**\*\*\*\*\*\***

**电信学院结课项目（大作业）报告**

**课 程 Java企业级开发技术**

**专 业 软件工程**

**班 级 软件工程\*\*\*\*班**

**学生姓名 \*\*\***

**学 号　 \*\*\*\*\*\*\***

目录

[1. 需求分析 1](#_Toc185952556)

[1.1 功能需求 1](#_Toc185952557)

[1.2 质量需求 1](#_Toc185952558)

[2. 概要设计 2](#_Toc185952559)

[2.1 系统开发及运行环境 2](#_Toc185952560)

[2.2 系统架构设计 2](#_Toc185952561)

[2.3 项目组织结构 项目组织结构如图3所示： 4](#_Toc185952562)

[2.4 类图 5](#_Toc185952563)

[2.4.1 实体关系 5](#_Toc185952564)

[2.4.2 依赖关系 5](#_Toc185952565)

[2.4.3 设计特点 5](#_Toc185952566)

[2.5 主要时序图 6](#_Toc185952567)

[3. 详细设计 7](#_Toc185952568)

[3.1 MeetingController（会议管理控制器） 8](#_Toc185952569)

[3.2 CreateMeetingController（会议创建控制器） 8](#_Toc185952570)

[3.3 UpdateMeetingController（会议更新控制器） 8](#_Toc185952571)

[3.4 MeetingService（会议服务接口） 9](#_Toc185952572)

[3.5 MeetingServiceImpl（会议服务实现） 9](#_Toc185952573)

[3.6 MeetingMapper（会议数据访问接口） 10](#_Toc185952574)

[3.7 会议列表页面 10](#_Toc185952575)

[3.7.1 功能说明 10](#_Toc185952576)

[3.7.2 页面结构 10](#_Toc185952577)

[3.7.3 交互设计 11](#_Toc185952578)

[3.8 会议创建/编辑页面 11](#_Toc185952579)

[3.9 登录/注册流程 12](#_Toc185952580)

[3.10 创建会议流程 12](#_Toc185952581)

[3.11 更新会议流程 13](#_Toc185952582)

[4. 主要代码 14](#_Toc185952583)

[4.1 creatMeetingController 14](#_Toc185952584)

[4.2 LoginController 15](#_Toc185952585)

[5. 测试结果 17](#_Toc185952586)

[6. 项目总结 20](#_Toc185952587)

[6.1 不足及需要改进的地方 20](#_Toc185952588)

[6.2 遇到的问题 21](#_Toc185952589)

[6.3 收获的知识和经验 23](#_Toc185952590)

**会议管理系统**

1. 需求分析

## 功能需求

## 质量需求

1. 概要设计

## 系统开发及运行环境

操作系统：Windows 11

Web服务器：Tomcat 8.5.99

JDK：JDK 11

IDE：IntelliJ IDEA 2023.3.4Ultimate

Mybatis：3.5.\*

Spring MVC：5.3.25

数据库：MySQL 8.0.37

数据库图形化工具：Navicat Premium 16

浏览器：Edge、Chrome

## 系统架构设计

根据功能的不同，会议管理管理系统项目结构可以划分为以下几个层次。

持久对象层（持久层或持久化层）：该层由若干持久化类（实体类）组成。

数据访问层（DAO层）：该层由若干DAO接口和MyBatis映射文件组成。DAO接口的名称统一以Mapper结尾，且MyBatis的映射文件名称要与接口的名称相同。

业务逻辑层（Service层）：该层由若干Service接口和实现类组成。逻辑层主要用于实现系统的业务逻辑。

Web表现层：该层主要包括Spring MVC中的Controller类和JSP页面。Controller类主要负责拦截用户请求，并调用业务逻辑层中相应组件的业务逻辑方法来处理用户请求，然后将处理结果返回给JSP页面。

