Online Bookstore

单元测试报告

版本1.2

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2024.4.20 | 1.0 | 完成了测试报告的大纲以及第1部分 | 安俊宇 |
| 2024.4.21 | 1.1 | 完成了第2部分和第3部分 | 安俊宇 |
| 2024.4.22 | 1.2 | 完成剩余部分 | 安俊宇 |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

1.6 测试环境 4

2. DD路径分析 5

2.1 Controller部分 5

2.2 Book&Cart部分 5

2.3 Order部分 6

2.4 User部分 7

2.5 其他文件 8

3. 数据流分析 9

3.1 Controller部分 9

3.2 Book & Cart部分 10

3.3 Order部分 10

3.4 User部分 11

3.5 其他文件 12

4. 测试结果分析 13

4.1 测试用例分析与补充 13

4.2 测试覆盖率分析 14

5. 建议措施 14

6. 测试结论 14

7. 贡献度 14

测试报告

# 简介

## 目的

本次测试针对Online Bookstore的后端代码采用白盒测试的方式进行了单元测试。本次测试的目的有以下几点：

·进一步确定并细化BookStore的功能模块，及对应软件构件的测试需求

·对后端代码进行全方位白盒测试，达到95%以上覆盖率

·进一步了解探索单元测试策略的原则，熟悉dd路径，数据流分析等内容，制定应对各种环境与要求的测试用例

·基于AI生成的测试用例，添加以手工测试用例添加，以完善白盒测试覆盖率

·为最终交付非Lab环境下可运行的BookStore提供测试支持

## 范围

本次需要测试的代码包括Online Bookstore后端项目中Controller、Dao（包含DaoImpl）、Entity、Service（包含ServiceImpl）等等文件夹中的所有代码。

本次测试中不包含Repository层代码的测试，Repository层是在应用程序中负责处理数据持久化和数据库操作的一层。它主要包含与数据存储相关的逻辑，提供了一种将应用程序中的数据对象持久化到数据库中的方式。我们使用了JpaRepository（是Spring Data JPA提供的一个接口，它扩展了JPA的标准接口）提供的一组基本的数据库操作方法，进行数据访问和操作。在使用junit5进行测试时，这部分默认是100%覆盖的。

## 定义、首字母缩写词和缩略语

暂无

## 参考资料

暂无

## 概述

该测试报告将重点展示被测代码的DD路径分析与数据流分析、本次测试的结果分析，测试代码详见文件夹。其中，DD路径与数据流分析将按以下几个部分展开：

·Controller：包含所有的Controller文件

·Book&Cart：包含Dao、Entity、Service等文件夹中涉及到Book和Cart的文件

·Order：包含Dao、Entity、Service等文件夹中涉及到Order的文件

·User：包含Dao、Entity、Service等文件夹中涉及到User的文件

·其他文件：比如interceptor、utils等文件夹中的文件

## 测试环境

·测试框架：Junit5

·Mock组件：Mockito 3.4.0 (版本不能过低，否则无法mock static method，也不能过高，因为此框架backward compatibility很差)

