**ourBlog设计报告**

**1、介绍**

**1.1、项目简介**

Vue+Spring Boot 前后端分离博客系统。

[**项目网站**](http://111.229.31.179/)

**需求分析与设计报告在word中**

**1.2、主要功能**

* 用户注册，登录，登出
* 使用markdown方法编辑输入文章
* 文章上传，可设置文章分类、标tag
* 文章可归档整理，按日期、标签、分类分页查看
* 根据点击次数添加热度功能
* 个性化，主题背景切换
* 评论系统
* 个性化，主题背景切换

**2、软件架构**

**2.1、后端技术**

| **技术** | **说明** |
| --- | --- |
| maven | 项目管理和构建自动化工具 |
| MyBatis-Plus | 代码生成器、性能分析插件、自动填充 |
| Spring Web | Spring Framework的Web模块 |
| Lombok | 简化对象封装工具 |
| jwt | 身份验证和信息交换 |

**2.2、前端技术**

| **技术** | **说明** |
| --- | --- |
| Vue.js | 响应式和组件化特性构建用户界面 |
| Element UI | 提供美观、实用的界面元素 |
| Spring Web | Spring Framework的Web模块 |
| Lombok | 简化对象封装工具 |
| jwt | 身份验证和信息交换 |

**3、使用说明**

**3.1、依赖配置**

**用maven配置后端起步依赖，测试jdk版本为11可供参考**

**3.2、数据库配置**

**ourBlog.sql导入数据库，后端项目的application.yml中配置你的数据库配置。**

datasource:

url: jdbc:mysql://# {本地的IP地址}:3306/{数据库名字}?useUnicode=true&characterEncoding=UTF-8&serverTimeZone=UTC

username: *#数据库用户名*

password: *#数据库连接密码*

driver-class-name: com.mysql.cj.jdbc.Driver

**前端运行**

依次在项目终端执行:

npm install

npm run dev (本地运行)

npm run build (打包)

**4、前端组件和技术**

un build (打包)

* **BaseHeader 组件**
  + **使用 Vue.js 和 Element UI 创建响应式的导航栏，集成了用户登录状态和主要导航链接。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-header>**

**<base-header :simple=true>**

**<!-- Navigation Links -->**

**</base-header>**

**</el-header>**

**</template>**

* **Index 组件**
  + **结合 Vue.js 和 Element UI，整合博客的核心功能区域，如文章列表和归档。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-container>**

**<el-main>**

**<article-scroll-page></article-scroll-page>**

**</el-main>**

**</el-container>**

**</template>**

* **Register 组件**
  + **利用 Vue.js 和 Element UI 的表单验证，实现用户注册功能，并通过 Vuex 管理注册状态。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-form ref="userForm" :model="userForm" :rules="rules">**

**<el-form-item prop="account">**

**<el-input placeholder="用户名" v-model="userForm.account"></el-input>**

**</el-form-item>**

**<!-- Other Form Items -->**

**</el-form>**

**</template>**

* **Login 组件**
  + **同样采用 Vue.js 和 Element UI，实现安全的登录功能，并使用 Vuex 维护登录状态。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-form ref="userForm" :model="userForm" :rules="rules">**

**<el-form-item prop="password">**

**<el-input placeholder="密码" type="password" v-model="userForm.password"></el-input>**

**</el-form-item>**

**<!-- Other Form Items -->**

**</el-form>**

**</template>**

* **Home 组件**
  + **作为应用的入口，整合 BaseHeader 和 BaseFooter 组件，使用 Vue Router 展示动态内容。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-container>**

**<base-header :activeIndex="activeIndex"></base-header>**

**<router-view class="me-container"/>**

**<base-footer v-show="footerShow"></base-footer>**

**</el-container>**

**</template>**

* **BlogAllCategoryTag 组件**
  + **通过 Vue.js 和 Element UI 的选项卡和列表组件，展示文章分类和标签，实现数据的动态加载和页面跳转。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-tabs v-model="activeName">**

**<el-tab-pane label="文章分类" name="category">**

**<ul class="me-allct-items">**

**<li v-for="c in categorys" :key="c.id" @click="view(c.id)" class="me-allct-item">**

**<!-- Category Item -->**

**</li>**

**</ul>**

**</el-tab-pane>**

**<!-- Other Tab Panes -->**

**</el-tabs>**

**</template>**

* **BlogArchive 组件**
  + **使用 Vue.js 和 Element UI 的列表和徽章组件，展示文章归档，并通过 API 获取数据。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-aside>**

