个人技术博客文档

1.项目使用技术

1. Spring Boot（最新 Spring Boot 3.x） -> 要求最低 JDK 17+
2. Spring Web（Spring MVC）
3. MyBatis
4. MyBatis Plus
5. MySQL（最新版 8.x）
6. Redis（3.x）-> 存储验证码/存储用户登录信息

1.1 项目前端资源文件

> 链接：https://pan.baidu.com/s/1ldn_c5ioi1OCawaoYNuWPQ
>
> 提取码：gee1

1.2 项目源码

> https://gitee.com/mydb/blog-springboot-mp

1.3 所需软件和安装包

所需软件如下，以下软件包都为 Windows 系统的安装包。

1.3.1 MySQL 服务器

Windows 平台安装包：mysql-installer-community-5.7.27.0.msi

右键管理员运行并安装，一路下一步即可。

> 下载地址（阿里云盘）：https://www.aliyundrive.com/s/K4EtYBw8EtS 提取码: 1kt0

1.3.2 Redis 服务器

使用微软提供的 Windows 安装包：Redis-x64-3.0.504.zip 解压即可使用。

1.3.3 JDK

Java 运行和开发环境，文件名：jdk-17_windows-x64_bin.exe

> 下载地址（阿里云盘）：https://www.aliyundrive.com/s/K4EtYBw8EtS 提取码: 1kt0

1.3.3 Idea 编写 Java 代码

文件名：idealU-2023.1.2.exe

需要破解，点击免费使用一个月，一个月 google 搜索最新的 Idea 激活码填上去即可。

> 下载地址（阿里云盘）：https://www.aliyundrive.com/s/K4EtYBw8EtS 提取码: 1kt0

1.3.4 Navicat（非必须，选装）

可以方便的操作 MySQL，如果 SQL 能力比较好，可以使用 MySQL 自带的控制台。

2.项目交互流程和页面展示

1、所有人实现进入到文章列表页，展示所有人发布的文章

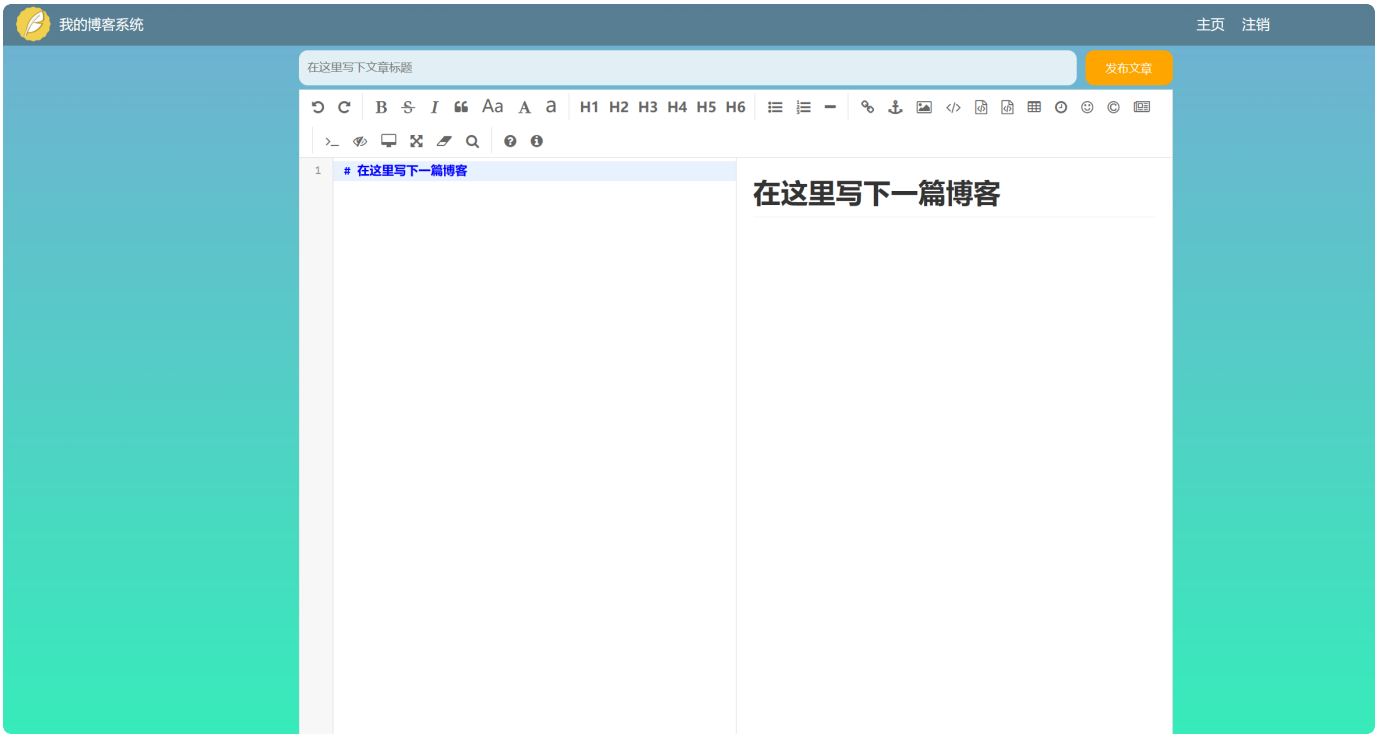
2、点击详情查看文章正文

