

# 计算机科学与工程学院课程设计报告

题目全称: 实现简单的微博平台

题目难度等级: 3

指导老师: 韩宏 职称: 副教授

序号	学生姓名	学号	班号	成绩
1	徐贤达	2016060601018	2016061101	
2	吴俊良	2016060203018	2016061101	
3				
4				
5				

(注: 学生姓名填写按学生对该课程设计的贡献及工作量由高到底排列, 分数按排名依次递减。序号排位为“1”的学生成绩最高, 排位为“5”的学生成绩最低。

主要任务:

用 Java, Js, C# 或其它语言, 实现简单地网页微博平台

详细功能描述:

微博平台的功能包括: (1) 用户注册; (2) 用户发表不超过 139 字的文本, 即微博; (3) 用户关注和取消关注其它用户; (4) 用户对微博加入评论 (不超过 100 字); (5) 用户主页可查看关注用户和自己发表的微博。

预期成果或目标:

微博平台演示程序

指导老师评语:

指导教师签字: \_\_\_\_\_

# 基于 SSM 框架的微博系统

## 摘要

在此课程设计中，本小组设计了一款功能齐全而又简便使用的微博系统。它是一套集浏览微博、发表微博、评论微博、转发微博、收藏微博、关注用户、添加好友和后台管理等为一体的 Web 应用系统，面向注册用户、一般用户和管理员这三类用户人群。在实现过程中，本小组后端采用的是 Spring + SpringMVC + Mybatis 的 SSM 框架，前端采用的是 Bootstrap 框架，数据库采用的是 MySQL，部署的服务器是阿里云服务器，网址是 <http://47.103.63.146:8080/weibo>

## 关键词

微博系统、Web 应用系统、SSM 框架、Bootstrap 框架，MySQL

# 目录

第一章 任务完成简述 .....	3
第二章 背景介绍 .....	3
2.1 微博的研究背景 .....	3
2.2 相关技术的介绍 .....	4
2.2.1 Spring 技术简介 .....	4
2.2.2 Spring MVC 技术简介 .....	5
2.2.3 MyBatis 技术简介 .....	6
2.2.4 Bootstrap 技术简介 .....	7
2.2.5 MySQL 技术简介 .....	7
第三章 需求分析 .....	7
3.1 产品介绍 .....	7
3.2 产品遵循的标准 .....	7
3.3 产品范围 .....	7
3.4 产品的功能性需求 .....	8
第四章 详细设计 .....	9
4.1 系统分析 .....	9
4.2 运行环境 .....	9
4.2.1 硬件环境 .....	9
4.2.2 软件环境 .....	9
4.3 功能设计 .....	10
4.3.1 管理员控制类设计 .....	10
4.3.2 用户控制类设计 .....	11
4.3.3 微博控制类设计 .....	13
4.3.4 评论控制类设计 .....	15
4.3.5 收藏控制类设计 .....	15
4.3.6 回复控制类设计 .....	16
4.3.7 关系控制类设计 .....	17
4.3.8 点赞控制类设计 .....	18
4.3.9 通知控制类设计 .....	18
4.4 数据库设计 .....	21
4.4.1 实体类设计 .....	21
4.4.2 ER 图 .....	24
第五章 系统测试 .....	25
5.1 测试目标 .....	25
5.2 测试环境 .....	25
5.3 测试情况 .....	25
5.3.1 游客功能测试执行情况 .....	25
5.3.2 微博用户功能测试执行情况 .....	27
5.3.3 管理员功能测试执行情况 .....	34

第六章 线上测试.....	38
6.1 测试目标.....	38
6.2 测试环境.....	38
6.3 测试 .....	38
第七章 总结与展望 .....	40
7.1 总结 .....	40
7.2 展望 .....	40
参考文献.....	41

# 第一章 任务完成简述

本小组成员共两人，分别是徐贤达和吴俊良。

此课程设计经历了需求分析，前期学习，环境搭建，软件编写，系统测试，软件优化大致这么几个环节，成功地实现课程设计的预期目标，搭建了微博系统，方便客户进行注册登录并进行浏览微博、发表微博、评论微博、转发微博、收藏微博、关注用户等操作。在此基础之上，本小组设计了后台的管理员系统，管理员能够很好地对微博系统内的用户和微博信息进行管理。最后，本小组将微博系统部署到了阿里云服务器上，成功运行。

在软件编写的任务分配上，徐贤达主要进行后端的开发，吴俊良主要进行前端的开发。

## 第二章 背景介绍

### 2.1 微博的研究背景

微博，就是微博客（MicroBlog）的简称，是一个基于用户关系的信息分享、传播以及获取的平台，用户可以通过 WEB、WAP 以及各种客户端组建个人社区，以 140 字左右的文字更新信息，并实现即时分享。陈永东，国内知名新媒体领域研究学者，他在国内率先给出了微博的定义。其中有五方面的理解：1、关注机制：分为单向和双向两种；2、内容简短：通常为 140 字；3、实时信息：最新的实时信息；4、广播式：公开的信息，谁都可以浏览以及转播；5、社交网络平台：把微博归为社交网络。

微博为广大用户提供了这样一个平台，你既可以以观众的身份在微博上浏览你感兴趣的信息，也可以以发布者的身份在微博上发布内容供别人浏览。发布的内容一般较短，通常为 140 字的限制，由此命名为微博。当然，也可以发布图片，分享歌曲、视频等。

微博有两方面的含义：首先，对于一些强调版面布置的博客来说，微博主要是由简单的只言片语组成，从这个角度来说，微博对用户的技术要求门槛很低，而且与博客相比，对语言的编排组织的要求没那么高。第二，微博浏览非常方便，由于微博开通的多种 API，所以用户可以通过手机、网络等方式来即时更新自己的个人信息，深受广大用户的喜爱。

微博的特点：信息获取具有很强的自主性、选择性；微博宣传的影响力具有很大弹性，与内容质量高度相关；内容短小精悍。微博的内容限定为 140 字左右，内容简短，不需长篇大论，门槛较低；信息共享便捷迅速，这也是微博最大的特点。

新浪微博是中国用户人数最多的微博平台之一，本小组在课程设计中也参考了新浪微博的设计。

## 2.2 相关技术的介绍

### 2.2.1 Spring 技术简介

Spring 是一个开源框架，它由 Rod Johnson 创建。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式，如下图所示。

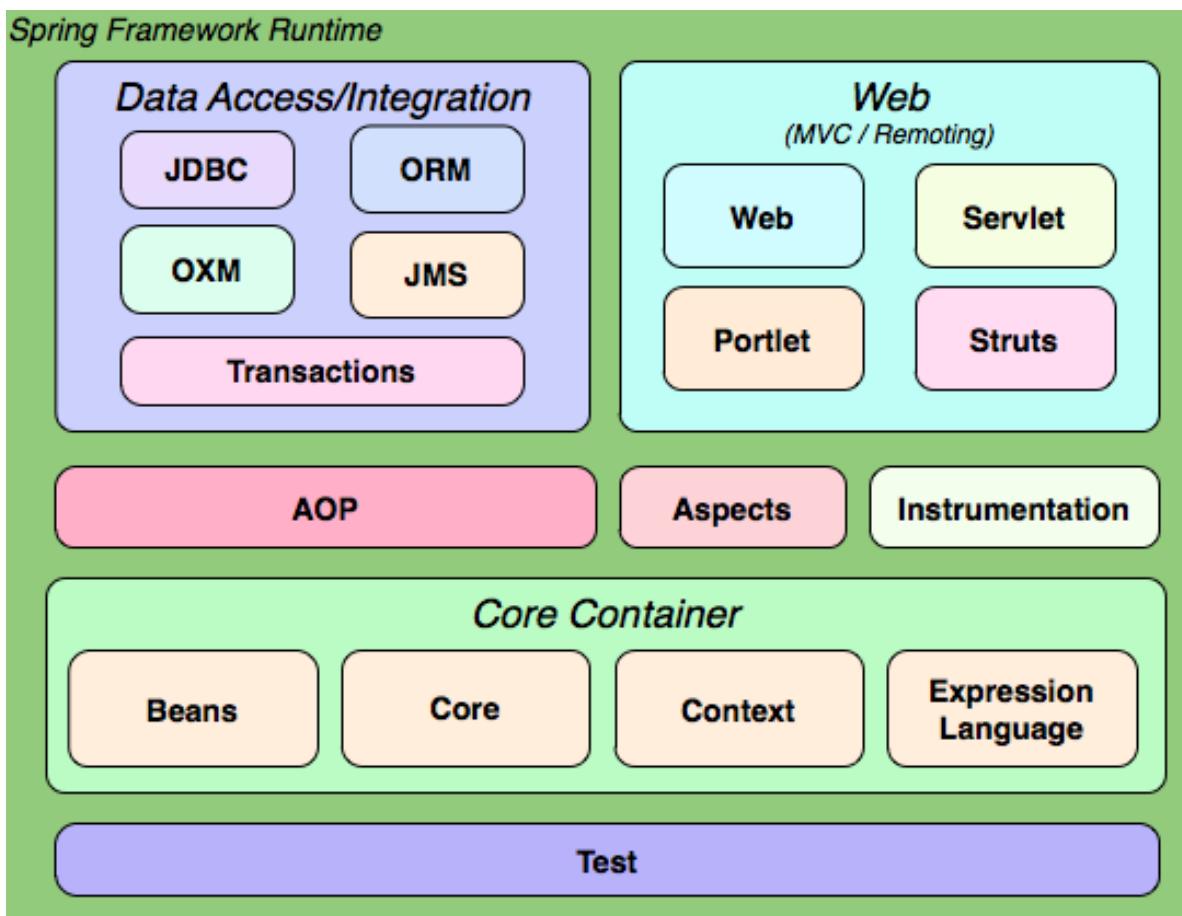


图 2.1 Spring 模块

Spring 框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。每个模块的功能如下：

**核心容器：**核心容器提供 Spring 框架的基本功能。核心容器的主要组件是 BeanFactory，它是工厂模式的实现。BeanFactory 使用控制反转（IOC）模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。

**Spring 上下文**: Spring 上下文是一个配置文件, 向 Spring 框架提供上下文信息。Spring 上下文包括企业服务, 例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。

**Spring AOP**: 通过配置管理特性, Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。所以, 可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP, 不用依赖 EJB 组件, 就可以将声明性事务管理集成到应用程序中。

**Spring DAO**: JDBC DAO 抽象层提供了有意义的异常层次结构, 可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理, 并且极大地降低了需要编写的异常代码数量 (例如打开和关闭连接)。Spring DAO 的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。

**Spring ORM**: Spring 框架插入了若干个 ORM 框架, 从而提供了 ORM 的对象关系工具, 其中包括 JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

**Spring Web 模块**: Web 上下文模块建立在应用程序上下文模块之上, 为基于 Web 的应用程序提供了上下文。所以, Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

**Spring MVC 框架**: MVC 框架是一个全功能的构建 Web 应用程序的 MVC 实现。通过策略接口, MVC 框架变成为高度可配置的, MVC 容纳了大量视图技术, 其中包括 JSP、Velocity、Tiles、iText 和 POI。

Spring 框架的功能可以用在任何 J2EE 服务器中, 大多数功能也适用于不受管理的环境。Spring 的核心要点是: 支持不绑定到特定 J2EE 服务的可重用业务和数据访问对象。毫无疑问, 这样的对象可以在不同 J2EE 环境 (Web 或 EJB)、独立应用程序、测试环境之间重用。

简单来说, Spring 是一个轻量级的控制反转 (IOC) 和面向切面 (AOP) 的容器框架。

## 2.2.2 Spring MVC 技术简介

模型(Model), 视图(View)和控制(Controller) 三个单词组成了 Model- View- Controller, 缩写即 MVC。MVC 模式的目的就是实现 Web 系统的职能分工。Model 层通常可以用 JavaBean 或 EJB 来实现系统中的业务逻辑。View 层通常用 JSP 来实现与用户的交互。Model 与 View 之间沟通的桥梁是 Controller 层, 它可以分派用户的请求并且选择恰当的视图以用于显示, 同时它也可以解释用户

的输入并将它们映射为模型层可执行的操作。

MVC 强制性的使应用程序的输入、处理和输出分开，它是一个设计模式。使用 MVC 应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。下图介绍了这几个模块的功能以及之间的关系。

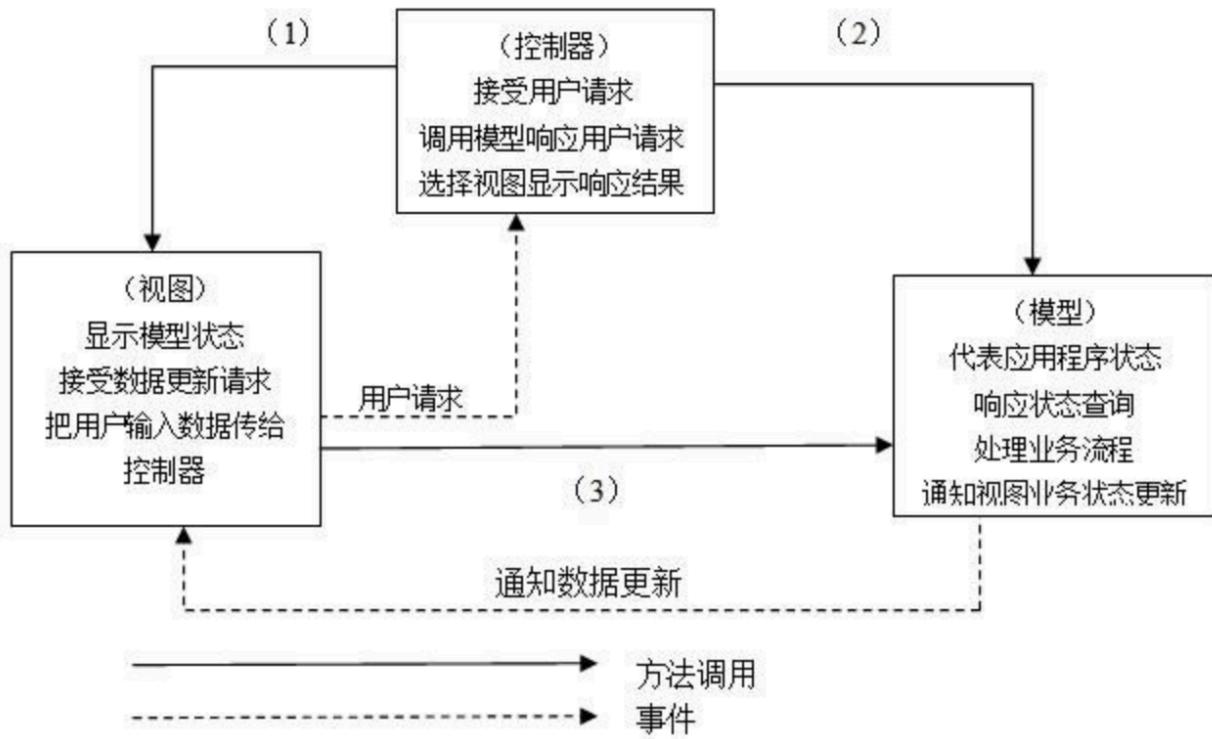


图 2.2 MVC 设计模式图

MVC 设计模式是一个很好创建软件的途径，它所提倡的一些原则，像内容和显示互相分离可能比较好理解。本小组在课程设计中采用的是 Spring MVC 框架，它是 Spring 的一个模块，提供了 MVC 架构和用于开发灵活和松散耦合的 Web 应用程序的组间。

### 2.2.3 MyBatis 技术简介

MyBatis 是一个优秀的持久层框架，它对 jdbc 的操作数据库的过程进行封装，使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、创建 connection、创建 statement、手动设置参数、结果集检索等 jdbc 繁杂的过程代码。

