**文本

描述已自动生成**

**Javaee实验报告**

指导教师：　　 　　董朝阳

姓　　名：

学　　号：　　 202145

专　　业：　 计算机科学与技术

班　　级：　　 　　 计科N212

基于SpringBoot实现学生体质管理系统

1. **项目总体说明**

**1.1项目总体说明**

此项目意在帮助学生体育管理员方便管理学生体质情况。主要功能有：

1、登录/注册：均需要学校管理员注册并登陆才能使用系统。

2、成绩查询：管理员可以根据学生学号/姓名查询该学生成绩。

3、成绩录入：管理员可以录入学生成绩，并进行审核和处理。

4、成绩修改：管理员可以查看学生信息、修改学生成绩操作。

5、学生删除：管理员可以删除指定学号、姓名的学生成绩。

6、界面设计：系统界面应该简洁、美观、易于操作，方便管理员使用。

**1.2开发环境的选择、安装与配置**

|  |  |
| --- | --- |
| 开发环境 | 配置 |
| Idea2022 | 创建javaWeb项目时，将前端html页面和后端各模块分开方便编写、维护 |
| Maven3.8.1 | 管理Java项目的构建、依赖、文档和发布。 |
| SpringBoot3.0.6 | 快速构建Web项目 |
| Tomcat9.0.71 | 在Idea中的sever中配置Tomcat服务器 |
| MySql 8.0.32 | 数据库管理 |
| Vue 2.6.10 | 更好地组织与简化前端开发，方便前后端数据交互 |

**1.3需要的jar包**

<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-web</artifactId>

<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-jdbc</artifactId>

<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-test</artifactId>

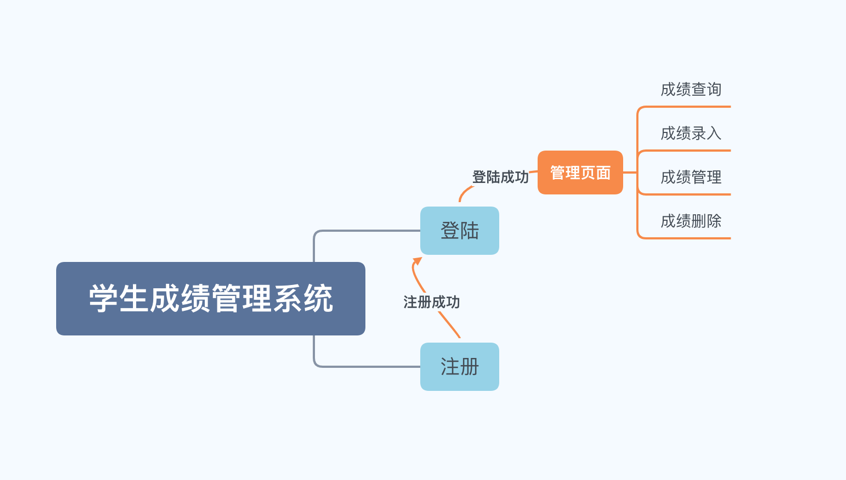
<artifactId>mysql-connector-java</artifactId>  
<version>8.0.32</version>

<artifactId>druid</artifactId>  
<version>1.2.15</version>

1. **项目设计概要**

**2.1项目需求分析**

(1)用例图

****

(2)数据库表关系图

图一.学生体质表 图二.用户登陆表

**图片包含 文本

描述已自动生成图形用户界面

中度可信度描述已自动生成**

**2.2系统总体设计**

本系统针对学生成绩管理系统，学生成绩管理建立了一个表stu:存储学生体质信息，对stu表编写了对应的Java类，执行特定的增加删除查询修改等一些列操作。之后按照业务逻辑编写对应HTML（静态）页面，这些页面通过调用JavaBean实现对数据库操作，从而实现系统功能。本系统编写过程中，对输入信息进行测试，有效避免输入为空和错误信息对系统的破坏，有较好的防止sql注入。

