毕业论文

学　　院：	天津理工大学华信软件学院
专业名称：	软件工程
所在班级：	2013 级 1 班
学　号：	20134940
姓　名：	郭强
课题名称：	比邻微博
分项题目	评论、站内信、管理
完成时间：	2017 年 05 月 01

比邻微博

——评论、站内信、管理

摘要

Spring Boot 是一款完全开源的框架，该框架由 Pivota 团队提供，该框架设计的目的就是用来简化新 Spring 应用的初始搭建和开发过程，这个框架使用了特定的方法进行配置，因此使开发人员不再需要定义样板化的配置工作，减少了开发周期，就是由于该框架的支持，使得比邻微博的开发周期大大减小。

整个系统大的方向个人负责两个板块，普通用户和管理员。

普通用户板块包含了用户的登录注册、站内信的推送、评论功能；登录注册采用了盐加密的方式，这样在一定程度上确保了用户的信息安全，即使在数据库被泄露的情况下，也不会使用户的密码轻易被破解；站内信推送给用户提供了一个交流和管理员与用户之间信息传递的一个平台，使用户能够最快的接收到站内通知；评论功能的亮点在于用户可以对已有的评论再进行评论从而形成一个评论树，丰富用户交流。

管理员板块包含了对用户的微博管理、对用户的基本信息的管理、以及对用户的评论的管理；管理员板块主要是对用户的一些基本信息的维护。

比邻微博是微型博客的一种简称，是一个基于用户关系的信息的分享、传播和获取的平台，用户可以用文字、图片、视频等等形式来分享关于自己的信息，使用户可以在第一时间内了解到朋友、新闻媒体、偶像歌手等关注用户的实时动态。

关键词：微博　比邻微博　Spring Boot　Spring　盐加密　站内信

Bi Lin Blog

——Comments、Station letter、Management

ABSTRACT

Spring Boot is a fully open source framework that is provided by the Pivota team, which is designed to simplify the initial setup and development of new Spring applications, which are configured using a specific method, so that developers No longer need to define the configuration of the template, reducing the development cycle, is due to the support of the framework, making the microblogging development cycle greatly reduced.

The overall direction of the whole system is responsible for two sections, the average user and the administrator.

The general user section contains the user's login registration, the station letter of the push, comment function; login registration using a salt encryption way, so to a certain extent, to ensure that the user's information security, even in the case of the database is leaked Will make the user's password easily cracked; station letter push to the user to provide a communication and management between the administrator and the user a platform for the transmission, so that users can receive the fastest notice of the station; comment feature highlights the user can The existing comments are then commented to form a review tree that enriches user communication.

The administrator section contains the management of the user's microblogging, the management of the basic information of the user, and the management of the user's comments. The administrator section is mainly the maintenance of some basic

information about the user.

Weibo is a short description of microblogging, a platform for sharing, disseminating and acquiring information based on user relationships. Users can share their own information in the form of text, pictures, videos, etc., so that users can A time to understand friends, news media, idol singer and other users concerned about the real-time dynamic.

Key Words：Microblogging　BILINMicroblogging　Spring Boot　Spring Salt encryption　Station letter

目　　录

第一章 绪论

1.1 研究背景

当前计算机网络通讯技术发达，移动和固定网络的迅速普及，人们对网络的使用率也逐渐提高，伴随移动互联网的发展和不断更新迭代，也沉淀出了很多优秀的互联网作品，比如 Facebook，新浪，Twitter 等社交产品。尽管这些优秀的产品经历了时间的验证也经历了用户的考验，但是仍然不能够满足用户对社交的需求，用户想要获取更多的身边发生的信息，想要接触更多的人，所以比邻微博便加入了社交的行业，为用户的社交出一份力，同时满足用户对社交的大部分需求。

1.2 目前的应用现状

在移动互联网的推动下，个人互联网应用发展整体呈现上升态势。尽管即时通信作为网民第一大上网应用，在高使用率水平的基础上继续攀升导致近两年微博的使用率在有所下降，但微博作为信息发布交流平台，其热点事件、事实新闻搜索的及时性；电子商务方面的推广作用；数据挖掘、情报搜集方面的重要性不可替代。本文主要对微博在电子商务、社交人群分析、社会影响三个方面进行了分析，实际上微博产生的大数据在情报搜集、社群检测、在线广告等方面都具有很高的价值。

随着微博的普及，微博已成为网民随时发出自己声音的便捷平台，然而因为微博信息发布的便捷性、信息传递的快速性以及普通网民对专业新闻知识的缺乏，微博同时也成为了谣言传播的载体和不满情绪的导火索，而微博营销价值对于电子商务企业重要性，也使得微博容易沦为商业的炒作工具。由于缺少信息审核机制，使得一些暴力、色情、虚假信息广告也在微博中迅速传播。

但是只要我们加强微博信息的审核和监管，促进其健康发展并合理运用这个平台，微博平台便能在电子商务、文化传播、社会进步等方面发挥更大的作用。

1.3 实现目标和意义

1.3.1 实现目标

本次课题的设计目标旨在为大众群体提供更加优质的社交服务，希望通过该社交平台能够给用户带来更加舒畅的社交体验，通过比邻微博解可以让用户随身边的人群有一个更加深入的了解拉近彼此的感情，让你时时刻刻关注身边的实事，从此不再是一个孤陋寡闻的人，也不至于让你感到社交尴尬。

1.3.2 系统意义

微博开辟了一个资讯高速流动时代。当今社会运转速度和人们的生活节奏越来越快，信息流动的速度越来越快，碎片化的内容比长篇表达更适合阅读，微博上的内容更适合时代的需要。微博充分满足了现代快节奏高压力社会下人们急剧上升的个人表达与倾诉沟通的需求。

微博关注与被关注的不对称人际关系加之独特的广播式信息流动模式，带给人们一种全新的沟通方式。用户在 140 个汉字之内更乐于记录与分享观点，发布新鲜事，发布新想法，发布新情绪，而不需要就一个情绪、一个灵感去编辑一个标题，一段经过加工的文字发表一篇文章。微博提供了一种全新的沟通工具，是一种介于人际对话互动和广播之间的媒介工具。

第二章 系统开发环境及相关技术简介

2.1 系统开发环境的简介

本系统所使用的开发平台为 Tomcat+Myeclipse2014，开发语言是 Java，后台数据库使用的是 mysql。

Myeclipse2014 作为 Java 的可视化开发平台，现在已经被越来越多的人使用。相比其他的 Java 开发工具，明显的有如下 3 点：

1、Myeclipse2014 作为 Java 语言的主要开发工具,其主要的一个特点就是开源免费,不同于其他的 Java 开发工具,Myeclipse2014 是一个纯开源的开发平台,可以根据开发者的不同,进行不同开发插件的集成。

2、Myeclipse2014 的起始界面更为友好，给初学者提供了很好的引导，还增加了自我学习的功能。

3、将 Windows 应用程序和 Web 应用程序明显的提了出来，Myeclipse2014 中提供了全新的网页设计器（称为 Visual Web Developer）。可见 Myeclipse2014 对 Web 应用方面的重视，Java 技术正得到不断的发展，一些新技术正被广泛宣传和推广。

2.1.1 Tomcat 平台简介

Tomcat 是一个免费的开源的 Serlvet 容器，它是 Apache 基金会的 Jakarta 项目中的一个核心项目，由 Apache，Sun 和其它一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 Jsp 规范总能在 Tomcat 中得到体现。Tomcat 被 JavaWorld 杂志的编辑选为 2001 年度最具创新的 java 产品，可见其在业界的地位。由于 Java 的跨平台特性，基于 Java 的 Tomcat 也具有跨平台性。

2.1.2 Java 简介

Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 程序设计语言（以下简称 Java 语言）和 Java 平台的总称。用 Java 实现的 HotJava 浏览器（支持 Java applet）显示了 Java 的魅力：跨平台、动态的 Web、Internet 计算。从此，Java 被广泛接受并推动了 Web 的迅速发展，常用的浏览器现在均支持 Java applet。另一方面，Java 技术也不断更新。

2.1.3 MySQL 简介

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗

下公司。MySQL 最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。

MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

2.1.4 Spring MVC 框架

Spring MVC 属于 SpringFrameWork 的后续产品，已经融合在 Spring Web Flow 里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用 Spring 进行 WEB 开发时，可以选择使用 Spring 的 SpringMVC 框架或集成其他 MVC 开发框架，如 Struts1，Struts2 等。

2.1.5 Spring Boot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot 致力于在蓬勃发展的快速应用开发领域（rapid application development）成为领导者的。

第三章 系统设计与需求分析

3.1 系统的定位

人们在虚拟社区中，借由彼此的互动沟通，渐渐了解和信赖对方，而互动和沟通的基础，建立在人们在兴趣、人际关系、幻想和交易四大基本需求之上。

微博作为社交性新闻传播，是新闻传播的“社会化”与“个性化”有机的结合。一方面使具有公共价值的新闻更容易在微博平台上广泛传播．新闻的针对性更强了，体现微博作为社会形新闻平台的魅力；另一方面每个人所关注的对象也不见得相同，获取的信息也有所差异，也就是说个人的杜交圈子与交往圈子决定了他获取信息的范围、数量。微博作为新型的传播平台有鲜明的个性化特点。

3.2 系统的总体规划

首先用户是我们产品得以成长的基石，我们的服务对象是大众，我们希望给大众提供更方便更快捷的信息共享平台，让用户能够随时随地了解身边发生了什么，同时让用户成就我们的未来，所以我们的产品服务的群体就是普通的用户群体只要你对社交感兴趣，你就可以成为我们的产品使用者，也就是我们服务的对象。

3.3 网站设计目标

3.3.1 普通用户

1. 用户注册以及密码的盐加密

这部分主要功能是实现用户注册，但是普通的加密方式存在安全隐患，数据库一旦出现泄漏或者被盗，用户的密码很容易被破解，本次项目通过使用加盐加密方式在一定程度上来解决这个问题，从而提高用户信息的安全性，保证用户的密码不会轻易被窃取。

2. 站内信的发送

站内信的功能主要是为方便用户信件或者消息往来而设置的服务功能，它主要服务于用户与用户的信息沟通，为用户与用户之间消息的传递起到很好的连接作用，另外在用户与系统消息的传递中起到一个传递作用，从而方便用户对信息的收集和管理。

