# 实验一 Servlet的使用

## 一、实验目的

熟悉Servlet的开发。

## 二、实验要求

1、掌握HttpServlet的使用

2、ServletConfig和ServletContext对象的使用

3、掌握HttpServletRequest的使用

4、掌握HttpServletResponse的使用

5、掌握Cookie的使用

6、掌握Session的使用

## 三、实验内容

1、编写一个servlet，实现统计网站被访问次数的功能。

2、编写Servlet，测试获取客户端表单中的文本框、复选框的信息

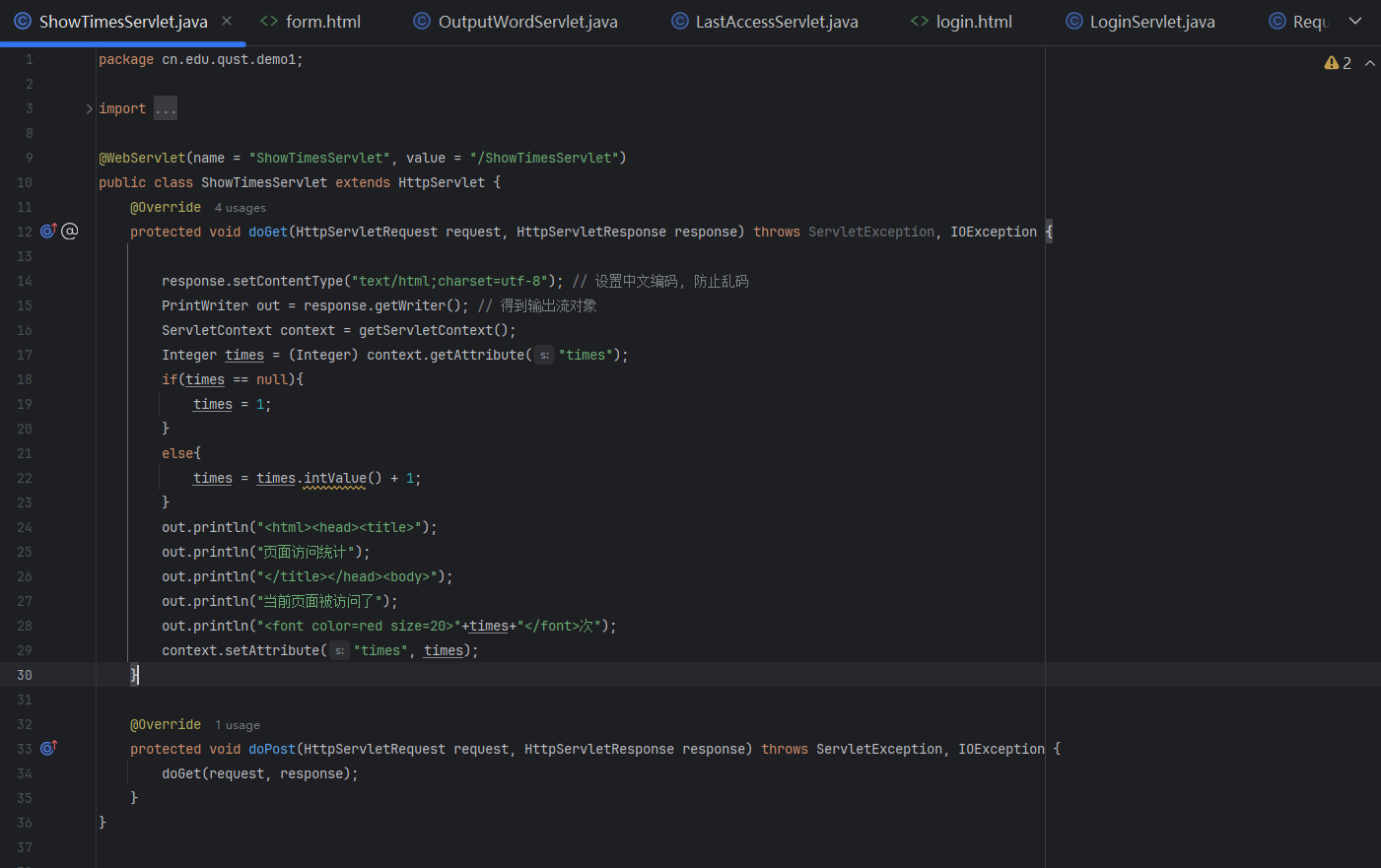
3、用Servlet向客户端输出一个二进制文件（例如一个Word文件）

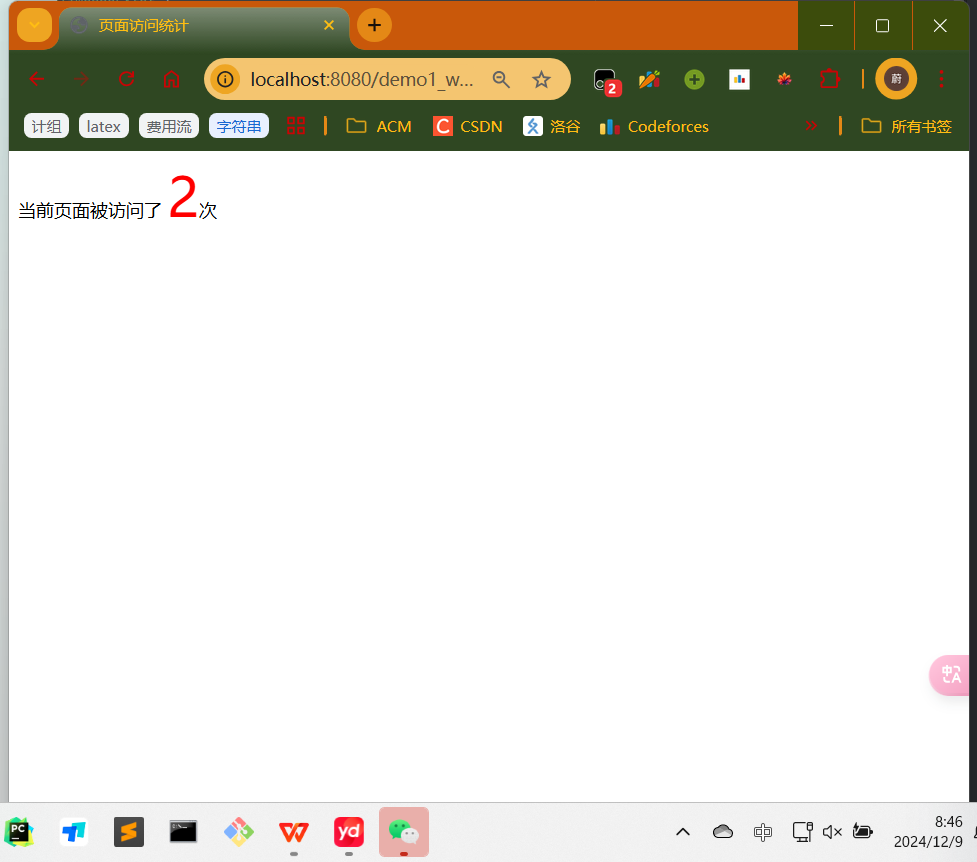
4、显示用户上次访问时间

5、编写一个用户登录页面，用Servlet判断用户名密码是否正确（假定正确的用户名是admin，密码是123），正确则用session记住登录用户名，跳转到一个欢迎页面，错误则给出提示

