**线上书店管理系统(onlineBSMS)**

**概要/详细设计**

当前版本：V0.2

最后修改时间：<2023年12月17日>

版权所有◎2023年

# 目录

[第一部分 引言 1](#_Toc153733244)

[1.1 系统标识 1](#_Toc153733245)

[1.2 编写目的 1](#_Toc153733246)

[1.3 读者群与知识储备 1](#_Toc153733247)

[1.4 术语与缩略词定义 2](#_Toc153733248)

[1.5 参考文献 2](#_Toc153733249)

[第二章 概要设计 4](#_Toc153733250)

[2.1 设计注意事项、原则、目标及相关规范 4](#_Toc153733251)

[2.1.1 基本需求说明 4](#_Toc153733252)

[2.1.2 软件平台和开发工具选择 5](#_Toc153733253)

[2.1.3 前后端开发框架选择 5](#_Toc153733254)

[2.1.4 安全设计 5](#_Toc153733255)

[2.1.5 相关规范 5](#_Toc153733256)

[2.2 总体结构设计 6](#_Toc153733257)

[2.3 构件设计 7](#_Toc153733258)

[2.3.1 构件1 初始化数据库 7](#_Toc153733259)

[2.3.2 构件2 插入书籍信息 7](#_Toc153733260)

[2.3.3 构件3 插入作者信息 8](#_Toc153733261)

[2.3.4 构件4 插入类别信息 9](#_Toc153733262)

[2.3.5 构件5 插入出版社信息 9](#_Toc153733263)

[2.3.6 构件6 插入顾客信息 10](#_Toc153733264)

[2.3.7 构件7 插入权限与折扣的对应信息 11](#_Toc153733265)

[2.3.8 构件8 插入订单信息 11](#_Toc153733266)

[2.3.9 构件9 插入订单与书的对应信息 12](#_Toc153733267)

[2.3.10 构件10 插入管理员的信息 12](#_Toc153733268)

[2.3.11 构件11 删除用户账号 13](#_Toc153733269)

[2.3.12 构件12 删除当前订单对应信息 14](#_Toc153733270)

[2.3.13 构件13 修改书籍相关信息 15](#_Toc153733271)

[2.3.14 构件14 修改当前订单对应信息 16](#_Toc153733272)

[2.3.15 构件15 查询并精准查看书籍对应信息 17](#_Toc153733273)

[2.3.15 构件16 书籍的二维统计 18](#_Toc153733274)

[2.4 图形用户接口设计 19](#_Toc153733275)

[2.5 数据库设计 19](#_Toc153733276)

[2.6 系统使用方法 22](#_Toc153733277)

[2.7 公共函数库 23](#_Toc153733278)

[第三章 详细设计 24](#_Toc153733279)

[3.1 开发环境概述 24](#_Toc153733280)

[3.1.1 版本控制工具 24](#_Toc153733281)

[3.1.2 编译环境 24](#_Toc153733282)

[3.1.3 源程序目录 24](#_Toc153733283)

[3.1.4 文档存储目录 28](#_Toc153733284)

[3.1.5 打包程序/编译顺序 42](#_Toc153733285)

[3.2 构件详细设计 42](#_Toc153733286)

[3.2.1 构件1 数据库初始化 42](#_Toc153733287)

[3.2.2 构件11 删除用户账号 43](#_Toc153733288)

[3.2.3 构件13 修改书籍相关信息 43](#_Toc153733289)

[3.2.4 构件15 查询并精准查看书籍对应信息 44](#_Toc153733290)

[3.2.5 构件16 书籍的二维统计 45](#_Toc153733291)

# 第一部分 引言

1.1 系统标识

项目名称：线上书店管理系统

项目简称：onlineBSMS

=1.2 编写目的

为了让该MIS的用户更好地使用该系统，同时记录系统的技术设计，进行存档，软件系统技术人员需要撰写该文档，对自己负责的部分进行详细说明，并确保其可读性，在各部分间做好对接，提高项目团队合作效率。

在实际的工程开发中，对设计文档的审查与质询能够大幅度减少后续编程和单元测试的错误，同时间接提供了一个发现需求设计不合理性的机会，与口头对接工作相比可以减少歧义现象的产生。

本文详细说明了图书管理系统的数据库设计和构件设计，意在帮助开发人员和测试人员更好地了解本系统，便于对本系统进行评估、测试与改进。

本文详细说明了onlineBSMS的总体结构、开放环境、网页（图形用户接口）设计、数据库设计、构件设计以及其他技术细节，能够帮助开发人员和测试人员更好地了解本系统，便于对本系统进行评估、测试与改进。

1.3 读者群与知识储备

**读者群**：参与该项目设计、开发与验收测试的所有技术人员、书店方、系统内测用户

**知识储备**：SQL语言、数据库编程(JDBC)、数据库系统原理与安全课程理论知识、汉字编码技术、MVC设计模式、关系数据库管理系统MySQL、JSP+Servlet框架、Tomcat Web服务器

1.4 术语与缩略词定义

* onlineBSMS(Online Bookstore Manage System)：线上书店管理系统
* MIS(Management Information System)：管理信息系统
* SQL(Structured Query Language)：结构化查询语言
* JDBC(Java DataBase Connectivity)：Java数据库连接
* MVC(Model-View-Controller)模式：模型-视图-控制器模式
* MySQL：最流行的关系型数据库管理系统之一
* JSP(Java Server Pages)：Java服务器页面，由Sun Microsystems公司倡导和许多公司参与共同创建的一种使软件开发者可以响应客户端请求，而动态生成HTML、XML或其他格式文档的Web网页的技术标准。
* Servlet(Server Applet)：服务器小程序，运行在Web服务器或应用服务器上的程序，作为来自Web浏览器或其他HTTP客户端的请求和 HTTP服务器上的数据库或应用程序之间的中间层。
* Tomcat：免费的开放源代码的Web应用服务器,由java语言编写，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是一个开源的Java Servlet容器，也是一个JSP容器，将Java Web应用程序部署到服务器上，同时也提供了Java Servlet和JSP的运行环境
* Bootstrap：全球最受欢迎的前端组件库，用于开发响应式布局、移动设备优先的WEB项目。Bootstrap5目前是Bootstrap的最新版本，是一套用于HTML、CSS和JS开发的开源工具集。它支持Sass变量和mixins、响应式网格系统、大量的预建组件和强大的JavaScript插件，助你快速设计和自定义响应式、移动设备优先的站点。

1.5 参考文献

1. <https://www.runoob.com/w3cnote/jdbc-use-guide.html>
2. <https://www.runoob.com/design-pattern/mvc-pattern.html>
3. <https://zhuanlan.zhihu.com/p/620770652>
4. <https://www.runoob.com/servlet/servlet-intro.html>
5. <https://www.runoob.com/jsp/jsp-tutorial.html>
6. <https://v5.bootcss.com/docs/getting-started/introduction/>
7. <https://www.runoob.com/bootstrap5/bootstrap5-tutorial.html>
8. https://www.runoob.com/bootstrap5/bootstrap5-tutorial.html