3. 评论中心

用户可以评论微博，同时用户也可以对微博下的评论信息进行评论，今后可能还会对其他元素或者内容进行评论，因此需要做出一种合理的设计，方便开发及扩展。

3.3.2 管理员

1. 对微博的管理

管理模块中微博的管理需要拥有管理员权限的用户才能对此部分进行操作，管理员可以查看所有的微博信息，并且对其进行维护，包括对用户发布的对不良微博信息的删除等。

2. 对用户的管理

管理模块中用户的管理同样需要拥有管理员权限的用户才能对此部分进行操作，管理员可以查看所有用户的信息，以及该用户的相关动态，管理员可以对违法发布不良信息的用户进行禁言或者注销用户信息。

3. 对评论的管理

管理模块中评论的管理同样也需要拥有管理员权限的用户才能对此部分进行操作，管理员可以查看所有的评论信息，并且对评论信息进行删除操作等。

3.4 功能模块图

本次设计的功能模块图如图 3.4，其包括两大功能模块，具体包括普通用户模块、管理员模块，下面分别简单介绍一下这两个模块所包含的子模块的功能，普通用户模块是呈献给大众来使用的，这个模块会有很漂亮的界面展示，用户选择登录注册，也可以方便的接收管理员的通知消息，增加了信息的流通，评论模块用户可以对微博进行评论，同时用户可以对已有的评论进行再评论，增加了微博的互动性，也提高了用户的交互体验。管理员模块是给系统维护人员提供的，该模块普通用户没有权限进入，只有管理员才可以进入该模块进行操作，管理员可以对微博进行管理，对普通用户进行维护，以及对已有的微博的评论进行维护。

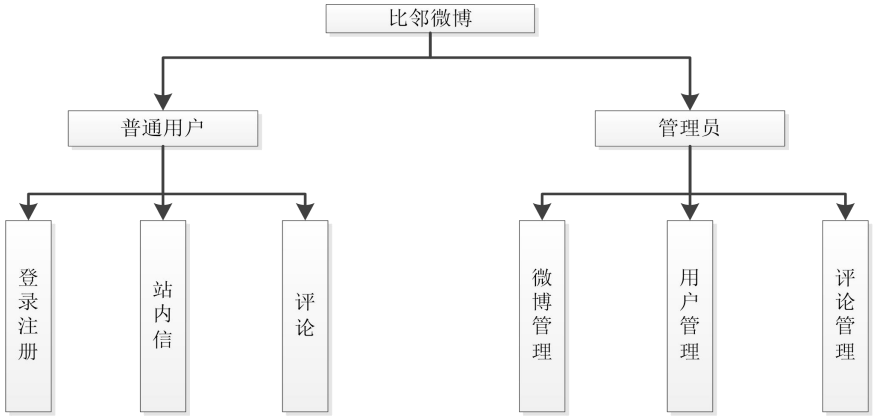

图 3.4 功能模块图

Fig. 3.4 Functional block diagram

3.5 系统用例图设计

3.5.1 普通用户模块用例图

如图 3.5.1 所示为普通用户对微博进行操作的用例图，用户打开微博的首页，然后选择登录或者注册功能，如果已经有账号，则可以直接登录进入系统，如果没有账号用户可以选择注册账号，注册完账号之后就可以进入本微博，在微博中普通用户可以浏览微博，或者发布自己的微博，同时用户可以在已经发布的微博下面进行评论，而且评论功能不局限于单条的用户评论，用户可以对已有的评论继续二次评论，或者回复评论的评论，另外站内信功能可以让用户和用户之间进行沟通，互相进行消息的传递，起到了以聊天的作用，另一方面用户可以利用站内信来接收管理员发送的一些通知等等。

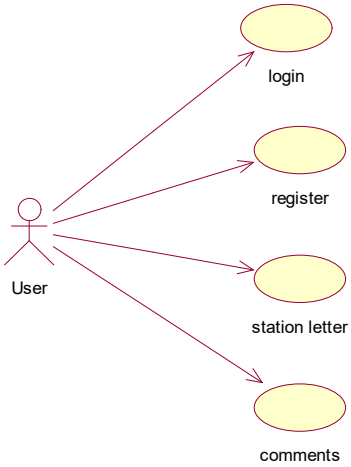

图 3.5.1 普通用户模块用例图

Fig. 3.5.1 Common user module use case diagram

如表 3.5.1 所示为普通用户的用例说明，该用例的范围是普通用户，前置条件是用户打开微博首页，也就是说用户能够成功的进入我们的微博首页，触发事件是用户能够正常登陆微博，然后就是主事件的操作步骤介绍，用户按照步骤一步步操作，如果遇到扩展事件或者错误时间系统都会给用户以提示用户具体的出错位置，这样能够很明确的让用户知道是自己的操作不当引起的错误还是由于系统原因引起的错误。

表 3.5.1 普通用户用例说明

Table 3.5.1 Common user use case description

范围	普通用户	
前置条件	用户打开微博首页	
触发事件	用户正常登录	
主事件流描述	步骤	活动
	1	系统提示用户输入账号和密码
	2	用户输入账号密码
	3	系统验证

		A1：验证失败 E1：无法识别用户身份
	4	用户进入微博
	5	微博显示备选项 B1：首页 B2：私信 B3：个人信息 B4：发布微博
	6	用户进行具体的操作
	7	用例结束
扩展事件 A1 描述	步骤	分支动作
	1	系统显示验证失败信息，并提示用户重新输入用户名和密码
	2	用户重新输入信息
	3	若连续三次验证失败，则拒绝访问
	4	若成功，返回主事件流第 4 步
错误流事件 E1 描述	步骤	分支动作
	1	系统显示无法识别用户身份
	2	返回主事件流第 2 步

3.5.2 管理员用例图

如图 3.5.2 所示为管理员模块的用例图，在该部分，只有具有管理员权限的用户才可以进入系统进行操作，普通用户不具备操作权限，管理员负责维护微博系统的信息的合法性，以及在发生数据混乱时对数据做及时的调整和恢复。

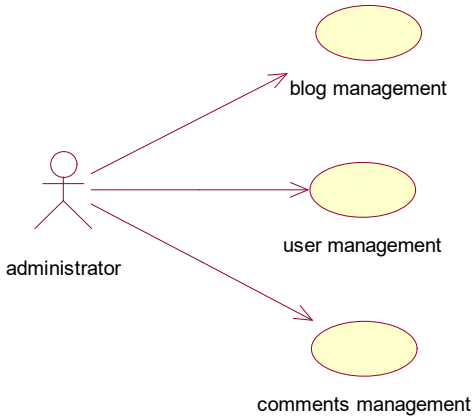

图 3.5.2 管理员用例图

Fig. 3.5.2 Administrator use case diagram

如表 3.5.3 所示为管理员用例说明的表格，在该表中清楚地说明了该用例的作用范围、前置条件、触发事件等的内容。从表只可以看出，该用例的作用范围是系统管理员，前置条件是管理员进入后台管理系统，触发事件就是管理员能够正常登录系统，也就是你的账户身份必须是管理员身份，否则的话是不能够进入该系统的，因为管理员具有很大的权限，主要负

责对系统的维护，如果普通用户具有管理员权限将可能因为误操作或者恶意操作对用户的数据信息的损失将会是毁灭性的。成功登录系统之后管理员既可以按照正常的流程来处理一系列的事件，如表的主事件流所示，如果遇到错误事件，系统同样会给予用户以提示。

表 3.5.2 管理员用例说明

Table 3.5.2 Administrator use case description

范围	系统管理员	
前置条件	管理员进入后台管理系统	
触发事件	管理员正常登录	
主事件流描述	步骤	活动
	1	系统提示管理员输入账号和密码
	2	管理员输入账号和密码
	3	系统验证 A1：验证失败 E1：无法访问管理员身份识别系统
	4	管理员进入管理系统主页
	5	系统显示备选项 B1：微博管理 B2：用户管理 B3：评论管理
	6	管理员进行具体的备选项操作操作
	7	用例结束
扩展事件 A1 描述	步骤	分支动作
	1	系统显示验证失败信息，并提示管理员重新输入用户名和密码
	2	管理员重新输入信息
	3	若连续三次验证失败，则拒绝访问
	4	若成功，返回主事件流第 4 步
错误流事件 E1 描述	步骤	分支动作
	1	系统显示无法访问管理员身份识别系统信息
	2	返回主事件流第 2 步

