**个人博客系统**

**职业教育学院 软件工程专业**

**学号：120402013068 姓名：谢圣强 指导教师：龚平**

**【摘　要】在工作学习过程中，往往需要记录自己的所学，正所谓好记性不如烂笔头。在过去使用的传统的笔记本的方式记录自己的学习经验笔记。在如今互联网技术，多媒体技术盛行，使用电脑记录自己的笔记，并且能丰富笔记内容。**

**本博客使用es2015(javascript)作为开发语言，采用B/S架构开发模式，前后端语言统一。使用nodejs express框架搭建后端架构，使用mysql 存储站点数据。本博客使用markdown语法书写文章，使用简洁的语法代替排版，还能方便插入，多媒体。网站分为前台展示跟后端管理，前台展示面向游客，游客可以阅读查看文章，并且能使用第三方登录来进行评论，后台管理部分则只面向网站管理员。使用了独特的单页应用技术。提供了文章的编辑发布，分类的管理，文件上传管理，以及站点设置。**

**【关键词】个人博客，nodejs，markdown，express，mysql，Spa.**

**目录**

# 引言

## 研究背景

随着互联网的发展，学习是越来越重要了，人们不可能不学习，如果不学习那他将被社会，被互联网淘汰。而博客作为学习的结晶，前人的经验，在学习的路上是很重要的组成部分。博客可以有效的结合图片，文字，能够给读者留下很深刻的影响，为学习带来便利。

博客具有了个人网站的自由精神，它既可以将个人日记，工作过程，学习笔记，生活见闻，闪现的灵感，记录起来，它能发挥出个人无限创造力，还能结交相识，汇聚朋友，进行深度交流。

并且如果能自己从零开始搭建一个博客系统，用来介绍自己，展示自己，这对于以后的个人的发展是有很大的好处。

目前就互联网个人建站的发展趋势，不可否认的，发达国家在相关领域有更长的积淀和更深度的普及，细看国外个人空间的租用，域名出售，开发者论坛都有不小的名气。由于国内对互联网的特殊管制，以及在个人网站信息监管流程上比较繁琐的步骤，造成了中国的个人博客站点比例远低于国外的现状。

本系统的设计意义就在于分享我的学习心得，展示我的作品，可将自己每日生活中发生的事情记录下来。表达个人思想：它可依照个人自由意志去撰写内容，传达个人的自由思想。分享个人心情：把生活中的不满、郁闷或愤怒宣泄到博客上。

因此，设计并制作一个个人博客系统，是有着非常积极的作用的

## 开发工具介绍

本项目使用的开发工具有：

Atom：它拥有高的自定义，可以配置项丰富。通过修改，安装插件，可以将atom 打造出一款专属编辑器，并且它插件安装简单，提供了类似npm包管理的功能。

Xshell 5:用于连接远程服务器的必要工具，它提供Ssh，sftp的连接方式，它支持动态端口的转发，自定义配色方案，等vb脚本。

Cmder:Windows必备神器,与cmd说再见！对于开发码字者，MAC和liunx果断要比windows更贴心。

## 可行性研究

### 经济可行性研究

开发此网站所需的工具皆为开源，免费工具，而其中最贵的开发服务器也使用了腾讯云云+校园计划的1元服务代替，开发成本低，而使用的技术却高效,nodejs 特有的v8引擎，执行速度远超ruby，python等脚本语言，前后端语言统一，方便维护，对程序要求低，只需掌握一门语言。与传统javaweb开发相比运行效率高，低成本，高质量。

### 技术可行性研究

本博客采用es2015(JavaScript)作为开发语言，使用nodejs开发后端框架，前后端分离，前端使用MVVM模式（vue+vue-router+vuex）开发，使用webpack打包资源，图片，依赖库都是用静态cdn加速。

虽然博文展示部分没有登录，但是后台管理必须要登录才能访问，所以使用了登录token机制来实现前后端分离下的登录校验，其中涉及token的存储，本系统使用redis 存储token。

项目使用git 进行版本控制。

## 系统开发及运行环境

### 开发环境

1. 硬件设备

电脑一台

1. 软件设备

开发所需要的软件程序,如表 1.4-1 所示

| **项目名** | **软件名称** |
| --- | --- |
| 开发平台 | Window |
| 开发工具 | Atom,Cmder,Xshell,mysql,redis |
| 开发环境 | Node7.0以上 |
| 浏览器 | Chrome |
| 测试软件 | Chrome |

### 生产环境

1.硬件设备

云虚拟主机一台

1. 环境：
2. nodejs7.0

# 相关技术介绍

## Node7.0

为什么要使用node7.0 发行版呢，而不是用6.0的稳定版 因为在node7.0以上的版本加入了async/await 的支持，这个对异步来说简直是福音，从此告别一堆一堆的回调函数，从而让语法更加简洁更加易读好维护。

nodejs背后的v8 引擎执行速度非常快，性能极好，在加上nodejs对一些特殊的用例进行了优化，提供代替api，使得v8 在非浏览器下运行的非常好。而且前后端语言统一。

