-- 创建数据库

CREATE DATABASE error\_management;

USE error\_management;

-- 用户表

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

role ENUM('tester', 'developer', 'manager', 'admin') NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- 项目表

CREATE TABLE Projects (

project\_id INT PRIMARY KEY AUTO\_INCREMENT,

project\_name VARCHAR(100) NOT NULL,

description TEXT,

start\_date DATE,

end\_date DATE,

manager\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (manager\_id) REFERENCES Users(user\_id)

);

-- 错误表

CREATE TABLE Bugs (

bug\_id INT PRIMARY KEY AUTO\_INCREMENT,

project\_id INT,

title VARCHAR(100) NOT NULL,

description TEXT,

priority ENUM('low', 'medium', 'high') NOT NULL,

status ENUM('new', 'in\_progress', 'resolved', 'closed') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

reporter\_id INT,

assignee\_id INT,

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id),

FOREIGN KEY (reporter\_id) REFERENCES Users(user\_id),

FOREIGN KEY (assignee\_id) REFERENCES Users(user\_id)

);

-- 错误历史记录表

CREATE TABLE BugHistory (

history\_id INT PRIMARY KEY AUTO\_INCREMENT,

bug\_id INT,

status ENUM('new', 'in\_progress', 'resolved', 'closed') NOT NULL,

comment TEXT,

attachment VARCHAR(255),

changed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

changed\_by INT,

FOREIGN KEY (bug\_id) REFERENCES Bugs(bug\_id),

FOREIGN KEY (changed\_by) REFERENCES Users(user\_id)

);

-- 项目成员表

CREATE TABLE ProjectMembers (

project\_id INT,

user\_id INT,

role ENUM('tester', 'developer', 'manager'),

PRIMARY KEY (project\_id, user\_id),

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- 系统日志表

CREATE TABLE SystemLogs (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

action VARCHAR(255),

log\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- 通知表

CREATE TABLE Notifications (

notification\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

message VARCHAR(255),

is\_read BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

-- 文件附件表

CREATE TABLE Attachments (

attachment\_id INT PRIMARY KEY AUTO\_INCREMENT,

bug\_id INT,

file\_path VARCHAR(255),

uploaded\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (bug\_id) REFERENCES Bugs(bug\_id)

);

-- 任务表（扩展功能）

CREATE TABLE Tasks (

task\_id INT PRIMARY KEY AUTO\_INCREMENT,

project\_id INT,

assigned\_to INT,

title VARCHAR(100) NOT NULL,

description TEXT,

status ENUM('pending', 'in\_progress', 'completed') DEFAULT 'pending',

due\_date DATE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id),

FOREIGN KEY (assigned\_to) REFERENCES Users(user\_id)

);

package com.example.utils;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

public class DBConnection {

private static Connection connection;

static {

try {

Properties props = new Properties();

props.load(DBConnection.class.getClassLoader().getResourceAsStream("database.properties"));

String url = props.getProperty("jdbc.url");

String username = props.getProperty("jdbc.username");

String password = props.getProperty("jdbc.password");

connection = DriverManager.getConnection(url, username, password);

} catch (Exception e) {

e.printStackTrace();

}

}

public static Connection getConnection() {

return connection;

}

}

jdbc.url=jdbc:mysql://localhost:3306/error\_management

jdbc.username=root

jdbc.password=yourpassword

jdbc.driverClassName=com.mysql.cj.jdbc.Driver

**用户管理模块**

**用户模型**

package com.example.model;

public class User {

private int userId;

private String username;

private String password;

private String role;

private String email;

public User(int userId, String username, String password, String role, String email) {

this.userId = userId;

this.username = username;

this.password = password;

this.role = role;

this.email = email;

}

// Getters and Setters

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**用户 DAO**

package com.example.dao;

import com.example.model.User;

import com.example.utils.DBConnection;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class UserDAO {

public User getUserByUsername(String username) throws SQLException {

Connection connection = DBConnection.getConnection();

String query = "SELECT \* FROM Users WHERE username = ?";

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, username);

ResultSet resultSet = preparedStatement.executeQuery();

if (resultSet.next()) {

return new User(

resultSet.getInt("user\_id"),

resultSet.getString("username"),

resultSet.getString("password"),

resultSet.getString("role"),

resultSet.getString("email")

);

}

return null;

