2020—2021学年第二学期

**《**Java EE企业级应用**》**课程大作业

设计题目： Springboot+Vue简易博客系统

**学号: 2020035321Z**

**姓名： 简腾飞**

**日期： 2021/6/19**

|  |  |  |  |
| --- | --- | --- | --- |
| **《Java EE企业级应用》评价标准** | | | |
| **项目** | **内容** | **单项分值** | **标准分值** |
| 软件  分析设计 | （1）分析合理 | 10 | 30 |
| （2）功能结构分析准确，有结构图 |
| （3）数据结构设计合理，Sql脚本正确 | 20 |
| （4）系统架构清晰，有相关说明 |
| （5）开发环境和运行环境明确 |
| （6）文档格式规范 |
| 软件实现 | （7）功能较完整、能正确运行 | 40 | 40 |
| （8）界面设计合理、易用 |
| （9）代码规范（命名、注释、格式） |
| （10）程序有一定难度（至少包含三个以上数据表） |
| 总体效果 | （11）项目完成总体质量 | 10 | 10 |
| （12）软件功能及技术水平 |
| 评语 |  | | |
| 得分 |  | | |

# 简易博客系统

作者：简腾飞

## 1.系统分析与设计

简单的博客系统，采用前后端分离的设计，后端使用Springboot作为基础框架，使用Vue作为前端框架。

1. 针对用户，系统应该具有登录，注册，忘记密码，修改密码，忘记密码等功能
2. 针对博客，博客应该也是基于用户来进行管理的，博客管理有发布博客，查看博客，用户管理自己已经发布的博客等功能。

### 1.1 系统分析

1. 功能性需求

1. 登录：用户登录时应该有表单校验，登录失败提示，重置表单等
2. 注册：用户注册应该有表单校验，重置表单，用户名已被注册提示等
3. 修改密码：用户登录成功后应该修改密码，查看个人信息功能
4. 忘记密码：用户忘记密码时，应该可以通过注册时的邮箱重置密码
5. 博客浏览: 任何人（不登录）浏览该网页时，应该都有浏览博客的权限
6. 发布博客：未登录时点击发布博客，应该跳转到登录页面，如果用户登录了，应该跳转到编辑新博客页面。编辑完成，点击发布，通过表单校验后，发布成功。
7. 查看博客：当查看到自己的博客时，应该会提示有修改和删除的功能，当查看到别人的博客时，应该只能查看
8. 管理已发布的博客：用户登录后，应该具有管理自己已发布博客的功能，包括编辑查看删除等。

2. 非功能需求

后端的某些接口，应该具有拦截功能，请求时需要判断用户是否登录，删除时，判断登录的用户和博客对应的用户是否为同一个用户。

3. 开发和运行环境

1. 开发环境：

后端：Windows10，idea2021+MySQL5.7+springboot2.3.2，

前端：Windows10，WebStorm 2021.1.1+Vue2.x+Element-Ui

1. 运行环境：

后端：Windows10，idea2021+MySQL5.7+springboot2.3.2，

前端：Windows10，WebStorm 2021.1.1+Vue2.x+Element-Ui

### 1.2 总体设计

**1．功能模块设计**

**总体设计采用前后端分离设计，后端采用springboot框架，前端采用vue框架，前后端分离。**

**用户模块：**

**界面采用element-ui，用现成的控价完成控件的设计，布局位置采用css调整。**

1. 登录：用户登录时应该有表单校验，登录失败提示，重置表单等
2. 注册：用户注册应该有表单校验，重置表单，用户名已被注册提示等
3. 修改密码：用户登录成功后应该修改密码，查看个人信息功能
4. 忘记密码：用户忘记密码时，应该可以通过注册时的邮箱重置密码

**以上功能前端采用，element-ui表单完成前端界面的编写，后端接口实现均采用Post请求。找回密码会将重置后的密码发送到个人邮箱。**

**博客模块：**

1. 博客浏览: 任何人（不登录）浏览该网页时，应该都有浏览博客的权限
2. 发布博客：未登录时点击发布博客，应该跳转到登录页面，如果用户登录了，应该跳转到编辑新博客页面。编辑完成，点击发布，通过表单校验后，发布成功。
3. 查看博客：当查看到自己的博客时，应该会提示有修改和删除的功能，当查看到别人的博客时，应该只能查看
4. 管理已发布的博客：用户登录后，应该具有管理自己已发布博客的功能，包括编辑查看删除等。

**博客概览，前端采用时间线插件和分页功能实现，博客编辑和修改采用mavon-editor实现编写和保存。博客管理采用表格的形式展示，后端接口采用分页形式查找。全部采用post的请求实现多条件查询。**

**2．数据库设计**

1. **m\_blog 博客表**

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **数据类型** | **是否为空** | **备注** |
| **blog\_id** | **bigint（20）** | **否** | **主键（自增）** |
| **user\_id** | **bigint（20）** | **否** | **外键（用户id）** |
| **blog\_title** | **varchar(255)** | **否** | **博客标题** |
| **blog\_** **description** | **varchar(255)** | **否** | **博客简述** |
| **blog\_** **content** | **longtext** | **否** | **博客内容** |
| **gmt\_create** | **datetime** | **否** | **博客创建日期** |
| **gmt\_modified** | **datetime** | **是** | **博客更正日期** |
| **blog\_** **deleted** | **int(5)** | **否** | **博客是否被删除** |