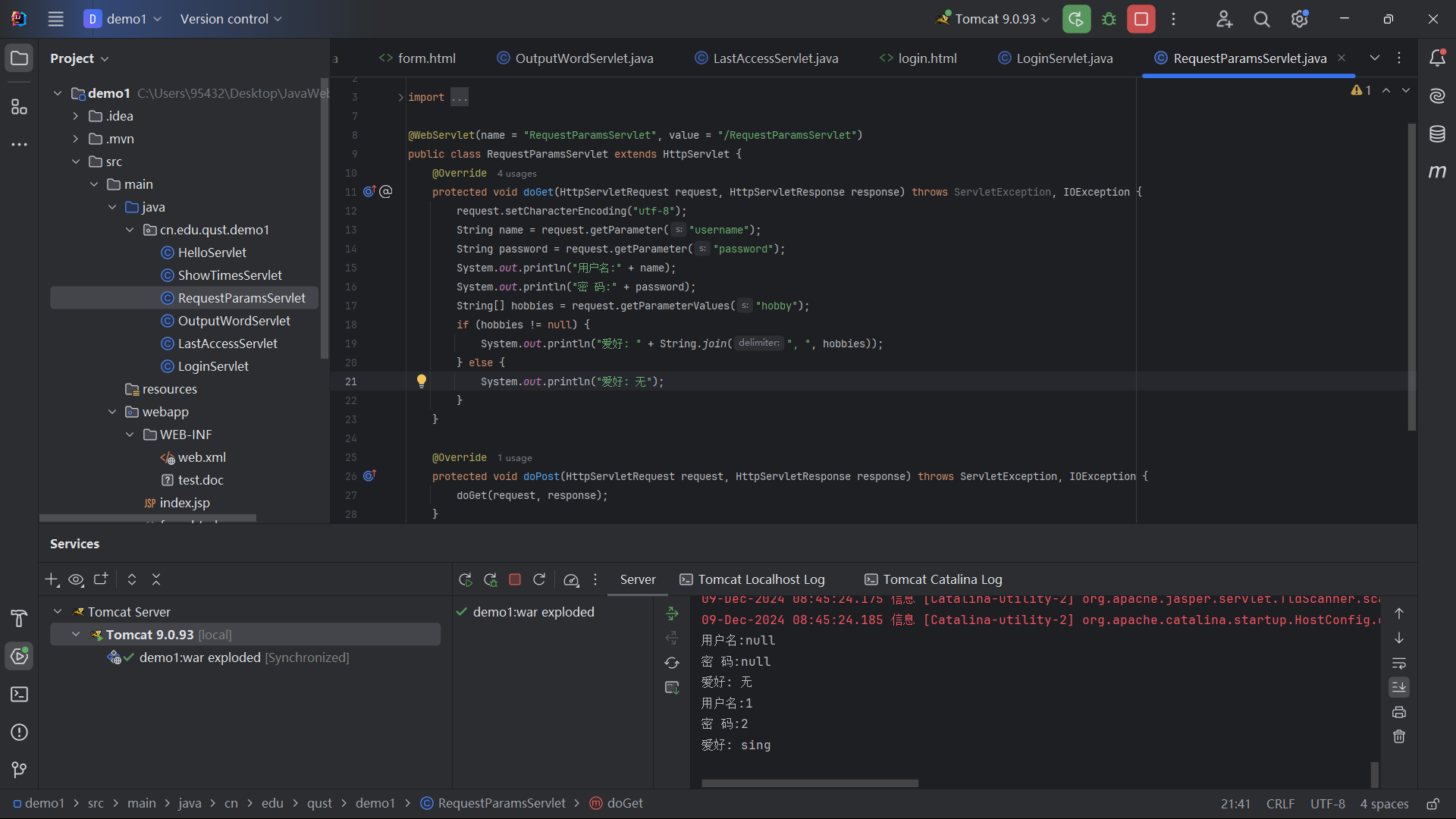
## 四、实验步骤

### 1、**编写一个servlet，实现统计网站被访问次数的功能。**



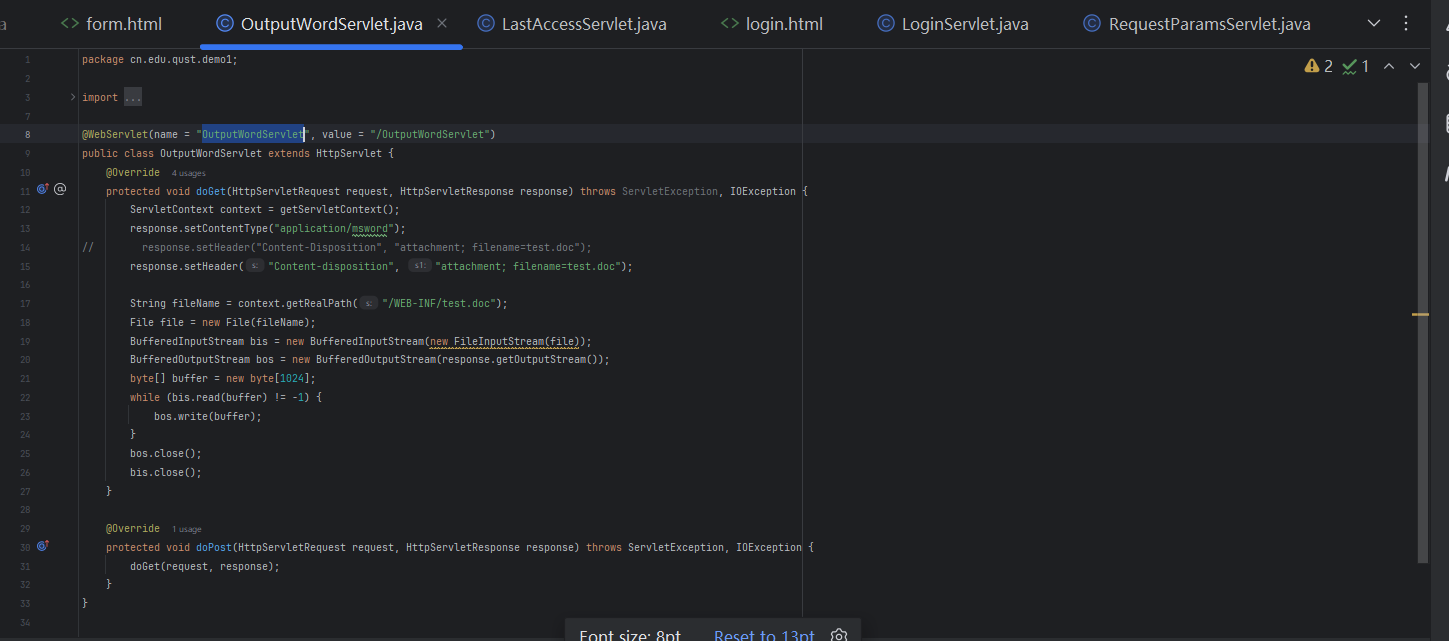


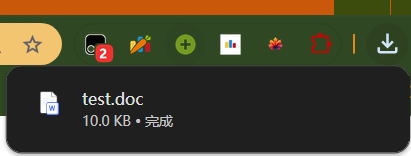
### 2、编写Servlet，测试获取客户端表单中的文本框、复选框的信息



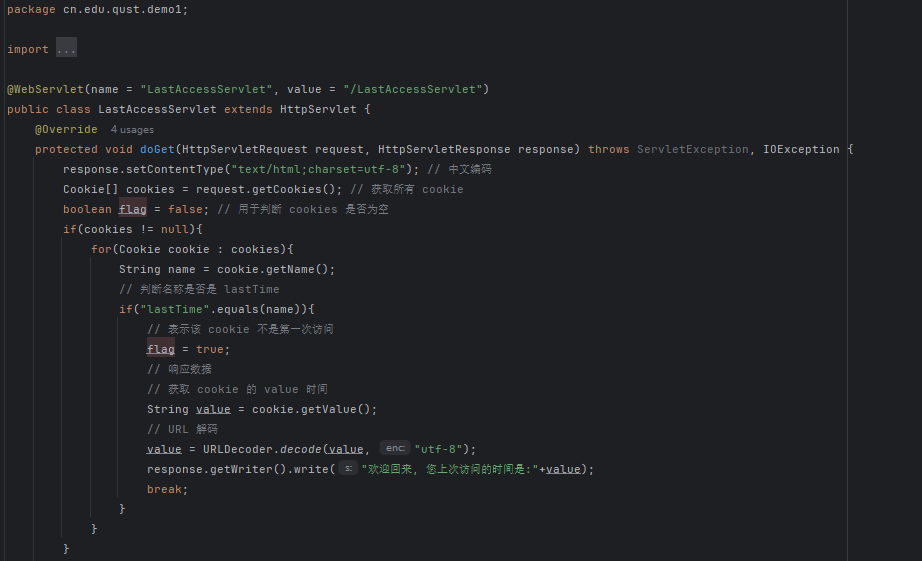


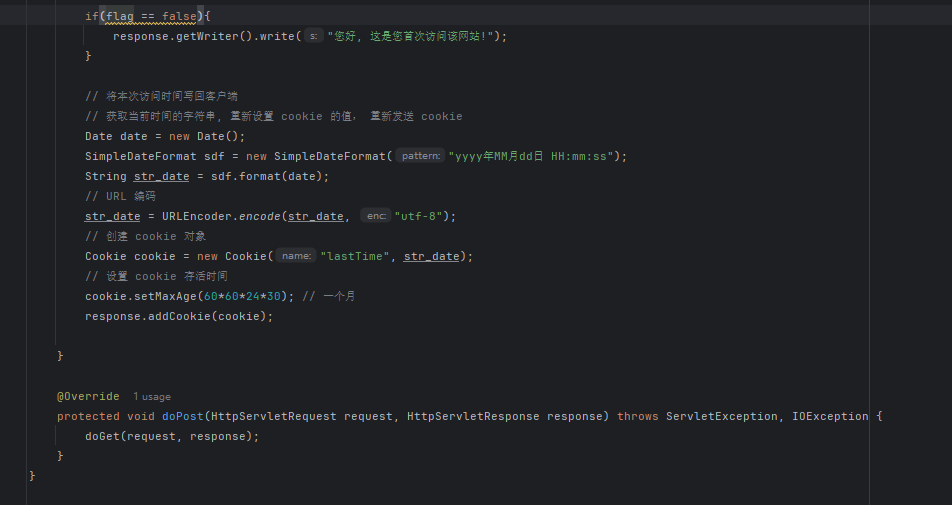
### **3、**用Servlet向客户端输出一个二进制文件（例如一个Word文件）

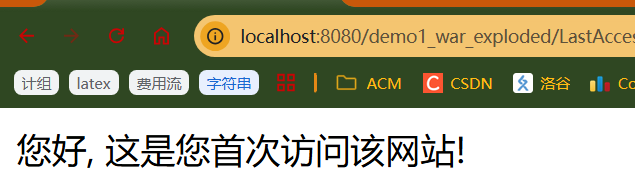




### 4、显示用户上次访问时间

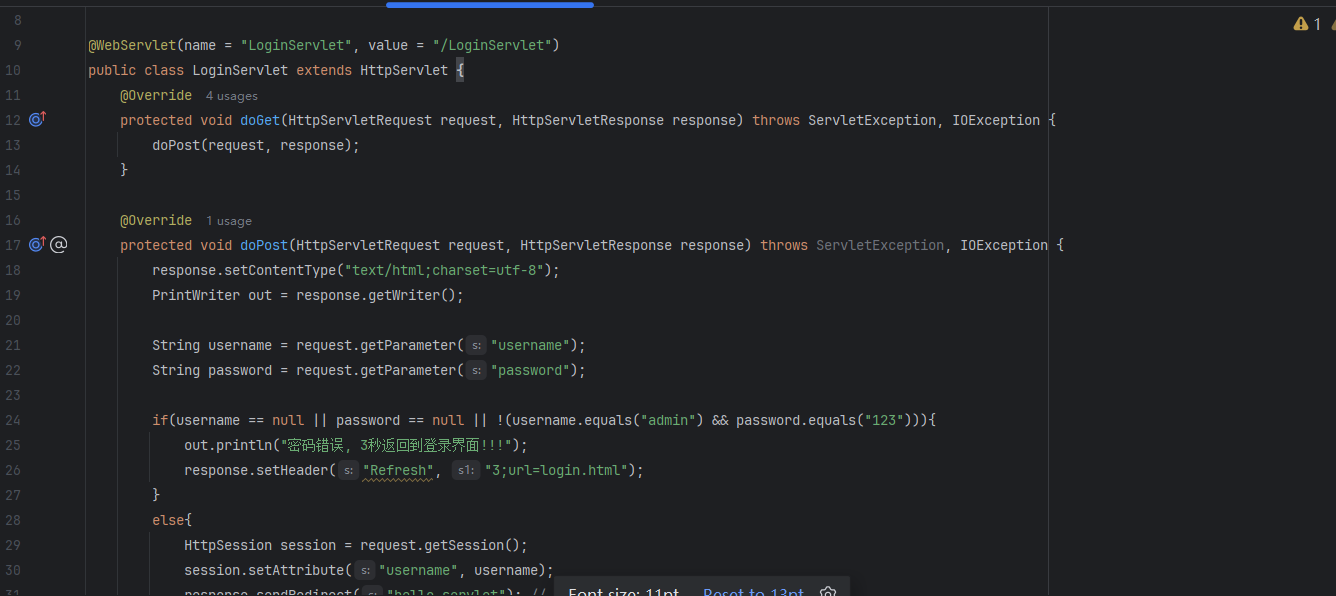








### 5、编写一个用户登录页面，用Servlet判断用户名密码是否正确（假定正确的用户名是admin，密码是123），正确则用session记住登录用户名，跳转到一个欢迎页面，错误则给出提示



登录成功 ：



登录失败 ：



## 五、实验总结

（1）通过本次实验，我全面掌握了 HttpServlet 的使用方法，包括其生命周期（init、service、destroy）的理解，以及如何在 doGet 和 doPost 方法中实现对客户端请求和响应的处理。

（2）熟悉了 ServletConfig 和 ServletContext 的使用方法，学会了 ServletConfig 获取初始化参数的技巧，以及通过 ServletContext 实现多个 Servlet 共享全局信息的能力，如网站访问次数的统计功能。

（3）通过实验实践了 HttpServletRequest 的使用，能够熟练获取客户端传递的表单数据（如文本框和复选框），并能够读取请求头信息和客户端 IP 地址等，进一步加深了对客户端请求的理解。

（4）掌握了 HttpServletResponse 的核心功能，包括设置响应头、返回 HTML 页面内容，以及通过 ServletOutputStream 实现二进制文件的输出，例如向客户端返回 Word 文件的下载功能。

（5）通过实验深入理解了 Cookie 的工作原理，并通过实践实现了使用 Cookie 保存用户上次访问时间的功能，加深了对客户端持久化数据存储的应用场景的认识。

（6）学会了利用 HttpSession 保存用户会话信息，成功实现了用户登录功能，能够根据登录结果进行页面跳转与提示信息展示，熟练掌握了 Session 在用户认证和状态管理中的实际应用。

