Web安全实训报告

基于java开发的学生成绩管理应用程序设计

姓名：杜鑫星

学号：201701060101

班级：信息安全2017级-1班

[第1章 引言 4](#_Toc23179185)

[1.1项目概要 4](#_Toc23179186)

[1.1.1摘要 4](#_Toc23179187)

[1.1.2可行性分析 4](#_Toc23179188)

[1.2项目功能的基本需求与其目的 4](#_Toc23179189)

[1.2.1功能需求 4](#_Toc23179190)

[1.2.2性能需求 5](#_Toc23179191)

[1.2.3安全需求 5](#_Toc23179192)

[第2章 概要设计 6](#_Toc23179193)

[2.1 开发环境 6](#_Toc23179194)

[2.2 环境搭建 6](#_Toc23179195)

[2.2.1 Tomcat 6](#_Toc23179196)

[2.2.2 Mysql与Navicat 6](#_Toc23179197)

[2.2.3 IDEA 6](#_Toc23179198)

[2.2.4 JDBC+Servlet 7](#_Toc23179199)

[2.3 前端设计HTML与CSS样式 7](#_Toc23179200)

[2.3.1 设计登录注册界面样式 7](#_Toc23179201)

[2.3.2 设计业务界面样式 7](#_Toc23179202)

[2.4 数据库设计 7](#_Toc23179203)

[2.5 MVC 设计模式 9](#_Toc23179204)

[2.6 系统功能设计 14](#_Toc23179205)

[2.7 算法流程设计 15](#_Toc23179206)

[2.8 Cookie与Session会话管理 17](#_Toc23179207)

[2.9 Web安全设计 17](#_Toc23179208)

[第3章 详细设计 18](#_Toc23179209)

[3.1 注册 18](#_Toc23179210)

[3.2 登录注销 19](#_Toc23179211)

[3.3 自动登录 24](#_Toc23179212)

[3.4 成绩增删改查实现 26](#_Toc23179213)

[3.4.1 增加学生成绩 26](#_Toc23179214)

[3.4.2 删除学生成绩 29](#_Toc23179215)

[3.4.2 更改学生成绩 31](#_Toc23179216)

[3.4.2 查询学生成绩 33](#_Toc23179217)

[3.5 验证码 35](#_Toc23179218)

[3.6 防SQL注入、XSS攻击 38](#_Toc23179219)

[第4章 功能演示 43](#_Toc23179220)

[4.1 注册 43](#_Toc23179221)

[4.1.1 注册前 43](#_Toc23179222)

[4.1.2 注册页面 44](#_Toc23179223)

[4.1.3 注册后 45](#_Toc23179224)

[4.2 登录注销 46](#_Toc23179225)

[4.2.1 登录 46](#_Toc23179226)

[4.2.2 注销 47](#_Toc23179227)

[4.3 自动登录 48](#_Toc23179228)

[4.4 成绩的增删改查 49](#_Toc23179229)

[4.4.1 学生查询成绩 49](#_Toc23179230)

[4.4.2 教师查询成绩 49](#_Toc23179231)

[4.4.3 教师添加成绩 49](#_Toc23179232)

[4.4.4 教师修改成绩 50](#_Toc23179233)

[4.4.5 教师删除成绩 51](#_Toc23179234)

[4.5 验证码 52](#_Toc23179235)

[4.6 防SQL注入、XSS攻击 53](#_Toc23179236)

[第5章 总结 54](#_Toc23179237)

[5.1 工作完成度 54](#_Toc23179238)

[5.2 系统缺陷与不足 54](#_Toc23179239)

[5.3 系统可扩展功能 54](#_Toc23179240)

[5.4 收获与不足 55](#_Toc23179241)

# 第1章 引言

## 1.1项目概要

### 1.1.1摘要

学生成绩管理系统目的是为了对学生的成绩进行信息化管理，方便的管理学生成绩，有助于学生成绩的存储和查询，降低教师管理的成本，提高工作效率。

根据涉及的数据，本项目数据库设计有六张数据表：学生表、教师（管理员）表、成绩表、课程表、班级表、登录表。学生表和教师表用来存储系统用户；成绩表用来存储学生成绩；课程表用来储存课程信息；班级表用来存储班级信息；登录表用来存储登录信息。

本项目使用MVC模块化封装设计模式，后端运用Servlet+JDBC+Dao等Java相关的技术开发，前端运用JSP+CSS+HTML等相关技术开发，实现了权限管理、登录登出、自动登录、验证码、防SQL注入、防XSS，成绩的增加、删除、修改、查询等功能。

### 1.1.2可行性分析

社会可行性分析：成绩是每一个大学生都会接触到的东西，不仅学生为之心动，为了管理学生的成绩，如果不用计算机来实现，老师们也会感觉很头疼麻烦，开发学生成绩管理系统后，让学生的成绩管理变的更加方便，学生也可以实现自助查询成绩功能，老师们也可以对学生成绩进行增删改查进行管理等。

经济可行性分析：该系统的开发调查主要是需要软件，这些软件都是免费的。主要的花销在于服务器的维护，除此之外没有其他的花销。

法律可行性分析：系统开发及维护所使用工具和技术及数据信息不违反法律。

## 1.2项目功能的基本需求与其目的

### 1.2.1功能需求

网页界面设计：利用HTML和CSS实现客户端前台设计

注册：实现可以动态添加使用用户

登录：实现连接服务器权限控制

注销：实现断开服务器连接

多用户管理：用户分为管理员、学生。其中管理员可以实现增加学生成绩信息，删除学生成绩信息，修改学生成绩信息，查找学生成绩信息，其中学生可以实现登录查询成绩功能。数据库（成绩数据）的增删改查：实现对成绩的基本管理

### 1.2.2性能需求

自动登录：运用Cookie与Session进行会话管理方便用户登录提高用户体验

### 1.2.3安全需求

本系统需要登录以后才可以使用。学生只能查看成绩，只有老师可以对成绩进行增删改查

验证码校验：防止恶意爆破网站及自动恶意访问等操作

Web软件安全：实现对SQL注入、XSS、点击劫持等攻击手段的防御

# 第2章 概要设计

## 2.1 开发环境

开发系统：MacOs10.14

开发平台：IntelliJ IEDA2019

开发工具包版本：Jdk1.8

开发引入工具Jar包：jstl.jar、mysql-connector-java-8.0.15.jar、standard.jar

数据库：MySQL5.6

数据库管理工具：NavicatPremium 12

## 2.2 环境搭建

### 2.2.1 Tomcat

在Mac系统使用Brew包管理工具（其他类Linux系统使用相应包管理工具下载即可）下载Tomcat8.0

命令：brew install tomcat@8

brew默认安装地址：~/User/local/Celler/tomcat@8

### 2.2.2 Mysql与Navicat

在Mac系统使用Brew包管理工具（其他类Linux系统使用相应包管理工具下载即可）下载TMysql5.6

命令：brew install mysql@5.6

brew默认安装地址：~/User/local/Celler/mysql@5.6

安装后自动启动Mysql，一步步配置mysql即可。注意：记住设置的密码（brew安装默认设置为123456）

安装Navicat，连接本地mysql数据库即可可视化操作数据表进行增删改查数据

### 2.2.3 IDEA

安装IDEA，新建项目，WEB-INF在新建classes和lib目录，分别用于存放编译好的类和扩展依赖

配置Project Structe，设置Modules依赖为JDK1.8，添加依赖包为tomcat8bin目录下的lib目录（里面包含了servlet等依赖），设置项目输出路径（out path）为WEB-INF的classes目录

添加Libaraies：新建lib目录的jstl.jar、mysql-connector-java-8.0.15.jar、standard.jar

设置服务器环境路径为tomcat8路径（设置好后如果启动项目IDEA自动以终端命令启动tomcat8）

### 2.2.4 JDBC+Servlet

将JDBC的jar包：mysql-connector-java-8.0.15.jar和Servlet的jar包：servlet-api.jar导入WEB-INF/lib目录下

## 2.3 前端设计HTML与CSS样式

### 2.3.1 设计登录注册界面样式

新建登录、注册HTML分三大模块：div里面设计表单；footer设计页脚；引入js来控制动画逻辑

新建CSS样式文件，在HTML文件标签添加属性引入CSS样式。分别设计文本、颜色、动画对应属性的样式

新建JavaScript文件，实现JavaScript代码逻辑

### 2.3.2 设计业务界面样式

业务界面采用Html+Jsp前端渲染技术。

学生端：主要对应查询成绩功能。使用Form表单接受学生传入参数。使用td标签来设计显示表格

教师端（管理员）：主要对应查询、更新成绩功能。查询成绩与学生端一致，增加成绩同样使用Form表单接受增加同学的信息参数，以td表单表格形式渲染；删除更改页面增加表单输入框来进行更新和删除按钮

## 2.4 数据库设计

数据库设计有六张数据表：学生表、教师（管理员）表、成绩表、课程表、班级表、登录表。学生表和教师表用来存储系统用户；成绩表用来存储学生成绩；课程表用来储存课程信息；班级表用来存储班级信息；登录表用来存储登录信息。

其中每张表的键为：学生表：stuId、教师表：tcId、成绩表：gradeId、课程表：CourseId、班级表：classId、登录表：id

根据数据库表设计Model层：主要实现每个模型的get和set方法进行第一层封装。比如：Administrator对象（其余Model类似，实现属性的get和set方法，不再赘述。）

package Model;

public class Administrator {

private String tcId; //教师编号

private String tcName; //教师姓名

private String tcClassId; //班级编号

private String tcSex; //教师性别

private String tcPassword; //教师密码

public void setTcId(String tcId){

this.tcId = tcId;

}

public void setTcName(String tcName){

this.tcName = tcName;

}

public void setTcClassId(String tcClassId){

this.tcClassId = tcClassId;

}

public void setTcSex(String tcSex){

this.tcSex = tcSex;

}

public void setTcPassword(String tcPassWord){

this.tcPassword = tcPassWord;

}

public String getTcId(){

return tcId;

}

public String getTcName(){

return tcName;

}

public String getTcClassId(){

return tcClassId;

}

public String getTcSex(){

return tcSex;

}

public String getTcPassword(){

return tcPassword;

}

}

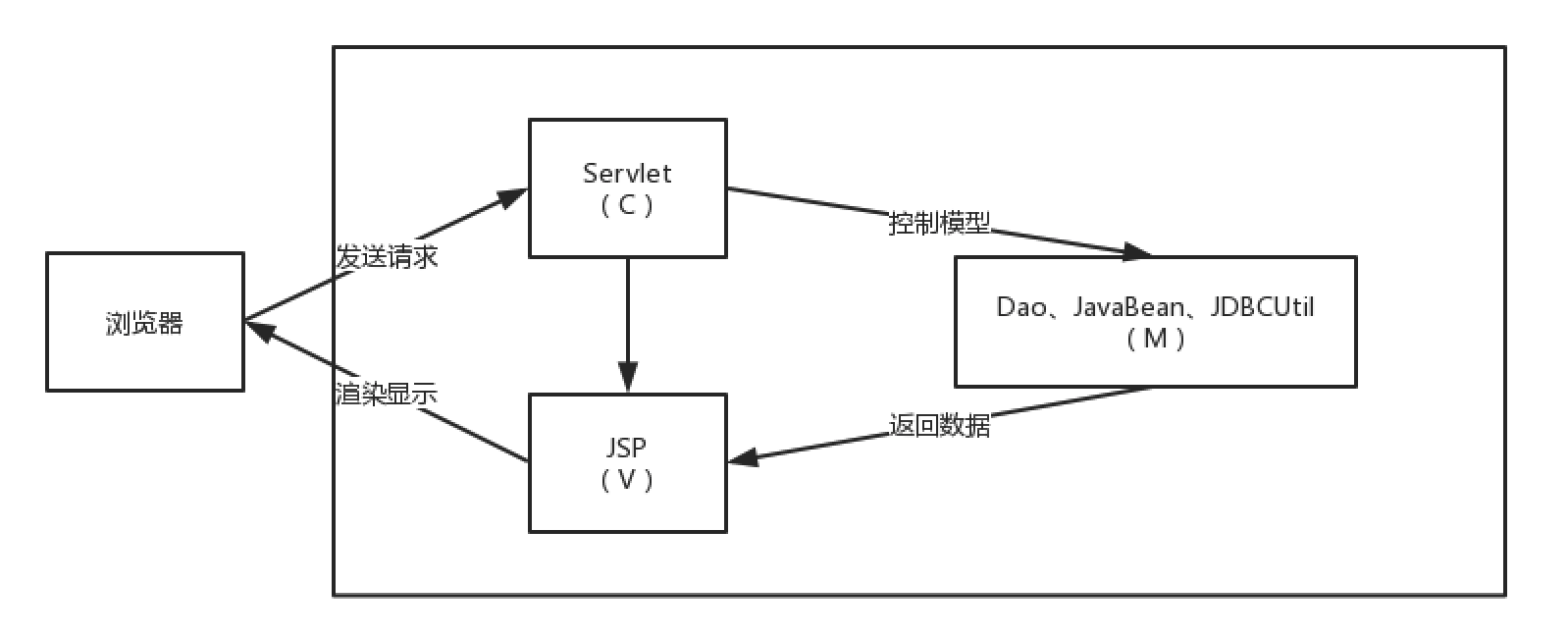
## 2.5 MVC 设计模式

MVC设计模式是一种软件设计典范。

用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。View（视图）是用户看到并与之交互的界面、Model（模型）表示类与业务方法逻辑、Controller（控制器）接受用户调用的输入并调用模型和视图去完成用户的需求。

