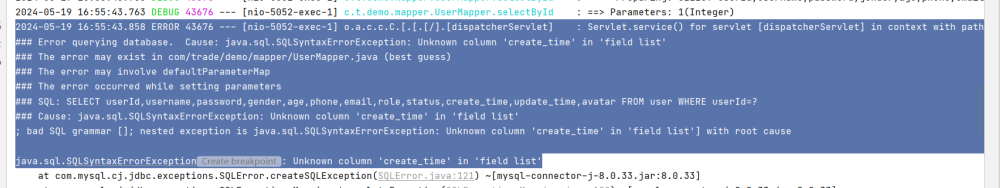
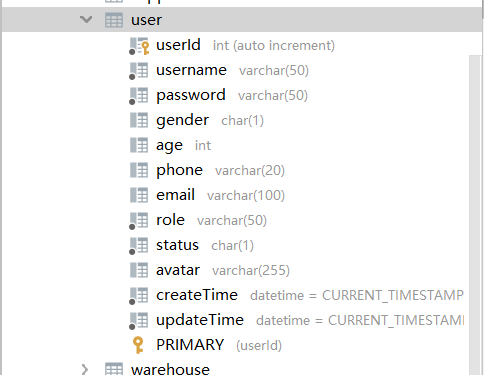
# Debug参考书



这个错误表明你的数据库中 **user** 表没有 **create\_time** 列，而实体类 **User** 和/或 MyBatis-Plus 查询试图访问该列。为了解决这个问题，需要确保数据库列名和实体类属性名之间的一致性。



我的user表是这样的,的确没有create\_time,出现错误的原因是,mybatis对实体类中属性中的大写字母对应数据库中的"\_小写字母". 例如, 我数据表中属性有一个为e\_mail,那么,java后端的实体类就应填写eMail, 这样,mybatis解析eMail才会对应e\_mail。

解决方法一：修改数据表结构，改为create\_time,update\_time

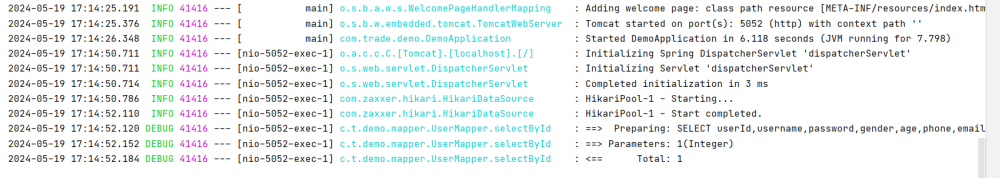
解决方法二:修改实体类和数据库表的映射关系

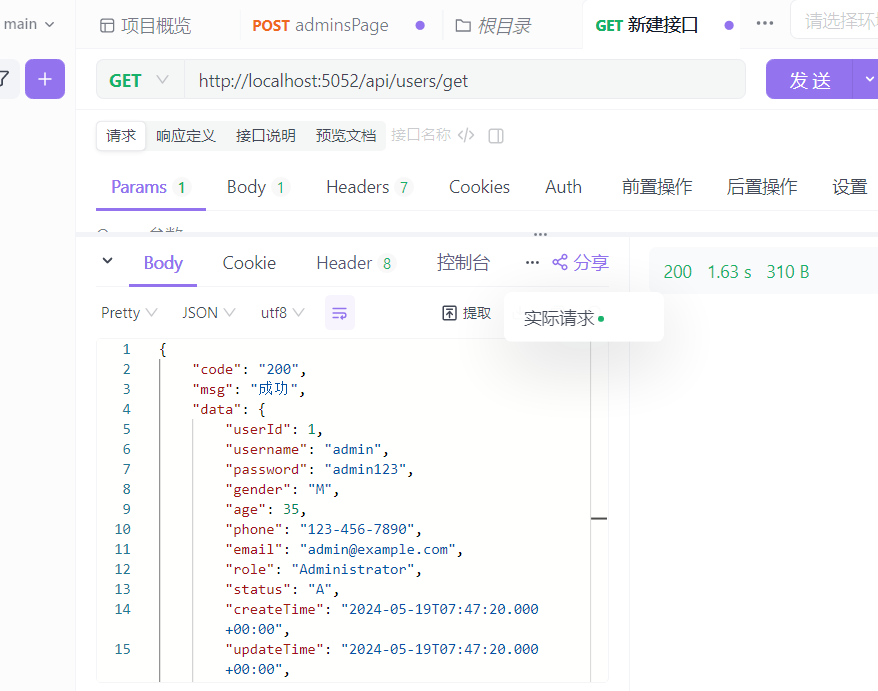
使用**@TableField** 注解来指定实体类字段与数据库表列之间的映射关系

@TableField("createTime")  
@ApiModelProperty(value = "创建时间")  
private Date createTime;  
@TableField("updateTime")  
@ApiModelProperty(value = "更新时间")  
private Date updateTime;

