本科学生毕业论文（设计）

题目（中文）： **基于SSM框架的B/S微博系统的设计与实现**

（英文）： **Design and Implementation of B/S Microblog System Based on SSM Framework**

姓 名  **刘帅**

学 号  **201506030107**

院 （系）  **电子与信息工程学院**

专业、年级  **计科1501**

指导教师  **段华斌 (讲师)**

年月 日

目 录

[1绪论 6](#_Toc6337583)

[2相关技术介绍 7](#_Toc6337584)

[2.1 架构概述 7](#_Toc6337585)

[2.2 关键技术简介 8](#_Toc6337586)

[2.3 开发工具 9](#_Toc6337587)

[2.4 本章小结 9](#_Toc6337588)

[3 系统需求分析 10](#_Toc6337589)

[3.1 可行性分析 10](#_Toc6337590)

[3.1.1技术可行性 10](#_Toc6337591)

[3.1.2 经济可行性 11](#_Toc6337592)

[3.1.3 操作可行性 11](#_Toc6337593)

[3.2 需求分析 11](#_Toc6337594)

[3.2.1 系统总体需求 11](#_Toc6337595)

[3.2.2 用例图需求 12](#_Toc6337596)

[4 系统功能设计 16](#_Toc6337597)

[4.1 系统类分析 16](#_Toc6337598)

[4.1.1 实体类 16](#_Toc6337599)

[4.1.2 控制器类 17](#_Toc6337600)

[4.1.3 业务逻辑类 20](#_Toc6337601)

[4.1.4 数据库交互类 21](#_Toc6337602)

[4.2 关键业务设计 22](#_Toc6337603)

[4.2.1 登陆系统 22](#_Toc6337604)

[4.2.1 用户信息系统 23](#_Toc6337605)

[4.2.2微博模块 24](#_Toc6337606)

[4.2.3 好友模块 24](#_Toc6337607)

[4.2.4 树洞模块 25](#_Toc6337608)

[4.2.5 后台管理员模块 25](#_Toc6337609)

[4.3 数据库设计 26](#_Toc6337610)

[4.3.1 概述 27](#_Toc6337611)

[4.3.2 概念设计 27](#_Toc6337612)

[4.3.3 数据库表 29](#_Toc6337613)

[5系统功能实现 33](#_Toc6337614)

[5.1 登陆系统 33](#_Toc6337615)

[5.2 用户信息系统 35](#_Toc6337616)

[5.3 微博模块 38](#_Toc6337617)

[5.4 好友模块 42](#_Toc6337618)

[5.5 神奇树洞模块 48](#_Toc6337619)

[5.6 后台管理员模块 51](#_Toc6337620)

[6系统测试 61](#_Toc6337621)

[6.1 系统测试综述 61](#_Toc6337622)

[6.2 测试用例 61](#_Toc6337623)

[6.2.1用户注册模块 61](#_Toc6337624)

[6.2.2用户登陆模块 62](#_Toc6337625)

[6.2.3修改用户信息模块 63](#_Toc6337626)

[6.2.4微博模块 64](#_Toc6337627)

[6.2.5用户好友模块 65](#_Toc6337628)

[6.2.6用户神奇树洞模块 66](#_Toc6337629)

[6.2.7管理员模块 67](#_Toc6337630)

[6.3 测试分析 69](#_Toc6337631)

[7 总 结 70](#_Toc6337632)

[参考文献 71](#_Toc6337633)

[致 谢 72](#_Toc6337634)

**摘 要**

自上个世纪末起。互联网技术得到飞速发展，时至今日信息时代的到来，短短十几年，互联网就让我们见识到了它的魅力，在我们的生活中，网络无处不在，而信息的爆炸性发展也拓展了人们获取信息的方式。

微博作为一种信息交流和分享的平台，在这个信息时代显得特别重要，人们可以随意浏览和发布信息，获取最新最热的咨询，而它的及时性和高互动性也使得它深受人们的喜爱，随着微博在网民中的火热，网络红人，网络热词等微博效应逐渐形成，该微博系统采用B/S架构，实现微博原有基本功能的基础上，更是在以往微博系统中添加了问题悬赏，采纳积分等新功能，提高用户体验，在后台管理中实现对微博和用户的综合管理，在实现技术方面，页面使用JSP，javaScript,ajax等技术提高系统页面与用户之间的交互性，后台使用java语言，服务器采用Tomcat服务器，使用SSM框架实现具体功能，在数据库方面使用Oracle数据库保存数据。

关键词： 微博；B/S架构；java；SSM框架；Oracle;

**ABSTRACT**

Since the end of last century. The expansion of the Internet on a global scale, our country get the rapid development of Internet technology, today the arrival of information age, short 10 years, the Internet let us insight into its charm, the life of people is inseparable from the network, and the explosive development of information also expanded the way people access to information.

Weibo as a platform of information exchange and sharing, appear particularly important in the information age, people can browse through and release information, for the latest hottest consulting, and its timeliness and high interactivity also makes it popular people's favorite, as weibo in the Internet, web, network hot words such as weibo effect gradually formed, the microblog system adopts B/S architecture, implementation of weibo browse and release, the basic function such as comments and collection, on the basis of more added in the previous weibo system problems, reward for new functions such as user tags, improve user experience, in the background in the management of weibo and the user's comprehensive management, in terms of implementation technology, use the JSP page, javaScript, ajax technology to improve system such as the page and the interaction between the user, the background using the Java language, SSM framework to achieve specific functions, the middle layer using Tomcat server, use Oracle database persistent data,

**Keywords:** Weibo；B/S architecture；java；SSM framework；Oracle；

# 1绪论

时至今日，网络对于现代人来说，早已成为人类科技发展进步的桥梁，而通过网络衍生出的一系列产品也不断的冲击着人们的日常生活，截至2018年年底，我国网名数量达到10亿，有超多一半多人在使用网络，而它天涯咫尺的作用，不但消除了人与人地域上的距离，更是拉近了心灵的距离，沟通与互动变的异常频繁与重要。

随着互联网新时代的来临，微博借着互联网的桥梁，逐渐进入网名的视野中，没有博客的长篇大论，也不需要严谨的逻辑层次，这使得网络中一大批的原创文章被生产发掘，短短几句话便可以在网络中激起千层浪，普通人也可能在一夜之间成为拥有数千万粉丝的“网红”。微博的便捷、原创、和草根性使它成为中国网民茶余饭后的网上生活。

早在2006年3月，Obvious推出了Twitter服务,这个世界上最早同时也是最著名的微博系统，而在那是，微博也仅仅只是为了给好友发送手机短信，在中国，饭否网的上线标志中国微博的开端，之后腾讯滔滔、叽歪、嘀咕等微博的开荒者终究也都没能站在最后。

2010年，我国的微博得到迅速发展，这一年，无论微博的用户还是影响力都达到前所未有的高度，以新浪门户为首的腾讯、新浪、搜狐等微博展现出全新的活力。到了2013年6月，中国微博用户规模高达3.31亿人，在微博中流动的信息有超过2亿多条，微博于2014年4月17日正式美国上市，到现在为止，微博市值已经突破百亿美金。

2010年11月，新浪微博推出群组功能，这个功能不但可以与好友实时联系，又可以随时发布最新信息，2012年添加新功能“悄悄关注”，在用户关注后不通知被关注用户，也不显示给被关注用户，2013年，微博推出包括iPhone和Android的移动客户端，新增“密友”功能，同年10月份新浪微博“粉丝服务平台”上线，粉丝服务平台帮助认证用户为订阅自己的用户提供精彩内容和互动服务，自此微博由“营销”向“营销+服务”转型！

而新浪微博依旧延续这自己的名人效应，一个普通账号，在没有粉丝关注的情况下，发布的微博却很难被他人看到，如果需要在微博中求助，没有“名人大V”的帮助，很难被别人所注意到。想要在微博中寻求帮助，就需要微博提供更多的服务。而本系统通过使用积分悬赏功能使得用户可以通过积分悬赏自己的问题，来让更多的用户回答自己的问题，得到更多人的帮助。

# 2相关技术介绍

## 2.1 架构概述

B/S架构(Browser/Server，浏览器/服务器模式)：是一种通过将浏览器作为客户端的网络结构模式，利用已经逐步成熟的web浏览器技术，结合浏览器的多种功能，使用浏览器来作为早先C/S(Client/Serve)架构下复杂的客户端，使用C/S架构使得用户的客户端得到统一，将软件系统的核心功能集中在服务器端，系统的升级和维护更加简单，开发人员只需要管理服务器就可以做到对如软件系统的更新和维护，B/S架构所带来的众多优点使得它成为将来信息化发展的主流方向。

MVC模式：即模型(Model),视图(View),控制器(Controller)是一种软件开发的设计模式，Model主要负责分离出来的业务逻辑的数据交互，View主要负责数据的提交和展示，Controller主要负责获取请求和请求的转发。

SSM框架(SpringMVC+Spring+Mybatis)：由这三个开源框架组成的web项目的框架，是标准的MVC模式，spring MVC负责请求的转发和视图管理，spring实现业务对象管理和事务控制，mybatis作为持久化引擎，相对于早先流行的SSH(Struts+Spring+Hibernate)框架来说，SSM作为轻量级框架更适合开发中小型项目。

## 2.2 关键技术简介

前端技术：

1.JSP(Java Server Pages)：嵌入了java代码的html页面，本质是一个servlet,它实现了在htmld的语法中扩展java代码，使用<% %>格式。

2.JavaScript：是一种弱类型的脚本语言，由ECMAScript描述JavaScript的基本对象和语法，文档对象模型(DOM)，浏览器对象模型(BOM)三部分组成。

3.Ajax(Asynchronous Javascript And XML)：异步的JavaScript和XML，实现前后台之间的异步交互，在不需要重新加载整个页面的前提下对页面上的部分内容做局部刷新。

后台技术：

1.java：一种面向对象的编程语言，主要特性包括，(1)简单性：抛弃了C++中复杂的语法和指针多继承等特征，开发人员不需要关注底层优化，只需要关注业务逻辑就行，(2)面向对象性，对程序员而言，只需要注意对应的数据和处理数据的方法，不用严格按照过程的方式来编写代码，因为java语言是面对对象的，因此也拥有面向对象的三大特征，继承、封装和多态，(3)跨平台性，java语言的跨平台性也就是JVM(java虚拟机)的跨平台性。Java文件编译后不会直接生成机器直接运行的二进制文件，而是编译成class文件，JVM会将编译的class文件根据不同操作系统下的JVM来生成对应系统的二进制文件，只需要编译一次，就可以在不同的平台上运行。

2.SpringMVC：是Spring框架提供的一个模块,通过实现MVC模式来地将数据、业务与展示进行分离，简化了Web开发。和其他MVC框架一样都是请求驱动,控制转发，它的核心Servlet是DispatcherServlet,即前端控制器，用来分发请求到控制器，它和Spring框架完全整合。这也使得SpringMVC成为目前非常流行的MVC框架。

3.Spring：一款轻量级java开发框架，Spring框架有着分层的体系结构，因此可以使用Spring中 独立的任意部分。而它的架构依旧是内在稳定的，Spring提供了J2EE应用各层解决方案，贯穿于三层架构的每一层，但Spring依旧可以很好的兼容其他框架，本项目主要用到了Spring中IOC(控制反转)和AOP(面向切面编程)模块。

4.Mybatis：是一个简化Java持久化层的轻量级开源框架。并且支持注解的Mapper，Mybatis消除了绝大部分的JDBC代码，使得java对象持久化到数据库的过程变的更加容易，相对于其他的java持久化框架，Mybatis的优点在于，消除了大量的JDBC冗余代码、简单易学、可见的Sql语句、提供了于Spring的整合，引入更好的性能。

## 2.3 开发工具

Tomcat服务器：是一个Web应用服务器，它是轻量且开源的，是中小型Web项目和开发调试和学习的首选。

Oracle数据库(Oracle Database)：是由Oracle公司开发的一款关系型数据库，是商业运用第一的关系型数据库，系统使用方便，功能强大，可移植性强。适用于各种大中小环境，在大数据时代，作为分布式数据库，它实现了分布式处理的功能，具有完整的数据管理功能、完备的关系型数据库、分布式处理功能。

Eclipse开发工具：一个基于java开源的可扩展开发平台，它不但包括java集成开发环境，还包括插件开发环境，如SVN、CVS等团队开发插件。

## 2.4 本章小结

本章主要介绍了开发项目用到的一些主要技术，项目所使用的架构和设计模式，项目中所使用到的主要框架技术，项目在浏览器端展示用到的前端技术和展示方式，后台代码使用的开发语言，使用的服务器技术，数据持久层所使用的数据库等，在本章最后又介绍了开发使用到的开发工具。

# 3 系统需求分析

## 3.1 可行性分析

### 3.1.1技术可行性

在已有技术方面，为了统一客户端，消除因版本升级和维护带来的复杂性，因此采用成熟的B/S架构在项目的实现上完全可行，在开发语言和框架方面，java和j2ee体系的强大，可以让开发人员精心的构建web项目，以及一系列的开源框架，都为项目的可行性提供了强大的依据，在服务器方面，使用开源服务器Tomcat,足以支持该小型项目的正常使用，而不断发展的前端技术和前端框架可以制作精美的前台页面，提高用户的体验和交互，这在项目的页面展示技术上完全可行，强大的关系型数据库为项目数据的持久化提供强有力的后援。综上所述，日趋成熟化的java和j2ee体系、完全开源的java框架和服务器、功能强大的关系型数据库、运用Web前端技术提供用户交互页面，所以该项目在技术方面完全可行。

### 3.1.2 经济可行性

在互联网发展到信息时代的今天，单一获取信息的方式已经不能满足人们，人们每天接触的信息越来越多，而获取信息的形式也越来越多，但大多数获取信息的方式，留给用户交互的方式并不多，大都数情况下，人们只能被动获取，而很难找到自己的喜好和需求来获取信息，而本微博项目可以让人们获取实时热点信息和他们所关注的信息，实时与微博信息交互。而使用大量成熟技术与开源框架下，也使得小项目的开发更加简单，经济，高效，因微博而兴起的微博效应也能带来一定的经济效益。

### 3.1.3 操作可行性

微博系统使用B/S架构，用户不需要下载客户端，只需要用户有浏览器，就可以在浏览器上登陆微博系统，微博系统的界面颜值高，用户交互性高，用户操作简单方便，只需要了解基本的计算机操作就能使用，用户体验性高。因此在系统操作上完全可行。

## 3.2 需求分析

### 3.2.1 系统总体需求

该微博系统主要由前台用户模块和后台管理模块组成，当用户进入首页时可以选择登陆或不登陆，登陆时可以使用已有账号登陆或注册新账号。用户未登陆时，在首页显示最近更新的热门微博，而登陆后的用户可以在首页看到 自己关注用户最近更新的微博。

未登陆用户只能搜索查看微博信息和访问用户主页，登陆用户可以登陆系统后修改自己的基本信息例如签名、性别等，在验证用户信息后还可以修改密码和密保信息。以及修改用户头像和密码，编辑自己的个人主页，对微博进行点赞、评论、收藏等功能，还可以关注/取消关注用户，拉黑用户、私聊用户等操作。

后台管理员可以查看系统所有的数据，包括用户、微博、评论、树洞、回答的总量，最近一个的数据库，最近一周的数据量。具体所具有的功能包括管理不良微博信息与不良账号，对微博、微博评论、树洞、回答等信息的删除和恢复功能，对不良账号的封禁等操作。