}

public boolean createUser(User user) throws SQLException {

Connection connection = DBConnection.getConnection();

String query = "INSERT INTO Users (username, password, role, email) VALUES (?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, user.getUsername());

preparedStatement.setString(2, user.getPassword());

preparedStatement.setString(3, user.getRole());

preparedStatement.setString(4, user.getEmail());

return preparedStatement.executeUpdate() > 0;

}

}

**用户控制器**

package com.example.controller;

import com.example.dao.UserDAO;

import com.example.model.User;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import java.io.IOException;

@WebServlet("/user")

public class UserServlet extends HttpServlet {

private final UserDAO userDAO = new UserDAO();

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String action = req.getParameter("action");

if ("login".equals(action)) {

handleLogin(req, resp);

} else if ("register".equals(action)) {

handleRegister(req, resp);

}

}

private void handleLogin(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

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

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

try {

User user = userDAO.getUserByUsername(username);

if (user != null && user.getPassword().equals(password)) {

HttpSession session = req.getSession();

session.setAttribute("user", user);

resp.sendRedirect("dashboard.jsp");

} else {

req.setAttribute("error", "Invalid credentials");

req.getRequestDispatcher("login.jsp").forward(req, resp);

}

} catch (Exception e) {

e.printStackTrace();

resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

}

}

private void handleRegister(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

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

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

String email = req.getParameter("email");

String role = req.getParameter("role");

User user = new User(0, username, password, role, email);

try {

if (userDAO.createUser(user)) {

resp.sendRedirect("login.jsp");

} else {

req.setAttribute("error", "Failed to register");

req.getRequestDispatcher("register.jsp").forward(req, resp);

}

} catch (Exception e) {

e.printStackTrace();

resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

}

}

}

**登录页面**

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form method="post" action="user">

<input type="hidden" name="action" value="login" />

<label>Username: <input type="text" name="username" required /></label><br/>

<label>Password: <input type="password" name="password" required /></label><br/>

<button type="submit">Login</button>

</form>

<p style="color: red;"><%= request.getAttribute("error") %></p>

</body>

</html>

**数据模型**

**错误模型 Bug.java**

package com.example.model;

import java.sql.Timestamp;

public class Bug {

private int bugId;

private int projectId;

private String title;

private String description;

private String priority;

private String status;

private Timestamp createdAt;

private Timestamp updatedAt;

private int reporterId;

private int assigneeId;

public Bug(int bugId, int projectId, String title, String description, String priority,

String status, Timestamp createdAt, Timestamp updatedAt, int reporterId, int assigneeId) {

this.bugId = bugId;

this.projectId = projectId;

this.title = title;

this.description = description;

this.priority = priority;

this.status = status;

this.createdAt = createdAt;

this.updatedAt = updatedAt;

this.reporterId = reporterId;

this.assigneeId = assigneeId;

}

// Getters and Setters

public int getBugId() {

return bugId;

}

public void setBugId(int bugId) {

this.bugId = bugId;

}

public int getProjectId() {

return projectId;

}

public void setProjectId(int projectId) {

this.projectId = projectId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getPriority() {

return priority;

}

public void setPriority(String priority) {

this.priority = priority;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public Timestamp getCreatedAt() {

return createdAt;

}

public void setCreatedAt(Timestamp createdAt) {

this.createdAt = createdAt;

}

public Timestamp getUpdatedAt() {

return updatedAt;

}

public void setUpdatedAt(Timestamp updatedAt) {

this.updatedAt = updatedAt;

}

public int getReporterId() {

return reporterId;

}

public void setReporterId(int reporterId) {

this.reporterId = reporterId;

}

public int getAssigneeId() {

return assigneeId;

}

public void setAssigneeId(int assigneeId) {

this.assigneeId = assigneeId;

}

}

**数据访问对象 (DAO)**

**错误 DAO BugDAO.java**

package com.example.controller;

import com.example.dao.BugDAO;

import com.example.model.Bug;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

@WebServlet("/bug")

public class BugServlet extends HttpServlet {

private final BugDAO bugDAO = new BugDAO();

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String action = req.getParameter("action");

if ("report".equals(action)) {

handleReportBug(req, resp);

} else if ("updateStatus".equals(action)) {

handleUpdateBugStatus(req, resp);

