ChatGPT

简化版小米商城 JSP 项目设计

数据库设计

一个简化的商城系统通常需要包含用户、商品、订单等基本数据表，用以存储用户信息、商品信息以及订单交易信息 [1]。本项目使用 MySQL 数据库，设计了如下几张表：用户表、管理员表、商品表、订单表和订单项表（订单明细表）。这些表通过主外键关系关联，以满足商城的基本功能需求。以下是各数据表的结构设计和 **CREATE TABLE** SQL 建表语句：

- **用户表（users）**：存储前台注册用户的信息。例如用户ID、用户名、密码等。本项目简化起见，仅包含登录所需的信息（不考虑密码加密等安全需求）。

```
CREATE TABLE users (
 id       INT PRIMARY KEY AUTO_INCREMENT,
 username   VARCHAR(50) NOT NULL UNIQUE,
 password   VARCHAR(50) NOT NULL
);
```

- **管理员表（admins）**：存储后台管理员账户，用于后台登录管理。字段包括管理员ID、用户名和密码。（这里将管理员与普通用户分开存表，方便初学者理解权限区别，也可以通过在用户表增加角色字段实现。）

```
CREATE TABLE admins (
 id       INT PRIMARY KEY AUTO_INCREMENT,
 username   VARCHAR(50) NOT NULL UNIQUE,
 password   VARCHAR(50) NOT NULL
);
```

- **商品表（products）**：存储商品的基本信息，如商品ID、名称、价格、库存数量、描述等。此表中的数据用于前台商品浏览以及后台商品管理功能。

```
CREATE TABLE products (
 id       INT PRIMARY KEY AUTO_INCREMENT,
 name      VARCHAR(100) NOT NULL,
 price     DECIMAL(10,2) NOT NULL,
 stock     INT NOT NULL,
 description TEXT
);
```

- **订单表（orders）**：存储订单主信息，每个订单由一个用户创建，包含订单ID、下单用户ID、下单日期、订单总额和状态等字段。其中用户ID为外键关联到用户表，以确定订单归属哪个用户；状态字段可用于标记订单是否已发货等。

```
CREATE TABLE orders (
 id       INT PRIMARY KEY AUTO_INCREMENT,
 user_id    INT NOT NULL,
 order_date DATETIME NOT NULL,
 status     VARCHAR(20) DEFAULT '未发货',
 total      DECIMAL(10,2) NOT NULL,
 FOREIGN KEY (user_id) REFERENCES users(id)
);
```

- **订单项表（order_items）**：存储订单的明细条目。由于一个订单可以包含多种商品，每条订单项记录对应订单中的一个商品，包括订单项ID、所属订单ID、商品ID、购买数量、成交时单价等字段。order_items 表通过外键关联订单表和商品表，以保持数据一致性。

```
CREATE TABLE order_items (
 id       INT PRIMARY KEY AUTO_INCREMENT,
 order_id   INT NOT NULL,
 product_id INT NOT NULL,
 quantity   INT NOT NULL,
 price      DECIMAL(10,2) NOT NULL,
 FOREIGN KEY (order_id) REFERENCES orders(id),
 FOREIGN KEY (product_id) REFERENCES products(id)
);
```

以上数据库表设计满足项目基本需求。其中我们为了简化**购物车**功能，并未设计购物车临时表，而是在用户浏览期间将购物车数据保存在服务器会话中；用户提交订单时，再将购物车中的商品持久化到订单及订单项表中。各表的建立语句提供了必要的字段和关系，初学者可直接使用这些 SQL 在 MySQL 数据库中创建相应表结构。

> **说明**：实际商城还可能需要商品分类表、购物车表、支付/配送信息表等 [1]，但在本练习项目中我们为了降低复杂度，省略了这些扩展表，仅保留上述核心表。

项目目录结构

项目按照简化的 MVC 架构组织，采用 JSP 页面作为视图（View），Servlet 作为控制器（Controller），数据库操作由一个工具类集中管理。整体目录结构清晰明了，便于初学者理解：

- **src/** - 后端源代码目录
- **DBHelper.java** – 数据库操作辅助类，封装所有 MySQL 的连接和CRUD操作
- **servlet/** – 存放所有控制器 Servlet 类
  - **RegisterServlet.java** – 用户注册处理
  - **LoginServlet.java** – 用户登录处理
  - **AddProductServlet.java** – 添加商品处理（后台）
  - **EditProductServlet.java** – 编辑商品处理（后台）
  - **DeleteProductServlet.java** – 删除商品处理（后台）
  - **OrderSubmitServlet.java** – 提交订单处理
  - **OrderListServlet.java** – 用户订单列表展示控制
  - **AdminLoginServlet.java** – 管理员登录处理

