命名规范

一.包命名规范

包名全部小写，连续的单词只是简单地连接起来，不使用下划线。 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。 例如：com.jiashuangkuaizi.kitchen

包名	此包中包含
com.xx.应用名称缩写.activity	页面用到的 Activit
com.xx.应用名称缩写.base	基础共享的类
com.xx.应用名称缩写.adapter	页面用到的 Adapt
com.xx.应用名称缩写.util	此包中包含：公共
com.xx.应用名称缩写.bean	下面可分：vo、po
com.xx.应用名称缩写.model	此包中包含：模型
com.xx.应用名称缩写.db	数据库操作类
com.xx.应用名称缩写.view (或者 com.xx.应用名称缩写.widget)	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.receiver	BroadcastReceive

注意： 关于包结构着重声明一下，无论你以哪种维度区分，结果只要简洁清晰就好。

二.类命名规范（类名是一个或多个单词组成，采用大驼峰命名，尽量要使类名简洁且利于描述）

Activity 命名规范：以 Activity 作为后缀。

具体：对应的英文描述+Activity

例如：登陆的页面: LoginActivity

Adapter 命名规范：以 Adapter 作为后缀。

具体：对应页面+对应的列表英文描述+Adapter

例如：登陆页面已经登陆的账户列表的 adapter：LoginUserAdapter

Bean 类的命名规范：以 Bean 作为后缀。

具体：对应页面+Bean

例如：登陆页面对应的 Bean:LoginBean

Listener 接口监听事件的命名规范：以 On 开头 Listener 作为后缀。

具体：On+事件描述+Listener

例如：点击事件：OnClickListener

对应的调用方法：setOnClickListener

Fragment 的类文件的命名规范：以 Fragment 最为后缀。

具体：碎片页面的布局+fragment

例如：工具页面用 fragment:ToolFragment

解析类：Parser 为后缀标识

例如：首页解析类 HomePosterParser

三.常量，变量命名规范

常量命名

常量命名采用全大些+下划线方式例如：

public static final int LOGIN_FLAG=1;

Intent 标记

intent 标记的 key 以 INTENT 开头：

public static final String INTENT_CAR_ID = "intentCarId";

变量以驼峰式命名，不要使用组合声明，比如 int a, b;。

布局文件中控件命名：功能 + “” + *控件简写 资源配置信息（String.xml 等） 模块 + 功能 + （状态）+ 具体类型 eg：home*toolbar_text_active_color = "#fff000";

说明： 代码中不能出现硬编码和魔术比 eg: xx.equals("1") or if(position == 3) 自行按照常量命名规范定义成常量

四.方法命名规范

方法命名规则采用小驼峰命名法例如:

onCreate(),onRun()，方法名一般采用动词或者动名词。 一般使用的方法名前缀。

getXX()返回某个值的方法

initXX() 初始化相关方法，比如初始化布局:initView()

checkXX()和 isXX()方法为 boolean 值的时候使用 is 或者 check 为前缀

saveXX() 保存数据

clearXX()和 removeXX() 清除数据

updateXX() 更新数据

processXX() 对数据进行处理

dispalyXX() 显示某某信息

draswXX() 绘制数据或者效果

另外对于方法的其他一些规范：

- 方法的参数尽可能不超过 4 个，多余 4 个考虑采用 builder 模式或者 javabean 形式
- 方法参数中尽量少使用 boolean,使用 boolean 传参不利于代码的阅读
- 方法长度尽量不超过普通一屏
- 一个方法只做一件事
- 如果一个方法返回的是一个错误码，可以使用异常
- 方法尽量避免返回 null
- 尽量避免使用 try catch 处理业务逻辑
- 尽可能不实用 null，替代为异常或者使用空的变量，比如 Collections.emptyList()

五.类声明规范

区块划分 建议使用注释将源文件分为明显的区块，区块划分如下

```
常量声明区


UI 控件成员变量声明区


普通成员变量声明区


内部接口声明区
```

