

**Web开发技术基础课程设计报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **班级** | **姓名** | **学号** | **学院** | **专业** | **分工** |
| 2018211314 | 陈姝仪 | 2018211507 | 计算机学院 | 网络工程 | 1.用户注册、登录、登出  2.游客模式的实现  3.微博评论  4.微博点赞 |
| 2018211313 | 曲珍莹 | 2018211166 | 计算机学院 | 网络工程 | 1.微博发布与储存  2.微博显示、分页、排序  3.发博时图片的处理  4.从微博进入发布者个人主页的实现 |
| 2018211314 | 沙尔娜 | 2018211294 | 计算机学院 | 网络工程 | 1.用户个人主页的实现  2.关注和粉丝的实现 |

**2021年1月5日**

**目录**

[**1.设计任务的描述** 4](#_Toc60781446)

[**2.总体方案设计说明** 4](#_Toc60781447)

[**2.1开发环境** 4](#_Toc60781448)

[**2.2功能模块划分** 4](#_Toc60781449)

[**2.3涉及到的技术** 4](#_Toc60781450)

[**3.各模块设计说明** 5](#_Toc60781451)

[**3.1用户账号服务** 5](#_Toc60781452)

[**3.1.1设计思路** 5](#_Toc60781453)

[**3.1.2后端实现** 5](#_Toc60781454)

[ 用拦截器实现权限管理 5](#_Toc60781455)

[ 注册 5](#_Toc60781456)

[ 登录 6](#_Toc60781457)

[ 登出 7](#_Toc60781458)

[**3.2微博发布** 8](#_Toc60781459)

[**3.2.1设计思路** 8](#_Toc60781460)

[**3.2.2前端代码** 8](#_Toc60781461)

[**3.2.3后端代码** 11](#_Toc60781462)

[**3.2.4样式展示** 12](#_Toc60781463)

[**3.3微博查看** 12](#_Toc60781464)

[**3.3.1设计思路** 12](#_Toc60781465)

[**3.3.2前端代码** 13](#_Toc60781466)

[**3.3.3后端代码** 14](#_Toc60781467)

[**3.3.4样式展示** 15](#_Toc60781468)

[**3.4微博点赞** 16](#_Toc60781469)

[**3.4.1设计思路** 16](#_Toc60781470)

[**3.4.2前端代码** 16](#_Toc60781471)

[**3.4.3后端代码** 17](#_Toc60781472)

[**3.5微博评论** 17](#_Toc60781473)

[ 查看评论 17](#_Toc60781474)

[**3.5.1设计思路** 17](#_Toc60781475)

[**3.5.2前端请求代码** 19](#_Toc60781476)

[**3.5.3后端代码** 19](#_Toc60781477)

[ 发表评论 19](#_Toc60781478)

[**3.5.1设计思路** 19](#_Toc60781479)

[**3.5.2前端代码** 20](#_Toc60781480)

[**3.5.3后端代码** 20](#_Toc60781481)

[**3.6关注和粉丝** 20](#_Toc60781482)

[**3.6.1设计思路** 20](#_Toc60781483)

[**3.6.2前端代码** 21](#_Toc60781484)

[**3.6.3后端代码** 22](#_Toc60781485)

[**3.6.4样式展示** 23](#_Toc60781486)

[**3.7个人主页** 24](#_Toc60781487)

[**3.7.1设计思路** 24](#_Toc60781488)

[**3.7.2前端代码** 24](#_Toc60781489)

[**3.7.3后端代码** 25](#_Toc60781490)

[**3.7.4样式展示** 26](#_Toc60781491)

[**3.8游客模式** 28](#_Toc60781492)

[**3.8.1设计思路** 28](#_Toc60781493)

[**3.8.2功能演示** 28](#_Toc60781494)

[**4.数据库说明** 30](#_Toc60781495)

[**4.1 Entity** 30](#_Toc60781496)

[**4.1.1 Sysrole** 30](#_Toc60781497)

[**4.1.2 Sysuser** 30](#_Toc60781498)

[**4.1.3 Message** 30](#_Toc60781499)

[**4.1.4 Comment** 31](#_Toc60781500)

[**4.1.5 Fan** 31](#_Toc60781501)

[**4.2 Repository** 32](#_Toc60781502)

[**4.2.1 Sysrole Repository** 32](#_Toc60781503)

[**4.2.2 Sysuser Repository** 32](#_Toc60781504)

[**4.2.3 Message Repository** 32](#_Toc60781505)

[**4.2.4 Comment Repository** 32](#_Toc60781506)

[**4.2.5 Fan Repository** 32](#_Toc60781507)

[**5.评价和改进意见** 32](#_Toc60781508)

[**5.1评价** 32](#_Toc60781509)

[**5.2心得体会** 33](#_Toc60781510)

[**5.3改进意见** 33](#_Toc60781511)

**1.设计任务的描述**

* 题目：简易版微博客系统的设计与实现
* 设定：设计实现一个简易的类“微博”的单体式Web应用。
* 基本功能要求：
* 用户账号服务：用户注册、用户登录、用户退出
* 微博查看：微博列表，支持按发布时间、评论数、点赞数等排序方式，微博列表可分页，未注册用户可以查看微博内容及评论。
* 微博发布：注册用户可以发布微博，微博内容可包含图片
* 评论：注册用户可对微博发布评论，点赞
* 关注：支持用户可以关注其他用户
* 扩展功能要求：
* 登录密码输入错误3次后账号锁定1小时
* 支持通过注册邮箱重置密码
* 支持@用户功能
* 采用AJAX技术减少界面的整体刷新次数

**2.总体方案设计说明**

**2.1开发环境**

本应用采用Intellij Idea 2020.2.3在win10环境下进行开发

**2.2功能模块划分**

* 用户账号服务（用户注册、登录、退出）
* 微博查看（微博列表，按发布时间、评论数、点赞数排序，分页，未注册用户可以查看微博内容及评论）
* 微博发布
* 微博点赞
* 微博评论
* 关注和粉丝
* 游客模式
* 个人主页

**2.3涉及到的技术**

* Thymeleaf
* H2 database
* Lombok
* Ajax
* JPA

**3.各模块设计说明**

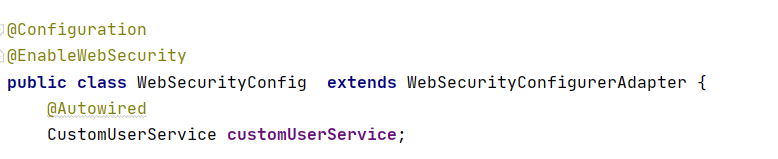
**3.1用户账号服务**

**3.1.1设计思路**

使用spring security的一整套解决方案。每一个系统用户都拥有自己的角色，将用户角色分为两类：user和tourist，user在登录之后拥有关注用户、发布微博、点赞微博、评论微博等一系列权限，而tourist仅拥有查看微博及评论、进入用户主页的权限。设置拦截器，根据当前用户访问的url实现权限管理，user用户可以访问所有url,tourist只能访问/tourist/\*\*路径下的资源