## Es2015

大众点名称叫es6，它是es5 的升级，它重新定义了语法规则，引用了新的关键字，新的声明方式，运算符。扩展了內建对象和函数。引入箭头函数，新的语言和库特性就像无主之宝，等待有识之士的发掘。新的语言特性涵盖范围甚广，小到受欢迎的语法糖，例如箭头函数（arrow functions）和简单的字符串插值（string interpolation），大到烧脑的新概念，例如代理（proxies）和生成器（generators），总之es6将彻底改变编写js的方式。

## Vue+Vue-router

Vue:构建用户界面的渐进式框架，vue核心是只关心视图层，它易上手，便于与第三方库整合，也能为复杂的单页应用提供驱动。在渲染界面的时候，dom的操作是很高的，虽然vue不能让这些操作变的更快，但是它们吧dom 的更新降低到最小。渲染方面，vue会尽量避免子组件的重新渲染。

Vue-router:控制单页应用的路由。

Vuex:在vue中每个组件的数据都是独立的，想要在个个组件中贡献数据，那就要使用vuex 来做状态管理器，vuex 的核心就是store 一个状态仓库，在其他的组件通过getter 来获取数据，通过action来修改。使得状态全局贡献。

## MVVM

数据双向绑定，使用vuejs来实现，vuejs它使用起来比reactjs简单，代码清晰，借鉴了angularjs的特点使得数据与视图绑定起来，数据源进行变化会立即反应到view 上。

它是对mvc模式的完全变革，它数据模型与数据双向绑定的思想作为核心，

## Webpack

[Webpack](https://github.com/webpack/webpack" \t "http://webpackdoc.com/_blank)是当下最热门的前端资源模块化管理和打包工具。它可以将许多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔，等到实际需要的时候再异步加载。通过 loader 的转换，任何形式的资源都可以视作模块。比如一些浏览器是不支持es2015的语法的，使用webpack将其打包进行转换后，可以生成es5 语法使其浏览器兼容。也将代码进行压缩，加速前端的访问速度。

## Git

Git是免费的开源的分布式版本控制系统。使用git可以有效的处理项目，管理代码更新，控制代码版本。

## Redis

Redis 是开源的，它使用c语言编写，是一个基于内存可持久的kv数据库，它提供了多种语言的api 这其中就包括了nodejs

## Mysql

它是一个由c++与c编写的[关系型数据库管理系统](http://baike.baidu.com/view/1450387.htm" \t "http://baike.baidu.com/item/mySQL/_blank),是MySQL AB 公司开发的，现在已被甲骨文公司收购。它支持系统多，[Linux](https://zh.wikipedia.org/wiki/Linux" \o "Linux)、[Mac OS](https://zh.wikipedia.org/wiki/Mac_OS" \o "Mac OS)、[Windows](https://zh.wikipedia.org/wiki/Windows" \o "Windows)等多种[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1" \o "操作系统)。为多种編程语言提供了[API](https://zh.wikipedia.org/wiki/API" \o "API),包括java（mysql-connector-java.Jar）。MYSQL性能高、成本低、跨平台，已经成为最流行的开源数据库。

# 系统分析

## 系统整体详细设计

页面展示：使用了近期上线的vue2.数据驱动页面，页面交互设计更加方便。分离前后端关注点，前端负责界面显示，后端负责数据存储和计算，各司其职，不会把前后端的逻辑混杂在一起；减轻服务器压力，服务器只用出数据就可以，不用管展示逻辑和页面合成，大大提高速度

后台服务：使用v8 引擎的nodejs7，运行速度快，前后端一个语言维护方便。Nodejs可以很快的在服务器上做原型，而且代码量少。

数据库：使用了开源的mysql，它安装方便，内置api丰富，体积小，跨平台等特点

## 系统模块功能分析

前台博文展示：这是博客展示部分，要求界面美观。主要功能有：最新博客列表，最近评论。分类浏览，归档，博主作品展示，博主个人介绍信息，友情链接。

1. 博文查看：将markdown语法渲染成html标记语言进行展示，展示内容有作者介绍，分类，标签；分享功能，评论文章功能，统计阅读量功能。以及目录查看功能
2. 分类查看：查看所有分类，能够显示分类的说明，分类的特有的图片，icon。点击进入分类，查看分类下所有的文章。
3. 归档：将文章按照月份归档成目录，已时间轴的形式展现出来。

后台博客管理：主要是对网站进行管理。主要功能有文章编辑，发布，修改。分类增删改查、上传管理、评论管理、用户管理、站点信息设置

1. 文章管理：以列表的形式查看所有的文章，能够对文章进行置顶，删除，发布，撤回修改文章缩略图功能。
2. 文章编辑：以分类为目录，选择，新建文章进行编辑发布，包括文章的标题，标签，内容，展示缩略图，加密等详细修改。
3. 上传管理：对站点内上传的文件进行统一管理，执行删除，批量上传操作。
4. 评论管理：对前台文章的评论进行管理，包括封禁ip，过滤反动字符，小广告等
5. 用户管理：本博客系统只有一个超级管理员，仅对超级管理员信息进行修改，包括说明，头像，密码等。
6. 站点信息设置：设置首页展示的样式，包括首页图片，首页个签，首页附加代码，站点统计代码。