**<ul class="me-month-list">**

**<li v-for="a in archives" :key="a.year + a.month" class="me-month-item">**

**<el-badge :value="a.count">**

**<el-button @click="changeArchive(a.year, a.month)">{{a.year +'年' + a.month + '月'}}</el-button>**

**</el-badge>**

**</li>**

**</ul>**

**</el-aside>**

**</template>**

* **BlogView 组件**
  + **集成 Markdown 编辑器和评论列表，使用 Vue.js 实现文章内容的展示和用户交互。**
  + **重点代码片段：**

**vue**

**<template>**

**<div class="me-view-body">**

**<div class="me-view-card">**

**<h1 class="me-view-title">{{article.title}}</h1>**

**<div class="me-view-author">**

**<img class="me-view-picture" :src="article.author.avatar"/>**

**<div class="me-view-info">**

**<span>{{article.author}}</span>**

**<!-- Other Author Info -->**

**</div>**

**</div>**

**<div class="me-view-content">**

**<markdown-editor :editor=article.editor></markdown-editor>**

**</div>**

**<!-- Comments and Other Info -->**

**</div>**

**</div>**

**</template>**

* **BlogWrite 组件**
  + **提供富文本编辑功能，结合 Vue.js 和 Element UI 的表单和对话框组件，实现文章创作和发布。**
  + **重点代码片段：**

**vue**

**<template>**

**<el-main>**

**<div class="me-write-title">**

**<el-input type="textarea" autosize v-model="articleForm.title" placeholder="请输入标题"></el-input>**

**</div>**

**<markdown-editor :editor="articleForm.editor"></markdown-editor>**

**<el-dialog title="摘要 分类 标签" :visible.sync="publishVisible">**

**<el-form :model="articleForm" ref="articleForm" :rules="rules">**

**<el-form-item prop="summary">**

**<el-input type="textarea" v-model="articleForm.summary" placeholder="请输入摘要"></el-input>**

**</el-form-item>**

**<!-- Other Form Items -->**

**</el-form>**

**<div slot="footer">**

**<el-button @click="publishVisible = false">取消</el-button>**

**<el-button type="primary" @click="publish('articleForm')">发布</el-button>**

**</div>**

**</el-dialog>**

**</el-main>**

**</template>**

* **App 组件**
  + **作为根组件，整合 GoTop 组件，使用 Vue.js 设置全局样式，确保应用的视觉效果和布局一致性。**
  + **重点代码片段：**

**vue**

**<template>**

**<div id="app">**

**<router-view/>**

**<go-top></go-top>**

**</div>**

**</template>**

**5、后端架构和技术**

**common**

通常包含项目中通用的类和接口：aop包含与面向切面编程相关的类，cache包含与缓存相关的配置

**handler**

包含异常处理器和登入拦截器等处理请求和响应的类。

**controller-service(imp)-dao**

请求响应三层架构

**utils**

包含工具类和帮助类，这些类提供一些实用功能，如日期处理、加密、验证等。

**vo**

包含视图对象（View Object）相关的类，这些对象用于封装将要显示在用户界面上的数据。

**请求获取文章信息示例代码：**

**controller**

@RestController  
@RequestMapping("articles")  
public class ArticleController {  
  
 */\*\*  
 \* 首页 文章列表  
 \* @param pageParams  
 \* @return  
 \*/* @Autowired  
 private ArticleService articleService;  
  
 @LogAnnotation(module = "文章",operator = "获取文章列表")  
 @Cache(expire = 5 \* 60 \* 1000,name = "获取文章列表")  
 @PostMapping  
 public Result listArticle(@RequestBody PageParams pageParams){  
 //ArticleVo 页面接收的数据  
// Result articles = articleService.listArticle(pageParams);  
  
 return articleService.listArticle(pageParams);  
 }

}

**Service**

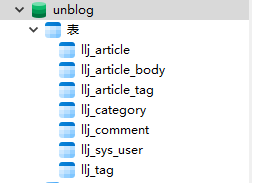
public Result listArticle(PageParams pageParams) {  
 Page<Article> page = new Page<>(pageParams.getPage(), pageParams.getPageSize());  
 LambdaQueryWrapper<Article> queryWrapper = new LambdaQueryWrapper<>();  
 if(pageParams.getCategoryId() != null){  
 queryWrapper.eq(Article::getCategoryId,pageParams.getCategoryId());  
 }  
 List<Long> tags = new ArrayList<>();  
 if(pageParams.getTagId() != null){  
 LambdaQueryWrapper<ArticleTag> qw = new LambdaQueryWrapper<>();  
 qw.eq(ArticleTag::getTagId,pageParams.getTagId());  
 List<ArticleTag> articleTags = articleTagMapper.selectList(qw);  
 for (ArticleTag articleTag : articleTags) {  
 tags.add(articleTag.getArticleId());  
 }  
 if(tags.size() > 0){  
 queryWrapper.in(Article::getId,tags);  
 }  
 }  
  
