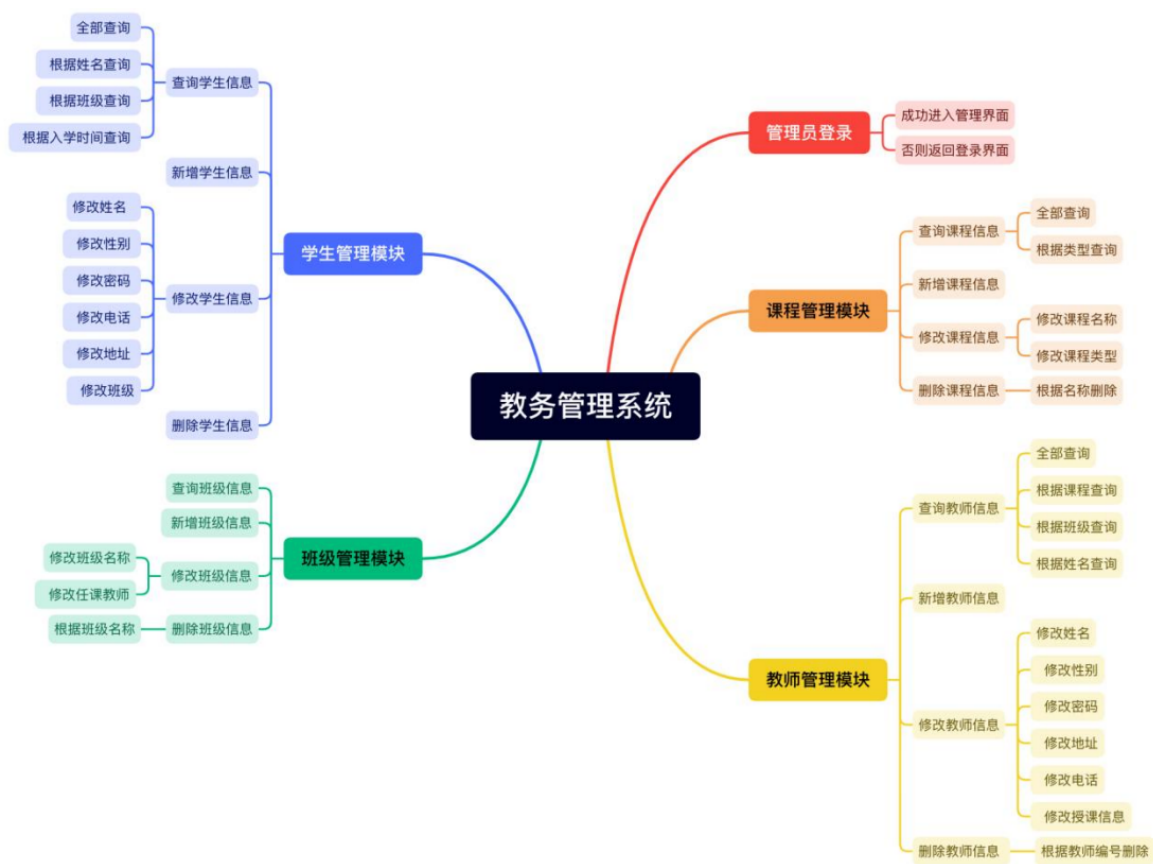


教务管理系统(下)

一、课程介绍




二、学生管理模块

2.1、创建实体类

```
1  /**
2   * 学生表实体类
3   */
4  @Getter
5  @Setter
6  @NoArgsConstructor
7  @AllArgsConstructor
8  public class Student {
9      private String id;
10     private String name;
11     private String sex;
12     private String phone;
13     private String address;
14     private String password;
15     private Date enrollmentTime;
16     private int classId;
17     private String photo;
```

```
18
19     public Student(String name, String sex, String phone, String address, Date
20     enrollmentTime, int classId,String photo) {
21         this.name = name;
22         this.sex = sex;
23         this.phone = phone;
24         this.address = address;
25         this.enrollmentTime = enrollmentTime;
26         this.classId = classId;
27         this.photo = photo;
28     }
}
```

