天津大学

电子商城设计文档

学　院 智能与计算学部

专　业 软件工程

组　号 18

组　员 吴海鹰

学　号 3018216274

组　员 高芳卓

学　号 3018216254

组　员 刘昕

学　号 3018216232

组　员 王子阳

学　号 3018216272

电子商城设计文档

电子商城设计文档

1、引言

1.1 编写目的

在需求分析阶段，我们已经将系统用户对本系统的需求做了详细的阐述，这些用户需求应经在需求说明书中获得，并在需求说明书中得到详尽的叙述及阐明。

本阶段已在系统需求分析的基础上，对网上商城做出设计说明，包括模块设计说明，Restful接口设计，类图设计，业务对象设计以及数据库设计。

1.2 项目背景

在需求分析的基础上完成网上商城的设计，实现前后端分离，主要设计内容有：

1. 数据库设计
2. 与前端交互的RESTful API设计
3. 采用SpringBoot+Mybatis框架实现说明
4. 前端设计简要说明

1.3 定义

术语	解释
RESTfulAPI	是目前比较成熟的一套互联网应用程序的API设计理论，是典型的基于HTTP的协议。
Springboot	Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。
Mybatis	MyBatis支持定制化 SQL、存储过程以及高级映射。避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Ordinary Java Object,普通的 Java对象)映射成数据库中的记录。
Controller	控制层，负责暴露接口
Service	业务逻辑层
HTML	HTML称为超文本标记语言，是一种标识性的语言。它包括一系列标签．通过这些标签可以将网络上的文档格式统一，使分散的Internet资源连接为一个逻辑整体。
CSS	层叠样式表(Cascading Style Sheets)是一种用来表现HTML或XML等文件样式的计算机语言。CSS不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。
JS	JavaScript(简称"JS")，是一种属于网络的高级脚本语言,已经被广泛用于Web应用开发,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。通常JavaScript脚本是通过嵌入在HTML中来实现自身的功能的。

2、需求分析复审

2.1 涉众分析

用户

- 用户可以实现注册和登录。
- 能够通过搜索找到喜欢的图书。
- 将选择的书籍添加到购物车。
- 查看购物车中的商品，勾选商品并下单
- 删除购物车中的商品。
- 查询自己的所有订单记录。

