# 网上书店系统实验报告

## 一、个人信息及学习体会；

#### 1.个人信息：

#### 2.学习体会：

通过学习JavaWEB，我收获了许多宝贵的经验和技能。首先，我了解了如何使用JavaEE和Spring框架来快速构建高效的Web应用程序。同时，学习JavaWEB还让我更好地理解了Web开发中的许多基本概念和技术，如HTTP协议、Servlet、JSP、AJAX等等。此外，我也意识到了软件设计和架构的重要性，以及如何避免常见的开发错误和漏洞。这些知识和技能不仅对我的个人成长和职业发展有很大帮助，也能够将我所学的运用到实际项目中，提高工作效率和质量。

在学习Spring的过程中，我深刻认识到了它的活性和高效性。通过对Spring框架的掌握，我能够更加有效地进行Web应用的开发和管理，同时也在代码重构、配置管理、AOP等面获得了很大的提升。此外，Spring框架提供的依赖注入和控制转机制，使得代码开发和管理更加便捷，也极大地减少了代码的重复性和维护成本。但是Spring的学习成本较高，本项目开发对Spring框架的应用就不太理想。

相比较来说，在普通javaee模式开发中，利用servlet技术和HTML或jsp网页编程技术，能够清楚明了的进行编程。采用mvc的设计模式，无论是用cookie还是session进行用户是否已经登录的判断，还是使用jdbc进行数据库与网页交互，都能直白易懂的明白每一步需要做什么，怎么做，唯一的缺点就是过于繁琐。然而，在使用Spring boot框架进行编程时，大量的细节被隐藏起来，导致开发人员往往不知道如何去编程，需要查阅大量的说明，开发成本降低，但是学习成本大大增加。

