|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 修订记录 | | | | | | | | | | | | |
| 版 本 | 修 订 描 述 | | | | | | | | | | 修订日期 | |
| A0 | 初次建立 | | | | | | | | | | 2020/1/18 | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
|  |  | | | | | | | | | |  | |
| 评审 | | | | | | | | | | | | |
| 部门 | | 评审 | | | 部门 | | 评审人 | | 部门 | | | 评审人 |
| 平台研发中心 | | 程绪伟 | | | TrackSolid 产品PDT经理 | | 陈新军 | | 有线定位产品PDT经理 | | | 张宁 |
| 项目管理中心 | | 段小权 | | | ISD产品PDT经理 | | 王胜 | | 无线定位产品PDT经理 | | | 王阳 |
| 规划中心 | | 郑旻琪 | | | Tracker产品PDT经理 | | 李胜如 | | 个人穿戴产品PDT经理 | | | 龚超 |
| 云平台智慧车联产品 | | 武宇飞 | | | DVR 产品PDT经理 | | 赵志文 | | 几米圈产品PDT经理 | | | 林时配 |
| 分发部门 | | | | | | | | | | | | |
| PDT管理 | | | 规划中心 | | | 项目管理中心 | | | | 国内营销--解决方案部 | | |
| 研发中心-测试中心 | | | 研发中心--云平台 | | | 国内营销-研发管理中心 | | | |  | | |
| 编制: 谢洪涛 | | | | 审核:－－ | | | | 批准:程绪伟 | | | | |

**目 录**

**[1. 引言](#_Toc28634_WPSOffice_Level1)** [3](#_Toc28634_WPSOffice_Level1)

[1.1. 更新记录 3](#_Toc24920_WPSOffice_Level2)

**[2. 编程规约](#_Toc24920_WPSOffice_Level1)** [3](#_Toc24920_WPSOffice_Level1)

[2.1. AS 规范 3](#_Toc24529_WPSOffice_Level2)

[2.2. 命名规范 3](#_Toc26_WPSOffice_Level2)

[First 一组变量中的第一个](#_Toc24920_WPSOffice_Level3) [6](#_Toc24920_WPSOffice_Level3)

[2.3. 代码样式规范](#_Toc17934_WPSOffice_Level2) [7](#_Toc17934_WPSOffice_Level2)

[2.3.1. 【推荐】 使用标准大括号样式](#_Toc24529_WPSOffice_Level3) [7](#_Toc24529_WPSOffice_Level3)

[2.3.2. 【推荐】编写简短方法](#_Toc26_WPSOffice_Level3) [8](#_Toc26_WPSOffice_Level3)

[2.3.3. 【推荐】 行长限制](#_Toc17934_WPSOffice_Level3) [9](#_Toc17934_WPSOffice_Level3)

[2.4. 资源文件规范](#_Toc20982_WPSOffice_Level2) [9](#_Toc20982_WPSOffice_Level2)

[2.4.1. 【推荐】动画资源文件（anim/ 和 animator/）](#_Toc20982_WPSOffice_Level3) [9](#_Toc20982_WPSOffice_Level3)

[2.4.2. 【推荐】颜色资源文件（color/） 9](#_Toc24164_WPSOffice_Level3)

[2.4.3. 【推荐】图片资源文件（drawable/ 和 mipmap/） 9](#_Toc3161_WPSOffice_Level3)

[2.4.4. 【推荐】 values 资源文件（values/） 9](#_Toc7593_WPSOffice_Level3)

[2.4.5. 【强制】colors.xml 9](#_Toc6495_WPSOffice_Level3)

[2.5. 注释规范](#_Toc24164_WPSOffice_Level2) [10](#_Toc24164_WPSOffice_Level2)

[2.5.1. 【强制】类注释](#_Toc21440_WPSOffice_Level3) [11](#_Toc21440_WPSOffice_Level3)

[2.5.2. 【强制】方法注释](#_Toc9923_WPSOffice_Level3) [12](#_Toc9923_WPSOffice_Level3)