**2.3 数据库设计**

1.user表包含两种属性：username，password用于存储管理员的登录用户名，密码。

2.stu表包含四种属性：id,name,sex,high,weight用于学生成绩的id， 姓名，性别，身高，体重。

3.数据库操作：插入，删除，查询，修改。

**2.4 资源清单和说明**  
 HTML页面3个  
 Bean类3个  
 数据库1个  
 数据库表2个  
 类中实用方法（不包含构造方法，setter，getter方法）10个

**2.5 主要界面设计效果截图和说明**

(1)注册页面

飞机在沙滩上

中度可信度描述已自动生成

(2)登陆页面

图形用户界面, 网站

描述已自动生成

(3)登陆成功，进入主页面

图形用户界面, 应用程序

描述已自动生成

(4)列出全部学生

图形用户界面, 应用程序, Teams

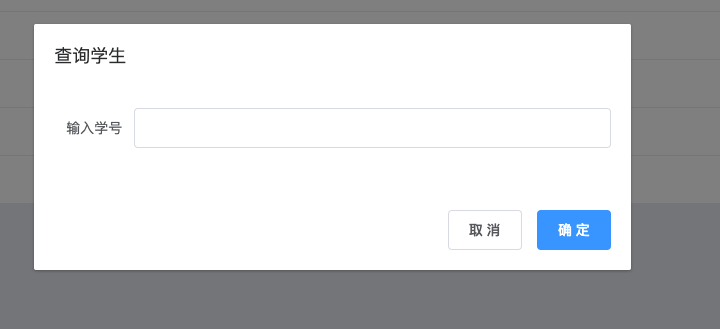
描述已自动生成

(5)添加页面

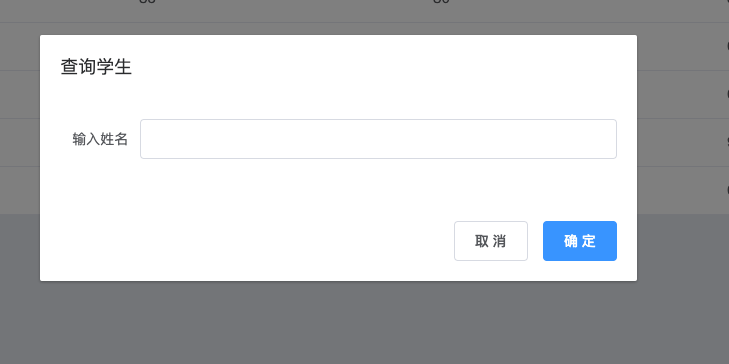
图形用户界面, 应用程序

描述已自动生成

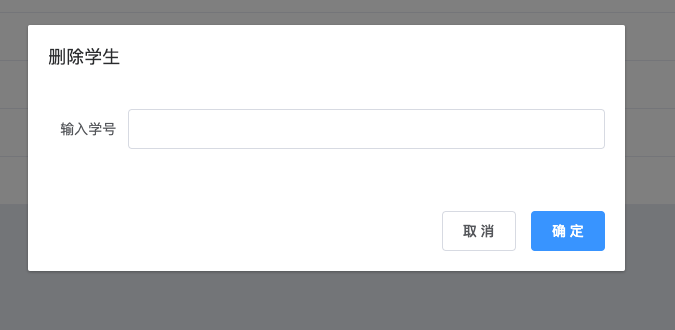
(6)学号查找学生



(7)姓名查找学生



(8)学号删除学生



(9)姓名删除学生



(10)学号修改学生

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

(11)姓名修改成绩

图形用户界面, 文本, 应用程序

描述已自动生成

1. 源代码

**3.1 Druid.priperties文件：**

driverClassName=com.mysql.cj.jdbc.Driver  
url=jdbc:mysql://localhost:3306/students  
username=root  
password=77777777

**3.2 static静态资源(代码略)：**

1. Vue.js
2. Element-ui.js
3. Axios.js
4. Home.html
5. Login.html
6. Regist.html

**3.3 Config**

**3.3.1 JdbcConfig:**

@Configuration  
public class JDBCUtils\_Druid {  
 private static DataSource dataSource;  
 static {  
 Properties pro=new Properties();  
 InputStream resourceAsStream = JdbcUtils.class.getClassLoader().getResourceAsStream("druid.properties");  
 try {  
 pro.load(resourceAsStream);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 try {  
 dataSource=DruidDataSourceFactory.createDataSource(pro);  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 }  
 public static Connection getConnection() throws SQLException {  
 return dataSource.getConnection();  
 }  
 public static DataSource getDataSource(){  
 return dataSource;  
 }  
 public static void close(ResultSet rs, Statement stmt, Connection conn){  
 if(rs!=null){  
 try{  
 rs.close();  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
 }  
 if(stmt!=null){  
 try{  
 stmt.close();  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
 }  
 if(conn!=null){  
 try{  
 conn.close();  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
 }  
 }  
}

