麦田守望者 (Bookstore) 项目软件文档

1. 项目概述 (Project Overview)

本项目是一个基于 Java Web 标准技术栈（JSP + Servlet + JDBC + MySQL）开发的 B/S 架构在线图书销售系统。系统采用经典的分层架构设计（MVC 模式变体），实现了从前台用户选购到后台管理员维护的完整电商业务流程。

- **核心技术**：Java SE, Servlet 3.0, JSP, JDBC, MySQL, HTML/CSS/JS (Bootstrap)。
- **架构模式**：Controller (Servlet) + Service (事务控制) + DAO (数据访问) + View (JSP)。
- **数据安全**：手动事务管理（Transaction Management），防止数据不一致。

---

2. 项目文件结构分析 (Project Structure Analysis)

项目的目录结构遵循标准的 Java Web 工程规范。

2.1 后端代码 (`src/com/bookstore/`)

包名 (Package)	描述	关键文件解析
`bean`	**实体类 (Model)** 对应数据库表结构，用于数据传输。	`UserInfo.java` (用户), `Book.java` (图书), `UserOrder.java` (订单), `CartItem.java` (购物车项), `UserPreference.java` (用户偏好用于推荐)。
`dao`	**数据访问接口** 定义数据库原子操作标准。	`baseDAO.java` (泛型DAO，封装通用的增删改查), `userDAO.java`, `bookDAO.java` 等接口定义。
`dao.Impl`	**数据访问实现** 编写具体的 SQL 语句。	`userDAOImpl.java`, `bookDAOImpl.java`, `adminDAOImpl.java`, `UserOrderDAOImpl.java` 等。
`service`	**业务逻辑接口** 定义高层业务功能。	`userService.java`, `bookService.java`, `UserOrderService.java` 等。
`service.Impl`	**业务逻辑实现** 核心层。负责**事务控制** (commit/rollback) 和复杂业务拼装。	`userServiceImpl.java` (级联删除逻辑), `UserOrderServiceImpl.java` (订单生成事务), `bookServiceImpl.java` (图书上下架)。
`controller`	**控制层 (Servlet)** 接收 HTTP 请求，调用 Service，转发视图。	`baseServlet.java` (利用反射分发 action 方法), `userServlet.java` (登录/注册), `cartServlet.java` (购物车), `adminServlet.java` (后台管理), `pageServlet.java` (首页展示)。

包名 (Package)	描述	关键文件解析
`utils`	**工具类**	`JDBCUtils.java` (数据库连接获取/关闭), `WebUtils.java` (将 Request 参数注入 Bean)。
`filter`	**过滤器**	`CharsetFilter.java` (处理中文乱码)。

2.2 前端资源 (`web/`)

目录/文件	描述
`admin/`	**后台管理页面**。核心文件`admin.jsp` 集成了图书管理、订单管理和用户管理的所有功能。
`user/`	**用户功能页面**。`login.jsp` (登录), `register.jsp` (注册), `cart.jsp` (购物车), `orderItems.jsp` (收银台), `detail.jsp` (商品详情), `profile.jsp` (个人中心)。
`static/`	**静态资源**。存放 CSS (`style.css`), JS (`bootstrap`), 图片 (`img/book/` 书籍封面)。
`index.jsp`	**入口页面**。通常直接转发到`pageServlet` 进行首页数据加载。
`front.jsp`	**前台首页**。展示图书列表、分类导航、搜索栏和推荐书籍。
`standard_template.jsp`	**开发规范模板**。定义了 JSP 页面的标准注释和结构。

3. 功能模块说明 (Functional Modules)

用户前台模块 (User Frontend)

