



**软件工程课程设计**

**题目：学生成绩信息管理系统**

**专业：计算机科学与技术**

**班级：14计本一班**

**组员：孔旻**

目录

[(一)、登录 3](#_Toc470106636)

[1.1登陆页面 3](#_Toc470106637)

[1.2 登录后台处理 4](#_Toc470106638)

[(二)、主页面 7](#_Toc470106639)

[(三)、编辑功能 11](#_Toc470106640)

[3.1 编辑页面 11](#_Toc470106641)

[3.1 编辑的servlet处理过程 13](#_Toc470106642)

[3.3 编辑模式的数据库操作代码（dao部分） 14](#_Toc470106643)

[（四）,查询模块 15](#_Toc470106644)

[4.1查询的英语成绩的servlet处理代码 15](#_Toc470106645)

[4.2 查询的数据库操作代码(dao) 17](#_Toc470106646)

[(五)、排序模块 17](#_Toc470106647)

[5.1排序班级的各科优秀率和不及格率servlet处理代码 17](#_Toc470106648)

[（六）、删除模块 27](#_Toc470106649)

[6.1删除模块的javascript代码，实现了复选的删除学生信息的代码 27](#_Toc470106650)

[6.2删除功能的servlet操作 28](#_Toc470106651)

[6.3 删除操作的数据库操作sql代码 29](#_Toc470106652)

**三详细设计**

身份验证是否成功

输入用户名和密码

退出

操作结果

学生信息管理

添加信息

按条件排序

按条件查询

登录

否

是

# (一)、登录

## 1.1登陆页面

Login.jsp

<div class="title"><h1>学生信息管理</h1></div>

<form action="${pageContext.request.contextPath}/LoginServlet?method=Login" method="post">

<input type="text" name="rootName" placeholder="请输入管理员账号"/><br><br>

<input type="password" name="rootPassword" placeholder="请输入管理员密码"/><br><br>

<input id="sbutton" type="submit" value="提交"/>

</form>

## 1.2 登录后台处理

LoginServlet.java

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

//访问该servlet直接转发到登录页面

request.getRequestDispatcher("/WEB-INF/jsp/login.jsp").forward(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

//将登录提交的页面分发到login方法中

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

if("Login".equals(method)){

this.Login(request, response);

}

}

protected void Login(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException{

//取得输入的用户名和密码

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

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

loginService loginService = new loginService();

try {

boolean flag = loginService.checkRoot(rootName, rootPassword);

//如果存在该用户，转发到学生信息管理的界面

if(flag){

request.getRequestDispatcher("/WEB-INF/jsp/XSGL.jsp").forward(request, response);

}

else{

//如果不存在该管理员用户，则转发到错误处理页面

request.setAttribute("message","你输入的有户名和密码不存在");

request.getRequestDispatcher("/WEB-INF/jsp/message.jsp").forward(request, response);

}

}catch(Exception e){

e.printStackTrace();

//如果不存在该管理员用户，则转发到错误处理页面

request.setAttribute("message","你输入的有户名和密码不存在");

request.getRequestDispatcher("/WEB-INF/jsp/message.jsp").forward(request, response);

}

}

# (二)、主页面

2.1 index.jsp

<div class="title">学生信息管理页面</div>

<div class="outer">

<div class="navigation">

<div id="cx"><h4>按条件查询</h4></div>

<div id="cxop" class="aaa">

<form action="${pageContext.request.contextPath}/selectServlet" method="post">

<select name="select" onchange="addSelect(this)">

<option ${select== 'id'? 'selected':''} value="id">学号</option>

<option ${select== 'name'? 'selected':''} value="name">姓名</option>

<option ${select== 'myClass'? 'selected':''} value="myClass">班级</option>

<option>课程</option>

</select>

<div id="div"></div>

<br><input type="text" name="selectname" value="${selectname}" /><input type="submit" value="查询" /><br><font

color="red" size="4"><span id="span"></span>

</font>

</form>

<p><jsp:include page="back.jsp" />

&nbsp;&nbsp;&nbsp;&nbsp;