微博查找模块：用户可以输入关键字来查找相对应的微博或查找用户。

微博发布模块：用户点击发布，在内容中添加自己想要发送的内容，可以选择表情，也可以插入图片，但对输入字符数有着限制，同时还可以插入一张图片。

微博评论模块，用户可以查看微博的评论，发布微博评论等。

神奇树洞模块：用户可以发布一个神奇树洞，树洞主要用来记载用户提出的各种问题，由其他用户来查看并回答问题，当回复者的答案被提问者采纳后，回答者可以增加自己的树洞积分，不同的树洞积分有着不同的树洞称号。

积分模块：用户每天登陆，发微博，做任务等方式可以增加自己的积分，不同的积分拥有不同的称号，神奇树洞模块的积分有着不同的称号，称号显示在用户名称的后面。

好友模块：用户可以查看系统中其他的用户，找到自己喜爱的用户然后关注他们，关注后可以在好友模块中查看自己关注的用户，以及好友最近发布的微博等信息，也可以私信好友，发送私信信息给好友。

### 3.2.2 用例图需求

1.当用户进入系统时候，可以选择登陆或注册用户，如果忘记了密码还可以通过密保问题来重置密码。

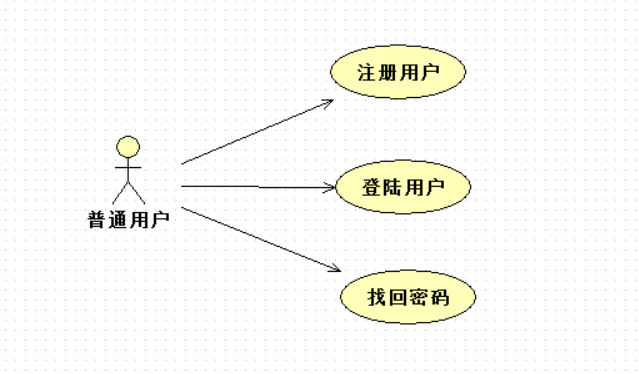


图3.1 用户初始用例图

2.当用户登陆后，可以管理用户的个人基本信息，修改用户基础信息，修改用户密保信息。修改用户头像等功能。

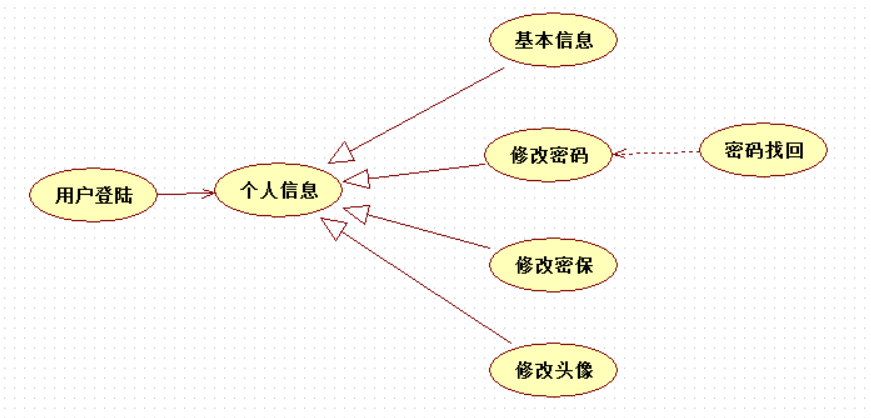


图3.2 用户信息管理用例图

1. 用户微博管理系统，当用户登陆进去系统时，可以在首页发布微博，通过关键字搜索微博内容中关键字的微博信息。产看微博，包括查看个人微博、好友微博、推荐微博。

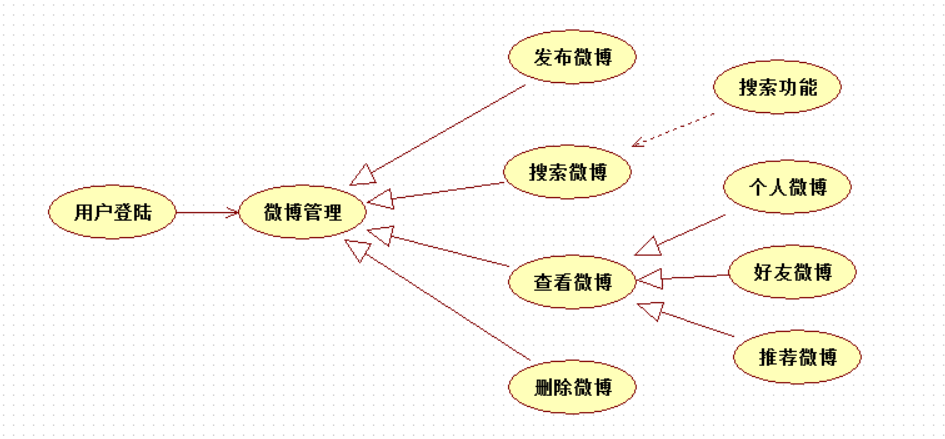


图3.3 微博管理用例图

1. 评论管理，评论管理依赖于微博模块，用户可以查看微博的评论，对微博信息发布评论，以及删除自己的评论。

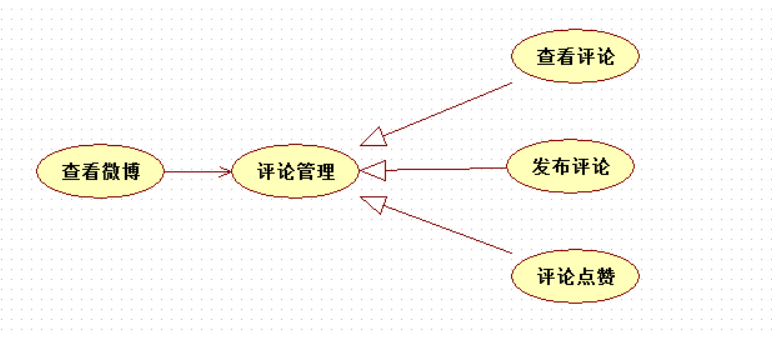


图3.4 微博评论管理用例图

1. 树洞管理，用户登陆后可以在树洞模块发布树洞问题，发布问题时可以选择悬赏的积分数目，同时减少自己的积分数，用户可以参加回答他人的树洞问题，当回答的答案被采纳时，就可以获得用户悬赏的积分数。

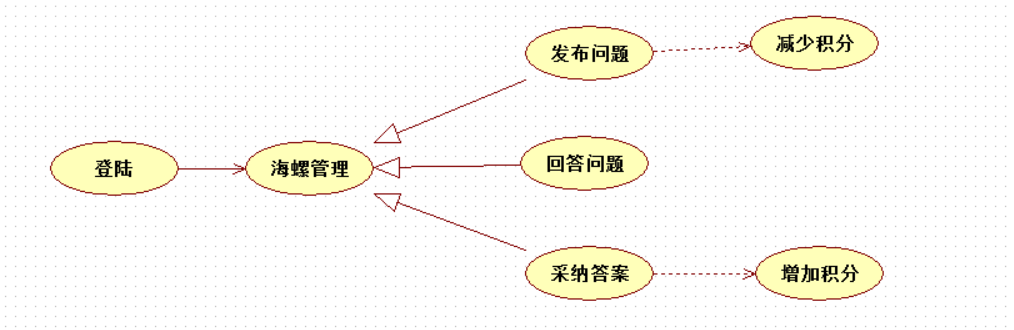


图3.5 树洞管理用例图

1. 好友管理，当用户登陆时可以关注系统推荐好友，也可以自己搜索用户，查看用户的主页面，添加关注或取消关注用户，还可以给用户发送私人信息，或者拉黑用户。

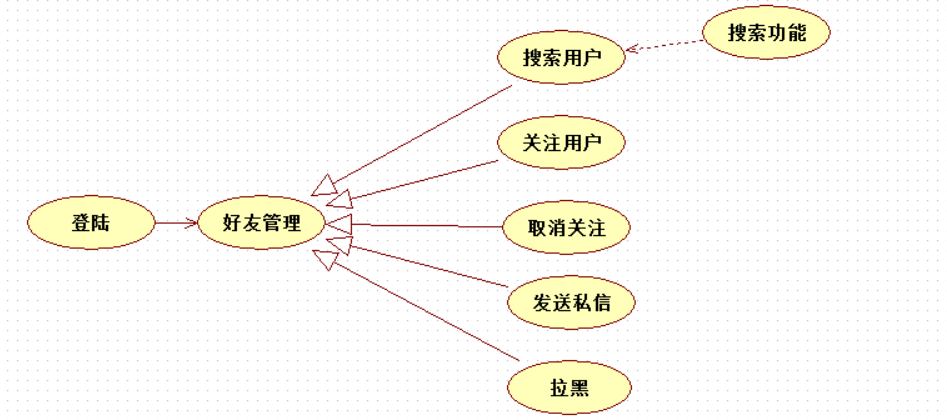


图3.6 好友管理用例图

7.消息通知，当用户的关注，微博评论，点赞，收藏时调用消息通知。

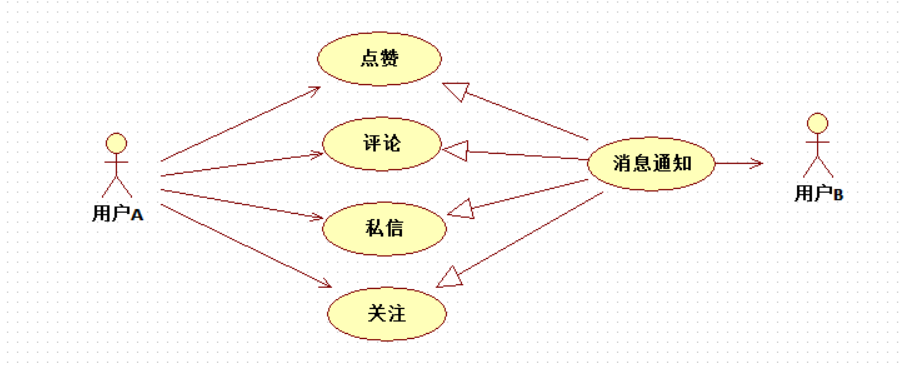


图3.7 好友管理用例图

8.管理员：管理员登陆系统，可以管理微博用户，对不良用户进行封禁和注销账户的操作，也可以恢复用户的状态，同时也可以对微博信息进行管理，如删除微博，恢复被管理员删除的微博信息等，对微博评论的删除和恢复等操作，如图3.2。

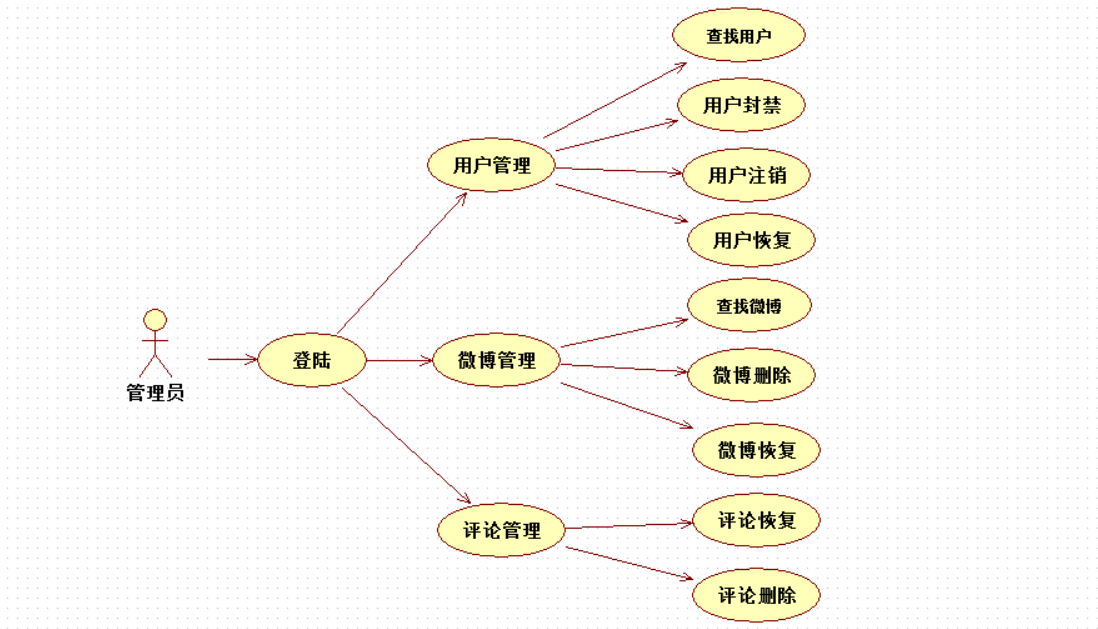


图3.8 管理员用例图

# 4 系统功能设计

## 4.1 系统类分析

4.1.1 实体类

实体类主要用来传递数据，主要包括User、Friend、PointAction、PointRecord、Weibo、WeiboCollect、WeiboComm、WeiboLike、WeiboTrans等，用户类中包括用户的基本信息，微博类中包括微博的基本信息同时包含实体用户类，好友类包括好友编号，好友创建时间与关注双方的用户类等，私信类包含私信双方的用户类与私信的基本属性，微博收藏类包含收藏的微博类与用户id等收藏属性，微博评论类包含微博类和用户类以及评论内容等属性，微博转发类包含用户类和微博类以及转发时间等属性，微博点赞类包含微博类和用户类以及点赞时间等属性。实体类之间的类关联关系如图4.1所示。

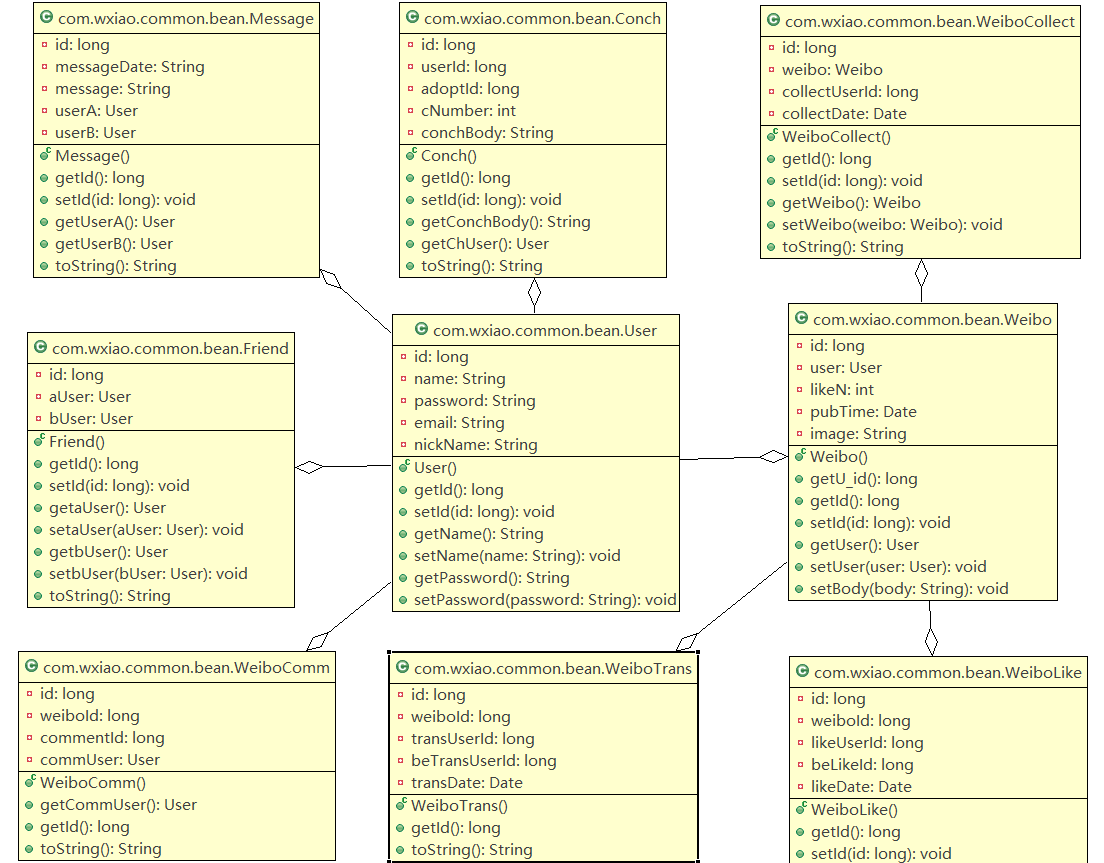


图4.1 Bean层类图

4.1.2 控制器类

在controller层包含MainController、WeiboController、FrendController、AdminController四个JAVA类，在SpringMVC框架中主要用来接收浏览发送给服务器的请求和数据处理并控制请求的转发，将从Service层中获取的数据响应给浏览器端。MainController主要用发来接收来自用户相关页面中提交的表单或链接请求，并将请求的参数传递到Controller中对应的方法中，携带数据响应给浏览器，在浏览器端显示数据，具体属性和方法如图4.2所示。

