

**2020** 届本科生毕业设计（论文）

题 目 基于JavaWeb技术的博客管理系统的设计与实现

学 生 姓 名 王鑫

学 号 316202060944

专 业 班 级 软件工程1609班

学 院 计算机与软件工程学院

指 导 老 师 徐茂生

完 成 日 期 2020年4月30日

教务处 制

**摘 要**

博客，很多人称他为Web2.0的核心代表，刚出现便在网上流行起来。那什么是博客呢？简单一点来说，博客是记载了人们日常发生的事、自己的兴趣爱好，把自己的生活和想表达的东西与博客上的其他人分享、沟通，与此同时也可以通过博客结交更多趣味相投的朋友。越来越多的博客门户的出现，这让网民们看到了其蕴含的巨大的信息价值和潜力。现在博客已然成为很多人获取信息的主要途径。

身为一名程序员，浏览博客网站是每天必不可少的一环，所以我设计并开发了一个具有用户使用和管理员管理的多功能博客管理平台，这个平台不仅用户操作方便舒适，并且后台管理人员也能轻松管理整个平台。真正意义上实现了社交、管理相结合。

本系统是基于JavaWeb技术开发的，前端使用目前主流的layui、jQuery等框架，这些框架具有易上手、开源、轻量的特点，这些特点在中小型项目中是至关重要的。后端使用springboot、mybatis-plus、shiro等框架。Springboot的自动装配可以使项目简单、轻松的运行起来。mybatis-plus实现对mybatis的封装使得mybatis使用起来更加方便、快捷。shiro用来给项目提供权限管理功能，集中管理整个项目的权限分配和权限管理。

**关键词：**博客； springboot； mybatis-plus

**ABSTRACT**

Blog, which many people call the core representative of Web 2.0, has been popular on the Internet since its appearance. So what is a blog? Simply put, blog is to record people's daily events, their interests and hobbies, share and communicate their lives and things they want to express with other people on the blog, and at the same time, make more interesting friends through the blog. More and more blog portals appear, which let netizens see its huge information value and potential. Nowadays, blog has become the main way for many people to get information.

As a programmer, browsing blog website is an essential part of every day, so I designed and developed a multi-functional blog management platform with user use and administrator management. This platform is not only convenient and comfortable for users, but also easy for background managers to manage the whole platform. In a real sense, it realizes the combination of social and management.

The system is developed based on Java Web technology. The front-end uses the current mainstream layui, jQuery and other frameworks. These frameworks are easy to use, open-source and lightweight. These characteristics are very important in small and medium-sized projects. The back end uses springboot, mybatis plus, Shiro and other frameworks. Spring boot's automatic assembly can make the project simple and easy to run. The encapsulation of mybatis by mybatis plus makes it more convenient and quick to use. Shiro is used to provide permission management function for the project, and to centrally manage the permission assignment and permission management of the whole project.

**Key words:** blog； springboot； mybatis-plus

目 录

[1 引言 - 1 -](#_Toc40019130)

[1.1 研究背景及意义 - 1 -](#_Toc40019131)

[1.2发展现状及趋势 - 1 -](#_Toc40019132)

[1.3 论文组织结构 - 2 -](#_Toc40019133)

[2 系统需求分析 - 4 -](#_Toc40019134)

[2.1 可行性分析 - 4 -](#_Toc40019135)

[2.1.1 经济可行性 - 4 -](#_Toc40019136)

[2.1.2 技术可行性 - 4 -](#_Toc40019137)

[2.1.3 操作可行性 - 4 -](#_Toc40019138)

[2.2 系统需求分析 - 4 -](#_Toc40019139)

[2.2.1 业务流分析 - 4 -](#_Toc40019140)

[2.2.2 数据流分析 - 5 -](#_Toc40019141)

[2.2.3 数据字典 - 5 -](#_Toc40019142)

[2.2.4 数据库需求分析 - 6 -](#_Toc40019143)

[3 系统概要设计 - 8 -](#_Toc40019144)

[3.1 系统总体设计结构 - 8 -](#_Toc40019145)

[3.2 数据库表设计 - 8 -](#_Toc40019146)

[3.2.1 ER图 - 8 -](#_Toc40019147)

[3.2.2 数据表设计 - 9 -](#_Toc40019148)

[3.3 接口设计 - 13 -](#_Toc40019149)

[3.3.1 用户接口设计 - 13 -](#_Toc40019150)

[3.3.2 内部接口设计 - 14 -](#_Toc40019151)

[3.4 系统出错处理设计 - 19 -](#_Toc40019152)

[4 系统详细设计 - 21 -](#_Toc40019153)

[4.1 系统登录模块 - 21 -](#_Toc40019154)

[4.2 注册模块 - 23 -](#_Toc40019155)

[4.3 用户主页模块 - 25 -](#_Toc40019156)

[4.4 用户个人中心模块 - 27 -](#_Toc40019157)

[4.5 我的博客主页模块 - 29 -](#_Toc40019158)

[4.6 博客页面模块 - 31 -](#_Toc40019159)

[4.7 新增评论模块 - 32 -](#_Toc40019160)

[4.8 消息模块 - 34 -](#_Toc40019161)

[4.9 撰写消息模块 - 36 -](#_Toc40019162)

[4.10 相关博客模块 - 38 -](#_Toc40019163)

[4.11 新增博客模块 - 40 -](#_Toc40019164)

[4.12 用户管理模块 - 43 -](#_Toc40019165)

[4.13 修改角色模块 - 44 -](#_Toc40019166)

[4.14 栏目管理模块 - 46 -](#_Toc40019167)

[4.15 新增栏目模块 - 47 -](#_Toc40019168)

[4.16 博客管理模块 - 49 -](#_Toc40019169)

[5 系统测试 - 52 -](#_Toc40019170)

[5.1 系统测试概述 - 52 -](#_Toc40019171)

[5.2 系统测试方法 - 52 -](#_Toc40019172)

[5.3 系统测试用例 - 52 -](#_Toc40019173)

[5.4 系统测试总结 - 54 -](#_Toc40019174)

[结论与展望 - 55 -](#_Toc40019175)

[致 谢 - 56 -](#_Toc40019176)

[参考文献 - 57 -](#_Toc40019177)

插图清单

[图 2-1 博客管理系统业务流图 - 5 -](#_Toc39630875)

[图 2-2 博客管理平台数据流图 - 5 -](#_Toc39630876)

[图 3-1 博客管理系统总体结构设计功能图 - 8 -](#_Toc39630877)

[图 3-2 博客管理系统ER图 - 9 -](#_Toc39630878)

[图 4-1 登录实现流程图 - 21 -](#_Toc39630879)

[图 4-2 系统登陆图 - 22 -](#_Toc39630880)

[图 4-3 注册模块流程图 - 24 -](#_Toc39630881)

[图 4-4 注册模块界面 - 24 -](#_Toc39630882)

[图 4-5 用户主页流程图 - 26 -](#_Toc39630883)

[图 4-6 用户主页界面 - 26 -](#_Toc39630884)

[图 4-7 用户个人中心流程图 - 28 -](#_Toc39630885)

[图 4-8 用户个人中心界面 - 28 -](#_Toc39630886)

[图 4-9 我的播客主页模块流程图 - 30 -](#_Toc39630887)

[图 4-10 我的博客主页界面 - 30 -](#_Toc39630888)

[图 4-11 博客页面模块流程图 - 31 -](#_Toc39630889)

[图 4-12 博客页面界面 - 32 -](#_Toc39630890)

[图 4-13 新增评论模块流程图 - 32 -](#_Toc39630891)

[图 4-14 新增评论界面 - 33 -](#_Toc39630892)

[图 4-15 消息模块流程图 - 34 -](#_Toc39630893)

[图 4-16 消息模块界面 - 34 -](#_Toc39630894)

[图 4-17 撰写消息模块 - 36 -](#_Toc39630895)

[图 4-18 撰写消息界面 - 37 -](#_Toc39630896)

[图 4-19 相关博客模块流程图 - 38 -](#_Toc39630897)

[图 4-20 相关博客界面 - 38 -](#_Toc39630898)

[图 4-21 新增博客模块流程图 - 40 -](#_Toc39630899)

[图 4-22 新增博客模块界面 - 40 -](#_Toc39630900)

[图 4-23 用户管理模块流程图 - 42 -](#_Toc39630901)

[图 4-24 用户管理模块界面 - 42 -](#_Toc39630902)

[图 4-25 修改角色模块流程图 - 43 -](#_Toc39630903)

[图 4-26 修改角色模块界面 - 44 -](#_Toc39630904)

[图 4-27 栏目管理模块流程图 - 45 -](#_Toc39630905)

[图 4-28 栏目管理模块界面 - 45 -](#_Toc39630906)

[图 4-29 新增栏目模块流程图 - 46 -](#_Toc39630907)

[图 4-30 新增栏目模块界面 - 47 -](#_Toc39630908)

[图 4-31 博客管理模块流程图 - 48 -](#_Toc39630909)

[图 4-32 博客管理模块界面 - 48 -](#_Toc39630910)

表格清单

[表 2-1 用户数据模型表 - 6 -](#_Toc39631032)

[表 3-1 用户表 - 9 -](#_Toc39631033)

[表 3-2 博客表 - 10 -](#_Toc39631034)

[表 3-3 动态表 - 10 -](#_Toc39631035)

[表 3-4 栏目表 - 11 -](#_Toc39631036)

[表 3-5 收藏记录表 - 11 -](#_Toc39631037)

[表 3-6 点赞记录表 - 11 -](#_Toc39631038)

[表 3-7 阅读记录表 - 12 -](#_Toc39631039)

[表 3-8 评论表 - 12 -](#_Toc39631040)

[表 3-9 关注表 - 12 -](#_Toc39631041)

[表 3-10 消息表 - 13 -](#_Toc39631042)

[表 3-11 角色表 - 13 -](#_Toc39631043)

[表 3-12 用户登录接口设计 - 14 -](#_Toc39631044)

[表 3-13 用户注册接口设计 - 14 -](#_Toc39631045)

[表 3-14 用户管理接口设计 - 15 -](#_Toc39631046)

[表 3-15 博客管理接口设计 - 16 -](#_Toc39631047)

[表 3-16 栏目管理接口设计 - 16 -](#_Toc39631048)

[表 3-17 评论管理接口设计 - 17 -](#_Toc39631049)

[表 3-18 关注管理接口设计 - 17 -](#_Toc39631050)

[表 3-19 消息管理接口设计 - 18 -](#_Toc39631051)

[表 3-20 上传管理接口设计 - 19 -](#_Toc39631052)

[表 3-21 博客es接口设计 - 19 -](#_Toc39631053)

[表 3-22 出错处理设计表 - 20 -](#_Toc39631054)

[表 5-1 测试项目表 - 50 -](#_Toc39631055)

[表 5-2 测试项目需求表 - 50 -](#_Toc39631056)

[表 5-3 测试项目用例表 - 51 -](#_Toc39631057)

**基于JavaWeb技术的博客管理系统的设计与实现**

（计算机与软件工程学院 20届 软件工程1609班） 指导教师：徐茂生

**1 引言**

**1.1 研究背景及意义**

1998年，博客在网上出现，直到2000年才开始流行起来[1]。此时博客正式进入中国，并且发展速度极快，但成绩始终趋于平淡，没有很好的效果。最开始，博主只是每天分享着自己每天的浏览心得和意见。但由于Blog的极速发展，Blog的目的性已经发生了天翻地覆的变化。每个博主发表Blog的目的性也有着很大的不同，博客一步步的改变着人们的交流方式，发表博客分享生活或者推销自己等等，每个人有着属于自己风格的博客，形成了拥有独立性的博客文化[2]。网上有很多博客平台，这些平台五花八门，有的使用起来繁琐，有的缺乏社交性，所以我设计了这个博客管理平台，它不仅使用方便还具有社交性和传播性。

Blog目前主要在三个方面应用火热：一、以推销自己为目的知识收集平台；二、一种新型人际交流模式；三、一个知识管理平台或以个人为中心的知识、信息管理平台。Blog可以成为每个人的学习工具，使用方便，大量吸引着网民的兴趣。我设计的平台更偏向于用户之间的交流、分享和在管理上方便易用。是人际交流模式和知识管理共存的多功能博客平台[3]。

现在Blog在商业、公司内部、校园领域、教育的实力不容小觑，它强化了互联网过去的很多功能，如：资料分享、通讯、交流等，把信息融为一个圈供里面的人分享，收纳。Blog已然成为人们提高自身的一种重要方式，一个优秀的博客平台更是能体现其特点[4]。

博客的发展与壮大对我国信息时代具有添砖加瓦的意义。希望我的博客平台在提升个人编码能力的同时，也能紧随时代的步伐，给网民提供一个交流、分享的平台。

**1.2发展现状及趋势**

博客的发展可以说是十分迅速，各行各业多多少少都有涉及。随着需求的增长，博客数量的增长是呈几何式的[5]。这些博客大多是由个人创建，在传递各类新闻事件、分享生活、分享知识心得等过程中起到了不可忽视的作用，为人们提供了一种新的知识分享，生活分享，新闻分享的平台[6]。

但是相比较发展更是迅猛的微博，博客的发展速度无法与微博相比。微博的便捷性，极低的门槛是博客无法办到的[7]。但博客相比微博更有目的性，用户粘性更大，人们会在博客平台找到属于自己的小圈子，在其中分享和收获其他兴趣相同的用户的分享。

博客甚至可以被称为是个人自媒体，热门、受关注度够高的个人博客甚至可以拥有一定的媒体影响力，在各个领域具有高度影响力的个人博客更是人们青睐的对象。乐于分享生活的人也可以在个人博客找到属于自己的小天地，他们不需要多高的知名度，他们只需要一个共同的爱好、共同的话题，这些共同点是博客最不可缺少、最精华、存粹的部分。

