武汉纺织大学

**java博客系统**

**院系：数学与人工智能学院**

**专业：软件工程**

**学号：1804230912**

**姓名：伦小颖**

**日期：2021年6月**

**摘要**

博客，仅[音译](https://baike.baidu.com/item/%E9%9F%B3%E8%AF%91" \t "_blank)，英文名为[Blogger](https://baike.baidu.com/item/Blogger)，为Web Log的混成词。它的正式名称为网络日记；又[音译](https://baike.baidu.com/item/%E9%9F%B3%E8%AF%91" \t "_blank)为[部落格](https://baike.baidu.com/item/%E9%83%A8%E8%90%BD%E6%A0%BC)或部落阁等，是使用特定的软件，在网络上出版、发表和张贴个人文章的人，或者是一种通常由个人管理、不定期张贴新的文章的[网站](https://baike.baidu.com/item/%E7%BD%91%E7%AB%99/155722)。博客上的文章通常以[网页](https://baike.baidu.com/item/%E7%BD%91%E9%A1%B5/99347)形式出现，并根据张贴时间，以倒序排列。通常具备[RSS](https://baike.baidu.com/item/RSS/24470)订阅功能。

[Blog](https://baike.baidu.com/item/Blog/70149)就是以网络作为载体，简易迅速便捷地发布自己的心得，及时有效轻松地与他人进行交流，再集丰富多彩的个性化展示于一体的综合性平台。不同的博客可能使用不同的编码，所以相互之间也不一定兼容。而且，很多博客都提供丰富多彩的模板或其他个性化方案，这使得不同的博客各具特色。

博客是继MSN、BBS、ICQ之后出现的第4种网络交流方式，现已受到大家的欢迎，是网络时代的个人“[读者文摘](https://baike.baidu.com/item/%E8%AF%BB%E8%80%85%E6%96%87%E6%91%98/1588509)”，是以超级链接为入口的网络日记，它代表着新的生活、工作和学习方式。许多博客专注在特定的课题上提供评论或[新闻](https://baike.baidu.com/item/%E6%96%B0%E9%97%BB" \t "_blank)，其他则被作为个人性的[日记](https://baike.baidu.com/item/%E6%97%A5%E8%AE%B0)。一个典型的博客结合了[文字](https://baike.baidu.com/item/%E6%96%87%E5%AD%97)、图像、其他博客或网站的[链接](https://baike.baidu.com/item/%E9%93%BE%E6%8E%A5)及其它与主题相关的[媒体](https://baike.baidu.com/item/%E5%AA%92%E4%BD%93)，能够让读者以[互动](https://baike.baidu.com/item/%E4%BA%92%E5%8A%A8/10073145)的方式留下意见，是许多博客的重要要素。大部分的博客内容以文字为主，但仍有一些博客专注在艺术、摄影、视频、音乐、[播客](https://baike.baidu.com/item/%E6%92%AD%E5%AE%A2)等各种主题。博客是社会媒体网络的一部分。比较著名的有[新浪](https://baike.baidu.com/item/%E6%96%B0%E6%B5%AA/125692)等博客。

本文基于博客，详细的介绍了如何利用java技术，采用面向对象的分析和设计方法，对该平台进行详细的需求分析和系统设计。使用springboot开发，前端使用Bootstrap。支持移动端自适应，配有完备的前台和后台管理功能。

关键词：Java； Springboot； Apache Shiro； 阿里云OSS； kaptcha

**ABSTRACT**

Blog, transliterated only, is a hybrid word of Web log. Its official name is network diary; It is also transliterated as blog or tribal Pavilion. It is a person who publishes, publishes and posts personal articles on the Internet by using specific software, or a website which is usually managed by individuals and posts new articles irregularly. Blog articles usually appear in the form of web pages and are arranged in reverse order according to the time of posting. It usually has RSS subscription function.

Blog is a comprehensive platform which takes the network as the carrier, simply, quickly and easily releases their own experience, timely, effectively and easily communicates with others, and then integrates rich and colorful personalized display in one. Different blogs may use different codes, so they are not necessarily compatible with each other. Moreover, many blogs provide colorful templates or other personalized solutions, which makes different blogs have their own characteristics.

Blog is the fourth way of network communication after MSN, BBS and ICQ, which has been welcomed by everyone. It is a personal "reader's Digest" in the network era and a network diary with hyperlinks as the entrance. It represents a new way of life, work and study. Many blogs focus on providing comments or news on specific topics, while others are used as personal diaries. A typical blog combines words, images, links of other blogs or websites and other media related to the theme, which can let readers leave their opinions in an interactive way. It is an important element of many blogs. Most of the blog content is mainly written, but there are still some blogs focusing on art, photography, video, music, podcasting and other topics. Blogs are part of the social media network. More famous are Sina and other blogs.

Based on blog, this paper introduces in detail how to use Java technology, using object-oriented analysis and design methods, to carry out detailed requirements analysis and system design for the platform. Using spring boot development, the front-end uses bootstrap. It supports mobile terminal self adaptation and is equipped with complete foreground and background management functions.

Keyword: Java； Springboot； Apache Shiro； Alibaba cloud OSS；kaptcha

目录

[1绪论 5](#_Toc77122569)

[1.1 研究背景 5](#_Toc77122570)

[1.2 研究目的 5](#_Toc77122571)

[1.3 研究意义 6](#_Toc77122572)

[1.4功能简介 7](#_Toc77122573)

[2 相关技术综述 8](#_Toc77122574)

[2.1 java 8](#_Toc77122575)

[2.2 Spring Boot 9](#_Toc77122576)

[2.3 Apache Shiro 10](#_Toc77122577)

[2.4 Bootstrap 13](#_Toc77122578)

[2.5 Websocket 14](#_Toc77122579)

[2.6 Freemarker 15](#_Toc77122580)

[2.6.2 性能特点 16](#_Toc77122581)

[2.7 开发环境介绍 18](#_Toc77122582)

[3 需求分析 21](#_Toc77122583)

[4．实现 34](#_Toc77122584)

[5.结束语 39](#_Toc77122585)

[附录 41](#_Toc77122586)

[附录1：登录相关代码 41](#_Toc77122586)

[附录2：验证码代码 45](#_Toc77122586)

[附录3：申请友链、评论、点赞等代码 46](#_Toc77122586)

[附录4：保存文章浏览记录代码 50](#_Toc77122586)

[附录5：单条发送和批量发送代码 51](#_Toc77122586)

## 1绪论

### 1.1 研究背景

知识改变命运，要想有所成就，我们必须拥有丰富的知识体系。从古至今，我们的知识都是从学习中得来的。古时的人们都是从书籍中学习知识，现如今随着互联网信息技术的迅猛发展，互联网已成为了一个人们信息交流的平台。人们查阅资料的途径已经不在单一，人们不仅可以将自己觉得有用的信息发布到网页上，也可以在网上查询各种有用的信息。在当今这个地球村的时代，除了传统的查阅信息的方法外，互联网上还拥有优厚而丰富的资源，我们可以通过互联网来搜索各种自己需要的信息，从中学习并丰富自己的知识体系，从而达到提高自己解决生活中各种问题的能力。

博客，是使用特定的软件，在网络上出版、发表和张贴个人文章的人，或者是一种通常由个人管理、不定期张贴新的文章的网站。它是一种网络交流的方式，现已受到大家的欢迎，是网络时代的个人“读者文摘”，是以超级链接为入口的网络日记，他是代表着新的生活、工作和学习的方式。

许多博客专注在特定的课题上提供评论或新闻，其他则被作为个人性的日记。一个典型的博客结合了文字、图像、其他博客或网站的连接及其他的与主题相关的媒体，能够让读者以互动的方式留下个人见解，是许多博客的重要要素。大部分的博客内容以文字为主，但仍有一些博客专注在艺术、摄影、视频、音乐等各种主题。

### 1.2 研究目的

根据目前人们对于博客使用的调查，针对于现在的流行的博客软件，大体可以将这些分为四种：

1、[托管博客](https://baike.baidu.com/item/%E6%89%98%E7%AE%A1%E5%8D%9A%E5%AE%A2" \t "_blank)：无须自己注册域名、租用空间和编制网页，只要去免费注册申请即可拥有自己的Blog空间，是最“多快好省”的方式。

2、自建独立网站的Blogger：有自己的域名、空间和页面风格，需要一定的条件。（例如自己需要会网页制作，需要懂得网络知识，当然，自己域名的博客更自由，有最大限度的管理权限。）

3、附属Blogger：将自己的Blog作为某一个网站的一部分(如一个栏目、一个频道或者一个地址)。这三类之间可以演变，甚至可以兼得，一人拥有多种博客网站。

4、独立博客：[独立博客](https://baike.baidu.com/item/%E7%8B%AC%E7%AB%8B%E5%8D%9A%E5%AE%A2" \t "_blank)一般指在采用独立域名和网络主机的博客，既在空间、域名和内容上相对独立的博客。独立博客相当于一个独立的网站，而且不属于任何其他网站。相对于BSP下的博客，独立博客更自由、灵活，不受限制。

博客(Blogger)的出现,使我们在互联网世界,第一次有了知识积累和文化指向。使人类由粗放的数字化生存,过渡为个人化的精确的目录式生存。博客们将工作、生活和学习融为一体,通过博客日志(Blog),将日常的思想精华及时记录并发布,萃取并链接全球最有价值、最相关、最有意思的信息与资源。在其形成的博客媒介下,更多的知识工作者能够零距离、零壁垒地汲取这些最鲜活的思想。博客传播的出现,标志着以“信息共享”为特征的第一代门户之后,追求“思想共享”为特征的第二代门户正在浮现,互联网开始真正凸现无穷的知识价值。博客传播是网络传播领域出现的个性鲜明的传播现象,其出现改变了网络传播的秩序,重新划分了网络传播的界限。博客的出现还打破了长久以来传统媒体对媒介所传播内容的控制,使得由美国[社会心理学家](https://baike.baidu.com/item/%E7%A4%BE%E4%BC%9A%E5%BF%83%E7%90%86%E5%AD%A6%E5%AE%B6/15671256" \t "_blank)、传播学四大先驱之一的卢因1947年率先提出的“把关人”理论受到强烈冲击,以至于更有甚者宣称,“守门人”时代结束。一个理论的建立,在某种意义上讲是现实实践的结论升华,它反过来必然要指导实践活动。一个理论的结束,对于现实生活是具有重大意义的。博客技术下发展起来的博客群体、博客媒介和博客传播,是否将重新定义互联网的界限。

### 1.3 研究意义

博客是互联网2.0下的产物,它的出现对以往的传播模式带来了很大冲击,它对话语权的配置和重构,使普通民众拥有了自由发布信息的权利。博客传播的出现,标志着在以“信息共享”为特征的第一代门户网站之后,追求“思想共享”为特征的第二代门户——博客托管网站正在蜂拥而起。 博客编辑,是指从事互联网博客托管网站内容建设的人员。博客编辑由于其编辑对象——博客的特殊性,也就随之具有了与传统媒体编辑大不相同的新特点和新职能。博客编辑对编辑对象的形式和内容上的编辑干预度降到了最低,在运用“把关”职能时,也只充当了“信息过滤者”。博客编辑的工作更多地集中在对博客用户的人性化管理,对博客信息资源的科学整合,以及发掘优秀博客文章和博客作者,以促使博客健康发展、努力提升品位上面。随着博客的不断发展,它其中的负面问题也不断凸显出来,如博客侵权事件不断发生,博客中色情、暴力内容屡见不鲜,所以,维护博客健康、有序发展的重担就落在了博客编辑的肩上。如何巧妙地通过“把关”、整合和去粗取精等编辑手段引导博客使用者接纳优秀博客文化、摒弃糟泊,是每个博客编辑应该考虑的问题。在工作中博客编辑会遇到一些问题,例如,该如何平衡维护“言论自由权”。