对于本项目，Model包主要存放根据数据表属性设计6大类，分别进行一层封装来提供给Controller实例化控制；View主要存放JSP文件、CSS样式文件以html+css+JavaScript+JSP来进行渲染显示前端页面；Controller主要接受View层传来的参数，实例化类然后调用相应方法来实现业务逻辑。



除此之外:

Util层包下对JDBC建立数据库连接、释放数据库连接进行一层封装。使得Dao层需要连接数据库操作直接调用接口进行连接释放

API功能：

1.实现加载驱动、并建立数据库连接API

2.实现关闭数据库连接、释放资源API

package Util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class JDBCUtils {

// 加载驱动，并建立数据库连接

public static Connection getConnection() throws SQLException,

ClassNotFoundException {

Class.forName("com.mysql.cj.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/jspweb1?serverTimezone=GMT";

String Name = "root";

String password = "123456";

Connection conn = DriverManager.getConnection(url, Name, password);

return conn;

}

// 关闭数据库连接，释放资源

public static void release(Statement stmt, Connection conn) {

if (stmt != null) {

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

stmt = null;

}

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

conn = null;

}

}

public static void release(ResultSet rs, Statement stmt,

Connection conn){

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}

rs = null;

}

release(stmt, conn);

}

}

Dao层包下对数据操作语言sql语句进行第二层封装。主要封装增删改查数据功能，以便Controller层进行调用API来直接操纵数据实现业务逻辑。

比如GradeDao类（其余根据数据表的5个Dao层类都是实现了增删改查的接口不再赘述）

package Dao;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.ArrayList;

import Model.Grade;

import Util.JDBCUtils;

public class GradeDao {

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

//添加成绩操作

public boolean insert(Grade grade){

try {

conn = JDBCUtils.getConnection();

stmt = conn.createStatement();

String sql = "insert into grade (stuId, stuName, courseId, tcId, courseGrade, classId) "

+ "values('"

+ grade.getStuId()

+ "','"

+ grade.getStuName()

+ "','"

+ grade.getCourseId()

+ "','"

+ grade.getTcId()

+ "','"

+ grade.getCourseGrade()

+ "','"

+ grade.getClassId()

+ "')";

System.out.println("添加SQL语句为：" + sql);

int num = stmt.executeUpdate(sql);

if (num > 0){

return true;

}

return false;

} catch (Exception ee) {

ee.printStackTrace();

} finally {

JDBCUtils.release(rs, stmt, conn);

}

return false;

}

//查询所有的grade对象

public ArrayList<Grade> findAll(){

ArrayList<Grade> list = new ArrayList<Grade>();

try{

conn = JDBCUtils.getConnection();

stmt = conn.createStatement();

String sql = "select \* from grade";

System.out.println("查询所有的SQL语句：" + sql);

rs = stmt.executeQuery(sql);

while (rs.next()) {

Grade grade = new Grade();

grade.setCourseId(rs.getString("courseId"));

grade.setStuName(rs.getString("stuName"));

grade.setCourseGrade(rs.getString("courseGrade"));

grade.setStuId(rs.getString("stuId"));

grade.setTcId(rs.getString("tcId"));

grade.setClassId(rs.getString("classId"));

list.add(grade);

}

return list;

} catch(Exception ee) {

ee.printStackTrace();

} finally {

JDBCUtils.release(rs, stmt, conn);

}

return null;

}

//根据ID查询

public ArrayList<Grade> find(String stuId){

ArrayList<Grade> list = new ArrayList<Grade>();

try{

conn = JDBCUtils.getConnection();

stmt = conn.createStatement();

String sql = "select \* from grade where stuId = '" + stuId + "'";

rs = stmt.executeQuery(sql);

System.out.println("单个查询SQL语句：" + sql);

while (rs.next()) {

Grade grade = new Grade();

grade.setCourseId(rs.getString("courseId"));

grade.setStuName(rs.getString("stuName"));

grade.setCourseGrade(rs.getString("courseGrade"));

grade.setStuId(rs.getString("stuId"));

grade.setTcId(rs.getString("tcId"));

grade.setClassId(rs.getString("classId"));

list.add(grade);

}

return list;

}catch(Exception ee){

ee.printStackTrace();

}finally{

JDBCUtils.release(rs, stmt, conn);

}

return null;

}

//删除成绩

public boolean delete(Grade grade){

try{

conn = JDBCUtils.getConnection();

stmt = conn.createStatement();

String sql = "delete from grade where stuId= '" + grade.getStuId()

+ "' and stuName = '" + grade.getStuName()

+ "' and courseId = '" + grade.getCourseId()

+ "' and tcId = '" + grade.getTcId()

+ "' and courseGrade = '" + grade.getCourseGrade()

+ "' and classId = '" + grade.getClassId()

+ "'";

System.out.println("删除的SQL语句是：" + sql);

int num = stmt.executeUpdate(sql);

if (num > 0){

return true;

}

return false;

} catch (Exception ee){

ee.printStackTrace();

} finally {

JDBCUtils.release(rs, stmt, conn);

}

return false;

}

//修改成绩

public boolean update(Grade grade){

try{

conn = JDBCUtils.getConnection();

stmt = conn.createStatement();

String sql = "update grade set "

+ "courseGrade='" + grade.getCourseGrade()

+ "' where stuId='" + grade.getStuId()

+"' and stuName= '" + grade.getStuName()

+"' and courseId='" + grade.getCourseId()

+"' and tcId='" + grade.getTcId()

+"' and classId='" + grade.getClassId()

+ "'";

System.out.println("修改的SQL语句是：" + sql);

int num = stmt.executeUpdate(sql);

if (num > 0){

return true;

}

return false;

} catch (Exception ee){

ee.printStackTrace();

} finally {

JDBCUtils.release(rs, stmt, conn);

}

return false;

}

}

## 2.6 系统功能设计

学生成绩管理系统具备以下功能：教师（管理员）学生分权限登录，查询学生成绩，更新学生成绩：增加学生成绩信息、删除学生成绩信息、修改学生成绩信息。

分权限登录：系统分为管理员和学生用户两个权限，因为管理员就是老师，所以账号秘密固定，但是学生有很多，所以在学生登录的过程中要对数据库中的学生进行查询，如果存在，而且密码正确，才能成功登录

增加学生成绩：该功能可以将学生的姓名，学号，班级，课程编号、教课老师、课程成绩、班级编号等信息录入系统

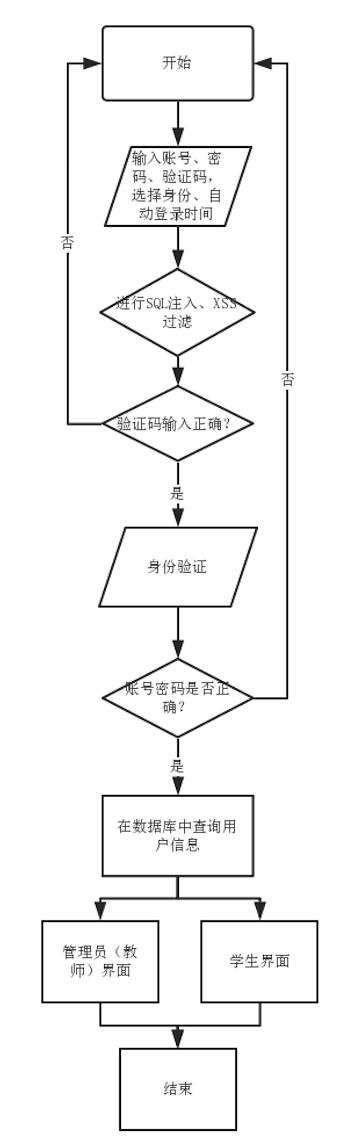
删除学生成绩：该功能可以将学生的信息全部列举出来，然后选择想要删除的学生的信息

修改学生成绩：该功能可以将学生的所有信息全部列举出来，然后再选择想要修改的学生的成绩直接进行修改

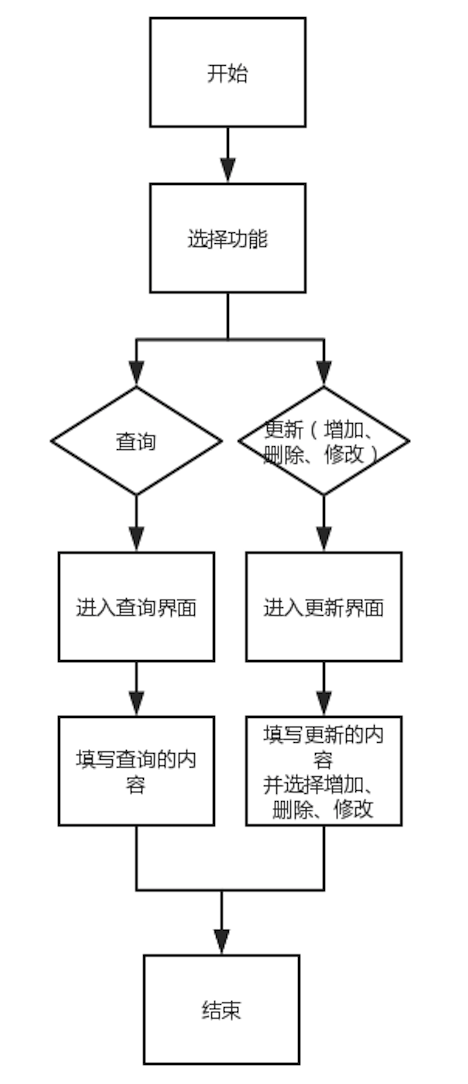
查询学生成绩信息：该功能可以在输入学号后显示出学生的各项成绩信息

## 2.7 算法流程设计

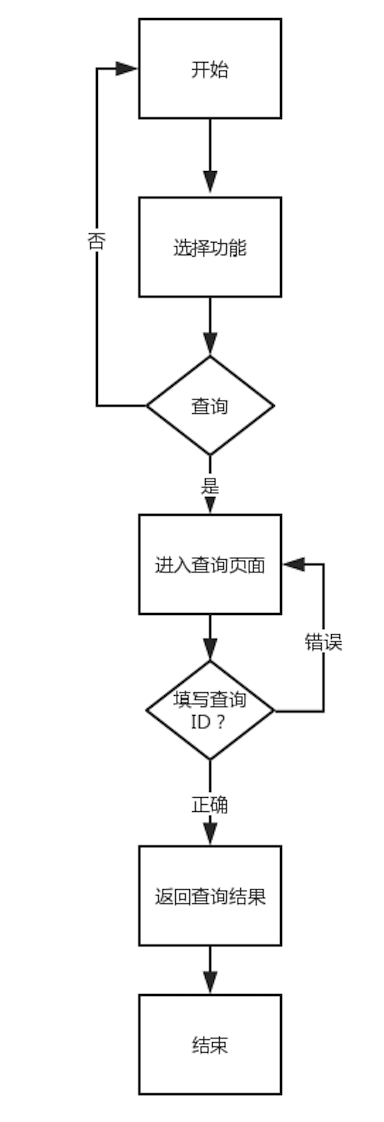
登录与注册流程图：



教师（管理员）业务界面功能程序执行流程图：



学生业务界面功能程序执行流程图：



## 2.8 Cookie与Session会话管理

HTTP协议是无状态协议所以必须使用技术进行标记用户。本项目使用了Cookie+Session会话管理等技术实现了登录、注销、权限控制、自动登录等业务需求。登录时，前端表单进行选择教师还是学生来传递不同参数，Servlet根据接受的参数不同将用户状态user存入session域，然后对应用户登录记录下，发送自动登录Cookie给浏览器端此地方用户的信息要进行加密以防Cookie劫持。登录成功后，根据用户不同跳转到学生业务界面或教师业务界面

## 2.9 Web安全设计

可控的输入点都是造成Web安全隐患的地方。要防御常见的Web攻击，不仅仅要在用户输入流环节进行控制，比如：过滤、编码、序列化等，还要在根本上解决Web安全问题。本项目针对常见的Web安全SQL注入、XSS攻击、劫持等做了全面的过滤。根本上防御SQL主语使用了预编译SQL语句实现，根本上防御XSS使用了HTML化防止用户输入当做XSS脚本执行。

# 第3章 详细设计

## 3.1 注册

在JSP包中的registerSucceed文件实现学生注册功能。

接受HTML表单传过来的参数： stuId（学号）、stuPassword（密码)、confirmstuPassword（确认密码）、stuName（学生姓名）、stuSex（学生性别）、stuClassId（学生班级号）

判断账号密码是否为空，如果为空则提示错误信息，如果不为空则向下执行

执行sql查询语句查询stuName是否已经注册，如果注册过则提示用户已经存在；如果没有注册过则判断两次密码是否一致，不一致则提示”密码输入不一致”，一致则执行SQL语句向学生信息数据表插入学生信息。

关闭数据库连接释放数据库连接

<%

Context ctx = null;

DataSource ds = null;

Statement stmt = null;

ResultSet rs = null;

Connection con = null;

//去除输入首尾

String stuId = request.getParameter("stuId").trim();