（7）本次实验让我系统学习了 Servlet 的基本开发流程，同时熟悉了 Cookie 和 Session 的应用场景，为后续学习更高层次的 Java Web 技术（如 JSP、Spring MVC 框架）提供了实践基础和理论支持。

# 实验二 JSP+Servlet+JavaBean技术的使用

## 一、实验目的

1、掌握mysql数据库的使用

2、掌握JDBC技术的使用

3、掌握MVC模式的概念

4、掌握JSP+Servlet+JavaBean技术的使用

5、掌握三层架构的使用

6、熟悉EL、JSTL的使用

## 二、实验要求

1、熟悉Mysql数据库的安装配置及第三方客户端工具navicat的使用

2、掌握JDBC相关API（Connection、Statement、ResultSet等）的使用

3、掌握JSP+Servlet+JavaBean的开发模式

4、熟悉EL、JSTL的使用

## 三、实验内容

实现对学生表（student）的增删改查等基本操作，表结构如下：

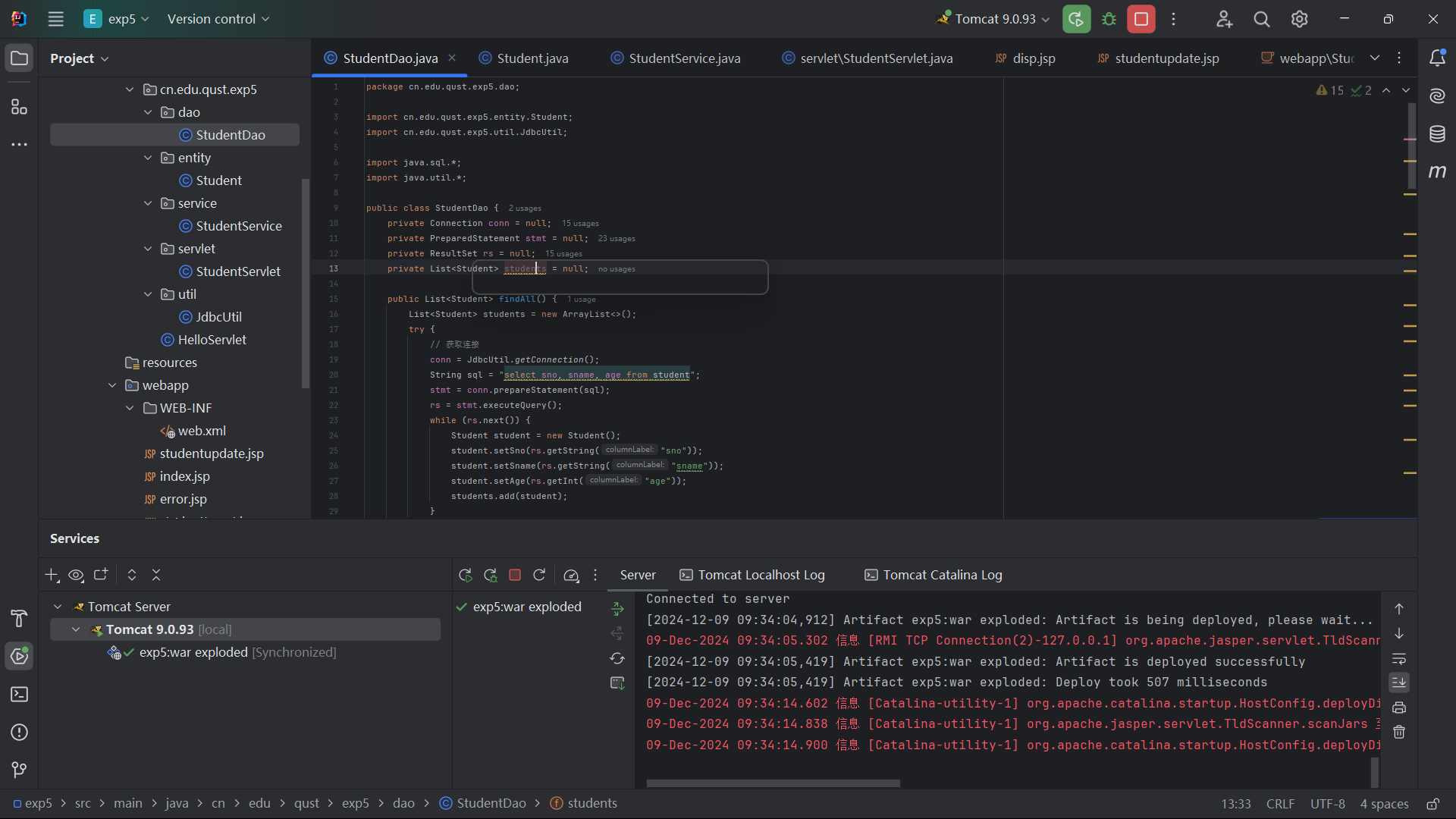
表名：student

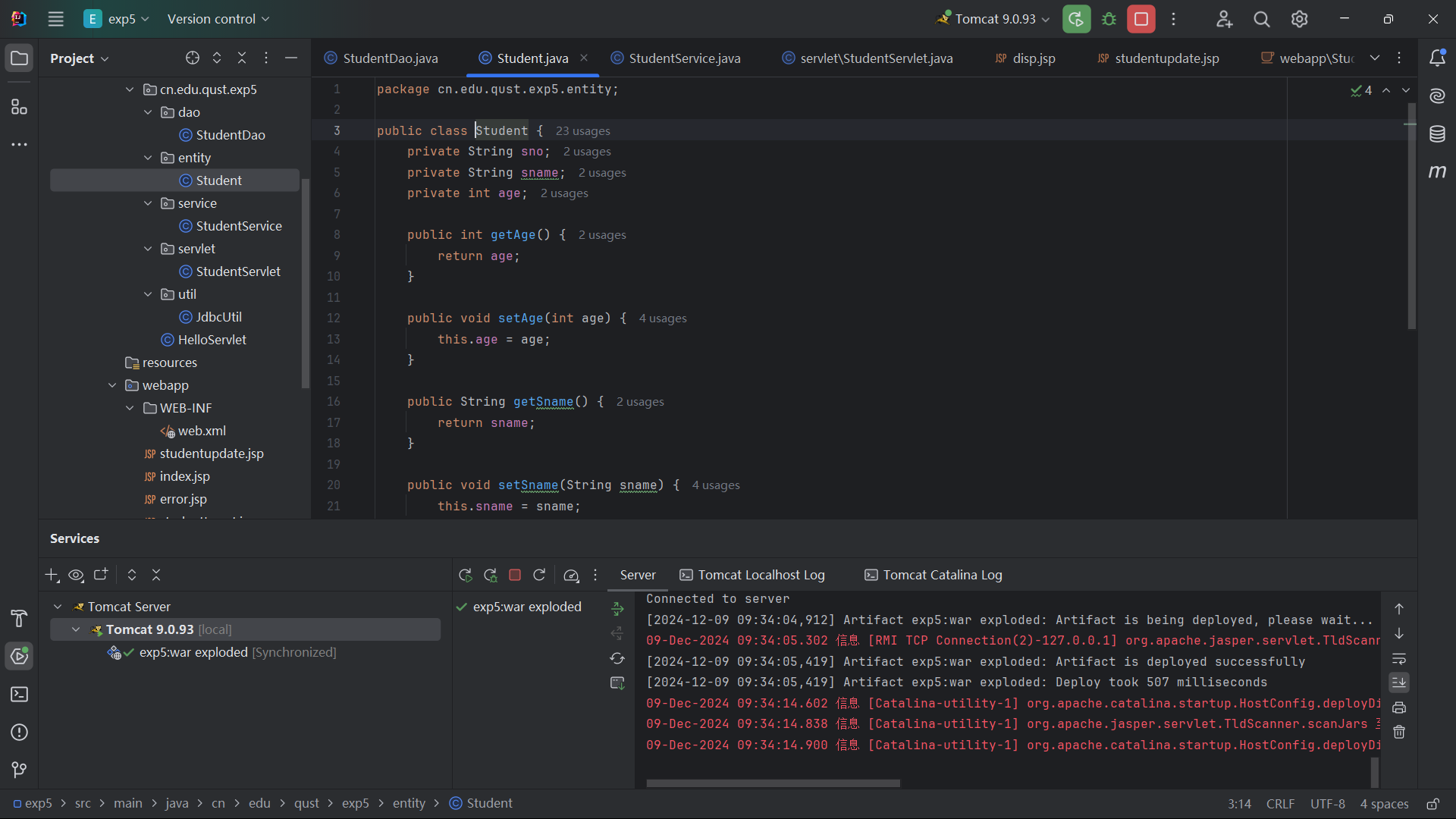
|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 备注 |
| sno | char(10) | 学号，主键 |
| sname | varchar(20) | 姓名 |
| age | int | 年龄 |

