天津大学

设计模式实验报告

学　　院　智算学部

专　　业　软件工程

学　　号　3019213043

姓　　名　刘京宗

题目要求：分析“东软教育实战化教学项目资源包—饿了么外卖平台”（以下简称“饿了么项目”）中存在的问题，并给出改进建议。可以从以下几方面分析：

（1） 软件架构设计中存在的问题

（2） 接口设计中存在的问题

（3） 违反面向对象设计原则的问题

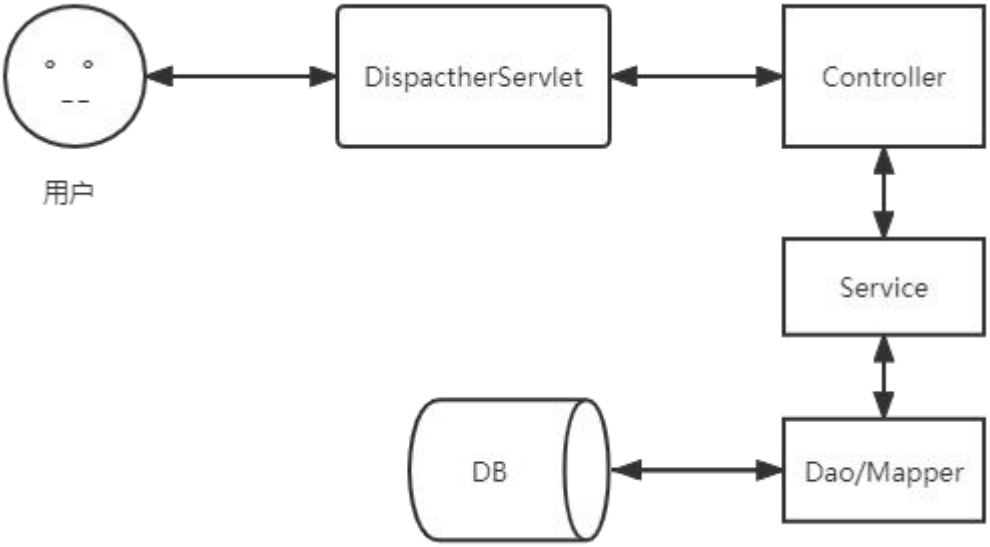
（4） 违反数据库设计原则的问题

（5） 违反用户使用的习惯和需求的问题

（6） 其它问题，如拼写错误、错别字等

分析内容包括“饿了么项目”从项目一到项目四的内容，重点分析项目三前端（VUE 技术）与项目四后端（Spring Boot 技术）结合中存在的问题，并给出相应的改进建议。

（1） 软件架构设计中存在的问题



采用 MVC 软件架构，这本身没有问题。但模块间通信时没有考虑到敏感数据，例如用户密码采用明文传输，明文存储，这显然是不安全的，可以考虑一些加密算法。且 Service 层的实现为贫血模型，这点在（3）中将较为详细论述。

（2） 接口设计中存在的问题

原项目中的所有接口的命名方式为动词+名词的方式，且全部为 post 请求。

应使用 RESTful 风格的接口设计。RESTfulAPI 中，URL 中只使用名词来指定资源，原则上不使用动词。多人协作时，统一风格，可提高沟通效率。具体如下：

1. GET（SELECT）：从服务器取出资源（一项或多项）；

2. POST（CREATE）：在服务器新建一个资源；
3. PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）；
4. DELETE（DELETE）：从服务器删除资源；

post 方法+FoodController/listFoodByBusinessId（**原项目接口**）
get 方法+Food/BusinessId（**RESTful 风格**）
显然 RESTful 风格的接口设计更为简洁。

（3） 违反面向对象设计原则的问题

Service 层的实现为贫血模型。以 User 为例，UserBo 是一个纯粹的数据结构，只包含数据，不包含任何业务逻辑。业务逻辑集中在 UserService 中。我们通过 UserService 来操作 UserBo。换句话说，Service 层的数据和业务逻辑，被分割为 BO 和 Service 两个类中。贫血模型是一种领域模型，其中领域对象包含很少或没有业务逻辑，它是一种面向过程的编程模式，它与面向对象设计的基本思想相悖。

应使用充血模型，Service 层包含 Service 类和 Domain 类两部分。Domain 就相当于贫血模型中的 BO。不过，Domain 与 BO 的区别在于它是基于充血模型开发的，既包含数据，也包含业务逻辑。而 Service 类变得非常单薄。

