|  |  |
| --- | --- |
| 文档名称 | 密级 |
| Android开发手册 | 机密 |
| 文档版本 | 共19页 |
| 1.0 |

Android开发手册

拟制 日期 2017-7-14

审核 日期

批准 日期



**北京君德财富投资管理股份有限公司**

**北京微金客科技有限公司**

版权所有 侵权必究

（仅供内部使用）

修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 日期 | 修订版本 | CR号 | 修改章节 | 修改描述 | 作者 |
| 2017-7-14 | 1.0 |  |  | 初稿完成 | 徐伟伟  崔新玥  洪瑞芬 |
|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

[1.概述 3](#_Toc28537)

[1.1 前言 3](#_Toc19800)

[1.2 目的 3](#_Toc14132)

[2.开发准备 4](#_Toc2916)

[2.1开发工具使用android studio。 4](#_Toc17656)

[2.2代码管理使用管理工具svn进行管理。 4](#_Toc29570)

[3.代码风格 4](#_Toc7429)

[3.1 文件格式和文件类型 4](#_Toc27720)

[3.2 程序块缩进 4](#_Toc6538)

[3.3 程序块空行 4](#_Toc1368)

[3.4 长语句与长表达式 4](#_Toc22819)

[3.5 长参数 4](#_Toc594)

[3.6 短语句 5](#_Toc6783)

[3.7 大括号{}的使用 5](#_Toc21111)

[3.8 自动换行 5](#_Toc13827)

[3.9 变量声明 5](#_Toc563)

[3.10 参数和返回值 6](#_Toc11676)

[3.11 import语句 6](#_Toc24102)

[3.12 使用标准的Java Annotation 7](#_Toc19496)

[3.13 Java异常的处理 8](#_Toc31600)

[3.14 其它 11](#_Toc22811)

[4.命名规范 11](#_Toc14145)

[4.1包名全部小写 11](#_Toc19015)

[4.2类名采用大驼峰命名法 11](#_Toc4093)

[4.3接口/抽象类命名方式和类名一样 12](#_Toc27892)

[4.4资源文件 12](#_Toc3191)

[4.5类方法采用 小驼峰 命名法 14](#_Toc2218)

[4.6控件id命名 15](#_Toc18958)

[4.7变量名采用小驼峰命名 16](#_Toc20827)

[5.编码规范 16](#_Toc20662)

[5.1 Activity当中的oncreate方法分为三个部分 16](#_Toc27602)

[5.2 布局的适配方案 17](#_Toc1066)

[5.3 activity之间的传值 17](#_Toc7818)

[5.4 数据承载体的使用 17](#_Toc994)

[5.5 控件的点击事件由代码控制 17](#_Toc23136)

[5.6 SharePreferences的建议 17](#_Toc6017)

[5.7 圆角图片的处理 17](#_Toc25024)

[5.8 本地数据库的操作 18](#_Toc29645)

[6 上线打包注意的细节 18](#_Toc18998)

## 1.概述

### 1.1 前言

团队协作能够显著有效的提高工作效率，而代码规范在团队协作当中也起了很重要的作用；这份文档参考了[Google Java 编程风格规范](http://www.hawstein.com/posts/google-java-style.html)和[Google官方Android编码风格规范](http://www.360sdn.com/2013/android_0517/144.html)。该文档仅供参考，适用于本公司整个android团队。