3、点击注册，在博客平台注册账号（之后就可以发布文章、管理文章、发评论等操作了）

登录和注册都需要每次生成不同的验证码，并且验证成功之后才能执行业务逻辑。

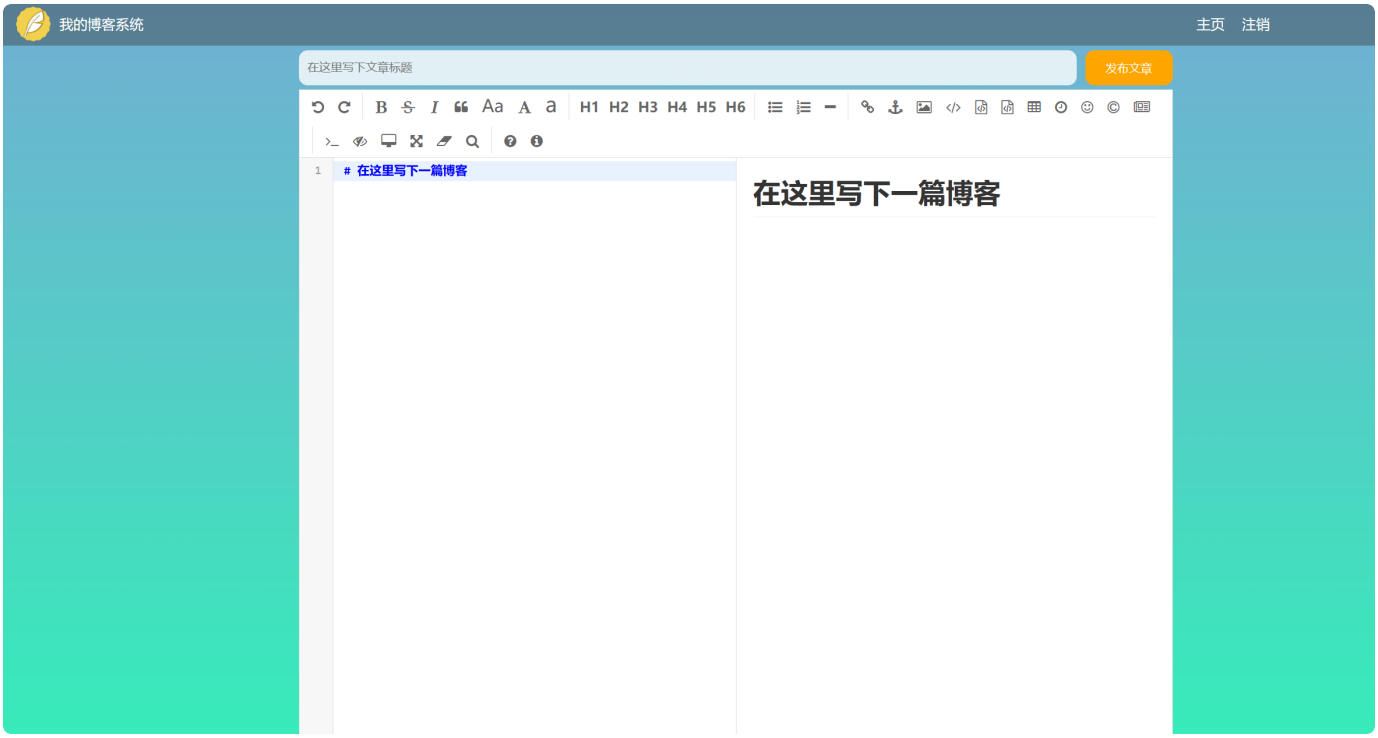
4、注册成功之后就可以登录了

5、登录完成，可以发表文章



6、发表完文章可以在我的文章列表中查看和管理

7、点击修改，到修改页面

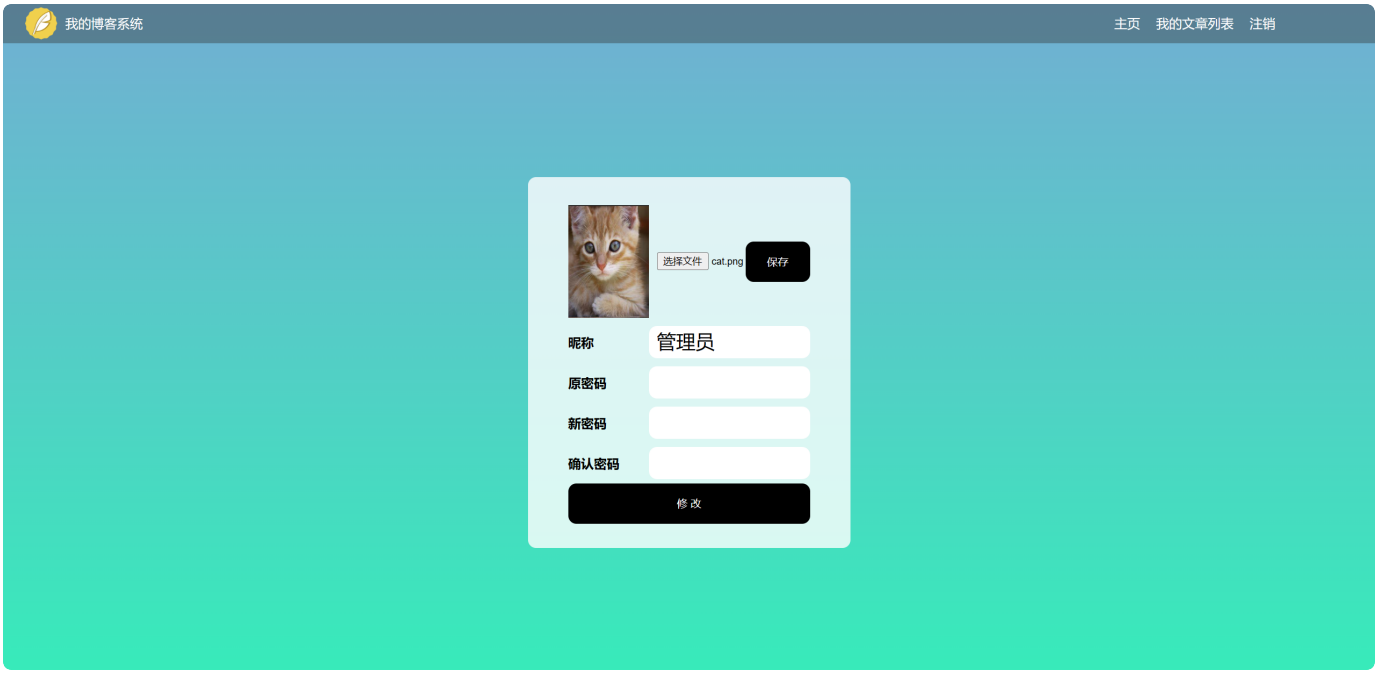


8、删除操作

删除操作，需要验证归属人，需要提供防止误删的功能。

9、评论相关功能

10、个人中心



3.数据库分析和设计

1. **用户表**
    a. 用户主键（用户编号）
    b. 登录用户名（需要添加唯一约束）
    c. 昵称
    d. 密码

e. 个人 github 地址
f. 头像
g. 状态（隐式字段）【正常、异常、被永久冻结、被临时冻结】

2. 文章表

a. 文章主键
b. 标题
c. 创建时间
d. 修改时间
e. 文章简介
f. 文章正文
g. 状态（隐私字段）【已发布、草稿】
h. 用户主键（文章作者id）
i. 访问量

