

# 北京邮电大学

## 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
2.总体方案设计说明 .....	4
2.1 开发环境 .....	4
2.2 功能模块划分 .....	4
2.3 涉及到的技术 .....	4
3.各模块设计说明 .....	5
3.1 用户账号服务 .....	5
3.1.1 设计思路 .....	5
3.1.2 后端实现 .....	5
● 用拦截器实现权限管理.....	5
● 注册.....	5
● 登录.....	6
● 登出.....	7
3.2 微博发布 .....	8
3.2.1 设计思路 .....	8
3.2.2 前端代码 .....	8
3.2.3 后端代码 .....	11
3.2.4 样式展示 .....	12
3.3 微博查看 .....	12
3.3.1 设计思路 .....	12
3.3.2 前端代码 .....	13
3.3.3 后端代码 .....	14
3.3.4 样式展示 .....	15
3.4 微博点赞 .....	16
3.4.1 设计思路 .....	16
3.4.2 前端代码 .....	16
3.4.3 后端代码 .....	17
3.5 微博评论 .....	17
● 查看评论.....	17
3.5.1 设计思路 .....	17
3.5.2 前端请求代码 .....	19
3.5.3 后端代码 .....	19
● 发表评论.....	19
3.5.1 设计思路 .....	19
3.5.2 前端代码 .....	20
3.5.3 后端代码 .....	20
3.6 关注和粉丝 .....	20
3.6.1 设计思路 .....	20
3.6.2 前端代码 .....	21
3.6.3 后端代码 .....	22
3.6.4 样式展示 .....	23

3.7 个人主页 .....	24
3.7.1 设计思路 .....	24
3.7.2 前端代码 .....	24
3.7.3 后端代码 .....	25
3.7.4 样式展示 .....	26
3.8 游客模式 .....	28
3.8.1 设计思路 .....	28
3.8.2 功能演示 .....	28
4.数据库说明 .....	30
4.1 Entity.....	30
4.1.1 Sysrole.....	30
4.1.2 Sysuser .....	30
4.1.3 Message .....	30
4.1.4 Comment .....	31
4.1.5 Fan.....	31
4.2 Repository.....	32
4.2.1 Sysrole Repository .....	32
4.2.2 Sysuser Repository .....	32
4.2.3 Message Repository .....	32
4.2.4 Comment Repository.....	32
4.2.5 Fan Repository .....	32
5.评价和改进意见 .....	32
5.1 评价 .....	32
5.2 心得体会 .....	33
5.3 改进意见 .....	33

## 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

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    CustomUserService customUserService;
```

- 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。

```
//
        .and().authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
        .antMatchers( ...antPatterns: "/h2-console/**", "/user/register", "/tourist/**").permitAll()
        !!!!!!!!!!!!!!! 权限管理
        .antMatchers( ...antPatterns: "/user/**").hasRole("USER");//Spring Security会自动为任何角色添加前缀ROLE_，如
```

不拦截静态资源：

```
@Override
public void configure(WebSecurity web) throws Exception {
    // 不拦截静态资源
    web.ignoring().antMatchers( ...antPatterns: "/js/**", "/css/**", "/images/**");
}
```

- 注册

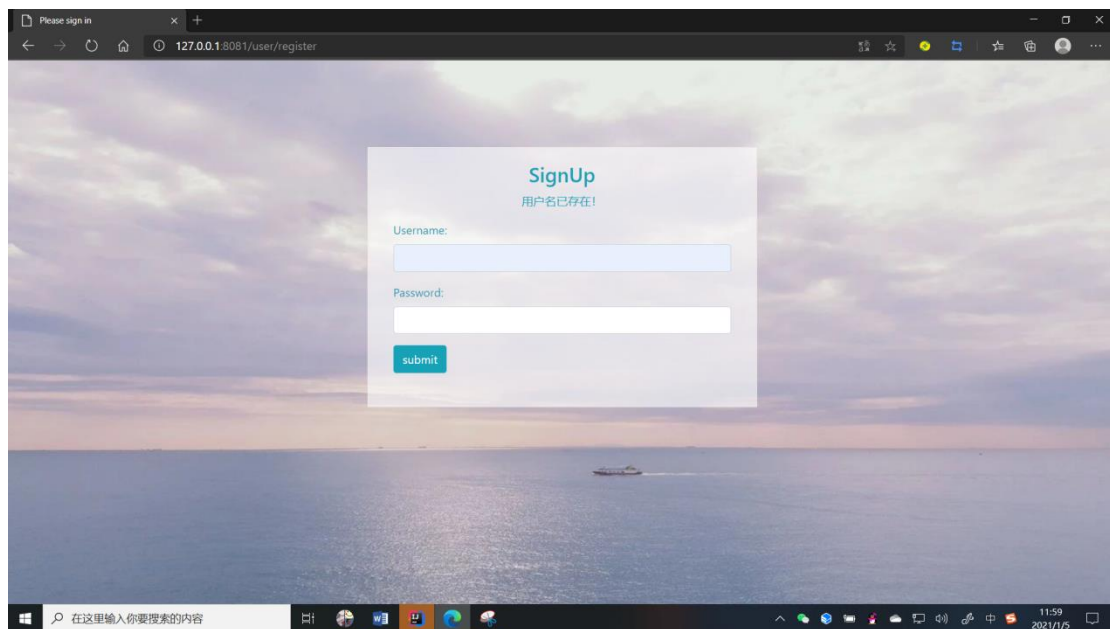
- 注册成功则为用户赋予 user 角色，并重定向到登录页面
- 注册失败则返回“用户名已存在”的信息给前端

