链滴

异常日志规范

作者：xixiaoming

原文链接：https://ld246.com/article/1542683076647

来源网站：链滴

异常处理

- 【强制】不要捕获Java类库中定义的继承自RuntimeException的运行时异常类,如:IndexOutOfBoundsException / NullPointerException,这类异常由程序员预检查来规避,保证程序健壮性.

正例:if(obj != null) {...}

反例:try { obj.method() } catch(NullPointerException e){ ... }

- 【强制】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.
- 【强制】一大段代码进行try-catch,这是不负责任的表现.catch时请分清稳定代码和非稳定代码,稳定代码指的是无论如何不会出错的代码.对于非稳定代码的catch尽可能进行区分异常类型,再做对应的异常处理.
- 【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.
- 【强制】有try块放到了事务代码中,catch异常后,如果需要回滚事务,一定要注意手动回滚事务.
- 【强制】finally块必须对资源对象、流对象进行关闭,有异常也要做try-catch.

说明: 如果JDK7,可以使用try-with-resources方法.

- 【强制】不能在finally块中使用return,finally块中的return返回后方法结束执行,不会再执行try块中return语句.
- 【强制】捕获异常与抛异常,必须是完全匹配.捕获异常必须是抛异常的父类.

说明: 如果预期抛的是绣球,实际接到的是铅球,就会产生意外情况.

- 【推荐】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下会返回null值.调用方需要进行null判断防止NPE问题.

说明: 本规约明确防止NPE是调用者的责任.即使被调用方法返回空集合或者空对象,对调用者来说,也并高枕无忧,必须考虑到远程调用失败,运行时异常等场景返回null的情况.

- 【推荐】防止NPE,是程序员的基本修养,注意NPE产生的场景:

1. 返回类型为包装数据类型,有可能是null,返回int值时注意判空.

反例: public int f(){ return Integer对象},如果为null,自动解箱抛NPE.

2. 数据库的查询结果可能为null.
3. 集合里的元素即使isNotEmpty,取出的数据元素也可能为null.
4. 远程调用返回对象,一律要求进行NPE判断.
5. 对于Session中获取的数据,建议NPE检查,避免空指针.
6. 级联调用obj.getA().getB().getC();一连串调用,易产生NPE.

反例: "一拍档客户"的返回值从空对象变成了null,导致线上故障,NPE无小事.

- 【推荐】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HSF调用优先考虑使用Result方式,封装isSuccess、"错误码"、"错简短信息".
- 【推荐】定义时区分unckecked / checked 异常,避免直接使用RuntimeException抛出,更不允许出Exception或者Throwable,应使用有业务含义的自定义异常.推荐业界或者集团已定义过的自定义异

,如:DaoException / ServiceException等.

- 【参考】避免出现重复的代码(Don’ t Repeat Yourself),即DRY原则.

说明: 随意复制和粘贴代码,必然会导致代码的重复,在以后需要修改时,需要修改所有的副本,容易遗漏.要时抽取共性方法,或者抽象公共类,甚至是共用模块.

正例: 一个类中有多个public方法,都需要进行数行相同的参数校验操作,这个时候请抽取:

private boolean checkParam(DTO dto){ ... }

日志规约

- 【强制】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)的API.

说明: 日志框架(SLF4J、JCL--Jakarta Commons Logging)的使用方式(推荐使用SLF4J):

使用SLF4J:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
private static final Logger logger = LoggerFactory.getLogger(Abc.class);
```

- 【强制】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.对于当天日志,以"用名.log"来保存,保存在/home/admin/应用名/logs/目录下,过往日志格式为: {logname}.log.{保存日},日期格式:yyyy-MM-dd

说明: 以mppserver应用为例,日志保存在/home/admin/mppserver/logs/mppserver.log,历史日志称为mppserver.log.2016-08-01

- 【强制】应用中的扩展日志(如打点、临时监控、访问日志等)命名方式:appName_logType_logNae.log.logType:日志类型,推荐分类有stats/desc/monitor/visit等;logName:日志描述.这种命名的好处通过文件名就可知道日志文件属于什么应用,什么类型,什么目的,也有利于归类查找.

正例: mppserver应用中单独监控时区转换异常,如:mppserver_monitor_timeZoneConvert.log

说明: 推荐对日志进行分类,错误日志和普通提示日志尽量分开存放,便于开发人员查看,也便于通过日志系统进行及时监控.

- 【强制】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,否则量的对象toString和字符串拼接会带来严重的性能问题.

正例: (条件)

```
if (logger.isDebugEnabled()) { logger.debug("Processing trade with id: " + id + " symbol: " symbol); }
```

 正例: (占位符)

```
logger.debug("Processing trade with id: {} and symbol : {} ", id, symbol);
```

- 【强制】避免重复打印日志,浪费磁盘空间,务必在log4j.xml中设置additivity=false.

正例:name="com.taobao.ecrm.member.config" additivity="false"

- 【强制】生产环境禁止直接使用System.out 或System.err 输出日志或使用e.printStackTrace()打异常堆栈.由于标准日志输出与标准错误输出文件每次Jboss重启时才滚动,如果大量输出送往这两个文件容易造成文件大小超过操作系统大小限制.
- 【强制】异常信息应该包括两类信息:案发现场信息和异常堆栈信息.如果不处理,那么往上抛.

正例: logger.error(各类参数或者对象toString + "_" + e.getMessage(), e);

输出的POJO类必须重写toString方法,否则只输出此对象的hashCode值(地址值),没啥参考意义.

- 【推荐】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.注意日输出的级别,error级别只记录系统逻辑出错、异常、或者重要的错误信息.如非必要,请不要在此场景打error级别,避免频繁报警.
- 【推荐】如果使用log.warn记录跟踪调试信息,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.
- 【参考】如果日志用英文描述不清楚,推荐使用中文注释.对于中文UTF-8的日志,在secureCRT中,set ncoding=utf-8;如果中文字符还乱码,请设置:全局>默认的会话设置>外观>字体>选择字符集gb2312 如果还不行,执行命令:set termencoding=gbk,并且直接使用中文来进行检索.