    - **OrderManageServlet.java** – 管理员查看/发货订单处理（或拆分为专门的发货 Servlet）
    - (其他 Servlet：如购物车添加/移除，也可在需要时创建)

- **WebContent/** (或 **WebRoot/**) - 前端页面及静态资源

- **index.jsp** – 前台首页（商品列表浏览页面）
- **register.jsp** – 用户注册页面
- **login.jsp** – 用户登录页面
- **cart.jsp** – 购物车页面
- **order_list.jsp** – 用户订单列表页面
- **product_detail.jsp** – 商品详情页面（用户点击商品查看更多信息）

- **admin/** – 后台管理相关 JSP 页面子目录

    - **admin_login.jsp** – 管理员登录页面
    - **admin_product_list.jsp** – 商品管理页面（显示商品列表，提供编辑/删除入口）
    - **admin_add_product.jsp** – 添加新商品的表单页面
    - **admin_order_list.jsp** – 订单管理页面（显示所有订单并提供“发货”操作）

- **WEB-INF/** – Web应用配置文件目录

- **web.xml** – Web应用部署描述符（配置 Servlet 映射等）

上述结构将前台页面和后台页面分开存放，Java Servlet 类集中在`servlet`包下。这样便于区分不同角色的资源，理清项目层次。初学者可以根据功能逐步找到对应的 JSP 页面和 Servlet 类。例如，前台注册功能对应的是 register.jsp 和 RegisterServlet.java；后台商品管理功能对应 admin_product_list.jsp、admin_add_product.jsp 以及相关的 Servlet 等。

(注：本项目未使用额外的分层，如独立的 Service 或 DAO 层，所有数据库访问通过 DBHelper 直接在 Servlet 中调用，从而简化架构。实际开发中可根据需要引入更多分层来提高可维护性。)

页面说明

本项目包含前台用户使用的页面和后台管理员使用的页面，两部分界面相对独立。页面主要以 JSP 实现，用于展示数据和提供表单交互。下面按前台和后台分别说明各页面的作用：

前台页面

前台页面供商城用户浏览商品和进行购物操作使用，典型页面包括：首页、商品详情、注册/登录、购物车和订单查看等 [2]：

- **首页（商品列表页）** – 这里以 index.jsp 或 product_list.jsp 实现商城首页，用于向用户展示商品列表。页面会遍历数据库中的所有商品，将商品名称、价格、简要描述等信息以列表形式显示，用户可点击某一商品查看详情或“加入购物车”。

- **商品详情页（product_detail.jsp）** – 展示单个商品的详细信息。当用户在商品列表点击某商品时，跳转到此页面。页面显示该商品的图片（如果有）、详细描述、价格、库存等，并提供“加入购物车”的按钮。用户点击按钮即发送加入购物车请求。

- **用户注册页（register.jsp）** – 提供新用户注册表单。表单包含用户名、密码（确认密码）等字段。用户提交注册表单后，RegisterServlet 处理注册逻辑，将新用户信息写入数据库，然后给出注册成功提示或跳转到登录页。

- **用户登录页（login.jsp）** – 提供用户登录表单。包含用户名和密码输入项。用户提交后由 LoginServlet 校验用户名密码，成功则建立会话（Session）并跳转到首页或用户主页，失败则返回登录页显示错误信息。登录后用户才能执行购物车、下单等操作（本项目简化为登录后才允许提交订单）。

- **购物车页（cart.jsp）** – 展示当前用户购物车中的商品列表。用户每次点击“加入购物车”后，相应商品会被加入购物车（存储在Session）。此页面遍历 Session 中保存的购物车商品集合，显示商品名称、数量、小计价格等，让用户确认。页面上提供“提交订单”按钮进入下单流程，或“删除”按钮可移除某件商品。删除操作一般通过调用 RemoveFromCartServlet 移除 Session 中的该商品，再刷新购物车页。

- **用户订单列表页（order_list.jsp）** – 用户下单后，可在此页面查看自己的订单历史。页面通过 OrderListServlet 查询当前登录用户在数据库中的订单记录，包括订单编号、下单时间、总金额、状态（例如“未发货”或“已发货”）等。订单列表页面可以按订单逐行显示，并可能提供查看订单明细的功能（例如点击订单号跳转查看该订单的所有商品项，初学者也可直接在列表中附带显示商品项简要信息）。在简化实现中，下单成功后直接跳转到该页面显示新订单。