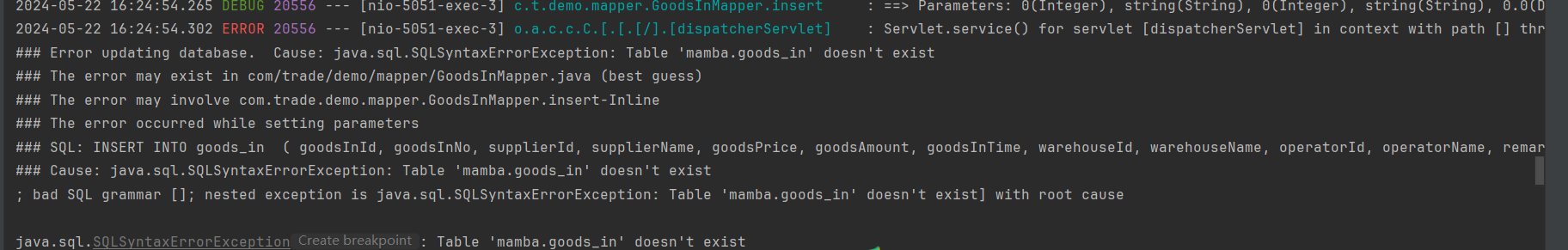
在 MyBatis-Plus 中，默认情况下，它会将实体类的驼峰命名法（CamelCase）转换为数据库表的下划线命名法（snake\_case）。因此，如果你的数据库表中的列名是 **create\_time**，而你的实体类属性名是 **createTime**，MyBatis-Plus 会自动进行映射。然而，如果数据库表中的列名不遵循这种命名规则（例如 **e\_mail**），那么需要明确地指定映射关系。