管理员

- 管理商城中书籍的上架和下架。
- 管理商城宣传和图书宣传。

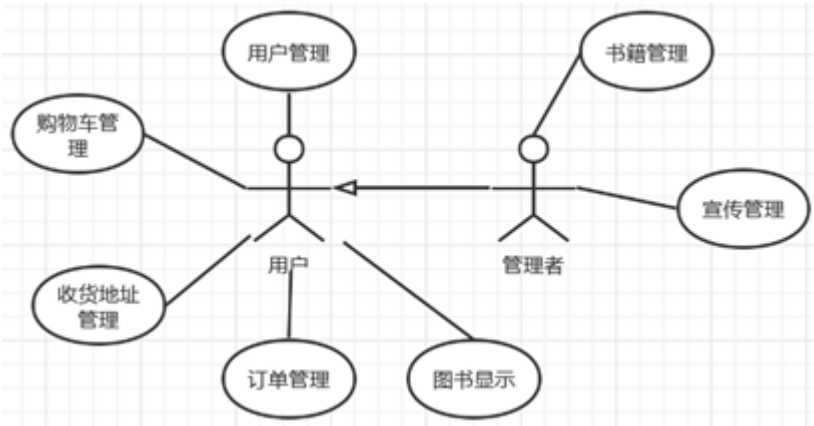
2.2 用例图

2.2.1 执行者分析

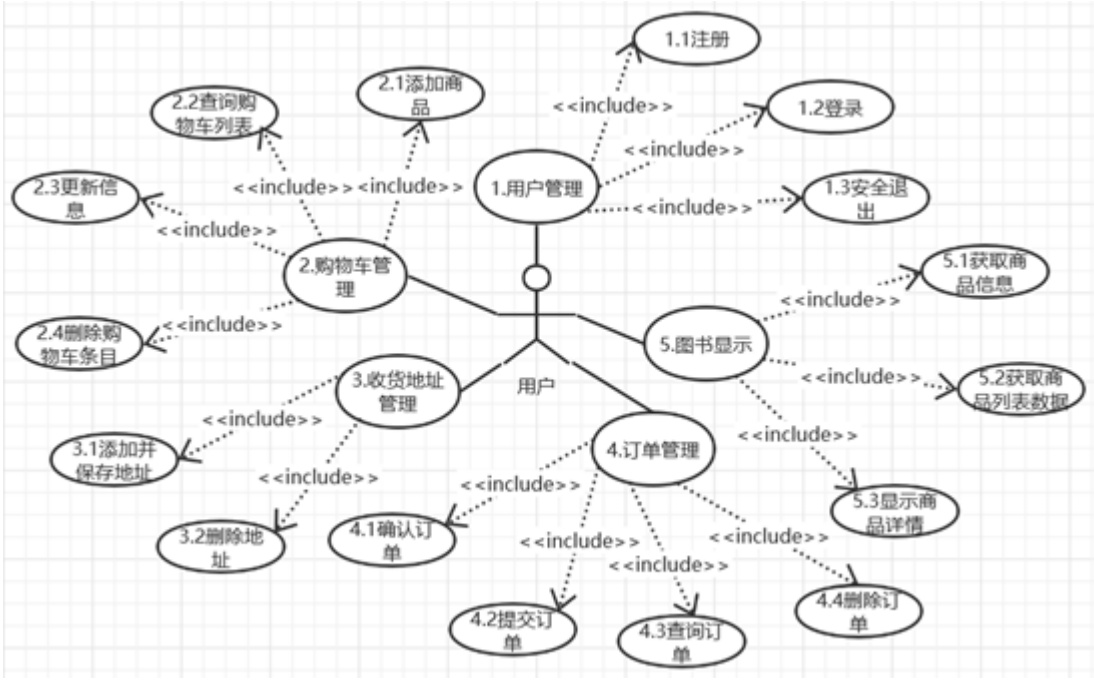
该系统执行者分为两类：用户和管理员。

两者的关系如图所示：

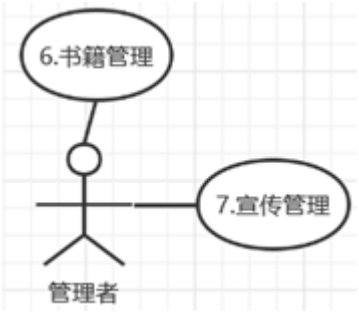
2.2.2 总用例图



2.2.3 用户用例



2.2.4 管理员用例

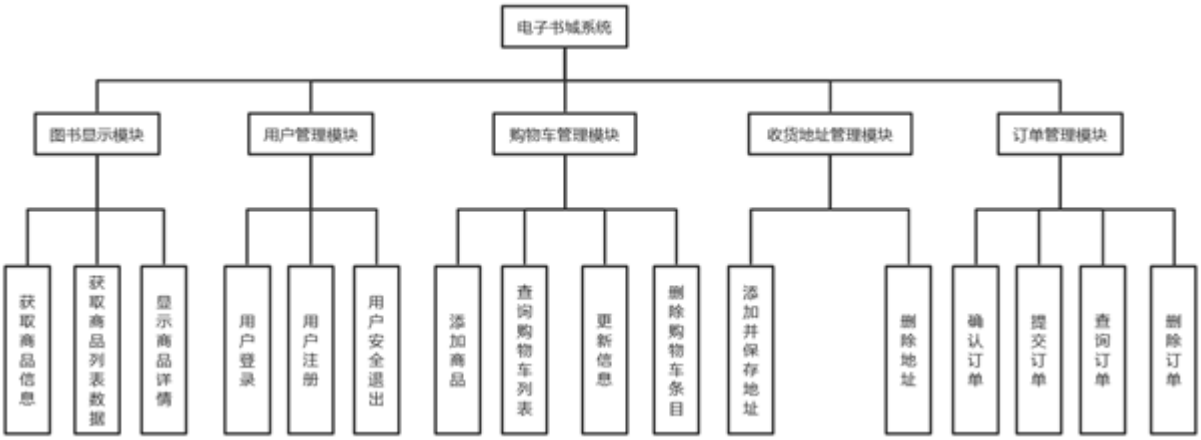


2.3 概要设计

根据需求分析复审，我们将电子商城系统依旧分为五大模块：

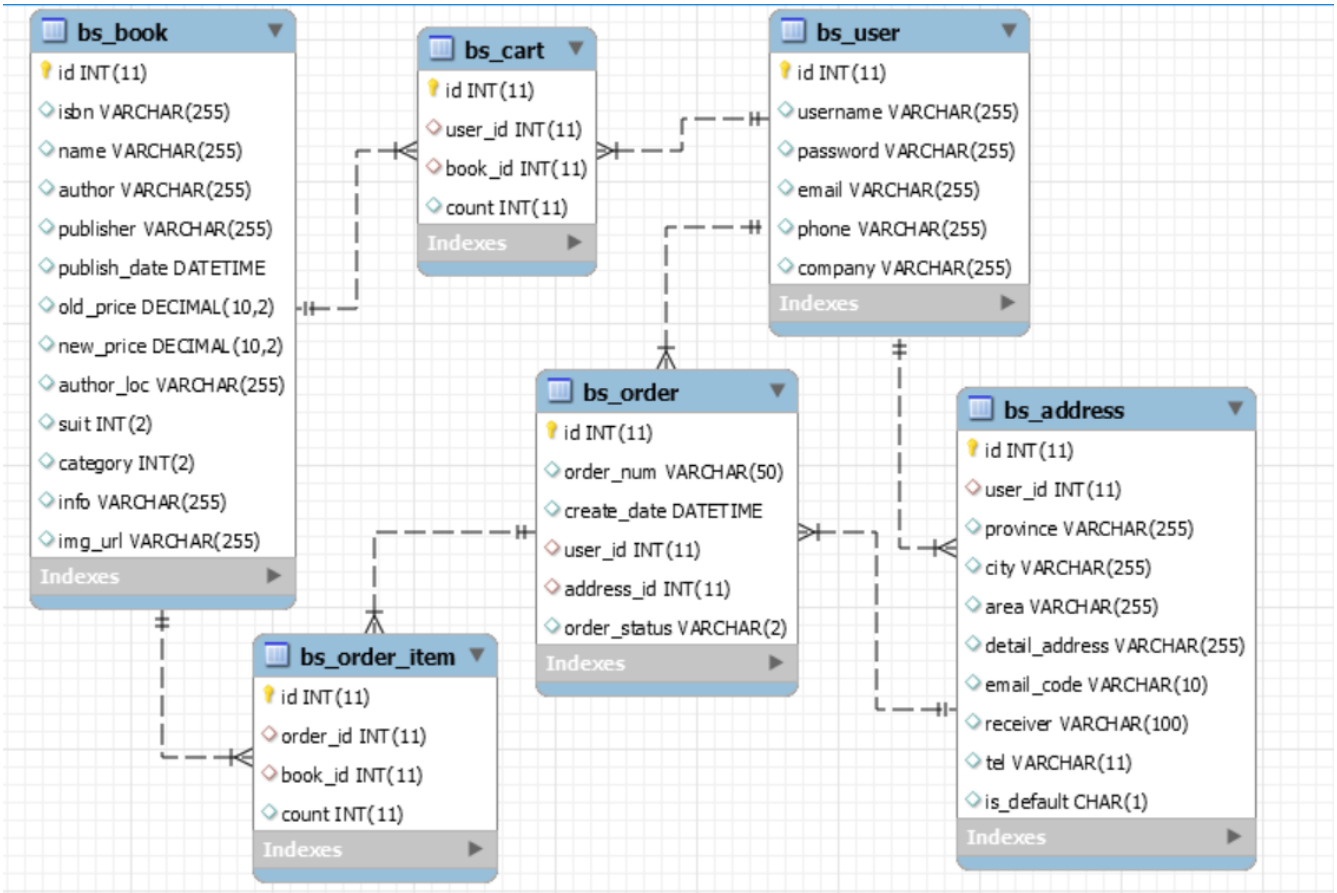
- 图书显示模块
- 用户管理模块
- 购物车管理模块
- 收货地址管理模块
- 订单管理模块

