东莞理工学院网络空间安全学院

实验报告

课程名称：Java EE程序设计 学期：2024年秋季

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 综合项目实战训练 | | | 实验序号 | 3 |
| 姓 名 |  | 学 号 |  | 班 级 |  |
| 实验地点 |  | 实验日期 |  | 指导老师 |  |
| 教师评语 |  | | | 实验成绩 |  |
| 同组同学 |  | | | | |
| 一、实验内容、要求  项目简介:  综合运用前面章节相关概念与原理，设计和开发一个信息管理系统，通过本实验项目的练习，有助于学生对Java Web技术的了解和认识，提高项目开发实践能力。主要内容为对项目进行需求分析，规划，框架设计，编码，布署，发布等。  系统描述：（填写选作系统的描述）  家校通管理系统（中）   * 系统共有三类用户，分别是管理员、老师、学生家长； * (1).老师使用系统前需要注册，其注册信息管理员审核通过后才能生效； (2).老师在登陆系统后可以创建新的班级并进行班级信息管理(加个删除吧)，新创建的班级信息在管理员审核通过后才能生效，班级信息一旦通过审核则不能再修改； (3).老师也可以加入已生效的班级，需班级创建者通过； (4).老师加入班级后可以发布班级通知，班级通知对该班级的所有家长及老师可见，通知发布后不可修改但可以删除； (5).同一班级的老师和家长之间也可以通过站内信息进行沟通；老师可以通过多条件查询对班级通知和站内信息进行查询； * 家长使用系统前需要进行注册；家长登陆后可以申请加入的班级，申请信息在班级创建者审核通过后生效；家长加入班级后可以查看班级通知；通过站内信息与班级老师沟通；家长可以通过多条件查询对班级通知和站内信息进行查询； * 管理员在登陆系统后可以进行教师信息、班级信息进行审核；管理员可以发布系统通知，系统通知对所有用户可见，通知发布后不可修改但可以删除；   **二、所采用的Java EE技术（如MVC架构，JSP/Servlet，EL，JSTL，JDBC等）**  MVC架构以及JSP/Servlet  **三、数据库设计（ER图）**  **graph**  **四、实验的主要模块功能及核心代码** 1. ****系统概述**** 系统的三类用户：管理员、老师、家长   * **管理员**： 1.可以审核教师信息、班级创建信息， 2.可以发布系统通知（所有人可见），通知发布后只能删除不能修改 * **老师**： 1.老师账号需要注册后，管理员审核通过才能使用 2.老师可以创建新的班级，并管理班级信息，新班级只有通过管理员审核才能生效，审核过后的班级信息不能修改； 3.老师也能加入通过审核的班级，需要班级创建者同意 4.老师进入班级后可以发布班级通知，对班级内的老师及其家长可见，通知只能删除不能修改 5.同一班级的老师和家长之间通过站内信息沟通 6.老师可以通过多条件查询对班级通知和站内信息进行查询 * **家长**： 1.家长使用系统前需要进行注册；   2.家长登陆后可以申请加入的班级，申请信息在班级创建者审核通过后生效； 3.家长加入班级后可以查看班级通知；通过站内信息与班级老师沟通 4.家长可以通过多条件查询对班级通知和站内信息进行查询； ****模块功能(核心代码)**** (1)注册与登录  **注册控制** package controller;  import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import service.RegisterService;  import java.io.IOException;  public class Register extends HttpServlet {  private final RegisterService registerService = new RegisterService();   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取表单参数  String identify = request.getParameter("identify");  String username = request.getParameter("username");  String password = request.getParameter("password");  String confirmPassword = request.getParameter("confirmPassword");   // 检查两次输入的密码是否匹配  if (!password.equals(confirmPassword)) {  request.setAttribute("errorMessage", "两次输入的密码不匹配！");  request.setAttribute("username", username);  request.getRequestDispatcher("register.jsp").forward(request, response);  return;  }   // 检查用户名是否已存在  if (registerService.isUsernameExists(username, identify)) {  request.setAttribute("errorMessage", "用户名已存在！");  request.setAttribute("username", username);  request.getRequestDispatcher("register.jsp").forward(request, response);  return;  }   // 注册用户  try {  registerService.registerUser(username, password, identify);  response.sendRedirect("index.jsp");  } catch (Exception e) {  e.printStackTrace();  request.setAttribute("errorMessage", "注册失败，请稍后再试！");  request.setAttribute("username", username);  request.getRequestDispatcher("register.jsp").forward(request, response);  }  } }  **登录控制**  package controller;  import service.LoginService;  import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.IOException;  public class Login extends HttpServlet {  private final LoginService loginService = new LoginService();   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String identify = request.getParameter("identify");  String username = request.getParameter("username");  String password = request.getParameter("password");   try {  if (loginService.userExists(username, identify)) {  if (loginService.validatePassword(username, password)) {  if (loginService.isApprovedOrParent(username, password)) {  int userId = loginService.getUserId(username, identify); // 从数据库获取用户ID  if (userId == -1) {  handleError(request, response, "无法找到用户ID", username);  return;  }  HttpSession session = request.getSession();  session.setAttribute("username", username);   // 根据角色重定向页面  if ("teacher".equals(identify)) {  session.setAttribute("teacherId", userId);  session.setAttribute("parentId", null);  session.setAttribute("adminId", null);  request.getRequestDispatcher("teacher.jsp").forward(request, response);  } else if ("parent".equals(identify)) {  session.setAttribute("parentId", userId);  session.setAttribute("teacherId", null);  session.setAttribute("adminId", null);  request.getRequestDispatcher("parent.jsp").forward(request, response);  } else if ("admin".equals(identify)) {  session.setAttribute("adminId", userId);  session.setAttribute("teacherId", null);  session.setAttribute("parentId", null);  request.getRequestDispatcher("admin.jsp").forward(request, response);  }  } else {  handleError(request, response, "用户未审核", username);  }  } else {  handleError(request, response, "密码错误", username);  }  } else {  handleError(request, response, "用户名不存在或者身份错误", username);  }  } catch (Exception e) {  e.printStackTrace();  throw new ServletException("登录处理出错", e);  }  }   private void handleError(HttpServletRequest request, HttpServletResponse response, String errorMessage, String username)  throws ServletException, IOException {  request.setAttribute("errorMessage", errorMessage);  request.setAttribute("username", username);  request.getRequestDispatcher("index.jsp").forward(request, response);  }   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.sendRedirect("index.jsp");  } }  解析：通过，首先通过DAO判断，注册的账号是否已存在或者登录的账号是否存在，session来存储着对应登录角色的Id作为身份判别唯一凭证。注册成功后，跳转回登录页面；登录成功后，跳转到对应登录角色页面  **管理员：**   1. **审核教师注册  审核教师控制 教师注册后需要管理员审核才行** package controller;  import model.Teacher; import service.AuditTeacherService;  import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.IOException; import java.sql.SQLException; import java.util.List;  // 审核教师的 Servlet public class AuditTeacherServlet extends HttpServlet {  private final AuditTeacherService auditTeacherService = new AuditTeacherService();   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  try {  // 获取待审核教师列表  List<Teacher> pendingTeachers = auditTeacherService.getPendingTeachers();  request.setAttribute("pendingTeachers", pendingTeachers);   // 转发到 JSP 页面  request.getRequestDispatcher("/admin/AuditTeacherRegistration.jsp").forward(request, response);  } catch (SQLException e) {  throw new ServletException("获取待审核教师列表时出错", e);  }  }   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String action = request.getParameter("action");  int teacherId = Integer.parseInt(request.getParameter("id"));   try {  // 处理审核操作（通过或拒绝）  auditTeacherService.handleTeacherApproval(teacherId, action);   // 重定向回审核页面  response.sendRedirect(request.getContextPath() + "/AuditTeacherServlet");  } catch (SQLException e) {  throw new ServletException("处理审核操作时出错", e);  }  } }   **TeacherDAO中数据库交互** // 获取待审核的教师列表 public List<Teacher> getPendingTeachers() throws SQLException {  List<Teacher> teachers = new ArrayList<>();  