如图1、图2所示。



图1 系统结构设计图示1

图示

描述已自动生成

图2 系统架构设计图示2

## 项目组织结构 项目组织结构如图3所示：

图片包含 文本

描述已自动生成

图3 项目组织结构示意图

## 类图

### 实体关系

Meeting和MeetingParticipant：一对多关系

User和MeetingParticipant：一对一关系

User和Meeting：一对多关系

### 依赖关系

Controller层依赖Service层接口

Service实现类依赖Mapper接口

遵循依赖倒置原则，面向接口编程

### 设计特点

清晰的职责划分：每个类都有明确的职责

良好的封装性：通过接口隔离实现细节

合理的依赖关系：遵循依赖倒置原则

完整的业务功能：覆盖会议管理的主要用例

可扩展性：通过接口定义便于扩展新功能

图示

描述已自动生成

图4 类图

## 主要时序图

登录功能时序图以及创建会议功能时序图如图5、图6所示,更多详见详细设计

**图示

描述已自动生成**

图5 登录时序图

**图示

描述已自动生成**

图6 创建会议时序图

1. 详细设计

借助uml工具对几个主要控制器、Service层、DAO层和表现层的详细设计作说明：

## MeetingController（会议管理控制器）

处理会议确认/取消请求

图片包含 应用程序

描述已自动生成

图7 会议管理控制器

## CreateMeetingController（会议创建控制器）

处理会议创建请求

处理日期格式转换

图形用户界面, 文本, 应用程序, Teams

描述已自动生成

图8 会议创建控制器

## UpdateMeetingController（会议更新控制器）

处理会议列表展示

处理会议编辑和更新

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图9 会议更新控制器

## MeetingService（会议服务接口）

图形用户界面, 文本, 应用程序

描述已自动生成

图10 会议服务接口

## MeetingServiceImpl（会议服务实现）

图示

描述已自动生成

图11 会议服务实现主要作用

## MeetingMapper（会议数据访问接口）

文本

描述已自动生成

图12 会议数据访问接口

## 会议列表页面

以会议列表页面的设计为例作详细说明：

### 功能说明

 显示用户创建的所有会议列表

 提供会议的基本信息展示

 支持会议的编辑操作

 提供创建新会议的入口

### 页面结构

**顶部导航栏**

 显示用户信息

 提供系统导航菜单

 包含登出功能

**主内容区**

 空状态提示：当无会议时显示提示信息

 会议列表表格：包含以下列：会议主题、会议地点、开始时间、结束时间、操作按钮（编辑）

### 交互设计

**列表展示**

 按时间倒序排列会议

 时间格式化为"yyyy-MM-dd HH:mm"

**操作按钮**

 编辑按钮：跳转到会议编辑页面

 创建按钮：跳转到会议创建页面

图示

描述已自动生成

图13 会议列表页面图示

## 会议创建/编辑页面

图示

描述已自动生成

图14 会议创建/编剧页面图示

## 登录/注册流程

**图示

描述已自动生成**图15 登录/注册流程时序图

## 创建会议流程

图示

描述已自动生成

图16 创建会议流程时序图

## 更新会议流程

图示

描述已自动生成

图17 更新会议流程时序图

1. 主要代码

## creatMeetingController

**@Controller  
@RequestMapping("/creat\_meeting")  
public class creatMeetingController {  
  
 @Autowired  
 private UserService userService; // 获取用户列表  
  
 @Autowired  
 private MeetingService meetingService; // 处理会议相关操作  
 @InitBinder  
 public void initBinder(WebDataBinder binder) {  
 binder.addCustomFormatter(new DateFormatter("yyyy-MM-dd'T'HH:mm"));  
 }  
  
 // 处理创建会议表单提交  
 @PostMapping("/create")  
 public String createMeeting(HttpServletRequest request,  
 @RequestParam("meetingTitle") String meetingTitle,  
 @RequestParam("start\_time") Date start\_time,  
 @RequestParam("end\_time") Date end\_time,  
 @RequestParam("meetingLocation") String meetingLocation,  
 @RequestParam(value = "userIds", required = false) List<Integer> userIds) {  
 user currentUser = (user) request.getSession().getAttribute("USER\_SESSION");  
   