博客是以自由、开放和共享为文化特征,通过图文音象等表现形式，围绕个人网络存在的五大功能，提供存取读写、组织沟通、评价交换等服务的一种社会化个人服务模式。它并不是纯粹的技术创新,但确是一种逐渐演变的网络应用,一种形式的变化。从某种意义上说,也是一种计算机源代码开放,它像一部傻瓜照相机一样,让所有使用者都成了程序员,这就大大消解了网络与人的距离,也消解了人与人的距离。博客的全民性,让它的传播方式成为所有人对所有人的传播。然而,博客并不如表面般繁荣,当博客毫不掩饰地在大众面前喧闹的时候,接踵而来的问题使得博客乱了方寸。 本文以非独立博客网站为研究对象,从传者、受者、内容、博客服务平台以及互动反馈等几个方面对博客进行分析,并从中得出了以下结论:博客参与者的盲目性导致了博客行为过程中的迷茫与厌倦;由于进入的零门槛和缺少监管,彻底颠覆互联网既有模式的博客,变成了新的信息垃圾场。博客正遭受低俗肤浅成风、网络侵权等因素的困扰；博客平台在服务、管理以及价值导向上忽略了用户体验；博客间的互动也呈现两个极端的尴尬局面。从总体上说，我国博客正处在一个混乱期,隐藏在虚热表面下的是种种的危机。

### 1.4功能简介

* 多种编辑器：支持wangEditor和Markdown两种富文本编辑器，可以自行选择
* 自动申请友情链接：在线申请友情链接，无需站长手动配置，只需申请方添加完站长的连接后自行申请即可
* 百度推送：支持百度推送功能，加速百度搜索引擎收录博文
* 评论系统：自研的评论系统，支持显示用户地址、浏览器和os信息，后台可审核评论、开启匿名评论、回复和邮件通知评论
* 权限管理：后台配备完善的权限管理
* 完善的 SEO 方案：自带robots、sitemap等seo模板，实现自动生成robots和sitemap
* 实时通讯：管理员可向在线的用户发送实时消息
* 系统配置支持快速配置：可通过后台手动修改诸如域名信息、SEO优化、赞赏码、七牛云以及更新维护通知等
* 多种文件存储：集成七牛云、阿里云OSS，实现文件云存储，同时支持本地文件存储
* 文章搬运工：集成blog-hunter实现“文章搬运工”功能，支持一键同步imooc、csdn、iteye或者cnblogs上的文章，可抓取列表和单个文章
* 第三方授权登录：集成JustAuthPlus实现第三方授权登录
* 自定义网站内容：管理员可自定义“关于本站”、“留言板”、“友情链接”、“免责声明”、“Footer”、“鼠标点击时的气泡文字”、“热门搜索的待选项”等内容
* 流控：针对异常IP的连续大量访问，系统会自动封禁该IP。

## 2 相关技术综述

### 2.1 java

#### 2.1.1 java简介

Java是由Sun Microsystems公司于1995年5月推出的Java面向对象程序设计语言和Java平台的总称。由James Gosling和同事们共同研发，并在1995年正式推出[ ]。

Java语言发展到今天经历了三个发展方向：J2SE，J2ME，J2EE，最核心的部分是J2SE，J2ME和J2EE是在J2SE基础之上发展起来的，3种技术的关系图如图2-1所示：

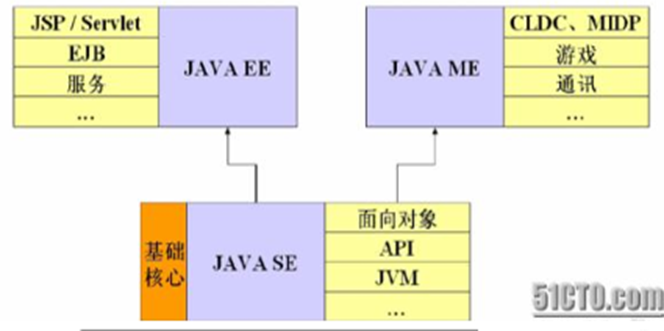


图2-1 J2SE，J2ME和J2EE三者间关系图

#### 2.1.2 java特性

* Java语言具有以下特点：
* 简洁有效：Java语言是一种相当简洁的“面向对象”程序设计语言
* 可移植性：java编写的程序可以在多个不同的平台上运行，基本上不需要做任何修改。
* 面向对象：面向对象是指以对象为基本粒度，其下包含属性和方法。Java是一种面向对象的语言。
* 较好的性能：JAVA语言源程序编写完成后，先使用java伪编译器进行伪编译，将其转换为中间码再解释，拥有较好的运行性能。

### 2.2 Spring Boot

#### 2.2.1 Spring Boot简介

Spring Boot是由Pivotal团队提供的全新[框架](https://baike.baidu.com/item/%E6%A1%86%E6%9E%B6/1212667" \t "_blank)，其设计目的是用来[简化](https://baike.baidu.com/item/%E7%AE%80%E5%8C%96/3374416)新[Spring](https://baike.baidu.com/item/Spring/85061)应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。

SpringBoot基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

#### 2.2.2 Spring Boot特征

* SpringBoot所具备的特征有：
* 可以创建独立的[Spring](https://baike.baidu.com/item/Spring/85061)应用程序，并且基于其Maven或Gradle插件，可以创建可执行的JARs和WARs；
* 内嵌Tomcat或Jetty等Servlet容器；
* 提供自动配置的“starter”项目对象模型（POMS）以简化[Maven](https://baike.baidu.com/item/Maven/6094909" \t "_blank)配置；
* 尽可能自动配置Spring容器；
* 提供准备好的特性，如指标、健康检查和外部化配置；
* 绝对没有代码生成，不需要XML配置。

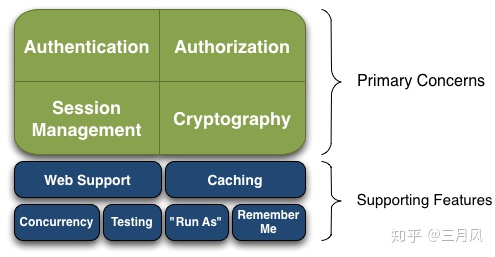
### 2.3 Apache Shiro

#### 2.3.1 Apache Shiro 简介

Apache Shiro 是 Java 的一个安全框架。目前，使用 Apache Shiro 的人越来越多，因为它相当简单，对比 Spring Security，可能没有 Spring Security 做的功能强大，但是在实际工作时可能并不需要那么复杂的东西，所以使用小而简单的 Shiro 就足够了。对于它俩到底哪个好，这个不必纠结，能更简单的解决项目问题就好了。

Shiro 可以非常容易的开发出足够好的应用，其不仅可以用在 JavaSE 环境，也可以用在 JavaEE 环境。Shiro 可以帮助我们完成：认证、授权、加密、会话管理、与 Web 集成、缓存等。这不就是我们想要的嘛，而且 Shiro 的 API 也是非常简单；

其基本功能点如下图所示：



Authentication：身份认证 / 登录，验证用户是不是拥有相应的身份；

Authorization：授权，即权限验证，验证某个已认证的用户是否拥有某个权限；即判断用户是否能做事情，常见的如：验证某个用户是否拥有某个角色。或者细粒度的验证某个用户对某个资源是否具有某个权限；

Session Manager：会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中；会话可以是普通 JavaSE 环境的，也可以是如 Web 环境的；

Cryptography：加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储；

Web Support：Web 支持，可以非常容易的集成到 Web 环境；

Caching：缓存，比如用户登录后，其用户信息、拥有的角色 / 权限不必每次去查，这样可以提高效率；

Concurrency：shiro 支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能把权限自动传播过去；

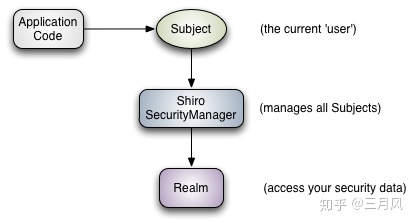
Testing：提供测试支持；

Run As：允许一个用户假装为另一个用户（如果他们允许）的身份进行访问；

Remember Me：记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录了。

记住一点，Shiro 不会去维护用户、维护权限；这些需要我们自己去设计 / 提供；然后通过相应的接口注入给 Shiro 即可。

#### 2.3.2 主要功能



三个核心组件：Subject, SecurityManager 和 Realms.

Subject：即“当前操作用户”。但是，在Shiro中，Subject这一概念并不仅仅指人，也可以是第三方进程、后台帐户（Daemon Account）或其他类似事物。它仅仅意味着“当前跟软件交互的东西”。

Subject代表了当前用户的安全操作，SecurityManager则管理所有用户的安全操作。

SecurityManager：它是Shiro框架的核心，典型的[Facade模式](https://baike.baidu.com/item/Facade%E6%A8%A1%E5%BC%8F/7557140)，Shiro通过SecurityManager来管理内部组件实例，并通过它来提供安全管理的各种服务。

Realm： Realm充当了Shiro与应用安全数据间的“桥梁”或者“连接器”。也就是说，当对用户执行认证（登录）和授权（访问控制）验证时，Shiro会从应用配置的Realm中查找用户及其权限信息。

从这个意义上讲，Realm实质上是一个安全相关的DAO：它封装了数据源的连接细节，并在需要时将相关数据提供给Shiro。当配置Shiro时，你必须至少指定一个Realm，用于认证和（或）授权。配置多个Realm是可以的，但是至少需要一个。

Shiro内置了可以连接大量安全数据源（又名目录）的Realm，如LDAP、关系数据库（JDBC）、类似INI的文本配置资源以及属性文件等。如果系统默认的Realm不能满足需求，你还可以插入代表自定义数据源的自己的Realm实现。

### 2.4 Bootstrap

#### 2.4.1 Bootstrap简介

Bootstrap是美国[Twitter](https://baike.baidu.com/item/Twitter/2443267)公司的设计师Mark Otto和Jacob Thornton合作基于HTML、CSS、[JavaScript](https://baike.baidu.com/item/JavaScript/321142" \t "_blank) 开发的简洁、直观、强悍的[前端](https://baike.baidu.com/item/%E5%89%8D%E7%AB%AF/5956545)开发框架，使得 Web 开发更加快捷。Bootstrap提供了优雅的HTML和CSS规范，它即是由动态CSS语言[Less](https://baike.baidu.com/item/Less)写成。Bootstrap一经推出后颇受欢迎，一直是[GitHub](https://baike.baidu.com/item/GitHub)上的热门开源项目，包括[NASA](https://baike.baidu.com/item/NASA)的MSNBC（[微软全国广播公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%A8%E5%9B%BD%E5%B9%BF%E6%92%AD%E5%85%AC%E5%8F%B8/8750737)）的Breaking News都使用了该项目。国内一些移动开发者较为熟悉的框架，如[WeX5](https://baike.baidu.com/item/WeX5)前端开源框架等，也是基于Bootstrap源码进行性能优化而来。

#### 2.4.2包含内容

基本结构：Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。这将在Bootstrap 基本结构部分详细讲解。