总之，辅修课程到此结束，javaee程序设计几乎可以说是之前所学之集大成者，带给我的收获有很多，希望日后在从事相关工作时能够更快捷方便入手。

## 二、系统架构及功能模块

#### 1.系统架构：

**使用软件：**

Spring Tool Suite 3.9.4.RELEASE，MySQL 8.0，Microsoft Frontpage。

**运行环境：**

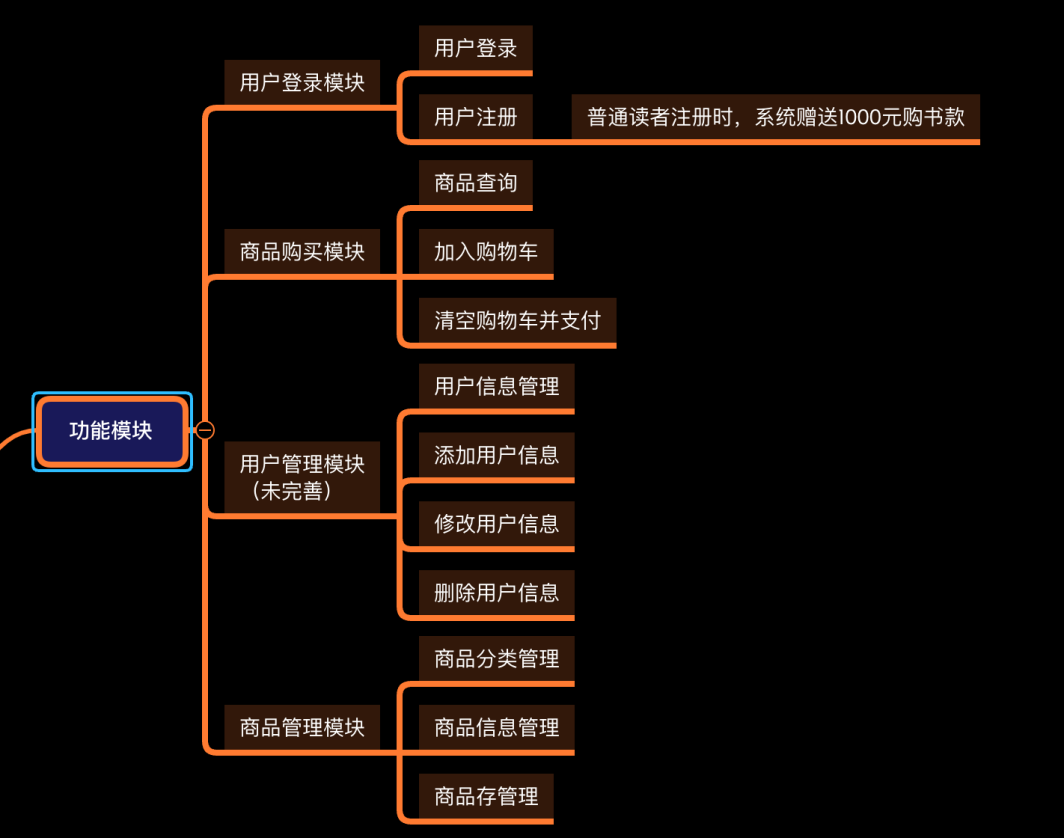
Windows 11系统，javaSE 1.8.1，java 8版本。

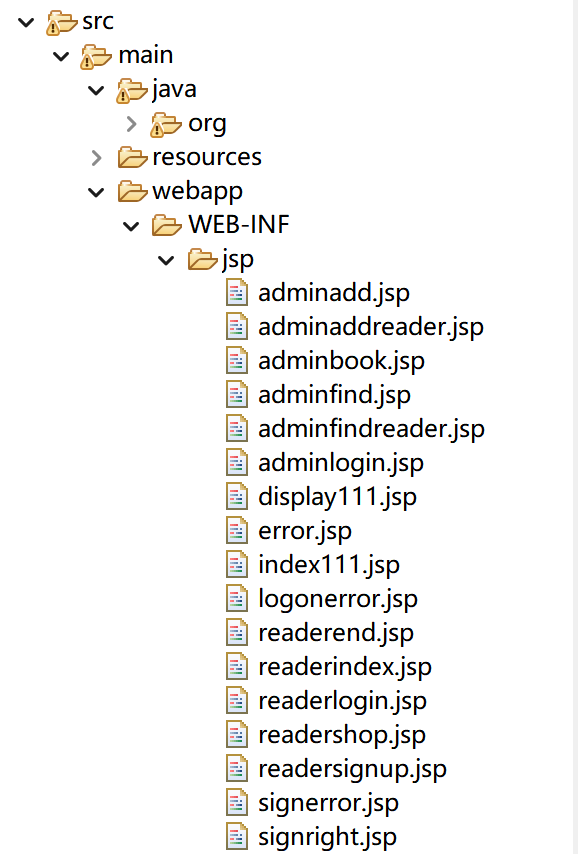
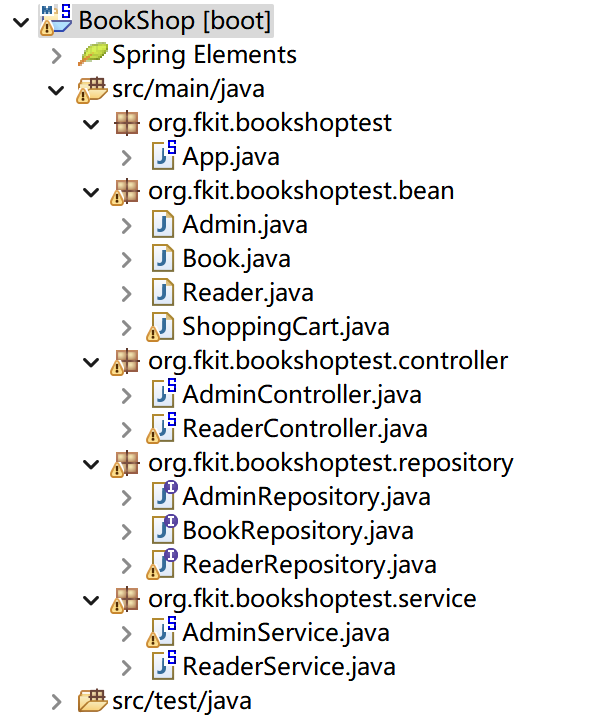
**具体架构：**

该网上书店系统的架构采用了Spring Boot框架作为核心，使用jsp网页编辑语言，集成servlet与MyBatis、Hibernate和Thymeleaf等开源项目，利用jdbc技术与数据库连接交互，利用Maven自动加载第三方库，最终项目为jar包形式，解压缩即可运行。其主要任务是简化Spring应用程序搭建和开发过程。该架构采用了微服务的设计理念，不同的业务模块独立部署，实现了高内聚低耦合的业务拆分，提高了系统的可扩展性和可维护性。在系统架构中，用户可以通过Web客户端进行访问，而后端则包括订单处理模块、图书管理模块等。

#### 功能模块：

网上书店利用Spring Boot框架开发，功能主要包括图书分类、图书搜索、购物车管理、订单管理、用户登录。系统提供清晰易懂的图书分类和搜索界面，用户可以通过关键字或者分类进行查询，并且在系统中览书籍的详细信息。用户购物车功能方便快捷，用户可以将心仪的商品加入购物车进行管理，同时，用户也够进行订单管理、支付等操作，保证用户使用系统的良好体验。





## 三、自己有特色的部分介绍

#### 1.注册账号

注册账号需要介入数据库进行插值，通过框架实现对数据库的接入，免去每次sql操作都需要重写相关配置和异常抛出等工作，只需载入sql语句读取执行即可。同时，确保注册值非空，进行了空值校验。若账号和密码至少存在一个空白提交，网页端弹出提示框并拒绝下一步访问。只有都进行输入有效文本后，才判定注册成功。

控制层代码：  
@RequestMapping("/add")

**public**  String add(HttpSession session) {

**if**(session.getAttribute("admin")==**null**)

**return** "adminlogin";

**else**

**return** "adminadd";

}