MyBatis 通过 xml 或注解的方式将要执行的各种 statement 配置起来，并通过 java 对象和 statement 中的 SQL 进行映射生成最终执行的 SQL 语句，最后由 MyBatis 框架执行 SQL 并将结果映射成 java 对象并返回。本小组在课程设计中采用 MyBatis 的一大原因是它可以大大简化 jdbc 的操作代码。

## 2.2.4 Bootstrap 技术简介

Bootstrap 是我们使用的前端框架。它是一款基于 HTML、CSS、JavaScript 开发的简洁、直观、强悍的前端开发框架，使得 Web 开发更加快捷。

## 2.2.5 MySQL 技术简介

MySQL 是一个小型关系型数据库管理系统，开发者为瑞典 MySQL AB 公司。由于其速度快、体积小、总体拥有成本低，尤其 MySQL 是开源的这一特点，所以许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。在此系统设计中，本小组应用 MySQL 的主要原因是：MySQL 数据库的体积小，而且运行速度快，总体拥有成本低。

# 第三章 需求分析

## 3.1 产品介绍

本小组设计的微博系统面向注册用户、一般用户和管理员这三类用户人群，是一套集浏览微博、发表微博、评论微博、转发微博、收藏微博、关注用户、添加好友和后台管理等为一体的 Web 应用系统。

## 3.2 产品遵循的标准

本小组设计的微博系统参考了新浪微博，并加以改进，具有以下特点：

1. 操作界面美观大方
2. 功能齐全，能够实现微博的正常功能
3. 开放性好，使用 SSM 框架，MVC 架构，数据库采用 MySQL

## 3.3 产品范围

本小组设计的微博系统的应用领域包括：

1. 方便用户和游客浏览微博
2. 方便游客注册为微博用户
3. 方便微博用户登录并修改个人资料
4. 方便微博用户发表微博、评论微博、转发微博、收藏微博和添加好友

## 5. 方便管理员对系统内用户和微博信息进行管理

### 3.4 产品的功能性需求

功能类别	功能名称、标识符	描述
管理员	登陆后台	凭登录名和密码进行登陆
	退出后台	操作结束后可以退出后台
	用户管理	查看用户信息并删除用户
	微博管理	删除部分微博
	修改密码	修改管理员密码
微博用户	登陆系统	凭登录名和密码进行登陆
	退出系统	操作结束后可以退出系统
	修改个人资料	修改个人资料
	关注用户与添加好友	关注某些用户，如果双方都关注对方，则变成好友
	发表微博	发表自己的微博
	评论微博	对注册用户所关注的用户进行评论以及回复
	转发微博	对其他用户所发的微博进行转发，从而变为自己的微博。
	点赞微博	对其他用户所发的微博进行点赞
	收藏微博	在个人主页或者在广播大厅中收藏别人的微博
游客	搜索微博和用户	在个人主页或者在广播大厅中搜索微博和用户
	浏览微博	浏览网站内的所有用户发表的消息
	注册系统	可以注册成为微博用户

## 第四章 详细设计

### 4.1 系统分析

本系统可以实现一个简易的类微博平台，用户通过该平台可发表、转发、评论、点赞微博，搜索微博和用户。用户还可以关注他人，接收到被关注者的最新动态。

### 4.2 运行环境

#### 4.2.1 硬件环境

本系统的运行硬件环境如下：

客户机：普通 PC

CPU：P4 1.8GHz 以上

内存：256MB 以上

能够运行 IE5.0 以上或者 Netscape4.0 以上版本的机器

分辨率：推荐使用 1024\*768 像素

WEB 服务器：

CPU：P4 2.0GHz

内存：1G 以上

硬盘：80G 以上

网卡：千兆

数据库服务器：

CPU：P4 2.0GHz

内存：1G 以上

硬盘：80G 以上

#### 4.2.2 软件环境

本系统的的软件运行环境如下：

操作系统：Unix/Linux/windows2000 或以上版本

数据库：MySQL Server 8.0

开发工具包：JDK Version 1.8

开发环境：Jetbrain IntelliJ Idea 2019.1.2

Web 服务器：Tomcat

浏览器：IE6.0 以上

### 4.3 功能设计

本微博平台的功能设计如下：

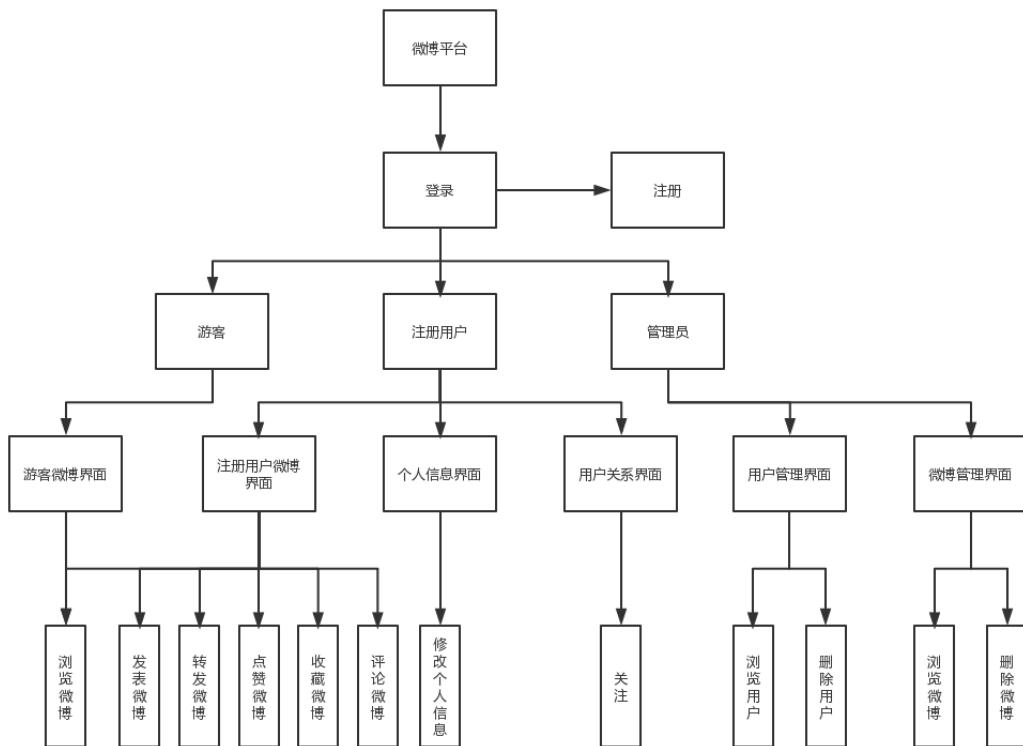


图 4.1 微博系统功能设计图

下面将分别介绍各功能控制类的接口：

#### 4.3.1 管理员控制类设计

该控制类具有如下功能：管理员登录与注册

类函数接口：

@Controller

```
public class AdminController {
```

```
// 管理员 Service
```

```
@Autowired
```

```
private AdminService adminService;
```

```

// 管理员注册
@RequestMapping(value="signin")
public String signin();

//管理员登录
@RequestMapping(value="loginAdmin", method=RequestMethod.POST)
public String loginAdmin(
    @RequestParam("username") String username,
    @RequestParam("password") String password
);
}

}

```

#### 4.3.2 用户控制类设计

该控制类具有如下功能：用户注册登录及其基本功能

类函数接口：

```

@Controller
public class UserController {

    // 用户 Service
    @Autowired
    private UserService userService;

    // 微博 Service
    @Autowired
    private WeiboService weiboService;

    // 关系 Service
    @Autowired
    private RelationService relationService;

    // 通知 Service
    @Autowired
    private MentionService mentionService;

    // date 格式化工具类
    DateConvert dateConvert;

    // 登录页面
    @RequestMapping(value = "login")
    public String login() throws Exception {
        return "login";
    }

    // 注册页面
    @RequestMapping(value = "goregister")

```

```
public String goregister() throws Exception {
    return "register";
}

// 用户退出
@RequestMapping(value = "exit")
public String exit(HttpServletRequest request) throws Exception {
    request.getSession().invalidate();
    return "login";
}

// 用户注册
@RequestMapping(value = "register", method = RequestMethod.POST)
public String register(@RequestParam("vCode") String vCode, @RequestParam("username") String username,
    @RequestParam("password1") String password1, @RequestParam("password2") String password2,
    HttpServletRequest request, HttpServletResponse response);

// 注册页面（生成验证码）
@RequestMapping(value = "toRegister")
public void toRegister(HttpServletRequest request, HttpServletResponse response);

// 用户登录
@RequestMapping(value = "loginUser", method = RequestMethod.POST)
public String loginUser(HttpServletRequest request, HttpSession session, UserVo userVo);

// 跳至修改用户信息视图
@SuppressWarnings("static-access")
@RequestMapping(value = "updateInfo")
public String updateInfo(HttpSession session, Model model);

// 提交修改用户信息
@RequestMapping(value = "submitInfo", method = RequestMethod.POST)
public String submitInfo(HttpSession session, Model model, MultipartFile user_face, UserVo userVo);

// 跳至修改密码页面
@RequestMapping(value = "toUpdatePassword")
public String toUpdatePassword(HttpSession session);

// 修改密码
@RequestMapping(value = "undatePassword")
public String undatePassword(HttpSession session, HttpServletRequest request, @RequestParam("oldpw") String oldpw,
    @RequestParam("newpw1") String newpw1, @RequestParam("newpw2") String newpw2);

// 去往我的主页
@SuppressWarnings("static-access")
```

```

@RequestMapping(value = "queryMinePage")
public String queryMinePage(HttpSession session, Model model, @RequestParam("pageNo") int pageNo);

// 访问用户主页
@RequestMapping(value = "queryUserPage")
public String queryUserPage(HttpServletRequest request, HttpSession session,
    @RequestParam("userId") int userId,
    @RequestParam("pageNo") int pageNo,
    Model model);

// 模糊查询用户列表
@RequestMapping("queryUserByWord")
public String queryUserByWord(
    Model model,
    HttpSession session,
    @RequestParam("keyWord") String keyWord);

}

}

```

### 4.3.3 微博控制类设计

该控制类具有如下功能：微博的所有相关操作

类函数接口：

@Controller

```
public class WeiboController {
```

// 微博 Service

@Autowired

```
private WeiboService weiboService;
```

// 用户 Service

@Autowired

```
private UserService userService;
```

// 评论 Service

@Autowired

```
private CommentService commentService;
```

// 回复 Service

@Autowired

```
private ReplyService replyService;
```

// 通知 Service

@Autowired

```
private MentionService mentionService;
```

// 点赞 Service

```
@Autowired
private LikesService likesService;

// 收藏 Service
@Autowired
private CollectService collectService;

// date 格式化工具类
private DateConvert dateConvert;

// 独立微博页面 详细评论页
@SuppressWarnings("all")
@RequestMapping(value = "/singleWeibo")
public void singleWeibo(HttpServletRequest session, HttpServletRequest request, HttpServletResponse response,
    @RequestParam("weiboid") Integer weiboid);

// 删除微博
@RequestMapping(value = "deleteWeibo", method = RequestMethod.GET)
public void deleteWeibo(@RequestParam("weiboid") Integer weiboid, HttpServletResponse response,
    HttpServletRequest request);

// 发送微博
@RequestMapping(value = "post")
public String post(HttpServletRequest request, HttpSession session, Model model, WeiboVo weiboVo);

// 图片上传
@RequestMapping(value = "upload", method = RequestMethod.POST)
public @ResponseBody String upload(MultipartFile file, Model model, HttpSession session);

// 遍历所有微博 实时
@SuppressWarnings("static-access")
@RequestMapping(value = "queryAllWeiboNow")
public String queryAllWeiboNow(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

// 遍历所有微博 好友圈
@SuppressWarnings("static-access")
@RequestMapping(value = "queryAllWeiboFriends")
public String queryAllWeiboFriends(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

// 遍历所有微博 首页
@SuppressWarnings("static-access")
@RequestMapping(value = "queryAllWeiboFollow")
public String queryAllWeiboFollow(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

// 根据关键字搜索相关微博
@RequestMapping(value = "queryWeiboByWord")
```

```

public String queryWeiboByWord(
    Model model,
    HttpSession session,
    @RequestParam("keyWord") String keyWord,
    @RequestParam("pageNo") int pageNo);
}

```

#### 4.3.4 评论控制类设计

该控制类具有如下功能：评论的添加和删除

类函数接口：

```

@Controller
public class CommentController {

```

```

    // 评论 Service
    @Autowired
    private CommentService commentService;

    @Autowired
    private static ObjectMapper MAPPER = new ObjectMapper();

    //添加评论
    @RequestMapping(value="comment",method = RequestMethod.POST,consumes = "application/json")
    @ResponseBody
    public CommentCustom comment(
        @RequestBody CommentCustom commentCustom);

    //遍历评论
    @RequestMapping(value="queryComment", method = RequestMethod.GET)
    @ResponseBody
    public String queryComment(
        HttpServletRequest request,
        @RequestParam("weiboid") int weiboid);

    //删除评论
    @RequestMapping(value="deleteComment")
    public void deleteComment(@RequestParam("commentId") int commentId);
}

```

#### 4.3.5 收藏控制类设计

该控制类具有如下功能：收藏添加和删除

类函数接口：

```

@Controller
public class CollectController {

```

```
// 收藏 Service
```

```

@.Autowired
private CollectService collectService;

// 收藏
@RequestMapping(value = "collect")
public void collect(
    @RequestParam("weiboid") int weiboid,
    HttpServletResponse response,
    HttpSession session);

// 取消收藏
@RequestMapping(value = "uncollect")
public void uncollect(
    @RequestParam("weiboid") int weiboid,
    HttpServletResponse response,
    HttpSession session);
}

```

#### 4.3.6 回复控制类设计

该控制类具有如下功能：回复的添加、删除和查询

类函数接口：

```

@Controller
public class ReplyController {

```

```

// 回复 Service
@Autowired
private ReplyService replyService;

//删除回复
@RequestMapping(value="deleteReply")
public void deleteReply(@RequestParam("replyId") int replyId);

//添加回复
@RequestMapping(value="reply",method = RequestMethod.POST,consumes = "application/json")
@ResponseBody
public ReplyCustom reply(
    @RequestBody ReplyCustom replyCustom );

//查询回复
@RequestMapping(value="queryReply", method = RequestMethod.GET)
@ResponseBody
public String queryReply(
    @RequestParam("commentId") int commentId );
}
```

#### 4.3.7 关系控制类设计

该控制类具有如下功能：关系的添加、删除和查询

类函数接口：

```
@Controller
public class RelationController {
    // 关系 Service
    @Autowired
    private RelationService relationService;

    // 通知 Service
    @Autowired
    private MentionService mentionService;

    // 用户 Service
    @Autowired
    private UserService userService;

    // 关注
    @RequestMapping(value = "follow")
    public void follow(HttpSession session,
                       HttpServletResponse response,
                       @RequestParam("flag") int flag, // 判断两人关系 1: 陌生 2: 有一人已关注
                       @RequestParam("userId") int userId);

    // 取关
    @RequestMapping(value = "unfollow")
    public void unfollow(HttpSession session,
                         HttpServletResponse response,
                         @RequestParam("flag") int flag, // 判断两人关系 1: 陌生 2: 有一人已关注
                         @RequestParam("userId") int userId);

    // 关注列表
    @RequestMapping("listFollow")
    public String listFollow(@RequestParam("userId") int userId,
                           HttpServletRequest request,
                           HttpSession session);

    // 粉丝列表
    @RequestMapping("listFans")
    public String listFans(@RequestParam("userId") int userId,
                          HttpServletRequest request,
                          HttpSession session);
}
```

#### 4.3.8 点赞控制类设计

该控制类具有如下功能：点赞的添加、删除和查询

类函数接口：

@Controller

```
public class LikesController {
```

@Autowired

```
    private LikesService likesService;
```

// 点赞

@RequestMapping(value="like" )

```
    public void like(
```

@RequestParam("weiboid") int weiboid,

HttpServletResponse response,

HttpSession session) ;