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;   try {  conn = DatabaseUtil.getConnection();  String query = "SELECT id, username, created\_at, status FROM users WHERE (role = 'teacher' OR role = 'admin') AND status = 'pending'";  stmt = conn.prepareStatement(query);  rs = stmt.executeQuery();   while (rs.next()) {  Teacher teacher = new Teacher();  teacher.setId(rs.getInt("id"));  teacher.setUsername(rs.getString("username"));  teacher.setCreatedAt(rs.getString("created\_at"));  teacher.setStatus(rs.getString("status"));  teachers.add(teacher);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }   return teachers; }  **封装业务逻辑** package service;  import dao.TeacherDAO; import model.Teacher;  import java.sql.SQLException; import java.util.List;  // 审核教师服务类 public class AuditTeacherService {  private final TeacherDAO teacherDAO;   // 构造方法，初始化 TeacherDAO  public AuditTeacherService() {  this.teacherDAO = new TeacherDAO();  }   // 获取待审核教师列表  public List<Teacher> getPendingTeachers() throws SQLException {  return teacherDAO.getPendingTeachers();  }   // 处理教师审核操作（通过或拒绝）  public void handleTeacherApproval(int id, String action) throws SQLException {  teacherDAO.updateTeacherStatus(id, action);  } }  （2）**审核班级创建  班级审核控制类** package controller;  import jakarta.servlet.\*; import jakarta.servlet.annotation.WebServlet; import jakarta.servlet.http.\*; import java.io.IOException; import java.sql.SQLException; import dao.ClassDAO;  @WebServlet("/AuditClassServlet") public class AuditClassServlet extends HttpServlet {   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String action = request.getParameter("action");  int classId = Integer.parseInt(request.getParameter("classId"));   ClassDAO classDAO = new ClassDAO();  try {  boolean isUpdated = false;  if ("approve".equals(action)) {  isUpdated = classDAO.updateClassStatus(classId, "approved");  } else if ("deny".equals(action)) {  isUpdated = classDAO.updateClassStatus(classId, "rejected");  }   if (isUpdated) {  request.getSession().setAttribute("message", "班级审核成功！");  } else {  request.getSession().setAttribute("error", "班级审核失败，请重试！");  }  } catch (SQLException e) {  e.printStackTrace();  request.getSession().setAttribute("error", "数据库操作错误，请稍后重试！");  }   // 重定向到审核页面   response.sendRedirect("ToShowAdminClass");  }   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  ClassDAO classDAO = new ClassDAO();  // 获取待审核的班级数据  try {  request.setAttribute("classList", classDAO.getPendingClasses());  } catch (SQLException e) {  throw new RuntimeException(e);  }  // 转发到 JSP 页面  request.getRequestDispatcher("admin/AuditClassCreation.jsp").forward(request, response);  } }  **ClassDAO与数据库交互**  **(查询未审核班级)** public List<Class> getPendingClasses() throws SQLException {  List<Class> classList = new ArrayList<>();  String query = "SELECT c.id, c.class\_name, c.created\_at, c.status, c.teacher\_id, u.username, c.class\_briefly " +  "FROM classes c " +  "JOIN users u ON c.teacher\_id = u.id WHERE c.status = 'pending'";  try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(query);  ResultSet rs = stmt.executeQuery()) {   while (rs.next()) {  Class newClass = new Class();  newClass.setId(rs.getInt("id"));  newClass.setClassName(rs.getString("class\_name"));  newClass.setCreatedAt(rs.getString("created\_at"));  newClass.setStatus(rs.getString("status"));  newClass.setTeacherId(rs.getInt("teacher\_id"));  newClass.setTeacherName(rs.getString("username"));  newClass.setClassBriefly(rs.getString("class\_briefly"));  classList.add(newClass);  }  }  return classList; }  **(改变审核班级状态)**  public boolean updateClassStatus(int classId, String status) throws SQLException {  String updateQuery = "UPDATE classes SET status = ? WHERE id = ?";  try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(updateQuery)) {  stmt.setString(1, status);  stmt.setInt(2, classId);  int rowsUpdated = stmt.executeUpdate();  return rowsUpdated > 0;  } }  **（3）管理系统通知**  1.通知展示：  **通知展示控制类:** import dao.NotificationDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class ToShowAdminNotification extends HttpServlet {  public void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException  {  // 设置响应内容类型  response.setContentType("text/html");  NotificationDAO Notification = new NotificationDAO();  List<Notification> res = null;   HttpSession session = request.getSession();  Integer adminId = (Integer) session.getAttribute("adminId");   try {  res = Notification.getNotificationsByAdmin(adminId);  } catch (SQLException e) {  throw new RuntimeException(e);  }  if(res != null && !res.isEmpty()){  request.setAttribute("notifications",res);  }  request.getRequestDispatcher("admin/manageNotifications.jsp").forward(request, response);  } }  **通过NotificationDAO与数据库进行交互** public List<Notification> getNotificationsByAdmin(int adminId) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();  try {  conn = DatabaseUtil.getConnection();  String query = "SELECT id, title, content, created\_at FROM admin\_notifications WHERE admin\_id = ?";  stmt = conn.prepareStatement(query);  stmt.setInt(1, adminId);  rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setId(rs.getInt("id"));  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));  notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }  2.通知添加  **添加通知控制类：** package controller;  import service.NotificationService; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession;  import java.io.IOException;  public class AddAdminNotificationServlet extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String title = request.getParameter("title");  String content = request.getParameter("content");   HttpSession session = request.getSession();  Integer adminId = (Integer) session.getAttribute("adminId");   if (adminId == null) {  session.setAttribute("error", "管理员未登录！");  response.sendRedirect("index.jsp");  return;  }   NotificationService notificationService = new NotificationService();  boolean isSuccess = notificationService.addNotification(adminId, title, content);   if (isSuccess) {  session.setAttribute("message", "通知发布成功！");  response.sendRedirect("ToShowAdminNotification");  } else {  session.setAttribute("error", "发布通知失败，请稍后再试。");  response.sendRedirect("ToShowAdminNotification");  }  } }  **通知删除控制类** package controller;  import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import service.NotificationService;  import java.io.IOException;    public class DeleteAdminNotification extends HttpServlet {  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  int id = Integer.parseInt(request.getParameter("notification\_id"));   HttpSession session = request.getSession();  Integer adminId = (Integer) session.getAttribute("adminId");   if (adminId == null) {  session.setAttribute("error", "管理员未登录！");  response.sendRedirect("index.jsp");  return;  }   NotificationService notificationService = new NotificationService();  boolean isSuccess = notificationService.deleteNotification(id);   if (isSuccess) {  session.setAttribute("message", "通知删除成功！");  response.sendRedirect("ToShowAdminNotification");  } else {  session.setAttribute("error", "删除通知失败，请稍后再试。");  response.sendRedirect("ToShowAdminNotification");  }  } }    **数据库操作对象类(包含通知删除)** package service;  import dao.NotificationDAO; import model.Notification;  import java.sql.SQLException; import java.util.List;  public class NotificationService {  private NotificationDAO notificationDAO;   public NotificationService() {  this.notificationDAO = new NotificationDAO();  }   public boolean addNotification(int adminId, String title, String content) {  try {  return notificationDAO.addNotification(adminId, title, content);  } catch (SQLException e) {  e.printStackTrace();  return false;  }  }   public boolean deleteNotification(int notificationId) {  try {  return notificationDAO.deleteNotification(notificationId);  } catch (SQLException e) {  e.printStackTrace();  return false;  }  } }   1. 管理员页面   **老师：**  **(1)创建班级** **创建班级控制类** package controller;  import jakarta.servlet.ServletException; import jakarta.servlet.annotation.WebServlet; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import service.ClassService;  import java.io.IOException;  @WebServlet("/CreateClassController") public class CreateClassServlet extends HttpServlet {   private final ClassService classService = new ClassService();   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取表单数据  String className = request.getParameter("className");  String classBriefly = request.getParameter("class\_briefly");  Integer teacherId = (Integer) request.getSession().getAttribute("teacherId");   // 防止 teacherId 未登录异常  if (teacherId == null) {  request.setAttribute("error", "您尚未登录，请重新登录！");  request.getRequestDispatcher("teacher/createClass.jsp").forward(request, response);  return;  }   try {  // 调用服务层处理创建班级的逻辑  String result = classService.createClass(className, teacherId, classBriefly);   if ("success".equals(result)) {  request.setAttribute("message", "班级创建成功，请等待管理员审核！");  } else {  request.setAttribute("error", result);  }  } catch (Exception e) {  e.printStackTrace();  request.setAttribute("error", "系统错误：班级创建失败，请稍后重试！");  }   // 转发回 JSP 显示结果  request.getRequestDispatcher("teacher/createClass.jsp").forward(request, response);  } }  **封装业务逻辑** package service;  import dao.ClassDAO; import java.sql.SQLException;  public class ClassService {   private final ClassDAO classDAO = new ClassDAO();   // 创建班级的业务逻辑  public String createClass(String className, int teacherId, String classBriefly) {  try {  // 检查班级名是否已存在  if (classDAO.isClassNameExists(className)) {  return "班级名已存在，请选择其他班级名！";  }   // 插入班级信息  boolean isSuccess = classDAO.createClass(className, teacherId, classBriefly);  return isSuccess ? "success" : "班级创建失败，请稍后重试！";  } catch (SQLException e) {  e.printStackTrace();  return "系统错误：班级创建失败！";  }  }  }  **数据访问对象类(DAO)**  // 创建班级 public boolean createClass(String className, int teacherId, String classBriefly) throws SQLException {  String query = "INSERT INTO classes (class\_name, teacher\_id, status, class\_briefly) VALUES (?, ?, 'pending', ?)";  try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(query)) {  stmt.setString(1, className);  stmt.setInt(2, teacherId);  stmt.setString(3, classBriefly);  return stmt.executeUpdate() > 0;  } }  **(2)加入班级**  **加入班级控制类** import com.example.myapplication.util.DatabaseUtil; import dao.TeacherDAO; import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.\*; import java.sql.\*;  public class JoinClassServlet extends HttpServlet {   // 处理POST请求  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  int classId = Integer.parseInt(request.getParameter("class\_id"));  Integer teacherId = (Integer) request.getSession().getAttribute("teacherId");   if (teacherId == null) {  response.sendRedirect("../index.jsp"); // 如果没有登录，跳转到登录页面  return;  }  TeacherDAO teacherDAO = new TeacherDAO();  try {  conn = DatabaseUtil.getConnection();   // 检查是否已经申请该班级  String checkQuery = "SELECT approval\_status FROM teacher\_classes WHERE teacher\_id = ? AND class\_id = ?";  stmt = conn.prepareStatement(checkQuery);  stmt.setInt(1, teacherId);  stmt.setInt(2, classId);  rs = stmt.executeQuery();   if (rs.next()) {  String approvalStatus = rs.getString("approval\_status");  if ("pending".equals(approvalStatus)) {  request.getSession().setAttribute("error", "您已经申请过该班级，待审核中");  } else if ("approved".equals(approvalStatus)) {  request.getSession().setAttribute("error", "您已经加入该班级");  } else if ("rejected".equals(approvalStatus)) {  request.getSession().setAttribute("error", "您曾申请加入该班级，但被拒绝");  }  } else {  // 插入申请记录  String fin = teacherDAO.TeacherJoinClass(teacherId,classId);    if (fin.equals("true")) {  request.getSession().setAttribute("message", "您的加入申请已提交，待审核");  } else {  request.getSession().setAttribute("error", "加入班级失败，请稍后再试");  }  }   // 重定向回班级加入页面  response.sendRedirect(request.getContextPath() + "/teacher/joinClass.jsp");  } catch (SQLException e) {  e.printStackTrace();  request.getSession().setAttribute("error", "数据库操作错误，请稍后重试");  response.sendRedirect(request.getContextPath() + "/teacher/joinClass.jsp");  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  } }  **数据访问对象类（DAO）** public String TeacherJoinClass(int teacherId,int classId) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  String insertQuery = "INSERT INTO teacher\_classes (teacher\_id, class\_id, approval\_status) VALUES (?, ?, 'pending')";  stmt = conn.prepareStatement(insertQuery);  stmt.setInt(1, teacherId);  stmt.setInt(2, classId);  int rowsAffected = stmt.executeUpdate();   if (rowsAffected > 0) {  return "true";  } else {  return "false";  }  **（3）管理班级**  **管理班级控制类**  import dao.ClassDAO; import dao.NotificationDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Class; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class ToShowTeacherClasses extends HttpServlet {  public void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException  {  // 设置响应内容类型  response.setContentType("text/html");  ClassDAO classDAO = new ClassDAO();  List<Class> res = null;   HttpSession session = request.getSession();  Integer teacherId = (Integer) session.getAttribute("teacherId");   try {  res = classDAO.getClassesbyTeacher(teacherId);  } catch (SQLException e) {  throw new RuntimeException(e);  }  if(res != null && !res.isEmpty()){  