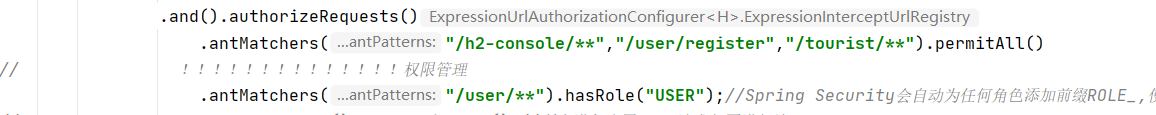
**3.1.2后端实现**

* 定义继承于WebSecurityConfigurerAdapter的类WebSecurityConfig

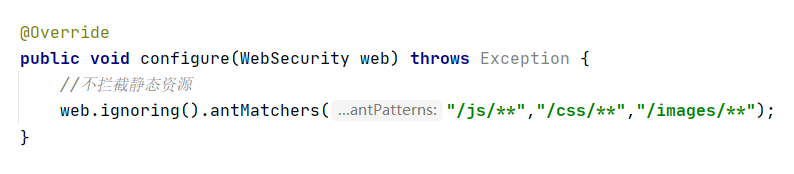


* 用拦截器实现权限管理

对于所有数据库访问请求、用户注册请求、游客访问请求放行，访问‘/user/\*\*’必须要有user权限，即当前用户已登录。

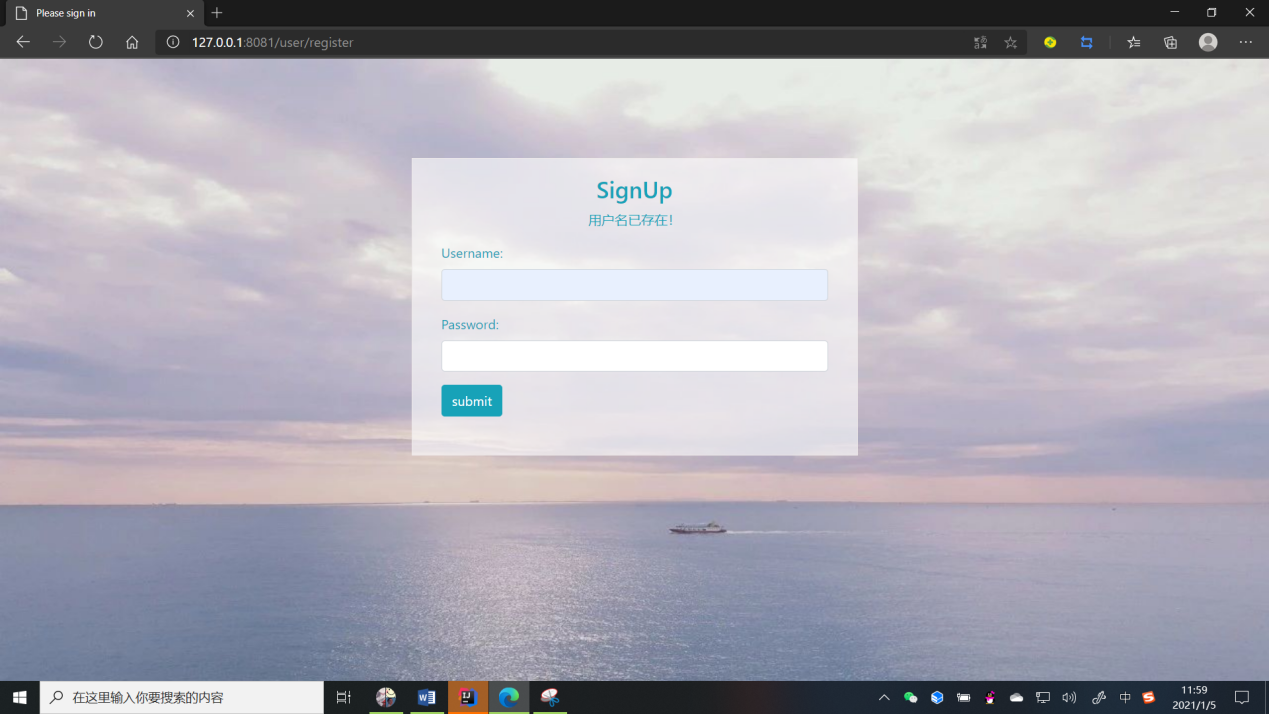


不拦截静态资源：



* 注册
  + 注册成功则为用户赋予user角色，并重定向到登录页面
  + 注册失败则返回“用户名已存在”的信息给前端

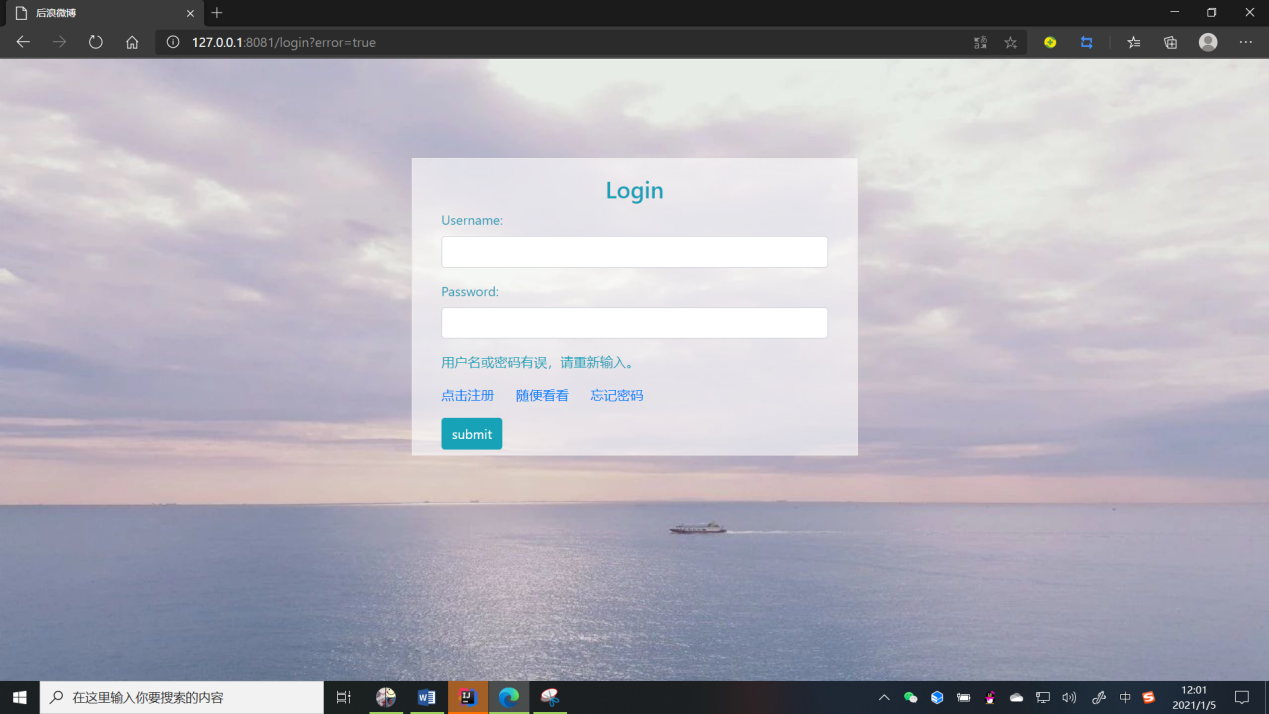




* 登录
  + 登录成功处理



* + 登录失败，AJAX提示错误信息

* 登出

重定向到登录页面



**3.2微博发布**

**3.2.1设计思路**

使用ajax实现在不刷新页面的情况下进行新发布微博的实时显示。同时在点击发布后ajax还会发post请求给”/user/home”，将微博内容等发给后端，存入h2数据库中。

在输入正文内容时，右上角会使用ajax技术计算当前输入字数并显示。

点击页面上的可以进行图片的选择，采用input标签的file进行图片的读入，点击可以看到预设表情并添加到正文框中，但目前表情还没有实现生成转义字符并存入数据库中。

**3.2.2前端代码**

**Html部分：**

<div class="col-sm-6 col-xs-12 my\_edit">  
 <div class="row" id="edit\_form">  
 <span class="pull-left" style="margin:15px;">编写新鲜事</span>  
 <span class="tips pull-right" style="margin:15px;"></span>  
 <form role="form" style="margin-top: 50px;">  
 <div class="form-group">  
 <div class="col-sm-12">  
 <div contenteditable="true" id="content" class="form-control "></div>  
 </div>  
 <div class="col-sm-12" style="margin-top: 12px;">  
 <span class="emoji">表情</span>  
  