// 取消赞

@RequestMapping(value="unlike" )

```
    public void unlike(
```

@RequestParam("weiboid") int weiboid,

HttpServletResponse response,

HttpSession session);

```
}
```

#### 4.3.9 通知控制类设计

该控制类具有如下功能：通知相关的所有操作

类函数接口：

@Controller

```
public class MentionController {
```

// 通知 Service

@Autowired

```
    private MentionService mentionService;
```

// 微博 Service

@Autowired

```
    private WeiboService weiboService;
```

// 点赞 Service

@Autowired

```
    private LikesService likesService;
```

// 评论 Service

@Autowired

```
    private CommentService commentService;
```

```
// 回复 Service
@Autowired
private ReplyService replyService;

// 收藏 Service
@Autowired
private CollectService collectService;

@Autowired
private static ObjectMapper MAPPER = new ObjectMapper();

// date 格式化工具类
private DateConvert dateConvert;

/***
 * ajax 长轮询 获得新通知
 *
 * @param session
 * @return
 * @throws Exception
 */
@ResponseBody
@RequestMapping(value = "getNotice")
public String getNotice(HttpSession session);

/***
 * 跳转到<转发过我的>
 *
 * @param session
 * @param model
 * @param pageNo
 * @return
 * @throws Exception
 */
@SuppressWarnings("static-access")
@RequestMapping(value = "toRepostPage")
public String toRepostPage(HttpSession session, Model model, @RequestParam("pageNo") int pageNo);

/***
 * 跳转到<收到的评论>
 *
 * @param session
 * @param model
 * @param pageNo
 * @return
 */

```

```

* @throws Exception
*/
@RequestMapping(value = "toCommentPage")
public String toCommentPage(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

/**
 * 跳转到<回复我的>
 *
 * @param session
 * @param model
 * @param pageNo
 * @return
 * @throws Exception
 */
@RequestMapping(value = "toReplyPage")
public String toReplyPage(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

/**
 * 跳转到<赞过我的>
 *
 * @param session
 * @param model
 * @param pageNo
 * @return
 * @throws Exception
 */
@RequestMapping(value = "toLikePage")
public String toLikePage(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

/**
 * 跳转到<我的赞>
 *
 * @param session
 * @param model
 * @param pageNo
 * @return
 * @throws Exception
 */
@SuppressWarnings("static-access")
@RequestMapping(value = "toMyLikes")
public String toMyLikes(HttpServletRequest session, Model model, @RequestParam("pageNo") int pageNo);

/**
 * 跳转到<我的收藏>
 *
 * @param session

```

```

* @param model
* @param pageNo
* @return
* @throws Exception
*/
@SuppressWarnings("static-access")
@RequestMapping(value = "toMyCollection")
public String toMyCollection(HttpSession session, Model model, @RequestParam("pageNo") int pageNo);
}

```

## 4.4 数据库设计

### 4.4.1 实体类设计

管理员类：

名称	代码	数据类型	长度
管理员 id	admin_id	int	11
用户名	username	varchar	20
密码	password	varchar	20

收藏类：

名称	代码	数据类型	长度
收藏 id	collect_id	int	11
微博 id	weibo_id	int	11
用户 id	user_id	int	11
收藏时间	collect_time	datetime	

评论类：

名称	代码	数据类型	长度
评论 id	comment_id	int	11
用户 id	user_id	int	11
微博 id	weibo_id	int	11
评论时间	comment_time	timestamp	
评论内容	comment_content	varchar	100

点赞类：

名称	代码	数据类型	长度

点赞 id	likes_id	int	11
用户 id	user_id	int	11
微博 id	weibo_id	int	11
点赞时间	like_time	datetime	

地区类：

名称	代码	数据类型	长度
地区 id	id	varchar	50
地区名	name	varchar	50

通知类：

名称	代码	数据类型	长度
通知 id	mention_id	int	11
用户 id	user_id	int	11
转发数	repostCount	int	11
评论数	commentCount	int	11
回复数	replyCount	int	11
点赞数	likeCount	int	11
粉丝数	fansCount	int	11

关系类：

名称	代码	数据类型	长度
关系 id	relation_id	int	11
被关注者 id	user_id	int	11
粉丝 id	follow_id	int	11
状态	state	int	11

回复类：

名称	代码	数据类型	长度
回复 id	reply_id	int	11
评论 id	comment_id	int	11
回复者 id	from_id	int	11

被回复者 id	to_id	int	11
回复内容	reply_content	varchar	100
时间	time	datetime	

用户类：

名称	代码	数据类型	长度
用户 id	user_id	int	11
用户名	username	varchar	50
密码	password	varchar	50
昵称	nickname	varchar	50
头像	face	varchar	50
性别	sex	int	11
生日	bir	date	
省	province	varchar	10
市	city	varchar	10

微博类：

名称	代码	数据类型	长度
微博 id	weibo_id	int	11
用户 id	user_id	int	11
发送时间	post_time	datetime	
微博内容	content	varchar	100
图片	pic1	varchar	50
图片 2	pic2	varchar	50
图片 3	pic3	varchar	50
图片 4	pic4	varchar	50
图片 5	pic5	varchar	50
图片 6	pic6	varchar	50
图片 7	pic7	varchar	50

图片 8	pic8	varchar	50
图片 9	pic9	varchar	50
原微博	original	int	11
转发 id	repost_id	int	11

#### 4.4.2 ER 图

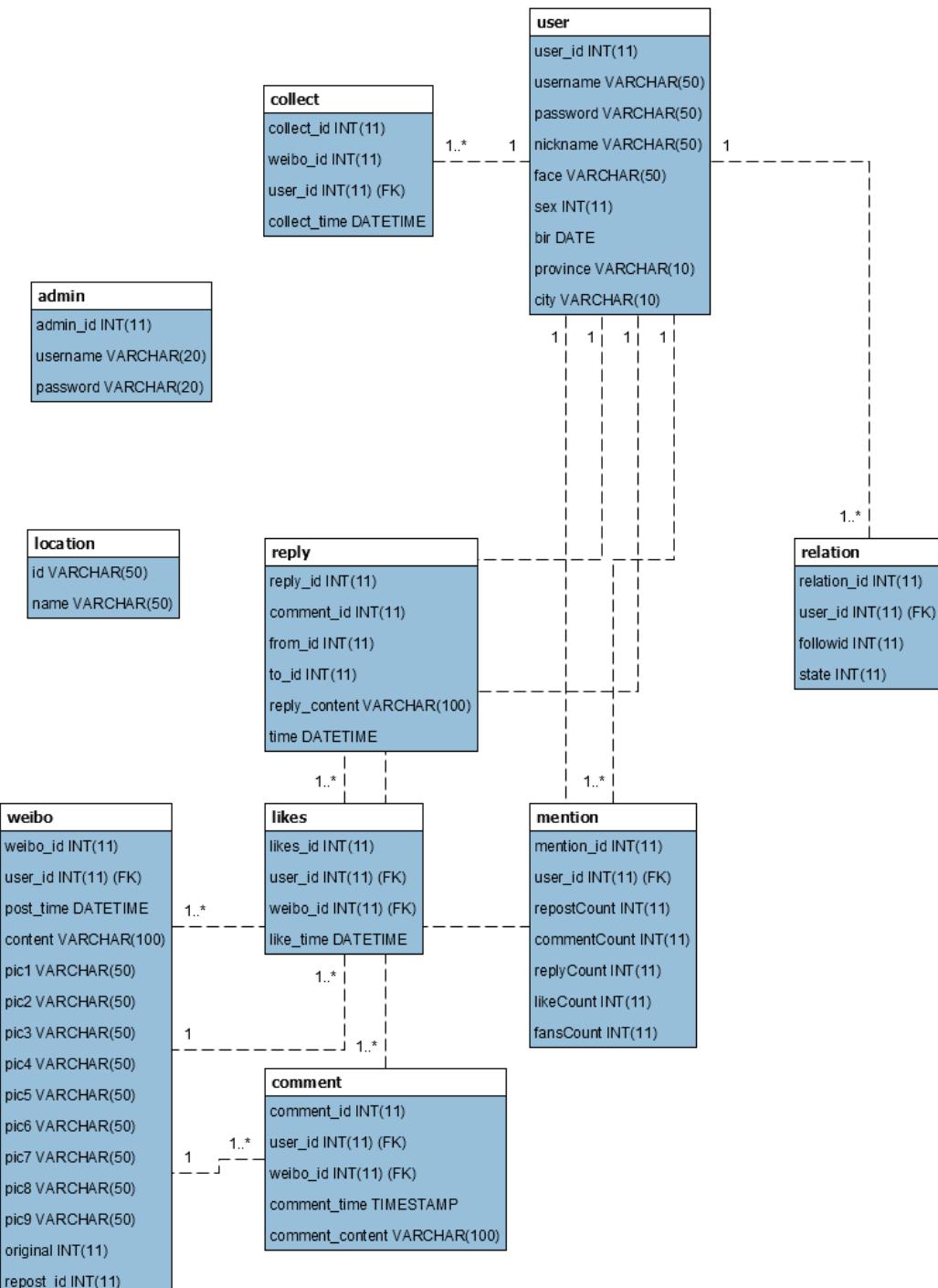


图 4.2 微博系统数据库 ER 图

# 第五章 系统测试

## 5.1 测试目标

本章中对于本小组设计的微博系统进行测试。

系统测试基于产品的功能性需求，功能类别为：游客、微博用户和管理员。

## 5.2 测试环境

操作系统：macOS Mojave 10.14.4

网页浏览器：Google Chrome 70.0.3729.169

虚拟服务器：Tomcat 8.5

处理器：Intel Core i7

内存：16 GB

## 5.3 测试情况

### 5.3.1 游客功能测试执行情况

微博系统的欢迎界面如下图所示：

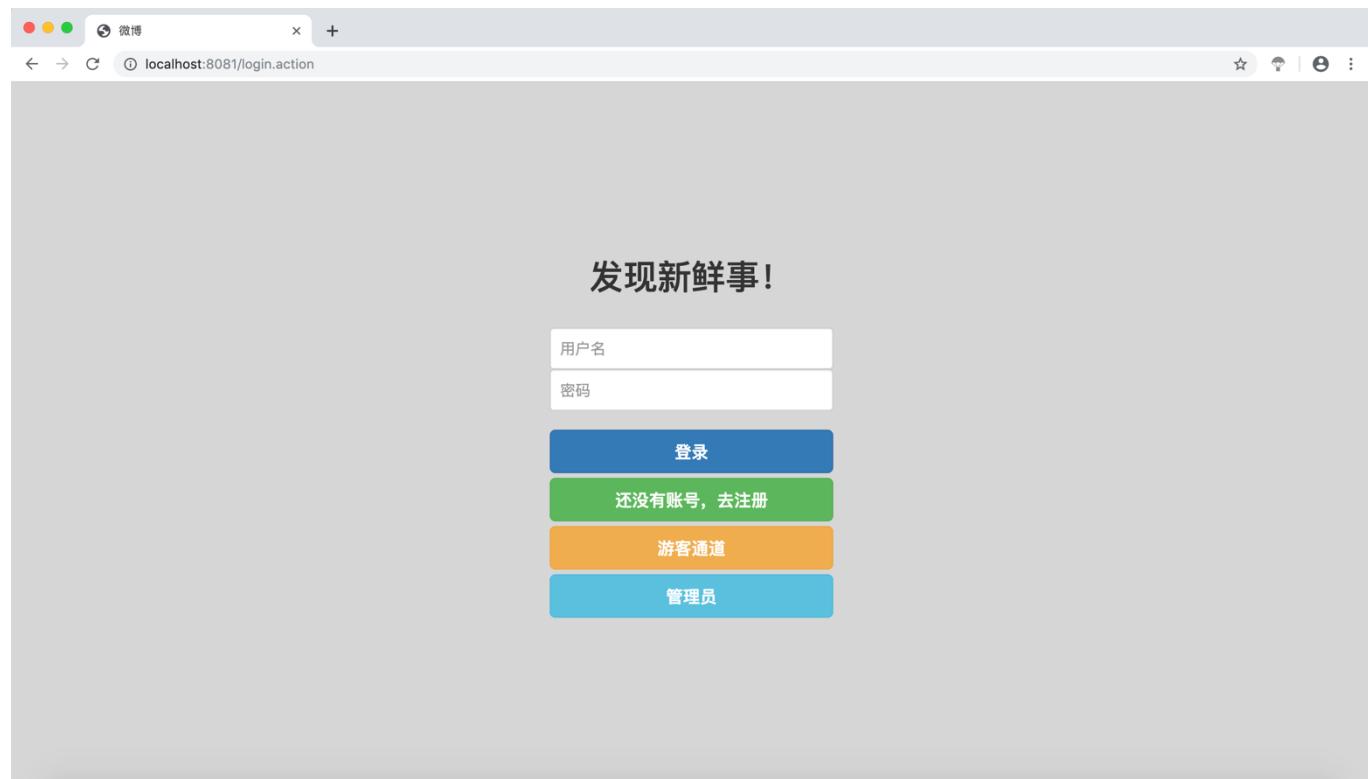


图 5.1 微博系统欢迎界面

我们首先测试游客功能，点击游客通道进入微博广播大厅，便可以浏览所有用户发表的微博。但是作为游客，功能仅限于浏览微博，不能对其他用户的微博进行诸如转发、点赞等操作，这里我们提示游客，如果需要拥有更多的功能，请先登录或者注册。执行效果如下图所示：

