****

**本科毕业设计**

**（2018届）**

|  |  |
| --- | --- |
| **题 目** | 基于SpringBoot的响应式技术博客的设计和实现 |
| **学 院** | 计算机学院 |
| **专 业** | 软件工程 |
| **班 级** | 14108413 |
| **学 号** | 14108337 |
| **学生姓名** | 朱文赵 |
| **指导教师** | 马虹 |
| **完成日期** | 2018年 5月 |

**诚 信 承 诺**

我谨在此承诺：本人所写的毕业论文《基于SpringBoot的响应式技术博客的设计和实现》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

**承诺人（签名）：**

**年 月**

摘 要

移动todo

**关键字：**Spring Boot; Blog; Java;

**ABSTRACT**

Spring boot Todo

**Keywords:** Spring Boot; Blog; Java;

# 1 绪论

## 1.1 项目背景

早期的博客为了给不同的人提供一些不同的东西，会挑选一些特别的网站，并作简单的介绍。久而久之，慢慢的形成了不同种类，从不同的角度，又可以分为多种类型，如用户角度，可以分为个人博客、企业博客；或者按功能分，可以分为技术类的博客，微型博客。

而技术博客在众多博客类型中又有着一个不一样的地位。这些博客记录了博主曾经走过的路，是如何考研，做项目，学习的历程，算法心得，思考的过程，面试的技巧，如何进入谷歌，Facebook，微软这样的大公司实习或在学习或者工作中学到的新技术或是碰到的技术难题以及解决的方法。每个人对待各种技术的看法，已经消耗的时间可能都不一样，但他们对此都有相同的态度。那就是都希望通过自己的坚持和努力，解决一个个难点后，把这种喜悦或者说是成就感分享出来，让别人也能都一起学习和借鉴这个历程，分享自己的快乐，分享自己的收获。如我一样，相信很多人在遇到一些“疑难杂症”的时候，都会去网上苦苦寻求，当他们读到了这些优秀的文字的时候，他们能够快速解决问题或者得到一个问题的解决思路。而后他们能在学会知识的同时，也懂得了记录和分享的重要性。

## 1.2 本课题的国内外研究现状

国内外有很多优秀的技术博客，比如说国内的CSDN，简书；国外的Tumblr，Github Pages，The Hacker Blog，Stack OverFlow。同时还有一些个人站点，如：程序猿DD，廖雪峰的官方网站和阮一峰博客等。这些都不失为值得学习借鉴的博客，对比他们的技术特点，不难总结出一下几点：

### 1.2.1出众的设计界面

注重“表面” 的开发，页面相当于一个门面。用户的体验，我觉得是第一要义，良好的设计能够给自己的门户网站带来不错的流量。在保证前端美观的同时，还提供了性能良好的后台，能够把内容更好的展示出来。一般国外的技术博客都有这个特点,相比之下国内的有些博客做的不是很好，究其原因是国内技术太过于守旧，没有拥抱新的技术和框架，不愿做出一些新的尝试。

### 1.2.2友好程度不一

说到友好程度，那必须得说一下国内的一些优秀的博客站点，总是有人能够在在一定的时间，去学习一些比较热门的技术，然后分享出来。其中不乏有英文的文档，他们大多会翻译一部分，然后结合自己的见解，分享出来给大家学习和借鉴。当然用户也可以第一时间采用谷歌搜索或者Stack OverFlow中搜索自己的想要的答案，也可以在相关的社区提问，或者在Github中提一个issure，在别人交流，得到一些解答的思路。不过后者会对用户上网会有一定的要求，同时也需要有一点英文功底。

### 1.2.3 自定义的成分

国外很多有一些博客能够支持用户自定义代码块，让用户小规模建站，让他们能够自由发挥，得到一个与众不同的Blog。而国内目前还不支持这一小功能。

### 1.2.4 机器学习

这一个国内外很多站点都是支持的，因为他们有庞大的数据量，能够有足够的训练集，来丰富他们的机器学习模型。以此来构建用户画像，智能推荐给用户真正感兴趣的东西，而不是一旦用户搜索了一个东西，就疯狂地、无意地的推荐这些信息。

### 1.2.5 是否提供注册

有的不提供注册，只能博主使用，如一些个人站点。人们只能以游客的方式访问，但是绝大部分都提供了评论的功能，能够保证博主和其他人的交流度。

## 1.3 本课题研究意义

随着现在信息化时代的发展，越来越多的人喜欢把自己的一些见解和心得记下来，放到一个平台上，希望别人来观看和交流。很多人都会选择一些现有的博客站点，当然还是有很大一部分人选择自己建站，那建站就会有很多方式。比如采用开源的博客框架，利用他人的服务器，快速搭建，不过这样会有一些限制，如内存大小是固定值，网络问题，有一些部署在国外，国内访问会很有一点棘手，同时也无法自定义一些十分个性化的功能。于是就有了另一种方式，自己从零开始搭建，采用自己喜爱的技术和框架，制作一个正真意义上的属于自己的博客。

于是本着一个给大家提供更多技术交流的地方，采用一些目前比较热门的框架，深入用户的需求，给用户带来一个易于使用的博客。因此选用 Spring boot来作为技术支持，让建站变得更加容易，更易于部署，更加容易和其他技术栈结合，让我们能够专注于开发逻辑，技术的选用，而不在搭建和繁杂的配置项上。同时希望能够做到事实的将用户的信息同步到一些大数据搜索平台上，给用户带来一种快速，响应式的的体验。

## 1.4 课题研究内容

本毕业设计选用SpringBoot框架，结合Thymeleaf，SpringData，SpringSecurity，Elasticsearch等技术，旨在为技术人员设计并实现一款用于记录并分享技术文档的技术博客。通过该技术博客，方便技术人员记录自己工作和学习过程中的点滴，不断地进行技术的总结和积累，从而提升自己的综合能力，并通过博客这一平台，把自己的知识、经验、教训分享给大家，为志同道合者提供一个相互交流、共同学习的平台，促使更多的人共同进步。学习到别人的一些良好的设计思路、编码风格和优秀的技术能力，使我们的设计初衷。本系统主要面向web端的用户，希望能给用户更多的学习和交流的选择。

本课题在做了的网络问卷调研和梳理用户的一些想法，明确了用户对于web端的博客有着一种简单快速优质的愿望。此次设计中将主要实现用户模块、角色模块、权限模块、博客模块、评论模块、点赞模块、分类模块、标签模块、搜索模块等功能模块。会根据用户平常阅读的喜好，搜索的习惯，智能推荐给用户相应的优质内容。方便技术人员能够更加自由的书写博客，以及分享自己的学习心得，结识一些有相同爱好的极客。后期也可根据用户的需求做适当调整。

## 1.5 本论文组织结构

论文各个章节安排如下：

第一章：绪论。主要介绍本课题的项目背景、国内外的研究现状，并解释了其研究意义和阐述了后续做的研究内容。

第二章：开发环境与相关技术介绍。本系统为Web端系统，虽在Windows下完成开发，但基本没有平台的约束性。主要语言采用Java8，结合Spring的技术栈和Maven等一些工具进行开发。

第三章：系统分析。对系统的可行性分析进行展开描述，以及描述了用户的需求和介绍了开发和运行环境。

第四章：系统设计。对整个课题结合用户需求，来有效的进行拆解，拆解为各个功能模块，以及对数据库、界面和系统架构的设计。

第五章：系统实现。对各个模块进行细化的描述，阐述其中的设计原理和思想以及具体的逻辑，将最终的成品展现出来。

第六章：软件测试。对各个模块进行相应的测试用例设计，已经采用相应的工具进行自动化测试和压力测试。

第七章：总结与展望。对整个课题的开发过程进行一个总结和展望。

# 2 开发环境与相关技术

## 2.1 Windows系统介绍

Windows系统的建立，是为了人们能够简单，快捷的操作我们的操作系统，所有的事情都只需要人们通过GUI来进行，而不是像程序员们需要通过命令行来进行交互。比如极其强大的资源管理器，能够将不同的内容穿梭在不同的文件夹中。此外window8开始，已经支持屏幕触控，不过要想进行该操作，只能在微端，类似平板或者支持可触控的显示器等条件下，才能正常开启。

Windows8中加入了一种新的开机模式，叫做混合启动，能够保证用户在下次开机时能够快速的进入，类似于一种休眠的状态，大幅度提高了开机的时候硬盘读取和初始化的时间。

当然，我们的开发者通常不会满足于这些简单的需求，因为总是有一些东西是GUI无法满足的，所以微软的windows提供了一套cmd命令来供开发者使用。比如查看两个不同的网络是否互通的ping操作和查看当前的网络连接状态的netstat的命令。

## 2.2 Java语言

Java是一种能够运行在JVM虚拟机中语言规范，他是一种高级程序设计语言，可以在不同的平台下运行。具有很多优秀的特性，它结合了一些语言的优点，抛弃了一些语言的设计不足，比如说C#语言中的菱形集成，同时也包装了一些比较难以理解的原理，比如指针。可谓是取其精华，去其糟粕。

它是一种面向对象的设计语言，比起一些面向过程的语言，有很好的可阅读性，能够让程序员们进行简单和复杂中进行自由组合出一套优雅的程序。

而本课题主要采用java8编写，大量采用了java8的新特性，比如时间api和stream流式操作（lambda表达式），比起以往的代码（以前的java版本）代码量能够在保持简洁下而不丧失良好的语义。其中lambda表达式算是一个核心突破，它吸取了一些先驱如JavaScript或者符合JVM语言规范的后来者如Scala等设计理念，从而产生了现在这一个优秀的api，极大地简化了程序的逻辑复杂度。另外Java8也在改进以前的一些api，比如集合中的map，在hash碰撞之后在链表和红黑树的切换机制，这个是在java7及以前版本是没有的。

## 2.3 Spring boot框架

本课题采用了当前java最热门的框架Spring boot来进行编写，主要目的是利用它的开箱即用的特性，它能够快速的集成其他框架，让开发者能够专注于自己的业务逻辑，而不是陷入一大堆的配置文件中或者项目搭建中。

Spring boot本质上并不是一个新的框架，他只是在原有的Spring框架中进行了进一步的封装，将它打造成为一个具有良好开闭原则的分布式框架。

该框架现在保持了极大的活跃度，其衍生产品Spring cloud也是在微服务中站住一个不可动摇的位置，比起阿里的Dubbo有着一些良好的生态支持。

总的来说该框架在将来必将取代现有的开发模式，类似于SSM（Spring+Spring MVC+Mybatis）的开发模式，这将成为一种主流，所以我选择了拥抱将来，拥抱新的技术。

## 2.4 Spring security 框架

Spring security在本课题中占了很大的部分，有着不可动摇的地位。它主要用来控制用户的访问权限，相比以往的进行Filter过滤器、Intercepter拦截器或者AOP面向切面编程，更加的灵活和自由，同时也更加简单。它支持Spring EL表达式，能够结合注解的同时做到更加细粒度的方法层面的控制。

作为同为开源权限控制框架的shiro，它相比Spring security来说就会有点太过于“粗糙”，不能达到后者的粒度控制大小，而且在生态支持力度，也没有后者这么良好。后者能够和Spring 有着天然良好的集成，是作为开发者权限控制的首要选择。

## 2.5 Spring data 框架

### 2.5.1 Spring data jpa

Spring data jpa作为Spring整合Hibernate的一套优质良好的ORM（Object Relational Mapping）框架，它即保持了JPA（Java Persistence Api）的设计规范，比起原始JDBC（Java DataBase Connectivity）具有良好的操作性和可读性，极大的提高了开发者的开发效率。

该框架在持久层中占据着一个重要的地位，它即支持简单的单表操作也支持复杂的多表连接已经手写sql的支持。同时和Mybatis都有同样的设计风格，那就是面向接口编程，能够让用户只需要定义接口，而不用去管他的实现。只不过前者能够支持通过方法名的命名规则来进行sql的转化，具有非常好的可扩展性。

同时该框架也是在微服务中首选的框架之一，因为微服务中因为需要各种各样的链路调用，所以它需要保证各个服务之间延迟时间在可控范围，然后该框架能够很好保持这种良好特性，同时也能让编程更加容易。

### 2.5.2 Spring data elasticsearch

Spring data elasticsearch同样是spring开源出来的针对大数据搜索平台elasticsearch的api封装。它将一个文档化和结构化的数据变成，我们开发者能够通过我们熟悉的方式，类似于spring data jpa的操作，进行操作，从而屏蔽掉一些复杂的实现细节，这也是一种优秀的设计模式构建方式。

它提供了和elastic的官方文档同步的操作操作api，保证开发者在阅读文档后能够利用java elasticsearch api进行平滑的操作。同样他也支持原始的JSON查询体进行查询，给用户提供了极大的便利。

当然该框架是对elasticsearch的封装，其关键还是在elasticsearch的本身，该搜索平台是现在的优秀分布式搜索平台之一，同比还有solr等。但前者具有更加灵活的api和搜索设计，后者呢语法会稍微复杂一点，且实时搜索效率不高，没有像elasticsearch那么高速的分布式搜索效率。Elasticsearch相比较传统的关系型数据库（Relational Database）在搜索效率上有着本质上的提升，因为前者在插入数据的时候，都是将该数据变为非结构化的数据，如JSON，没有复杂的表关联，而是一些平铺开的数据，同时加入索引（index），加快了数据的访问速度。不在是关系型数据库中的结构化检索，同时搜索语法也比较后者有着本质的提升。