String stuPassword = request.getParameter("stuPassword").trim();

String confirmstuPassword = request.getParameter("confirmstuPassword").trim();

String stuName = request.getParameter("stuName").trim();

String stuSex = request.getParameter("stuSex").trim();

String stuClassId = request.getParameter("stuClassId").trim();

try{

ctx = new InitialContext();

ds = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql");//mysql是在创建连接池时自己创建的名字

con = ds.getConnection();

stmt = con.createStatement();

if(stuId != null){

rs = stmt.executeQuery("select \* from student where stuID='"+stuId+"'");

if(rs.next()){

out.print("用户已经存在 "+"请<a href=\"../html/login.html\">注册</a>");

}else {

if(stuPassword.equals(confirmstuPassword)){

stmt.executeUpdate("insert into student (stuId,stuPassword,stuName,stuSex,stuClassId) values('"+stuId+"','"+stuPassword+"','"+stuName+"','"+stuSex+"','"+stuClassId+"')");

%>

注册成功！！！<br>

三秒后自动跳转到登录页面！！！<br>

如果没有跳转，请点击<a href="Login.jsp">这里</a>！！！

<span style="font-size:24px;"><meta http-equiv="refresh" content="3;URL=Login.jsp"> </span>

<%

}else{

out.print("密码输入不一致!!!<br>"+"重新<a href=\"registered.jsp\">注册</a>");

}

}

}else{

out.print("姓名不能为空");

}

}catch (Exception e){

out.print(e);

}finally {

if(rs!=null)

rs.close();

if(stmt!=null)

stmt.close();

if(con!=null)

con.close();

}

%>

## 3.2 登录注销

在Controller包中的LoginServlet类中重写doGet()、doPost()方法

doGet：

接受View层传过来的参数： username（用户名）、password（密码)、identity（权限认证码）、autologin（是否自动登录标记）、checkcode（验证码）

判断账号密码是否为空，如果为空则提示错误信息，如果不为空则向下执行

判断验证码是否正确，如果不正确提示验证码输入错误，如果正确则向下执行

判断identity参数是学生还是教师。如果为空则提示错误信息；如果是学生则实例化学生类，调用学生登录函数方法；如果是教师则实例化教师类，调用教师登录函数方法。

doPost：直接对doGet进行一次封装操作相同

package Controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import Dao.AdministratorDao;

import Dao.StudentDao;

import Dao.UserDao;

import Model.Administrator;

import Model.Student;

import Model.User;

/\*\*

\* Servlet implementation class LoginServlet

\*/

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

String username = request.getParameter("username");

String password = request.getParameter("password");

String identity = request.getParameter("identity");

String autoLogin = request.getParameter("autologin");

String checkCode = request.getParameter("check\_code");

String savedCode = (String) request.getSession().getAttribute("check\_code");

System.out.println(username + " " + password + " " + checkCode + " "

+ savedCode + " " + identity);

if(username != null && password != null){

System.out.println("账号密码不为空");

//账号密码不为空

if(checkCode.equals(savedCode)) {

//验证码正确

System.out.println("验证码正确");

if(identity !=null) {

if(identity.equals("admin")){ //教师

try{

System.out.println("进入教师");

AdministratorDao adminDao = new AdministratorDao();

Administrator admin = adminDao.find(username);

if (admin != null){

if (admin.getTcPassword().equals(password)) {

System.out.println("ok");

function1(username, password, autoLogin, 1, request, response);

}

} else {

request.setAttribute("errerMsg", "没有该教师");

request.getRequestDispatcher("/Jsp/Login.jsp").forward(request, response);

}

} catch (Exception ee) {

ee.printStackTrace();

}

} else if (identity.equals("student")){ //学生

try{

System.out.println("进入学生");

StudentDao stuDao = new StudentDao();

Student student = stuDao.find(username);

if (student != null) {

if (student.getStuPassword().equals(password)){

System.out.println("ok");

function1(username, password, autoLogin, 2, request, response);

}

} else {

request.setAttribute("errerMsg", "没有该学生");

request.getRequestDispatcher("/Jsp/Login.jsp").forward(request, response);

}

} catch (Exception ee) {

ee.printStackTrace();

}

}

} else {

request.setAttribute("errerMsg", "身份为空");

request.getRequestDispatcher("/Jsp/Login.jsp").forward(request, response);

}

} else {

//验证不正确

request.setAttribute("errerMsg", "验证码错误");

request.getRequestDispatcher("/Jsp/Login.jsp").forward(request, response);

}

} else {

//账号或密码为空

request.setAttribute("errerMsg", "用户名或密码不能为空");

request.getRequestDispatcher("/Jsp/Login.jsp").forward(request, response);

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

登录函数具体实现细节：

获取参数：username、password、autoLogin、num（标记用户是学生还是教师）

判断num的值来判断用户对象是学生还是教师，将用户状态user对象存入session域

实例化UserDao类，插入登录记录到user表

发送自动登录的Cookie：先判断num值来选择发送学生或教师的Cookie，然后将对应的Cookie发送

跳转至业务页面：如果num为1则跳转到教师业务界面；如果num=2则跳转到学生业务界面

private void function1(String username, String password, String autoLogin, int num,

HttpServletRequest request, HttpServletResponse response) throws IOException{

//将用户状态user对象存入session域

User user = new User();

user.setUsername(username);

user.setPassword(password);

if (num == 1) {

user.setIdentity("admin");

} else if (num ==2) {

user.setIdentity("student");

}

request.getSession().setAttribute("user", user);

//登陆记录

UserDao userDao = new UserDao();

userDao.insert(user);

//发送自动登陆的cookie

if(autoLogin != null){

//注意cookie中的密码要加密

if(num == 1){

Cookie cookie = new Cookie("autologin", username + "-" + password + "-" + "1");

cookie.setMaxAge(Integer.parseInt(autoLogin));

cookie.setPath(request.getContextPath());

response.addCookie(cookie);

}else if (num == 2){

Cookie cookie = new Cookie("autologin", username + "-" + password + "-" + "2");

cookie.setMaxAge(Integer.parseInt(autoLogin));

cookie.setPath(request.getContextPath());

response.addCookie(cookie);

}

}

if (num == 1) { //教师

//跳转至首页

response.sendRedirect(request.getContextPath() + "/Jsp/Admin.jsp");

} else if (num == 2){ //学生

//跳转至首页

response.sendRedirect(request.getContextPath() + "/Jsp/Student.jsp");

}

}

## 3.3 自动登录

在Controller包中的AutoLoginFilter类中运用Cookie会话、Filter过滤器技术。

单一的使用Cookie实现自动登录会造成每个Servlet类中出现大量重复的校验程序，所以加入Filter对服务器的所有请求进行拦截。

Filter会对用户的访问请求进行拦截，判断请求是否包含用户自动登录的Cookie。

如果包含，则获取Cookie中的用户名、密码、身份，并验证用户名、密码、身份是否正确。如果正确，则将用户的登录信息封装到User对象存入Session域中，完成自动登录，完成自动登录

详细逻辑流程：

使用Filter对服务器的所有请求进行拦截

获取名为autologin的cookie，如果不为空则进入过滤器

判断num为1则实例化教师Dao层类，执行查询用户名操作然后标记flag为1；判断num为2则实例化学生Dao层类，执行查询用户名操作然后标记flag为1；

判断flag如果为1则获取user对象存入session域，登录成功

放行filter拦截流量

package Controller;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest;

import Dao.AdministratorDao;

import Dao.StudentDao;

import Model.Administrator;

import Model.Student;

import Model.User;

/\*\*

\* Servlet Filter implementation class AutoLoginFilter

\*/

public class AutoLoginFilter implements Filter {

/\*\*

\* @see Filter#destroy()

\*/

public void destroy() {

// TODO Auto-generated method stub

}

/\*\*

\* @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)

\*/

public void doFilter(ServletRequest req, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

HttpServletRequest request = (HttpServletRequest) req;

//获取一个名为autologin的cookie

Cookie[] cookies = request.getCookies();

String autologin = null;

for (int i=0; cookies != null && i<cookies.length; i++){

if("autologin".equals(cookies[i].getName())){

//找到了指定的cookie

autologin = cookies[i].getValue();

break;

}

}

if(autologin != null){

//做自动登陆

System.out.println("进入过滤器。");

String [] parts = autologin.split("-");

String username = parts[0];

String password = parts[1];

String num = parts[2];

User user = new User();

int flag = 0;

if(num == "1"){

AdministratorDao adminDao = new AdministratorDao();

Administrator admin = adminDao.find(username);

if(admin != null) {

if (admin.getTcPassword().equals(password)){

flag = 1;

user.setIdentity("admin");

}

}

} else {

StudentDao stuDao = new StudentDao();

Student stu = stuDao.find(username);

if (stu != null){

if (stu.getStuPassword().equals(password)){

flag = 1;

user.setIdentity("student");

}

}

}

//检查用户名和密码

if (flag == 1 ){

//登陆成功，将用户状态user对象存入session域

user.setUsername(username);

user.setPassword(password);

request.getSession().setAttribute("user", user);

}

}

//放行

chain.doFilter(request, response);

}

/\*\*

\* @see Filter#init(FilterConfig)

\*/

public void init(FilterConfig fConfig) throws ServletException {

// TODO Auto-generated method stub

}

}

## 3.4 成绩增删改查实现

### 3.4.1 增加学生成绩

在Controller包中的AddScoreServlet类中重写doGet()、doPost()方法

业务逻辑流程：

获取stuId、stuName、courseId、tcId、courceGrade、classId参数

实例化StudentDao、CourseDao、AdministratorDao、classesDao、Gradedao，然后依次执行Dao层的find方法执行查询操作

实例化grade对象

获取form表单传递的添加值调用Model层grade类的set方法，判断姓名、cource、admin、classes等是否为空，不为空则执行Dao层gradeDao类的insert()方法插入数据到grade数据表

package Controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import Dao.AdministratorDao;

import Dao.ClassesDao;

import Dao.CourseDao;

import Dao.GradeDao;

import Dao.StudentDao;

import Model.Administrator;

import Model.Classes;

import Model.Course;

import Model.Grade;

import Model.Student;

/\*\*

\* Servlet implementation class AddScoreServlet

\*/

public class AddScoreServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=utf-8");

String stuId = request.getParameter("stuId");

String stuName = request.getParameter("stuName");

String courseId = request.getParameter("courseId");

String tcId = request.getParameter("tcId");

String courseGrade = request.getParameter("courseGrade");

String classId = request.getParameter("classId");

try{

StudentDao stuDao = new StudentDao();

CourseDao courseDao = new CourseDao();

AdministratorDao adminDao = new AdministratorDao();

ClassesDao classesDao = new ClassesDao();

GradeDao gradeDao = new GradeDao();

Student stu = stuDao.find(stuId);

Course course = courseDao.find(courseId);

Administrator admin = adminDao.find(tcId);

Classes classes = classesDao.find(classId);

Grade grade = new Grade();

grade.setStuId(stuId);

grade.setStuName(stuName);

grade.setCourseId(courseId);

grade.setTcId(tcId);

grade.setCourseGrade(courseGrade);

grade.setClassId(classId);

if (stu.getStuName() == null) {

request.setAttribute("errerMsg", "姓名为空");

request.getRequestDispatcher("/Jsp/AddScore.jsp").forward(request, response);

}

if(stu != null || stu.getStuName() == stuName){

if(course != null){

if(admin != null){

if(classes != null){

if(gradeDao.insert(grade)){

response.sendRedirect(request.getContextPath() + "/Jsp/AddScore.jsp");

}else{

request.setAttribute("errerMsg", "添加失败");

request.getRequestDispatcher("/Jsp/AddScore.jsp").forward(request, response);

}

}else{

request.setAttribute("errerMsg", "没有该班级");

request.getRequestDispatcher("/Jsp/AddScore.jsp").forward(request, response);

}

}else{

request.setAttribute("errerMsg", "没有该教师ID");

request.getRequestDispatcher("/Jsp/AddScore.jsp").forward(request, response);

}

} else {

request.setAttribute("errerMsg", "没有该课程ID");

request.getRequestDispatcher("/Jsp/AddScore.jsp").forward(request, response);

}

}

} catch (Exception ee) {

ee.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

### 3.4.2 删除学生成绩

在Controller包中的DelScoreServlet类中重写doGet()、doPost()方法

业务逻辑流程：

1．获取stuId、stuName、courseId、tcId、classId、grade参数

2．实例化StudentDao、CourseDao、AdministratorDao、classesDao、Gradedao，然后依次执行Dao层的find方法执行查询操作

3.实例化grade对象，获取form表单传递的添加值调用Model层grade类的set方法

4.获取成绩列表实例化GradeDao对象调用delete()方法循环删除指定stuId的所有信息。如果删除失败则报错：“删除失败”。

package Controller;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import Dao.GradeDao;

import Model.Grade;

/\*\*

\* Servlet implementation class DelScoreServlet

\*/

public class DelScoreServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=utf-8");

String stuId = request.getParameter("stuId");

String stuName = request.getParameter("stuName");

String courseId = request.getParameter("courseId");

String tcId = request.getParameter("tcId");

String classId = request.getParameter("classId");

String grade = request.getParameter("grade");

Grade grades = new Grade();

grades.setStuId(stuId);

grades.setStuName(stuName);

grades.setCourseId(courseId);

grades.setTcId(tcId);

grades.setClassId(classId);

grades.setCourseGrade(grade);

