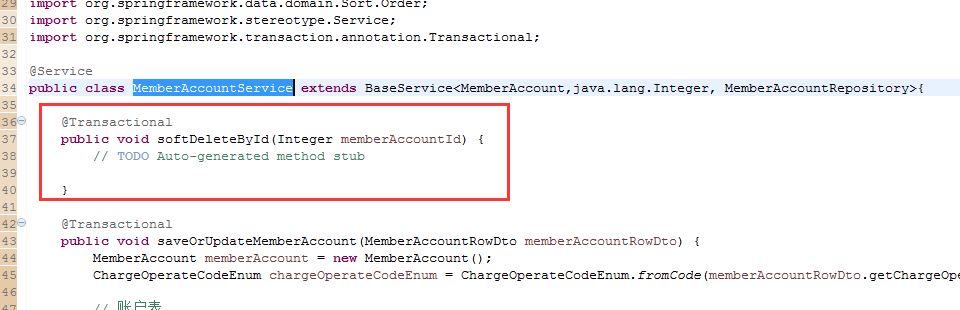
# BUG整理

# 可读性优化

## 项目残留无用代码

**内容：**



**可能导致后果：**

系统可读性变差，可维护性变低。还可以导致别人维护你的代码出现无用的工作。

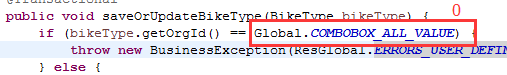
**解决方案：**

如果单元完成还需要临时留做测试请添加TODO以及说明文字为什么要保留，否则请注释或清理掉。

## 代码出现硬编码，没有考虑引用Global常量

**内容：**

代码里面不要出现硬编码





**可能导致后果：**

出现魔鬼数字，无法通过refactor来重构

**解决方案：**

代码中任何一个出现的数字你都必须有理由说明他为什么要直接用数字表示而无法使用GLOBAL的有意义的常量。例如size=0这个属于常规判断，而size<=999可能这个999可以抽离成常量BODY\_MAX\_SIZE变成size<=BODY\_MAX\_SIZE这样代码意义就更加明确