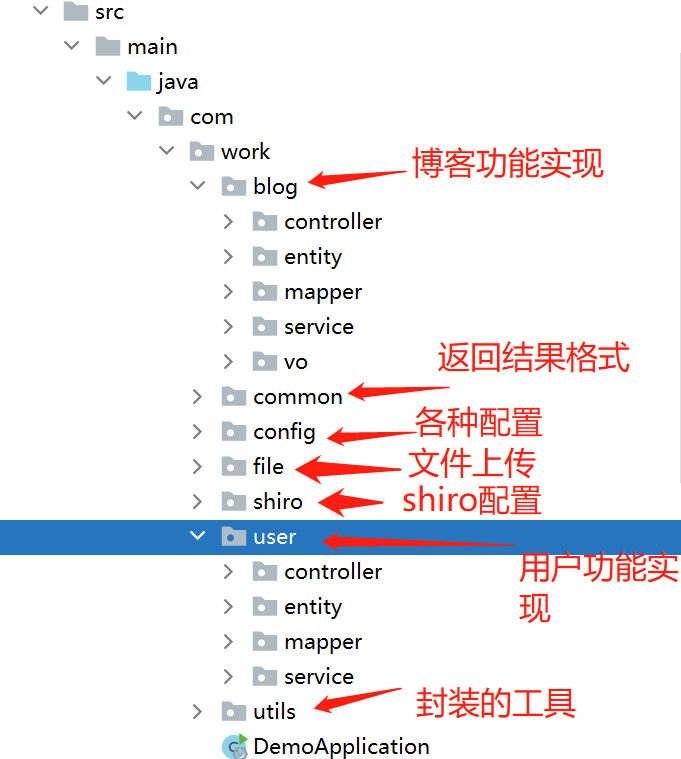
1. **m\_user用户表**

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **数据类型** | **是否为空** | **备注** |
| **user\_id** | **bigint(20)** | **否** | **主键（自增）** |
| **user\_name** | **varchar(64)** | **否** | **用户名** |
| **user\_avatar** | **varchar(255)** | **是** | **用户头像** |
| **user\_email** | **varchar(64)** | **否** | **用户邮箱** |
| **user\_** **password** | **varchar(64)** | **否** | **用户密码** |
| **user\_status** | **int(5)** | **否** | **是否被删除** |
| **user\_created** | **datetime** | **否** | **用户创建日期** |
| **last\_login** | **datetime** | **否** | **上一次登录时间** |

1. **用户博客表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **数据类型** | **是否为空** | **备注** |
| **user\_id** | **bigint(20)** | **否** |  |
| **blog\_id** | **bigint（20）** | **否** |  |

**3 包和文件夹划分**

****

**功能的包中的划分如下：**

****

### 1.3 实体类（数据模型）设计

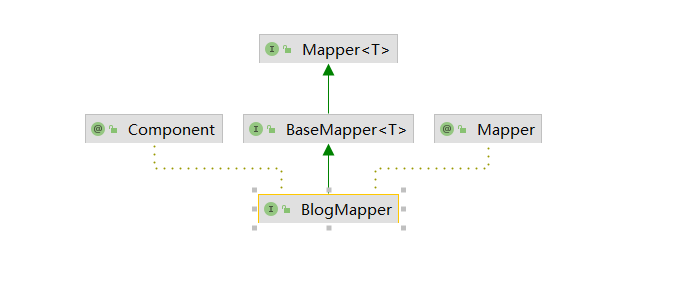
博客系统的实体类有2个。具体如下：

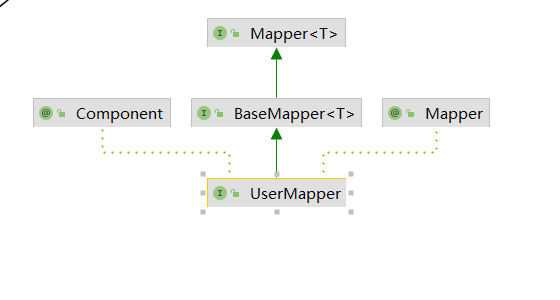
### 

### 1.4 接口设计

**1．数据访问接口设计**

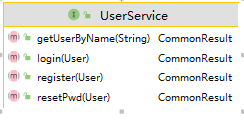
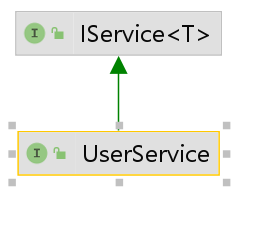
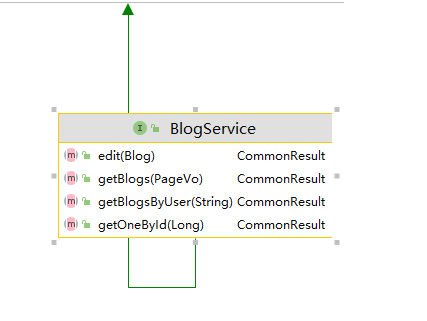
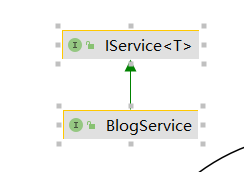
**数据访问层 我采用的是MyBatis-Plus 框架，其中内置了许多直接访问数据库的接口，不用自己去设计访问数据库的接口。**

