个人博客管理系统的设计与实现

摘 要

本文针对个人博客管理系统进行了系统架构分析和实验研究，针对特点建立了个人博客管理的SPA应用。本系统前后端分离， 使系统耦合度更低，每个模块之间的关联更小，开发效率更高，维护的成本也大大降低。

系统功能主要包括博客内容的查看、编辑、删除、投稿、保存上传；博客用户的个人资料编辑修改和保存；管理员对投稿日志的筛选，以及将日志推送到首页。由于本系统的定位是一个博客社区，不仅仅是只有博客的基础管理，每个用户之间还有更多的互动：用户之间可以进行关注，关注即订阅了该用户的文章，当用户切换到关注页下，可以点击关注用户的头像按照时间顺序或热度排名查看被关注用户的文章列表；用户可以对某篇文章进行评论、点赞等操作；当用户被关注或收到投稿相关的反馈时，该条内容会添加到通知模块，用户点击通知按钮时会弹出通知列表，点击某条通知即已读通知并从数据库里删除。本系统的编辑器使用markdown，不同于其他使用富文本编辑器的博客系统，该编辑器对比其他富文本编辑器的更出众的优点将会在下文提出。

对于本博客管理系统，进行了系统化的测试。根据需求文档对比当前系统是否符合规定的要求；确定该系统是否处于高风险的范围，是否安全；用户是否需要花费成本；对每个模块进行单元测试；评估维护的成本等。

**关键词：** SPA应用 markdown MVC

Design and Implementation of Personal Blog Management System

**ABSTRACT**

In this paper, the personal blog management system for the system architecture analysis and experimental research, for the characteristics of the establishment of a personal blog management SPA applications.The system back and forth separation, so that the system coupling is lower, the correlation between each module smaller, more efficient development, maintenance costs are greatly reduced.

System functions include the contents of the blog to view, edit, delete, submit, save upload; blog user's personal data editor to modify and save; administrator on the submission of log selection, and push the log to the home page. As the positioning of the system is a blog community, not only the basic management of the blog, there is more interaction between each user: the user can be concerned about, attention to subscribe to the user's article, when the user switch To the attention page, you can click on the user's attention to the user in accordance with the chronological or heat rankings to view the list of users concerned about the user; users can comment on an article, point of praise and so on; when the user is concerned or received contributions related to feedback , The contents of the article will be added to the notification module, the user clicks the notification button will pop up a notification list, click on a notice that has read the notice and deleted from the database. The system's editor uses markdown, which is different from other blog systems that use rich text editors, and the editor's superiority over other rich text editors will be presented below.

For this blog management system, a systematic test. According to the requirements of the document compared to the current system is consistent with the requirements; to determine whether the system is in a high risk range, whether the security; users need to spend costs; unit testing for each module;

**Key Words:** SPA markdown MVC引 言

Web 2.0出现以来，社交媒体上的内容和使用用户介入度急剧增长。博客已经成为与商业和个人生活的各个方面相关的记 录平台。 然而，我们没有适当的工具来正确地汇总和保存博客内容，以及有效地管理博客归档。 鉴于博客日益重要，建立一个稳定运行的系统以促进博客保护至关重要，保护我们遗产的重要部分，这对当代和后代来说将是有价值的。 在本文中，我将详细介绍在这个课题研究中所构建的一个具有体验流畅且稳定运行的博客展示和托管的平台，任何个人或组织都可以使用这个平台来保留他的博客。

利用这个的专门平台博客的特点，能够改进归档。 Web保存被定义为捕获，管理和保存网站和网页资源。 Web保存必须是一个从出生（即开始）到死亡（完成）的活动，它应该包括网络资源的一个完整生命周期。 最着名的网络归档计划是Internet Archive1自1996年以来一直在运作。另外还有各种各样的项目国家和国际组织正在开展网络维护工作相关活动。所有活跃的国家网络存档工作，作为以及一些学术网络档案是国际互联网的成员保存的。 但是，这是一项复杂的任务需要耗费很多资源。 因此，网络归档只有部分现有的网络被归档。与传统的媒体相反，如印刷业，使用网络可以极大地加快传播速度。 因此，博客的种类不仅仅是由规定主题的文档，还有其他参数，如每页的归档频率以及与页面请求相关的参数（例如浏览器，用户帐号，语言等）。

博客归档是Web归档的子类别。博客的意义在商业和私人生活的各个方面都不能低估。 博客是拉丁美洲的教学物理学用来促进时尚讨论法国的年轻人的产物。所有博客的集体成果，它们的内容，相互联系和影响构成了他们的作品博客圈，具有重要意义的独特的社会技术文物。然而，当前的网络存档工具导致博客保护的几个问题[1]。首先，采集和策划的工具使用基于时间表的方法确定内容应被捕获以便归档的时间点，导致信息丢失，如果它比其更新更频繁地更新被爬。另一方面，不必要的收获和存档的重复如果博客比爬网时间表更不频繁地更新，则会发生如果整个博客再次被收获，而不是选择性地收获新的页面。因此，考虑更新事件（例如新帖子，新评论等）作为爬行活动的触发器的方法将更为合适。

目 录

[第一章 任务概述 1](#_Toc7497)

[1.1目标 1](#_Toc28113)

[1.2本系统的特点和目标用户 1](#_Toc7343)

[第二章 需求分析 3](#_Toc14435)

[2.1基本需求 3](#_Toc3923)

[2.1.1功能需求 3](#_Toc25487)

[2.1.2性能需求 3](#_Toc5670)

[2.2运行环境需求 4](#_Toc301)

[2.2.1软件环境 4](#_Toc17830)