目前网上比较火热的博客平台基本上都是.NET技术和C语言技术搭建，但是Java一直都是网页平台搭建的选项之一，Java的平台无关性使项目可以跨平台运行，这不仅为维护项目提供了方便，也为平台编码者提供了方便。这也是本系统核心技术使用Java的原因，同时采用MySql数据库、springboot、mybatis-plus框架也给项目带来的新活力，这些技术都是开源的，方便学习，使开发者更容易地上手和开发项目。

**1.3 论文组织结构**

本文共5章，论文的组织结构及各章节介绍如下：

第一章为引言，对研究背景和意义做了描述，同时对国内外相关的系统现状进行介绍，并提出还没有解决的问题，以此来引出自己所写的毕业设计的研究意义，同时介绍自身选用的相关技术和设计所要达到的预期要效果。

第二章系统需求分析：从各个方面对博客管理系统进行详细的业务流程、可行性分析，以此熟悉系统的业务流程和完成可能性。

第三章系统概要设计：主要介绍系统的整体结构设计，数据库设计，然后再对系统的接口的设计进行统一介绍。

第四章系统详细设计：主要介绍系统各个模块的详细设计，包括用例图、界面、核心代码。

第五章系统测试：主要介绍如何对系统做一个详细的测试，说明整个系统的测试方法，以及系统测试的完成情况。

本次毕业设计中，根据需求分析、概要设计、详细设计以及系统测试这几个方向进行思考和实践，这一过程大大提升了个人分析能力与系统设计能力。

**2 系统需求分析**

**2.1 可行性分析**

在软件开发过程中对软件进行可行性分析是一个重要步骤。可行性分析主要从经济、技术、操作等方面分析所给出的解决方案是否可行，软件的可行性分析是通过对项目的市场需求、环境影响盈利能力等方面的研究分析[8]。

**2.1.1 经济可行性**

软件的经济可行性意味着有没有足够的资金来完成开发，同时也要分析这个软件能否给用户带来实际经济效益。本系统是为了用户可以方便的操作博客，在博客平台和其他用户分享交流，是个人独立研究设计，只是消耗时间成本，并不需要其他太多的消耗，所以，本系统的开发设计在经济上是完全可行的，不存在经济可行性问题[9]。

**2.1.2 技术可行性**

本系统是开发博客管理平台，用户通过自身新建博客、发表博客与其他用户产生联系、交流。另一方面，系统采用目前互联网主流的开发工具和开发框架，这些工具和框架都是通过开源的方式在互联网传播，技术本身并不复杂，根据自己的学习就可以基本掌握这些工具和框架的基本使用，从而达到开发项目的能力。所以，本系统不存在技术可行性问题。

**2.1.3 操作可行性**

本系统只需输入自身游戏和密码即可注册账号进入平台，系统是传统的web项目，符合网上所有网站简单方便的特点，用户可以轻松的实现注册、登录、创建博客、关注其他用户、评论博客等操作，用户不需要输入太多的个人信息。进入系统后，系统有相关提示系统提醒用户当前应该做什么、怎样做。所以，本系统不存在操作可行性问题。

**2.2 系统需求分析**

**2.2.1 业务流分析**

博客管理平台用户首先通过邮箱注册账号，使用账号密码登陆系统[10]。登录后可以对博客进行增删改查、给博客进行评论、发送消息、接收消息、关注用户、搜一搜等操作，管理员可以进行用户管理、博客管理、栏目管理等操作，操作过程中的数据都会保存到数据库中。博客管理系统业务流图如图2-1所示

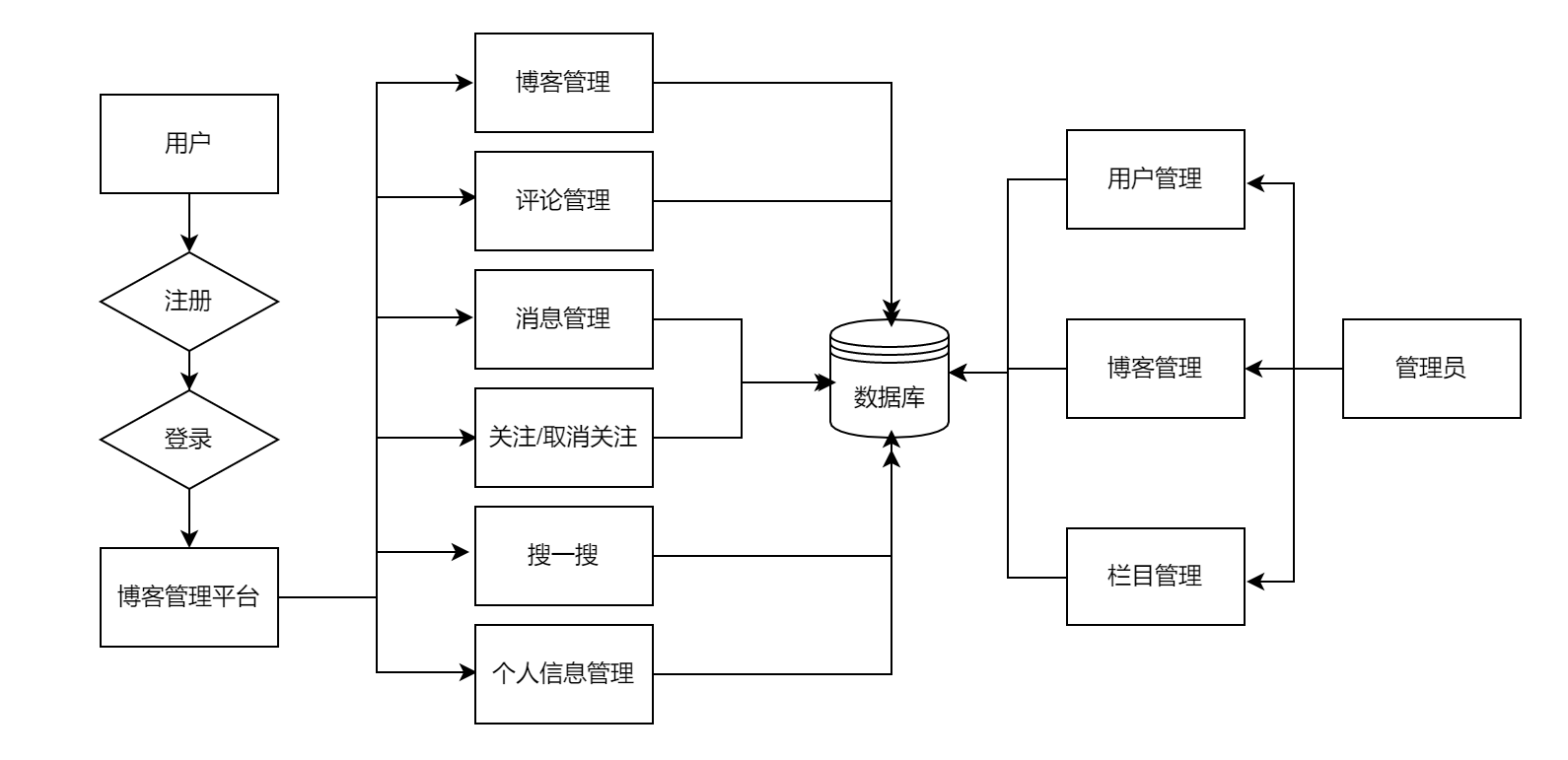


图 2-1 博客管理系统业务流图

**2.2.2 数据流分析**

博客管理系统的数据流是依赖用户对系统的操作来进行流向的，用户使用邮箱注册账号就是向系统输入数据，像博客管理、评论管理、消息管理、关注/取消关注、搜一搜、个人信息管理这些操作的数据都是双向同步的，只有用户进行操作，才会输出相应的数据，管理员对系统的管理数据也是双向同步的。博客管理系统数据流图如图2-2所示。

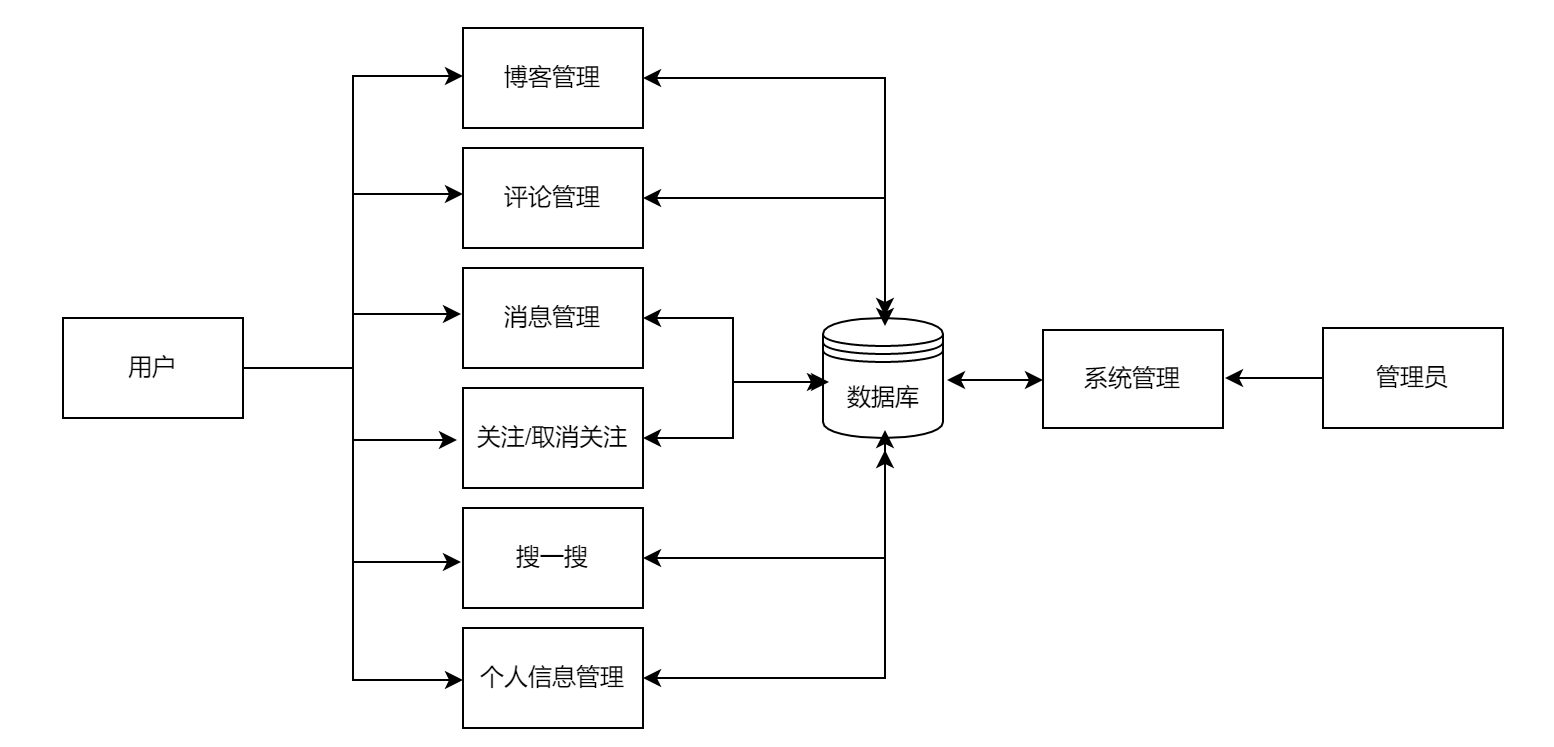


图 2-2 博客管理平台数据流图

**2.2.3 数据字典**

数据字典是一种通用的程序设计方法，数据字典标准是以概念数据模型为基础，提供基础数据集的空间与层次要素的标准定义，如数据存储、数据处理逻辑、数据流、数据结构、外部实体等。其功能可以将数据的各个元素做出详细描述。数据字典可称为数据的信息描述集合，方便访问的数据库目录[11]。被动数据字典是指修改时必须手工更新其内容的数据字典[12]。

博客管理系统的用户数据字典，是由用户详细信息组成的，这些详细信息保存在数据库中。如表2-1所示。

表 2-1 用户数据模型表

|  |  |
| --- | --- |
| 数据项存储名 | 用户信息 |
| 数据项来源 | 用户信息表 |
| 数据项去向 | 用户邮箱注册登录模块 |
| 数据项组成 | 用户信息=邮箱+用户名+ +激活秘钥+激活状态+密码 |
| 数据项描述 | 保存了用户的基本信息 |

**2.2.4 数据库需求分析**

博客管理系统使用了MySQL数据库，轻量级、开源等特点使他非常适合成为本次毕设的数据库。身为一种关系数据库管理系统，根据不同的数据将数据保存在不同的表中，大大增加了使用速度并且提高了使用灵活性[13]。对于个人使用者来讲，MySQL提供的功能已经足够了[14]。

通过对博客管理系统的功能需求分析、数据库需求分析，博客管理系统有如下功能需求信息：

管理员功能需求分析：

(1)用户管理：更改用户角色，禁用/启用用户账号，并把处理结果返回给用户；

(2)博客管理：下架博客，推荐/取消推荐博客，并把处理结果返回给用户；

(3)栏目管理：新增/修改/删除栏目。

用户功能需求分析：

(1)查看个人信息：用户可以查看自己的详细信息；

(2)编辑个人信息：用户可以编辑自己的个人信息；

(3)查看发表博客：用户可以查看自己发表的博客；

(4)查看草稿博客：用户可以查看自己设为草稿的博客；

(5)发表博客：用户可以发表自己的博客；

(6)撰写评论：用户可以给博客写评论；

(7)发送消息：用户可以给其他用户发送消息；

(8)查看收信箱：用户可以查看其他用户发送来的信息；

(9)查看发信箱：用户可以自己发送过的信息；

(10)查看未读信息：用户可以根据自己还未阅读的信息；

(11)使用搜一搜：用户可以使用搜一搜全局分词搜索博客；

(12)点赞博客：用户可以点赞自己觉得写的优秀的博客；

(13)收藏博客：用户可以收藏自己觉得值得收藏的博客；

**3 系统概要设计**

**3.1 系统总体设计结构**

博客管理系统总体分为6个大模块：用户对博客的管理模块，用户对自身个人信息管理的模块，用户对博客进行评论模块，管理员管理博客模块，管理员管理用户模块，用户收藏点赞博客模块，结构设计功能图如图3-1所示。