HttpSession session = request.getSession();

ArrayList<Grade> list = (ArrayList<Grade>) session.getAttribute("grade");

try {

GradeDao gradeDao = new GradeDao();

if (gradeDao.delete(grades)) {

for(int j=0;j<list.size();j++){

if(list.get(j) == null){

list.remove(j);

}

if(list.get(j)!=null && list.get(j).getStuId().equals(stuId)

&& list.get(j).getCourseId().equals(courseId)

&& list.get(j).getStuName().equals(stuName)

&& list.get(j).getTcId().equals(tcId)

&& list.get(j).getCourseGrade().equals(grade)

&& list.get(j).getClassId().equals(classId)){

list.remove(j);

break;

}

}

response.sendRedirect(request.getContextPath() + "/Jsp/UpdDelScore.jsp");

} else {

request.setAttribute("errerMsg", "删除失败");

request.getRequestDispatcher("/Jsp/UpdDelScore.jsp").forward(request, response);

}

} catch (Exception ee) {

ee.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

### 3.4.2 更改学生成绩

在Controller包中的UpdScoreServlet类中重写doGet()、doPost()方法

业务逻辑流程：

1．获取stuId、stuName、courseId、tcId、classId、grade参数

2.实例化Grade对象，获取form表单传递的更改值调用Model层grade类的set方法

3.获取成绩列表实例化GradeDao对象调用update()方法更新指定stuId的所有信息。如果删除失败则报错：“更新失败”。

package Controller;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import Dao.GradeDao;

import Model.Grade;

/\*\*

\* Servlet implementation class UpdScoreServlet

\*/

public class UpdScoreServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charcet=utf-8");

String stuId = request.getParameter("stuId");

String stuName = request.getParameter("stuName");

String courseId = request.getParameter("courseId");

String tcId = request.getParameter("tcId");

String classId = request.getParameter("classId");

String grade = request.getParameter("grade");

Grade grades = new Grade();

grades.setStuId(stuId);

grades.setStuName(stuName);

grades.setCourseId(courseId);

grades.setTcId(tcId);

grades.setClassId(classId);

grades.setCourseGrade(grade);

HttpSession session = request.getSession();

try{

GradeDao gradeDao = new GradeDao();

if (gradeDao.update(grades)) {

ArrayList<Grade> list = (ArrayList<Grade>) gradeDao.find(stuId);

session.setAttribute("grade", list);

response.sendRedirect(request.getContextPath() + "/Jsp/UpdDelScore.jsp");

} else {

request.setAttribute("errerMsg", "更新失败");

request.getRequestDispatcher("/Jsp/UpdDelScore.jsp").forward(request, response);

}

} catch(Exception ee) {

ee.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

### 3.4.2 查询学生成绩

在Controller包中的FindScoreServlet类中重写doGet()、doPost()方法

业务逻辑流程：

1．获取session，实例化user判断已登录的用户是学生还是教师。

2.实例化Dao层的studentDao对象根据传递的参数查找学生id，如果学生存在则进行下一步，否则抛出异常信息：“没有该学生”

3.实例化Dao层的GradeDao对象，获取成绩列表传递给前端渲染输出

package Controller;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import Dao.GradeDao;

import Dao.StudentDao;

import Model.Student;

import Model.User;

import Model.Grade;

/\*\*

\* Servlet implementation class FindScoreServlet

\*/

public class FindScoreServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=utf-8");

HttpSession session = request.getSession();

User users = (User)request.getSession().getAttribute("user");

String id = null;

if (users.getIdentity() == "admin") {

id = request.getParameter("id");

} else if (users.getIdentity() == "student") {

id = users.getUsername();

}

String func = request.getParameter("func");

try {

StudentDao stuDao = new StudentDao();

Student student = stuDao.find(id);

if(student != null){

try {

GradeDao gradeDao = new GradeDao();

ArrayList<Grade> list = (ArrayList<Grade>) gradeDao.find(id);

session.setAttribute("grade", list);

if(func.equals("show")){

response.sendRedirect(request.getContextPath() + "/Jsp/FindScore.jsp");

}else{

response.sendRedirect(request.getContextPath() + "/Jsp/UpdDelScore.jsp");

}

} catch(Exception ee) {

ee.printStackTrace();

}

}else{

request.setAttribute("errerMsg", "没有该学生");

if(func.equals("show")){

request.getRequestDispatcher("/Jsp/FindScore.jsp").forward(request, response);

}else{

request.getRequestDispatcher("/Jsp/UpdDelScore.jsp").forward(request, response);

}

}

} catch (Exception ee) {

ee.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

## 3.5 验证码

在Controller包中的CheckServlet类中实现验证码技术。先产生4位随机数和图片，然后将图像以字节流的形式输出至客户端，最后把随机数存储至Session域中。

业务逻辑流程：

1.设置浏览器不要缓存图片

2.创建内存图像没并获得其图形上下文

3.产生随机的认证码

4.产生图片

5.结束图像的绘制过程，完成图像

6.将图像输出到客户端

7.将当前验证码存储到Session中

package Controller;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.awt.\*;

/\*\*

\* Servlet implementation class CheckServlet

\*/

public class CheckServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

private static int WIDTH = 60;

private static int HEIGHT = 20;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession();

response.setContentType("image/jpeg");

ServletOutputStream sos = response.getOutputStream();

//设置浏览器不要缓存图片

response.setHeader("Pragma", "No-cache");

response.setHeader("Cache-Control", "no-cache");

response.setDateHeader("Expires", 0);

//创建内存图像并获得其图形上下文

BufferedImage image = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE\_INT\_RGB);

Graphics g = image.getGraphics();

//产生随机的认证码

char[] rands = generateCheckCode();

//产生图片

drawBackground(g);

drawRands(g, rands);

//结束图像的绘制过程，完成图像

g.dispose();

//将图像输出到客户端

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ImageIO.write(image, "JPEG", bos);

byte[] buf = bos.toByteArray();

response.setContentLength(buf.length);

//下面语句也可写成：bos.writeTo(sos);

sos.write(buf);

bos.close();

sos.close();

//将当前验证码存入到Session中

session.setAttribute("check\_code", new String(rands));

//直接使用下面的代码将有问题，session对象在提交响应前获取

// request.getSession().setAttribute("check\_code", new String(rands));

}

private void drawRands(Graphics g, char[] rands) {

g.setColor(Color.BLACK);

g.setFont(new Font(null, Font.ITALIC|Font.BOLD, 18));

//在不同的高度上输出验证码的每个字符

g.drawString("" + rands[0], 1, 17);

g.drawString("" + rands[1], 16, 15);

g.drawString("" + rands[2], 31, 18);

g.drawString("" + rands[3], 46, 16);

System.out.println(rands);

}

private void drawBackground(Graphics g) {

//画背景

g.setColor(new Color(0xDCDCDC));

g.fillRect(0, 0, WIDTH, HEIGHT);

//随机产生120个干扰点

for (int i=0; i<120; i++)

{

int x = (int)(Math.random() \* WIDTH);

int y = (int)(Math.random() \* HEIGHT);

int red = (int)(Math.random() \* 255);

int green = (int)(Math.random() \* 255);

int blue= (int)(Math.random() \* 255);

g.setColor(new Color(red, green, blue));

g.drawOval(x, y, 1, 0);

}

}

//生成一个4字符的验证码

private char[] generateCheckCode() {

//定义验证码的字符表

String chars = "0123456789zxcvbnmasdfghjklqwertyuiop";

char [] rands = new char[4];

for(int i=0; i<4; i++)

{

int rand = (int)(Math.random() \* 36);

rands[i] = chars.charAt(rand);

}

return rands;

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

## 3.6 防SQL注入、XSS攻击

在Controller包中的XssAndSqlFilter类和XssAndSqlHttpServlet类实现截取所有输入流量进行Filter过滤

业务逻辑流程：

XssAndSqlFilter类主要对请求拦截封装，XssAndSqlHttpServlet类实现主要业务逻辑：

获取用户的输入参数，将容易引起xss和sql注入漏洞的半角字符替换转义成全角字符

获取原始请求的静态方法，对易引发xss常见的字符进行匹配过滤

package Controller;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletRequest;

/\*\*

\* XssAndSqlFilter

\*/

public class XssAndSqlFilter implements Filter {

@Override

public void destroy() {

// TODO Auto-generated method stub

}

@Override

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

XssAndSqlHttpServletRequestWrapper xssRequest = new XssAndSqlHttpServletRequestWrapper((HttpServletRequest) request);

chain.doFilter(xssRequest, response);

}

@Override

public void init(FilterConfig arg0) throws ServletException {

// TODO Auto-generated method stub

}

}

package Controller;

import java.util.regex.Pattern;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletRequestWrapper;

/\*\*

\* XssAndSqlHttpServletRequestWrapper

\*/

public class XssAndSqlHttpServletRequestWrapper extends HttpServletRequestWrapper {

HttpServletRequest orgRequest = null;

public XssAndSqlHttpServletRequestWrapper(HttpServletRequest request) {

super(request);

orgRequest = request;

}

/\*\*

\* 覆盖getParameter方法，将参数名和参数值都做xss & sql过滤。<br/>

\* 如果需要获得原始的值，则通过super.getParameterValues(name)来获取<br/>

\* getParameterNames,getParameterValues和getParameterMap也可能需要覆盖

\*/

@Override

public String getParameter(String name) {

String value = super.getParameter(xssEncode(name));

if (value != null) {

value = xssEncode(value);

}

return value;

}

/\*\*

\* 覆盖getHeader方法，将参数名和参数值都做xss & sql过滤。<br/>

\* 如果需要获得原始的值，则通过super.getHeaders(name)来获取<br/>

\* getHeaderNames 也可能需要覆盖

\*/

@Override

public String getHeader(String name) {

String value = super.getHeader(xssEncode(name));

if (value != null) {

value = xssEncode(value);

}

return value;

}

//将容易引起xss & sql漏洞的半角字符直接替换成全角字符

private static String xssEncode(String s) {

if (s == null || s.isEmpty()) {

return s;

}else{

s = stripXSSAndSql(s);

}

StringBuilder sb = new StringBuilder(s.length() + 16);

for (int i = 0; i < s.length(); i++) {

char c = s.charAt(i);

switch (c) {

case '>':

sb.append("＞");// 转义大于号

break;

case '<':

sb.append("＜");// 转义小于号

break;

case '\'':

sb.append("＇");// 转义单引号

break;

case '\"':

sb.append("＂");// 转义双引号

break;

case '&':

sb.append("＆");// 转义&

break;

case '#':

sb.append("＃");// 转义#

break;

default:

sb.append(c);

break;

}

}

return sb.toString();

}

//获取最原始的request

public HttpServletRequest getOrgRequest() {

return orgRequest;

}

///获取最原始的request的静态方法

public static HttpServletRequest getOrgRequest(HttpServletRequest req) {

if (req instanceof XssAndSqlHttpServletRequestWrapper) {

return ((XssAndSqlHttpServletRequestWrapper) req).getOrgRequest();

}

return req;

}

//防止xss跨脚本攻击

public static String stripXSSAndSql(String value) {

if (value != null) {

// NOTE: It's highly recommended to use the ESAPI library and

// uncomment the following line to

// avoid encoded attacks.

// value = ESAPI.encoder().canonicalize(value);

// Avoid null characters

/\*\* value = value.replaceAll("", "");\*\*\*/

// Avoid anything between script tags

Pattern scriptPattern = Pattern.compile("<[\r\n| | ]\*script[\r\n| | ]\*>(.\*?)</[\r\n| | ]\*script[\r\n| | ]\*>", Pattern.CASE\_INSENSITIVE);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid anything in a src="http://www.yihaomen.com/article/java/..." type of e-xpression

scriptPattern = Pattern.compile("src[\r\n| | ]\*=[\r\n| | ]\*[\\\"|\\\'](.\*?)[\\\"|\\\']", Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);

value = scriptPattern.matcher(value).replaceAll("");

// Remove any lonesome </script> tag

scriptPattern = Pattern.compile("</[\r\n| | ]\*script[\r\n| | ]\*>", Pattern.CASE\_INSENSITIVE);

value = scriptPattern.matcher(value).replaceAll("");

// Remove any lonesome <script ...> tag

scriptPattern = Pattern.compile("<[\r\n| | ]\*script(.\*?)>", Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid eval(...) expressions

scriptPattern = Pattern.compile("eval\\((.\*?)\\)", Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid e-xpression(...) expressions

scriptPattern = Pattern.compile("e-xpression\\((.\*?)\\)", Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid javascript:... expressions

scriptPattern = Pattern.compile("javascript[\r\n| | ]\*:[\r\n| | ]\*", Pattern.CASE\_INSENSITIVE);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid vbscript:... expressions

scriptPattern = Pattern.compile("vbscript[\r\n| | ]\*:[\r\n| | ]\*", Pattern.CASE\_INSENSITIVE);

value = scriptPattern.matcher(value).replaceAll("");

// Avoid onload= expressions

scriptPattern = Pattern.compile("onload(.\*?)=", Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);

value = scriptPattern.matcher(value).replaceAll("");

}

return value;