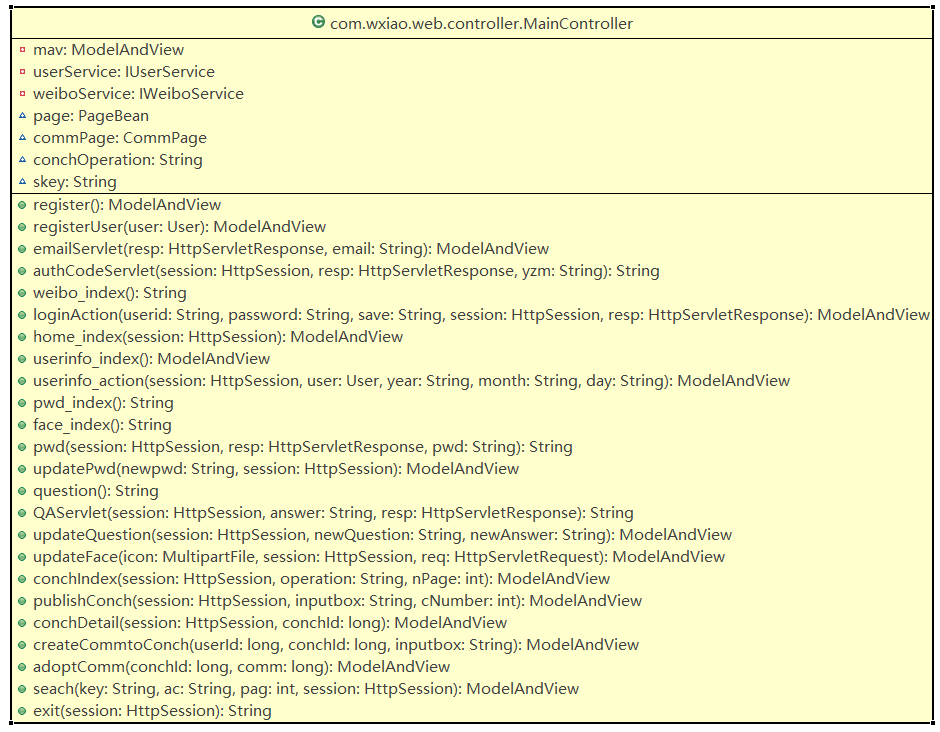


图4.2 用户控制器类图

WeiboController主要用发来接收来自微博相关页面中提交的表单或链接请求，并将请求的参数传递到Controller中对应的方法中，并获取到业务处理层中返回的数据，携带数据响应给浏览器，在浏览器端显示数据，具体属性和方法如图4.3所示。



图4.3 微博控制器类图

FriendController主要用发来接收来自好友相关页面中提交的表单或链接请求，并将请求的参数传递到Controller中对应的方法中，并获取到业务处理层层中返回的数据，携带数据响应给浏览器，在浏览器端显示数据，具体属性和方法如图4.4所示。

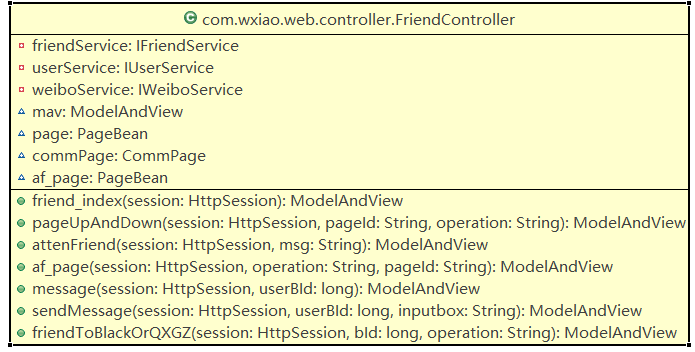


图4.4 好友控制器类图

AdminController主要用发来接收来自管理员相关页面中提交的表单或链接请求，并将请求的参数传递到Controller中对应的方法中，并获取到业务处理层层中返回的数据，携带数据响应给浏览器，在浏览器端显示数据，具体属性和方法如图4.5所示。



图4.5管理员控制器类图

4.1.3 业务逻辑类

在Service层中主要包含四个Service接口和他们的实现类，包括IUserService用来处理用户业务例如用户注册、登陆、修改个人信息等，如图4.6所示。IWeiboService用来处理微博相关的业务例如查找微博，发布微博，删除微博以及对微博的相关操作例如点赞、评论等功能的业务实现，如图4.7所示。IFriendService用来处理和用户好友相关的业务例如点赞、取消点赞、私信、拉黑等功能的业务实现，如图4.8所示。IAdmoinService用来处理和管理员相关的业务例如管理员登陆、图表展示、用户管理、微博管理、树洞管理功能的业务实现，如图4.8所示。

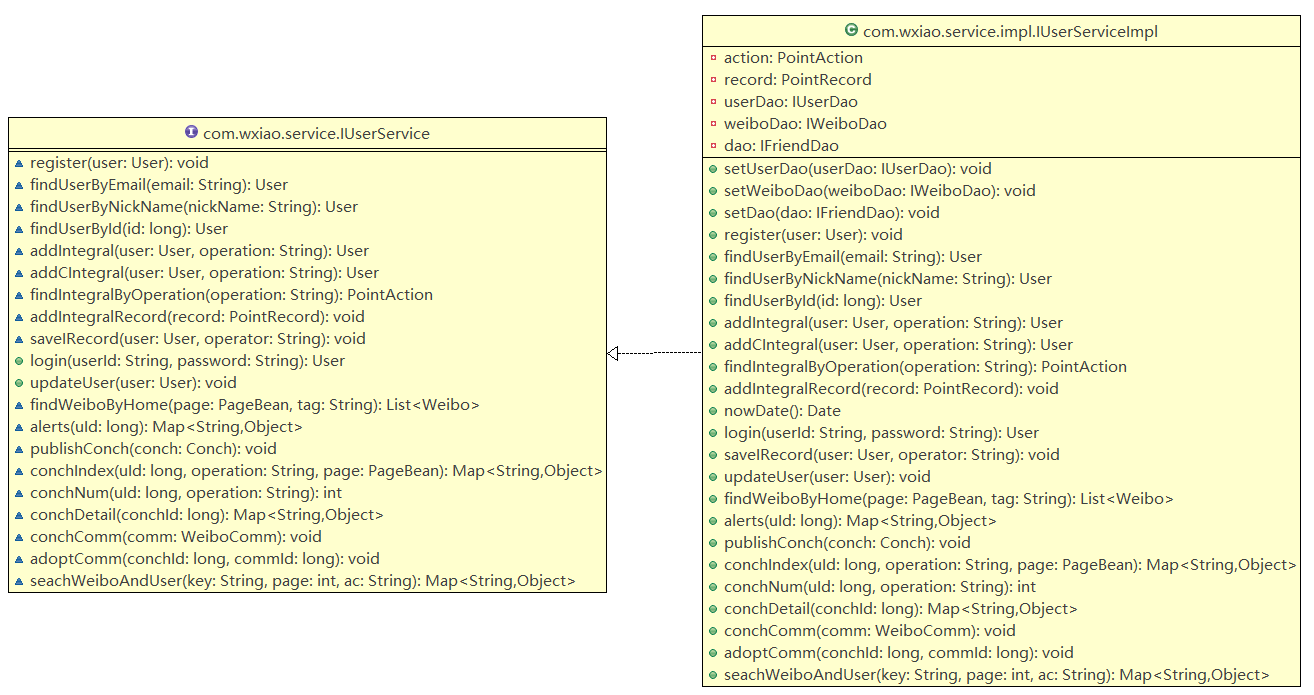


图4.6用户业务处理类图

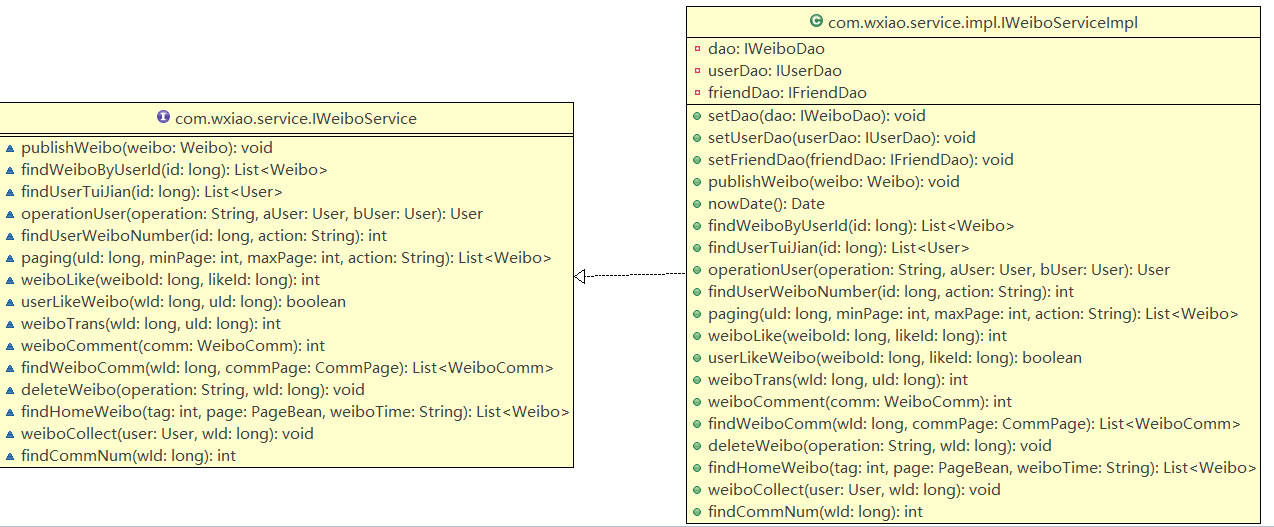


图4.7微博业务处理类图

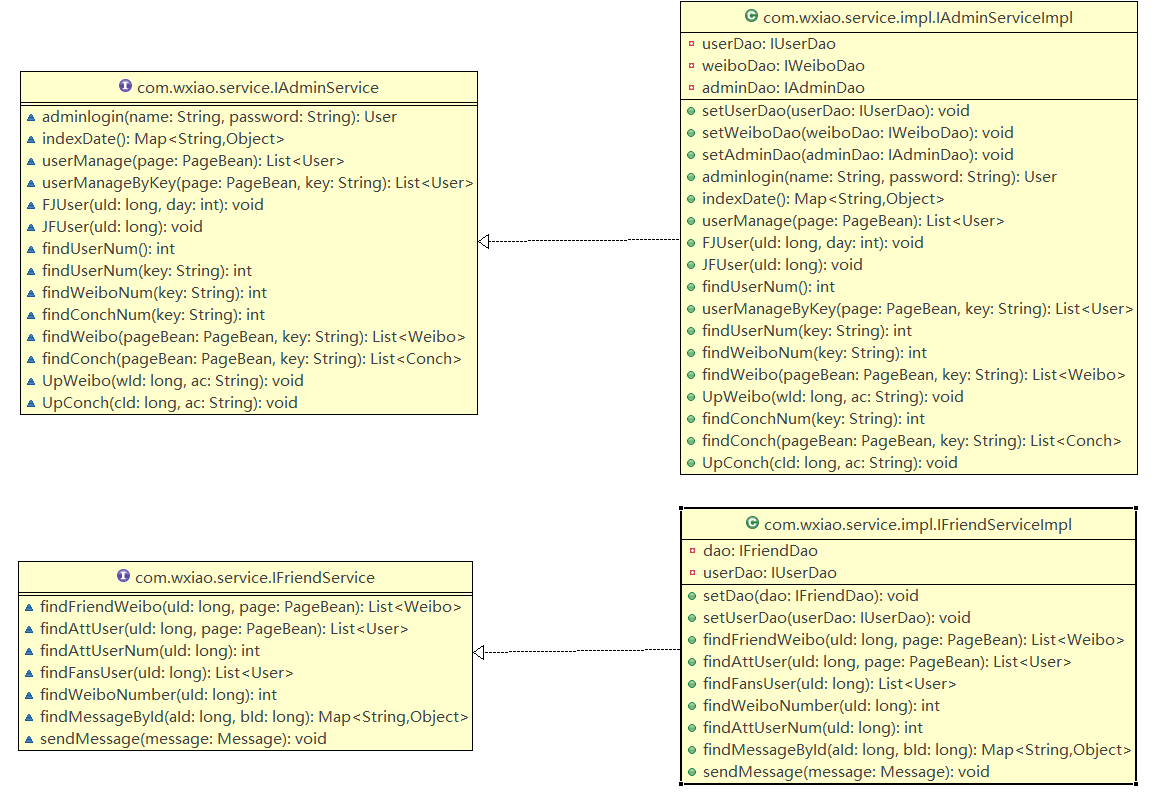


图4.8微博控制器类图

4.1.4 数据库交互类

由于系统采用了MyBatis持久化框架，开发人员不需要关注和数据库之间具体的JDBC代码，而只需要处理业务逻辑，因此只需要在Dao层接口中声明方法而不需要写接口的实现类来实现方法，则是通过配置对应的配置文件，在配置文件中编写对应接口方法中的SQL语句和数据库交互。

## 4.2 关键业务设计

### 4.2.1 登陆系统

CONTROLLER

SERVICE

DAO

DATABASE

用户

登陆

Add

用户输入

成功

成功

成功

成功

成功

Insert

Select

注册

Find

成功

图4.9系统登录顺序图

用户进入主页后，可以在左边选择注册用户，或者在右边登陆系统，在注册模块，用户输入用户邮箱，密码，昵称等信息，当用户输入邮箱后会通过Ajax将用户输入的邮箱传到后台，查找用户邮箱是否已经被注册，如果被注册则在页面提示用户该邮箱已被注册，在输入基本信息点击注册后，保存用户注册信息。

登陆功能，当用户输入邮箱地址和密码后，如果点击下次自动登陆密码在点击登陆后，判断用户的当前登陆时间和上次登陆时间是否为同一天，如果不是同一天则为用户增加登陆的 积分，然后跳转至微博首页。

### 4.2.1 用户信息系统

CONTROLLER

SERVICE

DAO

DATABASE

用户

查看信息

Update

用户输入

成功

修改成功

成功

成功

成功

Update

Select

修改信息

Find

成功

成功

图4.10用户修改信息顺序图

用户基本信息：在系统的首页中点击个人账号设置后，跳转到修改用户基本信息页面，用户可以修改这些基本信息，并将修改后的信息保存在数据库中。

修改密码：在用户个人资料页面点击修改密码，跳转修改密码页面，用户可以输入用户的当前密码，系统判断密码是否正确，如果密码不正确，显示当面密码错误，如果输入密码正确，用户则可以输入新密码，点击修改后将修改后的密码更新到数据库中。