2.2、分页查询学生信息


admin

学生管理

班级管理

教师管理

课程管理

2024年08月30日 15:44:12 星期五

主题

学生管理




关闭操作

退出

学生姓名: 学生姓名 班级: 请选择班级 入学时间: 入学时间 查询

+ 添加

批量删除

<input type="checkbox"/>	序号	学生姓名	性别	学生手机号	生源地	入学时间	所在班级	学生照片	操作
<input type="checkbox"/>	1	张三	男	18700000000	河南	2024-08-18	Java3班		<div>编辑 删除</div>
<input type="checkbox"/>	2	李四	女	18712345678	河南	2024-08-25	Java2班		<div>编辑 删除</div>
<input type="checkbox"/>	3	赵六	男	18798765432	河南	2024-08-27	Java1班		<div>编辑 删除</div>

首页

上页

1

下页

尾页

- 创建用于封装分页查询参数的类 PageQO

```
1  /**
2   * 封装分页查询数据
3   */
4  @Getter
5  @Setter
6  public class PageQO {
7      // 分页数据
8      private int currentPage = 1; // 当前页码,默认显示第一页
9      private int pageSize = 5; // 每页数据量,默认每页显示5条数据
10 }
```

- 引入用于封装分页结果的类 `PageResult`

```
1  /**
2   * 分页数据:
3   * 用户传入:
4   *     当前页          currentPage
5   *     每页数据量      pageSize
6   * 查询数据库:
7   *     总数据量        totalCount
8   *                      select count(*) from 表名
9   *     当前页数据      List<Object> data
10  *                      select * from 表名 limit 偏移量,每页数据量
11  * 程序计算:
12  *     上一页          prevPage
13  *                      prevPage = currentPage - 1 > 0 ? currentPage - 1 : 1
14  *     下一页          nextPage
15  *                      nextPage = currentPage + 1 > totalPage ? totalPage :
currentPage + 1
16  *     尾页(总页数)    totalPage
17  *                      totalPage = totalCount % pageSize == 0 ? totalCount /
pageSize : totalCount / pageSize + 1
18  */
19  @Getter
20  @ToString
21  public class PageResult{
22      private int currentPage;    // 当前页码
23      private int pageSize;      // 每页数据量 5
24      private int totalCount;    // 总数据量 15 3 15%5 = 0 16/5 = 3 余数1
25      private List<?> data;      // 当前页数据
26      private int prevPage;      // 上一页
27      private int nextPage;      // 下一页
28      private int totalPage;     // 总页数
29
30      public PageResult(int currentPage, int pageSize, int totalCount, List<?> data)
31      {
32          this.currentPage = currentPage;
33          this.pageSize = pageSize;
34          this.totalCount = totalCount;
35          this.data = data;
36
37          this.totalPage = totalCount % pageSize == 0 ? totalCount / pageSize :
totalCount / pageSize + 1;
38          this.prevPage = currentPage - 1 > 0 ? currentPage - 1 : 1;
39          this.nextPage = currentPage + 1 > totalPage ? totalPage : currentPage + 1;
40      }
41  }
```

- 定义Servlet, 获取页面传递过来的分页参数并分页查询学生信息

```

1  @WebServlet("/student")
2  @MultipartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      @Override
7      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
8          // 请求分发操作
9          // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
10         String func = req.getParameter("func");
11         switch (func) {
12             case "findAllStudent": findAllStudent(req, resp);break;
13         }
14     }
15
16     // 查询学生信息
17     private void findAllStudent(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
18         // 获取分页数据
19         String currentPage = req.getParameter("currentPage");
20
21         // 将上面的数据封装到StudentQO对象中
22         PageQO pageQO = new PageQO();
23
24         // 获取条件分页结果对象
25         PageResult pageResult = studentService.findAllStudent(studentQO);
26         // 将条件分页结果对象存储到请求域对象
27         req.setAttribute("pageResult",pageResult);
28
29         // 将学生信息显示到列表页面 student_list.jsp
30         req.getRequestDispatcher("student_list.jsp").forward(req,resp);
31     }
32 }

```

- 封装Service层

StudentService 接口

```

1  public interface StudentService {
2      // 返回结果:需要包含 当前页码, 每页数数据量, 总页数, 总数据量, 上一页, 下一页, 当前页数据
3      // 将需要包含的数据封装到对象中, 分页结果对象
4      PageResult findAllStudent(PageQO pageQO);
5  }