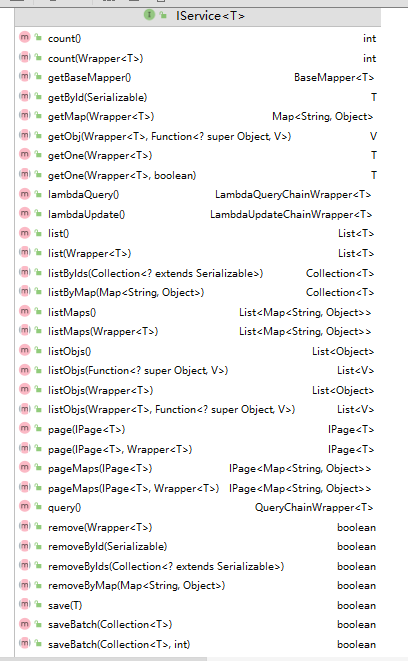




**2．业务逻辑接口设计**

**userservice和blogservice继承IService**

**、**



## 2 数据访问层实现

import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.work.user.entity.User;  
import org.apache.ibatis.annotations.Mapper;  
import org.springframework.stereotype.Component;  
  
*/\*\*  
 \* <p>  
 \* Mapper 接口  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-27  
 \*/*@Mapper  
@Component("UserMapper")  
public interface UserMapper extends BaseMapper<User> {  
  
}

package com.work.blog.mapper;  
  
import com.work.blog.entity.Blog;  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import org.apache.ibatis.annotations.Mapper;  
import org.springframework.stereotype.Component;  
  
*/\*\*  
 \* <p>  
 \* Mapper 接口  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-28  
 \*/*@Mapper  
@Component("BlogMapper")  
public interface BlogMapper extends BaseMapper<Blog> {  
  
}

对应的xml文件



## 3 业务逻辑层实现

public interface UserService extends IService<User> {  
 CommonResult login(User user);  
 CommonResult register(User user);  
 CommonResult resetPwd(User user);  
 CommonResult getUserByName(String username);  
}

public interface BlogService extends IService<Blog> {  
  
 CommonResult getOneById(Long blogId);  
  
 CommonResult getBlogs(PageVo pageVo);  
  
 CommonResult edit(Blog blog);  
  
 CommonResult getBlogsByUser(String userName);  
}

### 3.1 Service实现

UserService实现代码如下所示。

package com.work.user.service.impl;  
  
import cn.hutool.crypto.SecureUtil;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.work.common.results.CommonResult;  
import com.work.user.entity.User;  
import com.work.user.mapper.UserMapper;  
import com.work.user.service.UserService;  
import com.work.utils.EmailUtil;  
import com.work.utils.GetDateUtil;  
import com.work.utils.JwtUtils;  
import com.work.utils.RandomPwdUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import javax.annotation.Resource;  
import java.time.LocalDateTime;  
  
*/\*\*  
 \* <p>  
 \* 服务实现类  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-27  
 \*/*@Service  
public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements UserService {  
  
 @Resource  
 private JwtUtils jwtUtils;  
 @Autowired  
 private EmailUtil emailUtil;  
  
 @Override  
 public CommonResult login(User user) {  
 *//查询数据库* User userDb = this.getOne(new LambdaQueryWrapper<User>()  
 .eq(User::getUsername, user.getUsername()));  
  
 if (userDb == null) {  
 return CommonResult.*errorResponse*("该用户还未注册！");  
 } else {  
 if (userDb.getPassword().equals(SecureUtil.*md5*(user.getPassword()))) {  
 *//登陆成功 需要差生token* String token = jwtUtils.generateToken(userDb.getId());  
 LocalDateTime data = GetDateUtil.*getNetworkTime*();  
 *//1.具有转换功能的对象* userDb.setLastLogin(data);  
 this.updateById(userDb);  
 userDb.setToken(token);  
 return CommonResult.*successResponse*(userDb.setPassword(""), "登录成功");  
 } else {  
 return CommonResult.*errorResponse*("密码错误！");  
 }  
 }  
 }  
  
 public Boolean isRegister(String username) {  
 User user = this.getOne(new LambdaQueryWrapper<User>().eq(User::getUsername, username));  
 if (user != null)  
 return true;  
 return false;  
 }  
  
 @Override  
 public CommonResult register(User user) {  
 if (isRegister(user.getUsername())) {  
 return CommonResult.*errorResponse*("用户名已被注册！");  
 }  
 LocalDateTime dateTime = GetDateUtil.*getNetworkTime*();  
 String encodePwd = SecureUtil.*md5*(user.getPassword());  
 this.save(user.setPassword(encodePwd).setCreated(dateTime).setStatus(0));  
 return CommonResult.*successResponse*("", "注册成功！");  
 }  
  
  
 @Override  
 public CommonResult resetPwd(User user) {  
 String username = user.getUsername();  
 String email = user.getEmail();  
 if (isRegister(username)){  
 *//产生随机密码* String password =RandomPwdUtil.*genRandomNum*(8);  
 User userDb = this.getOne(new LambdaQueryWrapper<User>()  
 .eq(User::getUsername, username));  
 if (email.equals(userDb.getEmail())) {  
  
 userDb.setPassword(SecureUtil.*md5*(password));  
 this.updateById(userDb);  
 emailUtil.sendSimpleMessage("RestPassword","you password reset:\n "+password,"1308659229@qq.com",email);  
 return CommonResult.*successResponse*("验证通过", "重置密码已经发送到注册邮箱");  
 }else{  
 return CommonResult.*errorResponse*("用户名和邮箱不匹配！");  
 }  
 }  
 return CommonResult.*errorResponse*("该用户名未被注册！");  
 }  
  