修改用户头像：点击用户个人资料中修改头像，跳转至修改头像页面，用户选择头像文件，点击上传，将用户头像保存在服务器上，判断用户之前头像是为系统默认头像，如果不是就删除用户之前的头像图片，点击修改后将修改后的头像地址保存在数据库中。

修改密保：用户先要根据之前设置的密保问题来填写答案，如果密保答案错误，提示用户密保答案错误，如果密保答案正确，用户可以输入新的密保问题和密保答案，点击修改保存用户新的密保问题和答案。

### 4.2.2微博模块

CONTROLLER

SERVICE

DAO

DATABASE

用户

查看微博

Operation

发送请求

返回响应

成功

成功

成功

成功

Imsert

Select

微博操作

成功

成功

Update

Delete

Find

成功

成功

图4.11系统登录顺序图

发布微博：用户可以在首页发布微博，在微博信息中可以插入表情，也可以选择插入一张图片，当用户点击发布后，浏览器发送请求将前台页面表单中微博信息和图片信息传入后台控制器，将图片信息保存在服务器中，在数据库中只保存图片路径，最后将微博信息保存在数据库中。

搜索微博：用户可以在首页的搜索框中输入微博中提到的内容来搜索，系统在数据库中通过迷糊查询查询相关的微博信息。在页面中将搜索到的页面展示出来，并且将关键字标红显示。

微博操作；用户可以查看个人微博、好友微博、收藏微博等信息，对微博的操作有点赞，转发，收藏，评论，删除等。

### 4.2.3 好友模块

关注功能：当用户登陆系统后可以在页面右侧的推荐用户中选择需要关注的用户，或者是通过页面中的搜索功能来搜索用户，对搜索到的用户进行关注。关注用户后个人关注+1，同时被关注用户粉丝数+1。

取消关注：和关注功能类型，在导航栏中点击我的好友，在我的好友首页中查看我的关注好友，就能查看到我所关注的所有好友和好友数以及他们的个人信息，通过点击对应的取消关注就可以取消关注该用户，取消关注用户后个人关注数-1，同时被关注用户粉丝数-1。

拉黑用户：即修改好友表中对应的记录状态，被拉黑用户所发布的私信和微博信息不会被拉黑用户所看到。

私信：可以在页面的推荐用户上面查看当前登陆用户的未查看私信数，未读私信的数目通过数据库中对所有接收者为当前登陆用户的所有私信信息，且信息状态为未阅读的私信，将得到的数值展示在前台页面中显示。

发送私信：用户先选择要发送的用户，输入需要发送的私信信息，可以在私信中插入表情，点击发送后会将私信信息发送给对应用户。同时增加提示该用户的未读私信数。

搜索用户：在搜索用户页面中，用户可以输入用户昵称的关键字来模糊查询相关用户，并将查询到的用户展示在页面中，同时将用户输入的关键字标红显示。

用户主页面：在页面中，点击任意一个用户的名称或头像都会跳转至对应用户的个人主页，用户的个人主页显示用户的个人信息和用户最近发布的微博，按照时间倒序排列，用户也可以对微博的点赞、转发、评论、收藏做操作。

### 4.2.4 树洞模块

发布：用户通过点击导航栏中的神奇树洞进入树洞主页，在树洞页面的上面是发布树洞的问题框，下面的可以插入的表情按钮和问题的悬赏积分数，中间的导航栏可以选择查看最新发布、已解决、待解决、我的树洞不同的筛选条件。

查看树洞问题：用户点击任意树洞问题，系统跳转到展示树洞的具体信息的页面，上方展示发布用户的用户名、用户称号、发布日期、树洞问题、悬赏积分、问题的状态等，在中间显示问题的所有回复信息，回复人的用户名、称号、回答内容、回复日期，是否被采纳等。

回答树洞问题：在树洞问题详细详细信息页面的底部可以回答树洞问题，在输入框中输入回复的答案，插入表情信息等，点击回复后将回复信息保存在数据库中。

采纳问题答案：在树洞问题首页，用户通过点击我的树洞问题可以跳转至用户自己所发布的树洞问题页面，在自己所发布的树洞问题页面中可以选择自己认为最正确的答案，点击采纳后修改树洞状态，被采纳的答案变为采纳答案，增加采纳者的树洞积分。

4.2.5 后台管理员模块

管理员登陆：管理员可以通过在登陆页面中点击管理员登陆，跳转至管理员登陆页面。当管理员输入账号信息和密码信息后，如果用户名和密码正确就跳转到管理员首页。如果错误则给出提示。

管理员首页：在管理员首页中上方显示导航栏，在页面内容方面，通过四张图表来显示微博、用户、评、树洞、评论、回答等的总数以及当月数，在下方的柱状图中则显示距离今天最近的7天的数目。

用户管理首页：在页面上方显示导航栏，页面内容中显示所有用户的编号、邮箱、昵称、注册日期、上次登陆日期以及用户的状态，由于考虑到用户数量多以以及为了方便查找，因此在页面中一次显示10条用户数据，同时添加用户搜索功能，用户能够在输入用户昵称的关键字后来搜索用户，并且支持迷糊查询。 搜索用户:在用户管理页面输入用户昵称中的关键字来搜索用户，系统会将获取到的用户信息中昵 称为输入关键字的那部分显示为红色。

用户封禁：管理员可以通过点击用户管理页面操作一栏中的封禁按钮来对违规用户的封禁，管理员可以输入封禁的天数，那么在这个日期之前，用户是不能正常登陆的，管理员也可以通过点击解封来提前解除封禁用户的操作。

微博管理首页：管理员点击导航栏中的微博管理，系统会跳转至用户微博管理页面，显示系统中所用是微博信息，管理员也可以通过输入微博内容中的关键字来搜索微博，同时可以删除有不良信息的微博，或是恢复以被删除的微博。

微博搜索：管理员可以通过输入微博内容中的关键字来搜索在微博中存在该关键字的微博，搜索到的微博内容中的关键字会使用红色标注出来。

微博删除：在微博管理页面中，管理员可以根据微博的内容来判断微博是否违法等信息，如果微博信息中包含不良信息，管理员可以通过操作栏中的删除按钮来删除微博或者可以对已经删除的为微博做恢复操作。

树洞管理首页：管理员点击导航栏中的树洞管理可以跳转至树洞管理页面，在页面的上方为导航栏，页面内容则显示所有的树洞信息、树洞搜索框、以及下方的分页框，树洞信息包括编号、发布人名称、树洞的内容、发布日期、树洞状态以及可以执行的操作。

树洞搜索：在树洞管理首页中，考虑到树洞数量多不好查找的问题，因此设置了搜索功能，管理员可以能够在树洞搜索框中输入树洞内容中的关键字来搜索树洞问题，在展示搜索到的结果时会将搜索关键字使用红色标注出来，方便查看。

4.3 数据库设计

4.3.1 概述

微博系统数据库表主要包括：

1. 用户表：用来保存用户的个人信息，例如编号、昵称、姓名、密码。邮箱等信息，以用户编号作为主键，如表4-1。
2. 微博关注表：用来保存用户的关注用户信息，以关注ID作为主键，如表4-2所示。
3. 微博表：用来保存微博信息，例如微博发布时间，微博内容，微博点赞，转发，收藏数等，以微博编号作为主键，如表4-3所示。
4. 微博收藏表：用来保存用户收藏的微博信息，如微博编号，收藏时间等，以收藏编号作为主键，如表4-4所示。
5. 评论表：保存微博的评论消息，例如评论人编号、评论日期、评论内容等，以评论编号作为主键，如表4-5所示。
6. 积分表：用来保存用户获取积分的方式，例如通过每天的登陆发布微博等获取积分，以积分编号作为主键， 如表4-6所示。
7. 点赞表：用来保存微博的点赞信息，主要包括点赞人编号，点赞编号，点赞微博编号等信息。以点赞编号作为主键，如表4-7所示。
8. 积分流水表：用来保存用户获取积分的信息，主要包括积分编号、用户编号、获取日期等信息，以积分编号作为主键，如表4-8所示。
9. 转发表：用来保存用户转发的信息，主要有转发编号、转发人、微博编号等信息，以转发编号作为主键，如表4-9所示。

4.3.2 概念设计

一个用户可以发布多条微博，因此用户表和微博表之间存在一对多的关系如图4.12所示。

用户表

微博表

1

n

图4.12 用户表和微博表对应关系

1. 微博可以对应多个点赞、转发、收藏和评论，因此微博表和收藏、点赞、转发、收藏表之间存在一对多的对应关系，如图4.13所示。

微博表

收藏表

转发表

点赞表

评论表

图4.13 微博表和微博操作表对应关系

一个用户可以发布多个树洞，每个树洞问题可以对应多条评论。如图4.14所示。

用户表

树洞表

回复表

图4.14 用户表和树洞表对应关系

4.3.3 数据库表

1. 用户表：数据库表名USER\_TAB，引用序列名SEQ\_USER。

表4-1用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| U\_ID | NUMBER |  | 主键 | 用户编号 |
| U\_NICKNAME | VARCHAR2(20) | 20 |  | 用户昵称 |
| U\_NAME | VARCHAR2(20) | 20 |  | 真实姓名 |
| U\_PWD | VARCHAR2(20) | 20 |  | 密码 |
| U\_EMAIL | VARCHAR2(200) | 200 |  | 邮箱 |
| U\_SEX | VARCHAR2(8) | 8 |  | 性别 |
| U\_POSITION | VARCHAR2(50) | 50 |  | 住址 |
| U\_IMAGES | VARCHAR2(200) | 200 |  | 头像 |
| U\_SIGN | VARCHAR2(200) | 200 |  | 签名 |
| REGISTERTIME | DATE |  |  | 注册时间 |
| PWD\_QUESTION | VARCHAR2(100) | 100 |  | 密保问题 |
| PWD\_ANSWER | VARCHAR2(100) | 100 |  | 密保答案 |
| INTEGRAL | NUMBER |  |  | 用户积分 |
| CONCH\_INTEGRAL | NUMBER |  |  | 树洞积分 |
| A\_NUMBER | NUMBER |  |  | 关注数 |
| F\_NUMBER | NUMBER |  |  | 粉丝数 |
| CO\_NUMBER | NUMBER |  |  | 收藏数 |
| BLOG\_NUMBER | NUMBER |  |  | 微博数 |
| LASTLOGIN | DATE |  |  | 上次登陆时间 |
| U\_BIRTHDAY | VARCHAR2(50) | 50 |  | 生日 |
| U\_STATE | NUMBER |  |  | 用户状态 |

1. 关注表：数据库表FRIEND\_TAB，引用序列名SEQ\_FRIEND。

表4-2关注表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| F\_ID | NUMBER |  | 主键 | 编号 |
| U\_ID | NUMBER |  |  | 关注人编号 |
| F\_ATTEN | NUMBER |  |  | 被关注人编号 |

1. 微博表：数据库表名WEIBO\_TAB，引用索引名SEQ\_WEIBO。

表4-3微博表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| M\_ID | NUMBER |  | 主键 | 编号 |
| U\_ID | NUMBER |  |  | 发布人编号 |
| M\_TRANUM | NUMBER |  |  | 转发数 |
| M\_COMNUM | NUMBER |  |  | 评论数 |
| M\_COLNUM | NUMBER |  |  | 收藏数 |
| M\_LIKE | NUMBER |  |  | 点赞数 |
| M\_DATE | DATE |  |  | 发布日期 |
| M\_BODY | VARCHAR2(500) | 500 |  | 微博正文 |
| M\_IMAGE | VARCHAR2(200) | 200 |  | 微博图片 |
| M\_STATE | NUMBER |  |  | 微博状态 |
| M\_TAG | NUMBER |  |  | 微博标签 |
| M\_T\_STATE | NUMBER |  |  | 转发状态 |

1. 收藏表：数据库表名COLLECT\_TAB，引用索引名SEQ\_COLLECT。

表4-4收藏表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| C\_ID | NUMBER |  | 主键 | 编号 |
| M\_ID | NUMBER |  |  | 微博编号 |
| COLLECT\_USER\_ID | NUMBER |  |  | 收藏用户编号 |
| BE\_COLLECT\_ID | NUMBER |  |  | 微博发布人编号 |
| COLLECT\_DATE | DATE |  |  | 收藏日期 |
| COLLECT\_STATE | NUMBER |  |  | 收藏状态 |

1. 评论表：数据库表名COMM\_TAB，引用索引名SEQ\_COMM。

表4-5评论表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| C\_ID | NUMBER |  | 主键 | 编号 |
| M\_ID | NUMBER |  |  | 微博编号 |
| COMMENT\_ID | NUMBER |  |  | 评论人编号 |
| BE\_COMMENT\_ID | NUMBER |  |  | 微博发布人编号 |
| C\_DATE | DATE |  |  | 评论日期 |
| C\_BODY | VARCHAR2(500) | 500 |  | 评论内容 |
| C\_LIKE | NUMBER |  |  | 评论点赞数 |
| C\_STATE | NUMBER |  |  | 评论状态 |

1. 积分表：数据库表名INTEGRAL\_TAB，引用索引名SEQ\_INTEGRAL。

表4-6评论表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| I\_ID | NUMBER |  | 主键 | 编号 |
| M\_I\_OPERATION | VARCHAR2(50) | 50 |  | 获取积分方式 |
| I\_NUMBER | VARCHAR2(50) | 50 |  | 积分数 |
| I\_DESC | VARCHAR2(100) | 100 |  | 描述 |

1. 点赞表：数据库表名LIKE\_TAB，引用索引名SEQ\_LIKE。

表4-7点赞表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| LIKE\_ID | NUMBER |  | 主键 | 编号 |
| WEIBO\_ID | NUMBER |  |  | 微博编号 |
| LIKE\_USER\_ID | NUMBER |  |  | 点赞用户编号 |
| BE\_LIKE\_ID | NUMBER |  |  | 微博发布人编号 |
| LIKE\_DATE | DATE |  |  | 点赞日期 |

1. 积分流水表：数据库表名RECORD\_TAB，引用索引名SEQ\_RECORD。

表4-8积分流水表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| R\_ID | NUMBER |  | 主键 | 编号 |
| U\_EMAIL | VARCHAR2(50) | 50 |  | 用户邮箱 |
| I\_ID | NUMBER |  |  | 积分编号 |
| R\_DATE | DATE |  |  | 获取日期 |

1. 转发表：数据库表名TRANS\_TAB，引用索引名SEQ\_TRANS。

表4-9转发表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段描述  数据库字段 | 类型 | 长度 | 约束 | 中文描述 |
| T\_ID | NUMBER |  | 主键 | 编号 |
| M\_ID | NUMBER |  |  | 微博编号 |
| TRANS\_USER\_ID | NUMBER |  |  | 转发人编号 |
| BE\_TRANS\_USER\_ID | NUMBER |  |  | 微博发布人编号 |
| TRANS\_DATE | DATE |  |  | 转发日期 |

# 5系统功能实现

5.1 登陆系统

在系统登陆注册页面当用户输入邮箱后会通过Ajax将用户输入的邮箱传到后台控制器，调用Service层中对应的方法，是Service方法中调用Dao层接口查找用户邮箱是否已经被注册，如果被注册则通过前台javaScript显示在页面提示用户该邮箱已被注册，在输入基本信息点击注册后，将会把用户输入的注册信息通过浏览器发送请求到后台控制器中，控制器控制请求的转发页面和将用户注册信息传给Service,在Service中初始化用户的一些基本信息，例如默认头像、默认状态、初始化用户积分等操作，组装用户数据源，调用Dao层方法保存用户注册信息如图5.1所示。核心代码如下：

user.setRegisterTime(nowDate()); //使用当前时间作为用户注册时间

user.setLastLogin(nowDate());

user.setBirthday("2017-1-1");