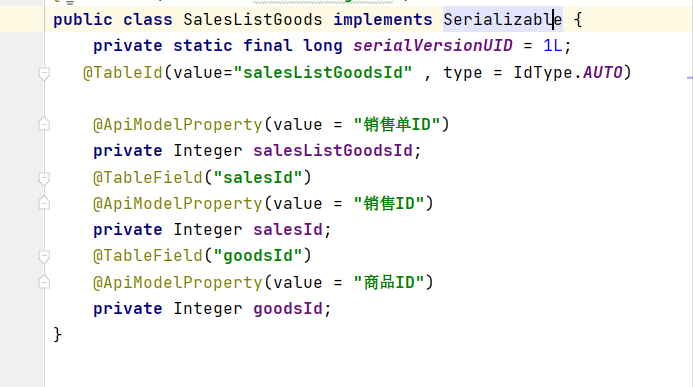
效果





发现也有表名的匹配的问题

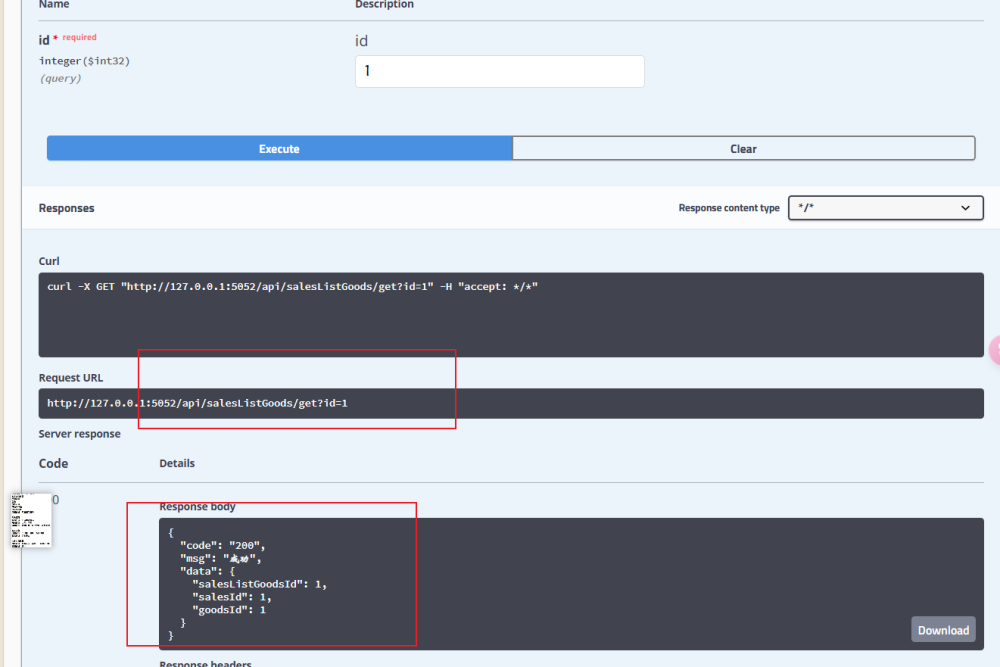


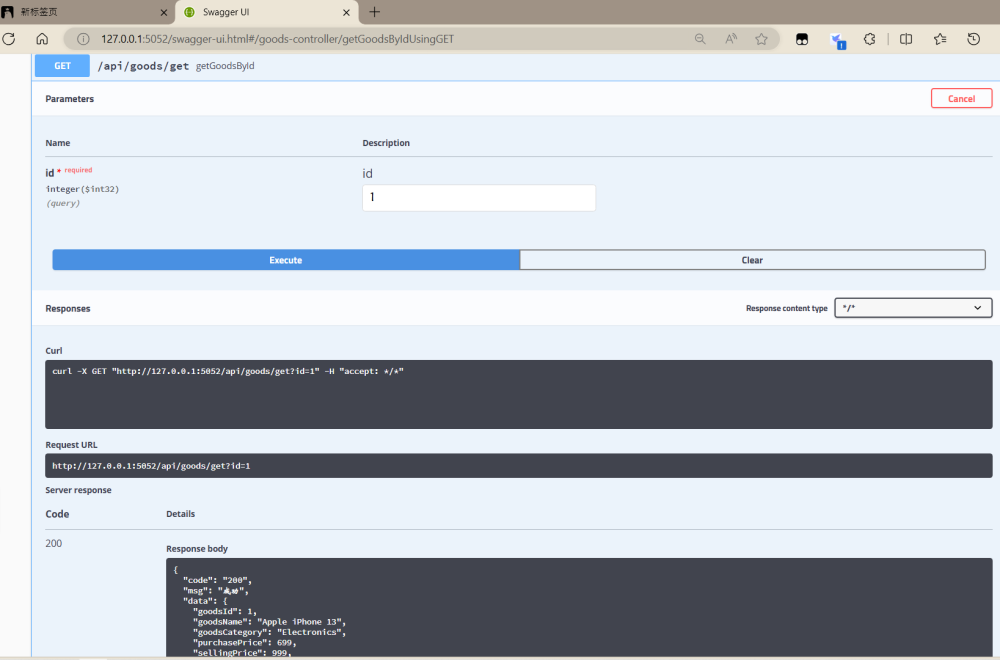


意外发现重修定义也有问题, 差点大改数据库,解决方案是给类加上表的注解,值为数据库中表的值



成功:

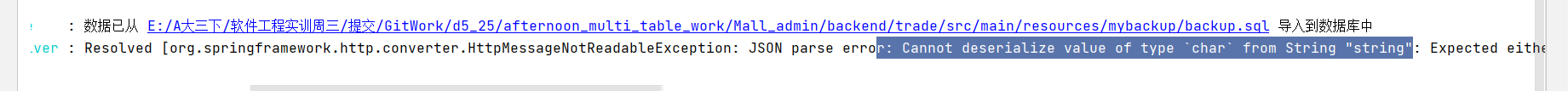


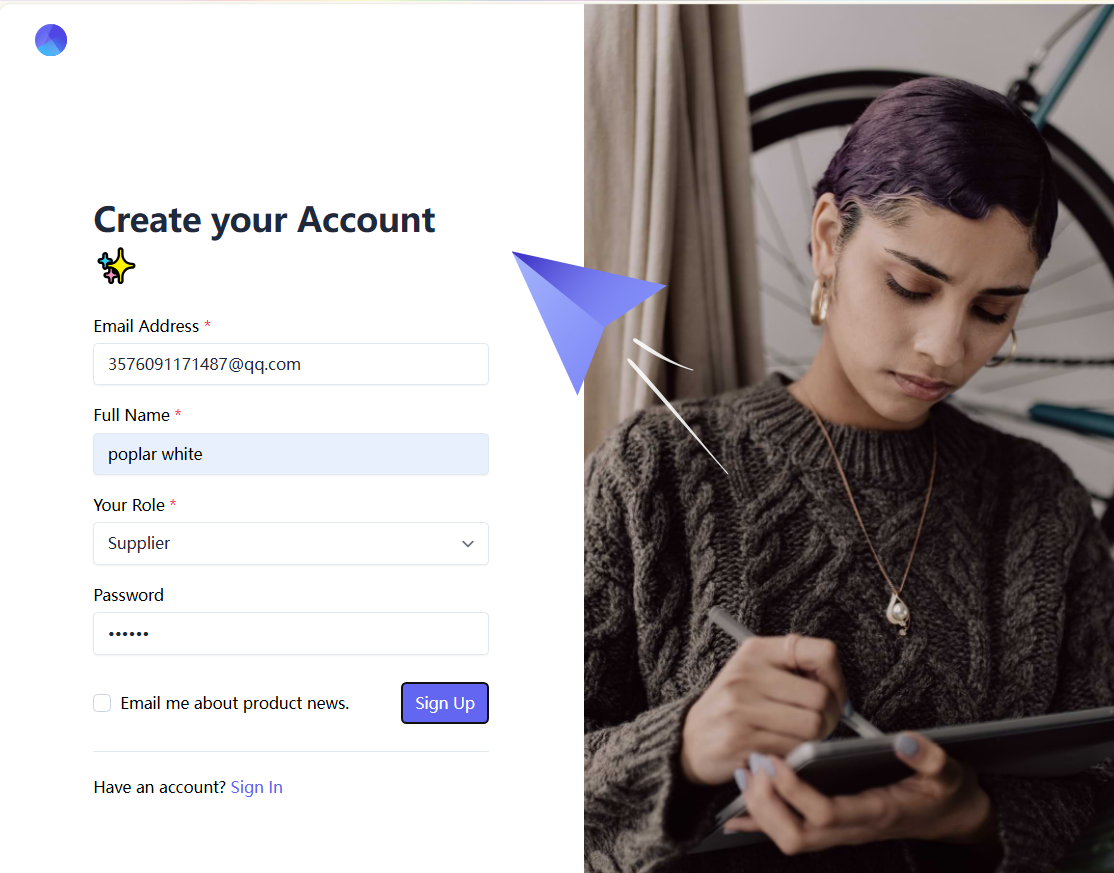
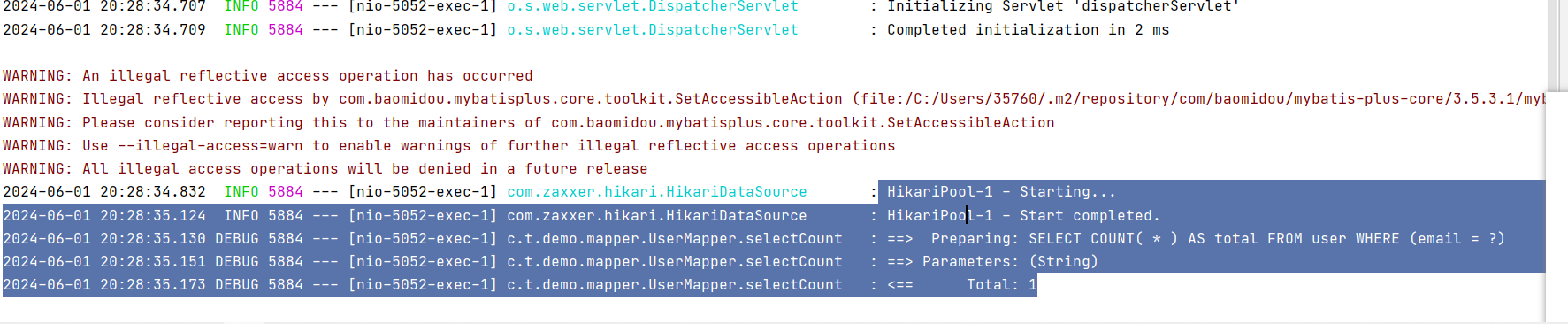