[2.5.3. 【推荐】块注释](#_Toc18229_WPSOffice_Level3) [13](#_Toc18229_WPSOffice_Level3)

[2.5.4. 【推荐】其他一些注释](#_Toc31135_WPSOffice_Level3) [14](#_Toc31135_WPSOffice_Level3)

**1.0引言**

**1.1更新记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **更新人** | **更新日期** | **说明** |
| **1.0** | 陈梦宣 | 2018/03/28 | 新建 |
| **1.1** | 李治健 | 2020/01/13 | 添加仓库规范，完善编程思维 |

# 2.0编程规约

**2.1 AS规范**

* 【推荐】尽量使用最新的稳定版的 IDE 进行开发；
* 【强制】编码格式统一为 UTF-8；
* 【强制】编辑完 .java、.xml 等文件后一定要 格式化，格式化，格式化（如果团队有

公共的样式包，那就遵循它，否则统一使用 AS 默认模板即可）；

* 【推荐】删除多余的 import，减少警告出现，可利用 AS 的 Optimize Imports（Settings

-> Keymap -> Optimize Imports）快捷键。

**2.2命名规范**

### 2.2.1【强制】包名

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。一级包名是顶级域名，通常为 com、edu、gov、net、org 等，二级

包名为公司名，三级包名根据应用进行命名，后面就是对包名的划分了，关于包名的划

分，推荐采用 PBF（按功能分包 Package By Feature）。

### 2.2.2【强制】类名

类名都以 UpperCamelCase 风格编写。

类名通常是名词或名词短语，接口名称有时可能是形容词或形容词短语。现在还没有特

定的规则或行之有效的约定来命名注解类型。

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 WelcomeActivity；

Adapter 类 Adapter 为后缀标识 新闻详情适配器 NewsDetailAdapter；

解析类 Parser 为后缀标识 首页解析类 HomePosterParser ；

工具方法类 Utils 或 Manager 为后缀标识 线程池管理类：ThreadPoolManager；

日志工具类：LogUtils（Logger 也可）；

打印工具类：PrinterUtils ；

数据库类 以 DBHelper 后缀标识 新闻数据库：NewsDBHelper ；

Service 类 以 Service 为后缀标识 时间服务 TimeService；

BroadcastReceiver 类 以 Receiver 为后缀标识 推送接收 JPushReceiver ；

ContentProvider 类 以 Provider 为后缀标识 ShareProvider；

自定义的共享基础类 以 Base 开头 BaseActivity, BaseFragment ；

测试类的命名以它要测试的类的名称开始，以 Test 结束。例如：HashTest 或

HashIntegrationTest；

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 Runnable、interface Accessible。

### 2.2.3【强制】方法名都以 lowerCamelCase 风格编写。

### 方法名通常是动词或动词短语。

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/check 为前缀标识；

getXX() 返回某个值的方法，使用 get 为前缀标识 ；

setXX() 设置某个属性值 ；

handleXX(), processXX() 对数据进行处理的方法 ；

displayXX(), showXX() 弹出提示框和提示信息，使用 display/show 为前缀标识 ；

updateXX() 更新数据 ；

saveXX(), insertXX() 保存或插入数据 ；

resetXX() 重置数据 ；

clearXX() 清除数据 ；

removeXX(), deleteXX() 移除数据或者视图等，如 removeView() ；

drawXX() 绘制数据或效果相关的，使用 draw 前缀标识 。

### 2.2.4【强制】常量名命名

常量名命名模式为 CONSTANT\_CASE，全部字母大写，用下划线分隔单词。那到底什么算是

一个常量？

每个常量都是一个 static final 字段，但不是所有 static final 字段都是常量。在决定一个字段是否是一个常量时，得考虑它是否真的感觉像是一个常量。例如，如果观测任何一个该实例的状态是可变的，则它几乎肯定不会是一个常量。只是永远不打算改变的对象一般是不够的，它要真的一直不变才能将它示为常量。