CSS：Bootstrap 自带以下特性：全局的 CSS设置、定义基本的 HTML 元素样式、可扩展的 class，以及一个先进的网格系统。这将在Bootstrap CSS部分详细讲解。

组件：Bootstrap 包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。这将在布局组件部分详细讲解。

JavaScript 插件：Bootstrap包含了十几个自定义的jQuery 插件。您可以直接包含所有的插件，也可以逐个包含这些插件。这将在Bootstrap插件部分详细讲解。

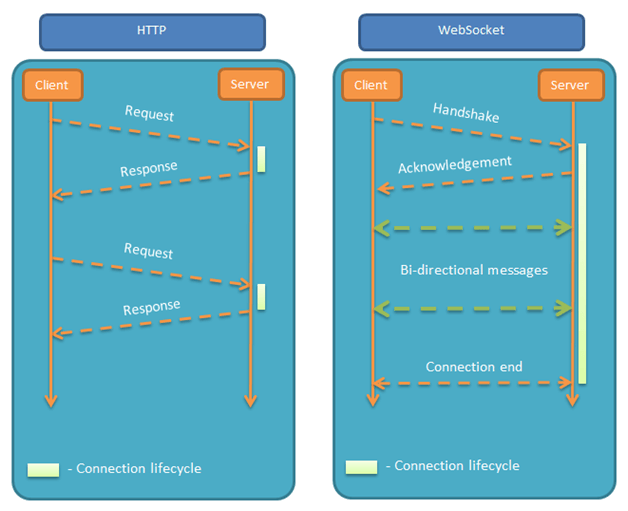
定制：可以定制Bootstrap的组件、LESS 变量和jQuery 插件来得到自己的版本。

### 2.5 Websocket

#### 2.5.1 Websocket简介

WebSocket是一种在单个[TCP](https://baike.baidu.com/item/TCP" \t "_blank)连接上进行[全双工](https://baike.baidu.com/item/%E5%85%A8%E5%8F%8C%E5%B7%A5" \t "_blank)通信的协议。WebSocket通信协议于2011年被[IETF](https://baike.baidu.com/item/IETF" \t "_blank)定为标准RFC 6455，并由RFC7936补充规范。WebSocket [API](https://baike.baidu.com/item/API)也被[W3C](https://baike.baidu.com/item/W3C" \t "_blank)定为标准。

WebSocket使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在WebSocket API中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。



#### 2.5.2 特点

（1）建立在 TCP 协议之上，服务器端的实现比较容易。

（2）与 HTTP 协议有着良好的兼容性。默认端口也是80和443，并且握手阶段采用 HTTP 协议，因此握手时不容易屏蔽，能通过各种 HTTP 代理服务器。

（3）数据格式比较轻量，性能开销小，通信高效。

（4）可以发送文本，也可以发送二进制数据。

（5）没有同源限制，客户端可以与任意服务器通信。

（6）协议标识符是ws（如果加密，则为wss），服务器网址就是 URL。

#### 2.5.3 优点

较少的控制开销。在连接创建后，服务器和客户端之间交换数据时，用于协议控制的数据包头部相对较小。在不包含扩展的情况下，对于服务器到客户端的内容，此头部大小只有2至10[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)（和数据包长度有关）；对于客户端到服务器的内容，此头部还需要加上额外的4字节的[掩码](https://baike.baidu.com/item/%E6%8E%A9%E7%A0%81" \t "_blank)。相对于HTTP请求每次都要携带完整的头部，此项开销显著减少了。

更强的实时性。由于协议是全双工的，所以服务器可以随时主动给客户端下发数据。相对于HTTP请求需要等待客户端发起请求服务端才能响应，延迟明显更少；即使是和Comet等类似的长轮询比较，其也能在短时间内更多次地传递数据。

保持连接状态。与HTTP不同的是，Websocket需要先创建连接，这就使得其成为一种有状态的协议，之后通信时可以省略部分状态信息。而HTTP请求可能需要在每个请求都携带状态信息（如身份认证等）。

更好的二进制支持。Websocket定义了[二进制](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6" \t "_blank)帧，相对HTTP，可以更轻松地处理二进制内容。

可以支持扩展。Websocket定义了扩展，用户可以扩展协议、实现部分自定义的子协议。如部分浏览器支持[压缩](https://baike.baidu.com/item/%E5%8E%8B%E7%BC%A9" \t "_blank)等。

更好的压缩效果。相对于[HTTP压缩](https://baike.baidu.com/item/HTTP%E5%8E%8B%E7%BC%A9" \t "_blank)，Websocket在适当的扩展支持下，可以沿用之前内容的[上下文](https://baike.baidu.com/item/%E4%B8%8A%E4%B8%8B%E6%96%87" \t "_blank)，在传递类似的数据时，可以显著地提高压缩率。

### 2.6 Freemarker

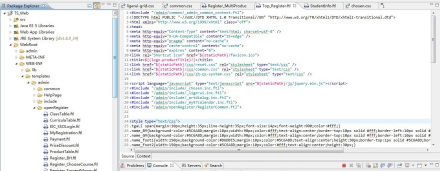
#### 2.6.1 Freemarker简介

FreeMarker是一款[模板引擎](https://baike.baidu.com/item/%E6%A8%A1%E6%9D%BF%E5%BC%95%E6%93%8E/907667" \t "_blank)： 即一种基于模板和要改变的数据， 并用来生成输出文本（[HTML](https://baike.baidu.com/item/HTML)网页、[电子邮件](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6/111106" \t "_blank)、[配置文件](https://baike.baidu.com/item/%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6/286550)、[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81/3969)等）的通用工具。 它不是面向最终用户的，而是一个Java类库，是一款程序员可以嵌入他们所开发产品的组件。

FreeMarker是[免费](https://baike.baidu.com/item/%E5%85%8D%E8%B4%B9/131326" \t "_blank)的，基于[Apache](https://baike.baidu.com/item/Apache/8512995)许可证2.0版本发布。其模板编写为FreeMarker Template Language（FTL），属于简单、专用的语言。需要准备数据在真实编程语言中来显示，比如[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)查询和业务运算， 之后模板显示已经准备好的数据。在模板中，主要用于如何展现数据， 而在模板之外注意于要展示什么数据。

## 2.6.2 性能特点

模板并没有包含[程序逻辑](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E9%80%BB%E8%BE%91" \t "_blank)来查找当前的访问者是谁，或者去查询数据库获取最新的产品。显示的数据是在FreeMarker之外准备的，通常是一些“真正的”编程语言（比如[Java](https://baike.baidu.com/item/Java/85979)）所编写的代码。模板作者无需知道这些值是如何计算出的。事实上，这些值的计算方式可以完全被修改，而模板可以保持不变，而且页面的样式也可以完全被修改而无需改动模板。当模板作者（设计师）和程序员不是同一人时，显示逻辑和业务逻辑相分离的做法是非常有用的，即便模板作者和程序员是一个人，这么来做也会帮助管理应用程序的复杂性。保证模板专注于显示问题（[视觉设计](https://baike.baidu.com/item/%E8%A7%86%E8%A7%89%E8%AE%BE%E8%AE%A1" \t "_blank)，布局和[格式化](https://baike.baidu.com/item/%E6%A0%BC%E5%BC%8F%E5%8C%96/2209232)）是高效使用模板引擎的关键。

[](https://baike.baidu.com/pic/freemarker/9489366/0/342ac65c10385343ac38b6929413b07ecb808889?fr=lemma&ct=single)MyEclipse工具下的编辑界面

##### 2.6.2.1通用性

能够生成各种文本：[HTML](https://baike.baidu.com/item/HTML)、[XML](https://baike.baidu.com/item/XML" \t "_blank)、[RTF](https://baike.baidu.com/item/RTF" \t "_blank)、Java源代码等等。

易于嵌入到产品中：[轻量级](https://baike.baidu.com/item/%E8%BD%BB%E9%87%8F%E7%BA%A7)；不需要[Servlet](https://baike.baidu.com/item/Servlet)环境。

插件式模板载入器：可以从任何源载入模板，如本地文件、数据库等等。

可以按所需生成文本：保存到本地文件；作为[Email](https://baike.baidu.com/item/Email" \t "_blank)发送；从Web应用程序发送它返回给Web浏览器。

##### 2.6.2.2 模板语言

所有常用的指令：[include](https://baike.baidu.com/item/include/5015256)、if/elseif/else、[循环结构](https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E7%BB%93%E6%9E%84" \t "_blank)。

在模板中[创建](https://baike.baidu.com/item/%E5%88%9B%E5%BB%BA" \t "_blank)和改变变量。

几乎在任何地方都可以使用复杂表达式来指定值。

命名的宏，可以具有位置参数和嵌套内容。

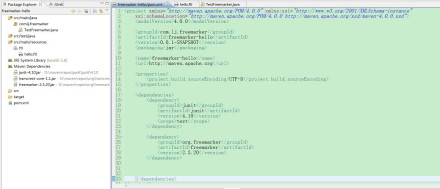
名字空间有助于建立和维护可重用的宏库，或者将一个大工程分成模块，而不必担心名字冲突。

输出转换块：在嵌套模板片段生成输出时，转换HTML转义、压缩、语法[高亮](https://baike.baidu.com/item/%E9%AB%98%E4%BA%AE" \t "_blank)等等；可以定义自己的转换。

##### 2.6.2.3 通用数据模型

FreeMarker不是直接反射到Java对象，Java对象通过插件式对象封装，以变量方式在模板中显示。

可以使用抽象（接口）方式表示对象（[JavaBean](https://baike.baidu.com/item/JavaBean" \t "_blank)、[XML](https://baike.baidu.com/item/XML" \t "_blank)文档、[SQL](https://baike.baidu.com/item/SQL" \t "_blank)查询结果集等等），告诉模板开发者使用。方法，使其不受技术细节的打扰。

[](https://baike.baidu.com/pic/freemarker/9489366/0/2fdda3cc7cd98d106f1a3c82263fb80e7aec90b2?fr=lemma&ct=single)xml配置

##### 2.6.2.4 为Web准备

在模板语言中内建处理典型Web相关任务（如[HTML](https://baike.baidu.com/item/HTML" \t "_blank)转义）的结构。

能够集成到Model2 Web应用框架中作为JSP的替代。

支持JSP标记库。

为MVC模式设计：分离可视化设计和应用程序逻辑；分离页面设计员和程序员。

##### 2.6.2.5 智能的国际化和本地化

字符集智能化（内部使用[UNICODE](https://baike.baidu.com/item/UNICODE" \t "_blank)）。

数字格式本地化敏感。

日期和时间格式本地化敏感。

非US[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)可以用作标识（如变量名）。

多种不同语言的相同模板。

##### 2.6.2.6 XML处理能力

<#recurse> 和<#visit>指令（2.3版本）用于递归遍历[XML](https://baike.baidu.com/item/XML" \t "_blank)树。

在模板中清楚和直接的访问XML[对象模型](https://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1%E6%A8%A1%E5%9E%8B)