@RequestMapping("/adddb")

**public**  String addDB(Book book,HttpSession session) {

**if**(session.getAttribute("admin")==**null**)

**return** "adminlogin";

**if**(adminService.insertBook(book)>0)

**return** "adminbook";

**else**

**return** "error";

}

@RequestMapping("/signdb")

**public**  String signDB( Reader reader,HttpSession session) {

//if( StringUtils.hasText(reader.getName()) && StringUtils.hasText(reader.getPassword()))

//return "signerror";

**if**( StringUtils.*isEmpty*(reader.getName()) || StringUtils.*isEmpty*(reader.getPassword()))

**return** "signerror";

//StringUtils.isEmpty(null)

//public String addStudent(@RequestBody Student stu){

//if(reader.getName()==null || reader.getPassword()==null)

// return "signerror";

//String name = RequestPath.getParameter("name");

//String password = Request.getParameter("password");

**if**(readerService.insertReader(reader)>0)

**return** "signright";

**else**

**return** "signerror";

}

除此之外，读者登录界面和注册界面可以互相跳转，方便查找。

#### 2.登录校验

登录验证是指在用户登录系统时对其身份进行确认和验证的过程，确保只有合法用户才能访问系统资源和进行相关操作。通过引入Spring Security框架，我们可以很方便地实现登录验证，并实现与Spring Boot框架的无缝整合，确保系统具有高可靠性和安全性。用户在登录时输入用户名和密码，系统将验证用户信息的正确性，在通过验证后，系统才会允许用户进入系统并进行相关操作。