</div>

<div id="sort"><h4>按条件排序</h4></div>

<div id="sortop">

<form action="${pageContext.request.contextPath}/OrderServlet" method="post">

<select name="select" onchange="change(this)">

<option value="id">学号</option>

<option>课程</option>

<option value="myclass">班级</option>

</select>

<div id="course"></div>

<select name="order">

<option value="asc">升序</option>

<option value="desc">降序</option>

</select>

<input type="submit" value="排序"/>

</form>

</div>

<div><h4>添加学生信息<a href="${pageContext.request.contextPath}/stuServlet?method=toAddJsp">

&nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</a></h4></div>

</div>

<div class="stu\_table">

<table id="tableId" border="0" align="center" width="850px" height="500px">

<tr>

<th>学号</th>

<th>姓名</th>

<th>性别</th>

<th>班级</th>

<th>英语</th>

<th>数学</th>

<th>语文</th>

</tr>

<c:forEach var="student" items="${Page.listStudent}">

<tr id="TrRed">

<td><input type="checkbox" value="${student.id}" />${student.id}</td>

<td>${student.name}</td>

<td>${student.gender}</td>

<td>${student.myClass}</td>

<td class="pg">${student.english}</td>

<td class="pg">${student.math}</td>

<td class="pg">${student.chinese}</td>

<td><a

href="${pageContext.request.contextPath}/stuServlet?method=findStudentById&id=${student.id}"

style="text-decoration:none">[编辑]</a></td>

</tr>

</c:forEach>

<tr>

# (三)、编辑功能

## 3.1 编辑页面

Update.jsp

<form action="${pageContext.request.contextPath}/stuServlet?method=update&id=${Stu.id}"

method="post">

<table border="0" align="center" width="300px" height="450px">

<tr>

<th>学号</th>

<td><input type="text" name="id" value="${Stu.id}" readonly="readonly"/></td>

</tr>

<tr>

<th>姓名</th>

<td><input type="text" name="name" value="${Stu.name}"/></td>

</tr>

<tr>

<th>性别</th>

<td>

<input ${Stu.gender=='男'?'checked':''} type="radio" name="gender" value="男"/>男

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input ${Stu.gender=='女'? 'checked':''} type="radio" name="gender" value="女"/>女

</td>

</tr>

<tr>

<th>班级</th>

<td>

<input type = "text" name="myClass" value="${Stu.myClass}"/>

</td>

</tr>

<tr>

<th>英语</th>

<td><input type="text" name="english" value="${Stu.english}"/></td>

</tr>

<tr>

<th>数学</th>

<td><input type="text" name="math" value="${Stu.math}"/></td>

</tr>

<tr>

<th>语文</th>

<td><input type="text" name="chinese" value="${Stu.chinese}"/></td>

</tr>

</table>

<div class="back">

<input type="submit" value="更新"/>

&nbsp;&nbsp;&nbsp;&nbsp; <jsp:include page="back.jsp"/>

</div>

</form>

* 1. 编辑的servlet处理过程

protected void update(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException{ int id = Integer.parseInt(request.getParameter("id"));

Student student = new Student();

Enumeration<String> enumers = request.getParameterNames();

while(enumers.hasMoreElements()){

String key = enumers.nextElement();

String[] values = request.getParameterValues(key);

if(!key.equals("method")){

try {

BeanUtils.setProperty(student, key, values);

} catch (Exception e) {

e.printStackTrace();

request.setAttribute("message","更新学生信息失败");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

}

}

## 3.3 编辑模式的数据库操作代码（dao部分）

public void UpdateStudentById(Student stu) throws SQLException{

QueryRunner qr = new QueryRunner(jdbcUtils.getCds());

String sql = "update student set Name=?,Gender=?,myClass=?,English=?,Math=?,Chinese=? where Id=?";

qr.update(sql, new Object[]{stu.getName(),stu.getGender(),stu.getMyClass(),stu.getEnglish(),stu.getMath(),stu.getChinese(),stu.getId()});

}

# （四）,查询模块

## 4.1查询的英语成绩的servlet处理代码

protected void findStudentByEnglish(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

try {

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

request.setAttribute("selectname", selectname);

List<Student> liststu = stuservice.findStudentByEnglish(selectname);

Page page = new Page();

page.setListStudent(liststu);

request.setAttribute("Page", page);

request.getRequestDispatcher("/WEB-INF/jsp/index.jsp").forward(request, response);

} catch (gradesException e) {

e.printStackTrace();

request.setAttribute("message", "请按照格式填写");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

request.setAttribute("message", "查询");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