对于在Swagger UI进行测试时,尤其是进行插入,update操作时,配置的样例数据的json是将属性自动填满了的,但有的类型不匹配。如status是char类型的，如果直接执行这个样例代码，会失败并后端idea查看可以知道错误类型(swagger错误认为status是string的)



后端提示：





前端在传入参数时,似乎出现错误

2024-06-01 20:28:35.130 DEBUG 5884 --- [nio-5052-exec-1] c.t.demo.mapper.UserMapper.selectCount : ==> Preparing: SELECT COUNT( \* ) AS total FROM user WHERE (email = ?)

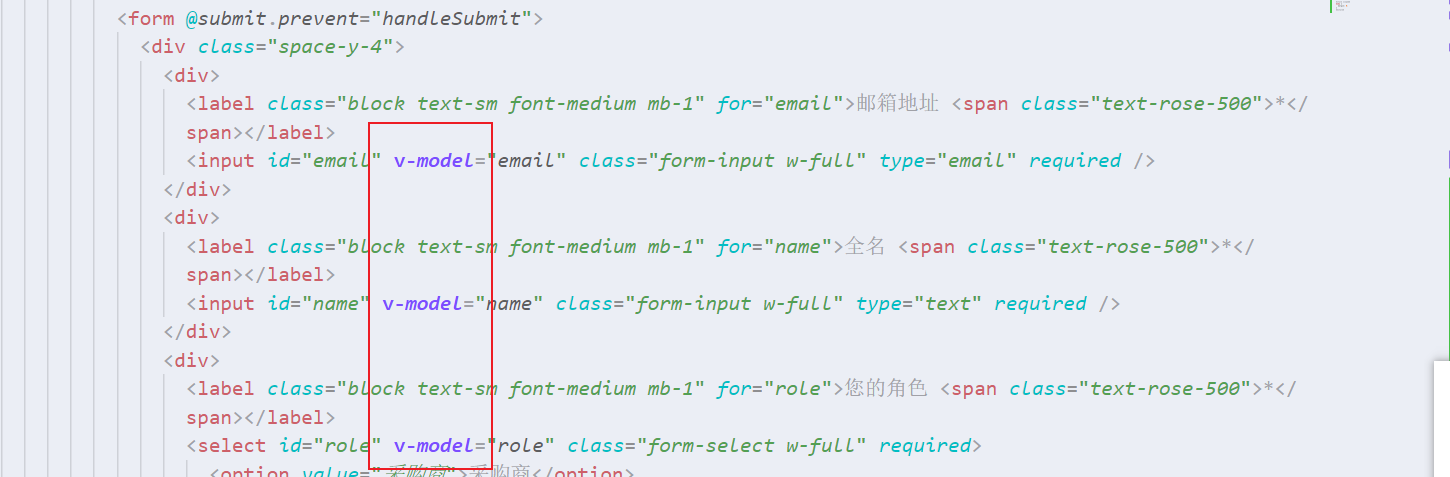
2024-06-01 20:28:35.151 DEBUG 5884 --- [nio-5052-exec-1] c.t.demo.mapper.UserMapper.selectCount : ==> Parameters**: (String)**

2024-06-01 20:28:35.173 DEBUG 5884 --- [nio-5052-exec-1] c.t.demo.mapper.UserMapper.selectCount : <== Total: 1