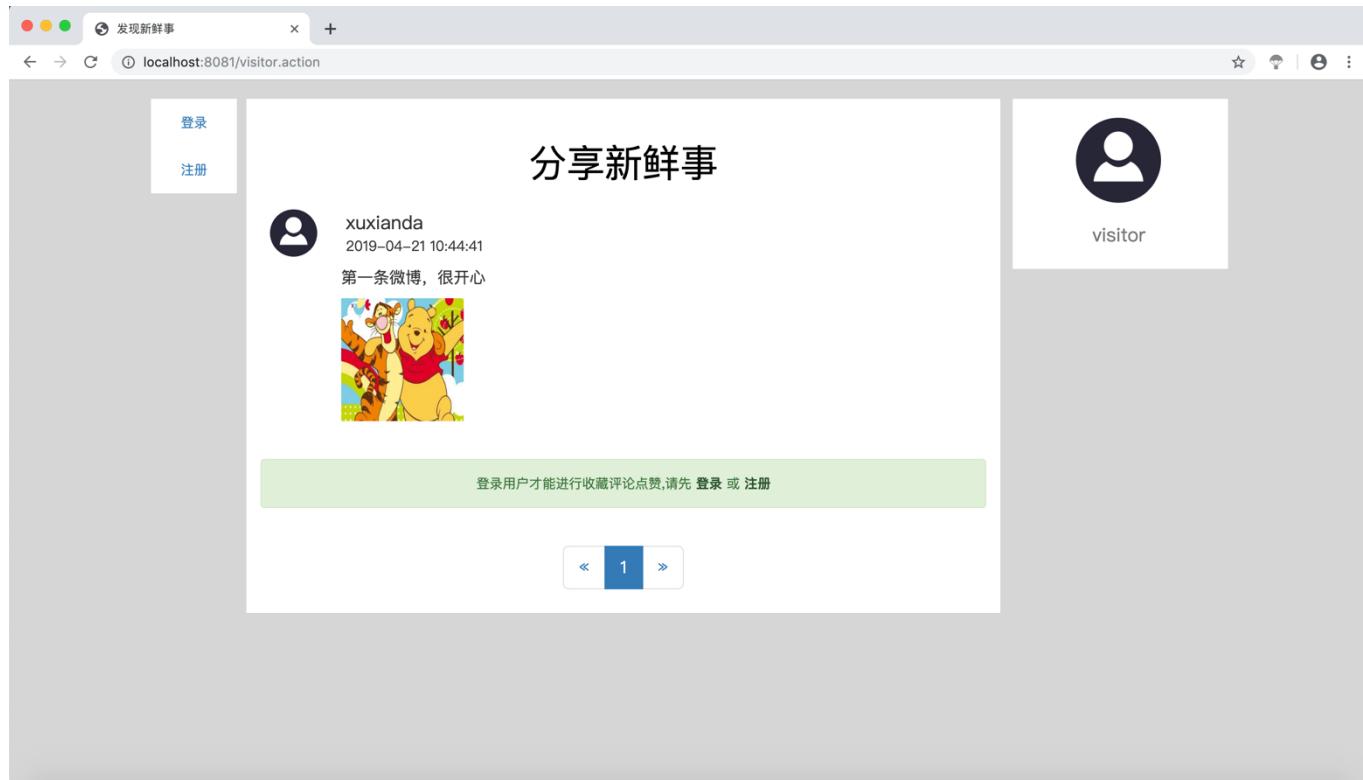


图 5.2 游客通道浏览界面

作为游客，可以注册为微博用户，拥有自己的账号。这里我们直接点击注册，进入注册页面，注册一个微博用户，如下图所示：

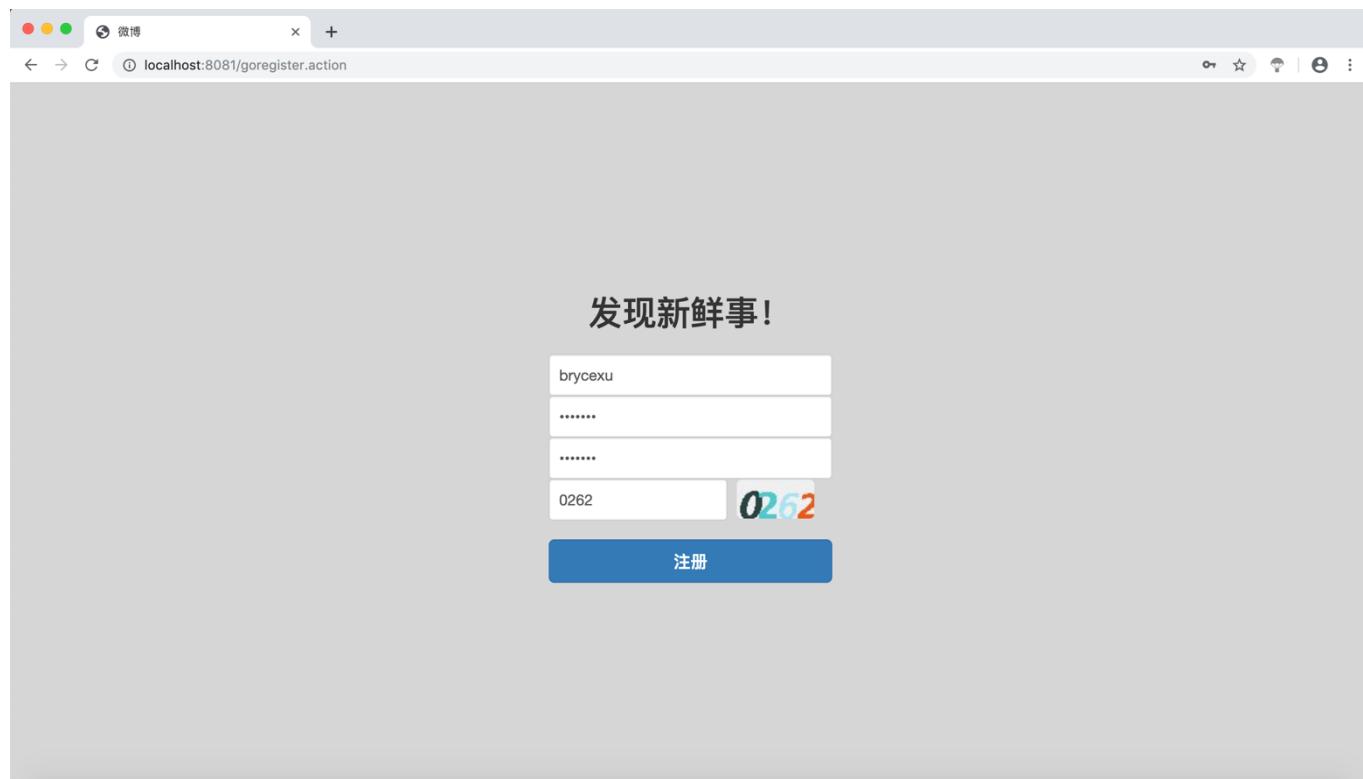


图 5.3 游客注册页面

在终端上，我们查看数据库里的变化，可以看出，我们成功地在存放用户的表中新建了用户，如下图所示：

```
XuXianda — mysql -u root -p — 133x30
| reply      |
| user       |
| weibo      |
+-----+
10 rows in set (0.00 sec)

mysql> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | username | password | nickname | face          | sex | bir   | province | city  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 23    | xuxianda | 6403838 | xuxianda | 639ff4a1-78ed-40b8-96a6-27c7c2e7081c.png | 1  | 1998-03-07 | 13    | 1303  |
| 33    | visitor   | 83df1f63-b1fe-48e0-b6ef-7d57c066b05c.png | 1  | 1920-06-05 | 24    | 2401  |
| 34    | wujunliang | 123456  | wujunliang | default.png | 1  | 2018-06-06 | 00    | 00    |
| 37    | duolali   | 6403838 | duolali   | default.png | 1  | 2019-01-01 | 00    | 00    |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | username | password | nickname | face          | sex | bir   | province | city  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 23    | xuxianda | 6403838 | xuxianda | 639ff4a1-78ed-40b8-96a6-27c7c2e7081c.png | 1  | 1998-03-07 | 13    | 1303  |
| 33    | visitor   | 83df1f63-b1fe-48e0-b6ef-7d57c066b05c.png | 1  | 1920-06-05 | 24    | 2401  |
| 34    | wujunliang | 123456  | wujunliang | default.png | 1  | 2018-06-06 | 00    | 00    |
| 37    | duolali   | 6403838 | duolali   | default.png | 1  | 2019-01-01 | 00    | 00    |
| 39    | brycexu   | 6403838 | brycexu   | default.png | 1  | 2019-01-01 | 00    | 00    |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

图 5.4 游客注册在数据库中的变化

### 5.3.2 微博用户功能测试执行情况

凭借用户名和密码，我们登录微博系统，如下图所示：

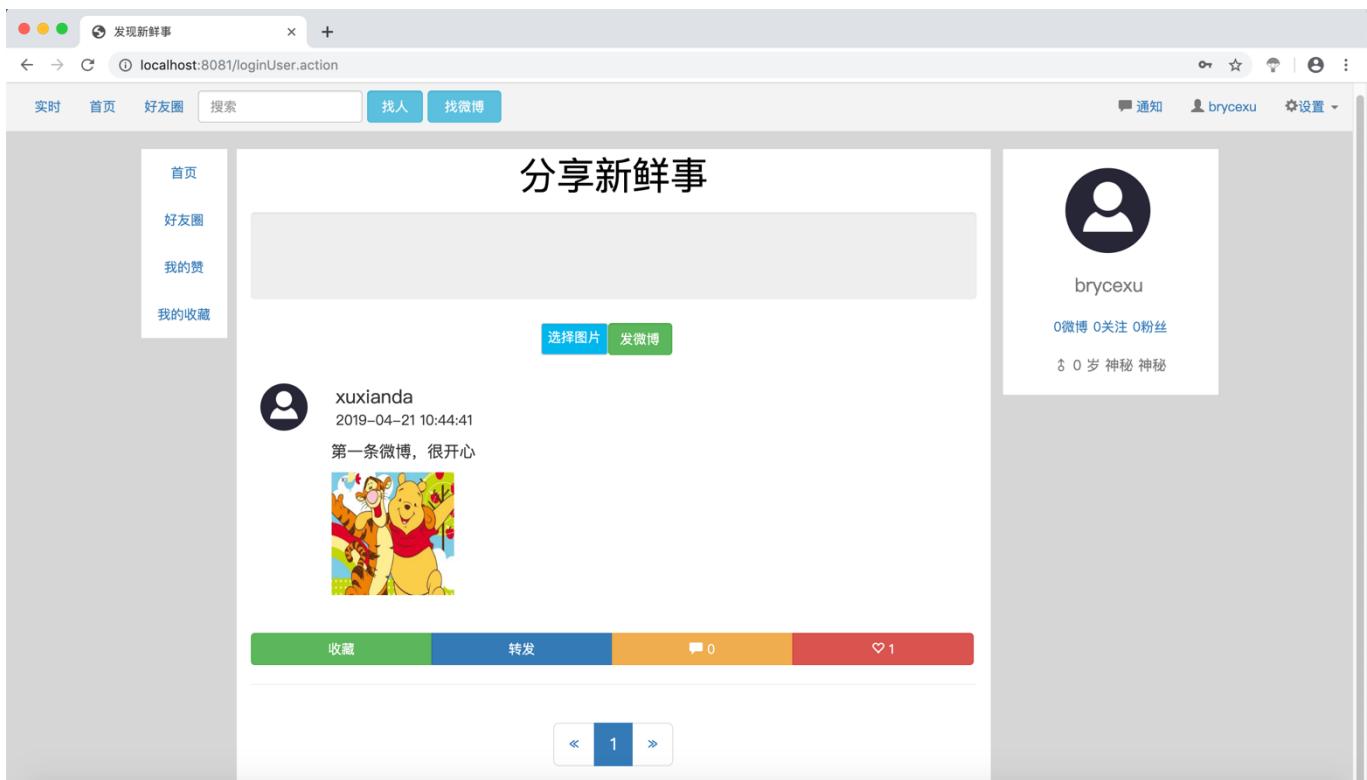


图 5.5 新注册用户登录界面

新注册用户个人信息基本为空，头像是默认的用户头像，我们可以进行修改，如下图所示：

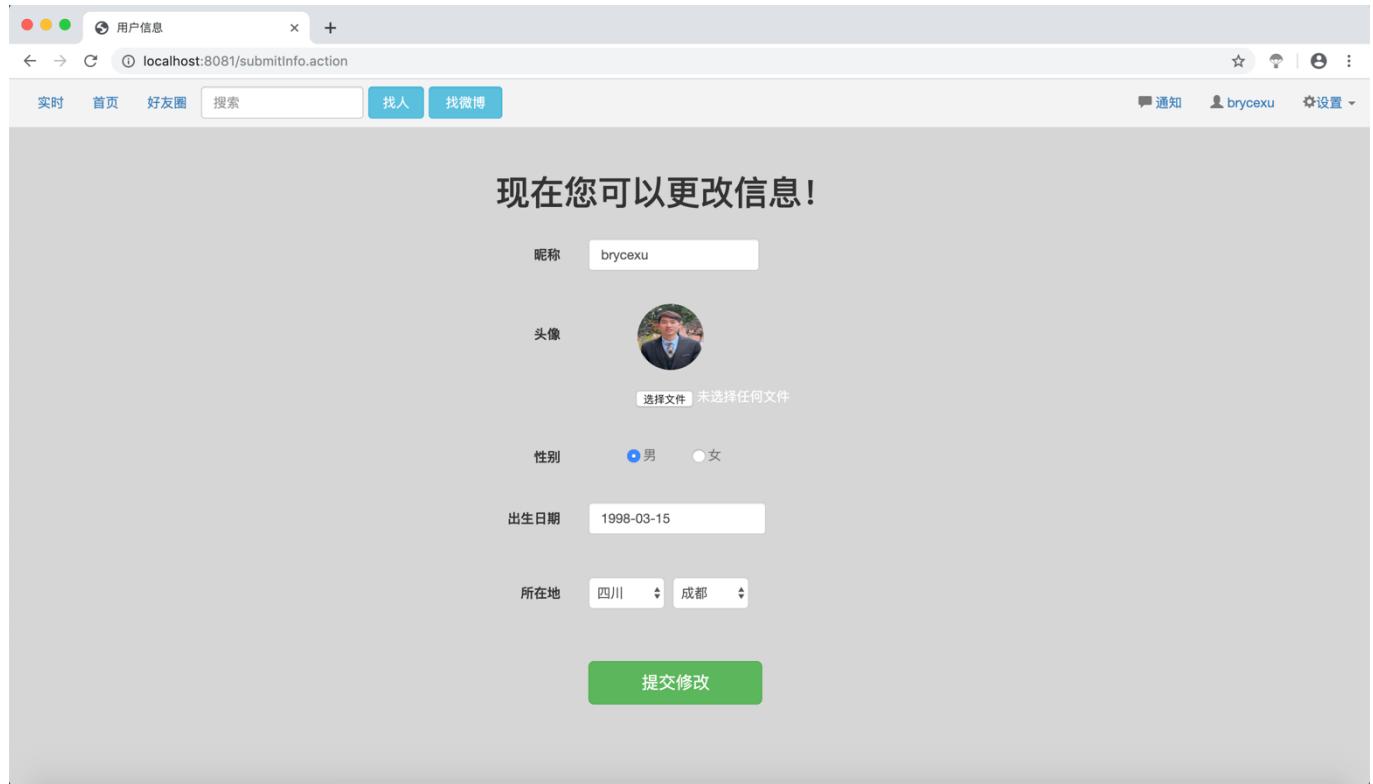


图 5.6 用户修改个人信息界面

修改完成后，我们打开终端，发现用户的信息成功地被修改了，如下图所示：

```
mysql> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | username | password | nickname | face          | sex | bir      | province | city |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 23     | xuxianda | 6403838 | xuxianda | 639ff4a1-78ed-40b8-96a6-27c7c2e7081c.png | 1   | 1998-03-07 | 13    | 1303  |
| 33     | visitor   | visitor   | visitor   | 83df1f63-b1fe-48e0-b6ef-7d57c066b05c.png | 1   | 1920-06-05 | 24    | 2401  |
| 34     | wujunliang | 123456   | wujunliang | default.png | 1   | 2018-06-06 | 00    | 00    |
| 37     | duolali   | 6403838 | duolali   | default.png | 1   | 2019-01-01 | 00    | 00    |
| 39     | brycexu  | 6403838 | brycexu  | 4fabde23-6090-48d1-b781-e4071e89915f.JPG | 1   | 1998-03-14 | 24    | 2401  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