User addUser = addIntegral(user, Constant.***REGISTER***); //增加用户积分

action = userDao.findIntegral(Constant.***REGISTER***); //查找注册操作得到的PointAction

record = **new** PointRecord(action, addUser.getEmail(), nowDate());

addIntegralRecord(record); //保存积分操作

userDao.saveUser(addUser); //保存用户注册信息

登陆功能，如果点击下次自动登陆密码在点击登陆后，先将用户名和密码信息传递到Service层做业务处理，再调用Dao层接口判断邮箱地址和密码信息是否正确，如果正确并且点击了下次登陆，就将邮箱名和密码保存在浏览器Cookie中，将登陆用户保存在Session中，在处理用户登陆的Service中判断用户的当前登陆时间和上次登陆时间是否为同一天，如果不是同一天则为用户增加登陆的积分，修改用户数据库中的积分数，然后跳转至微博首页，如图5.2所示。核心代码如下：

findUser = userDao.findUserByEmailAndPassword(userId, password);

if(findUser == null){

throw new UserServiceException("用户名或密码错误");

}else{

Date lastLogin = findUser.getLastLogin(); //上次登陆时间

Date now = nowDate(); //当前时间

if(lastLogin.~~getYear~~()!=now.~~getYear~~() || lastLogin.~~getMonth~~()!=now.~~getMonth~~() || lastLogin.~~getDate~~()!=now.~~getDate~~()){

//判断如果上次登陆时间和这次登陆时间不一样，则为用户增加登陆积分

findUser=addIntegral(findUser, Constant.*LOGIN*); //保存积分

saveIRecord(findUser, Constant.*LOGIN*); //保存积分记录

findUser.setLastLogin(now); //将上次登陆时间改为当前时间

userDao.updateUser(findUser); //更改用户信息

}

}



图5.1用户登陆和注册界面



图5.2微博主页界面

5.2 用户信息系统

用户基本信息：如图5.3所示当用户点击修改后浏览器会提交form表单，发送请求携带用户基本信息到SpringMVC的控制器中接收请求，并把form表单中的数据组装成User对象，作为参数传入Service层中对应的方法进行处理，调用Dao层接口修改用户的基本信息，并修改当前服务器中Session中的User对象信息。核心代码如下：

try {

userDao.updateUser(user);

} catch (DataAccessException e) {

throw new UserServiceException("修改失败，请重试！");

}



图5.3修改用户基本信息界面

修改密码：在用户个人资料页面点击修改密码，将会跳转至如图5-4所示的修改密码页面，用户可以输入用户的当前密码，前台页面通过Ajax发送异步请求，后台控制器接收请求，从数据库中获取用户当前密码是否正确，如果密码不正确，在前台页面中通过javaScript动态提示给用户当面密码错误，如果输入密码正确，用户则可以输入新密码，确认新密码，点击确认修改后浏览器提交form表单，将用户新密码传给后台Service层中对应的方法，在Service的方法中调用Dao层接口更改数据库中的用户密码，更改服务器Session中的用户信息。核心代码如下：

try {

user.setPassword(newpwd);

userService.updateUser(user);

session.setAttribute("loginUser", user);

mav.addObject("msg", "修改成功");

} catch (UserServiceException e) {

e.printStackTrace();

}



图5.4修改密码界面

修改用户头像：点击用户个人资料中修改头像，将会跳转至如图5.5所示的修改头像页面，用户选择头像文件，点击上传，form表单将图片传到后台控制器中，将用户头像保存在服务器上，判断用户之前头像是为系统默认头像，如果不是就删除用户之前的头像图片，并将用户新的头像地址保存在用户信息中，传入Service中调用Dao层接口修改数据库中用户的头像信息。核心代码如下：

if(!(icon.isEmpty())){ //如果文件存在，isEmpty文件为空

//保存文件目录

String path = req.getServletContext().getRealPath("/")+"WEB-INF/jsp/face/"+user.getId()+ico n.getOriginalFilename();

File newFile = new File(path);

//判断文件目录存在不存在，如果不存在就创建目录

if(!(newFile.getParentFile().exists())){

newFile.getParentFile().mkdirs();

}

try {

icon.transferTo(newFile); //转存文件

File oldIcon = new File(req.getServletContext().getRealPath("/")+"WEB-INF/jsp/"+user.getImages()); //删除之前的头像

if(!(user.getImages().equals("face/boy.png") || user.getImages().equals("face/girl.png"))){

//先判断用户之前的头像不是系统默认的头像，然后删除用户之前的头像

if(oldIcon.exists()){

oldIcon.delete();

}

}

user.setImages("face/"+user.getId()+icon.getOriginalFilename());

userService.updateUser(user); //保存用户头像

session.setAttribute("loginUser", user); //将保存后的用户头像信息保存在用户session中

} catch (Exception e) {

e.printStackTrace();

}

}else{

mav.addObject("msg", "图片不存在");

}



图5.5修改头像界面

修改密保：当用户点击修改密保页面时浏览器跳转至修改密保页面如图5-6所示，用户先要根据之前设置的密保问题来填写答案，前台页面通过Ajax将用户输入的密保答案传入后台控制器，与数据库中用户的密保问题答案做判断，如果密保答案错误，在页面上通过javaScript展示用户密保答案错误，如果密保答案正确，用户可以输入新的密保问题和密保答案，点击修改提交form表单后，浏览器发送请求在后台数据库中修改用户的密保问题和答案。核心代码如下：

userService.updateUser(user);

session.setAttribute("loginUser", user);

mav.setViewName("forward:question\_index");

mav.addObject("msg", "修改成功!");



图5.6修改密保界面

5.3 微博模块

发布微博：在用户主页如图5.7所示，用户可以发布微博，在微博信息中可以插入表情，也可以选择插入一张图片，在前台页面中，表情使用javaScript动态生成div标签并显示在页面上，当用户点击发布后，浏览器发送请求将前台页面表单中微博信息和图片信息传入后台控制器，将图片信息保存在服务器中，在数据库中只保存图片路径，从Session中获取当前登陆用户，组装微博数据源，将数据源传入Service中，调用Dao层接口将微博信息保存在数据库中。核心代码如下：

if(!(pic.isEmpty())){ //如果文件存在，isEmpty文件为空

//保存文件目录

String path = req.getServletContext().getRealPath("/")+"WEB-inf/jsp/upload/pic/"+user.getId()+pic.getOriginalFilename();

File newFile = new File(path);

//判断文件目录存在不存在，如果不存在就创建目录

if(!(newFile.getParentFile().exists())){

newFile.getParentFile().mkdirs();

}

try {

pic.transferTo(newFile); //转存文件

weibo.setImage("upload/pic/"+user.getId()+pic.getOriginalFilename());

} catch (Exception e) {

e.printStackTrace();

}

}else{

//没有带图片

weibo.setImage("null");

}

try {

//保存微博信息

weibo.setUser(user);

weiboService.publishWeibo(weibo); //保存微博信息

user.setBlogNumber(user.getBlogNumber()+1); //将用户的微博数+1

userService.updateUser(user); //保存更改后的用户信息

session.setAttribute("loginUser", user);

mav.addObject("msg", "发布成功！");

} catch (Exception e) {

e.printStackTrace();

mav.addObject("msg", e.getMessage());

}



图5.7发布微博信息界面

搜索微博：如图5.8所示，在微博首页搜索框输入关键字点击搜索提交form表单，浏览器发送请求将关键字传到后台控制器中，在数据库中通过迷糊查询查询相关的微博信息。在Service层中具体方法中调用Dao层接口获得相关的微博信息，遍历集合，组装微博的基本信息和发布人对象，将微博中表情转换成对应的gif图片，调整微博时间格式为对应格式，修改微博内容中搜索关键字为红色显示，将修改好的数据返回到前台页面展示，如图5.9所示。核心代码如下：

User user = (User) session.getAttribute("loginUser");

if("weibo".equals(ac)){

mav.setViewName("seachWeiboPage");

}else if("user".equals(ac)){

mav.setViewName("seachUserPage");

}

try {

Map<String, Object> map = userService.seachWeiboAndUser(skey,pag,ac);

mav.addObject("map", map);

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());

mav.addObject("tuijian", userTuiJian);

} catch (Exception e) {

e.printStackTrace();

}



图5.8发布微博信息界面



图5.9搜索微博显示界面

微博操作；点击微博的点赞，转发和收藏功能类型，通过Ajax将微博id提交请求到后台控制器，从Session中获取登陆用户的信息，传递参数到Service对应的方法中通过对微博不同的操作调用对应的Dao层接口将微博的点赞，转发和收藏信息保存在数据库中。点击评论后通过前台页面的点击事件跳转至JavaScript中通过Ajax发送异步请求到后台控制器中，将微博id传递到Service层中对应的方法中嗲用Dao层接口查找数据库中对应微博编号的评论信息，微博评论的分页通过对应的PageBean类控制，在数据库层通过SQL语句来控制分页要显示的条数，在控制器中传递json数据到前台页面展示。在评论信息的最后面可以发布用户自己的评论，可以添加表情，点击发表将form表单提交到后台控制器中，在后台代码中调用Dao层接口保存用户的评论信息。如图5.10所示。核心代码如下：

comm.setBeCommentId(dao.findUserIdbyWId(comm.getWeiboId()));

comm.setCommentLike(0); //默认点赞数为0

//保存评论信息

dao.saveWeiboComm(comm);

Weibo weibo = dao.findWeiboBywId(comm.getWeiboId());

//查找微博评论数

commNum = dao.findCommNum(comm.getWeiboId());

weibo.setComnum(commNum);

//更新微博评论数目

dao.updateWeibo(weibo);



图5.10微博正文操作界面

个人微博：如图5.11所示：在页面上方导航栏可以点击进入用户个人主页查看我的微博，在页面中页面上方展示导航栏，下面显示用户的基本信息，中间显示用户所发布过的微博，点击微博插入的图片还可以通过JavaScript将图片放大，在页面的右侧展示系统为用户推荐的好友。点击我的微博按钮后发送的请求会被后台控制器接收，从服务器Session中获取当前登陆用户的用户id,传入Service层，在Service层中调用Dao层接口从数据库中获取当前登陆用户的微博信息组装成List集合，遍历List集合，更改微博数据源的日期格式，调用工具类将微博正文中的表情替换成对应的图片信息，在微博的分页中，使用了Oralce数据库中的伪列来获取对应区间的微博信息，实现数据库层的分页，将所有需要在页面上展示的数据传递给控制器，控制器再将数据封装在Response响应中传递到前台页面。核心代码如下：

List<Weibo> list = weiboService.findHomeWeibo(0, page, null);

list = FaceFormat.*formt*(list);

for(int i=0;i<list.size();i++){

long uId = list.get(i).getU\_id(); //获取用户id

User weibo\_user = userService.findUserById(uId);

list.get(i).setUser(weibo\_user); //将user信息组装到weibo类中

boolean b = weiboService.userLikeWeibo(list.get(i).getId(), user.getId());

list.get(i).setB(b);

//重新查找未读私信数目

Map<String, Object> alerts = userService.alerts(user.getId());

session.setAttribute("alerts", alerts);

}

int pageNumber = 0;

page.setTotalPage(pageNumber/10+1); //确定所有的微博共有多少页

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());



图5.11个人微博界面

5.4 好友模块

查看好友微博：通过点击页面导航栏中我的好友来跳转至好友微博页面，在后台控制器中先从Session中获得当前登陆用户的id值，传递给Service，在Service中调用Dao层方法先在数据库的好友表中查找对应用户的所有好友信息，然后再在数据库微博表中查看这些好友对应的微博信息按照微博发布时间倒序排列，将所有查找到的微博信息和发布用户信息封装成List集合，遍历List集合修改微博日期各式，微博表情字符转换为对应图片名称，将加工过后的集合传递给控制器，控制器将所有的组装好的数据响应到前台页面中，在前台页面中通过C标签遍历List展示微博数据如图5.12所示。核心代码如下：

List<Weibo> list = friendService.findFriendWeibo(user.getId(), page);

list = FaceFormat.*formt*(list);

for(int i=0;i<list.size();i++){

long uId = list.get(i).getU\_id(); //获取用户id

User weibo\_user = userService.findUserById(uId);

list.get(i).setUser(weibo\_user); //将user信息组装到weibo类中

boolean b = weiboService.userLikeWeibo(list.get(i).getId(), user.getId());

list.get(i).setB(b);

comm = FaceFormat.*commFormt*(comm);

list.get(i).setComm(comm); list.get(i).setStr\_pubTime(TimeFormat.*dateToString*(list.get(i).getStr\_pubTime())

}

int pageNumber = friendService.findWeiboNumber(user.getId());

page.setTotalPage(pageNumber/10+1); //确定所有的微博共有多少页

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());

关注功能：当用户登陆系统后可以在页面右侧的推荐用户中选择需要关注的用户，点击关注后前台页面通过Ajax技术发送异步请求将被关注用户的用户id发送到后台控制器中，在控制器中获取Session中的登陆用户信息，将登陆用户的id和被关注用户的id传给Service层中对应的方法，在Service层对应的方法中做处理，组装好友信息数据源，调用Dao层接口保存用户的关注信息，同时更新用户的关注数，更新被关注用户的粉丝数，然后在前台页面中通过javaScript将页面文本信息关注修改为已关注。核心代码如下：

User user = (User)session.getAttribute("loginUser");

long bId = Long.*parseLong*(attId);

User bUser = userService.findUserById(bId);

weiboService.operationUser(operation, user, bUser);

取消关注：和关注功能类型，在导航栏中点击我的好友，在中间点击关注，就能查看到我所关注的所有好友和好友数以及他们的个人信息，如图5.13所示，点击取消关注，通过Ajax将好友id传递到后台控制器中，在控制器中获取请求中的用户信息，将好友id和用户id传入Service层中对应的方法，调用Dao层接口将对应用户id和好友id的好友信息删除掉，同时更改用户的关注数和被关注用户的粉丝数。然后返回响应在前台页面刷新好友列表。

else if(operation.equals("QXGZ")){

//取消关注用户操作

dao.canUser(aId,bId);

aUser.setAttention(aUser.getAttention()-1); //a用户关注数-1

bUser.setFans(bUser.getFans()-1); //b用户粉丝数-1

}

拉黑用户：和取消关注用户类似，只是在Dao层接口中，不是删除已有的用户记录而是逻辑删除，即修改好友表中对应的记录状态，被拉黑用户所发布的私信和微博信息不会被拉黑用户所看到。核心代码如下：

else if(operation.equals("LH")){

//拉黑用户 不接收该用户的私信

friendDao.updateFriendToBlack(aId, bId);

friendDao.updateFriendState(aId, bId);

}



图5.12好友微博信息界面



图5.13好友列表界面

私信：可以在页面的推荐用户上面查看当前登陆用户的未查看私信数，如图5.14所示，未读私信的数目通过数据库中对所有接收者为当前登陆用户的所有私信信息，且信息状态为未阅读的私信，将得到的数值传递给前台页面，前台页面中所有展示的数值通过EL表达式从服务器发送回来的响应中获取。点击未读私信或者通过图5.13中显示的好友列表中的私信按钮，跳转至用户私信页面。核心代码如下：

List<User> userList = dao.findUserByMessage(aId);

if(bId == -1 && userList.size()!=0){

//通过点击私信进入

bId = userList.get(0).getId();

