Web 大作业报告

1 需求分析

1.1 基础功能

1. 用户注册、登陆、退出：spring security
2. 微博列表排序（发布时间、评论数、点赞数）
3. 分页：下部按钮实现
4. 未注册用户：直接查看内容和评论
5. 微博发布：文字+图片
6. 点赞、评论
7. 关注

1.2 拓展功能

1. 错误锁定：spring security
2. 邮箱重置密码：通过用户名关联邮箱，发送修改密码的请求
3. @用户：正则表达式，高亮被@用户信息，点击可以跳转用户主页；被@用户有提示信息
4. AJAX减少刷新（点赞+评论）

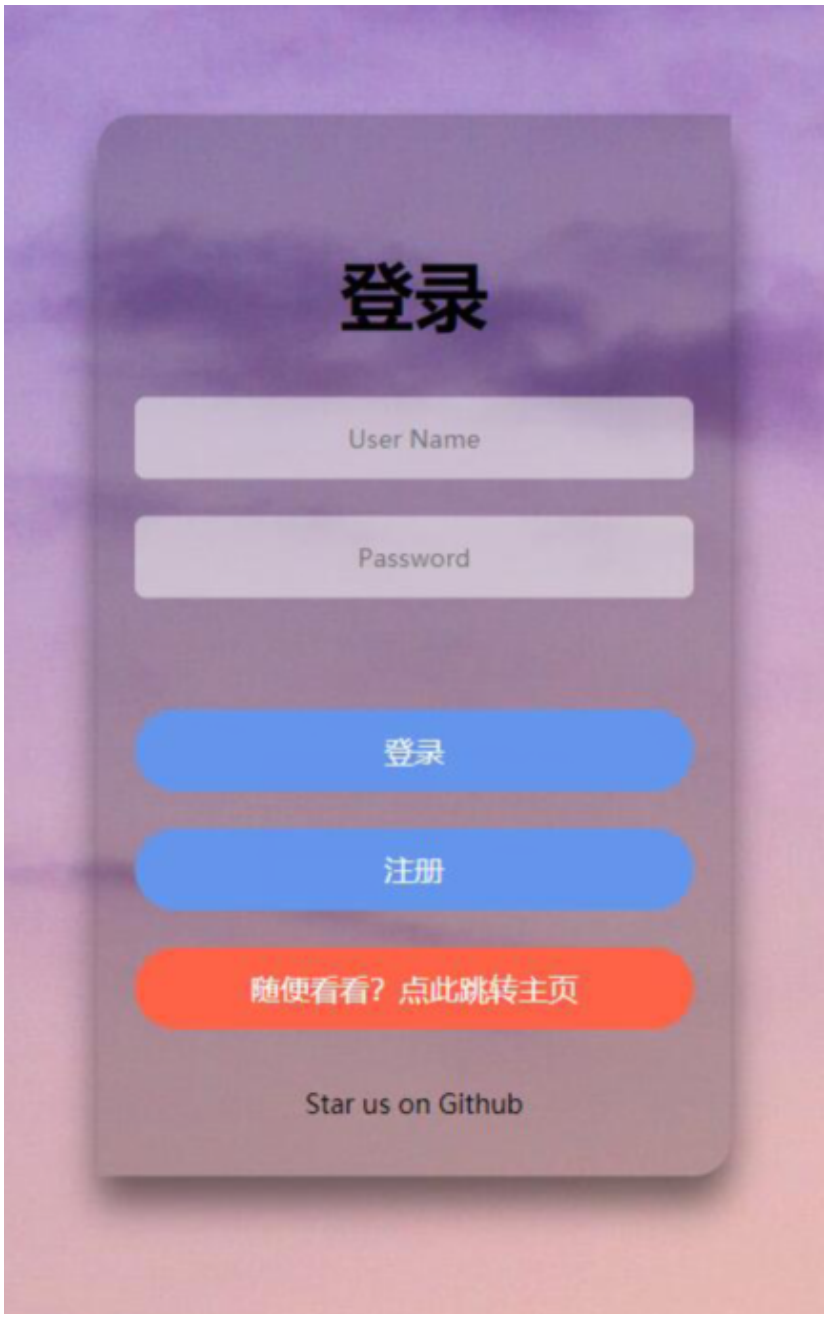
1.3 额外功能

1. 管理员账号才能访问h2数据库
2. 通过邮箱进行注册