本系统结构设计图如下：

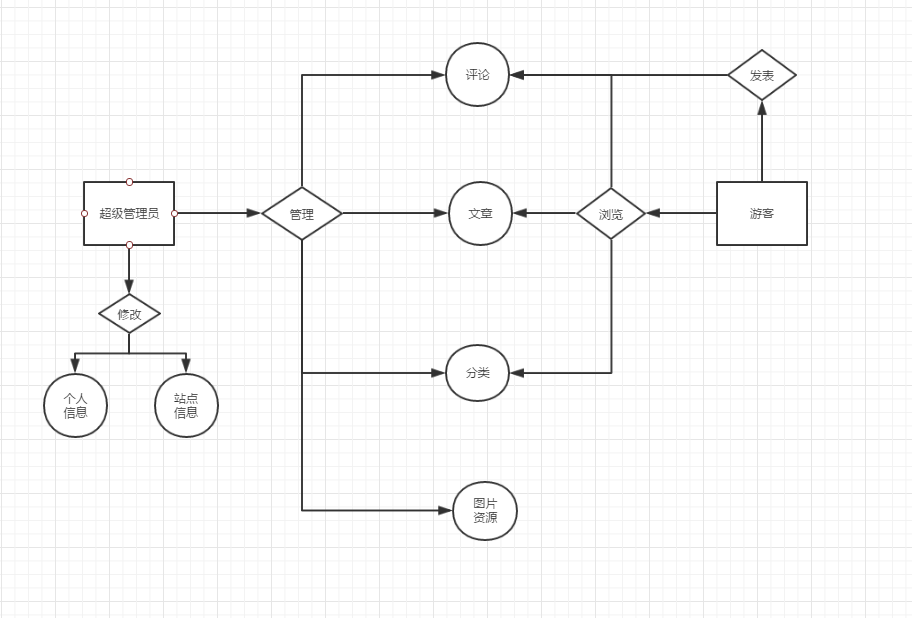


图 3.2-1 系统结构设计图

## 数据库详细设计

1. 用户表

用于记录本站点的超级用户。

表结构如下

| 表名 | J\_users（用户表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | bigint(20) unsigned | auto\_increment |
| User\_login | varchar(60) | 登录名称 |
| User\_pass | varchar(64) | 密码 |
| User\_nickname | varchar(50) | 昵称 |
| User\_email | varchar(100) | 邮箱 |
| User\_url | varchar(100) | 网址 |
| User\_registered | timestamp | 注册时间 |
| user\_status | int(11) | 状态 |
| display\_name | varchar(250) | 显示名称 |

1. 访客表

用于记录访问本站点的访客

| 表名 | j\_visitors（访客表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | bigint(20) unsigned | auto\_increment |
| Ip | varchar(30) | 来访Ip |
| create\_at | timestamp | 来访时间 |
| userName | varchar(60) | 如果是登录的记录账号 |
| address | varchar(255) | 来访人员物理地址 |
| isp | varchar(30) | Isp |
| originalUrl | varchar(300) | 访问的url |

1. 分类表

用户记录文章的分类，

| 表名 | j\_terms（分类表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| term\_id | bigint(20) unsigned | auto\_increment |
| name | varchar(200) | 分类名称 |
| slug | varchar(6) | 链接名称，防爬取 |
| taxonomy | varchar(32) | 分类标识 |
| parent | bigint(20) unsigned | 父分类 |
| count | bigint(20) | 分类文章数 |
| description | longtext | 分类说明 |
| delete\_at | timestamp | 删除时间 |
| create\_at | timestamp | 创建时间 |
| icon | varchar(200) | 分类图片 |

1. 评论表

记录下游客的评论

| 表名 | j\_comments（评论表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| comment\_id | bigint(20) unsigned | auto\_increment |
| comment\_post\_id | bigint(20) unsigned | 评论的文章 |
| comment\_author | tinytext | 评论者作者 |
| comment\_author\_email | varchar(100) | 评论者邮箱 |
| comment\_author\_url | varchar(200) | 评论者网站 |
| comment\_author\_ip | varchar(100) | 评论者IP |
| comment\_date | timestamp | 评论时间 |
| comment\_content | text | 评论内容 |
| comment\_approved | varchar(20) | 评论是否批准 |
| comment\_agent | varchar(255) | 评论者设备 |
| comment\_type | varchar(20) | 评论类型 |
| comment\_parent | bigint(20) unsigned | 父评论 |
| user\_id | bigint(20) unsigned | 是否登录，登录账号 |

1. 评论附加信息表

评论附加信息

| 表名 | j\_commentmeta（分类表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| meta\_id | bigint(20) unsigned | auto\_increment |
| comment\_id | bigint(20) unsigned | 评论id |
| meta\_key | varchar(255) | key |
| meta\_value | longtext | value |