```

@GetMapping("/register")
String register() { return "user/register"; }

@PostMapping("/register")
String registerDone(Model model, String username, String password){
    SysUser sysUser = sysUserRepository.findByUsername(username);
    if(sysUser != null){
        model.addAttribute("msg", "用户名已存在!");
        System.out.println("注册失败处理=====用户名已存在");
        return "user/register";
    }else {
        PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        sysUser = new SysUser();
        sysUser.setUsername(username);
        sysUser.setPassword(passwordEncoder.encode(password));
        SysRole sysRole = new SysRole();
        sysRole.setType("ROLE_USER");
        sysRoleRepository.save(sysRole);
        List<SysRole> roles = new ArrayList<>();
        roles.add(sysRole);
        sysUser.setRoles(roles);
        sysUserRepository.save(sysUser);
        System.out.println("注册成功处理=====");
        return "redirect:/user/login";
    }
}
}

```



## ● 登录

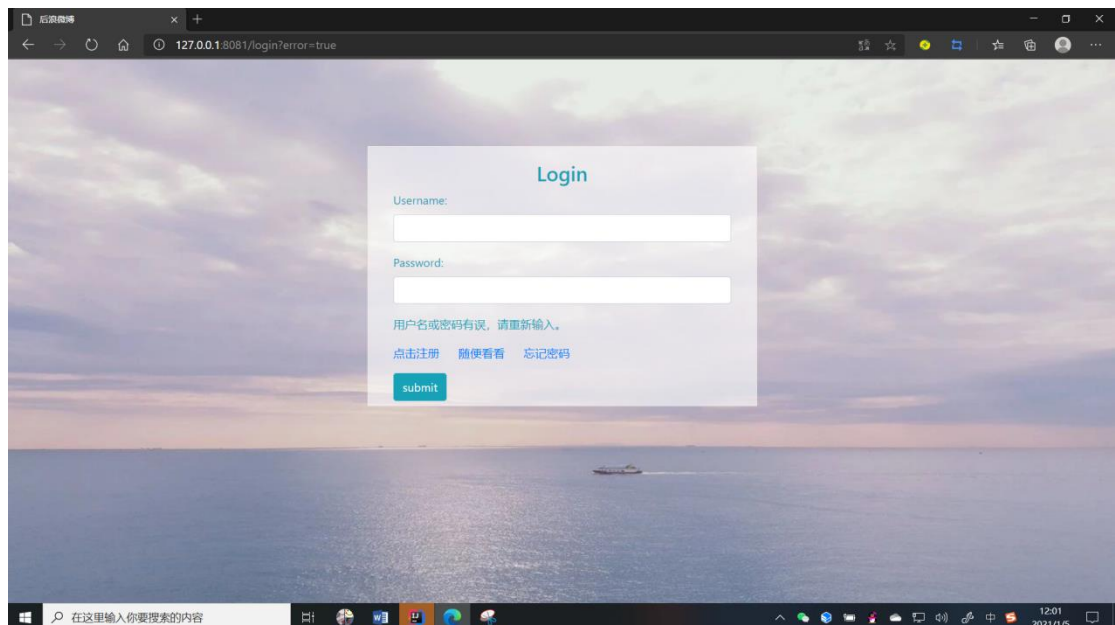
### ■ 登录成功处理

```
.successHandler((HttpServletRequest, HttpServletResponse, authentication) -> {
    System.out.println("登陆成功处理=====");
    // 跳转到微博整体首页，配置登录成功的回调
    HttpServletResponse.sendRedirect(s: "/user/home");
})
```

## ■ 登录失败，AJAX 提示错误信息

```
.failureForwardUrl("/loginfailure?error=true")
.failureHandler((HttpServletRequest, HttpServletResponse, e) -> {
    System.out.println("登陆失败处理=====重定向到登录页面");
    // 返回到登陆页面，可以采用AJAX技术实现，提示用户用户名不存在，密码不正确之类错误
    System.out.println(e.getMessage());
    String msg = null;
    if (e instanceof LockedException) {
        msg="账户被锁定，请联系管理员!";
    } else if (e instanceof CredentialsExpiredException) {
        msg="密码过期，请联系管理员!";
    } else if (e instanceof AccountExpiredException) {
        msg="账户过期，请联系管理员!";
    } else if (e instanceof DisabledException) {
        msg="账户被禁用，请联系管理员!";
    } else if (e instanceof BadCredentialsException) {
        msg="用户名或者密码输入错误，请重新输入!";
    }
    System.out.println(msg);