图 5.7 用户个人信息的修改在数据库中的体现

回到广播大厅之后，用户便可以在大厅发微博。

我们从电脑中选择并上传一张图片，然后配上文字：“这是我进行测试的微博”，如图所示：

我们发现，点击“发微博”按钮之后，用户的微博便出现在了广播大厅中，如图所示：

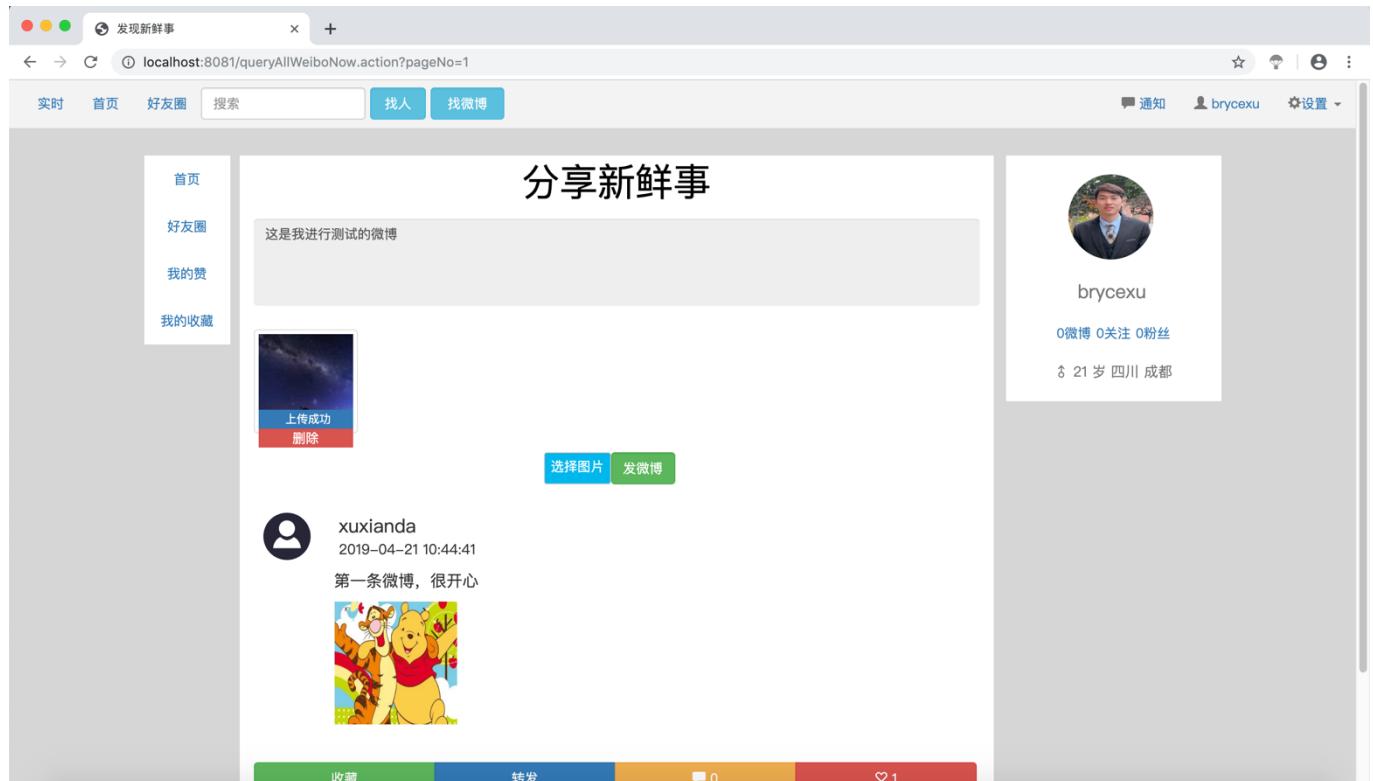


图 5.8 选择图片和文字进行微博的发表

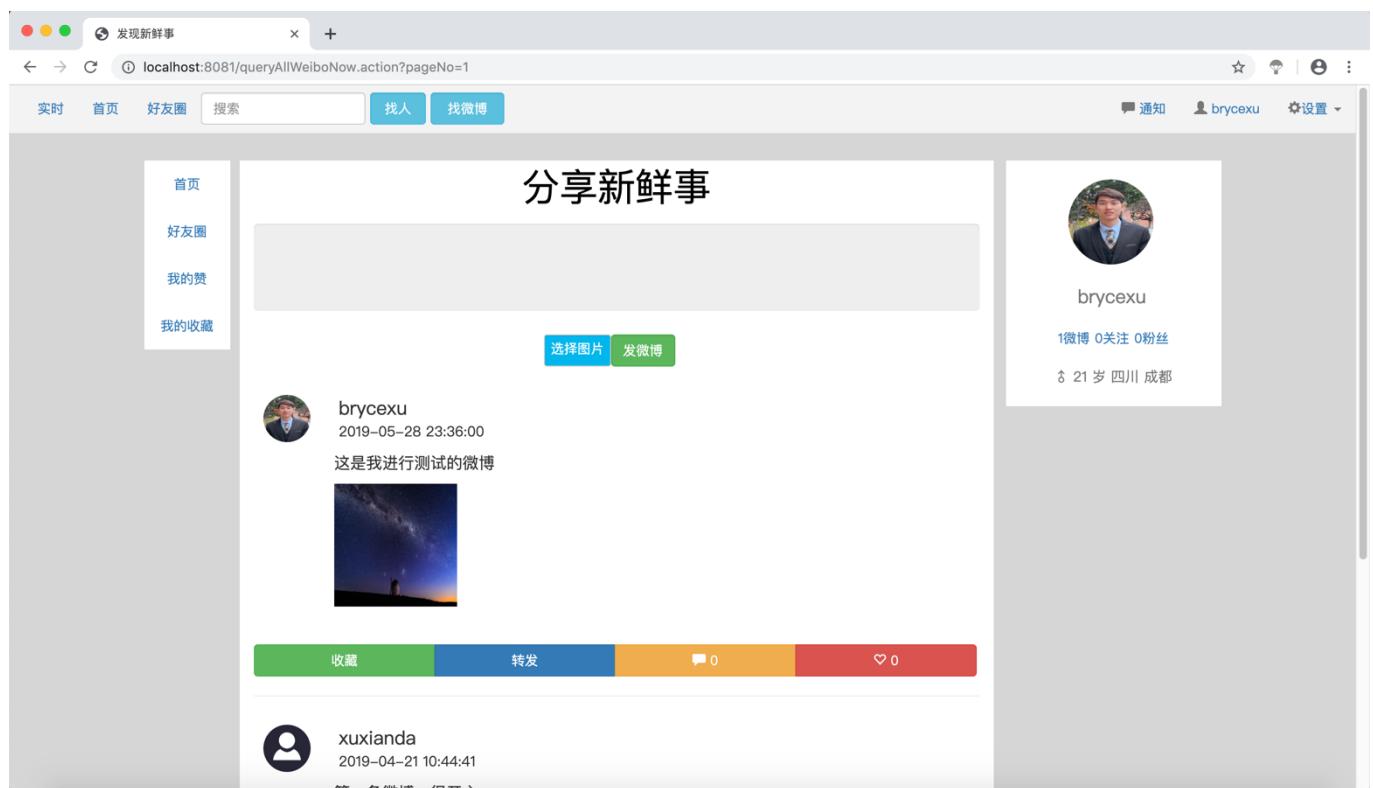


图 5.9 进行微博的发表

在测试发表微博功能之后，我们测试用户评论功能，我们对已经发表的一篇微博进行评论，评论的文字为：“这是我进行测试的评论”，如图所示：

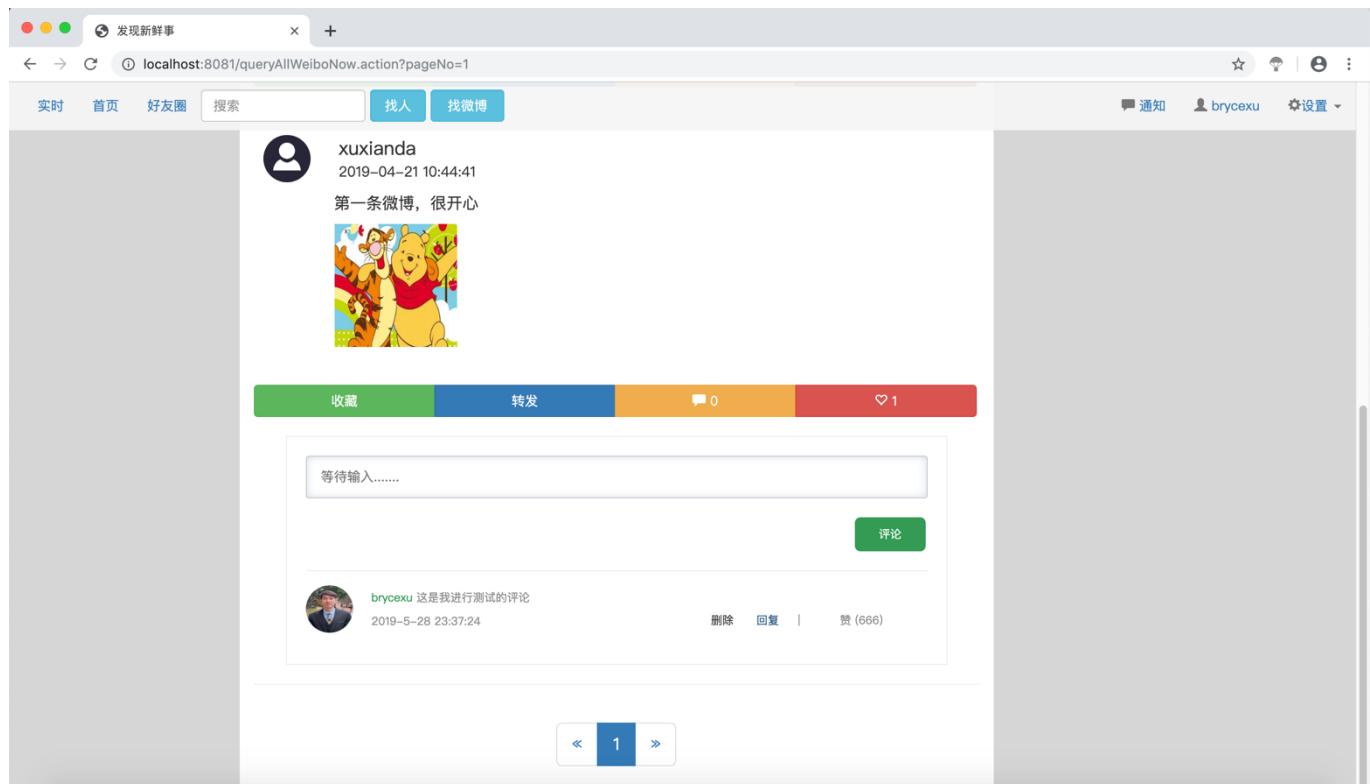


图 5.10 进行微博的评论

在测试评论微博功能之后，我们测试转发微博功能，我们对已经发表的一篇微博进行转发，该微博便成为了我们的微博，如图所示：

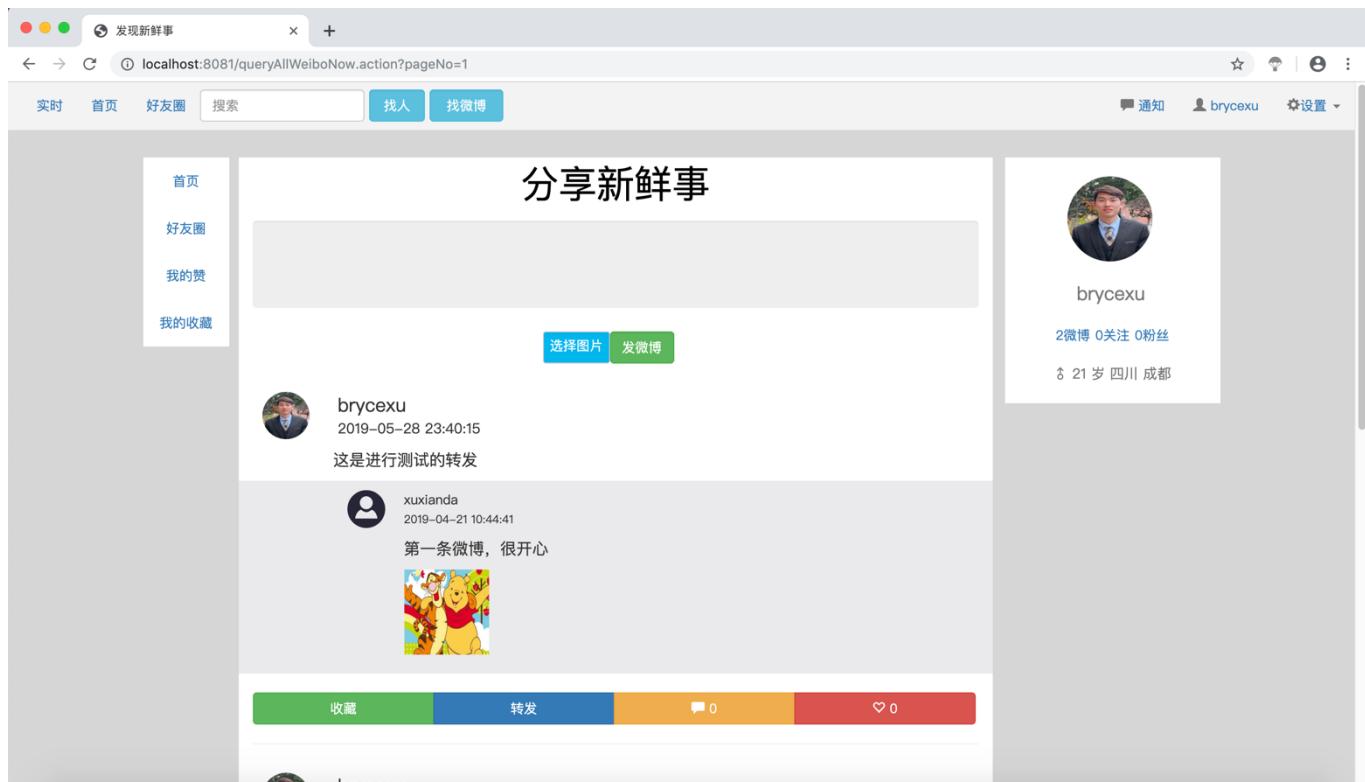


图 5.11 进行微博的评论