后台页面

后台页面供管理员登录后进行商城管理操作使用，包括商品管理和订单管理等功能页面：

- **管理员登录页（admin_login.jsp）** – 提供管理员账号登录的界面。管理员输入用户名和密码，由 AdminLoginServlet 进行验证。登录成功后进入后台管理界面（通常会跳转到管理首页或商品管理页）。

- **商品管理页（admin_product_list.jsp）** – 管理员登陆成功后首先看到的商品列表页面。页面通过 ProductListServlet（Admin） 从数据库读取所有商品信息，以表格形式列出，包括商品ID、名称、价格、库存等。 2 管理员可以在此页面执行管理操作，例如：点击“编辑”链接跳转到商品编辑页，或点击“删除”直接删除商品。页面还提供一个“添加商品”按钮，链接到添加商品页。

- **添加商品页（admin_add_product.jsp）** – 管理员添加新商品的表单页面。包含商品名称、价格、库存、描述等输入字段。管理员填写后提交表单，由 AddProductServlet 处理，将新商品数据插入数据库的商品表，然后重定向回商品管理页以查看新增结果。

- **编辑商品页（admin_edit_product.jsp）** – （可选）用于修改已有商品信息的页面。当管理员在商品管理页点击某商品的“编辑”时，EditProductServlet 拉取该商品当前信息，填充到此 JSP 表单中。管理员修改所需字段后提交，由 EditProductServlet 执行更新数据库操作，保存修改并跳转回商品管理页。为了简化，也可以不使用单独页面，直接在商品列表页实现快速编辑或在后台管理中省略商品编辑功能。

- **订单管理页（admin_order_list.jsp）** – 列出所有用户的订单信息，供管理员查看和处理发货。页面通过 OrderListAdminServlet 查询订单表中所有订单记录，显示订单号、下单用户、日期、总额和当前状态等。管理员可以在每条订单旁边看到“标记发货”或“确认发货”的操作按钮。点击后会发送请求到 ShipOrderServlet，将该订单状态更新为“已发货”，然后刷新订单列表页，使管理员能够看到状态变化。发货操作不需要额外页面，Servlet 处理完成后仍回到此列表。

后台页面通常还会有一个简单的导航或首页（如 admin_home.jsp），列出后台可管理的模块链接（商品管理、订单管理等）。在本项目中，可在管理员登录成功后直接跳转到商品管理页，导航在页面顶部以菜单链接形式提供。

Servlet 功能说明

为了实现上述前后台功能，每个重要的用户操作由相应的 Servlet 来处理业务逻辑。Servlet 充当控制器，接收页面表单或操作请求，调用数据库操作方法完成功能，然后将结果转发给 JSP 页面显示或重定向到相应页面。以下按前台和后台两类，列出主要的 Servlet 及其功能：

前台相关 Servlet

- **RegisterServlet（用户注册）** – 处理 register.jsp 提交的注册表单。Servlet 从请求中获取用户名、密码等参数，调用 `DBHelper.addUser(username, password)` 等方法将新用户数据插入数据库 [3]。如果注册成功，重定向到登录页面提醒用户登录；如果失败（例如用户名已存在），则转发回注册页并给出错误提示。

- **LoginServlet（用户登录）** – 处理 login.jsp 提交的登录表单。Servlet 调用 `DBHelper.validateUser(username, password)` 检查用户表中是否存在匹配的用户名和密码 [3]。验证通过则在 Session 中保存用户登录状态（例如保存用户ID或用户名），并重定向到商城首页或商品列表页；验证失败则转回登录页提示“用户名或密码错误”。

- **LogoutServlet（用户注销）** – 可选Servlet，实现用户退出登录的功能。将 Session 中的用户信息清除，然后重定向回首页或登录页。虽然需求未明确要求，此功能在实践中常用，初学者可考虑实现。

- **ProductListServlet（商品列表）** – 首页商品列表展示的控制器。Servlet 调用 `DBHelper.getAllProducts()` 查询商品表，获得商品列表数据，然后把数据存入请求属性并转发给 index.jsp（或 product_list.jsp）进行展示。用户无需提供输入，直接请求首页时该 Servlet 即被调用。这样可以确保页面不直接访问数据库逻辑，而是由 Servlet 准备数据。