 @Override  
 public CommonResult getUserByName(String username) {  
 User user =this.getOne(new LambdaQueryWrapper<User>().eq(User::getUsername,username));  
 return CommonResult.*successResponse*(user,"成功~");  
 }  
}

BlogService实现代码如下所示

package com.work.blog.service.impl;  
  
import cn.hutool.core.bean.BeanUtil;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.metadata.IPage;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.work.blog.entity.Blog;  
import com.work.blog.mapper.BlogMapper;  
import com.work.blog.service.BlogService;  
import com.work.blog.vo.PageVo;  
import com.work.common.results.CommonResult;  
import com.work.user.entity.User;  
import com.work.user.service.UserService;  
import com.work.utils.GetDateUtil;  
import com.work.utils.ShiroUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.util.Assert;  
import org.springframework.util.StringUtils;  
  
import java.util.List;  
  
*/\*\*  
 \* <p>  
 \* 服务实现类  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-28  
 \*/*@Service  
public class BlogServiceImpl extends ServiceImpl<BlogMapper, Blog> implements BlogService {  
  
 @Autowired  
 private UserService userService;  
 @Override  
 public CommonResult getOneById(Long blogId) {  
 Blog blog = this.getById(blogId);  
 Assert.*notNull*(blog, "该博文已被删除");  
 return CommonResult.*successResponse*(blog, "查询成功!");  
 }  
  
 @Override  
 public CommonResult getBlogs(PageVo pageVo) {  
  
 Integer pageSize = pageVo.getPageSize();  
 Integer pageNo = pageVo.getPageNo();  
 String blogName = pageVo.getBlogName();  
  
 Page page = new Page(pageNo,pageSize);  
 IPage pageData = this.page(page,new LambdaQueryWrapper<Blog>()  
 .like(!StringUtils.*isEmpty*(blogName),Blog::getTitle,blogName).orderByDesc(Blog::getGmtCreate));  
 return CommonResult.*successResponse*(pageData,"查询成功！");  
 }  
  
 @Override  
 public CommonResult edit(Blog blog) {  
 Blog blogTemp=null;  
 if (blog.getId()!=null){ *//提交编辑好的博客* blogTemp = this.getById(blog.getId());  
 *//比对用户id和博客的id是否一样，如果不一样则不能编辑* Assert.*isTrue*(blogTemp.getUserId().longValue()  
 == ShiroUtil.*getProfile*().getId().longValue(),"没有编辑权限");  
 }else{ *//添加新博客* blogTemp = new Blog();  
 *//这里我还需要去写入一个用户* User user =userService.getById(ShiroUtil.*getProfile*().getId());  
 blogTemp.setUserId(ShiroUtil.*getProfile*().getId());  
 blogTemp.setGmtCreate(GetDateUtil.*getNetworkTime*());  
 blogTemp.setStatus(0);  
 blogTemp.setAuthor(user.getUsername());  
 }  
 BeanUtil.*copyProperties*(blog,blogTemp,"id","userId","gmtCreate","status","author");  
 this.saveOrUpdate(blogTemp);  
 return CommonResult.*successResponse*(null,"操作成功！");  
 }  
  
 @Override  
 public CommonResult getBlogsByUser(String userName) {  
 List<Blog> blogs = this.list(new LambdaQueryWrapper<Blog>().eq(Blog::getAuthor,userName));  
 return CommonResult.*successResponse*(blogs,"查询成功~");  
 }  
}

## 

## 4 控制层实现

控制层主要用来处理用户请求，根据用户请求调用业务方法，并将结果转发给视图层。

UserController实现：

package com.work.user.controller;  
  
  
import com.work.common.results.CommonResult;  
import com.work.user.entity.User;  
import com.work.user.service.UserService;  
import org.apache.shiro.authz.annotation.RequiresAuthentication;  
import org.springframework.validation.annotation.Validated;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.annotation.Resource;  
  
*/\*\*  
 \* <p>  
 \* 前端控制器  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-27  
 \*/*@RestController  
@RequestMapping("/user")  
public class UserController {  
  
 @Resource  
 private UserService userService;  
 @PostMapping("/login")  
 public CommonResult login(@Validated @RequestBody User user){  
 return userService.login(user);  
 }  
  
 @PostMapping("/register")  
 public CommonResult register(@Validated @RequestBody User user){  
 return userService.register(user);  
 }  
  
 @PostMapping("/resetPwd")  
 public CommonResult resetPwd(@Validated @RequestBody User user){  
 return userService.resetPwd(user);  
 }  
 @RequiresAuthentication  
 @GetMapping("/logout")  
 public CommonResult logout(){  
*// SecurityUtils.getSubject().logout();* return CommonResult.*successResponse*(null,"退出成功");  
 }  
 *//更具用户名获取用户详细信息* @GetMapping("getUserByName/{username}")  
  
 public CommonResult getUserByName(@PathVariable("username") String username){  
 return userService.getUserByName(username);  
 }  
  
}

BlogController实现

package com.work.blog.controller;  
  
  
import com.work.blog.entity.Blog;  
import com.work.blog.service.BlogService;  
import com.work.blog.vo.PageVo;  
import com.work.common.results.CommonResult;  
import org.apache.shiro.authz.annotation.RequiresAuthentication;  
import org.springframework.util.Assert;  
import org.springframework.validation.annotation.Validated;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.annotation.Resource;  
  
