

浙江工业大学

Web 应用开发上机作业

2022 级



实验题目： 作业 5 基于 MVC 的数据库应用

学生姓名： 李飞飞

学生学号： 2022

任课教师： 赵 小 敏

专业班级： 2022 软件工程（移动应用开发 1）

学 院： 计算机科学与技术学院

2024 年 5 月 4 日

目 录

1、购物车.....	3
1.1 题目要求:	3
1.2 需求分析	3
1.2.1 功能描述.....	3
1.2.2 业务处理流程及要求.....	3
1.2.3 输入信息.....	5
1.2.4 输出信息.....	5
1.3 开发步骤.....	6
1.4 运行结果.....	6
1.5 问题及解决方案.....	11
1.6 实验收获.....	11
1.7 源代码.....	11
2. 管理系统.....	17
2.1 题目要求	17
2.2 需求分析	17
2.2.1 功能描述.....	17
2.2.2 业务处理流程及要求.....	18
2.2.3 输入信息.....	19
2.2.4 输出信息.....	20
2.3 开发步骤	20
2.4 运行结果	22
2.5 问题及解决方案	29
2.6 实验收获	29
2.7 源代码	29

1、购物车

1.1 题目要求：

测试查询商品信息的例子,使之能正常运行，并增加如下功能：

(1) 分别增加添加、修改和删除功能；

(2) 增加一个商品管理页面，包含添加商品、查询商品、查看所有商品等功能，查询结果列表的后面增加修改和删除功能。

1.2 需求分析

1.2.1 功能描述

1. 查询商品

用户可以通过商品 ID 或商品名称等参数查询商品信息。系统会返回匹配的商品记录，展示商品的详细信息，包括商品 ID、商品名称、价格、库存量、描述等。

2. 添加商品

提供一个商品信息的输入表单，允许用户输入新的商品数据，包括商品 ID、商品名称、价格、库存量和描述等。添加成功后，系统将新商品记录插入数据库，并在商品管理页面中显示该商品。

3. 删除商品

在查询结果列表中，提供删除按钮，允许用户删除特定的商品。点击删除按钮后，系统会从数据库中移除该商品，并更新商品管理页面中的商品列表。

4. 商品管理页面

增加一个统一的商品管理页面，包含添加商品、查询商品、查看所有商品等功能。在页面中，用户可以通过不同的方式添加、查询、修改、删除商品。在查询结果列表的末尾，增加修改和删除功能，方便用户对商品信息进行编辑或删除。