}

}

## 4.2 查询的数据库操作代码(dao)

public List<Student> selectStudentByEnglish(int low\_grades,int high\_grades) throws SQLException{

QueryRunner runner = new QueryRunner(jdbcUtils.getCds());

String sql = "select \* from student where English >=? and English<=?";

List<Student> liststu = runner.query(sql, new Object[]{low\_grades,high\_grades},new BeanListHandler<Student>(Student.class));

return liststu;

}

# (五)、排序模块

## 5.1排序班级的各科优秀率和不及格率servlet处理代码

protected void OrderAllClass(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

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

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

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

request.setAttribute("SELECT",selectClass);

if(course.equals("english")){

request.setAttribute("COURSE","英语");

}else if(course.equals("math")){

request.setAttribute("COURSE","数学");

}

else if(course.equals("chinese")){

request.setAttribute("COURSE","语文");

}

Map<String,List<MyClass>> map=null;

try {

if("great".equals(selectClass)&&"asc".equals(order)){

map = stuservice.QueryAllClassByGreateInAsc(course);

List<MyClass> listClassByAsc = map.get(course);

request.setAttribute("LISTCLASS",listClassByAsc);

request.getRequestDispatcher("/WEB-INF/jsp/listAllClass.jsp").forward(request, response);

}

else if("great".equals(selectClass)&&"desc".equals(order)){

map = stuservice.QueryAllClassByGreateInDesc(course);

List<MyClass> listClassByDesc = map.get(course);

request.setAttribute("LISTCLASS",listClassByDesc);

request.getRequestDispatcher("/WEB-INF/jsp/listAllClass.jsp").forward(request, response);

}else if("fail".equals(selectClass)&&"asc".equals(order)){

map = stuservice.QueryAllClassByFailInAsc(course);

List<MyClass> listClassByAsc = map.get(course);

request.setAttribute("LISTCLASS",listClassByAsc);

request.getRequestDispatcher("/WEB-INF/jsp/listAllClass.jsp").forward(request, response);

}else if("fail".equals(selectClass)&&"Desc".equals(order)){

map = stuservice.QueryAllClassByFailInDesc(course);

List<MyClass> listClassByDesc = map.get(course);

request.setAttribute("LISTCLASS",listClassByDesc);

request.getRequestDispatcher("/WEB-INF/jsp/listAllClass.jsp").forward(request, response);

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

request.setAttribute("message","排序每个班级的失败");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

}

}

5.2、排序各班级各科的逻辑处理代码（service）

//根据课程的不及格率来降序的显示每个班级的情况

public Map<String,List<MyClass>> QueryAllClassByFailInDesc(String course) throws Exception{

Map<String,List<MyClass>> map = null;

try {

map = this.QueryAllClassByCourse(course);

List<MyClass> listClassDesc = map.get(course);

Collections.sort(listClassDesc, new Comparator<MyClass>() {

public int compare(MyClass o1, MyClass o2) {

if(o1.getUnfail()>o2.getUnfail()){

return -1;

}

return 1;

}

});

map.put(course, listClassDesc);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

throw e;

}

return map;

}

//根据课程的不及格率来升序的显示每个班级的情况

public Map<String,List<MyClass>> QueryAllClassByFailInAsc(String course) throws Exception{

Map<String,List<MyClass>> map = null;

try {

map = this.QueryAllClassByCourse(course);

List<MyClass> listClassAsc = map.get(course);

Collections.sort(listClassAsc,new Comparator<MyClass>() {

public int compare(MyClass o1, MyClass o2) {

if(o1.getUnfail()>=o2.getUnfail()){

/\*System.out.println(o1.getUnfail());

System.out.println(o2.getUnfail());

System.out.println("1");\*/

return 1;

}

/\*System.out.println(o1.getUnfail());

System.out.println(o2.getUnfail());

System.out.println("-1");\*/

return -1;

}

});

map.put(course, listClassAsc);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

throw e;

}

return map;

}

//根据课程优秀率来降序的显示每个班级的情况

public Map<String,List<MyClass>> QueryAllClassByGreateInDesc(String course) throws Exception{

Map<String,List<MyClass>> map = null;

try {

//取得存有各班级某门课程情况的map对象

map = this.QueryAllClassByCourse(course);