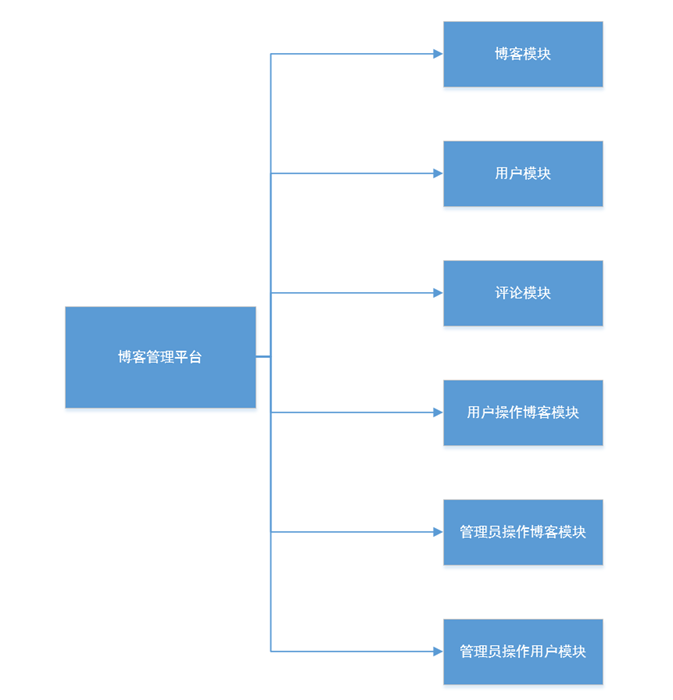


图 3-1 博客管理系统总体结构设计功能图

**3.2 数据库表设计**

**3.2.1 ER图**

博客管理系统的E-R图如图3-2所示。

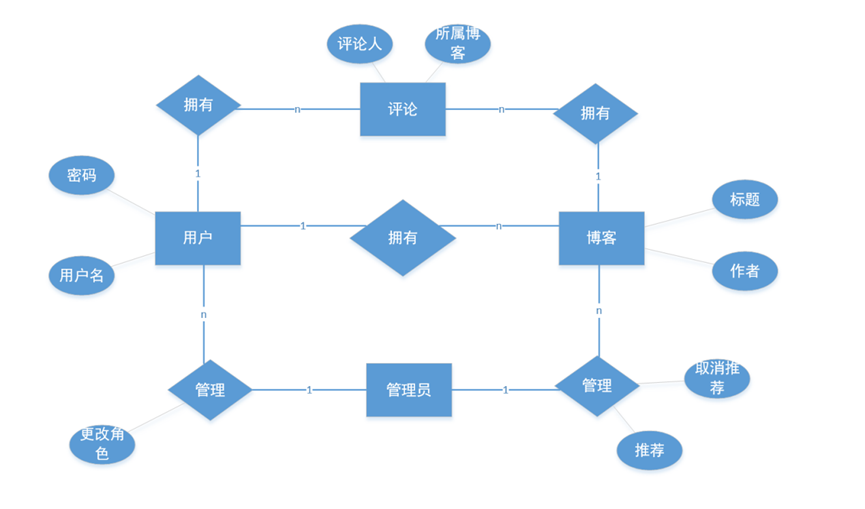


图 3-2 博客管理系统ER图

**3.2.2 数据表设计**

本系统在设计数据库时首先根据需求绘制E-R图，再将E-R图转换为多张表，系统数据库名blogsystem，其表设计分别如下所示。

（1）用户表设计，如表3-1所示。

表 3-1 用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| username | varchar(20) | N |  |  | 用户名 |
| email | varchar (50) | N |  |  | 邮箱 |
| password | varchar (50) | N |  |  | 密码 |
| name | varchar (50) | N |  |  | 姓名 |
| role\_code | varchar (50) | N |  |  | 角色code |
| follow\_num | int | N |  |  | 粉丝数 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |
| create\_time | datetime | N |  |  | 创建时间 |
| pic\_url | varchar (50) | N |  |  | 头像路径 |
| activity\_status | varchar (50) | N |  |  | 激活状态 |
| activity\_code | varchar (50) | N |  |  | 激活码 |

（2）博客表设计，如表3-2所示。

表 3-2 博客表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| title | varchar(50) | N |  |  | 标题 |
| author | varchar (50) | N |  |  | 作者Id |
| author\_name | varchar (50) | N |  |  | 作者用户名 |
| content | text | N |  |  | 内容 |
| summary | varchar (50) | N |  |  | 摘要 |
| create\_time | datetime | N |  |  | 创建时间 |
| update\_time | datetime | N |  |  | 更新时间 |
| release\_time | datetime | N |  |  | 发表时间 |
| status | varchar (50) | N |  |  | 状态 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |
| collect\_num | int | N |  |  | 收藏数 |
| comment\_num | int | N |  |  | 评论数 |
| view\_num | int | N |  |  | 浏览数 |
| label | varchar (50) | N |  |  | 标签 |
| like\_num | int | N |  |  | 点赞数 |
| is\_recommend | varchar (50) | N |  |  | 推荐标识 |
| category | varchar (50) | N |  |  | 栏目 |

（3）动态表设计，如表3-3所示。

表 3-3 动态表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| category | varchar(20) | N |  |  | 类型 |
| blog\_id | varchar (36) | N |  |  | 博客id |
| blog\_title | varchar (36) | N |  |  | 博客标题 |
| content | varchar (255) | N |  |  | 内容 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |
| create\_time | datetime | N |  |  | 创建时间 |
| user\_id | varchar (36) | N |  |  | 用户id |

（4）栏目表设计，如表3-4所示。

表 3-4 栏目表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| name | varchar(50) | N |  |  | 名称 |
| create\_time | datetime | N |  |  | 创建时间 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |

（5）收藏记录表设计，如表3-5所示。

表 3-5 收藏记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| user\_id | varchar(50) | N |  |  | 用户id |
| blog\_id | varchar (50) | N |  |  | 博客id |
| create\_time | datetime | N |  |  | 创建时间 |
| del\_tag | varchar(50) | N |  |  | 删除标识 |

（6）点赞记录表设计，如表3-6所示。

表 3-6 点赞记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| user\_id | varchar(50) | N |  |  | 用户id |
| blog\_id | varchar (50) | N |  |  | 博客id |
| create\_time | datetime | N |  |  | 创建时间 |
| del\_tag | varchar(50) | N |  |  | 删除标识 |

（7）阅读记录表设计，如表3-7所示。

表 3-7 阅读记录表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| user\_id | varchar(50) | N |  |  | 用户id |
| blog\_id | varchar (50) | N |  |  | 博客id |
| create\_time | datetime | N |  |  | 创建时间 |
| del\_tag | varchar(50) | N |  |  | 删除标识 |

（8）评论表设计，如表3-8所示。

表 3-8 评论表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| content | varchar(255) | N |  |  | 内容 |
| sender | varchar (50) | N |  |  | 发送人id |
| sender\_name | varchar (50) | N |  |  | 发送人用户名 |
| receiver | varchar (50) | N |  |  | 接收人id |
| receiver\_name | varchar (50) | N |  |  | 接收人用户名 |
| blog\_id | varchar (50) | N |  |  | 博客id |
| del\_tag | varchar (50) | N |  |  | 删除标识 |
| create\_time | datetime | N |  |  | 创建时间 |

（9）关注表设计，如表3-9所示。

表 3-9 关注表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| follower | varchar (50) | N |  |  | 关注人id |
| follower\_name | varchar (50) | N |  |  | 关注人用户名 |
| be\_follower | varchar (50) | N |  |  | 受关人id |
| be\_follower\_name | varchar (50) | N |  |  | 受关人用户名 |
| follower\_pic\_url | varchar (50) | N |  |  | 关注人头像 |
| be\_follower\_pic\_url | varchar (50) | N |  |  | 受关人头像 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |
| create\_time | datetime | N |  |  | 创建时间 |

（10）消息表设计，如表3-10所示。

表 3-10 消息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| title | varchar (50) | N |  |  | 标题 |
| content | varchar (50) | N |  |  | 内容 |
| type | varchar (50) | N |  |  | 类型 |
| receiver | varchar (50) | N |  |  | 接收人 |
| Receiver\_name | varchar (50) | N |  |  | 接收人用户名 |
| sender | varchar (50) | N |  |  | 发件人 |
| Sender\_name | varchar (50) | N |  |  | 发件人用户名 |
| create\_time | datetime | N |  |  | 创建时间 |
| Del\_tag\_re | varchar (50) | N |  |  | 收件删除标识 |
| Is\_read | varchar (50) | N |  |  | 阅读标识 |
| Del\_tag\_se | varchar (50) | N |  |  | 发件删除标识 |

（11）角色表设计，如表3-11所示。

表 3-11 角色表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否为空 | 默认值 | 唯一 | 备注 |
| id | varchar(36) | N |  | Y | Id |
| name | varchar(50) | N |  |  | 名称 |
| code | varchar (50) | N |  |  | code |
| create\_time | datetime | N |  |  | 创建时间 |
| del\_tag | varchar (50) | N |  |  | 删除标识 |

**3.3 接口设计**

**3.3.1 用户接口设计**

用户界面部分：通过需求分析，提供给用户一个方便操作、观感舒适的界面，方便用户使用系统。在界面方面，简单明了、简易操作是最基本的要求，同时页面布局也很重要，操作后的反馈，可互动部分的合理性，出错时的系统提示都应该重点突出。在用户接口方面也要做到操作简单、方便管理。设计上应该实现尽可能的解耦，以防止改一处而动全身。总的来说，系统的用户界面应作到可靠性、简单性、易学习和使用[15]。

**3.3.2 内部接口设计**

（1）用户登录接口设计，如表3-12所示。

表 3-12 用户登录接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| LoginController | login(String email,String username,String password) | 用username字段接受用户输入的用户名或者密码，通过正则表达式判断输入的是邮箱还是用户名，调用service方法验证用户输入的账号密码是否正确，根据是否通过验证返回相应提示信息到页面。 |
| Shiro | SecurityUtils.getSubject() | 实例化AuthenticationToken创建实例对象  调用SecurityUtils的Login方法完成系统登录。 |

（2）用户注册接口设计，如表3-13所示。

表 3-13 用户注册接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| RegisterController | register(User user) | 前端传来用户输入的用户信息，调用registerService的register方法，保存用户信息到数据库，给予用户一个未激活状态和一个激活码，发送激活链接到用户的邮箱，用户点击邮箱链接激活用户。 |
| checkCode(String code) | 点击邮箱中链接，调用checkCode接口，通过点击链接获取激活码，使用此码与数据库中用户存储激活码相比较，相同即时解除用户禁用状态。 |

（3）用户管理接口设计，如表3-14所示。

表 3-14 用户管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| UserController | queryList(Page<User> page, User user) | 根据Page类接收分页信息，User类为查询条件，根据查询条件查询相应用户列表，返回到页面。 |
| topBlogUserList(Page<User> page,Model model) | 调去userService中的topBlogUserList方法根据热度排序获取博主列表。 |
| goPersonPage(String id,Model model) | 根据id获取用户信息，返回用户信息到个人中心页面。 |
| goHeadPicPage(String id,Model model) | 跳转到用户更改头像页面。 |
| goUserChangeRole(String userId,Model model) | 跳转到用户更改角色页面。 |
| userChangeRole(String userId,String roleCode) | 根据id获取用户信息，更改用户角色code保存到数据库，返回提示信息。 |

（4）博客管理接口设计，如表3-15所示。

表 3-15 博客管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| BlogController | getBlogList(Page<Blog> page,String id, Model model) | 根据Page类接收分页信息，根据id获取该用户id下所发布的所有博客。 |
| saveOrUpdate(Blog blog) | 新增或修改博客。 |
| getTopViewList(Page<Blog> page, Model model) | 获取阅读排行榜博客列表，返回到相应页面。 |
| getAllBlogList(Page<Blog> page, Model model) | 获取首页所有已发布博客列表。 |
| getTenDaysTopLikeBlog(Page<Blog> page,Model model) | 获取10天内点赞数排行榜列表。 |
| get48HoursViewBlogList(Page<Blog> page, Model model) | 获取两天内阅读排行榜列表。 |

（5）栏目管理接口设计，如表3-16所示。

表 3-16 栏目管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| BlogCategoryController | goBlogCategoryListPage() | 跳转至栏目管理列表页面。 |
| goAddPage(String id,Model model) | 跳转至栏目新增页面。 |
| selectAll4Front(Page<BlogCategory> page, Model model) | 首页获取栏目列表。 |
| selectAll(Page<BlogCategory> page) | 栏目管理页面获取所有栏目。 |
| saveOrUpdate(BlogCategory blogCategory) | 管理页面新增修改栏目。 |

（6）评论管理接口设计，如表3-17所示。

表 3-17 评论管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| CommentController | addComment(Comment comment) | 新增评论接口。 |
| getPage(Page<Comment> page, String blogId, Model model) | 根据博客id获取该博客所有评论列表。 |
| deleteComment(String id) | 根据评论id删除评论。 |