·语言：Java 11 （Mockito 3.4.0所需的最低Java版本）

# DD路径分析

# Controller部分

该部分共包括以下7个文件：

·BookController.java

·CartController.java

·HomeController.java

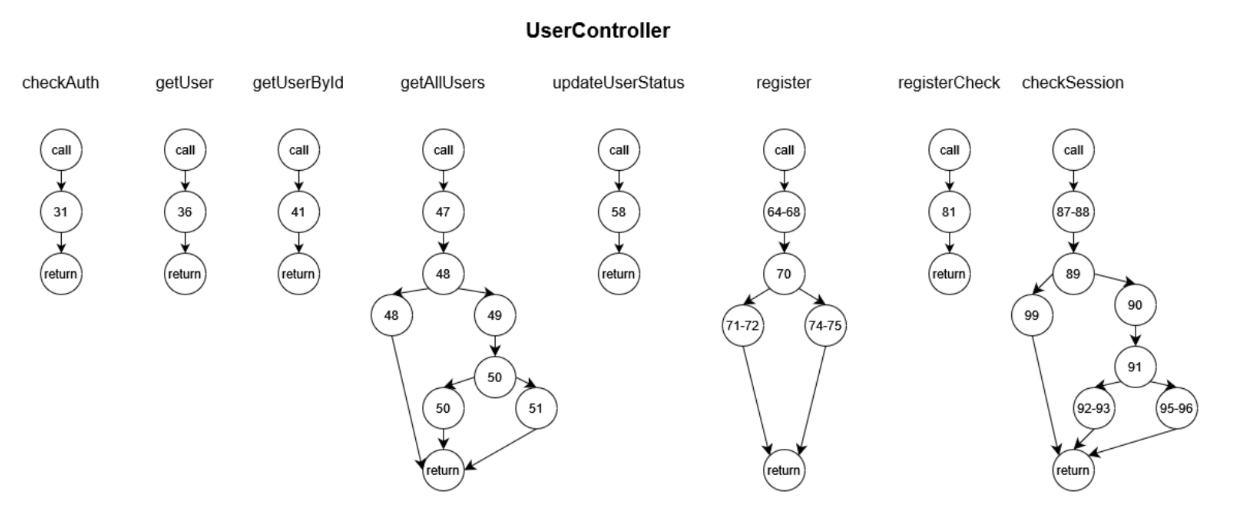
·LoginController.java

·OrderController.java

·OrderItemController.java

·UserController.java

本报告选取最具代表性的UserController进行展示（如下图），其他文件的DD路径图详见文件：Controller部分测试报告.pdf。



# Book&Cart部分

该部分共包括以下文件：

·BookDao.java（包含BookDaoImpl.java）

