**1 为什么要有编码规范**

编码规范对于程序员而言尤为重要，有以下几个原因：

- 一个软件的生命周期中，80%的花费在于维护

- 几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护

- 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新的代码

- 如果你将源码作为产品发布，就需要确任它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品

为了执行规范，每个软件开发人员必须一致遵守编码规范。每个人。

1. 【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例： \_name / \_\_name / $Object / name\_ / name$ / Object$

2. 【强制】代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式

也要避免采用。

正例： czsm / taobao / youku / hangzhou 等国际通用的名称，可视同英文。

反例： DaZhePromotion [ 打折 ] / getPingfenByName() [ 评分 ] / int 某变量 = 3

3. 【强制】类名使用 UpperCamelCase 风格，必须遵从驼峰形式，但以下情形例外： DO / BO /

DTO / VO / AO

正例： MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion

4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从

驼峰形式。

正例： localValue / getHttpMessage() / inputUserId

5. 【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

正例： MAX \_ STOCK \_ COUNT

反例： MAX \_ COUNT

6. 【强制】抽象类命名使用 Abstract 或 Base 开头 ； 异常类命名使用 Exception 结尾 ； 测试类

命名以它要测试的类的名称开始，以 Test 结尾。

7. 【强制】中括号是数组类型的一部分，数组定义如下： String[] args;

反例：使用 String args[] 的方式来定义。

9. 【强制】包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用

单数形式，但是类名如果有复数含义，类名可以使用复数形式。

正例： 应用工具类包名为 com . czsm . open . util 、类名为 MessageUtils

10. 【强制】杜绝完全不规范的缩写，避免望文不知义。

反例： AbstractClass “缩写”命名成 AbsClass；condition “缩写”命名成 condi ，此类随意缩写严重降低了代码的可阅读性。

11. 【推荐】为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词组合来表达其意。

正例：从远程仓库拉取代码的类命名为 PullCodeFromRemoteRepository。

反例：变量 int a; 的随意命名方式。

12. 【推荐】如果模块、接口、类、方法使用了设计模式，在命名时体现出具体模式。

说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。

正例： public class OrderFactory;

public class LoginProxy;

public class ResourceObserver;

14. 接口和实现类的命名规则：

【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部的实现类用 Impl 的后缀与接口区别。

正例： CacheServiceImpl 实现 CacheService 接口。

15. 【参考】枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。

说明：枚举其实就是特殊的常量类，且构造方法被默认强制是私有。

正例：枚举名字为 ProcessStatusEnum 的 成员名称： SUCCESS / UNKOWN \_ REASON 。

16. 【参考】各层命名规约：

A) Service / DAO 层方法命名规约

1 ） 获取单个对象的方法用 get 做前缀。

2 ） 获取多个对象的方法用 list 做前缀。

3 ） 获取统计值的方法用 count 做前缀。

4 ） 插入的方法用 save/insert 做前缀。

5 ） 删除的方法用 remove/delete 做前缀。

6 ） 修改的方法用 update 做前缀。

B) 领域模型命名规约

1 ） 数据对象： xxxDO ， xxx 即为数据表名。

2 ） 数据传输对象： xxxDTO ， xxx 为业务领域相关的名称。

3 ） 展示对象： xxxVO ， xxx 一般为网页名称。

4 ） POJO 是 DO / DTO / BO / VO 的统称，禁止命名成 xxxPOJO 。

( ( 二) ) 常量定义

1. 【强制】不允许任何魔法值 （ 即未经定义的常量 ） 直接出现在代码中。

反例： String key = " Id # taobao \_" + tradeId;

cache . put(key , value);

2. 【强制】 long 或者 Long 初始赋值时，使用大写的 L ，不能是小写的 l ，小写容易跟数字 1 混淆，造成误解。

说明： Long a = 2 l; 写的是数字的 21，还是 Long 型的 2?

3. 【推荐】不要使用一个常量类维护所有常量，按常量功能进行归类，分开维护。

说明：大而全的常量类，非得使用查找功能才能定位到修改的常量，不利于理解和维护。

正例：缓存相关常量放在类 CacheConsts 下 ； 系统配置相关常量放在类 ConfigConsts 下。

4. 【推荐】常量的复用层次有五层：跨应用共享常量、应用内共享常量、子工程内共享常量、包

内共享常量、类内共享常量。

1 ） 跨应用共享常量：放置在二方库中，通常是 client . jar 中的 constant 目录下。

2 ） 应用内共享常量：放置在一方库中，通常是 modules 中的 constant 目录下。

反例：易懂变量也要统一定义成应用内共享常量，两位攻城师在两个类中分别定义了表示

“是”的变量：

类 A 中： public static final String YES = " yes " ;

类 B 中： public static final String YES = " y " ;

A . YES . equals(B . YES) ，预期是 true ，但实际返回为 false ，导致线上问题。

3 ） 子工程内部共享常量：即在当前子工程的 constant 目录下。

4 ） 包内共享常量：即在当前包下单独的 constant 目录下。

5 ） 类内共享常量：直接在类内部 private static final 定义。

( 三) 代码 代码格式 格式

1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行 ； 如果

是非空代码块则：

1 ） 左大括号前不换行。

2 ） 左大括号后换行。

3 ） 右大括号前换行。

4 ） 右大括号后还有 else 等代码则不换行 ； 表示终止的右大括号后必须换行。

2. 【强制】 左小括号和字符之间不出现空格 ； 同样，右小括号和字符之间也不出现空格。详见

第 5 条下方正例提示。

反例： if (空格 a == b 空格)

3. 【强制】 if / for / while / switch / do 等保留字与括号之间都必须加空格。

4. 【强制】任何二目、三目运算符的左右两边都需要加一个空格。

说明：运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。

6. 【强制】注释的双斜线与注释内容之间有且仅有一个空格。

正例： // 注释内容，注意在 // 和注释内容之间有一个空格。

7. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：

1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。

2 ） 运算符与下文一起换行。

3 ） 方法调用的点符号与下文一起换行。

4 ） 方法调用时，多个参数，需要换行时，在逗号后进行。

8. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。

正例：下例中实参的" a ",后边必须要有一个空格。

method("a", "b", "c");

9. 【强制】 IDE 的 text file encoding 设置为 UTF -8 ; IDE 中文件的换行符使用 Unix 格式，

不要使用 Windows 格式。

11. 【推荐】方法体内的执行语句组、变量的定义语句组、不同的业务逻辑之间或者不同的语义之间插入一个空行。相同业务逻辑和语义之间不需要插入空行。

说明：没有必要插入多个空行进行隔开。