# 第二章 概要设计

2.1 设计注意事项、原则、目标及相关规范

在进行系统概要设计之前，根据需求分析说明书的有关要求和预定的设计目标，结合以前的设计经验，需要对整个系统设计所采用的技术、方法有个统一的评估和规划，然后再作一个平衡、折衷的选择。在此过程中，需要记录各种替代设计方案及其评估结论。所考虑的因素并不局限以下各个方面：

硬件/软件平台

开发/测试工具的选择依据

编程语言、代码规范

设计方法

内存管理、分配策略

设计假设、约束和依赖性，如硬件、软件环境限制，对第三方软件的依赖等

互操作要求

接口/协议要求

遵循标准要求

性能要求

系统的可扩展性

子系统或模块之间的数据交换、通讯方式/方法选择

系统设计目标：追求简单还是复杂；强调速度、内存占用；平台相关性；多进程、多线索

2.1.1 基本需求说明

本数据库管理系统意在对一个线上书店进行管理。面向的对象主要有普通用户和管理员两类，通过identity属性区分。

用户能够查看书籍信息，按照相关条件检索与统计，并管理自己的当前订单，进行增减与提交购买，并查看历史订单。当用户付款金额到达一定总量时，vip等级提升，购买时享受相应折扣。此外，该线上书店不收取任何运费。用户还能对自己的基本账号信息如邮箱、手机号等进行管理，并修改密码和销号。

管理员不进行购书操作，没有vip等级，但需要对书籍、用户以及相关账号进行定期维护，并统计分析相关信息。

总体而言，本系统实现了主要数据实体的录入、修改、删除功能，同时对主要存储数据，根据多个给定组合条件，实现查询功能，并针对可能的统计需求，设计二维统计表，将统计结果输出到csv文件，还实现了vip升级、级联删除、访问控制等其他功能。具体的功能介绍见8.系统运行结果截图。

2.1.2 软件平台和开发工具选择

关系数据库管理系统采用MySQL，更适用于本系统中轻量级数据的使用。

代码编写和运行工具采用Intellij IDEA，其功能强大，具有高效的开发工具、插件生态系统和友好的用户界面等，被广泛认可为一款优秀的集成开发环境。

Web服务器采用tomcat，相对于其他Java应用服务器来说比较轻量，启动和部署速度较快，适合于中小型项目和简单的应用程序，且对Servlet和JSP规范的支持很好，能够很好地运行这些类型的应用程序，同时可以与其他Apache项目（如Apache HTTP服务器）结合使用，也支持通过插件和扩展实现功能的增强，可扩展性强。

2.1.3 前后端开发框架选择

前端使用Bootstrap5+JSP+[html+javascript+css]，后端使用Servlet，系统采用基于JAVA的B/S架构，采用MVC设计模式，并用JDBC连接数据库。

在这种设计中，前端和后端逻辑分离，JSP用于构建动态页面，Servlet用于处理用户请求并生成动态内容。这种分离使得前端和后端开发可以更清晰地分工合作；同时，JSP和Servlet是Java EE的一部分，因此在使用这些技术时可以利用Java EE提供的丰富的功能和库；同时，JSP和Servlet提供了丰富的API和生态系统，可以方便地扩展应用程序的功能。

2.1.4 安全设计

由于涉及到用户密码等隐私信息，需要设法保证用户隐私。因此，本系统通过对用户密码取哈希来防止用户信息泄露或被非法访问。

2.1.5 相关规范

采用Java EE规范，相关内容如下：

**(1)JDBC**

Java EE规范定义了JDBC API，这是Java用于与数据库进行交互的标准接口。它允许开发人员编写Java应用程序来执行SQL查询、更新数据库记录等操作；JDBC 规范还包括数据源和连接池的标准配置，以及对各种数据库类型的标准化访问。

**(2)Servlet**

Java EE规范涉及Servlet，定义了如何创建和管理Servlet的标准方法，Servlet是用于处理Web请求的Java类，常用于构建动态Web应用程序；Java EE规范还包括关于Servlet生命周期、请求处理、会话管理、安全性等方面的详细标准。

**(3)JSP**

Java EE规范中也包含了关于JSP的规定，包括JSP页面的语法、指令、标签库、表达式语言的使用等方面的标准；此外，Java EE规范还定义了JSP和Servlet之间的协作机制，以实现动态Web页面的构建和呈现。

2.2 总体结构设计

以构件为基础的总体架构图如图 1所示：

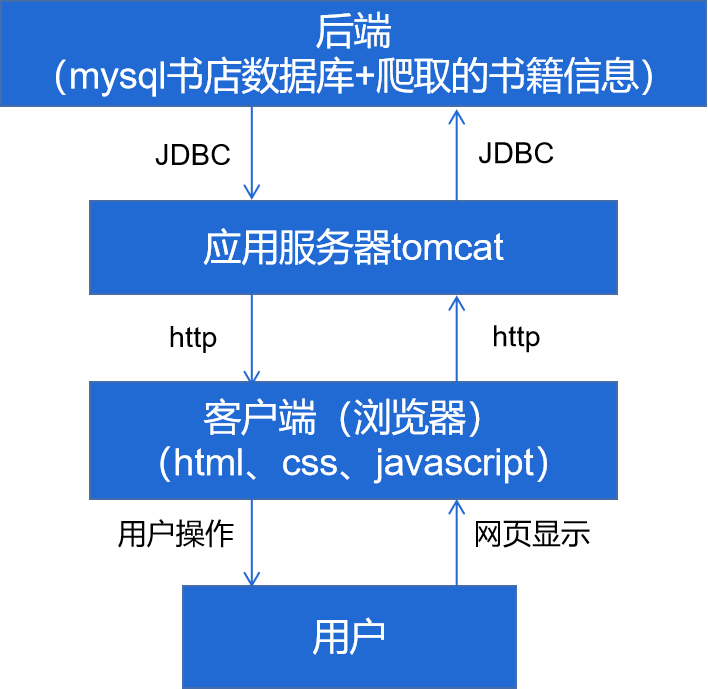


图 1 总体架构图

2.3 构件设计

2.3.1 构件1 初始化数据库

(1)功能描述

适当改造爬取的大规模数据，初始化线上书店管理系统数据库。管理员在初始时于后端执行依次执行drop.sql脚本，schema.sql脚本，data.sql脚本，用最初的数据初始化数据库。

**(2)接口规范**

功能描述：实现一个线上书店管理系统的初始化数据库功能，该功能实现较为简单，只需要依次执行三个SQL脚本：drop.sql、schema.sql、data.sql。这一过程将使用爬取的大规模数据，确保数据库得到最初的数据状态。

参数说明：

* drop.sql： 包含数据库清空的脚本，删除所有表和相关数据。
* schema.sql： 包含数据库表结构定义的脚本，创建必要的表和约束。
* data.sql： 包含初始数据插入的脚本，使用爬取的大规模数据填充数据库。