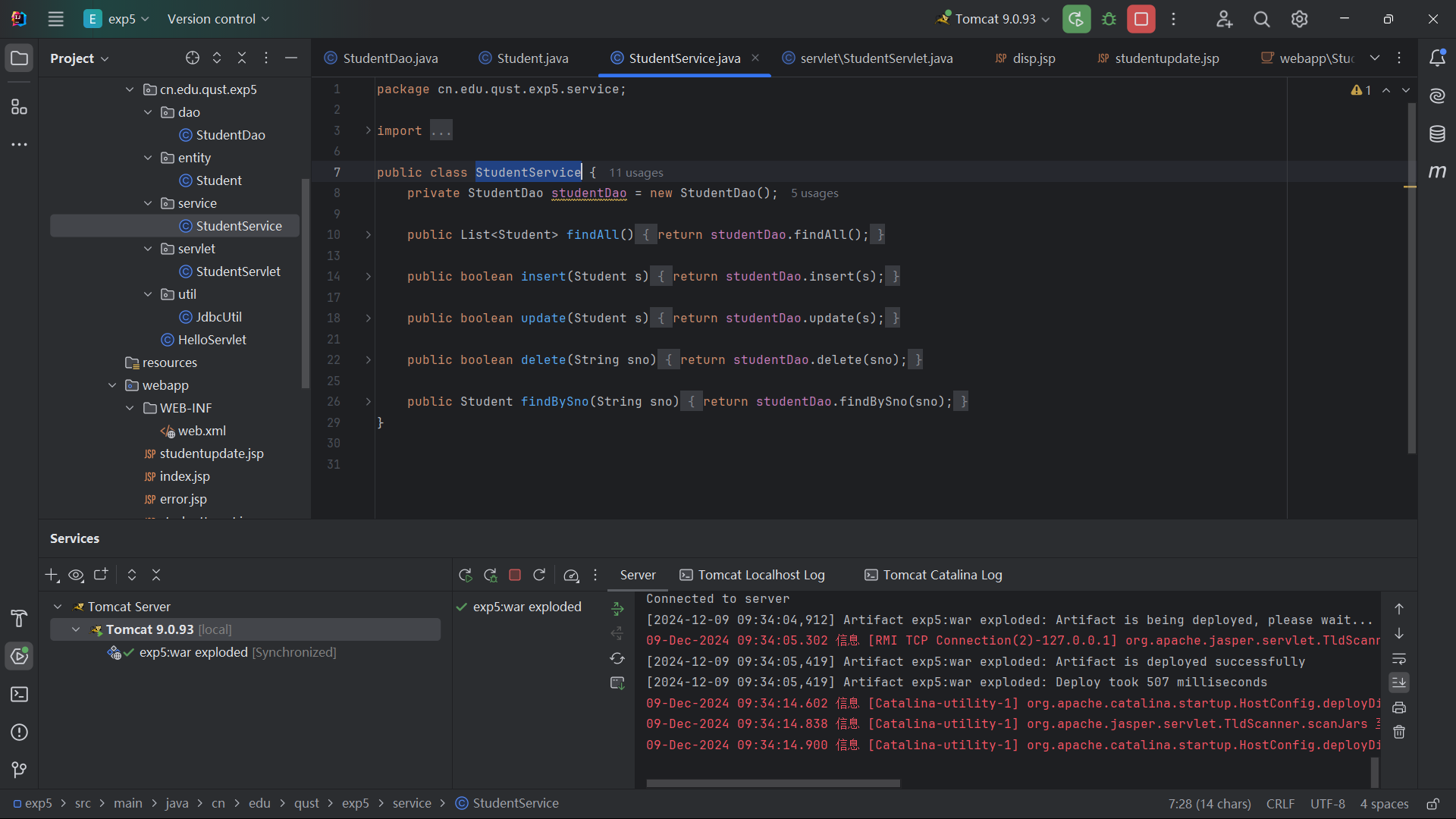
参考界面：

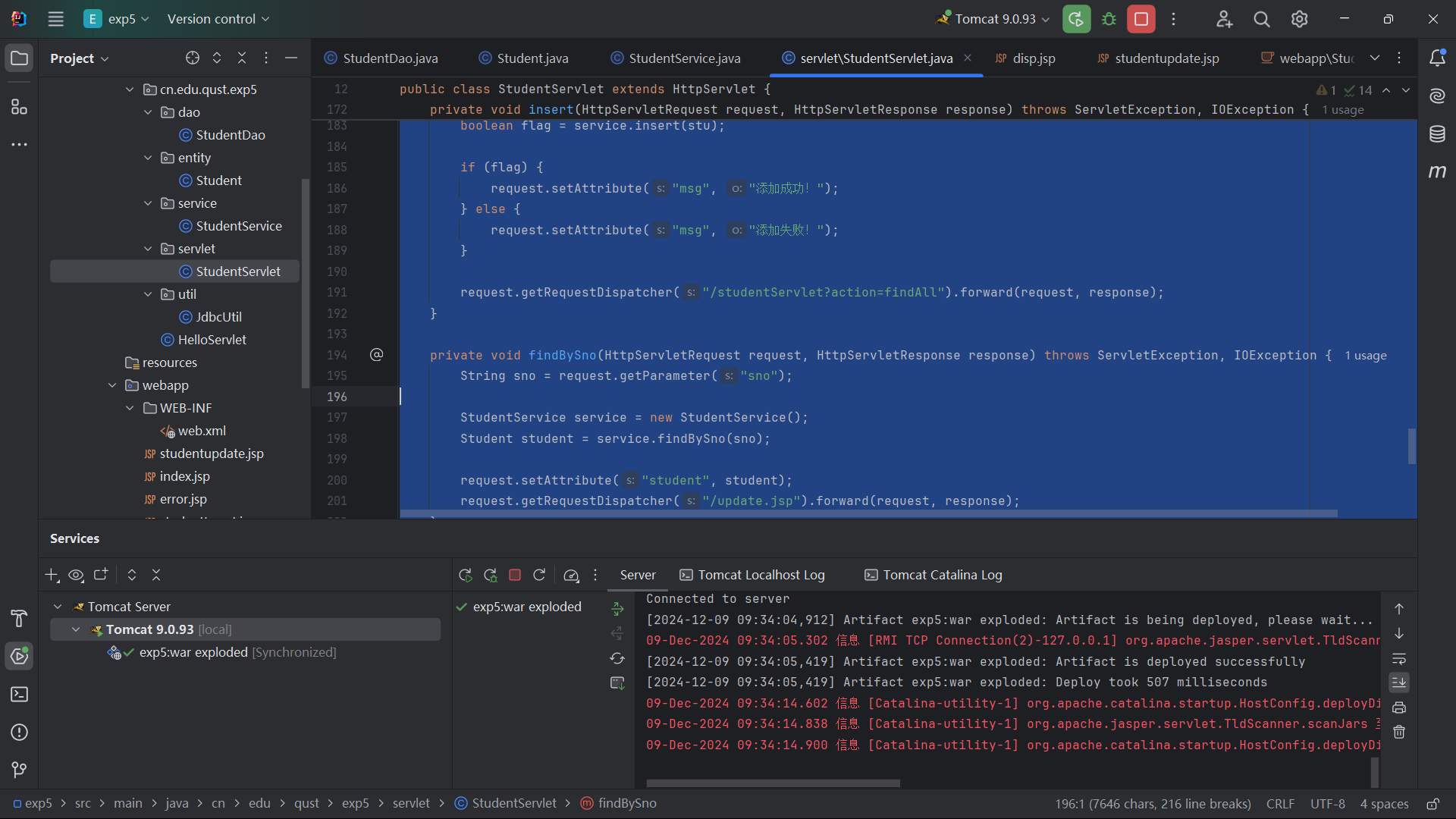


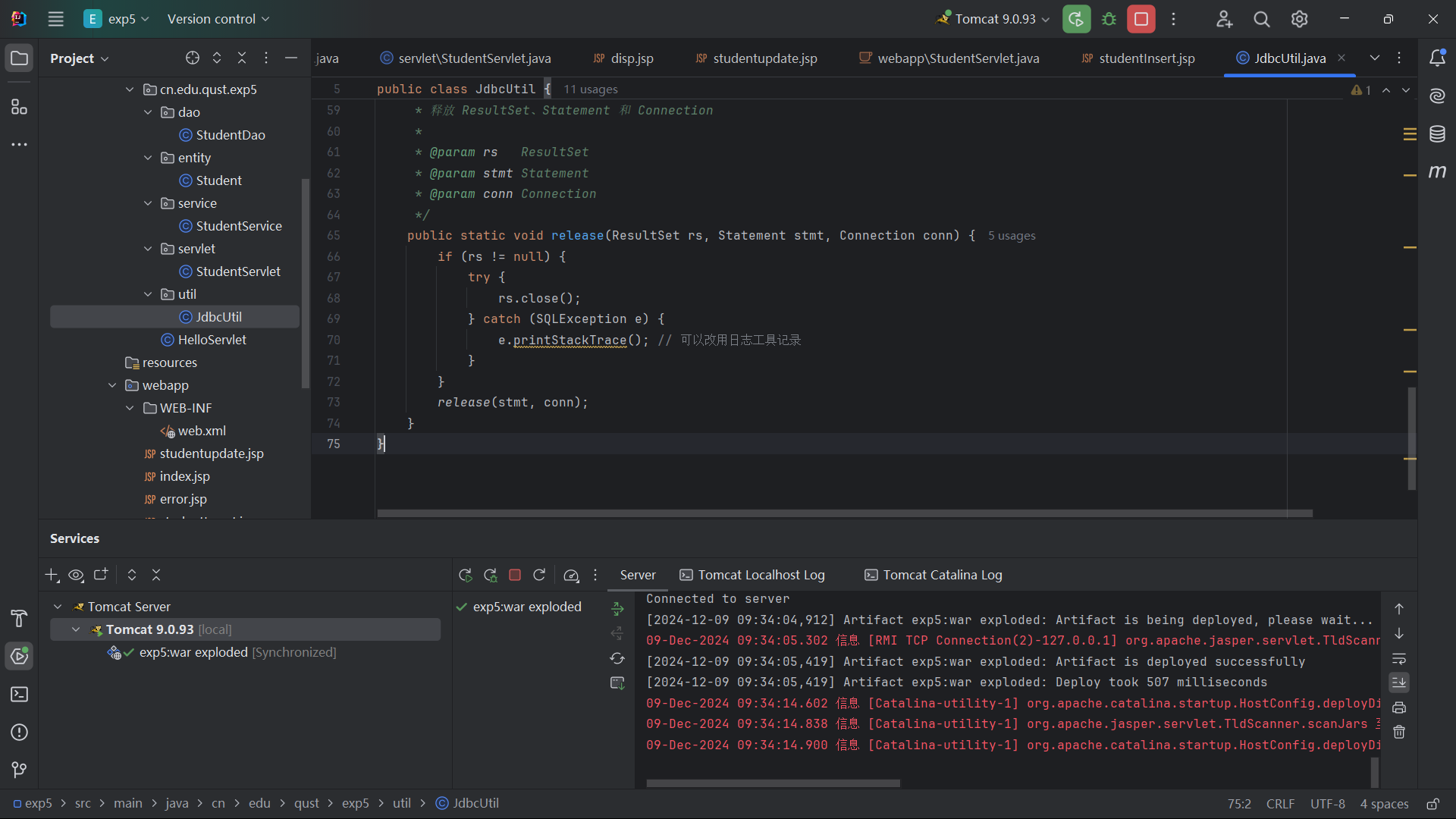
## 实验步骤













## 实验总结

(1) 熟悉了MySQL数据库和Navicat工具的使用  
本次实验的第一步是安装和配置MySQL数据库，并通过第三方客户端工具Navicat对数据库进行管理。在实践过程中，我掌握了数据库的基本操作，包括如何创建数据库、设计表结构以及通过Navicat实现对表数据的增、删、改、查等直观操作。这让我对MySQL数据库的基本使用有了更深入的理解，也为后续的开发工作打下了坚实的基础。同时，通过使用Navicat，我体会到第三方工具在提升开发效率和优化用户体验方面的优势。

(2) 掌握了JDBC相关API的使用  
实验中深入学习了JDBC技术的核心接口和类，并通过实例掌握了这些API的实际用法。首先，通过**Connection**接口，我学会了如何与数据库建立连接。其次，**Statement**接口用于执行SQL语句，而**ResultSet**则用于处理查询结果。我还深入研究了**PreparedStatement**的用法，它相比普通Statement更安全高效，能够有效防止SQL注入问题。此外，我还学习了如何在JDBC中实现事务操作，包括开启事务、提交事务和回滚事务，这些知识对于保证数据一致性和安全性非常重要。

(3) 理解了MVC模式的概念及实现方式  
在此次实验中，我通过实际开发体验了MVC模式的思想，并初步掌握了其实现方法。MVC模式是现代Web开发中的重要设计模式，通过将数据操作逻辑与用户界面分离，提高了代码的清晰性和可维护性。在实验中，**Model**部分由JavaBean实现，主要负责封装对学生表的增、删、改、查操作；**View**部分由JSP页面实现，用于展示学生信息和接收用户输入；**Controller**部分由Servlet完成，用来接收用户请求并调用Model进行业务逻辑处理。通过MVC模式的应用，我更加深刻地理解了分层设计的重要性。

