Java后端必备的开发规范

Java知音 今天

> 作者：silianpan
>
> juejin.im/post/5ada99fff265da0b8a672fbd

> 基于阿里巴巴JAVA开发规范整理
> https://github.com/alibaba/p3c

一、命名风格

【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
- 反例：macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion

【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

【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

【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。

【强制】Model 类中布尔类型的变量，都不要加 is，否则部分框架解析会引起序列化错误。

- 反例：定义为基本数据类型 Boolean isDeleted；的属性，它的方法也是 isDeleted()，RPC框架在反向解析的时候，“以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异常。

【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：CacheManagerImpl 实现 CacheManager 接口。

【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。

【推荐】接口类中的方法和属性不要加任何修饰符号（public 也不要加），保持代码的简洁 性，并加上有效 的Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是 与接口方法相关，并且是整个应用的基础常量。

- 正例：接口方法签名：void f(); 接口基础常量表示：String COMPANY = "alibaba";
- 反例：接口方法定义：public abstract void f();
- 说明：JDK8 中接口允许有默认实现，那么这个default方法，是对所有实现类都有价值的默 认实现。

【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。

【参考】各层命名规约：

Service/DAO 层方法命名规约

- 获取单个对象的方法用 get 做前缀。
- 获取多个对象的方法用 list 做前缀。
- 获取统计值的方法用 count 做前缀。
- 插入的方法用 save/insert 做前缀。
- 删除的方法用 remove/delete 做前缀。
- 修改的方法用 update 做前缀。

二、变量定义

【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。

三、代码格式

【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：

- 左大括号前不换行。
- 左大括号后换行。
- 右大括号前换行。
- 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。

【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。

- 反例：if (空格a == b空格)

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

【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。

【强制】采用 4 个空格缩进，禁止使用 tab 字符。

- 说明：Vue工程采用2个空格缩进

【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。

【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");

【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。

【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。

四、OOP规约

【强制】所有的覆写方法，必须加@Override 注解。

【强制】不能使用过时的类或方法。

【强制】Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals。

- 正例："test".equals(object);
- 反例：object.equals("test");

【强制】所有的相同类型的包装类对象之间值的比较，全部使用 equals 方法比较。

- 说明：对于 Integer var = ? 在 -128 至 127 范围内的赋值，Integer 对象是在 IntegerCache.cache 产生，会复用已有对象，这个区间内的 Integer 值可以直接使用==进行判断，但是这个区间之外的所有数据，都会在堆上产生，并不会复用已有对象，这是一个大坑， 推荐使用 equals 方法进行判断。

【强制】RPC 方法的返回值和参数必须使用包装数据类型。

【强制】构造方法里面禁止加入任何业务逻辑，如果有初始化逻辑，请放在 init 方法中。

【推荐】当一个类有多个构造方法，或者多个同名方法，这些方法应该按顺序放置在一起， 便于阅读。

【推荐】循环体内，字符串的连接方式，使用 StringBuilder 的 append 方法进行扩展。

- 说明：反编译出的字节码文件显示每次循环都会 new 出一个 StringBuilder 对象，然后进行 append 操作，最后通过 toString 方法返回 String 对象，造成内存资源浪费。
- 反例：

```
String str = "start";
for (int i = 0; i < 100; i++) {
str = str + "hello";
}
```

【推荐】慎用 Object 的 clone 方法来拷贝对象。说明：对象的 clone 方法默认是浅拷贝，若想实现深拷贝需要重写 clone 方法实现属性对象 的拷贝。

五、集合处理

【强制】使用集合转数组的方法，必须使用集合的 toArray(T[] array)，传入的是类型完全一样的数组，大小就是 list.size()。

- 说明：使用 toArray 带参方法，入参分配的数组空间不够大时，toArray 方法内部将重新分配 内存空间，并返回新数组地址；如果数组元素大于实际所需，下标为[list.size()]的数组 元素将被置为 null，其它数组元素保持原值，因此最好将方法入参数组大小定义与集合元素 个数一致。

- 正例：

```
List<String> list = new ArrayList<String>(2);
list.add("guan");
list.add("bao");
String[] array = new String[list.size()];
array = list.toArray(array);
```

- 反例：直接使用 toArray 无参方法存在问题，此方法返回值只能是 Object[]类，若强转其它 类型数组将出现 ClassCastException 错误。

【强制】不要在 foreach 循环里进行元素的 remove/add 操作。remove 元素请使用 Iterator方式，如果并发操作，需要对 Iterator 对象加锁。

- 正例：

```
Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
  String item = iterator.next();
  if (删除元素的条件) {
   iterator.remove();
  }
}
```