*/\*\*  
 \* <p>  
 \* 前端控制器  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-28  
 \*/*@RestController  
@RequestMapping("/blog")  
public class BlogController {  
  
 @Resource  
 private BlogService blogService;  
  
 *//根据id查博客* @GetMapping("/getOneById/{id}")  
 public CommonResult getOneById(@PathVariable("id") Long blogId){  
 return blogService.getOneById(blogId);  
 }  
 *//分页查询* @PostMapping("/getBlogs")  
 public CommonResult getBlogs(@RequestBody PageVo pageVo){  
 return blogService.getBlogs(pageVo);  
 }  
  
 *//编辑博客* @RequiresAuthentication *//需要登录注解* @PostMapping("/edit")  
 public CommonResult edit(@Validated @RequestBody Blog blog){  
 return blogService.edit(blog);  
 }  
 *//编辑博客* @RequiresAuthentication *//需要登录* @DeleteMapping("deleteById/{id}")  
 public CommonResult deleteById(@PathVariable("id") Long blogId){  
 boolean isRemoved = blogService.removeById(blogId);  
  
 Assert.*isTrue*(isRemoved,"删除失败！该博客可能已经被删除");  
 return CommonResult.*successResponse*(null,"删除成功！");  
 }  
 *//更具用户查博客* @RequiresAuthentication  
 @GetMapping("/getBlogsByUser/{username}")  
 public CommonResult getBlogsByUser(@PathVariable("username") String userName){  
 return blogService.getBlogsByUser(userName);  
 }  
  
}

### 4.1 用户控制类实现

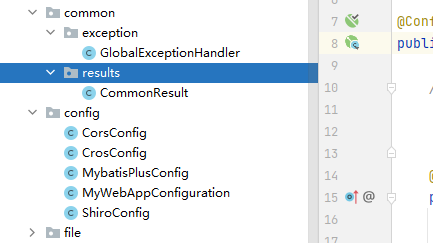
User entity实现：

package com.work.user.entity;  
  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import lombok.Data;  
import lombok.EqualsAndHashCode;  
import lombok.experimental.Accessors;  
  
import java.io.Serializable;  
import java.time.LocalDateTime;  
  
*/\*\*  
 \* <p>  
 \*  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-27  
 \*/*@Data  
@EqualsAndHashCode(callSuper = false)  
@Accessors(chain = true)  
@TableName("m\_user")  
  
public class User implements Serializable {  
  
 private static final long *serialVersionUID* = 1L;  
  
 @TableId(value = "id", type = IdType.*AUTO*)  
 private Long id;  
  
 private String username;  
  
 private String avatar;  
  
 private String email;  
  
 private String password;  
  
 private Integer status;  
 @JsonFormat(shape = JsonFormat.Shape.*STRING*,pattern = "yyyy-MM-dd",timezone = "GMT+8")  
 private LocalDateTime created;  
 @JsonFormat(shape = JsonFormat.Shape.*STRING*,pattern = "yyyy-MM-dd",timezone = "GMT+8")  
 private LocalDateTime lastLogin;  
  
 @TableField(exist = false)  
 private String token;  
  
}

blog entity实现：

package com.work.blog.entity;  
  
import com.baomidou.mybatisplus.annotation.\*;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import lombok.Data;  
import lombok.EqualsAndHashCode;  
import lombok.experimental.Accessors;  
  
import java.io.Serializable;  
import java.time.LocalDateTime;  
  
*/\*\*  
 \* <p>  
 \*  
 \* </p>  
 \*  
 \* @author jian  
 \* @since 2021-05-28  
 \*/*@Data  
@EqualsAndHashCode(callSuper = false)  
@Accessors(chain = true)  
@TableName("m\_blog")  
public class Blog implements Serializable {  
  
 private static final long *serialVersionUID* = 1L;  
  
 @TableId(value = "id", type = IdType.*AUTO*)  
 private Long id;  
  
 private Long userId;  
  
 private String title;  
  
 private String description;  
  
 private String content;  
  
 @JsonFormat(shape = JsonFormat.Shape.*STRING*,pattern = "yyyy-MM-dd",timezone = "GMT+8")  
 @TableField(fill = FieldFill.*INSERT*)  
 private LocalDateTime gmtCreate;  
  
 @JsonFormat(shape = JsonFormat.Shape.*STRING*,pattern = "yyyy-MM-dd",timezone = "GMT+8")  
 @TableField(fill = FieldFill.*INSERT\_UPDATE*)  
 private LocalDateTime gmtModified;  
 private Integer status;  
 @TableLogic  
 private Integer deleted;  
 private String author;  
}