 // 创建会议对象  
 meeting newMeeting = new meeting(meetingTitle, meetingLocation, start\_time, end\_time,   
 currentUser.getUser\_id(), "scheduled");  
   
 // 创建会议并获取返回值  
 Integer result = meetingService.createMeeting(newMeeting);  
   
 // 打印日志，检查meeting\_id是否正确设置  
 System.out.println("Created meeting with ID: " + newMeeting.getMeeting\_id());  
   
 if (result > 0 && newMeeting.getMeeting\_id() > 0) {  
 // 如果有选择参会人员  
 if (userIds != null && !userIds.isEmpty()) {  
 for (Integer user\_id : userIds) {  
 meeting\_participants participant = new meeting\_participants(  
 newMeeting.getMeeting\_id(), // 使用新生成的meeting\_id  
 user\_id,  
 "pending"  
 );  
 if (!meetingService.addParticipant(participant)) {  
 request.setAttribute("msg", "Failed to add participant: " + user\_id);  
 return "forward:/jsp/creat\_meeting.jsp";  
 }  
 }  
 }  
 return "redirect:/to\_home";  
 } else {  
 request.setAttribute("msg", "Failed to create meeting");  
 return "forward:/jsp/creat\_meeting.jsp";  
 }  
 }  
}**

## LoginController

**@Controller  
public class LoginController {  
 @Autowired  
 private LoginService loginService;  
 @Autowired  
 private UserService userService;  
 // 跳转到主页面  
 @RequestMapping("/toMainPage")  
 public String toMainPage() {  
 return "home";  
 }  
 @RequestMapping("/toLogin")  
 //redirect:login或者redirect:toLogin就会访问没有样式的jsp  
 public String toLogin() {  
 return "forward:jsp/login.jsp";  
 }  
 // 用户登录  
 @RequestMapping("/login")  
 public String login(user user, HttpServletRequest request) {  
 try {  
 user u = loginService.login(user);  
 if (u != null) {  
 if(loginService.updateStatus(u))//更新用户状态  
 {  
 request.getSession().setAttribute("USER\_SESSION", u);//将刚刚登录的用户放到会话里  
 return "redirect:to\_home";  
 }  
 else{  
 request.setAttribute("msg", "登录失败，请重试");  
 return "forward:/jsp/login.jsp";  
 }  
 }  
 request.setAttribute("msg", "用户名或密码错误");  
 return "forward:/jsp/login.jsp";  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("msg", "未知系统错误");  
 return "forward:/jsp/login.jsp";  
 }  
 }  
  
 // 用户注册  
 @RequestMapping("/register")  
 public String register(user user, HttpServletRequest request) {  
 System.out.println("register user= "+user.toString());  
 try {  
 // 检查用户名是否已存在  
 if (loginService.checkUserExists(user.getUsername())) {  
 request.setAttribute("msg", "用户名已存在");  
 return "forward:/jsp/login.jsp";  
 }  
  
 // 注册用户  
 boolean success = loginService.register(user);  
 if (success) {  
 request.setAttribute("msg", "注册成功，请登录");  
 return "forward:/jsp/login.jsp";  
 } else {  
 request.setAttribute("msg", "注册失败，请稍后再试");  
 return "forward:/jsp/login.jsp";  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("msg", "系统错误");  
 return "forward:/jsp/login.jsp";  
 }  
 }  
 @RequestMapping("/logout")  
 public String logout(HttpServletRequest request) {  
 // 从 session 中获取登录的用户对象  
 user user = (user) request.getSession().getAttribute("USER\_SESSION");  
 System.out.println("logout");  
  
 if (user != null) {  
 System.out.println("need logout user id:"+user.getUsername());  
 // 调用 service 层的 logout 方法更新用户状态为 inactive  
 if(loginService.logout(user.getUser\_id()))  
 {  
  
