**网上论坛系统详细设计报告**

1. **引言**

在科技迅速发展的今天，网络已经成为生活中不可或缺的一部分，计算机网络的普及使现代成为了一个信息爆炸的时代。无论是信息的交流，更新，传播，都进入了一个全新的领域。在这其中，论坛系统因其便捷性，简单性，普遍性成为了一种主流的信息交流平台。本系统即是在参考现有论坛系统的基础上完善功能，建造出一个稳定，方便的信息平台。

* 1. **标识**

本系统只适用于windows操作系统下安装及使用。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件状态：  [ ]草稿  [ √ ]正式发布  [ ]正在修改 | 报告编号 | iview网上论坛系统详细说明书 | | |
| 当前版本 | V1.0 | | |
| 编写人 | 刘雨君、张逸之、李文秀、刘欣悦、岳子琪 | 编写日期 | 2021-6-9- |
| 审批人 |  | 审批日期 | 2021-6-16- |
| 保密级别 | …… | | |

**1.2 系统概述**

网上论坛系统是便于用户之间交流的网络平台系统。实现游客浏览帖子，搜索帖子；游客注册正式用户，修改密码，正式用户修改信息、发布帖子、帖子管理，回复帖子等。

（1）软件系统的名称为IView论坛系统

（2）项目的任务提出者:本次课程设计指导老师：余仲星

开发者：刘雨君、张逸之、李文秀、刘欣悦、岳子琪

用户：用户较为广泛，无特定群体

编写网上论坛系统详细设计说明书的目的在于，从详细设计的角度明确网上论坛系统项目的业务品种、功能范围，使系统开发人员和产品管理人员明确产品功能，可以有针对性的进行系统开发、测试、验收等各方面的工作。

读者：项目经理、概要设计人员，编码人员，测试人员。

**1.3 文档概述**

本系统的网上论坛系统分为前后台两个管理系统:用户管理系统是在Internet上接受来自全国各地的用户访问的网上论坛系统 (网站)，管理员管理系统是在Internet内部用于管理员管理的网上论坛系统后台管理系统。本系统的用户管理系统实现的功能是游客、正式用户通过网络可以方便、快捷的发布帖子并回复帖子，可以通过系统实现线上交流。本系统的管理员管理系统实现的功能是论坛管理员，负责用户信息管理、帖子审核与管理、论坛建设，工作非常繁重，本系统能够帮助用户和管理员更加高效的工作。

**1.4功能需求**

（1）用户注册、登录、意见反馈；

（2）用户发表、编辑、删除、回复、浏览、检索帖子；

（3）管理员添加、删除、调整、置顶、隐藏论坛版块；

（4）管理员添加、删除用户和帖子。

**1.5 基线**

1**.** 网上论坛系统可行性分析报告 V1.0

2. 网上论坛系统软件需求规格说明 V1.0

3. 网上论坛系统软件架构文档 V1.0

**2 引用文件**

1.中华人民共和国国家标准GB T-8567-2006

**3 详细系统设计**

网上论坛系统的功能有前台系统和后台系统两个功能，如图3－1所示。

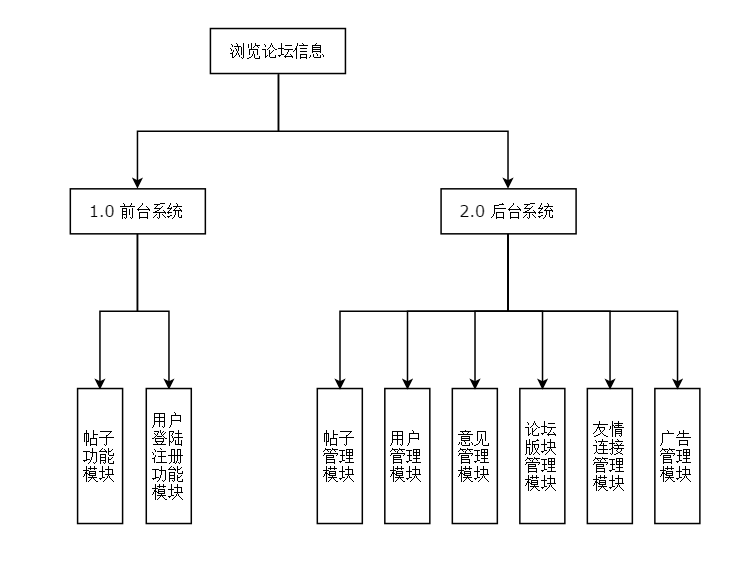


图3－1 网上购书系统功能图

**3.1 前台功能**

（1）用户模块：用户注册、登录、管理修改个人信息、意见反馈等。

（2）帖子功能模块：用户发表、编辑、删除、回复、浏览帖子。

**3.2 后台管理**

（1）管理员注册：网站管理人员注册，实现对系统后台的管理，对系统所有信息的控制。还需要对系统进行维护，同时还对普通用户的基本信息进行管理。

（2）帖子管理模块： 管理员可以删除、修改所有帖子。

（3）用户管理模块：管理普通用户，删除、修改、禁言等。

（4）意见管理模块：收集普通用户的意见反馈。

（5）论坛版块管理模块：管理员可以添加、删除、调整、置顶、隐藏论坛版块。

**3.3论坛主要处理内容**

（1）页面模块化：系统界面的设计使用模块进行处理，如把页面的头、尾页面内容，数据库的连接等做成单独的文件，在其它页面设计中需要时可以把其他页面的相应的内容包括进去了，有利于页面风格比较统一以及提高开发系统的效率。

（2）功能较完善，管理方便：功能方面包括讨论内容的展示、用户注册及登录、讨论的检索等各个方面，完整地实现了网上论坛系统的管理要求，同时由于应用了模板，对系统的管理维护非常方便。

**4. 数据描述**

1. 用户信息表：此表存放用户信息类型信息。

表4－1用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 宽度 |
| 用户ID | id | Int | 11 |
| 用户名 | username | Varchar | 10 |
| 密码 | password | Varchar | 20 |
| 性别 | sex | Varchar | 1 |
| 年龄 | age | Varchar | 3 |
| 手机号码 | telephone | Varchar | 12 |
| 电子邮箱 | email | Varchar | 40 |

1. 发帖信息表：此表存放发送的帖子的信息。

表4－2发帖信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 宽度 |
| 讨论ID | id | Int | 11 |
| 用户ID | userid | Int | 11 |
| 主题 | subject | Varchar | 60 |
| 内容 | content | LONGTEXT | Not null |
| 时间 | time | DATETIME | 6 |

1. 回帖信息表：此表存放图书的分类信息。

表4－3 回帖信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 宽度 |
| 分类唯一ID | PKId | Int | Not null |
| 分类ID号 | Description | Nvarchar | null |

（4）用户反馈信息表：此表存放用户反馈的信息。

表4－4用户反馈信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 字段名 | 类型 | 宽度 |
| 反馈ID | fadebackid | Int | 11 |
| 用户ID | userid | Int | 11 |
| 内容 | content | LONGTEXT | Not null |
| 时间 | time | DATETIME | 6 |

**5 程序设计说明**

逐个地给出各个层次中的每个程序的设计考虑。以下给出的提纲是针对一般情况的。对于一个具体的模块，尤其是层次比较低的模块或子程序，其很多条目的内容往往与它所隶属的上一层模块的对应条目的内容相同，在这种情况下，只要简单地说明这一点即可。图5－1是本程序系统的实际的基本的处理流程。