//取得存有各班级的某门课程的情况的list集合

List<MyClass> listClassDesc = map.get(course);

//对集合中的所有数据进行重新排序

Collections.sort(listClassDesc,new Comparator<MyClass>(){

//匿名内部类

public int compare(MyClass o1, MyClass o2) {

if(o1.getOutstanding()>o2.getOutstanding()){

return -1;

}

return 1;

}

});

map.put(course, listClassDesc);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

throw e;

}

return map;

}

//根据课程优秀率来升序的显示每个班级的情况

public Map<String,List<MyClass>> QueryAllClassByGreateInAsc(String course) throws Exception{

Map<String,List<MyClass>> map = null;

try {

map = this.QueryAllClassByCourse(course);

List<MyClass> listClassAsc = map.get(course);

Collections.sort(listClassAsc, new Comparator<MyClass>(){

public int compare(MyClass o1, MyClass o2) {

if(o1.getOutstanding()>=o2.getOutstanding()){

return 1;

}

return -1;

}

});

map.put(course, listClassAsc);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

throw e;

}

return map;

}

//根据课程查询所有班级情况

public Map<String,List<MyClass>> QueryAllClassByCourse(String course) throws Exception{

Map<String, List<MyClass>> map = new HashMap<String, List<MyClass>>();

List<MyClass> listMyclass = new ArrayList<MyClass>();

try {

DecimalFormat df = new DecimalFormat("##.00");

//循环每个班级

for(int i=1;i<=studao.QueryClassNumber();i++){

//为每个班级建立个对象

MyClass myclass = new MyClass();

//取得每个班级所有的学生人数

double allstuOfClass = studao.QueryStudentCountByClass("0"+i+"班");

//System.out.println(allstuOfClass);

//取得每个班级该课程及格的人数

double allUnFailstuOfCourse = studao.QueryStudentCountByFail(course, "0"+i+"班");

// System.out.println(allUnFailstuOfCourse);

//计算每个班级该课程的及格率

double unFailRate =Double.parseDouble(df.format(allUnFailstuOfCourse/allstuOfClass));

// System.out.println(unFailRate);

//取得每个班级该课程优秀的人数

double allGreatestuOfCourse = studao.QueryStudentCountByGreat(course, "0"+i+"班");

//System.out.println(allGreatestuOfCourse );

//计算每个班级该课程的优秀率

double GreateRate = Double.parseDouble(df.format(allGreatestuOfCourse/allstuOfClass));

//System.out.println(GreateRate);

//放入到MyClass对象中

myclass.setMyclass("0"+i+"班");

myclass.setUnfail(unFailRate);

myclass.setOutstanding(GreateRate);

listMyclass.add(myclass);

}

} catch (SQLException e) {

e.printStackTrace();

throw new Exception();

}

map.put(course, listMyclass);

return map;

}

# （六）、删除模块

## 6.1删除模块的javascript代码，实现了复选的删除学生信息的代码

function deleteAll(){

var strId="";

var tableElement = document.getElementById("tableId");

var size = tableElement.rows.length;

for(var i=1;i<size-2;i++){

var checkboxElement = tableElement.rows[i].cells[0].firstChild;

if(checkboxElement.checked){

strId += checkboxElement.value+"\_";

}

}

if(strId.length>0){

var url = "/XSGL/stuServlet?method=deleteStudent&strId="+strId;

window.location.href = url;

}else{

window.alert("请选中要删除的用户选项");

}

}

## 6.2删除功能的servlet操作

protected void deleteStudent(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

stuService stuservice = new stuService();

int id=0;

try {

String[] ids = strId.split("\_");

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

id = Integer.parseInt(ids[i]);

stuservice.deleteStudentById(id);

request.setAttribute("message","删除学生信息成功");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

}

} catch (SQLException e) {

e.printStackTrace();

request.setAttribute("message","删除学生信息失败");

request.getRequestDispatcher("/WEB-INF/jsp/message2.jsp").forward(request, response);

## 6.3 删除操作的数据库操作sql代码

//根据指定的id删除一条学生信息

public void DeleteStudentById(int id) throws SQLException{

QueryRunner qr = new QueryRunner(jdbcUtils.getCds());

String sql = "delete from student where id = ?";

qr.update(sql,id);