第四章 数据库设计

4.1 实体类类图

如图 4.1 所示为项目工程的部分类结构图，因为项目比较大，所以不能将所有的类添加进来

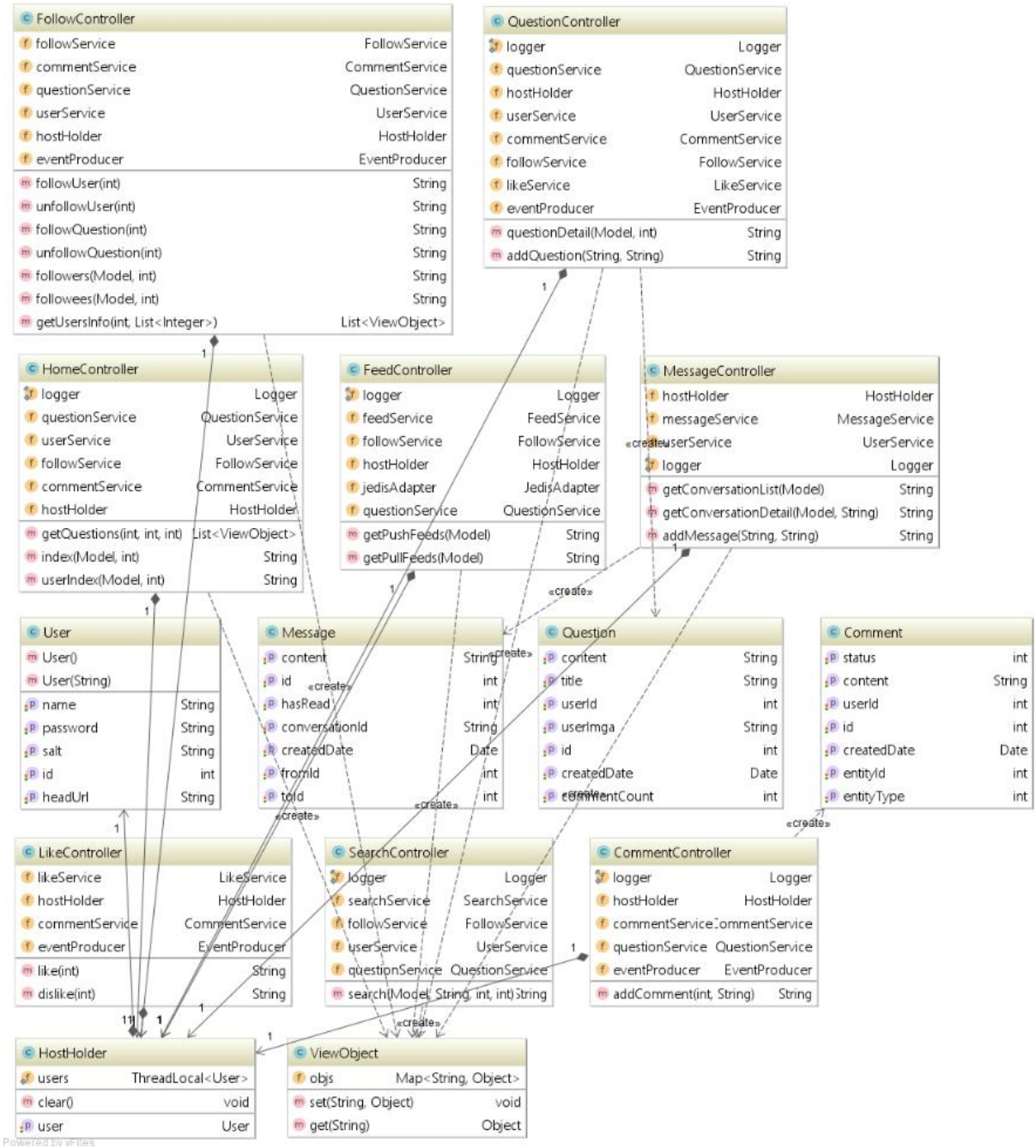

图 4.1 实体类图

Fig. 4.1 Entity class diagram

该类图是利用 IDE 附带的工具自动生成的，从图中可以简单地看出各个类之间的继承依赖组合关系，通过类图我们能够简单的了解项目的设计流程，以及项目的代码结构，对于我们了解项目的整体设计和架构有一个非常不错的帮助。

4.2 数据库表结构

4.2.1 USER 用户信息表

如表 4.2.1 所示为用户表主要用来存放用户的基本信息，包括了用户的用户名、密码、头像的重要信息，因为密码采用了盐加密的方式，所以即使直接打开数据表也不能直接获取到用户的密码，看到的仍然是加密后的密码，这样就为用户的信息安全做了一层保证。

表 4.2.1 用户表

Table 4.2.1 User table

字段名称	说明	功能	数据类型	长度
ID	用户 ID	唯一标识用户序列	int	11
NAME	用户名	唯一标识用户	varchar	64
PASSWORD	用户密码	用户密码，加密	varchar	128
SALT	加密字符	通过该字符进行加密	varchar	32
HEAD_URL	头像库链接	用户的头像	varchar	256

4.2.2 LOGIN_TICKET 登录记录表

如表 4.2.2 所示该表记录了用户的登录状态信息，以及用户的登录时间和退出系统时间，另外用户在登录系统之后会得到一个令牌，标识用户在登录状态一直有效其中 id 作为主键标识，user_id 与用户表进行关联可以知道是哪一个具体的用户，ticket 作为一个令牌，在登录期间一直有效，如果退出登录则令牌失效，expired 标识用户的登录时间，便于查询用户的登录记录，status 标识用户目前的登录状态。

表 4.2.2 登录记录表

Table 4.2.2 Log log

字段名称	说明	功能	数据类型	长度
ID	表序列	主键标识	int	11
USER_ID	用户 ID	唯一标识用户身份	int	11
TICKET	令牌	用户登录验证	varchar	45
EXPIRED	时间	记录用户登录时间	datetime	0
STATUS	状态	用户登录状态	int	11

4.2.3 评论信息表

如表 4.2.3 所示为评论表用来记录用户的评论信息，该表有七个字段，id 字段为表的主键，用来区别每一条记录，content 为评论的详细内容，user_id 标识那个用户进行的评论，entity_id

主要是与微博表进行关联用的，用来关联微博的 id，entity_type 作为预留字段标识微博的类型，created_date 表示评论的发布时间，可以让用户很清楚的知道该条评论是在何时进行发布的。

表 4.2.3 评论信息表

Table 4.2.3 Comment information table

字段名称	说明	功能	数据类型	长度
ID	教师 ID	主键标识	int	11
CONTENT	评论内容	评论内容信息	text	0
USER_ID	用户 ID	评论用户 ID	int	11
ENTITY_ID	发布微博 ID	发布的微博 ID	int	11
ENTITY_TYPE	微博类型	微博的类型	int	11
CREATED_DATE	创建时间	创建微博时间	datetime	0
STATUS	评论状态	评论的状态	int	11

4.2.4 站内信表

如表 4.2.4 所示为站内信记录了用户与用户之间的交流信息，以及系统管理员与用户之间的一些信息交流或者通知，该表包含了七个字段，其中 id 作为主键标识，from_id 标识信件的来源方是谁，该值与 user 表中的 id 值进行关联，to_id 和 from_id 的含义相似，标识接收方的用户是谁，与 user 表进行关联。Content 表示信件的内容。

表 4.2.4 站内信表

Table 4.2.4 Station letter

字段名称	说明	功能	数据类型	长度
ID	主键	主键标识	int	11
FROM_ID	信件来源	发件方 ID	int	11
TO_ID	接收方 ID	接收方 ID	int	11
CONTENT	信件内容	信件内容	text	0
CREATED_DATE	发送时间	发件时间	datetime	0
HAS_READ	是否阅读	是否已查收	int	11
CONVERSATION_ID	来源和去向连接	来源和去向连接	varchar	45

4.2.5 微博表

如表 4.2.5 所示为微博表用来存储用户发表的微博信息，该表具有六个字段，id 同其他标的设计结构类似，作为主键标识，title 标识用户的发出的微博标题，content 表示用户的正文内容，user_id 与用户表进行关联，表示是那个用户发送的微博，这样就很清楚的标识出了这个微博是属于谁的，created_date 标识微博的发表时间，便于用户进行查看或者记录自己的动态，可以相当于一个日记的功能，comment_count 标识该条微博含有多少条评论，如果用户删除了自己的评论，那么该条数目就会递减，管理员同样具有删除评论的权限。

表 4.2.5 微博表

Table 4.2.5 Micro-blog table

字段名称	说明	功能	数据类型	长度
ID	主键	唯一标识	int	11
TITLE	标题	微博标题	varchar	255
CONTENT	内容	微博内容	text	0
USER_ID	用户 ID	用户 ID	int	11
CREATED_DATE	创建时间	创建时间	datetime	0
COMMENT_COUNT	评论数	评论数量	int	11

4.3 数据库表关系

本次微博的数据库设计主要采用了一对一的关心进行，没有采用多对多和一对多的数据库关系，我们将多对多的数据关系进行了了拆分，而且在表结构的设计上，尽可能的达到了 3NF 的设计，降低冗余和传递依赖，尽可能的产生中间关联关系表。当然数据库的设计上还是存在缺陷的，可能只是目前阶段没有发现的一些问题，希望在不断迭代的过程中进行弥补这些缺陷，然后进行弥补，以求达到精准无误。

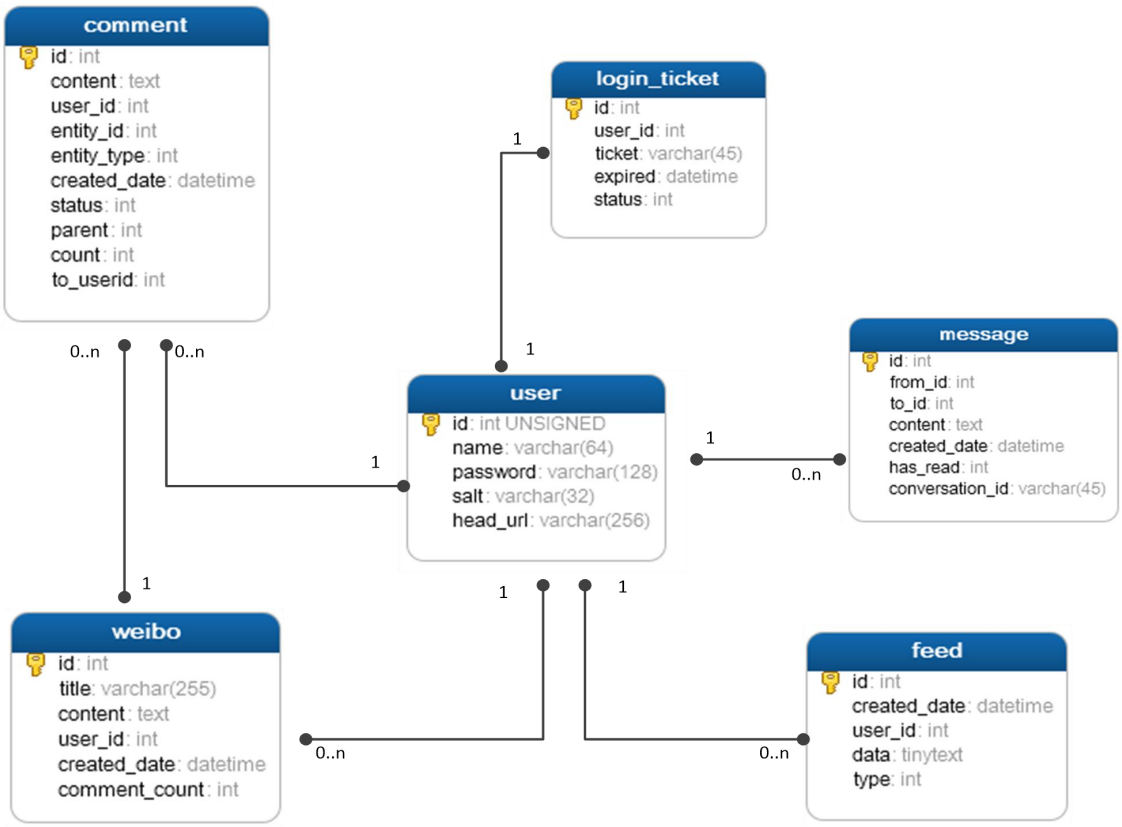

图 4.3 数据库表关系

Fig. 4.3 Database table relationships

第五章 系统详细设计

详细设计阶段的根本目标是确定应该具体地实现所要求的系统，也就是说，经过这个阶段的设计工作，应该得出对目标系统的精确的描述，从而在编码阶段可以把这个描述直接翻译成用某种程序设计语言书写的程序。

要保持程序分层设计的优点，就必须坚持层间的松散耦合关系。设计程序前，应先划分出可能的层次，以及此层次提供的服务和需要的服务。程序设计时，应尽量保持层间的隔离，只是使用层归档提供的服务。