### 2.5.3 Spring data mongodb

本课题采用Spring data mongodb来操作mongodb存储一些图片文件信息，利用它对二进制文件良好存储性能构建我们的文件服务器。它和上面提到的elasticsearch都有一个共同点，那就是存储结构都是非结构化的数据，都是采用key-value的JSON来存储，从某种意义上来说，提高了可读性和程序的适配性。相比xml的方式具有很大传输优势，这就是现在基本主流程序都是通过JSON来传输的原因。同样都是以文档（document）来定义属性，保持了NoSQL（Not Only SQL）的高效性和灵活性。

在当前的大数据时代，这种非结构化的数据能够帮助开发者进行一些良好的数据分析操作，mongodb也是同样具有全覆盖索引操作，对检索的支持也是十分良好。Spring data mongodb在保证不变该数据库的操作原理的同时，封装出对java程序员友好的操作api，这也体现了java多态的设计原则，极大简化了开发者开发的效率。

## 2.6 Maven 工具

Maven这个工具简直是java开发者的福音，开发者再也不用手动去晚上一个一个的去下载jar包，只需要在pom文件中添加jar的坐标，然后点击引入，maven就会自动从你的setting.xml文件中配置的仓库去查找jar，并进行下载。

除了刚刚讲的下载功能，它还有三个生命周期，默认（default），清洁（clean）和站点（site），能够帮助开发者进行不同的操作。这个在传统的项目中是做不到的，开发者只能手动去用java命令去构建，发布到相应的站点。这在现在提倡自动化发布的时代，很明显是不符的。

同时在maven中由于加jar包比较自由，所以会出现jar包冲突的情况，不过这个依然可以很优雅的解决，现在的各个ide都可以添加maven插件，来查看当前的jar包是否重复。这个在原始的java web项目中简直是一个噩梦，开发者必须要去查看tomact中打包的项目中的lib目录jar包是否有重复或者有高低版本问题，当那种jar包内部有冲突的时候，这种情况将很难查找。

## 2.7 Lombok 工具

本课题采用Lombok来简化代码，目的是减少java中实体类的“模板”信息，比如get、set方法和构造函数，通过注解形式，在类编译之后自动生成，不用手动维护。这样做能够保证在改字段声明的时候，不用手动去改其他的关联信息。

同时该jar包，还支持Builder构建者模式，能够通过在类中添加@Builder注解，动态生成构建者信息，在类的字段特别多的时候，将会解救开发者的代码，不再看起来那么的愚拙。如果没有这个jar包的支持，开发者要想实现这种构建器模式，也必须要十分呆滞的再代码中添加样板代码，这样费时也费力，可以说是一个苦力活。

## 2.8 设计模式

本课题全部采用面向接口的设计理念，同时采用策略模式来构建不同情况下不同的代码分支，保证代码清晰可读。同时代码保持开闭原则，尽量做到用实体类来进行操作，这样防止今后该栋接口，只需要在类中加减字段，而不需要改动方法传输结构。

# 3. 系统分析

## 3.1 可行性分析

### 3.1.1 技术可行性分析

技术操作上，本课题采用spring boot为框架主体结合Thymeleaf，SpringData，SpringSecurity，Elasticsearch等技术。这些都是一些开源技术，各个社区都十分的活跃，互联网和书籍中都有十分成功的案例教学。难点是如何才能保证它们在有机的结合起来，发挥出自己的作用，做到各司其职。

整体课题都基于mvc设计模式来构建，前后端传输采用通用数据格式JSON，做到前后端分离，降低耦合度。

由于自己在平时有在学习这些框架，平时会去阅读官方文档和在社区中进行交流和别人进行讨论，对整体要用到的技术内容，都有着一定了解和使用率。因此整个案例的构建在技术上可行的。

### 3.1.2 经济可行性分析

由于整个课题用到的技术都是开源的，所以基本不会存在购买资源的问题。同时编译器也是用的IDEA，目前享受学生正版优惠，因此也不会有版权问题。

同时开发本项目虽然会有一定的服务器开销，但系统一旦完成，将会给各个开发者提供一个新的选择来进行结束和交流，具有不错的用户流量。因此本课题在经济上是可行的。

### 3.1.3 操作可行性分析

本课题是用web端展现给用户，界面美观，具有良好的操作风格，简单容易容易上手，提示信息良好。因此，在操作上是可行的。

### 3.1.4 法律可行性分析

本项目在法律范围内进行部署，不会触碰到法律的底线，传播积极向上的内容，为净化网络环境做出贡献。因此在法律是可行的。

## 3.2 用户需求分析

用户的需求在开过程中是十分重要的，具有是非常关键的地位，一旦偏离了用户的需求和用户的痛点，那么开发的结果将会不尽人意。因此做好用户需求分析，将会给后续的开发过程中，树立一个方针。用户的整体活动图如下图所示：

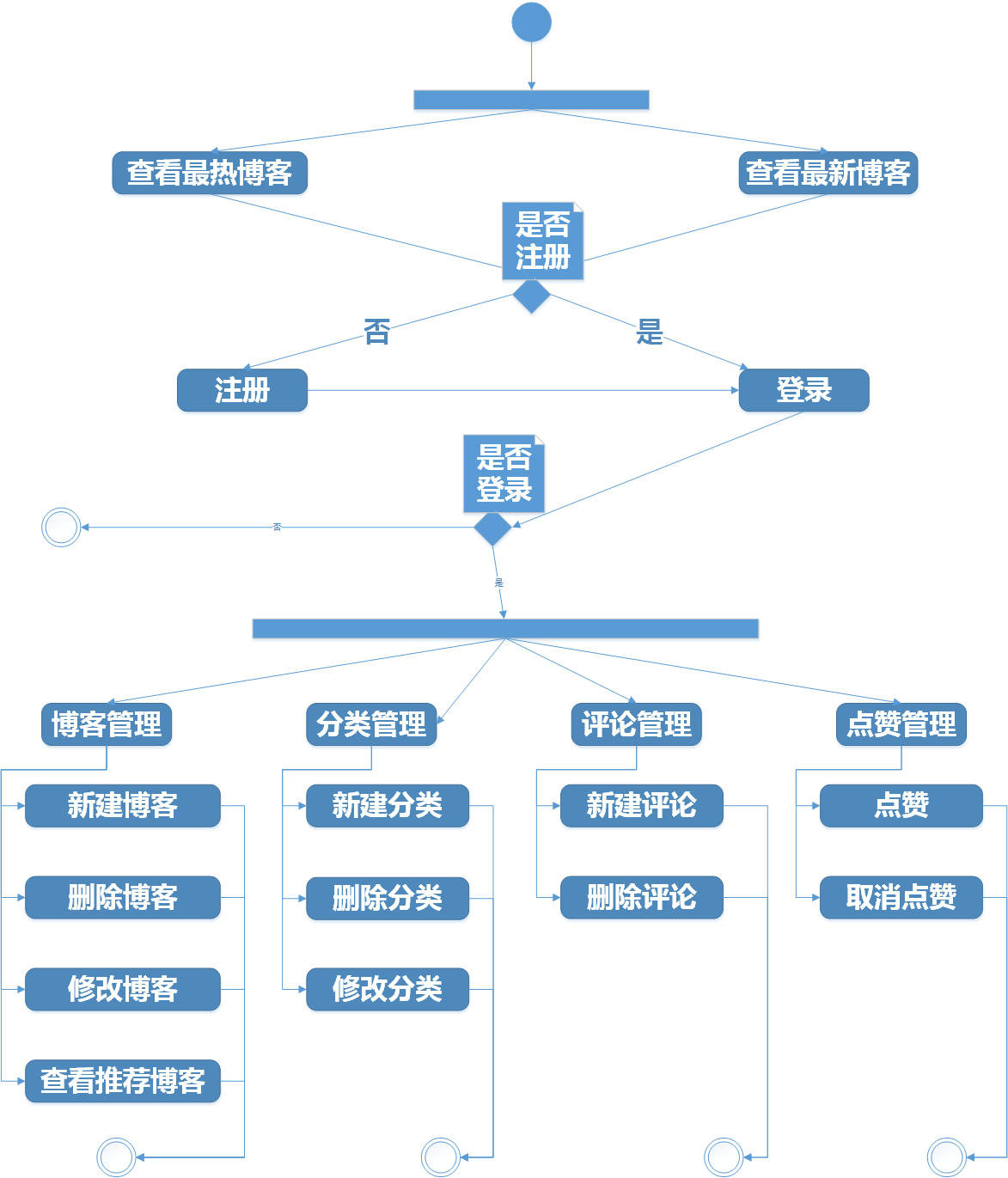


图3-1 系统整体活动图

根据需求分析，可以分为用户管理，全文搜索，博客管理，评论管理，点赞管理，安全管理，权限管理，个性推荐等用例，用例描述如下图所示：

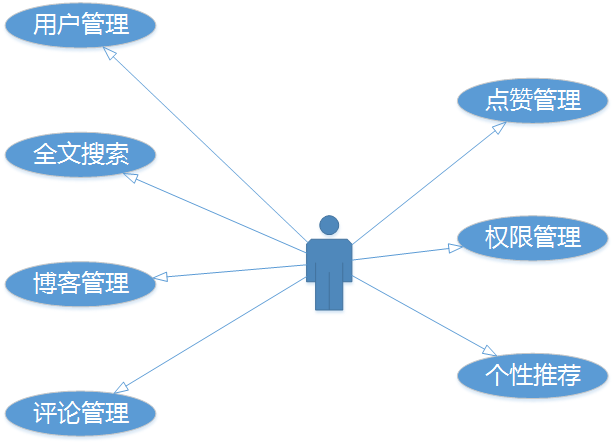


图3-2 系统总体用例图

### 3.2.1 用户管理模块

用户管理是在本系统中虽然只是一小部分，但是却把控着用户的操作权限，决定这用户是否权限操作一些模块。首先用户应当先注册，然后登陆，登陆后可以对自己的个人信息进行修改，如头像，昵称等个人信息。用户的简单活动图如下所示：

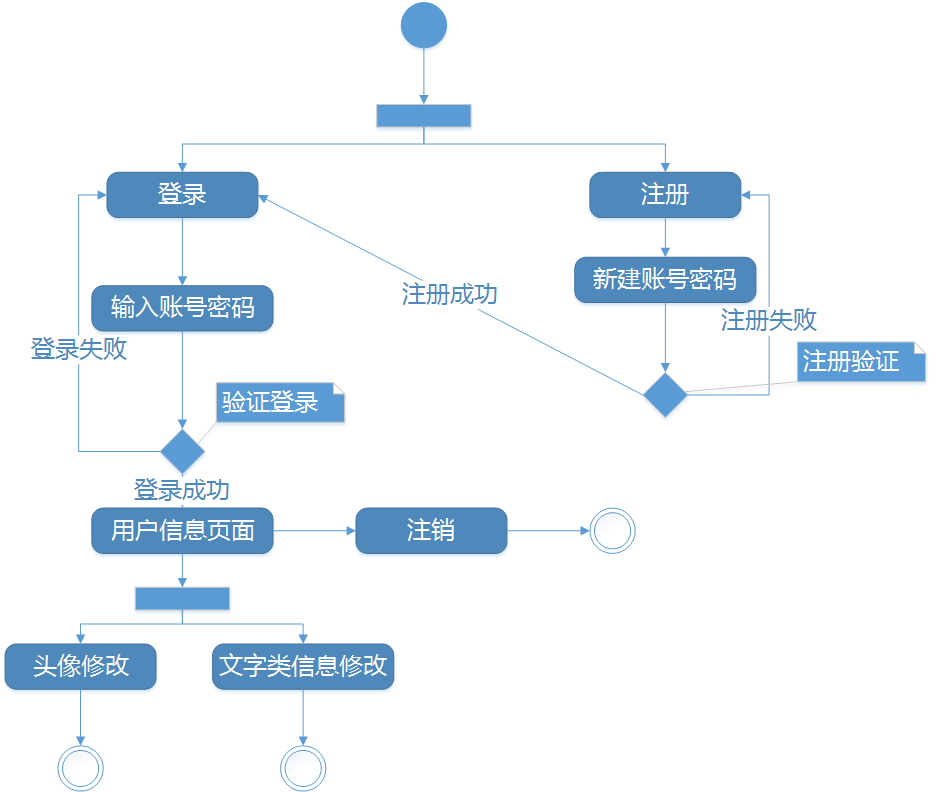


图3-3 用户管理活动图

用户管理模块包含用户登录、注册、注销、还有查看用户信息，用例描述如下：

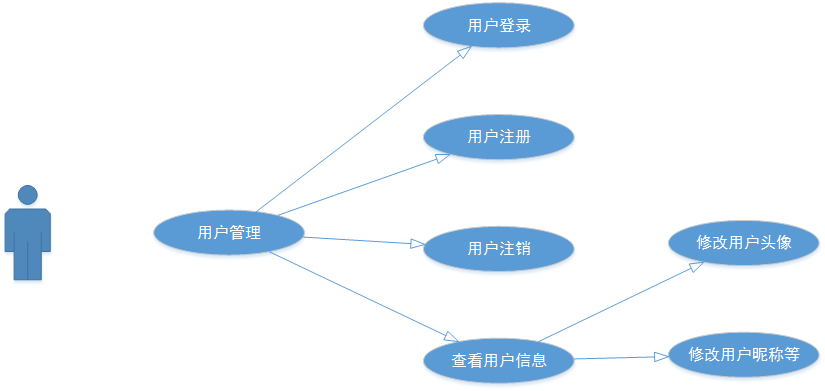


图3-4 用户管理用例图

用户登录：用户在登陆时输入邮箱和密码，验证信息是否和数据库匹配，匹配则登录至主页，否则则登录失败。登录成功后，系统会写入用户信息到cookie中，方便用户在不注销的情况下，下次能够正常登录进来。