初始化相关方法区

事件响应方法区

普通逻辑方法区

重载的逻辑方法区

发起异步任务方法区

异步任务回调方法区

生命周期回调方法区（除去 `onCreate()`方法）

内部类声明区

类成员排列通用规则

按照发生的先后顺序排列

常量按照使用先后排列

`UI` 控件成员变量按照 `layout` 文件中的先后顺序排列

普通成员变量按照使用的先后顺序排列

方法基本上都按照调用的先后顺序在各自区块中排列

相关功能作为小区块放在一起（或者作为一个封装体引入）

重载：永不分离

当一个类有多个构造函数，或是多个同名方法，这些函数/方法应该按顺序出现在一起，中间不要放进其它函数/方法。

代码格式范例

匿名类范例

```
return new MyClass() {
    @Override
    public void method() {
        if (condition()) {
            try {
                something();
            } catch (ProblemException e) {
                recover();
            }
        }
    }
};
```

枚举类范例

```
private enum Suit {
    CLUBS,
    HEARTS,
    SPADES,
    DIAMONDS
}
```

数组范例

```
new int[] {
        0, 1, 2, 3
}
new int[] {
        0,
        1,
        2,
        3
}
new int[] {
        0, 1,
        2, 3
}
new int[]{0, 1, 2, 3}
```

注解

注解紧跟在文档块后面，应用于类、方法和构造函数，一个注解独占一行。

```
@Partial
@Mock
DataLoader loader;
```

六.资源文件命名规范

对应的 activity 的 xml 文件命名规范：以 activity 作为前缀

具体：activity+"_"+页面描述

例如：登陆页面 activity 的布局文件：activity_login.xml

ListView 中的 item 布局文件命名规范：以 item_list 作为前缀，以页面名称作为中缀，以列表描述作为后缀。

具体："item_list_"+页面名称+列表描述

例如：登陆页面的所有登录过的用户的列表：item_list_login_users.xml

Dialog 布局文件命名规范：以 dialog 作为前缀，如果是通用的 dialog 则以 common 作为中缀以功能描述作为后缀。如果是对应页面定制的 dialog,以页面描述作为中缀，dialog 描述作为后缀。

具体：

通用 dialog：dialog+common+功能描述

页面定制 dialog：dialog+页面名称+功能描述

例如：

通用的 dialog 提示：dialog_common_hint.xml

登录页面密码错误提示：dialog_login_pwd_error.xml

values 下文件命名

1. strings.xml 命名

直接以对应的内容的英文单词组合命名,后续通用的提示语之类会写在库里。

例如：

```
    <string name="look_car_address">看车地址</string>
        <string name="manage">管理</string>
```

2. colors.xml 命名

通用的库里的会有对应的颜色。不是通用的按照规则描述。

具体：页面+"_"+描述+颜色名称

例如：

登录页面登录按钮字体颜色

```
<color name="login_activity_login_btn_text">#000000</color><!--登录页面登录按钮字体颜色  -->
```

注意：整个 APP 通用的颜色将用

common_描述

例如：

通用的 item 字体黑色　common_item_text

3. dimens.xml 文件命名

主要采用官方的命名规范。对具体的 dimens 单位进行描述，目前采用的方法将要改变。

例如：

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources><!--  -->
```

drawable 目录下资源文件命名规范：

- selector 文件的命名规范：以 selector 作为前缀，以某个页面的作为中缀如果是多个页面就以模块名称作为中缀或者是项目通用的就以 common 作为中缀，以功能描述作为后缀。 具体：

selector+页面名称+功能描述

selector+模块名称+功能描述

selector+common+功能描述

例如：

登陆页面的确认按钮：selector_login_confirm_btn.xml（login 为页面名称）

所有支付模块购买的按钮：selector_pay_buy_btn.xml(buy 为模块名称)

所有通用按钮的背景：selector_common_btn_bg.xml

- shape 自定义图形文件命名规范：以 shape 作为前缀，如果是通用的以 common 作为中缀，如果是单独页面的以页面名称作为中缀，或者以模块名称作为中缀，以功能描述作为后缀。具体形式喝第一条 selector 一样的规则。
- 图片资源命名规范 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

icon 图片资源以 ic 开头；

具体： 前缀+"_"+*页面名称*+"_"+描述 （如果页面和描述一致的则不用重复）

欢迎页面背景图 bg_welcome.png

登录页面 登录按钮背景 bg_login_btn.png

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
```