充血模型的数据和对应的业务逻辑被封装到同一个类中。因此，这种充血模型满足面向对象的封装特性，是典型的面向对象编程风格。对象自洽度很高，表达能力强，适合于复杂的企业业务逻辑实现，可复用程度高。

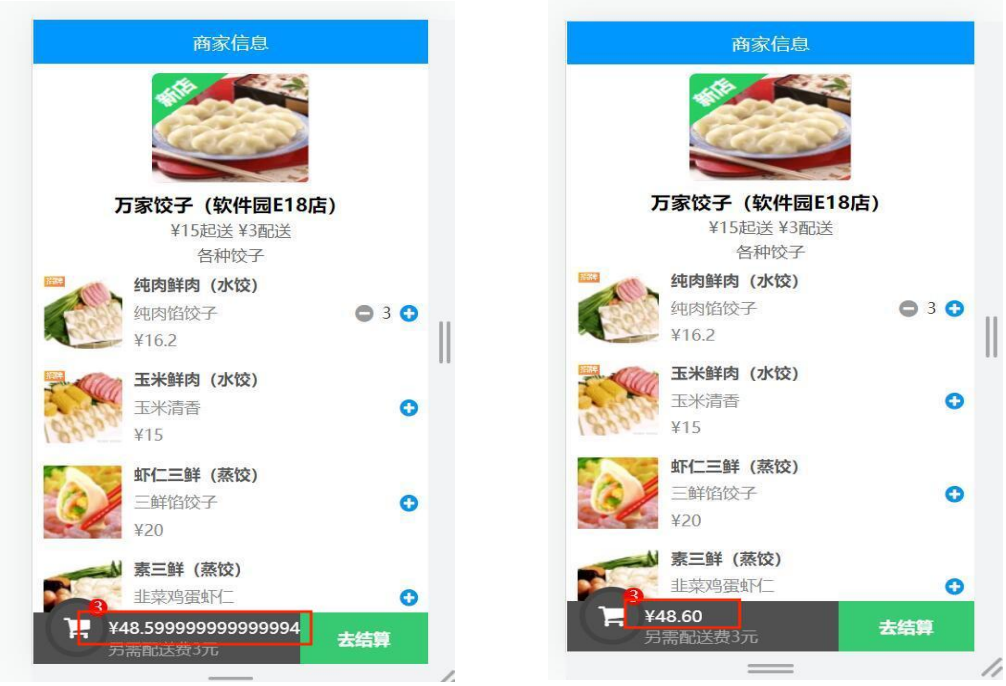
（4） 违反数据库设计原则的问题

daId	contactName	contactSex	contactTel	address	userId
1	那个谁	0	18523336666	天津大学北洋园校区格园一	18535806666

deliveryaddress 表中删除数据为直接删除，这就会导致其关联表 orders 在用户删除送货地址后，过往订单地址产生错误。应使用**逻辑删除**，即数据本身没有被删除，只是将 deleted 字段设置为 1。

（5） 违反用户使用的习惯和需求的问题

在订单页面，收餐人名称应与地址管理中的性名保持一致，而不是用户 ID，上图为修改后的正确示例。

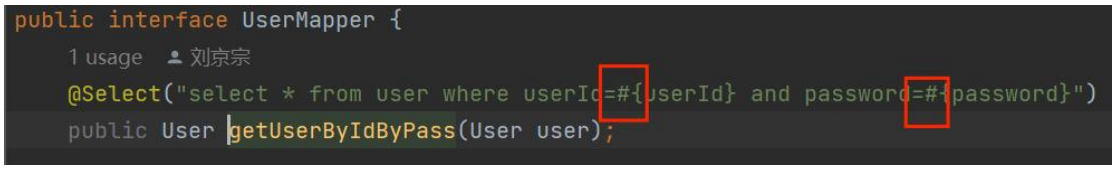


在付款页面，付款金额应保留小数点后两位。左图为原项目效果，右图为修改后效果。

（6） 其它问题，如拼写错误、错别字等

错别字问题：“登录”而不是“登陆”。

关于 sql 注入（“饿了么项目”中采用预编译的方式是正确的）：



在编写 mybatis 的映射语句时，若使用**“${xxx}”**这样的参数，sql 语句直接拼接，很容易被 sql 注入攻击。在项目中，我们采用**“#{xxx}”**这样的格式，采用预编译的方式可以防止 sql 注入。这一部分虽然没有问题，但在完成项目时没有留意到这一问题，在李老师课上提问后才回忆起来，故将这一部分补充到最后。