1. 任务概述
2. 目标

当今快节奏的社会中，人们往往忽略了文字的沉淀。该系统旨在搭建一个博客社区，给用户提供一个更多元化的文字交流平台，不仅仅是发博客，用户之间还可以进行更加现代化的互动。

1. 用户特点

该系统面向喜欢文字交流的年轻社群，尤其是能快速接受新鲜事物并且能够持续保持活跃状态的用户。

1. 需求分析
2. 基本需求
3. 功能需求

博客系统

登录注册

关注用户

通知模块

评论列表

日志编辑

日志查看

首页投稿

用户信息

博客系统

博客系统

1. 性能需求
2. 响应时间

规定服务器响应时间不超过0.5s，所以在初次加载时不应该出现白屏现象，给用户一个更好的体验。每次客户端向服务端请求时应在异步状态下进行操作。

1. 系统输入输出精度需求

规定用户输入空值时给出警告提示。

每次请求后的响应字段必须为全部可用字段，需要什么就响应什么，不能冗余，造成输出变大，响应时间变长这一问题。

1. 运行环境需求
2. 软件环境

操作系统：windows、linux、macOS

服务器：apache 、node v4.4.4（开发环境）

数据库环境：mysql（innoDB）

浏览器：chrome、firefox、safari

1. 硬件环境

处理器：AMD A6-4400M APU with Radeon HD Graphics

内存：6.00GB

系统类型：64位操作系统

1. E-r图
2. 用户实体

每个用户实体包括用户id，用户名（登录名），登录密码，昵称、性别、生日、常住地、常用邮箱、主页推荐标识符、获赞数、获得评论数等属性。

用户

id

password

sex

birth

username

isHot

articleNum

email

creatTime

commitNum

1. 通知实体

每个通知实体包括通知id、通知标题、通知类型（成功、失败和警告）、创建日期等属性。

通知

type

title

status

createTime

id

1. 文章实体

每个文章实体包括文章id、文章标题、内容、创建时间、作者名、获赞数、摘要等属性。

文章

writerId

likeNum

createTime

content

id

praiseList

shortCut

1. 评论实体

评论实体包括评论的id、对应的文章id、评论人的用户id、评论内容、创建时间等属性。

评论

createTime

content

userId

articleId

id

1. 系统设计
2. 流程图及流程说明

.1.登录注册

从首页点击登录进入登录页，填写用户名和登录密码进行登录；未注册则填写相关字段进行注册。

操作成功后跳转首页，进行其他操作；失败则继续当前操作直至成功或仅浏览首页。

进入首页

登陆页

进入其他模块

注册

已登录

未登录

注册后

未注册

1. 查看用户信息

点击用户头像，带着当前用户的id参数，跳转到个人信息页；

判断所带参数是否为当前用户，是则编辑个人的资料；否则浏览信息。

点击头像

是否为当前用户

浏览

编辑

保存

否

是

1. 日志查看/编辑

点击日志，带着用户id这个参数跳转到详情页；

进入后，判断是否为当前的日志作者：

是则进行管理、评论等操作；

否则只能评论点赞。

点击日志

评论

删除

当前用户是否为作者

编辑

保存

否

是

1. 通知

用户被关注或日志被点赞和评论时，插入通知队列中，当用户点击查看通知图标时，展示未读通知，关闭则从数据库里删除。

收到评论

点击图标

首页显示图标

未读消息条数增加

被关注

通知列表

1. 首页投稿

用户进入详情页，判断当前用户与文章作者是否一致：

是则通过服务器发来的isHot字段判断投稿状态，进行投稿操作；

否则不显示投稿按钮

管理员进行筛选，将投稿的文章显示在首页

否

点击投稿

显示通知

显示通知

系统管理员审核

首页显示

投稿按钮消失

（服务器）发送

isHot

（创作者）进入文章页

否

是

是

1. 算法设计
2. MVC设计模式

大多数开发者都知道M是数据模型,V是视图，C是控制器。在MVC里，View是可以直接访问Model的！从而，View里会包含Model信息，不可避免的还要包括一些业务逻辑。 MVC模型关注的是Model的不变，所以，在MVC模型里，Model不依赖于View，但是 View是依赖于Model的。不仅如此，因为有一些业务逻辑在View里实现了，导致要更改View也是比较困难的，至少那些业务逻辑是无法重用的。

1. MVVM设计模式

MVVM在概念上是真正将页面与数据逻辑分离的模式，它把数据绑定工作放到一个JS里去实现，而这个JS文件的主要功能是完成数据的绑定，即把model绑定到UI的元素上。