2.3.2 构件2 插入书籍信息

(1)功能描述

此构件为插入书籍信息。此功能要求管理员输入书籍的信息，包括国际标准书号、书名、书籍的简要描述、发行的年份、评论数、好评率、出版社名称、主要作者名字、类别名、单价，对书籍表格进行相应的操作。

**(2)对象模型**

Book对象模型：

title：String

isbn：String

publishHouse：publishHouse

author：Author

description：String

publishDate：String

category：Category

reviewAmount：Integer

goodRate：Double

price：Double

surplus：Integer

**(3)接口规范**

功能描述：根据管理员给定的参数插入书籍信息。

参数说明：输入为国际标准书号、书名、书籍的简要描述、发行的年份、评论数、好评率、出版社名称、主要作者名字、类别名、单价，输出为数据插入成功页面或数据插入失败页面。在输入中，国际标准书号、书名、发行的年份、评论数、好评率、出版社名称、主要作者名字、类别名、单价为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。需要注意2.5中提及的外键约束。

2.3.3 构件3 插入作者信息

(1)功能描述

此构件为插入作者信息。此功能要求管理员输入作者的信息，包括主要作者名字、作者国籍、出生年份、相关描述、发行的书的本数，对作者表格进行相应的操作。

**(2)对象模型**

Author对象模型：

name：String

nationality：String

birthDay：String

description：String

publishNumber：Integer

publishBook：List<Book>

**(3)接口规范**

功能描述：根据管理员给定的参数插入作者信息。

参数说明：输入为主要作者名字、作者国籍、出生年份、相关描述、发行的书的本数，输出为数据插入成功页面或数据插入失败页面。在输入中，主要作者名字、作者国籍、出生年份、发行的书的本数为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。需要注意publishNumber与book对象的对应数量对应，publishBook与book对象相对应。

2.3.4 构件4 插入类别信息

(1)功能描述

此构件为插入类别信息。此功能要求管理员输入类别的信息，包括类别名、属于该类别的书的总数，对类别表格进行相应的操作。

**(2)对象模型**

Category对象模型：

categoryName：String

totalNumber：Integer

bookList：List<Book>

**(3)接口规范**

功能描述：根据管理员给定的参数插入类别信息。

参数说明：输入为类别名、属于该类别的书的总数，输出为数据插入成功页面或数据插入失败页面。在输入中，类别名、属于该类别的书的总数为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。注意totalNumber和bookList的对应。

2.3.5 构件5 插入出版社信息

(1)功能描述

此构件为插入出版社信息。此功能要求管理员输入出版社的信息，包括出版社名称、出版社地址、发行的不同书的总量，对出版社表格进行相应的操作。

**(2)对象模型**

PublishHouse对象模型：

name：String

location：String

totalPublish：Integer

**(3)接口规范**

功能描述：根据管理员给定的参数插入出版社信息。

参数说明：输入为出版社名称、出版社地址、发行的不同书的总量，输出为数据插入成功页面或数据插入失败页面。在输入中，出版社名称、出版社地址、发行的不同书的总量为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。注意totalPublish需要和Book对象模型中属于该出版社的总数对应。

2.3.6 构件6 插入顾客信息

(1)功能描述

此构件为插入顾客信息。此功能要求管理员输入顾客的信息，包括用户id、用户昵称、用户密码sha256哈希值、用户邮箱、用户手机号、用户vip等级、用户买书的已付总金额、当前订单编号（通过查询语句获取）、用户默认地址，对顾客表单进行相应的操作。

**(2)对象模型**

Customer对象模型：

name：String

password：String

id：String

email：String

telephone：String

location：String

viplevel：Integer

orders：List<Order>

totalCost：Double

**(3)接口规范**

功能描述：根据管理员给定的参数插入顾客信息。

参数说明：输入为用户id、用户昵称、用户密码sha256哈希值、用户邮箱、用户手机号、用户vip等级、用户买书的已付总金额、当前订单编号、用户默认地址，输出为数据插入成功页面或数据插入失败页面。在输入中，用户id、用户昵称、用户密码sha256哈希值、用户手机号、识别用户是否为管理员（超级用户）为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。其中，viplevel字段与totalCost有对应的映射关系，具体在2.3.7中叙述。

2.3.7 构件7 插入权限与折扣的对应信息

(1)功能描述

此构件为插入权限与折扣的对应信息。此功能要求管理员输入权限与折扣的对应信息，包括用户vip等级、打折率，对折扣表格进行相应的操作。

**(2)对象模型**

Vip\_rate对象模型：

level：String

rate：Double

**(3)接口规范**

功能描述：根据管理员给定的参数插入权限与折扣的对应信息。

参数说明：输入为用户vip等级、打折率，输出为数据插入成功页面或数据插入失败页面。在输入中，用户vip等级、打折率为必选字段，不允许接受空值。其他字段为可选字段，允许接受空值。