[2.2.2硬件环境 4](#_Toc32188)

[2.3 E-r图 4](#_Toc17000)

[2.3.1用户实体 4](#_Toc30026)

[2.3.2 通知实体 5](#_Toc17122)

[2.3.3文章实体 5](#_Toc11404)

[2.3.4评论实体 5](#_Toc10410)

[第三章 系统设计 7](#_Toc14252)

[3.1流程图及流程说明 7](#_Toc13443)

[.3.1.1.登录注册 7](#_Toc25755)

[3.1.2查看用户信息 7](#_Toc13289)

[3.1.3日志查看/编辑 8](#_Toc27885)

[3.1.4通知 9](#_Toc15366)

[3.1.5首页投稿 9](#_Toc21377)

[第四章 算法设计 11](#_Toc25900)

[4.1 MVC设计模式 11](#_Toc13447)

[4.2 MVVM设计模式 12](#_Toc21326)

[4.2.1 MVVM介绍 12](#_Toc17062)

[4.2.2 MVVM模式的组件 12](#_Toc31498)

[4.2.3 解释 13](#_Toc16967)

[4.3系统交互模式设计 13](#_Toc20549)

[4.4技术选型 14](#_Toc7256)

[4.4.1 Vue 14](#_Toc12907)

[4.4.2 Webpack 15](#_Toc11263)

[4.4.3 Ajax 16](#_Toc19398)

[4.4.4 Apache+php+mysql 16](#_Toc18202)

[4.5 开发工具 17](#_Toc1171)

[4.5.1 WebStorm 17](#_Toc16252)

[4.5.2 PhpStorm 18](#_Toc28603)

[4.5.3 Sublime text3 18](#_Toc19263)

[第五章 数据库设计 19](#_Toc1628)

[第六章 接口设计 21](#_Toc29879)

[第七章 系统实现 26](#_Toc28802)

[7.1 开发步骤 26](#_Toc30392)

[7.2 主要配置文件的核心代码 26](#_Toc19650)

[7.2.1 package.json文件 26](#_Toc20559)

[7.2.2 vue-router配置文件 26](#_Toc13718)

[7.2.3 webpack配置文件 27](#_Toc29483)

[具体配置见附录3 27](#_Toc31397)

[第八章 程序的实现 28](#_Toc18999)

[第九章 谢 辞 29](#_Toc30059)

[参考文献 1](#_Toc581)

[附录1 1](#_Toc23499)

[附录2 4](#_Toc3244)

[附录3 6](#_Toc18585)

# 

# 第一章 任务概述

## 1.1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，不仅仅是发表文章，用户之间还可以进行更加现代化的互动。

本系统是用vue+php实现的一个完全基于浏览器的博客系统，注册用户拥有以下权限：

1. 登陆系统
2. 注册用户可以给某篇博客留言
3. 注册用户可以对某篇文章进行点赞操作
4. 注册用户可以拥有自己的头像昵称等信息
5. 注册用户可以编写自己的文章并保存到服务器上
6. 注册用户可以把自己的文章投稿到首页，让更多的用户查看或评论
7. 注册用户可以管理自己的文章列表，上传文章、删除指定文章、和编辑指定文章等操作
8. 注册用户可以管理自己的文章分类列表，编辑分类条目、添加一个分类、或删除某个分类等操作

管理员拥有以下权限：

1. 管理员可以对用户投稿的文章进行筛选，并放到首页上

非注册用户可以进行的操作：

1. 查看主页上的投稿文章
2. 注册账户

## 1.2本系统的特点和目标用户

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。

该系统特点：

1. 优雅。本系统使用基于markdown的编辑器，对于书写者来说，排版更加优雅，写作时更加专注于文字本身，而无需再考虑怎样排版更加好看；对于读者来说，使用统一的格式，阅读体验更佳。
2. 免费。用户不需要购买域名，不需要购买服务器，不需要花费任何的费用，只要注册一个用户名，就能拥有与自己用户名一样的基于本站的域名了。
3. 传播更广泛。用户可以通过投稿的方式，把自己的觉得满意的作品推送到首页，让更多的用户看到，传播更加广泛。而不仅仅像普通的个人主页那样：必须得进入目标用户的主页才能浏览信息，使信息流通变得更加闭塞。
4. 使用方便。使用该系统无需掌握任何的计算机知识，只要初步了解了markdown的基础语法，就可以写博客了。像使用电子邮箱一样简单可以对自己的博客进行管理：如：编辑、上传、删除等操作。

# 第二章 需求分析

## 2.1基本需求

### 2.1.1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

本系统的用户有三种身份，分别为：游客（即未注册/未登录用户），注册用户和管理员。要求每种角色使用系统，操作时都方便快捷。游客在没有登录的情况下可以查看博客的详情；已经注册并且登陆系统的用户享有更多的特权，可以对自己的博客进行相关的操作等；管理员可以对用户投稿文章进行筛选。

### 2.1.2性能需求

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。

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。

## 2.2运行环境需求

### 2.2.1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

### 2.2.2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

## 2.3 E-r图

### 2.3.1用户实体

每个用户实体包括用户id（id），用户名（username），登录密码（password），昵称（nickName）、性别（sex）、生日（birth）、常住地（place）、常用邮箱（email）、主页推荐标识符（isHot）、获赞数（likeNum）、获得评论数（commitNum）等属性。

用户

id

password

sex

birth

username

isHot

articleNum

email

creatTime

commitNum

### 2.3.2 通知实体

每个通知实体包括通知id、通知标题（title）、通知类型（即type，包含成功、失败和警告）、创建日期（createTime）等属性。

通知

type

title

status

createTime

id

### 2.3.3文章实体

每个文章实体包括文章id、文章标题（title）、内容（content）、创建时间（createTime）、作者名（writerId）、获赞数（likeNum）、摘要（shortCut）等属性。