·CartDao.java（包含CartDaoImpl.java）

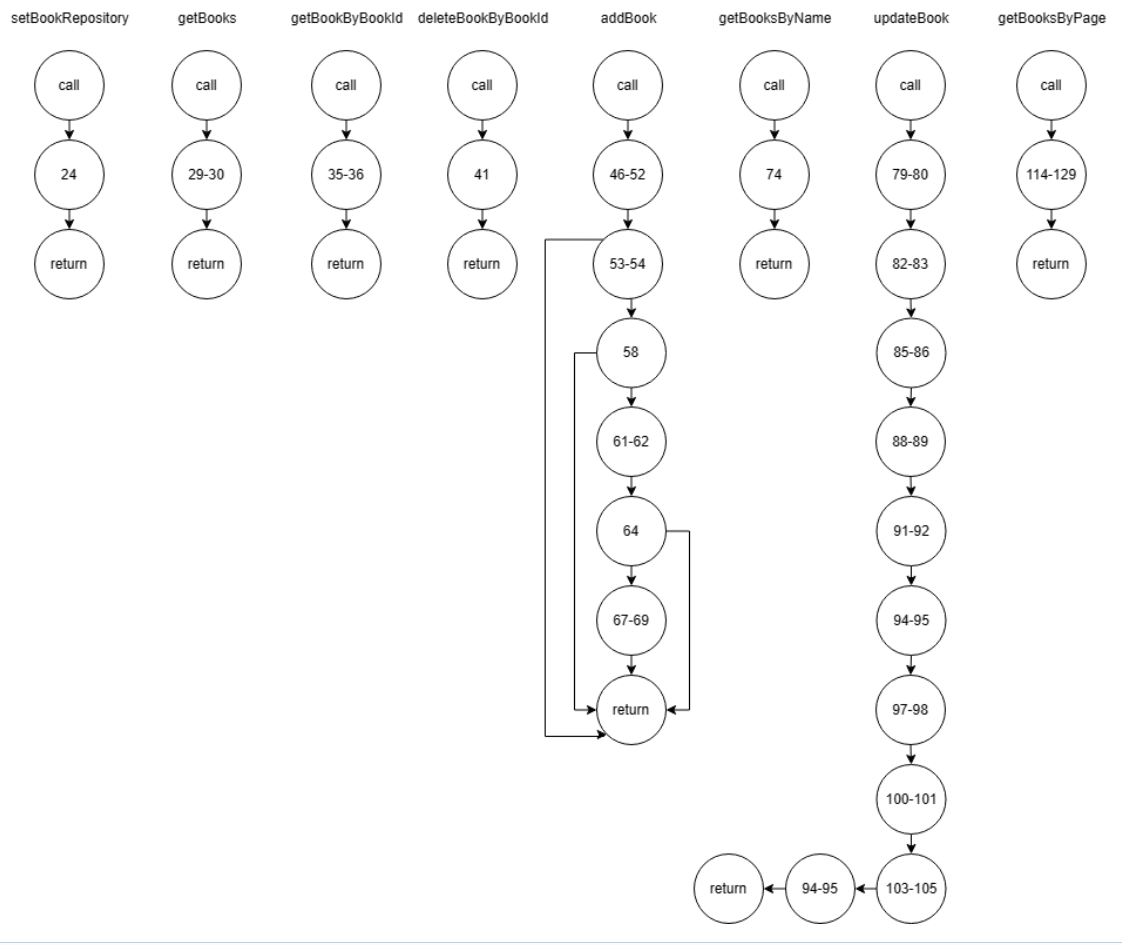
·Book.java

·CartItem.java

·BookService.java（包含BookServiceImpl.java）

·CartService.java（包含CartServiceImpl.java）

本报告选取BookDaoImpl进行展示（如下图），其他信息详见文件：book和cart部分测试报告.pdf。



# Order部分

该部分包括以下文件：

·OrderDao.java（包含OrderDao.java）

·OrderItem.java（包含OrderItem.java）

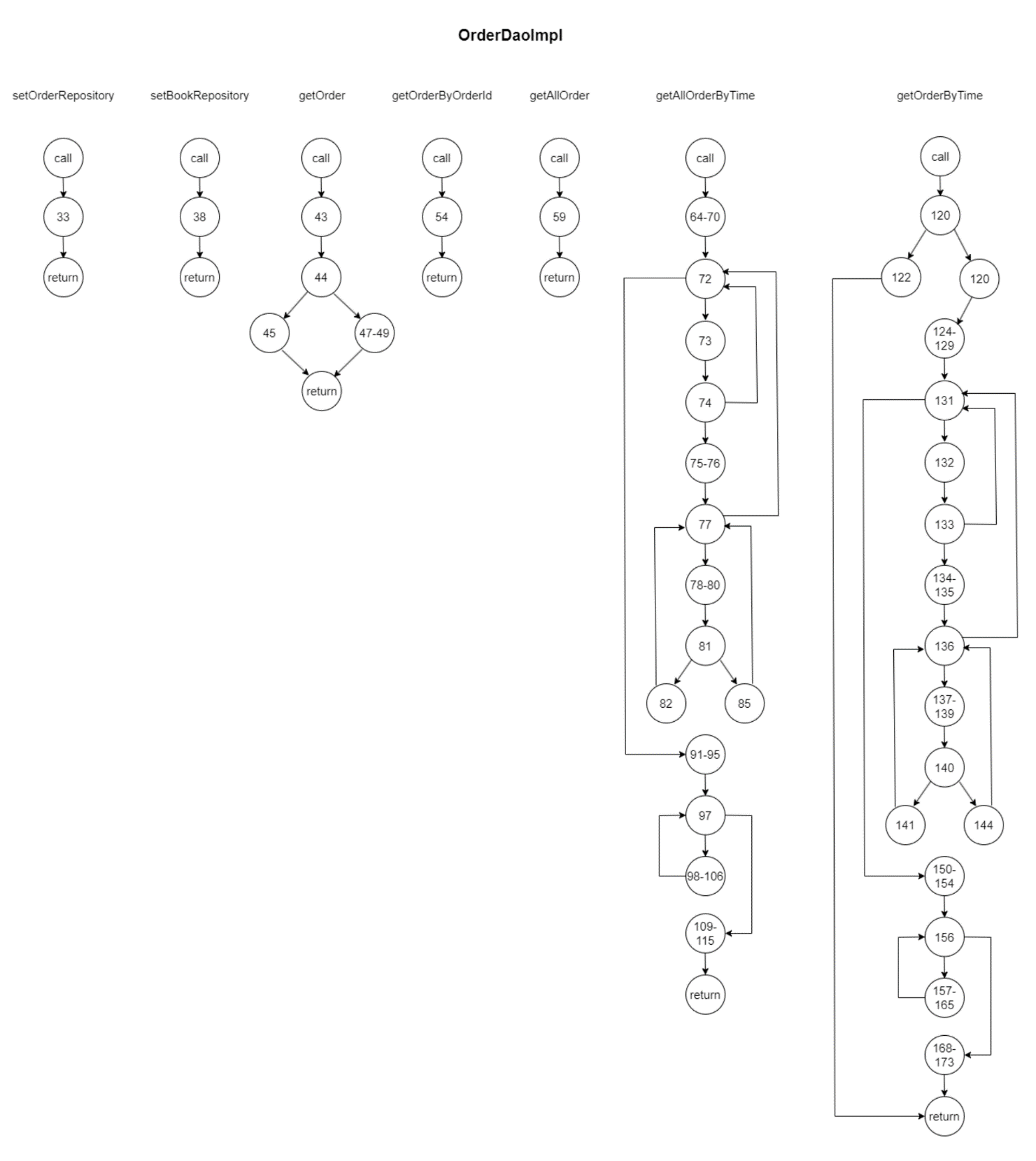
·Order.java

·OrderItem.java