 <span class="pic">图片 </span>  
 <span>  
 <input type="file" name="file" id="uploadImage" class="select\_Img" accept=".png, .jpg, .jpeg" style="display: none">  
 <img id="preview\_pic" class="preview" src="">  
 <a id="delpic" href="#">X</a>  
 </span>  
 <div class="myEmoji">  
 <ul id="myTab" class="nav nav-tabs">  
 <li class="active">  
 <a href="#set" data-toggle="tab">  
 预设  
 </a>  
 </li>  
 <li><a href="#hot" data-toggle="tab">热门</a></li>  
  
 </ul>  
 <div id="myTabContent" class="tab-content">  
 <div class="tab-pane fade in active" id="set">  
 <div class="emoji\_1"></div>  
 </div>  
 <div class="tab-pane fade" id="hot">  
 <div class="emoji\_2"></div>  
 </div>  
 </div>  
 </div><button type="button" id="send" class="btn btn-default pull-right disabled">发布</button>  
 </div>  
 </div>  
 </form>  
 </div>

**判断输入的字符串长度：**

$("#content").keyup(function () {  
 *//判断输入的字符串长度* var content\_len = $("#content").text().replace(/\s/g, "").length;  
  
 $(".tips").text("已经输入" + content\_len + "个字");  
  
 if (content\_len == 0) {  
 *// alert(content);* $(".tips").text("");  
 $("#send").addClass("disabled");  
 return false;  
 } else {  
 $("#send").removeClass("disabled");  
 }  
});

**图片预览及删除：**

//点击图片选定后实现图片预览

$(".pic").click(function () {  
 $(".select\_Img").click();  
})

var objUrl="#"; *//存放上传图片路径*$('.select\_Img').on('change',function(){*//图片上传* var fileList = this.files;  
  
 objUrl = getObjectURL(fileList[0]) ;  
 if (objUrl) {  
 $('#preview\_pic').show();  
 $('#preview\_pic').attr('src',objUrl);  
 $('#preview\_pic').attr('style','width:100px;height:70px');  
 }  
})

//删除上传图片  
$("#delpic").click(function (event){  
 objUrl="#";  
 $('#preview\_pic').hide();

document.getElementById('file').value='';

})

**发布微博及post请求：（新发的微博通过ajax显示在第一条，包括添加的图片）**

*//点击按钮发送内容*$("#send").click(function () {  
 var time = new Date();  
  
 var content = $("#content").text();  
 var uname = $("#uname").text();  
  
 if (objUrl != "#")  
 {  
 $("#new\_msg").html("<div class='col-sm-12 col-xs-12 message' > <img src='/images/icon.png' class='col-sm-2 col-xs-2' style='border-radius: 50%'><div class='col-sm-10 col-xs-10'><span style='font-weight: bold;'>" + uname + "</span> <br><small class='date' style='color:#999'>刚刚</small><div class='msg\_content'>" + content + "</div><img class='mypic' src="+objUrl+" ></div></div>");  
 $("#new\_msg").show();  
 }  
 else {  
 $("#new\_msg").html("<div class='col-sm-12 col-xs-12 message' > <img src='/images/icon.png' class='col-sm-2 col-xs-2' style='border-radius: 50%'><div class='col-sm-10 col-xs-10'><span style='font-weight: bold;'>" + uname + "</span> <br><small class='date' style='color:#999'>刚刚</small><div class='msg\_content'>" + content + "</div></div></div>");  
 $("#new\_msg").show();  
 }  
 var formData = new FormData();  
 formData.append("file", $("#uploadImage")[0].files[0]);  
 *alert*("content")  
 $.ajax({  
 type:"POST",*//提交数据的类型 POST/GET* url:"/user/home",*//提交的网址* data:{  
 "content":content,  
 "time":time,  
 "thumbUp":0,  
 "cmtNum":0},*//提交的数据* dataType: "json",*//"xml", "html", "script", "json", "jsonp", "text".//返回数据的格式* success : function(data) {  
 $("#content").empty(); *// 清空resText里面的所有内容*  
 },  
 error: function(){*//请求出错处理  
 alert*("发送出错");  
 }  
 });  
});

**3.2.3后端代码**

@PostMapping("/home")  
void get\_message(@RequestParam(value = "content") String content,  
 @RequestParam(value = "time") Date time,  
 @RequestParam(value = "thumbUp") Integer thumbUp,  
 @RequestParam(value = "cmtNum") Integer cmtNum) {  
 *//获取当前用户名和id* SysUser uid = (SysUser) SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  
 Long userid = uid.getId();  
 String uname = uid.getUsername();  
 Message m = new Message();  
  
 m.setContent(content); *//正文内容*  
 m.setTime(time); *//发布时间*  
 m.setUserid(userid); *//发布者id*  
 m.setThumbUp(thumbUp);*//起始点赞数0*  
 m.setName(uname); *//发布者用户名*  
 m.setCmtNum(cmtNum);*//起始评论数0*  
  
 messageRepository.save(m);  
}

**3.2.4样式展示**





**3.3微博查看**

**3.3.1设计思路**

结合Thymeleaf的元素迭代和数据库，实现微博列表的显示。

使用page对象进行分页的实现，并且通过修改sort的参数可以实现按照不同的字段进行排序。而这个参数的获取是靠点击排序的超链接，如果点击按时间排序，在href的超链接上会添加上一个参数sort\_method=time/thumbUp/cmtNum，后台接收到Get请求后，会根据sort\_method参数进行对应方式的排序，sort\_method的缺省值为time，即默认按照时间顺序排序。

分页的实现是在页面下方添加几个分页按钮，结合thymeleaf和page对象进行分页操作的实现。

**3.3.2前端代码**

**微博列表的显示**

<div class="row item\_msg" th:each="message:${messages}">  
 <div class="col-sm-12 col-xs-12 message">  
 <img src="/images/icon.png" class="col-sm-2 col-xs-2" style="border-radius: 50%">  
 <div class="col-sm-10 col-xs-10">  
 <span style="font-weight: bold;" th:text="${message.name}"}>Jennifer</span>  
 <br>  
 <small class="date" style="color:#999" th:text="${message.time}">刚刚</small>  
 <div class="msg\_content" th:text="${message.content}">  
 今天天气不错!  
 </div>  
 <br><br>  
*<!-- <img class="mypic" th:src="${message.picUrl}">-->* <br><br>  
 <span class="commentsAndThumbUps" th:id="${message.id}">  
 <img th:id="'img'+${message.id}" class="zan" src="/images/zan.png" width="13" height="13" alt="zan.png">  
 <span th:id="'zanNum'+${message.id} " class="zanNum" th:text="${message.thumbUp}" >666</span>  
 <span >  
 <button type="button" class="btn btn-primary" data-toggle="collapse" th:data-target="'#cmt'+${message.id}" width="15" height="15">评论</button>  
 <img th:id="'img2'+${message.id}" class="xinfeng" src="/images/xf.png" width="13" height="13" alt="xinfeng.png">  
 <span th:id="'cmtNum'+${message.id} " class="cmtNum" th:text="${message.cmtNum}" >666</span>  
 <div th:id="'cmt'+${message.id}" class="collapse">  
 Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
 sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,  
 quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.  
 </div>  
 </span>  
 </span>  
  
  
 </div>  
 </div>  
 </div>

**排序部分：**

<div class="row" id="sort">  
 <a th:href="@{/user/home(sort\_method=time)}">按时间排序</a>  
 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  
 <a th:href="@{/user/home(sort\_method=thumbUp)}">按点赞数排序</a>  
 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;  
 <a th:href="@{/user/home(sort\_method=cmtNum)}">按评论数排序</a>  
</div>