### 1.2 目的

形成统一的代码规范，减少编码过程中bug的数量，增加代码的观赏性、扩展性以及可维护性。

## 2.开发准备

### 2.1开发工具使用android studio。

### 2.2代码管理使用管理工具svn进行管理。

## 3.代码风格

### 3.1 文件格式和文件类型

utf-8格式和java文件。

### 3.2 程序块缩进

不以tab进行缩行处理，以space空格键进行缩行

### 3.3 程序块空行

a.空行将逻辑相关的代码块分隔开，以提高可读性。

b.一个源文件的两个片段之间。

c.类声明和接口声明之间。

d.两个方法之间。

e.方法内的布局变量与方法的第一条语句之间。

f.一个方法内的两个逻辑段之间。

### 3.4 长语句与长表达式

循环或判断当中若是有较长的表达式或语句，要进行适当的划分，一般以操作符为分割对象，操作符位于分隔行之首。

### 3.5 长参数

方法当中有多个参数时，以方法内部逗号进行划分，逗号位于分隔行之首。

### 3.6 短语句

每行只能存在一条语句，每条语句结束后应当进行换行处理。

### 3.7 大括号{}的使用

a.大括号一般与if else 、for、do while语句一起使用，即便是一条空语句也应该把大括号加上。

b.左大括号前不换行、左大括号后换行、右大括号前换行

c.如果右大括号后是一个语句、函数体或类的终止，则右大括号后换行，否则不换行。

|  |
| --- |
| public void method() {//Good  }  public void method(){//Bad  }  public void method()//Bad {  } |

### 3.8 自动换行

一般情况下，一行长代码为了避免超出列限制（80或100个字符）而被分成多行，很多时候，对于同一串代码已经有了好几种有效的自动换行方式。

### 3.9 变量声明

不要使用组合声明。如：int i, j,k;

|  |
| --- |
| int i, j, k;//Bad  int i;//Good  int j;//Good |

### 3.10 参数和返回值

a.一个方法的参数尽量不易过多，如若过多，做好换行处理

|  |
| --- |
| public void method(String params01 , String params02  , String params03…) {  }//good |

b.一个方法返回值是错误码，用异常处理去替代它

c.尽可能不要使用null，替代为异常或者空变量，List则可以返回Collections.emptyList形式。

### 3.11 import语句

a.impory语句不使用通配符,不换行。

|  |
| --- |
| import java.lang.\*;//Bad  import java.lang.string;//Good |

b.顺序和间距

|  |
| --- |
| Import语句可以分为以下几组，按照这个顺序，每组由一个空行分隔：  所有的静态导入独立成组  com.google imports(仅当这个源文件是在com.google包下)  第三方的包。每个顶级包为一组，字典序。例如：android，com，jnuit，org，sun  Java imports5.javax imports组内不空行，按字典序排序 |

### 3.12 使用标准的Java Annotation

a.Annotation位于java语言其它 修饰符之前。简单的marker annotation（@Override等）可以和语言元素放在同一行。如果存在多个annotation，或者annotation是参数化的，则应该按字母顺序各占一行来列出。

|  |  |
| --- | --- |
| 内建三种Annotation | Annotation注解内容 |
| @DePrecated | 只要某个语言元素已经不再建议使用了，必须  使用@DePrecated annotation。如果使用了@Deprecated annotation ,则必须同时进行@Deprecated javadoc标记，并且给出一个替代的实现方案。此外，@Deprecated 的方法仍然是能正常执行的。 |
| @Override | 只要某个方法覆盖了已过时的或继承自超类的方法，就必须使用@Override annotation。 |
| @SuppressWarnings | @SuppressWarnings annotation 仅用于无法消除编译警告的场合。如果警告确实经过测试”不可能消除”,则必须使用@SuppressWarnings annotation,已确保所有的代码警告都能真实反映代码中的问题。当需要使用@SuppressWarnings annotation时，必须在前面加上TODO注释行，用于解释“不可能解除警告”的条件 |

b.Todo注释一般应用于未完成的功能区域。

c.对那些临时性的、短期的、够棒但不完美的代码，使用TODO注释

TODO注释应该包含全部大写的TODO，后跟一个冒号：

//TODO: Remove this Code for version code

//TODO: Change this Code for reason

d.注释的语言当前以中文为主。

### 3.13 Java异常的处理

a.不要忽略异常，即使你的代码看起来永远不会出现这种情况或者处理这种异常并不重要，但忽略异常的这种行为会在代码中埋下一颗定时炸弹，so，在代码中以某种规矩处理所有的异常。