·OrderService.java（包含OrderService.java）

·OrderItemService.java（包含OrderItemService.java）

本报告中选取OrderDao进行展示（如下图），其他文件的DD路径分析详见文件：Order和OrderItem部分测试报告.pdf。



# User部分

该部分包括以下文件：

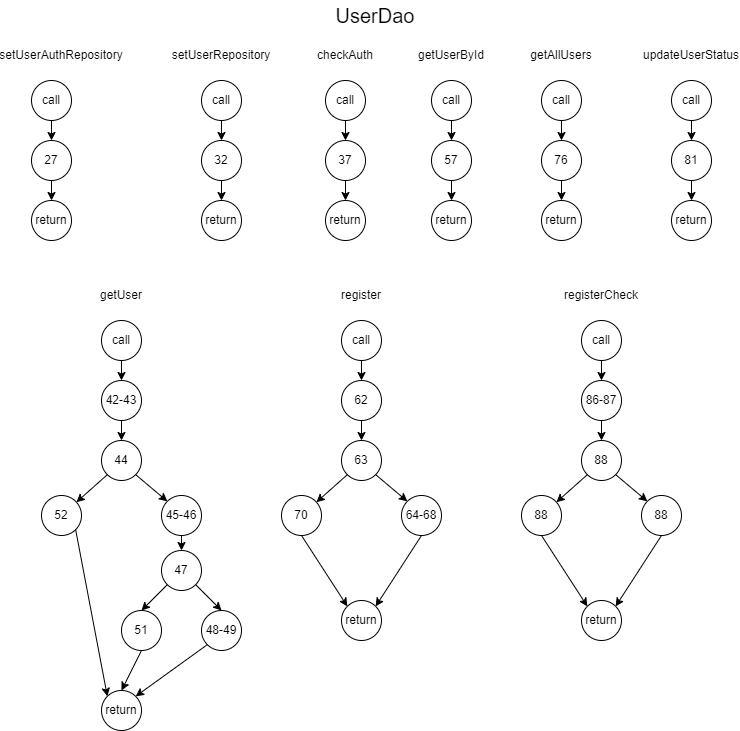
·UserDao.java（包含UserDaoImpl.java）

·User.java

·UserAuth.java

·UserService.java（包含UserServiceImpl.java）

本报告选取UserDao的DD路径图进行展示（如下图），其他文件的DD路径图详见文件：User部分测试报告.pdf。



# 其他文件

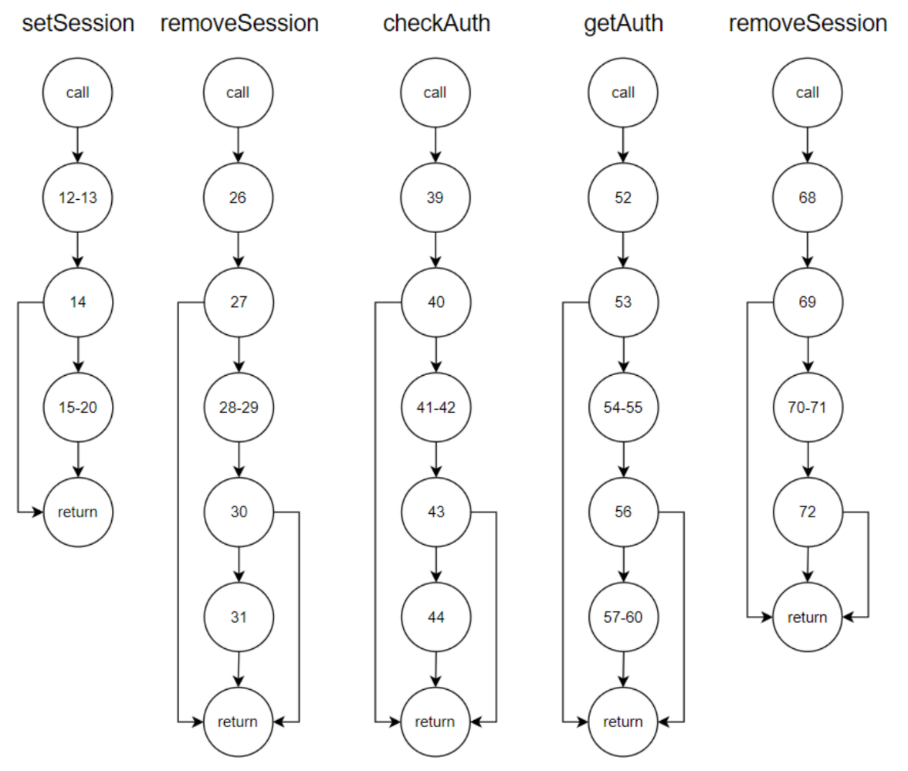