总体功能设计图如下：



3、数据库设计

3.1 ER图设计



3.2 数据库建表

3.2.1 用户信息表 bs_user：

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√		用户id
username	VARCHAR(255)	\	\	NULL	用户名
password	VARCHAR(255)	\	\	NULL	用户登录密码
email	VARCHAR(255)	\	\	NULL	邮箱
phone	VARCHAR(255)	\	\	NULL	电话
company	VARCHAR(255)	\	\	NULL	公司

建表语句：

```
CREATE TABLE `bs_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `phone` varchar(255) DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

3.2.2 用户地址表 bs_address

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√	\	地址id
user_id	INT(11)	FK	√	\	用户id
province	VARCHAR(255)	\	\	NULL	省
city	VARCHAR(255)	\	\	NULL	市
area	VARCHAR(255)	\	\	NULL	区/县
detail_address	VARCHAR(255)	\	\	NULL	详细地址（具体到门牌号）
email_code	VARCHAR(10)	\	\	NULL	邮编
receiver	VARCHAR(100)	\	\	NULL	收件人
tel	VARCHAR(11)	\	\	NULL	联系电话
is_default	CHAR(1)	'0'	\	NULL	是否默认地址，1为是；0为否

建表语句：

```
CREATE TABLE `bs_address` (
```

```
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) DEFAULT NULL,
  `province` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `area` varchar(255) DEFAULT NULL,
  `detail_address` varchar(255) DEFAULT NULL,
  `email_code` varchar(10) DEFAULT NULL,
  `receiver` varchar(100) DEFAULT NULL,
  `tel` varchar(11) DEFAULT NULL,
  `is_default` char(1) DEFAULT '0' COMMENT '0',
  PRIMARY KEY (`id`),
  KEY `fk_user_id` (`user_id`),
  CONSTRAINT `fk_user_id` FOREIGN KEY (`user_id`) REFERENCES `bs_user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
```

3.3.3 商品信息表（商品为图书）bs_book

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√	\	商品id
isbn	VARCHAR(255)	\	\	NULL	国际标准书号
name	VARCHAR(255)	\	\	NULL	商品（图书）名称
author	VARCHAR(255)	\	\	NULL	作者
publisher	VARCHAR(255)	\	\	NULL	出版商
publish_date	DATETIME	\	\	NULL	出版日期
old_price	DECIMAL(10,2)	\	\	NULL	旧价格
new_price	DECIMAL(10,2)	\	\	NULL	新价格
author_loc	VARCHAR(255)	\	\	NULL	作者介绍
suit	INT(2)	\	\	NULL	是否为套装
category	INT(2)	\	\	NULL	商品（图书）类别
info	VARCHAR(255)	\	\	NULL	商品（图书）详情
img_url	VARCHAR(255)	\	\	NULL	商品图片存放位置

建表语句

```
CREATE TABLE `bs_book` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `isbn` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `author` varchar(255) DEFAULT NULL,
  `publisher` varchar(255) DEFAULT NULL,
  `publish_date` datetime DEFAULT NULL,
```

```
  `old_price` decimal(10,2) DEFAULT NULL,
  `new_price` decimal(10,2) DEFAULT NULL,
  `author_loc` varchar(255) DEFAULT NULL,
  `suit` int(2) DEFAULT NULL,
  `category` int(2) DEFAULT NULL,
  `info` varchar(255) DEFAULT NULL,
  `img_url` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8;
```

3.3.4 购物车信息表 bs_cart

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√	\	购物车id
user_id	INT(11)	FK	√	\	用户id
book_id	INT(11)	FK	√	\	商品id
count	INT(11)	\	\	NULL	商品数目

建表语句

```
CREATE TABLE `bs_cart` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) DEFAULT NULL,
  `book_id` int(11) DEFAULT NULL,
  `count` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_cart_user_id` (`user_id`),
  KEY `fk_cart_book_id` (`book_id`),
  CONSTRAINT `fk_cart_book_id` FOREIGN KEY (`book_id`) REFERENCES `bs_book` (`id`),
  CONSTRAINT `fk_cart_user_id` FOREIGN KEY (`user_id`) REFERENCES `bs_user` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3.3.5 订单信息表 bs_order

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√	\	订单id
order_num	VARCHAR(50)	\	\	NULL	订单号
create_date	DATETIME	\	\	NULL	订单创建时间
user_id	INT(11)	FK	√	\	购买用户id
address_id	INT(11)	FK	√	\	地址id
order_status	VARCHAR(2)	\	\	NULL	支付状态

建表语句：

```
CREATE TABLE `bs_order` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `order_num` varchar(50) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `address_id` int(11) DEFAULT NULL,
  `order_status` varchar(2) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_order_user_id` (`user_id`),
  KEY `fk_order_addr_id` (`address_id`),
  CONSTRAINT `fk_order_addr_id` FOREIGN KEY (`address_id`) REFERENCES `bs_address` (`id`),
  CONSTRAINT `fk_order_user_id` FOREIGN KEY (`user_id`) REFERENCES `bs_user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

3.3.6 订单项目信息表 bs_order_item

Column Name	Datatype	PK/FK	Not Null	Default	Description
id	INT(11)	PK	√	\	订单项目id
order_id	INT(11)	FK	√	\	订单id
book_id	INT(11)	FK	√	\	商品id
count	INT(11)	\	\	NULL	商品数目

建表语句：