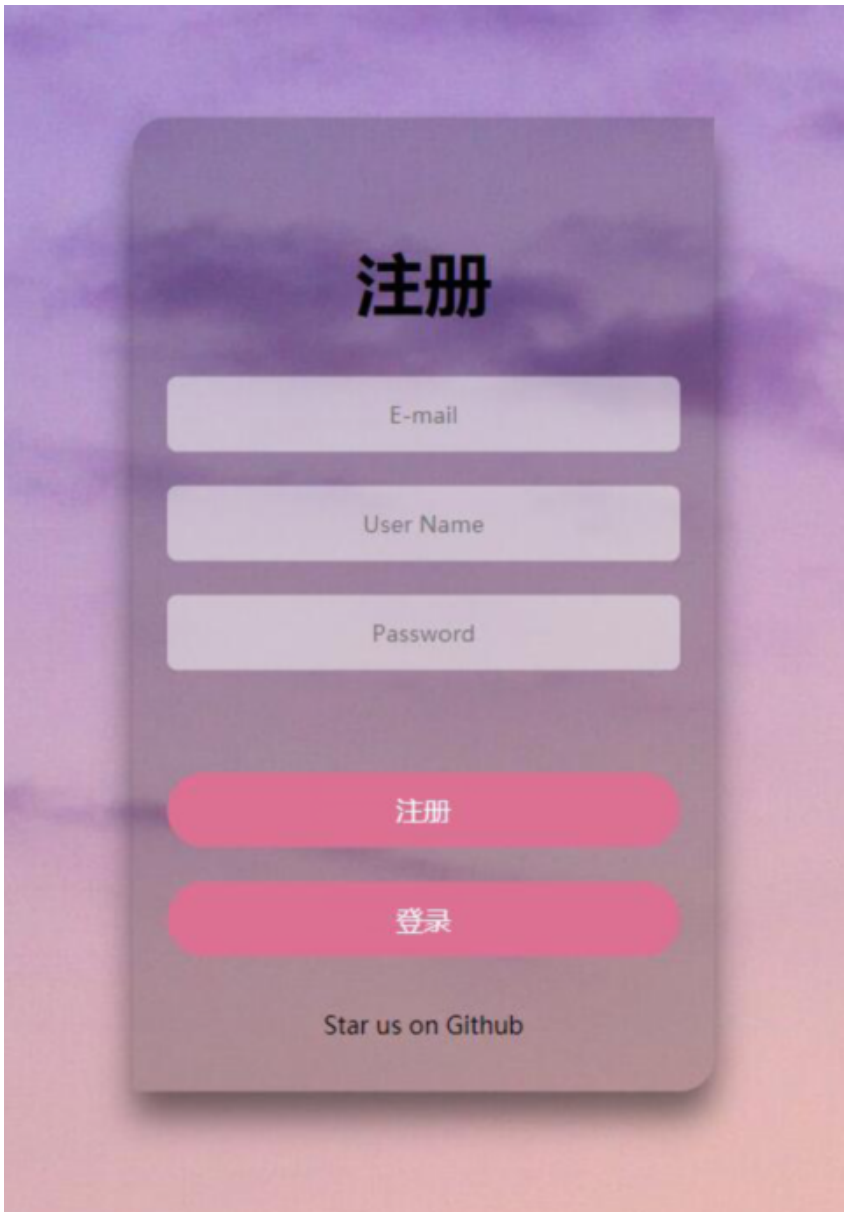
2 设计

2.1 html页面

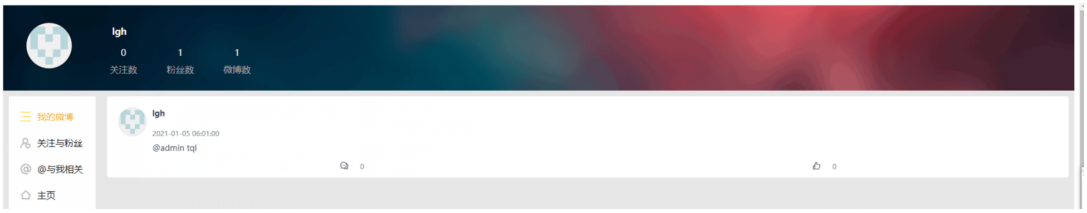
1. 登陆页面



2. 注册页面

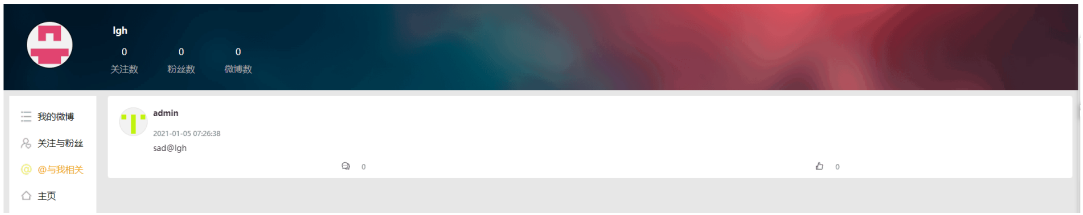


3. 我的页面（我的微博）



4. 我的页面（关注和粉丝）

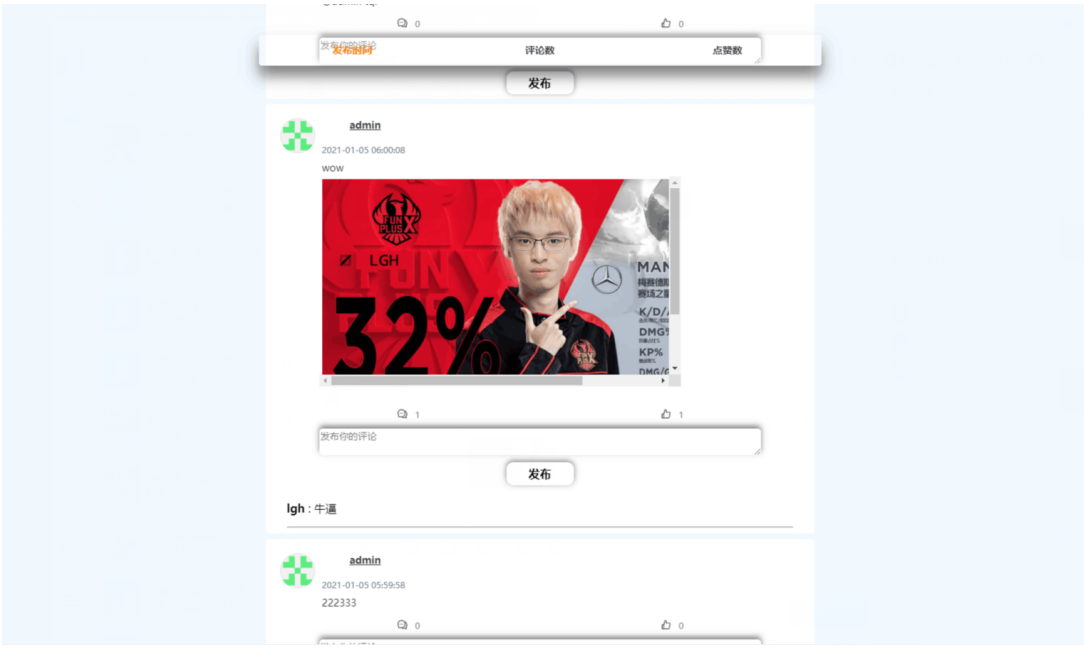
5. 我的页面（@信息显示）



6.

7. 微博内容页面

（排序➕发布➕主体内容）



2.2 数据

2.2.1 数据表

用户表 WeiboUser：

id	用户名 String username	密码 String password	邮箱 String email	粉丝 List<WeiboUser> fansUser	关注 List<WeiboUser> attentionUser	微博 List<Weibo> weibos	@我 List<Weibo> atMeWeibo	与我下班你管
	邮件锁定截至时间 String emailOutDate	有效位 String validCode	当前账号未被锁定 boolean AccountNonLocked	当前账号激活情况 boolean active	头像路径 String avatarPath	关注我的微博 List<Weibo> attentionWeibo	权限角色 List<WeiboRole> roles	