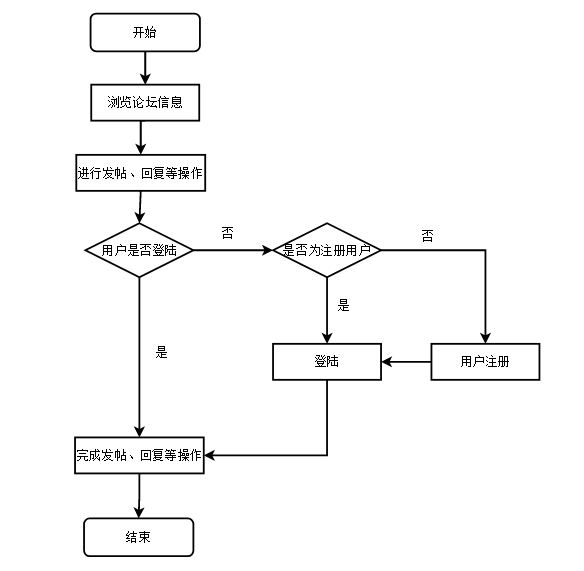
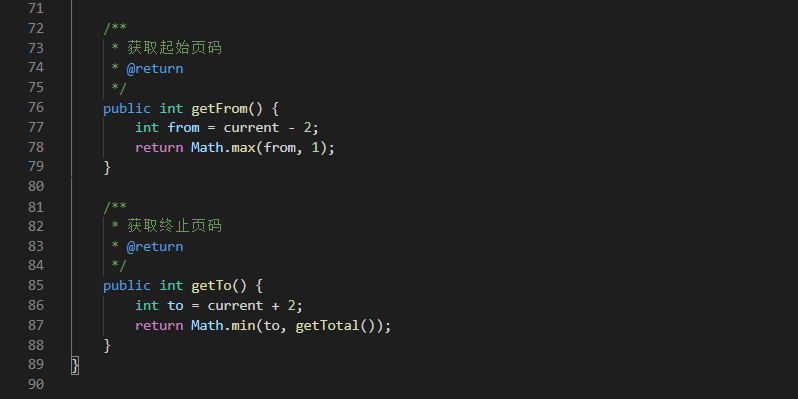


图5－1 网上论坛系统流程图

**5.1 & 5.2 程序描述及功能**

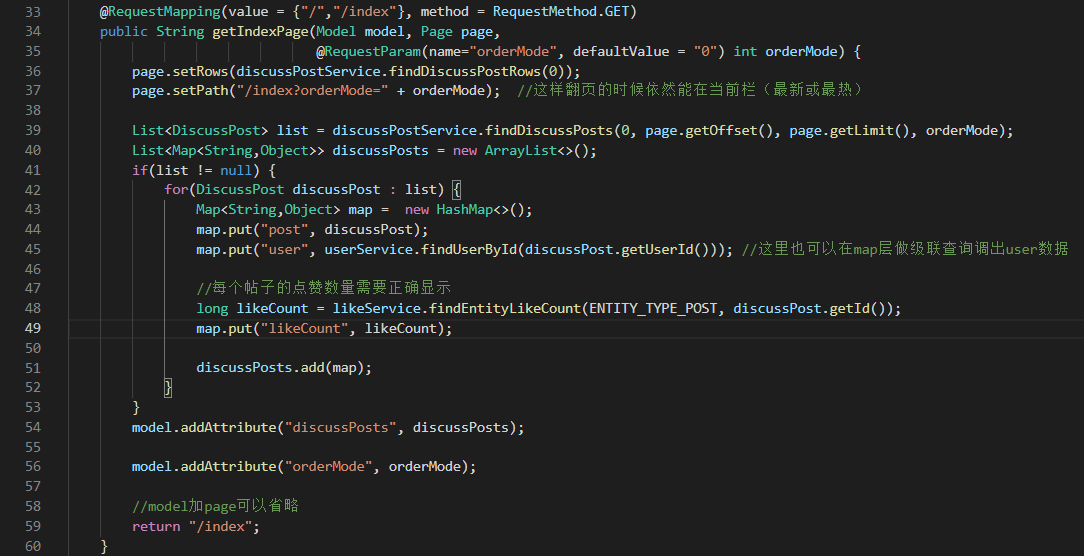
1. 权限模块
2. 首页：首页进行帖子的展示，依据一般后台开发业务流程进行实现：数据库建表->Java对应entity实体类->mapper层接口->接口对应的xml文件实现crud逻辑(Mybatis)->service层->controller层->页面。
3. 分页：通过Page类封装分页逻辑：





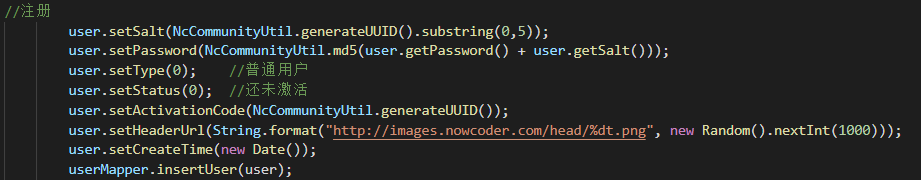
前台传来的current等参数通过controller的Page类参数进行封装，从而实现页面跳转。该模块可用于其他地方大量复用。

HomeController层对于Page类的使用：



1. 邮箱注册

对前台传来的注册表单数据进行判重判空和数据库匹配后，如果能注册，将用户数据插入数据库：



注意：密码的存储是经过加盐和md5加密的，防止密码泄露。数据库存储了该用户的盐和加密的密码。

然后发送激活邮件：



mailClient工具类封装了JavaMailSender进行邮箱激活，需要在配置文件中进行邮箱SMTP服务的配置。这里用子线程去发送邮件，防止卡顿时间过长。

邮箱里的激活链接即是通过"/activation/{userId}/{code}"访问路径修改用户的status字段，使其可用。

1. 登录

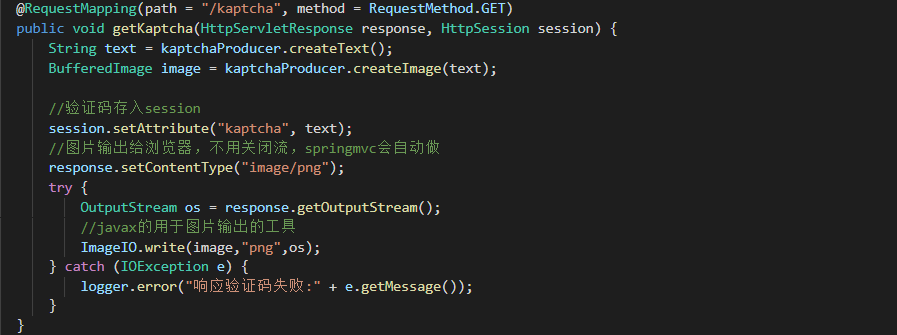
登录时检查信息正确性的逻辑和注册时基本一致，需要对账号进行非空、存在和激活的判断，对验证码进行判断，对密码进行非空和正确的判断，以及是否有rememberMe。登陆成功后生成LoginTicket存入数据库，记录了用户ID、ticket、过期时间等，ticket字段会被放入cookie中。

LoginTicket登录凭证之所以存入数据库，是考虑到了session在分布式环境下请求分发导致的会话状态无法保持的问题。

1. 验证码

使用google提供的Kaptcha实现验证码，登录时要检查验证码，逻辑如下：

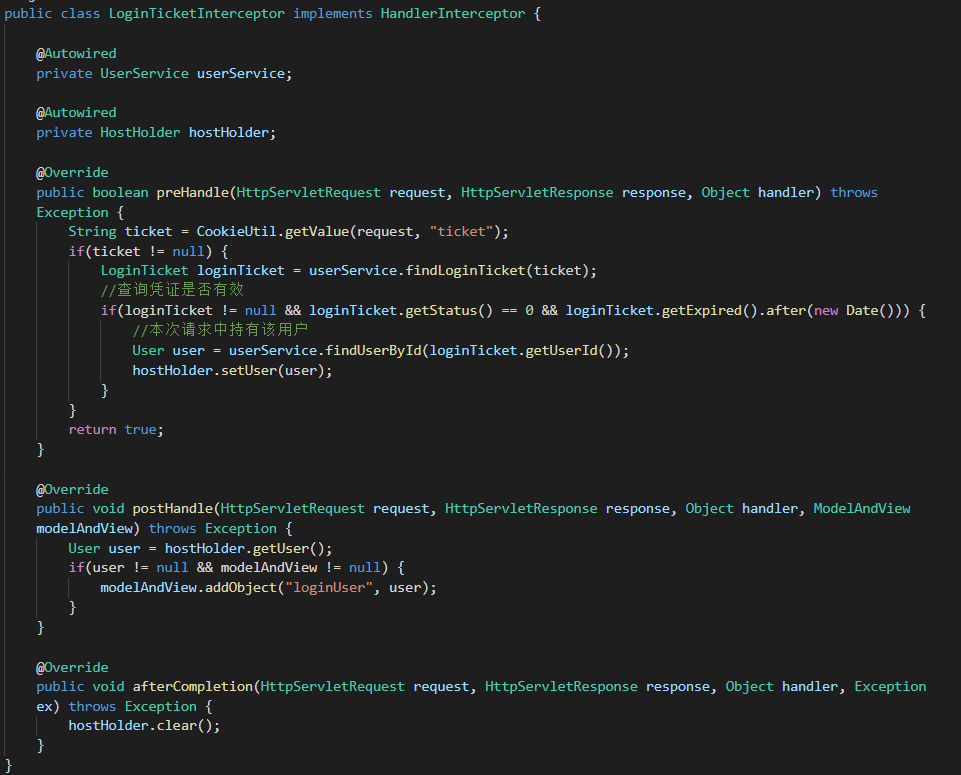
* 生成图片时存入session（后面用Redis优化）:



* 登录时从session中取值和表单值进行比对即可。

1. 状态保持

登录后需要进行状态保持，可以用前面提到的登录凭证+Interceptor+ThreadLocal实现：



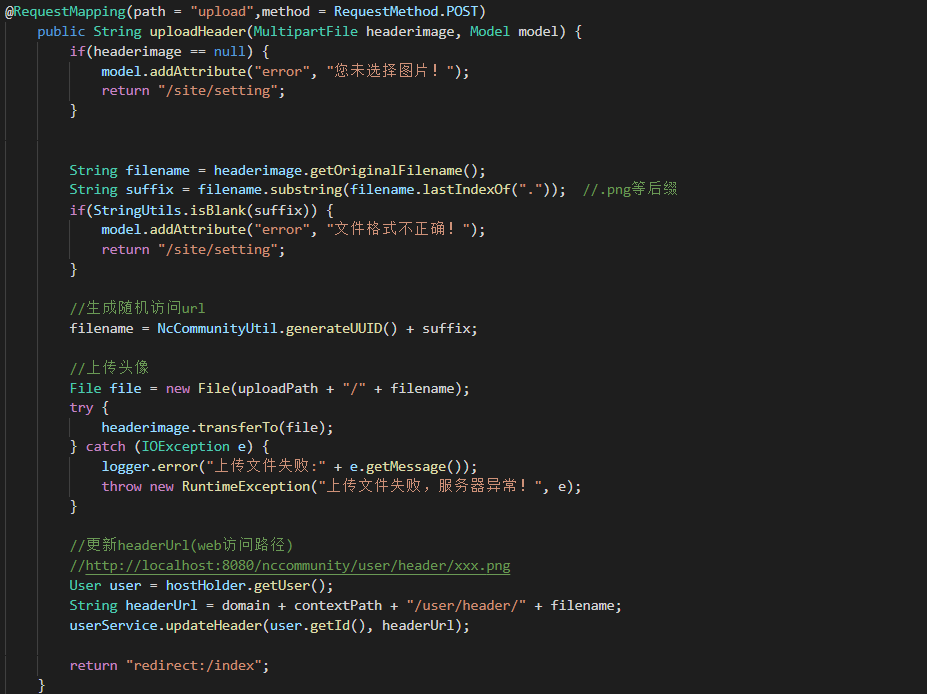
拦截器在preHandle时检查cookies中是否有有效ticket，有的话就在当前请求中持有用户信息。HostHolder类封装了ThreadLocal< User >，ThreadLocal的目的：Tomcat服务器会使用独立线程去处理每个请求，因此需要隔离多请求多用户，防止信息混乱。

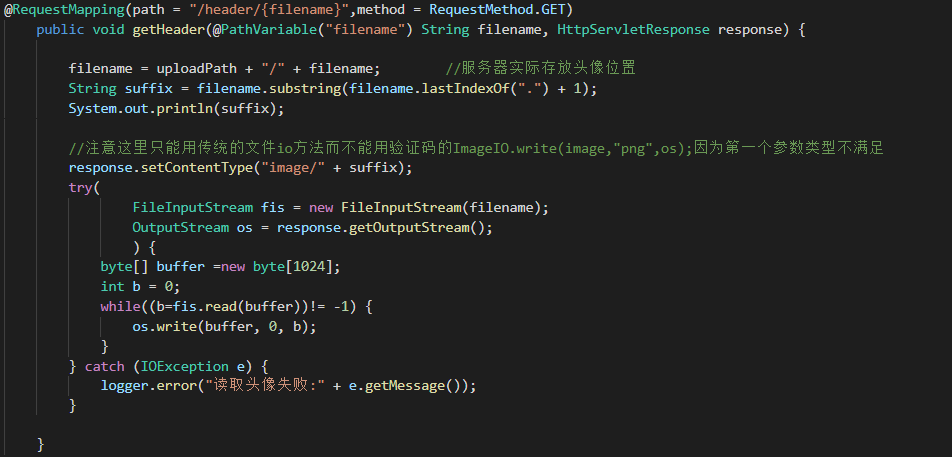
拦截器在postHandle时若发现该次请求中有用户信息，需要在 modelAndView中添加用户信息以保持状态。