用户注册：用户首次进入本系统，可以选择注册一个账号，成为本系统的用户，方便查看更为丰富的内容。用户输入自己的账号和密码，验证通过后，即可成为本站的用户。

用户注销：用户登录后，若想要推注很系统，可以选择注销操作，注销后将会清除cookie和thread local中用户信息的值。然后跳转到用户登录页，下次进入本站，要想查看一些比较丰富的内容，必须要进行登录。

查看用户信息：用户登录后可以选择查看自己的用户信息，其中包括对自己昵称等信息的修改和修改头像。

### 3.2.2 全文搜索模块

全文搜索模块是本课题中具有特色的一个模块，主要包括最新、最热查询、以及输入关键字查询和查看自己推荐的博客内容。同时还有显示活跃的用户和热门标签等。全文搜索活动图如下所示：

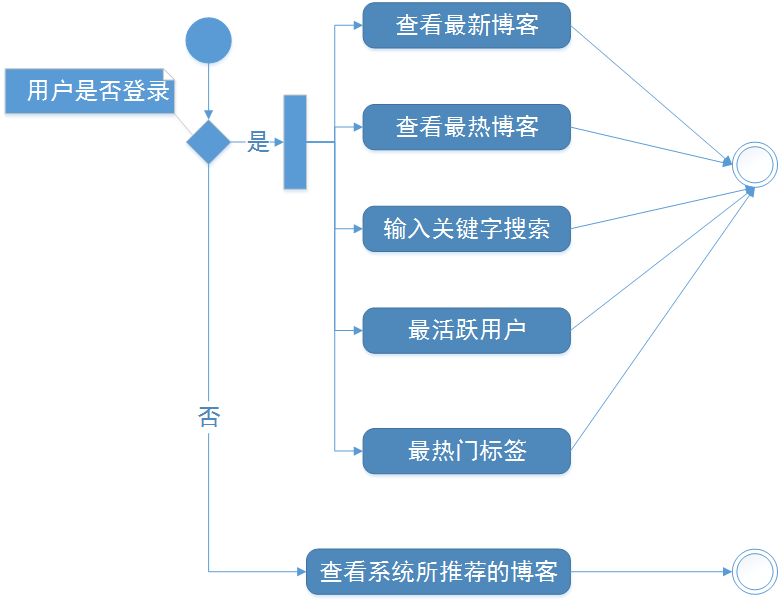


图3-5 全文搜索模块活动图

用户有些操作必须要在登陆后才能操作，比如查看智能推荐的内容等。简单用例图如下：

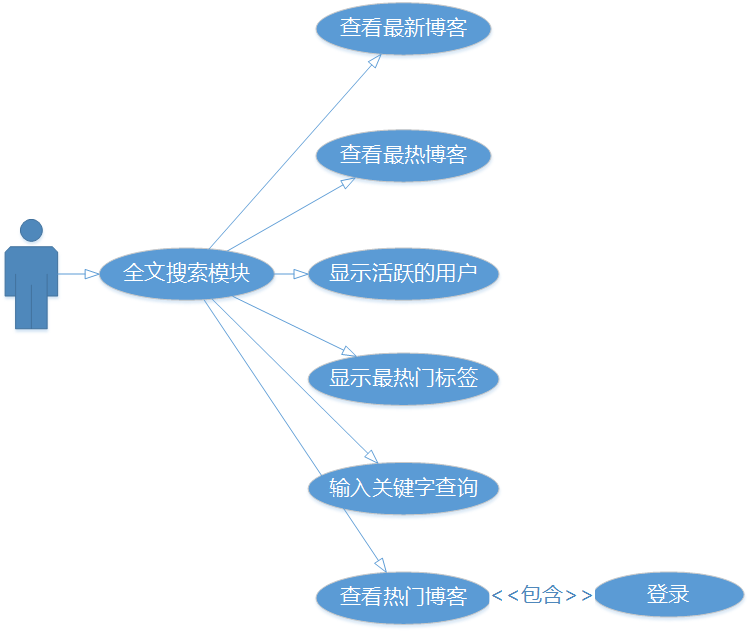


图3-6 全文搜索模块用例图

查看最新博客：用户可以点自己最新按钮查看目前博客中最新创建的博客，看到热门新鲜出炉的分享内容。

查看最热门博客：用户可以点自己最热按钮查看目前博客中最火热的博客列表，看到现在热门最关注的的博客内容。

输入关键字搜索：用户可以输入自己想要查找的内容，系统会根据关键字查找内容和标题匹配的博客内容，然后显示给用户。

显示热门标签：本课题将会显示最热门的用户自定义标签，同时也可以点击这些标签进行搜索。

显示活跃的用户：本课题将会显示最活跃的用户，根据创建博客的多少进行聚和统计。

查看推荐的内容：用户必须要先登录才能查看该模块下的内容，系统会根据用户的喜好，推荐给用户一些他自己喜欢看的内容。

### 3.2.3 博客管理模块

用户在登陆之后，可以创建、修改和删除博客，对自己的博客进行一个管理。博客管理活动图如下所示：

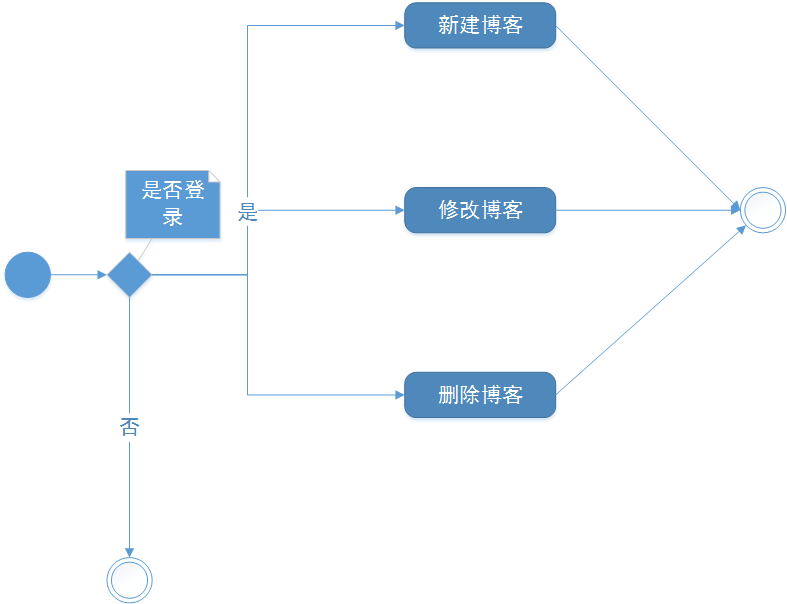


图3-7博客管理活动图

用例图如下所示：

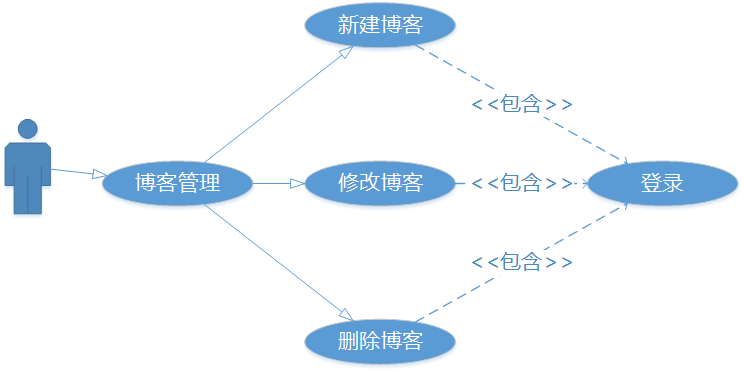


图3-8博客管理用例图

新建博客：用户登录后，可以新建博客，与别人分享自己的经验和快乐。

修改博客：用户登录后，可以点击自己之前的博客，进行修改。

删除博客：如果用户不想在管理这个博客，那么可以选择删除它。

### 3.2.4 评论管理模块

用户在点击一个博客后可以对改博客进行评论，同时也可以删除该评论。

评论模块活动图如下所示：

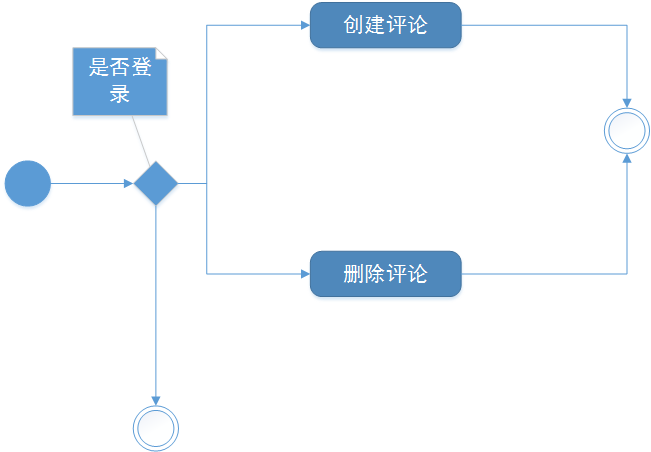


图3-9评论管理活动图

用例图如下所示：

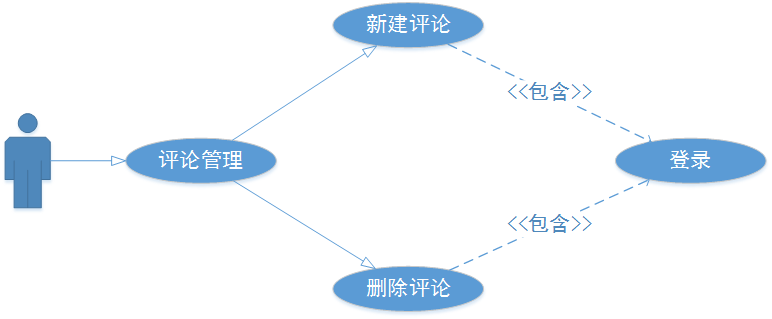


图3-10评论管理用例图

新建评论：用户可以对有想法的博客进行评论，和博主进行互动。

删除评论：用户可以删除自己不满意的评论。

### 3.2.5 点赞管理模块

点赞管理模块分为取消点赞和点赞，用户可以对自己喜欢的博客进行点赞操作，点赞管理活动图如下所示：、

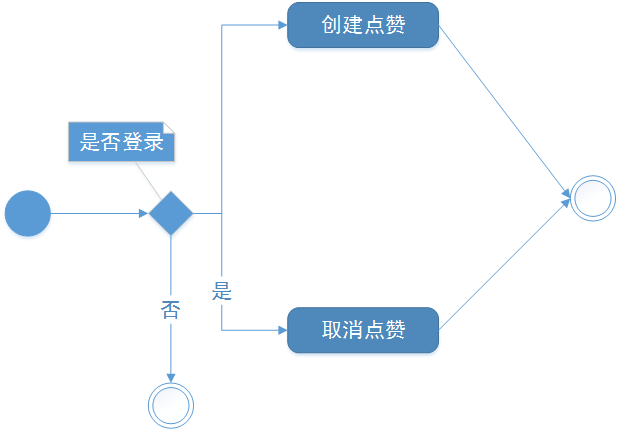


图3-11评论管理活动图

点赞用例图如下所示：

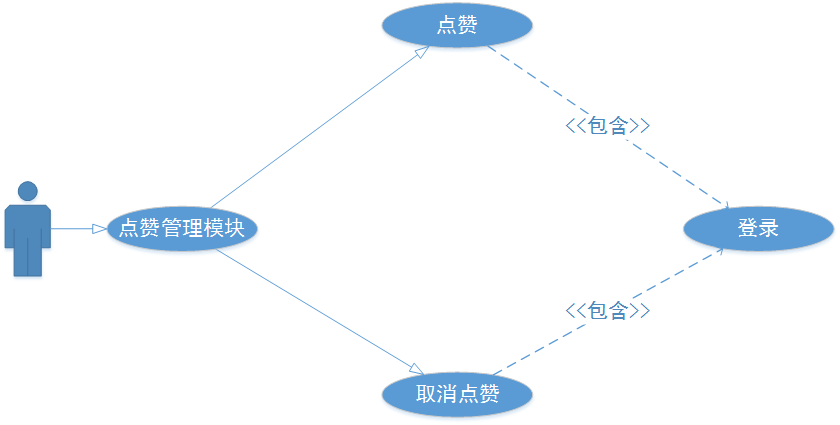


图3-12评论管理用例图

点赞：用户可以对自己喜欢的博客进行点赞，为博主的博客贡献一份力量。

取消点赞：同时用户可以对自己不再感兴趣的博客进行取消点赞。

### 3.2.6 权限设置模块

本文权限控制是非常关键的一部分，是过滤用户请求的重要环节，保证不同状态的用户能进行不同的操作。主要的权限分为管理员权限和用户权限。权限模块活动图如下所示：

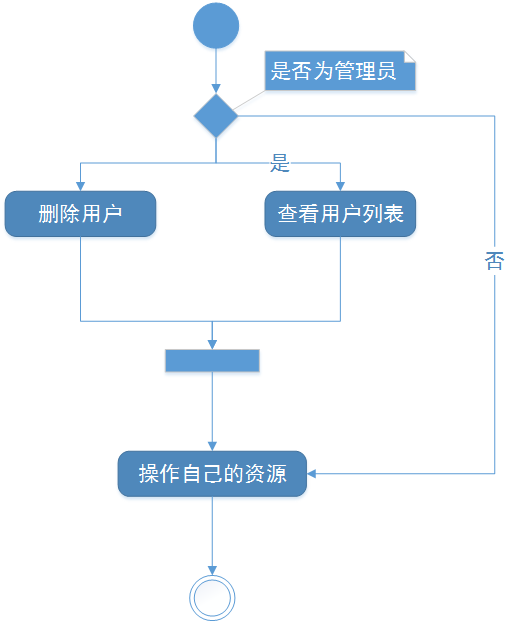


图3-13权限管理活动图

用例图如下所示：

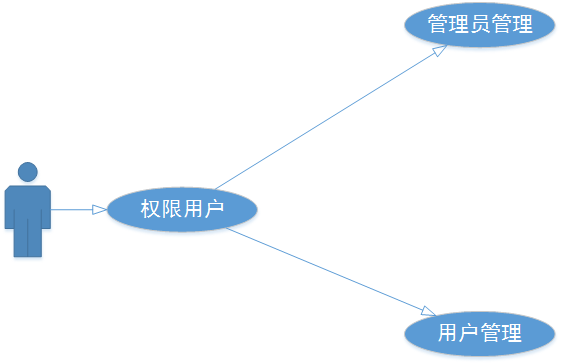


图3-14权限管理用例图

管理员权限：管理员可以查看整个用户列表，对用户进行一个管理操作，能够删除用户等。

用户权限：用户可以查看所有的博客，但是只能修改和删除自己的博客，同样点赞和评论也是只能删除自己锁创建的。保证了用户的权限的隔离。

### 3.2.7 个性推荐模块

个性推荐模块，是本课题的一个重点，能够在用户登录后，根据用户的喜好，智能推荐出按照匹配率排序的博客列表。个性推荐活动图如下图所示：

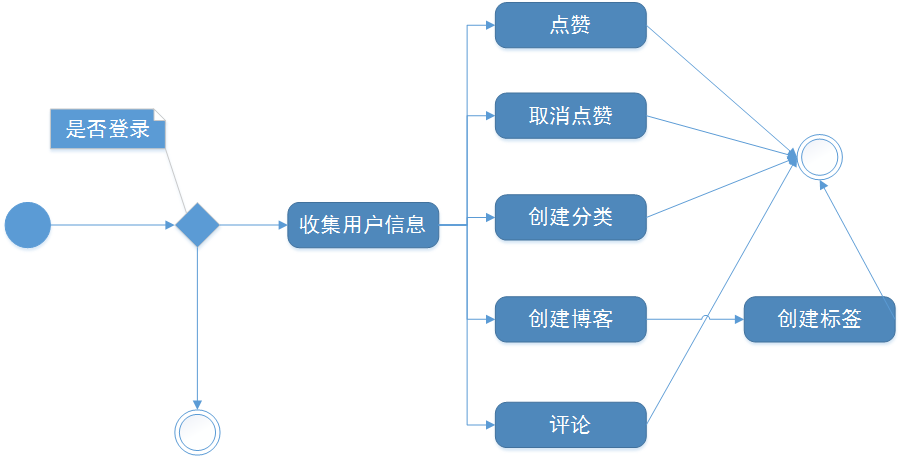


图3-15个性推荐模块活动图

用例图如下所示：



图3-16个性推荐模块用例图