文章

writerId

likeNum

createTime

content

id

praiseList

shortCut

### 2.3.4评论实体

评论实体包括评论的id、对应的文章id（articleId）、评论人的用户id（userId）、评论内容（content）、创建时间（createTime）等属性。

评论

createTime

content

userId

articleId

id

# 第三章 系统设计

## 3.1流程图及流程说明

### .3.1.1.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的文章或推荐关注的作者。

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

### 3.1.2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

### 3.1.3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；

进入这个详情页后，判断路由中的指定参数是否为当前的日志作者：

是则可以进行修改，删除，评论等操作；

否（即不是当前日志的作者）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

### 3.1.4通知

用户被关注或日志被点赞和评论时，插入通知队列中，当用户从主页点击“查看通知”这个图标时，展示该用户当前的未读通知，点击指定通知，则关闭该通知并从数据库里删除。

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

### 3.1.5首页投稿

当用户进入详情页时，立即判断当前用户的id是否与文章作者的id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，用户无法进行投稿操作。

管理员把投稿的文章进行筛选，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

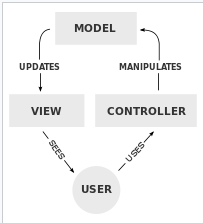
否

是

是

# 第四章 算法设计

## 4.1 MVC设计模式



对于MVC模式，大多数开发者都知道M是数据模型（Model），V是视图（View），C是控制器（Controller）。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[2]。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。

UI开发中最广泛的引用模式可能是模型视图控制器（MVC），它也是最引人注目的。坦率地说，很多经典的基于MVC的框架对于这些富客户端来说并不真实。 但是现在我们来看看它的起源[3]。

当我们看待MVC时，请务必记住，这是在任何比例上进行的非常大规模的关于用户交互工作的第一次尝试。 图形用户界面在70年代并不常见。 我刚描述的表单和控件模型在MVC之后，因为它更简单，并不总是在一个好的方式。 再次，我将使用评估示例讨论Smalltalk 80的MVC，但请注意，我正在使用Smalltalk 80的实际细节进行一些自由，以开始这是一个单色系统。

## 4.2 MVVM设计模式

### 4.2.1 MVVM介绍

Model-view-viewmodel（MVVM）是一种软件架构模式。MVVM有助于通过开发业务逻MVVM的视图模型是一个值转换器：意味着视图模型负责从模型中显示（转换）数据对象，使得对象容易管理和呈现[4]。 在这方面，视图模型比视图更模型，并处理大部分（如果不是全部）视图的显示逻辑。 视图模型可以实现调解器模式，组织对视图支持的一组用例的后端逻辑的访问。辑或后端逻辑（数据模型），分离图形用户界面的开发（无论是通过标记语言还是GUI代码）。

MVVM是Martin Fowler的演示模型设计模式的一个变体[5]。MVVM以同样的方式提取视图的状态和行为，但是Presentation Model会以不依赖于特定用户界面平台的方式抽象视图（创建视图模型）。

MVVM和Presentation Model都来源于模型 - 视图 - 控制器模式（MVC）。

MVVM由Microsoft架构师Ken Cooper和Ted Peters专门用于简化用户界面的事件驱动编程 - 通过利用Windows Presentation Foundation（WPF）（Microsoft的.NET图形系统）和Silverlight（WPF的Internet应用程序衍生）的功能。微软WPF和Silverlight架构师之一的约翰·戈斯曼（John Gossman）在2005年的博客上宣布MVVM。

Model-view-viewmodel也被称为model-view-binder，特别是在不涉及.NET平台的实现中。 ZK（使用Java编写的Web应用程序框架）和KnockoutJS（JavaScript库）使用model-view-binder

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。

### 4.2.2 MVVM模式的组件

1.模型（Model）

模型是指代表真实状态内容（面向对象的方法）的域模型，也可以是代表内容（以数据为中心的方法）的数据访问层。

2.视图（View）

与MVC和MVP模式一样，视图是用户在屏幕上看到的结构，布局和外观。

3.视图模型（View model）

View model是暴露公共属性和命令的视图的抽象概念。代替MVC模式的控制器或MVP模式的演示者，MVVM具有绑定器。在视图模型中，粘合剂介导视图和数据绑定器之间的通信[需要说明]视图模型已被描述为模型中数据的状态。

4.绑定器（Binder）

声明式数据和命令绑定在MVVM模式中是隐含的。在Microsoft解决方案堆栈中，Binder是一种名为XAML的标记语言。绑定器使开发人员不必编写繁杂的逻辑来同步视图模型和视图。当在Microsoft堆栈之外实现时，声明性数据绑定技术的存在是该模式的关键推动因素。

### 4.2.3 解释

MVVM旨在利用WPF（Windows Presentation Foundation）中的数据绑定功能，通过从视图层中删除几乎所有的GUI代码（“代码隐藏”），更好地促进了视图层开发与其余模式的分离。他们可以使用框架标记语言（例如，XAML）来创建数据绑定到视图模型，而不是用户体验（UX）开发人员编写GUI代码，而是由应用程序开发人员编写和维护。角色分离允许交互式设计人员专注于UX需求，而不是业务逻辑的编程。因此，应用程序的层可以在多个工作流中开发，以提高生产率。即使单个开发人员在整个代码库中工作，视图与模型的正确分离更有效率，因为用户界面通常根据最终用户反馈在开发周期中频繁更新。