 // 清除 session 中的用户对象  
 request.getSession().removeAttribute("USER\_SESSION");  
 // 跳转到登录页面或首页  
 //return "redirect:/login";  
 return "forward:jsp/login.jsp";  
 }  
 else {  
 System.out.println("登出失败");  
 request.setAttribute("msg", "登出失败");  
 return "redirect:to\_home";  
 }  
 } else {  
 System.out.println("用户未登录，跳转到登录页面");  
 // 用户未登录，跳转到登录页面  
 return "redirect:/login";  
 }  
 }  
}**

1. 测试结果

图形用户界面

描述已自动生成

图18 登录界面

图形用户界面, 应用程序, Teams

描述已自动生成

图19 首页概览

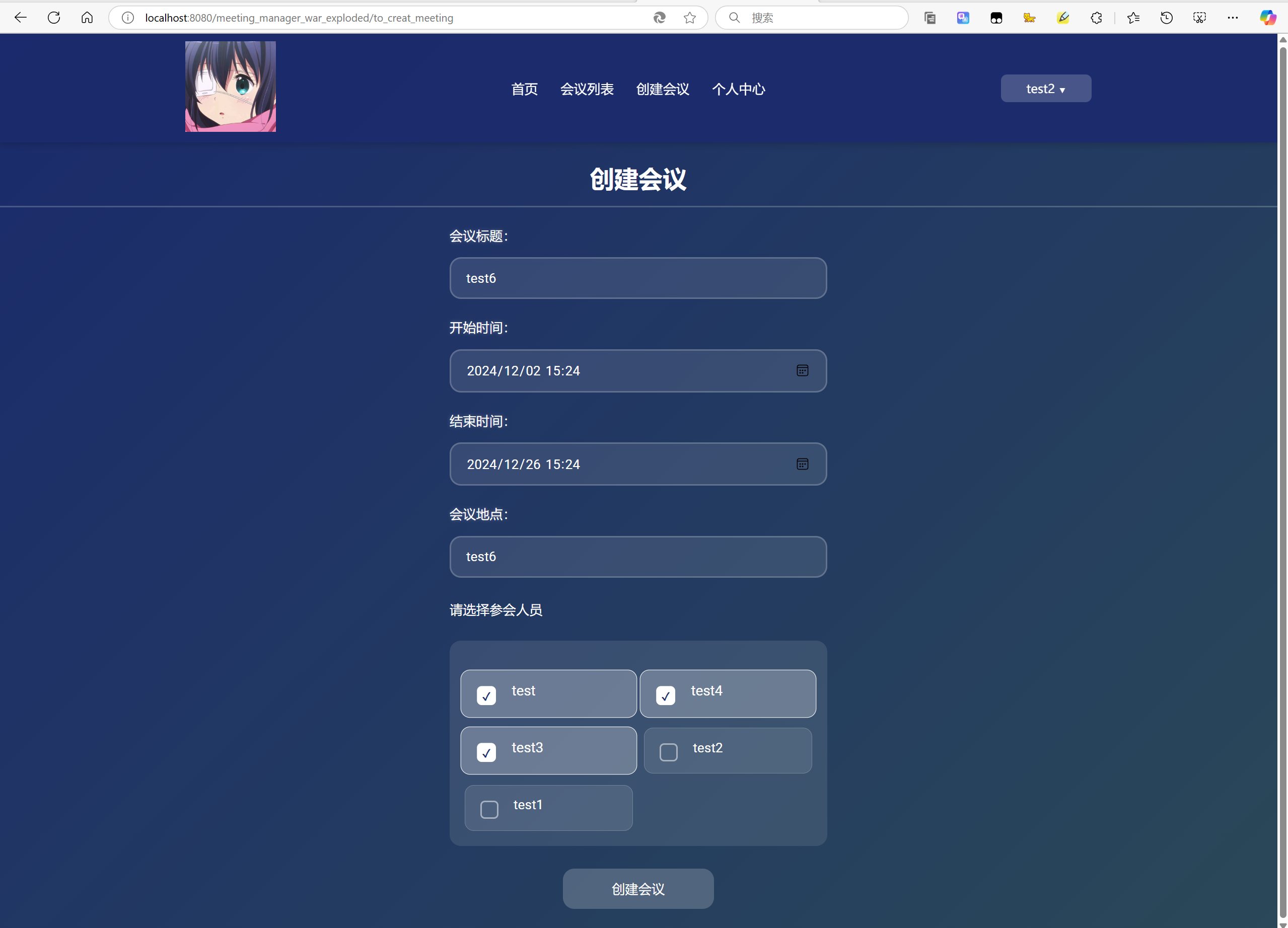


图20创建会议“test6”

电脑萤幕的截图

描述已自动生成

图21 查看会议列表，已经创建成功

图形用户界面, 应用程序

描述已自动生成

图22 修改个人资料（可以只更改基础资料，也可以修改密码）

图形用户界面, 应用程序, Teams