- **ProductDetailServlet（商品详情）** – 当用户点击某商品查看详情时，此Servlet根据请求参数的商品ID调用 `DBHelper.getProductById(id)` 查询商品详情数据。然后将商品对象放入请求并转发到 product_detail.jsp 显示完整信息。初学者也可以不使用独立Servlet，在商品列表页通过超链接携带商品ID直接访问详情页并在 JSP 内使用 `<jsp:useBean>` 加载数据，但为遵循MVC，这里建议使用Servlet来控制流程。

- **AddToCartServlet（加入购物车）** – 处理用户点击“加入购物车”操作的请求。Servlet从请求中读取商品ID（以及数量，可默认为1），然后将该商品加入用户的购物车数据结构中。购物车可以用 **HttpSession** 来存储，比如保存为 `List<CartItem>` 或 `Map<商品ID, 数量>`。如果该商品此前已在购物车中，则增加数量；否则新增一条。处理完成后通常重定向回之前的页面或刷新购物车数量提示等。本项目中，可选择直接重定向到 cart.jsp 购物车页以查看结果。

- **RemoveFromCartServlet（移出购物车）** – 处理用户在购物车页面删除某项商品的操作。根据请求参数的商品ID，从 Session 中的购物车集合移除对应商品项。然后重定向回 cart.jsp，使更新后的购物车列表重新显示。

- **OrderSubmitServlet（提交订单）** – 用户在购物车页点击“提交订单”时触发。Servlet 首先从 Session 中取出当前用户ID和购物车商品列表，接着调用 `DBHelper.createOrder(userId, cartItems)` 将订单写入数据库。该方法内部需要在订单表插入一条订单记录，并在订单项表插入多条记录（每个商品对应一条） [4]。订单插入成功后，清空用户 Session 中的购物车数据，避免重复下单。最后，

Servlet 重定向用户到订单列表页面或订单成功页面。在本设计中，选择重定向到用户的order_list.jsp，让用户立即看到刚提交的订单。

- **OrderListServlet（用户订单列表）** – 用于显示当前登录用户的所有订单。Servlet 会检查 Session 中保存的用户身份，如果未登录则可重定向到登录页（本项目简单假设已登录）。然后调用 `DBHelper.getOrdersByUser(userId)` 查询该用户的订单列表以及每个订单的订单项（可在DAO中通过SQL联表查询或分别查询）。查询结果放入 request 属性，随后前往 order_list.jsp。页面通过遍历这些订单数据，将订单号、日期、金额、状态等展示给用户。如果需要查看订单详情，初学者可以在订单列表中直接列出订单项；或者实现点击订单号通过请求参数让 OrderDetailServlet 查询并显示该订单的详细商品列表。

后台相关 Servlet

- **AdminLoginServlet（管理员登录）** – 处理 admin_login.jsp 的表单提交。与用户登录类似，Servlet 获取管理员用户名和密码，调用例如 `DBHelper.validateAdmin(username, password)` 来验证管理员表。成功则在 Session 中标记管理员已登录状态（如设置 `"adminLogin"` 属性），并重定向到后台管理主页或商品管理页；失败则返回登录页并提示错误。这样确保只有管理员能访问后续的管理功能 Servlet （这些 Servlet 在实现时可以检查 Session 中是否有管理员标记，没有则重定向回登录页，简单的权限控制）。

- **AddProductServlet（添加商品）** – 处理管理员在 admin_add_product.jsp 提交的新商品表单。Servlet 获取表单中的商品名称、价格、库存数量、描述等参数，调用 `DBHelper.addProduct(...)` 执行数据库插入操作，将新商品记录加入 **products** 表 [3]。插入完成后，Servlet 重定向回商品管理列表页（admin_product_list.jsp），以便管理员能立即看到新增的商品。若发生数据库错误（例如商品名称重复等），则可以跳回添加页面并显示错误信息（本项目不特别处理输入验证和错误，默认插入成功）。

- **EditProductServlet（编辑商品）** – 处理管理员修改商品信息的请求。该 Servlet 可能有两种用途：

- 当管理员在商品列表页点击“编辑”时，Servlet根据商品ID查询其当前信息（`DBHelper.getProductById`），将商品数据附加到请求，再转发到 admin_edit_product.jsp 填充表单供管理员修改。
- 当管理员提交编辑后的表单时，Servlet 获取修改后的字段值，调用 `DBHelper.updateProduct(id, newName, newPrice, ...)` 执行 UPDATE 更新数据库。完成后重定向回商品管理页。