微博表 Weibo：

id	内容 String weibo Text	评论数 int commen tNum	点赞数 int likeNu m	发布者 Weibo User weibo User	评论 List<Com ment> comments	发布时 间 String time	发布图片 List<Pict ure> pictures	@用户 List<Weib oUser> atUsers

评论表 Comment：

id	内容 String commentText	用户 WeiboUser weiboUser	微博 Weibo weibo

权限表：WeiboRole

id	权限名称 roleName

2.2.2 表项映射关系

用户 -> 微博（one -> many）

用户 -> 粉丝用户（one -> many）

用户 -> 权限（many -> many）

用户 -> @我的微博（many -> many）

用户 -> 关注用户（many -> many）

用户 -> @我的用户（many -> many）

微博 -> 发布用户（one -> one）

微博 -> @用户（many -> many）

微博 -> 评论（one -> many）

微博 -> 图片（one -> many）

评论 - > 用户（many -> one）

评论 - > 微博（many - > one）

图片 -> 用户（one - > one）

2.3 控制器 Controller

- AdminController

URL	访问时机及作用
/admin	权限页面

- ErrorController

URL	访问时机及作用
/404	错误处理页面（网页或文件未找到）
/403	错误处理页面（资源不可用）
/401	错误处理页面（用户无权限）
/400	错误处理页面（访问页面域名不存在或者请求错误）

- IndexController

URL	访问时机及作用
/	返回主页

- LoginController

URL	访问时机及作用
/login	登录页面

- RegisterController

URL	访问时机及作用
/register	登录页面
/activeUserEmail	激活用户邮箱页面
/activeUserEmail/reSend	重新发送激活用户邮箱页面
/resetMailSend	重置邮箱页面
/resetUserPassword	重置用户密码页面

- UserController

URL	访问时机及作用
/myWeibo	用户主页
/selfWeibo	访问登录用户自己的主页
/fans	粉丝页面
/at	@我的页面
/checkAttention	添加关注页面

- WeiboController

URL	访问时机及作用
/likeAdd	点赞（Ajax）
/commentAdd	添加评论（Ajax）
/commentShow	展示评论（Ajax）
/patternAt	添加关注（Ajax）
/release	发布微博（Ajax）
/mainPage	微博主页面

3 问题反思及实现

3.1 登录密码输入错误3次后账号锁定1小时

由于采用Spring security作为项目安全框架，因此用户entity已经实现了基于Spring security的UserDetails接口，通过该接口中entity的AccountNonLocked属性实现账户的锁定。因为该属性将会在Spring security过滤器链的前端（至少在密码验证过滤器的前端）进行判断，若判断不通过Spring security将会抛出LockedException，并发生登录错误事件。

接下来，需要在Spring security配置类中设置登录错误处理类即项目中的**LoginFailureHandler**。需要注意的是无论是账户锁定、账户未激活、密码错误均会进入该处理过程。为了检测用户短时间类多次触发登录错误的情况，利用API限流库ratelimitj的内置方案如下：

```
Set<RequestLimitRule> rules =Collections.singleton(RequestLimitRule.of(5, TimeUnit.MINUTES, 3));
    RequestRateLimiter limiter = new InMemorySlidingWindowRequestRateLimiter(rules);
```

这样设定表示每5分钟给予三次错误机会，否则触发限流处理。一旦处理函数检测到限流信号，则将AccountNonLocked属性设为false并返回401错误，这样用户账户将被锁定。

为了实现1小时后自动解锁用户账户，利用Spring task和java 线程池技术，首先在Spring的静态启动类设置静态的用户-解锁进程映射数据结构（利用Map实现）。并且，在用户从未锁定状态变为锁定状态时利用线程池调度执行解锁线程，设定时间片为1小时，并在一小时后执行，时间片对应的Cron表达式如下：

```
”0 0 1/1 * * ?”
```