## 5 系统配置



exception中是自定义的异常，result 中是接口返回给前端的结果封装，给予关键字，前端方便绑定数据。

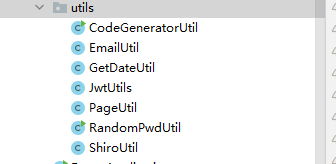
CrosConfig ：跨域请求配置

MybatisPlusConfig ：MyBatisPlus分页的配置

MyWebAppConfiguration：静态资源上传配置

ShiroConfig：用户管理框架shiro配置

工具类：



EmailUtil：邮件发送

GetDateUtil：从网络中获取当前时间

JwtUtils：用户登录后返回给前端的token

PageUtil：分页工具

RandomPwdUtil：产生随机密码

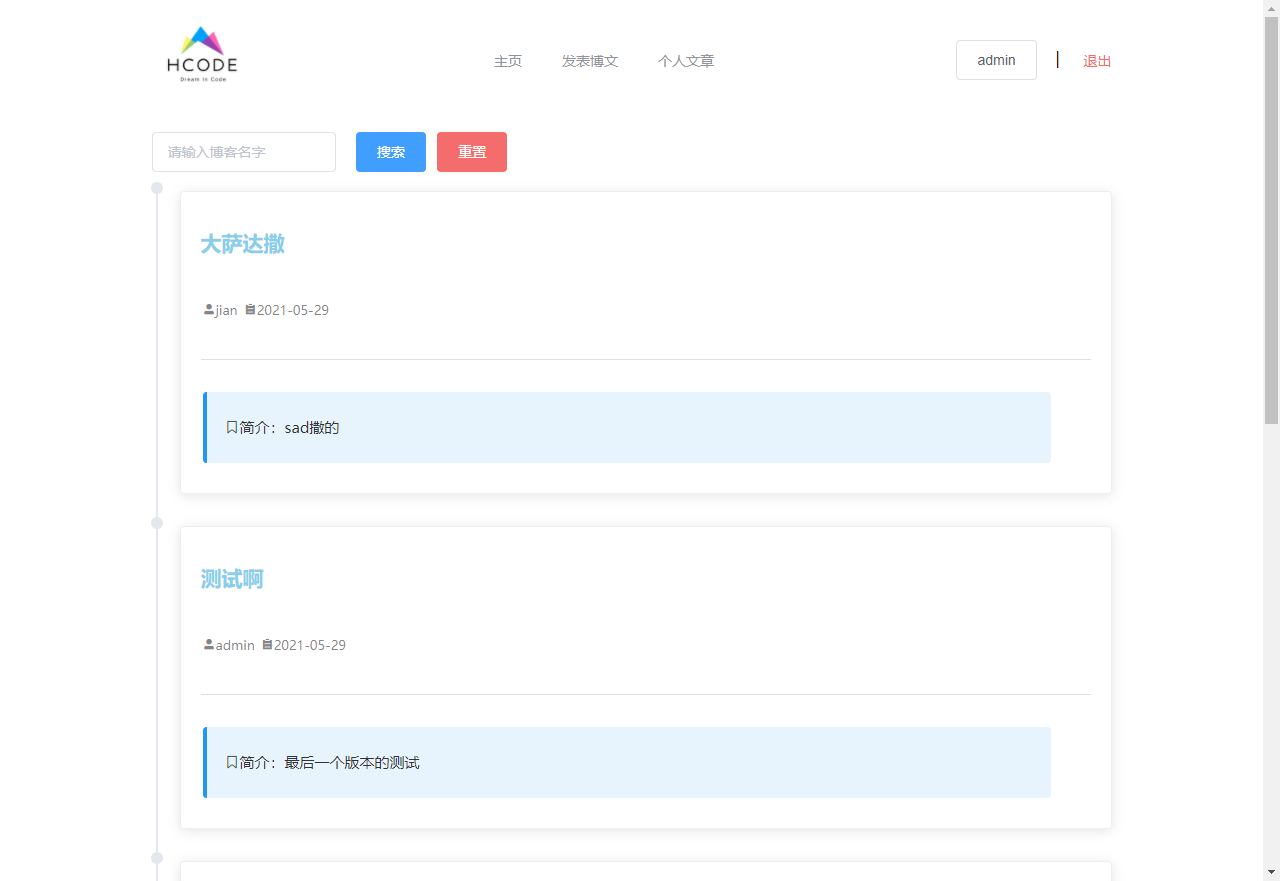