    HttpServletResponse.sendRedirect(s: "/login?error=true");
})
```



## ● 登出

重定向到登录页面

```

        .and().logout() LogoutConfigurer<HttpSecurity>
        .logoutUrl("/user/logout")
        .logoutSuccessHandler(((HttpServletRequest, HttpServletResponse, authentication) -> {
            System.out.println("登出成功处理=====重定向到登录页面");
            httpServletResponse.sendRedirect(s: "/user/login");
        })))
    }
}



```

## 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  图片 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  表情 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>
            <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>
        <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">
        

        <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>
<!--
            
            <br><br>
            <span class="commentsAndThumbUps" th:id="${message.id}">
                
                <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>

                
                <span th:id="'cmtNum'+${message.id}" class="cmtNum"
th:text="${message.cmtNum}">666</span>
                <div th:id="'cmt'+${message.id}" class="collapse">
```

elit, Lorem ipsum dolor sit amet, consectetur adipisicing  
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:

[illegible]

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.DISC, 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



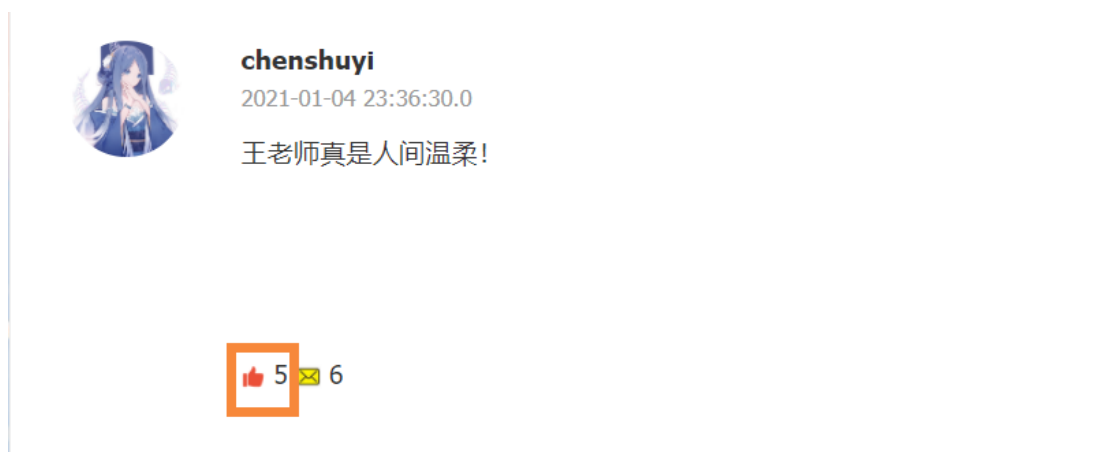


### 3.4 微博点赞

#### 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



```

<!--点赞-->
<script type="text/javascript">
$(document).ready(function() {
    $(".zan").click(function (event) // 当增加页面 电话输入框变化时 发送ajax请求看是否电话已存在
    {
        var imgid = event.target.id
        console.log("imgid=",imgid)
        // var v_id2 = $(this).attr("id");
        // console.log("v_id2",v_id2);
        msgid=imgid.slice(3);
        console.log("msgid=",msgid);

        $.ajax({
            url: "/user/zan",
            type: "GET",
            data: {"msgid":msgid},
            success: function (result) {
                $("#zanNum"+msgid).html(result)
            },
            error: function (xhr, textStatus, errorThrown) { /*错误信息处理*/
                alert("状态码: " + xhr.status);
                alert("状态: " + xhr.readyState); // 当前状态, 0-未初始化, 1-正在载入, 2-已经载入, 3-数据进行交互, 4-完成。
                alert("错误信息: " + xhr.statusText);
                alert("返回响应信息: " + xhr.responseText); // 这里是详细的信息
                alert("请求状态: " + textStatus);
                alert(errorThrown);
            }
        })
    })
})

```

### 3.4.3 后端代码

```


@ResponseBody
@GetMapping("/zan")
public String thumbUp(@RequestParam("msgid")String msgid) {
    Long id = Long.parseLong(msgid);
    Optional<Message> messageOptional = messageRepository.findById(id);
    if (messageOptional.isPresent()) {
        Message message = messageOptional.get();
        Integer num = message.getThumbUp();
        num=num+1;
        String ns = num.toString();
        message.setThumbUp(num);
        messageRepository.save(message);
        return ns;
    } else {
        return "failure";
    }
}