(4) 掌握了JSP+Servlet+JavaBean技术的综合应用  
本实验的核心是JSP、Servlet和JavaBean技术的协同使用。在项目开发过程中，JSP主要用于前端页面的显示，例如通过表单获取用户输入数据；Servlet负责处理用户请求，将前端传递的数据进行逻辑处理后调用JavaBean；JavaBean则作为Model层，封装了与数据库交互的具体逻辑。通过这一综合开发实践，我不仅对这三种技术的独立功能有了全面的了解，还掌握了它们之间的协作方式。这种清晰的职责分工使得代码的可读性和可维护性大大提升。

(5) 初步熟悉了三层架构的设计与使用  
本次实验通过实现表现层、业务逻辑层和数据访问层的分离，使我初步了解了三层架构的设计思路。在这种架构中，表现层负责与用户交互，业务逻辑层负责处理核心逻辑，数据访问层负责与数据库进行交互。分层设计减少了代码的耦合性，使得不同层次的功能可以独立开发和维护，从而显著提升了系统的可扩展性和稳定性。这种架构思想让我深刻认识到系统设计中规范化的重要性。

(6) 学习了EL表达式和JSTL的使用  
在JSP页面的开发中，我初步掌握了EL表达式和JSTL标签库的用法。EL表达式使得访问和显示JavaBean的属性变得非常方便，例如通过${student.sname}可以直接展示学生姓名，简化了代码逻辑。JSTL标签库则提供了许多常用的功能，例如通过<c:forEach>标签实现对学生信息的动态迭代展示，从而减少了页面中的Java代码量。这种方式不仅优化了代码结构，也显著提升了页面开发效率。

(7) 完成了对学生表（student）的增删改查功能  
实验的主要任务是对学生表的数据进行增删改查操作。通过设计合适的页面和功能接口，我实现了以下操作：

**新增**：用户通过页面表单提交学生信息后，系统将数据插入到数据库中。

**删除**：根据用户提供的学号删除对应学生信息。

**修改**：允许用户对学生信息进行更新并提交，系统将更新后的数据写入数据库。

**查询**：展示学生表中的所有信息，同时支持根据特定条件查询符合要求的数据。  
这些功能的实现让我更加熟悉了JSP、Servlet、JavaBean以及JDBC之间的协作过程，也提升了实际开发能力。

(8) 实验感悟  
本次实验让我深刻体会到分层设计、良好代码结构以及规范化开发的重要性。通过JSP+Servlet+JavaBean技术的综合应用，我不仅提升了Web应用开发能力，也掌握了现代Web开发的基本模式。此外，通过实践MVC模式和三层架构设计，我对软件设计思想有了更深刻的理解，认识到清晰的职责分工能够显著提升代码的可读性和系统的可维护性。实验还让我意识到，数据库操作和事务管理是确保系统稳定性和数据一致性的重要环节，未来需要进一步深入学习和实践相关内容。

# 实验三 SSM框架的使用

## 一、实验目的

1、掌握mysql数据库的使用

2、掌握jQuery的使用

3、掌握SpringMVC、Spring、Mybatis框架的使用

## 二、实验要求

1、熟悉前端JavaScript、jQuery的使用

2、熟悉SpringMVC、Spring、Mybatis框架的相关配置及使用

## 三、实验内容

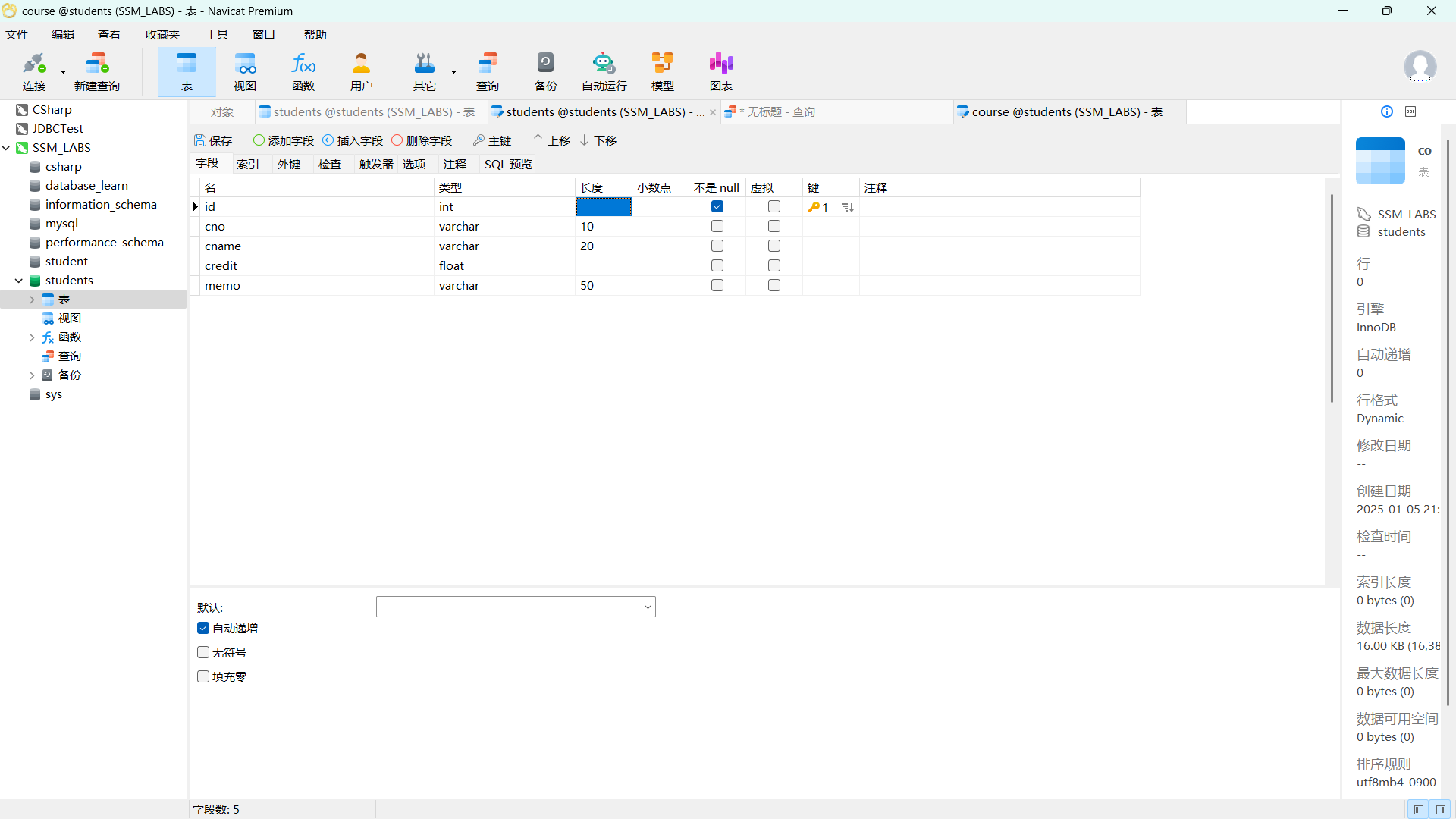
使用SSM框架实现对课程表（course）的增删改查等基本操作，表结构如下：

表名：course

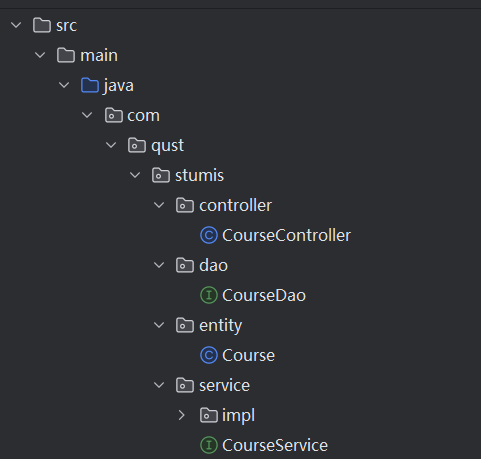
|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **备注** |
| id | int | 主键，自增1 |
| cno | varchar(10) | 课程号 |
| cname | varchar(20) | 课程名 |
| credit | float | 学分 |
| memo | varchar(50) | 备注 |