在afterCompletion清空信息即可。

1. 设置头像

File文件上传，修改用户的头像链接使其可以通过url访问头像图片。图片存放位置可暂存本地。

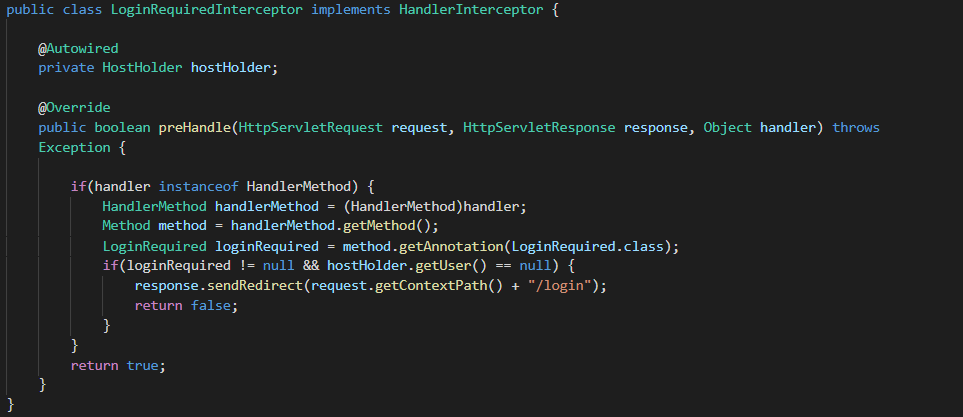




1. 简单的权限管理

设置页面和修改头像请求显然必须登录才能使用，可以通过注解进行简单的权限管理：

自定义@LoginRequired注解类，并添加到需要权限的方法上，然后通过拦截器进行判定。在访问当前方法时若有该注解则必须是已登录状态：

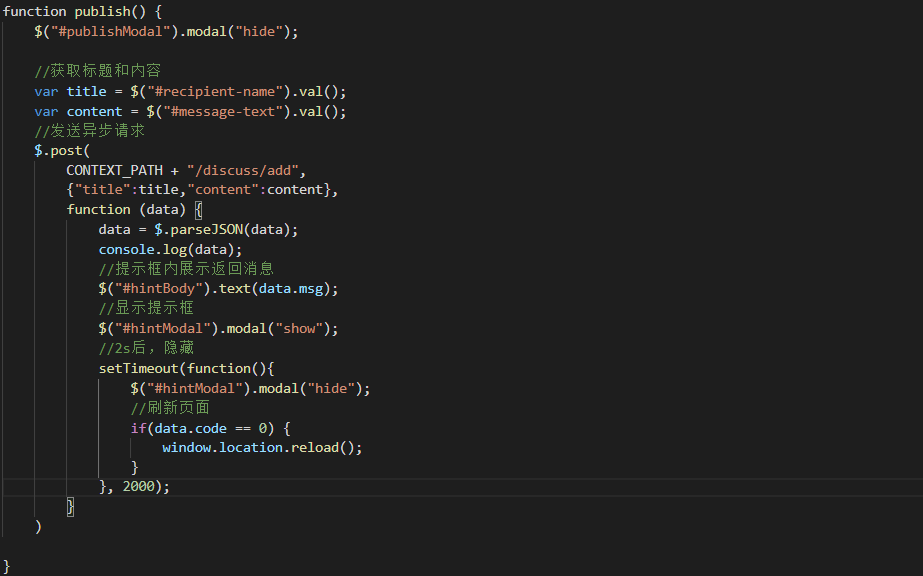


1. 发帖评论私信

发帖和评论为论坛核心功能，此外要能支持用户的私信。

1. 发帖

异步发送请求，以通过提示框展示提示信息，发帖后台就是普通的crud。



1. 评论

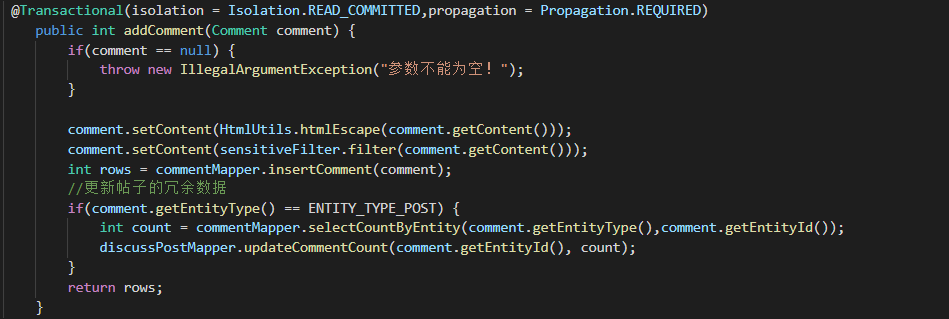
评论分为对帖子的评论（简称评论）和对评论的评论（简称回复）。

评论的entity如下：



entityType+entityId指示评论的对象（是帖子还是评论，然后具体Id为多少），当entityType为评论，且该条评论为回复时有效，为0表示该条回复评论的是对帖子的评论，非0则是代表该条回复评论的是回复。

另外注意，帖子entity中有个关于评论的冗余数据，因此有新评论产生时需要通过事务进行更新：



1. 私信

私信entity如下：



其中conversationId由fromId和toId拼接而成，小Id在前，如111\_112，表示111和112之间的私信。status记录私信是否已读，当用户进入私信详情页面时，会更新未读私信状态为已读。



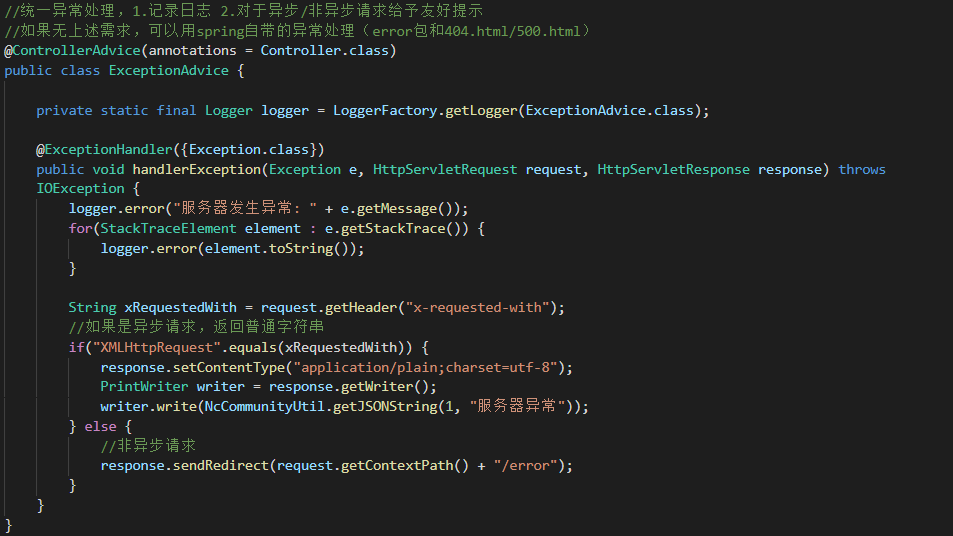
1. 统一异常处理

JavaWeb的思想是异常尽量不处理，而是往上层抛给controller去处理。spring对此提供了简单支持，若有4xx或5xx异常，则返回的页面为/error包下对应的4xx.html或5xx.html。但我们有如下需求：

1. 记录错误日志

2. 对于异步/非异步提供友好提示

因此我们可以用@ControllerAdvice注解实现异常处理：



1. 统一记录日志

我们需要知道哪些用户什么时候对什么方法进行了访问，因此最好能提供日志记录，一个解决思路是对于每个方法都进行硬编码记录日志，但这种思路显然违背了开闭原则，不利于扩展和维护。因此我们可以用AOP的思想解决问题，Spring对此提供了友好支持：



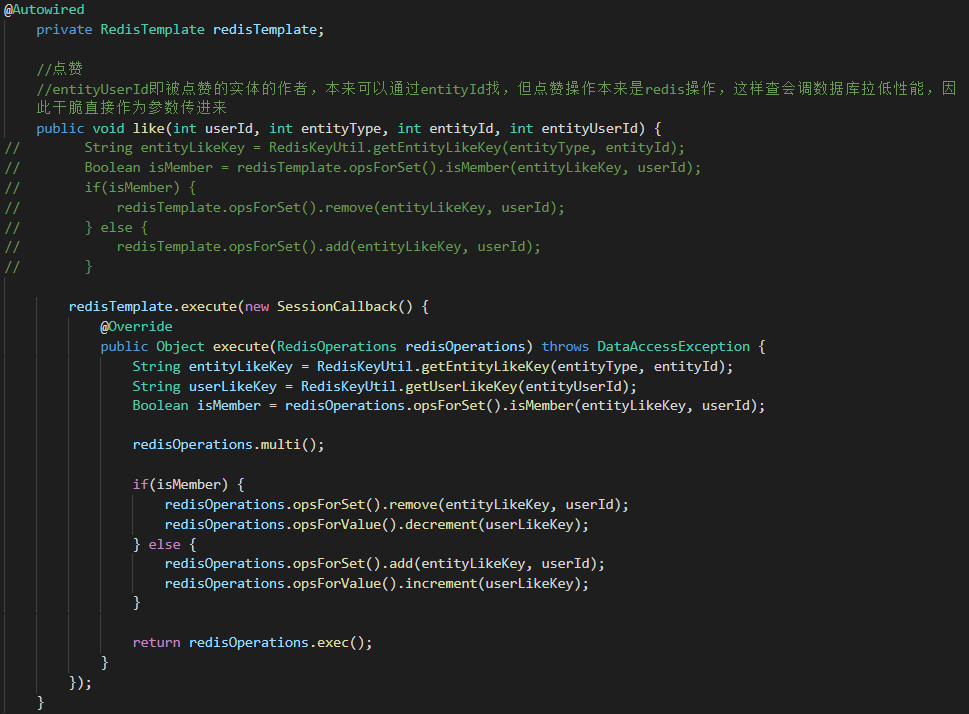
切入时机可以是pointcut调用前（@Before），pointcut调用后（@After），环绕pointcut（@Around），return后（@AfterReturning），异常后（@AfterThrowing）。这里只需@Before即可。

1. 点赞关注

点赞关注使用较频繁，是通过Redis实现的。我们在配置类中将RedisTemplate<String, Object>注入Spring容器，该Bean需要设置key和value的序列化方式（存入的有可能是对象，采用json格式进行序列化）。