## 展示内容没有从角色的角度来考虑

**内容：**

展示的内容要考虑导出给什么角色看，从角色的角度来思考自己的展示是否合理



**可能导致后果：**

客户会表示你根本没理解他想要的东西

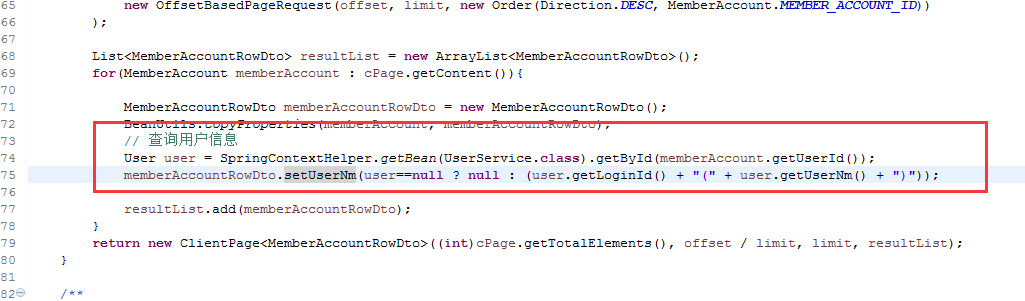
**解决方案：**

从使用角色的角度来思考一下问题，例如客户不是DBA，接触不到数据库，你这些数据呈现给他有用吗？有没有其它合理的模式？

## 表格列格式化显示渲染问题，在后端写死代码

**内容：**

视图的数据请不应该改变实体字段的内容定义！

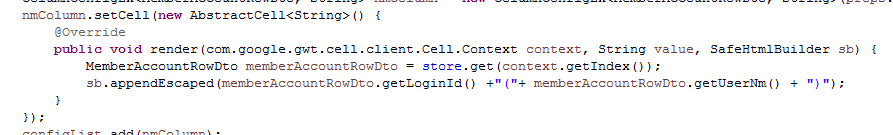
****

**可能导致后果：**

在后端的代码里写死，不宜维护

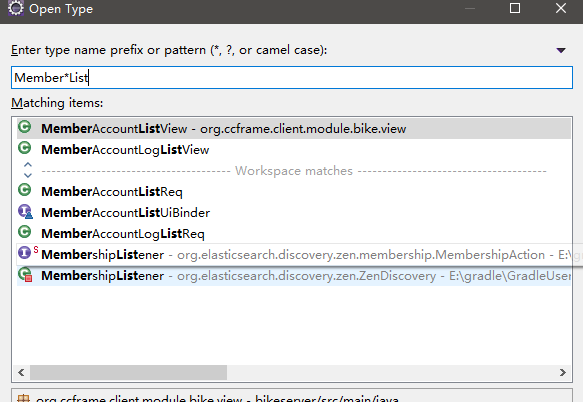
**解决方案：**

表格列格式化显示、拼接显示应使用表格列模式渲染



## 类命名是否符合规范，方便codeView

**内容：**



**可能导致后果：**

若类命名不符合规范，系统可读性变差，可维护性变低。别人接收项目则无法维护

**解决方案：**

GWT相关命名：

列表的视图用\*ListView、增加修改窗体用\*WindowView、

Service相关命名：

service与searchService

# 功能或性能问题

## 没有考虑过巨量数据的情况直接全部列出来循环过滤

**内容：**

findBykey或findAll这类函数for循环的时候没有考虑巨量数据，这等于是把系统所有的数据都要过一边这个查询才能执行完

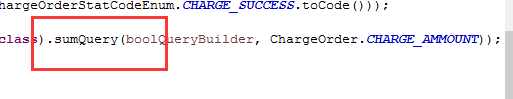
****

**可能导致后果：**

增大系统压力，甚至会卡死

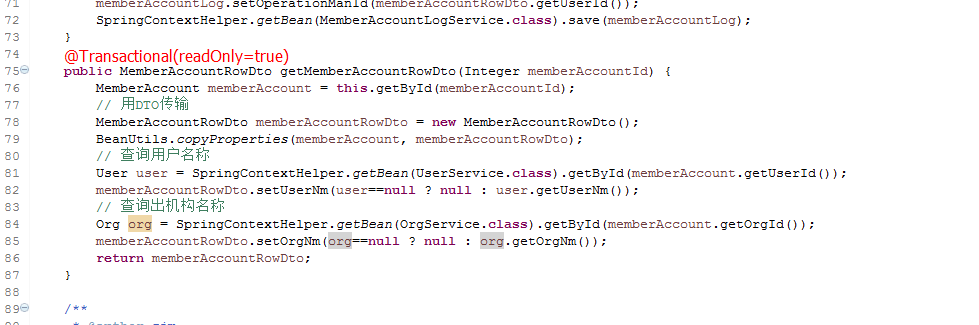
**解决方案：**

采用ES的聚集查询来实现汇总



## 漏掉事务申明

**内容：**

****

**可能导致后果：**

查询需要事务，否则会泄漏连接

**解决方案：**

Service类中，操作数据库的方法一定要检查是否添加了事物申明

## 前端的代码逻辑在后端处理

**内容：**

GWT前端的检验在后端重复校验

****

**可能导致后果：**

系统冗余

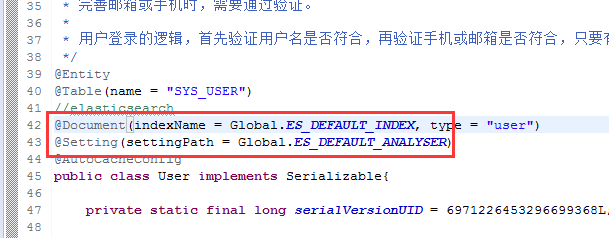
