方法上不应该不写注释

单词名称不应该无法识别(影响读代码的理解)

类的作者和方法的作者不应该不写

Controller层

类上不应该不加上swagger @Api注解



改为

@Api(value = "SayController|一个用来测试swagger注解的控制器")

方法上不应该不加上 @ApiOperation、@ApiImplicitParam注解



改为

@ApiOperation(value="根据用户编号获取用户姓名", notes="test: 仅1和2有正确返回")  
@ApiImplicitParam(paramType="query", name = "userNumber", value = "用户编号", required = true, dataType = "Integer")

分页对象应该在controller进行封装并传递到service层

业务参数应该封装到业务对象再传递给service层

判断字符串为空不应该直接判断而是要调用StringUtils类的isEmpty方法



改为



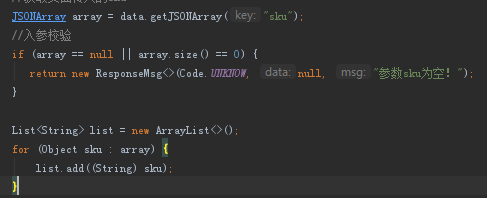
返回的异常信息不应该直接写中文,而是要用国际化(i18n),引用i18n code



请求类型不应该写为固定类

响应类型不应该写为固定类

数据集合转换不应该



改为

*List<String> list1 = array.toJavaList(String.class);*

Service层

方法上需要加入本地事务和分布式事务(lcn)注解

@LcnTransaction

或者

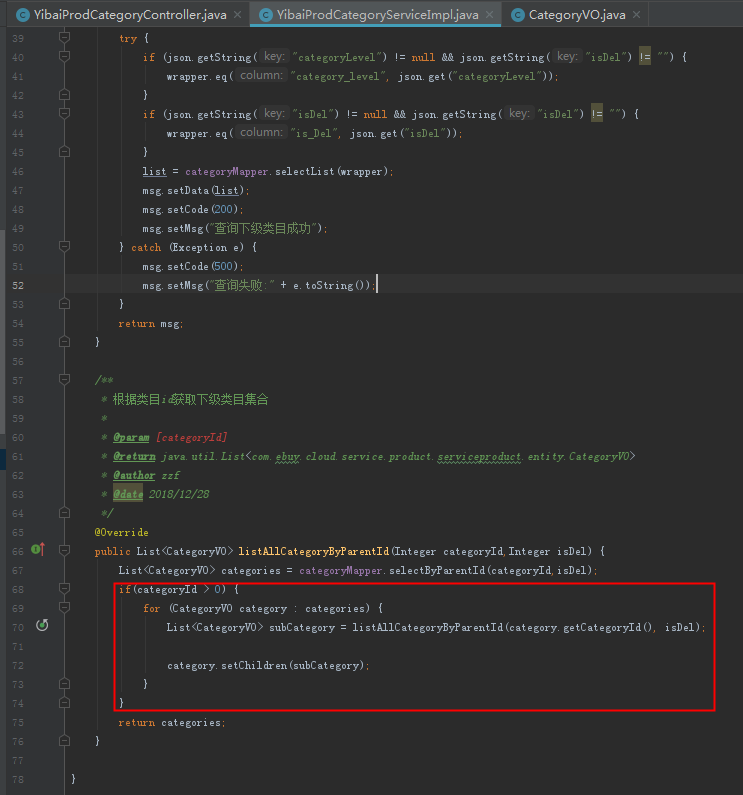
@TxcTransaction(propagation = DTXPropagation.SUPPORTS)  
@Transactional

公共响应对象不应该不提供构建函数，以提升代码风格

ResponseMsg msg = new ResponseMsg();  
msg.setData(list);  
msg.setCode(200);