优点[1]：

1、开发人员可以只关注整个结构中的其中某一层；

2、可以很容易的用新的实现来替换原有层次的实现；

3、可以降低层与层之间的依赖；

4、有利于标准化；

5、利于各层逻辑的复用。

概括来说，分层式设计可以达至如下目的：分散关注、松散耦合、逻辑复用、标准定义。

缺点:

“金无足赤，人无完人”，分层式结构也不可避免具有一些缺陷：

1、降低了系统的性能。这是不言而喻的。如果不采用分层式结构，很多业务可以直接造访数据库，以此获取相应的数据，如今却必须通过中间层来完成。

2、有时会导致级联的修改。这种修改尤其体现在自上而下的方向。如果在表示层中需要增加一个功能，为保证其设计符合分层式结构，可能需要在相应的业务逻辑层和数据访问层中都增加相应的代码。

关于第一个缺点，完全可以通过系统的缓存机制来减小对性能的影响。第二个缺点，我想只能通过采用一些设计模式来得到改善吧。

5.1 登录注册功能

5.1.1 登录注册顺序图

如图 5.1.1 所示为登录注册是顺序图，在该图中总共有六个模块，分别是用户，前端的展示页，一个控制层用来进行前端页面和后台交互的控制器，接下来是业务处理的 service 层，用户输入的信息从控制层传到业务层，会在该业务曾经进行相应的处理，然后在 DAO 层会进行数据库的操作，比如验证用户的身份是否合法，以及用户的密码是否正确或者用户是否存

在于本数据库系统中，最后数据库给予查询的结果，又通过反向的过程反送给前台页面，页面呈献给用户。

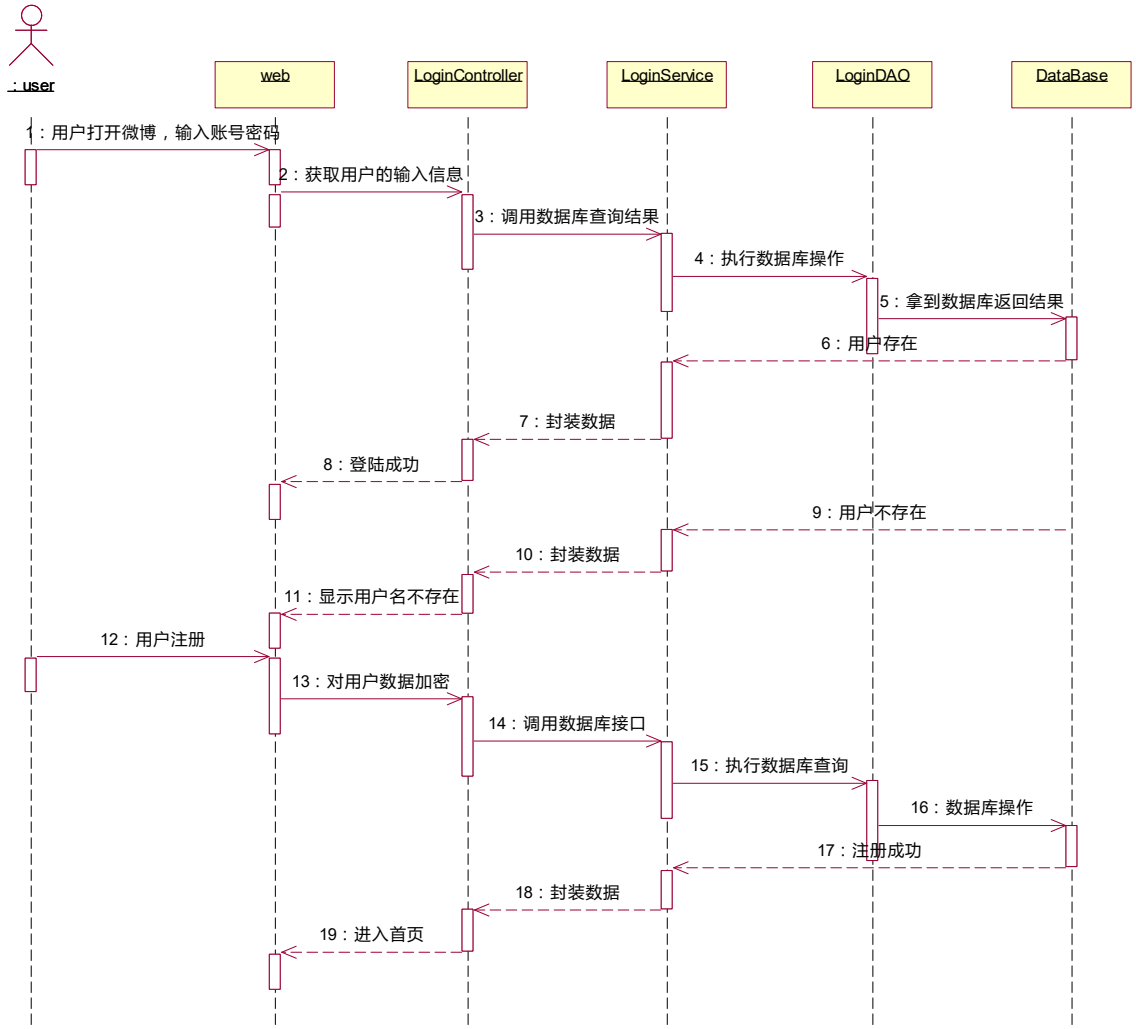

图 5.1.1 登录注册顺序图

Fig. 5.1.1 Login registration sequence

如表 5.1.1 所示为顺序图的基本说明，登录注册模块是整个项目的一个入口，如果登录注册功能做不好，对用户的个人信息是一个很致命的危害，会导致用户的信息外泄，一旦用户的信息外泄，将直接导致用户对我们的微博的安全性进行质疑，所以也就不会再相信我们的产品，所以登录注册功能的好坏在一定程度上影响了产品的用户的存留，在登录注册模块，我们使用了盐加密的方式对用户的账户密码进行了加密，这样即使被人攻陷我们的数据库，他们也不会获得用户的直接密码，所以在一定程度上对用户的信息进行了安全保证。用户打开我们的登录注册页面，如果已经拥有了我们系统的账号，则直接登录系统即可。

表 5.1.1 登录注册描述

Table 5.1.1 Login description

名称	类型	描述
Web	Jsp 页面	登录注册页

LoginController	控制类	与前台页面进行交互
LoginService	业务实现类	登录注册逻辑实现
LoginDAO	数据库操作类	用户信息数据库操作类
DB	数据库	用于数据存放

5.1.2 用户注册登录流业务程图

如图 5.1.2 所示为登录注册的业务流程图，如图所示，用户进入我们的系统首页，如果用户已经有了我们的微博账号，直接登录系统即可，如果没有我们的账号，用户则需要去注册

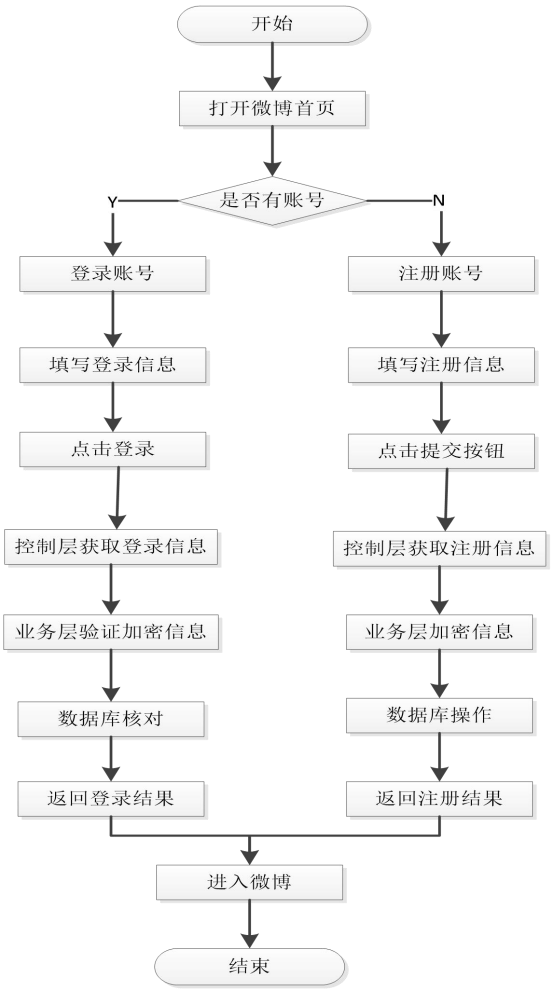

图 5.1.2 用户登录注册业务流程图

Fig. 5.1.2 User login registration business flow chart

一个，注册的时候，用户填写注册的基本信息，然后点击注册，这时候我们的后台会与前端进行交互，拿到前端的填写数据，然后对数据做基本的处理，因为注册信息中含有用户的密码等敏感信息，所以业务层需要对用户的密码进行加密处理，然后数据库操作层会对用户的信息进行查重，如果数据库中已经含有该用户的注册信息，则会提示注册失败，如果没有，数据库会插入该条记录，然后给前台返回注册结果，最后用户就会成功的进入微博系统。

5.2 站内信功能

5.2.1 站内信顺序图

如图 5.2.1 所示为站内信的顺序图，在该模块中同样也有六个板块，用户也是属于普通用户，普通用户进入微博，然后填写相应的站内信信息，这时候点击发送会出发控制层的逻辑，控制层接收到前端界面传递过来的信息，然后进一步传递到业务层，业务层拿到数据进一步处理，将处理结果发送到 DAO 层，DAO 层拿到数据进行数据库操作，数据库进行核对信息的合法性或者检测信息是否满足条件，然后插入数据库，然后将操作结果返回给 DAO，DAO 通过简单处理将数据给业务层，最后通过控制层返回给前台界面，前台界面接收到数据显示给用户操作是否成功。