此外，MVVM另一个重要特性，双向绑定。它更方便你同时维护页面上都依赖于某个字段的N个区域，而不用手动更新它们。

1. 系统交互模式设计

开发环境下：

在开发环境下，vue文件需要被webpack编译，而webpack则依赖于node作为服务器来进行热加载，所以ajax发送的请求跨域，需要在服务端写上跨域头（即cors）

apache

vue

php

ajax

node

webpack

生产环境下：

通过webpack打包后的vue文件则不需要webpack-dev-server来实施热加载了，直接把他放到apache下就好，这时不存在跨域的问题。

ajax

Html、js、css

apache

php

1. 技术选型
2. Vue

vue是是一套构建用户界面的渐进式框架，在2016年与react、angular并称为三大框架之一的前端开发框架，凭借其详细的官方文档、简单灵活的设计模式和高效率的开发流程，深受我国前端开发者的热爱和推崇。

Vue压缩后只有17kb，它采用自底向上增量开发的设计，只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。

1. Webpack

Webpack也是当下最流行的模块加载器和打包工具，他可以把js、jsx、css等前端资源文件打包为模块进行使用。在本系统中，因为使用到了vue文件，wepack的vue-loader将会自动将他编译成js，开发者不必进行各种配置。

1. Ajax

Ajax在近些年来已经成为与服务端交互工具的不二之选，使用ajax能够使代码更明确，前后端分离更清晰；最大的特点是异步，使服务器减轻负担，可以给用户更好的体验。

1. Apache+php+mysql

这套技术栈已经流行了很多年，主要是因为php对mysql都有非常友好的扩展。

Php稳定高效，最大的优点是在apache上不用配置很多，开发速率很快，代码简单易读，很适合中小型项目。

1. 开发工具
2. Webstorm

Webstorm是前端最优秀的ide，它集成了node、npm、webpack、eslint等各种js轮子的配置，自动纠错，代码高亮。

1. Phpstorm

看名字就知道和webstorm是来自一家公司，都是jet brain的旗舰产品，使开发过程更高效。

1. Sublime text3

Sublime是拥有非常多插件的一款文本编辑器，即使插件再多也不会造成卡顿的现象，使用手感非常流畅。

1. 数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | 字段名（主键加粗） | 字段类型 | 是否可为空（默认不可为空） | 描述 |
| notification | **Id** | Int(20) |  | 通知id |
|  | userid | Int(11) |  | 用户id |
|  | Title | Char(20) |  | 通知内容 |
|  | Type | Int(1) |  | 通知类型  1：成功  0：失败  -1：警告 |
|  | createTime | timestamp |  | 创建时间 |
| comment | **id** | Int(20) |  | 评论id |
|  | articleId | Int(11) |  | 文章id |
|  | userId | Int(11) |  | 用户id |
|  | content | Varchar(100) |  | 评论内容 |
|  | createTime | timestamp |  | 创建时间 |
| article | id | Int(20) |  | 文章id |
|  | userid | Int(11) |  | 作者id |
|  | username | Varchar(10) |  | 作者昵称 |
|  | title | Varchar(20) |  | 文章标题 |
|  | content | text |  | 文章内容 |
|  | categoryId | Int(10) |  | 分类id |
|  | likeNum | Int(3) |  | 喜欢人数 |
|  | commentNum | Int(3) |  | 评论数 |
|  | isHot | Smallint(1) |  | 是否推荐到首页 |
|  | createTime | timestamp |  | 创建时间 |
| attention | **id** | Int(11) |  | 关注id |
|  | userid | Int(11) |  | 用户id |
|  | writerid | Int(11) |  | 被关注用户id |
|  | writername | Varchar(20) |  | 被关注用户名 |
| userinfo | id | Int(11) |  | 用户id |
|  | nickname | Varchar(10) |  | 用户昵称 |
|  | email | Varchar(20) |  | 邮箱 |
|  | city | Varchar(8) |  | 居住城市 |
|  | birth | date |  | 生日 |
|  | sex | Binary(1) |  | 性别 |
|  | likeNum | Int(10) |  | 获赞人数 |
|  | fansNum | Int(10) |  | 粉丝数（被关注数） |
|  | isHot | Smallint(1) |  | 是否被推荐 |
| userlog | id | Int(11) |  | 用户id |
|  | username | Varchar(10) |  | 用户登录名 |
|  | psd | Varchar(15) |  | 用户密码 |
|  | nickname | Varchar(8) |  | 用户昵称 |
|  | type | Int(1) |  | 用户类型（1为管理员/0普通用户） |