- 反例：

```
List<String> list = new ArrayList<String>();
list.add("1");
list.add("2");
for (String item : list) {
 if ("1".equals(item)) {
  list.remove(item);
 }
}
```

【推荐】使用 entrySet 遍历 Map 类集合 KV，而不是 keySet 方式进行遍历。

六、控制语句

【强制】在一个 switch 块内，每个 case 要么通过 break/return 等来终止，要么注释说明程序将继续执行到哪一个 case 为止；在一个 switch 块内，都必须包含一个 default 语句并且 放在最后，即使它什么代码也没有。

【强制】在 if/else/for/while/do 语句中必须使用大括号。即使只有一行代码，避 单行的编码方式：if (condition) statements;

【推荐】除常用方法（如 getXxx/isXxx）等外，不要在条件判断中执行其它复杂的语句，将复 杂逻辑判断的结果赋值给一个有意义的布尔变量名，以提高可读性。

- 说明：很多 if 语句内的逻辑相当复杂，阅读者需要分析条件表达式的最终结果，才能明确什么 样的条件执行什么样的语句，那么，如果阅读者分析逻辑表达式错误呢?
- 正例：

```
//伪代码如下
final boolean existed = (file.open(fileName, "w") != null) && (...) || (...);
if (existed) {
  ...
}
```

- 反例：

```
if ((file.open(fileName, "w") != null) && (...) || (...)) {
  ...
}
```

【推荐】循环体中的语句要考量性能，以下操作尽量移至循环体外处理，如定义对象、变量、 获取数据库连接，进行不必要的 try-catch 操作（这个 try-catch 是否可以移至循环体外）。

【参考】下列情形，需要进行参数校验：

- 调用频次低的方法。
- 执行时间开销很大的方法。此情形中，参数校验时间几乎可以忽略不计，但如果因为参 数错误导致中间执行回退，或者错误，那得不偿失。
- 需要极高稳定性和可用性的方法。
- 对外提供的开放接口，不管是 RPC/API/HTTP 接口。
- 敏感权限入口。

七、注释规约

【强制】类、类属性、类方法的注释必须使用 Javadoc 规范，使用/*内容/ 格式，不得使用 // xxx 方式。

【强制】所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、 异常说明外，还必须指出该方法做什么事情，实现什么功能。

- 说明：对子类的实现要求，或者调用注意事项，请一并说明。

【强制】所有的类都必须添加创建者和创建日期。

【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释 使用/* */注释，注意与代码对齐。

【强制】所有的枚举类型字段必须要有注释，说明每个数据项的用途。

【推荐】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑 等的修改。

- 说明：代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，就失去了导航的意义。

【参考】谨慎注释掉代码。在上方详细说明，而不是简单地注释掉。如果无用，则删除。

【参考】对于注释的要求：第一、能够准确反应设计思想和代码逻辑；第二、能够描述业务含 义，使别的程序员能够迅速了解到代码背后的信息。完全没有注释的大段代码对于阅读者形同 天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路；注释也是给继任者看 的，使其能够快速接替自己的工作。

【参考】好的命名、代码结构是自解释的，注释力求精简准确、表达到位。避免出现注释的 一个极端：过多过滥的注释，代码的逻辑一旦修改，修改注释是相当大的负担。

【参考】特殊注释标记，请注明标记人与标记时间。注意及时处理这些标记，通过标记扫描， 经常清理此类标记。线上故障有时候就是来源于这些标记处的代码。

- 待办事宜（TODO）：（ 标记人，标记时间，[预计处理时间]） 表示需要实现，但目前还未实现的功能。这实际上是一个 Javadoc 的标签，目前的 Javadoc 还没有实现，但已经被广泛使用。只能应用于类，接口和方法（因为它是一个 Javadoc 标签）。
- 错误，不能工作（FIXME）：(标记人，标记时间，[预计处理时间]） 在注释中用 FIXME 标记某代码是错误的，而且不能工作，需要及时纠正的情况。

八、其他

【强制】在使用正则表达式时，利用好其预编译功能，可以有效加快正则匹配速度。说明：不要在方法体内定义：Pattern pattern = Pattern.compile(规则);

【强制】注意 Math.random() 这个方法返回是 double 类型，注意取值的范围 0≤x<1（能够 取到零值，注意除零异常），如果想获取整数类型的随机数，不要将 x 放大 10 的若干倍然后 取整，直接使用

Random 对象的 nextInt 或者 nextLong 方法。

【强制】获取当前毫秒数 System.currentTimeMillis(); 而不是 new Date().getTime();