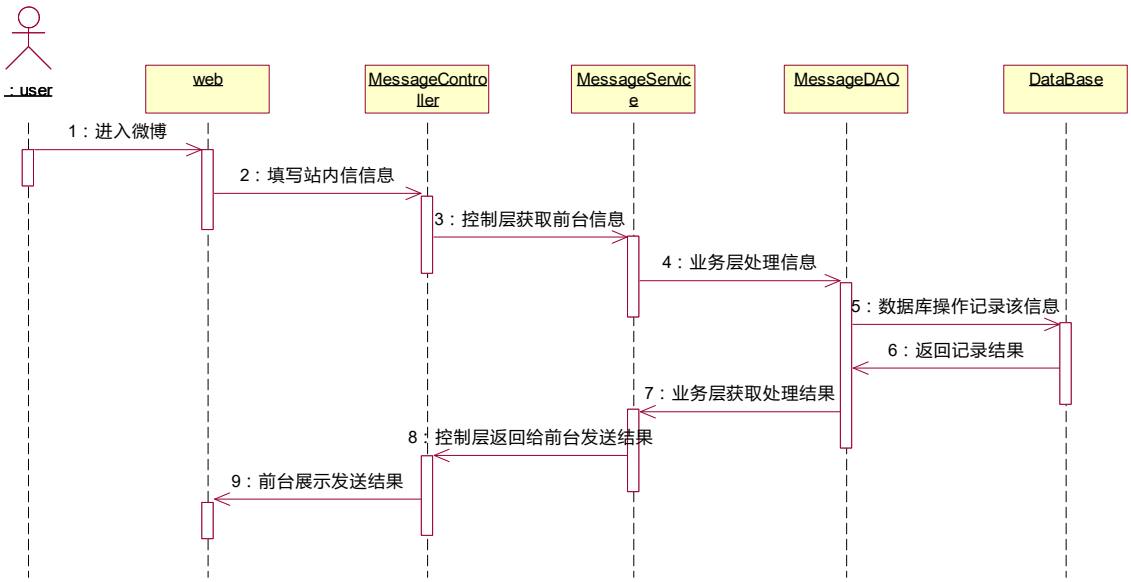

图 5.2.1 站内信顺序图

Fig. 5.2.1 Station letter sequence diagram

如表 5.2.1 所示为该顺序图的简单表结构，该模块主要是用户与用户之间的沟通，以及系统管理员给用户的一些重要通知，用户进入微博主页，点击私信，这时候会弹出用户的历史聊天记录，然后点击发送信息，会弹出一个小的对话框，用户填写需要发送的信息，指定人称，点击发送，这时候控制层会获取到前端界面的填写信息，然后业务层会去处理这些信息，数据库曾检查该信息的收信人是否存在，如果用户不存在，提示给用户相应的提示，如果存在，则投递到用户的 message 队列中，接收方在登录系统之后或者刷新页面之后就能够看到信息，与此同时，发件方的历史记录中也会记录该条信息，收件方可以选择回复也可以选择不回复等处理操作，另外系统管理员员可以通过私信给用户一些重要的提示，比如某某关注了您，或者你的微博有了新的动态等等之类的信息。

表 5.2.1 站内信描述

Table 5.2.1 Station letter description

名称	类型	描述
Web	Jsp 页面	微博主页
MessageController	控制类	与前端进行交互
MessageService	业务实现类	对信息进行处理
MessageDAO	数据库操作类	对数据库进行操作
DB	数据库	用于数据存放

5.2.2 站内信业务流程图

图 5.2.2 所示为站内信的设计流程图，前置条件是用户成功登陆进入系统，然后点击主页找到私信，此时会弹出对话框，用户需要填写收信人，以及发送的内容，点击发送私信，在填写完信息之后，并且发送信息，后台的控制层会与前台进行交互，拿到前台填写的信息，并进行检查信息的合法性，与此同时，会检测收信人是否存在，如果收信人不存在，则发送失败，并给前台以提示，如果用户存在，则投递到对方的消息队列中，用户上线之后或者在线的用户刷新页面就能够看到个人私信里面的内容。

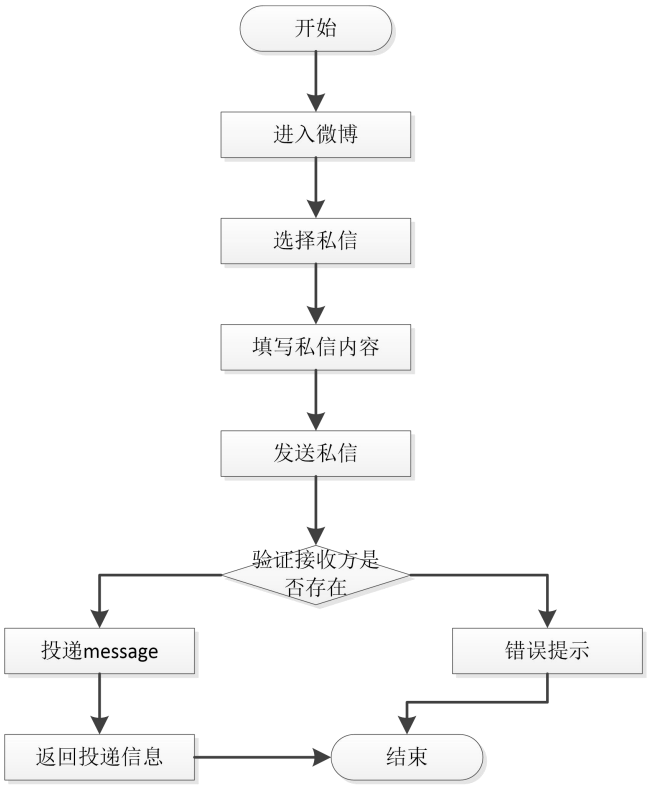

图 5.2.2 站内信业务流程图

Fig. 5.2.2 Station business flow chart

5.3 评论功能

5.3.1 评论顺序图

如图 5.3.1 所示为评论模块的顺序图，在该部分同样有六部分模块，分别是普通用户，普通用户选择一条微博，并进入微博的详情页，然后就能看到评论模块，填写评论内容，点击评论，这时候控制层就会获取到评论的内容发送给 service 层，业务层处理数据，接下来 DAO 会拿到这个数据，进行数据库的相关操作，插入数据库，出入评论完成之后会返回给 DAO 结果，DAO 将数据返回给业务层，业务层又把数据返回给控制层，控制层发送给前端界面，前端界面解析数据并最终显示评论的内容。

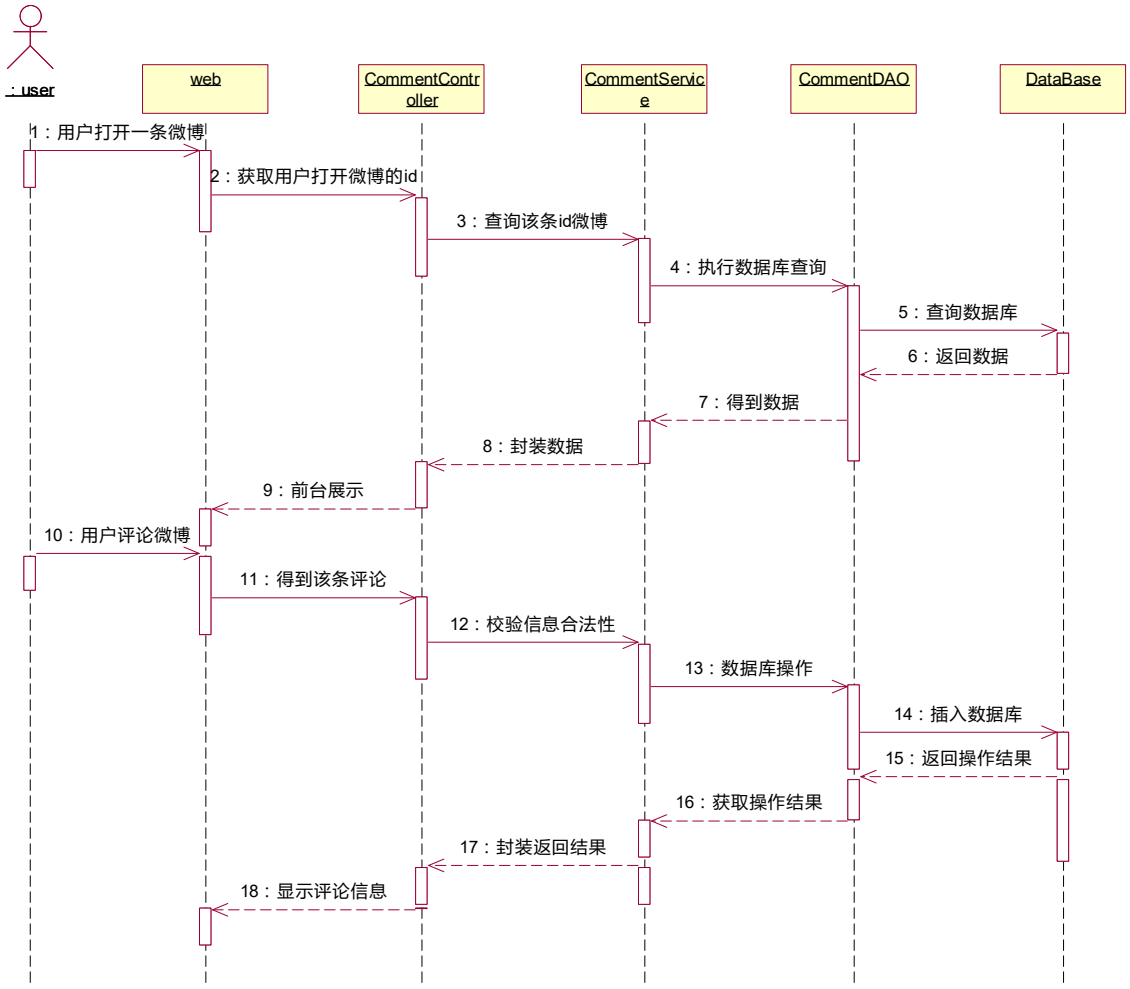

图 5.3.1 评论顺序图

Fig. 5.3.1 Comment sequence diagram

如表 5.3.1 所示为评论顺序图的描述，在该模块，用户进入微博首页，成功登陆，然后会展示多条微博，用户点击任意一条微博，会进入该微博的详情页，在该详情页，用户可以对该微博的信息进行评论，评论是任意的，用户填写评论之后，后台的控制层会获取前台的填写信息，然后在业务层进行数据的合法性的检验，然后数据库操作会把该条评论信息插入数

据库表，再次加载该条微博的时候，会附带这条评论信息显示，除此之外，用户还可以对已有的评论进行二次评论，用户选择需要的评论，点击旁边的回复按钮，会弹出回复的界面，用户填写评论，然后点击评论。评论完成之后会加载用户的评论信息，用户就可以看到已经评论的内容。

表 5.3.1 评论描述

Table 5.3.1 Comment description

名称	类型	描述
Web	Jsp 页面	微博的单项展示页
CommentController	控制类	与前台交互的类
CommentService	业务实现类	评论业务实现类
CommentDAO	数据库操作类	评论数据库操作类
DB	数据库	用于数据存放

5. 3. 2 评论功能业务流程图设计

图 5.3.2 所示为评论功能的业务流程图，用户需要成功登陆系统，点击任意一条微博，进入该条微博的详情页，在这里你会看到该条微博的详细信息，发布人，发布时间，同时你可以关注该条微博，如果该条微博有任何的更新或者评论等信息，你的站内信将会收到通知。用户填写评论，点击评论按钮，就可以将你的精彩评论发布上去，你的评论将会对所有人可见，所以别人也可以对你的评论进行二次评价，用户评价完你的评论之后，你的私信将会收到通知，可以点击跳转到微博详情页，查看更多的回复信息，同时也可以在私信里面查看回复的具体内容。