}

}

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String action = req.getParameter("action");

if ("list".equals(action)) {

handleListBugs(req, resp);

}

}

private void handleReportBug(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

try {

int projectId = Integer.parseInt(req.getParameter("projectId"));

String title = req.getParameter("title");

String description = req.getParameter("description");

String priority = req.getParameter("priority");

int reporterId = Integer.parseInt(req.getParameter("reporterId"));

int assigneeId = Integer.parseInt(req.getParameter("assigneeId"));

Bug bug = new Bug(0, projectId, title, description, priority, "new", null, null, reporterId, assigneeId);

int bugId = bugDAO.reportBug(bug);

if (bugId > 0) {

resp.sendRedirect("bugList.jsp");

} else {

req.setAttribute("error", "Failed to report bug");

req.getRequestDispatcher("reportBug.jsp").forward(req, resp);

}

} catch (Exception e) {

e.printStackTrace();

resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

}

}

private void handleUpdateBugStatus(HttpServletRequest req, HttpServletResponse resp) throws IOException {

try {

int bugId = Integer.parseInt(req.getParameter("bugId"));

String status = req.getParameter("status");

if (bugDAO.updateBugStatus(bugId, status)) {

resp.sendRedirect("bugList.jsp");

} else {

resp.sendError(HttpServletResponse.SC\_BAD\_REQUEST, "Failed to update bug status");

}

} catch (Exception e) {

e.printStackTrace();

resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

}

}

private void handleListBugs(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

try {

int projectId = Integer.parseInt(req.getParameter("projectId"));

List<Bug> bugs = bugDAO.getBugsByProject(projectId);

req.setAttribute("bugs", bugs);

req.getRequestDispatcher("bugList.jsp").forward(req, resp);

} catch (SQLException e) {

e.printStackTrace();

resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);

}

}

}

**控制器**

**错误控制器 BugServlet.java**

**package com.example.controller;**

**import com.example.dao.BugDAO;**

**import com.example.model.Bug;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.\*;**

**import java.io.IOException;**

**import java.sql.SQLException;**

**import java.util.List;**

**@WebServlet("/bug")**

**public class BugServlet extends HttpServlet {**

**private final BugDAO bugDAO = new BugDAO();**

**@Override**

**protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**String action = req.getParameter("action");**

**if ("report".equals(action)) {**

**handleReportBug(req, resp);**

**} else if ("updateStatus".equals(action)) {**

**handleUpdateBugStatus(req, resp);**

**}**

**}**

**@Override**

**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**String action = req.getParameter("action");**

**if ("list".equals(action)) {**

**handleListBugs(req, resp);**

**}**

**}**

**private void handleReportBug(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {**

**try {**

**int projectId = Integer.parseInt(req.getParameter("projectId"));**

**String title = req.getParameter("title");**

**String description = req.getParameter("description");**

**String priority = req.getParameter("priority");**

**int reporterId = Integer.parseInt(req.getParameter("reporterId"));**

**int assigneeId = Integer.parseInt(req.getParameter("assigneeId"));**

**Bug bug = new Bug(0, projectId, title, description, priority, "new", null, null, reporterId, assigneeId);**

**int bugId = bugDAO.reportBug(bug);**

**if (bugId > 0) {**

**resp.sendRedirect("bugList.jsp");**

**} else {**

**req.setAttribute("error", "Failed to report bug");**

**req.getRequestDispatcher("reportBug.jsp").forward(req, resp);**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**private void handleUpdateBugStatus(HttpServletRequest req, HttpServletResponse resp) throws IOException {**

**try {**

**int bugId = Integer.parseInt(req.getParameter("bugId"));**

**String status = req.getParameter("status");**

**if (bugDAO.updateBugStatus(bugId, status)) {**

**resp.sendRedirect("bugList.jsp");**

**} else {**

**resp.sendError(HttpServletResponse.SC\_BAD\_REQUEST, "Failed to update bug status");**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**private void handleListBugs(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**try {**