**分页部分：**

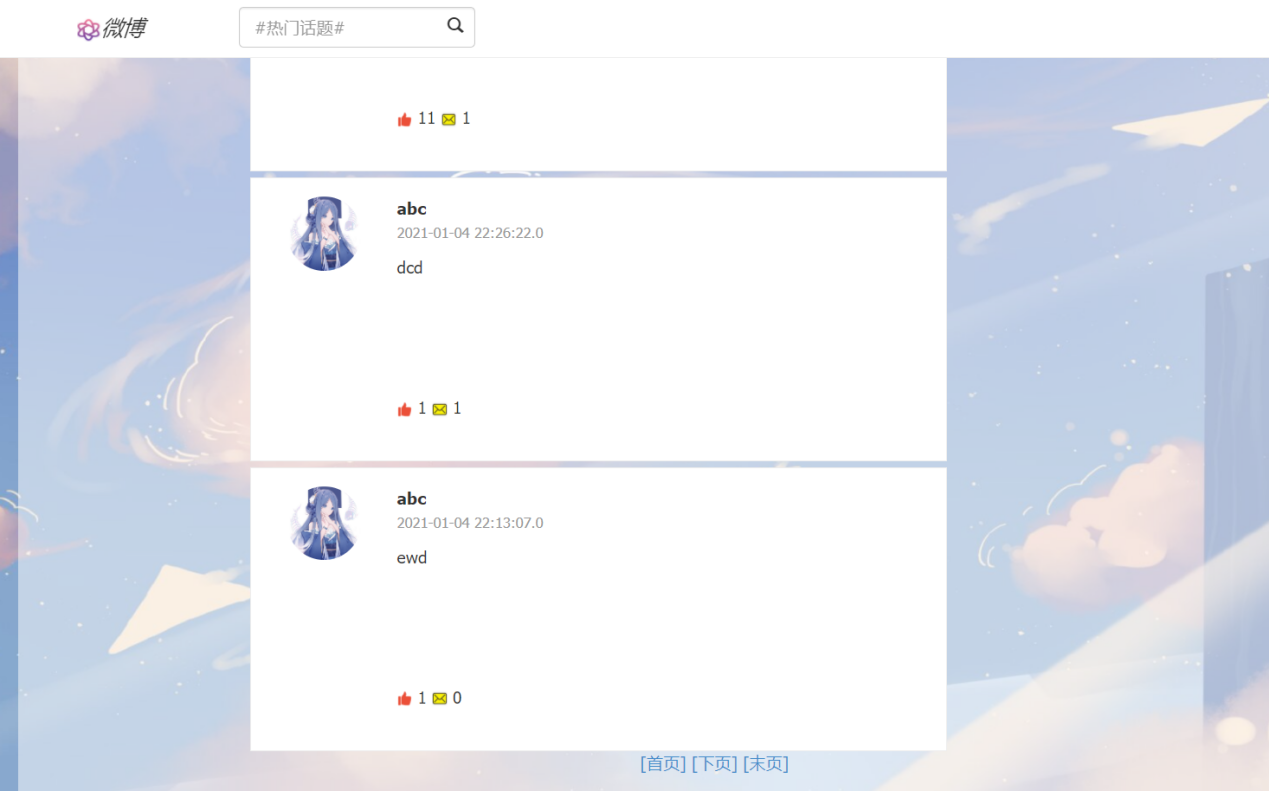
<div align="center">  
 <a th:href="@{/user/home(start=0)}">[首页]</a>  
 <a th:if="${not messages.isFirst()}" th:href="@{/user/home(start=${messages.number-1})}">[上页]</a>  
 <a th:if="${not messages.isLast()}" th:href="@{/user/home(start=${messages.number+1})}">[下页]</a>  
 <a th:href="@{/user/home(start=${messages.totalPages-1})}">[末页]</a>  
</div>

**3.3.3后端代码**

@GetMapping("/home")  
String home(Model model,  
 @RequestParam(value = "start",defaultValue = "0") Integer start,  
 @RequestParam(value = "limit",defaultValue = "9") Integer limit,  
 @RequestParam(value = "sort\_method",defaultValue = "time") String sort\_method)  
{  
 *//得到当前用户名* SysUser uid = (SysUser) SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  
 String uname = uid.getUsername();  
  
 start = start <0 ? 0 :start;  
 Sort sort = Sort.*by*(Sort.Direction.*DESC*,sort\_method);  
 Pageable pageable = PageRequest.*of*(start,limit,sort);  
 Page<Message> messages = messageRepository.findAll(pageable);  
 model.addAttribute("messages",messages);  
 model.addAttribute("uname", uname);  
 return "user/home";  
}