3. 评论表

a. 评论表主键
b. 文章编号
c. 用户主键（评论发表人是谁）
d. 评论的正文
e. 评论发表时间

SQL 脚本

1.创建用户表

```sql
-- 创建用户表
create table userinfo(
  uid bigint auto_increment primary key comment '主键',
  loginname varchar(50) not null unique comment '登录用户名',
  nickname varchar(50) default '' comment '昵称',
  `password` varchar(65) not null comment '密码',
  github varchar(255) comment 'github地址',
  photo varchar(255) comment '头像',
  `state` tinyint not null DEFAULT 1 comment '用户状态，1=正常|-1=异常|-2=永久冻结|-3=临时冻结'
);
insert into userinfo(loginname,`password`) values('admin','admin');
```

2.创建文章表

```sql
create table articleinfo(
  aid BIGINT auto_increment primary key comment '主键',
  `title` varchar(255)  not null comment '标题',
  createtime TIMESTAMP not null default CURRENT_TIMESTAMP() comment '创建时间',
  updatetime TIMESTAMP not null DEFAULT CURRENT_TIMESTAMP() comment '修改时间',
  `desc` varchar(255) not null comment '文章简介',
  `content` longtext not null comment '文章正文',
  `state` TINYINT DEFAULT -1 comment '状态：-1=草稿|1=已发布',
  uid bigint not null comment '作者id',
  `rcount` bigint default 1 comment '阅读量'
);
```

3.创建评论表

```java
-- 创建评论表
create table commentinfo(
    cid bigint auto_increment primary key comment '评论表的主键',
    aid bigint not null comment '文章表id',
    uid bigint not null comment '用户id',
    `content` varchar(500) not null comment '评论正文',
    createtime TIMESTAMP default CURRENT_TIMESTAMP() comment '评论的发表时间'
);
```

4.核心实现代码

4.1 项目常用配置

```properties
# 1.配置 MySQL 连接信息
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/mycnblog?characterEncoding=utf8
spring.datasource.username=root
# 注意：下面是你自己服务器的 MySQL 密码
spring.datasource.password=12345678
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# 2.配置 MyBatis XML 保存目录
mybatis.mapper-locations=classpath:mybatis/*Mapper.xml
# 配置 MyBatis 打印 SQL 日志
mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
logging.level.com.javacn.myblog=debug
# 配置 Redis 连接信息
spring.session.store-type=redis
spring.data.redis.host=127.0.0.1
spring.data.redis.password=
spring.data.redis.port=6379
server.servlet.session.timeout=1800
spring.session.redis.flush-mode=on_save
spring.session.redis.namespace=spring:session
```

4.2 MyBatis XML 模版

XML 复制代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="">

</mapper>
```

4.3 前后端交互代码

JavaScript 复制代码

```javascript
jQuery.ajax({
    url:"/user/reg",
    type:"POST",
    data:{
        "loginname":username.val(),
        "password":password.val(),
        "checkCode":checkcode.val()
    },success:function (res){
        // 接受返回结果
        // 4.根据返回的结果，将结果呈现给用户
        if(res.code==200 && res.data==1){
            // 添加成功
            alert("恭喜：注册成功！");
            location.href = "login.html"; // 跳转到登录页面
        }else{
            alert("抱歉：操作失败！"+res.msg);
        }
    }
});
```

4.4 用户权限效验拦截器实现代码

拦截器：

```java
@Component
public class LoginInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        // 1.先得到 HttpSession 对象
        HttpSession session = request.getSession(false);
        if(session!=null && session.getAttribute(AppVar.SESSION_KEY_USERINFO)!=null){
            // 已经登录
            return true;
        }
        // 代码执行到此处，说明用户未登录
        response.sendRedirect("/login.html");
        return false;
    }
}
```

拦截规则：