1. **用户认证：**
    - 注册（支持防重名校验）。
    - 登录验证。
    - 安全注销。
    - 个人信息修改与偏好设置。
2. **图书浏览：**
    - **分页展示**：支持海量数据的分页显示。
    - **分类筛选**：根据图书类别（如“小说”、“计算机”）筛选。
    - **智能推荐**：基于用户历史点击偏好 (`UserPreference`) 推荐相关分类书籍。
    - **搜索**：基于书名或作者的模糊查询。
3. **购物车管理：**
    - 添加商品（自动识别是否已存在，数量累加）。
    - 修改数量（支持前端校验）。
    - 清空/删除条目。
4. **订单处理：**
    - **下单**：将购物车内容转化为订单，原子性扣减库存。
    - **支付**：模拟余额支付，扣除用户余额 (`balance`)。
    - **历史查询**：查看订单状态（未付款/待发货/已发货/已收货）。

管理员后台模块 (Admin Backend)

1. **图书管理**：
   - **CRUD**：上传封面、新增图书、修改信息。
   - **状态管理：软删除**（放入回收站，`stock`设为-1）与 **物理删除**（彻底从数据库移除）。
   - **回收站**：一键还原下架图书。
2. **用户管理**：
   - **列表**：查看所有用户及余额。
   - **运维操作**：用户充值、密码重置。
   - **级联删除**：删除用户前自动清理其关联的购物车、订单及偏好数据（包含业务规则校验）。
3. **订单管理**：
   - **发货**：将"待发货"状态的订单更新为"已发货"。
   - **监控**：查看全平台所有订单详情。

---

4. 主要功能的流程与时序图

4.1 用户下单核心流程 (Create Order)

此流程涉及多个表的操作（`user_order`, `order_item`, `book`, `cart_item`），必须在同一个事务中完成。

```
sequenceDiagram
    autonumber
    participant User as 用户 (JSP)
    participant Servlet as OrderServlet
    participant Service as UserOrderServiceImpl
    participant CartDAO as CartItemDAO
    participant OrderDAO as UserOrderDAO
    participant ItemDAO as OrderItemDAO
    participant BookDAO as BookDAO
    participant DB as MySQL

    User->>Servlet: 请求生成订单 (createOrder)
    Servlet->>Service: 调用 createOrder(user, cartItemIds)

    Note over Service: 1. 开启事务 (setAutoCommit(false))

    Service->>CartDAO: 获取购物车选中项
    CartDAO-->>Service: List

    loop 遍历购物车项
        Service->>BookDAO: 检查库存 & 销量更新
        alt 库存不足
            Service->>DB: 回滚事务 (Rollback)
            Service-->>Servlet: 抛出异常/返回错误
        end
    end

    Service->>OrderDAO: 2. 插入订单主表 (UserOrder)
    Service->>ItemDAO: 3. 批量插入订单项 (OrderItem)
    Service->>CartDAO: 4. 清空购物车对应项 (deleteBatch)
```

```
    alt 所有操作成功
        Service->>DB: 提交事务 (Commit)
        Service-->>Servlet: 返回 orderId
        Servlet->>User: 跳转至支付页面 (orderItems.jsp)
    else 发生异常
        Service->>DB: 回滚事务 (Rollback)
        Servlet->>User: 提示下单失败
    end
```

4.2 订单取消流程 (Cancel Order)
此功能允许用户或管理员取消未完成的订单，涉及状态更新和潜在的库存回滚（可选）。

```mermaid
sequenceDiagram
    autonumber
    participant User as 用户/管理员
    participant Servlet as OrderServlet
    participant Service as UserOrderServiceImpl
    participant OrderDAO as UserOrderDAO
    participant BookDAO as BookDAO
    participant DB as MySQL

    User->>Servlet: 请求取消订单 (cancelOrder)
    Servlet->>Service: 调用 cancelOrder(orderId)

    Note over Service: 1. 开启事务 (setAutoCommit(false))

    Service->>OrderDAO: 查询当前订单状态
    OrderDAO-->>Service: UserOrder (status)

    alt 订单状态不可取消 (如已发货)
        Service-->>Servlet: 返回失败 (-1)
        Servlet->>User: 提示“订单已锁定，无法取消”
    else 订单状态可取消 (未付款/待发货)
        Service->>OrderDAO: 更新状态为“已取消”

        opt [可选逻辑] 回滚库存
            Service->>BookDAO: 增加对应书籍库存 (restoreStock)
        end

        Service->>DB: 提交事务 (Commit)
        Service-->>Servlet: 返回成功 (1)
        Servlet->>User: 刷新列表，显示“已取消”
    end
```