1.2.2 业务处理流程及要求

1. 添加商品

业务流程

用户在商品管理页面点击“添加商品”按钮。

系统跳转到添加商品页面，提供商品信息的输入表单。

用户填写商品信息（商品 ID、名称、价格、库存、描述等）。

用户提交表单后，系统将商品数据插入数据库。

系统返回商品管理页面，显示添加的商品。

业务要求

所有必要字段必须填写，如商品 ID、名称、价格。

商品 ID 必须唯一，不能与数据库中现有的商品 ID 重复。

添加成功后，系统应在管理页面显示新添加的商品。

2. 查询商品

业务流程

用户在商品管理页面输入查询条件（商品 ID 或商品名称）。

系统根据查询条件搜索数据库，返回匹配的商品列表。

用户查看查询结果中的商品信息。

业务要求

查询条件可以是商品 ID 或商品名称。

如果没有匹配结果，系统应显示相应的提示信息。

查询结果列表应包含商品的基本信息，并提供修改和删除功能。

3. 删除商品

业务流程

用户在查询结果列表中点击“删除”按钮。

系统提示用户确认删除操作。

用户确认后，系统从数据库中删除指定的商品。

系统返回商品管理页面，更新后的商品列表不再包含已删除的商品。

业务要求

删除操作应提供确认提示，防止意外删除。

删除成功后，系统应更新商品管理页面，确保已删除的商品不再显示。

删除操作应在数据库中物理删除商品。

4. 商品管理页面

业务流程

用户进入商品管理页面，显示所有商品的列表。

用户可以选择添加商品、查询商品。

在查询结果列表的末尾，提供修改和删除按钮。

用户可以选择执行对应的操作。

业务要求

商品管理页面应提供添加商品、查询商品的功能。

查询结果列表应提供修改和删除功能。

用户可以通过商品管理页面完成商品的基本操作

1.2.3 输入信息

1. 添加商品

商品 ID: 唯一标识符, 必须唯一且非空。

商品名称: 商品的名称, 非空。

价格: 商品的价格, 必须是正数。

库存: 商品的库存数量, 必须是非负数。

描述: 商品的描述, 允许为空。

2. 查询商品

商品 ID: 用于精确查找商品。

商品名称: 用于模糊搜索。

3. 删除商品

商品 ID: 用于标识要删除的商品。

1.2.4 输出信息

1. 查询商品

商品 ID: 查询到的商品的唯一标识符。

商品名称: 查询到的商品名称。

价格: 查询到的商品价格。

库存: 查询到的商品库存数量。

描述: 查询到的商品描述。

操作按钮: 提供修改和删除操作的按钮。

2. 添加商品

添加成功/失败: 操作结果的反馈。

新商品信息: 新添加的商品的详细信息。

错误消息: 如果添加失败, 系统提供错误提示。

3. 删除商品

删除成功/失败: 操作结果的反馈。

错误消息: 如果删除失败, 系统提供错误提示。

更新后的商品列表: 删除后商品管理页面应不再显示已删除的商品。

1.3 开发步骤

步骤 1：定义数据库表结构

商品 ID、名称、价格、库存、描述等。数据库结构应符合系统的需求。

步骤 2：实现添加商品功能

创建页面和后端逻辑，允许用户添加商品信息。前端页面提供商品信息的输入表单，后端负责将数据插入数据库，并提供操作结果的反馈。

步骤 3：实现查询商品功能

建立前端页面和后端逻辑，允许用户查询商品信息。前端提供查询条件的输入，后端根据条件在数据库中查找商品，并返回查询结果。

步骤 4：实现修改商品功能

在查询结果中，添加修改功能，允许用户编辑商品信息。前端页面应展示当前商品的详细信息，后端负责处理更新请求，将修改后的数据保存到数据库。

步骤 5：实现删除商品功能

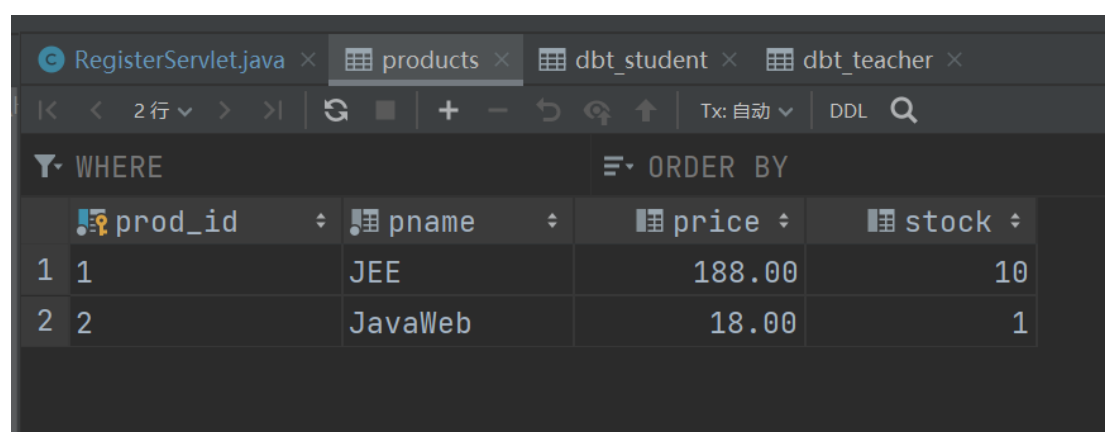
在查询结果中，增加删除功能，允许用户删除特定商品。后端处理删除请求，并从数据库中移除指定商品。