## 实验步骤

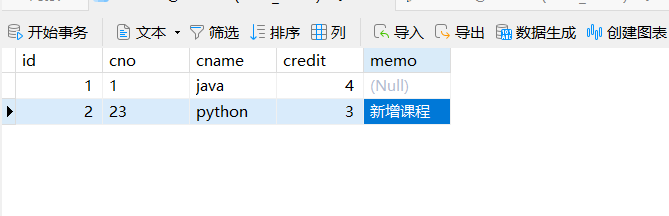
### 设计数据库表



### 设置如下项目结构



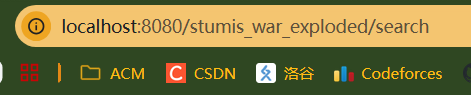
### 添加一些数据库记录，用于测试

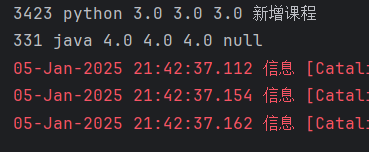


### 初步实现项目代码，使用控制台输出调试



### 访问对应链接，得到控制台输出结果



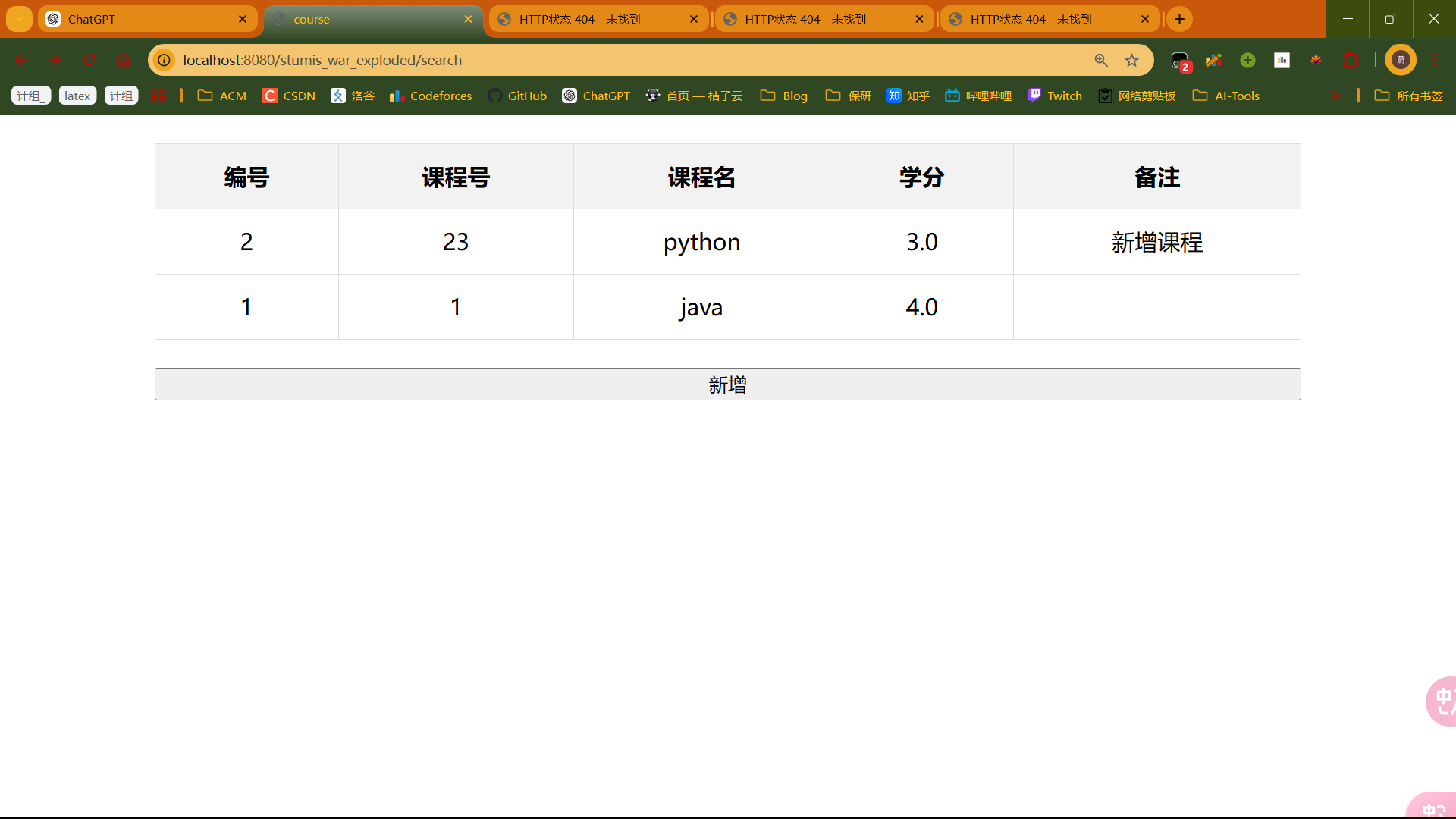


### 新建 course.jsp 文件

代码如下

<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
  
<html>  
<head>  
 <title>course</title>  
 <link rel="stylesheet" href="/css/bootstrap.min.css"/>  
 <script src="js/jquery.min.js"></script>  
 <script src="js/bootstrap.min.js"></script>  
 <style>  
 #container {  
 width: 80%; /\* 设置容器宽度为页面的 80% \*/  
 margin: 20px auto; /\* 上下 20px，左右居中 \*/  
 }  
  
 /\* 自定义样式：增加表格的框线 \*/  
 table {  
 width: 100%; /\* 使表格宽度自适应容器 \*/  
 border-collapse: collapse; /\* 合并单元格边框 \*/  
 }  
  
 th, td {  
 border: 1px solid #ddd; /\* 细边框 \*/  
 padding: 12px; /\* 增加单元格内边距 \*/  
 text-align: center;  
 }  
  
 th {  
 background-color: #f2f2f2; /\* 表头背景色 \*/  
 font-weight: bold;  
 }  
  
 /\* 新增按钮样式 \*/  
 #add {  
 margin-top: 20px; /\* 按钮与表格之间有一定间隔 \*/  
 width: 100%; /\* 让按钮宽度与表格一致 \*/  
 }  
 </style>  
 <script>  
 $().ready(function () {  
 $("#add").click(function () {  
 window.location.href = "showAdd"; // Redirect when the button is clicked  
 });  
 });  
 </script>  
</head>  
<body>  
<div id="container">  
 <table class="table table-bordered">  
 <thead>  
 <tr>  
 <th>编号</th>  
 <th>课程号</th>  
 <th>课程名</th>  
 <th>学分</th>  
 <th>备注</th>  
 </tr>  
 </thead>  
 <tbody>  
 <c:forEach var="cs" items="${courses}">  
 <tr>  
 <td>${cs.id}</td>  
 <td>${cs.cno}</td>  
 <td>${cs.cname}</td>  
 <td>${cs.credit}</td>  
 <td>${cs.memo}</td>  
 </tr>  
 </c:forEach>  
 </tbody>  
 </table>  
 <!-- 添加按钮，与表格宽度一致 -->  
 <button id="add" type="button" class="btn btn-primary">新增</button>  
</div>  
</body>  
</html>

### 7、实现对应数据表的访问界面



### 8、实现新增功能，表现为点击 **“新增”**按钮之后，跳转到如下页面 ：





输入数据，添加之后，效果如下 ：



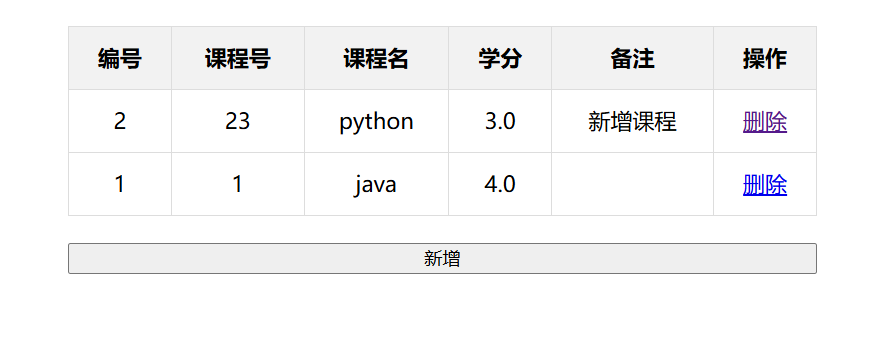
**其中，add.jsp 的代码如下 ：**

<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>studentAdd</title>  
 <link rel="stylesheet" href="css/bootstrap.min.css"/>  
 <script src="js/jquery.min.js"></script>  
 <script src="js/bootstrap.min.js"></script>  
 <style>  
 #container {  
 width: 400px;  
 margin: 30px auto;  
 }  
 </style>  
</head>  
<body>  
<div id="container">  
 <form class="form-horizontal" action="add" method="post">  
 <div class="form-group">  
 <label class="col-sm-2 control-label">编号</label>  
 <div class="col-sm-10">  
 <input type="text" class="form-control" name="id" placeholder="请输入编号"/>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="col-sm-2 control-label">课程号</label>  
 <div class="col-sm-10">  
 <input type="text" class="form-control" name="cno" placeholder="请输入课程号"/>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="col-sm-2 control-label">课程名</label>  
 <div class="col-sm-10">  
 <input type="text" class="form-control" name="cname" placeholder="请输入课程名"/>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="col-sm-2 control-label">学分</label>  
 <div class="col-sm-10">  
 <input type="text" class="form-control" name="credit" placeholder="请输入学分"/>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="col-sm-2 control-label">备注</label>  
 <div class="col-sm-10">  
 <input type="text" class="form-control" name="memo" placeholder="请输入备注"/>  
 </div>  
 </div>  
 <div class="form-group">  
 <div class="col-sm-offset-2 col-sm-10">  
 <button type="submit" class="btn btn-primary">添加</button>  
 </div>  
 </div>  
 </form>  
</div>  
</body>  
</html>

### 实现删除功能



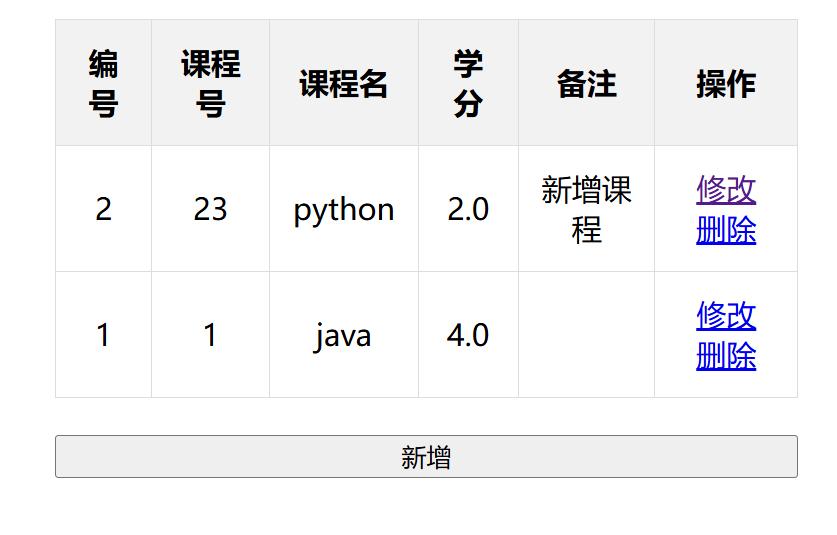
**点击删除按钮, 删除 “高等数学”课程 ：**



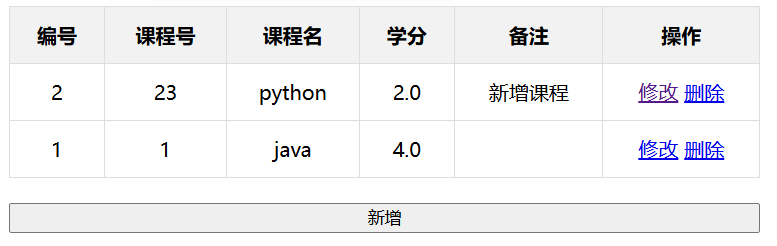
### 实现修改功能



**修改成功 ：**



### 最终效果如下 ：



### 12、代码

#### CourseController

package com.qust.stumis.controller;  
  
import com.qust.stumis.entity.Course;  
import com.qust.stumis.service.CourseService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.stereotype.Repository;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.servlet.ModelAndView;  
  
import java.util.List;  
  
@Controller  
public class CourseController {  
 @Autowired  
 CourseService courseService;  
  
 @RequestMapping("search")  
 public ModelAndView search(){  
 List<Course> list = courseService.search();  
  
// for(Course course : list){  
// System.out.println(course.getId()+' '+course.getCno()+' '+course.getCname()+' '+course.getCredit()+' '  
// +course.getCredit()+' '+course.getCredit()+' '+course.getMemo());  
// }  
  
 ModelAndView mv = new ModelAndView("course");  
 mv.addObject("courses", list);  
 return mv;  
  
 }  
  
 @RequestMapping("showAdd")  
 public String showAdd(){  
 return "add";  
 }  
  
 @RequestMapping("add")  
 public String add(Course course){  
 boolean flag = courseService.add(course);  
 if(flag){  
 return "redirect:search";  
 }  
 return null;  
 }  
  
 @RequestMapping("delete")  
 public String delete(String sno){  
 boolean flag = courseService.delete(sno);  
 if(flag){  
 return "redirect:search";  
 }  
 return null;  
 }  
  
 @RequestMapping("modify")  
 public ModelAndView modify(String cno){  
 Course course = courseService.findByCno(cno);  
 ModelAndView mv = new ModelAndView("update");  
 mv.addObject("course", course);  
 return mv;  
 }  
  
 @RequestMapping("save")  
 public String save(Course course){  
 boolean flag = courseService.add(course);  
 if(flag){  
 return "redirect:search";  
 }  
 return null;  
 }  
}

#### CourseDao

package com.qust.stumis.dao;  
  
import org.springframework.stereotype.Repository;  
import com.qust.stumis.entity.Course;  
  
import java.util.List;  
  
@Repository  
public interface CourseDao {  
 List<Course> search();  
  
 int add(Course course);  
  
 int delete(String cno);  
  
 int update(Course course);  
  