}

List<Message> list = dao.findMessageById(aId, bId);

for(int i=0;i<list.size();i++){

list.get(i).setUserA(userDao.findUserById(list.get(i).getUserAId()));

list.get(i).setUserB(userDao.findUserById(list.get(i).getUserBId()));

list.get(i).setMessageDate(TimeFormat.*dateToString*(list.get(i).getMessageDate()));

}

list = FaceFormat.*messageFormt*(list);

User user = userDao.findUserById(bId);



图5.14用户未读私信界面

查看私信：如图5.15所示，用户通过点击私信关注好友时，通过后台控制器跳转至我的私信用户页面，页面左侧显示和该用户最近的私信信息，页面右侧显示所有有过私信的用户列表，点击列表中的用户就可以直接私信这个用户，在私信中允许插入表情。在后台代码中，将要发送私信的用户id和从Session中获取的登陆用户id传递到Service层中对应的方法，调用Dao层接口在数据库中查找与该用户相关的所有私信记录同时修改和该用户的所有私信状态为以阅读，将获取的记录组装到集合中，遍历集合调用工具类修改私信的日期格式和文本中的表情格式，同时从数据库中获取和当前登陆用户有过私信记录的所有用户，将组装好的集合返回给控制器，控制器将数据响应给前台页面中，遍历集合中的数据，展示在前台JSP页面中。核心代码如下：

for(int i=0;i<userList.size();i++){

//查找来自该用户的未读私信

userList.get(i).setUnReadMessage(dao.findURMNByUID(aId, userList.get(i).getId()));

}

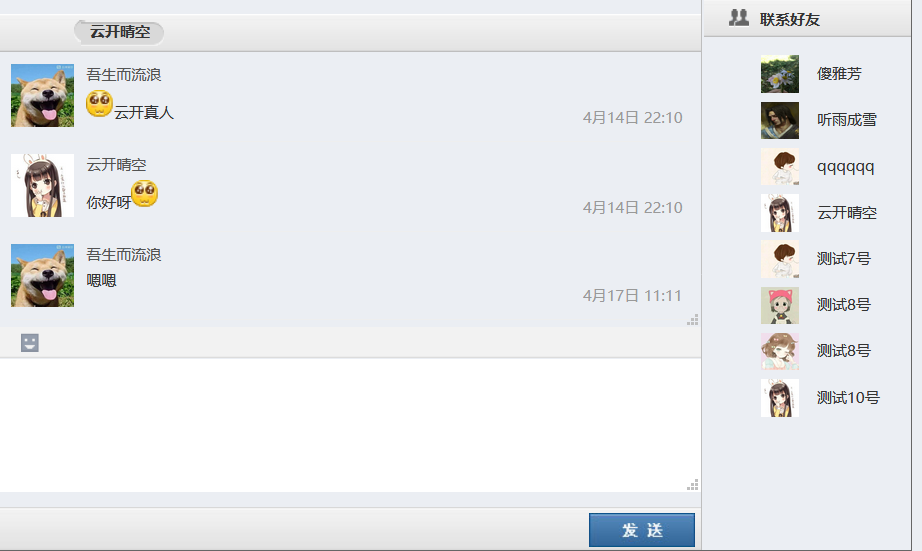


图5.15用户私信界面

发送私信：用户先选择要发送的用户，输入需要发送的私信信息，可以在私信中插入表情，点击发送后会提交form表单，浏览器发送请求到后台控制器中，控制器获取发送用户的id和接收用户的id，调用Service层中的方法，组装数据源为私信信息，设置私信信息为未阅读状态，在Service层中对应的方法中调用Dao层接口将私信信息保存在数据库中。核心代码如下：

friendService.sendMessage(message);

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());

mav.addObject("tuijian", userTuiJian);

//查找用户有私信的记录

Map<String, Object> map = friendService.findMessageById(user.getId(), userBId);

@SuppressWarnings("unchecked")

List<Message> Messagelist = (List<Message>) map.get("list");

User bUser = (User) map.get("bUser");

@SuppressWarnings("unchecked")

List<User> attUser = (List<User>)map.get("attUsers");

搜索用户：在搜索用户页面中，用户可以输入用户昵称的关键字来模糊查询相关用户，当用户输入要查询的用户昵称，浏览器发送请求携带关键字等信息跳转至控制器中特定的方法，在控制器方法内部调用Service中的方法处理逻辑，业务层调用Dao层接口中的查找方法查找用户昵称中包含有关键字的用户，将从数据库获得的对应用户组装成一个集合，遍历集合将用户昵称中包含的关键字改成红色，控制器返回响应跳转到搜索结果页面，遍历集合展示所有查找到的用户信息。如图5.16所示。核心代码如下：

Map<String, Object> map = userService.seachWeiboAndUser(skey,pag,ac);

mav.addObject("map", map);

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());

mav.addObject("tuijian", userTuiJian);



图5.16搜索用户界面

用户主页面：在页面中，点击任意一个用户的名称或头像都会跳转至对应用户的个人主页，用户的个人主页显示用户的个人信息和用户最近发布的微博，按照时间倒序排列，用户也可以对微博的点赞、转发、评论、收藏做操作。在后天代码中，当用户点击其他用户的头像或名称时，浏览器发送携带用户id的请求到后台控制器中。控制器调用Service中对应的方法，在Service方法中调用Dao层接口从数据库中查看用户的基本信息和用户的微博信息，将所有的数据存放在集合中。返回到控制器中，控制器携带数据返回到前台页面中做展示，如图5.17所示。核心代码如下：

List<Weibo> list = friendService.findFriendWeibo(user.getId(), page);

list = FaceFormat.*formt*(list);

for(int i=0;i<list.size();i++){

long uId = list.get(i).getU\_id(); //获取用户id

User weibo\_user = userService.findUserById(uId);

list.get(i).setUser(weibo\_user); //将user信息组装到weibo类中

boolean b = weiboService.userLikeWeibo(list.get(i).getId(), user.getId());

list.get(i).setB(b);

comm = FaceFormat.*commFormt*(comm);

list.get(i).setComm(comm); list.get(i).setStr\_pubTime(TimeFormat.*dateToString*(list.get(i).getStr\_pubTime()));

}



图5.17用户主页界面

## 5.5 神奇树洞模块

发布：当用户进入神奇树洞主页时，浏览器页面发送请求到后台控制器中，调用业务层中特定的方法，Service中调用Dao层接口在数据库中查找有关不同筛选条件的神奇树洞问题，并将查找到的问题封装在集合中，通过控制器发送服务器响应，跳转到神奇树洞首页，并循环展示所有的神奇树洞问题。如果用户需要发布问题，在填写了问题描述和所要悬赏的积分数后，点击发布，浏览器提交表单数据到后台控制器中，在Service层中首先判断用户的神奇树洞积分是否大于悬赏积分，如果小于悬赏积分就返回浏览器页面提示用户积分不足。如果积分足够就调用Dao层接口把用户的神奇树洞问题保存在数据库中，同时减少用户的神奇树洞积分修改用户基本信息。核心代码如下：

//神奇树洞分页

//获取神奇树洞总数

int num = userService.conchNum(user.getId(), conchOperation);

if(num%5 == 0){

page.setTotalPage(num/5);

}else{

page.setTotalPage(num/5+1);

}

page.setPage(1); //每5页一分页

page.setMinPage(1\*5-4);

page.setMaxPage(1\*5);

//查找神奇树洞显示

Map<String, Object> map = userService.conchIndex(user.getId(), conchOperation ,page);

List<User> userTuiJian = weiboService.findUserTuiJian(user.getId());



图5.18神奇树洞首页界面

查看神奇树洞问题：用户点击神奇树洞首页具体的问题时，浏览器发送请求给后台控制器中调用Service层对应的方法，在Service层中调用Dao层接口通过神奇树洞问题的编号来查看神奇树洞的具体信息，通过发布问题的用户id在数据库中查找对应的发布人信息，服务器返回响应到浏览器中，展示神奇树洞的具体信息，上方展示发布用户的用户名、用户称号、发布日期、神奇树洞问题、悬赏积分、问题的状态等，在中间显示问题的所有回复信息，回复人的用户名、称号、回答内容、回复日期，是否被采纳等，如图5.19所示。核心代码如下：

List<WeiboComm> list = weiboDao.findWeiboComm(conchId, 0, 20);

list = FaceFormat.*commFormt*(list);

for(int i=0;i<list.size();i++){ list.get(i).setCommentDate(TimeFormat.*dateToString*(list.get(i).getCommentDate()));

list.get(i).setCommUser(userDao.findUserById(list.get(i).getCommentId()));

}

map.put("conchComm", list);

图5.19查看问题具体界面

回答神奇树洞问题：在神奇树洞问题详细详细信息页面的底部可以回答神奇树洞问题，在输入框中输入回复的答案，插入表情信息，点击回复提交表单，浏览器请求携带表单数据到后台控制器中，被控制器中具体的方法接收，获取Session中的用户编号,组装回复信息的数据源，传递参数到Service中，在Service中调用Dao层接口保存神奇树洞的回复信息，如图5.20所示。核心代码如下：

comm.setCommentState(100); //100回复正常未采纳 101回复正常被采纳

weiboDao.saveWeiboComm(comm);

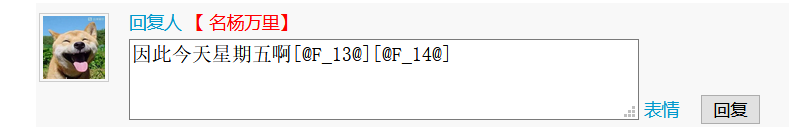


图5.20回复神奇树洞问题界面

采纳问题答案：在神奇树洞问题首页，用户通过点击我的神奇树洞问题可以跳转至用户自己所发布的神奇树洞问题页面，在自己所发布的神奇树洞问题页面中可以选择自己认为最正确的答案，被采纳的答案变为采纳答案，增加采纳者的神奇树洞积分，如图5.21所示，在后台代码中，当用户点击采纳后，浏览器请求携带神奇树洞问题id和问题回复信息被后台控制器中具体的方法接收，在控制器中调用Service层，业务层中调用Dao层接口修改数据库中神奇树洞问题的状态为已解决，修改评论表中被采纳用户的评论状态为被采纳，刷新前台神奇树洞问题页面。核心代码如下：

//设置神奇树洞状态为1

userDao.setConchState(conchId);

//设置神奇树洞评论状态为101

userDao.setConchComm(commId);

//为用户增加神奇树洞积分

Conch conch = userDao.findConchById(conchId);

User user = userDao.findUserById(userDao.findUserIdByCommId(commId));

user.setConchIntegral(user.getConchIntegral()+conch.getcNumber());



图5.21采纳问题答案界面

## 5.6 后台管理员模块

管理员登陆：管理员可以通过在登陆页面中点击管理员登陆，跳转至管理员登陆页面如图5.22所示，管理员登陆后，页面发送请求到后台控制器中，后台Controller层接收请求，将用户名和密码作为参数调用业务层中的方法，在Service层中调用Dao层接口和数据库中管理员账号表中查询，如果存在就返回管理员类，如果不存在，则抛出异常，异常层层上抛，在控制器层中接收，并将错误信息保存在方法的返回值中，在页面提示，如果用户名和密码正确，就跳转到Controller中管理员首页的处理方法中，在管理员首页的处理方法中获取需要显示的数据并展示。核心代码如下：

@RequestMapping("admin\_login")

public String adminIndex(){

return "admin/adminlogin";

}



图5.22管理员登陆界面

管理员首页：登陆后，系统会跳转到管理员首页，在首页中上方显示导航栏，在页面内容方面，通过四张图表来显示微博、用户、评、神奇树洞、评论、回答等的总数以及当月数，在下方的柱状图中则显示距离今天最近的7天的数目。后台首先在Controller层中跳转至管理员首页对应的处理方法中，调用Service层中对应的方法获取首页展示数据，并将获取到的数据组装到Map集合中，在服务层中调用Dao层中的方法来获取首页需要展示的用户、微博、评论、回答等数据保存在Map集合中，在控制层中获取到返回值数据并保存在服务器响应中，返回给前台页面使用EL表达式展示数据，如图5.23所示。核心代码如下：

//首页展示数据

try {

Map<String, Object> map = adminService.indexDate();

mav.addObject("map", map);

} catch (UserServiceException e) {

e.printStackTrace();

}

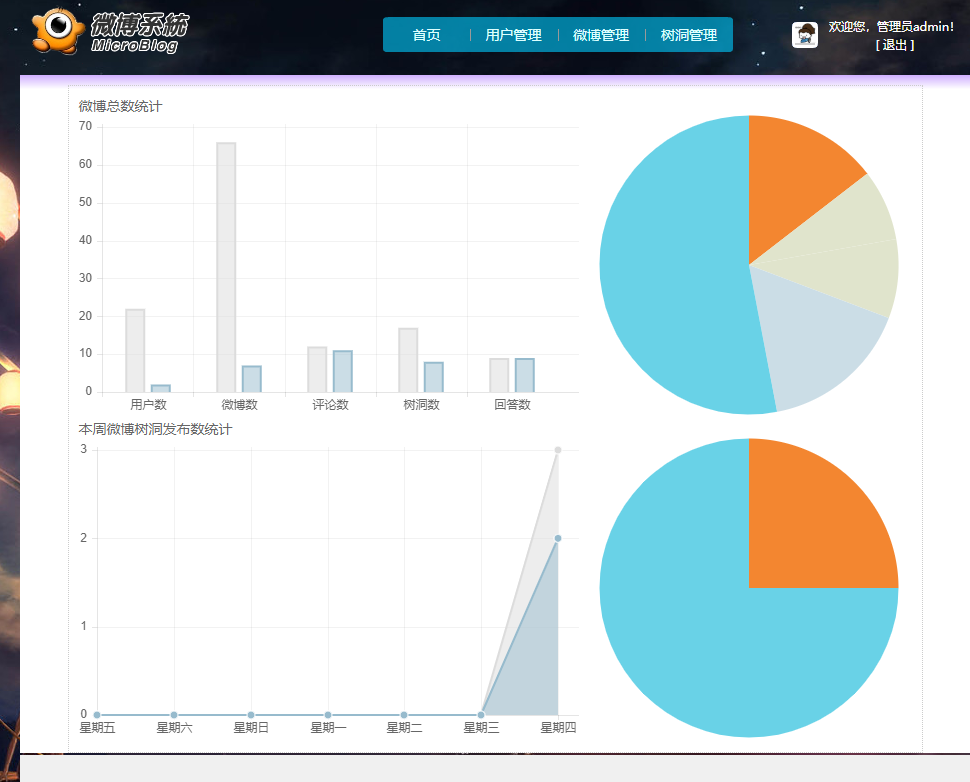


图5.23管理员首页界面

用户管理首页：后台代码通过调用Service层中对应的方法，Service方法里面则调用Dao层接口和数据库交互，获取数据库中所有的用户信息封装成List集合，返回给Controller层将用户List集合响应给浏览器，在页面中通过c标签遍历循环显示用户信息，页面的分页使用自定义分页类PageBean来保存分页信息，在数据库层做分页一次获取10条数据。核心代码如下：

int num = 0;

if(ukey == null){

num = adminService.findUserNum();

}else{

num = adminService.findUserNum(ukey);

}

if(num%10 == 0){

pageBean.setTotalPage(num/10);

}else{

pageBean.setTotalPage(num/10+1);

}

pageBean.setPage(page);

pageBean.setMinPage(page\*10-9);

pageBean.setMaxPage(page\*10);

List<User> list = null;

if(ukey == null){

list = adminService.userManage(pageBean);

}else{

list = adminService.userManageByKey(pageBean, ukey);