**服务层代码：  
public** **boolean** validate(Reader reader) {

Reader temp;

temp=readerRepository.selectByName(reader.getName());

**if**(temp==**null**) **return** **false**;

**if**(temp.getPassword().equals(reader.getPassword()))

{

reader.setId(temp.getId());

reader.setMoney(temp.getMoney());

**return** **true**;

}

**else** **return** **false**;

}

具体代码实现为利用validate这一布尔类实现校验。将输入名字和密码存储在中间变量，同时调取数据模型层的仓库操作，查询是否有相同名字和密码的记录，如果两者完全相同，则校验成功，返回true值，否则返回false值。在controller控制层，仅需利用布尔值进行相关操作和网页跳转即可。

#### 访问容错

凡需要登录才能访问的网页，如果在未经登录情况下直接访问，系统通过Spring框架中集成的session和servlet相关服务自动判定登录情况，未登录者直接跳转到登录界面进行登录，已经登录的则正常进行下一步访问。

**if**(session.getAttribute("reader")==**null**) **return** "readerlogin";

对购物车支付时，若1000元余额不足支付当前购物车内总价值，则系统支付失败，返回购物车添加页面。若库存图书数量少于购物车内添加图书数量，则系统支付失败，返回购物车添加界面。

控制层代码：  
@RequestMapping("/pay")

**public**  String pay(HttpSession session,Model model) {

**if**(session.getAttribute("reader")==**null**)

**return** "readerlogin";

ShoppingCart sc=**null**;

sc=(ShoppingCart)session.getAttribute("shopcart");

**if**(sc==**null**) **return** "error";

Reader reader=(Reader)session.getAttribute("reader");

**double** money=readerService.pay(reader, sc);

**if** (money<0)  **return** "error";

**else**

model.addAttribute("money",money);

//double quantity=readerService.pay(reader, sc);

//model.addAttribute("quantity",quantity);

//if (money>reader.getMoney()) return "余额不足";

session.removeAttribute("shopcart");

**return** "readerend";

}

服务层代码：  
@Transactional

**public** **double** pay(Reader reader,ShoppingCart sc) {

**double** money=reader.getMoney()-sc.getTotal();

reader.setMoney(money);

**if**(money<0)  **return** money;

readerRepository.update(reader);

//if <0?

List<Book> lb=sc.getCart();

**for**(Book book:lb) {

Book temp=bookRepository.findBookById(book.getId());

temp.setQuantity(temp.getQuantity()-book.getQuantity());

bookRepository.update(temp);