```

StudentServiceImpl 类

```

1  public class StudentServiceImpl implements StudentService {
2      private StudentDao studentDao = new StudentDaoImpl();

```

```

3      @Override
4      public PageResult findAllStudent(PageQO pageQO) {
5          // 业务逻辑代码
6          // 查询表中总的数量
7          int totalCount = studentDao.selectStudentCount();
8
9          // 调用 dao 操作数据库
10         List<Student> studentList = studentDao.selectAllStudent(pageQO);
11
12         return new
PageResult(pageQO.getCurrentPage(),pageQO.getPageSize(),totalCount,studentList);
13     }
14 }

```

- 封装DAO层

StudentDao 接口

```

1  /**
2   * 对Student操作的DAO接口，在接口中封装对表要做的增删改查操作
3   */
4  public interface StudentDao {
5      // 查询所有学生信息
6      List<Student> selectAllStudent(PageQO pageQO);
7      // 查询学生总数
8      int selectStudentCount();
9  }

```

StudentDaoImpl 类

```

1  public class StudentDaoImpl implements StudentDao {
2      // 创建dbutils中通过的操作数据的对象,传入连接池对象
3      private QueryRunner queryRunner = new QueryRunner(new ComboPooledDataSource());
4      @Override
5      public List<Student> selectAllStudent(PageQO pageQO) {
6          // 执行查询学生表信息操作
7          List<Student> studentList = null;
8          try {
9              StringBuilder sb = new StringBuilder("select * from student ");
10             // 拼接分页数据
11             // limit ?,?
12             // 第一个?: 从哪一行数据开始获取    = (当前页码 - 1) * 每页数据量
13             // 第二个?: 取多少条数据
14             sb.append(" limit ").append((pageQO.getCurrentPage() - 1) *
pageQO.getPageSize()).append(",").append(pageQO.getPageSize());
15
16             System.out.println(sb);
17             studentList = queryRunner.query(sb.toString(), new
BeanListHandler<Student>(Student.class));

```

```
18         } catch (SQLException e) {
19             throw new RuntimeException(e);
20         }
21
22         return studentList;
23     }
24
25     @Override
26     public int selectStudentCount() {
27         StringBuilder sb = new StringBuilder("select count(*) from student ");
28
29         int totalCount = 0;
30         try {
31             long count = (Long)queryRunner.query(sb.toString(),new
ScalarHandler());
32             totalCount = (int)count;
33         } catch (SQLException e) {
34             throw new RuntimeException(e);
35         }
36         return totalCount;
37     }
38 }
```

2.3、完善条件查询中的班级信息

admin

学生管理

班级管理

教师管理

课程管理

2024年08月30日 15:44:12 星期五

主题

学生管理

查询班级表中所有班级信息，显示在该下拉框中

学生姓名: 学生姓名 班级: 请选择班级 入学时间: 入学时间 查询

+ 添加

批量删除

<input type="checkbox"/>	序号	学生姓名	性别	学生手机号	生源地	入学时间	所在班级	学生照片	操作
<input type="checkbox"/>	1	张三	男	18700000000	河南	2024-08-18	Java3班		编辑 删除
<input type="checkbox"/>	2	李四	女	18712345678	河南	2024-08-25	Java2班		编辑 删除
<input type="checkbox"/>	3	赵六	男	18798765432	河南	2024-08-27	Java1班		编辑 删除

首页

上页

1

下页

尾页

- 创建班级实体类

```

1  @Getter
2  @Setter
3  @NoArgsConstructor
4  @AllArgsConstructor
5  public class Classes {
6      private int id;
7      private String name;
8  }

```

- 封装DAO层

ClassesDao 接口

```

1  public interface ClassesDao {
2      // 查询所有班级信息
3      List<Classes> selectAllClasses();
4  }

```

ClassesDaoImpl 类

```

1  public class ClassesDaoImpl implements ClassesDao {
2      private QueryRunner queryRunner = new QueryRunner(new ComboPooledDataSource());
3      @Override
4      public List<Classes> selectAllClasses() {
5          List<Classes> classesList = null;
6          try {
7              classesList = queryRunner.query("select * from classes", new
BeanListHandler<Classes>(Classes.class));
8          } catch (SQLException e) {
9              throw new RuntimeException(e);
10         }
11         return classesList;
12     }
13 }

