Java编写规范

一、命名规范

已经被使用的常量，不要重新定义

约定俗成的常量含义，不要重新定义。

努力避免硬编码。

每个模块，建议有独立的常量类。

方法名都以lowerCamelCase风格编写

类名都以UpperCamelCase风格编写

参数名以lowerCamelCase风格编写

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。

二、函数／方法

public 函数(方法)，对象参数 必须要处理参数为null的情况，

private 函数，对象参数 可以不用处理参数为null的情况(依情况而定)

方法名都以lowerCamelCase 开头单词小写后面驼峰风格编写

方法长度不超过50行

三、嵌套层级不要超过3层。

for，while，if ，switch 等。

方法中条件不成立直接return,不再向下执行

如：

public int xxx (String userId, String password, String email){

if(StringUtils.isEmpty(userId) || StringUtils.isEmpty(email)

||StringUtils.isEmpty(password)){

return null;

}

UserEntity ue = EntityProxy.OBJ.get(userId, UserEntity.class);

if(ue == null){

return null;

}

……...

}

避免前套层次过深，建议不超过三层

四、代码结构

1垂直

以下情况需要使用一个空行：

类内连续的成员之间：字段，构造函数，方法，嵌套类，静态初始化块，实例初始化块。

例外：两个连续字段之间的空行是可选的，用于字段的空行主要用来对字段进行逻辑分组。

在函数体内，语句的逻辑分组间使用空行。

类内的第一个成员前或最后一个成员后的空行是可选的(既不鼓励也不反对这样做，视个人喜好而定)。

要满足本文档中其他节的空行要求。

多个连续的空行是允许的，但没有必要这样做(我们也不鼓励这样做)。

一个类的总体行数尽量控制在400行左右(不超过一千行)。

五、资源处理

EntityTransaction tranx = null;

try {

// 获取数据库事务

tranx = em.getTransaction();

// 开始事务过程

if(!tranx.isActive()){

tranx.begin();

}

// 保存实体

Query query = em.createNamedQuery(jpqlName);

if(params!=null && !params.isEmpty()){

params.forEach((k,v)->{

query.setParameter(k, v);

});

}

query.executeUpdate();

// 提交事务

tranx.commit();

} catch (Exception ex) {

// 记录错误日志

DaoLog.LOG.error("删除对象异常");

DaoLog.LOG.error(ex.getMessage(), ex);

if(tranx != null){

tranx.rollback();

}

return false;

}finally{

if(tranx !=null && tranx.isActive()){

tranx.commit();

}

em.close();//注意 用完一定要释放

}

六、异常处理

比较底层的处理单元，建议抛出异常。

业务处理模块，处理异常的同时，异常必须要加日志！最好有finally处理。

方法返回结果，不要使用异常方式。

七、相同的代码快，不要到处出现或者重复出现！

相同代码提取处理,让代码可重用。

八、注释

1、源文件注释

源文件注释采用 /\* …… /，在每个源文件的头部要有必要的注释信息，包括：文件名；文件编号；版本号；作者；创建时间；文件描述包括本文件历史修改记录等。中文注释模版：

/\*\*

文 件 名 :

CopyRright (c) 2015-xxxx:

文件编号：

创 建 人：

日 期：

修 改 人：

日 期：

描 述：

版 本 号：

\*/

2、类（模块）注释：

类（模块）注释采用 /\* …… /，在每个类（模块）的头部要有必要的注释信息，包括：工程名；类（模块）编号；命名空间；类可以运行的JDK版本；版本号；

作者；创建时间；类（模块）功能描述（如功能、主要算法、内部各部分之间的关系、该类与其类的关系等，必要时还要有一些如特别的软硬件要求等说明）；

主要函数或过程清单及本类（模块）历史修改记录等。

3、接口注释：

接口注释采用 /\* …… /，在满足类注释的基础之上，接口注释应该包含描述接口的目的、它应如何被使用以及如何不被使用，块标记部分必须注明作者和版本。

在接口注释清楚的前提下对应的实现类可以不加注释。

4、构造函数注释：

构造函数注释采用 /\* …… /，描述部分注明构造函数的作用，不一定有块标记部分。

5、函数注释：

函数注释采用 /\* ……/，在每个函数或者过程的前面要有必要的注释信息，包括：函数或过程名称；功能描述；

输入、输出及返回值说明；调用关系及被调用关系说明等。函数注释里面可以不出现版本号（@version）。

6、方法注释：

方法注释采用 /\* …… /，普通成员方法要求说明完成什么功能，参数含义是什么且返回值什么；另外方法的创建时间必须注释清楚，为将来的维护和阅读提供宝贵线索。

7、方法内部注释：

控制结构，代码做了些什么以及为什么这样做，处理顺序等，特别是复杂的逻辑处理部分，要尽可能的给出详细的注释。

8、全局变量注释：

要有较详细的注释，包括对其功能、取值范围、哪些函数或者过程存取以及存取时注意事项等的说明。

9、局部（中间）变量注释：

主要变量必须有注释，无特别意义的情况下可以不加注释。

10、实参/参数注释：

参数含义、及其它任何约束或前提条件。

十、if else 条件含义要明确

如：

if (isOk) { //isOK 如何

return Response.status(200).entity(resp.getData()).build();

} else {

return Response.status(200).entity(resp.getErrorInfo()).build();

}

十一、逻辑控制，不要瀑布流！

尽量把条件不满足的情况写在某个逻辑块的前面(比如方法的最前面)，让不满足条件的情况快速失败，让代码整理结构清晰，可读。

十二、巧用构造函数构造者builder模式：

构造方法：UserDetailInfo userinfo = new UserDetailInfo(user);

builder方式：

Map<String, Object> oparams = ImmutableMap.<String, Object> builder()

.put("appid", ConfigUtil.APPID)// 服务号的应用号

.put("body", WeixinConstant.PRODUCT\_BODY)// 商品描述

.put("mch\_id", ConfigUtil.MCH\_ID)// 商户号 ？

.put("nonce\_str", PayCommonUtil.CreateNoncestr())// 16随机字符串(大小写字母加数字)

.put("out\_trade\_no", orderId)// 商品订单号

.put("total\_fee", "1")// 银行币种

.put("spbill\_create\_ip", ip)// IP地址

.put("notify\_url", ConfigUtil.NOTIFY\_URL) // 微信回调地址

.put("trade\_type", "APP")// 支付类型 app

.build();

十三、数据结构与业务处理（算法）分开

如： MVC MVVM 都可以参考

十四、关键业务添加日志记录

LoggerUtils.loginLogger.info(String.format("xx用户[%s] at %s 登陆xxx app", sb.toString(),DateUtils.getDateTime()));