1. 文章表

记录所有文章的信息

| 表名 | j\_posts（分类表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| id | bigint(20) unsigned | auto\_increment |
| post\_author | bigint(20) unsigned | 文章创建者 |
| post\_date | timestamp | 发布时间 |
| post\_content | longtext | 文章内容 |
| post\_title | text | 文章标题 |
| post\_excerpt | text | 文章摘录 |
| post\_status | varchar(20) | 文章状态 |
| comment\_status | varchar(20) | 评论状态 |
| ping\_status | varchar(20) | Ping状态 |
| post\_password | varchar(20) | 文章密码 |
| post\_name | varchar(200) | 文章缩略名 |
| term\_id | bigint(20) unsigned | 分类 |
| post\_modified | timestamp | 文章最后修改时间 |
| post\_content\_filtered | text | 文章过滤内容 |
| post\_parent | bigint(20) unsigned | 父文章 |
| guid | varchar(255) | 文章链接 |
| menu\_order | int(11) | 排序 |
| post\_type | varchar(20) | 文章类型 |
| comment\_count | bigint(20) | 评论数 |
| seq\_in\_nb | int(11) |  |
| delete\_at | timestamp | 删除时间 |
| create\_at | timestamp | 创建时间 |
| author | varchar(255) | 文章作者 |
| eye\_count | bigint(20) | 查看数 |

1. 文章附加信息表

文章的附加信息

| 表名 | j\_postsmeta（文章附加信息表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| meta\_id | bigint(20) unsigned | auto\_increment |
| Post\_id | bigint(20) unsigned | 评论id |
| Post\_key | varchar(255) | key |
| post\_value | longtext | value |

1. 站点表

记录站点信息

主要key 有'title','keyword', 'description', 'sign', 'background','avatar' 等

| 表名 | j\_site（站点表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Key | varchar(30) |  |
| Value | Longtext |  |

1. 阅读记录表

记录文章的阅读记录

| 表名 | j\_reads（分类表） | |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Ip | varchar(15) | Ip |
| Guid | varchar(255) | 文章链接 |
| create\_at | timestamp | 阅读时间 |
| useragent | varchar(255) | 设备 |

# 项目框架搭建

## 项目目录结构

项目的目录结构决定了项目的根基，决定了项目的扩展性。好项目的开始都是先从目录开始分，由于第一次自己摸索node项目，没有前人项目经验，在这次项目目录划分上改动了很多次，最终将目录划分如下:

├─ doc //文档说明，一些随便感言

├─ client 前端代码

│ ├─ public # 公共的资源文件目录

│ │ ├─css

│ │ ├─img

│ │ ├─js

│ ├─ src

│ │ ├─ bg # 后台管理模块

│ │ │ ├─ ...

│ │ ├─ blog # 前台展示模块

│ │ │ ├─ c.xxx.js # xxx 模块入口文件

│ │ │ ├─ ...

│ │ ├─ store # vuex仓库

│ │ ├─ components # 组件仓库，存储一些公共的组件

│ │ ├─ private.js # 私有的配置文件，不上传到代码服务器

├─ server # 后端代码

│ ├─ dao # 数据库链接，封装数据操作

│ ├─ router # 定义路由

│ ├─ views # 视图文件

│ ├─ test # 测试脚本

│ ├─ config.js # 配置文件

│ ├─ util.js # 工具类

├─ public # 线上用户访问的.html目录 webpack 打包生产的目录

├─ webpack #

│ ├─ webpack.base.js # webpack父配置，其他环境配置从其衍生

│ ├─ webpack.dev.js # 开发环境使用配置项

│ ├─ webpack.production.js # 生产环境配置项

│ ├─ webpack.dev.js # 数据库链接，封装数据操作

├─ package.json

├─ app.js # 项目入口文件

├─ private.js # 一些小秘密系统分析本系统采用前后端

## 项目配置

本次项目分为开发环境与部署环境，开发周期较长，经常会更新新功能上线，故将一些环境配置不一样的属性通过private.js 来加载，例如跨域请求的header头设置，在生产环境当然只能允许同源的请求，而本地开发的时候使用http://localhost:3878/ localhos域3878端口，会发生错误。

## 项目启动

利用package.json的script 项可以定义一些启动命令



图 4.3-1 启动命令设置

启动命令如下：

1. npm run dev # 开发环境

一键编译前后端代码，不产生文件。并且开启热部署，修改代码实时反馈到页面。

1. npm start # 生产环境

一键编辑前后端代码，生成静态文件，开启node 服务。

# 后台服务结构的设计

系统想要做的大，做的好，可扩展，维护方便，结构更是重中之重。虽然使用的是新学习的nodejs 但是基本MVC 结构还是套的上去的。从目录结构上看我已经做好了这个准备。首先是Model层