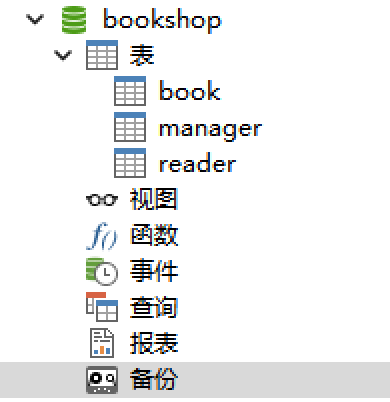
}

**return** money;

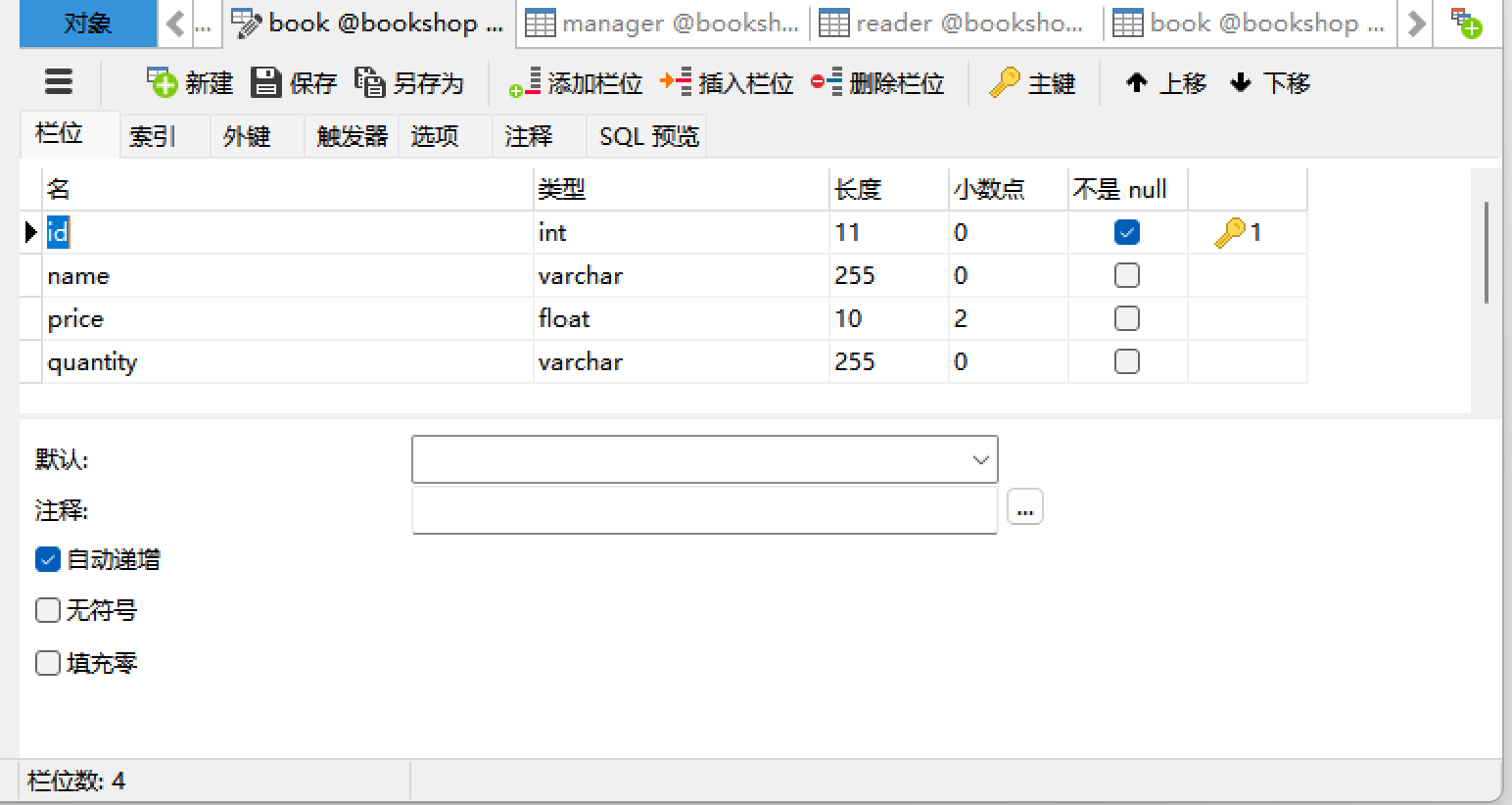
}

## 数据库表的设计

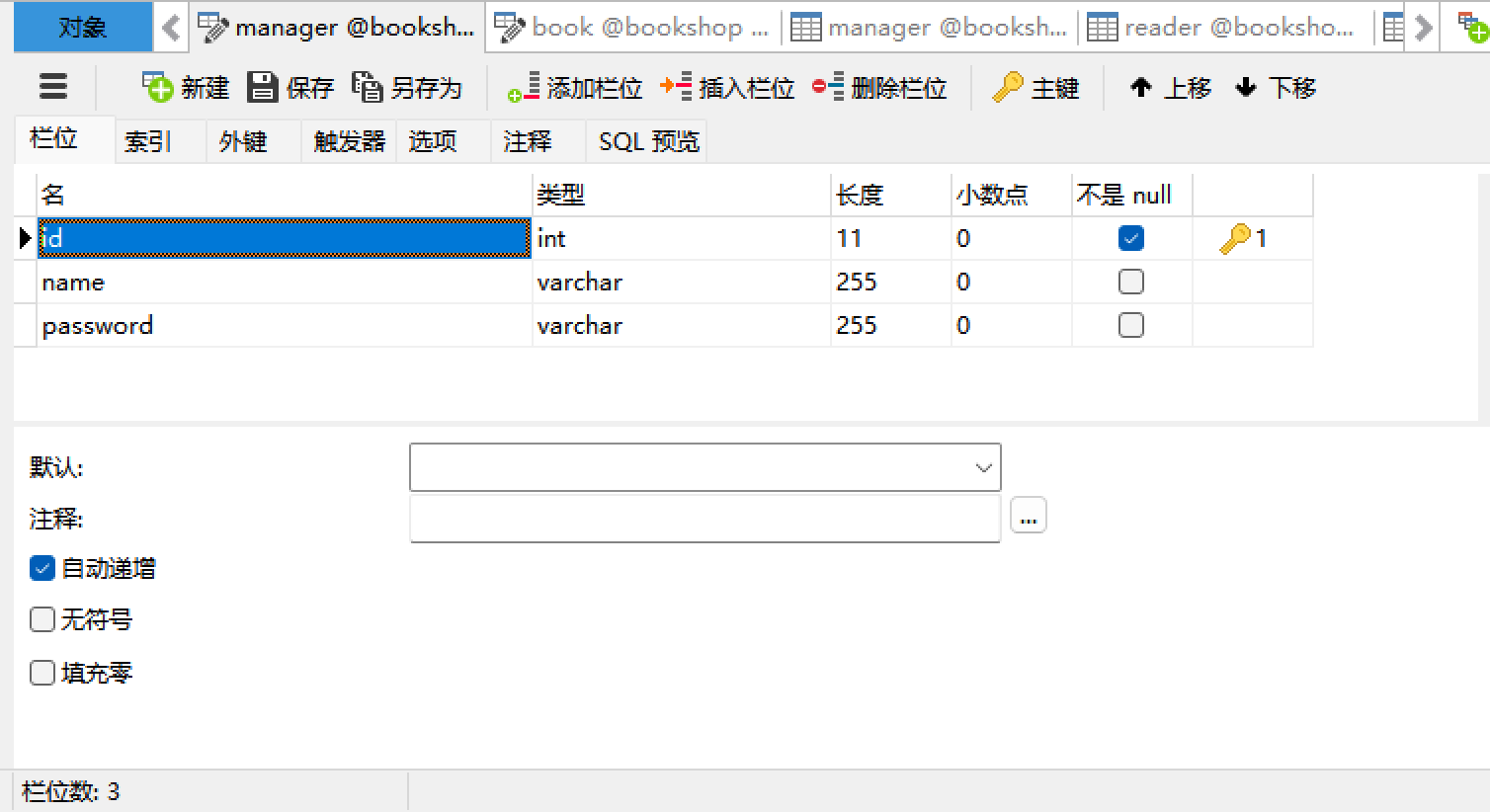
数据库表设计为三张单表，分别为book图书表，manager管理员表，reader读者表。