1. 接口设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 所在模块 | 接口功能 | 接口名称 | 请求字段 | 返回字段 |
| 登录页 | 登陆校验 | User/index/isLogin | username password | status(1成功{type（1是管理员，0是用户）}/0失败) |
| 注册页 | 注册 | User/index/regist | Username:用户名  Psd:密码  Nickname:用户昵称  Email  City:常住地  Birth：生日  Sex：性别（1男/0女） | Status（1成功/0失败（msg：失败信息）） |
| 文章编辑页 | 文章保存 | User/edit/saveContent | textId：文章id（新增为空）  userId：作者id  Title：文章标题  Content：文章内容  categoryId：分类id | Status（1成功/0失败（msg：失败信息）） |
|  | 获取用户的文章 分类 | User/edit/getArticleCategory | Userid:用户id | Status(1/0)  Category(id分类，title分类的名称) |
|  | 创建分类 | User/edit/addCategory | userid  Title:分类名称 | Status（1成功/0失败（msg：失败信息）） |
| 首页 | 获取推荐作者 | User/index/goodWriters | null | Status（1成功/0失败（msg：失败信息））Body({userId:用户id，userAvatar:用户头像，userName：用户名，likeNum：赞数，fansNum：粉丝数}) |
|  | 获取推荐文章 | User/index/goodArticle | null | Status（1成功/0失败（msg：失败信息））  Body（{textId，userId,title，content，categoryId，likenum：点赞人数，isHot：是否推荐}） |
|  | （管理员）设置推荐文章 | Admin/index/setHotArticle | articleId,status(1设为推荐，0取消推荐) | status(1成功/0失败)body(true/false) |
|  | （管理员）设置推荐作者 | Admin/index/setHotWriter | articleId,status(1设为推荐，0取消推荐) | status(1成功/0失败)  body(true/false) |
|  | （管理员）获取所有文章 | Admin/index/showArticleList | null | status(1成功/0失败)  Body[{articleId,title,status}] |
|  | 显示通知 | Index/showNotification | userId | status(1成功/0失败)  成功返回  notificationList[  {id,title,status(这个保留，是否已读删除待考虑),type:通知类型(1:成功 0:失败 -1：警告)}  ] |
|  | 关注作者 | Attention/followArticle | Id 当前用户  writerId 要关注的作者id  Writername 要关注的作者名字 | status(1成功/0失败) |
| 关注页 | 获得关注列表 | Attention/showWriterList | userId | status(1成功/0失败)  成功返回  writerList[  {writerId,name}] |
|  | 读取关注作者的文章列表 | Attention/showArticleList | writerId,type(  1:按照热门排序,0：按照时间排序) | status(1成功/0失败)  成功返回  articleList[  {articleId,title,shortCut:截取文章前100个字节}  ] |
| 关注页->文章详情页 | 展示文章详情 | Detail/showArticle | articleId | status(1成功/0失败)  成功返回  {writerId:作者id,writerName:作者名,writerAvatar：作者头像,content：文章内容，createTime：创建时间，praiseNum：点赞人数} |
| 详情页 | 展示文章评论 | Detail/showCommentList | articleId | status(1成功/0失败)  成功返回commentList[  {id:评论id,userId：评论人id，username：评论人名，createTime：创建时间，content：评论内容，isPraise:当前用户是否点过赞}] |
|  | 对文章进行评论 | Detail/doComment | articleId,userId,content | status(1成功/0失败) |
|  |  |  |  |  |

1. 系统实现
2. 开发步骤

我是先写的前端部分，使用vue的ui库iview，采用了当下最流行的material design来进行构建初级架构；在每个页面的跳转上使用了vue的官方轮子，vue-router，在视图层搭建一个路由；在和数据的交互方面，由于vue是mvvm架构，也就是数据驱动的，所以先构造json格式的假数据进行交互，等到后端写完接口再绑定数据，把假数据删了就好。

后端部分写的是原生的php，因为本系统有两个角色，分别是注册的用户和管理员，分成了两个文件夹，因为大多数都是注册用户在使用系统，每个分页都被分成了不同的文件，使维护的的时候可以更加明确方便。

接口方面，ajax请求chose字段暴露接口内容，后端根据chose的接口进行对数据的处理，每次响应字段包含status，即是否成功响应。

1. 主要配置文件的核心代码
2. 程序/算法测试

测试接口的时候先是进行单元测试，先调通接口，最后再往前端代码中连接。

1. 总结

经过大学这四年的熏陶，我对编程这方面的兴趣从0到1，不要小看这只是一个小小的字节变动，但其实这是个布尔值，学校的教育打开了我的兴趣开关，感谢老师的悉心栽培，我希望自己可以在这条道路上越走越远，将来可以报答母校，报效国家！