```

- 封装Service层

ClassesService 接口

```

1  public interface ClassesService {
2      List<Classes> findAllClasses();
3  }

```

ClassesServiceImpl 类

```

1 public class ClassesServiceImpl implements ClassesService {
2     private ClassesDao classesDao = new ClassesDaoImpl();
3     @Override
4     public List<Classes> findAllClasses() {
5         return classesDao.selectAllClasses();
6     }
7 }

```

- 修改StudentServlet中查询学生信息的方法 `findAllStudent()`

```

1 // 查询学生信息
2 private void findAllStudent(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException {
3     // 获取分页数据
4     String currentPage = req.getParameter("currentPage");
5
6     // 将上面的数据封装到StudentQO对象中
7     PageQO pageQO = new PageQO();
8
9     // 获取条件分页结果对象
10    PageResult pageResult = studentService.findAllStudent(studentQO);
11    // 将条件分页结果对象存储到请求域对象
12    req.setAttribute("pageResult", pageResult);
13
14    // 查询所有的班级信息
15    List<Classes> classesList = classesService.findAllClasses();
16    req.setAttribute("classesList", classesList);
17
18    // 将学生信息显示到列表页面 student_list.jsp
19    req.getRequestDispatcher("student_list.jsp").forward(req, resp);
20 }

```

2.4、条件查询学生信息



- 创建用于封装条件查询参数的类 `StudentQO`


```

1  /**
2   * qo:Query Object    封装查询时的条件数据
3   */
4  @Getter
5  @Setter
6  public class StudentQO extends PageQO{
7      // 条件数据
8      private String name;
9      private int classId;
10     private Date enrollmentTime;
11 }

```

- 修改StudentServlet中查询学生信息的方法 `findAllStudent()`

```

1  // 查询学生信息
2  private void findAllStudent(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
3      // 条件查询发送请求可能会带来五个数据 姓名、班级、入学时间 当前页码, 每页数据量(当前功能, 没有)
4      // 获取条件数据
5      String name = req.getParameter("name");
6      String classId = req.getParameter("classId");
7      String enrollmentTime = req.getParameter("enrollmentTime");
8
9      // 获取分页数据
10     String currentPage = req.getParameter("currentPage");
11
12     // 将上面的数据封装到StudentQO对象中
13     StudentQO studentQO = new StudentQO();
14     if(name != null && name.trim().length() > 0) {
15         studentQO.setName(name);
16     }
17     if(classId != null) {
18         studentQO.setClassId(Integer.parseInt(classId));
19     }
20     if(enrollmentTime != null && enrollmentTime.trim().length() > 0){
21         studentQO.setEnrollmentTime(DateTool.stringToDate(enrollmentTime));
22     }
23     if(currentPage != null && currentPage.trim().length() > 0) {
24         studentQO.setCurrentPage(Integer.parseInt(currentPage));
25     }
26
27     // 获取条件分页结果对象
28     PageResult pageResult = studentService.findAllStudent(studentQO);
29     // 将条件分页结果对象存储到请求域对象
30     req.setAttribute("pageResult", pageResult);
31
32     // 查询所有的班级信息

```

```

33     List<Classes> classesList = classesService.findAllClasses();
34     req.setAttribute("classesList",classesList);
35
36     // 将封装条件数据的对象存储到请求域对象中
37     req.setAttribute("qo",studentQO);
38     // 将学生信息显示到列表页面 student_list.jsp
39     req.getRequestDispatcher("student_list.jsp").forward(req,resp);
40 }

```

- 修改Service层代码

修改StudentService 接口中findAllStudent方法

```

1  PageResult findAllStudent(StudentQO studentQO);

```

修改StudentServiceImpl 类中findAllStudent方法

```

1  @Override
2  public PageResult findAllStudent(StudentQO studentQO) {
3      // 业务逻辑代码
4      // 查询表中总的数量
5      int totalCount = studentDao.selectStudentCount(studentQO);
6
7      // 调用 dao 操作数据库
8      List<Student> studentList = studentDao.selectAllStudent(studentQO);
9
10     return new
11     PageResult(studentQO.getCurrentPage(),studentQO.getPageSize(),totalCount,studentList);
12 }