个性推荐模块，我们需要采集用户的信息，后台会根据用户的点赞、评论、创建的博客和创建的标签来动态的解析，生成一个推荐列表给用户。

### 3.2.8分类管理模块

用户可以在创建博客的时候，选择创建好的分类，用来绑定每个博客属于什么分类。便于用户进行管理自己的博客列表。分类管理活动图如下所示：

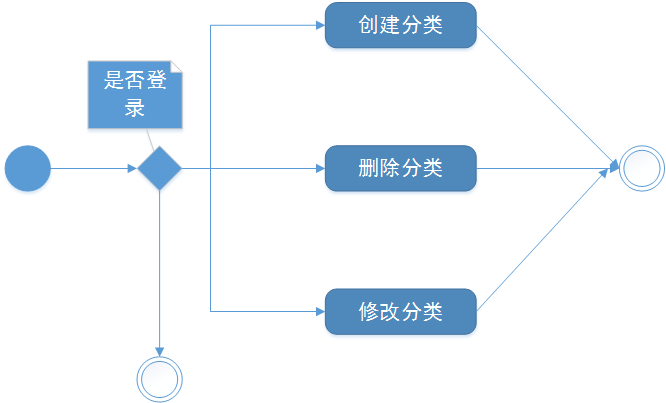


图3-17分类管理模块活动图

用例图如下所示：

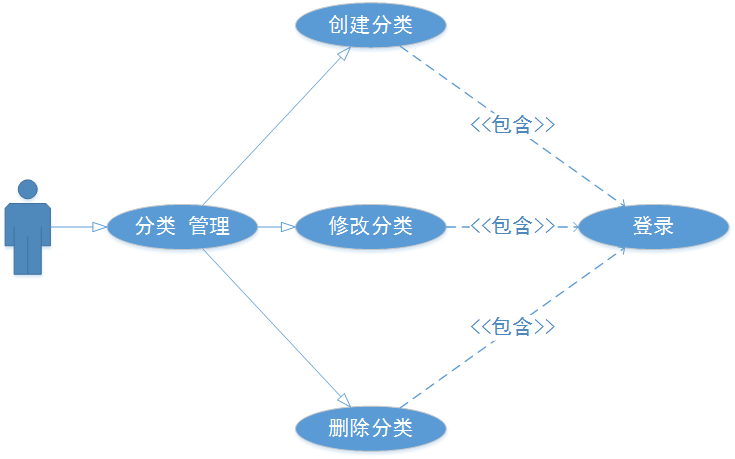


图3-18分类管理模块用例图

新建分类：用户可以根据自己的喜好，新建博客分类，用来管理自己的博客，让自己的博客看起来更加的条理化。

删除分类：用户如果对之前的分类不喜欢，那么可以删除该分类，该分类下的所有的博客列表会到一个未分类的栏目中。

修改分类：用户可以对自己所创建的博客分类进行修改，将他改为自己所想要的名字。

### 3.2.9标签管理模块

标签管理区别于用户管理，它主要是为了细化每个博客的所属标签，让同一个博客可以根据用户的喜欢，添加自己的标签。同时也方便后台进行数据的清洗和过滤。标签管理活动图如下图所示;

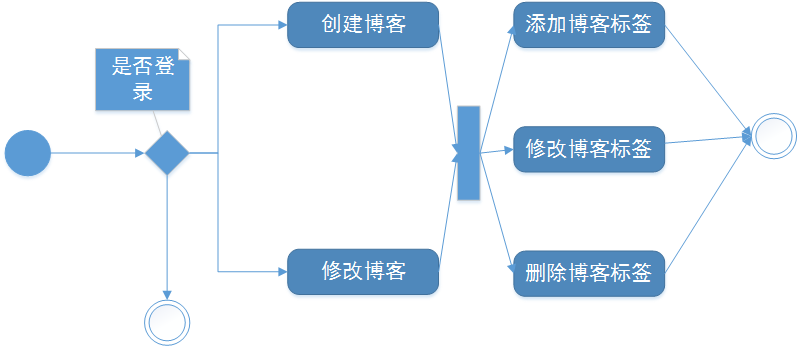


图3-19标签管理模块活动图

用例图如下所示：

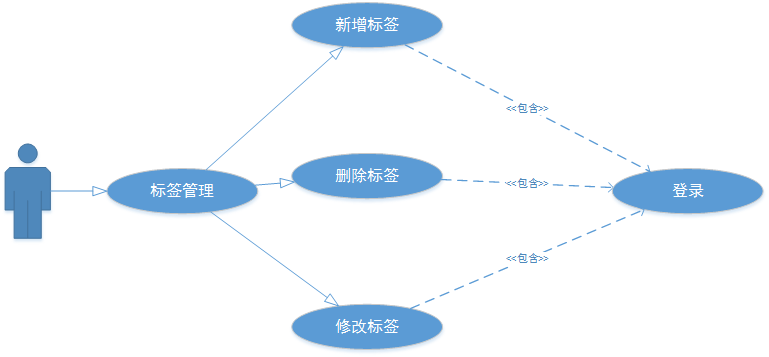


图3-20标签管理模块用例图

新建标签：用户在新建或者修改博客的时候可以点击标签模块，进行新建标签，丰富博客的内容。

删除标签：用户同样可以删除标签，剔除不合适的内容。

修改标签：同样用户也可以对原有的标签进行修改，使其更加语义明确和丰满。

## 3.3 非功能需求分析

### 3.3.1安全性

对用户的密码都进行加密的处理，避免明文传输，同时采用spring security作为权限管理框架，对用户的权限进行了方法级别的隔离，保证每个请求都是经过校验的，同时也保持了代码的简洁性。

### 3.3.2简约规范

在保证系统能够正常运行的时候，同时也保住了前后台代码的简洁性，保证后续重构和优化的便捷性。对用户的界面也采用响应式的规范来编写，界面美观。满足简约高效的规范。

### 3.3.3流畅高效

同时，本课题在保证用户能够正常运行的同时，也保证了性能的问题，采用elasticsearch进行全文搜索，相较于关系型数据库就会有很大的性能提升，保证了用户流程的操作感觉。同时也能准确的搜索到用户想要的。

### 3.3.4良好的用户体验

满足以上几点条件的同时，本课题也保证用户能够拥有良好的用户体验，无论是在视觉（前端）还是在性能（后台）都能够抓住用户的心。

## 3.4 开发和运行环境

这里对开发本课题的开发和运行环境做出简单的介绍，方便大家参考。

### 3.4.1 开发环境

设备：联系G510

CPU：Intel(R) Core(TM) 15-4210M CPU @ 2.60GHz 2.60 GHz

内存：12G

硬盘：512G

系统：Windows 8.1专业版

网络调试：Chrome 66.0.3359.139（正式版本）（64 位）

JDK：Java8

调试设备：IDEA 2018.1

### 3.4.2 运行环境

系统要求：没有要求

设备要求：Chrome55及以上

屏幕：无要求，可自动适配

硬盘：无要求

网络：wifi或者3G网络或者更快

# 4.系统设计

本课题遵循MVC设计模式和常用设计模式如策略模式和工厂模式设计理念，主要可以分为系统架构设计、功能模块设计、数据模型设计、数据模型设计、界面设计、系统框架设计。因此本章将会从这几个方面进行描述。

## 4.1 系统架构设计

系统的架构采用MVC的设计模式分为三层，主要为数据模型层、视图层和控制层。系统的数据集模型层主要有mysql、elasticsearch和mongodb提供支持，视图层采用web界面，同时适配移动端和网页端，控制层采用spring boot作为主体框架，采用java编写业务支撑。

数据模型层主要是为了支撑整个系统，是一个非常重要的部分，它作为数据库和java代码之间的连接桥梁，可以说是至关重要的。采用多个数据库来支持和维护该数据模型层，利用不同数据库的特性，比如关系型数据库的事务性和非关系型数据库的高效性等，发挥各个数据库的特长，为我们的系统带来非常不错的性能和功能的支持。

视图层主要是采用HTML5、CSS3和ES5以及thymeleaf来编写，同时采用bootstrap来适配移动端，给用户带来更好的体验。整个视图层包括用户的视觉上的效果，点击事件的处理和数据的渲染三个部分。

控制层是整个模块的核心，他连接了模型层和视图层，采用JSON的数据格式进行传输，便于移植和拓展。它负责将视图层传过来的请求分发到各个路由下，使用相应的方法进行解析和返回需要的数据。