```java
@Configuration
public class MyConfig implements WebMvcConfigurer {
    @Resource
    private LoginInterceptor loginInterceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(loginInterceptor)
                .addPathPatterns("/**") // 拦截所有请求
                .excludePathPatterns("/user/reg") // 排除注册接口（注册接口不拦截）
                .excludePathPatterns("/user/login")
                .excludePathPatterns("/user/getuser")
                .excludePathPatterns("/**/*.html")
                .excludePathPatterns("/css/**")
                .excludePathPatterns("/editor.md/**")
                .excludePathPatterns("/img/**")
                .excludePathPatterns("/js/**");
    }
}
```

4.5 密码加盐实现代码

Java | 复制代码

```java
public class PasswordUtil {

    /**
     * 加盐算法 -> 格式：盐值（32）$加密之后的密码（32）
     * @param password 原密码
     * @return
     */
    public static String encrypt(String password){
        // 1.生成盐值
        String salt = UUID.randomUUID().toString().replace("-","");
        // 2.使用加密算法将盐值+原密码进行加密
        String finalPassword = DigestUtils.md5DigestAsHex((salt+password).getBytes());
        // 3.将盐值和加密后的密码一起返回
        String dbPassword = salt+"$"+finalPassword;
        return dbPassword;
    }

    /**
     * 密码验证
     * @param inputPassword
     * @param dbPassword
     * @return
     */
    public static boolean decrypt(String inputPassword,String dbPassword){
        // 1.验证参数
        if(!StringUtils.hasLength(inputPassword) || !StringUtils.hasLength(dbPassword) ||
                dbPassword.length()!=65 || !dbPassword.contains("$")){
            return false;
        }
        // 2.将用户输入的密码和数据库的盐值进行加密，得到待验证的加密密码
        // 2.1 得到盐值 & 最终正确的密码
        String[] dbPasswordArray = dbPassword.split("\\$");
        String salt = dbPasswordArray[0];
        String finalPassword = dbPasswordArray[1];
        // 2.2 使用数据库的盐值+用户输入的密码进行加密=待验证的加密密码
        String userPassword = DigestUtils.md5DigestAsHex((salt+inputPassword).getBytes());
        // 3.将待验证密的加密密码和数据的加密的密码进行对比
        if(userPassword.equals(finalPassword)){
            return true;
        }
        // 4.将结果返回给调用方
        return false;
```

```
        }
    }
```

4.6 验证码

生成验证码：

Java | 复制代码

```java
@RestController
public class CaptchaController {

    @Value("${imagepath}")
    private String imagepath;
    @Autowired
    private RedisTemplate redisTemplate;

    @RequestMapping("/getcaptcha")
    public AjaxResult getCaptcha(){
        // 1.生成验证码到本地
        //定义图形验证码的长和宽
        LineCaptcha lineCaptcha = CaptchaUtil.createLineCaptcha(128, 50);
        String uuid = UUID.randomUUID().toString().replace("-","");
        // 图形验证码写出，可以写出到文件，也可以写出到流
        lineCaptcha.write(imagepath+uuid+".png");
        // url 地址
        String url = "/image/"+uuid+".png";
        // 将验证码存储到 redis
        redisTemplate.opsForValue().set(uuid,lineCaptcha.getCode());
        HashMap<String,String> result = new HashMap<>();
        result.put("codeurl",url);
        result.put("codekey",uuid);
        return AjaxResult.succ(result);
    }

}
```

配置本地图片的拦截映射规则：

```java
@Value("${imgpath}")
private String imagePath;

/**
 * 映射图片路径
 * @param registry
 */
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/image/**").addResourceLocations("file:"
+ imagePath);
}
```

5.项目可扩展功能

1. 文章草稿保存功能
2. 文章定时发布功能
3. 用户多次登录，账号冻结的业务
4. 找回密码功能（可通过邮件或短信验证码）
5. 文章点赞/踩
6. 注册成功之后，发送欢迎邮件

6.框架技术升级

1. 引入 Validator 框架，验证参数
2. 添加 XXL-Job 实现文章定时发布
3. 引入消息队列，登录时记录日志和安全验证（登录次数 + IP 黑名单 + 异地登录等）