```

- 修改Dao层代码

修改StudentDao 接口中selectAllStudent和selectStudentCount方法

```

1  // 查询所有学生信息
2  List<Student> selectAllStudent(StudentQO studentQO);
3
4  // 查询学生总数
5  int selectStudentCount(StudentQO studentQO);

```

修改StudentDaoImpl 类中selectAllStudent和selectStudentCount方法

```

1  @Override
2  public List<Student> selectAllStudent(StudentQO studentQO) {
3      // 执行查询学生表信息操作
4      // query() 执行查询语句
5      // update() 执行增删改语句

```

```

6      List<Student> studentList = null;
7      try {
8          StringBuilder sb = new StringBuilder("select * from student where 1 = 1 ");
9          // 拼接条件数据
10         if(studentQO.getName() != null) {
11             // select * from student where 1 = 1 and name like '%李%' and
12             sb.append("and name like '%").append(studentQO.getName()).append("%'");
13         }
14         if(studentQO.getClassId() != 0) {
15             // select * from student where 1 = 1 and name like '%李%' and classid =
16             1
17             sb.append(" and classid=").append(studentQO.getClassId());
18         }
19         if(studentQO.getEnrollmentTime() != null) {
20             // select * from student where 1 = 1 and name like '%李%' and classid =
21             1 and enrollmentTime = '1999-9-9'
22             sb.append(" and enrollmentTime =
23             ").append(DateTool.dateToString(studentQO.getEnrollmentTime())).append("'");
24         }
25         // 拼接分页数据
26         // limit ?,?
27         // 第一个?: 从哪一行数据开始获取    = (当前页码 - 1) * 每页数据量
28         // 第二个?: 取多少条数据
29         // select * from student where 1 = 1 and name like '%李%' and classid = 1
30         and enrollmentTime = '1999-9-9' limit 0,1
31         sb.append(" limit ").append((studentQO.getCurrentPage() - 1) *
32         studentQO.getPageSize()).append(",").append(studentQO.getPageSize());
33
34         System.out.println(sb);
35         studentList = queryRunner.query(sb.toString(), new BeanListHandler<Student>
36         (Student.class));
37     } catch (SQLException e) {
38         throw new RuntimeException(e);
39     }
40     return studentList;
41 }
42
43 @Override
44 public int selectStudentCount(StudentQO studentQO) {
45     StringBuilder sb = new StringBuilder("select count(*) from student where 1 = 1
46     ");
47     // 拼接条件数据
48     if(studentQO.getName() != null) {
49         // select * from student where 1 = 1 and name like '%李%' and
50         sb.append("and name like '%").append(studentQO.getName()).append("%'");
51     }
52     if(studentQO.getClassId() != 0) {

```

```

48         // select * from student where 1 = 1 and name like '%李%' and classid = 1
49         sb.append(" and classid=").append(studentQO.getClassId());
50     }
51     if(studentQO.getEnrollmentTime() != null) {
52         // select * from student where 1 = 1 and name like '%李%' and classid = 1
53         and enrollmentTime = '1999-9-9'
54         sb.append(" and enrollmentTime = ").append(studentQO.getEnrollmentTime()).append("");
55     }
56     int totalCount = 0;
57     try {
58         long count = (Long)queryRunner.query(sb.toString(),new ScalarHandler());
59         totalCount = (int)count;
60     } catch (SQLException e) {
61         throw new RuntimeException(e);
62     }
63     return totalCount;
64 }

```

2.5、添加学生信息

学生管理
关闭操作
退出

学生添加

学生姓名:

学生性别:

☒男
☐女

手机号码:

生源地:

入学时间:

所属班级:

Java1班

▼

学生照片:

选择文件

未选择任何文件

提交

- 在StudentServlet 添加用于显示添加页面和添加功能请求处理方法

```

1  @WebServlet("/student")
2  @MultipartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      private ClassesService classesService = new ClassesServiceImpl();
7      @Override
8      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