MVVM模式试图获得由MVC提供的功能开发分离的两个优点，同时利用数据绑定的优点和框架，通过将数据绑定到尽可能接近纯应用模型。它使用绑定器，视图模型和任何业务层的数据检查功能来验证传入的数据。结果是模型和框架驱动尽可能多的操作，消除或最小化直接操纵视图（例如，代码隐藏）的应用程序逻辑。

## 4.3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

## 4.4技术选型

### 4.4.1 Vue

1.概念

vue是是一套构建用户界面的渐进式框架[6]，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[7]。

1. 特性

模板

Vue使用基于HTML的模板语法，允许您将渲染的DOM声明性地绑定到底层的Vue实例的数据。所有Vue模板都是有效的HTML，可由规范兼容的浏览器和HTML解析器解析。在引擎盖下，Vue将模板编译成虚拟DOM渲染功能。结合反应性系统，Vue能够智能地找出要重新渲染的最小数量的组件，并在应用程序状态更改时应用最少量的DOM操作。

在Vue中，您可以使用模板语法或选择使用JSX直接编写渲染功能。为了这样做只需用render函数替换template选项。渲染功能为强大的基于组件的模式开辟了可能性 - 例如，新的过渡系统现在完全基于组件，在内部使用渲染功能。

活动

Vue最独特的功能之一是不引人注意的反应系统。模型只是简单的JavaScript对象。修改它们时，视图将更新。它使状态管理非常简单直观。 Vue提供了优化的重新渲染开箱即用，无需执行任何操作。每个组件在渲染过程中跟踪其反应依赖关系，因此系统会精确地知道何时重新呈现，以及哪些组件重新呈现。

组件

组件是Vue最强大的功能之一。在大型应用中，有必要将整个应用程序分成小型，独立的，经常可重复使用的组件，使开发变得易于管理。组件扩展基本的HTML元素以封装可重用的代码。在高层次上，组件是Vue编译器附加行为的自定义元素。在Vue中，组件本质上是一个具有预定义选项的Vue实例。下面的代码片段包含Vue组件的示例[8]。该组件显示按钮并点击按钮的次数打印：

Vue.component('buttonclicked', {

props: ["initial\_count"],

data: function() {var q = {"count": 0}; return q;} ,

template: '<button v-on:click="onclick">Clicked 0 times</button>'

,

methods: {

"onclick": function() {

this.count = this.count + 1;

}

},

mounted: function() {

this.count = this.initial\_count;

}

});

### 4.4.2 Webpack

Webpack是一个开源的JavaScript模块绑定器。 Webpack采用依赖关系的模块，并生成代表这些模块的静态资产[9]。它采用依赖关系，并生成依赖图，允许Web开发人员使用模块化方法来进行Web应用程序开发。 捆绑器可以从命令行使用，也可以使用名为webpack.config.js的配置文件进行配置。

需要Node.js才能安装webpack。 关于webpack的另一个重要方面是它可以通过使用装载机来高度可扩展[10]。 装载器允许开发人员在将文件捆绑在一起时编写他们想要执行的自定义任务。

Webpack使用标记代码分割来提供需要的代码。 ECMAScript的技术委员会39正在开发一个加载附加代码的函数的标准化：proposal-dynamic-import。

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。

### 4.4.3 Ajax

Ajax（“异步JavaScript和XML”的缩写）是一组Web开发技术，在客户端使用许多Web技术来创建异步Web应用程序。 使用Ajax，Web应用程序可以异步（在后台）将数据发送到服务器并从服务器检索，而不会影响现有页面的显示和行为。 通过将数据交换层与表示层分离，Ajax允许Web页面和扩展Web应用程序动态地更改内容，而无需重新加载整个页面。实际上，现代实现通常将JSON替换为XML，这是因为JavaScript本身的优点[11]。

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。

Ajax不是一种技术，而是一组技术。 可以组合使用HTML和CSS来标记和样式信息。 使用JavaScript访问DOM以动态显示，并允许用户与所呈现的信息进行交互。[12] JavaScript和XMLHttpRequest对象提供了一种在浏览器和服务器之间异步交换数据以避免全页重新加载的方法。

### 4.4.4 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。

Apache HTTP Server，俗称Apache，是基于Apache License 2.0的免费开源跨平台Web服务器软件。 Apache由Apache Software Foundation主持的开放社区开发和维护[13]。

最常用于Unix系统（通常是Linux），该程序也可用于Microsoft Windows。2.0版改进了对非Unix的支持，例如Windows和OS / 2（和eComStation）。Apache的旧版本被移植到例如OpenVMS，和NetWare。

最初基于NCSA HTTPd服务器，Apache的开发始于1995年初，NCSA代码停滞后工作。 Apache在万维网的初步发展中发挥了关键作用，迅速超越了NCSA HTTP作为主流的HTTP服务器，自1996年4月以来一直保持最受欢迎。2009年，它成为第一个提供更多服务的Web服务器软件超过1亿个网站。

截至2016年7月，Apache仍然是使用最广泛的Web服务器软件，估计占所有活动网站的46％，百分之百的网站占43％。

在Web开发的服务器端脚本语言中，不得不提到PHP，他也可以用作通用的编程语言。最初由Rasmus Lerdorf于1994年创建，PHP参考实现现在由PHP开发团队生成。PHP最初代表个人主页，但它现在代表递归的首字母缩写PHP：超文本预处理器。