**3.3.2 SpringMvcConfig(前端控制器、拦截器)**

@Configuration  
public class SpringMvcConfig implements WebMvcConfigurer {  
 @Autowired  
 private UserLoginInterceptor userLoginInterceptor;  
  
  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 InterceptorRegistration interceptorRegistration = registry.addInterceptor(userLoginInterceptor);  
 interceptorRegistration.addPathPatterns("/\*\*");  
 interceptorRegistration.excludePathPatterns(  
 "/jdbc/login",  
 "/jdbc/regist",  
 "/login.html",  
 "/regist.html",  
 "/js/\*\*",  
 "/imgs/\*\*",  
 "/element-ui/\*\*"  
 );  
 }  
}

**3.4 Entity类**

**3.4.1 Student.class**

@Repository  
public class Student {  
 private Integer id;  
 private String name;  
 private String sex;  
 private Double weight;  
 private Double high;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSex() {  
 return sex;  
 }  
  
 public void setSex(String sex) {  
 this.sex = sex;  
 }  
  
 public Double getWeight() {  
 return weight;  
 }  
  
 public void setWeight(Double weight) {  
 this.weight = weight;  
 }  
  
 public Double getHigh() {  
 return high;  
 }  
  
 public void setHigh(Double high) {  
 this.high = high;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", sex='" + sex + '\'' +  
 ", weight=" + weight +  
 ", high=" + high +  
 '}';  
 }  
}

**3.4.2 User.class**

@Repository  
public class User {  
 private String username;  
 private String password;  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "username='" + username + '\'' +  
 ", password='" + password + '\'' +  
 '}';  
 }  
}

**3.5 Handler**

**3.5.1 UserLoginInterceptor(登陆拦截器)**

@Component  
public class UserLoginInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 HttpSession session = request.getSession();  
 Object user = session.getAttribute("user");  
 if(user!=null){  
 return true;  
 }else {  
 response.sendRedirect(request.getContextPath()+"/login.html");  
 }  
 return false;  
 }

**3.6 Service**

**3.6.1 JdbcService**

添加学生：

public boolean add(Student student){  
  
 try{  
 conn=  
 conn= JDBCUtils\_Druid.getConnection();  
 conn.setAutoCommit(false);//开启事物  
 String sql="insert into stu(id,name,sex,high,weight)" +  
 "values(?,?,?,?,?)";  
 pstmt=conn.prepareStatement(sql);  
 pstmt.setInt(1,student.getId());  
 pstmt.setString(2,student.getName());  
 pstmt.setString(3,student.getSex());  
 pstmt.setDouble(4,student.getHigh());  
 pstmt.setDouble(5,student.getWeight());  
 int i = pstmt.executeUpdate();  
 conn.commit();//提交事物  
 if(i>0)  
 return true;  
 else return false;  
 } catch (SQLException e) {  
 try {  
 conn.rollback();  
 } catch (SQLException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
 JDBCUtils\_Druid.close(null,pstmt,conn);  
 return true;  
}

登陆操作：

public User login(User loginUser){  
 try {  
 String sql="select \* from user where username=? and password=? ";  
 User user = template.queryForObject(sql, new BeanPropertyRowMapper<User>(User.class),  
 loginUser.getUsername(), loginUser.getPassword());  
 return user;  
 }catch (DataAccessException e){  
 e.printStackTrace();  
 return null;  
 }  
}

查询所有学生操作：

public List<Student> selectAll(){  
 List<Student> students = template.query("select \* from stu",new BeanPropertyRowMapper<Student>(Student.class));  
 return students;  
}

根据学号查找学生操作：

public Student selectById(String id){  
 String sql="select \* from stu where id=?";  
 Student student = template.queryForObject(sql, new BeanPropertyRowMapper<Student>(Student.class), id);  
 return student;  
 }