```
CREATE TABLE `bs_order_item` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `order_id` int(11) DEFAULT NULL,
  `book_id` int(11) DEFAULT NULL,
  `count` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_item_order_id` (`order_id`),
  KEY `fk_book_id` (`book_id`),
  CONSTRAINT `fk_book_id` FOREIGN KEY (`book_id`) REFERENCES `bs_book` (`id`),
  CONSTRAINT `fk_item_order_id` FOREIGN KEY (`order_id`) REFERENCES `bs_order` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

4、Restful API 设计

4.1 图书显示模块

- /book/index 商城网站主页
- /book/getBookData 获取商品信息
- /book/bookList 商品列表页
- /book/getBookListData 获取商品列表数据
- /book/detail 显示商品详情

4.2 用户管理模块

- /user/checkUser/Name 验证用户用户名是否存在
- /user/register 用户注册
- /user/login 用户登录
- user/logout 用户安全退出

4.3 购物车管理模块

- /cart/add 向购物车中添加商品
- /cart/list 查询当前用户购物车列表
- /cart/update 更新购物车信息
- /cart/delete 删除购物车条目

4.4 收货地址管理模块

- /address/save 添加并保存地址
- /address/delete 删除地址

4.5 订单管理模块

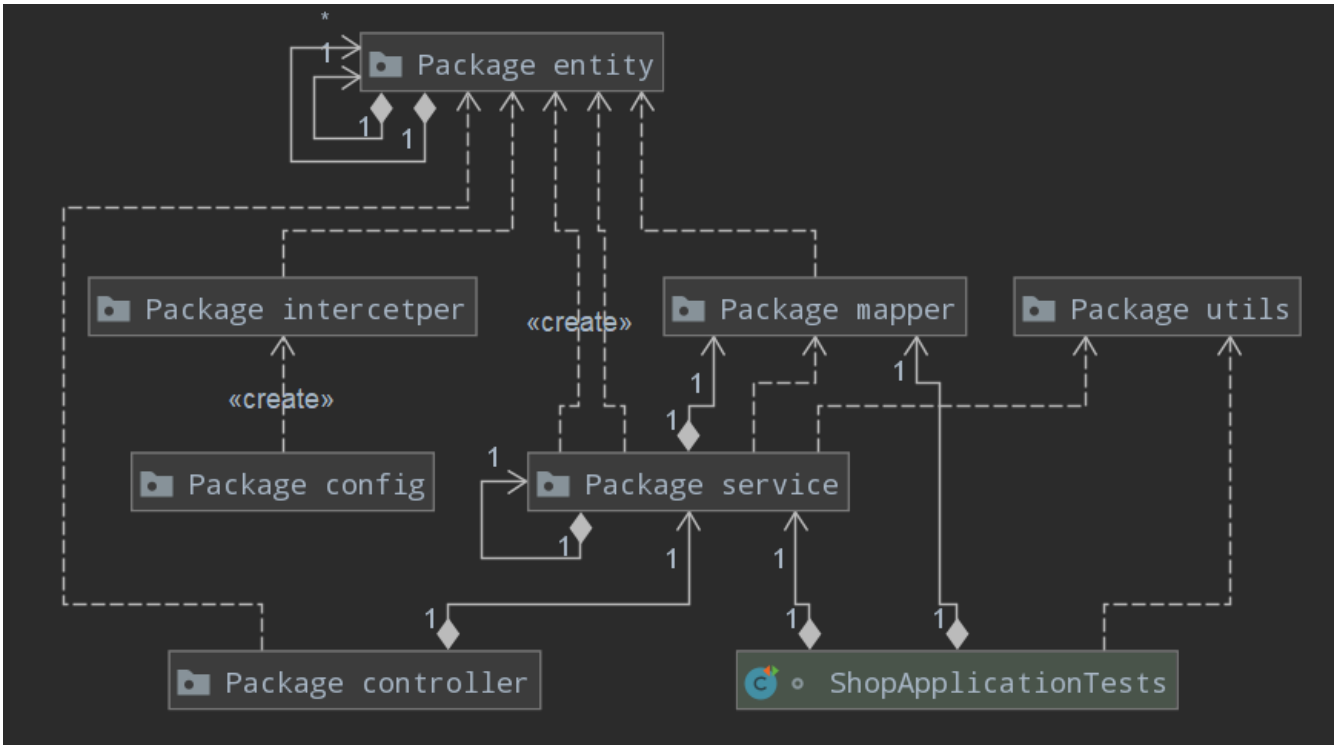
- /order/comfirm 确认订单
- /order/commitOrder 提交订单
- /order/list 显示用户订单列表（查询）
- /order/delete 删除订单

4.6 购物车管理模块

- /cart/add 添加购物车记录
- /cart/list 查询当前用户购物车列表
- /cart/update 更新购物车信息
- /cart/delete 删除购物车条目

5、详细设计与实现

采用SpringBoot+Mybatis框架设计，实现了entity层、mapper层、service层和controller层



5.1 Entity设计

entity层用于存放实体类，与数据库中属性值基本保持一致。

根据数据库建表的情况，数据库Table对应entity类，数据库属性对应类的成员变量，使用lombok自动生成get、set方法。

分别定义了用户实体类User、地址实体类Address、图书实体类Book、购物车实体类Cart、订单实体类Order、订单项目实体类OrderItem。

比如用户实体类定义如下：

```
/**
 * 用户实体类
 */
@Data
@TableName(value = "bs_user")
public class User {
    @TableId(type = IdType.AUTO)
    private int id;
    private String username;
    private String password;
    private String email;
    private String phone;
    private String company;
}
```