在解锁线程中，将通过调用静态用户-线程映射结构，在执行解锁操作后取消该线程的时间片调度，从而防止每个小时都会执行解锁线程，实现了在一小时后将用户解锁的功能。

3.2 邮箱重置密码

为了实现发送邮箱的功能，必须先实现异步邮件发送类。首先通过MD5数字证书技术加密密钥，最后使用Spring内置的JavaMailSender类实现邮件的发送。

为了实现异步调用JavaMailSender，使用RabbitMQ消息中间件，在config包中配置其高级消息队列设置，编写RabbitMQ监听器监听消息队列，最后在控制器中验证用户信息的正确性，如果验证通过，利用RabbitMQ内置模板的convertAndSend函数来发送验证邮箱。用户仅仅需要点击邮箱中的链接即可访问重置密码页面。

每个重置密码邮件的数字签名有30分钟的过期时间，若签名过期或是签名错误，将会在链接跳转页面给予提示并引导用户进入登录界面。若签名合法则显示注册成功提示，并在指定页面允许用户更改其密码。

该功能的主要实现难点在于使用RabbitMQ进行异步邮件发送。由于是一个完整的消息中间件框架，RabbitMQ的环境配置较为繁琐，并且其向Spring提供的API需要重新学习，具有一定上手难度。然而在粗略地了解了相关API功能和RabbitMQ运行原理后，可以发现RabbitMQ的使用方式并不是十分困难。在处理完验证邮件的发送过程后，剩下的控制器、视图页面编写等过程也能迎刃而解了。

3.3 @用户

为了实现@用户的功能，首先需要检测用户输入框中被@的用户并将其高亮显示。由于html中的textarea标签功能过于简单，故利用div标签加上属性contenteditable=”true”实现输入框富文本显示。

在JavaScript脚本中，通过绑定input propertychange事件，实时检测@用户文本，并将@至@后第一个空格间的字符串作为目标名字传给后端。后端控制器筛选目标名称（去重、去除数据库不存在的用户名），得到被@的用户（由于是实时响应输入的ajax，因此这些用户数据还需要额外存储在前端的隐藏标签中）。在微博发送表单提交后，后端将收到所有被@的用户名，依次将该微博存入对应用户的atMeWeibo列表（也即数据库一对多关系）中。在每个用户主页上，能够显示所有@本人的微博信息。

该功能的难点在于对于实时监听的数据如何进行存储（用户随时可能新@另一个用户）。在此处我们利用了隐藏input标签存储被识别的@后用户名，在用户发送微博进行提交后将这些数据也一并发送给后端，便于后端进行相应的操作。

3.4 Ajax减少刷新

在实现中，我们对点赞、评论、关注等多处采用Ajax技术，此处以关注为例（因为在关注的时候需要传参数多，实现更复杂）。

出现过的错误：

1. html中使用Ajax传递的参数书写形式不正确
2. 点赞数没有匹配当前weibo_id，导致在第二条微博处点赞时，点赞数显示在第一条微博
3. 在发布评论更新 评论显示栏 和 评论数 时，没有正确处理两者逻辑（因为Ajax是异步的，如果不放在success，可能会在还没有执行完评论数更新的时候就开始显示评论，此时新加入的评论就显示不出来了）

解决措施：

1. 使用正确的参数书写格式（见下文实现步骤）
2. 在传参时加入当前微博的id

```
<span class="icon-num" th:id="'likeNum'+${weibo.getId()}"
```

3. 把 评论显示栏 的函数 commenShow()放入 评论数 的success()函数内部

实现步骤：

首先，在myWeibo.html中对button标注id，onclick之后传递两个参数。此处参数传递书写格式应该注意形式正确（我们在此处传递过程中出现多次参数错误），正确形式如下：

```
<button class="attention-btn" id="attention-btn" th:onclick="attention([[${curId}]], [[${id}]])"
        th:if="${curId} ne ${id}">关注
</button>
```

其次，在json中设计attention函数，根据 curId 和 id 判读该登录用户访问的用户主页面是否是自己的用户主页，选择显示标签的格式（已关注 or 关注），并使用get请求发送到/checkAttention，其中携带数据 curId 和 id。

然后，在 UserController 使用 @ResponseBody 注解checkAttention函数，使用 @RequestParam 获取Ajax发送给后端的参数 curId 和 id，判断该登录用户访问的用户主页面是否是自己的用户主页，然后对应加入粉丝用户和关注用户列表。

最后，控制器将是否当前主页为已关注的用户的标志返回给前端，在 ajax 中更改 html 的样式。