该部分包含其他杂项文件，如下表：

·SessionValidateInterceptor.java

·MessageUtil.java

·SessionUtil.java

本报告选取SessionUtil的DD路径图进行展示（如下图），其他文件的DD路径图详见文件：Interceptor和utils部分测试报告.pdf。



# 数据流分析

# Controller部分

该部分包含的文件与上一节对应部分相同。本报告选取BookController的数据流分析表格进行展示（如下图），其他部分详见文件：Controller部分测试报告.pdf。



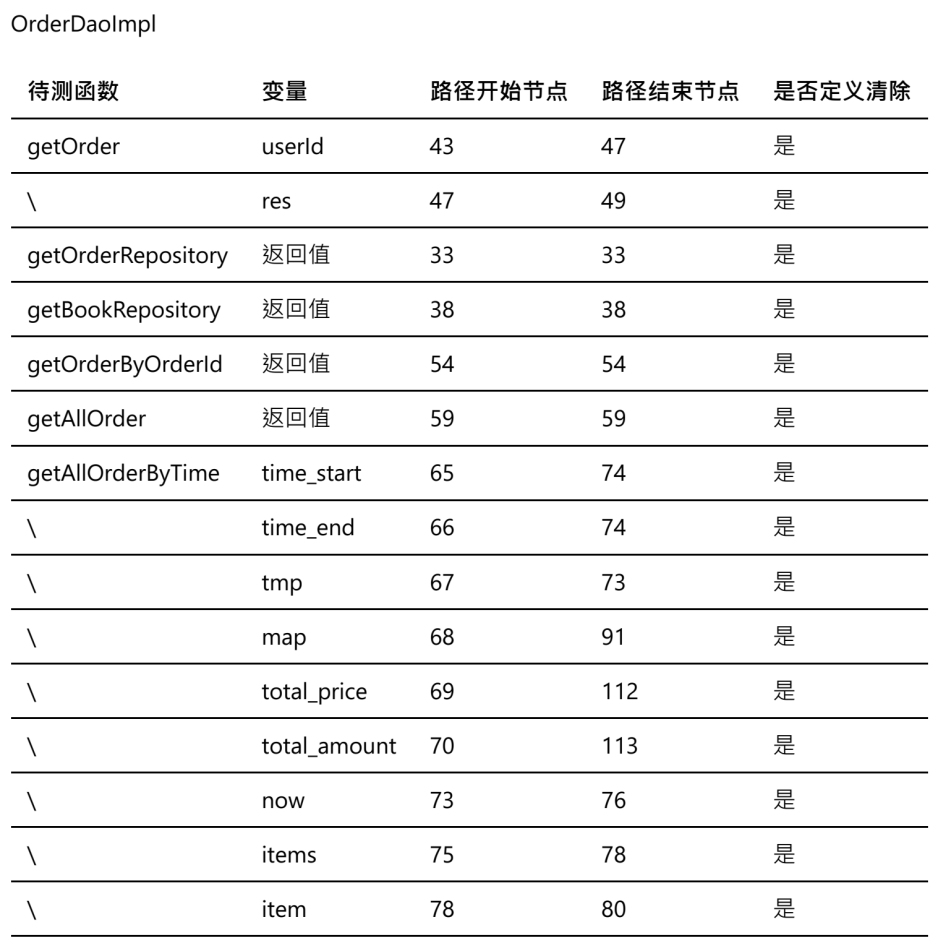
# Book&Cart部分

该部分包含的文件与上一节对应部分相同。本报告选取比较复杂的updatebook方法进行数据流分析（如下图），其他部分详见文件：book和cart部分测试报告.pdf。