5.2 Mapper层

对数据库进行数据持久化操作，其方法语句直接针对数据库操作。

继承`BaseMapper<T>`接口，根据用户需求选择在对应的Mapper类中实现对数据库的操作。部分具体实现在mapper.xml文件里。

如下展示了CartMapper接口中实现的根据id查询购物车记录等。

```
@Repository
public interface CartMapper extends BaseMapper<Cart> {

    //根据用户id查询购物车
    @Select("SELECT\n" +
            "\tbsc.*, bsb.NAME AS bookName, bsb.img_url AS img_url,\n" +
            "\tbsb.new_price AS new_price\n" +
            "FROM\n" +
            "\tbs_cart bsc\n" +
            "LEFT JOIN bs_book bsb ON bsc.book_id = bsb.id\n" +
            "WHERE\n" +
            "\tbsc.user_id = #{userId}")
    List<CartVo> findCartListByUserId(int userId);

    //根据购物车ids查询购物车记录
    @Select({
            "<script>" +
                "SELECT\n" +
                "\tbsc.*, bsb.NAME AS bookName, bsb.img_url AS img_url,\n" +
                "\tbsb.new_price AS new_price\n" +
                "FROM\n" +
                "\tbs_cart bsc\n" +
                "LEFT JOIN bs_book bsb ON bsc.book_id = bsb.id\n" +
                "WHERE bsc.id in\n" +
                "<foreach item = 'item' collection = 'ids' open = '(' separator = ',' close
= ')'>"+
                "#{item}" +
                "</foreach>" +
            "</script>"})
    List<CartVo> findCartListByIds(@Param("ids") List<String> ids);
}
```

5.3 Service层

Service层存放业务逻辑处理，在接口的实现方法继承`ServiceImpl<M extends BaseMapper<T>, T> implements IService<T>`,导入mapper层接口以及实体类，存放需要的处理业务逻辑的方法。

如下为UserService类，其中包含判断用户名是否存在以及登录验证密码是否正确等方法：

```
@Service
public class UserService extends ServiceImpl<UserMapper, User> {

    @Autowired
    private UserMapper userMapper;
    /**
     * 验证用户是否存在
     * @param username
```

```
     * @return
     */
    public String checkUser(String username){
        QueryWrapper<User> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("username",username);
        User user = userMapper.selectOne(queryWrapper);
        if(user == null){
            return "101";
            //用户不存在，可以进行注册
        }
        else{
            return "102";
            //用户已存在，不可注册
        }
    }
    /**
     * 登录验证
     * @param loginUser
     * @param session
     * @return
     */
    public String loginCheck(User loginUser, HttpSession session){
        QueryWrapper<User> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("username",loginUser.getUsername());
        User user = userMapper.selectOne(queryWrapper);

        if(user == null){
            return "101";//用户不存在
        }
        else{
            //判断密码是否正确
            if(loginUser.getPassword().equals(user.getPassword())){
                session.setAttribute("user",user);
                return "100";//密码正确
            }
            else{
                return "102";//密码不正确
            }
        }
    }
}
```

5.4 Controller层

控制器，与前端进行交互。导入service层，controller通过接收前端传过来的参数进行业务操作，在返回一个指定的路径或者数据表。

比如BookController类的代码如下：

```
/**
 * 图书控制器
 */
@Controller
```

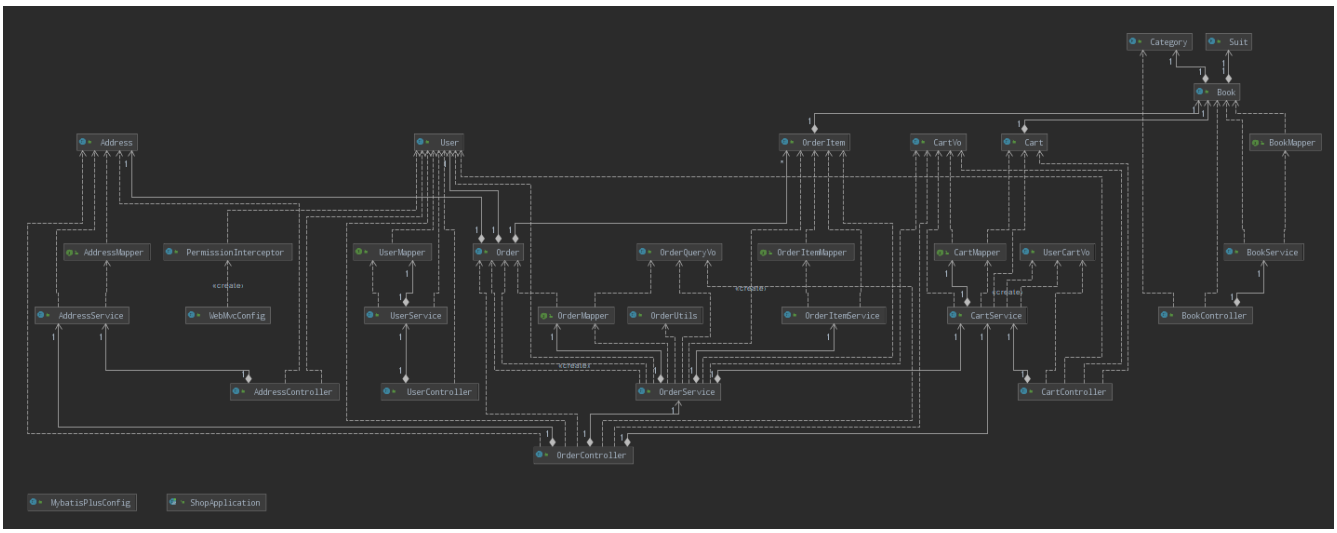
```java
@RequestMapping("/book")
public class BookController {

    @Autowired
    private BookService bookService;

    @RequestMapping("/index")
    public String index(){
        return "index";
    }
    /**
     * 获取图书信息
     */
    @RequestMapping("/getBookData")
    public String getBookData(Model model, int page, int category){
        IPage<Book> iPage = bookService.page(new Page<>(page,4), new QueryWrapper<Book>()
                .eq("category", Category.SELECTED));
        model.addAttribute("bookList",iPage.getRecords());
        model.addAttribute("pre", iPage.getCurrent() - 1);
        model.addAttribute("next", iPage.getCurrent() + 1);
        model.addAttribute("cur",iPage.getCurrent());
        model.addAttribute("last",iPage.getPages());
        model.addAttribute("category",category);
        return "bookData";
    }

    /**
     * 图书列表页
     */
    @RequestMapping("/bookList")
    public String bookList(String category, Model model){
        model.addAttribute("category",category);
        return "books_list";
    }
    /**
     * 获取图书列表数据
     */
    @RequestMapping("/getBookListData")
    public String getBookListData(String category, int page, int pageSize, Model model){
        IPage<Book> iPage = bookService.page(new Page<Book>(page, pageSize), new
QueryWrapper<Book>()
                .eq("category", Category.SELECTED));
        model.addAttribute("bookList",iPage.getRecords());
        model.addAttribute("pre", iPage.getCurrent() - 1);
        model.addAttribute("next", iPage.getCurrent() + 1);
        model.addAttribute("cur",iPage.getCurrent());
        model.addAttribute("pages",iPage.getPages());
        model.addAttribute("category",category);
        model.addAttribute("pageSize",pageSize);
        return "booksListData";
    }
    /**
     * 显示图书详情
```

```
     */
    @RequestMapping("/detail")
    public String detail(int id , Model model){
        Book book = bookService.getById(id);
        model.addAttribute("book", book);
        return "details";
    }
}
```