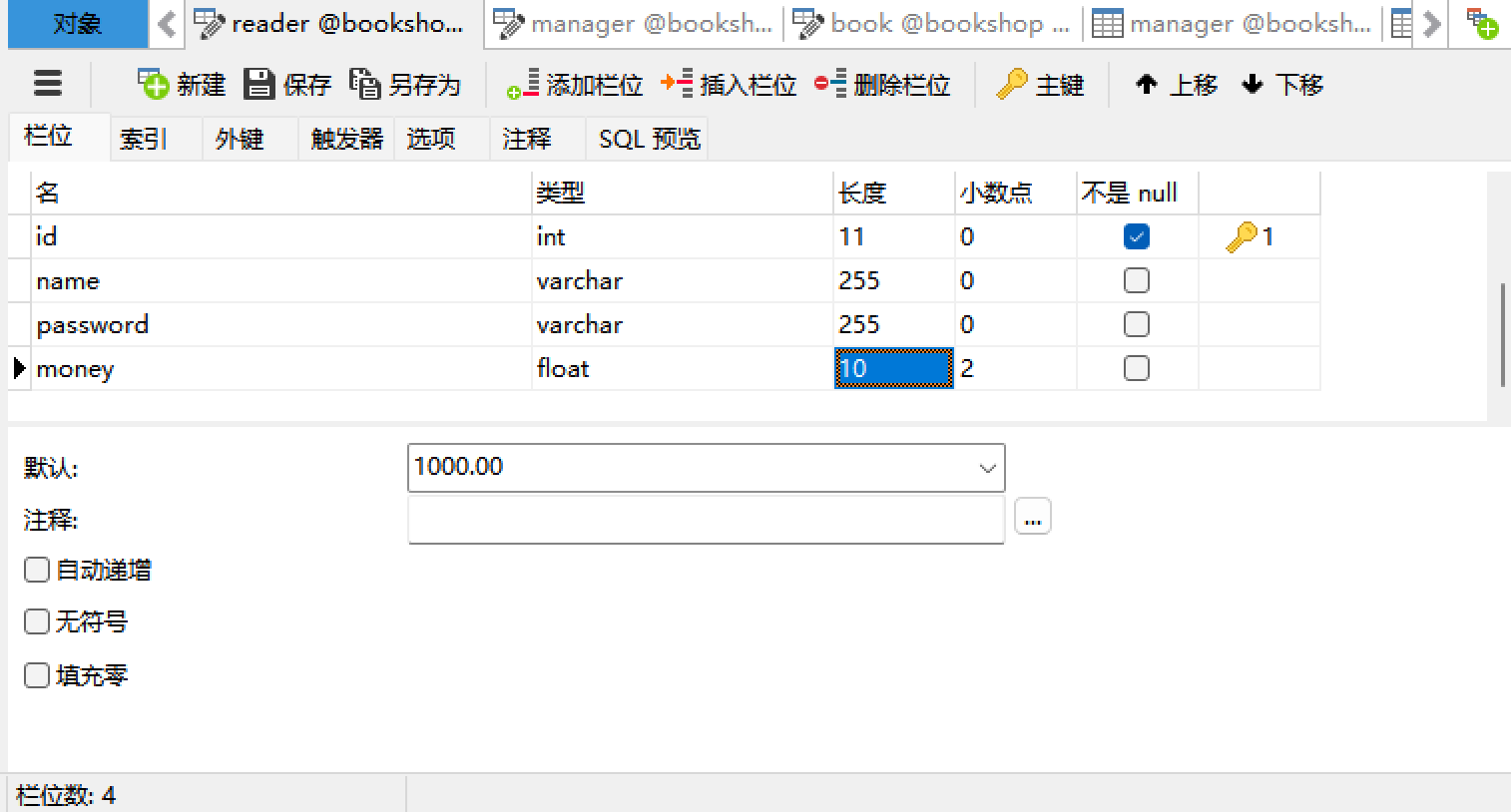
图书表具体设计与实现如下图所示。



管理员表具体设计与实现如下图所示。



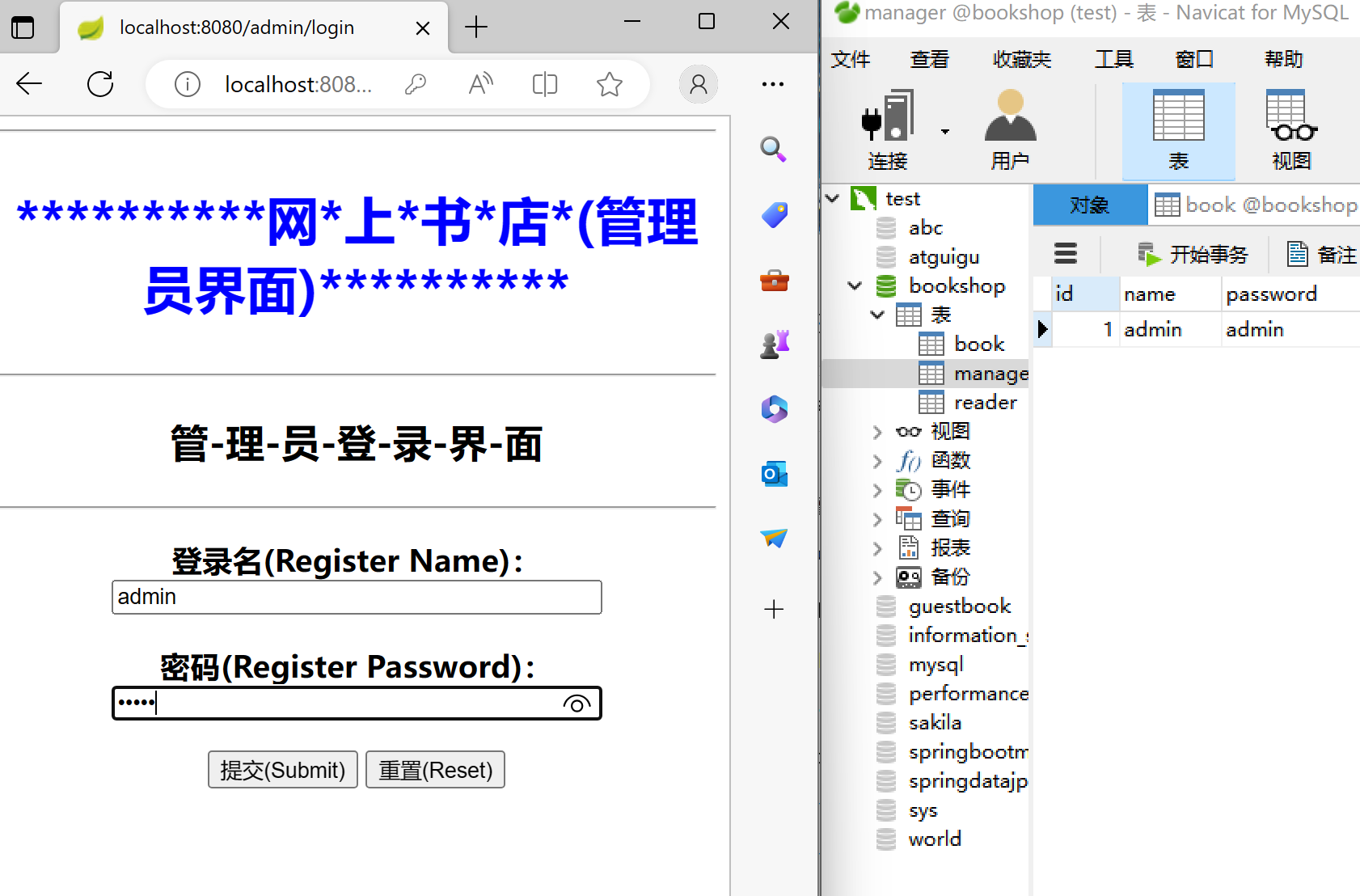
读者表具体设计与实现如下图所示。其中余额部分默认设置为1000.00，对应新注册用户赠送的1000元。