（7）关注管理接口设计，如表3-18所示。

表 3-18 关注管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| FollowController | follow(Follow follow) | 从前端获取Follow实体类，存入数据库实现关注。 |
| cancelFollow(Follow follow) | 从前端获取Follow实体类，存入数据库实现取消关注。 |
| getBeFollowerList4Side(Page<Follow> page,String id,Model model) | 个人中心页面侧边栏获取粉丝列表。 |
| getFollowerList4Side(Page<Follow> page,String id,Model model) | 个人中心页面侧边栏获取关注列表。 |
| getBeFollowerList(Page<Follow> page,String id,Model model) | 个人中心页面分页获取所有粉丝列表。 |
| getFollowerList(Page<Follow> page,String id,Model model) | 个人中心页面分页获取所有关注列表。 |

（8）消息管理接口设计，如表3-19所示。

表 3-19 消息管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| MsgController | addMsg(Msg msg) | 前端获取Msg对象，保存到数据库，新增消息。 |
| inList(Page<Msg> page, Model model) | 获取收信箱列表。 |
| outList(Page<Msg> page,Model model) | 获取发信箱列表。 |
| unReadList(Page<Msg> page,Model model) | 获取未读信息列表。 |

（9）上传管理接口设计，如表3-20所示。

表 3-20 上传管理接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| UploadController | uploadImg(@RequestParam("myFile") MultipartFile multipartFile, HttpServletRequest request, String username) | 获取前端上传的图片，通过虚拟路径储存到本地文件夹。 |
| uploadBlogImg(@RequestParam("myFile") MultipartFile multipartFile, HttpServletRequest request) | 获取前端上传的博客图片，通过虚拟路径储存到本地文件夹。 |

（10）博客es接口设计，如表3-21所示。

表 3-21 博客es接口设计

|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 说明 |
| BlogEsController | goAllSearch(Model model) | 跳转到全局搜索页面。 |
| goSearchList(Model model, String str) | 跳转到全局列表页面。 |
| searchAll(String str, Pageable pageable,Model model, Integer likeLimit, Integer viewLimit, Date dateLimit) | 根据条件分词搜索博客。 |

**3.4 系统出错处理设计**

本模块主要是针对出错的一些信息的处理方式，如表3-22所示。

表 3-22 出错处理设计表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **出错名称** | **系统输出信息** | **处理方法** |
| 1 | 用户名输入错误 | “不存在该用户！” | 进入登录页面。 |
| 2 | 密码错误 | “您的密码输入有误，请重新输入！” | 进入登录页面 |
| 3 | 无权限 | “您没有该操作的权限，请联系管理员” | 返回原页面 |
| 4 | 输入数据格式错误 | “输入格式不正确，请检查后再输入！” | 返回原输入页面 |
| 5 | 不允许为空的输入框输入空字符 | “请输入必填项!” | 弹回原输入页面的输入处 |
| 6 | 操作失败 | “该操作异常，请稍后再试！” | 返回原页面 |
| 7 | 系统故障 | “服务器维护中！暂停服务！” | 立即启用备用机，恢复故障 |

**4 系统详细设计**

**4.1 系统登录模块**

（1）模块流程图

本模块为用户管理员通用登录模块，输入者通过输入账号、密码进入系统。系统登录实现流程图如图4-1所示。

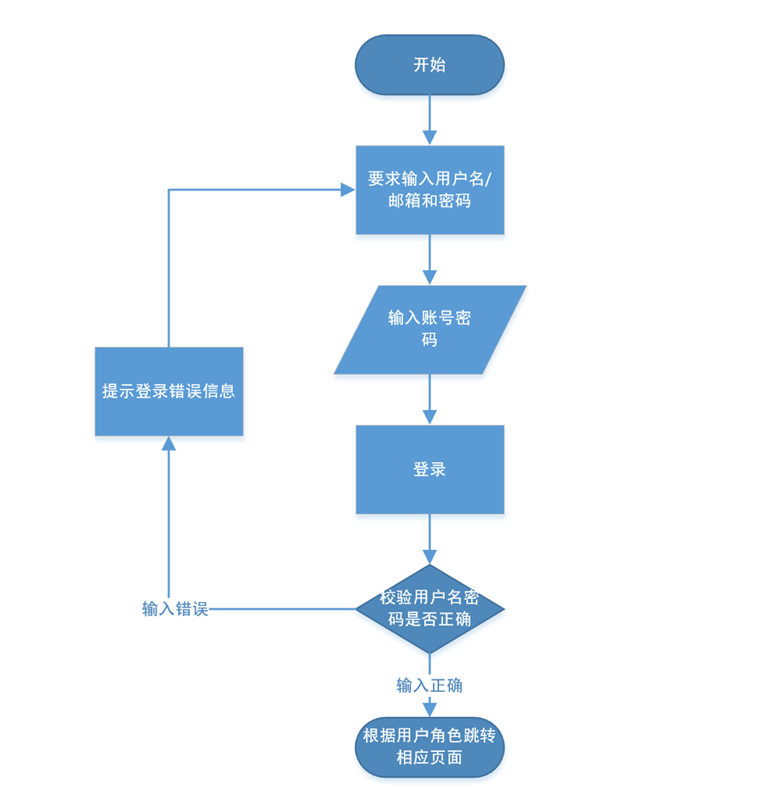


图 4-1 登录实现流程图

（2）模块界面展示

系统登录首页如图4-2所示。

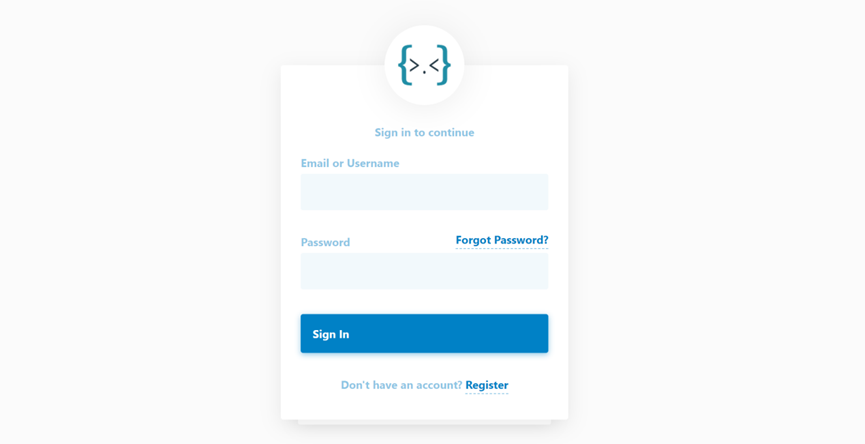


图 4-2 系统登陆图

（3）核心代码展示

先用SecurityUtils获取用户实例，通过输入的用户名，密码进行比较登录。

public String login(String username, String password) {

if(StringUtils.isEmpty(username))

{

return JSONUtil.error("请填写用户名/邮箱");

}

String msg="";

// 1、获取Subject实例对象

Subject currentUser = SecurityUtils.getSubject();

// 3、将用户名和密码封装到UsernamePasswordToken

//String md5Password = Encript.md5(password);

UsernamePasswordToken token = new UsernamePasswordToken(username,

password);

// 4、认证

try {

currentUser.login(token);// 传到MyAuthorizingRealm类中的方法进行认证

User user = (User) currentUser.getPrincipal();

if(user.getActivityStatus().equals(SysCode.USER\_ACTIVITY\_STATUS.WJH)){

return JSONUtil.error("账号处于未激活状态，请前往邮箱激活");

}

Session session = currentUser.getSession();

session.setAttribute("username", username);

return JSONUtil.success();

//return "/index";

}catch (UnknownAccountException e)

{

e.printStackTrace();

msg = "账号不存在";

}

catch (IncorrectCredentialsException e)

{

msg = "密码不正确";

}

catch (AuthenticationException e) {

e.printStackTrace();

msg="用户验证失败";

}

return JSONUtil.error(msg);

}

**4.2 注册模块**

（1）模块实现流程图

通过注册模块，输入用户名、邮箱、密码进行账号注册。注册模块流程图如图4-3所示。

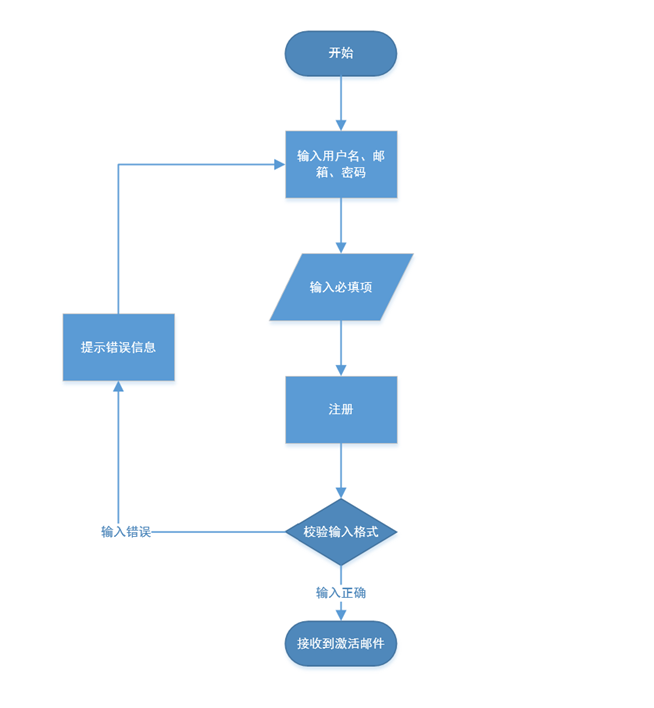


图 4-3 注册模块流程图

（2）模块界面展示

注册模块页面如图4-4所示。

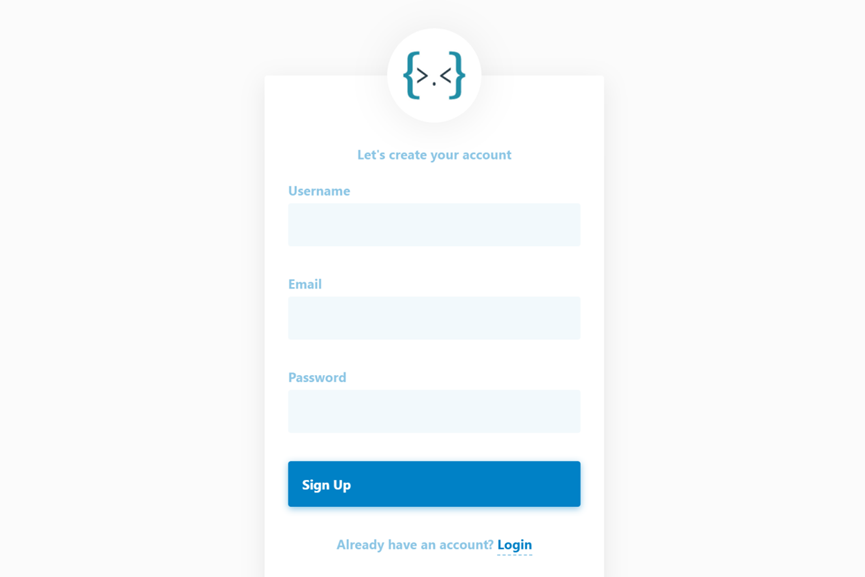


图 4-4 注册模块界面

（3）核心代码展示

通过前台接收用户对象并存入数据库，再调用邮箱发送接口进行激活口令发送。

public void register (User user) {

String activeCode = UUID.randomUUID().toString();

user.setActivityCode(activeCode);

user.setActivityStatus(SysCode.USER\_ACTIVITY\_STATUS.WJH);

user.setDelTag(SysCode.DELTAG.WSC);

user.setRoleCode(SysCode.ROLECODE.PTYH);

user.setCreateTime(new Date());

user.setPassword(Encript.md5(user.getPassword()));

userMapper.insert(user);

//主题

String subject = "来自博客网站的激活邮件";

//上面的激活码发送到用户注册邮箱

String context = "<a

href=\"http://localhost:8080/checkCode.do?code="+activeCode+"\">激活请点击:"+activeCode+"</a>";

mailService.sendMimeMail(user.getEmail(),subject,context);

}

@PostMapping(value = "/register.do")

@ResponseBody

public String register(User user){

loginService.register(user);

return JSONUtil.success("注册成功，请前往邮箱激活账号");

}

**4.3 用户主页模块**

（1）模块实现流程图

非管理员用户登录进入博客门户主页面。用户主页模块流程图如图4-5所示。

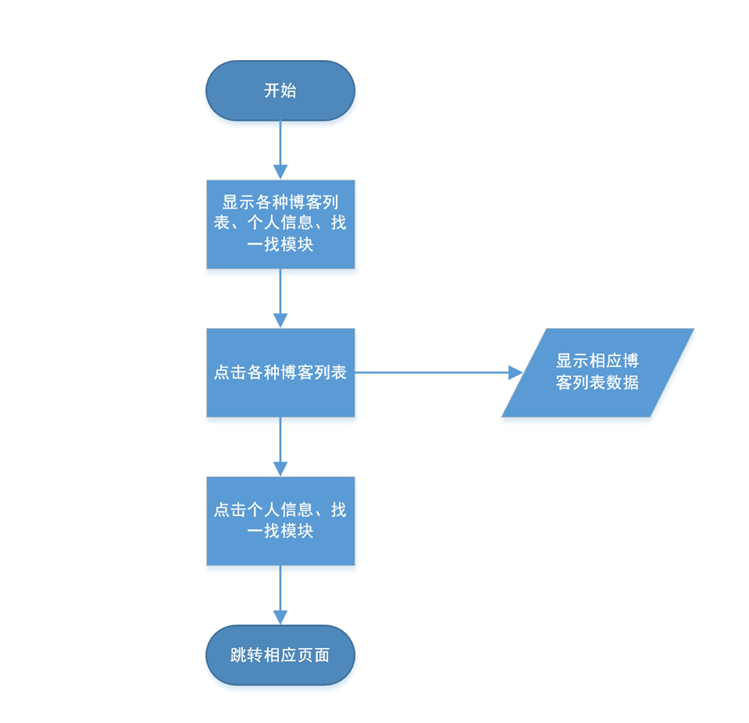


图 4-5 用户主页流程图

（2）模块界面展示

用户主页页面如图4-6所示。



图 4-6 用户主页界面

（3）核心代码展示

通过不同条件调用不同sql去数据获取博客列表数据。

@Select("SELECT sys\_blog.\*,user.pic\_url `authorPicUrl` FROM sys\_blog,user " +