## Model层

在本系统中称为dao，对应的文件目录是/server/dao/\*,为了方便业务层获取数据，以及每张表的实现，我封装一个base.dao.js ，其他的统一约定命名为xxx.dao.js (xxx为表名)。在这个类统一继承base.dao.js 。在base.dao.js中我定义了2个方法，一个execCallBack，和execTrans。execCallBack执行一个sql语句，并将结果通过回调函数的方式传回去，它提供1必选参数sql语句，3个可选参数，data,callback,resultFormat。execTrans方法则是执行一个事务，它的实现是将node-mysql的事物进行了封装。详细代码见图5.1-1



图5.1-1

有了这个2个方法，在其他表存取上只需根据需要提交参数即可，大大减少代码冗余。示例：如term.dao 在删除分类时需将分类下的所有文章移动到回收站可以构建如下代码：



图5.1-2

## Controller 层

在项目中有2个大模块，博文的展示跟后台的管理，所以路由上也分成2个模块。

一个bg，一个blog，在bg里定义的都是api操作，没有路由，因为后台管理部分使用的前后端分离模式，而博文展示使用的服务端渲染的模式，有数据的获取视图的控制。

## View 层

View 只有前台博文展示的页面，后台管理的页面之间放在了/public目录下。因为我的博客是放在公网上，而想要让搜索引擎收录的我的文章，使用前后端分离SEO比较难做，所以我采用曲线救国的办法，将博文展示部分改成服务端渲染，而后台管理部分只是管理员使用所以使用了前后端分离的技术。

# 前台页面展示设计与实现

## 网站首页设计与实现

### 设计

网站首页包括顶部导航区域，首页大图，首页个签，展示个性。往下翻阅依次 最新文章列表，一次加载10篇文章，拉到底部会在加载10个且不会重复加载，文章列表包含文章的标题，文章的抽象内容。右侧也不浪费，放置了标签列表，72小时热文榜，最新评论，友情链接，文章归档，最底部设计了一个专属版权信息栏。界面清晰，触控简明，内容丰富。

效果图6.1-1



图6.1-1 导航栏，首页个性签名

左右2栏的主内容区，还在右下角浮动显示了一个回到顶部的快速按钮、带有加密的博客鼠标经过时显示输入密码、右侧标签跑马灯式滚动。



图6.1-1 左右2栏的主内容区



图6.1-2 输入密码

### 实现

部分代码

1. 获取首页列表

安全性方面，由于首页使用了一个加载更多的异步请求，防止人爬取网站内容，我将ajax返回结果为html代码，直接插入到页面。当前直接插入html代码又会有xss 攻击的风险，在服务端也进行了过滤代码如下图6.1-3



图6.1-3

1. 鼠标经过显示输入密码控件

这个功能的实现是通过css 的伪类控制元素的显示实现的。具体思路是在有需要输入密码的地方加入一段密码框的html代码。并赋予样式，当鼠标移动到这个文章元素上时触发元素的hover伪类，然后将这个密码组件显示出来，代码如图6.1-4



图6.1-4

1. 导航根据用户操作显示

导航栏使用js 控制显示，正常预览网页往下方滚动时隐藏滚动条，向上滚动时，用户可能想看别的内容，则显示出导航栏，人性化的设计。代码如图 6.1-5



图 6.1-5

## 登录界面设计

### 设计

本系统没有登录界面。没有登录界面不代表没有用户登录，为了网站后台的安全，我将登录页面隐藏在首页，通过特殊的操作触发，然后输入账号密码进行登录，登录后手动访问后台链接进行管理。

这么做的原因是为了安全，如果有人想通过后台服务上传木马黑入的服务器，那他将连登录后台都找不到！这样做是很有意义。

前台博文展示部分用户的意义不大，后台管理才是用户权限的体现，而后台管理是前后端分离的，那么使用token来记住用户的登录状态是必须的。

登录效果图：

在首页标签分类栏，将鼠标放在java 标签上后输入账号，这个输入不显示的，输入错误也不提示，只有输入正确后才会提示。如图6.2-1



图6.2-1

### 实现

部分代码：

1. 登录请求的发起

使用了keyboardjs 来监听键盘响应事件，当用户按了回车后发送请求，登录成功提示。



图 6.2.2



图 6.2.3

1. Token的实现

发送请求后的创建token，token创建后保存在redis 本地服务器上设置上时间，然后在将生成后的token设置在cookie里，用户发起请求时会带上这个cookie，接着我在实现一个验证器验证请求的cookie即可。