request.setAttribute("Classes",res);  }  request.getRequestDispatcher("teacher/manageClass.jsp").forward(request, response);  } }  **（4）管理班级加入申请** **管理班级加入代码**  <tbody>  <%  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;   // 获取当前登录的教师ID  Integer teacherId = (Integer) request.getSession().getAttribute("teacherId");  if (teacherId == null) {  response.sendRedirect("../index.jsp"); // 如果没有登录，跳转到登录页面  return;  }   try {  conn = DatabaseUtil.getConnection();  int creatorId = (int) session.getAttribute("teacherId"); // 当前登录教师的ID   // 查询待审核的教师申请  String queryTeacher = "SELECT tc.id, tc.teacher\_id, tc.class\_id, tc.approval\_status, u.username, c.class\_name " +  "FROM teacher\_classes tc " +  "JOIN classes c ON tc.class\_id = c.id " +  "JOIN users u ON tc.teacher\_id = u.id " +  "WHERE c.teacher\_id = ? AND tc.approval\_status = 'pending'";  stmt = conn.prepareStatement(queryTeacher);  stmt.setInt(1, creatorId);  rs = stmt.executeQuery();   while (rs.next()) {  int requestId = rs.getInt("id");  String applicantName = rs.getString("username");  String className = rs.getString("class\_name");  String approvalStatus = rs.getString("approval\_status"); %> <tr>  <td><%= requestId %></td>  <td><%= applicantName %> (教师)</td>  <td><%= className %></td>  <td><%= approvalStatus %></td>  <td>  <form method="post" action="${pageContext.request.contextPath}/ApproveJoinRequestServlet" style="display: inline;">  <input type="hidden" name="requestId" value="<%= requestId %>">  <input type="hidden" name="action" value="approve">  <input type="hidden" name="type" value="teacher"> <!-- 如果是教师申请 -->  <button type="submit" class="btn" style="background-color: #4caf50;">通过</button>  </form>   <form method="post" action="${pageContext.request.contextPath}/ApproveJoinRequestServlet" style="display: inline;">  <input type="hidden" name="requestId" value="<%= requestId %>">  <input type="hidden" name="action" value="reject">  <input type="hidden" name="type" value="teacher"> <!-- 如果是教师申请 -->  <button type="submit" class="btn deny">拒绝</button>  </form>   </td> </tr> <%  } } catch (Exception e) {  e.printStackTrace(); %> <tr>  <td colspan="5">加载数据失败，请稍后重试！</td> </tr> <%  } finally {  DatabaseUtil.close(conn, stmt, rs);  } %> </tbody>  **（5）管理班级通知(查看和删除)** **通知删除控制类** package controller;  import com.example.myapplication.util.DatabaseUtil; import dao.NotificationDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse;  import java.io.IOException; import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.SQLException;  public class DeleteClassNotificationServlet extends HttpServlet {  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取通知ID  String notificationIdStr = request.getParameter("notification\_id");   if (notificationIdStr == null || notificationIdStr.isEmpty()) {  request.getSession().setAttribute("error", "通知ID无效！");  response.sendRedirect("/teacher/manageClassNotifications.jsp");  return;  }   int notificationId;  try {  notificationId = Integer.parseInt(notificationIdStr);  } catch (NumberFormatException e) {  request.getSession().setAttribute("error", "通知ID无效！");  response.sendRedirect("/teacher/manageClassNotifications.jsp");  return;  }   Connection conn = null;  PreparedStatement stmt = null;   try {  NotificationDAO notificationDAO = new NotificationDAO();    int rowsAffected = notificationDAO.deleteClassNotification(notificationId);   if (rowsAffected > 0) {  request.getSession().setAttribute("message", "通知删除成功！");  } else {  request.getSession().setAttribute("error", "通知删除失败，请重试！");  }   } catch (SQLException e) {  e.printStackTrace();  request.getSession().setAttribute("error", "服务器错误，请稍后重试！");  } finally {  DatabaseUtil.close(conn, stmt, null);  }   // 重定向到通知管理页面  response.sendRedirect("/teacher/manageClassNotifications.jsp");  } }  **数据访问对象类（DAO）** public int deleteClassNotification(int notificationId) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  try {  conn = DatabaseUtil.getConnection();  String query = "DELETE FROM class\_notifications WHERE id = ?";  stmt = conn.prepareStatement(query);  stmt.setInt(1, notificationId);  int rowsAffected = stmt.executeUpdate();  return rowsAffected;  } finally {  DatabaseUtil.close(conn, stmt, null);  } }  public List<Notification> getClassNotificationsByTeacher(int teacherId) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();  try {  conn = DatabaseUtil.getConnection();  // 修改SQL查询，联接查询班级名称  String query = "SELECT cn.id, cn.title, cn.content, cn.created\_at, cn.class\_id, c.class\_name\n" +  "FROM class\_notifications cn\n" +  "JOIN classes c ON cn.class\_id = c.id\n" +  "WHERE c.teacher\_id = ? -- 教师创建的班级的通知\n" +  "\n" +  "UNION\n" +  "\n" +  "SELECT cn.id, cn.title, cn.content, cn.created\_at, cn.class\_id, c.class\_name\n" +  "FROM class\_notifications cn\n" +  "JOIN classes c ON cn.class\_id = c.id\n" +  "JOIN teacher\_classes tc ON c.id = tc.class\_id\n" +  "WHERE tc.teacher\_id = ? -- 教师加入的班级的通知\n"  ;  stmt = conn.prepareStatement(query);  stmt.setInt(1, teacherId);  stmt.setInt(2, teacherId);  rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setId(rs.getInt("id"));  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setClass\_name(rs.getString("class\_name"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));   notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }  **（6）发送消息 发送消息控制类** package controller;  import dao.MessageDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse;  import java.io.IOException;  public class SendMessageServlet extends HttpServlet {   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取当前登录的用户ID (老师或者家长)  Integer senderId = (Integer) request.getSession().getAttribute("teacherId");  if( senderId == null ){  senderId = (Integer) request.getSession().getAttribute("parentId");  }   int receiverId = Integer.parseInt(request.getParameter("receiverId")); // 获取接收方的ID (家长)  String content = request.getParameter("content"); // 获取消息内容   MessageDAO messageDAO = new MessageDAO();  String string = messageDAO.sendMessage(senderId,receiverId,content);  request.setAttribute("resultMessage", string); //发送结果消息   // 发送完消息后，重定向回发送消息的页面  Integer teacherId = (Integer) request.getSession().getAttribute("teacherId");  if( teacherId != null ){  request.getRequestDispatcher("teacher/sendMessages.jsp").forward(request, response);  }else{  request.getRequestDispatcher("parent/sendMessages.jsp").forward(request, response);  }   } }  **数据访问对象类(DAO)**  public String sendMessage(int senderId,int receiverId,String content){  Connection conn = null;  PreparedStatement stmt = null;   try {  conn = DatabaseUtil.getConnection();   // 插入消息到 messages 表  String insertQuery = "INSERT INTO messages (sender\_id, receiver\_id, content, created\_at) VALUES (?, ?, ?, ?)";  stmt = conn.prepareStatement(insertQuery);  stmt.setInt(1, senderId);  stmt.setInt(2, receiverId);  stmt.setString(3, content);  stmt.setTimestamp(4, new Timestamp(System.currentTimeMillis())); // 当前时间   int rowsAffected = stmt.executeUpdate();  if (rowsAffected > 0) {  return "消息发送成功！";  } else {  return "消息发送失败，请稍后重试！";  }   } catch (Exception e) {  e.printStackTrace();  return "服务器错误，请稍后重试！";  } finally {  DatabaseUtil.close(conn, stmt, null);  } }  **(7)查询站内消息** **查询站内消息控制类** package controller;  import model.Message; import service.MessageService;  import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.IOException; import java.util.List;  public class QueryMessagesServlet extends HttpServlet {  private final MessageService messageService = new MessageService();   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  Integer userId = (Integer) session.getAttribute("parentId"); // 当前登录用户的 ID  if( userId == null ){  userId = (Integer) session.getAttribute("teacherId");  }   String sender = request.getParameter("sender"); //发送者  String receiver = request.getParameter("receiver"); //接受者  String dateFrom = request.getParameter("dateFrom"); //开始日期  String dateTo = request.getParameter("dateTo"); //结束日期  String role = request.getParameter("role"); // 判断用户是发送方还是接收方  String content = request.getParameter("content"); //获取查询文本   boolean isSender = "sender".equals(role);   List<Message> messages = messageService.getMessages(userId, sender, receiver, dateFrom, dateTo, isSender,content);   request.setAttribute("messages", messages);  Integer teacherId = (Integer) session.getAttribute("teacherId");  Integer parentId = (Integer) session.getAttribute("parentId");  if (teacherId != null){  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/queryMessages.jsp");  dispatcher.forward(request, response);  }else  if( parentId != null){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/queryMessages.jsp");  dispatcher.forward(request, response);  }  } }  **封装业务逻辑(service)** package service;  import dao.MessageDAO; import model.Message;  import java.util.List;  public class MessageService {  private final MessageDAO messageDAO = new MessageDAO();   public List<Message> getMessages(Integer userId, String sender, String receiver, String dateFrom, String dateTo, boolean isSender,String content) {  return messageDAO.queryMessages(userId, sender, receiver, dateFrom, dateTo, isSender,content);  } }  **数据访问对象类(DAO)**  public List<Message> queryMessages(Integer userId, String sender, String receiver, String dateFrom, String dateTo, boolean isSender,String content) {  List<Message> messages = new ArrayList<>();  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;   try {  conn = DatabaseUtil.getConnection();  StringBuilder query = new StringBuilder(  "SELECT u1.username AS sender, u2.username AS receiver, m.content, m.created\_at " +  "FROM messages m " +  "JOIN users u1 ON m.sender\_id = u1.id " +  "JOIN users u2 ON m.receiver\_id = u2.id " +  "WHERE 1=1 "  );   // 当前用户是发送方或接收方  if (isSender) {  query.append("AND m.sender\_id = ? ");  } else {  query.append("AND m.receiver\_id = ? ");  }   if (sender != null && !sender.isEmpty()) {  query.append("AND u1.username LIKE ? ");  }  if (receiver != null && !receiver.isEmpty()) {  query.append("AND u2.username LIKE ? ");  }  if (dateFrom != null && !dateFrom.isEmpty()) {  query.append("AND m.created\_at >= ? ");  }  if (dateTo != null && !dateTo.isEmpty()) {  query.append("AND m.created\_at <= ? ");  }  if (content != null && !content.isEmpty()) {  query.append(" AND m.content LIKE ?");  }   stmt = conn.prepareStatement(query.toString());  int paramIndex = 1;   // 当前用户的 ID  stmt.setInt(paramIndex++, userId);   if (sender != null && !sender.isEmpty()) {  stmt.setString(paramIndex++, "%" + sender + "%");  }  if (receiver != null && !receiver.isEmpty()) {  stmt.setString(paramIndex++, "%" + receiver + "%");  }  if (dateFrom != null && !dateFrom.isEmpty()) {  stmt.setDate(paramIndex++, Date.valueOf(dateFrom));  }  if (dateTo != null && !dateTo.isEmpty()) {  stmt.setDate(paramIndex++, Date.valueOf(dateTo));  }  if (content != null && !content.isEmpty()) {  stmt.setString(paramIndex++, "%" + content + "%");  }   rs = stmt.executeQuery();  while (rs.next()) {  Message message = new Message();  message.setSender(rs.getString("sender"));  message.setReceiver(rs.getString("receiver"));  message.setContent(rs.getString("content"));  message.setCreatedAt(rs.getTimestamp("created\_at"));  messages.add(message);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return messages; }  **(8)查询班级通知**  **查询班级通知控制类** package controller;  import dao.NotificationDAO; import jakarta.servlet.RequestDispatcher; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class SearchClassNotifications extends HttpServlet {   // 处理查询班级通知的请求  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  Integer userId = (Integer) session.getAttribute("parentId");  String role = "parent";  if ( userId == null){  userId = (Integer) session.getAttribute("teacherId");  role = "teacher";  }   if (userId == null) {  response.sendRedirect("index.jsp");  return;  }   // 获取查询参数  String title = request.getParameter("title");  String content = request.getParameter("content");  String startDate = request.getParameter("startDate");  String endDate = request.getParameter("endDate");     try {  NotificationDAO notificationDAO = new NotificationDAO();  List<Notification> notifications = notificationDAO.getClassNotifications(userId,role ,title,content, startDate, endDate);   // 将查询结果传递到 JSP  request.setAttribute("notifications", notifications);  if( role.equals("parent") ){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  } else if (role.equals("teacher")) {  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  }    } catch (SQLException e) {  e.printStackTrace();  request.setAttribute("errorMessage", "加载通知时发生错误，请稍后重试。");  if( role.equals("parent") ){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  } else if (role.equals("teacher")) {  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  }  }  } }  **数据访问对象类(DAO)**  public List<Notification> getClassNotifications(int userId, String role, String title, String content, String startDate, String endDate) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();   try {  conn = DatabaseUtil.getConnection();  String query = null;  // 构建 SQL 查询语句  if (role.equals("parent")) {  query = "SELECT c.title, c.content, c.created\_at FROM class\_notifications c " +  "JOIN parent\_classes p ON c.class\_id = p.class\_id " +  "WHERE p.parent\_id = ?";  } else if (role.equals("teacher")) {  query = "SELECT c.title, c.content, c.created\_at " +  "FROM class\_notifications c " +  "JOIN teacher\_classes tc ON c.class\_id = tc.class\_id " +  "JOIN classes cl ON c.class\_id = cl.id " +  "WHERE (tc.teacher\_id = ? AND tc.approval\_status = 'approved') " +  "OR (cl.teacher\_id = ?)";   }   // 根据条件拼接查询  if (title != null && !title.isEmpty()) {  query += " AND c.title LIKE ?";  }  if (content != null && !content.isEmpty()) {  query += " AND c.content LIKE ?";  }  if (startDate != null && !startDate.isEmpty()) {  query += " AND c.created\_at >= ?";  }  if (endDate != null && !endDate.isEmpty()) {  query += " AND c.created\_at <= ?";  }   stmt = conn.prepareStatement(query);   // 设置查询参数  stmt.setInt(1, userId);  if (role.equals("teacher")){  stmt.setInt(2, userId); // 同样设置教师ID，分别用于条件  }  int index = 2;  if( role.equals("teacher") ){  index += 1;  }  if (title != null && !title.isEmpty()) {  stmt.setString(index++, "%" + title + "%");  }  if (content != null && !content.isEmpty()) {  stmt.setString(index++, "%" + content + "%");  }  if (startDate != null && !startDate.isEmpty()) {  stmt.setDate(index++, Date.valueOf(startDate));  }  if (endDate != null && !endDate.isEmpty()) {  stmt.setDate(index++, Date.valueOf(endDate));  }   rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));  notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }  **(9)查看系统通知** **查看系统通知代码**  <tbody> <%  List<Notification> notifications = null;  NotificationDAO notificationDAO = new NotificationDAO();  try {  // 获取系统通知  notifications = notificationDAO.getAdminNotifications();   // 检查通知列表是否为空  if (notifications != null && !notifications.isEmpty()) {  for (Notification notification : notifications) { %> <tr>  <td><%= notification.getTitle() %></td>  <td><%= notification.getContent() %></td>  <td><%= notification.getCreatedAt() %></td> </tr> <%  } } else { %> <tr>  <td colspan="3" class="error-message">暂无通知</td> </tr> <%  } } catch (Exception e) {  e.printStackTrace(); %> <tr>  <td colspan="3" class="error-message">加载失败，请稍后重试。</td> </tr> <%  } %> </tbody>    **家长：** **（1）申请加入班级**  **加入班级控制类**： package controller;  import dao.ParentClassDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import model.Class;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class ParentJoinClassServlet extends HttpServlet {   private final ParentClassDAO parentClassDAO = new ParentClassDAO();   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取家长ID和班级ID  int classId = Integer.parseInt(request.getParameter("class\_id"));  int parentId = (Integer) request.getSession().getAttribute("parentId");   // 调用业务逻辑层申请加入班级  String message = null;  try {  message = parentClassDAO.applyForClass(parentId, classId);  } catch (SQLException e) {  throw new RuntimeException(e);  }   // 将消息存储在session中，供JSP页面使用  request.getSession().setAttribute("message", message);   // 重定向回班级列表页面  response.sendRedirect("ToShowParentClass");  }   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取家长ID  int parentId = (Integer) request.getSession().getAttribute("parentId");   try {  // 获取家长可加入的班级列表  List<Class> availableClasses = parentClassDAO.getAvailableClassesForParent(parentId);  // 将班级列表存放到请求属性中  request.setAttribute("availableClasses", availableClasses);  // 转发请求到JSP页面  request.getRequestDispatcher("ToShowParentClass").forward(request, response);  } catch (SQLException e) {  e.printStackTrace();  // 如果发生异常，设置错误信息  request.setAttribute("error", "加载班级信息失败，请稍后再试！");  // 转发到JSP页面并显示错误  request.getRequestDispatcher("ToShowParentClass").forward(request, response);  }  } }  **数据访问对象类** package dao;  import com.example.myapplication.util.DatabaseUtil; import java.sql.\*; import java.util.ArrayList; import java.util.List; import model.Class;  public class ParentClassDAO {   // 申请加入班级  public String applyForClass(int parentId, int classId) throws SQLException {  String checkQuery = "SELECT COUNT(\*) FROM parent\_classes WHERE parent\_id = ? AND class\_id = ?";  try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(checkQuery)) {  stmt.setInt(1, parentId);  stmt.setInt(2, classId);  try (ResultSet rs = stmt.executeQuery()) {  if (rs.next() && rs.getInt(1) > 0) {  return "您已经加入该班级！";  }  }  }   String insertQuery = "INSERT INTO parent\_classes (parent\_id, class\_id) VALUES (?, ?)";  try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(insertQuery)) {  stmt.setInt(1, parentId);  stmt.setInt(2, classId);  int rowsInserted = stmt.executeUpdate();  return rowsInserted > 0 ? "加入班级申请成功！" : "加入班级申请失败，请稍后再试！";  }  }   // 获取家长可加入的班级列表  public List<Class> getAvailableClassesForParent(int parentId) throws SQLException {  List<Class> classList = new ArrayList<>();  String query = "SELECT c.id, c.class\_name, c.class\_briefly, c.status " +  "FROM classes c WHERE c.status = 'approved' " +  "AND NOT EXISTS (SELECT 1 FROM parent\_classes pc WHERE pc.parent\_id = ? AND pc.class\_id = c.id)";   try (Connection conn = DatabaseUtil.getConnection();  PreparedStatement stmt = conn.prepareStatement(query)) {  stmt.setInt(1, parentId);  try (ResultSet rs = stmt.executeQuery()) {  while (rs.next()) {  Class newClass = new Class();  newClass.setId(rs.getInt("id"));  newClass.setClassName(rs.getString("class\_name"));  newClass.setClassBriefly(rs.getString("class\_briefly"));  newClass.setStatus(rs.getString("status"));  classList.add(newClass);  }  }  }  return classList;  } }  **(2)查看班级通知**  **查看班级通知控制类** import dao.NotificationDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class ToShowParentClassNotification extends HttpServlet {  public void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException  {  // 设置响应内容类型  response.setContentType("text/html");   // 创建 NotificationDAO 实例  NotificationDAO notificationDAO = new NotificationDAO();  List<Notification> res = null;   // 获取当前会话的家长ID  HttpSession session = request.getSession();  Integer parentId = (Integer) session.getAttribute("parentId");   // 获取家长相关的班级通知  try {  res = notificationDAO.getClassNotificationsByParent(parentId);  } catch (SQLException e) {  e.printStackTrace();  throw new ServletException("数据库操作失败", e);  }   // 如果查询到通知，将结果放入请求属性中  if (res != null && !res.isEmpty()) {  request.setAttribute("notifications", res);  } else {  request.setAttribute("error", "没有找到任何通知。");  }   // 转发到家长通知页面  request.getRequestDispatcher("parent/viewClassNotifications.jsp").forward(request, response);  } }  **数据访问对象类** public List<Notification> getClassNotificationsByParent(int parentId) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();  try {  conn = DatabaseUtil.getConnection();  String query = "SELECT