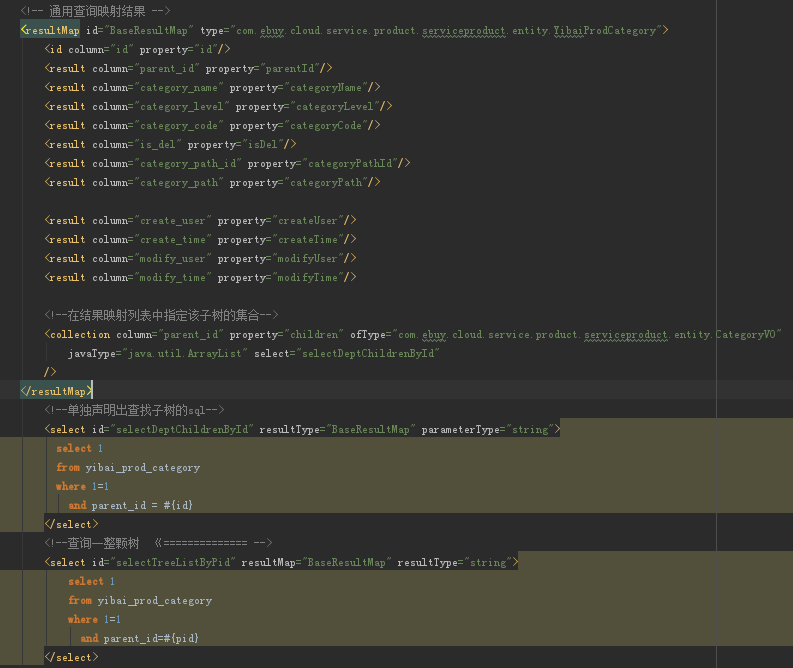
msg.setMsg("查询下级类目成功");

自关联查询不应该放到业务层去做,而是应该在sql层完成

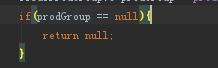


应该修改为(该类中自关联的List属性不能写到toString中,死循环)

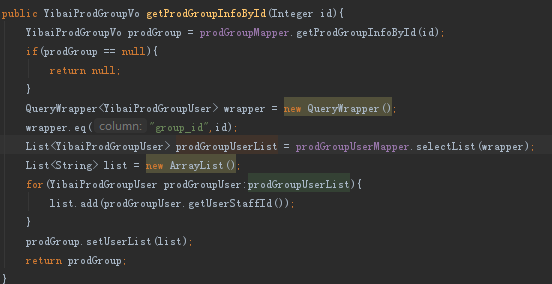
private List<CategoryVO> children;