步骤 6：创建商品管理页面

设计一个综合的商品管理页面，包含添加商品、查询商品、查看所有商品等功能。在查询结果列表的末尾，提供修改和删除选项，方便用户对商品进行管理。

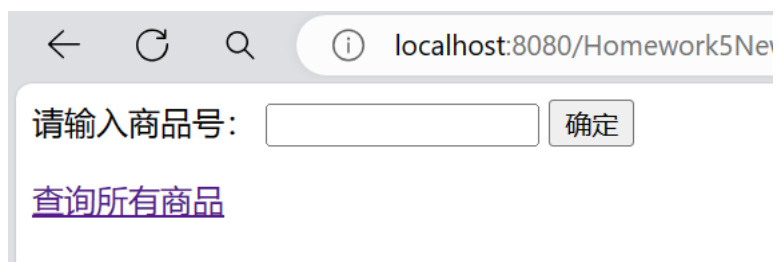
1.4 运行结果

- 数据库条目



	prod_id	pname	price	stock
1	1	JEE	188.00	10
2	2	JavaWeb	18.00	1

- 查询页面

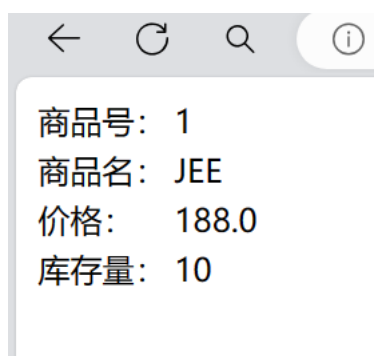


← ↻ 🔍 ⓘ localhost:8080/Homework5New

请输入商品号: 确定

[查询所有商品](#)

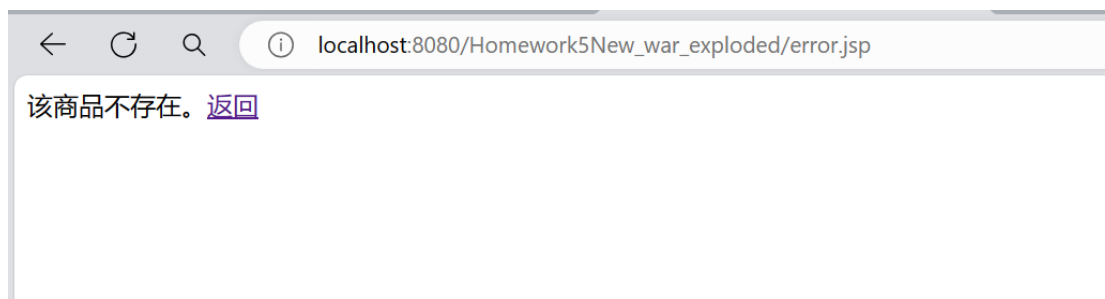
- 编号查询



← ↻ 🔍 ⓘ localhost:8080/Homework5New

商品号: 1
商品名: JEE
价格: 188.0
库存量: 10

- 找不到



← ↻ 🔍 ⓘ localhost:8080/Homework5New_war_exploded/error.jsp

该商品不存在。 [返回](#)

● 查询全部

localhost:8080/Homework5New_war_e

商品号	商品名	价格	数量	删除
1	JEE	188.0	10	删除
2	JavaWeb	18.0	1	删除

添加产品

产品 ID:

产品名称:

价格:

库存:

添加产品

- 添加 添加后会马上刷新页面展示出新商品

← ↻ 🔍 ⓘ localhost:8080/Homework5New_

商品号	商品名	价格	数量	删除
1	JEE	188.0	10	删除
2	JavaWeb	18.0	1	删除

添加产品

产品 ID:

产品名称:

价格:

库存:

添加产品

商品号	商品名	价格	数量	删除
1	JEE	188.0	10	删除
2	JavaWeb	18.0	1	删除
3	tesst	1.0	1	删除

添加产品

产品 ID:

产品名称:

价格:

库存:

添加产品

- 删除 删除后会马上刷新页面不展示删除的

商品号	商品名	价格	数量	删除
1	JEE	188.0	10	删除
2	JavaWeb	18.0	1	删除

添加产品

产品 ID:

产品名称:

价格:

库存:

添加产品

1.5 问题及解决方案

1.6 实验收获

1. 深入了解数据库操作

在实验中，我学习了如何设计数据库表结构，理解了数据表的关系，以及如何使用 SQL 语句进行查询、插入、更新、删除等操作。

2. 熟悉 Servlet 和 JSP 的交互

实验中，我体验了 Servlet 和 JSP 之间的交互过程。Servlet 用于处理 HTTP 请求、与数据库交互，并将数据传递给 JSP 页面，而 JSP 页面用于展示数据并与用户交互。这让我更好地理解 Web 应用的后端逻辑与前端展示之间的关系。