```

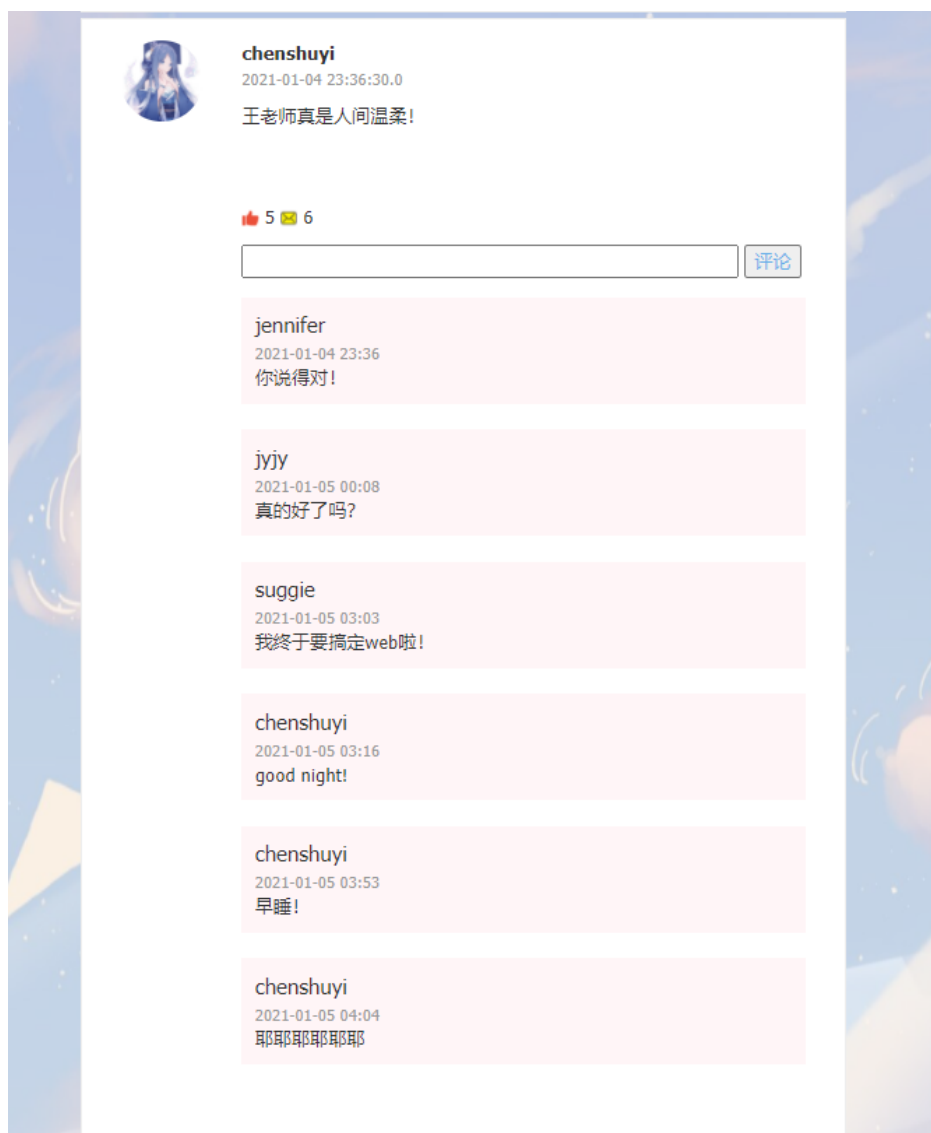
## 3.5 微博评论

### ● 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

```
<!--显示评论-->
<script language="javascript">
function sendId(event){
    var img2Id =event.target.id
    // var img2Id = $(this).attr("id");
    console.log("img2Id",img2Id)
    msgId=img2Id.slice(4);
    console.log(msgId);
    $.ajax({
        url: "/user/comment",
        type: "GET",
        data: {"msgId":msgId},
        success: function (result) {
            $("#text" + msgId).html(result);
            alert("刷新评论返回成功!")
            console.log("返回成功!");
        },
        error: function (xhr, textStatus, errorThrown) {
            /*错误信息处理*/
            alert("状态码: " + xhr.status);
            alert("状态: " + xhr.readyState);//当前状态,0-未初始化, 1-正在载入, 2-已经载入, 3-数据进行交互, 4-完成。
            alert("错误信息: " + xhr.statusText);
            alert("返回响应信息: " + xhr.responseText);//这里是详细的信息
            alert("请求状态: " + textStatus);
            alert(errorThrown);
        }
    })
}
</script>
```

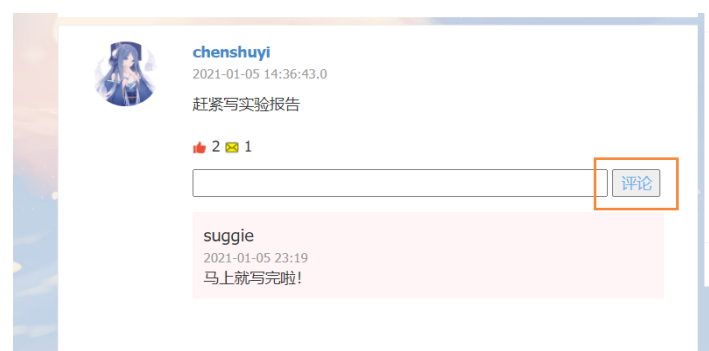
### 3.5.3 后端代码


```
@GetMapping("/comment")
public Object comment(@RequestParam("msgId")String msgId,Model model){
    System.out.println(msgId);
    Long id = Long.parseLong(msgId);

    List<Comment> comments=commentRepository.findByMsgId(id);
    model.addAttribute("comments", comments);
    return "/user/comments";
}
```