 // 是否置顶进行排序  
  
 if (!StringUtils.isBlank(pageParams.getYear())&&!StringUtils.isBlank(pageParams.getMonth())){  
 // 创建 SimpleDateFormat 用于解析和格式化日期  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
 try {  
 // 获取 2024年12月1日 00:00:00 的时间戳  
 Calendar calendar = Calendar.getInstance();  
 calendar.set(Integer.parseInt(pageParams.getYear()), Integer.parseInt(pageParams.getMonth()) - 1, 1, 0, 0, 0);  
 long startTime = calendar.getTimeInMillis();  
 // 获取 2024年12月31日 23:59:59 的时间戳  
 calendar.set(Integer.parseInt(pageParams.getYear()), Integer.parseInt(pageParams.getMonth()) - 1, 31, 23, 59, 59);  
 long endTime = calendar.getTimeInMillis();  
 queryWrapper.between(Article::getCreateDate,startTime,endTime);  
 }catch (Exception e){  
  
 }  
 }  
  
  
  
// 根据创建时间排序  
 queryWrapper.orderByDesc(Article::getWeight, Article::getCreateDate);  
 Page<Article> articlePage = articleMapper.selectPage(page, queryWrapper);  
 List<Article> records = articlePage.getRecords();  
 List<ArticleVo> articleVoList = copyList(records, true, true);  
 return Result.success(articleVoList);  
 }

**Mapper**

@Repository  
public interface ArticleBodyMapper extends BaseMapper<ArticleBody> {