1.7 源代码

不展示 ppt 已给出的

● Product

```
package zjut.lff.homework5new.pojo;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/*
 * @ Author      : Li Feifei
 * @ Date        : Created in 9:38 2024/5/4
 * @ Description:
 */
//@SuppressWarnings("all")
@Data
public class Product {
    private String prodId;
    private String pname;
    private double price;
    private int stock;
}
```

● displayAllProduct.jsp

```
<%--
Created by IntelliJ IDEA.
User: Li Feifei
```

```

Date: 2024/5/4
Time: 9:47
--%>
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="zjut.lff.homework5new.pojo.Product" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>显示所有商品</title>
</head>
<body>
<table border="1">
    <tr>
        <td>商品号</td>
        <td>商品名</td>
        <td>价格</td>
        <td>数量</td>
        <td>删除</td>
    </tr>
    <c:forEach var="product" items="${sessionScope.productList}"
varStatus="status">
        <!--为奇数行和偶数行设置不同的背景颜色-->
        <c:if test="${status.count%2==0}">
            <tr style="background: #eeeeff">
        </c:if>
        <c:if test="${status.count%2!=0}">
            <tr style="background: #dedeff">
        </c:if>
        <!--用 EL 访问作用域变量的成员-->
        <td>${product.prodId}</td>
        <td>${product.pname}</td>
        <td>${product.price}</td>
        <td>${product.stock}</td>
        <td>
            <!-- 添加表单，提交删除请求 -->
            <form action="AddDeleteProductServlet" method="get"
onsubmit="return confirmDelete();">
                <!-- 将 productid 作为隐藏字段传递 -->
                <input type="hidden" name="productid"
value="${product.prodId}">
                <input type="submit" value="删除"
style="background-color: red; color: white;">
            </form>
        </td>
    </c:forEach>

```

```

        </tr>
    </c:forEach>
</table>
<h2>添加产品</h2>
<form action="AddDeleteProductServlet" method="post">
    <label for="prodId">产品 ID:</label>
    <input type="text" id="prodId" name="prodId" required><br><br>

    <label for="pname">产品名称:</label>
    <input type="text" id="pname" name="pname" required><br><br>

    <label for="price">价格:</label>
    <input type="number" id="price" name="price" step="0.01"
required><br><br>

    <label for="stock">库存:</label>
    <input type="number" id="stock" name="stock" required><br><br>

    <input type="submit" value="添加产品">
</form>
</body>
</html>

```

● AddDeleteProductServlet

```

package zjut.lff.homework5new.controller;
/*
 * @ Author      : Li Feifei
 * @ Date        : Created in 14:42 2024/5/8
 * @ Description: ${description}
 */

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