}

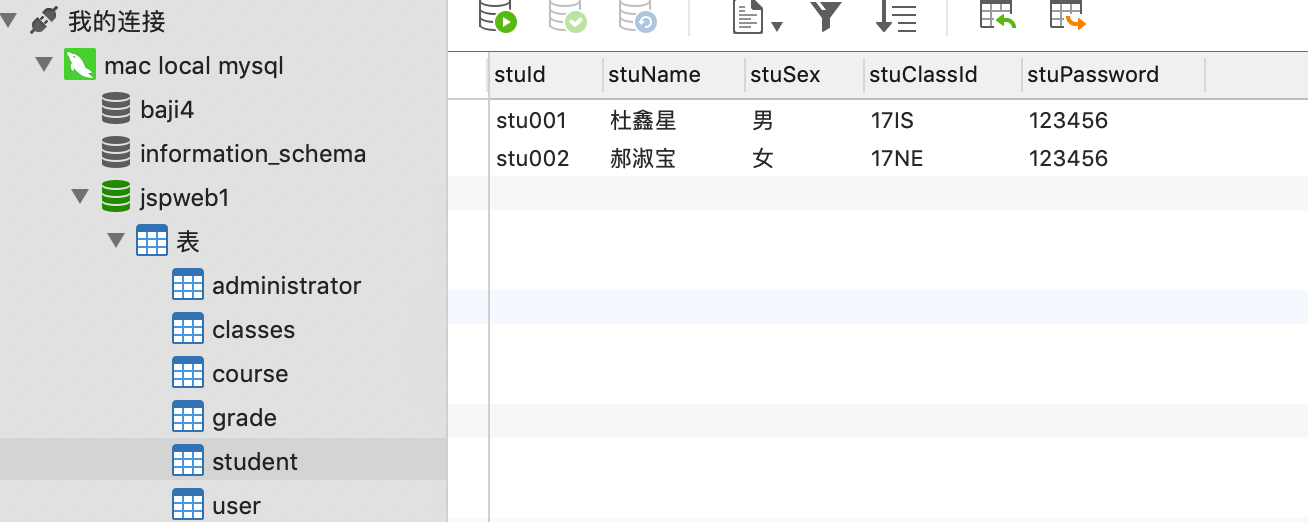
}

# 第4章 功能演示

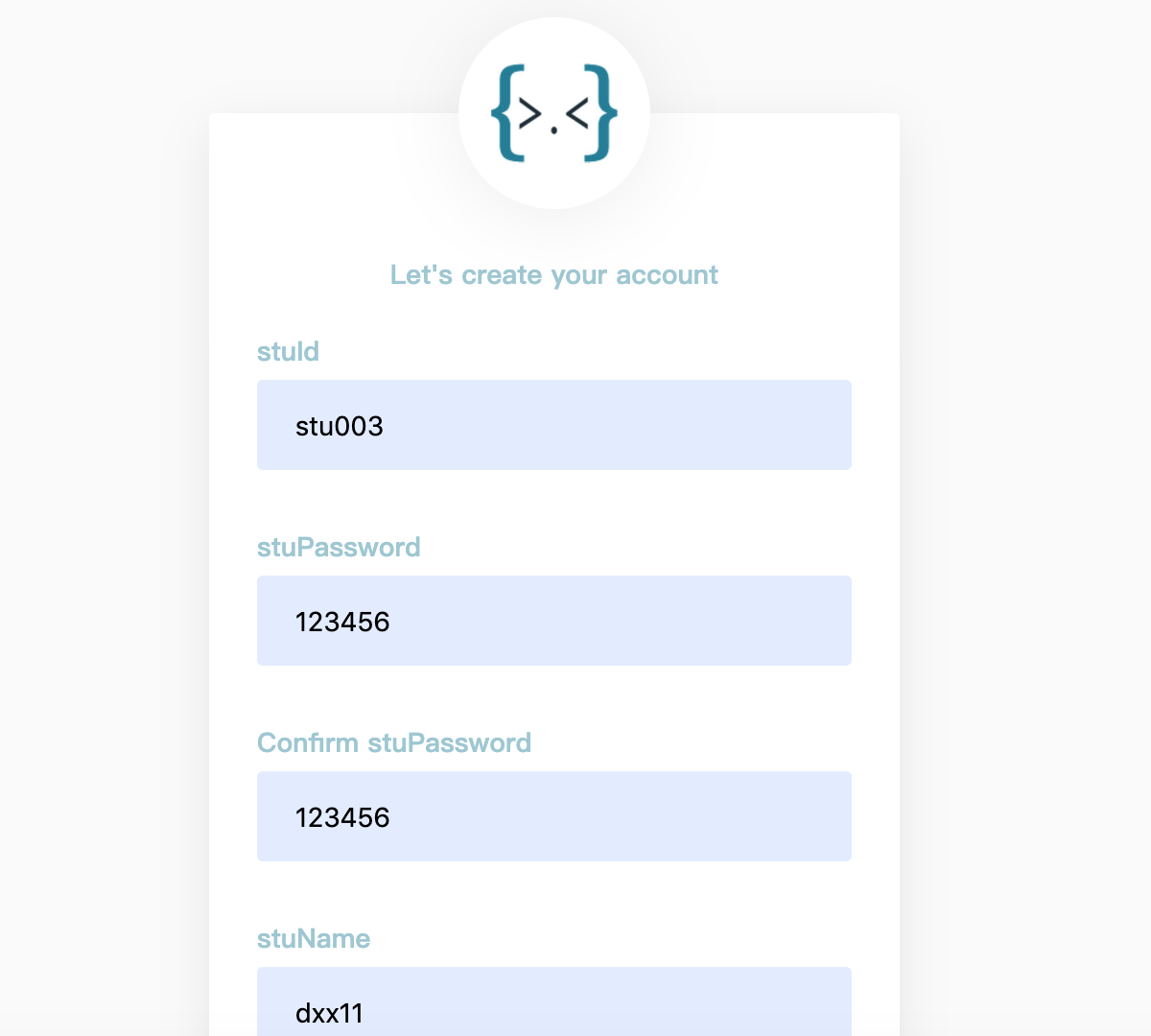
## 4.1 注册

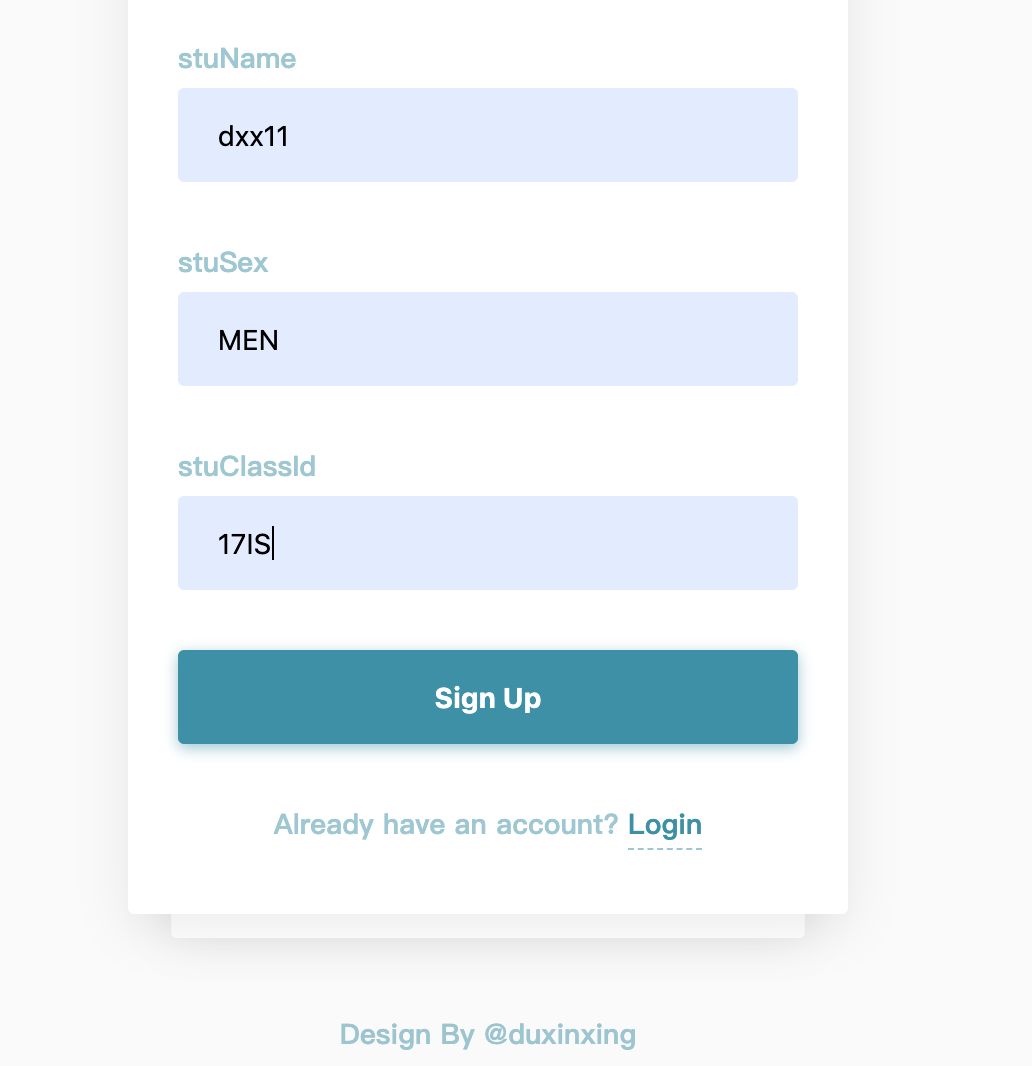
### 4.1.1 注册前

注册前数据库中student表数据：



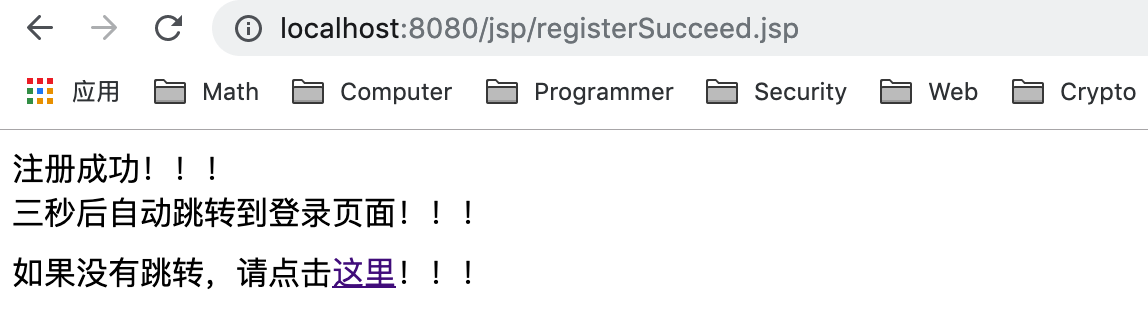
### 4.1.2 注册页面





### 4.1.3 注册后

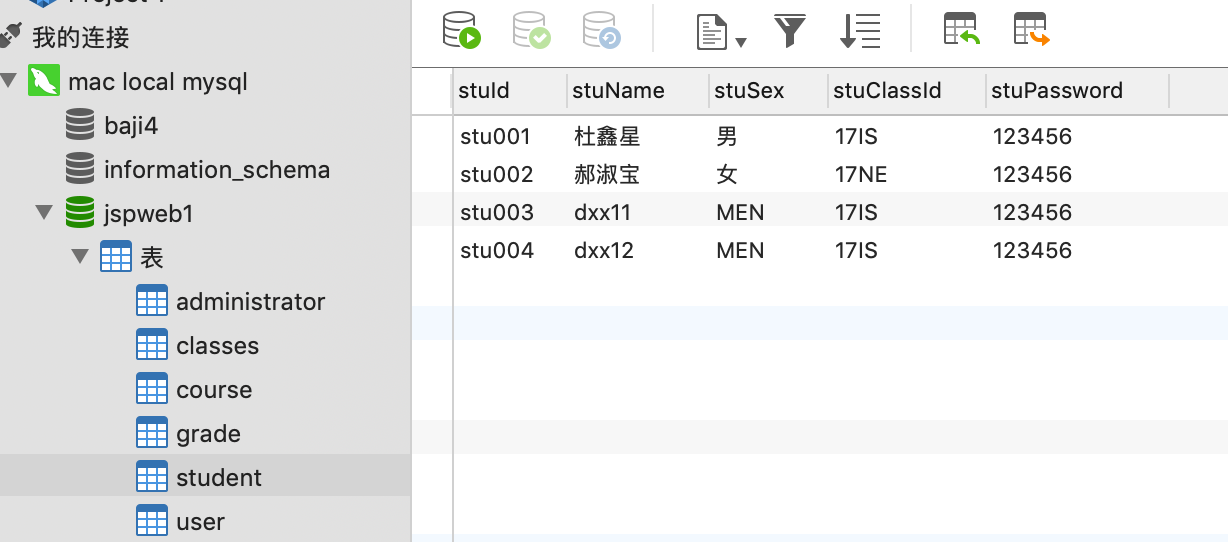
如果用户没有被注册：



如果用户已经注册：



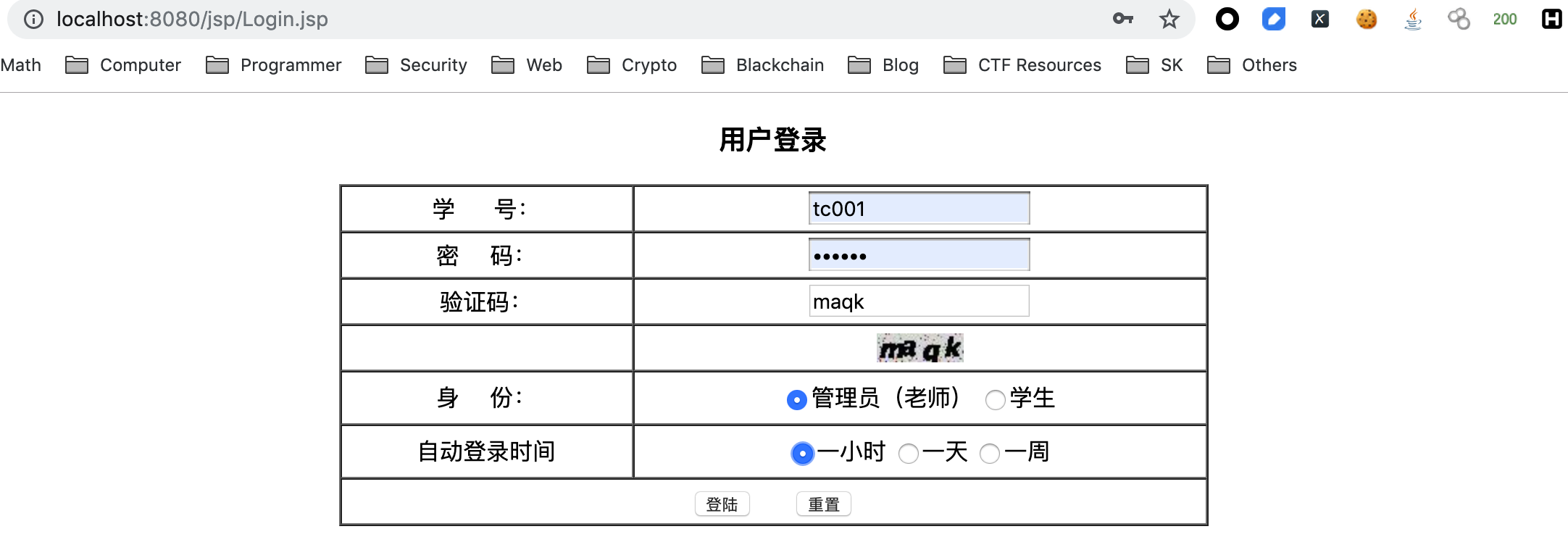
数据库信息更新：



## 4.2 登录注销

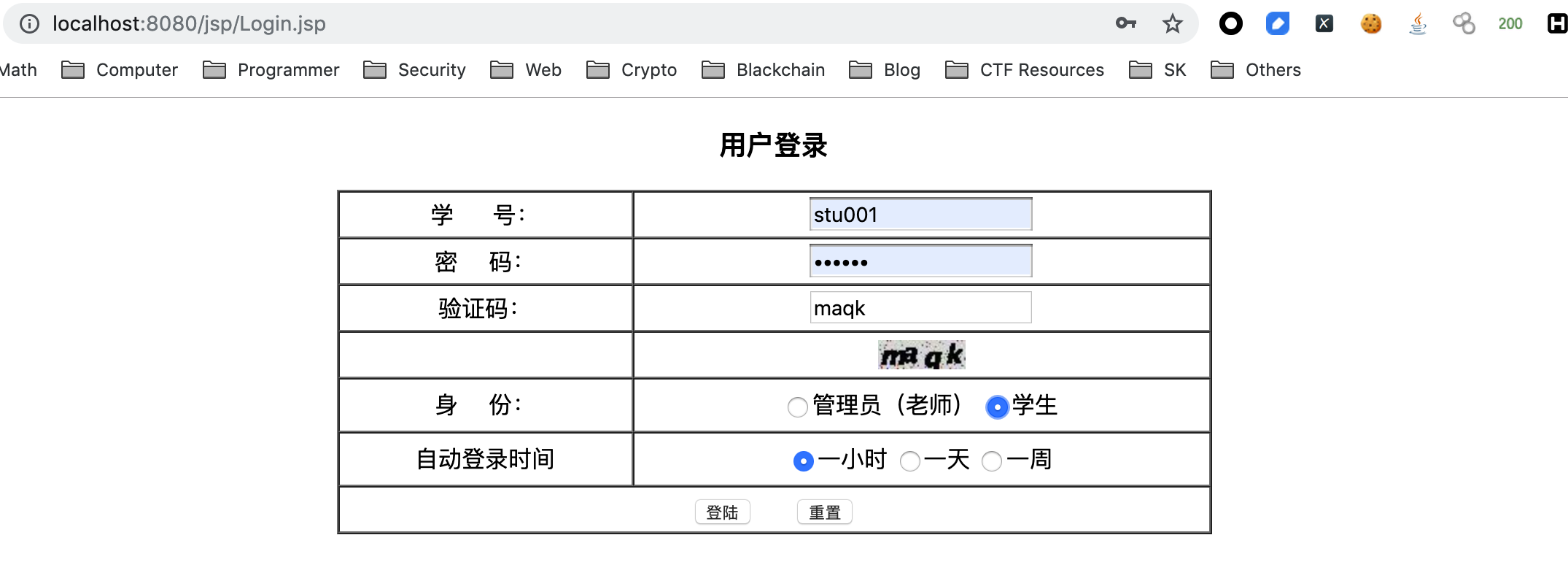
### 4.2.1 登录

教师登录：





学生登录：



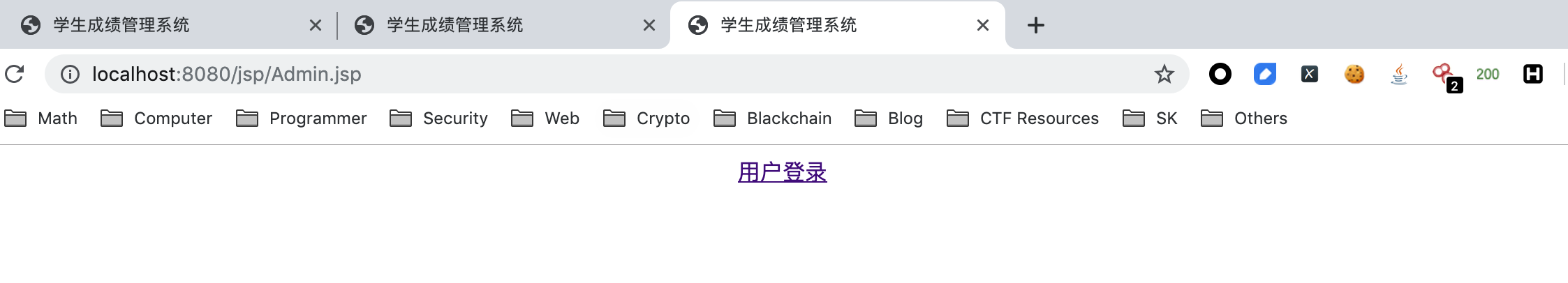