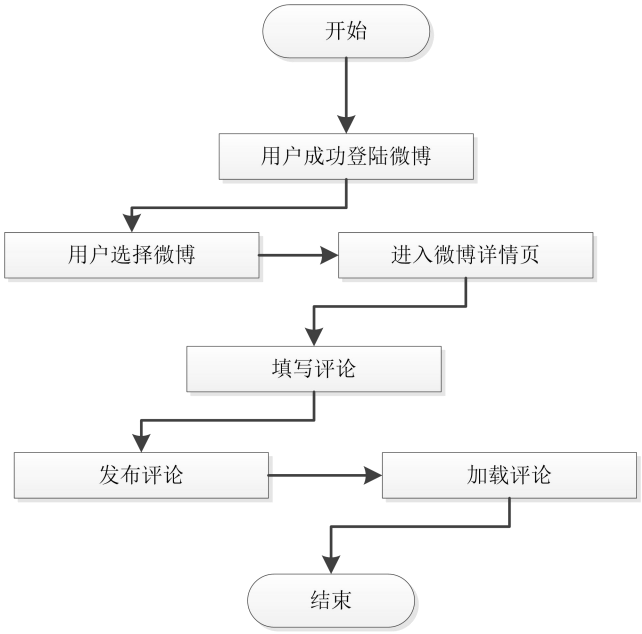

图 5.3.2 评论功能业务流程图

Fig. 5.3.2 Functional business flow chart

第六章 编码和测试及运行效果

6.1 测试的基础

6.1.1 测试的目标

测试的根本目标就是在软件投入生产性运行之前，尽可能多地发现软件中的错误，最终给用户提供具有一定可信度质量的软件，目前软件测试是对软件规格说明、设计和编码的最后复审，仍是保证软件质量的关键性步骤。在谈到软件测试时，许多人都引用 Grenford J. Myers 在《The Art of Software Testing》一书中的观点：

1．测试是为了发现程序中的错误而执行程序的过程[2]；

2．好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案；

3．成功的测试是发现了至今为止尚未发现的错误的测试。

6.1.2 测试的方法

软件测试有两种方法：白盒法和黑盒法。

黑盒测试法是在知道产品应该具有的功能后，通过测试来检测是否每个功能都能实现的测试方法；白盒测试法是在知道产品的内部工作过程时，通过测试来检验是否按照规格说明说的规定正常运行的方法。

粗看起来，不论采用上述那种测试方法，只要对每一种可能的情况都进行测试，就可以得到完全正确的程序。包含所有可能情况的测试称为穷尽测试，对于实际程序而言，穷尽测试通常是不可能做到的。使用黑盒测试法为了做到穷尽测试，要对的进行测试的所有输入数据进行各种可能值的排列组合，然而，由此得到的应该测试的情况，数字往往达到实际上根本无法测试的程度。实践表明，用无效的输入数据进行测试往往能比有效的输入数据发现更多的错误。但是使用白盒测试法和使用黑盒测试法一样也不可能做到穷尽测试。

6.1.3 测试的步骤

在实现组将验证所开发的程序后，交至测试组，由测试组的相关工作人员进行测试，测试一般有以下几个步骤：

1．测试人员要仔细阅读有关资料，包括设计文档、规格说明、测试大纲、使用说明、、测试内容及测试的通过准则，做到整体了解系统，然后写测试的计划，测试用例，为测试做充足的准备。

2．为了保证测试的质量，将测试过程分成几个阶段，即：代码审查、单元测试、集成测

试和验收测试。

3．代码会审：代码会审是由一组人通过阅读、讨论和争议对程序进行静态分析的过程。会审小组在充分阅读待审程序文本、控制流程图及有关要求、规范等文件基础上，召开代码会审会，程序员逐句讲解程序的逻辑，并展开热烈的讨论甚至争议，以揭示错误的关键所在。实践表明，程序员在讲解过程中能发现许多自己原来没有发现的错误，而讨论和争议则进一步促使了问题的暴露。

经过上述的测试过程对软件测试后，软件基本满足开发的要求，测试任务完成，验收完毕后，便可以将软件发送至用户。

6.2 测试用例设计

6.2.1 后台登陆模块测试用例

测试条件：管理员(UN)AND 密码(PW)

其约束形式为：D（D1，D2）；

其处理方式为：(管理员(UN)AND 密码(PW))

约束集合为{(t,t),(f,f),(t,f), (f,t)}

白盒测试：

测试案例 1：UN=Admin，PW=Admin，(t,t)

测试案例 2：UN=admin，PW=admin，(f,f)

测试案例 3：UN=Admin，PW=admin，(t,f)

测试案例 4：UN=123456，PW=“Admin”，(f,t)

黑盒测试：

运用等价分类法划分等价类补充用例，有效等价类如测试案例 1；无效等价类如测试案例 2（有效等价类、无效等价类较多，此处只举一例）然后用猜错法附加用例。

测试案例 5：UN=%￥#，PW=Admin（无效等价类）

测试案例 6：UN=Admin，PW=Admin（有效等价类）

测试案例 7：UN=“空”，PW=“空”（猜错法）

测试案例 8：UN=张三，PW=zhangsan，C=KRAW（猜错法）

测试案例 9：UN=Admin，PW=“空”，C=“空”（猜错法）

测试案例 10：UN=Admin，PW=Admin，C=“空”（猜错法）

如表 6.2.1 所示为管理员登录模块的测试用例的测试结果分析，以及各个测试用例的详细测试，测试的预计产生行为，和实际测试的结果，另外附带失败原因。

表 6.2.1 登陆模块测试用例

Table 6.2.1 Login module test case

案例	应产生行为	实际测试结果	失败原因
1. UN=Admin，PW=Admin，	登录后台系统	成功	
2. UN=admin，PW=admin，	提示用户名和密码错误	成功	
3. UN=Admin， PW=admin，	提示用户密码为空	成功	
4. UN=123456， PW=Admin，	提示用户用户名不存在	成功	
5. UN=%￥#，PW=Admin，	提示用户用户名不正确	成功	
6. UN=Admin，PW=Admin，	登录后台系统	成功	
7. UN=“空”，PW=“空”	提示用户输入用户名、密码	成功	
8. UN=张三，PW=zhangsan，	提示用户只允许管理员登陆	失败（只是提示用户名密码错误）	未能细化错误原因只是笼统的提示用户登录失败
9. UN=Admin，PW=“空”	提示用户输入密码	成功	
10. UN=Admin，PW=Admin	提示用户输入验证码	成功	

6.2.2 站内信测试用例

站内信发送条件：用户名(USER)AND 消息内容(MSG)

其约束形式为：D（D1，D2）；

其处理方式为：(用户名(USER)AND 消息内容(MSG))

约束集合为{(t,t),(f,f),(t,f), (f,t)}

白盒测试：

测试案例 1：USER=101，MSG=测试消息，(t,t)

测试案例 2：USER=1001，MSG=null，(f,f)

测试案例 3：USER=101，MSG=null，(t,f)

测试案例 4：USER=1001，MSG=测试消息，(f,t)

黑盒测试：

运用等价分类法划分等价类补充用例，有效等价类如测试案例 1；无效等价类如测试案例 2（有效等价类、无效等价类较多，此处只举一例），然后用边界值法和猜错法附加用例，

测试案例 5：USER=%￥#，MSG=测试消息（无效等价类）

测试案例 6：USER=101，MSG=测试消息（有效等价类）

测试案例 7：USER=null，MSG=null（猜错法）

测试案例 8：USER=1011，MSG=测试消息（猜错法）

测试案例 9：USER=101，MSG=null（猜错法）

如表 6.2.2 所示为站内信测试用例详细数据，在该表中包括了测试的各种情况，也就是案例，以及预估机会发生的行为或者预估记得操作结果，以及实际产生的操作的结果，另外对操作的结果进行了定性的分析。

表 6.2.2 站内信测试用例

Table 6.2.2 Station test case

案例	应产生行为	实际测试结果	失败原因
1. USER=101，MSG=测试消息	提示发送成功	成功	
2.USER=1001，MSG=null	提示用户名不存在	用户不存在 成功	
3. USER=101，MSG=null	提示请输入私信内容	请填写私信内容 成功	
4.USER=1001，MSG=测试消息	提示用户名不存在	用户不存在 成功	
5.USER=%￥#，MSG=测试消息	提示用户名不存在	用户不存在 成功	
6.USER=101，MSG=测试消息	提示发送成功	成功	
7.USER=null，MSG=null	提示用户名不存在	用户不存在 成功	
8.USER=1011，MSG=测试消息	提示用户名不存在	用户不存在 成功	
9.USER=101，MSG=null	提示请输入私信内容	请填写私信内容 成功	

第七章 运行效果

整个项目基本上到此告一段落，整个项目开发耗时两个月半，开发工具使用了 Git 所以在代码的管理上还算项目当方便，而且对于两个人不能够在一起同使开发也起到了协同帮助作用，耗费人力二人，整体功能基本实现，与预期效果基本相符，而且在 UI 的设计上基本上不算特别的难看，使用了当前比较流行的前端框架，唯一的不足就是在界面的设计上存在缺陷，由于没有专业的美工人员，所以界面部分看起来并不是相当的完美，下面将整个项目的运行效果进行截图。

7.1 管理模块

7.1.1 整体树结构

如图所示，这是整体的项目构架的树形结构，从图中可以看出，项目分为三个模块，分别为图表统计、用户列表、微博列表，此次项目中我主要负责管理模块。微博统计主要是预留出来的一部分模块，为后续的舆情分析做扩展，用户列表展示当前的用户，微博列表展示当前所有的微博信息。管理员可以在后台管理界面方便的管理用户信息和微博信息。高效快捷方便。

图 7.1.1 项目的树结构

Fig. 7.1.1 Project tree structure

7.2 首页

如图 7.2 所示为管理系统的首页，在该首页包含了五部分内容，第一部分是三个引导按钮，该按钮可以直通相应的界面，分别展示响应的内容，接下来是微博统计，该统计模块是预留出来的功能，作为后面的舆情分析用，另外在图标旁边还有四个组件，可以看到当月增加的用户，以及总的用户数，还有总的微博数，在统计模块下面是热门微博的排行榜，点击该微