【推荐】任何数据结构的构造或初始化，都应指定大小，避免数据结构无限增长吃光内存。

九、异常处理

【强制】捕获异常是为了处理它，不要捕获了却什么都不处理而抛弃之，如果不想处理它，请 将该异常抛给它的调用者。最外层的业务使用者，必须处理异常，将其转化为用户可以理解的 内容。

十、MySQL数据库

建立表规约

【强制】表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只 出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。

- 说明：MySQL 在 Windows 下不区分大小写，但在 Linux 下默认是区分大小写。因此，数据库名、表名、字段名，都不允许出现任何大写字母，避免节外生枝。
- 正例： aliyun_admin， rdc_config， level3_name 反例： AliyunAdmin， rdcConfig， level_3_name

【强制】禁用保留字，如 desc、range、match、delayed 等，请参考 MySQL 官方保留字。

【强制】主键索引名为 pk_字段名；唯一索引名为 uk_字段名；普通索引名则为 idx_字段名。 说明：pk_ 即 primary key；uk_ 即 unique key；idx_ 即 index 的简称。

【强制】小数类型为 decimal，禁止使用 float 和 double。

【强制】如果存储的字符串长度几乎相等，使用 char 定长字符串类型。

【强制】varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 text。

【强制】表必备三个字段：id，create_time, update_time, delete_flag

【强制】对于Boolean型的字段，采用decimal类型

【强制】表和字段都需要添加注释信息。

【推荐】单表行数超过 500 万行或者单表容量超过 2GB，才推荐进行分库分表。 说明：如果预计三年后的数据量根本达不到这个级别，请不要在创建表时就分库分表。

【参考】合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检 索速度。

索引规约

【强制】业务上具有唯一特性的字段，即使是多个字段的组合，也必须建成唯一索引。

- 说明：不要以为唯一索引影响了 insert 速度，这个速度损耗可以忽略，但提高查找速度是明 显的；另外，即使在应用层做了非常完善的校验控制，只要没有唯一索引，根据墨菲定律，必 然有脏数据产生。

【强制】超过三个表禁止 join。需要 join 的字段，数据类型必须绝对一致；多表关联查询时， 保证被关联的字段需要有索引。

- 说明：即使双表 join 也要注意表索引、SQL 性能。

【强制】在 varchar 字段上建立索引时，必须指定索引长度，没必要对全字段建立索引，根据 实际文本区分度决定索引长度即可。

- 说明：索引的长度与区分度是一对矛盾体，一般对字符串类型数据，长度为 20 的索引，区分度会高达 90%以上，可以使用 count(distinct left(列名, 索引长度))/count(*)的区分度 来确定。

【参考】创建索引时避免有如下极端误解：

- 宁滥勿缺。认为一个查询就需要建一个索引。
- 宁缺勿滥。认为索引会消耗空间、严重拖慢更新和新增速度。
- 抵制惟一索引。认为业务的惟一性一律需要在应用层通过“先查后插”方式解决。

SQL语句

【强制】不要使用 count(列名)或 count(常量)来替代 count()，*count()*是 SQL92 定义的标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。

- 说明：count(*)会统计值为 NULL 的行，而 count(列名)不会统计此列为 NULL 值的行。

【强制】count(distinct col) 计算该列除 NULL 之外的不重复行数，注意 count(di col1, col2) 如果其中一列全为 NULL，那么即使另一列有不同的值，也返回为 0。

【强制】当某一列的值全是 NULL 时，count(col)的返回结果为 0，但 sum(col)的返回结果为NULL.

【强制】不得使用外键与级联，一切外键概念必须在应用层解决。

【强制】禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。

【推荐】in 操作能避免则避免，若实在避免不了，需要仔细评估 in 后边的集合元素数量，控 制在 1000 个之内。

【参考】如果有全球化需要，所有的字符存储与表示，均以 utf-8 编码，注意字符 的区别。

- 说明：SELECT LENGTH("轻松工作")；返回为 12 SELECT CHARACTER_LENGTH("轻松工作")；返回为 4 如果需要存储表情，那么选择 utfmb4 来进行存储，注意它与 -8 编码的区别。

【强制】更新数据表记录时，必须同时更新记录对应的 update_time 字段值为当前时间。

【参考】@Transactional 事务不要滥用。事务会影响数据库的 QPS，另外使用事务的地方需 要考虑各方面的回滚方案，包括缓存回滚、搜索引擎回滚、消息补偿、统计修正等。

推荐阅读(点击即可跳转阅读)

1.SpringBoot内容聚合
2.面试题内容聚合
3.设计模式内容聚合
4.Mybatis内容聚合
5.多线程内容聚合

觉得不错？欢迎转发分享给更多人

我知道你 “在看”