在测试转发微博功能之后，我们测试点赞微博功能，我们还是对同一微博进行点赞，如图所示：

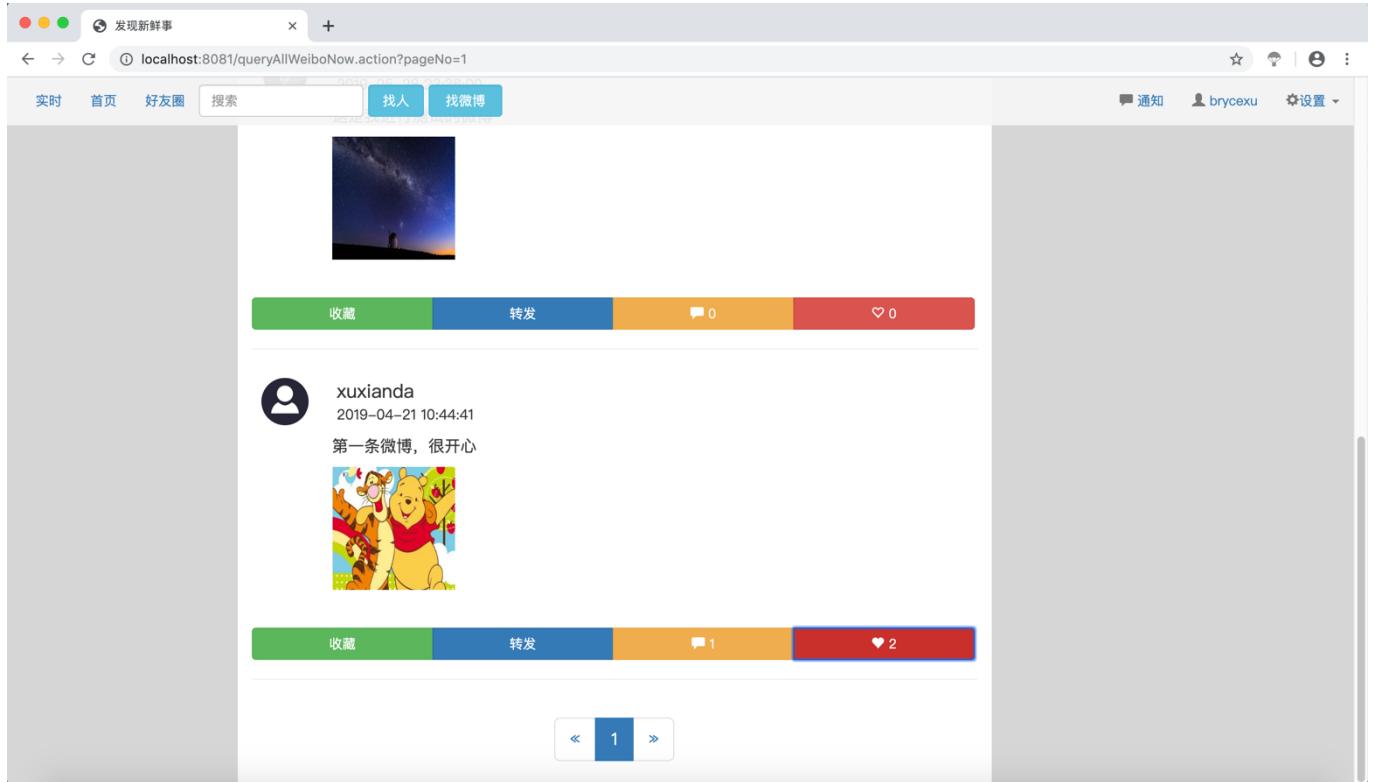


图 5.12 进行微博的点赞

在测试点赞微博功能之后，我们测试收藏微博功能，我们对此微博进行收藏，这样，在“我的收藏”页面，我们可以看到我们收藏的微博，如图所示：

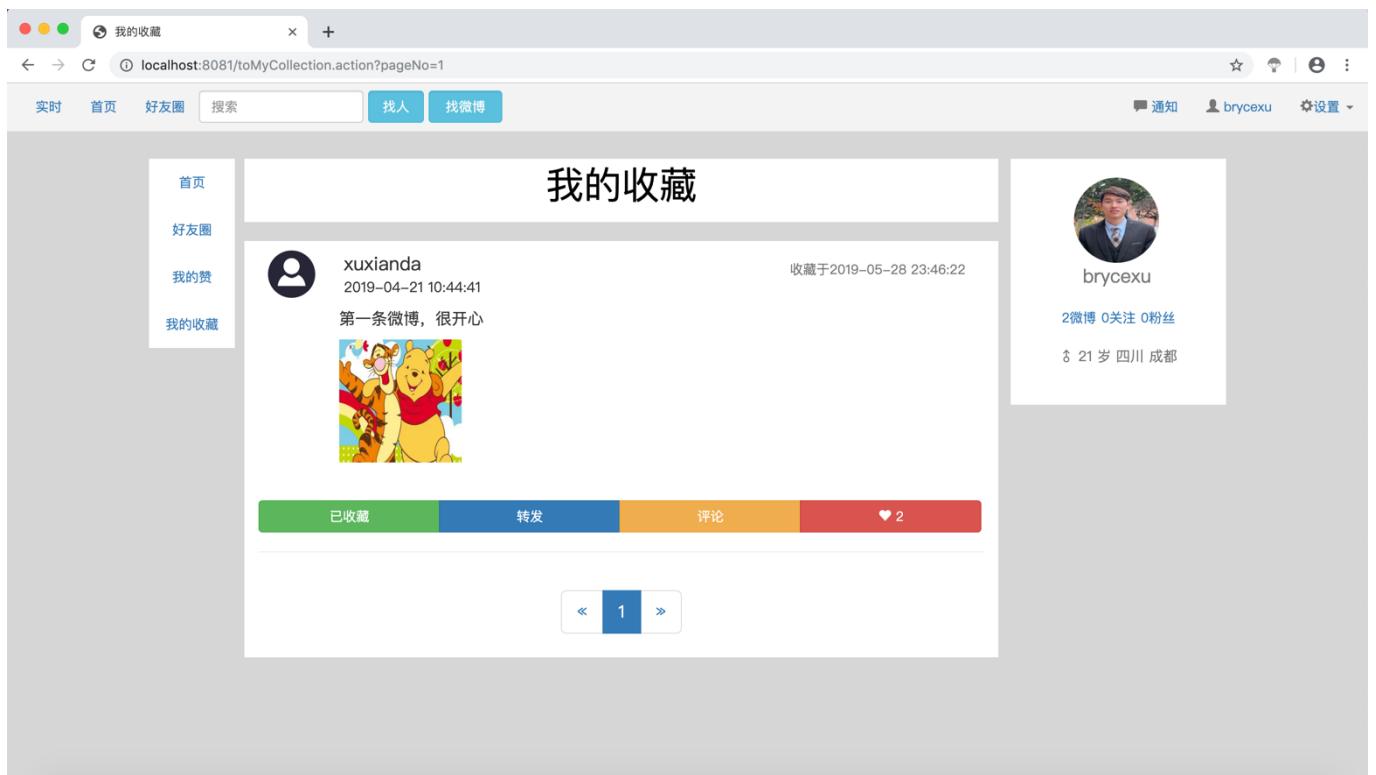


图 5.13 进行微博的收藏

在测试收藏微博功能之后，我们测试添加好友功能。本微博系统好友的要求是双方互相关注对方，则自动成为好友。我们先在当前用户下关注用户 xuxianda，如图所示：

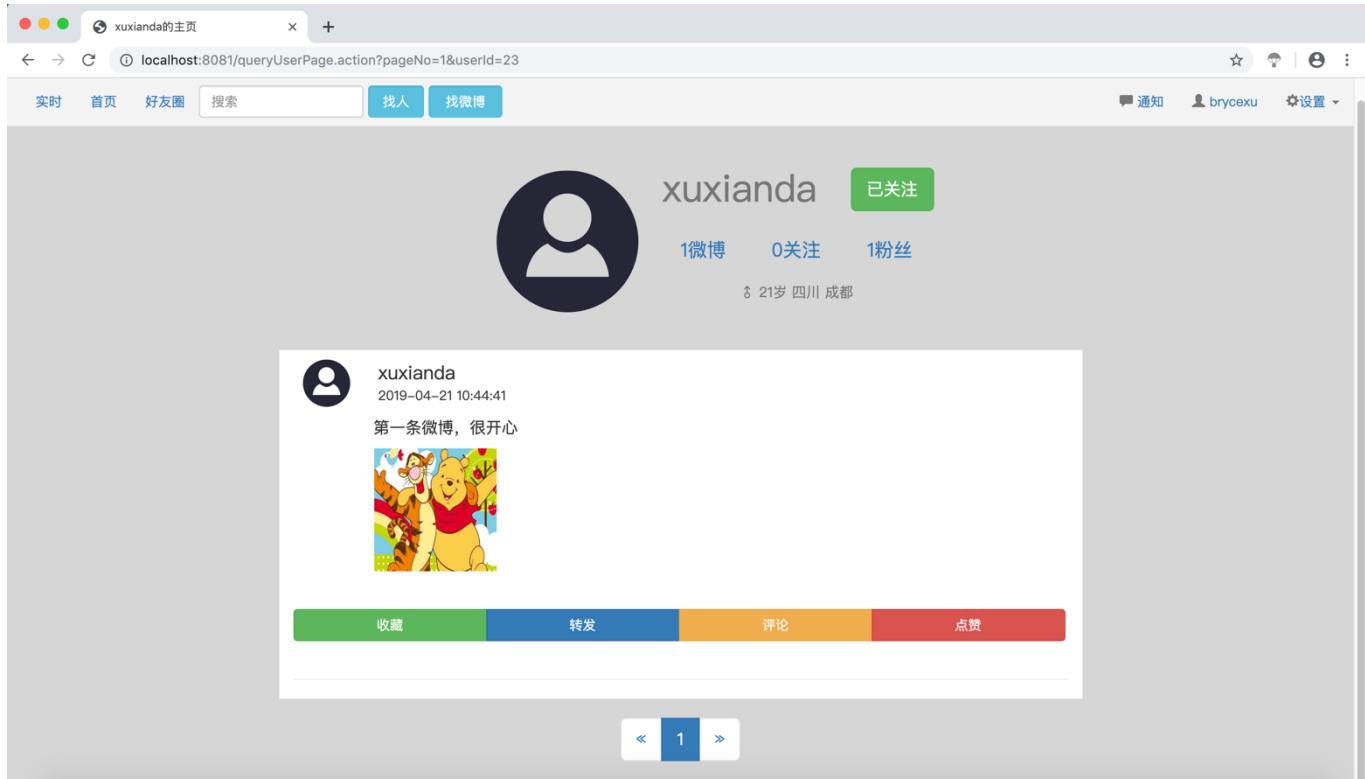
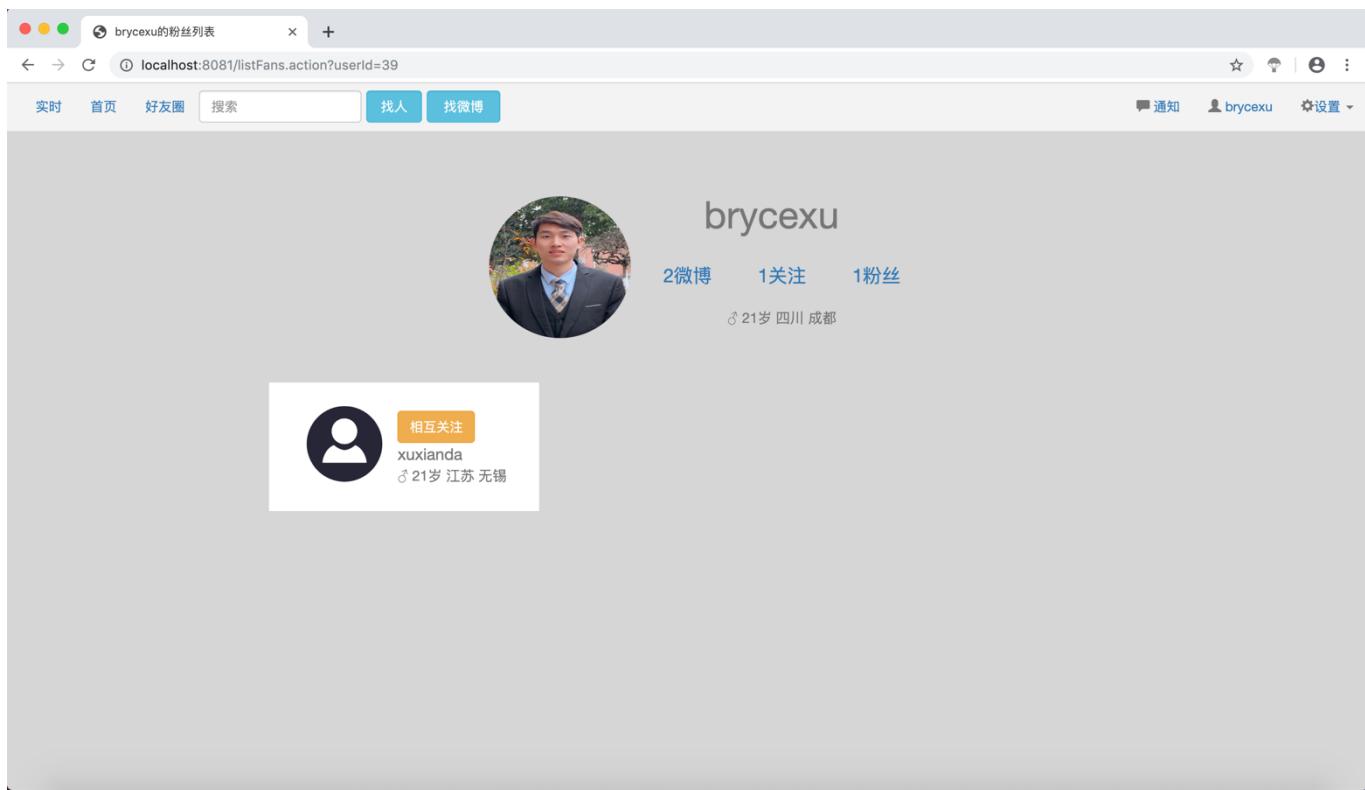


图 5.14 进行用户的关注

我们登录用户 xuxianda，关注用户 brycexu，再登录 brycexu 发现已经互相关注，如图所示，这样，在好友圈中，用户 brycexu 和用户 xuxianda 便可看到对方的微博。