c.title, c.content, c.created\_at FROM class\_notifications c " +  "JOIN parent\_classes p ON c.class\_id = p.class\_id " +  "WHERE p.parent\_id = ? ORDER BY c.created\_at DESC";  stmt = conn.prepareStatement(query);  stmt.setInt(1, parentId);  rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));  notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }  **(3)查看系统通知** **查看系统通知控制类；** import dao.NotificationDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class ToShowParentAdminNotification extends HttpServlet {  public void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException  {  // 设置响应内容类型  response.setContentType("text/html");   // 创建 NotificationDAO 实例  NotificationDAO notificationDAO = new NotificationDAO();  List<Notification> res = null;   // 获取当前会话的家长ID  HttpSession session = request.getSession();  Integer parentId = (Integer) session.getAttribute("parentId");   // 获取家长相关的班级通知  try {  res = notificationDAO.getAdminNotifications();  } catch (SQLException e) {  e.printStackTrace();  throw new ServletException("数据库操作失败", e);  }   // 如果查询到通知，将结果放入请求属性中  if (res != null && !res.isEmpty()) {  request.setAttribute("Adminnotifications", res);  } else {  request.setAttribute("error", "没有找到任何通知。");  }   // 转发到家长通知页面  request.getRequestDispatcher("parent/viewSystemNotifications.jsp").forward(request, response);  } }  **数据访问对象类：** public List<Notification> getAdminNotifications() throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();  try {  conn = DatabaseUtil.getConnection();  String query = "SELECT id, title, content, created\_at FROM admin\_notifications";  stmt = conn.prepareStatement(query);  rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setId(rs.getInt("id"));  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));  notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }  （4）与老师沟通  **发送消息控制类（跟老师的发送消息是同一个类）**  package controller;  import dao.MessageDAO; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse;  import java.io.IOException;  public class SendMessageServlet extends HttpServlet {   @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // 获取当前登录的用户ID (老师或者家长)  Integer senderId = (Integer) request.getSession().getAttribute("teacherId");  if( senderId == null ){  senderId = (Integer) request.getSession().getAttribute("parentId");  }   int receiverId = Integer.parseInt(request.getParameter("receiverId")); // 获取接收方的ID (家长)  String content = request.getParameter("content"); // 获取消息内容   MessageDAO messageDAO = new MessageDAO();  String string = messageDAO.sendMessage(senderId,receiverId,content);  request.setAttribute("resultMessage", string); //发送结果消息   // 发送完消息后，重定向回发送消息的页面  Integer teacherId = (Integer) request.getSession().getAttribute("teacherId");  if( teacherId != null ){  request.getRequestDispatcher("teacher/sendMessages.jsp").forward(request, response);  }else{  request.getRequestDispatcher("parent/sendMessages.jsp").forward(request, response);  }  } }  **数据访问对象类(DAO)的引用**  public String sendMessage(int senderId,int receiverId,String content){  Connection conn = null;  PreparedStatement stmt = null;   try {  conn = DatabaseUtil.getConnection();   // 插入消息到 messages 表  String insertQuery = "INSERT INTO messages (sender\_id, receiver\_id, content, created\_at) VALUES (?, ?, ?, ?)";  stmt = conn.prepareStatement(insertQuery);  stmt.setInt(1, senderId);  stmt.setInt(2, receiverId);  stmt.setString(3, content);  stmt.setTimestamp(4, new Timestamp(System.currentTimeMillis())); // 当前时间   int rowsAffected = stmt.executeUpdate();  if (rowsAffected > 0) {  return "消息发送成功！";  } else {  return "消息发送失败，请稍后重试！";  }   } catch (Exception e) {  e.printStackTrace();  return "服务器错误，请稍后重试！";  } finally {  DatabaseUtil.close(conn, stmt, null);  } }  (5)查询站内消息  **查询站内消息控制类** package controller;  import model.Message; import service.MessageService;  import jakarta.servlet.\*; import jakarta.servlet.http.\*; import java.io.IOException; import java.util.List;  public class QueryMessagesServlet extends HttpServlet {  private final MessageService messageService = new MessageService();   @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  Integer userId = (Integer) session.getAttribute("parentId"); // 当前登录用户的 ID  if( userId == null ){  userId = (Integer) session.getAttribute("teacherId");  }   String sender = request.getParameter("sender"); //发送者  String receiver = request.getParameter("receiver"); //接受者  String dateFrom = request.getParameter("dateFrom"); //开始日期  String dateTo = request.getParameter("dateTo"); //结束日期  String role = request.getParameter("role"); // 判断用户是发送方还是接收方  String content = request.getParameter("content"); //获取查询文本   boolean isSender = "sender".equals(role);   List<Message> messages = messageService.getMessages(userId, sender, receiver, dateFrom, dateTo, isSender,content);   request.setAttribute("messages", messages);  Integer teacherId = (Integer) session.getAttribute("teacherId");  Integer parentId = (Integer) session.getAttribute("parentId");  if (teacherId != null){  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/queryMessages.jsp");  dispatcher.forward(request, response);  }else  if( parentId != null){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/queryMessages.jsp");  dispatcher.forward(request, response);  }  } }  **封装业务逻辑(service)** package service;  import dao.MessageDAO; import model.Message;  import java.util.List;  public class MessageService {  private final MessageDAO messageDAO = new MessageDAO();   public List<Message> getMessages(Integer userId, String sender, String receiver, String dateFrom, String dateTo, boolean isSender,String content) {  return messageDAO.queryMessages(userId, sender, receiver, dateFrom, dateTo, isSender,content);  } }  **数据访问对象类(DAO)**  public List<Message> queryMessages(Integer userId, String sender, String receiver, String dateFrom, String dateTo, boolean isSender,String content) {  List<Message> messages = new ArrayList<>();  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;   try {  conn = DatabaseUtil.getConnection();  StringBuilder query = new StringBuilder(  "SELECT u1.username AS sender, u2.username AS receiver, m.content, m.created\_at " +  "FROM messages m " +  "JOIN users u1 ON m.sender\_id = u1.id " +  "JOIN users u2 ON m.receiver\_id = u2.id " +  "WHERE 1=1 "  );   // 当前用户是发送方或接收方  if (isSender) {  query.append("AND m.sender\_id = ? ");  } else {  query.append("AND m.receiver\_id = ? ");  }   if (sender != null && !sender.isEmpty()) {  query.append("AND u1.username LIKE ? ");  }  if (receiver != null && !receiver.isEmpty()) {  query.append("AND u2.username LIKE ? ");  }  if (dateFrom != null && !dateFrom.isEmpty()) {  