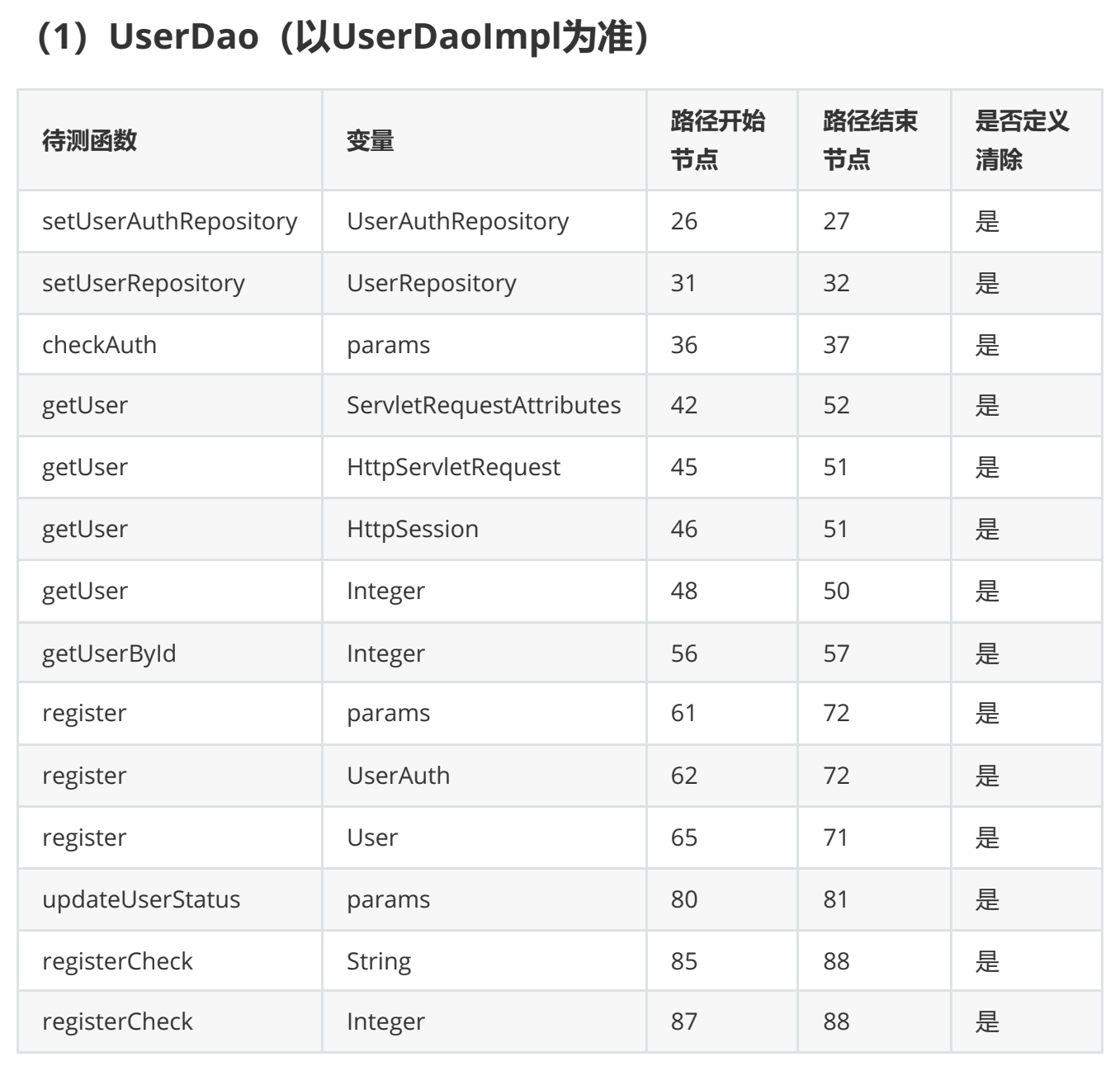
# Order部分

该部分包含的文件与上一节对应部分相同。本报告选取OrderDao中的部分方法进行数据流分析（如下图），其他部分详见文件：Order和OrderItem部分测试报告.pdf。