 Course findByCno(String cno);  
}

#### Course

package com.qust.stumis.entity;  
  
public class Course {  
 private int id;  
 private String cno;  
 private String cname;  
 private float credit;  
 private String memo;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getCno() {  
 return cno;  
 }  
  
 public void setCno(String cno) {  
 this.cno = cno;  
 }  
  
 public String getCname() {  
 return cname;  
 }  
  
 public void setCname(String cname) {  
 this.cname = cname;  
 }  
  
 public float getCredit() {  
 return credit;  
 }  
  
 public void setCredit(float credit) {  
 this.credit = credit;  
 }  
  
 public String getMemo() {  
 return memo;  
 }  
  
 public void setMemo(String memo) {  
 this.memo = memo;  
 }  
}

#### CourseServiceImpl

package com.qust.stumis.service.impl;  
  
import com.qust.stumis.dao.CourseDao;  
import com.qust.stumis.entity.Course;  
import com.qust.stumis.service.CourseService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CourseServiceImpl implements CourseService {  
  
 @Autowired  
 CourseDao courseDao;  
  
 @Override  
 public List<Course> search() {  
 return courseDao.search();  
 }  
  
 @Override  
 public boolean add(Course course) {  
 int rtn = courseDao.add(course);  
 return rtn > 0;  
 }  
  
 @Override  
 public boolean delete(String cno) {  
 int rtn = courseDao.delete(cno);  
 return rtn > 0;  
 }  
  
 @Override  
 public boolean update(Course course){  
 int rtn = courseDao.update(course);  
 return rtn > 0;  
 }  
  
 @Override  
 public Course findByCno(String cno) {  
 return courseDao.findByCno(cno);  
 }  
}

#### CourseService

package com.qust.stumis.service;  
  
import com.qust.stumis.entity.Course;  
  
import java.util.List;  
  
public interface CourseService {  
 List<Course> search();  
  
 boolean add(Course course);  
  
 boolean delete(String cno);  
  
 boolean update(Course course);  
  
 Course findByCno(String cno);  
}

#### CourseDao.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper PUBLIC  
 "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.qust.stumis.dao.CourseDao">  
 <select id="search" resultType="com.qust.stumis.entity.Course">  
 select \* from course  
 </select>  
  
 <insert id="add">  
 insert into course(id, cno, cname, credit, memo) values (#{id}, #{cno}, #{cname}, #{credit}, #{memo})  
 </insert>  
   
 <delete id="delete" parameterType="String">  
 delete from course where cno=#{cno}  
 </delete>  
  
 <update id="update" parameterType="com.qust.stumis.entity.Course">  
 update course set id=#{id},cno=#{cno},cname=#{cname},credit=#{credit},memo=#{memo} where cno=#{cno}  
 </update>  
  
 <select id="findByCno" parameterType="String" resultType="com.qust.stumis.entity.Course">  
 select \* from course where cno=#{cno}  
 </select>  
</mapper>

## 实验总结

**(1) MySQL数据库的使用**  
通过本次实验，我系统性学习并掌握了MySQL数据库在实际项目中的应用。在实验过程中，我设计并实现了 course 表的创建，包括主键自增设置、字段类型选择和表结构优化。在增删改查功能的开发中，我熟悉了SQL语句的基本写法和优化技巧，例如通过 WHERE 子句实现条件查询，利用索引提升查询效率，并结合分页查询应对大数据量场景。此外，我还学习了事务处理的基本操作，确保了数据操作的原子性和一致性。同时，通过MyBatis与数据库的结合使用，我熟悉了SQL映射文件的配置，掌握了通过动态SQL简化复杂查询逻辑的技巧，大大提升了开发效率和代码的可维护性。

**(2) 前端技术和jQuery的掌握**  
实验中，我进一步熟练了jQuery在前端开发中的使用技巧。通过jQuery的选择器，我能够快速操作DOM元素，实现页面的动态渲染和交互功能。在实现数据交互时，我重点学习了 AJAX 技术，成功实现了页面无刷新更新操作，提升了用户体验。同时，我还掌握了jQuery的事件绑定机制，通过监听用户操作（如点击、鼠标悬停）实现了多种动态效果。在数据校验方面，我结合jQuery实现了表单校验功能，确保了提交数据的合法性和完整性。此外，通过与后端的协作，我也熟悉了如何通过前端技术与SSM框架完成高效交互，为后续开发积累了宝贵经验。

**(3) SpringMVC框架的实践**  
在SpringMVC的实践中，我对其控制器设计、路由映射和视图解析有了全面了解。通过配置 @Controller 和 @RequestMapping，我实现了多种URL请求的精确映射，并利用Model和ModelAndView传递数据，成功将后台处理结果返回前端。我还学习了如何通过全局异常处理机制提升程序的健壮性，并在文件上传、下载功能中对SpringMVC的文件流处理有了更深入的理解。这部分实践让我对SpringMVC的核心概念和应用场景有了更深刻的体会。

**(4) Spring框架的深入应用**  
在Spring框架部分，我重点学习了IOC（控制反转）和AOP（面向切面编程）的使用方法。在IOC方面，我通过XML配置和注解方式管理Bean，深刻理解了Spring容器的作用及其对解耦的贡献。在AOP方面，我掌握了如何通过切面技术实现日志记录、权限校验等横切关注点的统一处理，减少了代码的冗余。在事务管理部分，我通过配置事务切面，确保了多表操作中的数据一致性，有效避免了操作失败导致的数据错误问题。这些实践让我更全面地掌握了Spring的核心功能和实际开发中的应用技巧。

**(5) MyBatis框架的学习与整合**  
在MyBatis框架的学习中，我深刻理解了ORM（对象关系映射）的思想，并成功通过Mapper接口实现了数据库表与Java对象的映射。在动态SQL部分，我学会了使用 if、choose、foreach 等标签编写灵活的查询语句，显著提高了复杂查询的可读性和维护性。此外，我还尝试使用MyBatis Generator工具自动生成Mapper文件和实体类，大幅提升了开发效率。通过将MyBatis与Spring整合，我了解了如何在Service层调用Mapper接口完成业务逻辑的实现，进一步强化了对分层架构的理解。

**(6) 项目综合开发能力的提升**  
通过完成课程表管理系统的开发，我从整体上理解了前后端协同开发的流程和最佳实践。从需求分析、数据库设计，到SSM框架的集成，再到前端页面与后端逻辑的交互，我熟悉了完整的开发流程。特别是在功能实现的过程中，我逐步培养了发现问题、解决问题的能力。例如，在处理数据更新时，我优化了前端和后端的数据同步逻辑，避免了潜在的竞态条件问题。通过这些实践，我不仅提高了技术能力，也积累了团队协作和项目管理的经验，为今后参与大型项目开发打下了基础。

**(7) 开发工具和调试技巧的应用**  
实验过程中，我熟悉了Eclipse/IntelliJ IDEA等IDE的使用，掌握了如何通过调试功能定位和解决代码中的问题。此外，我还学习了使用Postman测试后端接口，通过模拟不同请求方法（GET、POST、PUT、DELETE），验证了后端接口的正确性和鲁棒性。在前端开发中，我通过浏览器开发者工具（F12）进行DOM调试和网络请求分析，进一步提高了调试效率。这些工具和技巧的熟练使用，让我在开发过程中更加高效和从容。

**(8) 综合实验经验总结**  
本次实验让我从理论到实践全方位掌握了SSM框架及相关技术的应用。实验内容覆盖了前端、后端、数据库的多个关键环节，使我对Web开发有了系统性的认识。在实践中，我不仅提升了技术水平，还培养了严谨的开发思维和团队合作精神。这次实验为我未来的学习和工作奠定了坚实的基础，也激发了我对Web开发领域更深入学习的兴趣。

# 实验四 SpringBoot的使用

## 一、实验目的

1、掌握SpringBoot、SpringMVC、Spring、Mybatis框架的使用

2、掌握模板引擎thymeleaf的使用

## 二、实验要求

1、熟悉前端模板引擎themeleaf的使用

2、熟悉SpringBoot框架的相关配置及使用

## 三、实验内容

使用SpringBoot实现对图书表（book）的增删改查等基本操作，表结构如下：

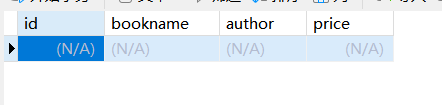
表名：book

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **备注** |
| id | int | 主键，自增1 |
| bookname | varchar(30) | 书名 |
| author | varchar(20) | 作者 |
| price | float | 价格 |

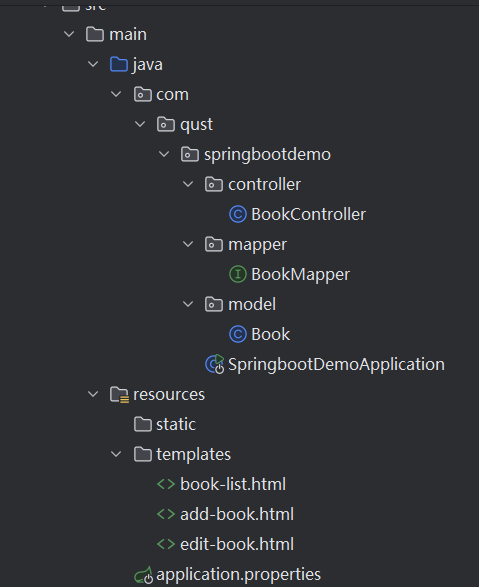
## 实验步骤

### 初始化项目并创建数据表





### 项目结构设置如下



### 编写对应代码

#### com.qust.springbootdemo.controller.BookControoler

package com.qust.springbootdemo.controller;  
  
import com.qust.springbootdemo.model.Book;  
import com.qust.springbootdemo.mapper.BookMapper;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@Controller  
public class BookController {  
  
 @Autowired  
 private BookMapper bookMapper;  
  
 @GetMapping("/")  
 public String getAllBooks(Model model) {  
 List<Book> books = bookMapper.getAllBooks();  
 model.addAttribute("books", books);  
 return "book-list";  
 }  
  
 @GetMapping("/add")  
 public String addBookForm() {  
 return "add-book";  
 }  
  
 @PostMapping("/add")  
 public String addBook(Book book) {  
 bookMapper.insertBook(book);  
 return "redirect:/";  
 }  
  
 @GetMapping("/edit/{id}")  
 public String editBookForm(@PathVariable int id, Model model) {  
 Book book = bookMapper.getBookById(id);  
 model.addAttribute("book", book);  
 return "edit-book";  
 }  
  
 @PostMapping("/edit")  
 public String editBook(Book book) {  
 bookMapper.updateBook(book);  
 return "redirect:/";  
 }  
  
 @GetMapping("/delete/{id}")  
 public String deleteBook(@PathVariable int id) {  
 bookMapper.deleteBook(id);  
 return "redirect:/";  
 }  
}

#### com.qust.springbootdemo.mapper.BookMapper

package com.qust.springbootdemo.mapper;  
  
import com.qust.springbootdemo.model.Book;  
import org.apache.ibatis.annotations.\*;  
  
import java.util.List;  
  
@Mapper  
public interface BookMapper {  
 @Insert("INSERT INTO book(bookname, author, price) VALUES(#{bookname}, #{author}, #{price})")  
 void insertBook(Book book);  
  
 @Delete("DELETE FROM book WHERE id = #{id}")  
 void deleteBook(int id);  
  
 @Update("UPDATE book SET bookname = #{bookname}, author = #{author}, price = #{price} WHERE id = #{id}")  
 void updateBook(Book book);  
  