### 2.7 开发环境介绍

#### 2.7.1 IDEA简介

IDEA 全称 IntelliJ IDEA，是[java](https://baike.baidu.com/item/java/85979)[编程语言](https://baike.baidu.com/item/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80/9845131)开发的集成环境。IntelliJ在业界被公认为最好的java开发工具，尤其在智能代码助手、代码自动提示、[重构](https://baike.baidu.com/item/%E9%87%8D%E6%9E%84/2182519)、[JavaEE](https://baike.baidu.com/item/JavaEE/3066623)支持、各类版本工具([git](https://baike.baidu.com/item/git/12647237)、[svn](https://baike.baidu.com/item/svn/3311103" \t "_blank)等)、[JUnit](https://baike.baidu.com/item/JUnit/1211849" \t "_blank)、[CVS](https://baike.baidu.com/item/CVS/405463" \t "_blank)整合、代码分析、 创新的[GUI](https://baike.baidu.com/item/GUI/479966)设计等方面的功能可以说是超常的。IDEA是[JetBrains](https://baike.baidu.com/item/JetBrains/7502758" \t "_blank)公司的产品，这家公司[总部](https://baike.baidu.com/item/%E6%80%BB%E9%83%A8/5289033)位于[捷克共和国](https://baike.baidu.com/item/%E6%8D%B7%E5%85%8B%E5%85%B1%E5%92%8C%E5%9B%BD/418555)的首都[布拉格](https://baike.baidu.com/item/%E5%B8%83%E6%8B%89%E6%A0%BC/632)，开发人员以严谨著称的[东欧](https://baike.baidu.com/item/%E4%B8%9C%E6%AC%A7/7149362)[程序员](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E5%91%98/62748)为主。它的[旗舰](https://baike.baidu.com/item/%E6%97%97%E8%88%B0/4724820)版本还支持[HTML](https://baike.baidu.com/item/HTML/97049)，[CSS](https://baike.baidu.com/item/CSS/5457" \t "_blank)，[PHP](https://baike.baidu.com/item/PHP/9337" \t "_blank)，[MySQL](https://baike.baidu.com/item/MySQL/471251" \t "_blank)，[Python](https://baike.baidu.com/item/Python/407313" \t "_blank)等。免费版只支持Java,[Kotlin](https://baike.baidu.com/item/Kotlin/1133714)等少数语言。

#### 2.7.2 MySql简介

MySQL是开源多线程的关系数据库管理系统，由Michael “Monty” Winenius在1995年创建的，目前MySQL AB在全球大约有超过6000000的用户。

MySQL的插件式存储引擎，将数据库理论发挥到淋漓尽致，完美的映射了数据库的外模式和内模式理论，据MySQL官方称，MySQL数据库占有全球数据库25%的市场份额、互联网公司80%的市场份额。

MySQL5.5融合了MySQL数据库和InnoDB存储引擎的优点，能够提供高性能的数据管理解决方案，包括InnoDB作为默认的数据库存储引擎、提高了Windows系统下的系统性能和可扩展性、改善性能和可扩展性、提高实用性。MySQL的优势还在于他的免费开源，能够占据很大的市场比例。

#### 2.7.3 Redis简介

redis是一个key-value[存储系统](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F)。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list([链表](https://baike.baidu.com/item/%E9%93%BE%E8%A1%A8))、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些[数据类型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B" \t "_blank)都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了[memcached](https://baike.baidu.com/item/memcached" \t "_blank)这类key/value存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。[1]

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

#### 2.7.4 Navicat for MySQL简介

Navicat for MySQL是一套管理和开发MySQL或MariaDB的理想解决方案，支持单一程序，可同时连接到MySQL和MariaDB。这个功能齐备的前端软件为数据库管理、开发和维护提供了直观而强大的图形界面，给MySQL或MariaDB新手以及专业人士提供了一组全面的工具。

#### 2.7.5 Tomcat8.0简介

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。在开发过程中，可以将JavaWeb项目部署到Tomcat上，实现模拟效果，极大程度的方便了开发者。在项目开发完成后，也可以将Tomcat作为服务器部署线上的项目。

#### 2.7.6 阿里云服务器简介

阿里云服务器（Elastic Compute Service, ECS）是一种处理能力可弹性伸缩的计算服务，其管理方式比物理服务器更简单高效。 阿里云服务器帮助您快速构建更稳定、安全的应用，降低开发运维的难度和整体IT成本，使您能够更专注于核心业务创新。

阿里云服务器是基于阿里云自主研发的飞天大规模分布式计算系统，通过虚拟化技术整合IT资源，为各行业提供互联网基础设施服务。云服务器具有自助管理、数据安全保障、自动故障恢复和防网络攻击等高级功能，能够帮您简化开发部署过程，降低运维成本，构建按需扩展的网站架构，从而更适应互联网应用快速多变的特性。

### 3 需求分析

#### 3.1 功能需求分析

Java博客系统面向的用户是全体社会人员，人们可以利用此系统来管理自己的博客，从而达到方便多次获取信息渠道。该系统可以分为以下几大功能模块：

|  |  |  |
| --- | --- | --- |
| 模块 | 释义 | 备注 |
| blog-core | 核心业务类模块，提供基本的数据操作、工具处理等 | 该模块只是作为核心依赖包存在 |
| blog-admin | 后台管理模块 | 该模块作为单独项目打包部署 |
| blog-web | 前台模块 | 该模块作为单独项目打包部署 |
| blog-file | 文件上传相关模块 | 依赖包 |

#### 3.2系统设计

##### 3.2.1 数据库设计

数据库设计是Java博客系统中重要的设计部分，需要根据系统中的所需的功能来设定数据库中具体的数据表数量及其属性，整个系统中的数据交互都需要借助数据库实现。

Java博客系统数据库总共设定了二十个数据表，这些数据表的名称、作用及其属性如下：

1. biz\_article表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `title` | 文章标题 | varchar(255) | 是 |  |
| `user\_id` | 用户ID | bigint(20) | 是 |  |
| `cover\_image` | 文章封面图片 | varchar(255) | 是 |  |
| `qrcode\_path` | 文章专属二维码地址 | varchar(255) | 是 |  |
| `is\_markdown` | 是否使用markdown | tinyint(1) |  |  |
| `content\_md` | markdown版的文章内容 | mediumtext | 是 |  |
| `content` | 文章内容 | mediumtext | 是 |  |
| `top` | 是否置顶 | tinyint(1) | 是 |  |
| `type\_id` | 类型 | bigint(20) | 是 |  |
| `status` | 状态 | tinyint(1) | 是 |  |
| `recommended` | 是否推荐 | tinyint(1) | 是 |  |
| `original` | 是否原创 | tinyint(1) | 是 |  |
| `description` | 文章简介 | varchar(300) | 是 | 最多200字 |
| `keywords` | 文章关键字 | varchar(200) |  | 优化搜索 |
| `comment` | 是否开启评论 | tinyint(1) | 是 |  |
| `password` | 文章私密访问时的密钥 | varchar(255) |  |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_article\_look表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `article\_id` | 文章ID | bigint(20) | 是 |  |
| `user\_id` | 已登录用户ID | bigint(20) | 是 |  |
| `user\_ip` | 用户IP | varchar(50) | 是 |  |
| `look\_time` | 浏览时间 | datetime(0) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_article\_love表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `article\_id` | 文章ID | bigint(20) | 是 |  |
| `user\_id` | 已登录用户ID | bigint(20) | 是 |  |
| `user\_ip` | 用户IP | varchar(50) | 是 |  |
| `look\_time` | 浏览时间 | datetime(0) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_article\_tags表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `tag\_id` | 标签表主键 | bigint(20) |  |  |
| `article\_id` | 文章ID | bigint(20) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_comment表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `sid` | 被评论的文章或者页面的ID | bigint(20) | 是 |  |
| `user\_id` | 评论人的ID | bigint(20) | 是 |  |
| `pid` | 父级评论的id | bigint(20) | 是 |  |
| `qq` | 评论人的QQ | varchar(13) | 是 | 未登录用户 |
| `nickname` | 评论人的昵称 | varchar(13) | 是 | 未登录用户 |
| `avatar` | 评论人的头像地址 | varchar(255) | 是 |  |
| `email` | 评论人的邮箱地址 | varchar(100) | 是 | 未登录用户 |
| `url` | 评论人的网站地址 | varchar(100) | 是 | 未登录用户 |
| `status` | 评论的状态 | enum | 是 |  |
| `ip` | 评论时的ip | varchar(64) | 是 |  |
| `lng` | 经度 | varchar(50) | 是 |  |
| `lat` | 纬度 | varchar(50) | 是 |  |
| `address` | 评论时的地址 | varchar(100) | 是 |  |
| `os` | 评论时的系统类型 | varchar(64) | 是 |  |
| `os\_short\_name` | 评论时的系统的简称 | varchar(10) | 是 |  |
| `browser` | 评论时的浏览器类型 | varchar(64) | 是 |  |
| browser\_short\_name | 评论时的浏览器的简称 | varchar(10) | 是 |  |
| `content` | 评论的内容 | varchar(2000) | 是 |  |
| `remark` | 备注 | varchar(100) | 是 | 审核不通过时添加 |
| `support` | 支持（赞） | int(10) | 是 |  |
| `oppose` | 反对（踩） | int(10) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_tags表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `name` | 书签名 | varchar(50) | 是 |  |
| `description` | 描述 | varchar(100) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. biz\_type表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `pid` |  | bigint(20) |  |  |
| `name` | 文章类型名 | varchar(50) |  |  |
| `description` | 类型介绍 | varchar(100) | 是 |  |
| `sort` | 排序 | int(10) |  |  |
| `icon` | 图标 | varchar(100) |  |  |
| `available` | 是否可用 | tinyint(1) |  |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_config表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `config\_key` | 配置关键字 | varchar(50) | 是 |  |
| `config\_value` | 配置项内容 | longtext | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_link表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `url` | 链接地址 | varchar(255) | 是 |  |
| `name` | 链接名 | varchar(55) | 是 |  |
| `description` | 链接介绍 | varchar(255) | 是 |  |
| `email` | 友链站长邮箱 | varchar(100) | 是 |  |
| `qq` | 友链站长QQ | varchar(13) | 是 |  |
| `favicon` |  | varchar(255) | 是 |  |
| `status` | 状态 | tinyint(1) | 是 |  |
| `home\_page\_display` | 是否首页显示 | tinyint(1) | 是 |  |
| `remark` | 备注 | varchar(100) | 是 |  |
| `source` | 资源 | enum('ADMIN','AUTOMATIC') | 是 | 管理员添加、自动申请 |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_log表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `user\_id` | 已登录用户ID | bigint(20) | 是 |  |
| `type` | 日志类型 | enum('SYSTEM','VISIT','ERROR') | 是 | 系统操作日志，用户访问日志，异常记录日志 |
| `log\_level` | 日志级别 | enum('ERROR','WARN','INFO') | 是 |  |
| `content` | 日志内容 | varchar(2000) | 是 | （业务操作） |
| `params` | 请求参数 | varchar(2000) | 是 | （业务操作） |
| `spider\_type` | 爬虫类型 | varchar(20) | 是 | 当访问者被鉴定为爬虫时该字段表示爬虫的类型 |
| `ip` | 操作用户的ip | varchar(64) | 是 |  |
| `ua` | 操作用户的user\_agent | varchar(500) | 是 |  |
| `os` | 评论时的系统类型 | varchar(64) | 是 |  |
| `browser` | 评论时的浏览器类型 | varchar(64) | 是 |  |
| `request\_url` | 请求的路径 | varchar(3000) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_notice表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `user\_id` | 被通知的用户ID | bigint(20) | 是 |  |
| `status` | 通知状态 | enum('RELEASE','NOT\_RELEASE') | 是 |  |
| `title` | 通知的标题 | varchar(200) | 是 |  |
| `content` | 通知的内容 | varchar(2000) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_resources表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `name` | 资源名称 | varchar(100) | 是 |  |
| `type` | 资源类型 | varchar(50) | 是 |  |
| `url` | 资源链接 | varchar(200) | 是 |  |
| `permission` |  | varchar(200) | 是 |  |
| `external` | 是否外部链接 | tinyint(1) | 是 |  |
| `icon` | 菜单图标 | varchar(100) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_role表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `name` | 角色名 | varchar(100) | 是 |  |
| `description` | 描述 | varchar(100) |  |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_role\_resources表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `role\_id` | 角色id | bigint(20) | 是 |  |
| `resources\_id` | 资源id | bigint(20) |  |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_template表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `ref\_key` | 键 | varchar(100) | 是 |  |
| `ref\_value` | 模板内容 | text | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_update\_recorde表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `version` | 更新版本 | varchar(100) | 是 |  |
| `description` | 更新记录备注 | varchar(2500) | 是 |  |
| `recorde\_time` | 项目更新时间 | datetime(0) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_user表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `username` | 用户名 | varchar(25) | 是 |  |
| `password` | 登录密码 | varchar(55) | 是 |  |
| `nickname` | 昵称 | varchar(25) | 是 |  |
| `mobile` | 手机号 | varchar(20) | 是 |  |
| `qq` | QQ | varchar(13) | 是 |  |
| `email` | 邮箱地址 | varchar(255) | 是 |  |
| `company` | 公司 | tinyint(1) | 是 |  |
| `birthday` | 生日 | date | 是 |  |
| `gender` | 性别 | smallint(2) | 是 |  |
| `avatar` | 头像地址 | varchar(255) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