创建token：这里使用了jsonwebtoken 插件来加密token，它的需要一个secret 字符来加密token。图 6.2.4



图 6.2.4 token的加密，中间件

在发起登录请求后将用户信息生成token，并存入redis服务器上。如图6.2.5

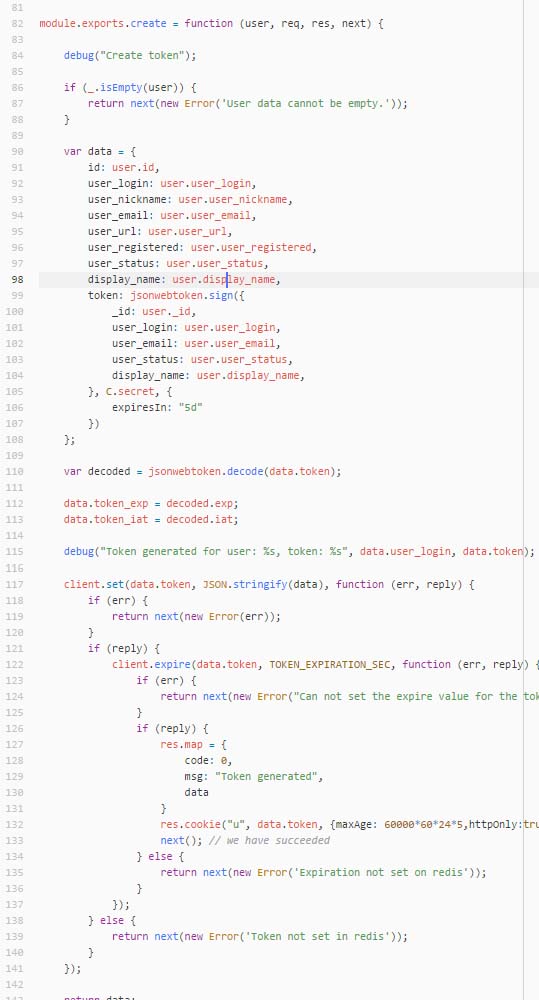


图6.2.5 生成token 存入redis

1. Token 验证中间件的实现

图6.2.4 启用中间件，图6.2-5，图6.2-6中间件的实现



图6.2-5



图6.2-6

## 文章展示设计与实现

### 设计

从安全性考虑，博文的链接如果是数字型的话回来又被爬取的风险，从此考虑我将博文的链接使用objectId 进行编码，生成一个唯一的id 当做文章的固定链接。

页面设计上，保留了与首页一致的导航栏，底部版权栏。本系统的文章都是使用markdown语法进行书写，所以在生成页面时需将markdown语法转换成html展示出来。本本系统采用了功能完善的editor.md 插件来进行转换，editor.md不仅支持markdown语法还能转义公式，流程图，等多种语法，大大丰富了我的博客内容体系。