PHP代码可能嵌入到HTML或HTML5标记中，或者可以与各种Web模板系统，Web内容管理系统和Web框架结合使用。 PHP代码通常由作为Web服务器中的模块实现的PHP解释器或通用网关接口（CGI）可执行文件来处理。 Web服务器软件将解释和执行的PHP代码的结果组合在一起，该代码可以是生成的网页的任何类型的数据，包括图像。 PHP代码也可以使用命令行界面（CLI）执行，可用于实现独立的图形应用程序[14]。

由Zend Engine提供支持的标准PHP解释器是根据PHP许可证发布的免费软件。 PHP已经被广泛移植，可以免费部署在几乎所有操作系统和平台上的大多数Web服务器上。

PHP语言在没有书面形式规范或标准的情况下才会演变，直到2014年，将规范的PHP解释器作为事实上的标准。自2014年以来，工作已经开始创建一个正式的PHP规范。

MySQL是一个开源关系数据库管理系统（RDBMS），联合创始人Michael Widenius的女儿的名字和“SQL”，结构化查询语言的缩写。 MySQL开发项目已经根据GNU通用公共许可证的条款以及各种专有协议规定了其源代码。 MySQL由一家盈利性公司（由Oracle Corporation拥有的瑞典公司MySQL AB）所拥有和赞助。对于专有使用，可以使用几种付费版本，并提供附加功能。

MySQL是LAMP开源Web应用软件堆栈（和其他“AMP”堆栈）的核心组件。 LAMP是“Linux，Apache，MySQL，Perl / PHP / Python”的缩写。使用MySQL数据库的应用程序包括：TYPO3，MODx，Joomla，WordPress，phpBB，MyBB和Drupal[15]。 MySQL也用于许多高调的大型网站，包括Google （虽然不是搜索），Facebook， Twitter，Flickr，和YouTube。

## 4.5 开发工具

### 4.5.1 WebStorm

Webstorm是前端最优秀的ide，它集成了node、npm、webpack、eslint等各种js开发所用的插件的配置，自动纠错，代码高亮。它也集成了当前多种版本控制工具，大部分的功能都可以使用。最主要的是它支持node.js的调试，不需要繁杂的配置就可以进行多种操作。令人欣喜的是，webstorm内置ftp，直接就可以将他部署到远程服务器上。

### 4.5.2 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。经过简单的配置就可以和你的MYSQL连接到一起，实时纠错，语法高亮。可以直接在里面浏览数据库，浏览远程服务器上的文件，也是内置ftp，可以直接将程序部署到服务器上，可以在这个IDE上进行任何从开发到部署环节中的操作。

### 4.5.3 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。Sublime Text是具有Python应用程序编程接口（API）的专有跨平台源代码编辑器。 它本身支持许多编程语言和标记语言，其功能可以由具有插件的用户扩展，通常由社区构建并在免费软件许可证下维护。

以下是Sublime Text的功能列表：

“转到任何东西”，快速导航到文件，符号或行

“命令调色板”使用自适应匹配来快速键盘调用任意命令

同时编辑：同时对多个选定区域进行相同的交互式更改

基于Python的插件API

项目特定的偏好

通过JSON设置文件进行广泛的可定制性，包括特定于项目和平台的设置

跨平台（Windows，macOS，Linux）

兼容来自TextMate的许多语言语法

# 第五章 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） |

# 第六章 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）） |
| 文章列表页 | 拉取文章列表 | User/edit/getArticleList | userId:用户登录名 | Status(1/0)  articles(id：文章id，title：文章标题，content：文章内容：likeNum：点赞数,createTime：创建时间) |
|  | 删除文章 | User/edit/deleteArticle | Id:文章id | Status(1/0)（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) |
|  |  |  |  |  |

# 第七章 系统实现

## 7.1 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。

## 7.2 主要配置文件的核心代码

### 7.2.1 package.json文件

Package.json实际上是由node本身来使用的。当你需要引用一个外部的库或框架时，node中的npm模块会将这个库的依赖库或文件写入package.json，安装的依赖在node\_modules文件夹下；当你想根据package.json安装依赖时，只需要运行命令行npm install即可。

具体配置见附录2

### 7.2.2 vue-router配置文件

vue-router是Vue.js的官方路由器。它与Vue.js核心深入整合，使Vue.js构建单页应用程序变得轻而易举。他的特点包括：

1. 嵌套路由/视图映射
2. 路由参数，查询，通配符
3. 模块化的基于组件的路由器配置
4. 查看Vue.js转换系统提供的转换效果
5. 细粒度导航控制
6. 与自动活动CSS类的链接
7. HTML5历史记录模式或哈希模式，在IE9中自动回退
8. 可定制的滚动行为

具体配置见附录1

### 7.2.3 webpack配置文件

在webpack的配置中，有三个核心文件，分别是webpack.base.conf.js，webpack.dev.con.js，webpack.prod.conf.js。

看文件的名字就知道他们是要处理什么的配置文件了。Webpack.base.conf.js是基础配置，主要包括唯一入口的文件路径，打包后的出口文件路径，各种静态资源的loader等。Webpack.dev.conf.js是在开发环境中用到的，主要包括wepack-dev-server的配置。Webpack.prod.conf.js是在生产环境中使用的，主要是来配置打包后的生成文件路径和打包方式等。