**3.3.4样式展示**





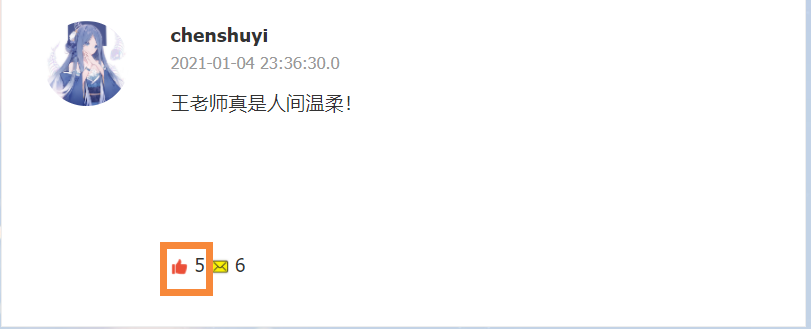


**3.4微博点赞**

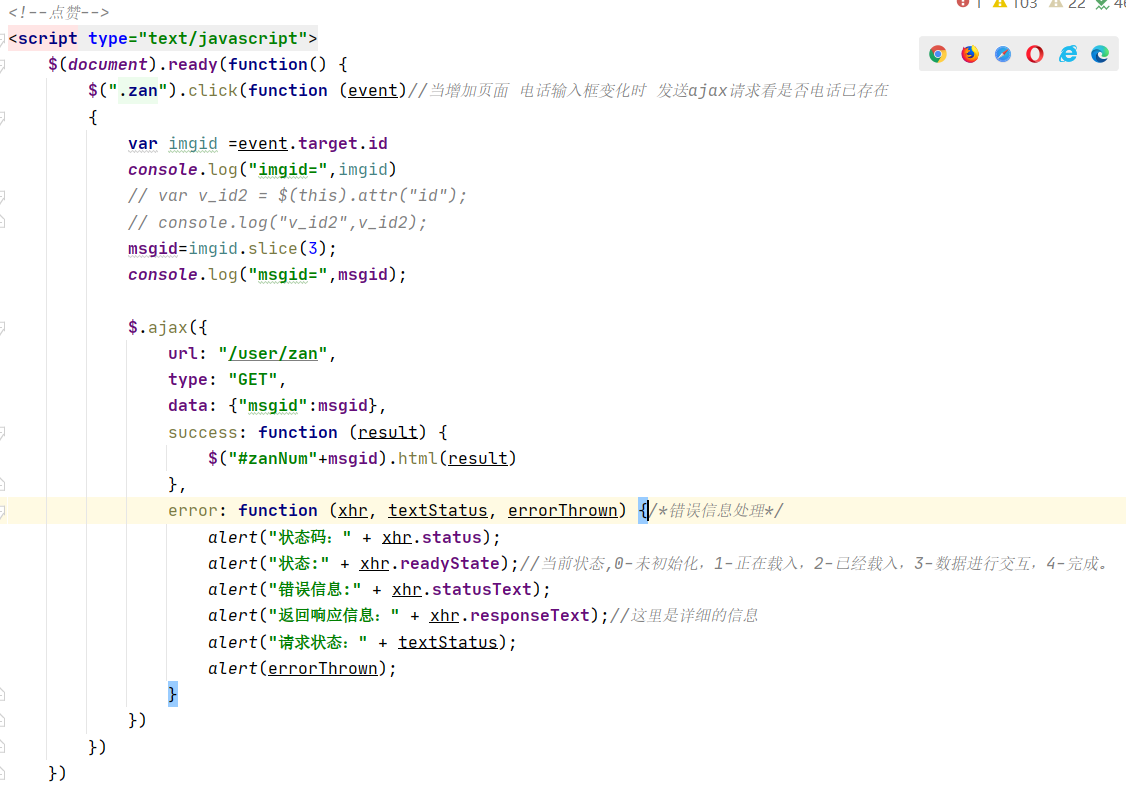
**3.4.1设计思路**

点击图片，实现被点击微博点赞数+1。

点击，前端侦听到onlick事件，采用AJAX发送请求给后端，后端相应微博的点赞数+1，更新数据库，并将点赞数返回给前端。



**3.4.2前端代码**



**3.4.3后端代码**



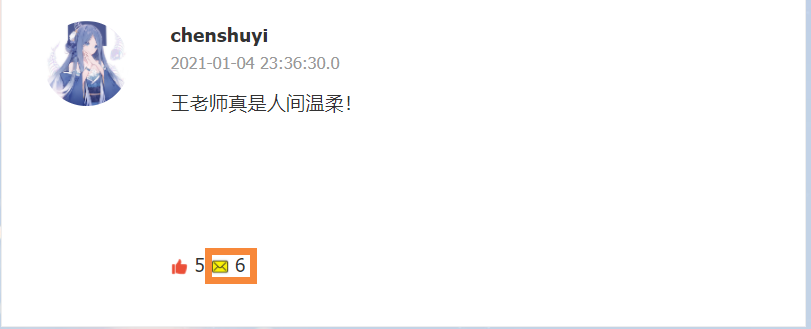
**3.5微博评论**

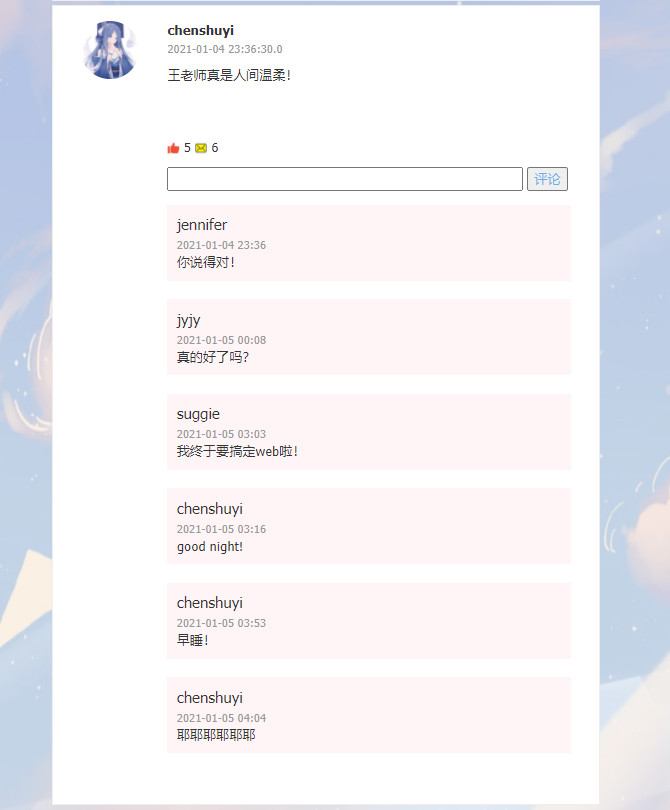
* 查看评论

**3.5.1设计思路**

点击图片，实现评论区的隐藏与展开。

点击图片，触发onclick事件，前端js向后端请求待显示的评论，后端放回数据，展示在下滑的评论框中





**3.5.2前端请求代码**



**3.5.3后端代码**



* 发表评论

**3.5.1设计思路**



在输入框里编写评论之后，点击提交表单，后端更新数据库，前端整体页面刷新。

**3.5.2前端代码**



**3.5.3后端代码**



**3.6关注和粉丝**

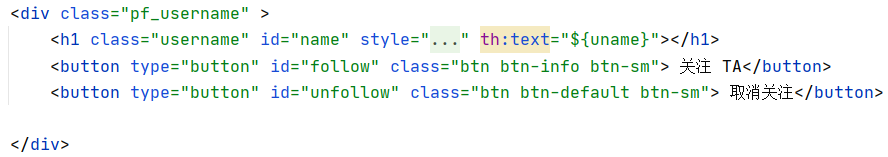
**3.6.1设计思路**

点击首页中某一条消息的用户名，可以进入该用户的个人主页，在个人主页里有“关注TA”和“取消关注”按钮。用ajax方法实现点击按钮后，往后台传递当前个人主页的用户名和“type”（1表示关注，0表示取消关注）这两个参数。后台根据用户名判断这对关注的关系是否存在，并据此进行。

数据库表示关注的表：如果用户x关注了用户y，则记录x和y的用户名。

**3.6.2前端代码**

前端personal.html中“关注TA”和“取消关注”按钮：



点击这两个按钮触发的事件用Ajax，向后台发送post请求来实现：

*//点击: +关注*$("#follow").click(function () {  
 var name = $("#name").text();  
 *// alert("name="+name);* $.ajax({  
 type: "POST",  
 url: "/user/personal",  
 data: {"person":name,"type":1},  
 *//contentType: "charset=utf-8",* dataType: "json",  
 success: function (attention) {  
 *alert*(attention);  
 },  
 error: function (attention) {  
 *alert*(attention);  
 }  
 });  
});  
*//取消关注*$("#unfollow").click(function () {  
 var name = $("#name").text();  
 $.ajax({  
 type: "POST",  
 data: {"person":name,"type":0},  
 url: "/user/personal",  
 *//contentType: "charset=utf-8",* dataType: "json",  
 success: function (attention) {  
 *alert*(attention);  
 },  
 error: function (attention) {  
 *alert*(attention);  
 }  
 });  
});

**3.6.3后端代码**

后端controller根据type参数来判定进行的是关注操作还是取消关注操作。

若为关注操作，判断这对“x关注y”的关系是否存在以及x与y是否相等（自己关注自己）；若不是，则执行关注操作，将x和y存入数据库里。

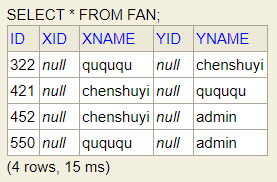
若为取消关注操作，判断这对关系“x关注y”是否存在，若存在则执行取关操作，将这条信息从数据库中删除；若不存在，则不执行取消关注操作。

添加@ResponseBody注解是为了解决post请求500的问题。

@ResponseBody  
@PostMapping("/personal")  
public String follow(@RequestParam("person") String person,  
 @RequestParam("type") Integer type)  
{  
 SysUser uid = (SysUser) SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  
 String uname = uid.getUsername();  
 *//判断是否已经关注了这个用户* String attention; *//回显信息* boolean flag=fanRepository.existsByXnameAndYname(uname, person);  
 if (type==1) *//关注* {  
 if (flag||uname.equals(person))  
 {  
 attention="已关注";  
 System.*out*.println("已关注");  
 }  
 else  
 {  
 Fan f = new Fan();  
 f.setXname(uname);  
 f.setYname(person);  
 fanRepository.save(f);  
 attention="关注成功";  
 System.*out*.println("关注成功");  
 }  
 }  
 else *//取消关注* {  
 if (flag)  
 {  
 Fan f= fanRepository.findByXnameAndYname(uname, person);  
 fanRepository.delete(f);  
 System.*out*.println("取消关注成功");  
 attention="取消关注成功";  
 }  
 else  
 {  
 System.*out*.println("未关注");  
 attention="未关注";  
 }  
 }  
 if (uname.equals(person)){  
 attention="不能关注自己";  
 }  
 return attention;  
}