|  |
| --- |
| public void stringToInt（String value）{  try{  int intValue = Integer.parseInt(value);  }catch(NumberFormatException){  }  }  🡪空的catch语句在java中会让人感到忧虑，绝对不要这样做，替代方案有以下几种。  向方法的调用者抛出异常  public void stringToInt(String value) throws NumberFormatException{  int intValue = Integer.parseInt(value);  }  根据抽象级别抛出新的异常。  public void stringToInt(String value) throws SelfDefineException{  try{  int intValue = Integer.parseInt(value);  }catch(NumberFormatException){  throw new SelfDefineException(“value”+invalid number);  }  }  默默的处理错误并在catch中替换为合适的值。  public void stringToInt（String value）{  int intValue;  try{  intValue = Integer.parseInt(value);  }catch(NumberFormatException){  intValue = 0;  }  }  捕获异常并生成一个新的RuntimeException。  public void stringToInt（String value）{  try{  int intValue = Integer.parseInt(value);  }catch(NumberFormatException){  throw new RuntimeException(“value”+invalid number);  }  }  如果忽略异常比较合适，那就忽略吧，但得加上说明注释。  public void stringToInt（String value）{  try{  int intValue = Integer.parseInt(value);  }catch(NumberFormatException){  //number format just ignore invalid data  }  } |

b.不要捕获顶级的Exception

|  |
| --- |
| 绝大部分情况下，捕获顶级的Exception或Throwable都是不合适的，Throwable更不合适，因为它还包含了Error异常。  try{  ioOperationMethod();//may be throw IOException  classCastOperationMethod();//may be throw CastException  objectInvokeMethod();//may be throw NullPointException  }catch(Exception e){  handlerError();//just handler with one operation  }  不可直接全部捕获为Exception异常，正常的处理方式为：  分开捕获每一种异常，在一条try语句后面跟随多个catch语句块。  再次抛出异常，很多时候我们不需要捕获这个异常，只要让这个方法抛出这个异常即可。 |

### 3.14 其它

## 4.命名规范

### 4.1包名全部小写

连续的单词只是简单的链接起来，不适用下划线；采用反域名命名规则，一级包名为公司性质，二级包名为公司名或个人，三级包名为应用，四级包名为模块名或层级名等，例如：com.vjinke。

### 4.2类名采用大驼峰命名法

既采用功能名+类型名的命名方式,尽量避免缩写，除非该缩写是众所周知的；如果类名众包含缩写单词，则单词缩写的每个字母均为大写。对于继承自安卓组件的类来说，类名应该以该组件名结尾，例如 ： SignInActivity , SignInFragment , ImageUploaderService , ChangePasswordDialog ;对于工具类来说，命名方式应该以其完成功能开始,以 Utils 结束 ，例如 ： HttpUtils , ImageUtils .

|  |  |  |  |
| --- | --- | --- | --- |
| 类 | 包名 | 命名格式 | 示例 |
| Activity | com.vjinke  .ui.activity | 功能名+activity | MainActivity  LoginActivity… |
| Service | com.vjinke  .service | 功能名+Service | DownLoadService… |
| BroadCastReceiver | com.vjinke  .service | 功能名+Receiver | JpushReceiver  AlarmReceiver… |
| Fragment | com.vjinke  .ui.fragment | 功能名+Fragment | MainFragment  FoundFragment… |
| Dialog | com.vjinke  ui.widget | 功能名+Dialog | CustomDialog  RoundProgressBar… |
| Adapter | com.vjinke  ui.adapter | 功能名+Adapter | GoodsAdapter  CouponAdapter… |
| 基础功能类 | com.vjinke  .app | Base+父类名 | BaseActivity  BasePresenter… |
| 工具类和管理类 | com.vjinke  .ui.utils | 功能名+Utils/Manager | DbUtils  FileUtils… |

### 4.3接口/抽象类命名方式和类名一样

一般在接口类前加上大写的I/A，表示这是一个Interface/Abstract类。例：IView,Ipresenter…

### 4.4资源文件

以小写加下划线\_的方式命名

#### 4.4.1 drawable文件的命名规范：

|  |  |  |
| --- | --- | --- |
| Asset Type | Prefix 前缀 | Example |
| Action bar | ab\_ | ab\_stacked.9.png |
| Button | btn\_ | btn\_send\_pressed.9.png |
| Dialog | dialog\_ | dialog\_top.9.png |
| Divider | divider\_ | divider\_horizontal.png |
| Icon | ic\_ | ic\_star.png |
| Menu | menu\_ | menu\_submenu\_bg.9.png |
| Notification | notification\_ | notification\_bg.9.png |
| Tabs | tab\_ | tab\_pressed.9.png |