### 2.2.5【强制】非常量字段命名

非常量字段以 lowerCamelCase 风格的基础上改造为如下风格：基本结构为 scope{Type0}VariableName{Type1}、type0VariableName{Type1}、variableName{Type1}。

### 2.2.6【推荐】 scope（范围）

非公有，非静态字段命名以 m 开头。

静态字段命名以 s 开头。

其他字段以小写字母开头。

例如：

public class MyClass {

public int publicField;

private static MyClass sSingleton;

int mPackagePrivate;

private int mPrivate;

protected int mProtected;

}

使用 1 个字符前缀来表示作用范围，1 个字符的前缀必须小写，前缀后面是由表意性强的一个单词或多个单词组成的名字，而且每个单词的首写字母大写，其它字母小写，这样保证了对变量名能够进行正确的断句。

### 2.2.7【推荐】 Type0（控件类型）

考虑到 Android 众多的 UI 控件，为避免控件和普通成员变量混淆以及更好地表达意思，所有用来表示控件的成员变量统一加上控件缩写作为前缀。例如：mIvAvatar、rvBooks、

flContainer。

### 2.2.8【推荐】VariableName（变量名）

变量名中可能会出现量词，我们需要创建统一的量词，它们更容易理解，也更容易搜索。

例如：mFirstBook、mPreBook、curBook。

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

Pre 一组变量中的上一个

Cur 一组变量中的当前变量

### 2.2.9【强制】Type1（数据类型）

对于表示集合或者数组的非常量字段名，我们可以添加后缀来增强字段的可读性，比如：

集合添加如下后缀：List、Map、Set。

数组添加如下后缀：Arr。

例如：mIvAvatarList、userArr、firstNameSet。

注意：如果数据类型不确定的话，比如表示的是很多书，那么使用其复数形式来表示也可，例如 mBooks。

### 2.2.10【强制】参数名

参数名以 lowerCamelCase 风格编写，参数应该避免用单个字符命名。

### 2.2.11【强制】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，局部变量名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。

即使局部变量是 final 和不可改变的，也不应该把它示为常量，自然也不能用常量的

规则去命名它。

### 2.2.12【强制】临时变量

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++)。

### 2.3代码样式规范

### 2.3.1【推荐】 使用标准大括号样式

左大括号不单独占一行，与其前面的代码位于同一行：

class MyClass {

int func() {

if (something) {

// ...

} else if (somethingElse) {

// ...

} else {

// ...

}

}

}

我们需要在条件语句周围添加大括号。例外情况：如果整个条件语句（条件和主体）适合放在同一行，那么您可以（但不是必须）将其全部放在一行上。例如，我们接受以下样式：

if (condition) {

body();

}

同样也接受以下样式：

if (condition) body();

但不接受以下样式：

if (condition)

body(); // bad!

### 2.3.2【推荐】编写简短方法

在可行的情况下，尽量编写短小精炼的方法。我们了解，有些情况下较长的方法是恰当的，因此对方法的代码长度没有做出硬性限制。如果某个方法的代码超出 40 行，请考虑是否可以在不破坏程序结构的前提下对其拆解。

### 2.3.3【推荐】 行长限制

代码中每一行文本的长度都应该不超过 100 个字符。虽然关于此规则存在很多争论，但最终决定仍是以 100 个字符为上限，如果行长超过了 100（AS 窗口右侧的竖线就是设置的行宽末尾 ），我们通常有两种方法来缩减行长。