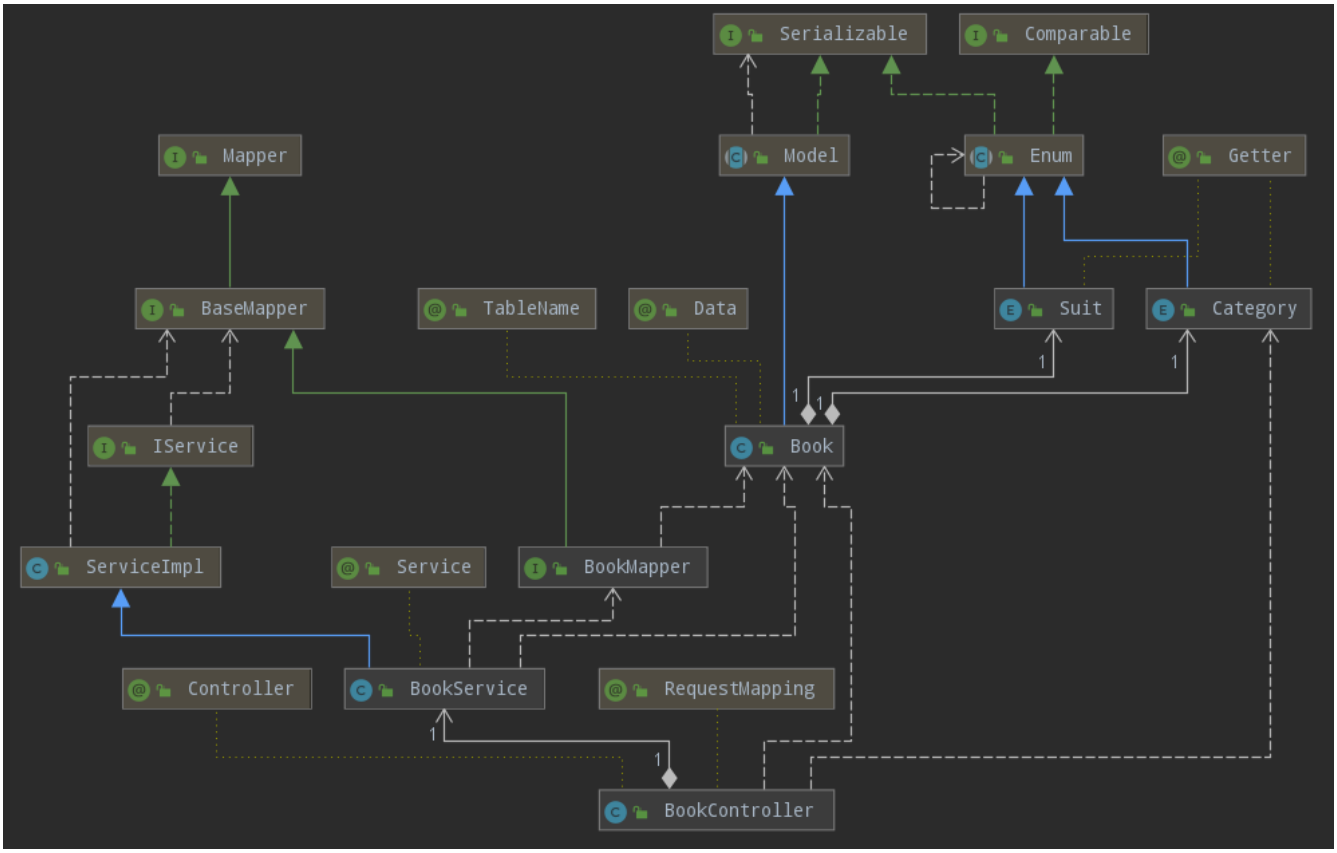
5.5 类图

(以下为代码实现后利用idea导出的类图)