#### 4.4.2 icons文件的命名规范

|  |  |  |
| --- | --- | --- |
| Asset Type | Prefix 前缀 | Example |
| Icons | ic\_ | ic\_star.png |
| Launcher icons | ic\_launcher | ic\_launcher.png |
| Menu icons and Action Bar icons | ic\_menu | ic\_menu\_archive.png |
| Status bar icons | ic\_stat\_notify | ic\_stat\_notify\_msg.png |
| Tab icons | ic\_tab | ic\_tab\_recent.png |
| Dialog icons | ic\_dialog | ic\_dialog\_info.png |

#### 4.4.3 选择器状态文件的命名规范

|  |  |  |
| --- | --- | --- |
| State | Suffix尾缀 | Example |
| Normal | \_normal | btn\_order\_normal.png |
| Pressed | \_pressed | btn\_order\_pressed.png |
| Focused | \_focused | btn\_order\_focused.png |
| Disabled | \_disabled | btn\_order\_disabled.png |
| Selected | \_selected | btn\_order\_selected.png |

#### 4.4.4 布局文件

|  |  |  |
| --- | --- | --- |
| Component组件 | **Class Name** | **Example** |
| Activity | MainActivity | activity\_main.xml |
| Fragment | SiteFragment | fragment\_site.xml |
| Dialog | LoadingDialog | dialog\_loading.xml |
| Adapter Item |  | item\_site.xml |
| 其他视图 |  | view\_head.xml |

#### 4.4.5 string.xml、dimens.xml、colors.xml、styles.xml命名方式 ，遵循完整性 规范性 有序性原则，分块注释，不同模块区分开

|  |
| --- |
| * layout常量要在string.xml中进行定义，且每个常量根据功能模块自成一组，如：   <!—我的关注-->  <string name=”cancel\_follow”>取消关注</string>  …  <!—我的优惠劵-->  <string name=”no\_used\_toast”>您没有未使用的优惠劵哟~</string>  …   * layout当中控件的margin、padding以及TextView的文本大小值定义在dimens.xml文件当中，sp采取的命名方式为size\_具体数值，dp采取的命名方式为dp\_具体数值，如：   <dimen name=”size\_10”>10sp</dimen>  …  <dimen name=”dp\_10”>10dp</dimen>  …  layout当中的复用样式定义到styles.xml当中。 |

#### 4.4.6 动画文件

|  |  |  |
| --- | --- | --- |
| 动画效果 | 命名风格 | **Example** |
| 淡入/淡出 | fade\_in/fade\_out | fade\_in.xml |
| 从右淡入 | fade\_right\_in | fade\_right\_in.xml |
| 从右弹入 | push\_right\_in | push\_right\_in.xml |
| 从右滑入 | slide\_in\_from\_right | slide\_in\_from\_right.xml |

### 4.5类方法采用 小驼峰 命名法

根据函数所完成功能命名 ，如 changView()，在函数头写对于函数功能、参数和返回值的注释，一个函数请尽量保持在50行 之内 如：

/\*\*

\* 获取两个数中最大的一个

\* @param value1 参与比较的第一个数

\* @param value2 参与比较的第二个数

\* @return 两个参数中最大的一个数

\*/

public int max(int value1, int value2) {

return (value1 > value2) ? value1 : value2;

}

### 4.6控件id命名

命名模式为：view缩写\_模块名称\_view的逻辑名称

|  |  |
| --- | --- |
| 控件 | 前缀缩写 |
| RelativeLayout | rl |
| LinearLayout | ll |
| FrameLayout | fl |
| TextView | tv |
| Button | btn |
| ImageButton | ibtn |
| ImageView | iv |
| CheckBox | cb |
| RadioButton | rb |
| EditText | et |
| ToggleButton | tbtn |
| ProgressBar | pb |
| VideoView | vv |
| WebView | wv |
| ScrollView | sv |
| ListView | lv |
| GridView | gv |
| RecycleView | rv |