//封装mybatis-plus  
}

**6、数据库**



llj\_article：文章基本信息

llij\_article\_body：存放文章内容

llj\_article\_tag：存放文章对应标签

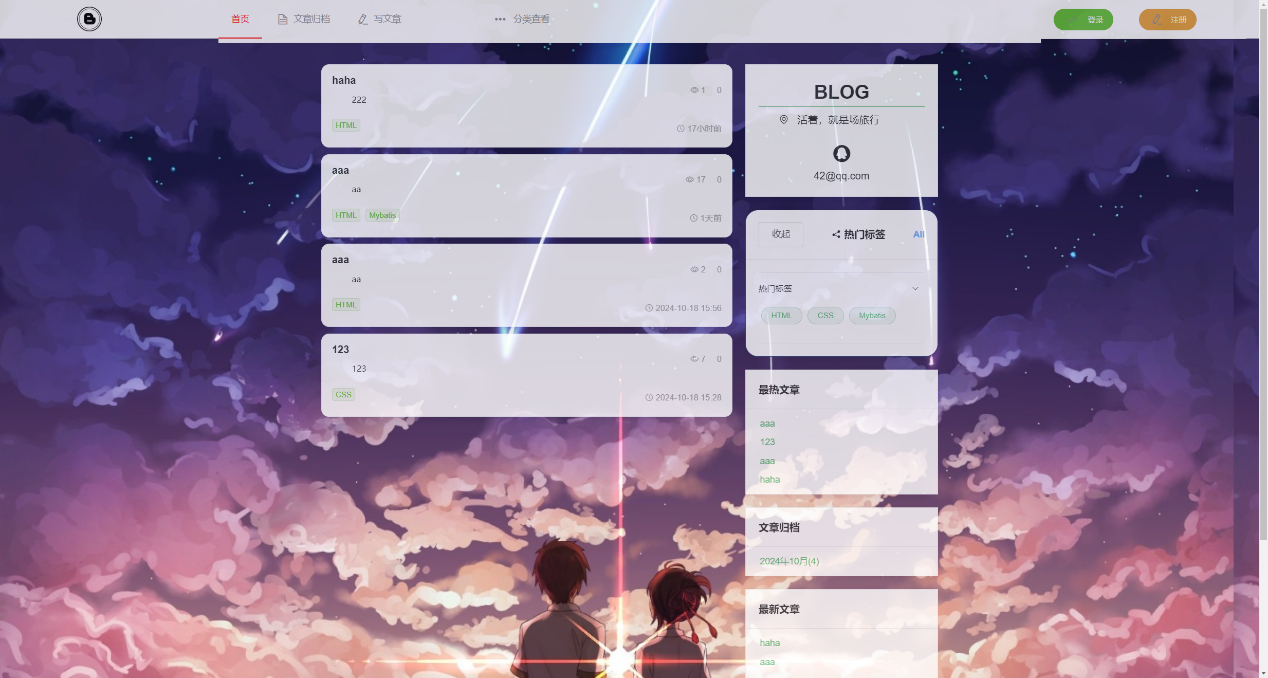
llij\_category：存放分类信息

llj\_comment：存放评论信息

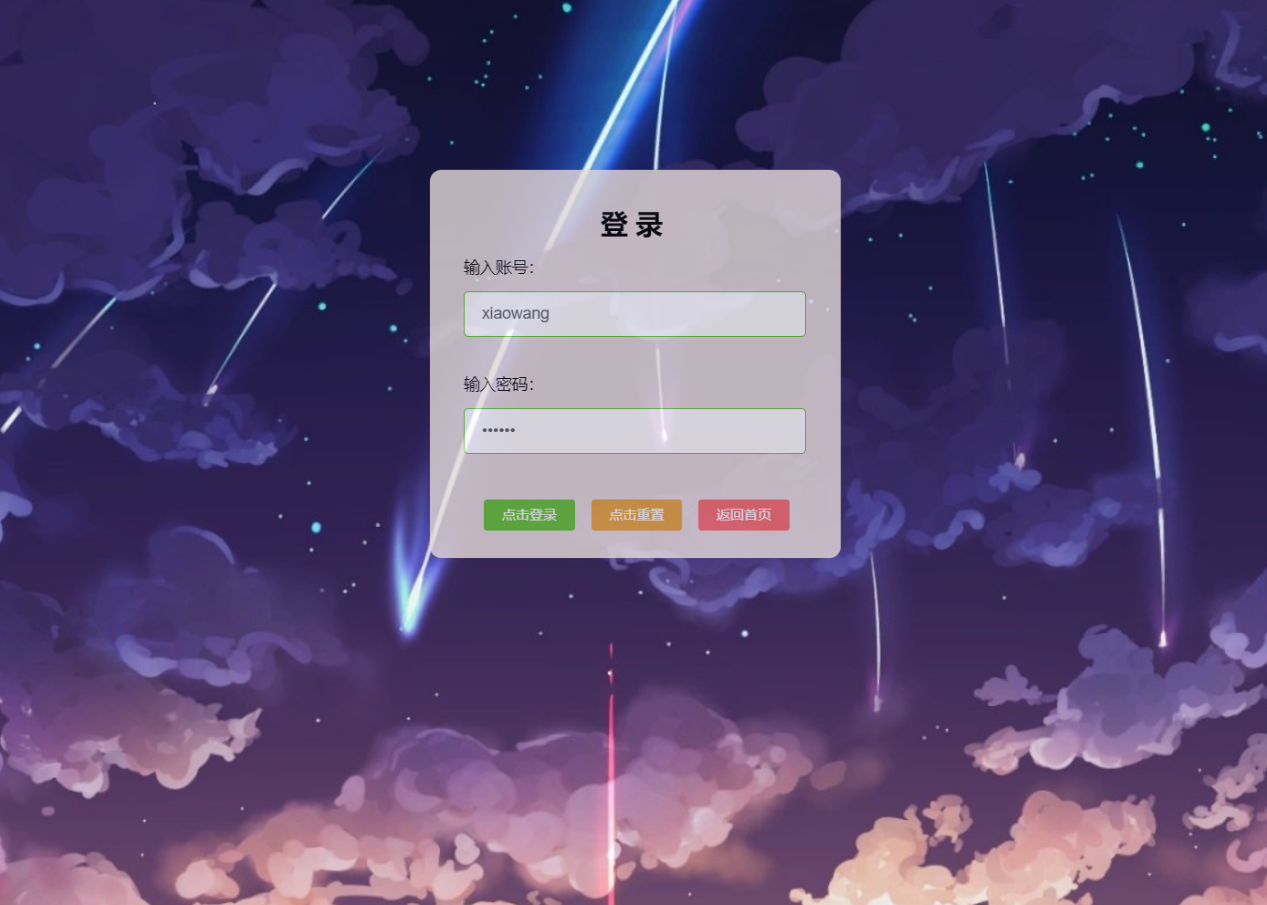
llj\_sys\_user：存放用户信息

llj\_tag：存放标签信息  
  
**7、项目演示**

1、主界面



2、登入界面



3、编辑文章界面



4、分类选择界面



5、评论界面



6、分类查询界面