}

mav.addObject("userList", list);

mav.addObject("pageBean",pageBean);



图5.24用户管理界面

搜索用户:在用户管理页面输入用户昵称中的关键字来搜索用户，系统会将获取到的用户信息中昵称为输入关键字的那部分显示为红色。在后台代码中，点击搜索会将管理员输入的用户昵称关键字传给后台控制器中，在控制器中调用Service层中对应的搜索用户的方法，在Service层中调用Dao层接口在数据库中通过迷糊查询来获取用户。并将获取到的数据层层返回，在Controller中响应给前台页面，然后在页面中做展示，如图5.25所示。核心代码如下：

list = adminDao.findAllUserByKey(key,page.getMinPage(),page.getMaxPage());

for(int i=0;i<list.size();i++){

list.get(i).setNickName(FaceFormat.*keyFormt*(list.get(i).getNickName(), key));

}

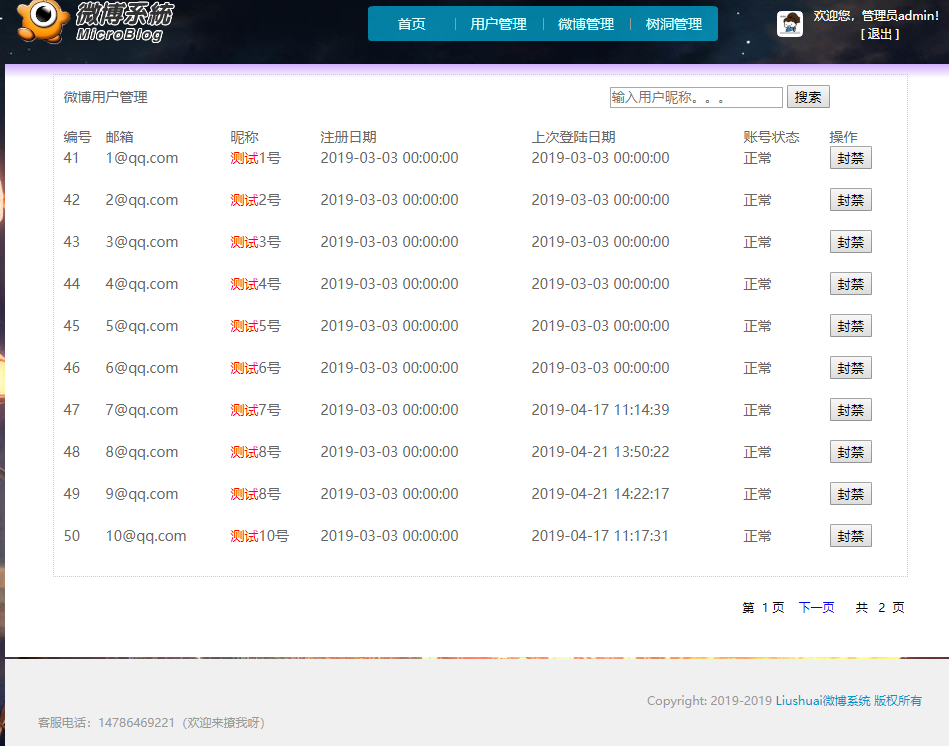


图5.25管理员搜索用户界面

用户封禁：管理员可以通过点击用户管理页面操作一栏中的封禁按钮来对违规用户的封禁，管理员可以输入封禁的天数，那么在这个日期之前，用户是不能正常登陆的，管理员也可以通过点击解封来提前解除封禁用户的操作，具体页面如图5.26所示。在后台代码中则是在Service层中调用Dao层接口，更改用户状态和封禁日期，解除封禁和封禁用户类似，因此不做具体说明。核心代码如下：

mav.setViewName("redirect:userManage");

try {

adminService.FJUser(uId, day); //uId封禁用户编号，day封禁天数

} catch (UserServiceException e) {

e.printStackTrace();

}

mav.addObject("page", 1);

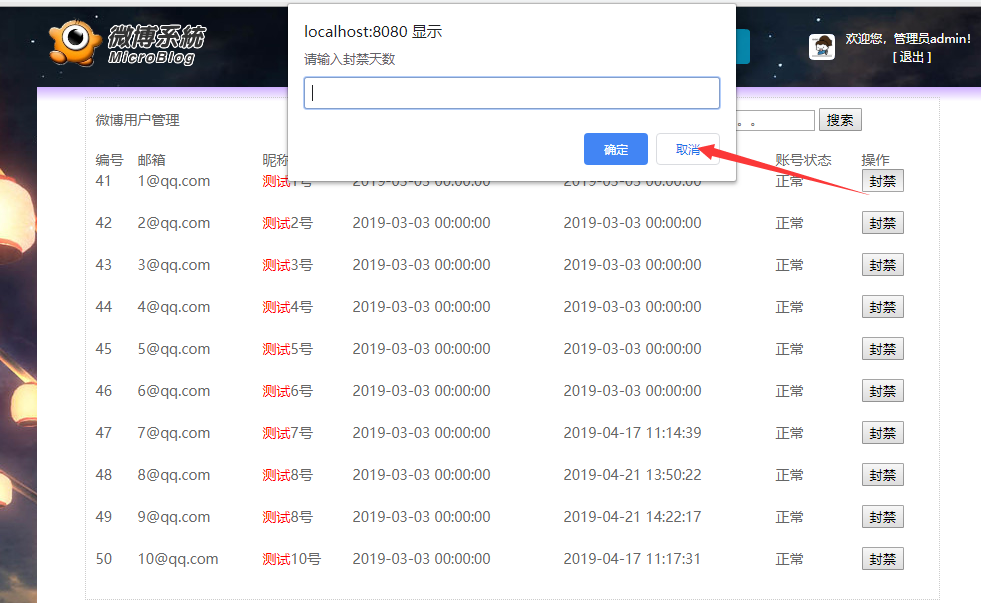


图5.26封禁用户界面

微博管理首页：管理员点击导航栏中的微博管理，系统会跳转至用户微博管理页面，显示系统中所用是微博信息，管理员也可以通过输入微博内容中的关键字来搜索微博，同时可以删除有不良信息的微博，或是恢复以被删除的微博，微博管理页面如图5.27所示，在后台代码中，当用户点击导航栏中微博管理时，浏览器会发送相应的请求到SpringMVC框架的Controller中对用的更能处理方法中，在控制器中调用Service层中显示所有微博的方法中，在Service层中再调用Dao层方法获取所有的微博信息，在分页方面没有采用在前端页面中做分页的方法，而是在数据库中通过Oracle的伪列来做分页，一次获取10条数据，最后在Controller中将获取到的微博信息响应给浏览器，浏览器中通过c标签遍历显示微博信息。核心代码如下：

String nkey = "";

if(key == null){

nkey = "1=1";

}else{

nkey = "m\_body like '%"+key+"%'";

}

list = adminDao.findWeiboByKey(nkey, pageBean.getMinPage(), pageBean.getMaxPage());

list = FaceFormat.*formt*(list);

for(int i=0;i<list.size();i++){

list.get(i).setStr\_pubTime(TimeFormat.*timeFormat*(list.get(i).getPubTime()));

list.get(i).setUser(userDao.findUserById(list.get(i).getU\_id()));

if(key != null){

list.get(i).setBody(FaceFormat.*keyFormt*(list.get(i).getBody(), key));

}

}



图5.27微博管理界面

微博搜索：管理员可以通过输入微博内容中的关键字来搜索在微博中存在该关键字的微博，搜索到的微博内容中的关键字会使用红色标注出来，具体页面如图5.28所示，在后台代码中，控制器中特定的方法接收浏览发送的搜索微博请求，调用Service层中对应的方法，将关键字作为参数传给Dao层接口中，在Dao层接口中查找数据库微博表中微博内容包含该关键字的微博信息，在数据库中则是通过模糊查询来查找对应微博。然后将查找到微博信息封装到List集合中，层层返回到Controller层中对应的方法，在方法中将数据响应给浏览器，浏览器接收响应在页面中通过c标签展示数据。核心代码如下：

if(key == null){

nkey = "1=1";

}else{

nkey = "m\_body like '%"+key+"%'";

}

list = adminDao.findWeiboByKey(nkey, pageBean.getMinPage(), pageBean.getMaxPage());



图5.28管理员搜索微博界面

微博删除：在微博管理页面中，管理员可以根据微博的内容来判断微博是否违法等信息，如果微博信息中包含不良信息，管理员可以通过操作栏中的删除按钮来删除微博或者可以对已经删除的为微博做恢复操作，当用户点击删除时，页面会携带着微博编号等参数发送请求给服务器，请求会被控制器中对应的方法所接收，将微博编号作为参数嗲用Service中对应的方法，在方法中调用Dao层接口在修改数据库中对应微博编号的微博状态为已删除，恢复微博和删除类似，只是修改数据库中对应微博信息状态为正常即可。核心代码如下：

if("sc".equals(ac)){

adminDao.upWeiboWithState(wId, 1);

}else if("hf".equals(ac)){

adminDao.upWeiboWithState(wId, 0);

}

神奇树洞管理首页：管理员点击导航栏中的神奇树洞管理可以跳转至神奇树洞管理页面，如图5.29所示，在页面的上方为导航栏，页面内容则显示所有的神奇树洞信息、神奇树洞搜索框、以及下方的分页框，神奇树洞信息包括编号、发布人名称、神奇树洞的内容、发布日期、神奇树洞状态以及可以执行的操作，在后台代码方面，当用户点击导航栏中的神奇树洞管理时，浏览器发送请求，请求在控制器中被对应的方法接收,Service中调用Dao层接口在数据库神奇树洞表中查找所有的神奇树洞信息，保存在List集合中，遍历List集合组装发布人用户信息，修改日期格式等，最后将组装号的List集合返回到Controller中，响应给浏览器跳转神奇树洞管理首页，通过使用c标签遍历显示查找到的神奇树洞信息，因为系统中使用的分页方法一致，因此在这里不再赘述。核心代码如下：

int num = adminService.findConchNum(ckey);

if(num%10 == 0){

pageBean.setTotalPage(num/10);

}else{

pageBean.setTotalPage(num/10+1);

}

pageBean.setPage(page);

pageBean.setMinPage(page\*10-9);

pageBean.setMaxPage(page\*10);

List<Conch> list = adminService.findConch(pageBean, ckey);



图5.29神奇树洞管理首页界面

神奇树洞搜索：在神奇树洞管理首页中，考虑到神奇树洞数量多不好查找的问题，因此设置了搜索功能，管理员可以通过在神奇树洞搜索框中输入神奇树洞内容中的关键字来搜索神奇树洞问题，在展示搜索到的结果时会将搜索关键字使用红色标注出来，方便查看。具体页面如图5.30所示。在后台代码中，当用户输入关键字点击搜索时，浏览器发送请求到控制器中指定方法接收，在Controller中调用Service层中对应的方法处理业务逻辑，然后在Service层中调用Dao层接口通过模糊查询在数据库神奇树洞表中查找对应的神奇树洞信息，最后在Controller中将查找到的神奇树洞信息响应给浏览器，在页面中通过c标签遍历展示数据。因为系统使用一样的分页方法，因此不做赘述。核心代码如下：

if(key == null){

nkey = "1=1";

}else{

nkey = "CONCHBODY like '%"+key+"%'";

}

System.*out*.println(nkey);

list = adminDao.findConchByKey(nkey, pageBean.getMinPage(), pageBean.getMaxPage());

list = FaceFormat.*conchFormt*(list);



图5.30神奇树洞搜索界面

# 6系统测试

## 6.1 系统测试综述

代码编写阶段主要采用了单元测试来测试代码是否存在缺陷，系统的功能能否正常运行，软件能否达到预期目的，在软件系统完成后编写相应模块的测试案例，根据测试案例中的步骤来测试系统中是否存在缺陷，同时采用黑盒测试和白盒测试来完善系统功能，优化软件代码，提高软件系统的质量。

## 6.2 测试用例

6.2.1用户注册模块

表6-1用户注册

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB001 | | | |
| 功能描述 | | | 用户注册 | | | |
| 功能URL | | | http://localhost:1921/weibo/register\_index | | | |
| 用例目的 | | | 测试用户是否能注册成功 | | | |
| 前提条件 | | | 进入用户注册界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 输入已被注册的邮箱号 | 系统提示该邮箱已被注册 | | 系统提示该邮箱已被注册 | 正确 |  |
| 2 | 不输入邮箱号 | 系统提示请输入邮箱号 | | 系统提示请输入邮箱号 | 正确 |  |
| 3 | 输入正确的邮箱号但是不输入密码 | 系统提示用户输入密码 | | 系统提示用户输入密码 | 正确 |  |
| 4 | 输入正确的邮箱号输入密码长度小于6位 | 系统提示用户密码格式错误 | | 系统提示用户密码格式错误 | 正确 |  |
| 5 | 不输入确认密码 | 系统提示两次输入的密码不一致 | | 系统提示两次输入的密码不一致 | 正确 |  |
| 6 | 不输入密保问题 | 系统提示用户输入密保问题 | | 系统提示用户输入密保问题 | 正确 |  |
| 7 | 输入密保问题不输入密保答案 | 系统提示用户输入密保答案 | | 系统提示用户输入密保答案 | 正确 |  |
| 8 | 输入密保答案不输入密保问题 | 系统提示用户输入密保问题 | | 系统提示用户输入密保问题 | 正确 |  |
| 9 | 输入错误的验证码 | 系统提示用户输入的验证码错误 | | 系统提示用户输入的验证码错误 | 正确 |  |
| 10 | 全部输入正确 | 系统正确跳转至登陆页面 | | 系统正确跳转至登陆页面 | 正确 |  |

6.2.2用户登陆模块

表6-2用户登陆

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB002 | | | |
| 功能描述 | | | 用户登录 | | | |
| 功能URL | | | http://localhost:1921/weibo/register\_index | | | |
| 用例目的 | | | 测试用户是否能登录成功 | | | |
| 前提条件 | | | 进入用户登录界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 输入不存在的用户名，不填写密码 | 系统提示用户名或密码错误 | | 系统提示登录名或密码错误 | 正确 |  |
| 2 | 输入正确的用户名输入错误的密码 | 系统提示用户名或密码错误 | | 系统提示登录名或密码错误 | 正确 |  |
| 3 | 输入错误的用户名输入正确的密码 | 系统提示用户名或密码错误 | | 系统提示登录名或密码错误 | 正确 |  |
| 4 | 只填写用户名 | 系统提示用户名或密码错误 | | 系统提示用户名或密码错误 | 正确 |  |
| 5 | 只填写密码 | 系统提示用户名或密码错误 | | 系统提示用户名或密码错误 | 正确 |  |
| 6 | 输入正确用户名和密码 | 系统进入用户首页 | | 系统进入用户首页 | 正确 |  |