具体配置见附录3

# 程序的实现

## 8.1 开发流程

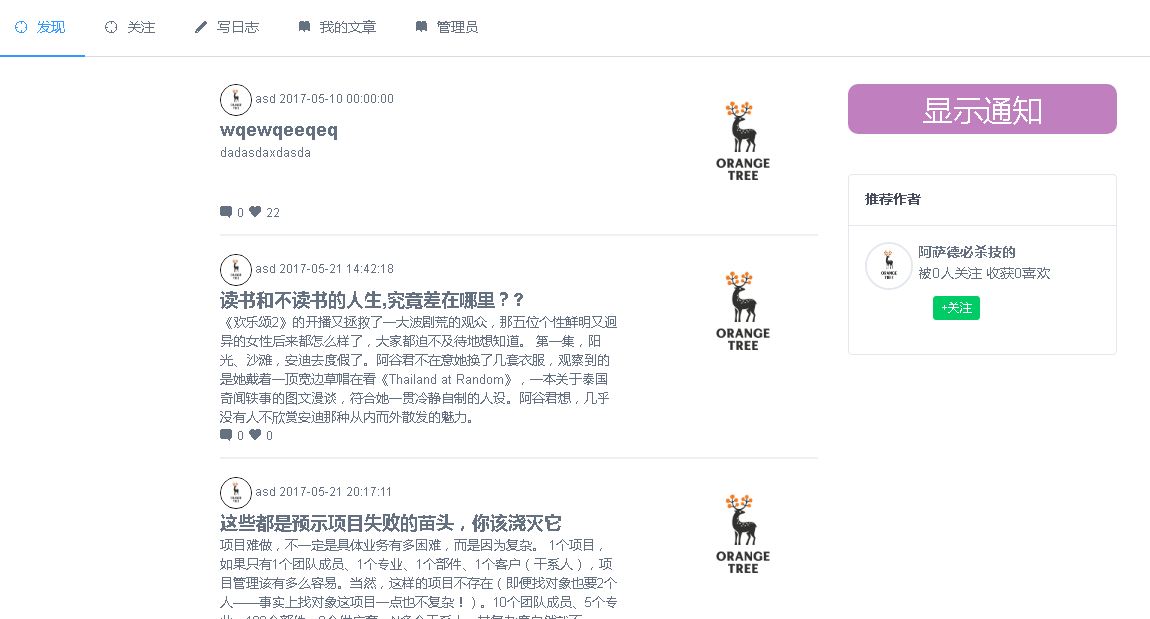
先配置好开发环境，写前端的界面，然后是编写假数据写业务逻辑；局部测试完成后，编写服务端；用一个test页进行单元测试，测试通过后，在接口文档上标注完成标识；全部接口调通后，前端绑定接口，删除假数据。

## 8.2 部署流程

运行npm run build，webpack会自动把所有前端代码打包好，先本地测试一下是否在不发请求的状态下能正常运行，由于开发环境是跨域的，所以在生产环境下先更改请求信息，如果不打算二次开发或进行维护，服务端可以删除用于跨域请求的代码，来保证安全性，之后把所有打包好的程序放到远程服务器上就可以了。

## 8.3 设计结果展示

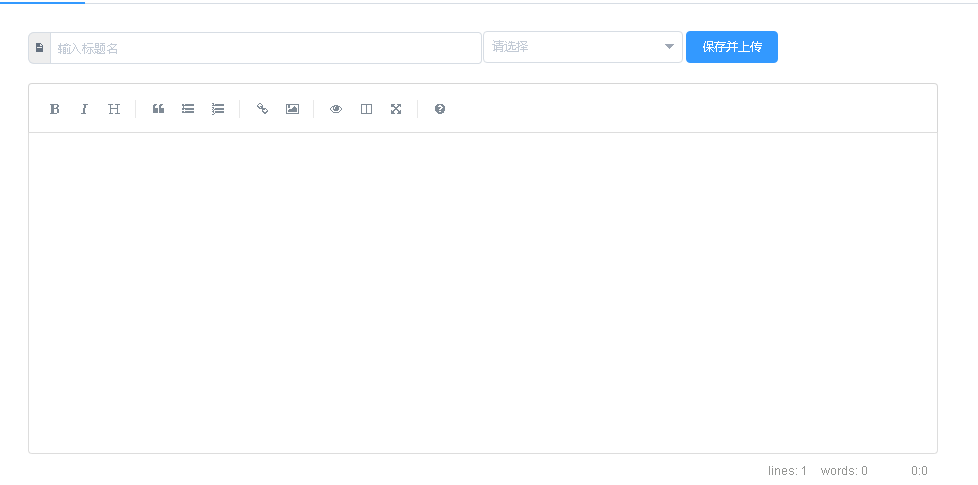
首页



关注页



编辑页



文章列表页



管理员设置页



# 第九章 谢 辞

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！

# 参考文献

[1] Towards building a blog preservation platform[A]. CERN.World Wide Web, 2014, Vol.17 (4), pp.799-825

[2] 李展飞. Web软件系统开发框架设计在MVC模式的实现[J]. 电子技术与软件工程,2017,(08):61.

[3] wikipedia,Model–View–Controller History.

<http://wiki.c2.com/?ModelViewControllerHistory>,2013-12-09.

[4] 陈涛. MVVM设计模式及其应用研究[J]. 计算机与数字工程,2014,(10):1982-1985.

[5] "The MVVM Pattern". Msdn.microsoft.com. <https://msdn.microsoft.com/en-us/library/hh848246.aspx>,2016-08-29.

[6] "Introduction — Vue.js". <https://vuejs.org/v2/guide/#What-is-Vue-js.html>, 2017-03-11.

[7]  Evan You. ["First Week of Launching Vue.js"](http://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html). <http://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html,>2017-03-11.

[8] 易剑波. 基于MVVM模式的WEB前端框架的研究[J]. 信息与电脑(理论版),2016,(19):76-77+84.

[9] "Webpack: An Introduction - Wisdom Geek". Wisdom Geek.,2017-01-12. Retrieved 2017-01-16. <https://www.wisdomgeek.com/web-development/webpack-introduction>

[10] 彭娜. 基于Node.JS博客系统的设计与实现[D].大连理工大学,2013.