**int projectId = Integer.parseInt(req.getParameter("projectId"));**

**List<Bug> bugs = bugDAO.getBugsByProject(projectId);**

**req.setAttribute("bugs", bugs);**

**req.getRequestDispatcher("bugList.jsp").forward(req, resp);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**}**

**JSP 页面**

**报告错误页面 reportBug.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Report Bug</title>**

**</head>**

**<body>**

**<h1>Report Bug</h1>**

**<form method="post" action="bug">**

**<input type="hidden" name="action" value="report" />**

**<input type="hidden" name="reporterId" value="<%= session.getAttribute("userId") %>" />**

**<label>Project ID: <input type="text" name="projectId" required /></label><br/>**

**<label>Title: <input type="text" name="title" required /></label><br/>**

**<label>Description: <textarea name="description" required></textarea></label><br/>**

**<label>Priority:**

**<select name="priority" required>**

**<option value="low">Low</option>**

**<option value="medium">Medium</option>**

**<option value="high">High</option>**

**</select>**

**</label><br/>**

**<label>Assignee ID: <input type="text" name="assigneeId" required /></label><br/>**

**<button type="submit">Submit</button>**

**</form>**

**</body>**

**</html>**

**错误列表页面 bugList.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Bug List</title>**

**</head>**

**<body>**

**<h1>Bug List</h1>**

**<table border="1">**

**<tr>**

**<th>ID</th>**

**<th>Title</th>**

**<th>Description</th>**

**<th>Priority</th>**

**<th>Status</th>**

**<th>Assignee</th>**

**</tr>**

**<%**

**List<Bug> bugs = (List<Bug>) request.getAttribute("bugs");**

**if (bugs != null) {**

**for (Bug bug : bugs) {**

**%>**

**<tr>**

**<td><%= bug.getBugId() %></td>**

**<td><%= bug.getTitle() %></td>**

**<td><%= bug.getDescription() %></td>**

**<td><%= bug.getPriority() %></td>**

**<td><%= bug.getStatus() %></td>**

**<td><%= bug.getAssigneeId() %></td>**

**</tr>**

**<% }**

**}**

**%>**

**</table>**

**</body>**

**</html>**

**数据模型**

**项目模型 Project.java**

**package com.example.model;**

**import java.sql.Date;**

**import java.sql.Timestamp;**

**public class Project {**

**private int projectId;**

**private String projectName;**

**private String description;**

**private Date startDate;**

**private Date endDate;**

**private int managerId;**

**private Timestamp createdAt;**

**private Timestamp updatedAt;**

**public Project(int projectId, String projectName, String description, Date startDate, Date endDate, int managerId, Timestamp createdAt, Timestamp updatedAt) {**

**this.projectId = projectId;**

**this.projectName = projectName;**

**this.description = description;**

**this.startDate = startDate;**

**this.endDate = endDate;**

**this.managerId = managerId;**

**this.createdAt = createdAt;**

**this.updatedAt = updatedAt;**

**}**

**// Getters and Setters**

**public int getProjectId() {**

**return projectId;**

**}**

**public void setProjectId(int projectId) {**

**this.projectId = projectId;**

**}**

**public String getProjectName() {**

**return projectName;**

**}**

**public void setProjectName(String projectName) {**

**this.projectName = projectName;**

**}**

**public String getDescription() {**

**return description;**

**}**

**public void setDescription(String description) {**

**this.description = description;**

**}**

**public Date getStartDate() {**

**return startDate;**

**}**

**public void setStartDate(Date startDate) {**

**this.startDate = startDate;**

**}**

**public Date getEndDate() {**

**return endDate;**

**}**

**public void setEndDate(Date endDate) {**

**this.endDate = endDate;**

**}**

**public int getManagerId() {**

**return managerId;**

**}**

**public void setManagerId(int managerId) {**

**this.managerId = managerId;**

**}**

**public Timestamp getCreatedAt() {**

**return createdAt;**

**}**

**public void setCreatedAt(Timestamp createdAt) {**

**this.createdAt = createdAt;**

**}**

**public Timestamp getUpdatedAt() {**

**return updatedAt;**

**}**

**public void setUpdatedAt(Timestamp updatedAt) {**

**this.updatedAt = updatedAt;**

**}**

**}**

**数据访问对象 (DAO)**

**项目 DAO ProjectDAO.java**

**package com.example.dao;**

**import com.example.model.Project;**

**import com.example.utils.DBConnection;**

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class ProjectDAO {**

**public int createProject(Project project) throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "INSERT INTO Projects (project\_name, description, start\_date, end\_date, manager\_id) " +**