"WHERE sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

"order by sys\_blog.release\_time desc")

IPage<Blog> selectAll(Page<Blog> page);

@Select("select \* from sys\_blog where id in (\n" +

"select tt.id from (\n" +

"select blog\_id \"id\" from sys\_blog\_view \n" +

"WHERE

(TIME\_TO\_SEC(TIMEDIFF(SYSDATE(),sys\_blog\_view.create\_time)))/3600<=48 \n" +

"GROUP BY blog\_id order by count(sys\_blog\_view.id) desc LIMIT 10 ) tt\n" +

"\n" +

")")

IPage<Blog> get48HoursViewBlogList(Page<Blog> page);

@Select("select sys\_blog.\*,user.pic\_url `authorPicUrl` from sys\_blog,user where sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

"and sys\_blog.id in (" +

"select tt.blog\_id from" +

"(" +

"select DISTINCT(blog\_id) from sys\_comment where

sender = #{userId} " +

") tt"+

")")

IPage<Blog> getMyCommentList(Page<Blog> page,@Param("userId")String userId);

**4.4 用户个人中心模块**

（1）模块实现流程图

用户进入个人中心可以查看自己最新的动态、个人资料、关注的人、粉丝、与自己相关的博客和自己的博客等。用户个人中心模块流程图如图4-7所示。

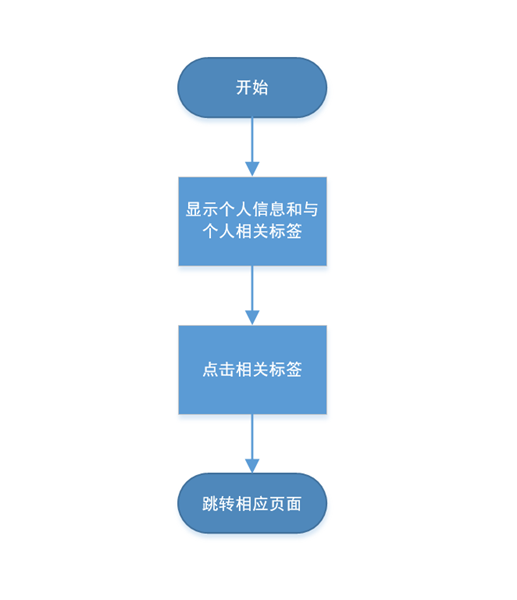


图 4-7 用户个人中心流程图

（2）模块界面展示

用户个人中心页面如图4-8所示。

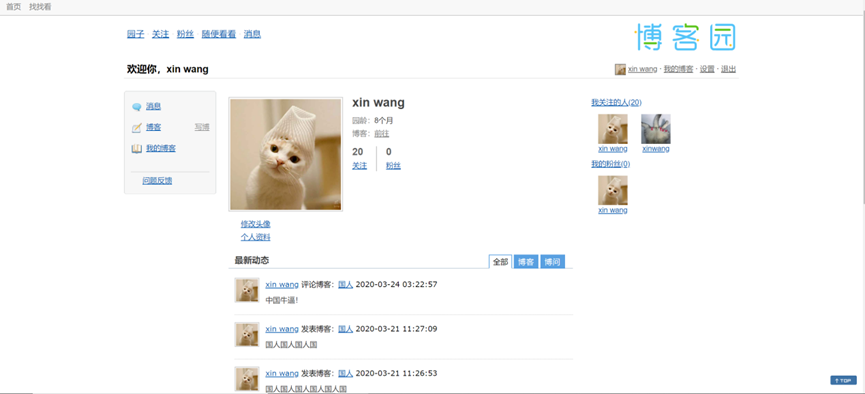


图 4-8 用户个人中心界面

（3）核心代码展示

分别创建QueryWrapper对象获取动态列表，关注者列表，粉丝列表。

@Override

public List<Activity> getActitvityList(Page<Activity> page,String id) {

QueryWrapper<Activity> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("user\_id",id);

queryWrapper.orderByDesc("create\_time");

IPage<Activity> activityIPage = activityMapper.selectPage(page,queryWrapper);

return activityIPage.getRecords();

}

@Override

public IPage<Follow> getBeFollowerList4Side(Page<Follow> page,String id) {

QueryWrapper<Follow> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("del\_tag",SysCode.DELTAG.WSC);

queryWrapper.eq("follower",id);

return followMapper.selectPage(page,queryWrapper);

}

@Override

public IPage<Follow> getFollowerList4Side(Page<Follow> page, String id) {

QueryWrapper<Follow> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("del\_tag",SysCode.DELTAG.WSC);

queryWrapper.eq("be\_follower",id);

return followMapper.selectPage(page,queryWrapper);

}

**4.5 我的博客主页模块**

（1）模块实现流程图

显示所有用户发表的博客，点击列表中博客，进入博客页面。点击上方工具栏返回主页和个功能页面。右侧显示博客公告，相关搜索，各功能按钮。我的博客主页模块流程图如图4-9所示。

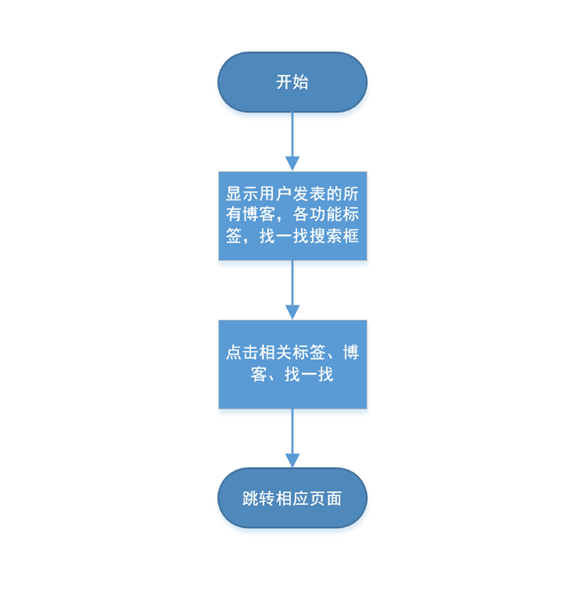


图 4-9 我的播客主页模块流程图

（2）模块界面展示

我的博客页面如图4-10所示。

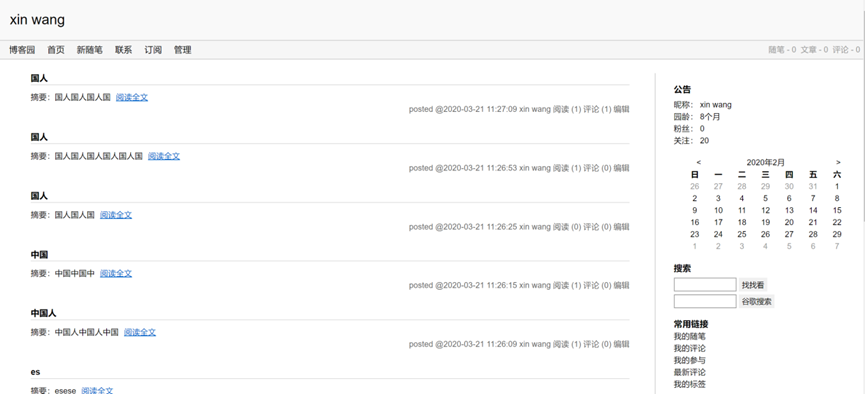


图 4-10 我的博客主页界面

（3）核心代码展示

通过Page对象实现分页查询，获取该userId下的所有已发布博客列表。

public IPage<Blog> selectPageByUserId(Page<Blog> page, String userId) {

QueryWrapper<Blog> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("author",userId);

queryWrapper.eq("status",SysCode.BLOG\_STATUS.RELEASED);

queryWrapper.eq("del\_tag",SysCode.DELTAG.WSC);

queryWrapper.orderByDesc("release\_time");

return blogMapper.selectPage(page,queryWrapper);

}

**4.6 博客页面模块**

（1）模块实现流程图

通过功能栏返回首页、新增博客、管理博客；左侧显示博客主页面的右侧功能栏，右侧大篇幅显示博客的具体内容，内容下方提供好文要顶、关注作者、收藏该文功能。最下方显示该博客的评论列表、并提供新增评论功能。博客页面模块流程图如图4-11所示。

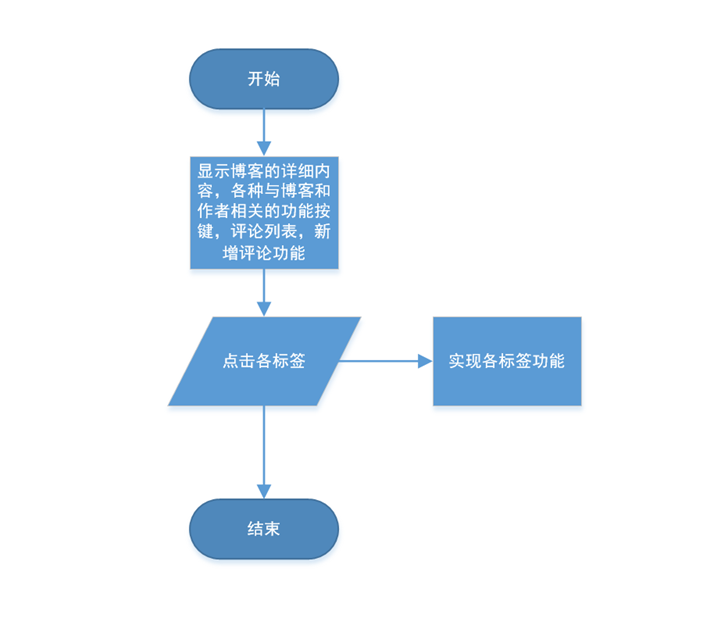


图 4-11 博客页面模块流程图

（2）模块界面展示

博客页面如图4-12所示。

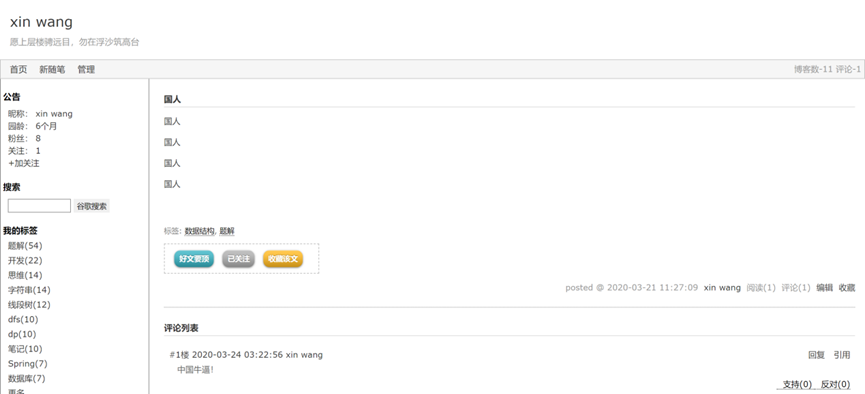


图 4-12 博客页面界面

（3）核心代码展示

根据博客id向数据库获取博客全部信息，同时获取用户头像url。

@Select("SELECT sys\_blog.\*,user.pic\_url `authorPicUrl` FROM sys\_blog,user " +

"WHERE sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

" and sys\_blog.id = #{id} "+

"order by sys\_blog.release\_time desc")

Blog selectById(@Param("id") String id);

**4.7 新增评论模块**

（1）模块实现流程图

输入评论内容必填项，评论使用富文本框编辑器，可添加各种评论形式。评论内容不能为空。如果为空，点击提交时页面提醒“评论内容不能为空”，如果输入内容，提示“评论成功”。评论列表即时刷新显示相关评论。新增评论模块流程图如图4-13所示。