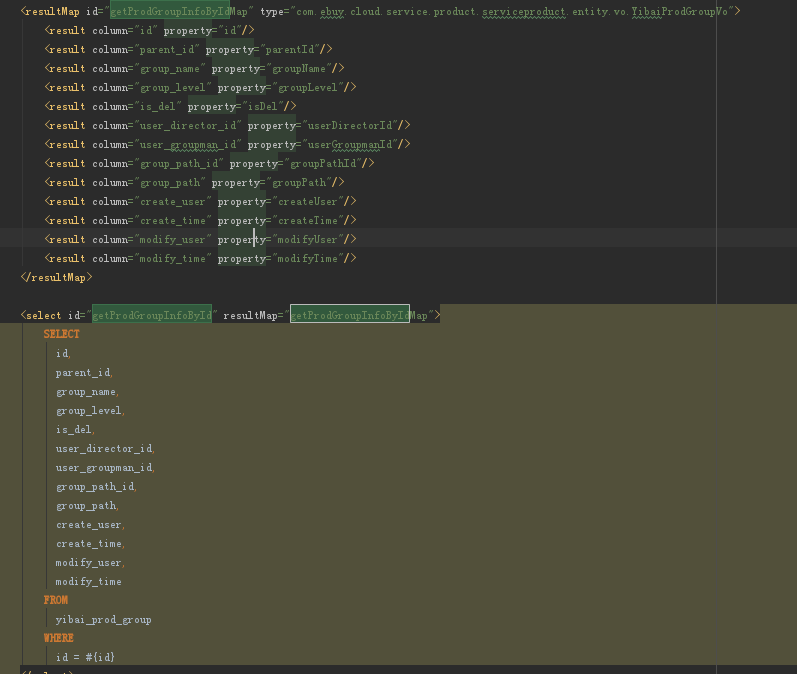


判断逻辑不应该重复

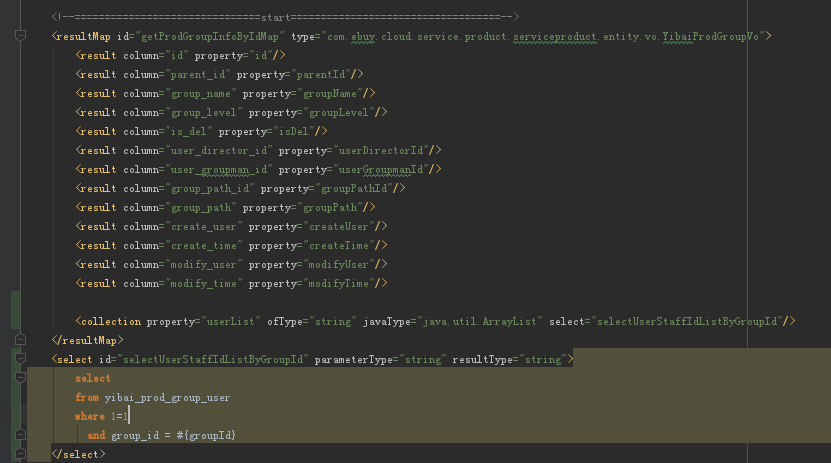


单条SQL的逻辑不应该拆成多条

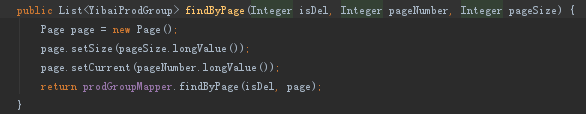


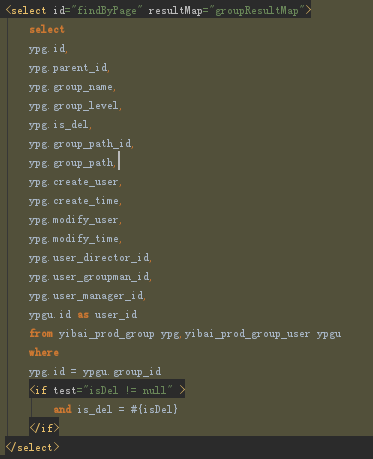


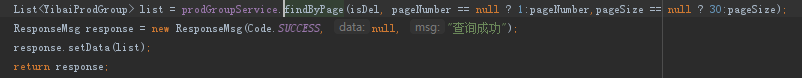
改为



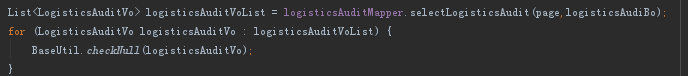
逻辑表达不应该混乱

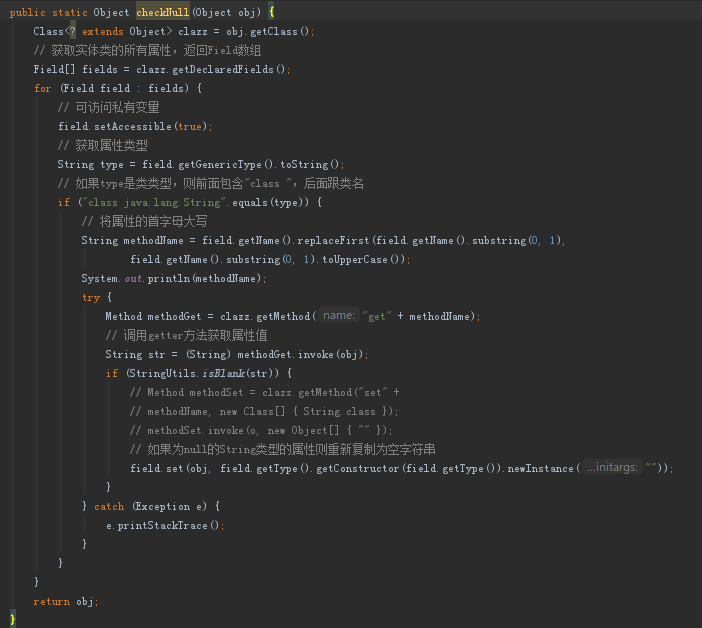






空值替换的做法造成的性能有影响

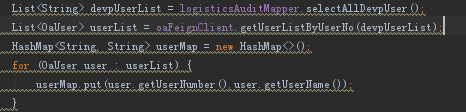




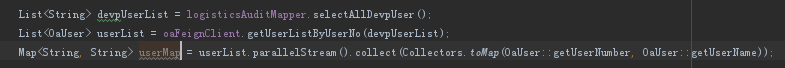
改为



转换list为map不应该使用单线程遍历



改为java8的stream 并行流

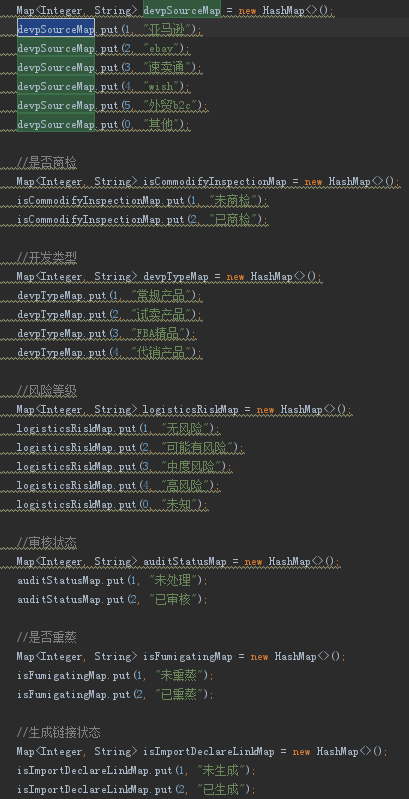


Map<String, String> userMap = userList.parallelStream().collect(Collectors.*toMap*(OaUser::getUserNumber, OaUser::getUserName));

或者这种

groupNameList.parallelStream().collect(Collectors.*toMap*(t -> Integer.*valueOf*(String.*valueOf*(t.get("id"))), t -> String.*valueOf*(t.get("groupName"))));

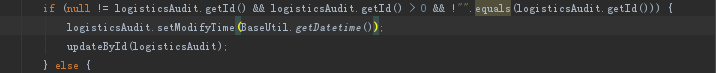
固定数据不能再代码中写死