**"VALUES (?, ?, ?, ?, ?)";**

**PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS);**

**preparedStatement.setString(1, project.getProjectName());**

**preparedStatement.setString(2, project.getDescription());**

**preparedStatement.setDate(3, project.getStartDate());**

**preparedStatement.setDate(4, project.getEndDate());**

**preparedStatement.setInt(5, project.getManagerId());**

**preparedStatement.executeUpdate();**

**ResultSet resultSet = preparedStatement.getGeneratedKeys();**

**if (resultSet.next()) {**

**return resultSet.getInt(1);**

**}**

**return -1;**

**}**

**public List<Project> getAllProjects() throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "SELECT \* FROM Projects";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**ResultSet resultSet = preparedStatement.executeQuery();**

**List<Project> projects = new ArrayList<>();**

**while (resultSet.next()) {**

**Project project = new Project(**

**resultSet.getInt("project\_id"),**

**resultSet.getString("project\_name"),**

**resultSet.getString("description"),**

**resultSet.getDate("start\_date"),**

**resultSet.getDate("end\_date"),**

**resultSet.getInt("manager\_id"),**

**resultSet.getTimestamp("created\_at"),**

**resultSet.getTimestamp("updated\_at")**

**);**

**projects.add(project);**

**}**

**return projects;**

**}**

**}**

**控制器**

**项目控制器 ProjectServlet.java**

**package com.example.controller;**

**import com.example.dao.ProjectDAO;**

**import com.example.model.Project;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.\*;**

**import java.io.IOException;**

**import java.sql.Date;**

**import java.sql.SQLException;**

**import java.util.List;**

**@WebServlet("/project")**

**public class ProjectServlet extends HttpServlet {**

**private final ProjectDAO projectDAO = new ProjectDAO();**

**@Override**

**protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**String action = req.getParameter("action");**

**if ("create".equals(action)) {**

**handleCreateProject(req, resp);**

**}**

**}**

**@Override**

**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**String action = req.getParameter("action");**

**if ("list".equals(action)) {**

**handleListProjects(req, resp);**

**}**

**}**

**private void handleCreateProject(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {**

**try {**

**String projectName = req.getParameter("projectName");**

**String description = req.getParameter("description");**

**Date startDate = Date.valueOf(req.getParameter("startDate"));**

**Date endDate = Date.valueOf(req.getParameter("endDate"));**

**int managerId = Integer.parseInt(req.getParameter("managerId"));**

**Project project = new Project(0, projectName, description, startDate, endDate, managerId, null, null);**

**int projectId = projectDAO.createProject(project);**

**if (projectId > 0) {**

**resp.sendRedirect("projectList.jsp");**

**} else {**

**req.setAttribute("error", "Failed to create project");**

**req.getRequestDispatcher("createProject.jsp").forward(req, resp);**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**private void handleListProjects(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**try {**

**List<Project> projects = projectDAO.getAllProjects();**

**req.setAttribute("projects", projects);**

**req.getRequestDispatcher("projectList.jsp").forward(req, resp);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**}**