ShiroUtil：工具

## 6 视图层设计

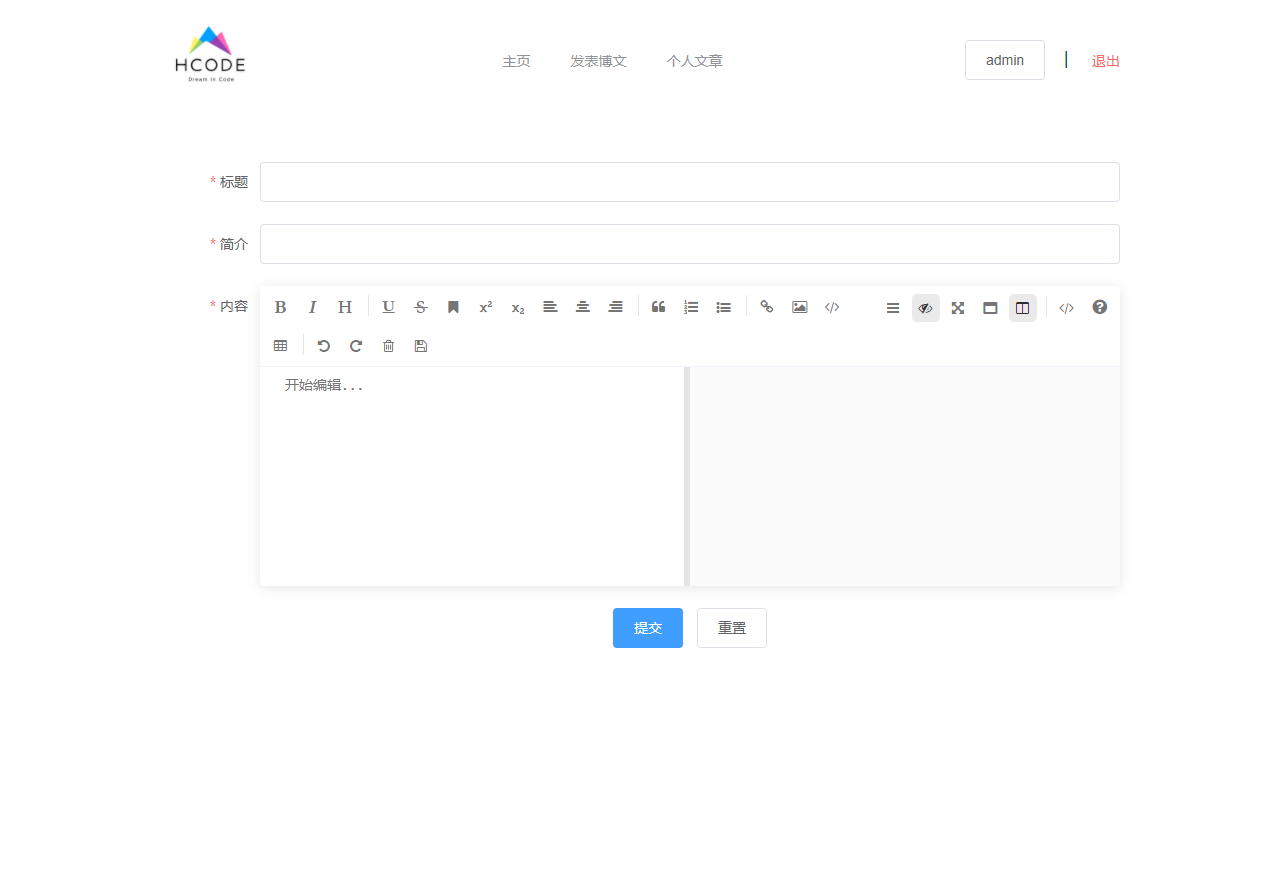
视图层主要采用Vue.js+element-ui实现，数据请求使用axios，得到json数据使用js与控件绑定

### 6.1 界面的总体设计

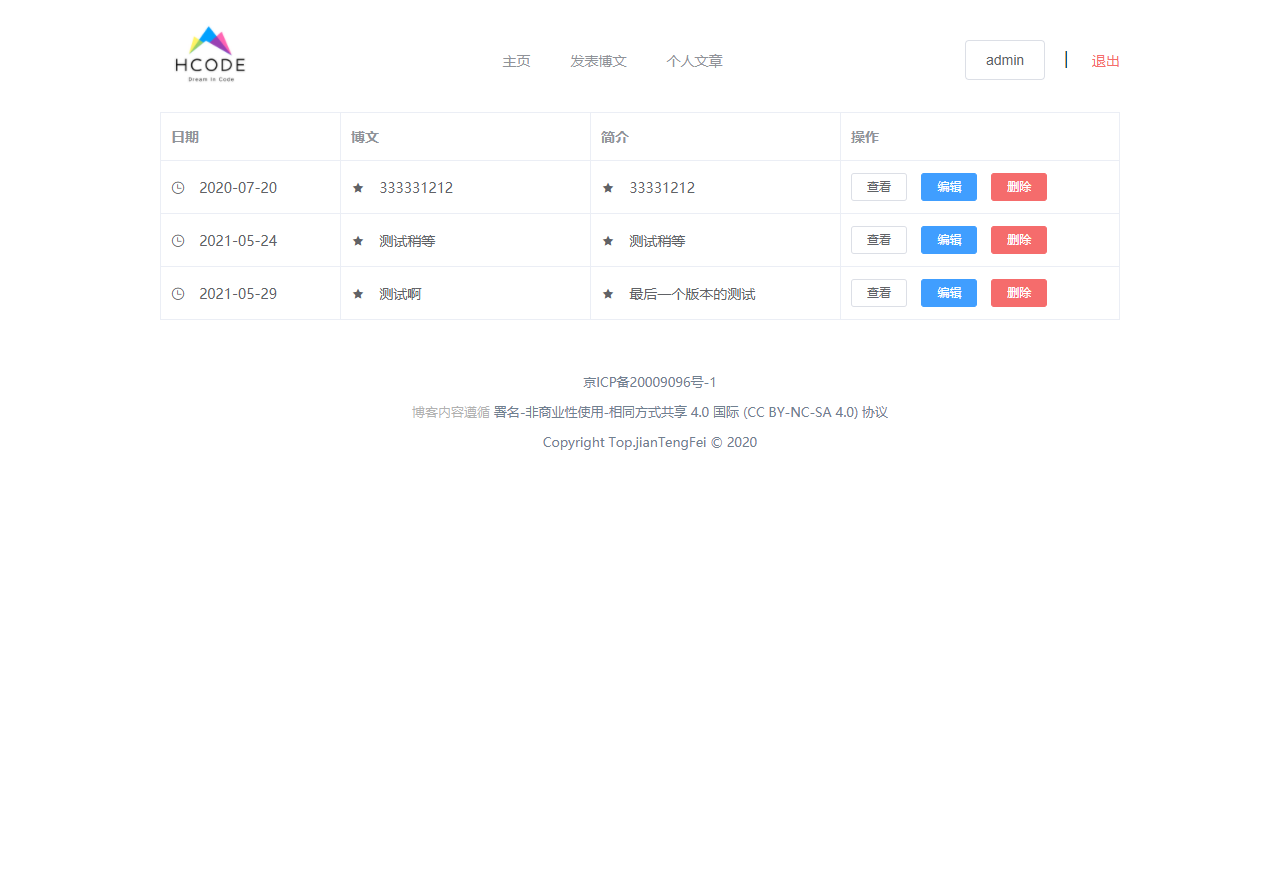
1. 主页：



1. 博客撰写/修改页面：



1. 个人文章管理：



1. 博客查看：



1. 登录：



1. 忘记密码：



1. 注册：



### 6.2 用户视图设计

用户视图层是展示给用户看的，用户视图层展示相关功能给用户看的，接收用户输入的内容，比如用户通过注册功能，输入用户名和密码，用户视图层也可以校验简单的逻辑，比如用户注册时两次输入的密码是否一致；