根据姓名查找学生操作：  
public Student selectByName(String name){  
 String sql="select \* from stu where name=?";  
 Student student = template.queryForObject(sql, new BeanPropertyRowMapper<Student>(Student.class),name);  
 return student;  
 }  
根据学号删除学生操作：  
public int deleteById(String id){  
 String sql="delete from stu where id=?";  
 int update = template.update(sql,id);  
 return update;  
 }  
根据姓名删除学生操作：  
 public int deleteByName(String name){  
 String sql="delete from stu where name=?";  
 int update = template.update(sql,name);  
 return update;  
 }  
根据学号更新学生：

public int updateById(Integer id, Map<String,Object> cur) throws SQLException {  
 int update = 0;  
 try {  
 conn = JDBCUtils\_Druid.getConnection();  
 conn.setAutoCommit(false);//开启事物  
 // 假设 conn 是已经创建好的数据库连接对象  
 String sql = "UPDATE stu SET ";  
 List<Object> params = new ArrayList<>();  
 if (cur.containsKey("name") && !cur.get("name").equals("")) {  
 sql += "name = ?, ";  
 params.add(cur.get("name").toString());  
 }  
 if (cur.containsKey("sex") && !cur.get("sex").equals("")) {  
 sql += "sex = ?, ";  
 params.add(cur.get("sex").toString());  
 }  
 if (cur.containsKey("high") && cur.get("high") != null) {  
 sql += "high = ?, ";  
 params.add(cur.get("high").toString());  
 }  
 if (cur.containsKey("weight") && cur.get("weight") != null) {  
 sql += "weight = ?, ";  
 params.add(cur.get("weight").toString());  
 }  
 sql = sql.substring(0, sql.length() - 2);  
 sql += " WHERE id = ?";  
 params.add(id);  
 System.out.println(sql);  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 for (int i = 0; i < params.size(); i++) {  
 stmt.setObject(i + 1, params.get(i));  
 }  
 update= stmt.executeUpdate();  
 conn.commit();  
 if(update>0){  
 return update;  
 }else{  
 return 0;  
 }  
 } catch (SQLException e) {  
 try {  
 conn.rollback();  
 } catch (SQLException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
 JDBCUtils\_Druid.close(null, pstmt, conn);  
 return update;  
}

根据姓名更新学生  
public int updateByName(String name, Map<String,Object> cur) throws SQLException {  
 int update = 0;  
 try {  
 conn = JDBCUtils\_Druid.getConnection();  
 conn.setAutoCommit(false);//开启事物  
 // 假设 conn 是已经创建好的数据库连接对象

String sql = "UPDATE stu SET ";  
 List<Object> params = new ArrayList<>();  
 if (cur.containsKey("sex") && !cur.get("sex").equals("")) {  
 sql += "sex = ?, ";  
 params.add(cur.get("sex").toString());  
 }  
 if (cur.containsKey("high") && cur.get("high") != null) {  
 sql += "high = ?, ";  
 params.add(cur.get("high").toString());  
 }  
 if (cur.containsKey("weight") && cur.get("weight") != null) {  
 sql += "weight = ?, ";  
 params.add(cur.get("weight").toString());  
 }

sql = sql.substring(0, sql.length() - 2);  
 sql += " WHERE name = ?";  
 params.add(name);  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 for (int i = 0; i < params.size(); i++) {  
 stmt.setObject(i + 1, params.get(i));  
 }  
 update= stmt.executeUpdate();  
 conn.commit();  
 if(update>0){  
 return update;  
 }else{  
 return 0;  
 }  
 } catch (SQLException e) {  
 try {  
 conn.rollback();  
 } catch (SQLException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
 JDBCUtils\_Druid.close(null, pstmt, conn);  
 return update;  
 }

添加登陆用户操作：  
public boolean addUser(User user){  
 try{  
 conn= JDBCUtils\_Druid.getConnection();  
 conn.setAutoCommit(false);//开启事物  
 String sql="insert into user(username,password)" +  
 "values(?,?)";  
 pstmt=conn.prepareStatement(sql);  
 pstmt.setString(1,user.getUsername());  
 pstmt.setString(2,user.getPassword());  
 int i = pstmt.executeUpdate();  
 conn.commit();//提交事物  
 String sql2="select \* from user where username=?";  
 User usr2= template.queryForObject(sql2, new BeanPropertyRowMapper<User>(User.class),user.getUsername());  
 if(i>0&&usr2!=null)  
 return true;  
 else return false;  
 } catch (SQLException e) {  
 try {  
 conn.rollback();  
 } catch (SQLException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
 JDBCUtils\_Druid.close(null,pstmt,conn);  
 return true;  
 }

**3.6 Controller**

**3.6.1 JdbcController.java**

@Autowired  
private HttpServletRequest request;  
DruidService druidService=new DruidService();

**登陆请求操作**：  
@PostMapping("/login")  
public String login(@RequestBody User user) {  
 User loginUser = druidService.login(user);  
 if(loginUser!=null){  
 HttpSession session = request.getSession();  
 session.setAttribute("user",user);  
 return "ok";  
 }else {  
 return "err";  
 }  
}