```

```

@WebServlet(name = "AddDeleteProductServlet", value =
"/AddDeleteProductServlet")
public class AddDeleteProductServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    Connection dbconn = null;

    public void init() {
        String driver = "com.mysql.cj.jdbc.Driver";
        String dburl =
"jdbc:mysql://localhost:3306/javaweb?serverTimezone=Asia/Shanghai&use
Unicode=true&characterEncoding=utf-8";
        String username = "root";
        String password = "254519";
        try {
            Class.forName(driver);
            dbconn = DriverManager.getConnection(dburl, username,
password);
            System.out.println(dburl + username + password);
        } catch (ClassNotFoundException e1) {
            System.out.println(e1);
        } catch (SQLException e2) {
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        String productid = request.getParameter("productid");

        try {
            // SQL 语句，用于根据 productid 删除商品
            String sql = "DELETE FROM products WHERE prod_id = ?";

            // 使用 PreparedStatement 来设置参数并执行 SQL
            PreparedStatement pstmt = dbconn.prepareStatement(sql);
            pstmt.setString(1, productid);

            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0)
                response.sendRedirect("queryproduct.do");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    @Override
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        // 从请求中获取产品信息
        String prodId = request.getParameter("prodId");
        String pname = request.getParameter("pname");
        double price =
            Double.parseDouble(request.getParameter("price"));
        int stock = Integer.parseInt(request.getParameter("stock"));
        PreparedStatement pstmt = null;
        try {
            // 插入 SQL 语句
            String sql = "INSERT INTO products (prod_id, pname, price,
stock) VALUES (?, ?, ?, ?)";

            // 使用 PreparedStatement 来防止 SQL 注入
            pstmt = dbconn.prepareStatement(sql);
            pstmt.setString(1, prodId);
            pstmt.setString(2, pname);