#### ● 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

```
<div th:id="'cmt'+${message.id}" class="collapse">
    <form role="form" style="..." method="post" th:action="@{/user/comment}">
        <div class="form-group">
            <input type="text" name="content" style="width: 88%"/>
            <input type="hidden" name="msgId" th:value="${message.id}" />
            <input type="submit" name="comment" value="评论" style="color: #66afe9" />
        </div>
    </form>
    <div th:id="'text'+${message.id}">
```

### 3.5.3 后端代码

```
@PostMapping("/comment")
public String comments(@RequestParam("msgId")String msgId,@RequestParam("content")String content,Model model){

    Long msgId = Long.parseLong(msgId);
    SysUser uid = (SysUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    Long userId = uid.getId();
    String name=uid.getUsername();
    Date time=new Date();
    //    System.out.println(msgId);
    //    System.out.println(content);
    //    System.out.println(userId);
    //    System.out.println(time);
    Comment c = new Comment();
    Optional<Message> messageOptional=messageRepository.findById(msgId);
    if(messageOptional.isPresent()) {
        c.setMessage(messageOptional.get());
        Message messagetemp= messageOptional.get();
        messagetemp.setCmtNum(messagetemp.getCmtNum()+1);
        messageRepository.save(messagetemp);
    }
    c.setMsgId(msgId);
    c.setContent(content);
    c.setTime(time);
    c.setPublisherName(name);
    c.setPublisherId(userId);
    commentRepository.save(c);//将评论存入数据库
    List<Comment> comments=commentRepository.findByMsgId(msgId);

    model.addAttribute("comments", comments);
    return "redirect:/user/home";
}
```

## 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：

```
<div class="pf_username" >
  <h1 class="username" id="name" style="..." th:text="${uname}"></h1>
  <button type="button" id="follow" class="btn btn-info btn-sm"> 关注 TA</button>
  <button type="button" id="unfollow" class="btn btn-default btn-sm"> 取消关注</button>
</div>
```

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，若存在则执行取消关注操作，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);

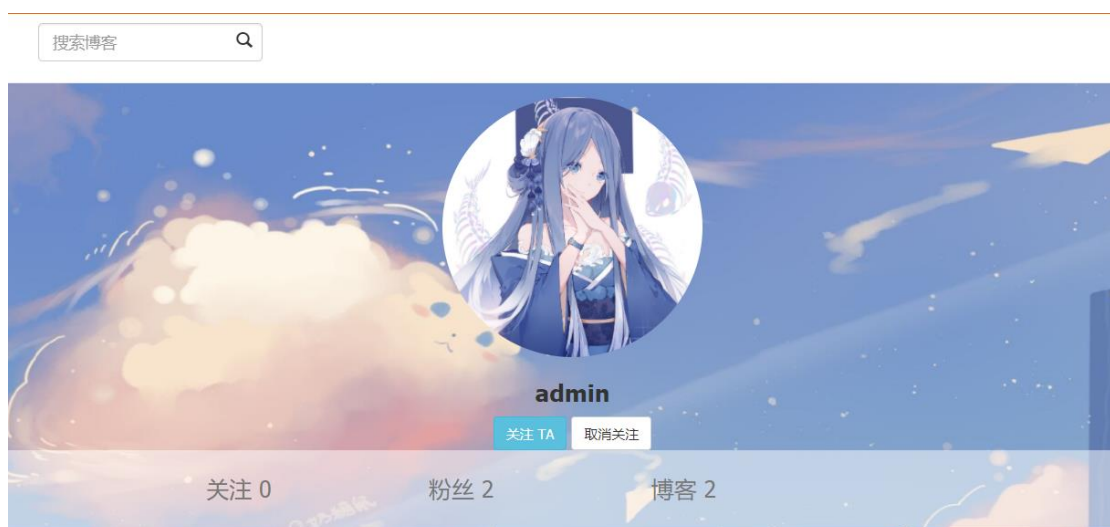
```

```

        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



SELECT \* FROM FAN;

ID	XID	XNAME	YID	YNAME
322	null	quququ	null	chenshuyi
421	null	chenshuyi	null	quququ
452	null	chenshuyi	null	admin
550	null	quququ	null	admin

(4 rows, 15 ms)

(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。

```
<a style="..." th:text="${message.name}" th:href="@{/user/personal(name=${message.name})}">Jennifer</a>
```

进入自己的主页。

```
<li><a th:href="@{/user/personal(name=${uname})}" ><i class="glyphicon glyphicon-user" th:text="${uname}" id="uname"> Jennifer</i>
```

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。



```

<div class="col-sm-4 col-xs-4">
    <div th:text="${follows}">111</div>
    <a class="sort" href="/user/follow">关注</a>
</div>
<div class="col-sm-4 col-xs-4">
    <div th:text="${fans}">111</div>
    <a class="sort" href="/user/fan">粉丝</a>
</div>