1. sys\_user\_role表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | bigint(20) | 是 | AUTO\_INCREMENT |
| `user\_id` | 用户id | bigint(20) | 是 |  |
| `role\_id` | 角色id | bigint(20) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

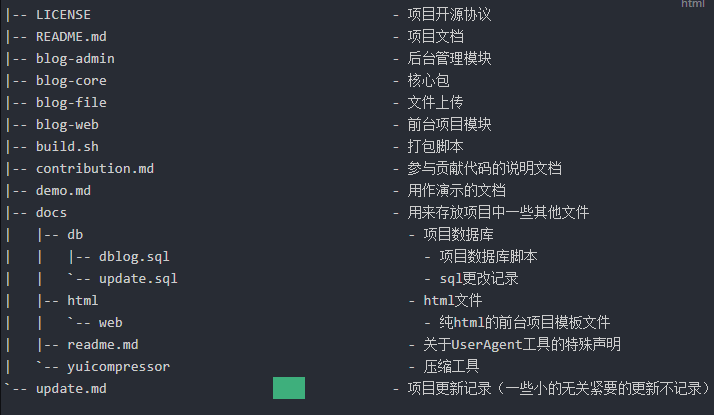
1. biz\_file表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | int(11) | 是 | AUTO\_INCREMENT |
| `user\_id` | 用户id | int(11) | 是 |  |
| `storage\_type` |  | varchar(55) | 是 |  |
| `original\_file\_name` |  | varchar(25) | 是 |  |
| `size` | 文件大小 | bigint(20) | 是 |  |
| `suffix` |  | varchar(13) | 是 |  |
| `width` |  | int(10) | 是 |  |
| `height` |  | int(10) | 是 |  |
| `file\_path` | 文件路径 | date | 是 |  |
| `full\_file\_path` | 绝对路径 | smallint(2) | 是 |  |
| ``file\_hash` ` |  | varchar(255) | 是 |  |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

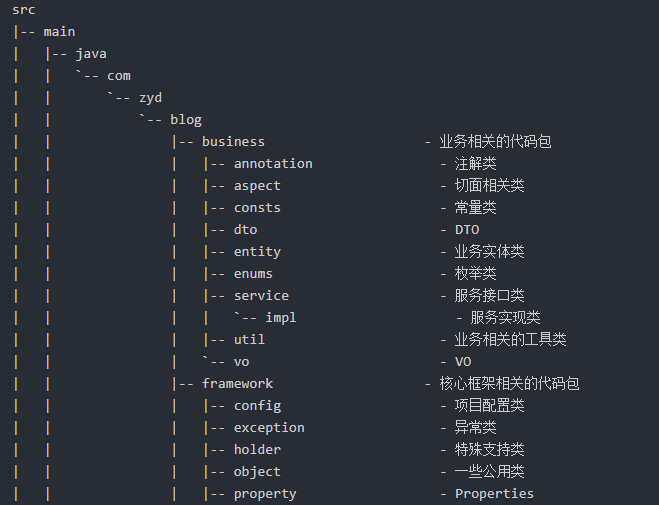
1. sys\_social\_config表

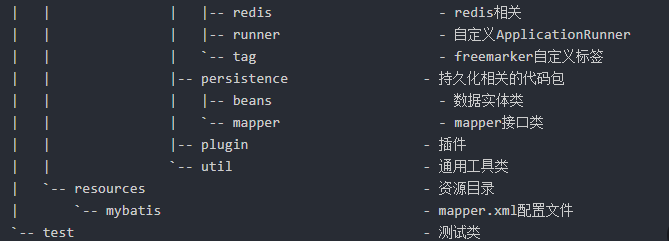
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 描述 | 类型（长度） | 非空 | 说明 |
| `id` | 编号 | int(11) | 是 | AUTO\_INCREMENT |
| `user\_id` | 用户id | int(11) | 是 |  |
| `client\_id` | 应用ID | varchar(55) | 是 |  |
| `client\_secret` | 应用密钥 | varchar(25) | 是 |  |
| `redirect\_uri` | 跳转的地址 | bigint(20) | 是 | 认证成功后 |
| `platform\_name` | 应用名 | varchar(13) | 是 |  |
| `platform` | 第三方平台 | int(10) | 是 |  |
| `logo` | 应用 LOGO | int(10) | 是 |  |
| `alipay\_public\_key` | 支付宝公钥 | date | 是 |  |
| `union\_id` | 是否需要申请unionid | varchar(255) | 是 |  |
| `stack\_overflow\_key` |  | varchar(200) | 是 |  |
| `agent\_id` | 授权方的网页应用ID | varchar(200) | 是 |  |
| `scope` | 授权范围 | varchar(200) | 是 |  |
| `available` | 当前配置 | bit(1) | 是 | 是否可用 |
| `create\_time` | 添加时间 | datetime(0) | 是 |  |
| `update\_time` | 更新时间 | datetime(0) | 是 |  |

#### 3.3项目结构简介

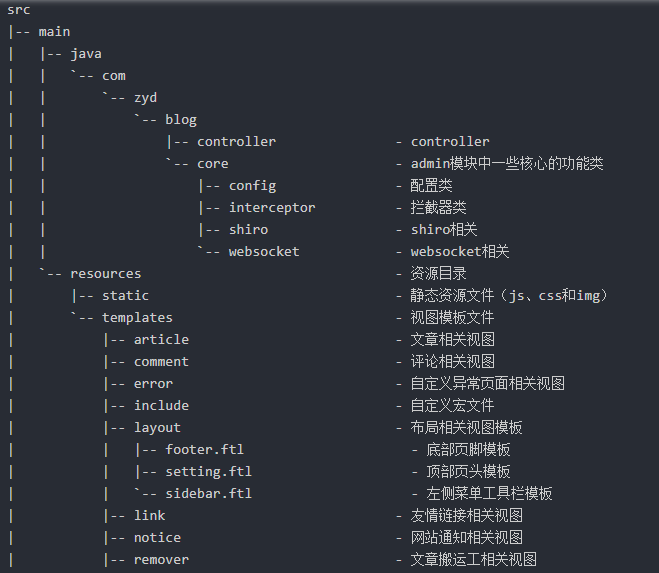


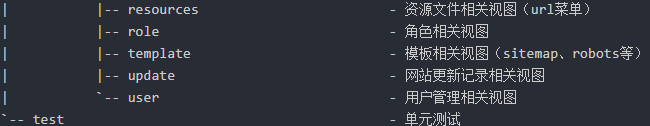
##### 3.3.1 blog-core目录结构简介



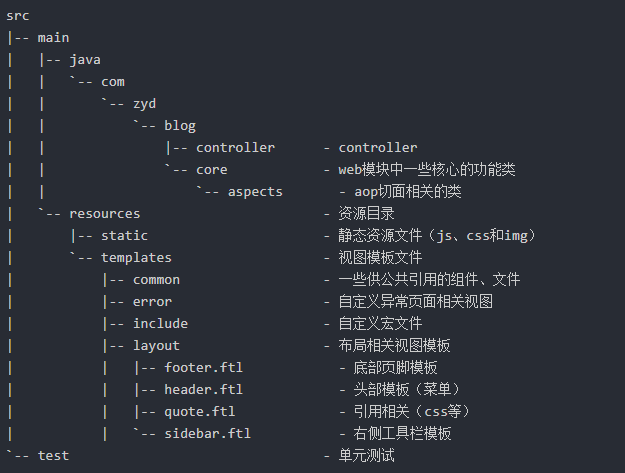


##### 3.3.2 blog-admin目录结构简介

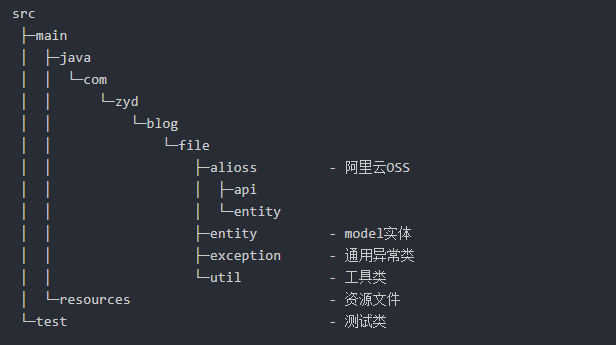




##### 3.3.3 blog-web目录结构简介

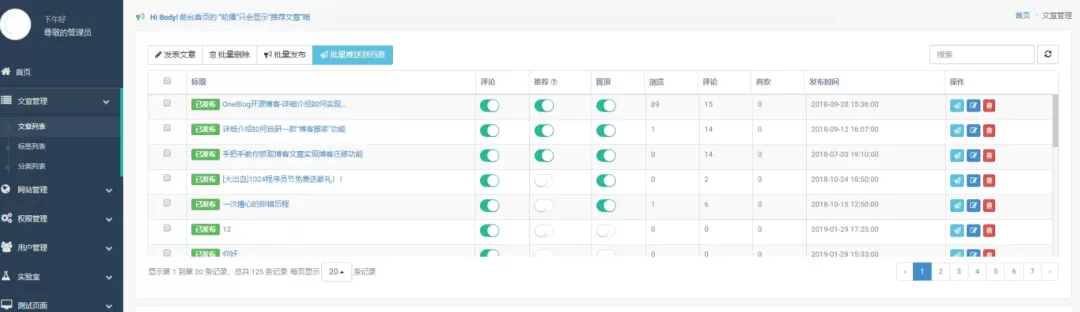
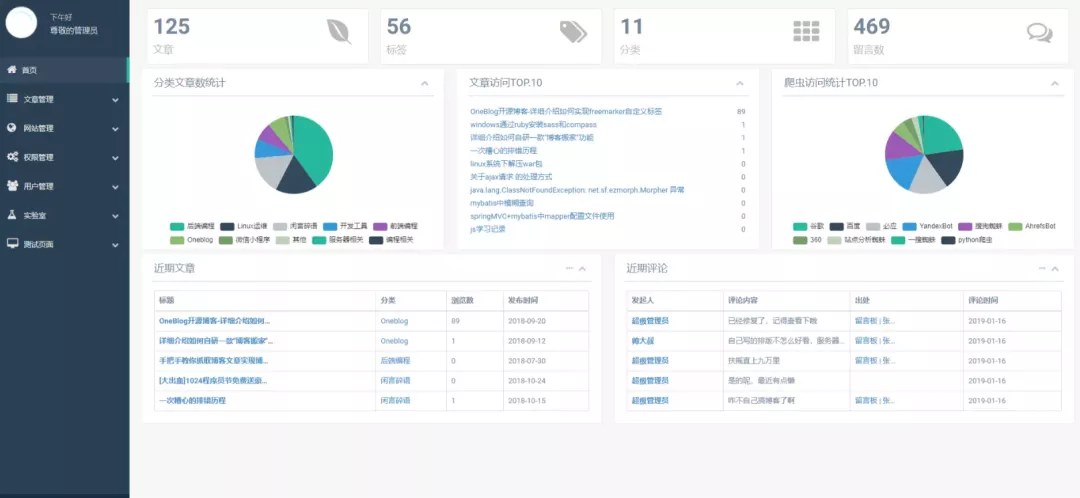