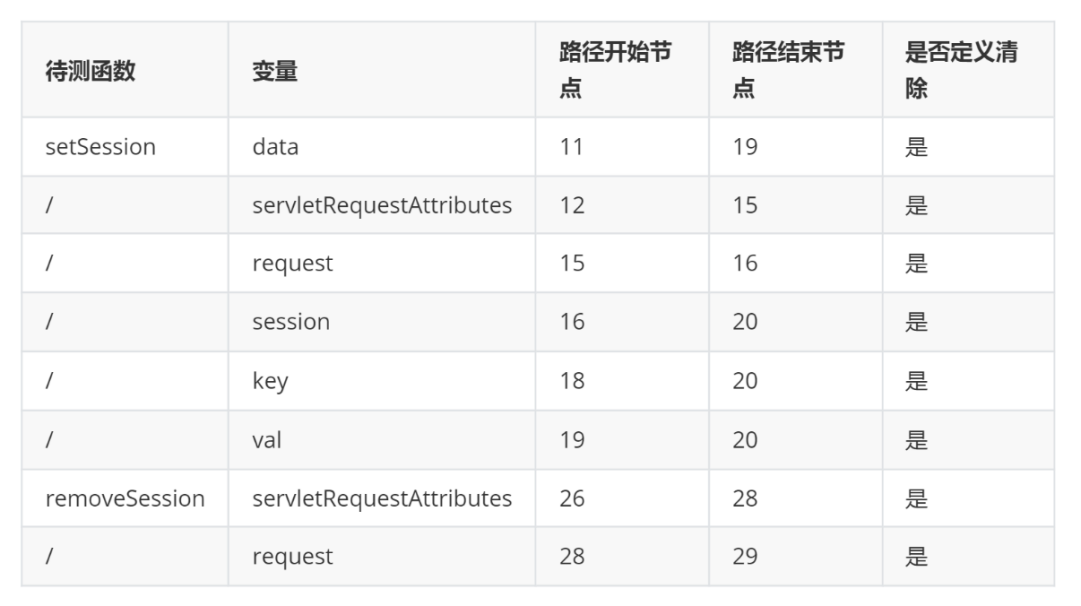
# User部分

该部分包含的文件与上一节对应部分相同。本报告选取UserDao中的部分方法进行数据流分析（如下图），其他部分详见文件：User部分测试报告.pdf。



# 其他文件

该部分包含的文件与上一节对应部分相同。本报告选取SessionUtil中的部分方法和参数进行数据流分析（如下图），详见其他文件：Interceptor和utils部分测试报告.pdf。



# 测试结果分析

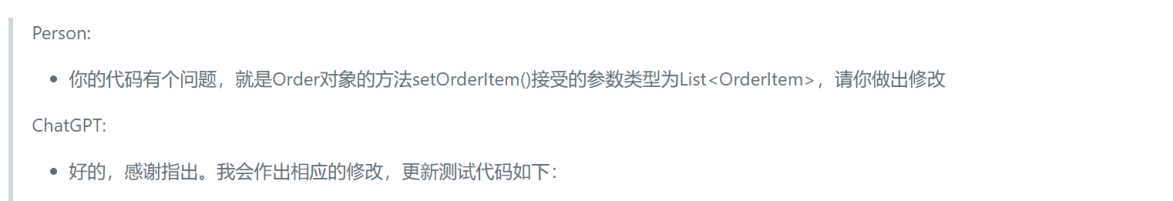
# 测试用例分析与补充

本报告以OrderDaoImpl.java中的getAllOrderByTime方法为例，对GPT生成的对话记录进行分析。由于篇幅问题，本报告不会呈现代码部分，具体信息以及其他部分的GPT对话记录详见【AI\_Export】文件夹。