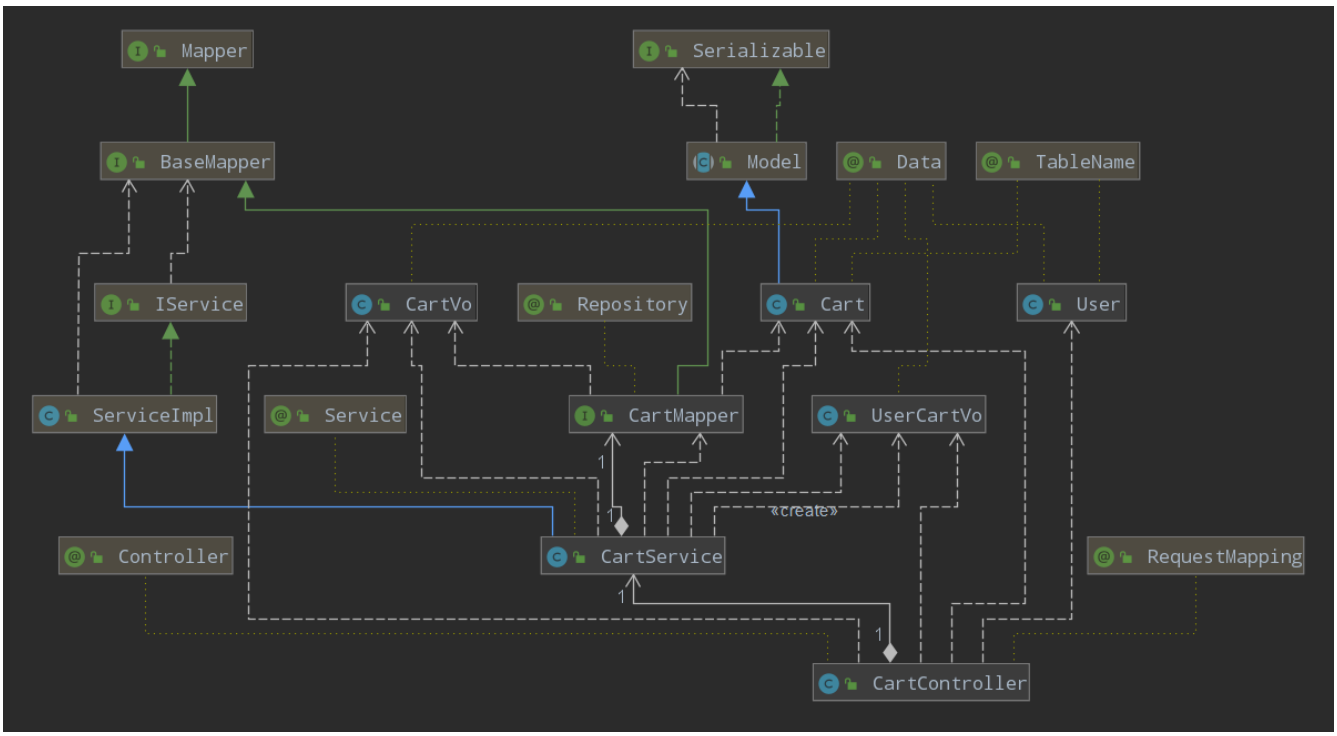
- 整体类图



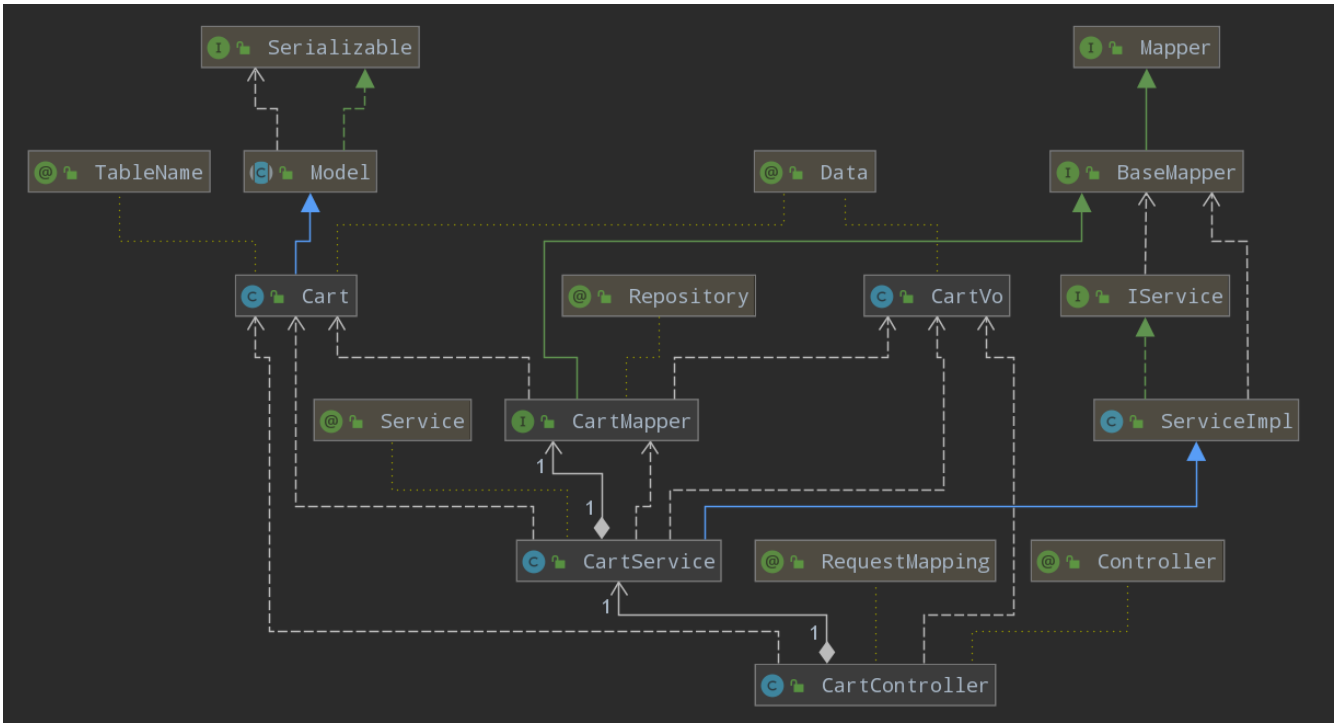
- 图书部分


Serializable
Comparable
Mapper
Model
Enum
Getter
BaseMapper
TableName
Data
Suit
Category
1
1
1
1
IService
Book
ServiceImpl
Service
BookMapper
Controller
BookService
RequestMapping
1
1
BookController