1.找回密码：



2.登录

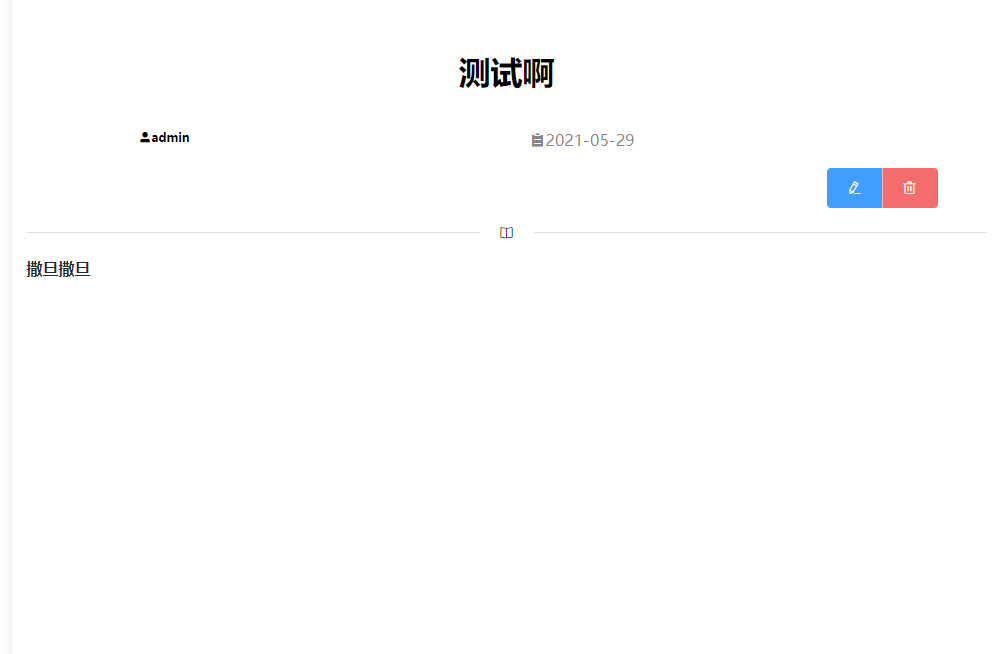


3.注册：



4.博客查看

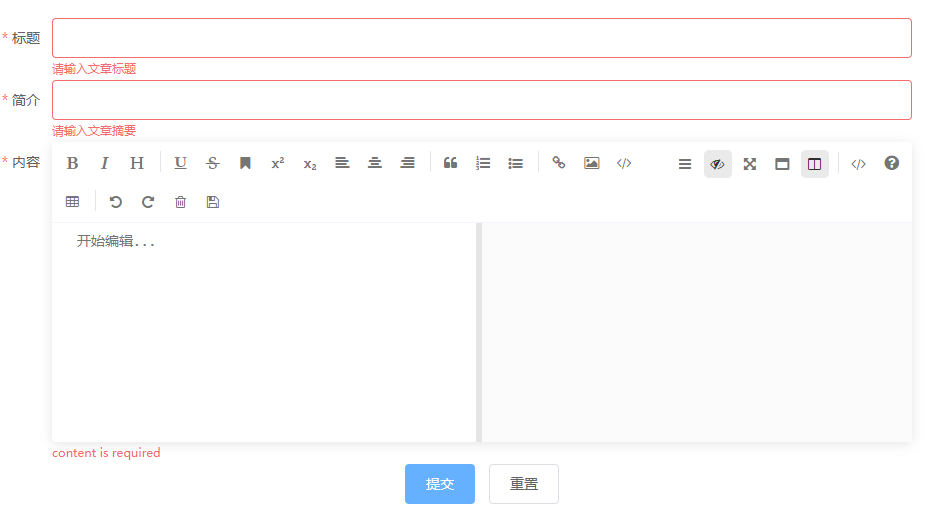
当博客属于当前用户时：会有编译和删除



当博客不属于当前用户时：



5.文章提交校验:



## 7 总结及自我评价

通过本学期该课程的学习，我了解到了Java企业级应用的设计模式，已经所使用的的框架，包括MyBatis，springsecurity，Spring等，还有对数据库的一些操作框架，例如jpa，jdbcTemplate。本次作业是通过自己学习一步一步的实现，css是借用别人的，因为css我还是不太会使用，如果自己写会出现一些布局乱套的现象。本次作业设计比较简单，是我不足的地方，但是采用了前后端分离的框架，是前后端更加专注于做自己的事情。用户管理采用的shiro框架，配置起来比较麻烦，同时系统缺少权限管理，这是后续待完善的地方。感谢杨老师对我们的悉心指导。希望未来对Java后端开发有更深的了解。