首先向GPT描述需要测试的函数以及进行的一些初始化：



初版测试代码已经较为完善，但还无法直接运行。第一个问题是代码中对bookRepository.getBookByBookId(int)打桩时，返回的是一个在构造时传入参数的Book对象，而这种构造方法在原项目中并未实现，因此需要将这部分修改为先创建空的Book对象，再设置其具体参数。第二个问题是Order.setOrderItem()方法接受的参数类型为List<OrderItem>，而非代码中的OrderItem。因此将这个问题告诉GPT，它给出了修改后的测试代码。



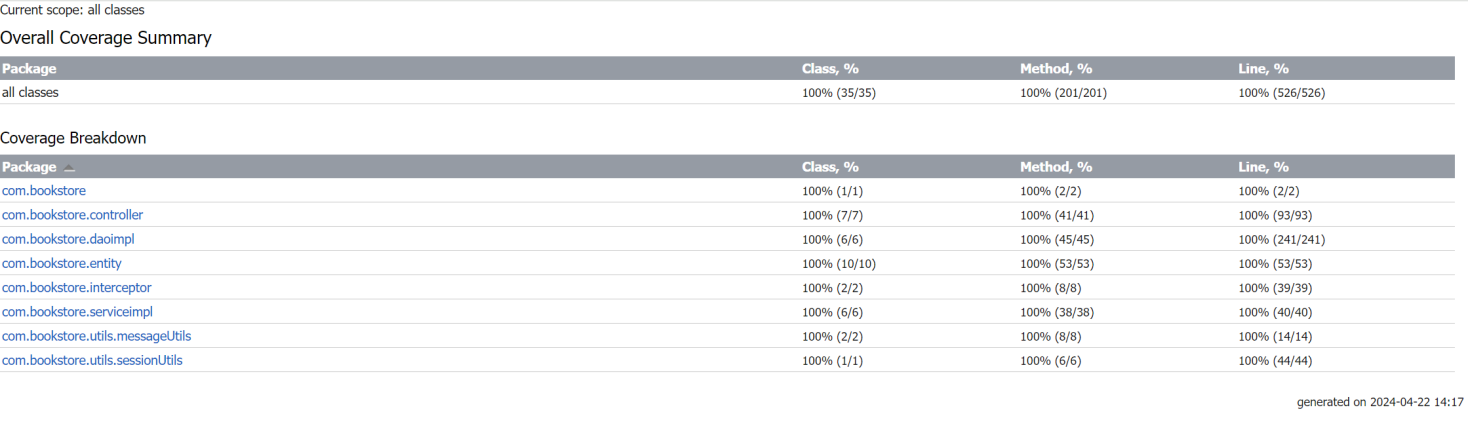
再次运行修改后的测试代码，成功完成了运行，但是发现测试并未通过。原因为对待测函数返回值中元素个数的断言失败，待测函数最终返回的List<OrderStat>实际包含了各个OrderItem以及总和的数量与价格，因此其中的元素个数应为n+1，在本例中即为2+1=3。将此处断言进行修改，再次运行测试代码，发现成功通过了测试，但是有一个条件分支里的语句没有被覆盖。这个条件分支在检查到重复的bookId时才会跳转，因此将该信息告知GPT，它做出相应修改以满足该条件。



GPT修改完后的最终测试代码成功通过了测试。

# 测试覆盖率分析

整个后端项目的测试覆盖率已达到100%（如下图）。详见：html\_report/index.html。



# 建议措施

本次测试的测试用例借助了ChatGPT，取得了良好的效果。将测试环境和配置等等信息提供给ChatGPT后，再提供被测代码，ChatGPT能快速生成测试用例和代码，大大减少了单元测试中的重复劳动，极大提高了测试效率，解放了人力。诚然，如本报告4.1部分中提到，ChatGPT对于较为复杂的代码生成的测试代码还有瑕疵，还需要人工的修正和补充。所以，对于单元测试这类工程量大、重复劳动多的任务，人工与ChatGPT结合的方式仍是目前的最优解。

# 测试结论

本次单元测试，采用了白盒测试的测试方法，利用人工与ChatGPT结合的方式，对Online Bookstore的后端代码部分进行了测试。测试过程中，对每个文件中的关键代码都进行了DD路径分析与数据流分析。同时，测试代码的测试覆盖率达到了100%，并未发现bug。

综上所述，Online Bookstore项目的后端代码通过本次单元测试。

# 贡献度

本次作业各组员贡献度均为25%。