初学者也可以简化为只实现提交部分，而在商品列表页直接使用表单编辑，不另开页面。但分开页面和Servlet有助于结构清晰。

- **DeleteProductServlet（删除商品）** – 处理管理员在商品列表页执行删除操作的请求。请求会携带要删除的商品ID，Servlet 调用 `DBHelper.deleteProduct(id)` 执行数据库删除（DELETE）。然后重定向回 admin_product_list.jsp 刷新列表。需要注意，如果有订单已经包含该商品，删除商品可能导致外键问题或订单中出现“孤儿”商品。简化起见，可不考虑关联约束或者在数据库设计时未开启外键检查。实际应用中通常禁止删除有订单关联的商品，而采用下架标记等方式。

- **ProductListServlet（Admin）** – 用于后台商品列表页面的数据准备。管理员登录后访问 admin_product_list.jsp 时，可以由一个 Servlet 来查询所有商品数据。它类似前台的商品列表 Servlet，调用 `DBHelper.getAllProducts()` 获取全部商品集合，保存到请求属性，再转发给 JSP 展示。由于代码逻辑与前台列出所有商品类似，可以复用相同的方法，不同之处在于后台页面可以有操作列。如需区分也可写一个独立的 AdminProductListServlet。

- **OrderListAdminServlet（订单管理）** – 负责查询所有用户订单供后台查看。Servlet 调用 `DBHelper.getAllOrders()` 获取订单表的全部记录（可以包含关联的用户名等信息，或在 JSP 中通过订单的 user_id 去用户表查）。查询结果放入请求后，转发到 admin_order_list.jsp。管理员据此看到订单列表，包括订单号、用户、金额、状态等。

- **ShipOrderServlet（订单发货）** – 处理管理员点击“发货”操作。请求中会包含需要发货的订单ID，Servlet 调用 `DBHelper.updateOrderStatus(orderId, "已发货")` 将该订单状态字段更新为已发货。然后重定向回订单管理列表页。刷新后的列表会显示该订单状态改变，使管理员确认操作成功。类似地，如果需要也可以实现“取消订单”等其它管理操作。

  提示：为了保证后台管理的安全性，以上所有管理员相关Servlet在执行操作前都应验证当前会话是否有管理员登录标识。但由于本项目不考虑完整的安全措施，只要通过 AdminLoginServlet 登录后Session有记录即可。未登录的访问可以被简单地重定向回 admin_login.jsp。

数据库操作封装说明和示例代码

本项目将所有数据库的连接和读写操作封装在一个名为 **DBHelper** 的 Java 类中。采用这种集中式封装的目的在于：一方面简化数据库访问逻辑，让初学者不用为每个表编写重复的JDBC代码；另一方面，当未来需要更改数据库实现或优化时，只需修改这一处封装即可，维护方便 [3] 。通过 DBHelper 类提供的统一接口，业务代码（如各 Servlet）可以专注处理业务流程，将数据存取委托给DBHelper完成 [5] 。

DBHelper.java 主要功能包括：加载数据库驱动、建立数据库连接、执行SQL语句（查询或更新）以及封装常用的CRUD方法。例如，我们在该类中定义如下成员和方法：

```
public class DBHelper {
  // 数据库连接配置
  private static final String URL = "jdbc:mysql://localhost:3306/xiaomi_mall";
  private static final String USER = "root";
  private static final String PASS = "123456";

  static {
    try {
      // 加载MySQL JDBC驱动
      Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    }
  }

  // 获取数据库连接
  public static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASS);
  }

  // 示例方法：验证用户登录
  public static boolean validateUser(String username, String password) {
    String sql = "SELECT * FROM users WHERE username=? AND password=?";
    try (Connection conn = getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
      ps.setString(1, username);
```

```
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();
            return rs.next(); // 如有结果则表示用户名密码匹配
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    // 示例方法：添加新用户（注册）
    public static int addUser(String username, String password) {
        String sql = "INSERT INTO users(username, password) VALUES(?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setString(1, username);
            ps.setString(2, password);
            return ps.executeUpdate(); // 返回插入成功的行数 (1为成功)
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return 0;
    }

    // （类似地，还可以有商品、订单的增删改查方法，例如：）
    public static List<Product> getAllProducts() { ... }   // 查询所有商品
    public static int addProduct(String name, double price, int stock, String desc) { ... }
    public static Product getProductById(int id) { ... }
    public static int updateProduct(int id, String name, double price, int stock, String desc) { ... }
    public static int deleteProduct(int id) { ... }
    public static int createOrder(int userId, List<CartItem> items) { ... } // 创建订单和订单项
    public static List<Order> getOrdersByUser(int userId) { ... }
    public static List<Order> getAllOrders() { ... }
    public static int updateOrderStatus(int orderId, String status) { ... }
    // ... 其他所需数据库操作方法
}
```