2.3.8 构件8 插入订单信息

(1)功能描述

此构件为插入订单信息。此功能要求用户输入选择的书籍，代码逻辑对信息进行自动处理，加入用户对应的当前订单中，同时OrderBook对象模型也需要做相应的改动。

**(2)对象模型**

Order对象模型：

orderNumber：String

customer：Customer

updateTime：String

address：String

price：Double

amount：Integer

bookList：List<Book>

**(3)接口规范**

功能描述：根据代码逻辑以及相关参数插入订单信息。

参数说明： orderNumber自增，updateTime取当前时间，address与用户的默认地址对应，一个订单只能属于一个用户。

2.3.9 构件9 插入订单与书的对应信息

(1)功能描述

此构件为插入订单与书的对应信息，由代码自动处理。

**(2)对象模型**

OrderBook对象模型：

ISBN:String

order\_id:int

order\_sum:int

book\_name:String

price:double

**(3)接口规范**

功能描述：用户对当前订单的书籍进行添加后代码逻辑自动执行

参数说明：参数与Book对象模型、Order对象模型有所对应

2.3.10 构件10 插入管理员的信息

(1)功能描述

此构件为管理员的信息。此功能要求管理员输入另一个管理员的信息，包括昵称、密码、id、邮箱(可选)、手机号、地址(可选)，对管理员表格进行相应的操作。

**(2)对象模型**

Administrator对象模型：

name：String

password：String

id：String

email：String

telephone：String

location：String

**(3)接口规范**

功能描述：根据管理员给定的参数插入其他管理员信息。

参数说明：思路与2.3.6类似

2.3.11 构件11 删除用户账号

(1)功能描述

删除id与管理员输入相同的对应元组，相关订单信息也需要级联删除。

**(2)对象模型**

Customer对象模型：

name：String

password：String

id：String

email：String

telephone：String

location：String

viplevel：Integer

orders：List<Order>

totalCost：Double

Order对象模型：

orderNumber：String

customer：Customer

updateTime：String

address：String

price：Double

amount：Integer

bookList：List<Book>

OrderBook对象模型：

ISBN:String

order\_id:int

order\_sum:int

book\_name:String

price:double

**(3)接口规范**

功能描述：管理员发现用户异常信息后，删除对应id的信息，并对订单相关内容进行级联删除。

参数说明：输入为管理员输入的用户id并点击提交按键，展示删除的订单信息。其中，用户id为必填字段，不允许接受空值。

2.3.12 构件12 删除当前订单对应信息

(1)功能描述

用户在确定不购买某一ISBN对应的书后，提交信息，将其从当前订单中删除，修改订单相关属性值或者删除相关元组。

**(2)对象模型**

Order对象模型：

orderNumber：String

customer：Customer

updateTime：String

address：String

price：Double

amount：Integer

bookList：List<Book>

OrderBook对象模型：

ISBN:String

order\_id:int

order\_sum:int

book\_name:String

price:double

**(3)接口规范**

功能描述：用户对需要在当前订单中删除的ISBN进行选择，删除关联信息

参数说明：输入为用户输入的ISBN信息并点击提交按键，删除成功。其中，ISBN为必填字段，不允许接受空值。同时，金额和数量信息会由更改。

2.3.13 构件13 修改书籍相关信息

(1)功能描述

为保证信息的时效性，管理员需要对书籍的信息进行定期维护。

**(2)对象模型**

Book对象模型：

title：String

isbn：String

publishHouse：publishHouse

author：Author

description：String

publishDate：String

category：Category

reviewAmount：Integer

goodRate：Double

price：Double

surplus：Integer

Author对象模型：

name：String

nationality：String

birthDay：String

description：String

publishNumber：Integer

publishBook：List<Book>

Category对象模型：

categoryName：String

totalNumber：Integer

bookList：List<Book>

PublishHouse对象模型：

name：String

location：String

totalPublish：Integer

**(3)接口规范**

功能描述：修改一些不违反外键约束的属性，如修改description的内容或更改书籍的作者为数据库中存在的其他作者，并同时修改出版社或作者的出版总数。

参数说明：根据需求选择需要更改的属性并做对应的修改，不接受破坏外键约束等恶意修改。

2.3.14 构件14 修改当前订单对应信息

(1)功能描述

用户在确定更改某一ISBN对应的书的欲购买数目后，提交信息，修改订单相关属性值。

**(2)对象模型**

Order对象模型：

orderNumber：String

customer：Customer

updateTime：String

address：String

price：Double

amount：Integer

bookList：List<Book>

OrderBook对象模型：

ISBN:String

order\_id:int

order\_sum:int

book\_name:String

price:double

**(3)接口规范**

功能描述：用户对需要输入在当前订单中的ISBN值和更改后的数目，实现对当前订单的更新。

参数说明：输入为用户输入的ISBN信息和更改后的订购量，并点击提交按键，更新成功。其中，ISBN和更新后的订购量为必填字段，不允许接受空值。同时，金额和数量信息会由更改。

2.3.15 构件15 查询并精准查看书籍对应信息

(1)功能描述

可以选择根据一个和多个属性进行书籍查询，最后也可以点击链接跳转至对应内容。

**(2)对象模型**

Book对象模型：

title：String

isbn：String

publishHouse：publishHouse

author：Author

description：String

publishDate：String

category：Category

reviewAmount：Integer

goodRate：Double

price：Double

surplus：Integer

Author对象模型：

name：String

nationality：String

birthDay：String

description：String

publishNumber：Integer

publishBook：List<Book>

Category对象模型：

categoryName：String

totalNumber：Integer

bookList：List<Book>

PublishHouse对象模型：

name：String

location：String

totalPublish：Integer

**(3)接口规范**

功能描述：用户选择查询条件，获得符合条件的书籍元组，随后可以点击链接跳转至书籍、作者、类别、出版商详情页。

参数说明：查询条件为书名（下拉栏选择）、出版时间（对应年份、下拉栏选择）、作者（下拉栏选择）、出版社（下拉栏选择）、分类（下拉栏选择）、好评率（<20%,20%-50%,50%-80%,80%-90%,>90%）、评论数（=0、1-1500、1501-15000、15000-100000、>100000）、价格（<20,20-50,50-100,>100）、默认排序方式（如按照价格升序等），可以选择一个或多个条件进行查询，获取满足条件的书籍完整元组信息。

2.3.15 构件16 书籍的二维统计

(1)功能描述

此功能为根据作者和类型（category）两个维度，设计梯度进行分类，统计每个类别的书籍数量。此功能要求用户在网站发起统计命令，系统可以根据用户的命令，对数据库中的相关表格进行相应的操作，并将统计结果总结为二维表，输出为csv/excel文件。

**(2)对象模型**

Book对象模型：

title：String

isbn：String

publishHouse：publishHouse

author：Author

description：String

publishDate：String

category：Category

reviewAmount：Integer

goodRate：Double

price：Double

surplus：Integer

Author对象模型：

name：String

nationality：String

birthDay：String

description：String

publishNumber：Integer

publishBook：List<Book>

Category对象模型：

categoryName：String

totalNumber：Integer

bookList：List<Book>

**(3)接口规范**

功能描述：根据作者和类型（category）两个维度，设计梯度进行分类，统计每个类别的书籍数量。

参数说明：功能无输入，输出为统计结果的表格和一个对应的csv/excel文件。

2.4 图形用户接口设计

详见系统运行截图文件

2.5 数据库设计

本系统采用MySQL数据库进行永久存储。

数据库表设计(schema or table design)如下，其余细节在文档1-3中展示，其中取值范围只叙述数据类型没有说明的部分。

**(1)book**

表 2 book关系描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 | 约束解释 | 索引 | 索引解释 |
| ISBN | 以97开头 | PK |  | ind\_book | 便于检索数据量较为庞大的书实体 |
| NOT NULL |  |
| book\_name |  | NOT NULL |  |  |  |
| book\_description |  |  |  |  |  |
| time |  | NOT NULL |  |  |  |
| comment\_num | 0 | NOT NULL |  |  |  |
| pos\_rate | 0.0-1.0 |  |  |  |  |
| press\_name |  | fk\_book\_press | FK |  |  |
| NOT NULL |  |
| author\_name |  | fk\_author\_press | FK |  |  |
| NOT NULL |  |
| category\_name |  | fk\_category\_press | FK |  |  |
| NOT NULL |  |
| price | >0 | NOT NULL |  |  |  |

**(2)author**

表 3 author关系描述

|  |  |  |
| --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 |
| author\_name |  | PK |
| NOT NULL |
| nationality | 国籍的英语形式表示 | NOT NULL |
| birth |  | NOT NULL |
| author\_description |  |  |
| pub\_sum | >0 | NOT NULL |

**(3)category**

表 4 category关系描述

|  |  |  |
| --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 |
| category\_name |  | PK |
| NOT NULL |
| sum | >0 | NOT NULL |

**(4)press**

表 5 press关系描述

|  |  |  |
| --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 |
| press\_name |  | PK |
| NOT NULL |
| location | >0 | NOT NULL |
| all\_sum | >0 | NOT NULL |