4.3 书籍删除流程 (Book Deletion - Physical)
管理员在后台物理删除书籍。为了防止数据库外键报错，必须先清理购物车中引用该书的记录。

```mermaid
sequenceDiagram
    autonumber
    participant Admin as 管理员
    participant Servlet as AdminServlet
```

```
    participant Service as BookServiceImpl
    participant CartDAO as CartItemDAO
    participant BookDAO as BookDAO
    participant DB as MySQL

    Admin->>Servlet: 请求彻底删除书籍 (deleteBooks)
    Servlet->>Service: 调用 delete(bookIds)

    Note over Service: 1. 开启事务 (setAutoCommit(false))

    loop 遍历每个 bookId
        Note right of Service: 解决外键约束: cart_item.book_id
        Service->>CartDAO: 删除引用该书的购物车项 (deleteByBookId)
        Service->>BookDAO: 删除书籍本体 (deleteById)
    end

    alt 操作成功
        Service->>DB: 提交事务 (Commit)
        Service-->>Servlet: 返回成功
        Servlet->>Admin: 更新列表，Toast提示“删除成功”
    else 发生异常 (如书籍被订单引用)
        Service->>DB: 回滚事务 (Rollback)
        Servlet->>Admin: 提示“删除失败，可能存在关联订单”
    end
```

4.4 用户级联删除流程 (User Deletion - Cascading)

这是系统中风险最高的操作。删除用户必须清理其所有痕迹（订单、购物车、偏好），否则会产生脏数据或报错。

```mermaid
sequenceDiagram
    autonumber
    participant Admin as 管理员
    participant Servlet as AdminServlet
    participant Service as UserServiceImpl
    participant OrderDAO as UserOrderDAO
    participant ItemDAO as OrderItemDAO
    participant CartDAO as CartItemDAO
    participant PrefDAO as UserPreferenceDAO
    participant UserDAO as UserDAO
    participant DB as MySQL

    Admin->>Servlet: 请求删除用户 (deleteUser)
    Servlet->>Service: 调用 delete(ids)

    Note over Service: 1. 业务校验 & 开启事务
    Service->>OrderDAO: 检查是否有未完成订单 (statusCheck)

    alt 有未完成订单 (status 0 or 1)
        Service-->>Servlet: 返回 -1 (拒绝操作)
        Servlet->>Admin: 提示“用户有未完结交易，禁止删除”
    else 无未完成订单
        Service->>OrderDAO: 获取用户所有 order_id
        OrderDAO-->>Service: List orderIds
```

```
        opt 如果 orderIds 不为空
            Service->>ItemDAO: 批量删除订单项 (order_item)
        end

        Service->>OrderDAO: 删除订单主表 (user_order)
        Service->>CartDAO: 删除购物车 (cart_item)
        Service->>PrefDAO: 删除用户偏好 (user_preference)
        Service->>UserDAO: 删除用户本体 (user_info)

        Service->>DB: 提交事务 (Commit)
        Service-->>Servlet: 返回成功
    end

## 5. 重点方法详细解析

### 5.1 用户级联删除逻辑 (`userServiceImpl.delete`)
**位置**: `src/com/bookstore/service/Impl/userServiceImpl.java`

此方法是系统中逻辑最复杂的方法之一，展示了如何手动处理 JDBC 事务以及维护数据库的引用完整性。

* **功能**：批量删除用户，并清理该用户关联的所有数据（订单、购物车、偏好）。
* **安全机制与流程**：
    1. **业务锁（前置校验）**：首先调用 `userOrderDAO.statusCheck`。如果用户有“进行中”的订单（状态为 0-未付款 或 1-待发货），则回滚事务并返回 `-1`，拒绝删除操作，防止产生交易纠纷。
    2. **事务原子性**：使用 `connection.setAutoCommit(false)` 开启事务，确保以下所有删除步骤要么全部成功，要么全部回滚。
    3. **级联删除顺序**（严格遵守外键约束）：
        * **Step 1**: 删除订单项 (`order_item`)。
            * *细节*：先通过 `userOrderDAO.getOrderIds` 查出该用户所有订单 ID。
            * *防空判断*：检查 `order_ids` 列表是否为空，防止传给 SQL `IN` 子句空参数导致报错。
        * **Step 2**: 删除订单主表 (`user_order`)。
        * **Step 3**: 删除购物车项 (`cart_item`)。
        * **Step 4**: 删除用户偏好 (`user_preference`)。
        * **Step 5**: 最后删除用户本体 (`user_info`)。

### 5.2 动态反射参数封装 (`baseServlet` & `cartServlet`)
**位置**: `src/com/bookstore/controller/baseServlet.java`, `cartServlet.java`

系统未使用传统的 `request.getParameter("name")` 手动逐一赋值，而是利用 Java 反射机制实现了通用的请求分发和参数注入。

* **请求分发 (Dispatcher)**：
    在 `baseServlet` 中：
    ```java
    // 1. 获取请求参数 action 的值 (例如 "login")
    String action = request.getParameter("action");
    // 2. 利用反射获取当前 Servlet 子类中名为 login 的方法
    Method method = this.getClass().getDeclaredMethod(action,
HttpServletRequest.class, HttpServletResponse.class);
    // 3. 动态调用该方法
```
```