**3.6.4样式展示**



（XID和YID为废弃列，因数据库没有重新create所以保留在里面，实际数据库中没有）

**3.7个人主页**

**3.7.1设计思路**

点击首页右上角的用户名可以进入个人主页，每条微博显示的发博人的名字也是超链接，可以进入对应用户的主页。

在个人主页中，可以进行用户的关注和取关，并且查看该用户发布的微博，进行排序、点赞、评论。



同时，在首页的右侧会显示当前用户的名称，粉丝数，关注数，微博数。其中关注和粉丝为超链接，可以点击查看关注人列表和粉丝列表。



**3.7.2前端代码**

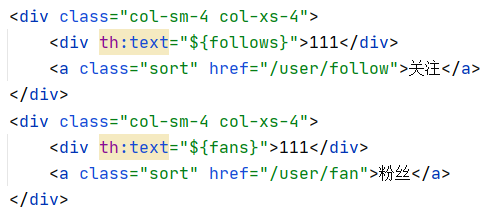
点击每条微博的发博用户名，可以进入他的主页。



进入自己的主页。



点击关注和分析，可以进入新页面查看关注列表和粉丝列表。



关注列表和粉丝列表的HTML代码（以关注为例）：

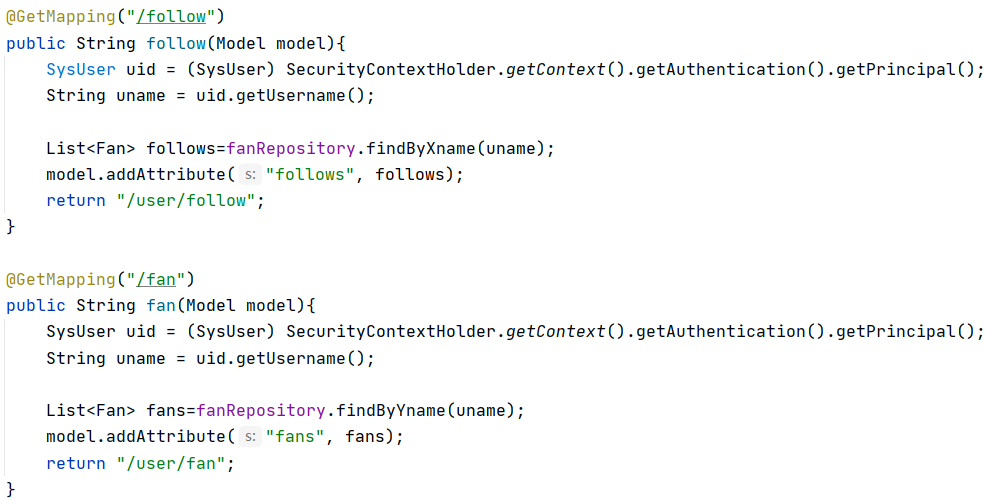
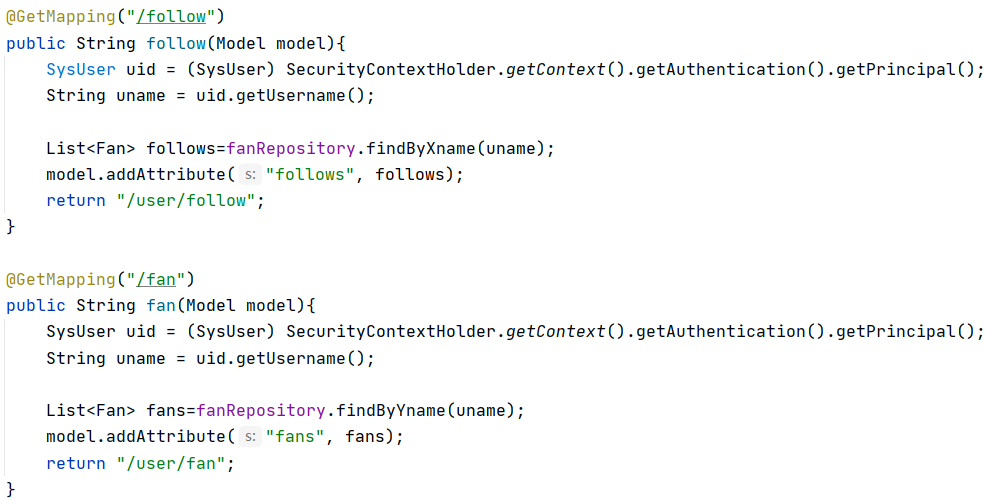


**3.7.3后端代码**

查询粉丝数和关注数和发博数：



跳转到关注列表和粉丝列表：

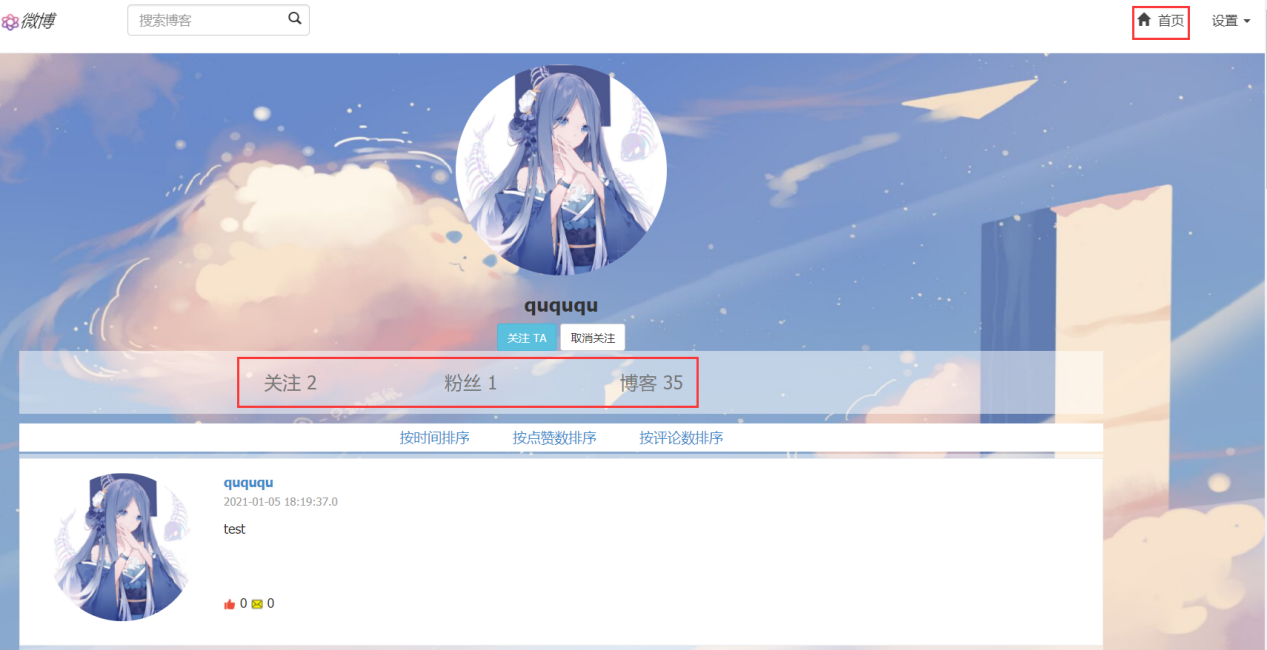


个人主页的微博显示，分页和排序：

@GetMapping("/personal") *//点消息里的名字*String personal(Model model,  
 @RequestParam(value = "start",defaultValue = "0") Integer start,  
 @RequestParam(value = "limit",defaultValue = "9") Integer limit,  
 @RequestParam("name") String name,  
 @RequestParam(value = "sort\_method",defaultValue = "time") String sort\_method)  
{  
 *//得到当前用户名* SysUser uid = (SysUser) SecurityContextHolder.*getContext*().getAuthentication().getPrincipal();  
 String uname = uid.getUsername();  
  
 start = start <0 ? 0 :start;  
 Sort sort = Sort.*by*(Sort.Direction.*DESC*,sort\_method);  
 Pageable pageable = PageRequest.*of*(start,limit,sort);  
  