**(5)customer**

表 6 customer关系描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 | 约束解释 |
| customer\_id | 8-16位，由字母和数字组成 | PK |  |
| NOT NULL |  |
| nickname |  | NOT NULL |  |
| pw | 64位十六进制表示 | NOT NULL |  |
| mail\_id | 符合邮箱规范 |  |  |
| telephone | 11位数字 | NOT NULL |  |
| level | 0-6 | fk\_customer\_level | FK |
| purchase\_sum | 0 |  |  |
| identity | ‘super’或‘normal’ | NOT NULL |  |
| current\_order\_id | 1 | fk\_current | FK |
| def\_location |  |  |  |

**(6)vip\_rate**

表 7 vip\_rate关系描述

|  |  |  |
| --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 |
| level | 0-6 | PK |
| NOT NULL |
| rate | 1.0,0.9,0.8,0.7,0.6,0.5,0.4 | PK |
| NOT NULL |

**(7)orders**

表 8 orders关系描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 | 约束解释 | 索引 | 索引解释 |
| order\_id | 1 | PK |  | ind\_order | 便于检索数据量较为庞大的订单实体 |
| NOT NULL |  |
| update\_time |  | NOT NULL |  |  |  |
| receipt\_place |  | NOT NULL |  |  |  |
| customer\_id | 8-16位，由字母和数字组成 | NOT NULL |  |  |  |
| fk\_customer\_order | FK |
| price\_sum | 0 | NOT NULL |  |  |  |

**(8)order\_book**

表 9 order\_book关系描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 取值范围 | 约束设计 | 约束解释 |
| ISBN | 以97开头 | PK |  |
| NOT NULL |  |
| fk\_ob2 | FK |
| order\_id | 1 | NOT NULL |  |
| PK |  |
| fk\_ob1 | FK |
| order\_sum | >0 | NOT NULL |  |

2.6 系统使用方法

* 数据库配置为：

使用utf8mb4字符集；

创建数据库dbsc\_lab3，用户dbsc\_lab3，用户密码设为dbsc；

执行SQL语句：grant all on dbsc\_lab3.\* to dbsc\_lab3；

依次运行脚本文件onlineBS\_schema.sql、onlineBS\_data.sql。

* 系统startup：

**完成数据库配置后**

将打包的项目文件jspservlet.war放置于tomcat的webapps文件夹下；

运行tomcat；

访问<http://localhost:8080/jspservlet/>；

* 系统shutdown：关闭tomcat服务器连接与对应网页
* 源代码编译运行：

使用IDEA2023.3.1专业版打开项目

Project Structure ->Project中选择jdk1.8

Project Structure -> Facets中Add->web

Project Structure -> Artifacts -> Web Application: Exploded -> from Modules，选择相应模块即可

Run-> Edit Configurations

Tomcat Server->Local

Application server->configure，选择tomcat安装路径

Deployment -> + -> Artifact

* 高可靠性设计方案：实现代码的可靠性，通过相关检查机制实现，具体机制可见系统截图文档
* 项目环境为mysql 8.0.35，apache-tomcat-9.0.83，java 8。

2.7 公共函数库

jstl.jar

mysql-connector-j-8.0.32.jar

standard.jar

javax.annotation.jar

javax.ejb.jar

javax.jms.jar

javax.persistence.jar

javax.resource.jar

javax.servlet.jar

javax.servlet.jsp.jar

javax.transaction.jar

# 第三章 详细设计

3.1 开发环境概述

3.1.1 版本控制工具

git

3.1.2 编译环境

* 操作系统：Windows11
* 开发工具与环境：Mysql v8.0.29，IDEA Intellij 2023.2.5，Tomcat v9.0.83，JDK8

3.1.3 源程序目录

**数据库部分：**

D:.

book\_data.csv（导出的统计表）

generate.py（构造数据）

generate\_pw.py（构造数据）

hash\_pw.py（构造数据）

onlineBS\_data.sql（导入数据）

onlineBS\_drop.sql（删除模式）

onlineBS\_schema.sql（创建模式）

pw.csv（存储普通用户的明文密码）

**前端部分：**

web

│ account.jsp（普通用户账户管理界面）

│ add\_admin.jsp（添加管理员用户界面）

│ add\_book.jsp（添加书籍界面）

│ admins.jsp（展示管理员用户界面）

│ admin\_account.jsp（管理员账户管理界面）

│ admin\_sidebar.jsp（管理员界面侧边栏）

│ author\_detail.jsp（作者详情界面）

│ books.jsp（书籍管理界面）

│ book\_detail.jsp（书目详情页面）

│ change\_account\_info.jsp（修改账户信息界面）

│ change\_account\_passwd.jsp（修改账户密码界面）

│ current\_order.jsp（当前订单展示界面）

│ customers.jsp（普通用户管理界面）

│ history\_order.jsp（历史订单展示界面）

│ home.jsp（普通用户书籍展示界面）

│ index.jsp（默认主界面）

│ login.jsp（登录界面）

│ press\_detail.jsp（出版商详情页面）

│ register.jsp（注册界面）

│ sidebar.jsp（普通用户侧边栏界面）

│

├─assets

│ ├─css //bootstrap5的css文件夹（具体文件略）

│ ├─img

│ │ backimg.jpg //登录界面的背景图片

│ │ web\_icon.svg //网站图标

│ │

│ ├─js //bootstrap5的js文件夹（具体文件略）

│ ├─self\_css // 自己编写的部分css文件

│ │ self\_content.css

│ │ self\_sidebar.css

│ │ self\_table.css

│ │

│ └─self\_js //自定义的部分辅助文件

│ delete\_cookies.jsp

│ verify\_cookies.js

│

└─WEB-INF

│ web.xml

│

└─lib

jstl.jar

mysql-connector-j-8.0.32.jar

standard.jar

**后端部分：**

src

│

└─com

└─jspservlet

├─dao //数据库交互相关

│ AdminDao.java（管理员的数据库操作）

│ BookControl.java（对书籍的所有操作）

│ BoughtBookControl.java（对已购买书籍的操作）

│ CustomerDao.java（普通用户的数据库操作）

│ HistoricalControl.java（历史订单的操作）

│ ManageAccount.java（账户管理操作）

│ OrderControl.java（对订单的操作）

│ UserDao.java（AdminDao.java和CustomerDao.java的父类）

│

├─entity //实体相关

│ Administrator.java

│ Author.java

│ Book.java

│ Category.java

│ Customer.java

│ Order.java

│ OrderBook.java

│ PublishHouse.java

│ Sha256.java

│ Stash.java

│ User.java

│

├─servlet //前端交互相关

│ AccountServlet.java

│ AddAdminServlet.java

│ AddBookServlet.java

│ AddOrderBookServlet.java

│ AdminsServlet.java

│ AuthorDetailServlet.java

│ BookDetailServlet.java

│ ChangeAccountInfoServlet.java

│ ChangeOrderBookServlet.java

│ CountServlet.java（二维统计与导出功能）

│ CurrentOrderServlet.java

│ CustomersServlet.java

│ DeleteCustomerServlet.java

│ DeleteSelfServlet.java

│ HistoryOrdersServlet.java

│ HomeServlet.java

│ LoginServlet.java

│ MoveCurrentOrderBookServlet.java

│ PayOrderServlet.java

│ PressDetailServlet.java

│ RegisterServlet.java

│

└─util //辅助类

dbConnectUtil.java

3.1.4 文档存储目录

卷 Data 的文件夹 PATH 列表

卷序列号为 048E-D4CB

D:.