展示部分比较容易，关键是做一些内容上的优化，比如SEO关键词，说明，google等主流搜索引擎的description 属性。

效果如下：



图6.3-1 文章展示



图6.3-2 文章展示-自动目录

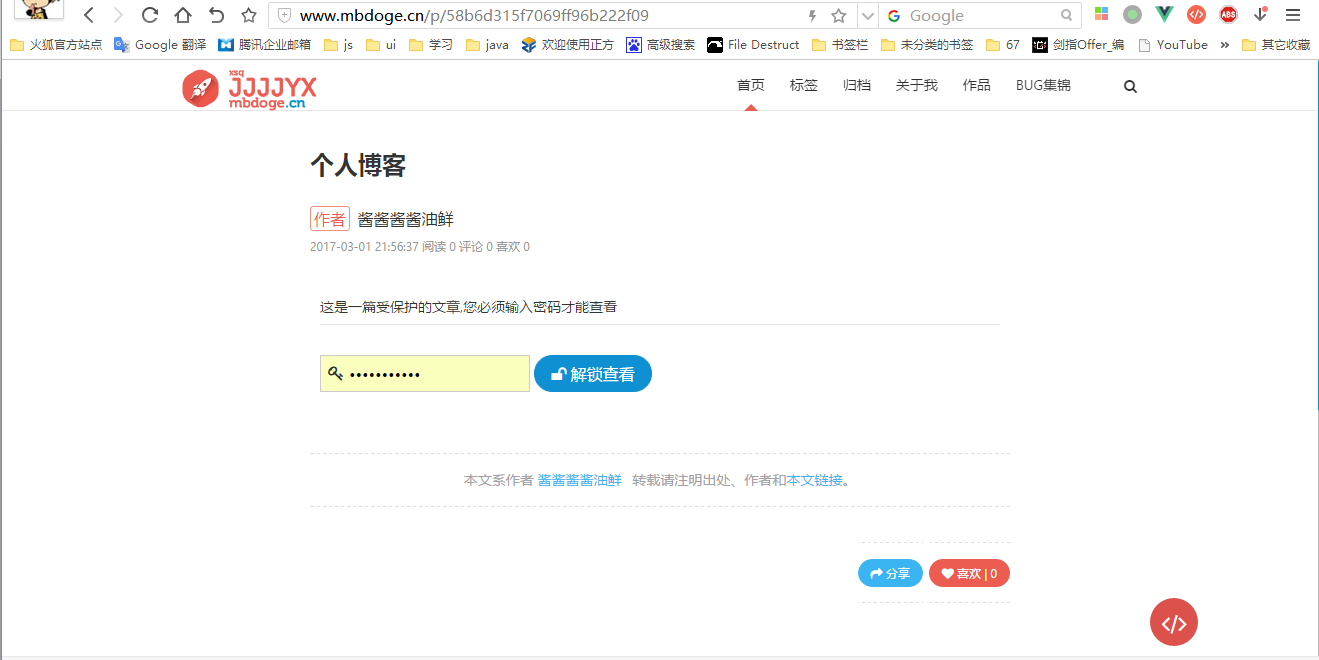


图6.3-3 文章展示-访问密码



图6.3-4评论

### 实现

1. 获取文章内容

根据guid 获取文章内容



图 6.3-5 获取文章 路由

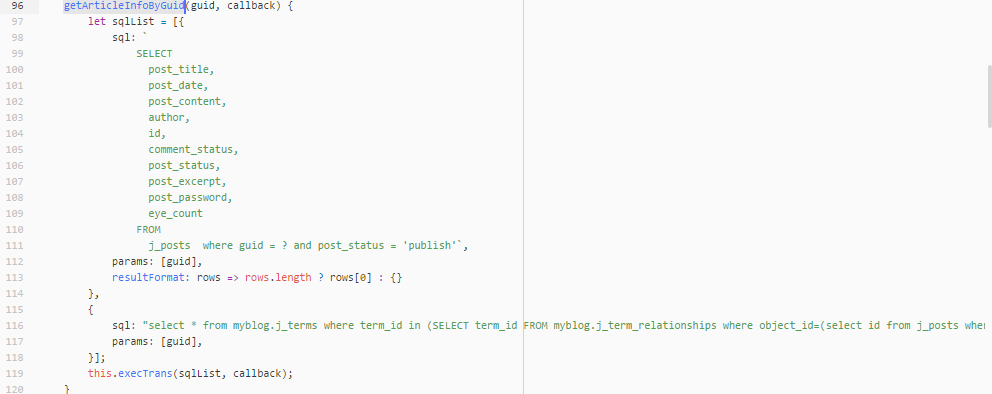


图 6.3-6 获取文章 数据

1. 渲染文章



图6.3-7 渲染文章

1. SEO

对SEO认知不是很足展示做到这样



图6.3-8 渲染文章

## 访问优化

# 后台管理的设计与实现

## MVVM 模式的应用

与传统前端项目相比，本次项目最大的区别就是应用了mvvm模式，页面完全由数据驱动，数据的改变会映射到视图上。举例，模拟一个购买商品业务场景：统计商品总价格，输入项：商品规格，物品数量，输出项：总价格。在传统的html 中会这样写到：



图7.1-1传统写法

那利用vue框架我们可以这样写道，构建html结构，绑定数字，为总价绑定计算属性，代码如下



图7.1-2 MVVM模式下的写法

对比结果显而易见，传统写法代码复杂，涉及选择器，事件绑定，读取表单值，设置值，修改dom元素。如果业务需求增加，改动起来将变的非常复杂，比如有8个可输入项，若干个输出结果项，相互影响结果，那代码维护起来非常的吃力。而使用数据驱动视图，只需绑定将数据安装语法绑定即可，任你需求改动我只需按部就班绑定，无需再维护dom的渲染，只需关注具体需求。

在后台管理页面设计中就大量应用了这种模式，比如文章列表的渲染，使用vuex 来存放postsList文章，减少异步请求次数，减少服务器负担。在文章列表管理模块取得数据，绑定视图，如果在视图中修改数据则影响全局变化。

图7.1-3 绑定 文章列表

还可以更复杂的处理，比如加入搜索功能，分页功能。只需要在计算属性实现一个过滤方法，将满足输入项的结果数据返回，利用js数组的原生方法就能简单的实现。

而分页就在过滤后的基础上，在将数组根据分页条件切割，然后在绑定一些按钮来改变页数，翻页即可

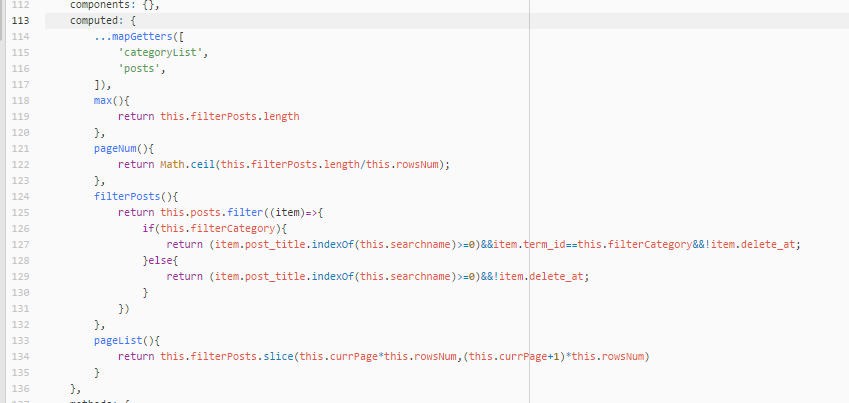


图7.1-3 获取 文章列表，分页显示

如果这些东西使用传统的方式，代码量是很多的，或者会加入很多插件，增加维护成本。

## 文章编辑功能的设计与实现

### 设计

文章的编辑我使用比较经典的3栏布局，将文章，分类，编辑抽象成为树的形式，在最左侧是放着分类列表，点击分类列表查看分类下的文章，点击文章进行编辑文章，文章的编辑使用了editor.md markdown 语法编辑器。在编辑文章过程中，需要频繁的插入图片，但是选择图片上传很麻烦，然后自己扩展了编辑器的实现，增加了粘贴上传图片的功能，这算是本博客的亮点之一。每一栏的上方还加入了快速新建按钮，方便操作。