最后，我们进行搜索的测试。

我们首先搜索用户，如果搜索“xu”，结果如下图所示：

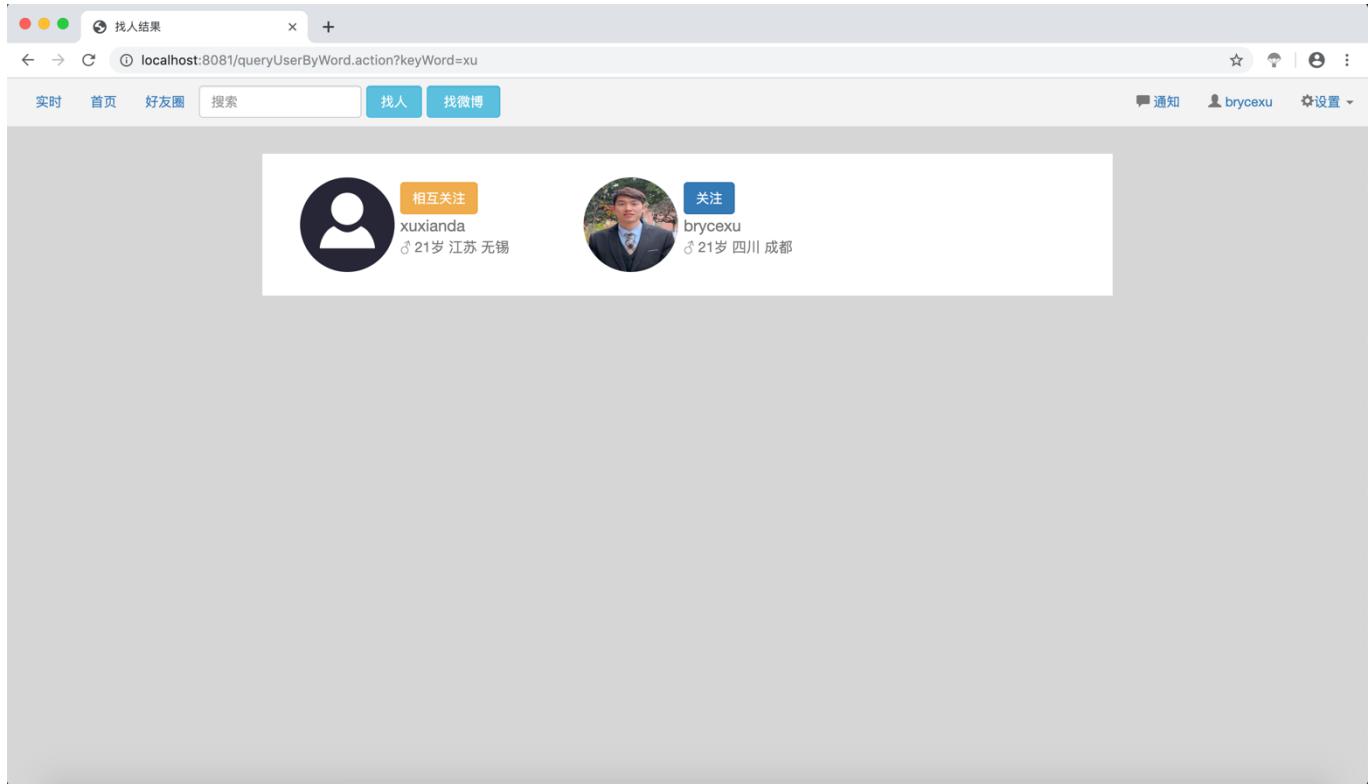


图 5.16 进行用户的搜索

我们然后搜索微博，如果搜索“转发”，结果如下图所示：

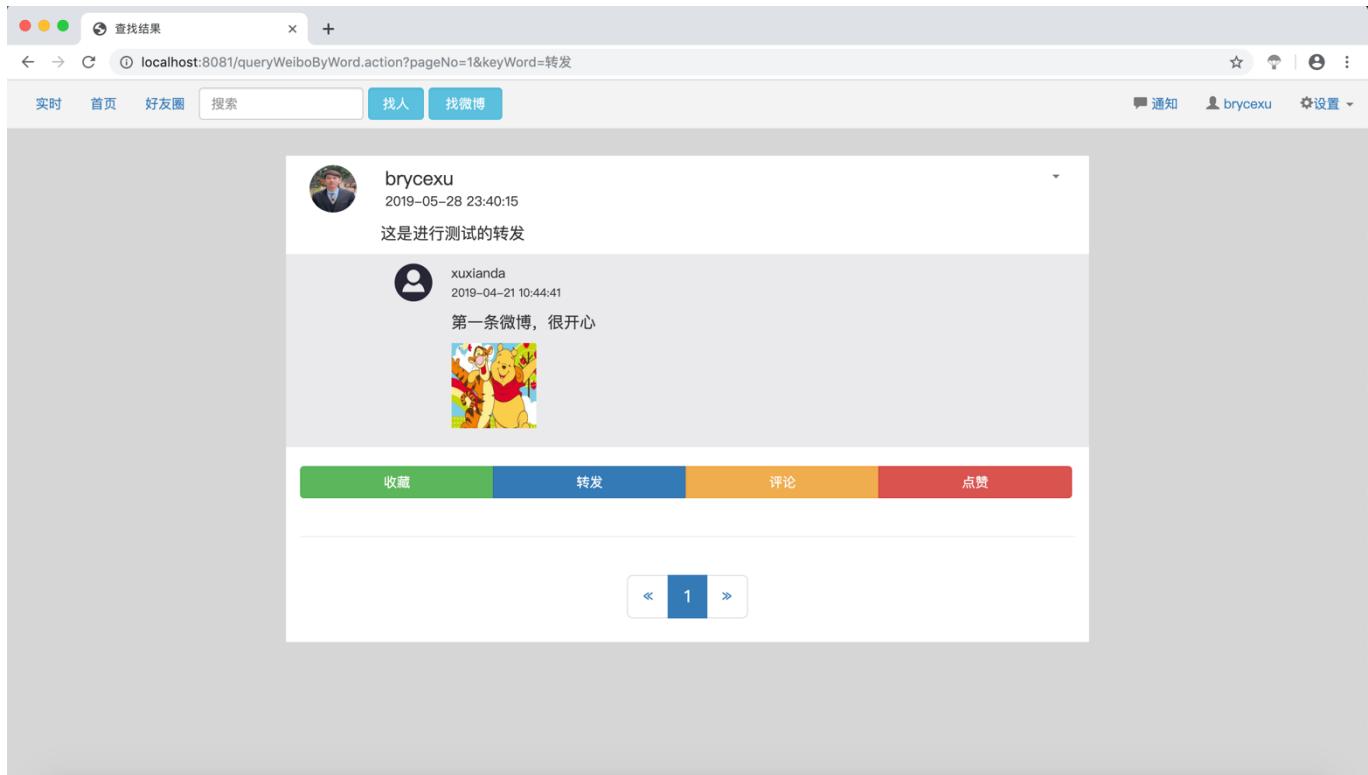


图 5.17 进行微博的搜索

### 5.3.3 管理员功能测试执行情况

管理员可以在后台系统中对用户和微博信息进行管理，下面我们再后台系统中对管理员的功能进行测试。

首先，我们在微博系统的欢迎界面中点击“管理员”，进入管理员登录界面，如下图所示：

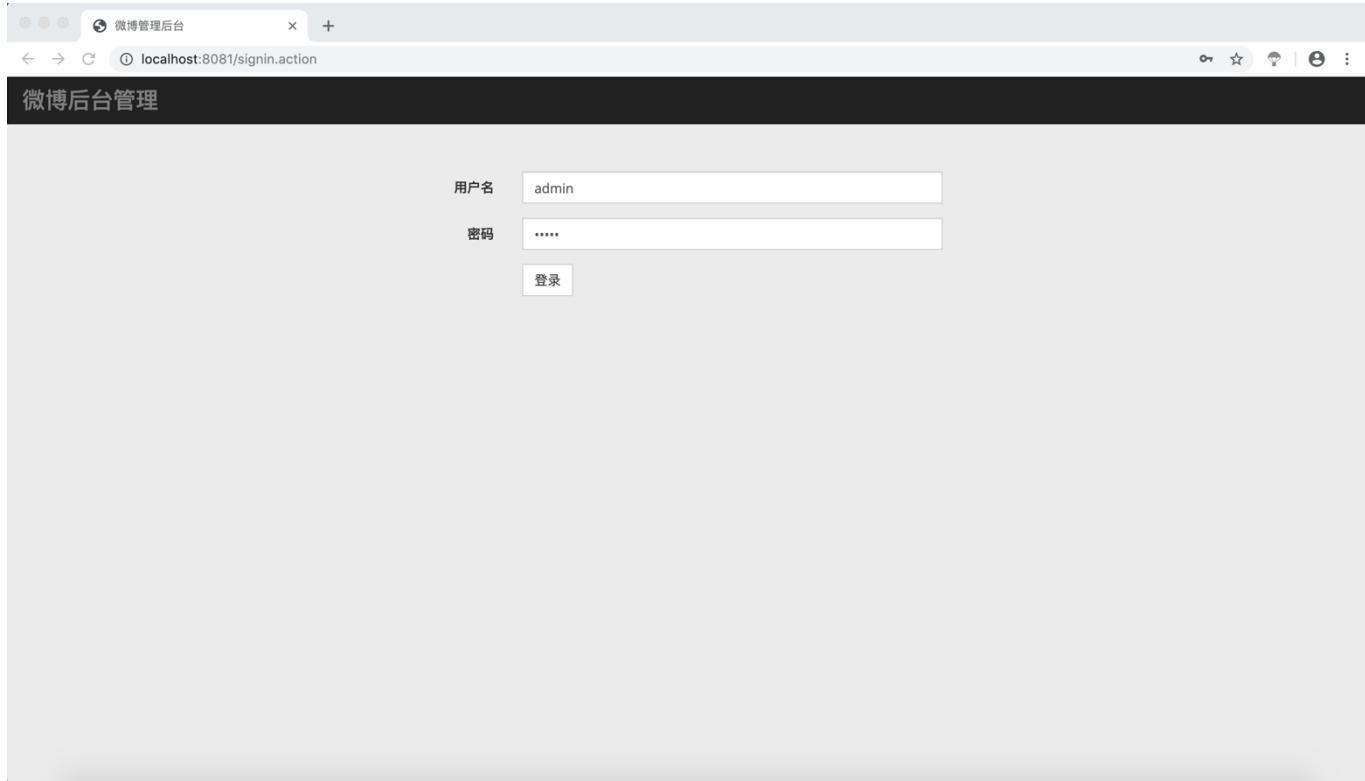


图 5.18 管理员登录界面

在微博管理界面，我们可以看到广播大厅中所有的微博，如下图所示：

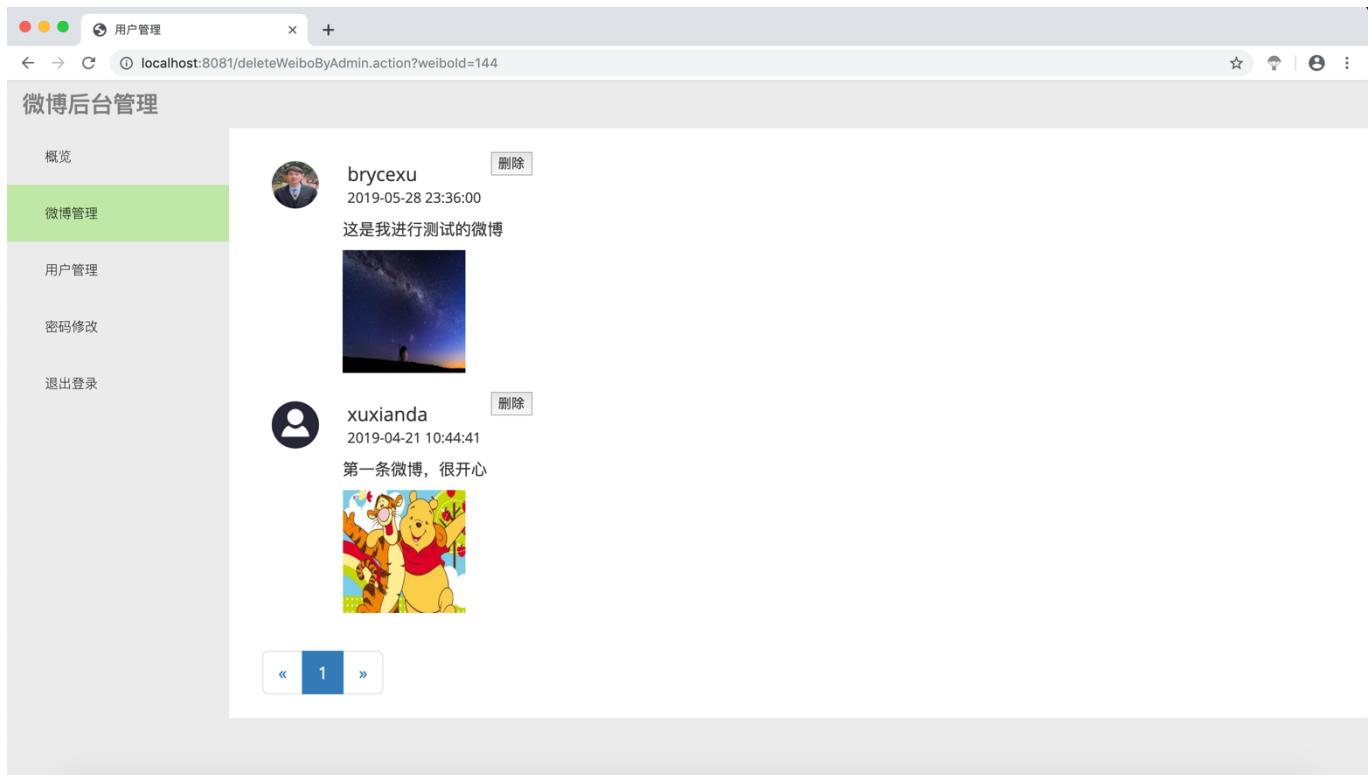
A screenshot of the "微博管理" (Weibo Management) interface. The left sidebar has tabs: "概览" (Overview), "微博管理" (Weibo Management, highlighted in green), "用户管理" (User Management), "密码修改" (Password Change), and "退出登录" (Logout). The main content area displays three tweets from users "brycexu" and "xuxianda".

- Tweet 1: User brycexu, posted on 2019-05-28 23:40:15. Content: "这是进行测试的转发".
- Tweet 2: User xuxianda, posted on 2019-04-21 10:44:41. Content: "第一条微博, 很开心". Includes a small cartoon image of Winnie the Pooh.
- Tweet 3: User brycexu, posted on 2019-05-28 23:36:00. Content: "这是我进行测试的微博". Includes a large image of a night sky with stars and a silhouette of a person.
- Tweet 4: User xuxianda, posted on 2019-04-21 10:44:41. Content: "第一条微博, 很开心". Includes a small cartoon image of Winnie the Pooh.

Each tweet has a "删除" (Delete) button to its right.

图 5.19 管理员微博管理界面

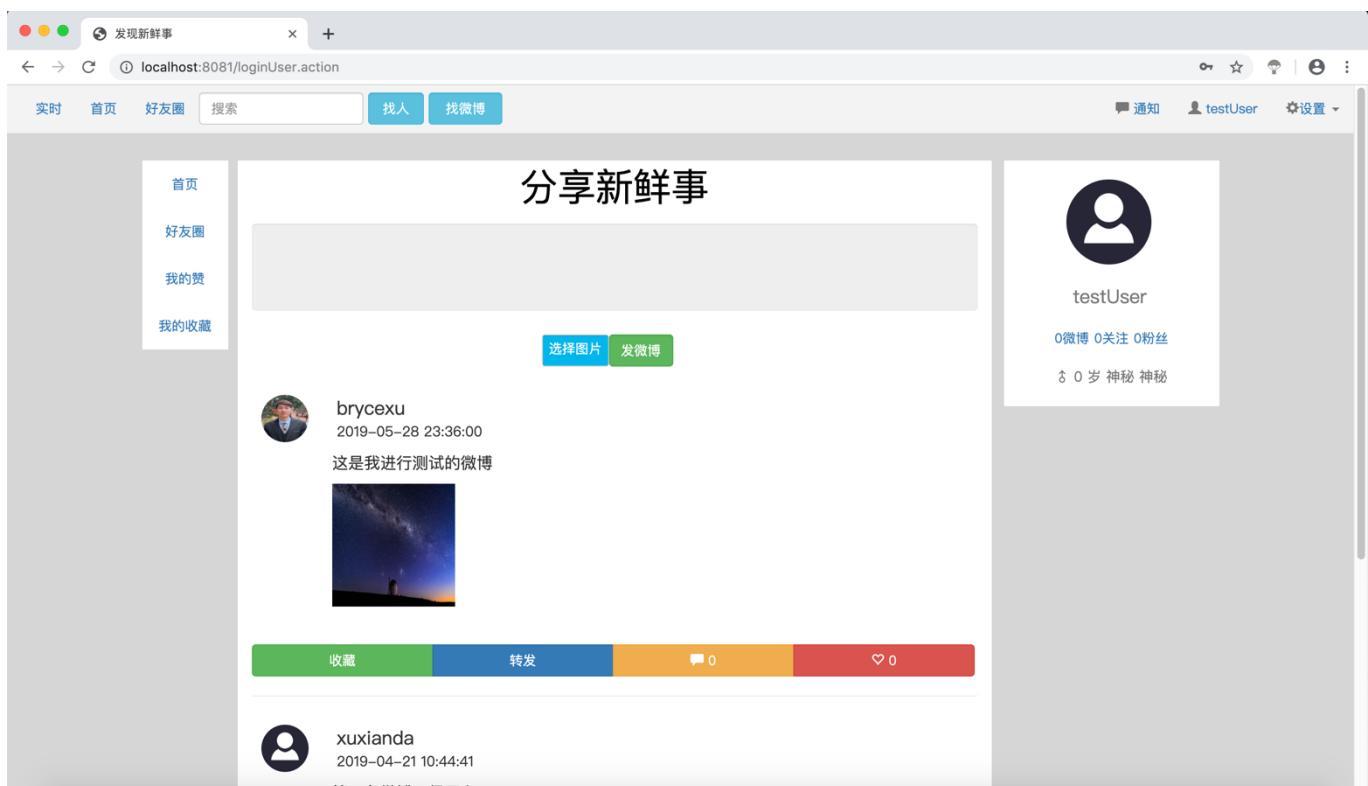
管理员可以对微博进行管理，这里，我们假设管理员需要删除最上面转发的微博，则点击右侧“删除”按钮，将微博删除，删除的结果如下图所示：



The screenshot shows a web-based administration interface for managing Weibo posts. On the left, a sidebar lists '概览', '微博管理' (selected), '用户管理', '密码修改', and '退出登录'. The main content area displays two tweets. The first tweet is from 'brycexu' at 2019-05-28 23:36:00, with the text '这是我进行测试的微博' and an image of a night sky. The second tweet is from 'xuxianda' at 2019-04-21 10:44:41, with the text '第一条微博, 很开心' and an image of Winnie the Pooh. Each tweet has a 'Delete' button to its right. At the bottom, there is a navigation bar with a blue '1' button.

图 5.20 管理员微博删除在后台中的结果

为了测试管理员用户管理功能，我们新建微博用户，用户名是 testUser，登录微博系统之后，发现被删除的微博被成功删除，如下图所示：



The screenshot shows a Weibo frontend interface. At the top, a header bar includes links for '实时', '首页', '好友圈', '搜索', '找人', '找微博', '通知', 'testUser' (logged in), and '设置'. On the left, a sidebar has buttons for '首页', '好友圈', '我的赞', and '我的收藏'. The main content area features a '分享新鲜事' (Share Fresh News) input field with '选择图片' and '发微博' buttons. Below this, a tweet from 'brycexu' is displayed, while the tweet from 'xuxianda' is missing. The right side shows a user profile for 'testUser' with a large placeholder icon, 0 Weibo, 0 Following, 0 Fans, and a status of '0岁 神秘 神秘'.