改为

配置文件中读取并初始化

业务层不要设置修改时间(受数据库版本影响)



改为

让数据库自动更新该时间

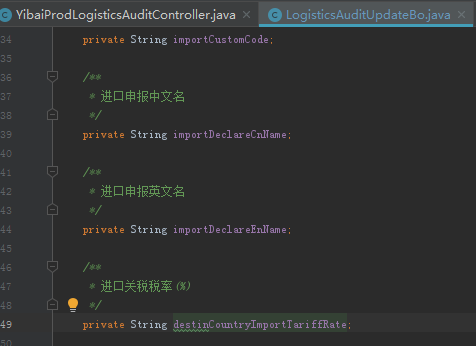
业务层不要设置创建时间(受数据库版本影响)



改为

让数据库自动创建该时间

BO业务对象不应该将应该为BigDecimal类型的设置为String



改为

*/\*\*  
 \* 进口关税税率(%)  
 \*/*private BigDecimal destinCountryImportTariffRate;

日志打印不规范



改为(待定)

修改操作不应该不添加本地事务注解、分布式事务(lcn)注解

判断逻辑不应该反过来(影响阅读)





改为:

if (null == id || id < 0){

值判断不应该为具体值



改为

将是改为Y

使用stream时不应该使用stream()



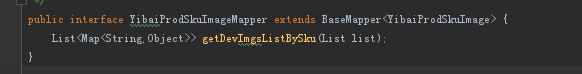
改为

oaUserList = oaUserList.parallelStream().filter(user -> user.getUserName().contentEquals(userName)).collect(Collectors.*toList*());

Mapper层

不要全部手写dao层而是要尽可能的用BaseMapper,例如mybatis plus

类上不要全不写SpringBootCache注解(以提升系统查询性能,缓解数据库压力)



改为类上:

@MapperScan

@CacheConfig(cacheNames = "system:model:table:user")

方法上:

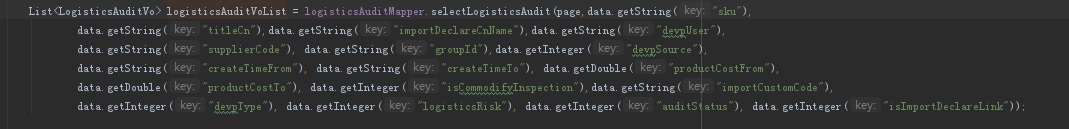
@Cacheable(key = "#p1+'\_'+#p2")

接口类中的方法不应该写public(默认public)