```
    method.invoke(this, request, response);
    ```
* **自动 Bean 注入**：
    在 `cartServlet.add` 等方法中，使用 `WebUtils.param2Bean(request, CartItem.class)`。
    * **关键约束**：前端 JSP/HTML 中的 `name` 属性（如 `book_id`）必须与后端 Java Bean 的属性名完全一致，否则反射机制无法找到对应的 Setter 方法进行注入（这正是导致 `ForeignKey` 报错的常见原因）。

5.3 智能推荐算法 (`bookServiceImpl.recommend`)
位置: `src/com/bookstore/service/Impl/bookServiceImpl.java`

实现了基于用户行为的简单推荐系统，根据用户是否登录提供不同的推荐策略。

* **逻辑解析**：
    1. **游客模式**：
        * 如果传入的 User 对象为 `null`，直接调用 `bookDAO.getSalesRank` 查询销量最高的前 4 本书。
    2. **会员模式**：
        * 从 `user_preference` 表中查询该用户点击/购买权重最高的分类 ID (`getTop1Category`)。
        * **冷启动处理**：如果找不到偏好记录（新用户），降级为游客模式（热销推荐）。
        * **分类推荐**：如果找到偏好（例如“计算机”类），则调用 `bookDAO.getBooksByCategory` 查询该分类下的书籍。
        * **补全逻辑**：如果该分类下的书籍不足 4 本（例如只有 2 本），系统会再次调用热销榜数据，补足剩余的展示位，确保前端页面布局整齐。

5.4 数据库连接管理规范
位置: `src/com/bookstore/utils/JDBCUtils.java` 及各 Service 实现类

为了防止 **数据库连接泄露 (Connection Leak)** 导致系统瘫痪，Service 层的所有方法都必须遵循严格的资源释放模式。

* **标准代码模板**：
    ```java
    Connection connection = null;
    try {
        // 1. 获取连接
        connection = JDBCUtils.getConnection();

        // 2. 开启事务 (针对增删改操作)
        connection.setAutoCommit(false);

        // 3. 执行 DAO 业务操作...

        // 4. 提交事务
        connection.commit();
    } catch (Exception e) {
        // 5. 异常回滚
        if (connection != null) {
            try { connection.rollback(); } catch (SQLException ex) { ex.printStackTrace(); }
        }
        e.printStackTrace();
```

```
        } finally {
            // 6. 【关键】必须显式关闭连接，将其归还给连接池或断开 TCP 连接
            JDBCUtils.close(connection, null);
        }
        ```
```
```