```

关注列表和粉丝列表的 HTML 代码（以关注为例）：

```

<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>关注列表</title>
</head>
<body>
    <h1 style="color:hotpink">关注列表</h1>
    <a href="/user/home">返回首页</a>
    <br><br><br>
    <div th:each="follow:${follows}" class="collapse">
        
        <span th:text="${follow.Yname}" style="font-size: 30px">
    </div>
</body>
</html>

```

### 3.7.3 后端代码

查询粉丝数和关注数和发博数：

```

Integer follows, fans, welogs;
follows = fanRepository.countByXname(name);
fans = fanRepository.countByYname(name);
welogs = messageRepository.countByName(name);
model.addAttribute(s: "follows", follows);
model.addAttribute(s: "fans", fans);
model.addAttribute(s: "welogs", welogs);

```

跳转到关注列表和粉丝列表：

```

@GetMapping("/follow")
public String follow(Model model){
    SysUser uid = (SysUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    String uname = uid.getUsername();

    List<Fan> follows=fanRepository.findByXname(uname);
    model.addAttribute(s: "follows", follows);
    return "/user/follow";
}

@GetMapping("/fan")
public String fan(Model model){
    SysUser uid = (SysUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    String uname = uid.getUsername();

    List<Fan> fans=fanRepository.findByYname(uname);
    model.addAttribute(s: "fans", fans);
    return "/user/fan";
}

```

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();

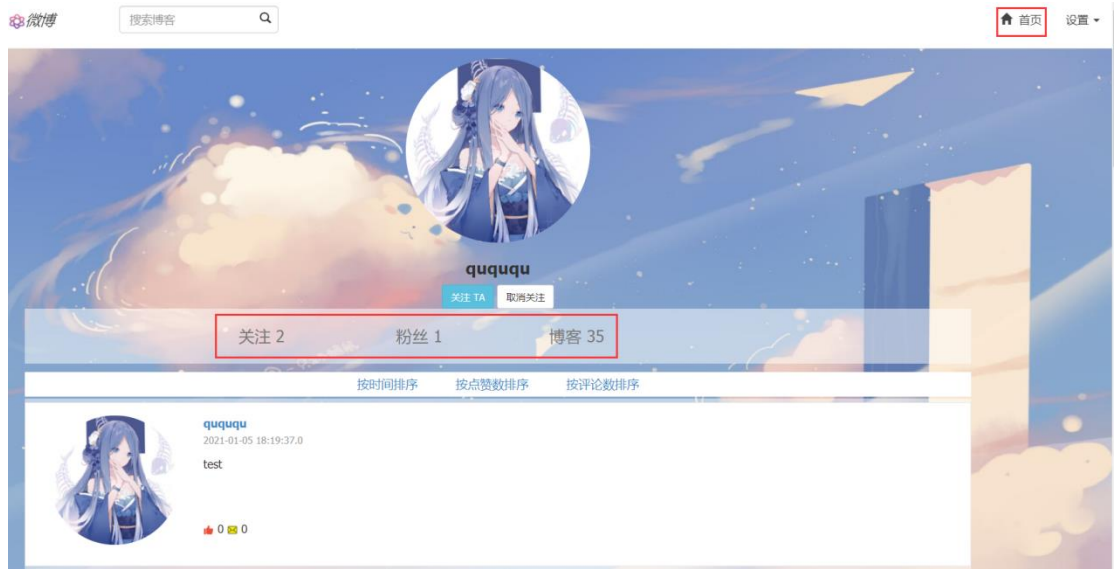
    start = start < 0 ? 0 : start;
    Sort sort = Sort.by(Sort.Direction.DISC,sort_method);
    Pageable pageable = PageRequest.of(start,limit,sort);

    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



## 关注列表

[返回首页](#)



chenshuyi



admin

## 粉丝列表

[返回首页](#)