 @Select("SELECT \* FROM book WHERE id = #{id}")  
 Book getBookById(int id);  
  
 @Select("SELECT \* FROM book")  
 List<Book> getAllBooks();  
}

#### com.qust.springbootdemo.model.Book

package com.qust.springbootdemo.model;  
  
public class Book {  
 private int id;  
 private String bookname;  
 private String author;  
 private float price;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getBookname() {  
 return bookname;  
 }  
  
 public void setBookname(String bookname) {  
 this.bookname = bookname;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public float getPrice() {  
 return price;  
 }  
  
 public void setPrice(float price) {  
 this.price = price;  
 }  
   
}

#### com.qust.springbootdemo.SpringDemoApplication

package com.qust.springbootdemo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringbootDemoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(SpringbootDemoApplication.class, args);  
 }  
  
}

#### templates.book-list.html

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Book List</title>  
</head>  
<body>  
<h1>Book List</h1>  
<table>  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Book Name</th>  
 <th>Author</th>  
 <th>Price</th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="book : ${books}">  
 <td th:text="${book.id}"></td>  
 <td th:text="${book.bookname}"></td>  
 <td th:text="${book.author}"></td>  
 <td th:text="${book.price}"></td>  
 <td>  
 <a th:href="@{/edit/{id}(id=${book.id})}">Edit</a>  
 <a th:href="@{/delete/{id}(id=${book.id})}">Delete</a>  
 </td>  
 </tr>  
 </tbody>  
</table>  
<a href="/add">Add New Book</a>  
</body>  
</html>

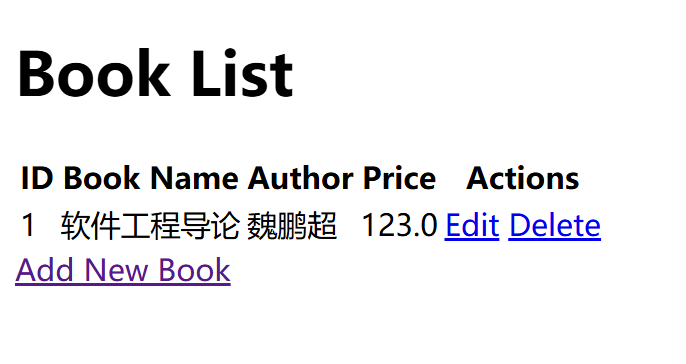
#### templates.add-book.html

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Add Book</title>  
</head>  
<body>  
<h1>Add New Book</h1>  
<form action="/add" method="post">  
 <label for="bookname">Book Name:</label>  
 <input type="text" id="bookname" name="bookname" required><br>  
  
 <label for="author">Author:</label>  
 <input type="text" id="author" name="author" required><br>  
  
 <label for="price">Price:</label>  
 <input type="number" step="0.01" id="price" name="price" required><br>  
  
 <button type="submit">Add Book</button>  
</form>  
</body>  
</html>

#### templates.edit-book.html

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Edit Book</title>  
</head>  
<body>  
<h1>Edit Book</h1>  
<form action="/edit" method="post">  
 <input type="hidden" name="id" th:value="${book.id}">  
  
 <label for="bookname">Book Name:</label>  
 <input type="text" id="bookname" name="bookname" th:value="${book.bookname}" required><br>  
  
 <label for="author">Author:</label>  
 <input type="text" id="author" name="author" th:value="${book.author}" required><br>  
  
 <label for="price">Price:</label>  
 <input type="number" step="0.01" id="price" name="price" th:value="${book.price}" required><br>  
  
 <button type="submit">Update Book</button>  
</form>  
</body>  
</html>

### 运行代码，添加测试数据，效果如下

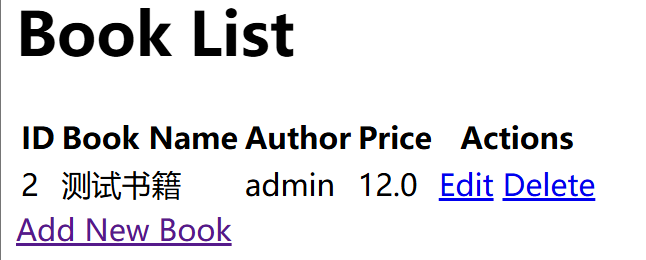


**我们点击 Add New Book :**





**添加成功，接下来测试删除功能，点击 Delete**

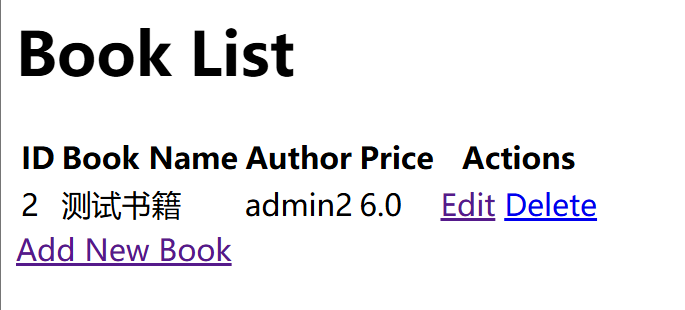


**成功删除**

**最后测试 Edit**



修改成功



## 五、实验总结

通过本次Spring Boot实验，我深入了解了Spring Boot、SpringMVC、MyBatis、Thymeleaf等技术的结合使用，掌握了如何搭建一个基于Spring Boot的图书管理系统。以下是对本次实验的总结，分为8个方面进行详细分析：

#### （1）Spring Boot的简洁性与高效性

Spring Boot作为一个简化了Spring框架配置的开发框架，它的出现极大地减少了开发者的配置工作。在本次实验中，我们通过Spring Initializr生成了项目模板，自动引入了所需的依赖和配置，使得项目的启动变得非常便捷。Spring Boot的“约定大于配置”原则，让我们无需为基础配置浪费大量时间，只需要关注业务逻辑的开发。比如，数据库的配置、Spring MVC的配置、MyBatis的配置都可以通过少量的注解和配置文件完成，大大提高了开发效率。

Spring Boot的自动配置特性也让我在实际开发中不需要手动配置复杂的XML文件和Java配置类。这使得Spring Boot成为一个非常适合初学者和快速开发项目的框架。

#### （2）Spring MVC框架的角色与作用

Spring MVC是Spring框架的一个重要模块，负责请求的接收、路由、处理和响应的返回。在本次实验中，Spring MVC的使用主要体现在请求处理、数据传递和页面跳转上。通过@Controller注解标记控制器类，通过@RequestMapping等注解定义请求的映射路径，从而实现前端与后端的交互。

例如，在BookController类中，我们通过@GetMapping和@PostMapping分别处理查询和新增图书的请求，利用Thymeleaf模板引擎返回数据到前端页面。Spring MVC框架通过这些注解简化了控制器的定义，让开发者专注于业务逻辑的实现，而无需关注请求分发和视图解析等繁琐工作。

#### （3）MyBatis框架的使用与优势

MyBatis作为一个轻量级的ORM框架，在本次实验中发挥了重要作用。MyBatis使得我们能够通过映射SQL语句与Java对象进行灵活的映射操作。在实验中，我们通过BookMapper接口与MyBatis的注解方式定义了增、删、改、查等SQL操作，而不需要编写复杂的JDBC代码或手动处理数据库连接。

MyBatis的优势在于它提供了比JPA更细粒度的SQL控制，让开发者能够完全控制SQL执行的过程，并且它支持多种复杂的SQL语句，如连接查询、子查询等。在实际开发中，MyBatis提供了丰富的扩展功能，如缓存、分页等，使得它在性能和灵活性方面表现得非常优秀。

#### （4）Thymeleaf模板引擎的使用

Thymeleaf是Spring Boot默认支持的模板引擎，广泛用于动态网页的渲染。在本次实验中，我们利用Thymeleaf进行页面渲染，将后端数据展示到前端页面。

Thymeleaf语法简单且功能强大，可以直接通过模板表达式从控制器传递的Model中获取数据，并在HTML页面中进行渲染。通过th:text、th:href等属性，我们能够灵活地插入动态数据，生成动态网页。例如，在book-list.html页面中，我们使用th:each循环遍历图书数据列表，并通过th:text将书籍的详细信息渲染到页面上。

Thymeleaf的优势在于它支持HTML5的标准语法，能够与传统HTML页面兼容，且具备丰富的条件渲染和迭代功能，使得它在前后端渲染中表现得非常出色。

#### （5）前后端分离与页面交互

本次实验中，我们通过Spring Boot的Spring MVC和Thymeleaf模板引擎，实现了一个简单的前后端交互。在前端页面中，用户可以查看所有书籍信息、添加书籍、编辑书籍以及删除书籍，而后端则处理用户的请求，并与数据库进行交互。

前后端交互主要通过HTTP请求来完成。在Spring MVC中，我们定义了多个请求映射方法（如/add、/edit、/delete等）来响应前端的操作。通过这些方法，用户提交的表单数据被传递到控制器中，控制器根据请求类型执行相应的数据库操作，并最终通过Thymeleaf返回动态页面。

#### （6）数据库设计与SQL操作

在本次实验中，数据库表book设计相对简单，包含id、bookname、author和price四个字段。通过这些字段，实验的增删改查操作得以实现。我们在MyBatis的BookMapper接口中使用了注解来编写SQL语句，通过@Select、@Insert、@Update、@Delete等注解实现了对数据库表的基本操作。

数据库设计和SQL操作是整个实验的核心，通过MyBatis和Spring Boot的结合，我们能够快速实现数据库的增删改查功能。在实际开发中，数据库设计是构建系统的基础，合理的表设计和SQL查询能够有效提升系统的性能和可维护性。

#### （7）项目的模块化与可扩展性

在本次实验中，我们将不同的功能模块进行了合理的拆分，比如将数据库操作封装在MyBatis的Mapper接口中，将请求处理封装在Controller类中，业务逻辑与数据访问层之间通过Service类进行解耦。这样的模块化设计使得项目的结构清晰，代码的可维护性得到了提升。

此外，Spring Boot的自动配置和约定优于配置的理念，让我们可以轻松扩展项目功能。例如，未来我们可以在原有基础上加入分页、排序等功能，甚至可以集成其他技术如Spring Security进行权限控制，提升系统的安全性。

#### （8）实践中的挑战与解决方案

尽管Spring Boot和MyBatis提供了强大的功能，但在实际操作中也遇到了一些挑战。例如，如何确保数据库操作的正确性、如何处理前端与后端数据的格式匹配等问题。通过查看文档和参考网上的解决方案，我们逐步克服了这些难题。

例如，在处理表单数据时，我们通过@RequestParam和@ModelAttribute注解将前端表单数据与后端的Java对象进行绑定，确保数据传递的正确性。此外，关于SQL注入和事务管理等问题，我们也通过学习和实践，掌握了如何安全地执行数据库操作，确保系统的稳定性和安全性。

### 总结

通过本次实验，我不仅掌握了Spring Boot、SpringMVC、MyBatis、Thymeleaf等技术的使用，还加深了对前后端分离、数据库设计、以及模块化开发的理解。在整个项目开发过程中，我体会到了Spring Boot框架的优势，能够快速搭建一个高效、可扩展的Web应用。同时，也在实践中解决了多个技术难题，提升了自己的开发能力。