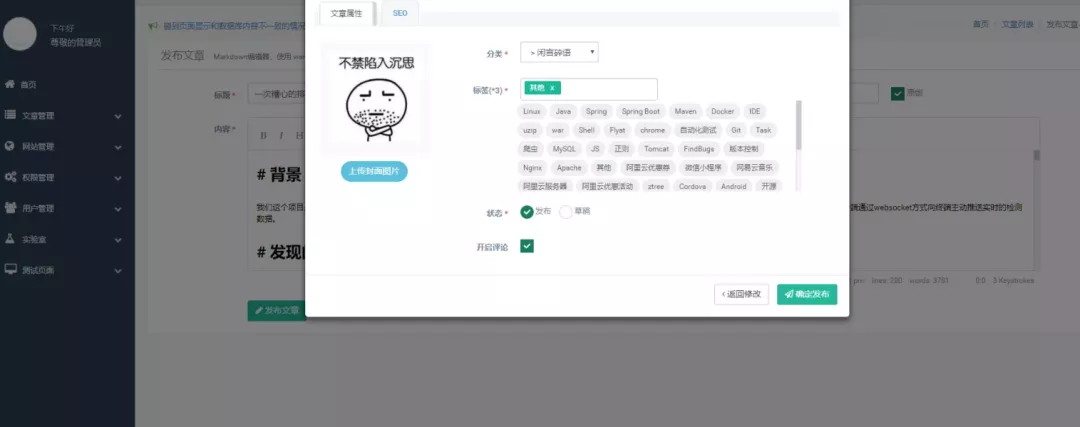
##### 3.3.4 blog-file目录结构简介

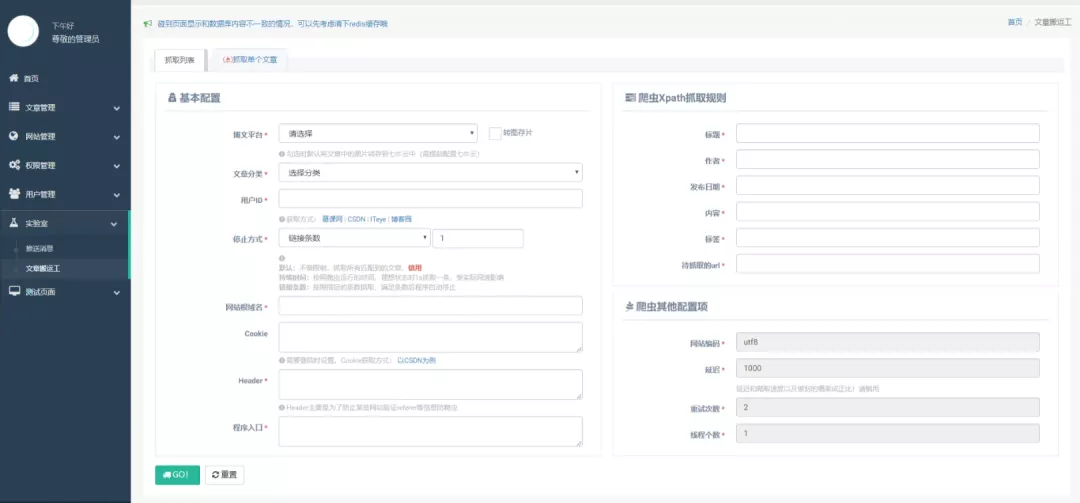
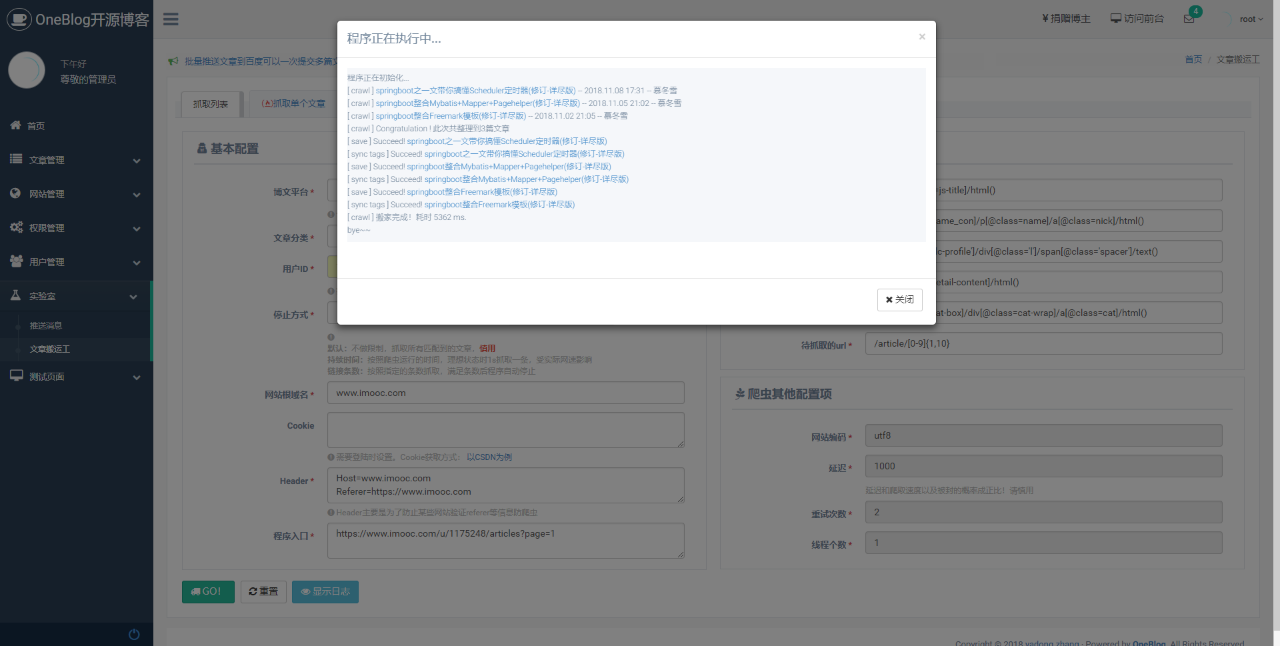