chenshuyi

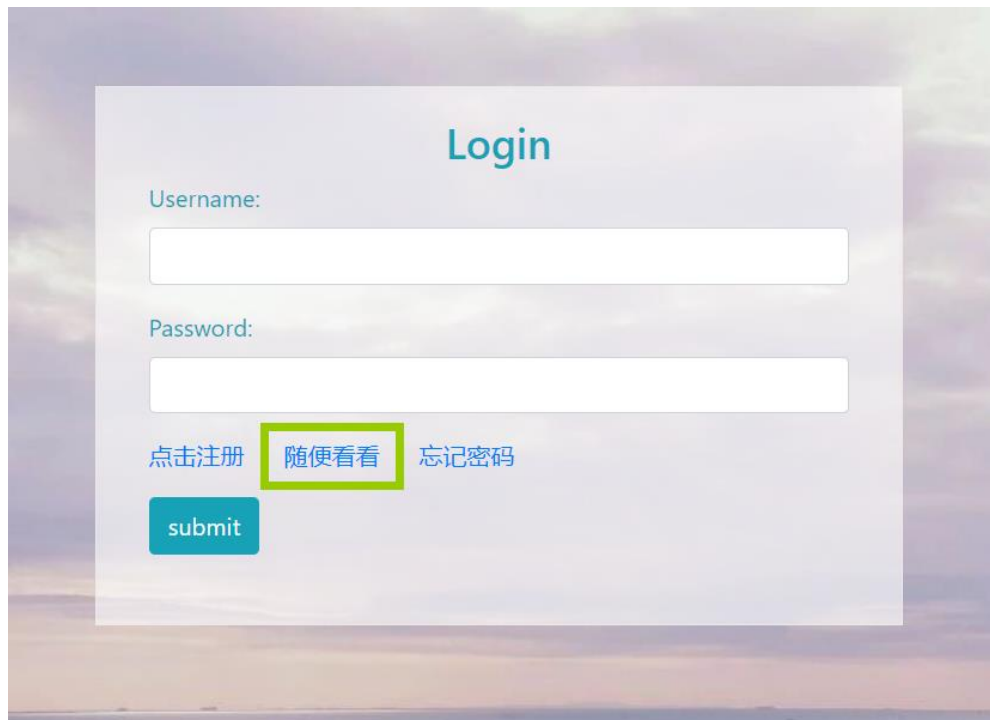
### 3.8 游客模式

#### 3.8.1 设计思路

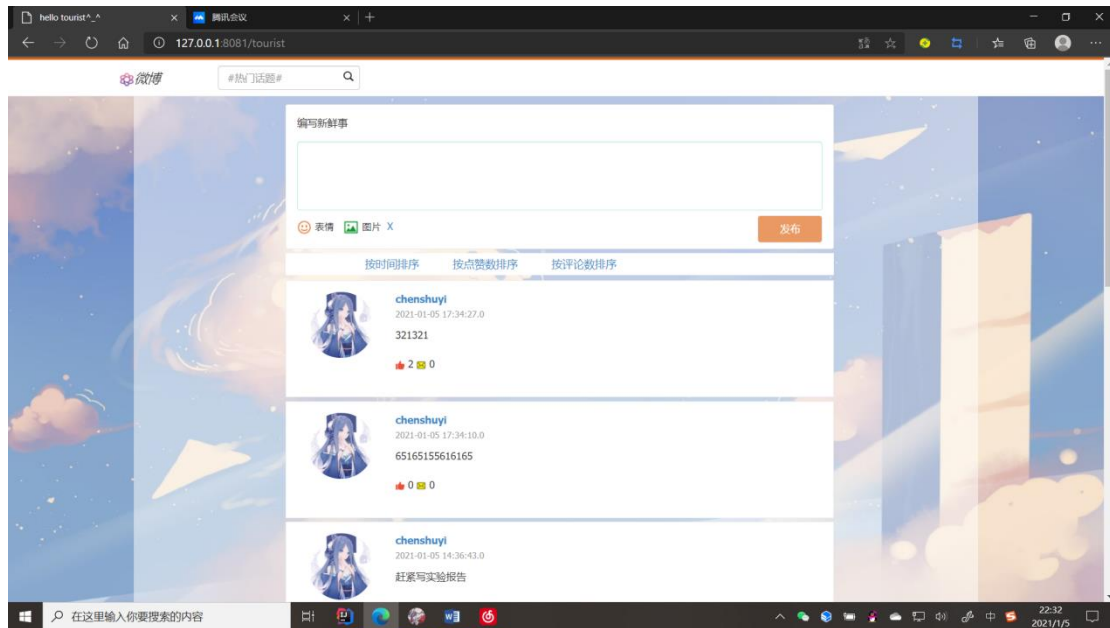
通过’/tourist’进入游客首页页面，这个页面游客只能查看。只需将用户首页稍作修改，将相应的按钮等禁用、去除评论区的输入框即可。