**JSP 页面**

**创建项目页面 createProject.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Create Project</title>**

**</head>**

**<body>**

**<h1>Create Project</h1>**

**<form method="post" action="project">**

**<input type="hidden" name="action" value="create" />**

**<label>Project Name: <input type="text" name="projectName" required /></label><br/>**

**<label>Description: <textarea name="description" required></textarea></label><br/>**

**<label>Start Date: <input type="date" name="startDate" required /></label><br/>**

**<label>End Date: <input type="date" name="endDate" required /></label><br/>**

**<label>Manager ID: <input type="text" name="managerId" required /></label><br/>**

**<button type="submit">Create</button>**

**</form>**

**</body>**

**</html>**

**项目列表页面 projectList.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Project List</title>**

**</head>**

**<body>**

**<h1>Project List</h1>**

**<table border="1">**

**<tr>**

**<th>ID</th>**

**<th>Name</th>**

**<th>Description</th>**

**<th>Start Date</th>**

**<th>End Date</th>**

**<th>Manager</th>**

**</tr>**

**<%**

**List<Project> projects = (List<Project>) request.getAttribute("projects");**

**if (projects != null) {**

**for (Project project : projects) {**

**%>**

**<tr>**

**<td><%= project.getProjectId() %></td>**

**<td><%= project.getProjectName() %></td>**

**<td><%= project.getDescription() %></td>**

**<td><%= project.getStartDate() %></td>**

**<td><%= project.getEndDate() %></td>**

**<td><%= project.getManagerId() %></td>**

**</tr>**

**<% }**

**}**

**%>**

**</table>**

**</body>**

**</html>**

**项目列表页面 projectList.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Project List</title>**

**</head>**

**<body>**

**<h1>Project List</h1>**

**<table border="1">**

**<tr>**

**<th>ID</th>**

**<th>Name</th>**

**<th>Description</th>**

**<th>Start Date</th>**

**<th>End Date</th>**

**<th>Manager</th>**

**</tr>**

**<%**

**List<Project> projects = (List<Project>) request.getAttribute("projects");**

**if (projects != null) {**

**for (Project project : projects) {**

**%>**

**<tr>**

**<td><%= project.getProjectId() %></td>**

**<td><%= project.getProjectName() %></td>**

**<td><%= project.getDescription() %></td>**

**<td><%= project.getStartDate() %></td>**

**<td><%= project.getEndDate() %></td>**

**<td><%= project.getManagerId() %></td>**

**</tr>**

**<% }**

**}**

**%>**

**</table>**

**</body>**

**</html>**

**通知模型 Notification.java**

**package com.example.model;**

**import java.sql.Timestamp;**

**public class Notification {**

**private int notificationId;**

**private int userId;**

**private String message;**

**private boolean isRead;**

**private Timestamp createdAt;**

**public Notification(int notificationId, int userId, String message, boolean isRead, Timestamp createdAt) {**

**this.notificationId = notificationId;**

**this.userId = userId;**

**this.message = message;**

**this.isRead = isRead;**

**this.createdAt = createdAt;**

**}**

**// Getters and Setters**

**public int getNotificationId() {**

**return notificationId;**

**}**

**public void setNotificationId(int notificationId) {**

**this.notificationId = notificationId;**

**}**

**public int getUserId() {**

**return userId;**

**}**

**public void setUserId(int userId) {**

**this.userId = userId;**

**}**

**public String getMessage() {**

**return message;**

**}**

**public void setMessage(String message) {**

**this.message = message;**

**}**

**public boolean isRead() {**

**return isRead;**

**}**

**public void setRead(boolean read) {**

**isRead = read;**

**}**

**public Timestamp getCreatedAt() {**

**return createdAt;**

**}**

**public void setCreatedAt(Timestamp createdAt) {**

**this.createdAt = createdAt;**

**}**

**}**

**通知 DAO NotificationDAO.java**

**package com.example.dao;**

**import com.example.model.Notification;**

**import com.example.utils.DBConnection;**

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class NotificationDAO {**

**public boolean createNotification(int userId, String message) throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "INSERT INTO Notifications (user\_id, message, is\_read) VALUES (?, ?, false)";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**preparedStatement.setInt(1, userId);**

**preparedStatement.setString(2, message);**

**return preparedStatement.executeUpdate() > 0;**

**}**

**public List<Notification> getUserNotifications(int userId) throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "SELECT \* FROM Notifications WHERE user\_id = ? ORDER BY created\_at DESC";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**preparedStatement.setInt(1, userId);**

**ResultSet resultSet = preparedStatement.executeQuery();**

**List<Notification> notifications = new ArrayList<>();**

**while (resultSet.next()) {**

**Notification notification = new Notification(**

**resultSet.getInt("notification\_id"),**

**resultSet.getInt("user\_id"),**

**resultSet.getString("message"),**

**resultSet.getBoolean("is\_read"),**

**resultSet.getTimestamp("created\_at")**

**);**

**notifications.add(notification);**

**}**

**return notifications;**

**}**

**public boolean markAsRead(int notificationId) throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "UPDATE Notifications SET is\_read = true WHERE notification\_id = ?";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**preparedStatement.setInt(1, notificationId);**

**return preparedStatement.executeUpdate() > 0;**

**}**

**}**

**通知服务类 NotificationService.java**

**package com.example.service;**

**import com.example.dao.NotificationDAO;**

**import java.sql.SQLException;**

**public class NotificationService {**

**private final NotificationDAO notificationDAO = new NotificationDAO();**

**public void sendNotification(int userId, String message) {**

**try {**

**notificationDAO.createNotification(userId, message);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**通知控制器 NotificationServlet.java**