博可以进入微博的详情页，会在后续介绍，旁边是一个消息的展示页面，可以用来展示一些新闻或者站内通知之类的信息，作为预留模块，再最后一个模块是最新微博，最新微博展示最新发布的微博，同样点击进去会展示微博详情。

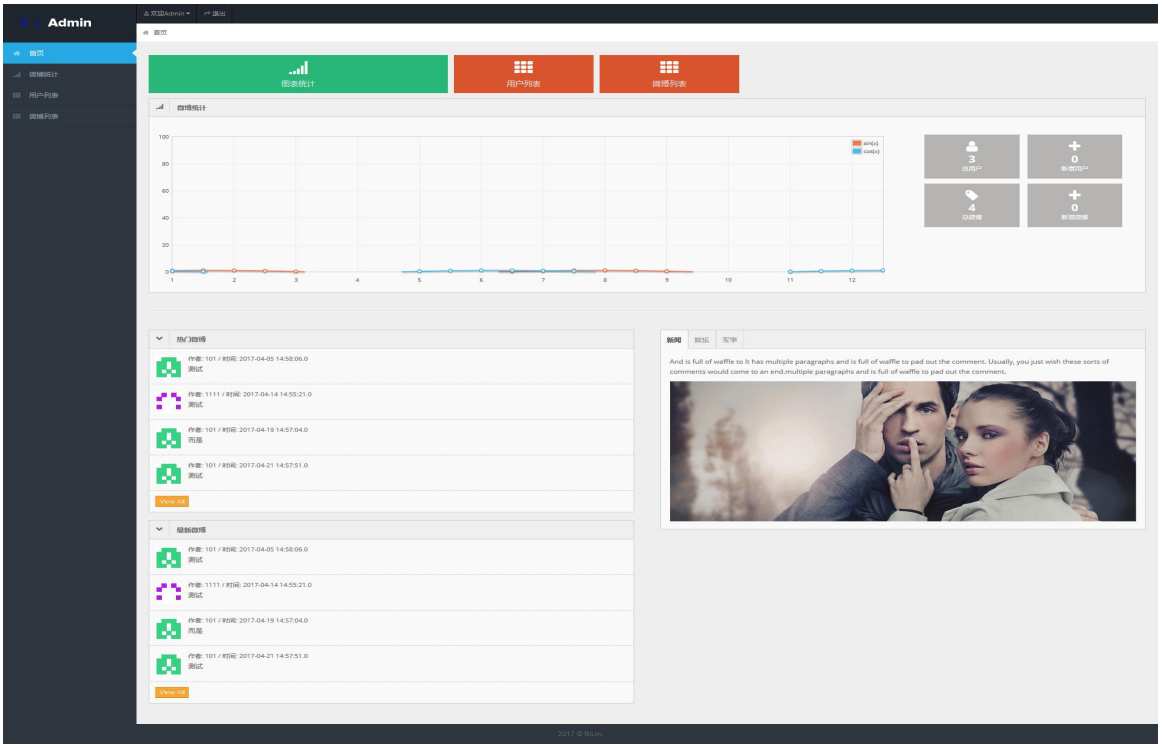

图 7.2 首页

Fig. 7.2 Home page

7.3 用户列表

如图 7.3.1 所示为用户信息列表的维护界面，该界面可以对用户的基本信息进行维护，包括向用户发送私信，删除用户信息，点击私信按钮之后会弹出私信对话框，界面如图 7.3.2 所示

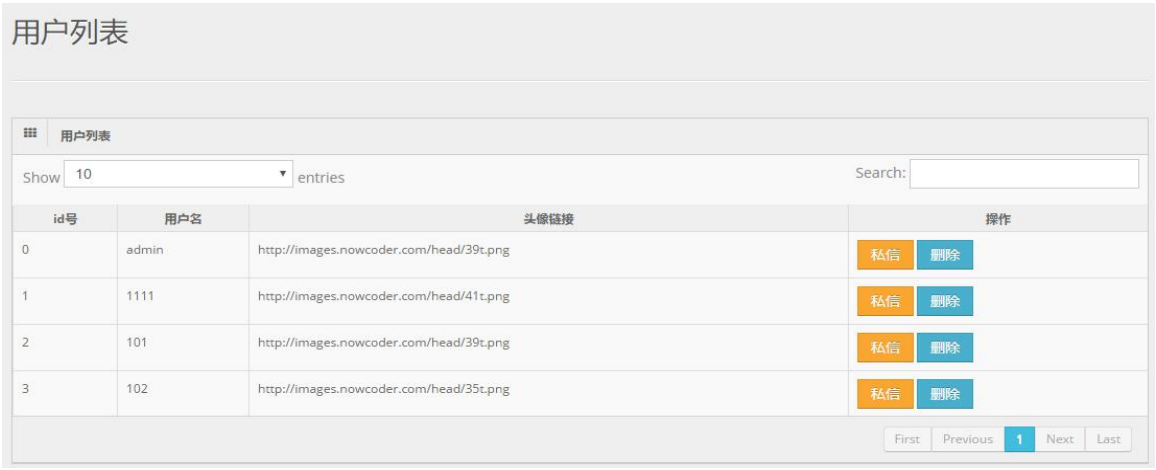

图 7.3.1 用户列表

Fig. 7.3.1 User list

在该界面清楚地标识了管理员需要向那个用户发送私信内容，并且管理员可以取消私信，也可以点击发送按钮进行发送私信。很方便管理员对普通用户的管理信息的通知，非常的高效快捷而又方便。

图 7.3.2 私信界面

Fig. 7.3.2 Private letter

图 7.3.3 所示为删除用户的界面，管理员点击删除按钮，弹出确认删除用户的界面，进行二次确认，这样做的好处是防止误删除操作，从一定程度上降低了操作的失误，提高了用户信息的安全保证。同时这样的提示给人的感觉非常的人性化，不会导致一键点击直接删除的移位情况发生，给用户留下了很多的选择余地。

图 7.3.3 删除用户

Fig. 7.3.3 Delete user

7.4 微博列表

如图 7.4.1 该界面管理员用来维已经发布的微博信息，管理员可以看到是哪些用户的微博，同时可以对微博进行删除操作，另外管理界面还可以设置每页显示多少条记录，可以尽可能的方便使用者，另外搜索按钮可以动态的搜索用户指定的内容，提高了工作效率，使用户不至于找不到信息而感到盲目，当然在微博量比较少的时候赶不到搜索的重要性，但是当微博数量达到了上千的时候，就能够很明显的体现出搜索功能的作用了，另外也可点击微博标题

进入该微博查看微博的具体信息。同时列表会显示微博的发布时间以及微博的评论数。点击一条微博进入查看如图 7.4.2 所示。

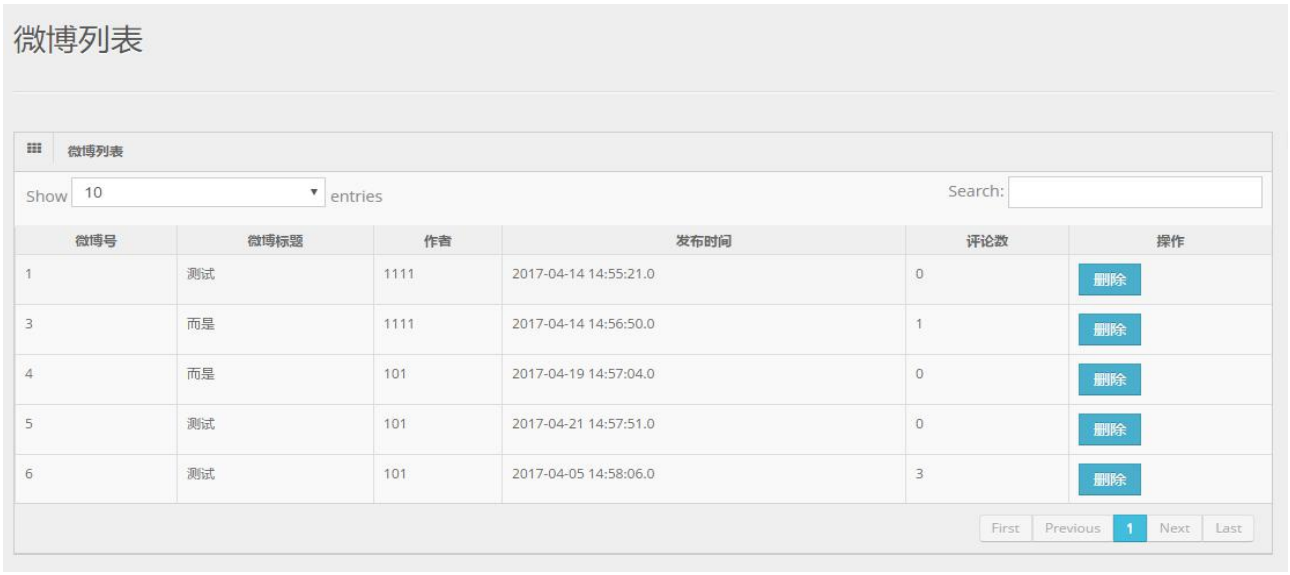

图 7.4.1 微博列表

Fig. 7.4.1 Micro-blog list

图 7.4.2 所示为微博详情界面，在该界面清楚地显示了微博的标题、作者、发布时间、以及微博内容，包括评论信息，评论信息包括了评论者，评论时间，评论内容，管理员可以对评论内容进行删除。

图 7.4.2 微博详情

Fig. 7.4.2 Micro-blog

以上就是微博的整个后台管理模块，很方便的一个后台管理功能，能够帮助管理员大大的减轻不必要的体力劳动，提高管理者的需求。让管理员不再因为繁琐的数据而感到焦虑和烦躁，面对纯粹的数据，在 UI 界面的情况下管理员面对的将不再是一堆乱数据，而是有漂亮的 UI

的数据，再剩下的就是前台的信息展示。

7.5 站内信

如图 7.5.1 所示为前台给用户呈现的内容，即站内信通知，该模块，用户会收到来自用户以及来自管理员的通知，在该界面会显示来自那个用户，以及包括哪些内容，信件的时间，及信件的个数等。