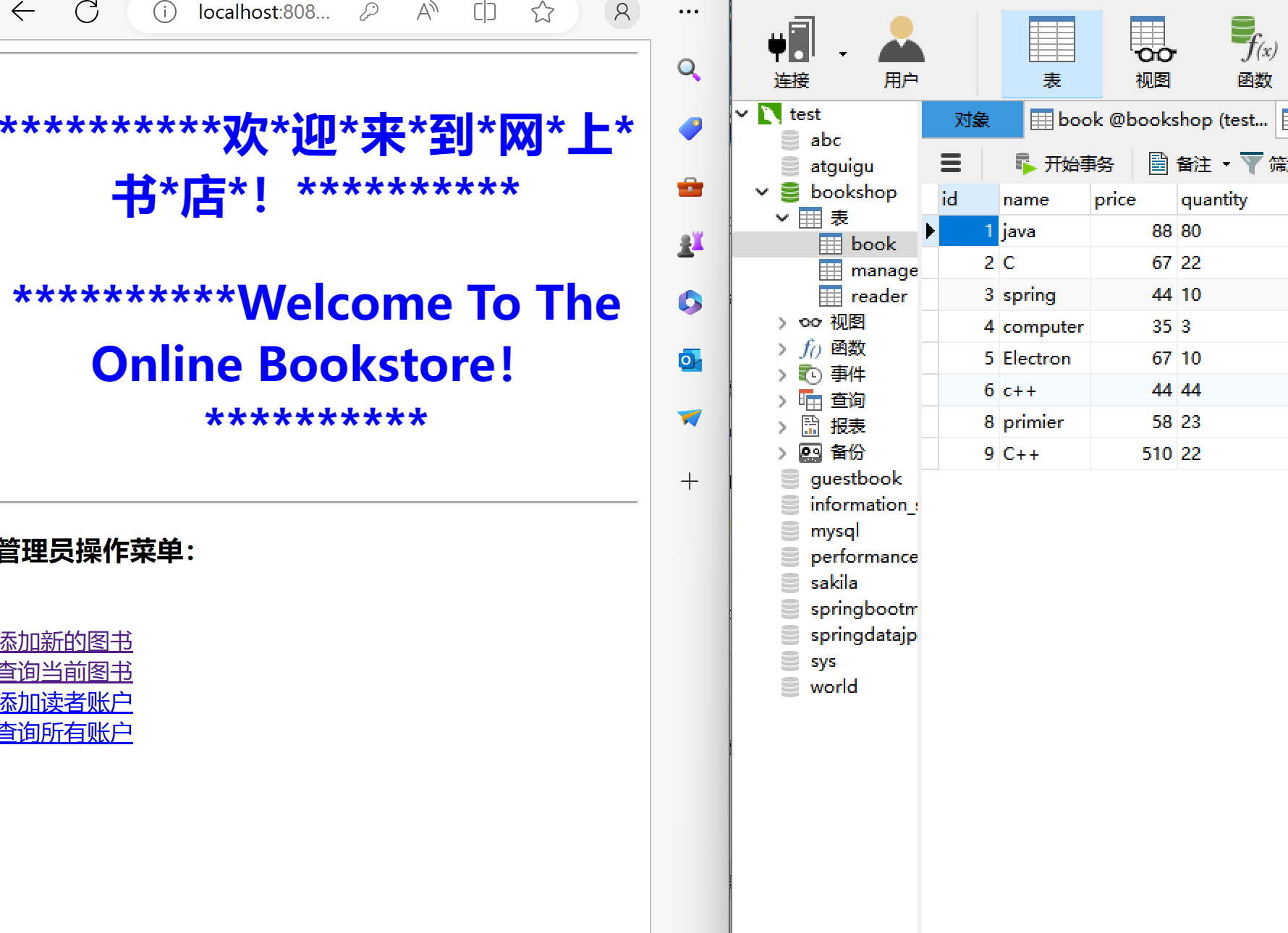
## 五、系统界面截图

#### 1.管理员操作界面

**（1）管理员登录**



**（2）管理员操作页面与数据库变动情况**

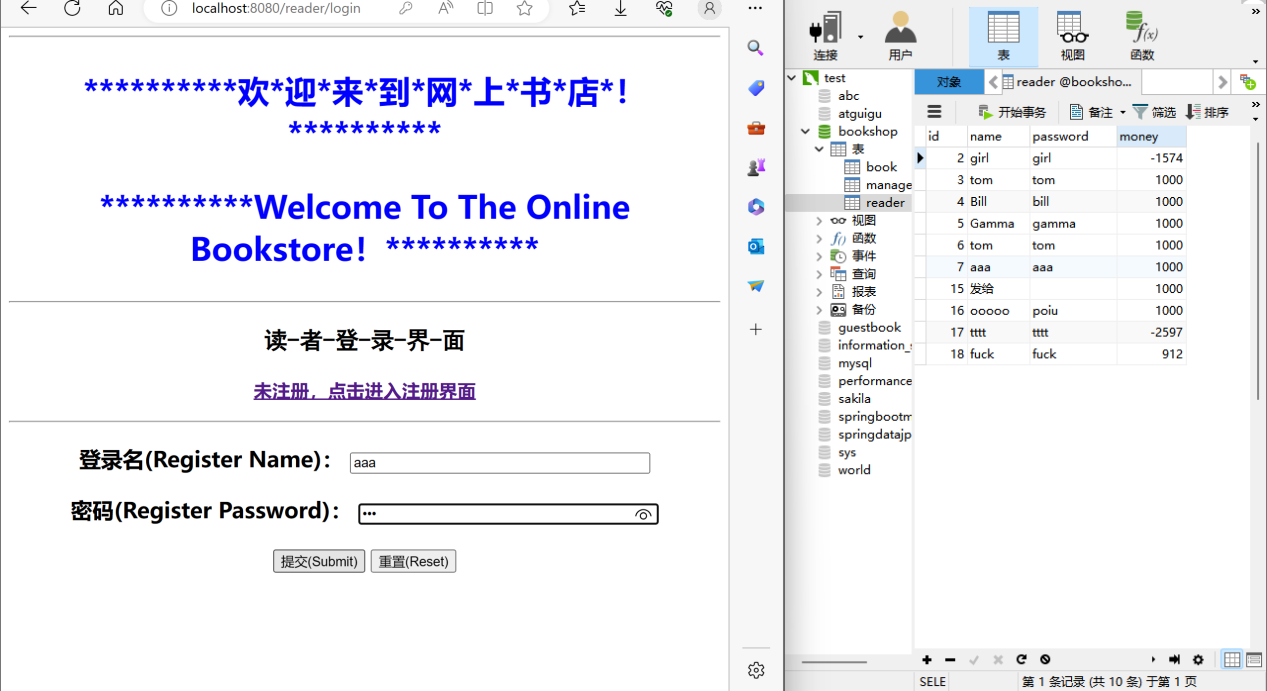


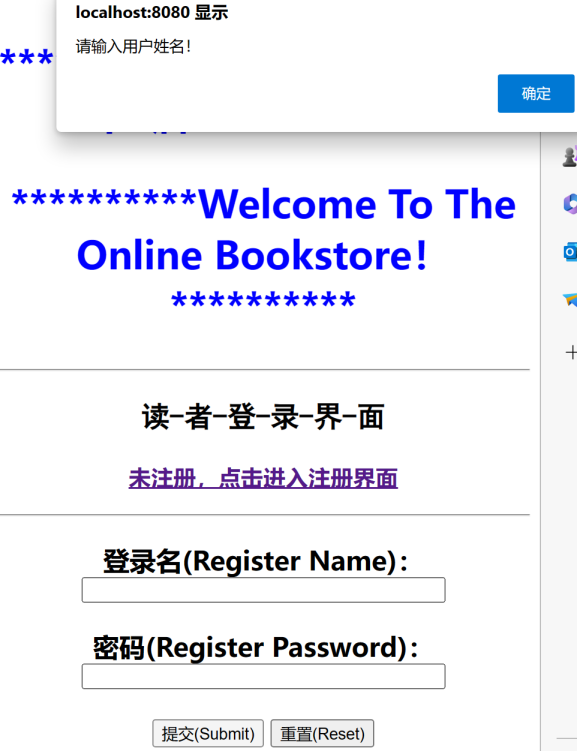