[11] 温立辉. AJAX异步交互技术浅析[J]. 山东工业技术,2017,(04):213.

[12] ["JSON: The Fat-Free Alternative to XML"](http://www.json.org/xml.html). www.json.org. <http://www.json.org/xml.html>

. 2017-02-17

[13] 周奎,王超,黄连丽. 基于PHP与MySQL的教务管理系统设计[J]. 软件导刊,2017,(05):89-90.

[14] 李华明. 基于PHP和MySQL的网上购物系统设计与实现[D].电子科技大学,2014.

[15] 赵红霞,王建. 基于PHP+MySQL结构的微课在线学习系统设计与实现[J]. 信息通信,2017,(03):84-85.

# 附录1

本系统的全部源码托管到了我的github上：<https://github.com/mytac/blog-system-with-vue>

Vue-router配置：

// main.js

import Vue from 'vue'

import App from './App'

import iView from 'iview';

import $ from 'jquery';

window.$ = $

window.jQuery = $

export default $

import 'iview/dist/styles/iview.css';

import signin from './components/sign/signin'

import signup from './components/sign/signup'

import index from './components/index/index'

import attention from './components/attention/attention.vue'

import edit from './components/edit/edit.vue'

import articles from './components/articles/articles.vue'

import detail from './components/articles/detail.vue'

import mainview from'./components/attention/mainview.vue'

import atten\_hot from './components/attention/sub\_hot.vue'

import atten\_new from './components/attention/sub\_new.vue'

import blogs from './components/index/recommendBlog.vue'

import admin from './components/admin/main.vue'

import test from './components/test.vue'

var VueRouter = require('vue-router')

Vue.use(VueRouter)

Vue.use(iView)

new Vue({

el: 'body',

components: { App }

})

var router=new VueRouter()

//路由

router.map({

'/':{

component:index

},

'/:userId': {

name:'index',

component: index

},

'/signin': {

component: signin

},

'/signup': {

component: signup

},

'/attention':{

component:attention,

subRoutes:{

'/':{

component:mainview,

subRoutes:{

'/':{

component:blogs

},

'/:choseType':{ //hot

name:'blogsType',

component:blogs

}

}

},

'/:switchname':{

name:'attention',

component:mainview,

subRoutes:{

'/':{

component:blogs

},

'/:choseType':{ //hot

name:'blogsType',

component:blogs

}

}

}

}

},

'/edit/:userId/:textId':{

name:'edit',

component:edit

},

'/add/:userId':{

name:'add',

component:edit

},

'/articles/:userId':{

name:'articles',

component:articles

},

'/detail/:userId/:articleId':{

name:'detail',

component:detail

},

'/admin/:userId':{

name:'admin',

component:admin

},

'/test':{

name:'test',

component:test

}

})

router.start(App,'#app')

# 附录2

依赖文件

//package.json

{

"name": "blog-system-with-vue",

"version": "1.0.0",

"description": "A Vue.js project",

"author": "Tactics Chang <my\_tac@126.com>",

"private": true,

"scripts": {

"dev": "node build/dev-server.js",

"build": "node build/build.js",

"test": ""

},

"dependencies": {

"babel-runtime": "^6.0.0",

"bootstrap-webpack": "0.0.5",

"exports": "0.0.1",

"exports-loader": "^0.6.4",

"imports-loader": "^0.7.1",

"iview": "^0.9.16",

"jquery": "^3.2.1",

"less": "^2.7.2",

"less-loader": "^4.0.0",

"markdown": "^0.5.0",

"style-loader": "^0.15.0",

"vue": "^1.0.21",

"vue-router": "^0.7.13",

"vue-simplemde": "^0.3.7",

"webpack": "^1.14.0"

},

"devDependencies": {

"babel-core": "^6.0.0",

"babel-loader": "^6.0.0",

"babel-plugin-transform-runtime": "^6.0.0",

"babel-preset-es2015": "^6.0.0",

"babel-preset-stage-2": "^6.0.0",

"babel-register": "^6.0.0",

"bootstrap": "^3.3.7",

"connect-history-api-fallback": "^1.1.0",

"css-loader": "^0.23.0",

"eventsource-polyfill": "^0.9.6",

"express": "^4.13.3",

"extract-text-webpack-plugin": "^1.0.1",

"file-loader": "^0.8.4",

"function-bind": "^1.0.2",

"html-webpack-plugin": "^2.8.1",

"http-proxy-middleware": "^0.12.0",

"json-loader": "^0.5.4",

"ora": "^0.2.0",

"shelljs": "^0.6.0",

"url-loader": "^0.5.7",

"vue": "^1.0.26",

"vue-hot-reload-api": "^1.2.0",

"vue-html-loader": "^1.0.0",

"vue-html5-editor": "^1.0.2",

"vue-loader": "^8.3.0",

"vue-style-loader": "^1.0.0",

"webpack": "^1.12.2",

"webpack-dev-middleware": "^1.4.0",

"webpack-hot-middleware": "^2.6.0",

"webpack-merge": "^0.8.3"

}

}

# 附录3

// webpack.base.conf.js

var path = require('path')

var config = require('../config')

var utils = require('./utils')

var projectRoot = path.resolve(\_\_dirname, '../')

module.exports = {

entry: {

app: './src/main.js'

},

output: {

path: config.build.assetsRoot,

publicPath: process.env.NODE\_ENV === 'production' ? config.build.assetsPublicPath : config.dev.assetsPublicPath,

filename: '[name].js'