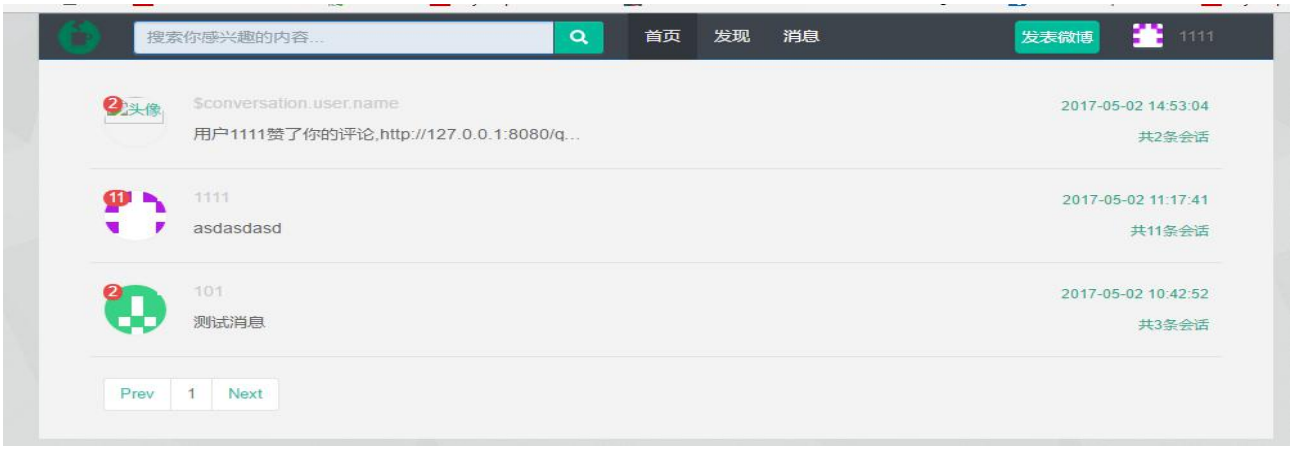

图 7.5.1 站内信

Fig. 7.5.1 Letter station

点击其中一条站内信，会进入站内信详情页，如图 7.5.2 所示，在该页面，显示了来自哪个用户的信息，包括了用户的头像，用户名，和用户发送的内容，通过分页组件，可以浏览下一页内容。

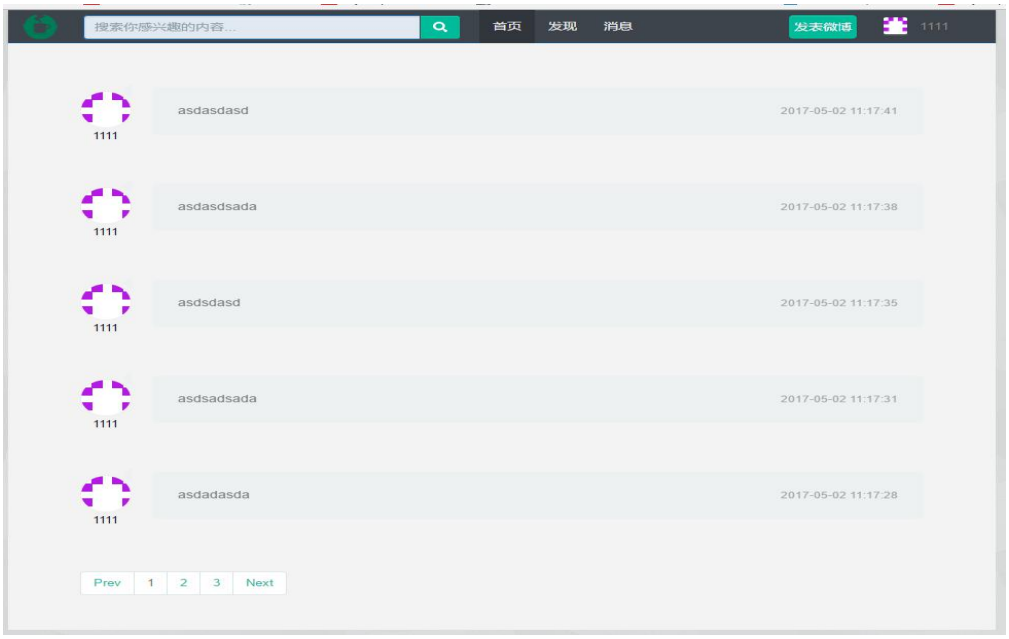

图 7.5.2 站内信详情

Fig. 7.5.1 Letter station infomation

第八章 代码实现

前面几章对整个项目的基本介绍已经基本上告一段落，包括项目的运行测试等都顺利完成，所以在本章要介绍的就是这些功能是如何通过代码来实现的，

由于项目中牵扯的功能过于多，所以我们只进行部分功能的代码解析，下面一站内信部分的代码为示例进行讲解。

站内信的总体设计上是通过前端界面用户填写站内信的收件人，并且填写信件内容，然后点击发送，这时候后台的控制层就会获取到前端界面用户填写的内容，然后将这部分内容发送给业务层。

8.1 站内信控制层代码

控制层的主要作用是起到一个桥梁作用，沟通前端和后台的一个作用，因为前端界面的用户填写信息必须通过一个桥梁之类的东西，将前端的信息传递给后台，控制层的作用就是起到这样的一个传递作用，很好的衔接了这两个独立的模块，方便了前后台的一个数据交换和数据展示。接下来是控制层的详细代码展示。

```
package com.hx.controller;
@Controller
public class CommentController {
    private static final Logger logger = LoggerFactory.getLogger(CommentController.class);
    @Autowired
    HostHolder hostHolder;
    @Autowired
    CommentService commentService;
    @Autowired
    QuestionService questionService;
    @Autowired
    EventProducer eventProducer;
    @RequestMapping(path = {"/addComment"}, method = {RequestMethod.POST})
    public String addComment(@RequestParam("questionId") int questionId,
                             @RequestParam("content") String content) {
        try {
            Comment comment = new Comment();
            comment.setContent(content);
```

```
                if (hostHolder.getUser() != null) {
                    comment.setUserId(hostHolder.getUser().getId());
                } else {
                    comment.setUserId(WeiBoUtil.ANONYMOUS_USERID);
                    // return "redirect:/reglogin";
                }
                comment.setCreatedDate(new Date());
                comment.setEntityType(EntityType.ENTITY_QUESTION);
                comment.setEntityId(questionId);
                commentService.addComment(comment);

                int count = commentService.getCommentCount(comment.getEntityId(),
comment.getEntityType());
                questionService.updateCommentCount(comment.getEntityId(), count);
                eventProducer.fireEvent(new
EventModel(EventType.COMMENT).setActorId(comment.getUserId())
                        .setEntityId(questionId));
            } catch (Exception e) {
                logger.error("增加评论失败" + e.getMessage());
            }
            return "redirect:/question/" + questionId;
        }
}
```

8.2 业务层代码

业务层的主要功能是将控制层获取到的时候拿过来，进行进一步的操作，其实业务层是一个传递信息的介质，他沟通的是 DAO 和控制层之间的数据交互，控制层拿到数据要将数据传给业务层，业务层再将数据发送给 DAO 层，作进一步处理，因为在项目中我的业务层的逻辑比较少，所以业务层的作用只起到了一个沟通的桥梁作用，没有起到他的实际作用。对业务逻辑的编写，以下是业务层的详细代码。

```
package com.hx.service;
@Service
public class MessageService {
    @Autowired
```

```
    MessageDAO messageDAO;
    @Autowired
    SensitiveService sensitiveService;
    public int addMessage(Message message) {
        message.setContent(sensitiveService.filter(message.getContent()));
        return messageDAO.addMessage(message) > 0 ? message.getId() : 0;
    }
    public List<Message> getConversationDetail(String conversationId, int offset, int limit) {
        return    messageDAO.getConversationDetail(conversationId, offset, limit);
    }
    public List<Message> getConversationList(int userId, int offset, int limit) {
        return    messageDAO.getConversationList(userId, offset, limit);
    }
    public int getConversationUnreadCount(int userId, String conversationId) {
        return messageDAO.getConversationUnreadCount(userId, conversationId);
    }
}
```

8.3 数据库操作层

数据库层也叫 DAO 层，负责从接收 service 层传递过来的数据，进行数据库的一些列操作，包括增删查改等操作，在这个层，数据库操作完成之后会将操作的结果返回过来，不管是增删改查的操作最后都会有一个操作结果返回回来，DAO 才拿到这个操作结果之后会进一步的返回给 service 层，service 拿到数据之后一般会做进一步的处理，比如封装数据，封装成前端界面能够解析的数据，然后发送到控制层，控制层负责将其发送给前端界面，前端界面利用标签来解析这些数据并显示。下面是 DAO 层的代码。

```
package com.hx.dao;
@Mapper
public interface MessageDAO {
    String TABLE_NAME = " message ";
    String INSERT_FIELDS = " from_id, to_id, content, has_read, conversation_id, created_date
";
    String SELECT_FIELDS = " id, " + INSERT_FIELDS;

    @Insert({"insert into ", TABLE_NAME, "(", INSERT_FIELDS,
```

```
            ")                                                                    values
(#{fromId},#{toId},#{content},#{hasRead},#{conversationId},#{createdDate})"})
    int addMessage(Message message);

    @Select({"select ", SELECT_FIELDS, " from ", TABLE_NAME,
            " where conversation_id=#{conversationId} order by created_date desc limit
#{offset}, #{limit}"})
    List<Message> getConversationDetail(@Param("conversationId") String conversationId,
                                        @Param("offset") int offset,
                                        @Param("limit") int limit);

    @Select({"select ", INSERT_FIELDS, " , count(id) as id from ( select * from ",
TABLE_NAME,
            " where from_id=#{userId} or to_id=#{userId} order by created_date desc) tt group
by conversation_id order by created_date desc limit #{offset}, #{limit}"})
    List<Message> getConversationList(@Param("userId") int userId,
                                      @Param("offset") int offset,
                                      @Param("limit") int limit);

    @Select({"select count(id) from ", TABLE_NAME, " where has_read=0 and to_id=#{userId}
and conversation_id=#{conversationId}"})
    int getConversationUnreadCount(@Param("userId") int userId, @Param("conversationId")
String conversationId);
}
```

参考文献

[1] Martin Fowler. Patterns of Enterprise Application Architecture. 机械工业出版社 2010

[2] Grenford J. Myers. The Art of Software Testing . Wiley 2004

致　　谢

这次毕业论文能够得以顺利完成，并非我一人之功劳，是所有指导过我的老师，帮助过我的同学和一直关心支持着我的家人对我的教诲、帮助和鼓励的结果。我要在这里对他们表示深深的谢意!

感谢教过我的每一位老师，四年的生活相处不久，却从你们身上学到了太多，必将终身受益，是你们诲人不倦才有了现在的我。

感谢我的父母，没有你们，就没有我的今天，你们的支持与鼓励，永远是支撑我前进的最大动力。

感谢兰心序，安农礼堂里挥汗如雨，日月湖畔闲庭信步，绿荫场上把酒言欢……最难忘的记忆里都有你身影。感谢一起欢笑一起惆怅的日子，不论何时，请不要忘记最初的梦想。