 Page<Message> messages = messageRepository.findByName(name, pageable);  
 model.addAttribute("messages", messages);  
 model.addAttribute("uname", name);  
 model.addAttribute("sort",sort\_method);  
  
 Integer follows, fans, welogs;  
 follows = fanRepository.countByXname(name);  
 fans = fanRepository.countByYname(name);  
 welogs = messageRepository.countByName(name);  
 model.addAttribute("follows",follows);  
 model.addAttribute("fans",fans);  
 model.addAttribute("welogs",welogs);  
  
 return "user/personal";  
  
}

**3.7.4样式展示**





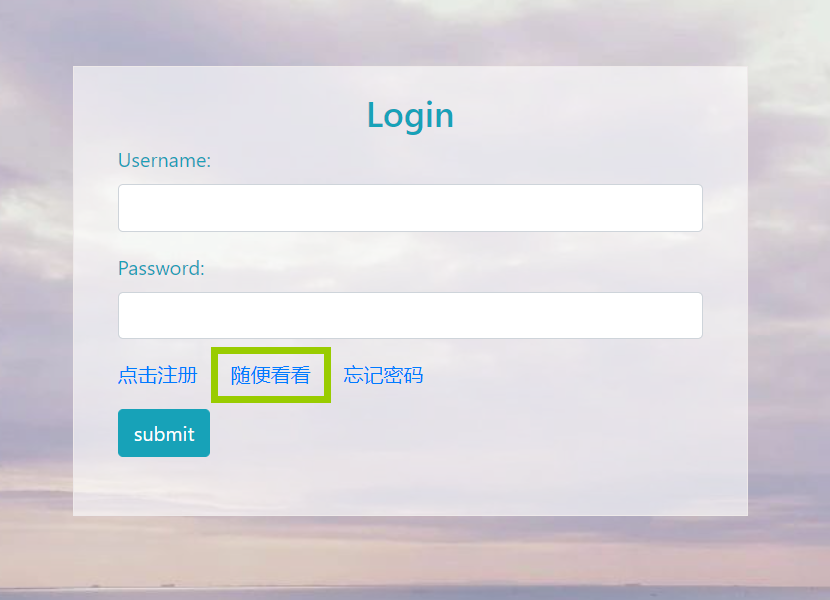
**3.8游客模式**

**3.8.1设计思路**

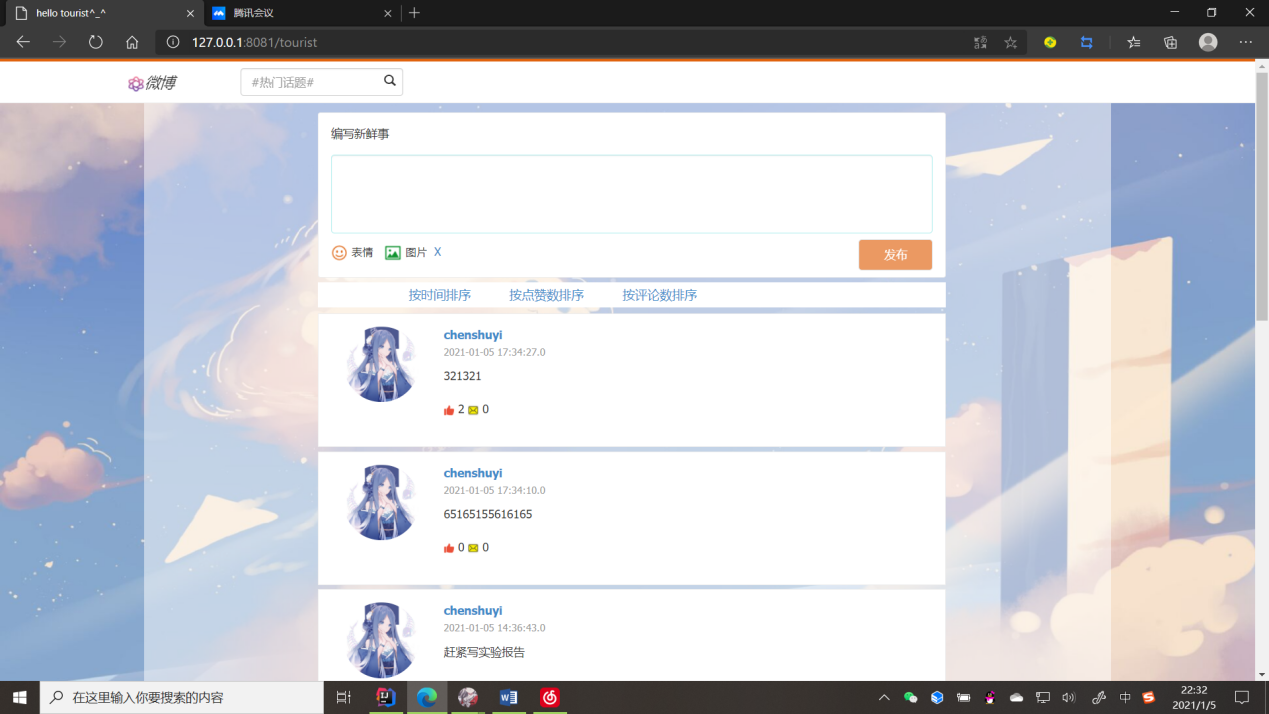
通过’/tourist’进入游客首页页面，这个页面游客只能查看。只需将用户首页稍作修改，将相应的按钮等禁用、去除评论区的输入框即可。