1. 点赞

点赞用Ajax实现，可对帖子也可对评论点赞。根据返回的点赞状态和点赞数量进行正确的局部更新。Service方法如下：



其中RedisKeyUtil是一个构造Redis key的工具类，生成的key是由各个字段用冒号隔开的（Redis的惯用key命名方式）。

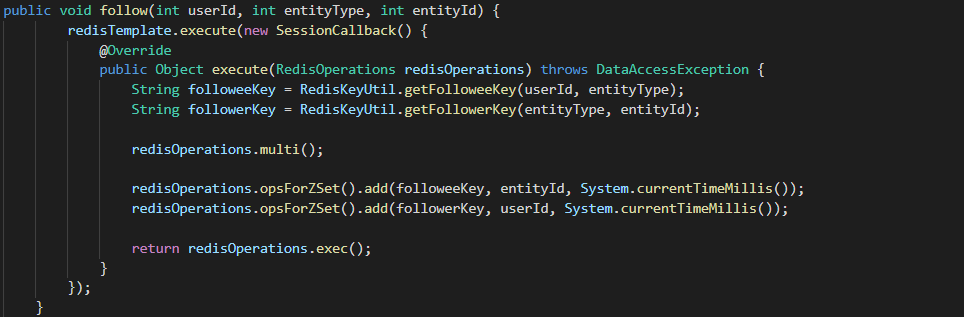
我们通过set数据结构记录某个实体（如帖子）的点赞用户Id，此外用一个string数据结构记录某个用户获得的赞总数。

这两个数据结构的操作需要用到Redis的事务一并实现。

1. 关注

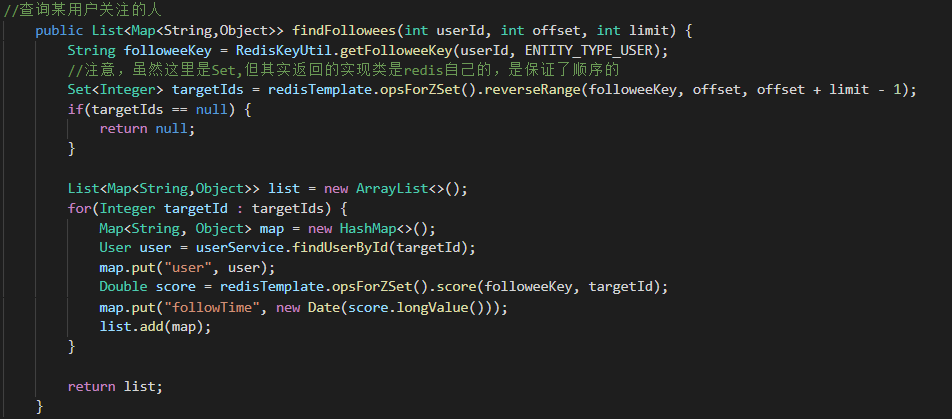
关注的对象可以是用户、帖子、评论。和点赞的区别在于：key 中需要包含关注者和被关注者这两个变量。

Service层定义follow和unfollow等方法，试举一例：



这里要用zset数据结构，在进行关注的人/粉丝列表显示的时候，可以根据关注时间进行排序显示。

你可以查看别人的关注列表，并对列表中的用户进行关注。



1. 缓存优化

下面用Redis缓存进行了三处地方的性能优化。

1. 代替session存储验证码

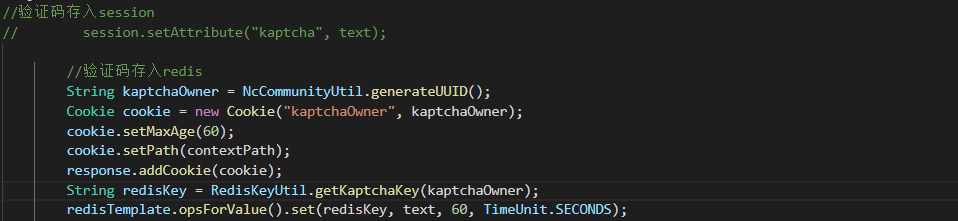
理由如下：

- 验证码可能会频繁访问和刷新

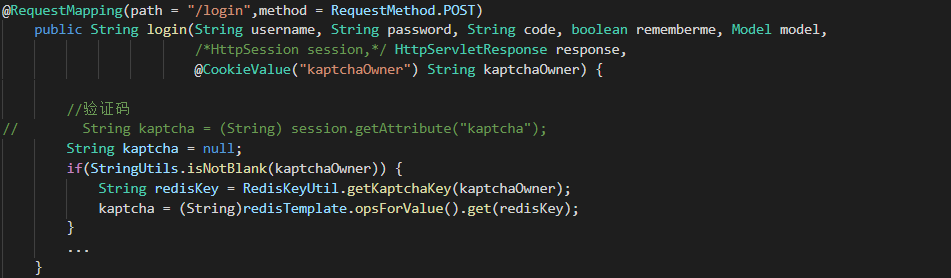
- 验证码只需要暂存，不需要长期保存

- 如未来涉及到分布式部署，能避免session共享的问题。

存验证码（"/kaptcha"请求）：



登录表单提交时取验证码：



1. 存储登录凭证

每次对网站的请求都会通过拦截器获取登录凭证，因此考虑到凭证的时效性和经常性，可以改为用Redis存储凭证而不是用数据库存储。

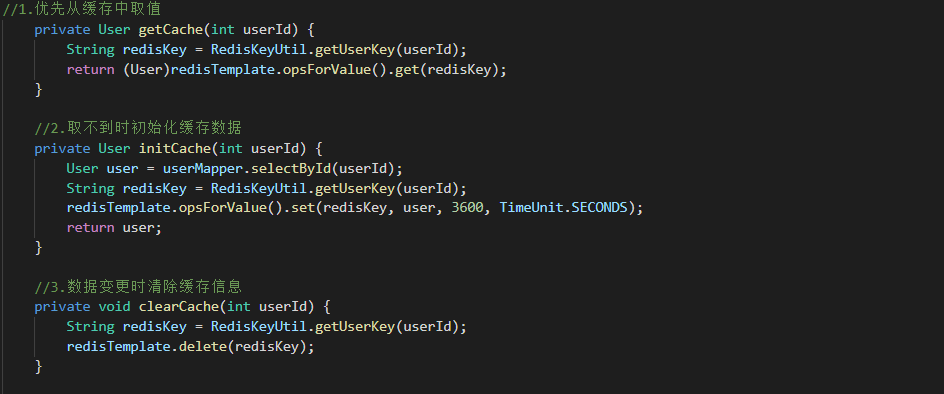
1. 存储用户信息

同上，每次对网站的请求都会通过拦截器获取登录凭证，然后再获取用户信息以保持登录状态，因此对于findUserById这个方法，有必要做Redis缓存：



这样，整个拦截器所调用的方法就不会涉及到数据库了。

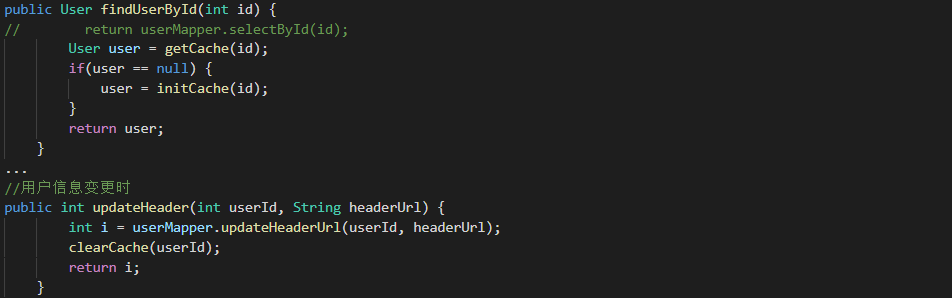
但用户信息的存储会涉及到Redis和MySQL的缓存不一致问题，需要解决：



关于缓存不一致问题，有很多解决方法，这里采用的是：当数据变更时，先更新数据库，再删除缓存。

当然无论是先更新数据库还是先删除缓存，都会有并发访问情况下的不一致问题和第二步操作失败的问题。

前一个问题可以采用延迟双删策略来解决。后一个问题可以用重试机制来解决。



1. 系统通知

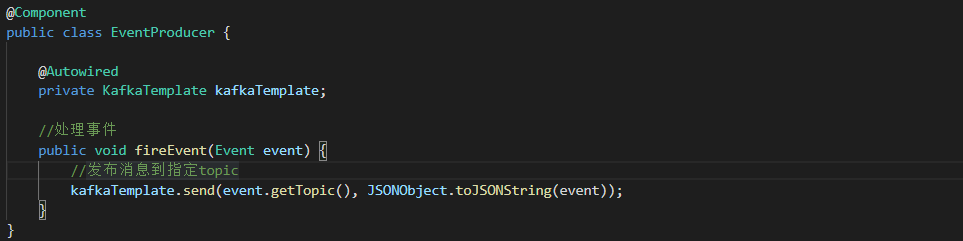
用Kafka做消息队列也能对系统进行优化。

原先Controller层的一些实现逻辑，可以转移到EventConsumer类中实现，享有消息队列异步削峰解耦的优势。例如系统通知的实现，本身和点赞、关注、评论的逻辑关联不强，且这些动作频繁发生，因此可以通过异步实现，提高性能。在例如后面用到的ES数据库的更新也可以用消息队列来实现。