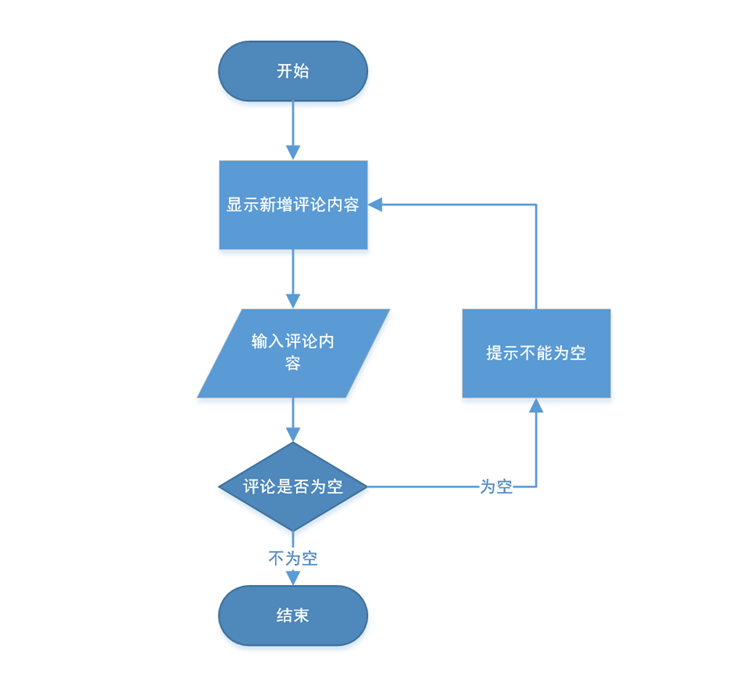


图 4-13 新增评论模块流程图

（2）模块界面展示

新增评论页面如图4-14所示。

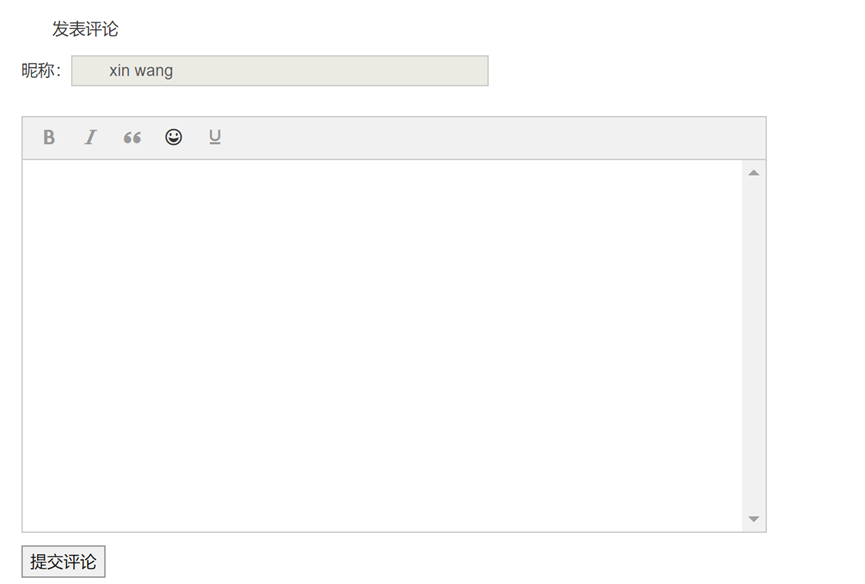


图 4-14 新增评论界面

（3）核心代码展示

从前端获取评论对象存入数据库并增加博客评论数和添加评论动态。

@Override

@Transactional

public void addComment(Comment comment) {

comment.setCreateTime(new Date());

comment.setDelTag(SysCode.DELTAG.WSC);

commentMapper.insert(comment);

Blog blog = blogService.selectById(comment.getBlogId());

blogService.addCommentNum(blog);

Activity activity = new Activity();

activity.setBlogTitle(blog.getTitle());

activity.setCategory(SysCode.ACTIVITY\_CATEGORY.COMMENT);

activity.setContent(comment.getContent());

activity.setCreateTime(new Date());

activity.setDelTag(SysCode.DELTAG.WSC);

activity.setUserId(comment.getSender());

activityService.addActivity(activity);

}

public String addComment(Comment comment){

commentService.addComment(comment);

return JSONUtil.success(SysCode.TIPMESSAGE.COMMENTSUCCESS);

}

**4.8 消息模块**

（1）模块实现流程图

用户通过信息模块查看所有收到的消息，消息可分为门户官方发来的公告、通知或者其他用户发来的私信，点击发件箱显示用户个人发送给其他用户的消息，点击未读消息显示所有其他用户或官方发来的还未读的消息。点击撰写新消息显示新增消息模块。可点击信件后方的x符号删除显示消息。消息模块流程图如图4-15所示。

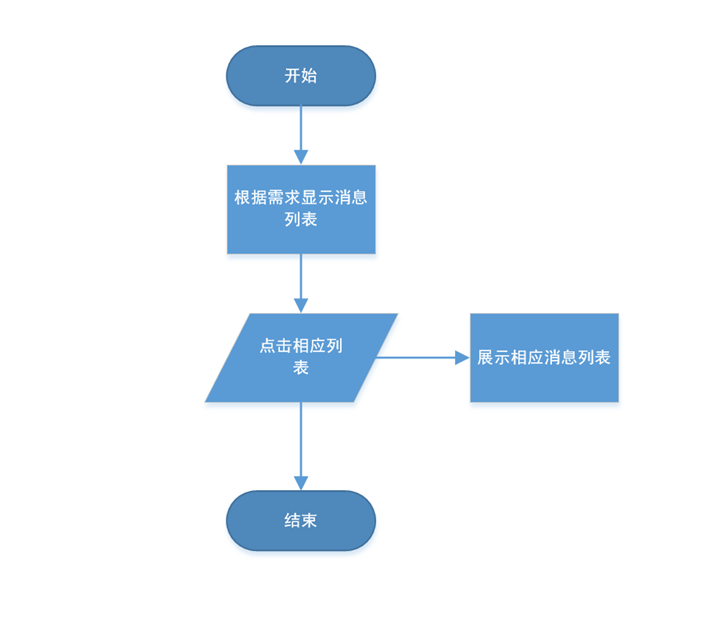


图 4-15 消息模块流程图

（2）模块界面展示

消息模块页面如图4-16所示。



图 4-16 消息模块界面

（3）核心代码展示

创建不同QuerryWrapper对象以获取收件箱、发件箱、未读消息列表。

@Override

public IPage<Msg> inList(Page<Msg> page, User user) {

QueryWrapper<Msg> queryWrapper = new QueryWrapper();

queryWrapper.eq("del\_tag\_re",SysCode.DELTAG.WSC);

queryWrapper.eq("receiver",user.getId());

queryWrapper.orderByDesc("create\_time");

return msgMapper.selectPage(page,queryWrapper);

}

@Override

public IPage<Msg> outList(Page<Msg> page, User user) {

QueryWrapper<Msg> queryWrapper = new QueryWrapper();

queryWrapper.eq("del\_tag\_se",SysCode.DELTAG.WSC);

queryWrapper.eq("sender",user.getId());

queryWrapper.orderByDesc("create\_time");

return msgMapper.selectPage(page,queryWrapper);

}

@Override

public IPage<Msg> unReadList(Page<Msg> page, User user) {

QueryWrapper<Msg> queryWrapper = new QueryWrapper();

queryWrapper.eq("del\_tag\_re",SysCode.DELTAG.WSC);

queryWrapper.eq("receiver",user.getId());

queryWrapper.eq("is\_read",SysCode.IS\_READ.NO);

queryWrapper.orderByDesc("create\_time");

return msgMapper.selectPage(page,queryWrapper);

}

**4.9 撰写消息模块**

（1）模块实现流程图

撰写消息模块，通过输入其他用户的用户名来确认想要发送的对象；标题主要概括这条消息的主要内容；内容为这条消息的主要内容。点击发送校验各输入项内容，所有输入项均为必填项，收件人必须是网站已存在的用户，标题不能超过100个字符，内容不得超过1000个字符。输入错误会根据内容提醒相应错误提示，如无问题，直接提示发送成功。撰写消息模块流程图如图4-17所示。

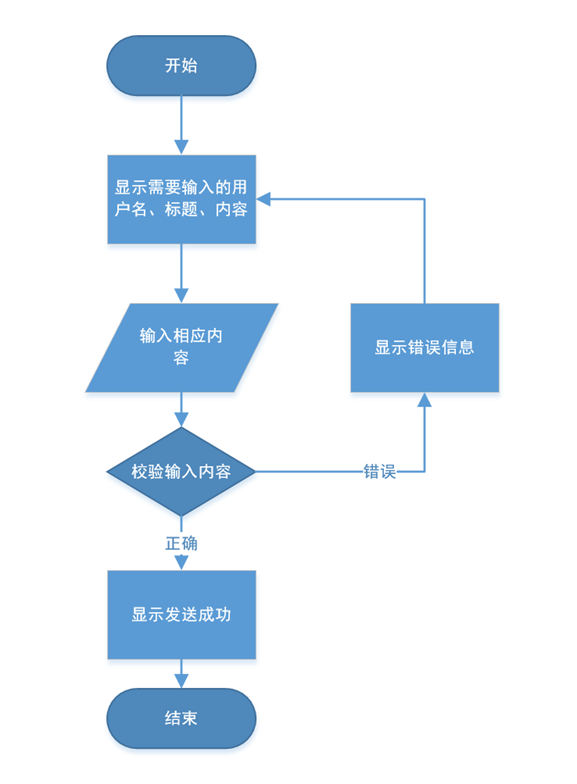


图 4-17 撰写消息模块

（2）模块界面展示

撰写消息页面如图4-18所示。



图 4-18 撰写消息界面

（3）核心代码展示

从前端获取消息对象，通过消息的接收人获取全部用户信息并存入数据库。

@Override

public void addMsg(Msg msg) {

QueryWrapper<User> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("username",msg.getReceiverName());

User user = userService.selectOne(queryWrapper);

msg.setCreateTime(new Date());

msg.setDelTagRe(SysCode.DELTAG.WSC);

msg.setDelTagSe(SysCode.DELTAG.WSC);

msg.setIsRead(SysCode.IS\_READ.NO);

msg.setReceiver(user.getId());

msgMapper.insert(msg);

}

**4.10 相关博客模块**

（1）模块实现流程图

相关博客模块功能：显示用户日常浏览所收藏的博客列表；用户关注博主的所发表的博客列表；用户日常点赞过的博客列表；用户曾经评论过的博客列表。点击博客进入该博客页面。相关博客模块流程图如图4-19所示。

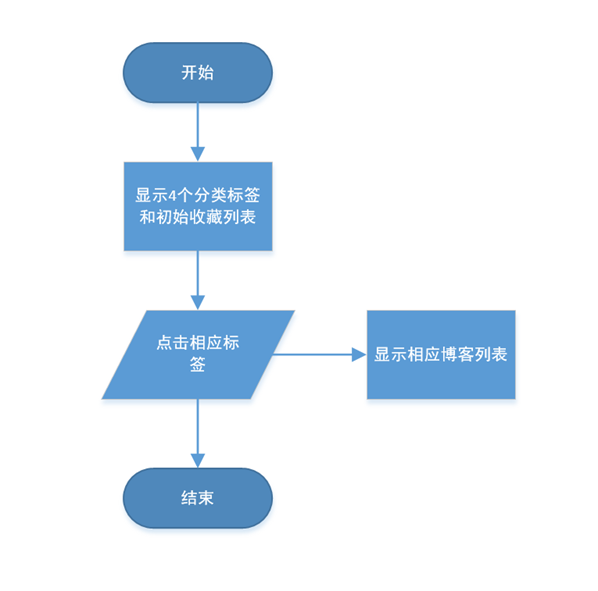


图 4-19 相关博客模块流程图

（2）模块界面展示

相关博客页面如图4-20所示。



图 4-20 相关博客界面

（3）核心代码展示

根据不同sql获取收藏博客、推荐博客、评论过博客列表。

@Select("select sys\_blog.\*,user.pic\_url `authorPicUrl` from sys\_blog,user where sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

"and sys\_blog.id in (" +

"select tt.blog\_id from" +

"(" +

"select DISTINCT(blog\_id) from sys\_comment where sender =

#{userId} " +

") tt"+

")")

IPage<Blog> getMyCommentList(Page<Blog> page,@Param("userId")String userId);

@Select("select sys\_blog.\*,user.pic\_url `authorPicUrl` from sys\_blog,user where sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

"and sys\_blog.id in (" +

"select tt.blog\_id from" +

"(" +

"select blog\_id from sys\_blog\_like where user\_id = #{userId} " +

") tt"+

")")

IPage<Blog> getILikeList(Page<Blog> page,@Param("userId")String userId);

@Select("select sys\_blog.\*,user.pic\_url `authorPicUrl` from sys\_blog,user where sys\_blog.author=user.id and sys\_blog.del\_tag='1' and sys\_blog.status = 'released'" +

"and sys\_blog.id in (" +

"select tt.blog\_id from" +

"(" +

"select blog\_id from sys\_blog\_favorite where user\_id = #{userId} " +

") tt"+

")")

IPage<Blog> getMyFavoriteList(Page<Blog> page,@Param("userId")String userId);

**4.11 新增博客模块**

（1）模块实现流程图

新增博客模块，通过填写博客标题，选择博客门户提供的栏目，通过富文本框编辑器填写博客的内容，最后通过点击发布来实现博客发布，或者点击存为草稿暂时存为草稿。校验：标题是必填项，长度不能超过100个字符串；栏目是必选项；内容是必填项，长度不得超过10000个字。新增博客模块流程图如图4-21所示。