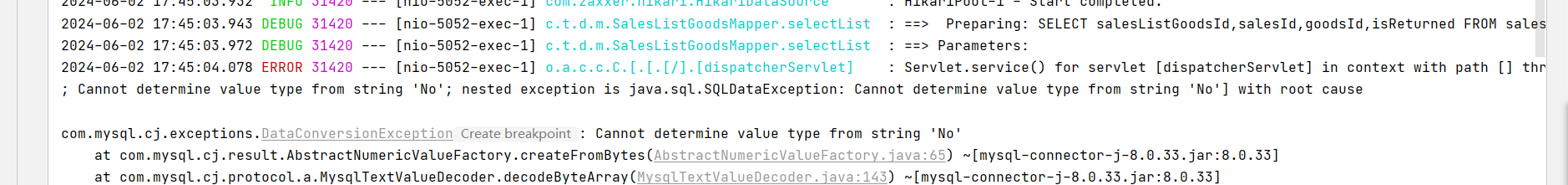
可见传参并没有传入输入的数据。

这是由于前端form表单没有绑定v-model进行信息的绑定,导致表单提交时能够正确获取数据

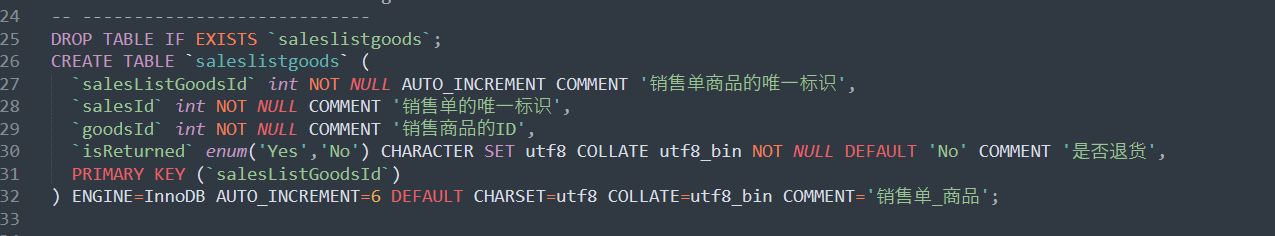
解决部分代码如下:



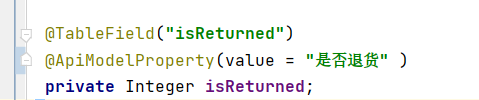
在进行查询代售商品时,出现了有关是否退货属性的错误,



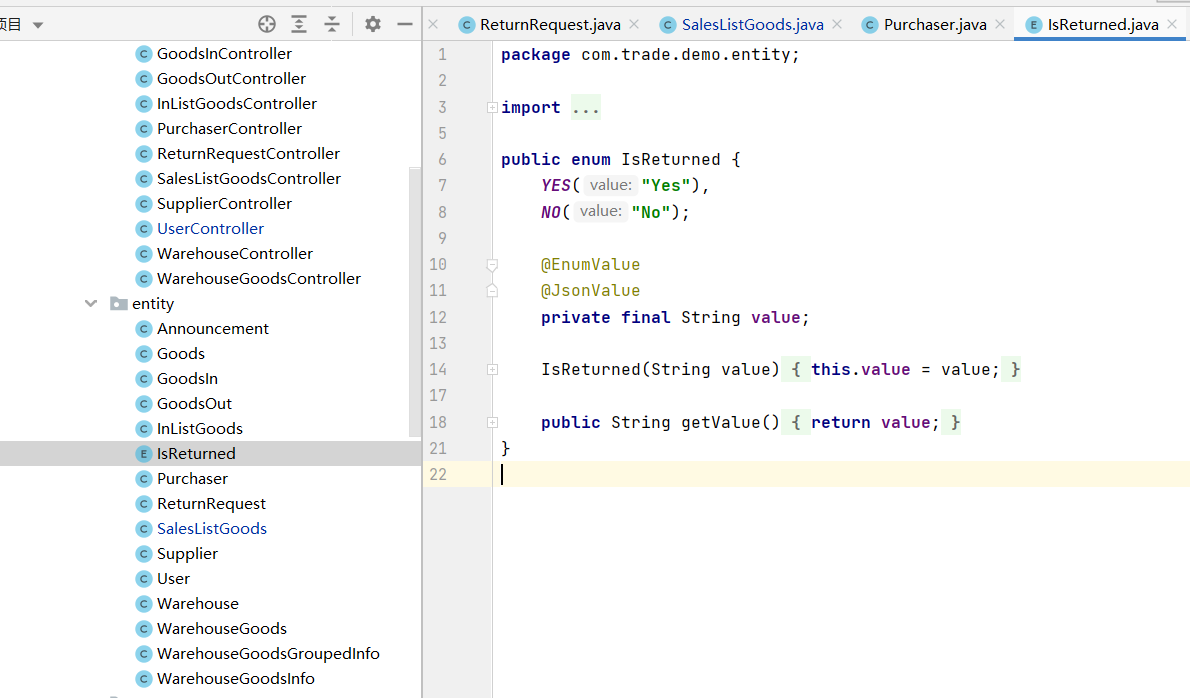
相关数据表如下



可以看到, 这里是定义的枚举类型, idea提示,cannot determine value type , 查看java中的实体类



发现给的是Integer, 找到问题所在

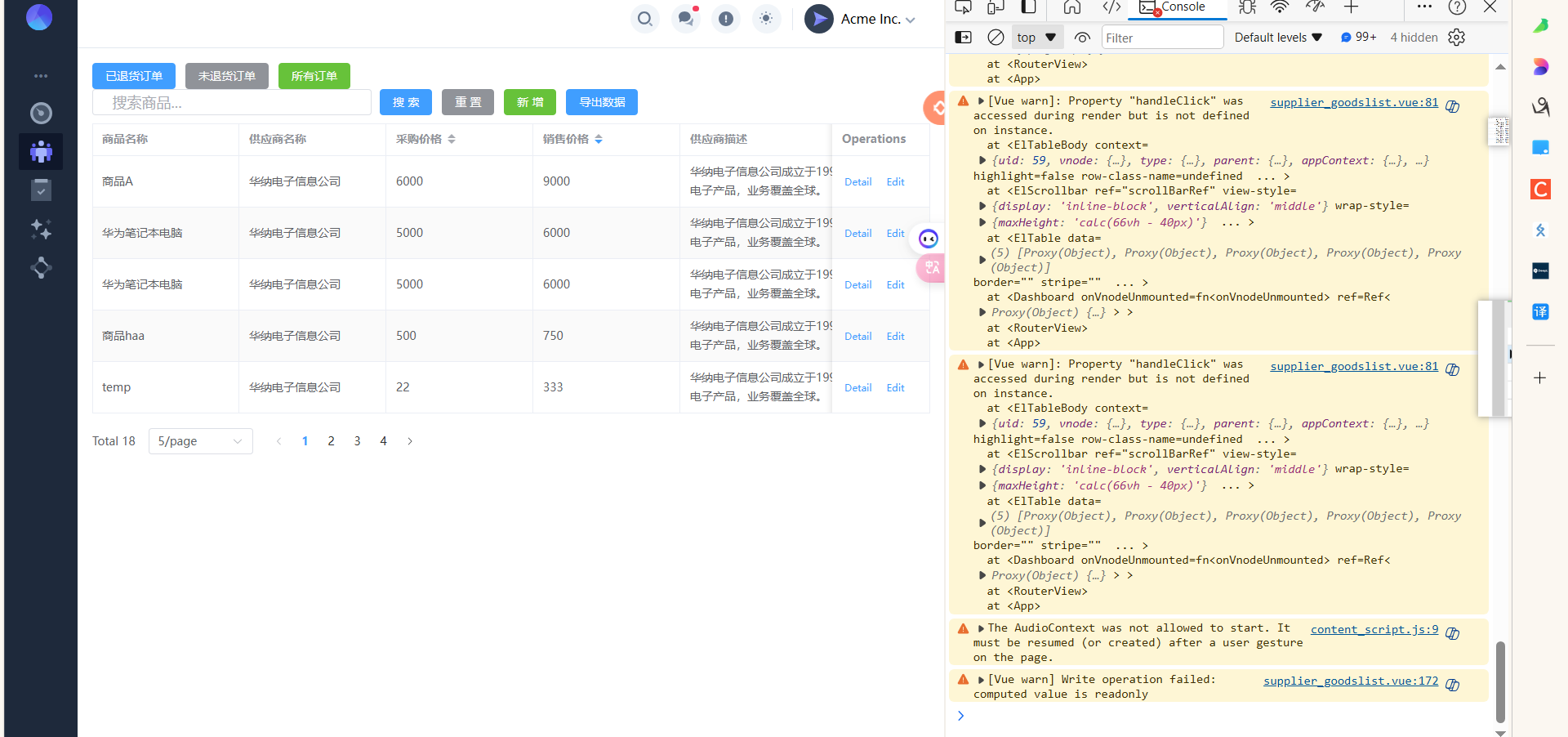


添加上述自定义的实体类, @TableField(**"isReturned"**)  
@ApiModelProperty(value = **"是否退货"** )  
**private** IsReturned **isReturned**;

将属性改为这个即可,如下图请求成功。



前端渲染问题:



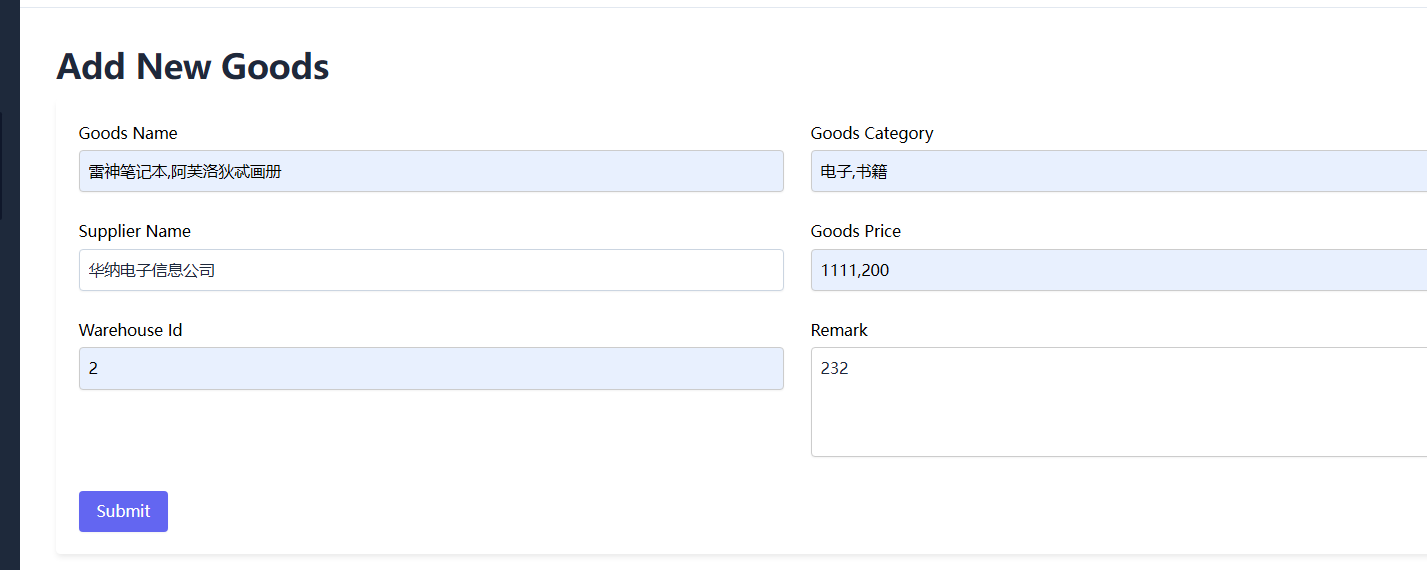
提示为拒接写操作,这个问题是在我进行通过按钮过滤退货订单, 未退货订单出现的。

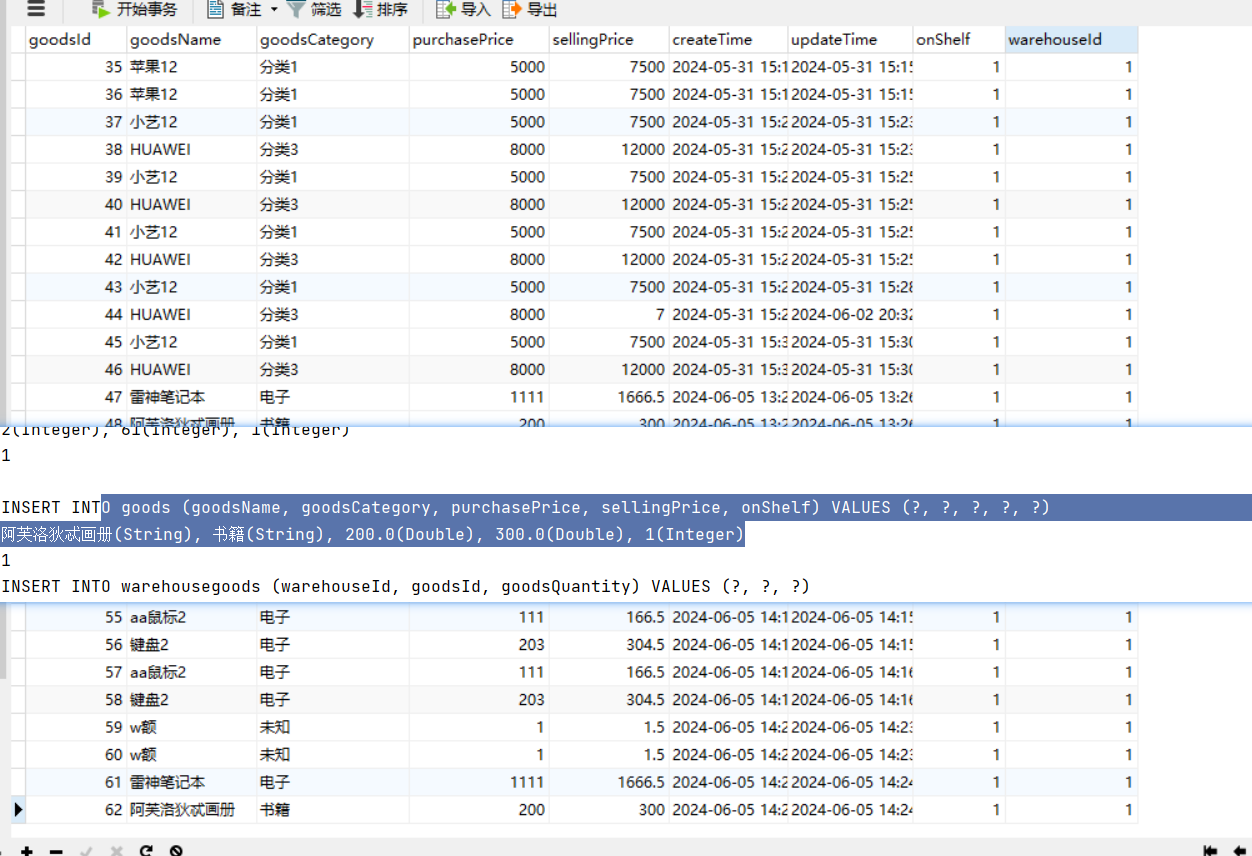


原因: 计算属性 filteredGoods 中进行写操作，而计算属性是只读的。计算属性是用于派生状态，而不用于修改状态。

解决方法:

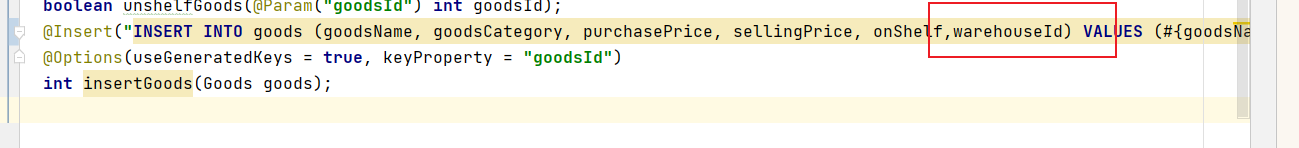
请求数据后,将数据进行存储,再对数据进行computed

发现商品的插入, warehouseId一直为1,而供应商是提供的2, 



由sql语句,可以看到,插入的属性缺少了warehouseId,这可能导致每次插入使用的是默认的1.检查。

在持久层进行补充：

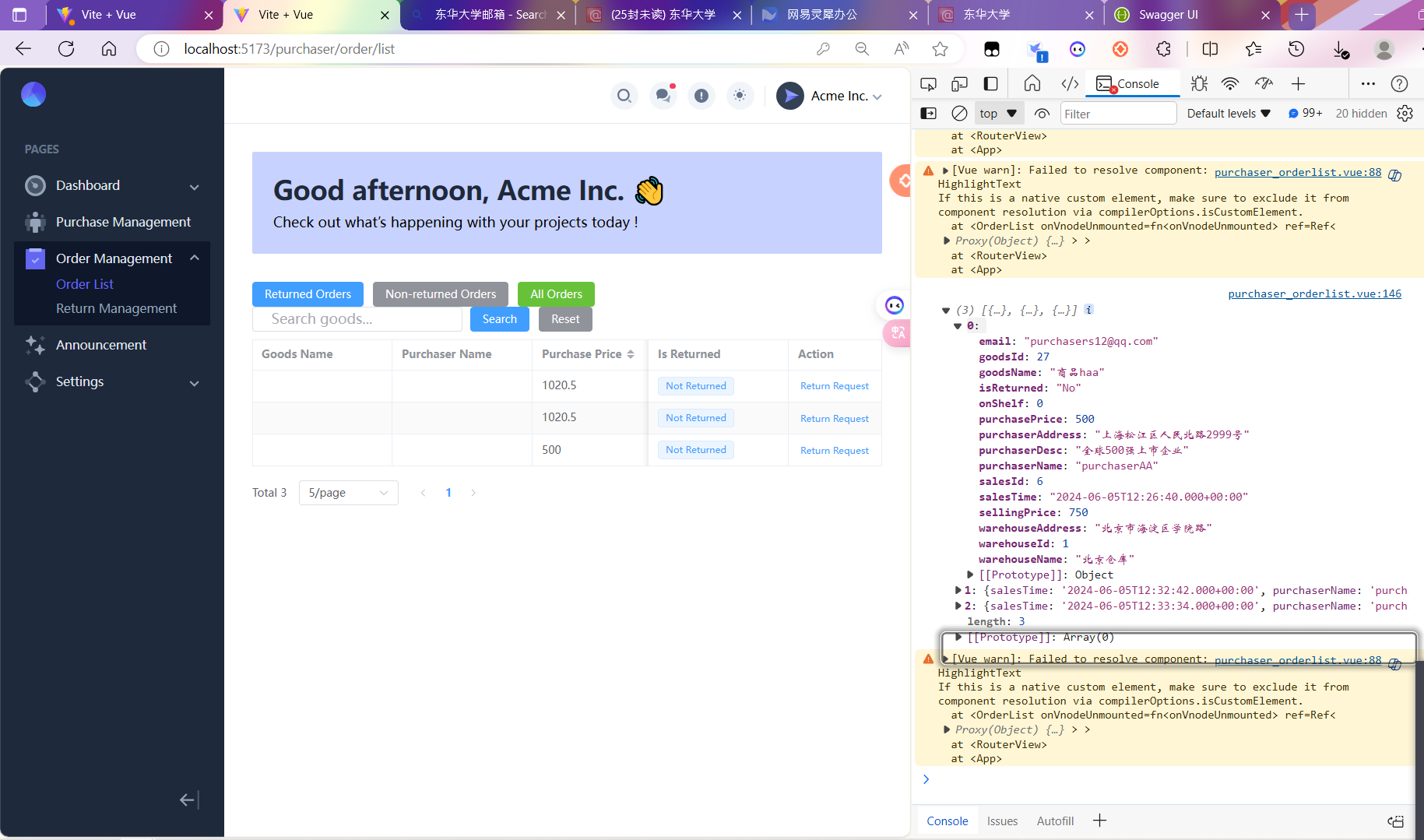


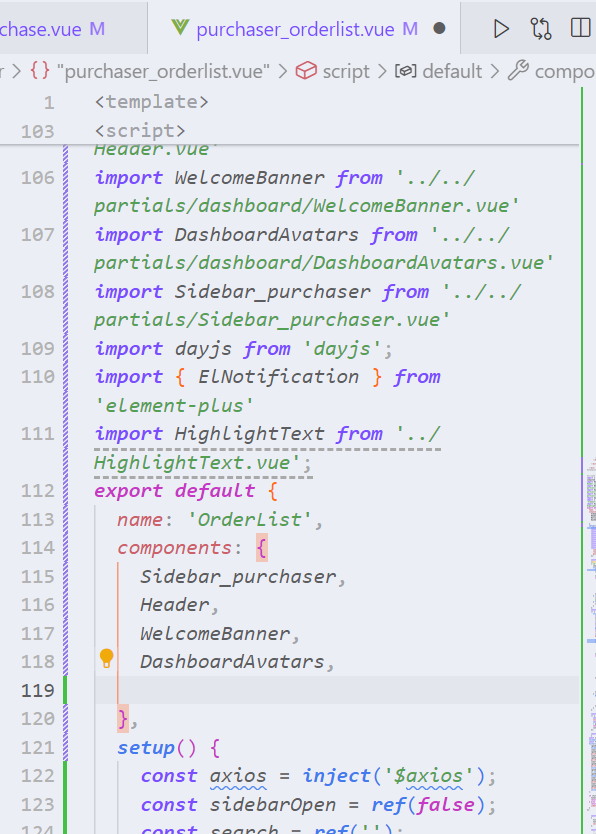
问题结论：这个问题是为了使用mybatis的usesGeneratedKeys，手写的插入，后续goods表修改属性时，忘记同步更改这个sql导致的。

实现这个页面时,出现了部分渲染成功,部分失效。

检查prop绑定名，无误后仍失效。

这个问题比较隐蔽，考虑到右侧提示component 没有解析到





在components处进行导入HighlightText,解决问题。之所以出现部分未显示，是因为相关的属性被HighlightText组件进行了搜索检索高亮绑定，而HighlightText为export 说明，导致渲染失效。