### 4.2.2 注销

注销前打开新网页直接访问Admin.jsp页面因设置了自动登录所以会自动进入到教师页面



注销后再打开新网页直接访问Admin.jsp页面因Cookie已经被删除故无法自动登录，提示用户登录



## 4.3 自动登录

登录页面有选择自动登录的时间选项：



已经登录的教师页面如下：



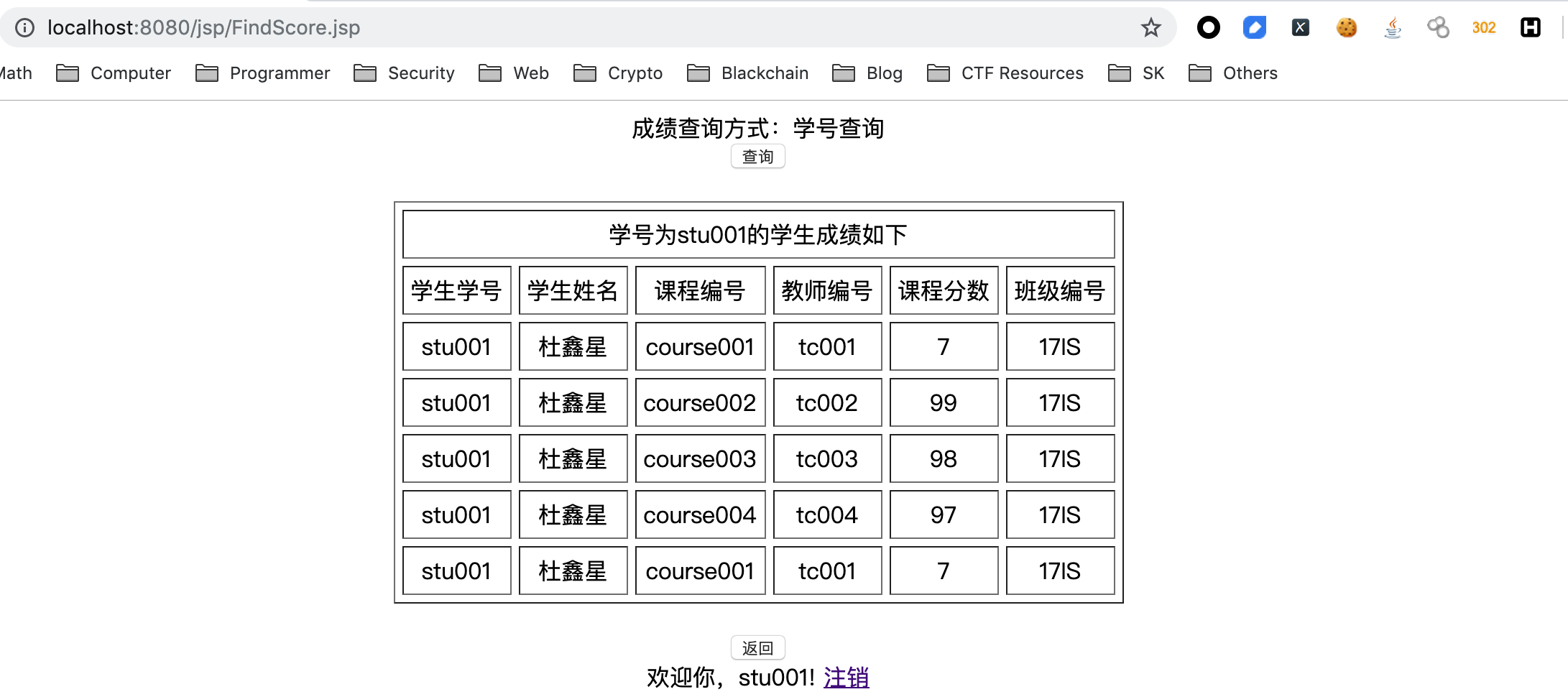
打开新网页直接访问URL会自动登录：



## 4.4 成绩的增删改查

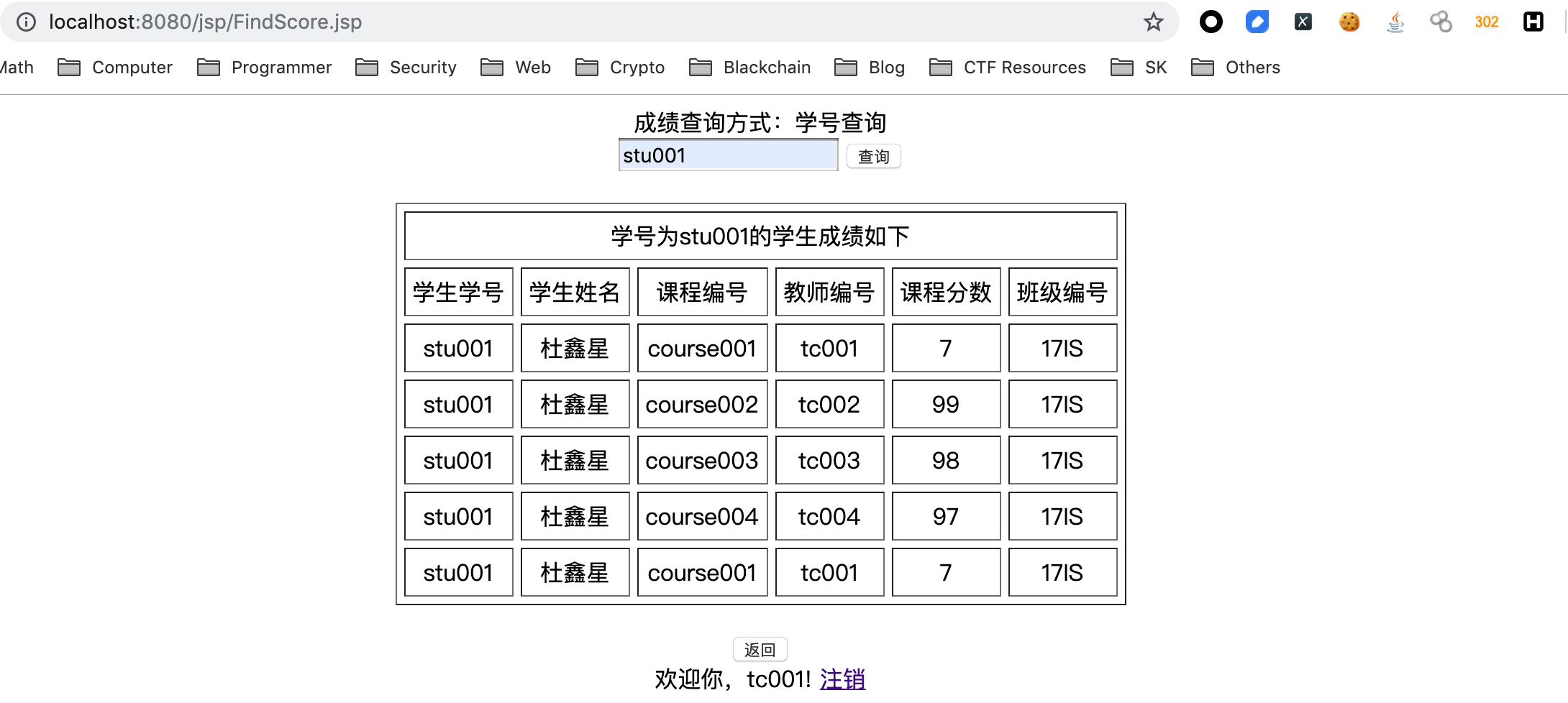
### 4.4.1 学生查询成绩

不用输入信息，直接根据登录的学生信息查询：



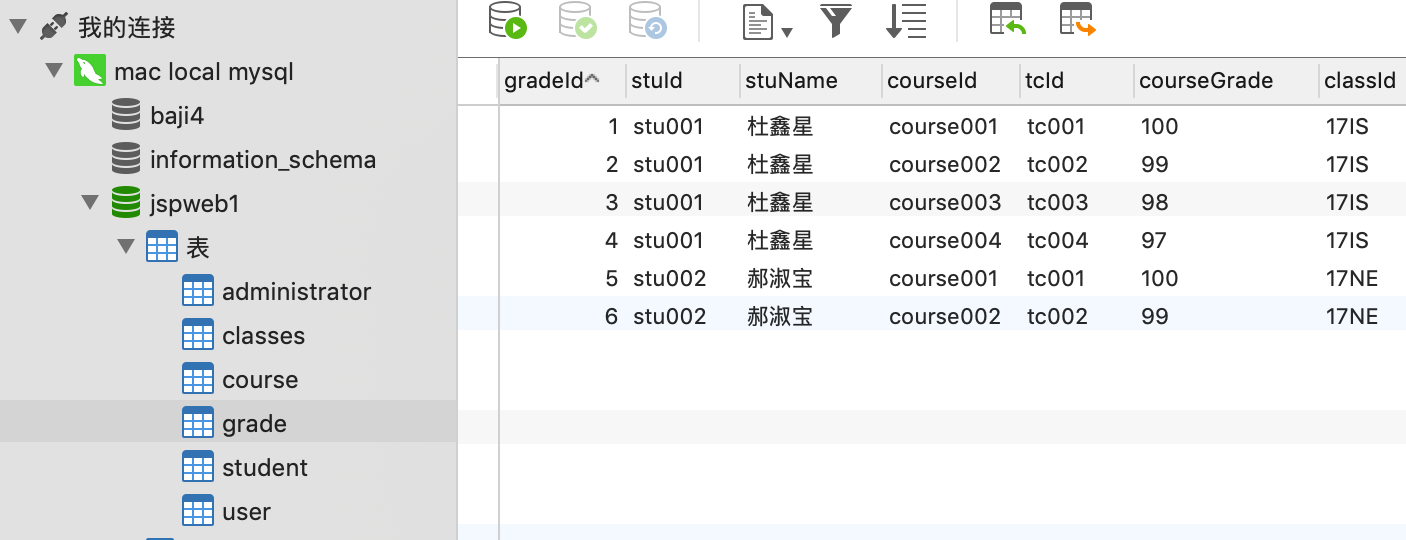
### 4.4.2 教师查询成绩

输入学生学号进行成绩查询：



### 4.4.3 教师添加成绩

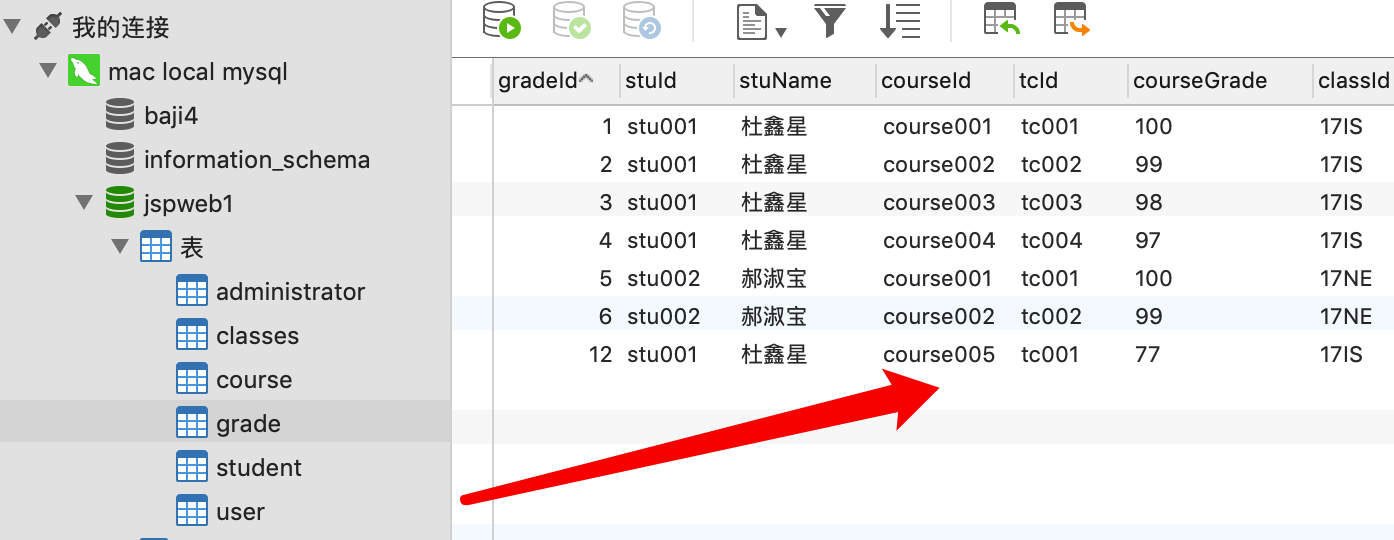
添加成绩前grade数据表信息：



添加成绩页面：

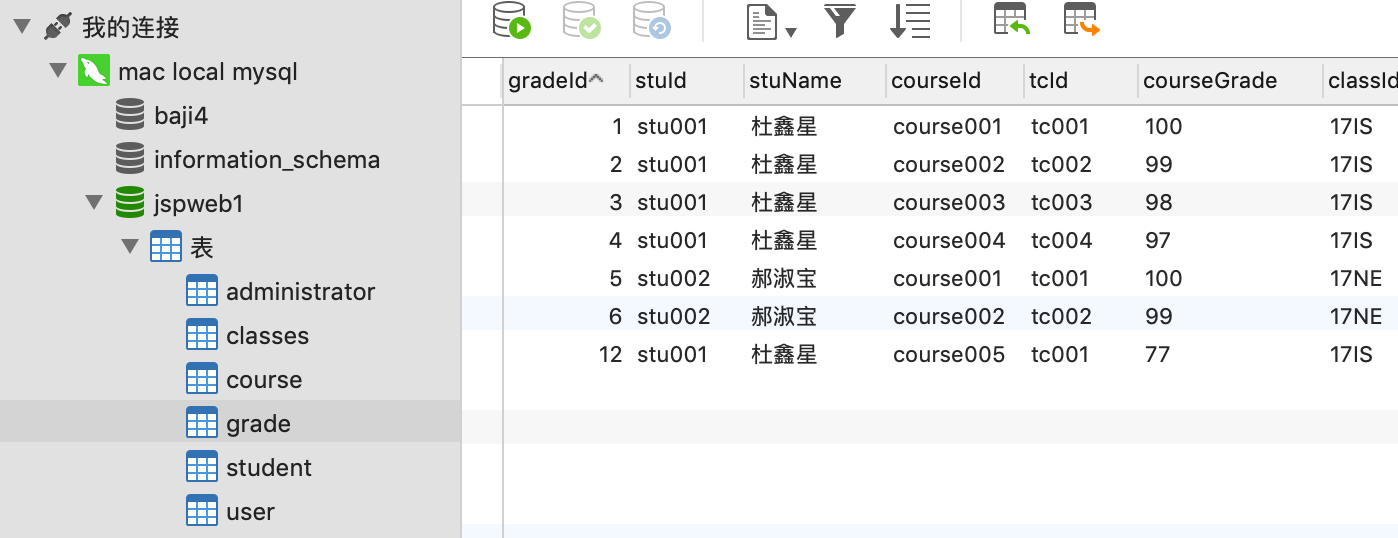