**package com.example.controller;**

**import com.example.dao.NotificationDAO;**

**import com.example.model.Notification;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.\*;**

**import java.io.IOException;**

**import java.sql.SQLException;**

**import java.util.List;**

**@WebServlet("/notifications")**

**public class NotificationServlet extends HttpServlet {**

**private final NotificationDAO notificationDAO = new NotificationDAO();**

**@Override**

**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**try {**

**int userId = Integer.parseInt(req.getParameter("userId"));**

**List<Notification> notifications = notificationDAO.getUserNotifications(userId);**

**req.setAttribute("notifications", notifications);**

**req.getRequestDispatcher("notifications.jsp").forward(req, resp);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**@Override**

**protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**try {**

**int notificationId = Integer.parseInt(req.getParameter("notificationId"));**

**if (notificationDAO.markAsRead(notificationId)) {**

**resp.sendRedirect("notifications?userId=" + req.getParameter("userId"));**

**} else {**

**resp.sendError(HttpServletResponse.SC\_BAD\_REQUEST, "Failed to mark notification as read");**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**}**

**通知列表页面 notifications.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Notifications</title>**

**</head>**

**<body>**

**<h1>Notifications</h1>**

**<table border="1">**

**<tr>**

**<th>ID</th>**

**<th>Message</th>**

**<th>Status</th>**

**<th>Action</th>**

**</tr>**

**<%**

**List<Notification> notifications = (List<Notification>) request.getAttribute("notifications");**

**if (notifications != null) {**

**for (Notification notification : notifications) {**

**%>**

**<tr>**

**<td><%= notification.getNotificationId() %></td>**

**<td><%= notification.getMessage() %></td>**

**<td><%= notification.isRead() ? "Read" : "Unread" %></td>**

**<td>**

**<% if (!notification.isRead()) { %>**

**<form method="post" action="notifications">**

**<input type="hidden" name="notificationId" value="<%= notification.getNotificationId() %>" />**

**<input type="hidden" name="userId" value="<%= notification.getUserId() %>" />**

**<button type="submit">Mark as Read</button>**

**</form>**

**<% } %>**

**</td>**

**</tr>**

**<% }**

**}**

**%>**

**</table>**

**</body>**

**</html>**

**系统日志模型 SystemLog.java**

**package com.example.model;**

**import java.sql.Timestamp;**

**public class SystemLog {**

**private int logId;**

**private int userId;**

**private String action;**

**private Timestamp logTime;**

**public SystemLog(int logId, int userId, String action, Timestamp logTime) {**

**this.logId = logId;**

**this.userId = userId;**

**this.action = action;**

**this.logTime = logTime;**

**}**

**// Getters and Setters**

**public int getLogId() {**

**return logId;**

**}**

**public void setLogId(int logId) {**

**this.logId = logId;**

**}**

**public int getUserId() {**

**return userId;**

**}**

**public void setUserId(int userId) {**

**this.userId = userId;**

**}**

**public String getAction() {**

**return action;**

**}**

**public void setAction(String action) {**

**this.action = action;**

**}**

**public Timestamp getLogTime() {**

**return logTime;**

**}**

**public void setLogTime(Timestamp logTime) {**

**this.logTime = logTime;**

**}**

**}**

**系统日志 DAO SystemLogDAO.java**

**package com.example.dao;**

**import com.example.model.SystemLog;**

**import com.example.utils.DBConnection;**

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class SystemLogDAO {**

**public void recordLog(int userId, String action) throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "INSERT INTO SystemLogs (user\_id, action) VALUES (?, ?)";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**preparedStatement.setInt(1, userId);**

**preparedStatement.setString(2, action);**

**preparedStatement.executeUpdate();**

**}**

**public List<SystemLog> getAllLogs() throws SQLException {**

**Connection connection = DBConnection.getConnection();**

**String query = "SELECT \* FROM SystemLogs ORDER BY log\_time DESC";**

**PreparedStatement preparedStatement = connection.prepareStatement(query);**

**ResultSet resultSet = preparedStatement.executeQuery();**

**List<SystemLog> logs = new ArrayList<>();**

**while (resultSet.next()) {**

**SystemLog log = new SystemLog(**

**resultSet.getInt("log\_id"),**

**resultSet.getInt("user\_id"),**

**resultSet.getString("action"),**

**resultSet.getTimestamp("log\_time")**

**);**

**logs.add(log);**

**}**

**return logs;**

**}**

**}**

**系统日志服务类 SystemLogService.java**

**package com.example.service;**

**import com.example.dao.SystemLogDAO;**

**import java.sql.SQLException;**

**public class SystemLogService {**

**private final SystemLogDAO systemLogDAO = new SystemLogDAO();**

**public void logAction(int userId, String action) {**

**try {**

**systemLogDAO.recordLog(userId, action);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**系统日志控制器 SystemLogServlet.java**