```

```

9      // 请求分发操作
10     // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
11     String func = req.getParameter("func");
12     switch (func) {
13         case "findAllStudent": findAllStudent(req, resp);break;
14         case "insertStudentPage": insertStudentPage(req, resp);break;
15         case "insertStudent": insertStudent(req, resp);break;
16         case "findStudentById": findStudentById(req, resp);break;
17     }
18 }
19
20 // 学生添加
21 public void insertStudent(HttpServletRequest req, HttpServletResponse resp)
throws IOException, ServletException {
22     // 获取表单数据
23     String name = req.getParameter("name");
24     String sex = req.getParameter("sex");
25     String phone = req.getParameter("phone");
26     String address = req.getParameter("address");
27     String enrollmentTime = req.getParameter("enrollmentTime");
28     String classId = req.getParameter("classId");
29     Part part = req.getPart("photo");
30     try {
31         String photo = UploadUtil.upload(part, req);
32         // 将上面的表单数据封装到学生对象
33         Student student = new Student(name, sex, phone, address,
DateTool.stringToDate(enrollmentTime), Integer.parseInt(classId),photo);
34
35         // 调用service 添加学生信息
36         studentService.insertStudent(student);
37
38         // 添加成功后,跳转到学生列表页,查询学生信息,展示最新数据
39         // 方案1.直接调用findAllStudent方法 但是有问题,页面一刷新,重新添加数据,因为
跳转列表页面用的是转发,转发是地址不会发生变化
40         // 方案2.重定向到findAllStudent请求上
41         resp.sendRedirect("student?func=findAllStudent");
42
43     }catch(Exception e) {
44         String errMsg = e.getMessage();
45         req.setAttribute("errMsg",errMsg);
46         insertStudentPage(req, resp);
47     }
48
49
50 }
51 // 学生添加页面
52 public void insertStudentPage(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
53     // 查询所有的班级信息

```

```

54         List<Classes> classesList = classesService.findAllClasses();
55
56         // 将班级信息存储到请求域对象
57         req.setAttribute("classesList", classesList);
58
59         // 转发到student_add.jsp
60         req.getRequestDispatcher("student_add.jsp").forward(req, resp);
61     }
62 }

```

- 修改Service层代码，加入添加学生方法

StudentService接口

```

1 void insertStudent(Student student);

```

StudentServiceImpl类

```

1 @Override
2 public void insertStudent(Student student) {
3     // 完善学生信息
4     // 生成密码 手机号码后六位 12345678910
5     student.setPassword(student.getPhone().substring(5));
6
7     // 学号 入学年份 + 班级编号 + 数字 yyyy-mm-dd
8     String year = DateTool.dateToString(student.getEnrollmentTime()).substring(0,
9     4);
10
11     // 获取学生中最大的学号
12     int maxId = studentDao.selectMaxId();
13     // 如果maxId == 0,说明该班级执行没有学生
14     if(maxId == 0) { // 班级第一个学生
15         student.setId(year + student.getClassId() + 1);
16     } else {
17         student.setId(maxId + 1 + "");
18     }
19
20     // 调用 dao保存学生信息方法
21     studentDao.insertStudent(student);
22 }

```

- 修改Dao层代码，加入添加学生方法

StudentDao接口

```

1 // 添加学生信息
2 void insertStudent(Student student);
3
4 // 查询学生最大的学号
5 int selectMaxId();

```

StudentDaoImpl类


```

1 @Override
2 public void insertStudent(Student student) {
3     try {
4         queryRunner.update("insert into student values (?,?,?,?,?,?,?,?,?)",
5
6             student.getId(),student.getName(),student.getSex(),student.getPhone(),student.getAddress(),
7
8             student.getPassword(),DateTool.dateToString(student.getEnrollmentTime()),student.getClassId(),
9
10            student.getPhoto()
11        );
12    } catch (SQLException e) {
13        throw new RuntimeException(e);
14    }
15 }
16
17 @Override
18 public int selectMaxId() {
19     int maxId = 0;
20     try {
21         String id = (String)queryRunner.query("select max(id) from student",new
22             ScalarHandler());
23         if(id != null) {
24             maxId = Integer.parseInt(id);
25         }
26     } catch (SQLException e) {
27         throw new RuntimeException(e);
28     }
29     return maxId;
30 }

```

2.6、修改页面学生数据回显

学生修改

学生姓名:	<input type="text" value="赵六"/>
学生密码:	<input type="password" value="....."/>
学生性别:	<input checked="" type="radio"/> 男 <input type="radio"/> 女
手机号码:	<input type="text" value="18798765432"/>
生源地:	<input type="text" value="河南"/>
入学时间:	<input type="text" value="2024-08-27"/>
所属班级:	<input type="text" value="Java1班"/>
学生照片:	<div><input type="button" value="选择文件"/> 未选择任何文件</div> <div></div> <div><input type="button" value="提交"/></div>