添加成绩后grade数据表信息：

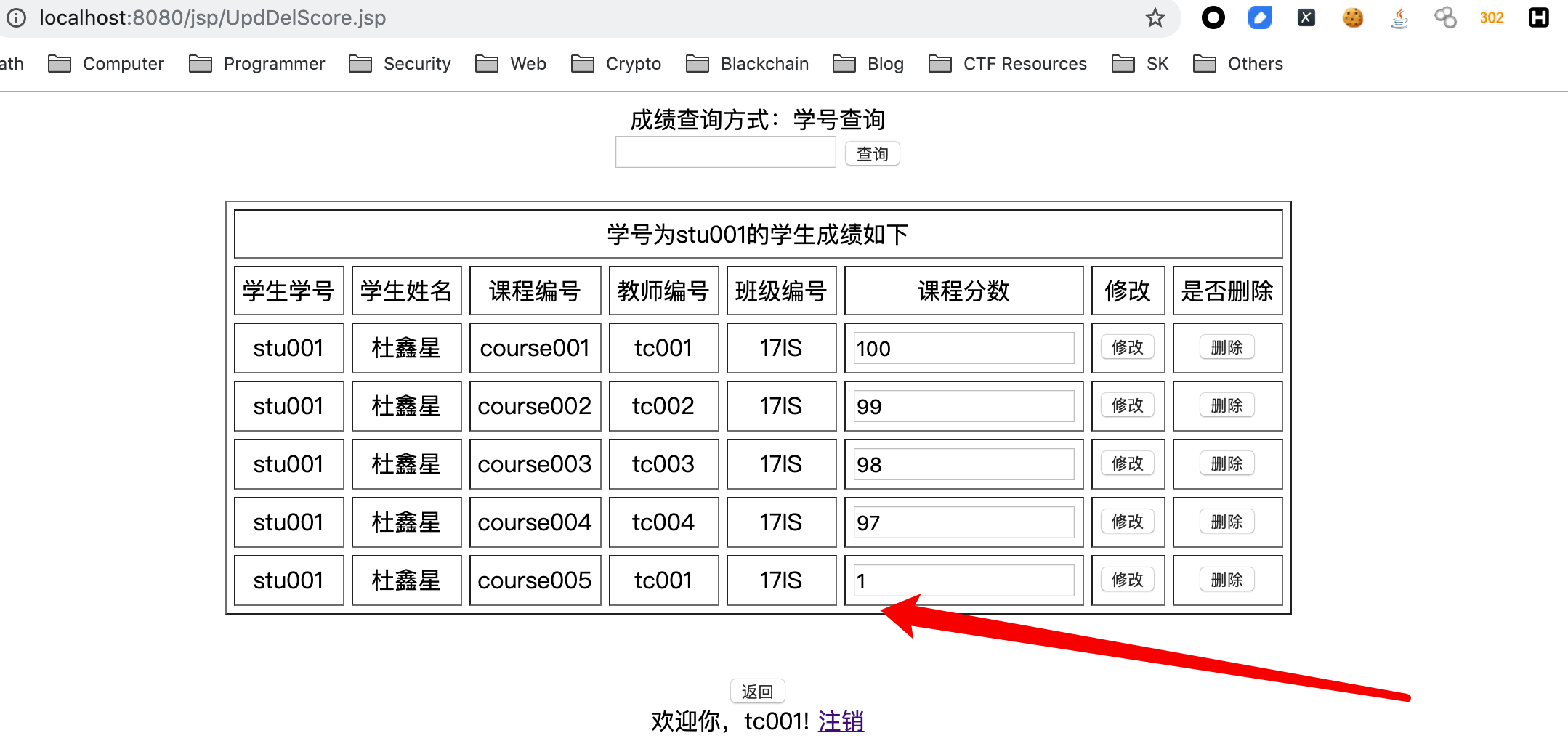


### 4.4.4 教师修改成绩

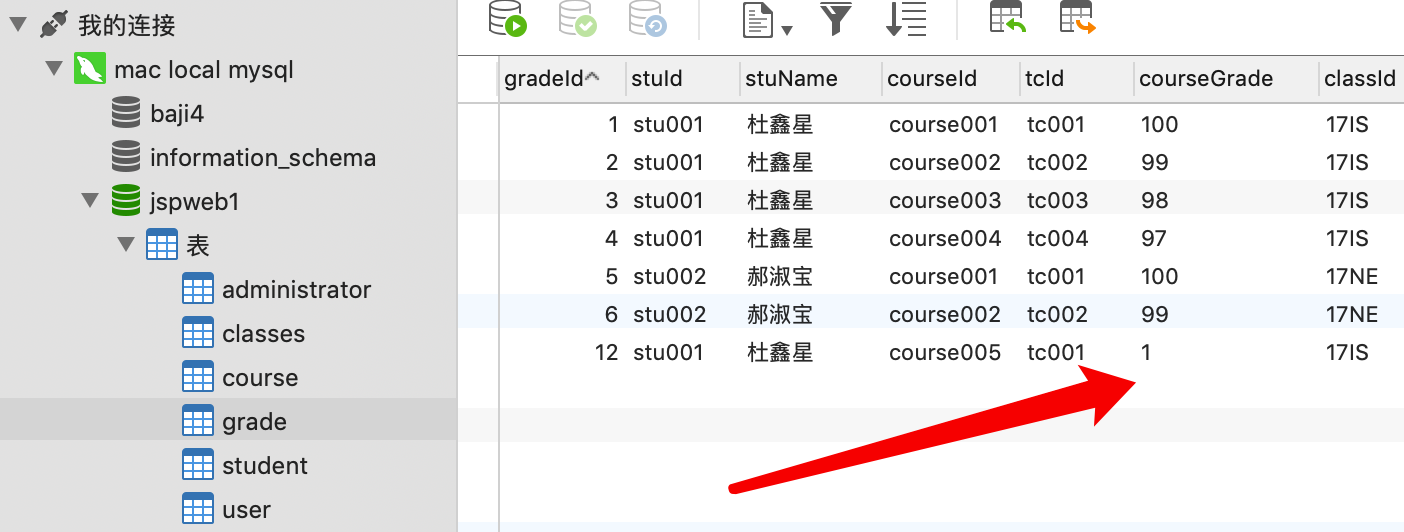
修改成绩前grade数据表信息：



修改成绩页面：把刚才添加进入的课程成绩改为1

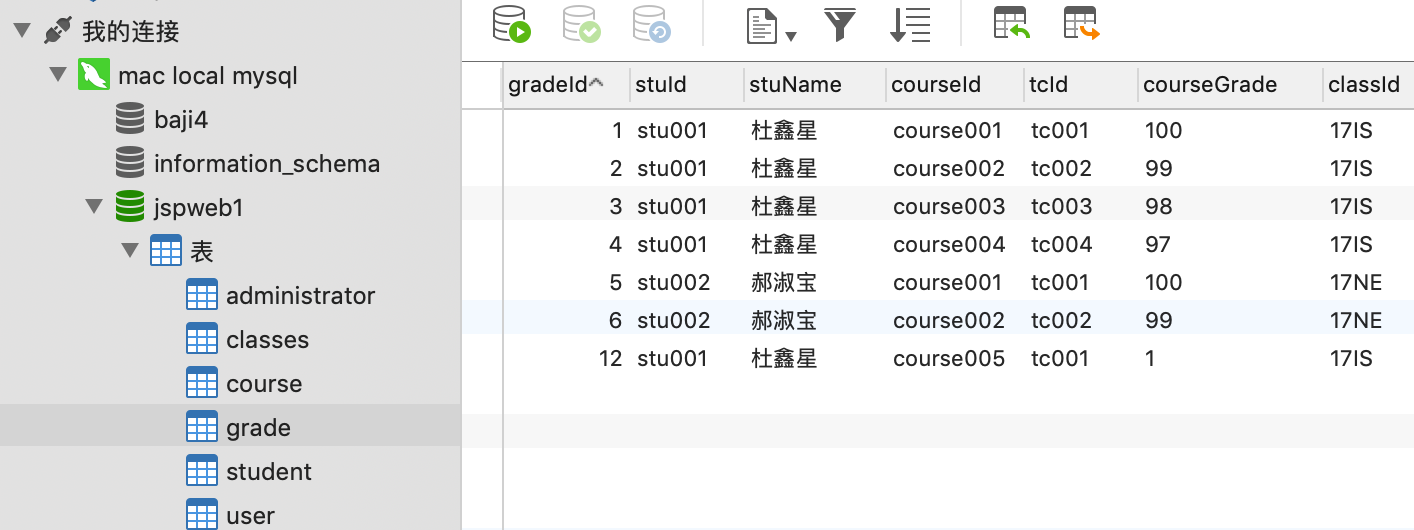


修改成绩后grade数据库信息：

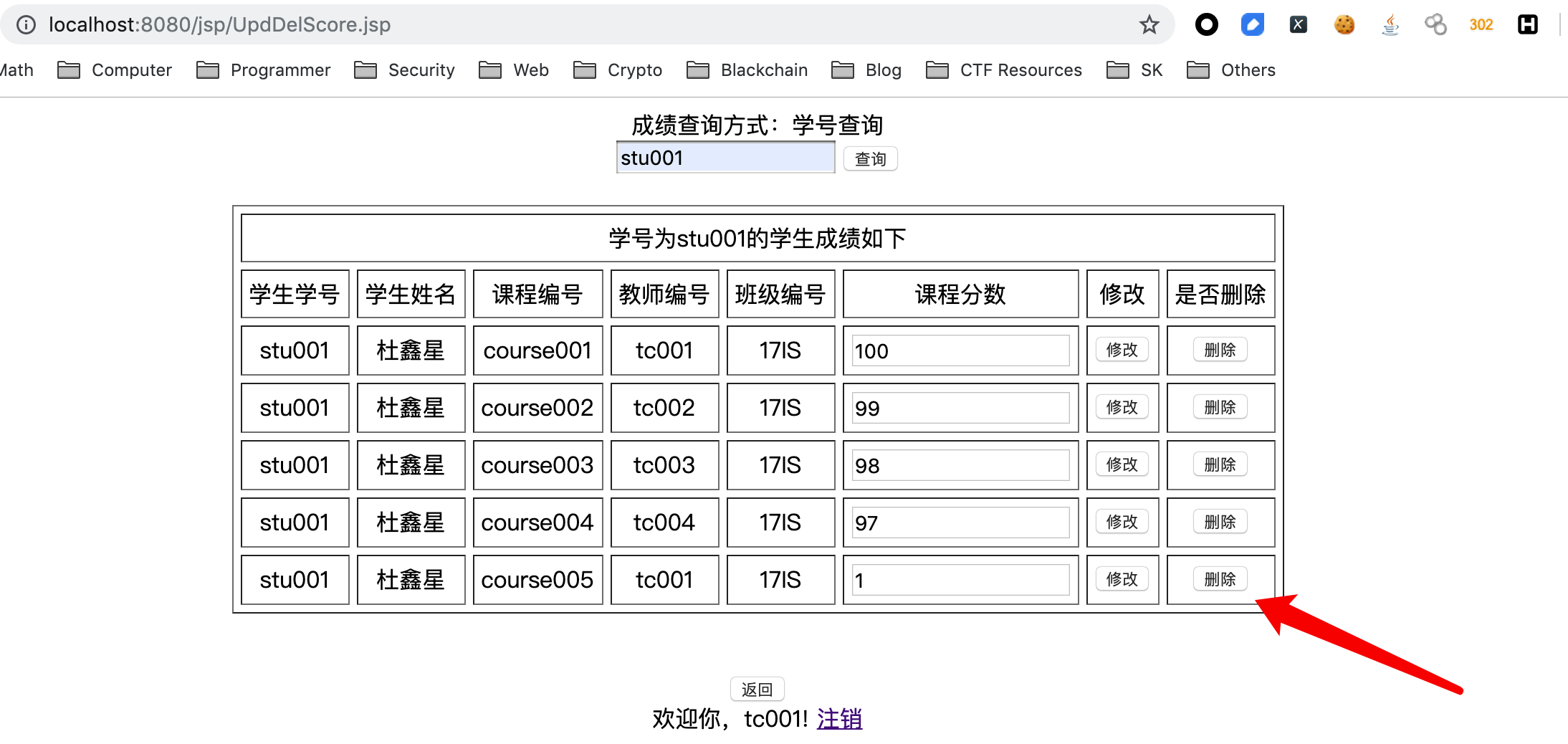


### 4.4.5 教师删除成绩

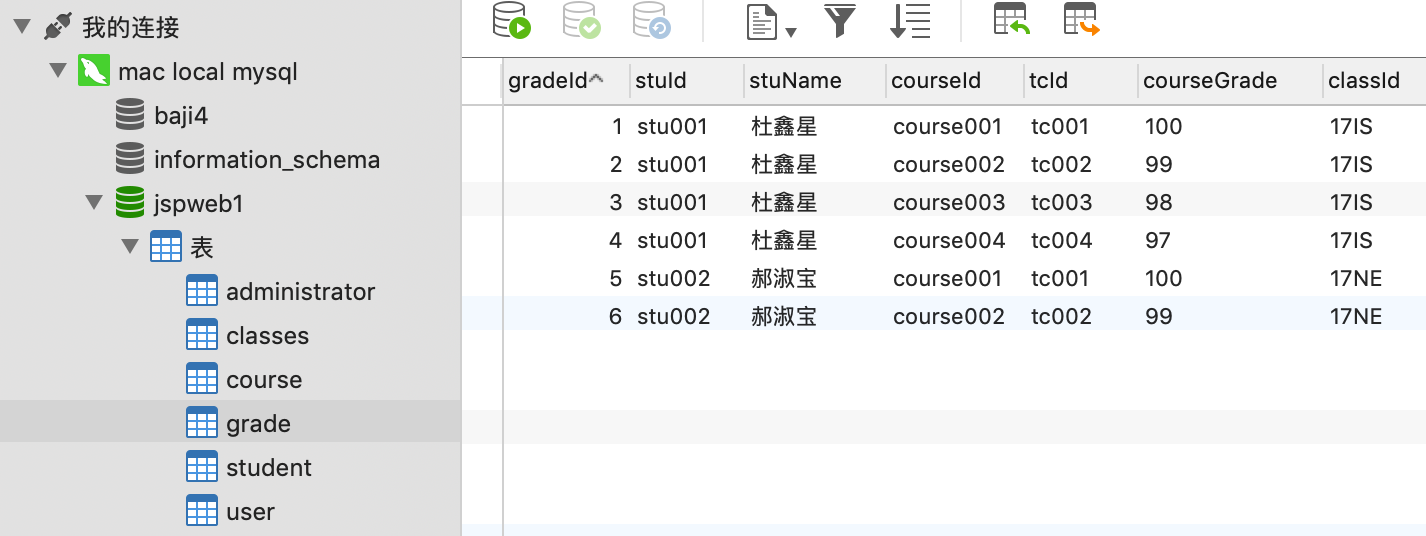
删除成绩前grade数据表信息：



删除成绩页面：



删除成绩后grade数据表信息：



## 4.5 验证码

在登录页面显示验证码：



当验证码输入错误时会提示“验证码输入错误”且刷新验证码：

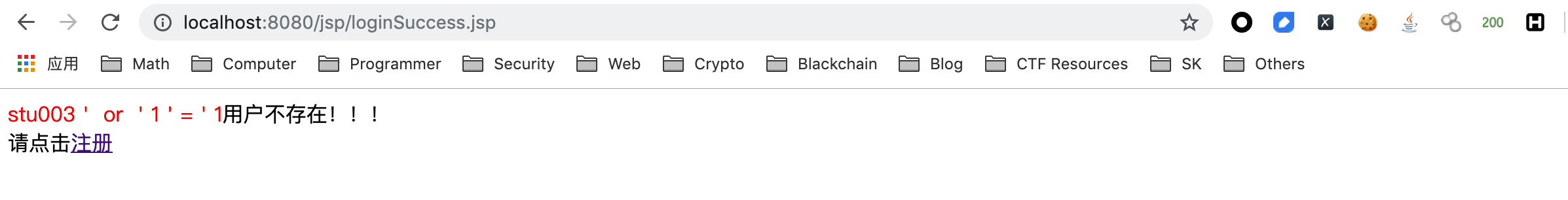