**package com.example.controller;**

**import com.example.dao.SystemLogDAO;**

**import com.example.model.SystemLog;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.\*;**

**import java.io.IOException;**

**import java.sql.SQLException;**

**import java.util.List;**

**@WebServlet("/systemLogs")**

**public class SystemLogServlet extends HttpServlet {**

**private final SystemLogDAO systemLogDAO = new SystemLogDAO();**

**@Override**

**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**try {**

**List<SystemLog> logs = systemLogDAO.getAllLogs();**

**req.setAttribute("logs", logs);**

**req.getRequestDispatcher("systemLogs.jsp").forward(req, resp);**

**} catch (SQLException e) {**

**e.printStackTrace();**

**resp.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR);**

**}**

**}**

**}**

**权限校验中间件 AuthorizationFilter.java**

**package com.example.middleware;**

**import javax.servlet.\*;**

**import javax.servlet.annotation.WebFilter;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import java.io.IOException;**

**@WebFilter("/\*")**

**public class AuthorizationFilter implements Filter {**

**@Override**

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

**HttpServletRequest httpRequest = (HttpServletRequest) request;**

**HttpServletResponse httpResponse = (HttpServletResponse) response;**

**HttpSession session = httpRequest.getSession(false);**

**String path = httpRequest.getRequestURI();**

**// Public pages (e.g., login, register)**

**if (path.endsWith("login.jsp") || path.endsWith("register.jsp") || path.endsWith("user")) {**

**chain.doFilter(request, response);**

**return;**

**}**

**// Check session for authentication**

**if (session == null || session.getAttribute("user") == null) {**

**httpResponse.sendRedirect("login.jsp");**

**return;**

**}**

**// Role-based access control**

**String role = (String) session.getAttribute("role");**

**if (path.startsWith("/admin") && !"admin".equals(role)) {**

**httpResponse.sendError(HttpServletResponse.SC\_FORBIDDEN, "Access Denied");**

**return;**

**}**

**chain.doFilter(request, response);**

**}**

**}**

**系统日志页面 systemLogs.jsp**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>System Logs</title>**

**</head>**

**<body>**

**<h1>System Logs</h1>**

**<table border="1">**

**<tr>**

**<th>ID</th>**

**<th>User ID</th>**

**<th>Action</th>**

**<th>Log Time</th>**

**</tr>**

**<%**

**List<SystemLog> logs = (List<SystemLog>) request.getAttribute("logs");**

**if (logs != null) {**

**for (SystemLog log : logs) {**

**%>**

**<tr>**

**<td><%= log.getLogId() %></td>**

**<td><%= log.getUserId() %></td>**

**<td><%= log.getAction() %></td>**

**<td><%= log.getLogTime() %></td>**

**</tr>**

**<% }**

**}**

**%>**

**</table>**

**</body>**

**</html>**

**数据备份脚本**

**#!/bin/bash**

**DB\_NAME="error\_management"**

**BACKUP\_PATH="./backup"**

**DATE=$(date +%Y%m%d\_%H%M%S)**

**FILENAME="${BACKUP\_PATH}/backup\_${DB\_NAME}\_${DATE}.sql"**

**mkdir -p "$BACKUP\_PATH"**

**mysqldump -u root -p --databases "$DB\_NAME" > "$FILENAME"**

**echo "Backup completed: $FILENAME"**

**数据恢复脚本**

**#!/bin/bash**

**DB\_NAME="error\_management"**

**BACKUP\_FILE=$1**

**if [ -z "$BACKUP\_FILE" ]; then**

**echo "Usage: $0 <backup\_file.sql>"**

**exit 1**

**fi**

**mysql -u root -p "$DB\_NAME" < "$BACKUP\_FILE"**

**echo "Restore completed: $BACKUP\_FILE"**