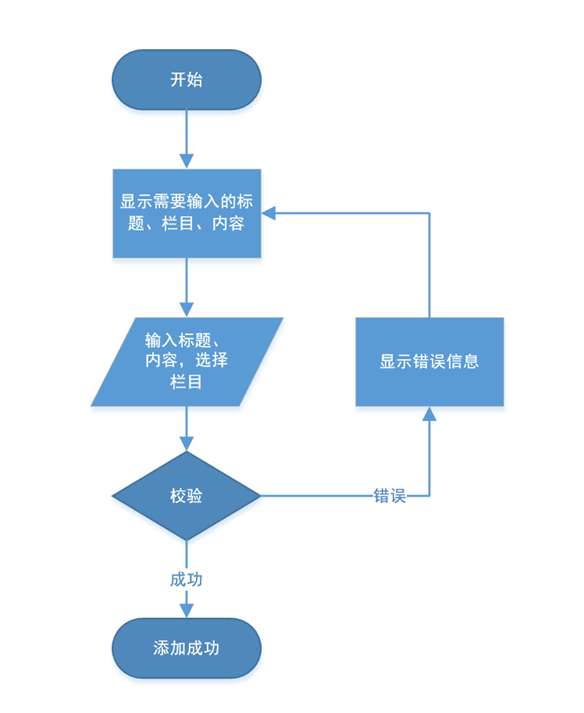


图 4-21 新增博客模块流程图

（2）模块界面展示

新增博客页面如图4-22所示。

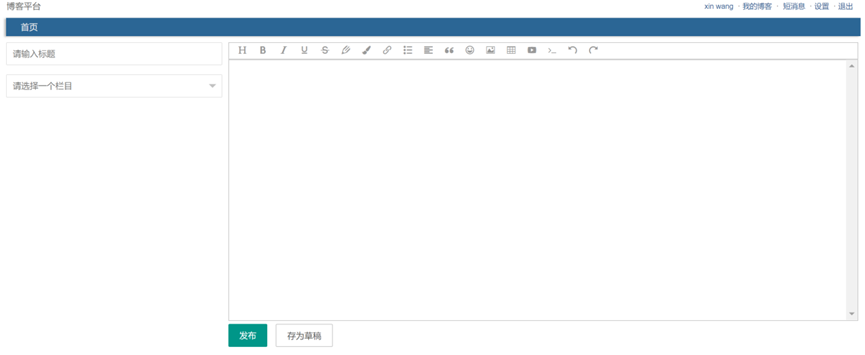


图 4-22 新增博客模块界面

（3）核心代码展示

新增和修改通用接口，根据是否有id判断是新增还是修改，获取博客对象新增或修改博客信息。

@Override

public String saveOrUpdate(Blog blog) {

User user = (User)SecurityUtils.getSubject().getPrincipal();

if(StringUtils.isEmpty(blog.getId())){

blog.setCreateTime(new Date());

blog.setAuthor(user.getId());

blog.setAuthorName(user.getUsername());

blog.setCollectNum(SysCode.NUM.ZERO);

blog.setCommentNum(SysCode.NUM.ZERO);

blog.setLikeNum(SysCode.NUM.ZERO);

blog.setDelTag(SysCode.DELTAG.WSC);

blog.setIsRecommend(SysCode.IS\_RECOMMEND.NO);

if(blog.getStatus().equals(SysCode.BLOG\_STATUS.RELEASED)){

blog.setReleaseTime(new Date());

}

blogMapper.insert(blog);

activityService.addActivity(addActivity(blog));

BlogEs blogEs = new BlogEs();

BeanUtils.copyProperties(blog,blogEs);

blogEsService.addBlogEs(blogEs);

return JSONUtil.success(SysCode.TIPMESSAGE.SAVESUCCESS);

}

else{

blog.setUpdateTime(new Date());

blogMapper.updateById(blog);

return JSONUtil.success(SysCode.TIPMESSAGE.UPDATESUCCESS);

}

}

**4.12 用户管理模块**

（1）模块实现流程图

用户管理模块可以通过上方用户名、姓名、角色来搜索相应用户，可以通过修改角色按钮修改用户的角色。用户管理模块流程图如图4-23所示。

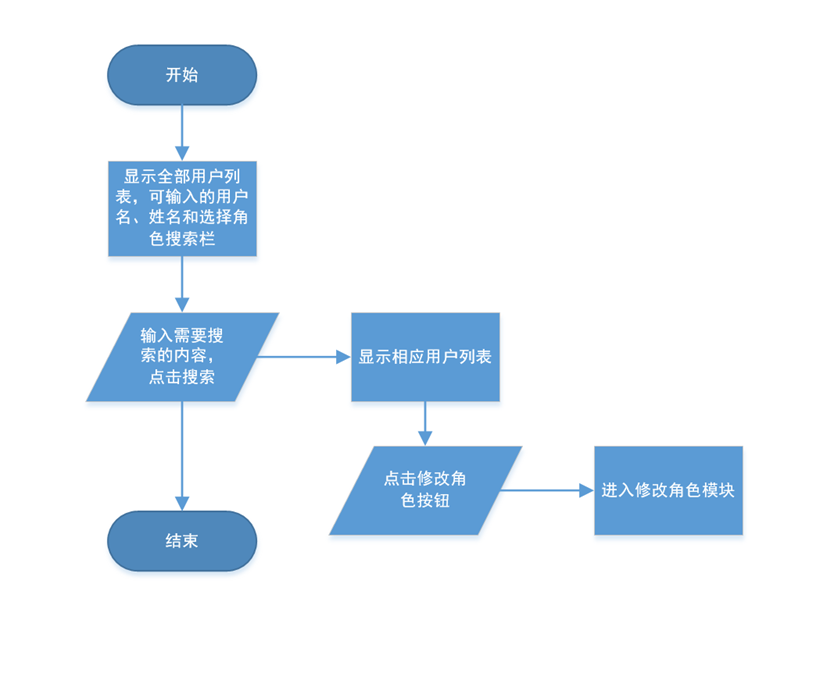


图 4-23 用户管理模块流程图

（2）模块界面展示

用户管理页面如图4-24所示。

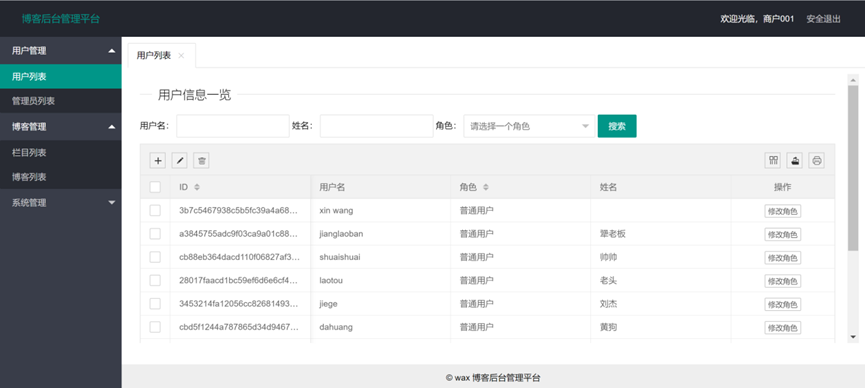


图 4-24 用户管理模块界面

（3）核心代码展示

Page对象提供分页功能，User对象提供搜索条件，同过创建Querry对象进行数据列表搜索。

@Override

public IPage<User> selectPage(Page<User> page, User user) {

QueryWrapper<User> queryWrapper = new QueryWrapper<>();

if(!StringUtils.isEmpty(user.getUsername())){

queryWrapper.like("username", user.getUsername());

}

if(!StringUtils.isEmpty(user.getRoleCode())){

queryWrapper.eq("role\_code",user.getRoleCode());

}

queryWrapper.eq("del\_tag", SysCode.DELTAG.WSC);

queryWrapper.orderByDesc("create\_time");

return userMapper.selectPage(page,queryWrapper);

}

**4.13 修改角色模块**

（1）模块实现流程图

选择想要更改的角色，点击修改进行修改。修改成功后提示“修改角色成功”。修改角色模块流程图如图4-25所示。

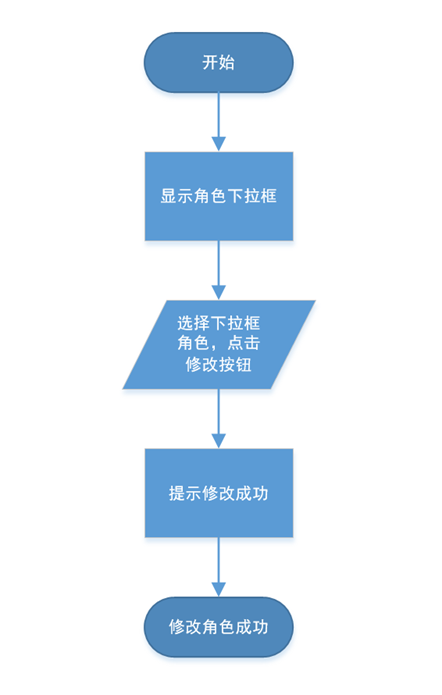


图 4-25 修改角色模块流程图

（2）模块界面展示

修改角色页面如图4-26所示。

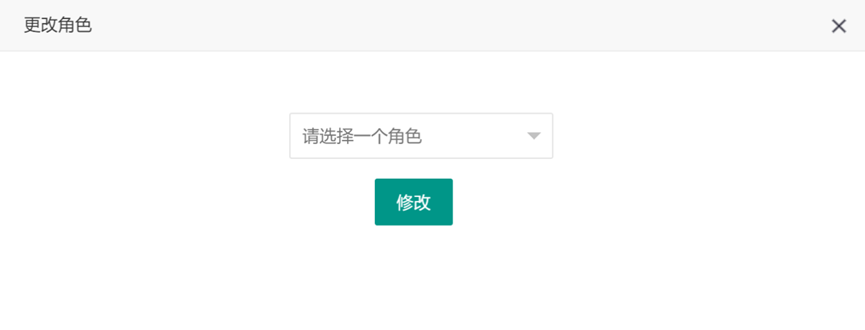


图 4-26 修改角色模块界面

（3）核心代码展示

Model提供返回页面数据功能，获取所有角色列表返回到修改角色页面，并且跳转到修改角色页面。

@RequestMapping(value="/goUserChangeRole.do")

public String goUserChangeRole(String userId,Model model){

List<Role> list = roleService.selectAll();

model.addAttribute("roleList",list);

model.addAttribute("userId",userId);

return "/user/userChangeRole";

}

**4.14 栏目管理模块**

（1）模块实现流程图

通过栏目名搜索想要查看的栏目。栏目管理模块流程图如图4-27所示。

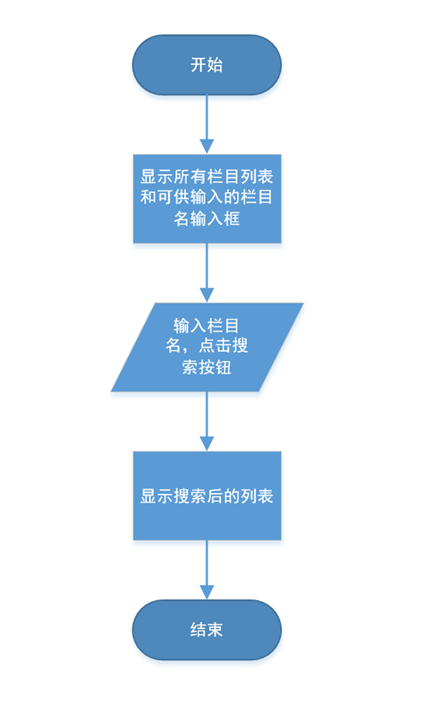


图 4-27 栏目管理模块流程图

（2）模块界面展示

栏目管理页面如图4-28所示。

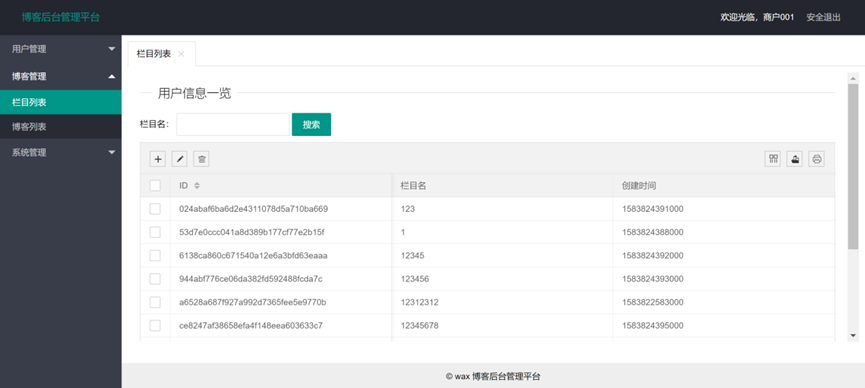


图 4-28 栏目管理模块界面

（3）核心代码展示

通过@GetMapping注解调用selectAll方法，Page提供分页功能，获取所有栏目列表返回给页面数据。

@GetMapping(value = "/selectAll.do")

@ResponseBody

public String selectAll(Page<BlogCategory> page){

IPage<BlogCategory> ipage = blogCategoryService.selectAll(page);

return JSONUtil.layUITable(ipage.getRecords(),ipage.getTotal());

}

@Override

public IPage<BlogCategory> selectAll(Page<BlogCategory> page) {

QueryWrapper<BlogCategory> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("del\_tag", SysCode.DELTAG.WSC);

return blogCategoryMapper.selectPage(page,queryWrapper);

}

**4.15 新增栏目模块**

（1）模块实现流程图

通过输入栏目名，点击立即提交生成新栏目。成功后提示添加成功。新增栏目模块流程图如图4-29所示。

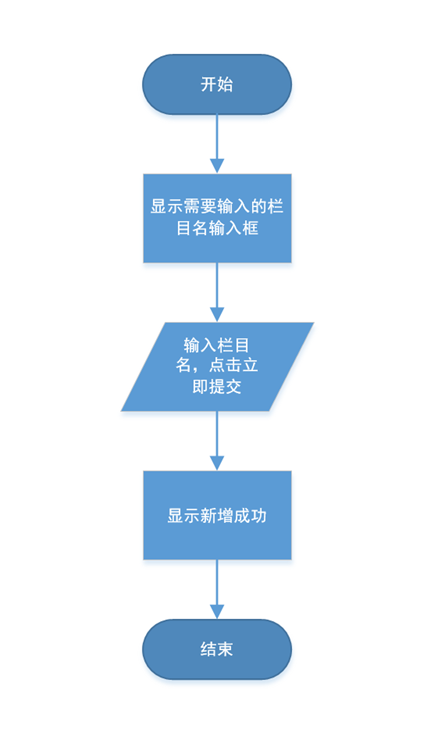


图 4-29 新增栏目模块流程图

（2）模块界面展示

新增栏目页面如图4-30所示。

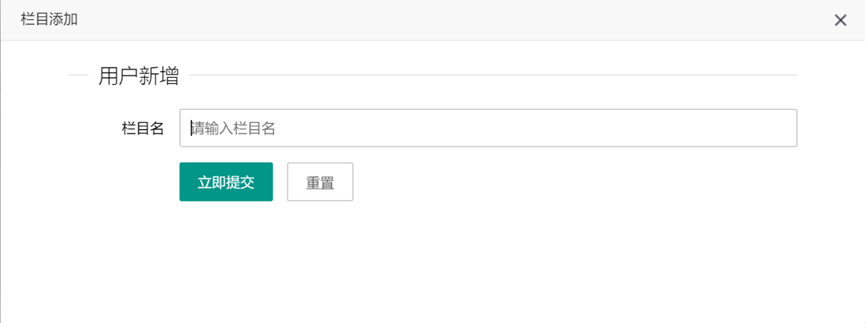


图 4-30 新增栏目模块界面

（3）核心代码展示

新增修改通用接口，根据栏目id判断是新增还是修改，改变数据库数据。

@Override

public void saveOrUpdate(BlogCategory blogCategory) {

if(StringUtils.isEmpty(blogCategory.getId())){

//add

blogCategory.setCreateTime(new Date());

blogCategory.setDelTag(SysCode.DELTAG.WSC);

blogCategoryMapper.insert(blogCategory);