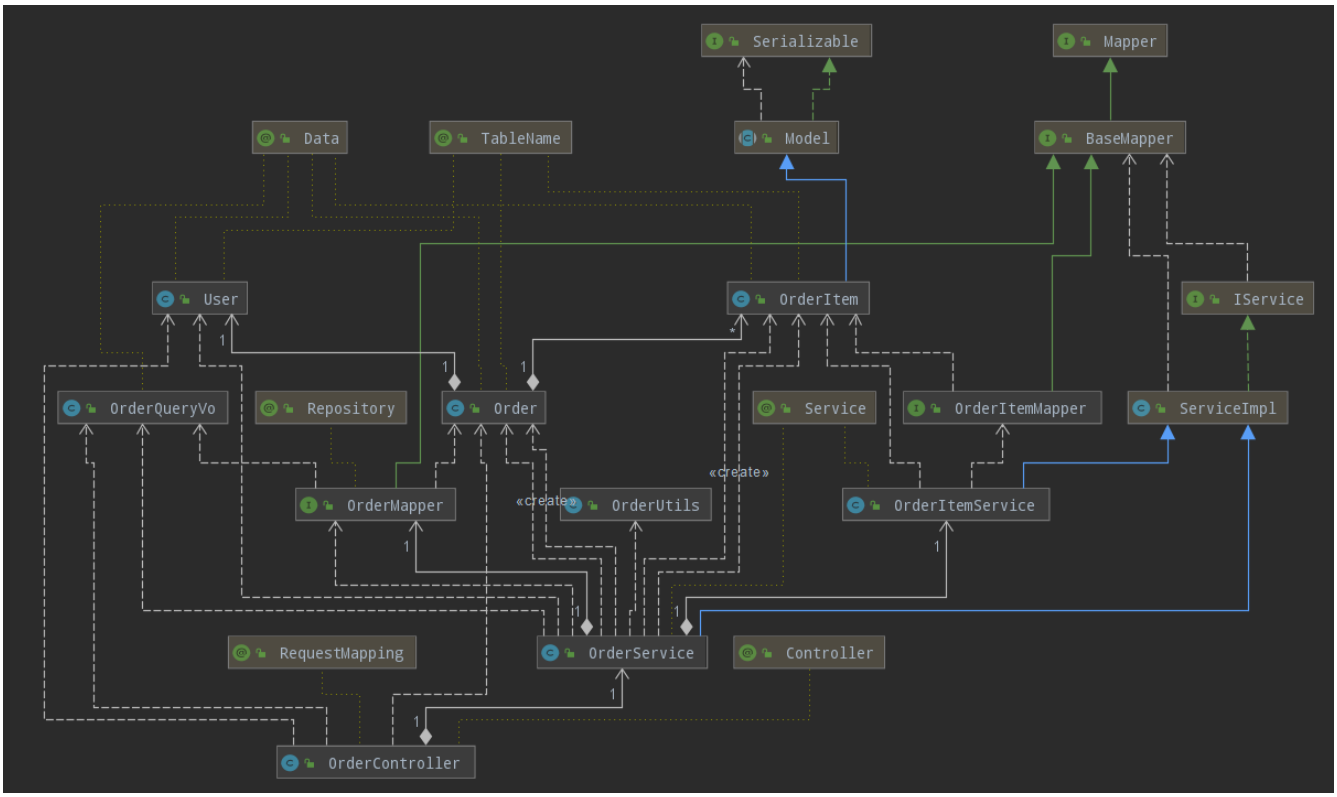
- 地址部分


Mapper
Serializable
BaseMapper
Model
Data
TableName
IService
CartVo
Repository
Cart
User
ServiceImpl
Service
CartMapper
UserCartVo
1
1
«create»
Controller
CartService
RequestMapping
1
1
CartController

- 购物车部分



- 订单部分



6、前端设计简要说明

由于对前端知识并未深入了解，本次前端网页中使用到的html文件、css文件以及js文件都有参考。下面对前端文件做出说明：

html放置在resources下的templates目录中

名称	说明
bookModal.html	包含用户注册与登录的模态框
carousel.html	包含显示在多个页面的轮播图
footer.html	包含显示在每个页面底部的信息
header.html	包含显示在每个页面上部的导航栏

名称	说明
index.html	商城首页
bookData.html	商城数据页，包含分页功能
books_list.html	商品列表，用于展示商品
bookListData.html	列表数据页，包含分页功能
detail.html	图书详情页面+图书推荐模块+添加至购物车
cart.html	购物车界面
comfirm_order.html	提交订单界面，包含添加地址模态框以及地址列表显示
order_list.html	订单列表界面，包含查询功能等
orderData.html	订单列表数据页，含分页功能

html文件中还引入了js和css文件，此处参考了一些教程，不再做具体阐述。

7、小结

本次通过自己动手实现系统后端，学习了Springboot+Mybatis的配置以及使用；通过查询资料完成前端的显示，对html、css以及js语言有了更进一步的了解，但本系统仅仅是一个简单的电子商城平台，希望今后继续深入学习，实现更多商品的展示以及连接第三方支付软件等功能，成为一个更完整的电子商城系统。