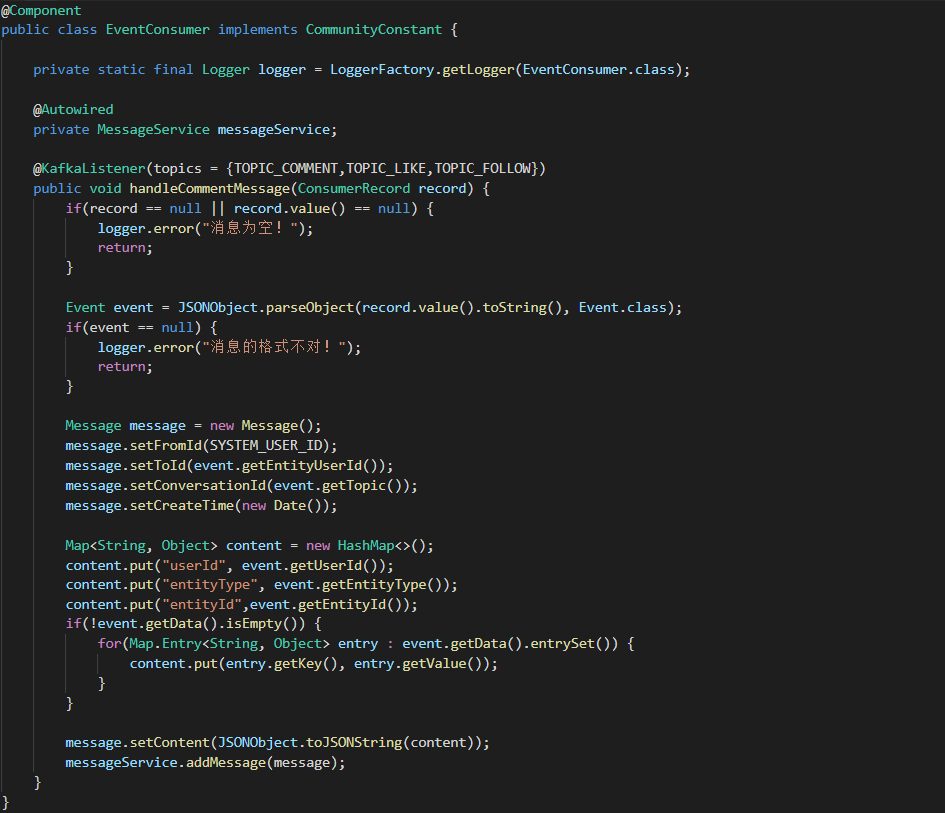
比如点赞时：



生产者：



消费者：

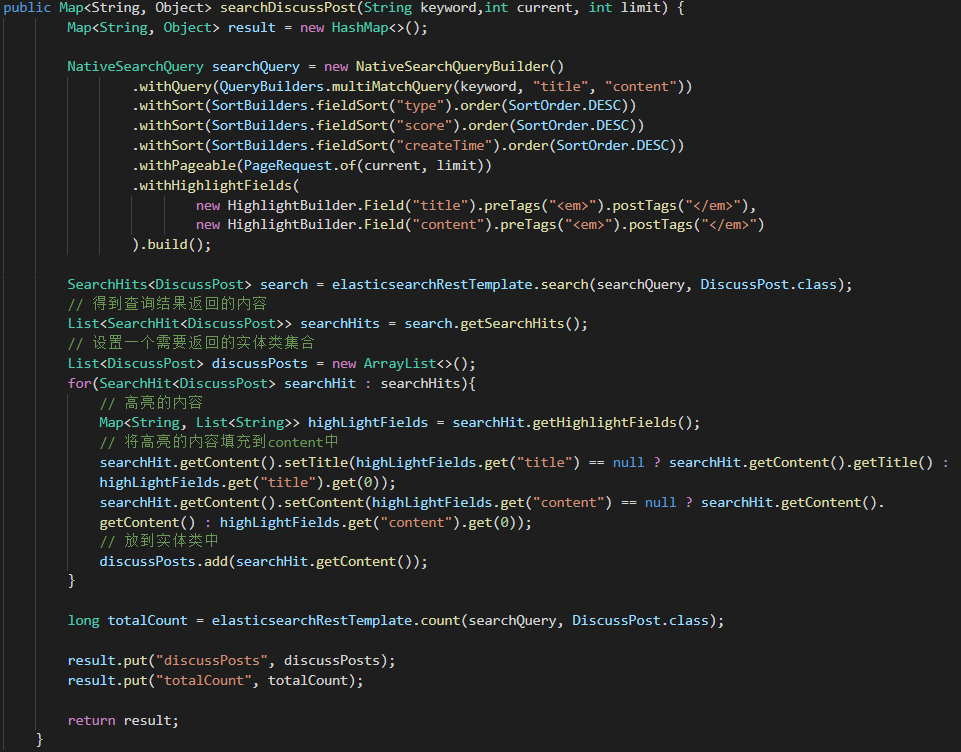


至于系统消息的查看、列表等实现，则基本与私信的实现差不多。（数据库中from\_id为1表示这不是普通私信而是系统通知）

1. 搜索

Elasticsearch相当于特殊的数据库，搜索就是对这个数据库的搜索。

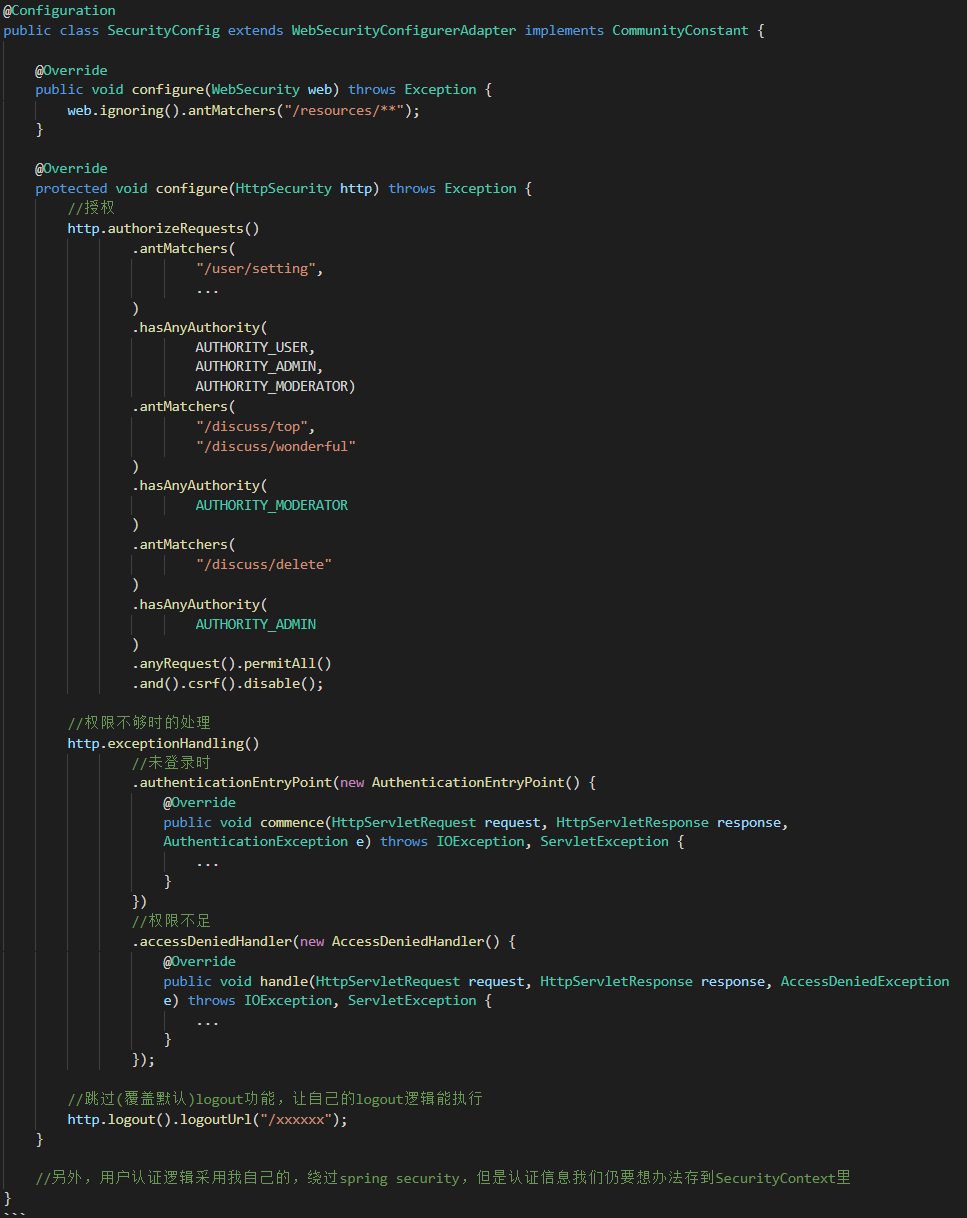
ElasticsearchService需要完成三个方法：save、delete和search，重点是search：