query.append("AND m.created\_at >= ? ");  }  if (dateTo != null && !dateTo.isEmpty()) {  query.append("AND m.created\_at <= ? ");  }  if (content != null && !content.isEmpty()) {  query.append(" AND m.content LIKE ?");  }   stmt = conn.prepareStatement(query.toString());  int paramIndex = 1;   // 当前用户的 ID  stmt.setInt(paramIndex++, userId);   if (sender != null && !sender.isEmpty()) {  stmt.setString(paramIndex++, "%" + sender + "%");  }  if (receiver != null && !receiver.isEmpty()) {  stmt.setString(paramIndex++, "%" + receiver + "%");  }  if (dateFrom != null && !dateFrom.isEmpty()) {  stmt.setDate(paramIndex++, Date.valueOf(dateFrom));  }  if (dateTo != null && !dateTo.isEmpty()) {  stmt.setDate(paramIndex++, Date.valueOf(dateTo));  }  if (content != null && !content.isEmpty()) {  stmt.setString(paramIndex++, "%" + content + "%");  }   rs = stmt.executeQuery();  while (rs.next()) {  Message message = new Message();  message.setSender(rs.getString("sender"));  message.setReceiver(rs.getString("receiver"));  message.setContent(rs.getString("content"));  message.setCreatedAt(rs.getTimestamp("created\_at"));  messages.add(message);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return messages; }  (6)查询班级通知 **查询班级通知控制类** package controller;  import dao.NotificationDAO; import jakarta.servlet.RequestDispatcher; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServlet; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import jakarta.servlet.http.HttpSession; import model.Notification;  import java.io.IOException; import java.sql.SQLException; import java.util.List;  public class SearchClassNotifications extends HttpServlet {   // 处理查询班级通知的请求  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  Integer userId = (Integer) session.getAttribute("parentId");  String role = "parent";  if ( userId == null){  userId = (Integer) session.getAttribute("teacherId");  role = "teacher";  }   if (userId == null) {  response.sendRedirect("index.jsp");  return;  }   // 获取查询参数  String title = request.getParameter("title");  String content = request.getParameter("content");  String startDate = request.getParameter("startDate");  String endDate = request.getParameter("endDate");     try {  NotificationDAO notificationDAO = new NotificationDAO();  List<Notification> notifications = notificationDAO.getClassNotifications(userId,role ,title,content, startDate, endDate);   // 将查询结果传递到 JSP  request.setAttribute("notifications", notifications);  if( role.equals("parent") ){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  } else if (role.equals("teacher")) {  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  }    } catch (SQLException e) {  e.printStackTrace();  request.setAttribute("errorMessage", "加载通知时发生错误，请稍后重试。");  if( role.equals("parent") ){  RequestDispatcher dispatcher = request.getRequestDispatcher("parent/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  } else if (role.equals("teacher")) {  RequestDispatcher dispatcher = request.getRequestDispatcher("teacher/classNotificationsQuery.jsp");  dispatcher.forward(request, response);  }  }  } }  数据访问对象类(DAO) public List<Notification> getClassNotifications(int userId, String role, String title, String content, String startDate, String endDate) throws SQLException {  Connection conn = null;  PreparedStatement stmt = null;  ResultSet rs = null;  List<Notification> notifications = new ArrayList<>();   try {  conn = DatabaseUtil.getConnection();  String query = null;  // 构建 SQL 查询语句  if (role.equals("parent")) {  query = "SELECT c.title, c.content, c.created\_at FROM class\_notifications c " +  "JOIN parent\_classes p ON c.class\_id = p.class\_id " +  "WHERE p.parent\_id = ?";  } else if (role.equals("teacher")) {  query = "SELECT c.title, c.content, c.created\_at " +  "FROM class\_notifications c " +  "JOIN teacher\_classes tc ON c.class\_id = tc.class\_id " +  "JOIN classes cl ON c.class\_id = cl.id " +  "WHERE (tc.teacher\_id = ? AND tc.approval\_status = 'approved') " +  "OR (cl.teacher\_id = ?)";   }   // 根据条件拼接查询  if (title != null && !title.isEmpty()) {  query += " AND c.title LIKE ?";  }  if (content != null && !content.isEmpty()) {  query += " AND c.content LIKE ?";  }  if (startDate != null && !startDate.isEmpty()) {  query += " AND c.created\_at >= ?";  }  if (endDate != null && !endDate.isEmpty()) {  query += " AND c.created\_at <= ?";  }   stmt = conn.prepareStatement(query);   // 设置查询参数  stmt.setInt(1, userId);  if (role.equals("teacher")){  stmt.setInt(2, userId); // 同样设置教师ID，分别用于条件  }  int index = 2;  if( role.equals("teacher") ){  index += 1;  }  if (title != null && !title.isEmpty()) {  stmt.setString(index++, "%" + title + "%");  }  if (content != null && !content.isEmpty()) {  stmt.setString(index++, "%" + content + "%");  }  if (startDate != null && !startDate.isEmpty()) {  stmt.setDate(index++, Date.valueOf(startDate));  }  if (endDate != null && !endDate.isEmpty()) {  stmt.setDate(index++, Date.valueOf(endDate));  }   rs = stmt.executeQuery();  while (rs.next()) {  Notification notification = new Notification();  notification.setTitle(rs.getString("title"));  notification.setContent(rs.getString("content"));  notification.setCreatedAt(rs.getTimestamp("created\_at"));  notifications.add(notification);  }  } finally {  DatabaseUtil.close(conn, stmt, rs);  }  return notifications; }    **五、程序运行时的输入数据/输出结果**  **（管理员默认账号密码均为 admin）**   1. 首先新创建一个老师账号，名字密码均为qqq   **2.直接登录会提示用户未审核**    **3.需要登录管理员对老师进行过审**      **4.在管理员界面可以对系统通知进行管理（也能进行发布通知）**        **5.现在回到老师，审核过后可以登录进去了**    **6.老师可以创建班级**      **这时可以发现自己创建的班级进入待审核状态**    **7.现在切换到管理员进行对班级审核 （可以发现我们新创的班级在带审核列表，点击“通过”来通过审核）**    **8.返回老师页面，班级已经通过了审核**    **9.可以发布班级通知了**  **（对照上面的表，输入班级id）**      **10.现在换到家长视角，（www为家长名字和密码）**    **点击“申请加入班级”，加入刚刚创建的班2412**    **11.回到老师视角，通过家长加入班级审核**      **12.老师现在可以发送消息给www家长了**      **13.登录家长可以发现接受到老师的消息**    **发送消息回去**    **14.返回老师界面**  **可以发现家长发的消息**  **15.班级通知查询**    **六、问题与收获**  ****问题：  1.****在jsp页面怎么将数据，提前处理然后展示在页面，而不是通过按钮刷新才展示  2.怎么区分管理员、老师和管理员，session该存储哪个ID  3.老师和班级，班级和通知，通知和老师怎么链接在一起，通过什么链接  4.老师进入班级后，应该可以看的到自己进入的班级的其它老师发布的通知  5.通知查询时，能通过标题、通知内容和发送时间查询；而信息查询能通过区分接受者和发送者，然后通过内容和时间查询  等等问题  ****Java EE技术深入理解****：  我们对Java Web技术有了更深入的了解，包括Servlet、JSP、、JDBC等关键技术。  能使用MVC架构继续项目搭建，能区分Model,DAO,Service,Control在MVC模式上的作用  通过这次项目实践，我对JavaEE 的技术有了详细的了解，较好地运用知识  ****项目开发实践能力提升****：  经历了从需求分析、规划、框架设计、编写代码到整个项目开发流程。  通过这个过程，提升了项目开发实践能力，学会了如何在实际项目中运用所学知识。  ****问题解决能力增强****：  在遇到问题时，学会了首先找到问题大致位置，然后通过调试，一步步找到问题的位置，找到详细的问题  然后分析问题原因，寻找解决方案并进行验证，一步步进行调试，直至解决问题。  通过解决问题，提升了自身的逻辑思维能力和问题解决能力。  ****技术视野拓展****：  通过这次项目，了解到许多的编程知识，拓宽了我的视野  为我们未来的学习和职业发展提供了更广阔的技术视野和选择空间。 | | | | | |
|  | | | | | |