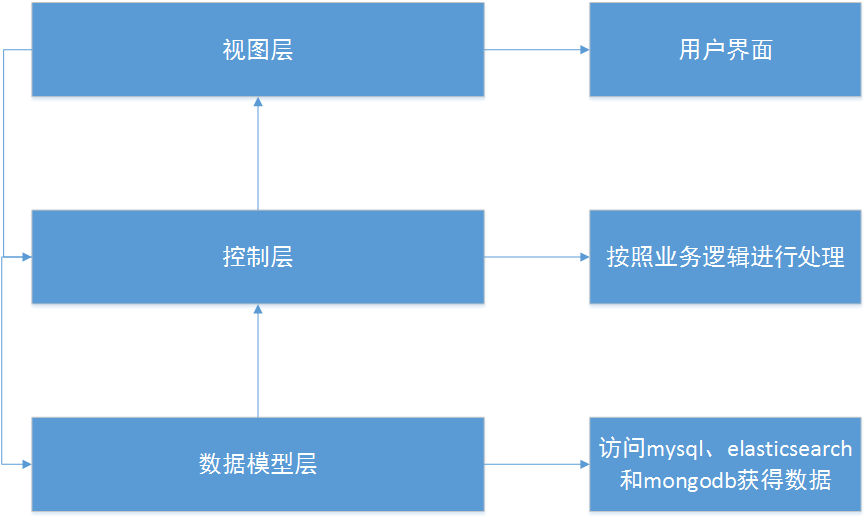


图4-1 系统结构图

## 4.2 功能模块设计

本系统的定位是一个供技术人员交流和分享的技术博客。通过该技术博客，方便技术人员记录自己工作和学习过程中的点滴，不断地进行技术的总结和积累，从而提升自己的综合能力，并通过博客这一平台，把自己的知识、经验、教训分享给大家，为志同道合者提供一个相互交流、共同学习的平台，促使更多的人共同进步。

本毕业设计中将主要实现用户模块、角色模块、权限模块、博客模块、评论模块、点赞模块、分类模块、标签模块、搜索模块等功能模块。会根据用户平常阅读的喜好，搜索的习惯，智能推荐给用户相应的优质内容。方便技术人员能够更加自由的书写博客，以及分享自己的学习心得，结识一些有相同爱好的极客。后期也可根据用户的需求做适当调整。

具体的功能如下：



图4-2 系统功能结构图

### 4.2.1用户管理模块：

用户可以进行登录、注册和搜索其他用户，超级管理员具有用户的增删改查的超级权限。同时本课题可以实现自动登录的功能，只要用户不注销，那么下次他可以免登陆，直接变成已经登录的状态。

### 4.2.2全文搜索模块：

此模块为本项目的一个亮点，采用大数据分析平台elasticsearch，将后台数据库中的博客信息，用户信息，标签信息。用户在全文搜索的时候，后台直接从elasticsearch中进行查询，增加了响应速度和响应精确性。比直接用sql语句中的like查询面积要广，而且还具有分词效果，匹配度高的显示在前面，低的显示在后面，真正达到全文搜索。全文覆盖的效果。

### 4.2.3博客管理模块：

此模块是本系统的一个重点。用户可以自由的对博客增删改查，同时可以对博客设置相应的标签。还可以对博客进行最新最热排序。同时系统还会对每一个博客进行阅读量的统计，让别的技术人员能够更加直观的感受到该博文的受欢迎程度，同时也给全文搜索给了一个指标。

### 4.2.4评论管理模块：

用户可以对博客进行评论，让更多的人看到自己的想法，能够与别人进行交流和互动。同时也能和博主有着思想的碰撞。

### 4.2.5点赞管理模块：

通评论模块一样，用户可以对自己喜欢的博文进行点赞，或者对自己已经不感兴趣的博文进行取消点赞。同时也给全文搜索提供一个硬性指标。

### 4.2.6权限设置模块：

用户只能对自己编写的博客才有删除的权限，对其他的无法进行删除操作。用户只能在自己拥有该权限的时候，才能进行相应的操作，这个通过SpringSecurity来进行管理。且权限粒度除了区分到方法级别上，还设置到了类级别，比如“/admin”和“/u”这两个路由下的资源必须要具有相应的角色才能够访问。

### 4.2.7个性推荐模块：

主要是靠后台进行数据筛选，比如必要在改用户是为登录的状态，而不是为匿名用户（未登录）的状态。具体要采集的数据主要为该用户自己创建的博客分类和具体属于的标签，还有就是用户电子和评论过的博客列表，并对这些博客列表进行筛选，选择具体属于的分类和标签。再去elasticsearch中进行查询该特征值的博客列表。

## 4.3 数据模型设计

根据本课题的需求，确定其数据模型如下所示：

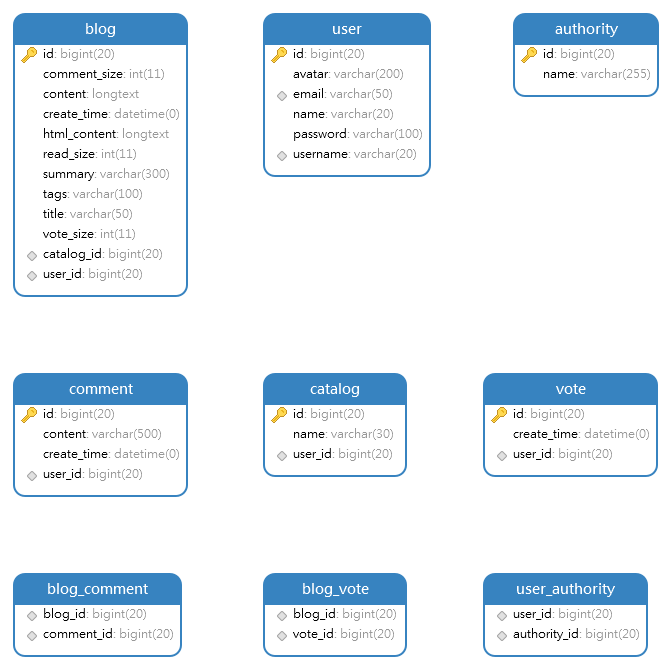


图4-3 数据模型图

### 4.3.1博客模型：

博客模型是用来存储和博客相关的一些属性，主要包括博客的编号、评论量、博客内容（markdown）、创建时间、网页内容、阅读量、摘要、标签、标题、点赞量、目录编号和用户编号。博客模型主要属性如下：

表4-1 博客模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 博客编号 |
| comment\_size | int | 评论量 |
| content | longtext | 博客内容 |
| create\_time | datetime | 创建时间 |
| html\_content | longtext | 网页内容 |

续表：4-1

|  |  |  |
| --- | --- | --- |
| read\_size | int | 阅读量 |
| summary | varchar | 摘要 |
| tags | varchar | 标签 |
| title | varchar | 标题 |
| vote\_size | int | 点赞量 |
| catalog\_id | bigint | 分类编号 |
| user\_id | bigint | 用户编号 |

### 4.3.2用户模型：

用户模型是用来存贮用户的一些基本信息，包括用户编号、头像地址、电子邮箱、用户名、密码。昵称等信息。用户模型主要属性如下：

表4-2 用户模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 用户编号 |
| avatar | varchar | 用户头像 |
| email | varchar | 用户邮箱 |
| name | varchar | 昵称 |
| password | varchar | 密码 |
| username | varchar | 用户名 |

### 4.3.3博客评论模型：

博客评论模型主要是为了存放博客的评论的信息，这里没有放在vote表中，而是单独抽离出来一张表格。主要包括博客编号和评论编号。博客评论模型主要属性如下：

表4-3 博客评论模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| blog\_id | bigint | 博客编号 |
| commetn\_id | bigint | 评论编号 |

### 4.3.4评论模型：

评论模型主要包括评论编号、评论内容、创建时间和用户编号。评论模型的主要属性如下：

表4-4 评论模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 评论编号 |
| content | varchar | 评论内容 |
| create\_time | datatime | 创建时间 |
| user\_id | bigint | 用户编号 |

### 4.3.5分类模型：

分类模型主要为了存放每个用户的具体的分类信息，方便用户按照分类进行管理博客。主要包括分类编号、分类名字和用户编号。分类模型主要属性如下：

表4-5 分类模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 分类编号 |
| name | varcahr | 分类名字 |
| user\_id | bigint | 用户编号 |

### 4.3.6博客点赞模型：

博客点赞模型主要为了存放博客的点赞信息，具体是那个用户进行的点赞操作。主要有博客编号和点赞编号。博客点赞模型主要属性如下：

表4-6 博客点赞模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| blog\_id | bigint | 博客编号 |
| vote\_id | bigint | 点赞编号 |

### 4.3.7点赞模型：

点赞模型主要包括点赞编号、创建时间和用户编号。点赞模型主要属性如下：

表4-7点赞模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 点赞编号 |
| create\_time | datetime | 创建时间 |
| user\_id | bigint | 用户编号 |

### 4.3.8权限模型：

权限模型主要包括权限编号和权限名字。权限模型的主要属性如下：

表4-8点赞模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| id | bigint | 权限编号 |
| name | varchar | 权限名字 |

### 4.3.9用户权限模型：

用户权限模型主要为了存放具体每个用户对应的权限信息。权限模型的主要属性如下：

表4-9权限模型表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 字段说明 |
| user\_id | bigint | 用户编号 |
| authority\_id | bigint | 权限编号 |