描述已自动生成

图23 接受会议邀请

截图里有图片

描述已自动生成

图24 接受成功

1. 项目总结

本次会议管理系统项目开发历时半个月，基本实现了预期的功能目标。系统采用SSM的技术架构，实现了用户管理、会议管理、参会管理等核心功能模块。通过JSP技术构建前端界面，实现了良好的用户交互体验。项目整体运行稳定，基本满足了用户的日常会议管理需求。

## 不足及需要改进的地方

在项目初始阶段，我进行了详细的项目分析：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图25 项目设计笔记

首先在功能方面我还有许多功能未实现：缺少会议提醒功能，无法实现自动提醒参会人员，会议室管理功能也不够完善，管理员及其相关功能未实现。会议的信息不够全面，比如缺少会议描述等。

其次在技术层面，前端采用的JSP技术相对陈旧，用户体验有待提升。

系统性能优化不足，在数据量大时可能存在响应延迟。

安全性方面的考虑也不够充分，如密码加密方式较为简单，权限控制粒度不够细致。

## 遇到的问题

由于对SSM框架的项目还不熟悉，导致在项目初始阶段的项目配置和环境配置就耗时好几天。

在项目开发过程中，不是在解决bug，就是在解决bug的路上，碰到的问题实在是太多太多，因此我将碰到的问题及解决过程进行了记录：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图26 问题笔记