6.2.3修改用户信息模块

表6-3修改用户信息

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB003 | | | |
| 功能描述 | | | 修改用户个人信息 | | | |
| 功能URL | | | http://localhost:1921/weibo/userinfo\_index | | | |
| 用例目的 | | | 测试用户是否能正确修改个人信息 | | | |
| 前提条件 | | | 进入修改用户个人信息界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 修改用户所在地 | 系统提示修改个人信息成功并刷新个人信息 | | 系统提示修改个人信息成功并刷新个人信息 | 正确 |  |
| 2 | 修改用户性别 | 系统提示修改个人信息成功并刷新个人信息 | | 系统提示修改个人信息成功并刷新个人信息 | 正确 |  |
| 3 | 修改用户签名 | 系统提示修改个人信息成功并刷新个人信息 | | 系统提示修改个人信息成功并刷新个人信息 | 正确 |  |
| 4 | 修改密码时输入错误的当前密码 | 系统提示密码错误 | | 系统提示密码错误 | 正确 |  |
| 5 | 输入的新密码长度小于6位 | 系统提示密码长度为6-16位 | | 系统提示密码长度为6-16位 | 正确 |  |
| 6 | 输入的新密码长度大于16位 | 系统提示密码长度为6-16位 | | 系统提示密码长度为6-16位 | 正确 |  |
| 7 | 输入与新密码不同的确认密码 | 系统提示两次输入的密码不一致 | | 系统提示两次输入的密码不一致 | 正确 |  |
| 8 | 未选择头像图片时点击上传 | 系统提示文件格式有误 | | 系统提示文件格式有误 | 正确 |  |
| 9 | 选择非图片格式文件上传 | 系统提示文件格式有误 | | 系统提示文件格式有误 | 正确 |  |
| 10 | 修改密保时输入错误的密保答案 | 系统提示答案错误 | | 系统提示答案错误 | 正确 |  |
| 11 | 不输入新的密保问题 | 系统提示密保问题不能为空 | | 系统提示密保问题不能为空 | 正确 |  |
| 12 | 不输入新的密保答案 | 系统提示密保答案不能为空 | | 系统提示密保答案不能为空 | 正确 |  |

6.2.4微博模块

表6-4用户微博模块

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB004 | | | |
| 功能描述 | | | 用户微博操作 | | | |
| 功能URL | | | http://localhost:1921/weibo/home\_index | | | |
| 用例目的 | | | 测试用户能否正常对微博操作 | | | |
| 前提条件 | | | 进入微博首页界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 发布微博不输入内容点击发布 | 系统提示微博内容不能为空 | | 系统提示微博内容不能为空 | 正确 |  |
| 2 | 发布微博输入内容大于144个字符 | 系统提示微博内容长度太长 | | 系统提示微博内容长度太长 | 正确 |  |
| 3 | 发布微博插入非图片格式文件 | 系统提示图片格式错误 | | 系统提示图片格式错误 | 正确 |  |
| 4 | 发布微博插入图片大于2M | 系统提示图片太大 | | 系统提示图片太大 | 正确 |  |
| 5 | 点击微博中的点赞按钮 | 系统提示微博点赞数+1 | | 系统提示微博点赞数+1 | 正确 |  |
| 6 | 点击微博中的转发按钮 | 系统提示转发成功并且在个人微博中能看到转发的微博 | | 系统提示转发成功并且在个人微博中能看到转发的微博 | 正确 |  |
| 7 | 点击微博中的评论按钮 | 系统提示评论成功，并且看查看到 | | 系统提示评论成功，并且看查看到 | 正确 |  |
| 8 | 点击微博中的收藏按钮 | 系统提示收藏成功，在我的收藏中可以查看到 | | 系统提示收藏成功，在我的收藏中可以查看到 | 正确 |  |
| 9 | 点击微博中的删除按钮 | 系统提示删除成功 | | 系统提示删除成功 | 正确 |  |
| 10 | 点击个人原创微博 | 系统显示个人所有原创微博 | | 系统显示个人所有原创微博 | 正确 |  |
| 11 | 点击个人转发微博 | 系统显示个人所有转发微博 | | 系统显示个人所有转发微博 | 正确 |  |
| 12 | 点击个人收藏微博 | 系统显示个人所有收藏微博 | | 系统显示个人所有收藏微博 | 正确 |  |

6.2.5用户好友模块

表6-5用户好友模块

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB005 | | | |
| 功能描述 | | | 用户好友管理 | | | |
| 功能URL | | | http://localhost:1921/weibo/friend\_index | | | |
| 用例目的 | | | 测试用户是否正确管理好友 | | | |
| 前提条件 | | | 进入用户好友界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 点击我的好友按钮 | 系统跳转至用户好友页面 | | 系统跳转至用户好友页面 | 正确 |  |
| 2 | 点击好友微博按钮 | 系统显示所有好友最近发布的微博 | | 系统显示所有好友最近发布的微博 | 正确 |  |
| 3 | 点击我的关注 | 系统显示用户关注的所有好友 | | 系统显示用户关注的所有好友 | 正确 |  |
| 4 | 点击我的粉丝 | 系统显示所有关注用户的粉丝 | | 系统显示所有关注用户的粉丝 | 正确 |  |
| 5 | 点击好友私信 | 系统跳转至对应用户的私信页面 | | 系统跳转至对应用户的私信页面 | 正确 |  |
| 6 | 点击用户关注 | 系统提示好友关注成功 | | 系统提示好友关注成功 | 正确 |  |
| 10 | 点击用户取消关注用户 | 系统提示以取消关注该用户 | | 系统提示以取消关注该用户 | 正确 |  |
| 11 | 点击拉黑好友 | 系统提示已成功拉黑用户 | | 系统提示已成功拉黑用户 | 正确 |  |

6.2.6用户神奇树洞模块

表6-6用户神奇树洞模块

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB006 | | | |
| 功能描述 | | | 用户神奇树洞管理 | | | |
| 功能URL | | | http://localhost:1921/weibo/conch\_index?operation=NEWS&nPage=1 | | | |
| 用例目的 | | | 测试用户是否正确执行对神奇树洞的操作 | | | |
| 前提条件 | | | 进入用户登录界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 点击神奇神奇树洞 | 系统跳转至用户神奇树洞首页 | | 系统跳转至用户神奇树洞首页 | 正确 |  |
| 2 | 不输入内容点击发布 | 系统提示神奇树洞问题不能为空 | | 系统提示神奇树洞问题不能为空 | 正确 |  |
| 3 | 输入的悬赏积分数大于用户积分数 | 系统提示用户名积分不足 | | 系统提示用户名积分不足 | 正确 |  |
| 4 | 点击最新发布 | 系统显示最新发布神奇树洞问题 | | 系统显示最新发布神奇树洞问题 | 正确 |  |
| 5 | 点击已解决 | 系统显示已解决的神奇树洞问题 | | 系统显示已解决的神奇树洞问题 | 正确 |  |
| 6 | 点击未解决 | 系统显示未解决的神奇树洞问题 | | 系统显示未解决的神奇树洞问题 | 正确 |  |
| 7 | 点击我的神奇树洞 | 系统显示我的神奇树洞问题 | | 系统显示我的神奇树洞问题 | 正确 |  |
| 8 | 点击神奇树洞问题 | 系统跳转至问题详细页面 | | 系统跳转至问题详细页面 | 正确 |  |
| 9 | 不输入回复内容点击回复按钮 | 系统提示回复内容不能为空 | | 系统提示回复成功 | 错误 | E01 |
| 10 | 输入正确的回复内容点击回复按钮 | 系统提示回复成功并显示回复内容 | | 系统提示回复成功并显示回复内容 | 正确 |  |
| 11 | 点击采纳按钮 | 神奇树洞状态变为已解决同时该回复变为已采纳 | | 神奇树洞状态变为已解决同时该回复变为已采纳 | 正确 |  |

6.2.7管理员模块

表6-7管理员模块

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 功能测试 | | | | | | |
| 概述 | | | | | | |
| 测试编号 | | | WB002 | | | |
| 功能描述 | | | 管理员模块 | | | |
| 功能URL | | | http://localhost:1921/weibo/adminIndex | | | |
| 用例目的 | | | 测试管理员用户功能是否正常 | | | |
| 前提条件 | | | 进入管理员首页界面 | | | |
| 测试操作 | | | | | | |
| 编号 | 输入/动作 | 期望的输出响应 | | 实际情况 | 是否正确 | 错误编号 |
| 1 | 点击管理员首页 | 系统跳转至管理员首页 | | 系统跳转至管理员首页 | 正确 |  |
| 2 | 鼠标移动至图表上方 | 系统显示图表详细信息 | | 系统显示图表详细信息 | 正确 |  |
| 3 | 点击用户管理 | 系统跳转至用户管理页面 | | 系统跳转至用户管理页面 | 正确 |  |
| 4 | 点击用户封禁按钮 | 系统提示管理员输入封禁天数 | | 系统提示管理员输入封禁天数 | 正确 |  |
| 5 | 不输入封禁天数 | 系统提示天数不能为空 | | 系统提示天数不能为空 | 正确 |  |
| 6 | 输入字符串 | 系统提示天数格式错误 | | 系统提示天数格式错误 | 正确 |  |
| 9 | 输入正确天数 | 系统显示封禁到期时间 | | 系统显示封禁到期时间 | 正确 |  |
| 10 | 点击取消封禁 | 系统显示用户状态正常 | | 系统显示用户状态正常 | 正确 |  |
| 11 | 点击微博管理 | 系统跳转至微博管理页面 | | 系统跳转至微博管理页面 | 正确 |  |
| 12 | 点击删除微博按钮 | 系统显示该微博被删除 | | 系统显示该微博被删除 | 正确 |  |
| 13 | 点击恢复被删除微博按钮 | 系统显示微博状态正常 | | 系统显示微博状态正常 | 正确 |  |
| 14 | 搜索框不输入文字点击搜索 | 系统显示所有的信息 | | 系统显示所有的信息 | 正确 |  |
| 15 | 搜索框输入关键字点击搜索 | 系统显示搜索到相关的信息 | | 系统显示搜索到相关的信息 | 正确 |  |
| 16 | 点击神奇树洞管理按钮 | 系统跳转至神奇树洞管理页面 | | 系统跳转至神奇树洞管理页面 | 正确 |  |
| 17 | 点击删除神奇树洞按钮 | 系统显示神奇树洞已被删除 | | 系统显示神奇树洞已被删除 | 正确 |  |
| 18 | 点击恢复神奇树洞按钮 | 系统显示神奇树洞状态正常 | | 系统显示神奇树洞状态正常 | 正确 |  |
| 19 | 点击页面下一页 | 系统跳转至页面下一页 | | 系统跳转至页面下一页 | 正确 |  |
| 20 | 点击页面上一页 | 系统跳转至页面上一页 | | 系统跳转至页面上一页 | 正确 |  |

6.3 测试分析

因为在代码的编写阶段中对系统中每一个功能都做了相应的单元测试，在单元测试阶段就已经对系统中存在的大量缺陷作出了修改，在后期系统完成之后根据测试用例来测试系统后仅发现及少量缺陷，但是没有缺陷的系统是不存在的，因此在系统中还存在一些没有被发现的缺陷，需要在长期的使用中不断完善。对于错误编号E01的错误在对代码查看后发现在前台页面未编写JavaScript代码对文本框内容做验证，在修改了代码之后再次验证，系统可以成功提示用户输入的回复内容不能为空。

# 7 总 结

时光荏苒，岁月如梭，转眼之间为期半年的毕业设计以及论文的编写终于落下帷幕，回顾毕业设计的每一个阶段都使我受益匪浅，在毕设初期，经过了长达一周的深思熟虑之后我决定将B/S微博系统作为我这次毕业设计和论文的选题，因为平时对微博的接触以及近几年微博的火热程度，使我对微博系统的具体功能和优缺点都有了一个全体的把控，这也使得在之后系统功能设计时能够更加的得心应手，在毕业设计中期时进入代码编写阶段时，我选择了J2EE体系开发Web项目的B/S微博，使用面向对象语言JAVA作为开发语言，使用个人熟练掌握的SSM框架来搭建系统，SSM框架强大的功能减少了大量的冗余代码，使得系统代码的编写更加轻松，提高了系统的开发效率，然而开发一个系统并不是那么简单就能完成，在代码的编写阶段，问题接连出现，但在指导老师的指导下以及自己通过网上查阅资料，最终解决了这些问题，提高了自己代码的编写能力，同时也提高了自我学习能力，这对以后的生活学习和工作中都有着非同寻常的意义，在项目编写完成后的测试阶段，之前在编写阶段没有被发现的系统缺陷逐渐跃出水面，在每一次修复这些问题的时候，我都能感觉到自己的能力在慢慢提高，一个没有缺陷的系统是不存在的，经过对测试用例中的测试方案一一测试，然后修复掉一个又一个的缺陷后，微博系统中大多数可见性高的BUG都被修复了，但是在之后的使用中还会不断的完善系统让它变的更加成熟，而不单单只是作为一份毕设设计而存在。

在这次毕业设计系统的开发中也让我看到了自身的一些问题，例如前端技术的不熟练导致在前台页面的修改和开发中浪费了大量的时间，微博系统中关键性的用户交互不够美观，用户体验性差，以及在系统类设计时没能正确的把控全局，设计出合理的接口，只能在后期代码编写阶段中不断的去完善。

总而言之，在对微博系统的开发中，我学习到了很多以前没有注意到和忽略到的东西，也使我认识到了自身的一些缺陷，让我在以后的生活和工作中都能更好的认识自己，提高自己的能力，然后服务于社会，做一个对社会发展有帮助的人。

# 参考文献

[1] 贾文潇,邓俊杰. 基于Java的Web开发技术浅析[J]. 电子测试,2016

[2] 李传扬. 微博分析系统的设计与实现[D]. 北京邮电大学 2015

[3] 刘运臣. 网站设计与建设[M]. 清华大学出版社, 2008

[4] 秦雅华. 基于WEB2.0的微博网站的设计与实现[D]. 北京工业大学 2012

[5] 陈玲,夏汛. 利用Mybatis的动态SQL实现物理分页[J]. 数字技术与应用. 2011(11)

[6] 萨师煊,王珊. 数据库系统概论(第三版)[M].北京：高等教育出版社，1998

[7] 基于Java的数据库访问技术研究[J]. 科技资讯. 2009(04)

[8] 张峰. 基于Ajax技术与J2EE框架的Web应用研究与实现[D]. 中国地质大学 2008

[9] 基于Java多线程技术的网络编程[J]. 电脑编程技巧与维护. 2009(22)

[10] 李威. 一种小型实用即时网络聊天通讯系统的设计[J]. 长江大学学报(自然科学版). 2011(12)

[11] 钟睿祺. 基于微博嵌入小伙伴阅读网的分析与设计[D]. 华南理工大学 2011

[12] 王少锋编著.面向对象技术UML教程[M]. 清华大学出版社, 2004

[13] 徐春绵. 关于网站开发相关问题的探究[J]. 通讯世界. 2015(09)

[14] 张宇,王映辉,张翔南. 基于Spring的MVC框架设计与实现[J]. 计算机工程. 2010(04)

[15] 胡以谰,张立平. J2EE开发模式的选择[J]. 计算机系统应用. 2002(08)

[16] 王丽爱. 《Java程序设计》课程网站的设计与实现[J]. 电脑知识与技术. 2016(27)

[17] 荣艳冬. 关于Mybatis持久层框架的应用研究[J]. 信息安全与技术. 2015(12)

# 致 谢

转眼间，四年过去了，而四年前初入校门时青涩的身影依旧在脑海中挥之不去，似乎大学生活即将完结，但人总是要学会长大，独自面对未来的艰难险阻，而在大学中的这四年时光，无论是课题上的专注，课间的嬉戏，运动场中的挥汗如雨，这一切都是我青春中最美好的回忆，感谢众多老师的的传道解惑，感谢同学之间的互帮互助，感谢父母亲人的含辛茹苦，如果没有你们的支持，就不会有现在的我，在这里我还还要感谢我的指导老师，回忆在毕设以及论文选题的初期对我的悉心教导，在我系统代码实现中遇到难以解决的问题时，指导老师都会耐心指导我完成代码和论文的编写。由于时间安排不当，在工作期间编写了毕设代码以及论文，感谢大学对我学习和工作的支持和理解，

最后再次感谢一直陪伴我的同学亲人，良师益友对我这四年以来的帮助和照顾，感谢段华斌老师对我毕设及论文的悉心指导，感谢论文中所有参考文献的原作者，感谢所有帮助过我的人