上面的代码片段演示了 DBHelper 的部分实现。其中，静态代码块注册了 MySQL JDBC驱动；`getConnection()` 方法获取数据库连接；`validateUser()` 和 `addUser()` 分别封装了检查用户登录和插入新用户的逻辑，使用 `PreparedStatement` 防止SQL注入并简化参数设置。其他方法（如商品增删改查、订单创建等）也遵循类似模式：先获取连接，准备SQL语句，设置参数，执行SQL，然后处理结果（返回影响行数或结果对象）。

通过这样的封装，业务层（Servlet）在需要进行数据库操作时，只需调用相应的 DBHelper 静态方法即可，无需重复编写繁琐的 JDBC 代码。例如，在 **RegisterServlet** 中处理注册时，可以这样调用：

```
// 从请求获取用户名和密码
String username = request.getParameter("username");
String password = request.getParameter("password");
// 调用DBHelper添加用户
int result = DBHelper.addUser(username, password);
```

```
if (result > 0) {
    // 注册成功，重定向到登录页
    response.sendRedirect("login.jsp");
} else {
    // 注册失败，返回注册页并提示 (省略具体实现)
    request.setAttribute("error", "注册失败，请重试");
    request.getRequestDispatcher("register.jsp").forward(request, response);
}
```

类似地，**LoginServlet** 中可以用 `DBHelper.validateUser(u, p)` 返回的布尔值决定登录是否成功；**AddProductServlet** 可以调用 `DBHelper.addProduct(...)` 将新商品写入数据库；**OrderSubmitServlet** 可以调用 `DBHelper.createOrder(userId, cartItems)` 完成下单操作等。由于所有数据库访问逻辑都集中在 DBHelper 中，实现这些功能所需的SQL细节对 Servlet 来说是透明的。正如一篇经验总结所指出的，**将所有数据库操作集中在DBHelper类中，开发者可以将注意力集中在业务逻辑上，而不必纠结于繁琐的数据库细节** [5]。

需要注意的是，此种做法适用于简化的学习场景，在实际复杂项目中，可能会进一步拆分为DAO层、使用连接池和ORM框架等。本项目的 DBHelper 实现未考虑性能优化和安全（例如未使用连接池、密码明文保存等），目的是以最直接的方式展示 JSP+Servlet+JDBC 的开发流程，方便初学者快速上手练习。

总结

以上内容完整介绍了一个简化版小米商城 Java Web 项目的设计方案，包括数据库设计、项目结构、主要页面及对应功能、关键Servlet逻辑，以及数据库操作的封装实现 [2] [1] [3]。整个项目遵循简化的 MVC 模式：利用 JSP 页面展示数据和提供交互界面，Servlet 处理业务流程控制，所有数据存取通过一个集中管理的 DBHelper 类与 MySQL 数据库交互。这种结构清晰且易于搭建，非常适合初学者练习掌握 JSP/Servlet 开发、电商网站基本功能以及 JDBC数据库编程等知识点。在实际开发中，可以在此基础上逐步引入更完善的架构和优化，例如表单验证、错误处理、MVC 分层扩展、安全权限控制等，不断迭代完善商城系统功能。通过本项目的练习，相信读者能对 JSP+Servlet 网页开发有一个系统的认识，为后续学习更复杂的Java Web技术（如Spring框架等）打下坚实基础。

---

[1] [4] 商城数据库表设计介绍 - 王大军 - 博客园
https://www.cnblogs.com/FondWang/p/12550415.html

[2] 基于servlet+jsp 的 java 简单版商城项目_jsp servlet maven商城-CSDN博客
https://blog.csdn.net/CDWLX/article/details/97248530

[3] [5] DbHelper数据操作类 - 腾讯云开发者社区 - 腾讯云
https://cloud.tencent.com/developer/information/DbHelper%E6%95%B0%E6%8D%AE%E6%93%8D%E4%BD%9C%E7%B1%BB