            pstmt.setDouble(3, price);
            pstmt.setInt(4, stock);

            // 执行插入操作
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0)
                response.sendRedirect("queryproduct.do");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // 在需要时关闭数据库连接和 PreparedStatement
            try {
                if (pstmt != null) pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```
public void destroy() {  
    try {  
        dbconn.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```


2. 管理系统

2.1 题目要求

(1) 用户注册 **register.jsp**。如果是教师，则可输入姓名、身份证号、工号、密码、学院、角色（系统管理员、校级管理员、院级管理员、普通教师）等信息，注册的信息写入教师表 **dbt_teacher** 中；如果是学生，则可输入姓名、身份证号、学号、密码、学院、专业、班级等信息，注册的信息存入学生表 **dbt_student**。要求密码采用 **SM3** 国密算法加密后存入数据库。要求教师的身份证号、工号和学生的身份证号、学生不能重复注册，否则跳转至 **failed.jsp** 提示“您输入的身份证号、工号或学号已被注册!”。

(2) 用户登录 **login.jsp**。选择用户类型(教师或学生)，输入登录名、密码，教师的登录名为工号，学生的登录名为学号。如果教师用户登录成功后，跳转至 **teacher.jsp** 显示该教师的姓名、身份证号、工号、密码、学院、角色等信息；如果是学生用户登录成功后，跳转至 **student.jsp** 显示该学生的姓名、身份证号、学号、密码、学院、专业、班级等信息；如果登录失败，则跳转至 **error.jsp**，显示“您的用户名或密码错误”，3 秒后自动跳转至登录页面 **login.jsp**。

2.2 需求分析

2.2.1 功能描述

(1) 用户注册 **register.jsp**

用户可以在注册页面注册为教师或学生，并填写对应的详细信息。

教师注册：教师可以输入姓名、身份证号、工号、密码、学院和角色。角色可以是“系统管理员”、“校级管理员”、“院级管理员”、“普通教师”之一。注册信息写入 **dbt_teacher** 表。

学生注册：学生可以输入姓名、身份证号、学号、密码、学院、专业和班级。注册信息写入 **dbt_student** 表。

要求

密码加密：在注册时，密码必须使用 **SM3** 国密算法加密后存入数据库，确保密码安全。

唯一性验证：教师的身份证号和工号、学生的身份证号和学号在数据库中不能重复。重复时，应跳转至 **failed.jsp**，提示“您输入的身份证号、工号或学号已被注册!”。

（2）用户登录 login.jsp

用户可以选择教师或学生身份登录，并输入对应的登录名和密码。

教师登录：教师的登录名是工号，用户需要输入工号和密码。登录成功后，跳转至 `teacher.jsp`，显示教师的详细信息，包括姓名、身份证号、工号、学院、角色等。

学生登录：学生的登录名是学号，用户需要输入学号和密码。登录成功后，跳转至 `student.jsp`，显示学生的详细信息，包括姓名、身份证号、学号、学院、专业、班级等。

登录失败：如果登录名或密码错误，跳转至 `error.jsp`，显示“您的用户名或密码错误”，并在 3 秒后自动跳转回 `login.jsp`。

要求

正确验证：确保教师的工号和学生的学号在数据库中存在，并与加密后的密码匹配。

错误处理：在登录失败时，应返回错误信息，并提供返回登录页面的机制。

2.2.2 业务处理流程及要求

（1）用户注册流程

流程

用户访问 `register.jsp` 页面，选择注册身份（教师或学生）。

根据所选身份，填写注册表单。教师填写姓名、身份证号、工号、密码、学院、角色；学生填写姓名、身份证号、学号、密码、学院、专业、班级。

用户提交表单后，系统验证输入信息，包括身份证号、工号或学号的唯一性。

如果验证通过，系统将注册信息写入对应的数据库表（`dbt_teacher` 或 `dbt_student`）。

如果验证失败，系统跳转至 `failed.jsp`，提示用户“您输入的身份证号、工号或学号已被注册！”。