• 提取一个局部变量或方法（最好）。

• 使用换行符将一行换成多行。

不过存在以下例外情况：

• 如果备注行包含长度超过 100 个字符的示例命令或文字网址，那么为了便于剪切和

粘贴，该行可以超过 100 个字符。

• 导入语句行可以超出此限制，因为用户很少会看到它们（这也简化了工具编写流程）。

### 2.4资源文件规范

### 2.4.1【推荐】动画资源文件（anim/ 和 animator/）

安卓主要包含属性动画和视图动画，其视图动画包括补间动画和逐帧动画。属性动画文

件需要放在 res/animator/ 目录下，视图动画文件需放在 res/anim/ 目录下。

命名规则：{模块名\_}逻辑名称。

说明：{} 中的内容为可选，逻辑名称 可由多个单词加下划线组成。

例如：refresh\_progress.xml、market\_cart\_add.xml、market\_cart\_remove.xml。

如果是普通的补间动画或者属性动画，可采用：动画类型\_方向 的命名方式。

例如：

名称 说明

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

### 2.4.2【推荐】颜色资源文件（color/）

专门存放颜色相关的资源文件。

命名规则：类型\_逻辑名称。

例如：sel\_btn\_font.xml。

颜色资源也可以放于 res/drawable/ 目录，引用时则用 @drawable 来引用，但不推荐

这么做，最好还是把两者分开。

### 2.4.3【推荐】图片资源文件（drawable/ 和 mipmap/）

res/drawable/ 目录下放的是位图文件（.png、.9.png、.jpg、.gif）或编译为可绘制

对象资源子类型的 XML 文件，而 res/mipmap/ 目录下放的是不同密度的启动图标，所

以 res/mipmap/ 只用于存放启动图标，其余图片资源文件都应该放到 res/drawable/

目录下。

### 2.4.4【推荐】 values 资源文件（values/）

values/ 资源文件下的文件都以 s 结尾，如 attrs.xml、colors.xml、dimens.xml，起

作用的不是文件名称，而是 <resources> 标签下的各种标签，比如 <style> 决定样式，

<color> 决定颜色，所以，可以把一个大的 xml 文件分割成多个小的文件，比如可以有

多个 style 文件，如 styles.xml、styles\_home.xml、styles\_item\_details.xml、

styles\_forms.xml。

### 2.4.5【强制】colors.xml

<color> 的 name 命名使用下划线命名法，在你的 colors.xml 文件中应该只是映射颜

色的名称一个 ARGB 值，而没有其它的。不要使用它为不同的按钮来定义 ARGB 值。

例如，不要像下面这样做：

<resources>

<color name="button\_foreground">#FFFFFF</color>

<color name="button\_background">#2A91BD</color>

<color name="comment\_background\_inactive">#5F5F5F</color>

<color name="comment\_background\_active">#939393</color>

<color name="comment\_foreground">#FFFFFF</color>

<color name="comment\_foreground\_important">#FF9D2F</color>

...

<color name="comment\_shadow">#323232</color>

使用这种格式，会非常容易重复定义 ARGB 值，而且如果应用要改变基色的话会非常困难。同时，这些定义是跟一些环境关联起来的，如 button 或者 comment，应该放到一个按钮风格中，而不是在 colors.xml 文件中。

相反，应该这样做：

<resources>

<!-- grayscale -->

<color name="white" >#FFFFFF</color>

<color name="gray\_light">#DBDBDB</color>

<color name="gray" >#939393</color>

<color name="gray\_dark" >#5F5F5F</color>

<color name="black" >#323232</color>

<!-- basic colors -->

<color name="green">#27D34D</color>

<color name="blue">#2A91BD</color>

<color name="orange">#FF9D2F</color>

<color name="red">#FF432F</color>

</resources>

向应用设计者那里要这个调色板，名称不需要跟 "green"、"blue" 等等相同。

"brand\_primary"、"brand\_secondary"、"brand\_negative" 这样的名字也是完全可以接受的。像这样规范的颜色很容易修改或重构，会使应用一共使用了多少种不同的颜色变得非常清晰。通常一个具有审美价值的 UI 来说，减少使用颜色的种类是非常重要的。

### 2.5注释规范

### 为了减少他人阅读你代码的痛苦值，请在关键地方做好注释。

### 2.5.1【强制】类注释

### 每个类完成后应该有作者姓名和联系方式的注释，对自己的代码负责。

/\*\*

\* <pre>

\* author : Blankj

\* e-mail : xxx@xx

\* time : 2017/03/07

\* desc : xxxx 描述

\* version: 1.0

\* </pre>

\*/public class WelcomeActivity {

...