图 5.21 管理员微博删除在微博系统中的结果

我们再次进入管理员后台，测试管理员对用户进行管理的功能。我们再用户管理界面可以看到微博系统中所有微博用户的个人信息，如下图所示：

用户名	密码	性别	年龄	籍贯	操作
xuxianda	6403838	男	21	江苏无锡	<button>删除</button>
visitor	visitor	男	99	四川成都	<button>删除</button>
wujunliang	123456	男	1	神秘神秘	<button>删除</button>
duolali	6403838	男	0	神秘神秘	<button>删除</button>
brycexu	6403838	男	21	四川成都	<button>删除</button>
testUser	123456	男	0	神秘神秘	<button>删除</button>

图 5.22 管理员在用户管理界面查看所有微博用户个人信息

作为测试，我们删除新注册的用户 testUser，在数据库中验证我们的操作，如下图所示：

```
XuXianda — mysql -u root -p — 130x30
| weibo      |
+-----+
10 rows in set (0.00 sec)

[mysql]> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | username | password | nickname | face          | sex | bir   | province | city  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 23    | xuxianda | 6403838 | xuxianda | 639ff4a1-78ed-40b8-96a6-27c7c2e7081c.png | 1  | 1998-03-07 | 13  | 1303  |
| 33    | visitor   | visitor   | visitor   | 83df1f63-b1fe-48e0-b6ef-7d57c066b05c.png | 1  | 1920-06-05 | 24  | 2401  |
| 34    | wujunliang | 123456   | wujunliang | default.png                                | 1  | 2018-06-06 | 00  | 00    |
| 37    | duolali   | 6403838 | duolali   | default.png                                | 1  | 2019-01-01 | 00  | 00    |
| 39    | brycexu   | 6403838 | brycexu   | 4f4bde23-6090-48d1-b781-e4071e89915f.JPG | 1  | 1998-03-14 | 24  | 2401  |
| 40    | testUser   | 123456   | testUser   | default.png                                | 1  | 2019-01-01 | 00  | 00    |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

[mysql]> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | username | password | nickname | face          | sex | bir   | province | city  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 23    | xuxianda | 6403838 | xuxianda | 639ff4a1-78ed-40b8-96a6-27c7c2e7081c.png | 1  | 1998-03-07 | 13  | 1303  |
| 33    | visitor   | visitor   | visitor   | 83df1f63-b1fe-48e0-b6ef-7d57c066b05c.png | 1  | 1920-06-05 | 24  | 2401  |
| 34    | wujunliang | 123456   | wujunliang | default.png                                | 1  | 2018-06-06 | 00  | 00    |
| 37    | duolali   | 6403838 | duolali   | default.png                                | 1  | 2019-01-01 | 00  | 00    |
| 39    | brycexu   | 6403838 | brycexu   | 4f4bde23-6090-48d1-b781-e4071e89915f.JPG | 1  | 1998-03-14 | 24  | 2401  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

图 5.23 管理员删除用户在数据库中的验证

通过数据库的验证，管理员删除用户功能测试成功。

最后，我们测试管理员密码的修改，界面如下图所示：

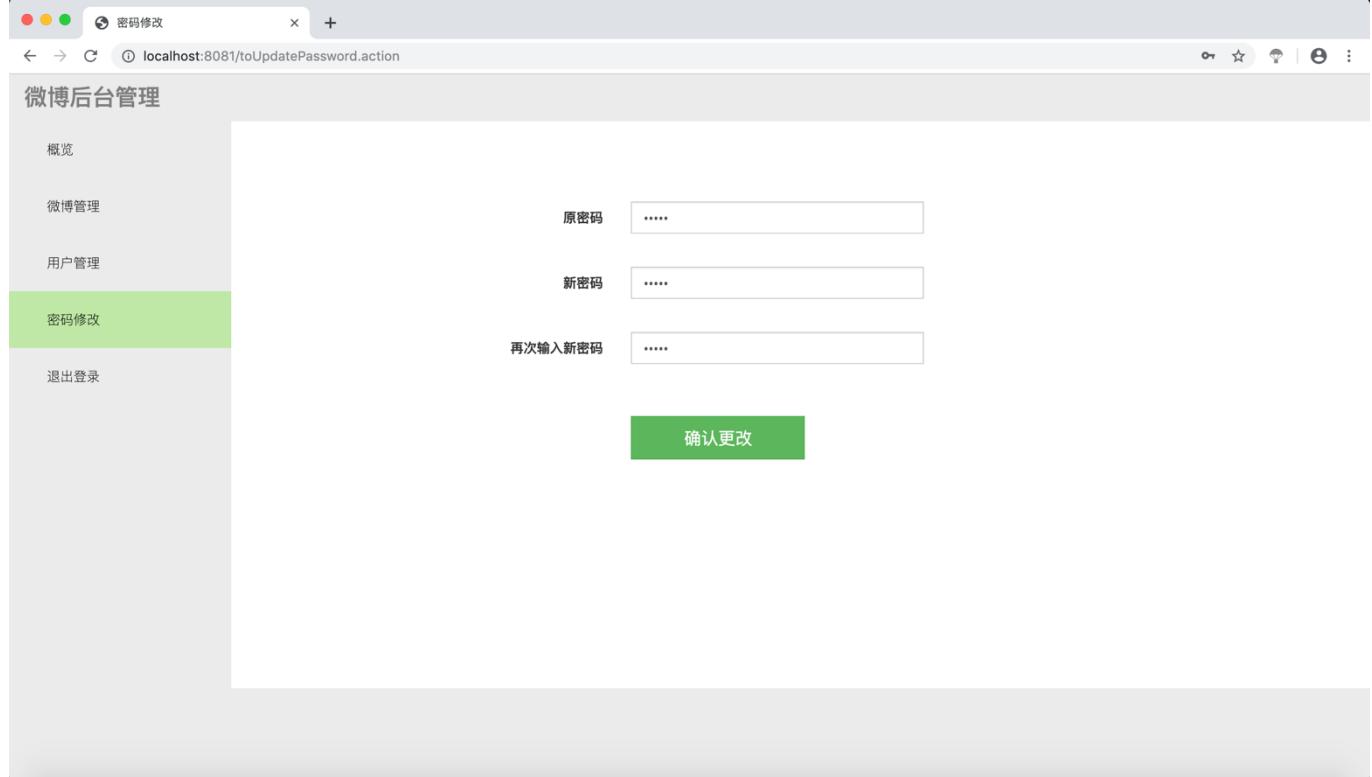


图 5.24 管理员密码修改界面

经测试，密码修改测试成功。

# 第六章 线上测试

## 6.1 测试目标

本小组对上线的微博系统进行线上测试。

## 6.2 测试环境

云服务器：阿里云

操作系统：Windows Server 2019 数据中心版

虚拟服务器：Tomcat 8.5

CPU：1 核

内存：2 GiB

带宽：1 Mbps

公网 IP：47.103.63.146

## 6.3 测试

在浏览器中输入微博系统网址：<http://47.103.63.146:8080/weibo>

欢迎界面如下图所示：

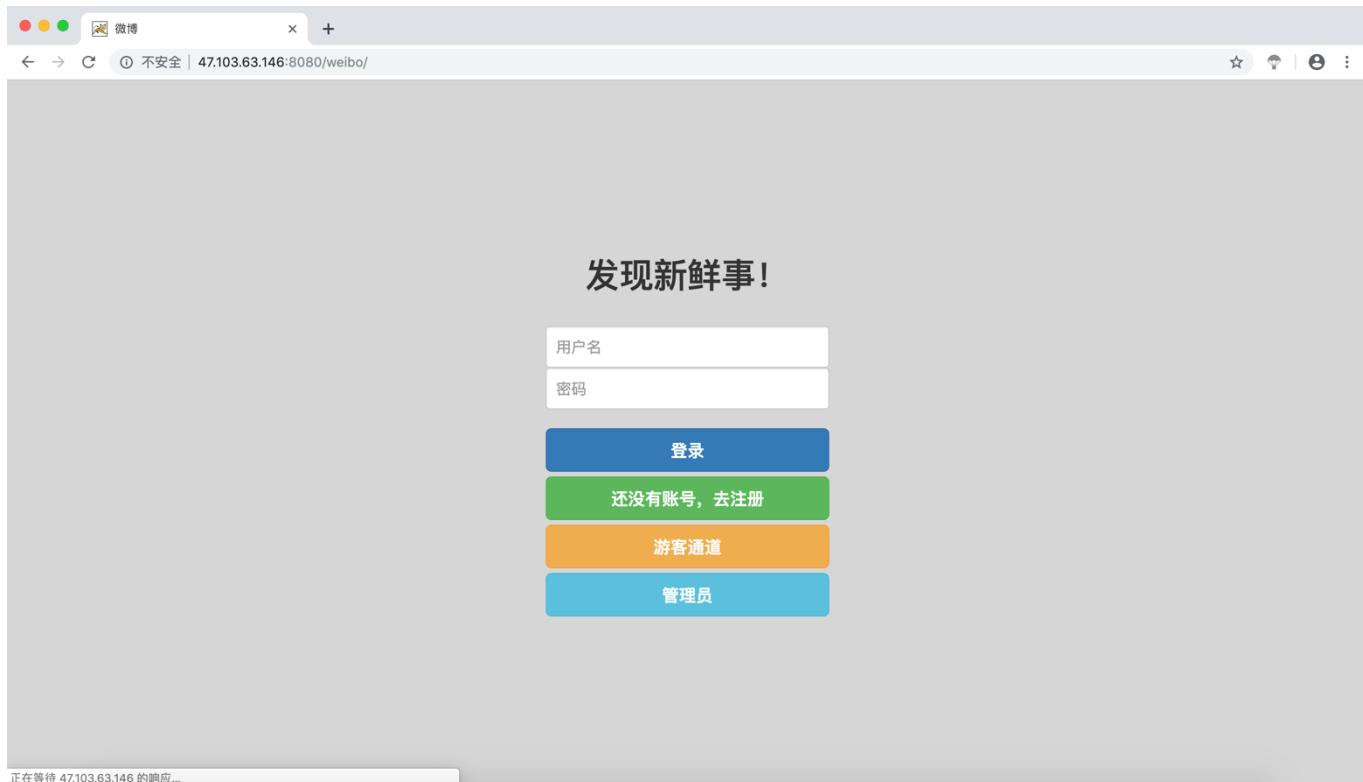


图 6.1 线上测试的欢迎界面

成功进入广播大厅，如下图所示：

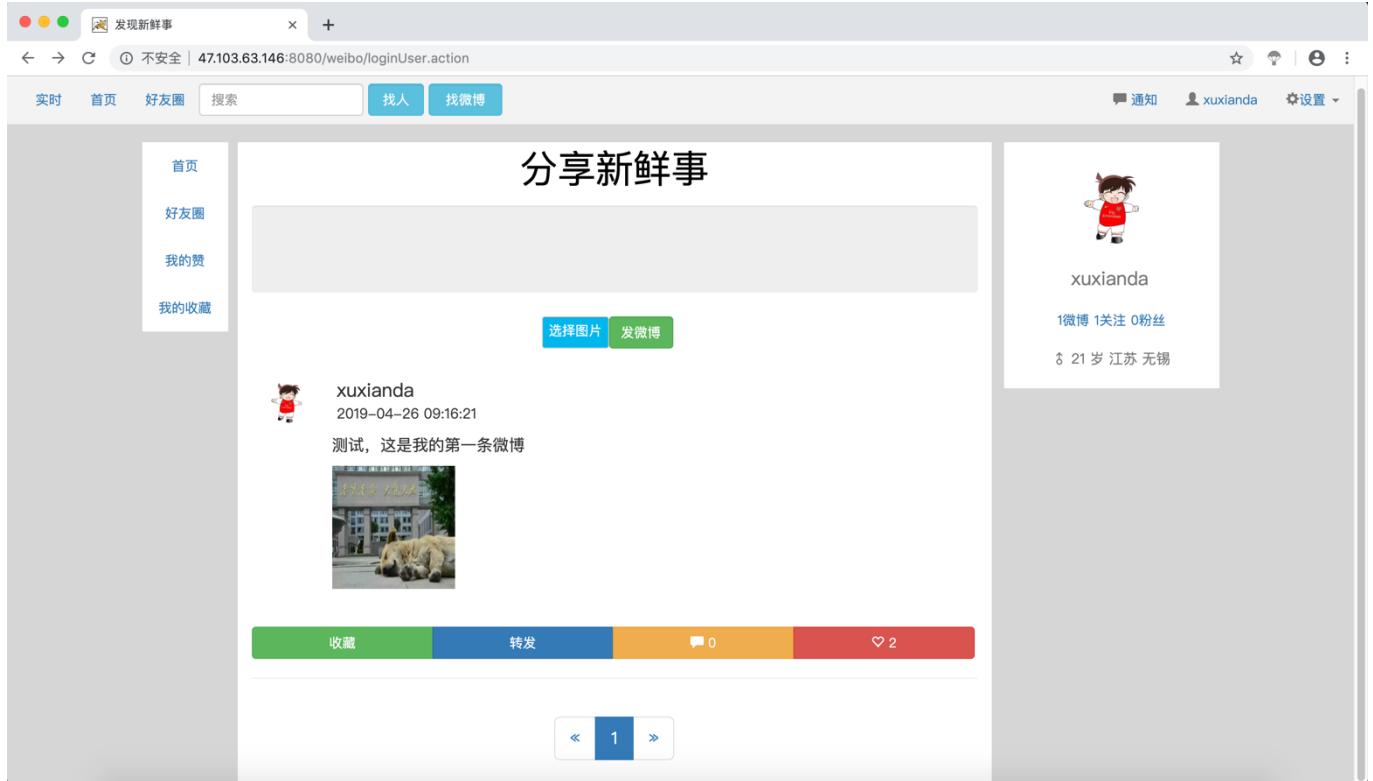


图 6.2 线上测试的广播大厅

线上测试成功。

# 第七章 总结与展望

## 7.1 总结

微博系统为广大的微博用户带来了良好的娱乐效果，也是一个沟通与交友的平台。本小组设计的微博系统面向游客、微博用户和管理员这三类群体，主要实现了以下的功能：

1. 游客浏览微博功能。
2. 游客注册为微博用户功能。
3. 微博用户发表微博、评论微博、点赞微博、转发微博、收藏微博功能。
4. 微博用户修改个人信息功能。
5. 微博用户关注其他用户与添加好友功能。
6. 管理员对微博和用户进行管理功能。
7. 管理员修改后台登录密码功能。

在系统测试中，我们在本地机器对于这些功能进行了逐一测试，测试结果良好。在线上测试中，我们对于我们部署在云服务器上的微博系统进行测试，测试结果良好。

由于时间仓促及本小组技术能力有限，微博系统的功能并不是很完善，系统中还有很多功能有待开发扩展，希望能够在以后的工作和学习中不断地去完善与改进。

## 7.2 展望

随着人们的需求不断地在变更，本小组设计的微博系统还有待在以下几个方面做进一步的发展与完善：

1. 增加好友之间的聊天功能，使得好友间的沟通更加方便。
2. 增加管理员在广播大厅的全员通知功能，使得管理员能更方便地通知所有微博用户。

# 参考文献

- [1] WALLS C, ZHANG W. Spring 实战[M]. 第 4 版. 北京: 人民邮电出版社, 2016.
- [2] Accessing data with MySQL[EB/OL]. Spring.io, 2019. (2019). <https://spring.io/guides/gs/accessing-data-mysql/>.
- [3] Spring Projects[EB/OL]. Spring.io, 2019. (2019). <https://spring.io/projects/spring-boot/>.
- [4] 明日科技. Java Web 项目开发实战入门.第 1 版.长春:吉林大学出版社, 2017.