**3.8.2功能演示**

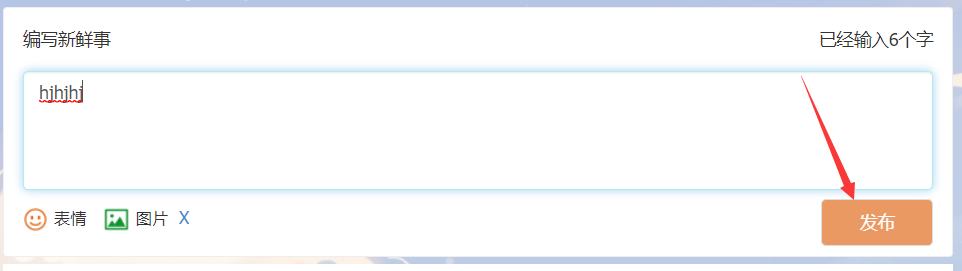
点击登录页面进入游客模式。

****

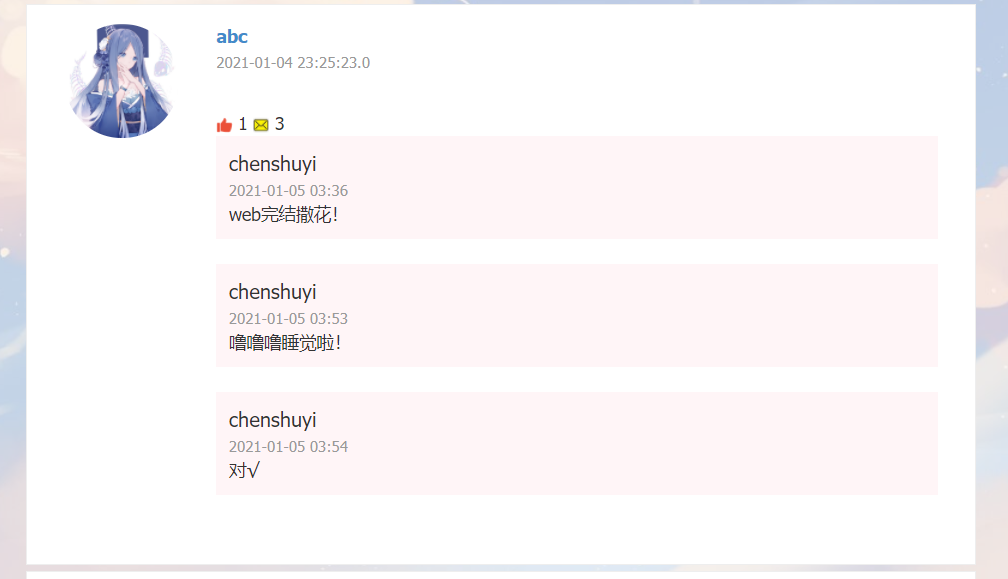
进入游客页面。



游客模式下不能发博，发布按钮失效。



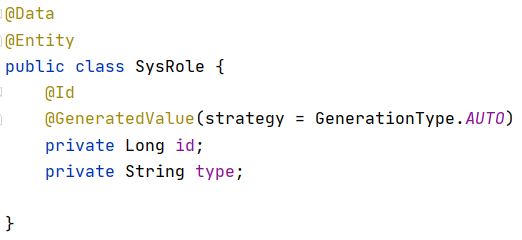
点击显示评论，此时没有编写评论的文本框，失效，点击无反应。



**4.数据库说明**

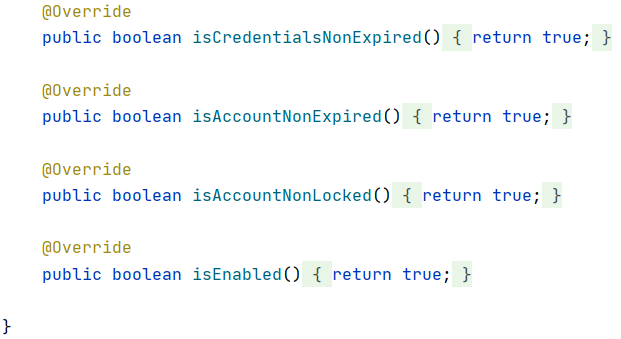
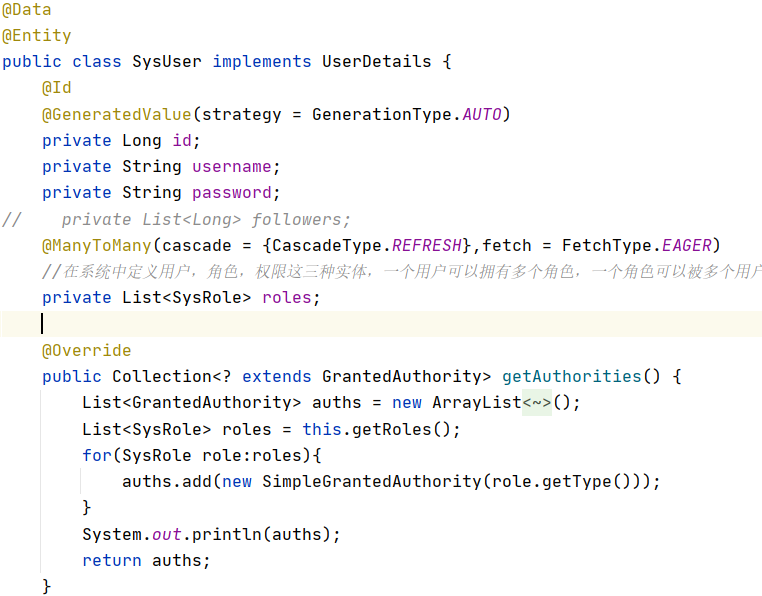
**4.1 Entity**

**4.1.1 Sysrole**



规定每一个用户的角色（游客或已注册用户），角色决定用户的权限。

**4.1.2 Sysuser**



用户的实体类，用来存储每个用户的各种信息，如用户名，密码等。

**4.1.3 Message**



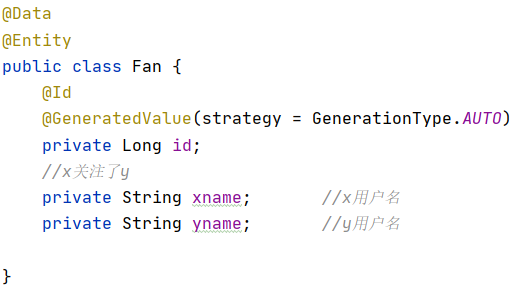
用来存放微博的内容，各个字段的含义在图中已经写出。

**4.1.4 Comment**



用来存放评论信息。

**4.1.5 Fan**



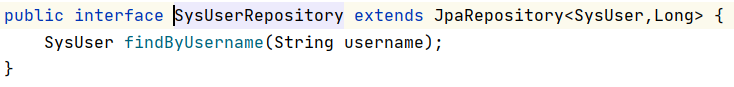
用来存放粉丝和关注的对应关系。

**4.2 Repository**

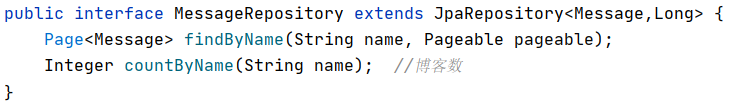
**4.2.1 Sysrole Repository**



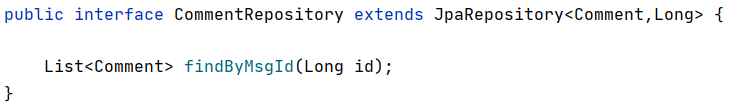
**4.2.2 Sysuser Repository**



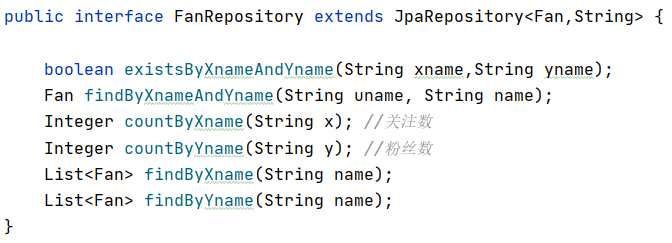
**4.2.3 Message Repository**



**4.2.4 Comment Repository**



**4.2.5 Fan Repository**



**5.评价和改进意见**

**5.1评价**

页面设计比较美观，运用了ajax技术，使得页面的刷新不是非常频繁，用户体验较好。

**5.2心得体会**

Web课程涉及到的知识点很多，没能很好地消化。在开卷考试前把本学期的所有内容都认真看过一遍后，对整体的知识有了更好的把握，对问题的解决变得更加有针对性，能想到应该使用何种方法，并据此进行资料的获取和代码的编写，来实现自己的功能。

在写大作业过程中也出现过许许多多的问题，有些通过搜索很快能解决，有些则不然。在写的过程中经常会推翻以前写的东西，对之前的数据库设置等进行改进，比如在写排序通过url传递参数时，发现参数的值包含下划线会出现问题，只能读取到下划线前一段，于是便将数据库中该字段的名字进行去掉下划线的修改，正确实现了功能。

在写微博图片的存取时，方法出现了一些问题。原方案是将图片通过input file添加，将二进制流存入数据库，然后将二进制流转成base64格式进行图片的显示。经过老师的提点，明白了这种方法其实很少用到，更多的是用upload上传到一个指定的地方储存起来，在数据库中存一个名字即可。

写评论模块的时候，遇到了各种各样的问题，在这个过程中对js、AJAX等技术有了更深入的了解。在写前后端数据交互的过程中，感受了MVC设计模式的奥义，感叹其精妙。

感谢老师一个学期的悉心教导和耐心解答！

**5.3改进意见**

可以实现用户忘记密码之后通过邮箱重新设置密码的功能。