}

具体可以在 AS 中自己配制，进入 Settings -> Editor -> File and Code Templates -> Includes -> File Header，输入

/\*\*

\* <pre>

\* author : ${USER}

\* e-mail : xxx@xx

\* time : ${YEAR}/${MONTH}/${DAY}

\* desc :

\* version: 1.0

\* </pre>

\*/

这样便可在每次新建类的时候自动加上该头注释。

### 2.5.2【强制】方法注释

### 每一个成员方法（包括自定义成员方法、覆盖方法、属性方法）的方法头都必须做方法

### 头注释，在方法前一行输入 /\*\* + 回车 或者设置 Fix doc comment（Settings -> Keymap

### -> Fix doc comment）快捷键，AS 便会帮你生成模板，我们只需要补全参数即可，如下

### 所示。

/\*\*

\* bitmap 转 byteArr

\*

\* @param bitmap bitmap 对象

\* @param format 格式

\* @return 字节数组

\*/

public static byte[] bitmap2Bytes(Bitmap bitmap, CompressFormat format) {

if (bitmap == null) return null;

ByteArrayOutputStream baos = new ByteArrayOutputStream();

bitmap.compress(format, 100, baos);

return baos.toByteArray();

}

### 2.5.3【推荐】块注释

### 块注释与其周围的代码在同一缩进级别。它们可以是 /\* ... \*/ 风格，也可以是 // ...

### 风格（//后最好带一个空格）。对于多行的 /\* ... \*/ 注释，后续行必须从 \* 开始，

### 并且与前一行的 \* 对齐。以下示例注释都是 OK 的。

/\*

\* This is

\* okay.

\*/

// And so// is this.

/\* Or you can

\* even do this. \*/

### 注释不要封闭在由星号或其它字符绘制的框架里。

### Tip：在写多行注释时，如果你希望在必要时能重新换行（即注释像段落风格一样），那么使用 /\* ... \*/。

### 2.5.4【推荐】其他一些注释

### AS 已帮你集成了一些注释模板，我们只需要直接使用即可，在代码中输入 todo、fixme 等这些注释模板，回车后便会出现如下注释。

### // TODO: 17/3/14 需要实现，但目前还未实现的功能的说明// FIXME: 17/3/14 需要修正，甚至代码是错误的，不能工作，需要修复的说明

# 3.0编程思维

1、应遵循模块化、功能化原则，降低耦合，提高模块及框架的可扩展性；  
2、应遵循面向对象、面向函数式编程，少用全局类型变量，少用临时参数；  
3、任何选择型代码块需要全部考虑选择类型，任何返回型代码块需要考虑所有返回的可能性；  
4、功能逻辑或需求应考虑所有异常或非正常调用及请求，而非简单的完成需求。  
5、一般情况下，任何请求和调用应该有相应的回调或反馈；  
6、你敲的每行代码，应思考是否是最优代码，是否是冗余代码，是否可以改善？

7、多与其他人沟通，多探讨代码逻辑、代码的严谨性；

8、新增功能及bug修复，应考虑修改的代码是否会影响其他功能；

# 4.0代码仓库规范

**master**：主分支，用于备份及查看；  
**dev**：开发分支，用于日常项目开发及调试；  
**release**：项目结版及后期维护拉取代码发布等；  
**test**：日常测试及无关项目功能的临时使用分支；

1、项目代码在发版、重大更改、重要功能添加，都需要进行代码提交；  
2、在不影响项目其他功能及模块的原则上，必须每日提交代码推送远程仓库分支；  
3、提交代码，必须写清楚修改的内容，包括：新增或修改了什么功能、调整了什么地方、修 复了什么BUG等；  
4、项目发版必须打版本号Tag；  
5、在项目发版或重大更新及调整之后，需要发起master分支合并请求，用SE主导；

6、重要更新及发版之后，需要发起主分支Master合并请求；