},

resolve: {

extensions: ['', '.js', '.vue'],

fallback: [path.join(\_\_dirname, '../node\_modules')],

alias: {

'src': path.resolve(\_\_dirname, '../src'),

'assets': path.resolve(\_\_dirname, '../src/assets'),

'components': path.resolve(\_\_dirname, '../src/components')

}

},

resolveLoader: {

fallback: [path.join(\_\_dirname, '../node\_modules')]

},

module: {

loaders: [

{

test: /bootstrap\/js\//,

loader: 'imports?jQuery=jquery'

},

{

test: /\.vue$/,

loader: 'vue'

},

{

test: /\.js$/,

loader: 'babel',

include: projectRoot,

exclude: /node\_modules/

},

{

test: /\.json$/,

loader: 'json'

},

{

test: /\.html$/,

loader: 'vue-html'

},

{

test: /\.(png|jpe?g|gif|svg)(\?.\*)?$/,

loader: 'url',

query: {

limit: 10000,

name: utils.assetsPath('img/[name].[hash:7].[ext]')

}

},

{

test: /\.(woff2?|eot|ttf|otf|woff)(\?.\*)?$/,

loader: 'url',

query: {

limit: 10000,

name: utils.assetsPath('fonts/[name].[hash:7].[ext]')

}

}

]

},

vue: {

loaders: utils.cssLoaders()

}

}

//webpack.dev.conf.js

var config = require('../config')

var webpack = require('webpack')

var merge = require('webpack-merge')

var utils = require('./utils')

var baseWebpackConfig = require('./webpack.base.conf')

var HtmlWebpackPlugin = require('html-webpack-plugin')

// add hot-reload related code to entry chunks

Object.keys(baseWebpackConfig.entry).forEach(function (name) {

baseWebpackConfig.entry[name] = ['./build/dev-client'].concat(baseWebpackConfig.entry[name])

})

module.exports = merge(baseWebpackConfig, {

module: {

loaders: utils.styleLoaders({ sourceMap: config.dev.cssSourceMap })

},

// eval-source-map is faster for development

devtool: '#eval-source-map',

plugins: [

new webpack.DefinePlugin({

'process.env': config.dev.env

}),

// https://github.com/glenjamin/webpack-hot-middleware#installation--usage

new webpack.optimize.OccurenceOrderPlugin(),

new webpack.HotModuleReplacementPlugin(),

new webpack.NoErrorsPlugin(),

// https://github.com/ampedandwired/html-webpack-plugin

new HtmlWebpackPlugin({

filename: 'index.html',

template: 'index.html',

inject: true

})

]

})

// webpack.prod.conf.js

var path = require('path')

var config = require('../config')

var utils = require('./utils')

var webpack = require('webpack')

var merge = require('webpack-merge')

var baseWebpackConfig = require('./webpack.base.conf')

var ExtractTextPlugin = require('extract-text-webpack-plugin')

var HtmlWebpackPlugin = require('html-webpack-plugin')

var env = process.env.NODE\_ENV === 'testing'

? require('../config/test.env')

: config.build.env

var webpackConfig = merge(baseWebpackConfig, {

module: {

loaders: utils.styleLoaders({ sourceMap: config.build.productionSourceMap, extract: true })

},

devtool: config.build.productionSourceMap ? '#source-map' : false,

output: {

path: config.build.assetsRoot,

filename: utils.assetsPath('js/[name].[chunkhash].js'),

chunkFilename: utils.assetsPath('js/[id].[chunkhash].js')

},

vue: {

loaders: utils.cssLoaders({

sourceMap: config.build.productionSourceMap,

extract: true

})

},

plugins: [

// http://vuejs.github.io/vue-loader/workflow/production.html

new webpack.DefinePlugin({

'process.env': env

}),

new webpack.optimize.UglifyJsPlugin({

compress: {

warnings: false

}

}),

new webpack.optimize.OccurenceOrderPlugin(),

// extract css into its own file

new ExtractTextPlugin(utils.assetsPath('css/[name].[contenthash].css')),

// generate dist index.html with correct asset hash for caching.

// you can customize output by editing /index.html

// see https://github.com/ampedandwired/html-webpack-plugin

new HtmlWebpackPlugin({

filename: process.env.NODE\_ENV === 'testing'

? 'index.html'

: config.build.index,

template: 'index.html',

inject: true,

minify: {

removeComments: true,

collapseWhitespace: true,

removeAttributeQuotes: true

// more options:

// https://github.com/kangax/html-minifier#options-quick-reference

},

// necessary to consistently work with multiple chunks via CommonsChunkPlugin

chunksSortMode: 'dependency'

}),

// split vendor js into its own file

new webpack.optimize.CommonsChunkPlugin({

name: 'vendor',

minChunks: function (module, count) {

// any required modules inside node\_modules are extracted to vendor

return (

module.resource &&

/\.js$/.test(module.resource) &&

module.resource.indexOf(

path.join(\_\_dirname, '../node\_modules')

) === 0

)

}

}),

// extract webpack runtime and module manifest to its own file in order to

// prevent vendor hash from being updated whenever app bundle is updated

new webpack.optimize.CommonsChunkPlugin({

name: 'manifest',

chunks: ['vendor']

})

]

})

if (config.build.productionGzip) {

var CompressionWebpackPlugin = require('compression-webpack-plugin')

webpackConfig.plugins.push(

new CompressionWebpackPlugin({

asset: '[path].gz[query]',

algorithm: 'gzip',

test: new RegExp(

'\\.(' +

config.build.productionGzipExtensions.join('|') +

')$'

),

threshold: 10240,

minRatio: 0.8

})

)

}

module.exports = webpackConfig