│ 1.数据库概念设计文档.pdf

│ 2.数据库逻辑设计文档.pdf

│ 3.数据库物理设计文档.xlsx

│ 4.系统详细设计文档.docx

│ 8.系统运行结果截图.pdf

│ 小组任务分工.txt

│ 整体感想.pdf

│

├─5.数据库脚本以及相关文件

│ book\_data.csv

│ generate.py

│ generate\_pw.py

│ hash\_pw.py

│ onlineBS\_data.sql

│ onlineBS\_drop.sql

│ onlineBS\_schema.sql

│ pw.csv

│

├─6.代码相关说明

│ 源代码文件说明.md

│ 源代码文件说明.pdf

│ 编译、运行、配置说明.md

│ 编译、运行、配置说明.pdf

│

└─7.前后端完整源代码及二进制打包文件

│ jspservlet.war

│

└─jspservlet

│ .gitignore

│ jspservlet.iml

│ README.md

│

├─lib

│ javax.annotation.jar

│ javax.ejb.jar

│ javax.jms.jar

│ javax.persistence.jar

│ javax.resource.jar

│ javax.servlet.jar

│ javax.servlet.jsp.jar

│ javax.transaction.jar

│

├─out

│ ├─artifacts

│ │ ├─jspservlet

│ │ │ jspservlet.war

│ │ │

│ │ └─jspservlet\_war\_exploded

│ │ │ account.jsp

│ │ │ add\_admin.jsp

│ │ │ add\_book.jsp

│ │ │ admins.jsp

│ │ │ admin\_account.jsp

│ │ │ admin\_sidebar.jsp

│ │ │ author\_detail.jsp

│ │ │ books.jsp

│ │ │ book\_detail.jsp

│ │ │ change\_account\_info.jsp

│ │ │ change\_account\_passwd.jsp

│ │ │ current\_order.jsp

│ │ │ customers.jsp

│ │ │ history\_order.jsp

│ │ │ home.jsp

│ │ │ index.jsp

│ │ │ login.jsp

│ │ │ press\_detail.jsp

│ │ │ register.jsp

│ │ │ sidebar.jsp

│ │ │

│ │ ├─assets

│ │ │ │ │

│ │ │ │ └─font

│ │ │ │ │ bootstrap-icons.css

│ │ │ │ │ bootstrap-icons.json

│ │ │ │ │ bootstrap-icons.min.css

│ │ │ │ │ bootstrap-icons.scss

│ │ │ │ │

│ │ │ │ └─fonts

│ │ │ │ bootstrap-icons.woff

│ │ │ │ bootstrap-icons.woff2

│ │ │ │

│ │ │ ├─css

│ │ │ │ bootstrap-grid.css

│ │ │ │ bootstrap-grid.css.map

│ │ │ │ bootstrap-grid.min.css

│ │ │ │ bootstrap-grid.min.css.map

│ │ │ │ bootstrap-grid.rtl.css

│ │ │ │ bootstrap-grid.rtl.css.map

│ │ │ │ bootstrap-grid.rtl.min.css

│ │ │ │ bootstrap-grid.rtl.min.css.map

│ │ │ │ bootstrap-reboot.css

│ │ │ │ bootstrap-reboot.css.map

│ │ │ │ bootstrap-reboot.min.css

│ │ │ │ bootstrap-reboot.min.css.map

│ │ │ │ bootstrap-reboot.rtl.css

│ │ │ │ bootstrap-reboot.rtl.css.map

│ │ │ │ bootstrap-reboot.rtl.min.css

│ │ │ │ bootstrap-reboot.rtl.min.css.map

│ │ │ │ bootstrap-utilities.css

│ │ │ │ bootstrap-utilities.css.map

│ │ │ │ bootstrap-utilities.min.css

│ │ │ │ bootstrap-utilities.min.css.map

│ │ │ │ bootstrap-utilities.rtl.css

│ │ │ │ bootstrap-utilities.rtl.css.map

│ │ │ │ bootstrap-utilities.rtl.min.css

│ │ │ │ bootstrap-utilities.rtl.min.css.map

│ │ │ │ bootstrap.css

│ │ │ │ bootstrap.css.map

│ │ │ │ bootstrap.min.css

│ │ │ │ bootstrap.min.css.map

│ │ │ │ bootstrap.rtl.css

│ │ │ │ bootstrap.rtl.css.map

│ │ │ │ bootstrap.rtl.min.css

│ │ │ │ bootstrap.rtl.min.css.map

│ │ │ │

│ │ │ ├─img

│ │ │ │ backimg.jpg

│ │ │ │ web\_icon.svg

│ │ │ │

│ │ │ ├─js

│ │ │ │ bootstrap.bundle.js

│ │ │ │ bootstrap.bundle.js.map

│ │ │ │ bootstrap.bundle.min.js

│ │ │ │ bootstrap.bundle.min.js.map

│ │ │ │ bootstrap.esm.js

│ │ │ │ bootstrap.esm.js.map

│ │ │ │ bootstrap.esm.min.js

│ │ │ │ bootstrap.esm.min.js.map

│ │ │ │ bootstrap.js

│ │ │ │ bootstrap.js.map

│ │ │ │ bootstrap.min.js

│ │ │ │ bootstrap.min.js.map

│ │ │ │

│ │ │ ├─self\_css

│ │ │ │ self\_content.css

│ │ │ │ self\_sidebar.css

│ │ │ │ self\_table.css

│ │ │ │

│ │ │ └─self\_js

│ │ │ delete\_cookies.jsp

│ │ │ verify\_cookies.js

│ │ │

│ │ └─WEB-INF

│ │ │ web.xml

│ │ │

│ │ ├─classes

│ │ │ │ Servlet.class

│ │ │ │

│ │ │ └─com

│ │ │ └─jspservlet

│ │ │ ├─dao

│ │ │ │ AdminDao.class

│ │ │ │ BookControl.class

│ │ │ │ BoughtBookControl.class

│ │ │ │ CustomerDao.class

│ │ │ │ HistoricalControl.class

│ │ │ │ ManageAccount.class

│ │ │ │ OrderControl.class

│ │ │ │ UserDao.class

│ │ │ │

│ │ │ ├─entity

│ │ │ │ Administrator.class

│ │ │ │ Author.class

│ │ │ │ Book.class

│ │ │ │ Category.class

│ │ │ │ Customer.class

│ │ │ │ Order.class

│ │ │ │ OrderBook.class

│ │ │ │ PublishHouse.class

│ │ │ │ Sha256.class

│ │ │ │ Stash.class

│ │ │ │ User.class

│ │ │ │

│ │ │ ├─servlet

│ │ │ │ AccountServlet.class

│ │ │ │ AddAdminServlet.class

│ │ │ │ AddBookServlet.class