Mapper-Xml中

javaType不应该不为该类的全包名

请求类型不应该为具体字段的类型而是应该为类的类型，以提高拓展性



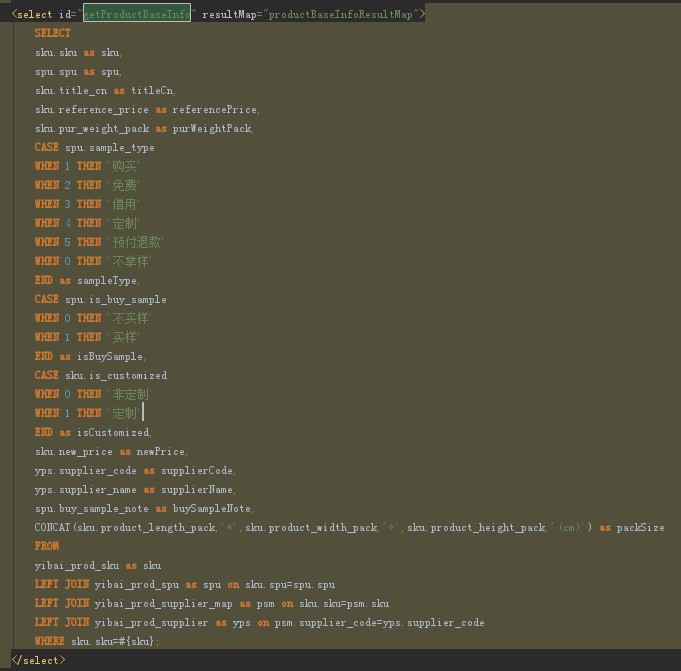
不要手动声明数据字段到实体类属性(下划线转驼峰),应该为全局配置

**mybatis**:  
 **configuration**:  
 **map-underscore-to-camel-case**: **true**

不要手动指定一对多、一对一配置,使用自动配置

公共sql不应该每个子类维护一份而是放到公共mapper文件中,而在具体mapper中通过include标签从该公共mapper文件的命名空间中去引用

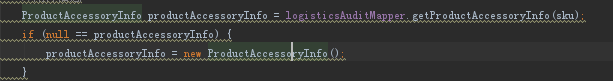
Sql中不应该设置具体到值



改为

根据code从i18n中拿取

查询判断的写法不应该过于重复



改为

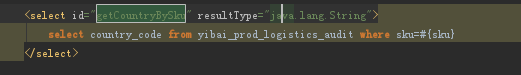


根据编码取具体值操作不应该直接从数据库查询sql 

改为

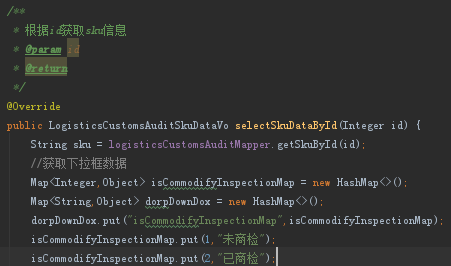
从单独的缓存中取

返回值类型/参数类型如果为基本类型则不应该带包名

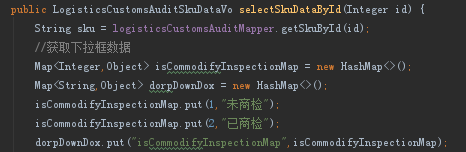




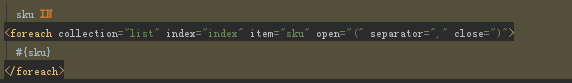
集合设值逻辑不清晰



改为

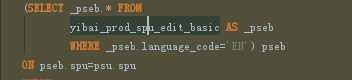


Sql中in中的item的数量,不应该不受到控制(受数据库所在服务器的性能影响,mysql没有跟oracle一样限制1000条)



改为多个in

Sql中不要写\*



Bean层

不应该放到业务工程内部而是要放到业务公共模块并去依赖

类名上不应该不写表名

字段上不应该不写swagger注解以标注该字段的信息

@ApiModelProperty(value="医生姓名" ,required=true)

公共属性不应该每个去写一份而是抽取到公共实体类中,并且子类去继承

*/\*\*  
 \* 删除标记，1：已删除，0：保留的  
 \*/*private Boolean isDel;  
*/\*\*  
 \* 创建人  
 \*/*private String createUser;  
*/\*\*  
 \* 创建时间  
 \*/*private LocalDateTime createTime;  
*/\*\*  
 \* 修改人  
 \*/*private String modifyUser;  
*/\*\*  
 \* 修改时间  
 \*/*private LocalDateTime modifyTime;