- 在StudentServlet 添加用于显示修改页面请求处理方法

```
1  @WebServlet("/student")
2  @MultipartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      private ClassesService classesService = new ClassesServiceImpl();
7      @Override
8      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
9          // 请求分发操作
10         // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
11         String func = req.getParameter("func");
12         switch (func) {
13             case "findAllStudent": findAllStudent(req, resp);break;
14             case "insertStudentPage": insertStudentPage(req, resp);break;
15             case "insertStudent": insertStudent(req, resp);break;
16             case "findStudentById": findStudentById(req, resp);break;
17         }
18     }
19     // 根据学生id查询学生信息到学生修改页面
20     public void findStudentById(HttpServletRequest req,HttpServletResponse resp)
throws ServletException, IOException {
21         // 查询所有班级信息
22         List<Classes> classesList = classesService.findAllClasses();
23
24         // 将班级信息存储到 请求域
```



```

25         req.setAttribute("classesList",classesList);
26
27         // 获取学生id
28         String id = req.getParameter("id");
29         // 根据学生id查询学生信息
30         Student student = studentService.findStudentById(id);
31
32         // 将查询到的学生存储到请求域
33         req.setAttribute("student",student);
34         // 跳转到修改页面, 回显学生    -- 转发
35         req.getRequestDispatcher("student_update.jsp").forward(req,resp);
36     }
37 }

```

- 修改Service层代码, 加入根据id查询学生方法

StudentService接口

```

1 Student findStudentById(String id);

```

StudentServiceImpl类

```

1 @Override
2 public Student findStudentById(String id) {
3     return studentDao.selectStudentById(id);
4 }

```

- 修改Dao层代码, 加入根据id查询学生方法

StudentDao接口

```

1 // 根据id查询学生信息
2 Student selectStudentById(String id);

```

StudentDaoImpl类

```

1 @Override
2 public Student selectStudentById(String id) {
3     Student student = null;
4     try {
5         student = queryRunner.query("select * from student where id = ?", new
        BeanHandler<Student>(Student.class),id);
6     } catch (SQLException e) {
7         throw new RuntimeException(e);
8     }
9     return student;
10 }

```

2.7、修改学生信息

- 在StudentServlet 添加修改学生请求处理方法

```
1  @WebServlet("/student")
2  @MultipartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      private ClassesService classesService = new ClassesServiceImpl();
7      @Override
8      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
9          // 请求分发操作
10         // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
11         String func = req.getParameter("func");
12         switch (func) {
13             case "findAllStudent": findAllStudent(req, resp);break;
14             case "insertStudentPage": insertStudentPage(req, resp);break;
15             case "insertStudent": insertStudent(req, resp);break;
16             case "findStudentById": findStudentById(req, resp);break;
17             case "updateStudent": updateStudent(req, resp);break;
18         }
19     }
20     // 修改学生信息
21     public void updateStudent(HttpServletRequest req,HttpServletResponse resp)
throws ServletException, IOException {
22         // 获取表单数据
23         String id = req.getParameter("id");
24         String password = req.getParameter("password");
25         String name = req.getParameter("name");
26         String sex = req.getParameter("sex");
27         String phone = req.getParameter("phone");
28         String address = req.getParameter("address");
29         String enrollmentTime = req.getParameter("enrollmentTime");
30         String classId = req.getParameter("classId");
31         Part part = req.getPart("photo");
32         String photo = "";
33         if(part.getSize() == 0) {
34             photo = req.getParameter("oldPhoto");
35         }else {
36             photo = UploadUtil.upload(part, req);
37         }
38         // 将上面的表单数据封装到学生对象
39         Student student = new Student(id,name, sex, phone,
address,password,DateTool.stringToDate(enrollmentTime),
Integer.parseInt(classId),photo);
40
41         // 调用service修改学生信息
```

```

42         studentService.updateStudent(student);
43
44         // 学生修改完毕后,跳转到列表页面,展示最新数据
45         resp.sendRedirect("student?func=findAllStudent");
46     }
47 }

```

- 修改Service层代码，加入修改学生方法

StudentService 接口

```

1 void updateStudent(Student student);