## 4.6 防SQL注入、XSS攻击

所有用户可控的输入都做了过滤，以防报错注入或泄露敏感信息导致猜测实现防御的机制所以不能提示任何过滤或输入错误信息。经过验证是可以过滤SQL注入、XSS等Web非法操作的：







# 第5章 总结

## 5.1 工作完成度

本次JavaWeb安全程序设计实现了权限管理、登录登出、会话管理自动登录、验证码，成绩的增加、删除、修改、查询等功能。基本完成了业务需求

## 5.2 系统缺陷与不足

系统运用的查找算法都是暴力算法，如果面对很多数据的话，系统运行会缓慢，可以运用二分查找算法或者改变存储的数据结构进行优化。

系统编码不够灵活，编码转换方面容易产生乱码。

## 5.3 系统可扩展功能

增加选课功能；

增加多个管理员功能；

增加学生自助修改密码功能；

增加根据成绩进行课程排名；

增加一系列功能做成教务系统。

## 5.4 收获与不足

通过本次JavaWeb安全程序设计，学会了Web开发的MVC+Dao+JDBC的设计模式；较熟练掌握了javaBean编程；会对JSP前端页面进行简单的设计；跟深入理解了数据库的设计与开发；更加熟悉了JavaWeb环境配置与IDE使用。也存在一些不足，比如文档不够全面，无法很好的敏捷开发，以后项目开发应该给予注意改进。