之后将部分问题及解决办法分享到了CSDN：

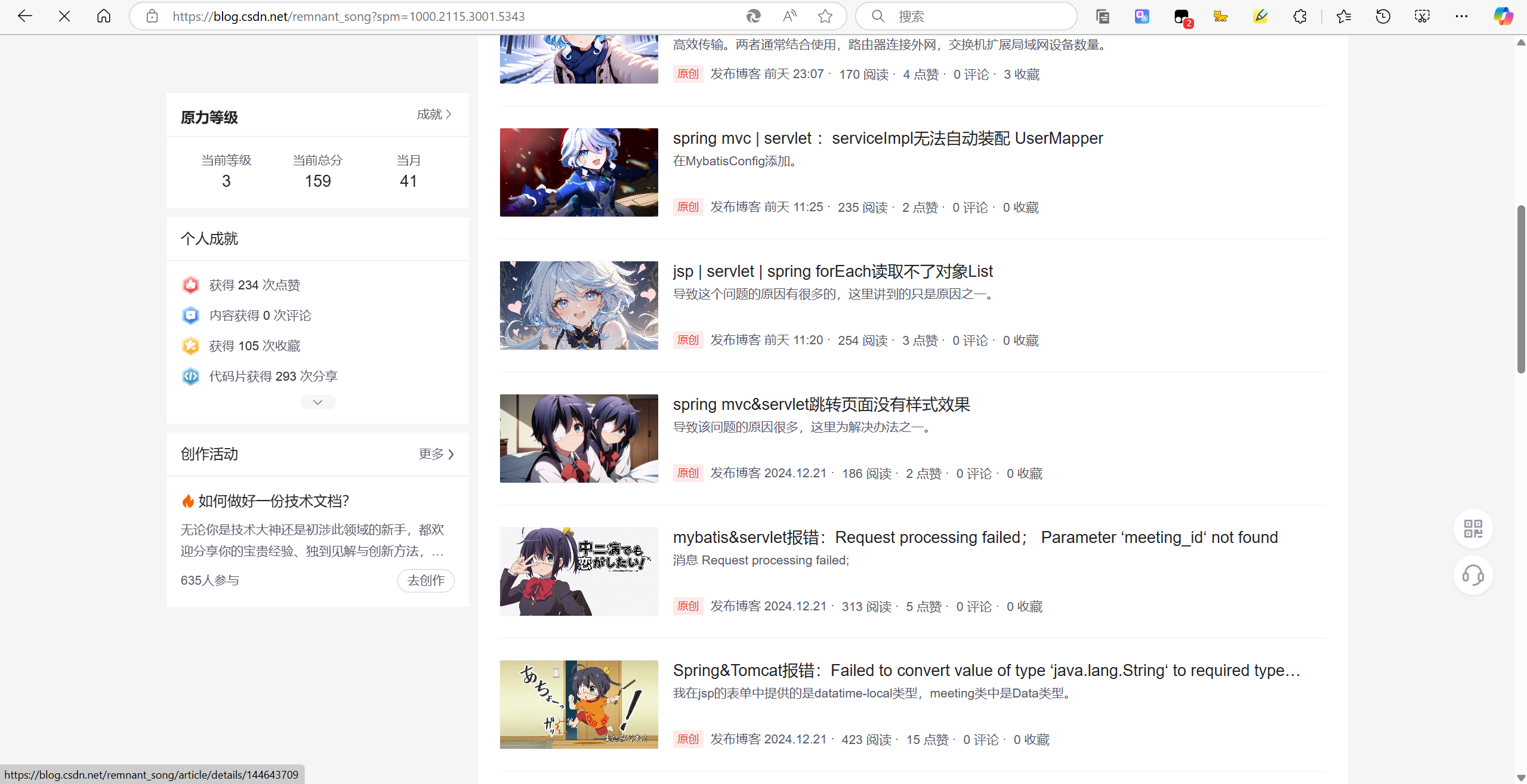


图27 个人CSDN

由于两周时间中碰到的问题太多了，无法一一列举，就只列举一部分：

（1）、Impl无法自动装配 UserMapper

MybatisConfig添加

@Configuration

@MapperScan("mapper")

（2）、jsp forEach读取不了对象List

添加<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

（3）、报错：Failed to convert value of type ‘java.lang.String‘ to required type ‘java.util.Date‘

在@InitBinder 中注册 DateFormatter，它将 yyyy-MM-dd'T'HH:mm 格式的字符串转换为 Date 对象。

## 收获的知识和经验

问题的解决过程不仅提升了技术能力，也积累了宝贵的项目经验。

通过本项目的开发，我在多个方面都有了显著提升。技术层面上，深入理解了Spring MVC的工作原理，掌握了MyBatis的多表查询和动态SQL技术，提高了数据库设计和优化能力。在项目管理方面，掌握了Maven项目构建和依赖管理。同时，问题解决能力、团队协作能力也得到了很大提升。最重要的是，通过这个项目，我对企业级应用开发有了更全面的认识，理解了从需求分析到系统实现的完整开发流程。

未来，我计划继续完善这个系统，添加会议提醒功能，优化会议室管理模块，同时考虑将前端框架升级为Vue.js，引入Redis缓存提升系统性能。这些改进将使系统更加完善，更好地满足用户需求。这次项目开发的经验和教训，将成为我继续进步的宝贵财富。