## 4.4 界面设计

### 4.4.1登录界面：

登录界面需要用户填写账号信息和密码，同时还可以点击记住我选择框。输入完毕后点击登录按钮即可。下面是登录界面的演示，此处用适配移动端形式展示：



图4-4 用户登录设计图

### 4.4.2注册界面：

用户注册界面需要用户填写账号和、邮箱信息、真实姓名和密码输入无误后点击提交按钮，即可注册成为本博客系统中的用户。下面是注册页面的展示，此处用适配移动端形式展示：



图4-5 用户登录设计图

### 4.4.3博客主页界面：

本课题博客主页面可以分为用户登录和未登录两种状态，其中区别主要是在右上角的按钮显示不同。

已经登录后的主页面如下图所示：

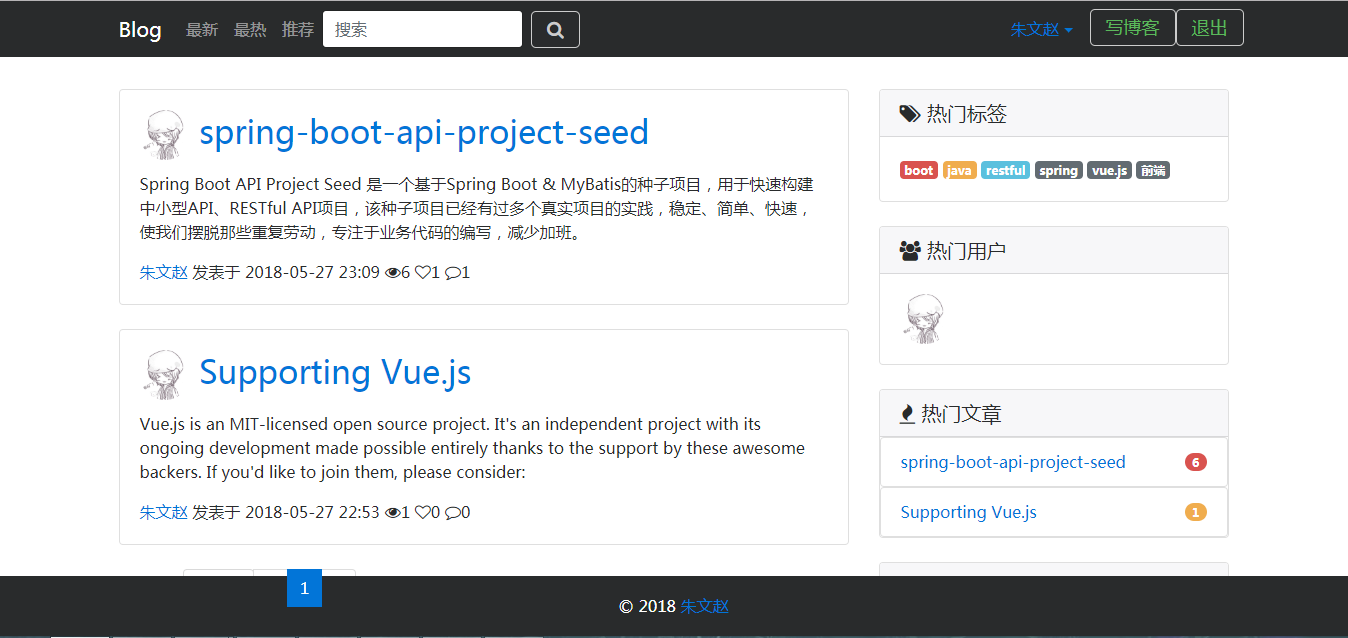


图4-6 用户已经登录主页面设计图

未登录的主页面如下图所示：

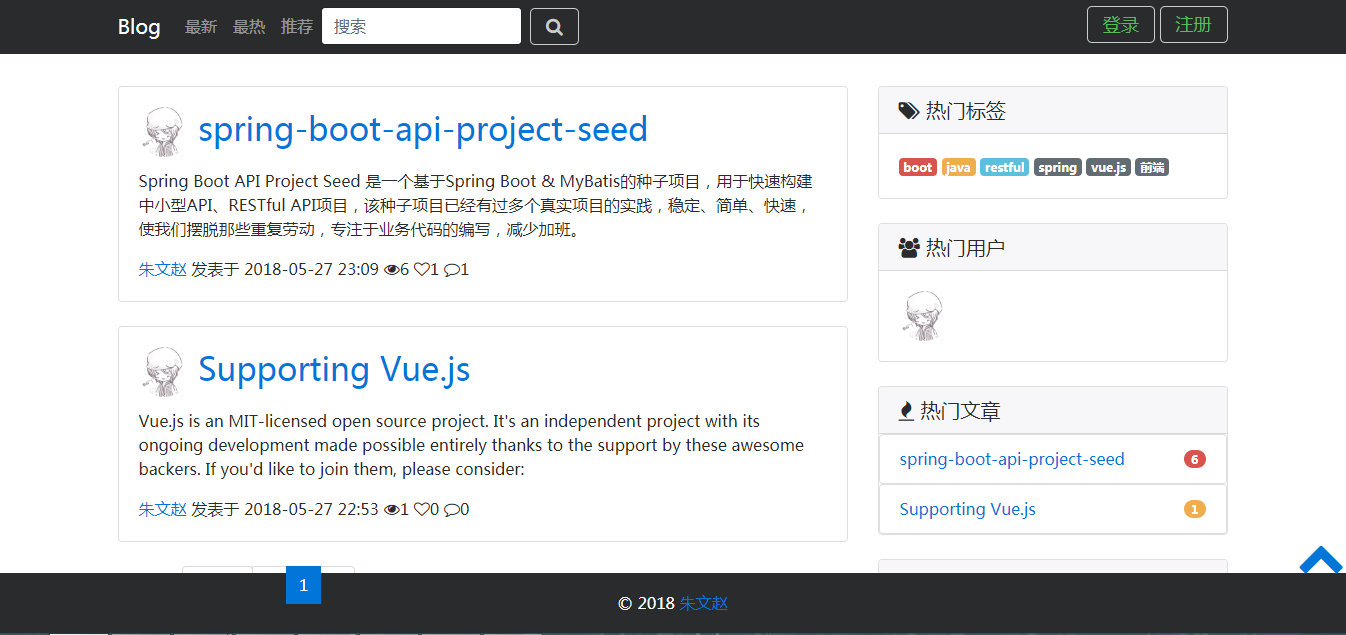


图4-7 用户未登录主页面设计图

其中头部主要内容是本系统名称、最新排序按钮、最热排序按钮、还有推荐排序按钮和搜索框。用户可以根据自己的需要添加相应的按钮进行查看相应的博客。

左下半部分为博客列表，显示当前操作下的博客排序列表。右下半部分分为热门标签、热门用户和热门文章三个部分，主要是为了统计相应的信息，使博客看起来更加的完整。

### 4.4.4创建博客界面：

用户登录后可以点击写博客按钮，然后进入到写博客页面，输入相应的博客标题、摘要、内容，然后输入相应的标签和选择博客的分类，点击创建按钮即可，界面如下图所示：

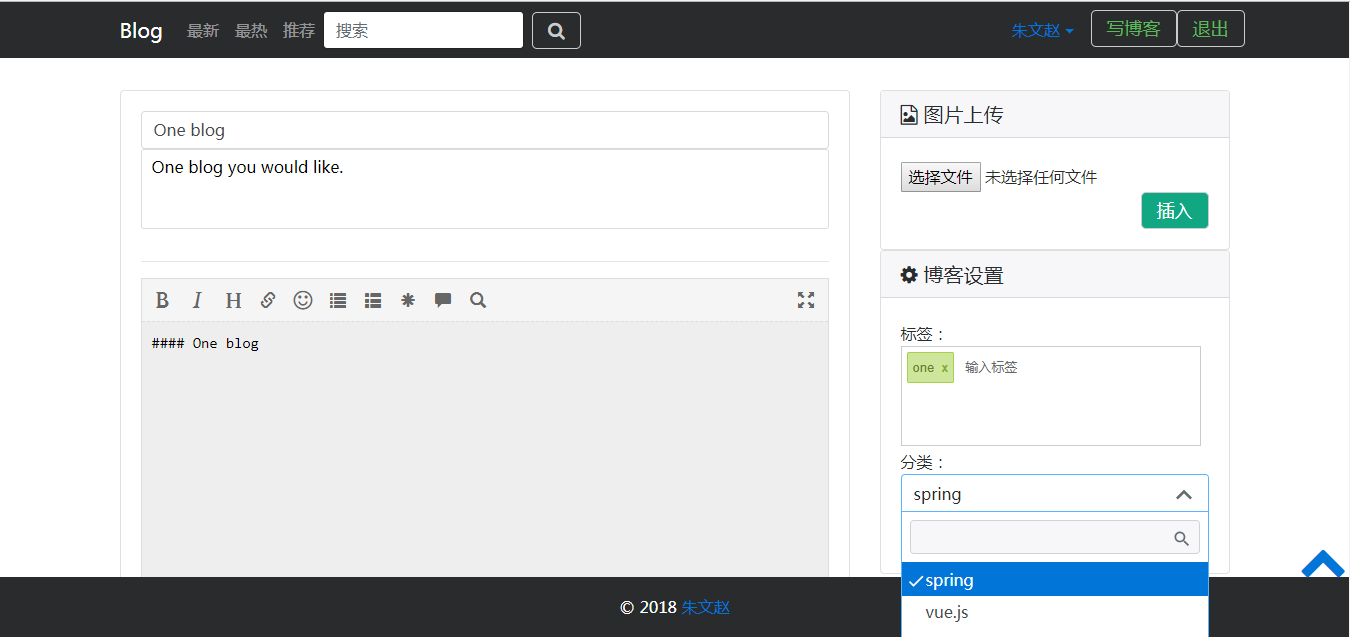


图4-8创建博客页面设计图

### 4.4.5查看博客界面：

用户点击博客列表中任意一条，即可进入到查看页面，这里又需要判断该博客的创建者是否为当前用户，若是则显示编辑按钮，若不是则不显示。下面是这两中情况的界面设计图：

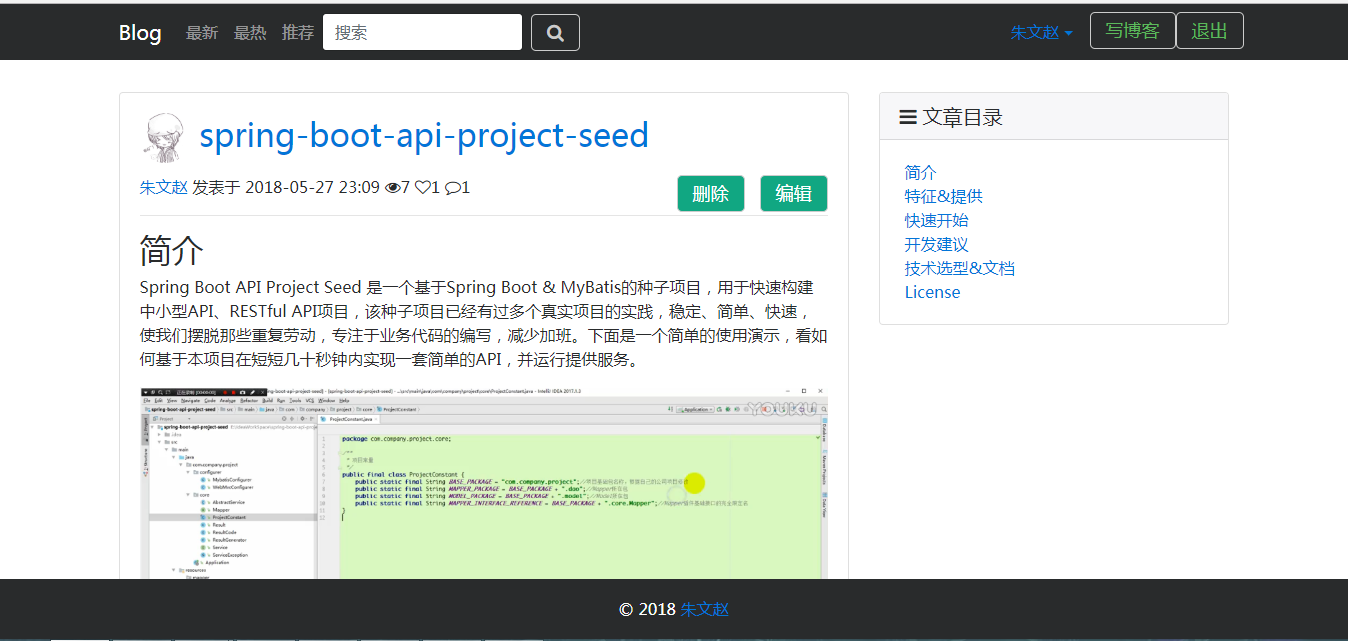


图4-9查看自己的博客页面设计图

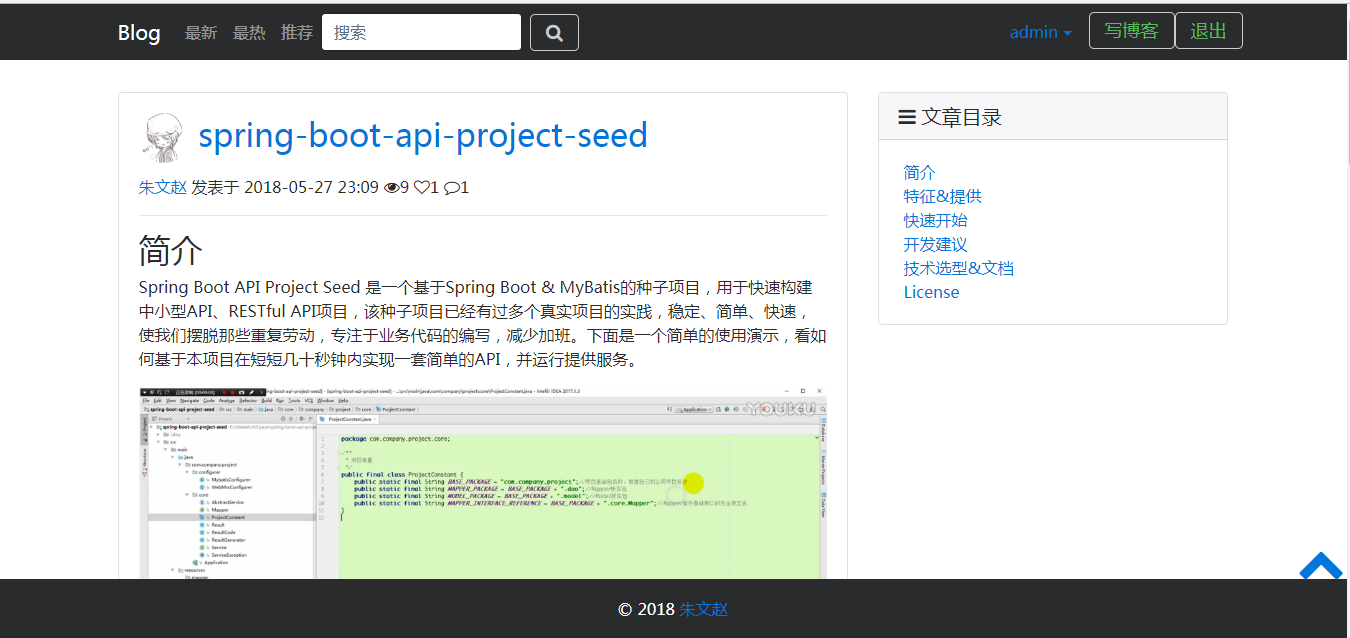


图4-10查看非自己的博客页面设计图

### 4.4.6点赞界面：

用户在点击查看一个博客列表的时候，可以 进行点赞操作。如下图所示：

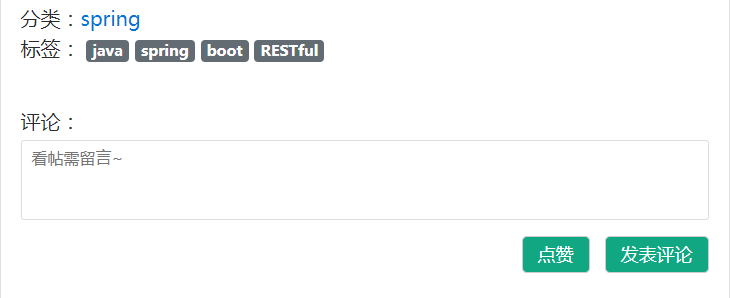


图4-11博客点赞设计图

### 4.4.7评论界面：

用户同样也可以对博客进行评论操作，如下图所示：

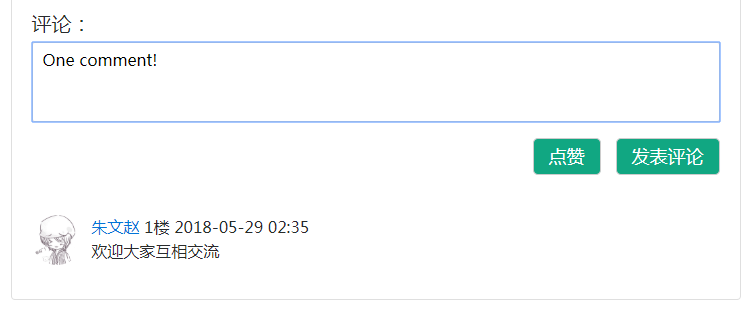


图4-12博客评论设计图

### 4.4.8用户主页界面：

用户点击个人主页按钮即可跳转到该页面，这里显示了用户的头像信息昵称邮箱地址还有分类各个分类信息，其次右半部分显示的则是自己所创建的博客列表，和主页相同可以进行最热、最新和关键词搜索。如下图所示：