在文章保存方面，文章编辑后自动保存，或者可以通过快捷键ctrl+s 快速保存文章，防止文章丢失，如果没有保存文章关闭网页将会提示。如图7.2-1 所示

文章的编辑采用了特有vue-router的嵌套路由，将打开的路径直接体现在url上，在不慎刷新页面后还能回到上次编辑位置，如图7.2-1 所示url



图 7.2-1 编辑文章界面

编辑文章栏是个从上至下的布局，按操作习惯以此是标题栏，编辑器工具栏内容编辑区。以及可以浮动显示的预览栏，工具栏。如果想要设置文章的标签，可以在工具栏点击标签按钮，会下拉出标签的选项如图7.2-2 所示

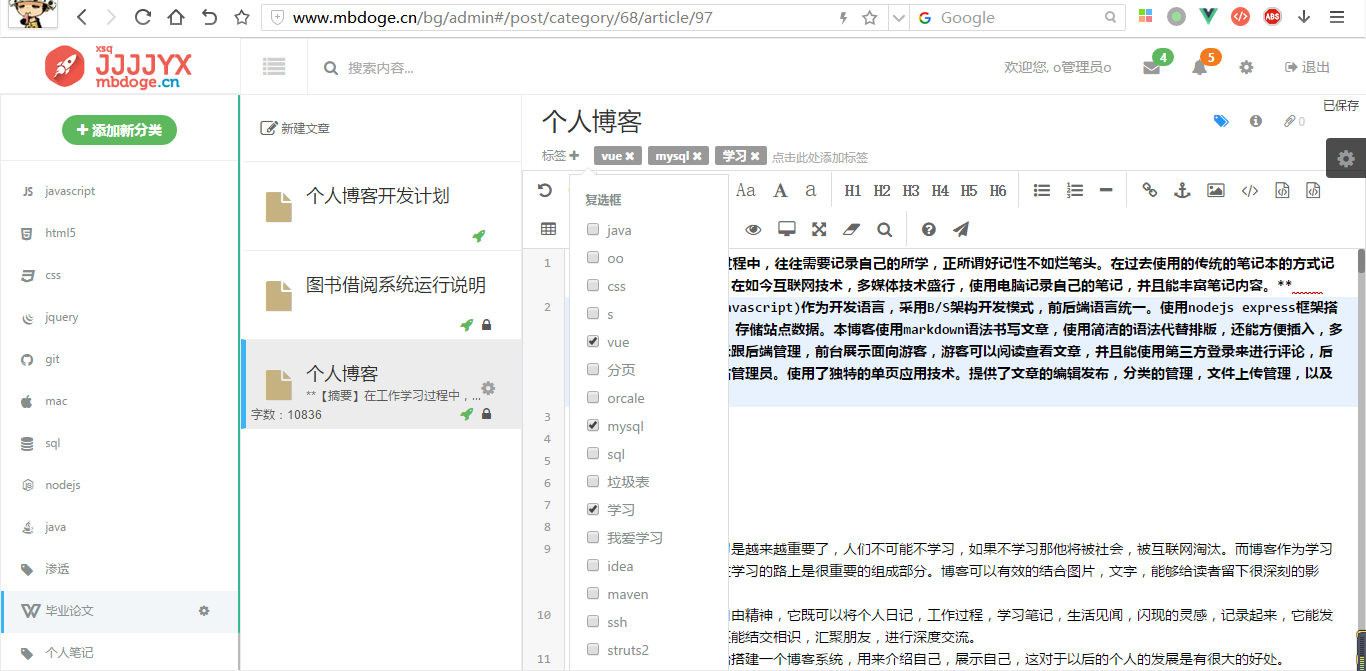


图 7.2-2 编辑文章界面-设置标签-隐藏预览

在工具栏中还能设置文章属性，加密文章，公开，发布文章等操作。



图 7.2-2 编辑文章界面-设置标签-隐藏预览

### 实现

1. 3栏布局+ 嵌套路由

这个3栏布局还得加上嵌套路由的结构总体实现如下，定义路由：图7.2.3，因为组件个数较多嵌套3级略显麻烦，附加使用命名视图方法解决。



图 7.2-2 文章编辑-路由定义



图 7.2-3 文章编辑-嵌套路由-命名视图

图 7.2-4 文章编辑-路由定义

1. 图片粘贴上传
2. 添加文章标签
3. 自动保存

## 分类的管理的设计与实现

## 站点设置的设计与实现

# 总结

# 致谢