要求

密码加密：在将密码存入数据库之前，必须使用 SM3 国密算法对密码进行加密。

唯一性验证：确保教师的身份证号、工号和学生的身份证号、学号在数据库中不重复。

注册成功后：系统可重定向至 `login.jsp` 或其他适当的页面。

注册失败时：系统跳转至 `failed.jsp`，并提供返回注册页面的选项。

（2）用户登录流程

流程

用户访问 `login.jsp` 页面，选择登录身份（教师或学生）。

根据所选身份，输入登录名和密码。教师使用工号作为登录名，学生使用学号作为登录名。

用户提交登录表单后，系统验证登录名和密码。

如果验证通过，系统根据身份跳转到不同的页面。教师跳转至 `teacher.jsp`，学生跳转至 `student.jsp`。

如果验证失败，系统跳转至 `error.jsp`，提示“您的用户名或密码错误”，并在 3 秒后自动返回 `login.jsp`。

要求

登录验证：确保登录名和加密后的密码与数据库中对应的记录匹配。

身份区分：根据用户选择的身份，进行不同的验证。教师验证工号，学生验证学号。

登录失败时：系统跳转至 `error.jsp`，并提供 3 秒后自动返回登录页面的机制。

登录成功后：教师跳转至 `teacher.jsp`，学生跳转至 `student.jsp`，并显示对应的详细信息。

2.2.3 输入信息

教师注册

姓名

身份证号

工号

密码

学院

角色（系统管理员、校级管理员、院系级管理员、普通教师之一）

学生注册

姓名

身份证号

学号

密码

学院

专业

班级

这些信息用于在数据库中创建新的教师或学生记录。系统在提交前会验证必要字段的完整性，并检查身份证号、工号、学号是否唯一。

用户登录

用户登录时，需选择身份类型（教师或学生）并提供以下信息：

教师登录

工号（作为登录名）

密码

学生登录

学号（作为登录名）

密码

在登录时，系统根据输入的登录名和密码在数据库中进行验证，确保用户信息匹配并允许登录。

2.2.4 输出信息

用户注册

- **注册成功/失败：**反馈注册操作的结果。
 - 如果成功，系统可以重定向到 **login.jsp** 或其他页面。
 - 如果失败，系统跳转至 **failed.jsp**，提示“您输入的身份证号、工号或学号已被注册！”
- **错误信息：**在注册失败的情况下，输出具体错误信息，提示用户重复或不合法的输入。

用户登录

- **登录成功/失败：**反馈登录操作的结果。
 - 如果成功，系统根据用户身份跳转至相应页面。教师跳转至 **teacher.jsp**，学生跳转至 **student.jsp**，并显示用户详细信息。
 - 如果失败，系统跳转至 **error.jsp**，提示“您的用户名或密码错误”，并在 3 秒后自动跳转回 **login.jsp**。
- **用户信息：**在成功登录后，输出用户的详细信息。
 - **教师信息：**显示教师的姓名、身份证号、工号、学院、角色等。
 - **学生信息：**显示学生的姓名、身份证号、学号、学院、专业、班级等。

2.3 开发步骤

步骤 1：设计数据库

确定系统所需的数据库表结构，确保表格能够存储教师和学生的注册信息。

确保字段设计满足要求，如教师和学生的唯一性约束、加密密码存储等。

步骤 2：实现用户注册功能

创建用户注册页面，允许用户输入所需信息进行注册。

实现后端处理逻辑，接收注册表单数据并验证必要字段。

在将密码存入数据库前，使用 SM3 加密。

添加唯一性验证，确保教师的身份证号、工号和学生的身份证号、学号不重复。

在成功注册后，重定向至适当页面；如果失败，跳转至 `failed.jsp` 并显示错误信息。

步骤 3：实现用户登录功能

创建用户登录页面，允许用户选择身份类型（教师或学生）并输入登录名和密码。

实现后端处理逻辑，接收登录表单数据并验证用户身份。

根据用户身份，在数据库中验证登录名和加密后的密码。

登录成功后，重定向至相应的页面，展示用户的详细信息；登录失败时，跳转至 `error.jsp` 并提供返回登录页面的机制。

步骤 4：构建用户信息展示页面

创建教师和学生的展示页面，分别用于显示登录成功后对应用户的信息。

确保页面能展示用户的详细信息，如教师的姓名、身份证号、工号、学院、角色，学生的姓名、身份证号、学号、学院、专业、班级等。

步骤 5：测试和验证

对系统进行全面测试，确保注册和登录功能正确。

验证唯一性约束，确保无法重复注册。

验证加密过程，确保密码在数据库中存储时已加密。

验证登录过程，确保验证逻辑正确，能正确识别有效和无效用户。

在登录失败时，验证错误处理和跳转机制。

步骤 6：完善错误处理和用户反馈

在各个功能模块中增加错误处理逻辑，确保系统能够处理各种异常情况。

提供清晰的用户反馈，帮助用户理解注册和登录过程中的错误原因。