**注册请求操作**：  
@PostMapping("/regist")  
public String regist(@RequestBody User registUser) {  
 boolean flag=druidService.addUser(registUser);  
 if (flag) {  
 return "ok";  
 }  
 else {  
 return "err";  
 }  
}

**查询所有请求操作**：  
@PostMapping("/selectAllServlet")  
public List<Student> selectAll(){  
 List<Student> students = this.druidService.selectAll();  
 return students;  
}

**学号查询请求操作：**  
@GetMapping ("/selectByIdServlet")  
public Student selectByIdServlet(@RequestParam String id){  
 System.out.println(id);  
 Student student=null;  
 if(id!=null) {  
 student = druidService.selectById(id);  
 }  
 return student;  
}

**姓名查询请求操作：**  
@RequestMapping("/selectByNameServlet")  
public Student selectByNameServlet(@RequestParam(value = "name") String name){  
 Student student=new Student();  
 if(name!=null) {  
 student = druidService.selectByName(name);  
 }  
 return student;  
}

**学号删除请求操作：**  
@RequestMapping("/deleteByIdServlet")  
public String deleteByIdServlet(@RequestParam String id){  
 int result=0;//判断是否删除成功  
 if(id!=null){  
 result = druidService.deleteById(id);  
 }  
 if (result==0){  
 return "error";  
 }else {  
 return "success";  
 }  
}

**姓名删除请求操作：**  
@RequestMapping("/deleteByNameServlet")  
public String deleteByNameServlet(@RequestParam(value = "name") String name){  
 int result=0;//判断是否删除成功  
 if(name!=null){  
 result = druidService.deleteByName(name);  
 }  
 if (result==0){  
 return "error";  
 }else {  
 return "success";  
 }  
}

**添加请求操作：**  
@RequestMapping("/addServlet")  
public String addServlet(@RequestBody Student student){  
 boolean flag = druidService.add(student);  
 if (flag){  
 return "success";  
 }else {  
 return "error";  
 }  
}

**学号更新请求操作：**  
@RequestMapping("/updateByIdServlet")  
public String updateByIdServlet(@RequestBody Student student){  
 Map<String,Object> cur=new HashMap<>();  
 int update=0;  
 cur.put("name",student.getName());  
 cur.put("sex",student.getSex());  
 cur.put("high",student.getHigh());  
 cur.put("weight",student.getWeight());  
 try {  
 update = druidService.updateById(student.getId(),cur);  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 if(update>0){  
 return "success";  
 }else {  
 return "error";  
 }  
}

**姓名更新请求操作：**  
@RequestMapping("/updateByNameServlet")  
public String updateByNameServlet(@RequestBody Student student) {  
 Map<String,Object> cur=new HashMap<>();  
 int update=0;  
 cur.put("sex",student.getSex());  
 cur.put("high",student.getHigh());  
 cur.put("weight",student.getWeight());  
 try {  
 update = druidService.updateByName(student.getName(), cur);  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
 if(update>0){  
 return "success";  
 }else {  
 return "error";  
 }  
}

1. **项目关键性问题分析及技术解决方案**

**关键性问题:**

1、在使用SpringBoot时，无法获取到JdbcTempla对象进行动态sql操作。

2、没有使用mybatis,只使用了jdbc的基本操作，在进行学生消息更新时，无法实行动态sql语句

**解决方案：**

1. 由于需要用DataSource对象进行动态sql操作，但SpringBoot自动装配DataSource对象，因此需要排除自动装配，实现手动进行datasource连接。

**以下是具体实现手动装配DataSource**

2、运用Map集合，实行key对应属性名，value对应参数，没有则为null,再根据Map集合遍历，实行sql拼接。

**以下是具体实现动态sql代码：**

if (cur.containsKey("name") && !cur.get("name").equals("")) {  
 sql += "name = ?, ";  
 params.add(cur.get("name").toString());  
 }  
 if (cur.containsKey("sex") && !cur.get("sex").equals("")) {  
 sql += "sex = ?, ";  
 params.add(cur.get("sex").toString());  
 }  
 if (cur.containsKey("high") && cur.get("high") != null) {  
 sql += "high = ?, ";  
 params.add(cur.get("high").toString());  
 }  
 if (cur.containsKey("weight") && cur.get("weight") != null) {  
 sql += "weight = ?, ";  
 params.add(cur.get("weight").toString());  
 }