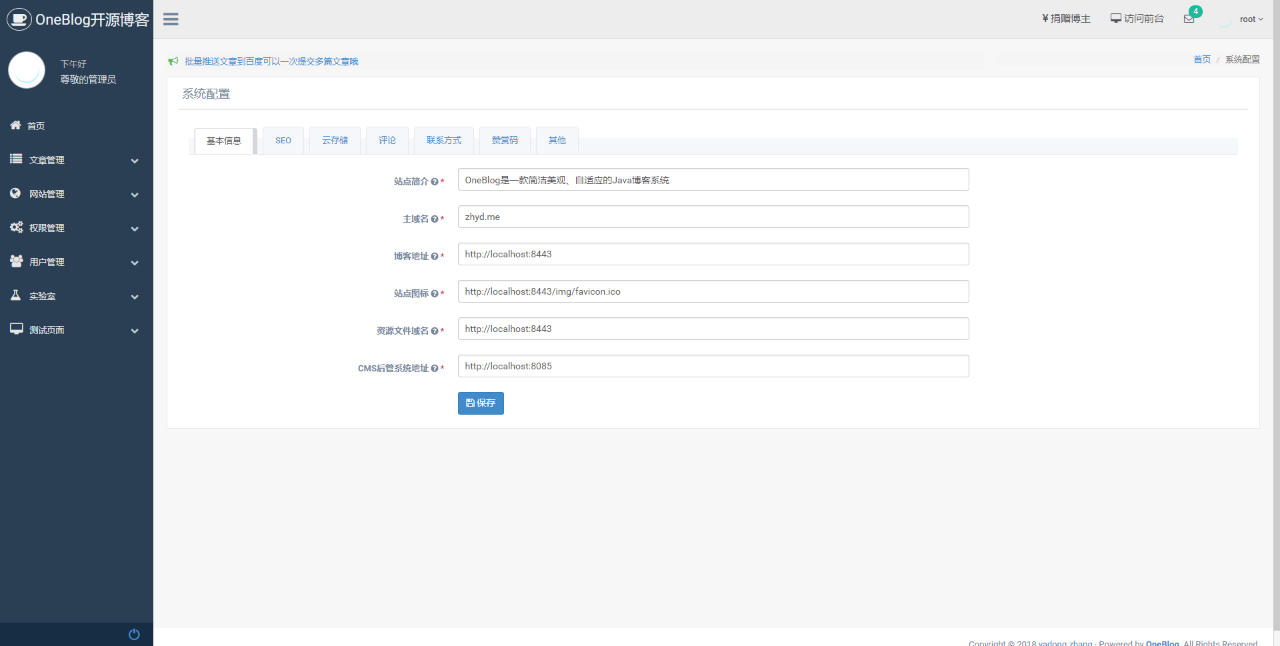
### 4．实现

#### 4.1 后台页面

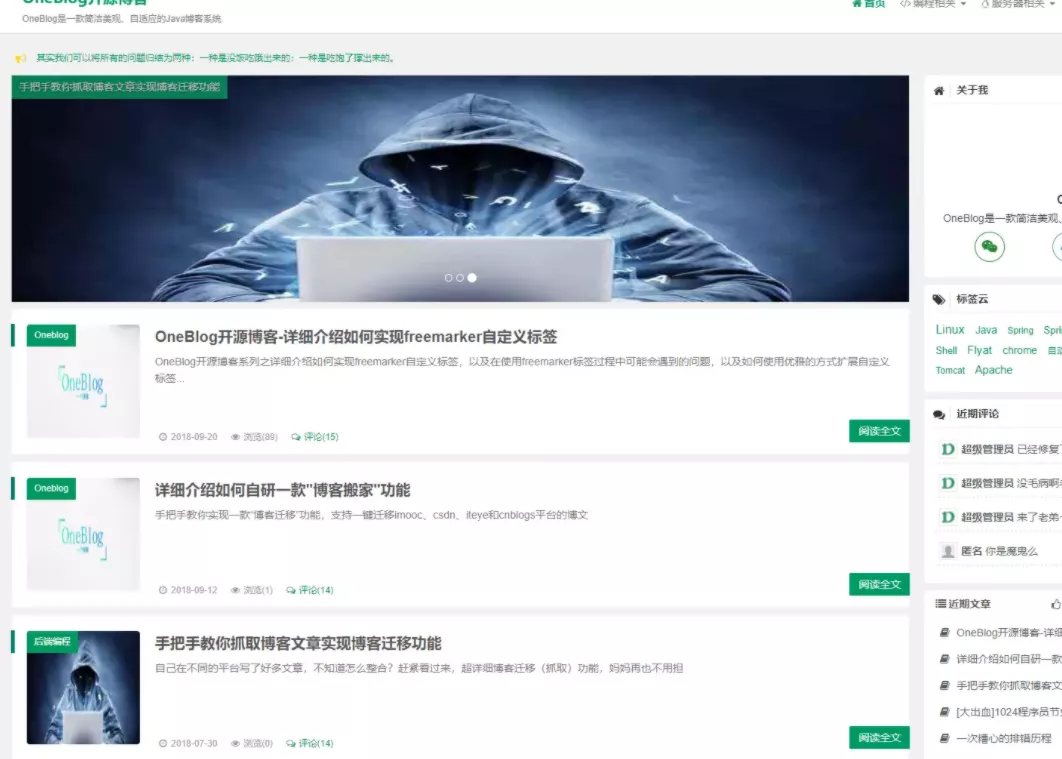








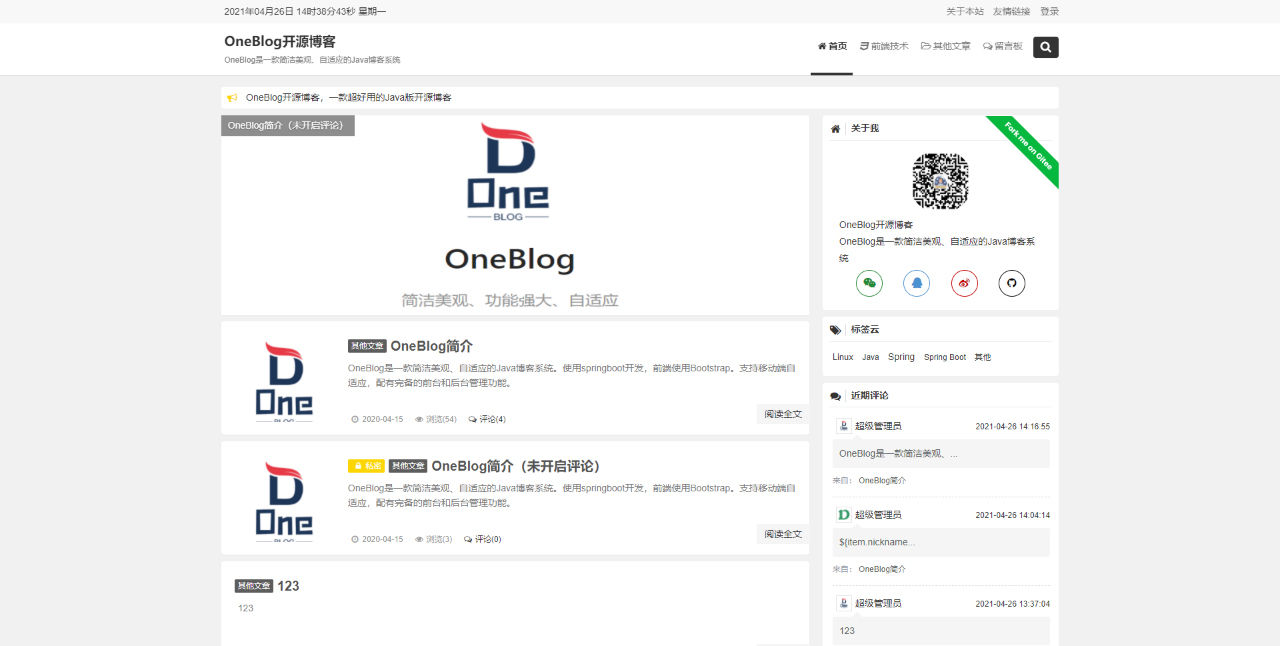
#### 4.2 前台页面







#### 4.3

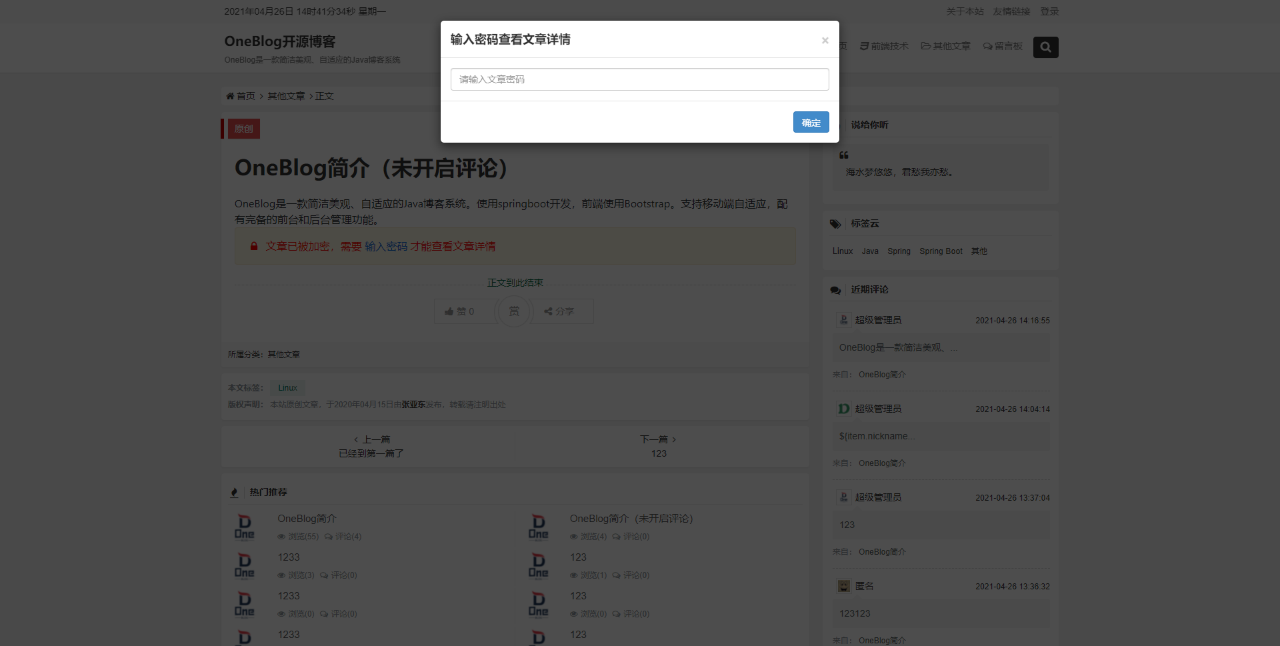


##### 4.3.1 OneBlog简介





##### 4.3.2 文章加密需输入密码



### 5.结束语

经过一这段时间的艰苦奋斗，基本上对Java博客系统的熟悉运用，本系统采用了Spring Boot、Apache Shiro 等框架。在此之前，对于后台框架以及前端技术这些知识，仅仅是了解一二、还不太熟悉原理。因此，这些知识对我来说还是十分陌生。如果不是使用框架开发的话，开发起来会很麻烦，项目也没有条理性，不适合对其进行管理。在实践中检验理论，不断加深我的理解。在使用过程中遇到了许多的问题，基本每天从早到晚都在做查询相关资料，这需要我有耐心，顽强的毅力，永不放弃的精神。在实际开发中碰到困难时，需要到网上去查看相关资料、各种技术文档以及视频教程。

通过本次课程懂得了理论与实际相结合是很重要的只 有理论知识是远远不够的，只有把所学的理论知识与实践相结合起来，从理论中得出结论，提高自己的实际动手能力和独立思考的能力。在实际开发过程中遇到问题解决问题，在这个过程中又学习到了新的知识，可以说是受益匪浅。

### 附录

#### 附录1：登录相关代码

package com.zyd.blog.controller;

import com.zyd.blog.business.annotation.BussinessLog;

import com.zyd.blog.business.entity.UserPwd;

import com.zyd.blog.business.service.SysUserService;

import com.zyd.blog.framework.holder.RequestHolder;

import com.zyd.blog.framework.object.ResponseVO;

import com.zyd.blog.framework.property.AppProperties;

import com.zyd.blog.util.ResultUtil;

import com.zyd.blog.util.SessionUtil;

import lombok.extern.slf4j.Slf4j;

import org.apache.shiro.SecurityUtils;

import org.apache.shiro.authc.UsernamePasswordToken;

import org.apache.shiro.subject.Subject;

import org.apache.shiro.web.util.SavedRequest;

import org.apache.shiro.web.util.WebUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.util.StringUtils;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

/\*\*

\* 登录相关

\*

\* @author yadong.zhang (yadong.zhang0415(a)gmail.com)

\* @version 1.0

\* @website https://docs.zhyd.me

\* @date 2018/4/24 14:37

\* @since 1.0

\*/

@Slf4j

@Controller

@RequestMapping(value = "/passport")

public class PassportController {

@Autowired

private AppProperties config;

@Autowired

private SysUserService userService;

@BussinessLog("进入登录页面")

@GetMapping("/login")

public ModelAndView login(Model model) {

model.addAttribute("enableKaptcha", config.isEnableKaptcha());

return ResultUtil.view("/login");

}

/\*\*

\* 登录

\*

\* @param username

\* @param password

\* @return

\*/

@BussinessLog("[{1}]登录系统")

@PostMapping("/signin")

@ResponseBody

public ResponseVO submitLogin(String username, String password, boolean rememberMe, String kaptcha) {

if (config.isEnableKaptcha()) {

if (StringUtils.isEmpty(kaptcha) || !kaptcha.equals(SessionUtil.getKaptcha())) {

return ResultUtil.error("验证码错误！");

}

SessionUtil.removeKaptcha();

}

UsernamePasswordToken token = new UsernamePasswordToken(username, password, rememberMe);

//获取当前的Subject

Subject currentUser = SecurityUtils.getSubject();

try {

// 在调用了login方法后,SecurityManager会收到AuthenticationToken,并将其发送给已配置的Realm执行必须的认证检查

// 每个Realm都能在必要时对提交的AuthenticationTokens作出反应

// 所以这一步在调用login(token)方法时,它会走到xxRealm.doGetAuthenticationInfo()方法中,具体验证方式详见此方法

currentUser.login(token);

SavedRequest savedRequest = WebUtils.getSavedRequest(RequestHolder.getRequest());

String historyUrl = null;

if(null != savedRequest) {

if(!savedRequest.getMethod().equals("POST")) {

historyUrl = savedRequest.getRequestUrl();

}

}

return ResultUtil.success(null, historyUrl);

} catch (Exception e) {

log.error("登录失败，用户名[{}]：{}", username, e.getMessage());

token.clear();

return ResultUtil.error(e.getMessage());

}

}

/\*\*

\* 修改密码

\*

\* @return

\*/

@BussinessLog("修改密码")

@PostMapping("/updatePwd")

@ResponseBody

public ResponseVO updatePwd(@Validated UserPwd userPwd, BindingResult bindingResult) throws Exception {

if (bindingResult.hasErrors()) {

return ResultUtil.error(bindingResult.getFieldError().getDefaultMessage());

}

boolean result = userService.updatePwd(userPwd);

SessionUtil.removeAllSession();

return ResultUtil.success(result ? "密码已修改成功，请重新登录" : "密码修改失败");

}

/\*\*

\* 使用权限管理工具进行用户的退出，跳出登录，给出提示信息

\*

\* @param redirectAttributes

\* @return

\*/

@BussinessLog("退出系统")