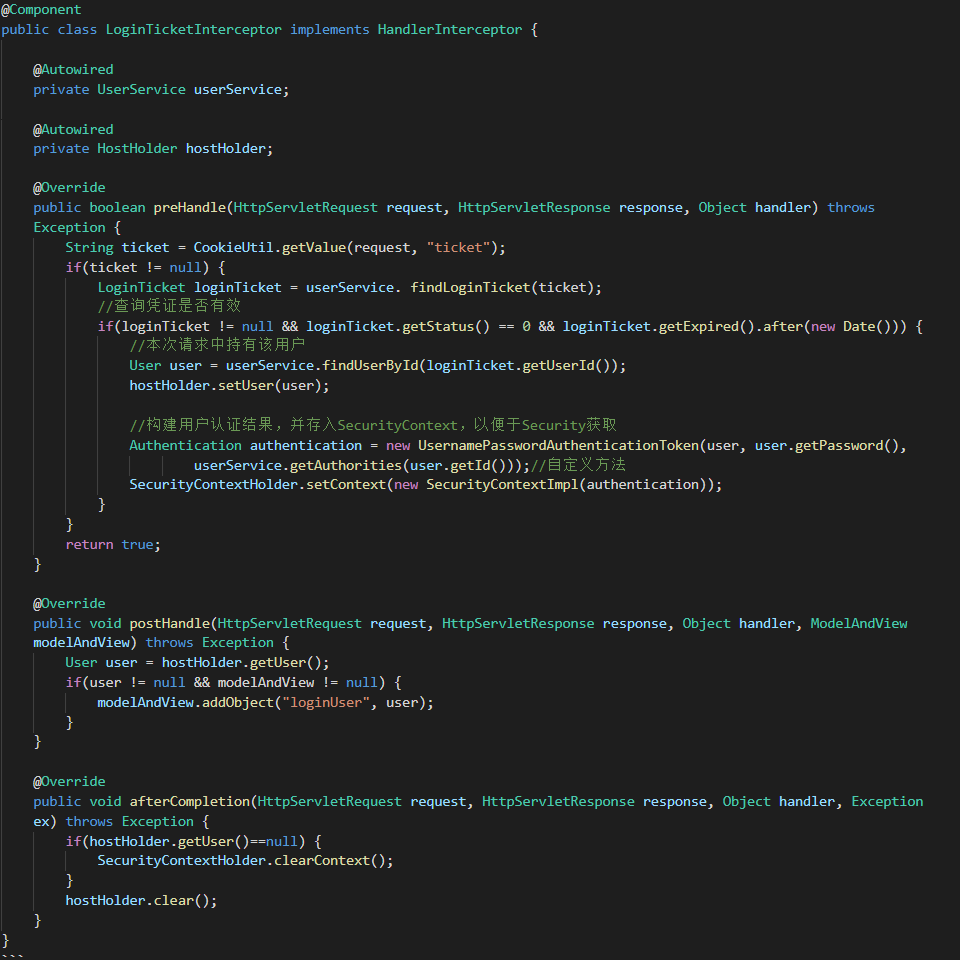
1. 其他
2. 认证授权

废弃了之前采用拦截器实现的登录检查，使用 Spring Security框架来进行统一认证授权管理。Security底层原理蛮复杂的，我们这里对其进行简单的使用。

授权方面，见Security配置类：



而用户认证方面，由于我们采用自定义的认证方式，因此无需采用Security提供的方式，但我们认证信息仍需要存到SecurityContext里，拦截器需要改进：



认证信息会在preHandle加入，afterafterCompletion处删除，logout时也会删除。

关于认证信息何时删除的思考：

1. 每次在afterCompletion时删除：

Security是基于Filter的，如果每次在afterCompletion时删除，那么下次请求时首先到达Filter，由于没有认证信息，会被判定权限不够，直接跳转到登录页面，即使已经登录。

2. 在afterCompletion时不删除，只在logout时删除：

如果这样虽然能保证权限与请求匹配，但是由于登录凭证会过期，用户信息会被清除，但认证信息却不会被清除，用户即使没登录也能访问，显然不合常理。

3. 因此正确做法为afterafterCompletion处如果没有用户信息就删除认证信息，logout时也删除。

1. 置顶加精删除

Thymeleaf有对Security的支持，可以从SecurityContext从获得权限信息。从而赋予用户不同的权限（置顶、加精、删除）。

1. 网站统计
2. UV独立访客统计



1. DAU日活统计

类似UV统计，只是DAU统计操作的是bitmap：



1. 热帖排行

使用Quartz实现热帖排行和更新，相比 JDK 的 ScheduledExecutorService 和 Spring 的 ThreadPoolTaskScheduler 的优势：

* Quartz 实现定时任务所依赖的参数是保存在数据库中，数据库只有一份，所以不会冲突。
* 而 ScheduledExecutorService 和 ThreadPoolTaskScheduler 是基于内存的，在分布式环境中，多台服务器会重复执行定时任务，产生冲突。