```

StudentServiceImpl 类

```

1 @Override
2 public void updateStudent(Student student) {
3     studentDao.updateStudent(student);
4 }

```

- 修改Dao层代码，加入修改学生方法

StudentDao 接口

```

1 void updateStudent(Student student);

```

StudentDaoImpl 类

```

1 @Override
2 public void updateStudent(Student student) {
3     try {
4         queryRunner.update("update student set name = ?,sex = ?,phone = ?,address =
5         ?,password = ?,enrollmentTime = ?,classid = ?,photo = ? where id = ?",
6
7         student.getName(),student.getSex(),student.getPhone(),student.getAddress(),student
8         .getPassword(),student.getEnrollmentTime(),student.getClassId(),student.getPhoto(),
9         student.getId()
10        );
11    } catch (SQLException e) {
12        throw new RuntimeException(e);
13    }
14 }

```

2.8、根据id删除学生信息

- 在StudentServlet 添加根据id删除学生请求处理方法

```
1  @WebServlet("/student")
2  @MultipartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      private ClassesService classesService = new ClassesServiceImpl();
7      @Override
8      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
9          // 请求分发操作
10         // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
11         String func = req.getParameter("func");
12         switch (func) {
13             case "findAllStudent": findAllStudent(req, resp);break;
14             case "insertStudentPage": insertStudentPage(req, resp);break;
15             case "insertStudent": insertStudent(req, resp);break;
16             case "findStudentById": findStudentById(req, resp);break;
17             case "updateStudent": updateStudent(req, resp);break;
18             case "deleteStudent": deleteStudent(req, resp);break;
19         }
20     }
21     // 删除单个学生信息
22     public void deleteStudent(HttpServletRequest req,HttpServletResponse resp)
throws IOException {
23         // 获取要删除学生信息的id
24         String id = req.getParameter("id");
25         // 调用service删除学生信息方法
26         studentService.deleteStudent(id);
27
28         // 学生信息删除完毕后,,跳转到列表页面,展示最新数据
29         resp.sendRedirect("student?func=findAllStudent");
30     }
31 }
```

- 修改Service层代码，加入删除学生方法

StudentService 接口

```
1 void deleteStudent(String ids);
```

StudentServiceImpl 类

```

1  @Override
2  public void deleteStudent(String ids) {
3      studentDao.deleteStudent(ids);
4  }

```

- 修改Dao层代码，加入删除学生方法

StudentDao接口

```

1  // 删除学生信息
2  // delete from student where id = ?
3  // delete from student where id in (?)
4  void deleteStudent(String ids);

```

StudentDaoImpl类

```

1  @Override
2  public void deleteStudent(String ids) {
3      try {
4          queryRunner.update("delete from student where id in (" + ids + ")");
5      } catch (SQLException e) {
6          throw new RuntimeException(e);
7      }
8  }

```

2.9、批量删除学生信息

- 在StudentServlet 添加批量删除学生请求处理方法

```

1  @WebServlet("/student")
2  @MultiPartConfig
3  public class StudentServlet extends HttpServlet {
4      // 创建service对象
5      private StudentService studentService = new StudentServiceImpl();
6      private ClassesService classesService = new ClassesServiceImpl();
7      @Override
8      protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
9          // 请求分发操作
10         // 区分当前请求是什么操作请求,调用下面对应的方法处理操作
11         String func = req.getParameter("func");
12         switch (func) {
13             case "findAllStudent": findAllStudent(req, resp);break;
14             case "insertStudentPage": insertStudentPage(req, resp);break;
15             case "insertStudent": insertStudent(req, resp);break;
16             case "findStudentById": findStudentById(req, resp);break;
17             case "updateStudent": updateStudent(req, resp);break;
18             case "deleteStudent": deleteStudent(req, resp);break;

```

```
19         case "batchDeleteStudent": batchDeleteStudent(req, resp);
20     }
21 }
22 //批量删除学生信息
23 public void batchDeleteStudent(HttpServletRequest req, HttpServletResponse resp)
throws IOException {
24     // 获取要删除的学生id
25     String ids = req.getParameter("ids");
26     // 调用service删除学生信息方法
27     studentService.deleteStudent(ids);
28
29     // 学生信息删除完毕后,,跳转到列表页面,展示最新数据
30     resp.sendRedirect("student?func=findAllStudent");
31 }
32 }
```