@GetMapping("/logout")

public ModelAndView logout(RedirectAttributes redirectAttributes) {

// http://www.oschina.net/question/99751\_91561

// 此处有坑： 退出登录，其实不用实现任何东西，只需要保留这个接口即可，也不可能通过下方的代码进行退出

// SecurityUtils.getSubject().logout();

// 因为退出操作是由Shiro控制的

redirectAttributes.addFlashAttribute("message", "您已安全退出");

return ResultUtil.redirect("index");

}

}

#### 附录2：验证码代码

package com.zyd.blog.controller;

import com.zyd.blog.plugin.kaptcha.Captcha;

import com.zyd.blog.plugin.kaptcha.GifCaptcha;

import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* 验证码

\*

\* @author yadong.zhang (yadong.zhang0415(a)gmail.com)

\* @version 1.0

\* @website https://docs.zhyd.me

\* @date 2018/4/24 14:37

\* @since 1.0

\*/

@Slf4j

@Controller

public class KaptchaController {

@GetMapping("/getKaptcha")

@ResponseBody

public void getKaptcha(HttpServletResponse response) {

try {

response.setHeader("Pragma", "No-cache");

response.setHeader("Cache-Control", "no-cache");

response.setDateHeader("Expires", 0);

response.setContentType("image/gif");

/\*\*

\* gif格式动画验证码

\* 宽，高，位数。

\*/

Captcha captcha = new GifCaptcha(146,33,4);

//输出

captcha.out(response.getOutputStream());

} catch (Exception e) {

e.printStackTrace();

log.error("获取验证码异常：{}", e.getMessage());

}

}

}

#### 附录3：申请友链、评论、点赞等代码

package com.zyd.blog.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.zyd.blog.business.annotation.BussinessLog;

import com.zyd.blog.business.entity.Article;

import com.zyd.blog.business.entity.Comment;

import com.zyd.blog.business.entity.Link;

import com.zyd.blog.business.enums.CommentStatusEnum;

import com.zyd.blog.business.enums.PlatformEnum;

import com.zyd.blog.business.service.BizArticleService;

import com.zyd.blog.business.service.BizCommentService;

import com.zyd.blog.business.service.SysLinkService;

import com.zyd.blog.business.service.SysNoticeService;

import com.zyd.blog.business.vo.CommentConditionVO;

import com.zyd.blog.framework.exception.ZhydArticleException;

import com.zyd.blog.framework.exception.ZhydCommentException;

import com.zyd.blog.framework.exception.ZhydLinkException;

import com.zyd.blog.framework.object.ResponseVO;

import com.zyd.blog.util.RestClientUtil;

import com.zyd.blog.util.ResultUtil;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.util.StringUtils;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* 网站接口类，申请友链、评论、点赞等

\*

\* @author yadong.zhang (yadong.zhang0415(a)gmail.com)

\* @version 1.0

\* @website https://docs.zhyd.me

\* @date 2018/4/18 11:48

\* @since 1.0

\*/

@Slf4j

@RestController

@RequestMapping("/api")

public class RestApiController {

@Autowired

private SysLinkService sysLinkService;

@Autowired

private BizCommentService commentService;

@Autowired

private BizArticleService articleService;

@Autowired

private SysNoticeService noticeService;

@PostMapping("/autoLink")

@BussinessLog(value = "自助申请友链", platform = PlatformEnum.WEB)

public ResponseVO autoLink(@Validated Link link, BindingResult bindingResult) {

log.info("申请友情链接......");

log.info(JSON.toJSONString(link));

if (bindingResult.hasErrors()) {

return ResultUtil.error(bindingResult.getFieldError().getDefaultMessage());

}

try {

sysLinkService.autoLink(link);

} catch (ZhydLinkException e) {

log.error("客户端自助申请友链发生异常", e);

return ResultUtil.error(e.getMessage());

}

return ResultUtil.success("已成功添加友链，祝您生活愉快！");

}

@PostMapping("/qq/{qq}")

@BussinessLog(value = "获取QQ信息", platform = PlatformEnum.WEB)

public ResponseVO qq(@PathVariable("qq") String qq) {

if (StringUtils.isEmpty(qq)) {

return ResultUtil.error("");

}

Map<String, String> resultMap = new HashMap<>(4);

String nickname = "匿名";

String json = RestClientUtil.get("https://users.qzone.qq.com/fcg-bin/cgi\_get\_portrait.fcg?uins=" + qq, "GBK");

if (!StringUtils.isEmpty(json)) {

try {

json = json.replaceAll("portraitCallBack|\\\\s\*|\\t|\\r|\\n", "");

json = json.substring(1, json.length() - 1);

log.info(json);

JSONObject object = JSONObject.parseObject(json);

JSONArray array = object.getJSONArray(qq);

nickname = array.getString(6);

} catch (Exception e) {

log.error("通过QQ号获取用户昵称发生异常", e);

}

}

resultMap.put("avatar", "https://q1.qlogo.cn/g?b=qq&nk=" + qq + "&s=40");

resultMap.put("nickname", nickname);

resultMap.put("email", qq + "@qq.com");

resultMap.put("url", "https://user.qzone.qq.com/" + qq);

return ResultUtil.success(null, resultMap);

}

@PostMapping("/comments")

@BussinessLog(value = "评论列表", platform = PlatformEnum.WEB, save = false)

public ResponseVO comments(CommentConditionVO vo) {

vo.setStatus(CommentStatusEnum.APPROVED.toString());

return ResultUtil.success(null, commentService.list(vo));

}

@PostMapping("/comment")

@BussinessLog(value = "发表评论", platform = PlatformEnum.WEB)

public ResponseVO comment(Comment comment) {

try {

commentService.comment(comment);

} catch (ZhydCommentException e) {

return ResultUtil.error(e.getMessage());

}

return ResultUtil.success("评论发表成功，系统正在审核，请稍后刷新页面查看！");

}

@PostMapping("/doSupport/{id}")

@BussinessLog(value = "点赞评论{1}", platform = PlatformEnum.WEB)

public ResponseVO doSupport(@PathVariable("id") Long id) {

try {

commentService.doSupport(id);

} catch (ZhydCommentException e) {

return ResultUtil.error(e.getMessage());

}

return ResultUtil.success("");

}

@PostMapping("/doOppose/{id}")

@BussinessLog(value = "点踩评论{1}", platform = PlatformEnum.WEB)

public ResponseVO doOppose(@PathVariable("id") Long id) {

try {

commentService.doOppose(id);

} catch (ZhydCommentException e) {

return ResultUtil.error(e.getMessage());

}

return ResultUtil.success("");

}

@PostMapping("/doPraise/{id}")

@BussinessLog(value = "点赞文章{1}", platform = PlatformEnum.WEB)

public ResponseVO doPraise(@PathVariable("id") Long id) {

try {

articleService.doPraise(id);

} catch (ZhydArticleException e) {

return ResultUtil.error(e.getMessage());

}

return ResultUtil.success("");

}

@PostMapping("/listNotice")

@BussinessLog(value = "公告列表", platform = PlatformEnum.WEB, save = false)

public ResponseVO listNotice() {

return ResultUtil.success("", noticeService.listRelease());

}

@PostMapping("/verifyArticlePassword")

@BussinessLog(value = "验证文章密码", platform = PlatformEnum.WEB, save = false)

public ResponseVO verifyArticlePassword(Long articleId, String password) {

if (StringUtils.isEmpty(password)) {

return ResultUtil.error("文章密码错误");

}

Article article = articleService.getByPrimaryKey(articleId);

if (null == article) {

return ResultUtil.error(String.format("文章【%s】不存在！", articleId));

}

if (!article.getPrivate() || StringUtils.isEmpty(article.getPassword())) {

return ResultUtil.error(String.format("文章【%s】未加密！", articleId));

}

if (article.getPassword().equals(password)) {

return ResultUtil.success("文章密码验证通过", article.getContent());

}

return ResultUtil.error("文章密码错误");

}

}

#### 附录4：保存文章浏览记录代码

package com.zyd.blog.runner;

import com.zyd.blog.core.schedule.ArticleLookTask;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.boot.ApplicationArguments;

import org.springframework.boot.ApplicationRunner;

import org.springframework.stereotype.Component;

/\*\*

\* 执行保存文章浏览记录任务

\*

\* @author kongchong

\* date: 2019-07-10 11:17

\*/

@Slf4j

@Component

public class TaskRunner implements ApplicationRunner {

@Value("${server.port}")

private int port;

@Autowired

private ArticleLookTask articleLookTask;

@Override

public void run(ApplicationArguments args) {

log.info("博客部署完成，博客访问地址：http://localhost:" + port);

articleLookTask.save();

}

}

#### 附录5：单条发送和批量发送代码

package com.zyd.blog.core.websocket.util;

import lombok.extern.slf4j.Slf4j;

import org.apache.commons.codec.Charsets;

import org.springframework.util.CollectionUtils;

import javax.websocket.Session;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.net.URLEncoder;

import java.util.Set;

/\*\*

\* websocket工具类，支持单条发送和批量发送

\*

\* @author yadong.zhang (yadong.zhang0415(a)gmail.com)

\* @version 1.0

\* @website https://docs.zhyd.me

\* @date 2018/4/18 11:48

\* @since 1.0

\*/

@Slf4j

public class WebSocketUtil {

private static final String ONLINE\_MSG\_KEY = "online";

private static final String NOTIFICATION\_MSG\_KEY = "notification";

private WebSocketUtil() {

// 私有化构造方法，禁止new

}

/\*\*

\* 根据消息类型，生成发送到客户端的最终消息内容

\*

\* @param type

\* 消息类型

\* @param content

\* 消息正文

\* @return

\*/

private static String generateMsg(String type, String content) {

return String.format("{\"fun\": \"%s\", \"msg\":\"%s\"}", type, content);

}

/\*\*

\* 发送在线用户的消息

\*

\* @param msg

\* @param sessionSet

\*/

public static void sendOnlineMsg(String msg, Set<Session> sessionSet) {

broadcast(generateMsg(ONLINE\_MSG\_KEY, msg), sessionSet);

}

/\*\*

\* 发送通知的消息

\*

\* @param msg

\* @param sessionSet

\*/

public static void sendNotificationMsg(String msg, Set<Session> sessionSet) throws UnsupportedEncodingException {

// 为了防止消息中存在特殊字符（比如换行符）等造成前台解析错误，此处编码一次。前台对应的需要解码

broadcast(generateMsg(NOTIFICATION\_MSG\_KEY, URLEncoder.encode(msg, Charsets.UTF\_8.displayName())), sessionSet);

}

/\*\*

\* 向客户端发送消息

\*

\* @param message

\* 消息内容

\* @param session

\* 客户端session

\* @throws IOException

\*/

private static void sendMessage(String message, Session session) {

try {

session.getAsyncRemote().sendText(message);

} catch (Exception e) {

log.error("[Socket] websocket-->向客户端发送数据发生异常", e);

}

}

/\*\*

\* 群发

\*

\* @param message

\* 消息内容

\* @param sessionSet

\* 客户端session列表

\* @throws IOException

\*/

private static void broadcast(String message, Set<Session> sessionSet) {

if (CollectionUtils.isEmpty(sessionSet)) {

return;

}

// 多线程群发

for (Session entry : sessionSet) {

if (null != entry && entry.isOpen()) {

sendMessage(message, entry);

} else {

sessionSet.remove(entry);

}

}

}

}