#### 3.8.2 功能演示

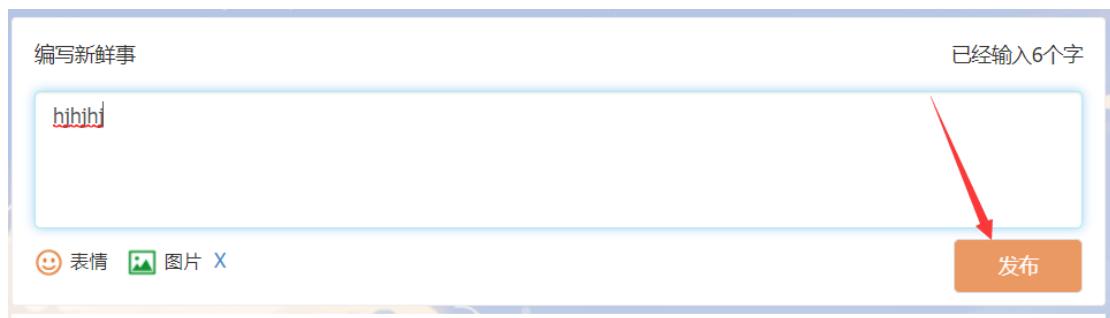
点击登录页面[随便看看](#)进入游客模式。





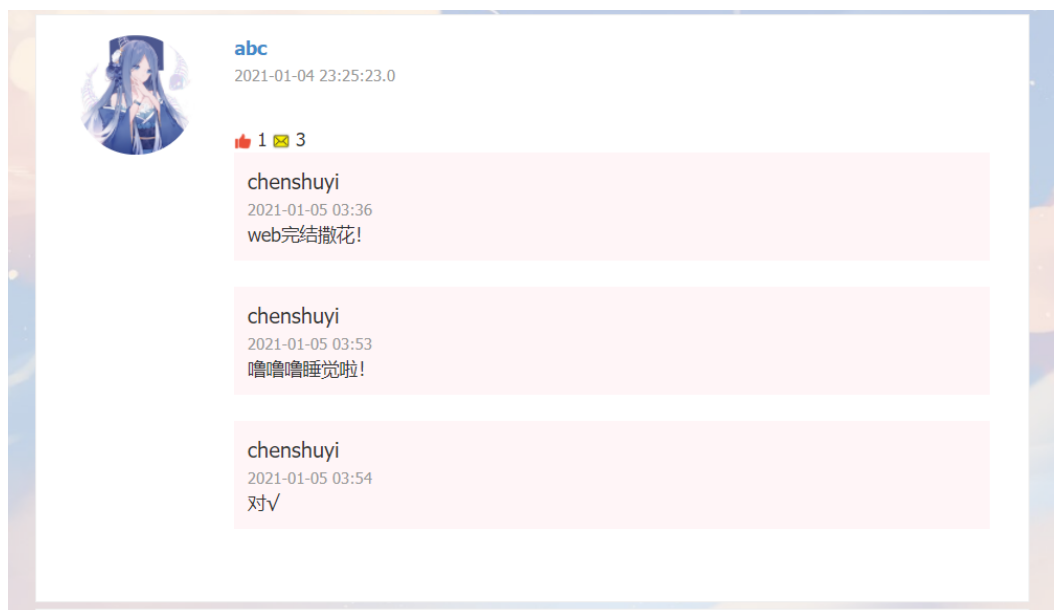
进入游客页面。



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

```
@Data
@Entity
public class SysRole {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String type;
}
```

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

```
@Data
@Entity
public class SysUser implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    // private List<Long> followers;
    @ManyToMany(cascade = {CascadeType.REFRESH}, fetch = FetchType.EAGER)
    // 在系统中定义用户，角色，权限这三种实体，一个用户可以拥有多个角色，一个角色可以被多个用户
    private List<SysRole> roles;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    List<GrantedAuthority> auths = new ArrayList<>();
    List<SysRole> roles = this.getRoles();
    for(SysRole role:roles){
        auths.add(new SimpleGrantedAuthority(role.getType()));
    }
    System.out.println(auths);
    return auths;
}

@Override
public boolean isCredentialsNonExpired() { return true; }

@Override
public boolean isAccountNonExpired() { return true; }

@Override
public boolean isAccountNonLocked() { return true; }

@Override
public boolean isEnabled() { return true; }
}
```

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

```

@Data
@Entity
public class Message {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;           //消息id
    private Long userid;       //用户id
    private String name;       //用户名
    private Date time;         //发布时间
    private String content;    //正文
    private Integer thumbUp;   //点赞数
    private Integer cmtNum;    //评论数

    private String picUrl;     //图片的url

    // @OneToMany(mappedBy = "message", cascade = CascadeType
    // private List<Comment> comments = new ArrayList<>();
}

```

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

```

@Data
@Entity
public class Comment {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private Long msg_id;       //针对于哪条微博的评论
    private Long publisher_id; //发布者id
    private String content;    //文字内容
    private Date time;         //发布时间

    // @JsonIgnore
    // @ManyToOne
    // @JoinColumn
    // private Message message;
}

```

用来存放评论信息。

#### 4.1.5 Fan

```

@Data
@Entity
public class Fan {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    //x关注了y
    private String xname;      //x用户名
    private String yname;      //y用户名
}

```

用来存放粉丝和关注的对应关系。

## 4.2 Repository

### 4.2.1 Sysrole Repository

```
public interface SysRoleRepository extends JpaRepository<SysRole,Long> {  
}
```

### 4.2.2 Sysuser Repository

```
public interface SysUserRepository extends JpaRepository<SysUser,Long> {  
    SysUser findByUsername(String username);  
}
```

### 4.2.3 Message Repository

```
public interface MessageRepository extends JpaRepository<Message,Long> {  
    Page<Message> findByName(String name, Pageable pageable);  
    Integer countByName(String name); //博客数  
}
```

### 4.2.4 Comment Repository

```
public interface CommentRepository extends JpaRepository<Comment,Long> {  
  
    List<Comment> findByMsgId(Long id);  
}
```

### 4.2.5 Fan Repository

```
public interface FanRepository extends JpaRepository<Fan,String> {  
  
    boolean existsByXnameAndYname(String xname,String yname);  
    Fan findByXnameAndYname(String uname, String name);  
    Integer countByXname(String x); //关注数  
    Integer countByYname(String y); //粉丝数  
    List<Fan> findByXname(String name);  
    List<Fan> findByYname(String name);  
}
```

## 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称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。