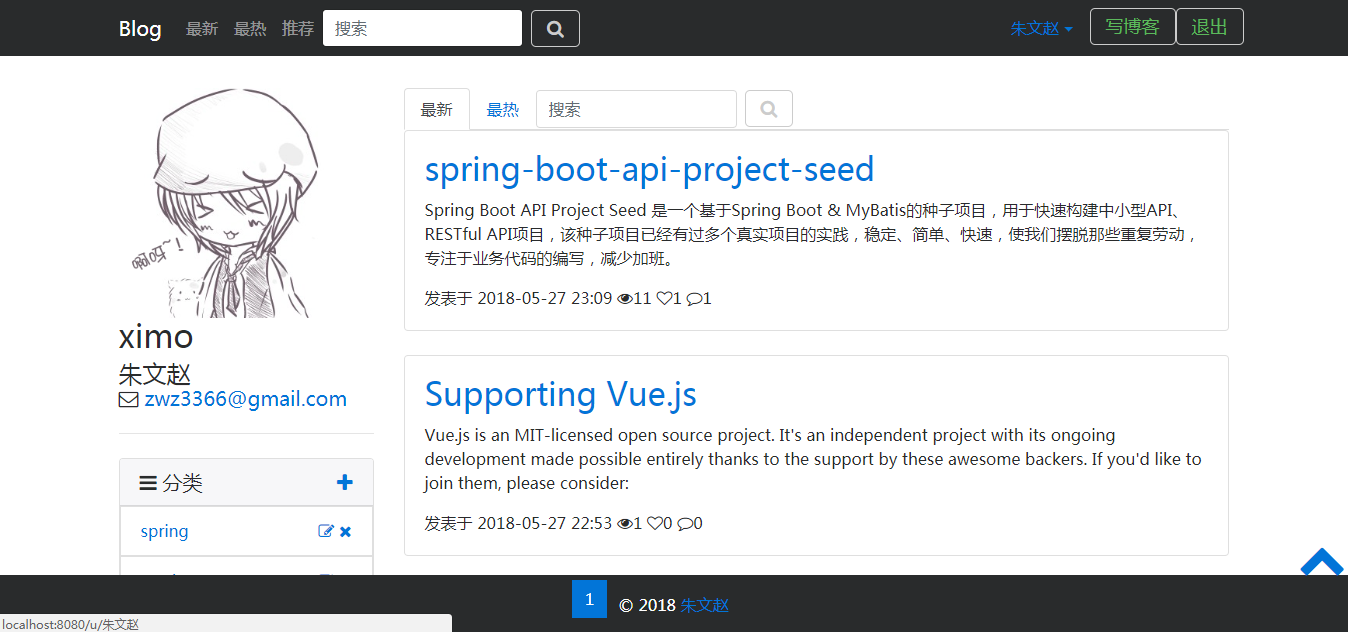


图4-13个人主页设计图

### 4.4.8个人设置界面：

用户点击个人设置按钮，即可跳转到该页面，该页面主要可以编辑自己的头像、邮箱、姓名和密码等信息。如下图所示：

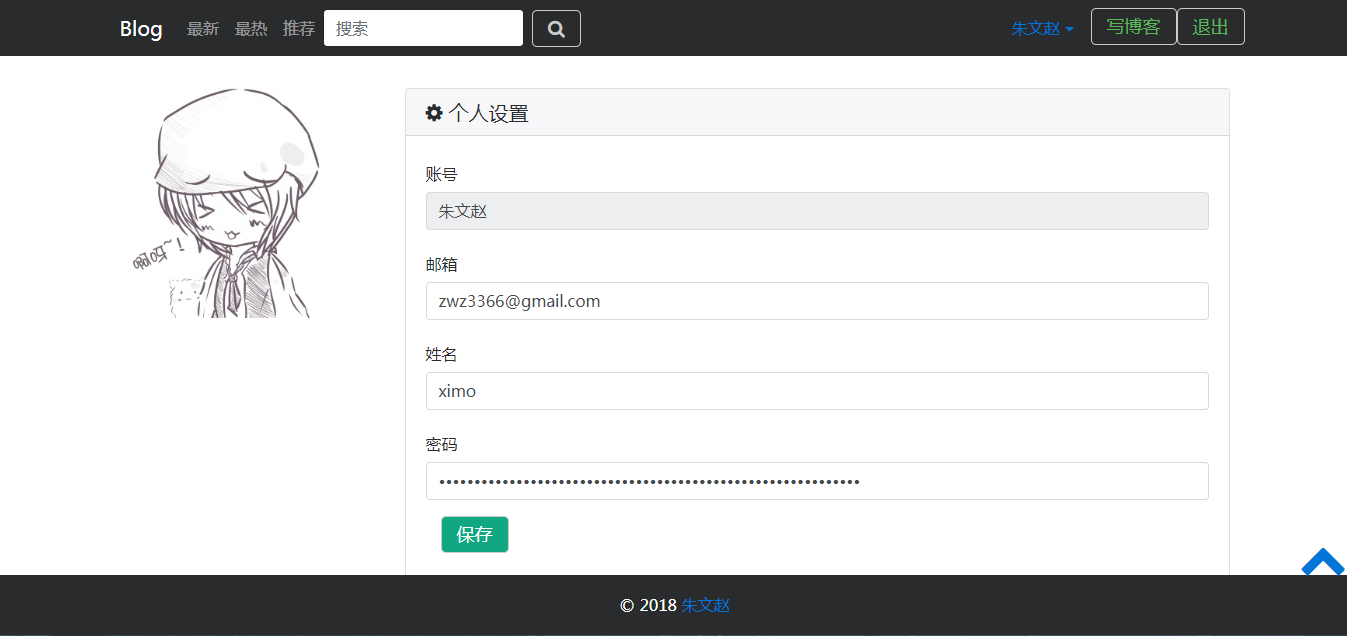


图4-14个人设置设计图

### 4.4.9头像变更切图界面：

用户点击自己的头像即可对自己的头像进行编辑，可进行左右移动。如下图所示：

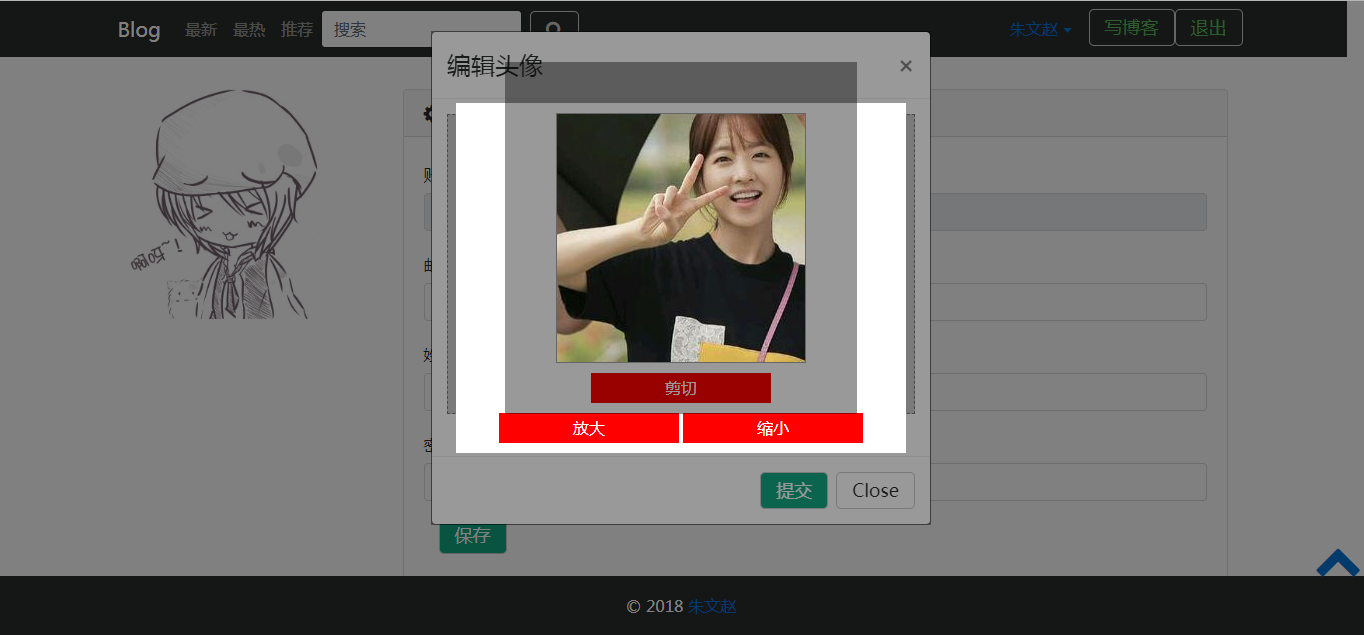


图4-14头像变更设计图

### 4.4.10分类管理界面：

用户点击个人主页，然后在下方即可看到对应的分类信息，点击添加即可添加分类，也可以选择编辑和删除分类的操作。如下图所示：



图4-15分类管理设计图

## 4.5 系统框架设计

本课题按照模型层、控制层和视图层来进行设计，通过这三个模块的相互协调来正常运作整个项目。

### 4.5.1模型层：

模型层在整个系统中属于最为底层的模块，他通过java中类的概念和数据库中的表一一对应，屏蔽了一些具体的细节，开发者只需要通过类与类之间的调用即可操作数据库中国的表。其中类和数据库表具体的对应关系如下所示：

1. Blog类对应数据库中blog表。
2. Catalog类对应数据库表中catalog表。
3. Commont类对应数据库表中comment表。
4. User类对应数据库表中的user表。
5. Vote类对应数据库表中的vote表。
6. EsBlog类对应elasticsearch中的blog索引（index）。
7. File类对应mongodb中的File文档（document）。

### 4.5.2控制层：

控制层主要负责前端的请求和数据的返回，作为连接视图层和模型的桥梁，可以说有着至关重要的地位。具体的控制类如下所示：

AdminController：管理员控制器，用例处理管理员相关逻辑。

BlogController：博客控制器，用例处理博客相关逻辑。

CatalogController：分类控制器，用例处理分类相关逻辑。

CommentController：评论控制器，用例处理评论相关逻辑。

MainController：主控制器，用来处理一些公共的页面跳转。

UserController用户控制器，用来处理用户相关逻辑。

UserspaceController：用户控件控制器，用来处理用户主页的控制器。

VoteController：点赞控制器：用来处理点赞的相关逻辑。

### 4.5.3视图层：

视图层主要为了显示控制层返回的数据，给用户一个美观的界面。本课题中的视图主要包括如下几个：

* + 1. admin

index.html主要用来显示管理员的一些操作的界面

* + 1. users

add.html主要用来为管理员进行添加用户的界面

edit.html主要用来为管理员进行编辑用户的界面

list.html主要用来显示所有的用户列表。

* + 1. userspace

avatar.html主要用来编辑头像

blog.html主要用来显示每个用户下的博客列表

blogedit.html主要用来标记自己的博客信息

catalog.html主要用来创建自己的分类信息

profile.html主要用来编辑自己的个人信息

u.html查看自己的所有信息

* + 1. fragments

foot.html公共部分用来处理整个系统的底部

header.html公共部分用来处理整个系统的头部

page.html公共分页的模块

* + 1. other

index.html主要用来处理博客主页页面

login.html主要用来处理登录的页面

register.html主要用来处理注册的页面

search.html主要用来处理搜索的页面

# 5. 系统实现

本章节主要对实现细节进行概述，主要从三个方面进行概述

## 5.1 数据获取层

本节主要对开发中的数据传输进行描述，主要可以归纳为包括用户管理、博客管理、点赞管理，评论管理，分类管理等模块。

### 5.1.1 用户管理模块

1. 登录接口：

由于我们采用spring security作为权限框架，所以我们的登录将会变得很简单，我们只需要调用框架中的“/login”接口即可。

请求地址：/login

请求方式：POST

接口参数：该接口直接采用spring security中的login接口进行操作，所以我们没有不需要管具体实现的细节，只需要传输参数即可。

表5-1 用户登录请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | “username”： “root” |
| password | 密码 | 必选 | “password”: 123456 |

接口返回：验证用户信息是否正确，正确后先往cookie中存放用户数据，然后跳转到用户刚刚请求的地址。

1. 注册接口：

方法签名如下：

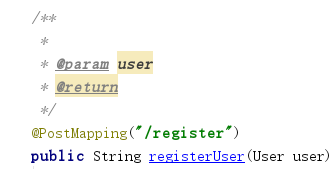


图5-1 用户注册方法签名

用户在创建账号的时候，将会调用这个接口。

请求地址：/register

请求方式：POST

接口参数：该接口传入用户的的基本信息，进行创建用户。

表5-2 注册请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | “username”:“root” |
| passwrod | 密码 | 必选 | “password”:123456 |
| email | 邮箱 | 必选 | “email”：“zwz3366@gmail.com” |
| name | 姓名 | 必选 | “name”：“zwz” |

接口返回，采用model的形式进行返回，如下所示：

表5-3注册方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| userId | 新增用户的编号 | userId:3 |