│ │ │ │ AddOrderBookServlet.class

│ │ │ │ AdminsServlet.class

│ │ │ │ AuthorDetailServlet.class

│ │ │ │ BookDetailServlet.class

│ │ │ │ ChangeAccountInfoServlet.class

│ │ │ │ ChangeOrderBookServlet.class

│ │ │ │ CountServlet.class

│ │ │ │ CurrentOrderServlet.class

│ │ │ │ CustomersServlet.class

│ │ │ │ DeleteCustomerServlet.class

│ │ │ │ DeleteSelfServlet.class

│ │ │ │ HistoryOrdersServlet.class

│ │ │ │ HomeServlet.class

│ │ │ │ LoginServlet.class

│ │ │ │ MoveCurrentOrderBookServlet.class

│ │ │ │ PayOrderServlet.class

│ │ │ │ PressDetailServlet.class

│ │ │ │ RegisterServlet.class

│ │ │ │

│ │ │ └─util

│ │ │ dbConnectUtil.class

│ │ │

│ │ ├─lib

│ │ │ jstl.jar

│ │ │ mysql-connector-j-8.0.32.jar

│ │ │ standard.jar

│ │ │

│ │ └─test

│ └─production

│ └─jspservlet

│ └─com

│ └─jspservlet

│ ├─dao

│ │ AdminDao.class

│ │ BookControl.class

│ │ BoughtBookControl.class

│ │ CustomerDao.class

│ │ HistoricalControl.class

│ │ ManageAccount.class

│ │ OrderControl.class

│ │ UserDao.class

│ │

│ ├─entity

│ │ Administrator.class

│ │ Author.class

│ │ Book.class

│ │ Category.class

│ │ Customer.class

│ │ Order.class

│ │ OrderBook.class

│ │ PublishHouse.class

│ │ Sha256.class

│ │ Stash.class

│ │ User.class

│ │

│ ├─servlet

│ │ AccountServlet.class

│ │ AddAdminServlet.class

│ │ AddBookServlet.class

│ │ AddOrderBookServlet.class

│ │ AdminsServlet.class

│ │ AuthorDetailServlet.class

│ │ BookDetailServlet.class

│ │ ChangeAccountInfoServlet.class

│ │ ChangeOrderBookServlet.class

│ │ CountServlet.class

│ │ CurrentOrderServlet.class

│ │ CustomersServlet.class

│ │ DeleteCustomerServlet.class

│ │ DeleteSelfServlet.class

│ │ HistoryOrdersServlet.class

│ │ HomeServlet.class

│ │ LoginServlet.class

│ │ MoveCurrentOrderBookServlet.class

│ │ PayOrderServlet.class

│ │ PressDetailServlet.class

│ │ RegisterServlet.class

│ │

│ └─util

│ dbConnectUtil.class

│

├─src

│ └─com

│ └─jspservlet

│ ├─dao

│ │ AdminDao.java

│ │ BookControl.java

│ │ BoughtBookControl.java

│ │ CustomerDao.java

│ │ HistoricalControl.java

│ │ ManageAccount.java

│ │ OrderControl.java

│ │ UserDao.java

│ │

│ ├─entity

│ │ Administrator.java

│ │ Author.java

│ │ Book.java

│ │ Category.java

│ │ Customer.java

│ │ Order.java

│ │ OrderBook.java

│ │ PublishHouse.java

│ │ Sha256.java

│ │ Stash.java

│ │ User.java

│ │

│ ├─servlet

│ │ AccountServlet.java

│ │ AddAdminServlet.java

│ │ AddBookServlet.java

│ │ AddOrderBookServlet.java

│ │ AdminsServlet.java

│ │ AuthorDetailServlet.java

│ │ BookDetailServlet.java

│ │ ChangeAccountInfoServlet.java

│ │ ChangeOrderBookServlet.java

│ │ CountServlet.java

│ │ CurrentOrderServlet.java

│ │ CustomersServlet.java

│ │ DeleteCustomerServlet.java

│ │ DeleteSelfServlet.java

│ │ HistoryOrdersServlet.java

│ │ HomeServlet.java

│ │ LoginServlet.java

│ │ MoveCurrentOrderBookServlet.java

│ │ PayOrderServlet.java

│ │ PressDetailServlet.java

│ │ RegisterServlet.java

│ │

│ └─util

│ dbConnectUtil.java

│

└─web

│ account.jsp

│ add\_admin.jsp

│ add\_book.jsp

│ admins.jsp

│ admin\_account.jsp

│ admin\_sidebar.jsp

│ author\_detail.jsp

│ books.jsp

│ book\_detail.jsp

│ change\_account\_info.jsp

│ change\_account\_passwd.jsp

│ current\_order.jsp

│ customers.jsp

│ history\_order.jsp

│ home.jsp

│ index.jsp

│ login.jsp

│ press\_detail.jsp

│ register.jsp

│ sidebar.jsp

│

├─assets

│ ├─css

│ │ bootstrap-grid.css

│ │ bootstrap-grid.css.map

│ │ bootstrap-grid.min.css

│ │ bootstrap-grid.min.css.map

│ │ bootstrap-grid.rtl.css

│ │ bootstrap-grid.rtl.css.map

│ │ bootstrap-grid.rtl.min.css

│ │ bootstrap-grid.rtl.min.css.map

│ │ bootstrap-reboot.css

│ │ bootstrap-reboot.css.map

│ │ bootstrap-reboot.min.css

│ │ bootstrap-reboot.min.css.map

│ │ bootstrap-reboot.rtl.css

│ │ bootstrap-reboot.rtl.css.map

│ │ bootstrap-reboot.rtl.min.css

│ │ bootstrap-reboot.rtl.min.css.map

│ │ bootstrap-utilities.css

│ │ bootstrap-utilities.css.map

│ │ bootstrap-utilities.min.css