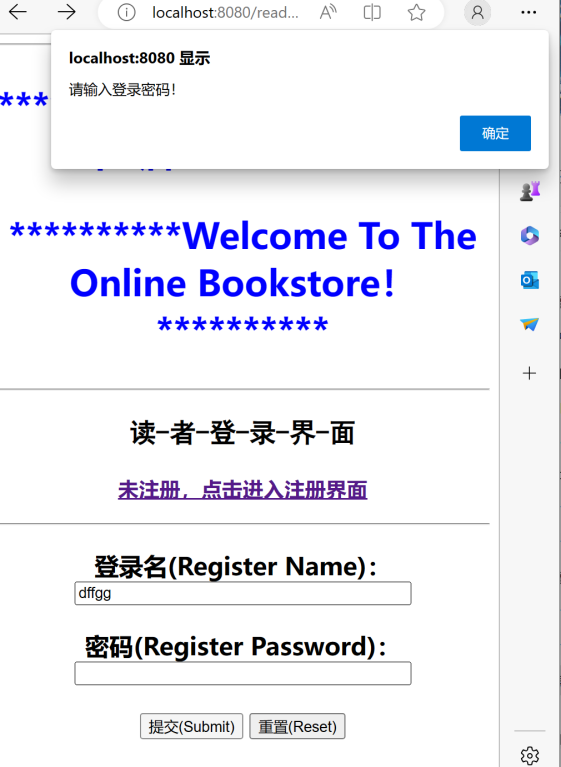
#### 读者操作界面

1. **读者注册界面**



**（2）读者登录界面**

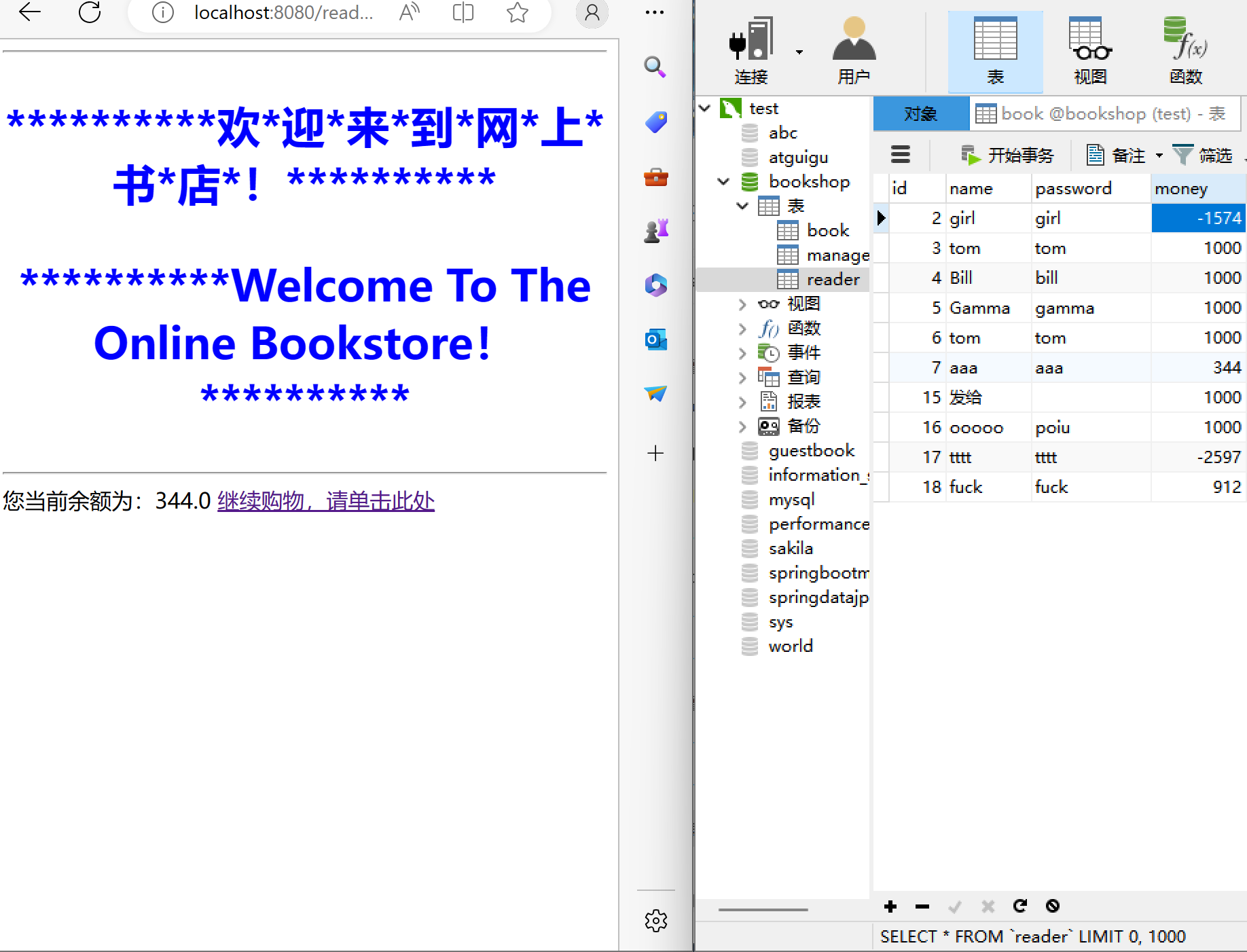




**（3）添加购物车和支付界面**



**（4）支付界面**



## 六．系统代码附录（参见spring boot工程提交）