}

else {

//update

blogCategoryMapper.updateById(blogCategory);

}

}

**4.16 博客管理模块**

（1）模块实现流程图

博客管理模块显示博客列表，包含标题、作者、点赞数、浏览数、发表时间属性。可通过标题、作者进行博客的模糊查询。操作栏按钮：前往，跳转到该博客页面；推荐，将博客列为编辑推荐博客；取消推荐，将博客从编辑推荐博客列表中移除。博客管理模块流程图如图4-31所示。

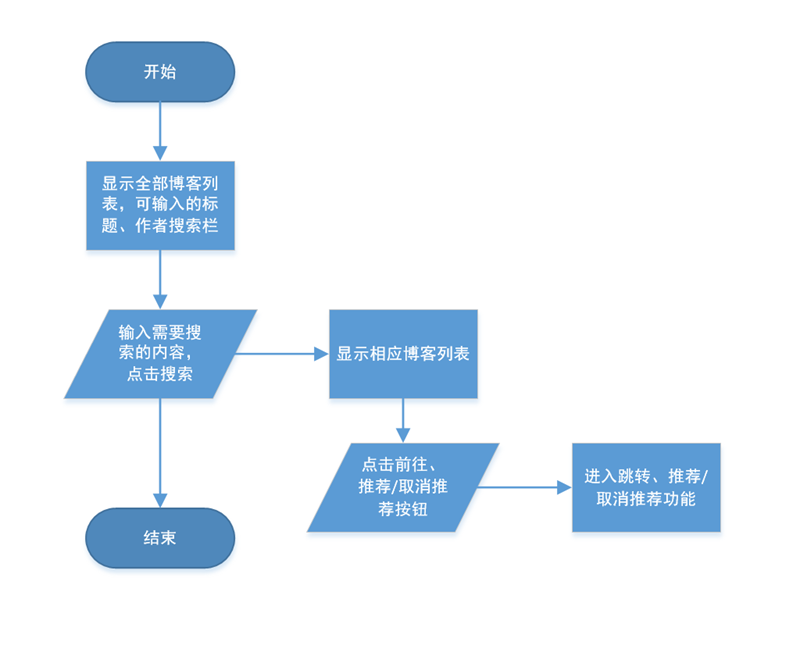


图 4-31 博客管理模块流程图

（2）模块界面展示

栏目管理页面如图4-32所示。

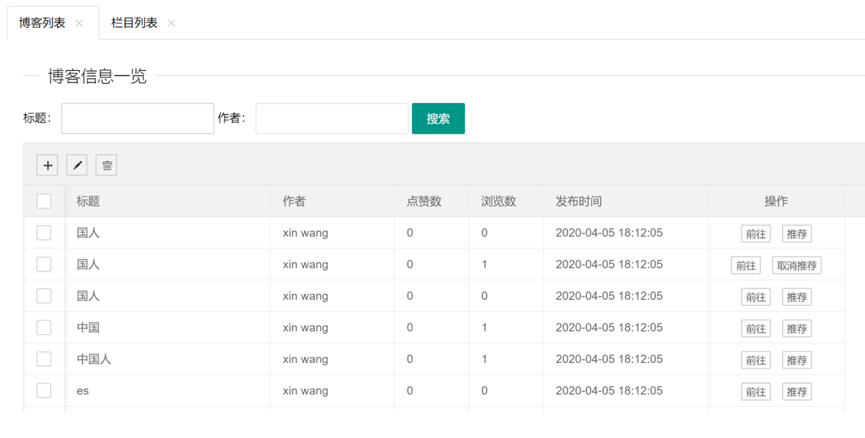


图 4-32 博客管理模块界面

（3）核心代码展示

推荐和取消推荐博客方法，通过id获得博客id，根据id更改数据库中是否被推荐数据。

@Override

public void recommend(String id) {

Blog blog = new Blog();

blog.setId(id);

blog.setIsRecommend(SysCode.IS\_RECOMMEND.YES);

blogMapper.updateById(blog);

}

@Override

public void cancelRecommend(String id) {

Blog blog = new Blog();

blog.setId(id);

blog.setIsRecommend(SysCode.IS\_RECOMMEND.NO);

blogMapper.updateById(blog);

}

**5 系统测试**

**5.1 系统测试概述**

在每个软件的系统设计完成之后，都需要对完成的软件进行完整且系统的测试。该测试的目的主要在于帮助我们发现编写时遗漏的错误，如果不能及时发现，可能在后面的编写中产生更大的错误。系统测试之所以能成为整个软件开发时最重要的一环，在于他能给使用者提供一个安全可靠的使用环境，能够决定系统软件是否具备上线到市场的资格。如何挑选测试用例也是开发时的重点，好的测试用例能够使测试进行的更顺利，能够更全面的发现问题。

**5.2 系统测试方法**

软件的测试方法大致可分为两大类，一种为黑盒测试，另一种为白盒测试。这两种测试着重点各不相同，黑盒测试着重于功能测试，不去考虑方法内部业务逻辑，只去要求输入同样的参数，得出同样的结果。白盒测试则不仅要去求输入参数与输出结果，同时还需要去测试系统内部逻辑结构，对系统业务逻辑中潜藏的缺陷进行测试，所以白盒测试相较于黑盒测试更能发现深层次的问题。

博客管理系统选择使用黑盒测试作为测试方法。

**5.3 系统测试用例**

（1）测试项目，如表5-1所示。

表 5-1 测试项目表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | 测试项编号 | 测试内容 | 测试优先级 |
| 0001 | A0001 | 博客系统 | 高 |
| 0002 | B0002 | 登录功能 | 低 |
| 0003 | C0003 | UI界面 | 低 |

（2）测试需求，如表5-2所示。

表 5-2 测试项目需求表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试功能 | 测试优先级 |
| A0001 | 用户注册 | 高 |
| A0002 | 用户登录 | 高 |
| A0003 | 用户激活 | 高 |
| B0001 | 发表博客 | 高 |
| B0002 | 点赞博客 | 高 |
| B0003 | 评论博客 | 高 |
| B0004 | 收藏博客 | 高 |
| C0001 | 查看消息列表 | 高 |
| C0002 | 发送消息 | 高 |
| D0001 | 搜一搜功能 | 高 |
| D0002 | 修改个人信息 | 高 |

（3）测试用例，如表5-3所示。

表 5-3 测试项目用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试需求 | 博客管理系统 | | | |
| 描述 | 客户端所有功能的测试 | | | |
| 优先级 | 高 | | | |
| 预置条件 | 管理员登录平台系统（账号：Admin 密码：Admin）  用户登录平台系统（账号：xinwang 密码：123456） | | | |
| 测试时间 | 2020.05.01 | 测试人员 | 王鑫 | |
| 测试用例序号 | 输入条件 | 操作步骤 | 预期输出 | 测试结果 |
| A0001 | 注册用户 | 进入注册页面，输入邮箱、用户名、密码，点击注册按钮 | 注册成功 | 测试通过 |
| A0002 | 用户登录 | 进入登陆页面，输入邮箱/用户名，点击登录按钮 | 登陆成功 | 测试通过 |
| A0003 | 用户激活 | 获取注册激活邮箱，点击邮箱中激活链接 | 激活成功 | 测试通过 |
| B0001 | 发表博客 | 进入新增博客页面，输入博客标题、正文，点击发表按钮 | 成功发表博客 | 测试通过 |
| B0002 | 点赞博客 | 进入相关点击相关博客点赞按钮 | 成功点赞博客 | 测试通过 |
| B0003 | 评论博客 | 输入评论必填项，在相关博客下方评论 | 成功评论博客 | 测试通过 |
| B0004 | 收藏博客 | 点击相关博客收藏按钮 | 成功收藏博客 | 测试通过 |
| C0001 | 查看消息列表 | 进入个人消息页面 | 正常显示消息列表 | 测试通过 |
| C0002 | 发送消息 | 输入相关必填项，点击发送按钮 | 成功发送消息 | 测试通过 |
| D0001 | 搜一搜功能 | 输入相关搜索内容，点击搜索按钮 | 成功查找相应博客 | 测试通过 |
| D0002 | 修改个人信息 | 修改将要修改的信息 | 修改成功 | 测试通过 |

**5.4 系统测试总结**

通过这次系统测试发现了自身在编写代码上还有很多不足。有的错误是因为注意力不集中，粗心大意产生的错误；还有的是因为代码不规范导致的改一处而动全身的错误，这就是编码时没有考虑到合理的解耦带来的麻烦。通过这些错误也让我更好的理解了系统测试的重要性，没有测试就相当于一台好车没有好的护理，即使车再好也会出各种各样的问题。

整个项目测试下来使我的编码能力，思维缜密能力都得到的提高。我相信，今后的我还会不断提高。

**结论与展望**

开发博客管理系统的过程是有趣的，它不仅稳固了我过去学过的知识，还帮助我学习了很多新知识。通过之前实习六个月学习到的项目管理知识，博客管理系统采用了类似的设计方式。虽然很多当面做不到尽善尽美，比如注释很多没有按照标准进行标注，代码冗余还是存在，没有尽可能地实现代码的可重用性，但是整个项目还是令我收获颇多，因为项目为一个人独立完成，没有前端的帮忙，自己独立完成前后端联调，学习到了很多前端的知识和框架。这点对以后工作中会起到很大的帮助作用，写后端接口时会多考虑前端的实现以更好地进行前后端互连。

通过博客管理系统我更多的接触到了mybatis-plus和elasticsearch这种新技术，通过学习这两种新知识使我找到了学习新知识时应该使用的方式，不应该片面的去单纯使用一个知识，而是要理解一个知识，其实每个新知识的学习都是相通的，不在于你去学习什么，而是在于你怎么去学习。在以后的工作中一定会遇到很多新知识需要去学习，但拥有好的学习方法，新知识自然会容易掌握。

**致 谢**

在这次毕业设计中，感谢徐茂生老师负责对我进行指导，期间我遇到了很多难题和疑问。面对我遇到的种种困难，每次徐老师都能根据我设计的内容、论文的写、图纸的绘制等各个方面提供关键并宝贵的意见，对我进行细心指导。通过与老师的沟通与交流也使我学习到了很多，完成一件事时一定要用心用心再用心，做每件事时不能受外界环境所影响，这样不仅处理不好外界环境，手中的事也完成不好，再次感谢徐老师与我沟通时的耐心和理解。古话说：授人以鱼不如授人以渔，凡事总有解决的办法，没有一件事是没有经过而直接到结果的，只有掌握了解决事物的办法才能不惧困难，解决问题。

作者：王鑫

2020年05月01日

**参考文献**

[1] 贾晓芳, 沈泽刚. JavaWeb应用开发中的常见乱码形式及解决方法[J]. 软件导刊(4),214-216.

[2] 苏航. 接口的意义及在Java Web三层架构中的作用分析[J]. 西部皮革, 2016(24),19-19.

[3] 赵璞. Java Web程序开发中字符乱码的原因与解决办法[J]. 电脑编程技巧与维护, 2016(20),24-24.

[4] 李健, 田立锋, 郭瑞. Java SSH框架在Java Web中的应用[J]. 电脑与电信(3),69-72.

[5] 徐雯,高建华.基于Spring MVC及MyBatis的Web应用框架研究[J].微型电脑应用,2012,28(7),1-4.

[6] 陈涛, 黄艳峰. Java Web开发中文件上传方法研究与实现[J]. 电脑知识与技术, 2016, 12(11),48-49.

[7] 吴沧舟, 兰逸正, 张辉, et al. 基于MySQL数据库的优化[J]. 电子科技, 2013,26(9),182.

[8] 辛立伟, 王水萍. Java Web开发框架-Struts框架[J]. 福建电脑, 2006(9),151-152.

[9] 杨家炜. 基于Spring Boot的web设计与实现[J]. 轻工科技, 2016(7),86-89.

[10] 陈华恩. JAVA设计模式研究之抽象工厂模式[J]. 电脑知识与技术, 2010, 6(9),2245-2246.

[11] 杨新艳, 苏爱玲. Java设计模式在基于MVC模式系统中的应用研究[J]. 计算机光盘软件与应用, 2014(13),128-129.

[12] 刘敏娜. 基于JAVA Web的教材管理系统的设计与实现[J]. 数字技术与应用, 2016(1),173-174.

[13] 陈玲. 基于nginx、tomcat、redis的高性能Java Web开源架构[J]. 计算机光盘软件与应用, 2014(22),296-296.

[14] 杨涵夫. 基于Spring技术的油田生产数据采集系统[J]. 油气田地面工程(11),101-102.

[15] G. Bollella, J. Gosling. The real-time specification for Java[J]. Computer, 2002, 33(6),47-54.