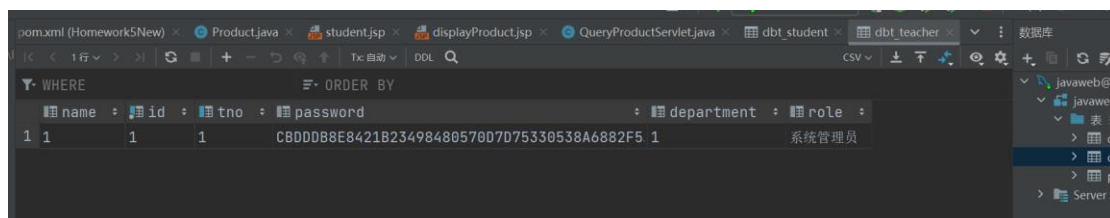
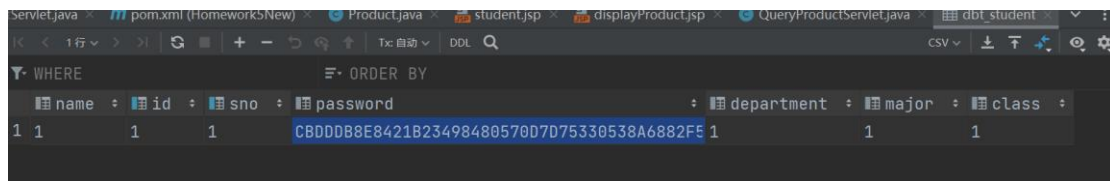
确保操作失败时，用户能够得到适当的提示和指导。

2.4 运行结果

- 首页



- 数据库



● 学生注册

登录

用户类型 学生 ☒ 教师 ☐

姓 名

请输入姓名

身份 证号

请输入身份证号

学 号

请输入学号

密 码

请输入密码

学 院

请输入学院

专 业

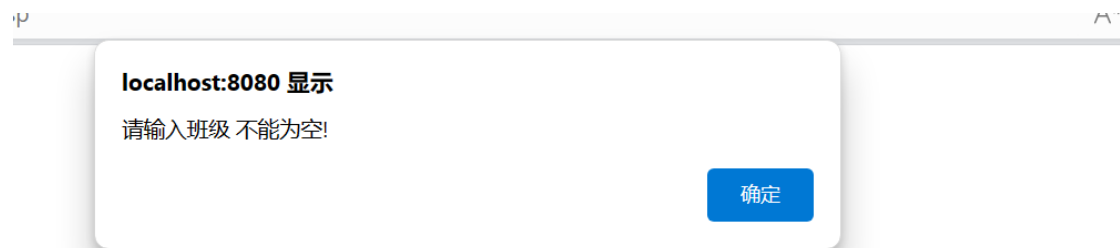
请输入专业

班 级

请输入班级

登录

- 信息缺失，其它也有，这里只展示一种情况



登录

用户类型 学生 ☒ 教师 ☐

姓 名

身份 证号

学 号

密 码

学 院

专 业

班 级

登录

- 注册成功后数据库新增条目

The screenshot shows a database management interface with a table containing user registration information. The table has columns for name, id, sno, password, department, major, and class. The second row of data is highlighted with a red line.

	name	id	sno	password	department	major	class
1	1	1	CBDDDB8E8421B23498480570D7D75330538A6882F5	1	1	1	
2	飞	111	2021057...	6DF72957D3B4D3C585B4F3FF3E04565FBE4750915F	计院	软件工程	软工2206

- 教师注册

登录

用户类型 学生 ☐ 教师 ☒

姓 名

身份 证号

工 号

密 码

学 院

角 色

登录

- 类别错误

oded/test2/register.jsp

localhost:8080 显示

角色类型必须是：系统管理员、校级管理员、院级管理员、普通教师之一！

确定

登录

用户类型 学生 ☐ 教师 ☒

姓 名 小敏

身份 证号 222

工 号 111

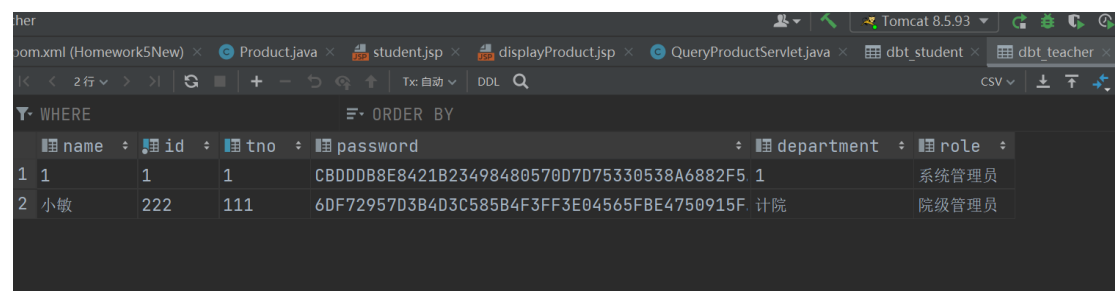
密 码 111

学 院 计院

角 色 1

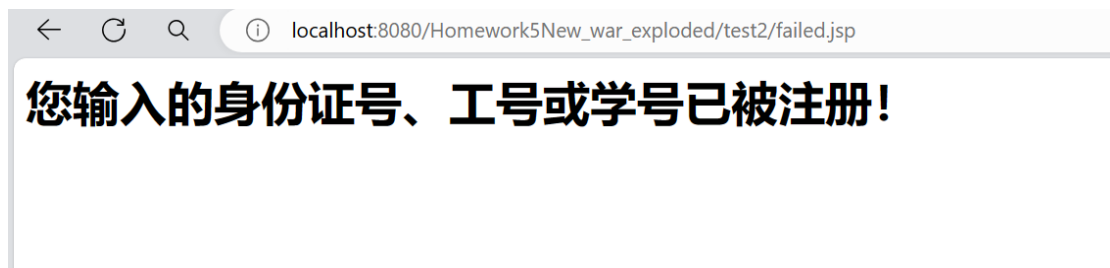
登录

- 注册后数据库



	name	id	tno	password	department	role
1	1	1	1	CBDDDB8E8421B23498480570D7D75330538A6882F5	1	系统管理员
2	小敏	222	111	6DF72957D3B4D3C585B4F3FF3E04565FBE4750915F	计院	院级管理员

- 注册冲突



- 学生登录

登录

用户类型 学生 ☒ 教师 ☐

学 号

密 码



- 教师登录

登录

用户类型 学生 ☐ 教师 ☒

工 号

密 码

登录

← ↻ 🔍 ⓘ localhost:8080/Homework5N

老师信息

姓名:	小敏
身份证号:	222
工号:	111
学院:	计院
角色:	院级管理员

- 登录失败

← ↻ 🔍 ⓘ localhost:8080/Homework5New_war_exploded/test2/error.jsp

您的用户名或密码错误

2.5 问题及解决方案

- 密码加密

引入依赖

```
<!-- 加密 -->
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcprov-jdk15on</artifactId>
  <version>1.68</version>
</dependency>
```

2.6 实验收获

1. 深入理解 Web 应用的开发流程

本实验涉及数据库设计、后端逻辑实现、前端页面设计、数据验证和用户反馈等各个环节，使我对完整的 Web 应用开发流程有了更加深入的理解。

2. 学习了数据库操作

在实验中，我学会了如何设计数据库表结构，并实现了基本的 CRUD 操作（创建、读取、更新、删除）。我了解到如何确保数据的唯一性，如何对密码进行加密存储，以及如何通过 SQL 查询和操作数据库。

3. 学会了错误处理和用户反馈

实验中，我遇到了一些错误和异常情况，这促使我学习了如何在 Web 应用中进行错误处理，并向用户提供适当的反馈。在注册和登录过程中，我学会了如何根据不同情况显示合适的错误消息，并提供用户友好的操作提示。

2.7 源代码

代码较多，放附件里