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

动画命名例子	描述
fade_in	淡入
fade_out	淡出

动画命名例子	描述
push_down_in	从下方推入
push_down_out	从下方推出
push_left	推向左方
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

layout 中的 id 命名（layout 文件夹下）

命名模式为：view 缩写 *view 的逻辑名称*
使用 AndroidStudio 的插件 ButterKnife Zelezny，生成注解非常方便。
*如果不使用 ButterKnife Zelezny，则建议使用 view 缩写做后缀，如：username*tv（展示用户名的 TextView）

编程实践

@Override：能用则用

只要是合法的，就把@Override 注解给用上。

捕获的异常：不能忽视

除了下面的例子，对捕获的异常不做响应是极少正确的。(典型的响应方式是打印日志，或者如果它被认为是不可能的，则把它当作一个 AssertionError 重新抛出。) 如果它确实是不需要在 catch 块中做任何响应，需要做 ———– 加以说明(如下面的例子)。

```
try {
    int i = Integer.parseInt(response);
    return handleNumericResponse();
} catch (NumberFormatException ok) {
    // it's not numeric; that's fine, just continue
}
```

```
return handleTextResponse(response);
```

补充规范 1，类或接口定义时扩展或者继承的接口需要换行，逗号在前置位 eg：public final class MainActivity extends CwtBaseActivity implements MainView, BottomNavigationBar.OnTabSelectedListener {

2，定义类或者接口 首行留空格，末尾不留；其次每个逻辑分区都要留单行空格（除普通常量定义外）

3，请自行控制类、接口、及其成员的访问权限，尽量缩小访问权限

4，类如果不是为扩展而设计的请加 final 修饰符

5，遇到 switch 语句时，酌情考虑将每个 case 语句拆分成当个函数调用

6，定义类或者接口变量时，请使用超类或者顶层接口定义变量

7，请使用 JDK7 及以下版本的语法糖

8，遇到处理文件时请使尽量用 nio2 提供的 api

9，处理资源信息时请尽量使用 TWR 语法

10，接口定义不加“I”前缀，实现添加"impl"后缀，如果有必要请提供默认的实现和其它实现 （默认实现:Default+接口名+impl，其它实现 :具体扩展功能 + 接口 + impl）

11，具有 controller 功能的类或者接口定义 都以功能+“manager”

12，所有表示层设计的实体定义功能+“model”

13，所有存在序列化对象类定义请自行实现 Parcelable 接口或者 Serializable（通过 Intent 传递实现 Parcelable 接口，序列化本地实现 Serializable）

14，重写父类方法时一定要查看父类方法定义，然后再决定是扩展父类方法还是完全重写实现该父类方法

15，命名不该出现汉语拼音！可以通过 google 翻译进行查询或者咨询其他组员

16，请慎重使用匿名内部类

17，不使用 new Thread（）子线程使用方式，如果有需要请使用线程池

18，访问后台 api 接口时传递的参数一定要统一定义并且有一定的模块职责区分开

19，在能使用 application context 时请不要使用 activity context 或着 service context

20，尽量避免使用标签类，请使用多态实现

21，vcs 版本库中不存在任何注释掉的代码块

22，提交代码时注释格式为[版本号][修改类型][修改的模块] eg：[1.2][fix bug][首页列表滑动冲突 bug]

23，不存在公有的对象属性，请提供公有的 set/get 方法

24，请尽量避免使用过时的 api 或者高于项目要求的最高版本的 api

25，当一个类或者接口只在当前类使用，请尽量定义成内部类形式

26，使用 handler 消息分发时，hander 定义成静态的或者以弱引用形式持有 handler 对象

27，再次声明一下，命名一定不能有歧义，一定要说明用途也就是用途无歧义

28，尽量不重复写代码。一处代码重复出现三次就应该考虑重新设计该实现逻辑

29，不提倡过度的注释信息，修改被注释的逻辑定义块，请思考是否需要更新注释信息

30，注释首先应该被添加在接口定义处，如果存在多个实现请各自添加相应的注释信息

last but not least 设计程序代码时请务必思考六大设计原则，尤其是开闭原则。