### 5.1.2 博客管理模块

博客管理模块是本课题的重点，主要有最新、最热、推荐博客接口，以及查看单个博客信息，创建博客和修改博客等接口。

1）全文搜索博客

最新、最热、关键字搜索和推荐博客，是本课题的重中之重，它结合了elasticsearch来进行全文搜索，比关系型数据库有明显的优势。同时该方法除了返回博客列表外还返回一些聚和信息，比如最活跃的用户，最热门的标签等信息，其中方法签名如下：



图5-2全文搜索博客方法签名

请求地址：/blogs

请求方式：GET

请求参数：该接口用来查看博客信息，用户可以输入关键字进行搜索，也可以直接点击最新和最热或者推荐链接进行操作。

表5-4全文搜索博客方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| order | 排序字段 | 可选 | “order”:“hot” |
| keyword | 关键字 | 可选 | “password”:123456 |
| async | 是否异步请求 | 可选 | “async”：true |
| pageIndex | 当前页 | 可选默认为0 | “pageIndex”：1 |
| pageSize | 一页显示多少条 | 可选，默认为10 | “pageSize”：10 |

请求结果：返回为携带博客信息的model，渲染好页面后进行返回。Model中主要包含以下信息，如下所示：

表5-5全文搜索博客方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| order | 排序字段 | “order”:“hot” |
| keyword | 关键字 | “password”:123456 |
| page | 分页信息 | “page”：“...” |
| blogList | 总条数 | “blogList”：[] |

2) 查看单个博客

用户可以直接点击一个博客进行查看，后台会根据是否是自己的博客进行相应的返回，方法签名如下：

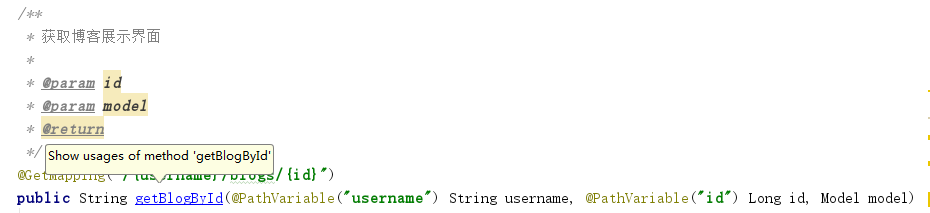


图5-3查看单个博客方法签名

请求地址：/u/{username}/blogs/{id}

请求方式：GET

请求参数：通过RESTful url传输用户名和博客id，具体描述如下：

表5-6查看单个博客方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | /u/zwz/blogs/1 |
| id | 博客编号 | 必选 | /u/zwz/blogs/1 |

接口返回：该博客的具体信息页面，具体采用model返回。如下所示：

表5-7查看单个博客方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| isBlogOwner | 是否是博客所有者 | “isBlogOwner”:true |
| blogModel | 关键字 | “blogModel”:“...” |
| currentVote | 分页信息 | “currentVote”：“...” |

3) 创建/修改博客:

由于创建和修改博客的区别只有博客编号的有无，所以将二者放到一起来进行传输。方法签名如下：



图5-4创建/修改博客方法签名

请求地址：/u/{username}/blogs/edit

请求方式：POST

请求参数：第一个参数为用户名，采用@ PathVariable注解绑定，第二个为博客的整个对象，为一个JSON结构。具体如下所示：

表5-8创建或者修改博客方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | /u/zwz/blogs/edit |
| blog | 博客 | 必选 | “blog”:“...” |

接口返回，采用JSON的格式进行返回，如下所示：

表5-9创建或者修改博客方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| id | 博客编号 | “id”:1 |
| title | 标题 | “title”:“One title” |
| summary | 摘要 | “summary”：“one summary” |
| content | 内容 | “content”:“one content” |
| htmlContent | 网页内容 | “htmlContent”：“<div>...<div>” |
| user | 用户信息 | “user”:“...” |
| createTime | 创建时间 | “createTime”:“20180510” |
| readSize | 阅读量 | “readSize”：10 |
| commentSize | 评论量 | “commentSize”：2 |
| voteSize | 点赞量 | “voteSize”：7 |
| comments | 评论列表 | “comments”：[] |
| votes | 点赞列表 | “votes”:[] |
| catalog | 分类 | “catalog”:[] |
| tags | 标签 | “tags”:“one,two” |
| success | 成功与否 | “success”:true |
| message | 消息 | “message”:“处理成功” |

### 5.1.3 点赞管理模块

1) 点赞

点赞用户和博主的一个交流方式，表达了用户对博主的博客的赞赏。其方法签名如下：

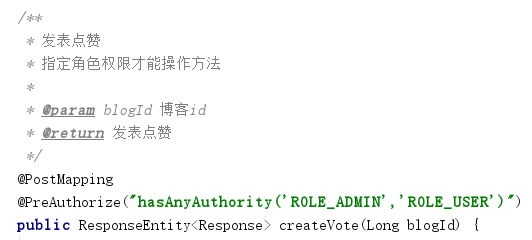


图5-5点赞方法签名

请求地址：/votes

请求方式：POST

请求参数：该接口是进行点赞，为一个固有的动作，所以只需要传一个博客编号。具体如下所示：

表5-10点赞操作请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| blogId | 博客编号 | 必选 | “blogId”:2 |

接口返回，采用JSON的格式进行返回，如下所示：

表5-11点赞操作返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| redirectUrl | 重定向url | “redirectUrl”:“/u/zwz/blogs/1” |
| success | 成功与否 | “success”:true |
| message | 消息 | “message”:“处理成功” |

2) 取消点赞

取消点赞表达了用户对该博客的不满意的心情，将会给博主敲一个警钟。其方法签名如下：

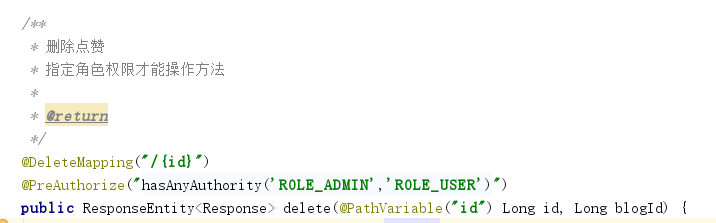


图5-6取消点赞方法签名

请求地址：/votes

请求方式：DELETE

请求参数：该接口是取消点赞，需要传一个博客编号和点赞编号。具体如下所示：

表5-12取消点赞方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| blogId | 博客编号 | 必选 | “blogId”:2 |

接口返回，返回类型为JONS，如下所示：

表5-13取消点赞法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| success | 成功与否 | “success”:true |
| message | 消息 | “message”:“处理成功” |

### 5.1.4评论管理模块

1）获取评论列表

当用户查看博客单个博客的时候，会异步加载博客的评论信息，后者在用户点赞或者取消点赞的时候，也会去异步加载改博客的评论信息。其方法签名如下：

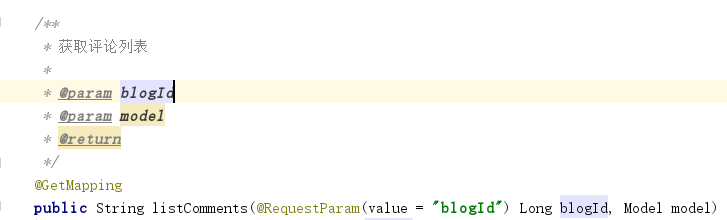


图5-7获得博客评论列表方法签名

请求地址：/comments

请求方式：GET

请求参数：该接口为获得某一个博客下的评论列表，所以只需要传输一个博客编号就行。具体如下所示：

表5-14获得评论列表方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| blogId | 博客编号 | 必选 | “blogId”:2 |

接口返回，采用model返回，其中model具体携带的信息如下所示：

表5-15获得评论列表方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| commentOwner | 是否是评论所有者 | “commentOwner”:true |
| comments | 评论列表 | “comments”:[...] |

2）发表评论

用户对于自己喜欢的博客可以进行发表评论的操作。其方法签名如下：

图5-8创建评论方法签名

请求地址：/comments

请求方式：POST

请求参数：该接口为创建评论，所以需要传入博客编号和评论内容就可以。具体如下所示：

表5-16创建评论请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| blogId | 博客编号 | 必选 | “blogId”:2 |
| commentContent | 评论内容 | 必选 | “commentContent”：“...” |

接口返回，采用JSON返回，其中JSON具体携带的信息如下所示：

表5-17创建评论法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| message | 返回信息 | “message”：“成功” |
| success | 是否成功 | “success”：true |

3) 删除评论

用户可以自己所创建的博客评论进行删除，方法签名如下：

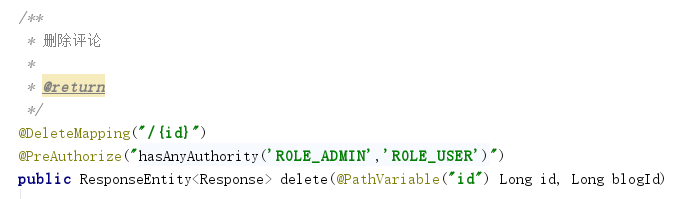


图5-9删除评论方法签名

请求地址：/comments

请求方式：DELETE

请求参数：该接口为删除评论，所以需要传入博客编号和评论编号。具体如下所示：

表5-18删除评论方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| blogId | 博客编号 | 必选 | “blogId”:2 |
| id | 评论编号 | 必选 | “id”：3 |

接口返回，采用JSON返回，其中JSON具体携带的信息如下所示：

表5-19删除评论方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| message | 返回信息 | “message”：“成功” |
| success | 是否成功 | “success”：true |

### 5.1.5分类管理模块

1）获取分类列表

该接口为了获得博客的分类列表，方法签名如下：

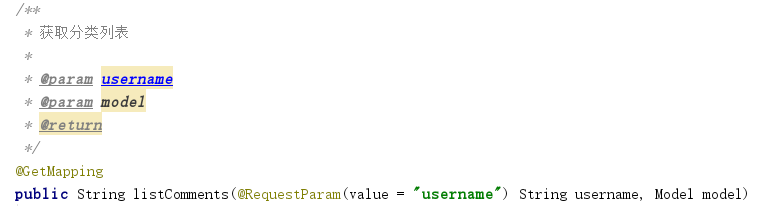


图5-10获得分类列表方法签名

请求地址：/catalogs

请求方式：GET

请求参数：该接口为获得分类列表，所以需要传入用户名。具体如下所示：

表5-20获得分类列表方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | “username”:“zwz” |

接口返回，采用Model返回，其中Model具体携带的信息如下所示：

表5-21获得分类列表方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| isCatalogsOwner | 是否是分类的所有者 | “isCatalogsOwner”：true |
| catalogs | 分类 | “catalogs”：[] |

2)创建分类：

用户可以创建分类，丰富博客的管理模块。方法签名如下所示：

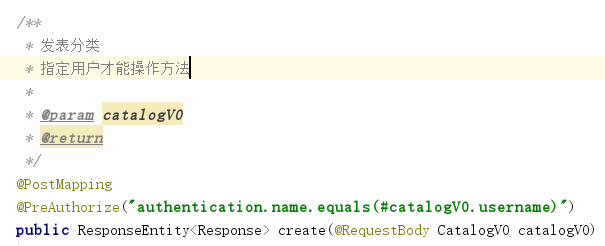


图5-11创建分类方法签名

请求地址：/catalogs

请求方式：POST

请求参数：该接口为创建分类，需要传入用户名和分类信息。具体如下所示：

表5-22创建分类方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | “username”:“zwz” |
| catalog | 分类信息 | 必选 | “catalogs”:“...” |

接口返回，采用JSON返回，其中JSON具体携带的信息如下所示：

表5-23创建分类方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| message | 提示信息 | “message”：“成功” |
| success | 成功与否 | “success”：true |

3）删除分类

用户可以删除自己所定义的博客类型标签，具体方法签名如下：



图5-12删除分类方法签名

请求地址：/catalogs

请求方式：DELETE

请求参数：该接口为删除分类，需要传入分类编号和用户名信息。具体如下所示：

表5-24删除分类方法请求参数

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 意义 | 是否可选 | 案例 |
| username | 用户名 | 必选 | “username”:“zwz” |
| id | 分类编号 | 必选 | “id”:1 |

接口返回，采用JSON返回，其中JSON具体携带的信息如下所示：

表5-25删除分类方法返回参数

|  |  |  |
| --- | --- | --- |
| 字段 | 意义 | 案例 |
| message | 提示信息 | “message”：“成功” |
| success | 成功与否 | “success”：true |

## 5.2 数据构建层

这一节，将具体讲述各个模块的核心代码，将数据拼装的一个过程细化描述。

### 5.2.1 用户管理模块

### 5.2.2 博客管理模块

### 5.2.3 分类管理模块

### 5.2.4 点赞管理模块

### 5.2.5 评论管理模块

### 5.2.6 权限管理模块

## 5.3 数据展示层

# 6. 软件测试

## 6.1 测试环境

## 6.2 测试方法

## 6.3 测试用例设计

## 6.4 测试结果

# 7. 总结和展望

## 7.1 毕业设计工作总结

## 7.2 展望

# 参考文献

# 致谢