│ │ bootstrap-utilities.min.css.map

│ │ bootstrap-utilities.rtl.css

│ │ bootstrap-utilities.rtl.css.map

│ │ bootstrap-utilities.rtl.min.css

│ │ bootstrap-utilities.rtl.min.css.map

│ │ bootstrap.css

│ │ bootstrap.css.map

│ │ bootstrap.min.css

│ │ bootstrap.min.css.map

│ │ bootstrap.rtl.css

│ │ bootstrap.rtl.css.map

│ │ bootstrap.rtl.min.css

│ │ bootstrap.rtl.min.css.map

│ │

│ ├─img

│ │ backimg.jpg

│ │ web\_icon.svg

│ │

│ ├─js

│ │ bootstrap.bundle.js

│ │ bootstrap.bundle.js.map

│ │ bootstrap.bundle.min.js

│ │ bootstrap.bundle.min.js.map

│ │ bootstrap.esm.js

│ │ bootstrap.esm.js.map

│ │ bootstrap.esm.min.js

│ │ bootstrap.esm.min.js.map

│ │ bootstrap.js

│ │ bootstrap.js.map

│ │ bootstrap.min.js

│ │ bootstrap.min.js.map

│ │

│ ├─self\_css

│ │ self\_content.css

│ │ self\_sidebar.css

│ │ self\_table.css

│ │

│ └─self\_js

│ delete\_cookies.jsp

│ verify\_cookies.js

│

└─WEB-INF

│ web.xml

│

└─lib

jstl.jar

mysql-connector-j-8.0.32.jar

standard.jar

3.1.5 打包程序/编译顺序

* 首先配置数据库

使用utf8mb4字符集；

创建数据库dbsc\_lab3，用户dbsc\_lab3，用户密码设为dbsc；

执行SQL语句：grant all on dbsc\_lab3.\* to dbsc\_lab3；

依次运行脚本文件onlineBS\_schema.sql、onlineBS\_data.sql，创建数据库。

* 使用IDEA2023.3.1专业版打开项目：

Project Structure ->Project中选择jdk1.8

Project Structure -> Facets中Add->web

Project Structure -> Artifacts -> Web Application: Exploded -> from Modules，选择相应模块即可

Run-> Edit Configurations

Tomcat Server->Local

Application server->configure，选择tomcat安装路径

Deployment -> + -> Artifact

* 随后运行项目即可，登录网页<http://localhost:8080/jspservlet/>即可开始使用系统

3.2 构件详细设计

由于部分构件功能类似，故该部分只对2.3中的部分构件进行阐述，编号与2.3中保持一致。

3.2.1 构件1 数据库初始化

此功能为数据初始化，由管理员进行，需要插入一些初始数据，同时需要满足系统设计的逻辑规范。

**(1)源程序列表**

* onlineBS\_drop.sql：首先实现对可能干扰项的删除
* onlineBS\_schema.sql：实现模式创建
* onlineBS\_data.sql：导入初始数据

**(2)主要设计思想**

由于管理员具有对数据的最初控制权，故所有数据的初始化由管理员在后端自行执行相应的sql语句完成。

3.2.2 构件11 删除用户账号

此构件功能为管理员的操作，按照id删除用户账号。此功能要求管理员能够删除对应id用户的所有相关信息，包括用户id、昵称、手机号、邮箱、已支付金额、vip等级等，系统可以根据输入，对数据库中的order表格、order\_book表格进行相应的操作（级联删除）。

**(1)源程序列表**

* Customer.java：顾客类
* DeleteCustomerServlet.java：实现删除用户类
* UserDao.java: 用户可以进行的操作

**(2)主要设计思想**

管理员在网页中对应的框内输入需要删除的用户id，点击相应按钮后，请求传给服务器，由DeleteCustomerServlet文件进行处理。由于在登录之后，服务器维持session，记录用户的账号和身份信息，故请求DeleteCustomerServlet文件中先检查是否登录，之后检查是否是管理员用户。如果符合前面两个条件则调用UserDao.java中的函数deleteAccount，该函数可以实现对数据库进行目标用户id删除的操作，Servlet文件调用函数并获得结果后，对结果进行处理，根据处理结果重定向到相应的jsp文件，生成不同的html页面返回给用户。

3.2.3 构件13 修改书籍相关信息

此功能为修改书籍相关信息。此功能要求管理员输入需要更改的相关信息以及更改后的语气结果，系统可以根据管理员的输入，对数据库中的书籍、出版社、作者等表格进行相应的操作。

**(1)源程序列表**

* AddBookServlet.java：实现管理员添加书籍
* AdminDao.java：管理员可以进行的操作

**(2)主要设计思想**

管理员在网页中对应的框内输入需要添加书的相关信息，点击提交按钮后，表单form向服务器的AddBookServlet发送post请求，传递相关信息。同构件11，AddBookServlet先根据session中保存的信息验证进行操作的人员的管理员身份，如果是管理员，则调用AdminDao文件中的addBookInfo函数，该函数可以实现对数据库进行书籍的插入，AddBookServlet文件调用函数并获得结果后，对结果进行处理，根据处理结果选择调用不同的输出jsp文件，展示不同的网页页面，用户得到网页页面，获得操作结果。

3.2.4 构件15 查询并精准查看书籍对应信息

**(1)源程序列表**

* AuthorDetailServlet.java：展示并协助查询作者信息的相关内容
* BookDetailServlet.java：展示并协助查询书籍信息（尤其是ISBN）的相关内容
* HomeServlet.java：传入参数进行查询
* PressDetailServlet.java：展示并协助查询出版社信息的相关内容
* BookControl.java：协助查询，进行书籍实体的控制
* author\_detail.jsp：展示作者具体信息
* book\_detail.jsp：展示书籍具体信息
* books.jsp：书籍管理对应页面
* home.jsp：普通用户可见的书籍信息页面
* press\_detail.jsp：展示出版社具体信息

**(2)主要设计思想**

书籍展示页面由home.jsp或books.jsp（取决于用户身份）获取所有书籍信息后生成，其中为每个查询返回的书目条目书名、作者和出版社属性构造了相应的链接，用户可通过点击相应的链接进行具体信息的查询。当点击相应链接时，浏览器向服务器发送相应的get请求，并传递相应的查询参数。根据链接的不同，请求分别传给，BookDetailServlet文件，AuthorDetailServlet文件，PressDetailServlet文件，并在相应的servlet中调用Dao部分相应的查询函数，相应的Servlet文件调用函数并获得结果后，对结果进行处理，根据处理结果选择重定向到不同的jsp文件（具体jsp文件在上面已经列出），生成不同的网页页面返回给浏览器。

3.2.5 构件16 书籍的二维统计

此功能为根据作者和书籍种类两个维度，设计梯度进行分类，统计每个类别的书籍数量。此功能要求用户在网站发起统计命令，系统可以根据用户的命令，对数据库中的相关表格进行相应的操作，并将统计结果总结为二维表，输出为csv/excel文件。

**(1)源程序列表**

* CountServlet.java：进行查询并创建二维表
* home.jsp：相应的前端代码

**(2)主要设计思想**

用户在网页点击统计按钮后，浏览器生成get请求传给服务器的CountServlet，服务器调用相应的查询方法，获得统计结果，之后对结果进行处理，包括建表，创建行列等，最后根据处理结果输出一个csv文件，其中包含各作者各类书的数目数量，进而获得操作结果。

需要注意的是，该功能没有输入，在数据库信息不变的情况下，所有用户进行统计操作得到的是同一个文件。这是因为，该功能固定以作者和种类两个维度进行统计，故在数据库信息不变的情况下统计结果始终不变。