**Dialog**。

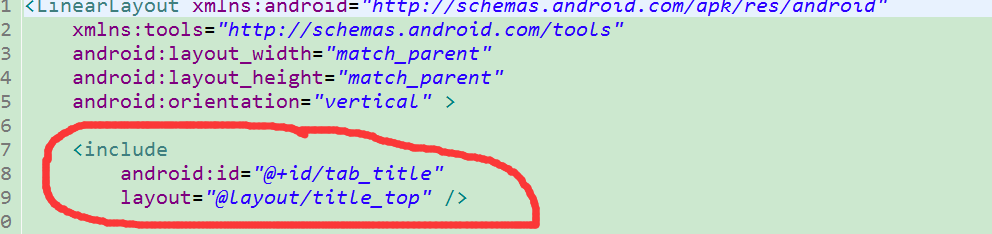
注：我们写好一个**DialogFragment**的继承类，**setArguments**传入参数，就可以做适当的改变。

1. **SVN上分支与主干开发**

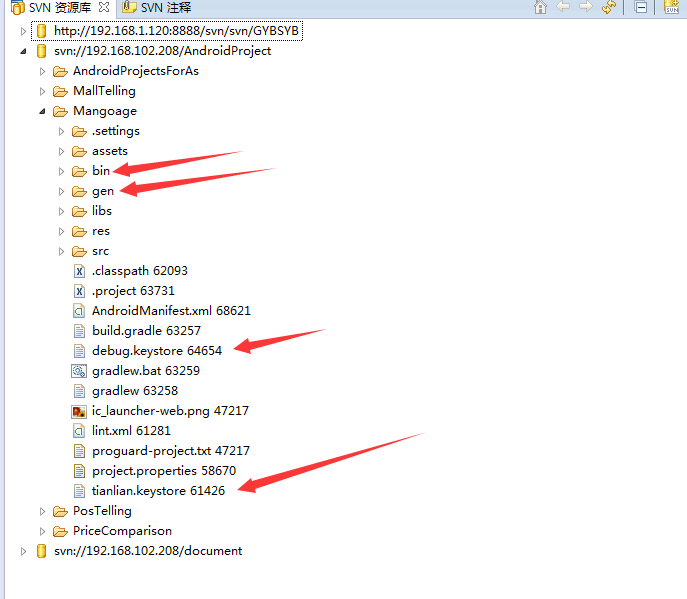
主干维持随时可发布的版本，开发新版本的话，单独拉出来一个分支。这点切记！

1. **布局文件复用**

有的时候，很多页面的头部都是一样的，我们可以只用**include**做嵌套，不同的地方，在代码里面做控制，如下图



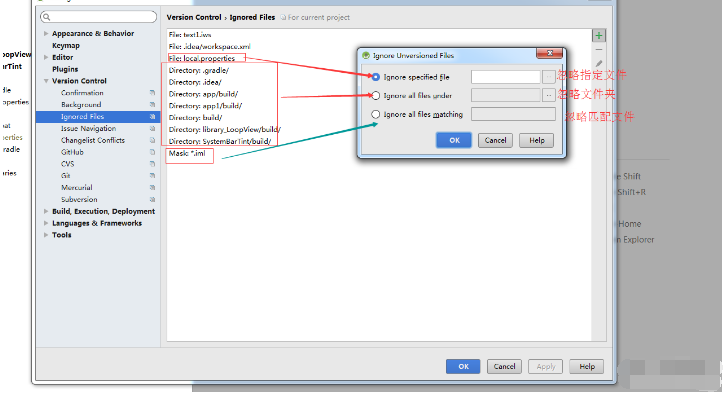
**Eclipse和SVN的使用**



上图是天联智家的项目，存在两个问题，1**.bin**目录和**gen**目录下是自动生成的文件不要提交 2..**keytore**文件是**apk**打包的密钥，不要提交**svn**，更不要放置在工程里面，一旦被人反编译出来，拿着你的密钥，后果非常危险。

**Android Studio 和SVN的使用**

**Android Studio上传svn，同样需要忽略很多自动生成文件，具体对照下图：**



**代码质量控制**

**注：完成一个小功能就提交，多次提交，不要写了非常独到的代码再提交，每次提交svn，写清楚注释，改了什么功能，或者修改了什么bug.**

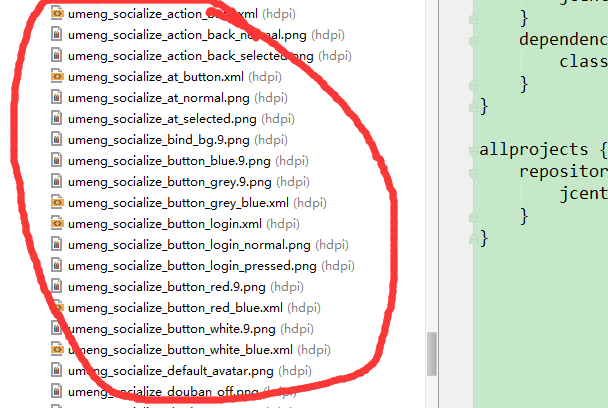
1. **svn提交代码**

往svn提交代码前，务必保证代码编译问题且自测没问题，再上传svn,不提交和项目无关的文件

1. **代码审核**

鉴于目前安卓组就两个人，个人建议，在提交svn前，相互审核彼此的代码，包括代码的整洁度，代码的冗余度，代码的效率，以及业务逻辑是否合理。

1. **第三方的jar以及第三方源代码的引入**



如上图所示：

**红线圈住的是友盟的资源文件，因为引入是jar包，如果友盟有升级sdk，那么可能需要改动这些资源文件，但是如果以库文件形式引入的话，只需要换成最新的库文件，其他地方无需改动**

原则上，不建议在自己工程里面引入过多的第三方jar，或者源代码，如果业务上需要，建议双方讨论之后再以库文件或者jar的形式引入，

**4.关于Android Studio**

Android Studio与Eclipse有很大不同，首先Android Studio（以下简称as）代码检查机制比Eclipse更严格，这对我们编写代码是好事，也要求我们规范代码

1. **数据库问题**

如果客户端需要缓存数据，用到数据库，需要数据库的版本控制，以及跨版本升级问题。

客户端数据的管理，为防止**sql**注入，建议使用**greendao**开源框架。

1. **安全问题**

包括Apk代码混淆和接口安全问题，首先混淆代码是必须的，但是假如我们的代码被反编译，接口访问地址被拿到，那么接口会不会被人恶意访问，以及传送数据的是否安全，需要前后端特别留意的。