热帖排行和更新的实现逻辑如下：

1. 帖子的score字段计算方法自定义为log(精华分+评论数 \* 10+点赞数 \* 2)+(发布时间-纪元)。

2. 一旦涉及到上述score会变化的操作，如帖子被设为精华，或帖子有新的评论等，帖子id会被放入Redis的set中。

3. 每隔一段时间执行定时任务，会从set中pop帖子id出来进行分数的刷新。



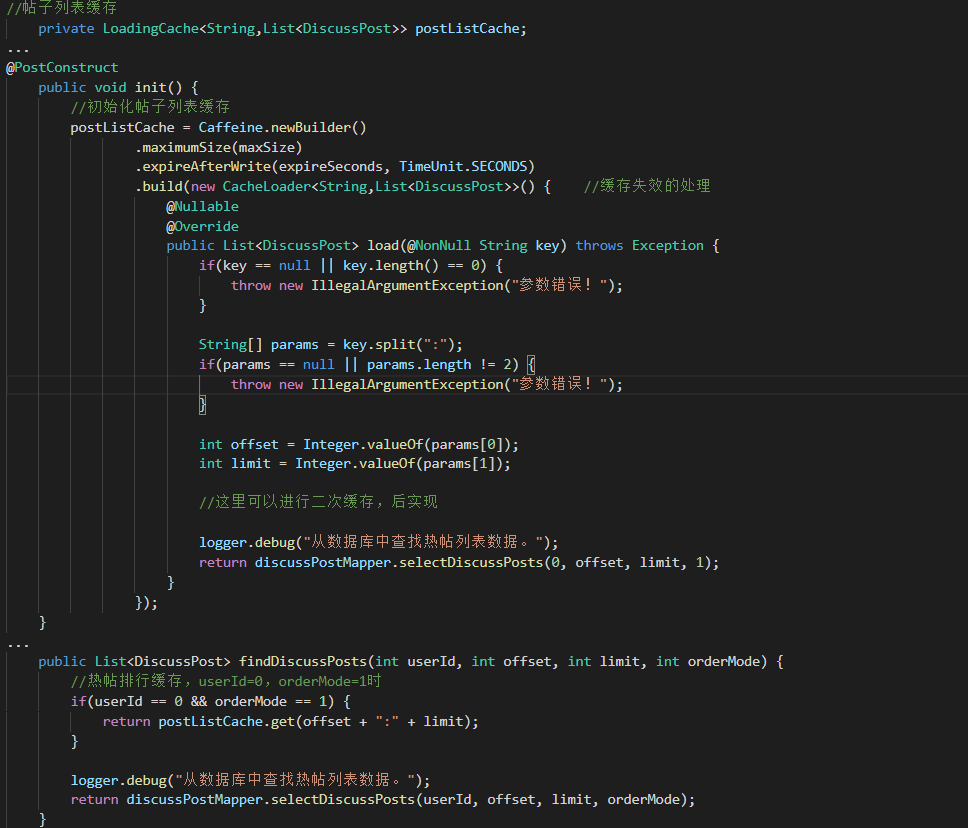
热帖页面根据帖子的score进行排序。

1. 本地缓存

在实际的部署中，服务器缓存由于就在服务器本机内，因此对性能的提升相比Redis更高。

对于热帖等与用户状态无关的内容可以存到本地缓存中。本项目用Caffeine实现，主要缓存帖子列表和帖子总数。

需要定义两个缓存管理器，一个针对热帖列表，一个针对帖子总数，可以通过@PostConstruct进行初始化。



**5.3性能**

（1）查询效率尽可能做到精准，信息维护做到简单易用，不需要对用户进行培训。

（2）用户输错数据都有提示信息，具有较好的容错性能。

**5.4输入项**

用户查询输入项

表5－1 网上论坛系统查询输入项表

|  |  |  |
| --- | --- | --- |
| **数据项** | **类型** | **数据精度** |
| 关键词 | 字符型 | 60 |

**5.5输出项**

用户查询输出项

表5－9 网上论坛系统查询输出项表

|  |  |  |
| --- | --- | --- |
| **数据项** | **类型** | **数据精度** |
| 主题 | 字符型 | 60 |
| 用户名 | 字符型 | 10 |
| 内容 | 字符型 |  |
| 时间 | 字符型 | 6 |
| 回复用户名 | 字符型 | 10 |
| 回复内容 | 字符型 |  |
| 回复时间 | 字符型 | 6 |

**5.7流程逻辑**

网上论坛系统的流程逻辑图如图5－4所示。

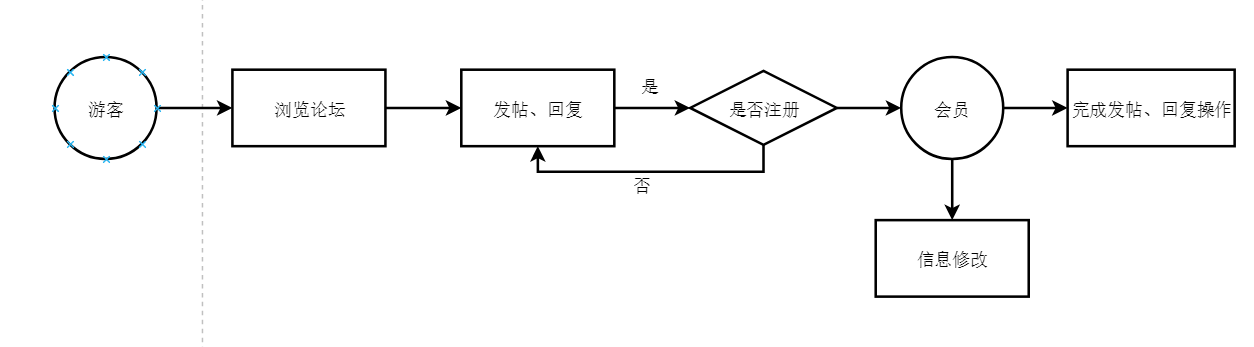


图5－4 网上论坛系统的流程逻辑图

**6 接口**

用户信息、管理员信息、发帖信息、回帖信息的接口图如图6－1、图6－2和图6－3所示。

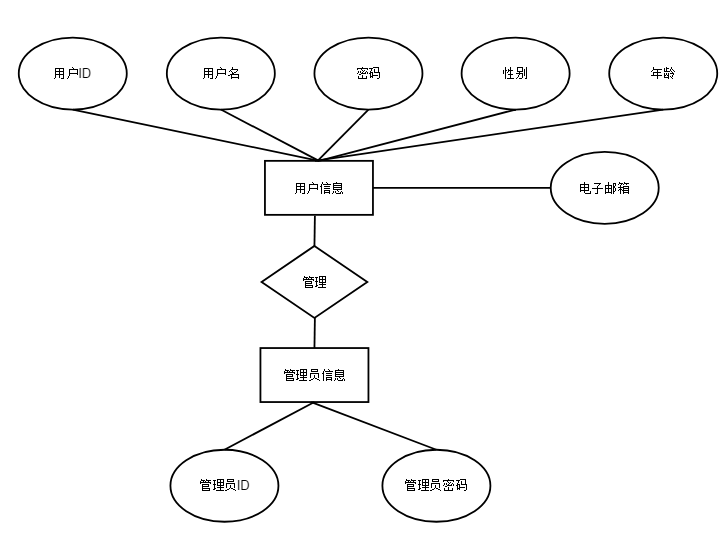


图6－1 用户和管理员信息接口图

**7 注释设计**

说明准备在本程序中安排的注释，如：

1. 加在模块首部的注释；
2. 加在各分枝点处的注释；
3. 对各变量的功能、范围、缺省条件等所加的注释；
4. 对使用的逻辑所加的注释等等。

**8 限制条件**

本程序仅支持在线的发帖功能，即以上功能都是基于联网条件实现，同时小组成员能力有限，此网上论坛系统无法进行一些复杂操作，比如在论坛中发视频、音频文件等，

**9 测试计划**

测试主要针对论坛系统的功能，需要以管理员与用户的视角分别进行测试，管理员主要测试能否进行注册，删除、修改任意帖子以及管理用户等操作，用户主要测试能否进行注册，发帖删帖等操作。

**10 尚未解决的问题**

此论坛系统无法发送大文件或者视频音频等

**11 参考资料**

[1]李玉林、王 岩 著，《ASP.NET2.0网络编程从入门到精通》， 清华大学出版社 2006.9

[2]郝 刚 主编，袁永刚、严治国、何宇光 著，《ASP.NET2.0开发指南》， 人民邮电出版社 2007.3

[3]张海藩，《软件工程导论》， 清华大学出版社 2003.12

[4]刘竹林编著，《软件工程案例开发与实践》，清华大学出版社 2009.8

**12 总结**

网上论坛系统是一种通过论坛来征得自己想要的信息的在线交流方式，有更高的效率和时效性。也最节约成本和资源，在论坛的交流过程中，无论喜怒哀乐都是在虚拟的环境中进行，也避免的正面的尴尬和冲突等，在论坛中，可以演变成任何角色，变换多种身份。论坛应具有知识性，趣味性，公平性，和谐性等优点。论坛给天下的网民都提供了交流的平台，为网络的互动性作出了巨大的贡献，通过论坛，网民们得以更方便的交流，更便捷的发表自己的观点

网上论坛系统是包含多种功能的在线系统。在这次设计中，小组完成了用户注册、登录、发帖、删帖等基本功能，对软件的开发过程有了进一步的认识，通过本次系统的开发也加深了对所学知识的巩固。