**解决方案：**

在GWT前端进行相应的逻辑、校验处理

## entity类的IndexName索引使用不同的字段

**内容：**

不同的entity类的IndexName处应用了不同的字段

****

**可能导致后果：**

否则会造成索引库之间的隔离，查询的时候无法跨表查询

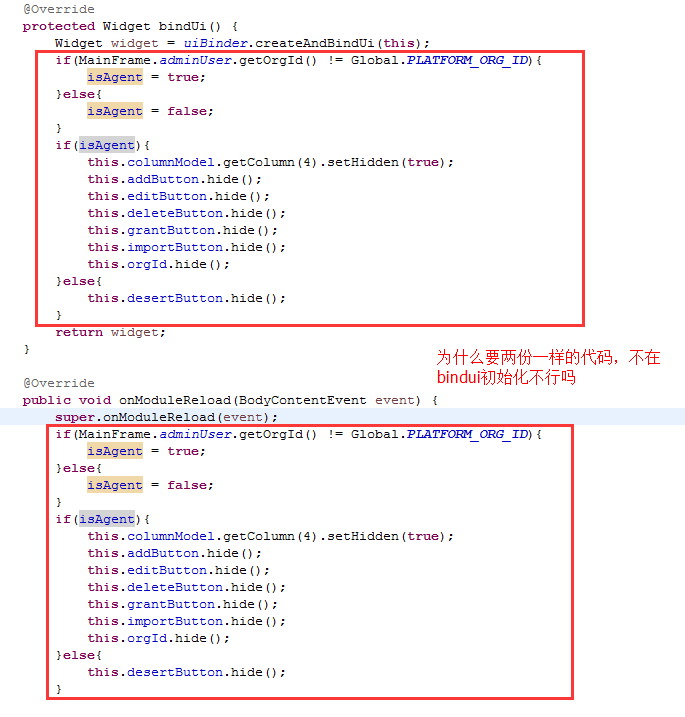
**解决方案：**

IndexName此处应用同一个

## GWT的初始化应放在bindUI（），逻辑处理放在onMoudleRoload（）

**内容：**

逻辑代码写在了只执行一次的bindUI()函数

****



**可能导致后果：**

**解决方案：**

逻辑代码写在onMoudleRoload()，如判断总平台和运营商的登陆

## 区别ViewUtil和Info.display

**内容：**

****



**可能导致后果：**

**解决方案：**

ViewUtil的使用：

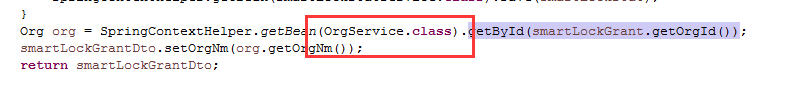
ViewUtil.messageBox("成功添加运营商", message);

Info的使用：

Info.display("操作完成", "单车类型" + (bikeTypeId == null ? "新增" : "修改") + "成功");

## getByKey用searchService，用索引查询

**内容：**

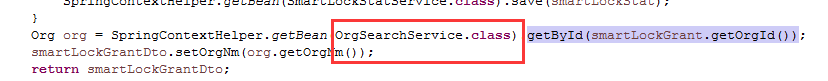


**可能导致后果：**

无法使用索引查询

**解决方案：**

getByID()、getByKey等查询方法应使用searchService来查询



## 主键生产策略，用GenerationType.IDENTITY

**内容：**

主键生产策略应使用GenerationType.IDENTITY，而不是GenerationType.AUTO

****

**可能导致后果：**

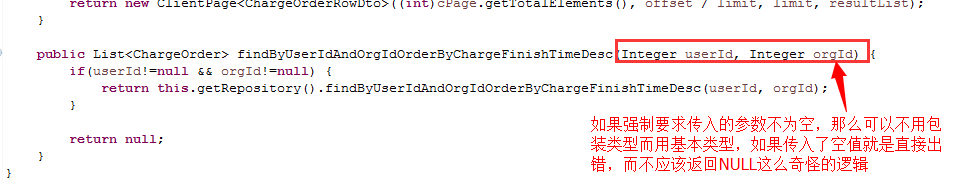
数据库操作时主键无法自增

**解决方案：**

应使用IDENTITY，而不是AUTO

## 方法参数类型传值问题，是包装类型还是基本类型

**内容：**

****

**可能导致后果：**

如果参数传入空值就会直接出错

**解决方案：**

应使用基本类型