技术管理文档-Java

学号：2022141461136 姓名：刘俊伶

tips

本文档参考的是阿里巴巴Java开发手册 终极版
《阿里巴巴Java开发手册（终极版）》免费下载*在线阅读*藏经阁-阿里云开发者社区
我会分为`编程规约` `异常日志` `MySQL数据库` 三个方面简述

一、编程规约

1.1命名风格

强制性

1. 不能以下划线或者美元作为开头结尾
2. 严禁使用拼音和英文混合的方式，更不能直接使用中文
3. 所有类名都必须使用UpperCamelCase驼峰命名法
4. 方法参数名使用lowerCamleCase
5. 常量命名必须全部大写，单词间用下划线隔开

推荐

1. 接口类等等使用了设计模式的要在命名时体现具体模式
2. 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思

允许

枚举类名带上Enum后缀，枚举成员名称需要全部大写

1.2常量定义

强制性

1. 不允许使用任何魔法值
2. long或者Long赋值，使用大写L，不能写小写l，容易混淆，例如21L不能写成2ll

推荐

1. 不要使用一个常量类维护所有常量，按照常量功能进行归类，分开维护
2. 常用的复用层次有五层：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： 阿里巴巴Java开发手册 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。

1.3代码格式

强制性

1. 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. 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。
3. if/for/while/switch/do等保留字与括号之间都必须加空格。
4. 任何二目、三目运算符的左右两边都需要加一个空格。
5. 采用4个空格缩进，禁止使用tab字符。 说明：如果使用tab缩进，必须设置1个tab为4个空格。IDEA设置tab为4个空格时， 请勿勾选Use tab character；而在eclipse中，必须勾选insert spaces for tabs。

推荐

1. 没有必要增加若干空格来使某一行的字符与上一行对应位置的字符对齐。

允许

1. 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。

1.4集合处理

强制性

1. 关于hashCode和equals的处理，遵循如下规则： 1） 只要重写equals，就必须重写hashCode。 2） 因为Set存储的是不重复的对象，依据hashCode和equals进行判断，所以Set存储的 对象必须重写这两个方法。 3） 如果自定义对象做为Map的键，那么必须重写hashCode和equals。
2. ArrayList的subList结果不可强转成ArrayList，否则会抛出ClassCastException 异常，即 java.util.RandomAccessSubList cannot be cast to java.util.ArrayList. 说明：subList 返回的是 ArrayList 的内部类 SubList，并不是 ArrayList ，而是 ArrayList 的一个视图，对于SubList子列表的所有操作最终会反映到原列表上。
3. 在subList场景中，高度注意对原集合元素个数的修改，会导致子列表的遍历、增加、 删除均会产生 ConcurrentModificationException 异常。

推荐

1. 集合初始化时，指定集合初始值大小。
2. 使用entrySet遍历Map类集合KV，而不是keySet方式进行遍历。：keySet其实是遍历了2次，一次是转为Iterator对象，另一次是从hashMap中取出 key 所对应的value。而entrySet只是遍历了一次就把key和value都放到了entry中，效 率更高。如果是JDK8，使用Map.foreach方法。
3. 高度注意Map类集合K/V能不能存储null值的情况，如下表格：

集合类	Key	Value	Super	说明
Hashtable	不允许为 null	不允许为 null	Dictionary	线程安全
ConcurrentHashMap	不允许为 null	不允许为 null	AbstractMap	锁分段技术（JDK8:CAS）
TreeMap	不允许为 null	允许为 null	AbstractMap	线程不安全
HashMap	允许为 null	允许为 null	AbstractMap	线程不安全

允许

合理利用好集合的有序性(sort)和稳定性(order)，避免集合的无序性(unsort)和 不稳定性(unorder)带来的负面影响。

二、异常日志

2.1异常处理

强制性

1. Java 类库中定义的一类RuntimeException可以通过预先检查进行规避，而不应该 通过catch 来处理，比如：IndexOutOfBoundsException，NullPointerException等等。 说明：无法通过预检查的异常除外，如在解析一个外部传来的字符串形式数字时，通过catch NumberFormatException 来实现。
2. 异常不要用来做流程控制，条件控制，因为异常的处理效率比条件分支低。
3. 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。

推荐

1.方法的返回值可以为null，不强制返回空集合，或者空对象等，必须添加注释充分 说明什么情况下会返回null值。调用方需要进行null判断防止NPE问题。
2. 定义时区分unchecked / checked 异常，避免直接抛出new RuntimeException()， 更不允许抛出Exception或者Throwable，应使用有业务含义的自定义异常。推荐业界已定义 过的自定义异常，如：DAOException / ServiceException等。

允许

1. 在代码中使用“抛异常”还是“返回错误码”，对于公司外的http/api开放接口必须 使用“错误码”；而应用内部推荐异常抛出；跨应用间RPC调用优先考虑使用Result方式，封 装isSuccess()方法、“错误码”、“错误简短信息”。

2.2日志规约

强制性

1. 应用中不可直接使用日志系统（Log4j、Logback）中的API，而应依赖使用日志框架 SLF4J 中的API，使用门面模式的日志框架，有利于维护和各个类的日志处理方式统一。 import org.slf4j.Logger; import org.slf4j.LoggerFactory; private static final Logger logger = LoggerFactory.getLogger(Abc.class);
2. 日志文件推荐至少保存15天，因为有些异常具备以“周”为频次发生的特点。
3. 对trace/debug/info级别的日志输出，必须使用条件输出形式或者使用占位符的方 式。

推荐

1. 谨慎地记录日志。生产环境禁止输出debug日志；有选择地输出info日志；如果使 用warn来记录刚上线时的业务行为信息，一定要注意日志输出量的问题，避免把服务器磁盘 撑爆，并记得及时删除这些观察日志。
2. 可以使用warn日志级别来记录用户输入参数错误的情况，避免用户投诉时，无所适 从。注意日志输出的级别，error级别只记录系统逻辑出错、异常等重要的错误信息。如非必 要，请不要在此场景打出error级别。

三、MySQL数据库

3.1建表规约

强制性

1. 表达是与否概念的字段，必须使用is_xxx的方式命名，数据类型是unsigned tinyint （1表示是，0表示否）。
2. 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。
3. 表名不使用复数名词。

推荐

1. 表的命名最好是加上“业务名称_表的作用”。
2. 库名与应用名称尽量一致

允许

1. 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。

3.2索引规约

强制性

1. 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。
2. 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
3. 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。

推荐

1. 如果有order by的场景，请注意利用索引的有序性。order by 最后的字段是组合 索引的一部分，并且放在索引组合顺序的最后，避免出现file_sort的情况，影响查询性能。 正例：where a=? and b=? order by c; 索引：a_b_c
2. 利用覆盖索引来进行查询操作，避免回表。

允许

防止因字段类型不同造成的隐式转换，导致索引失效。

3.3ORM映射

强制性

1. 在表查询中，一律不要使用 * 作为查询的字段列表，需要哪些字段必须明确写明。（减少获取成本）
2. POJO类的布尔属性不能加is，而数据库字段必须加is_，要求在resultMap中进行 字段与属性之间的映射。
3. 不要用resultClass当返回参数，即使所有类属性名与数据库字段一一对应，也需 要定义；反过来，每一个表也必然有一个与之对应。
4. iBATIS自带的queryForList(String statementName,int start,int size)不推 荐使用。

推荐

@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。

允许

@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。