### 4.7变量名采用小驼峰命名

使用标准的Java命名方法，不推荐使用Google的m命名法。例如：

private String userName; 而不推荐使用 private String mUserName;

a.类中控件名称必须与xml布局id保持一致。

b.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。

c.局部变量名

|  |
| --- |
| * 局部变量名以LowerCamelCase风格编写，比起其它类型的名称；局部变量可以有更为宽松的缩写。 * 虽然缩写更宽松，但还是要避免用单字符进行命名，除了临时变量和循环变量。 * 即使局部变量final是不可改变的，也不应该把它表示为常量，自然不能用常量的规则去命名它。 * 临时变量通常被取名为i，j，k，m和n，它们一般用于整型；c，d，e，它们一般用于字符型。如：for(int i=0 ; i<len;i++),并且它和第一个单词间没有空格。 |
|  |

## 5.编码规范

### 5.1 Activity当中的oncreate方法分为三个部分

a.initVariable()🡪初始化变量以及获取Intent传值。

b.initView()🡪控件的初始化操作。

c.initData()🡪当前界面的数据获取操作。

### 5.2 布局的适配方案

xml当中的每个控件都得在activity当中初始化完毕后，重新调取测量方法例：

|  |
| --- |
| * measure(mViewFoot,0,142) * measure(mIvTabOne,78,78) * measure(mIvTabTwo,78,78)   …  measure内部实现原理🡪  ViewGroup.LayoutParams params = view.getLayoutParams();  params.width = resetWidth;  params.height = resetHeight;  view.setLayoutParams(params); |

### 5.3 activity之间的传值

activity之间的传值坚持用Intent携带序列化实体数据的方式。禁止为了省事使用全局变量进行传值的方式。

### 5.4 数据承载体的使用

为节省内存，使用ArrayList<>作为数据的承载体，而不是HashMap；HashMap<>由android自封装的SparseArray<>(或者ArrayMap<>)进行取代。

### 5.5 控件的点击事件由代码控制

简单逻辑可以在xml文件中通过onclick标签实现.

### 5.6 SharePreferences的建议

SharePreferences只用于简单的配置消息，对于对象，还是需要保存在本地文件中。

### 5.7 圆角图片的处理

项目当中的单一圆角背景图片，使用自定义shape进行处理。

### 5.8 本地数据库的操作

数据库采用项目中自定义的脚本进行表的创建删除以及数据的增删改查操作。

|  |
| --- |
| * 脚本升级文件的命名方式 (dbName\_oldVer\_newVer)，如：tunhuoji.db\_1\_2🡪 表示将tunhuoji数据库版本从1升到2。 * 数据库中表的命名规范，全部大写，单词用下划线\_链接。 * 增加表的升级脚本   @orm.create(MyPointModel)  @orm.create(ConvertGoodsModel);  @orm.create(ConvertRecordModel);   * 删除表的升级脚本   drop table if exists MAIN\_HOT\_LIST\_MODEL;  drop table if exists FOLLOW\_ARTICLE\_NEW\_MODEL;  drop table if exists FOLLOW\_ARTICLE\_HOT\_MODEL;   * 如果表中有增加、减少、修改列(对象对应的字段增加、减少、修改)   drop table if exists MSG\_SYSTEM\_LIST\_MODEL;  @orm.create(MsgSystemListModel);  //先删除原先的表再创建  drop table if exists ORDERS\_SHOP\_MODEL;  @orm.create(OrdersShopModel); |

## 6 上线打包注意的细节

a.更改<meta-data…></meta-data>当中的请求接口，第三方的数据地址由测试环境更改为正式环境。

b.关闭app本身应用的log日志开关以及第三方sdk应用的log日志开关。

检查应用的版本号，做版本号的递增处理。

c.build.gradle文件当中 对minifyEnable做混淆的操作，在proguard-rules.pro做对应保留类的操作。

d.更改ApplicationConsts中的RELEASE\_FLAG值为REL\_OFFICIAL。网络请求地址变为正式环境地址。