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。

**关键词：** SPA应用 前后分离 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.

**Key Words:** SPA Separation between front-end and back-end MVC引 言

Web 2.0出现以来社交媒体内容和用户参与度急剧增加。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保存必须是一个开始完成活动，它应该包含网络资源的整个生命周期。 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。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](#_Toc22846)

[1.1目标 1](#_Toc25835)

[1.2本系统的特点和目标用户 1](#_Toc18473)

[第二章 需求分析 3](#_Toc3729)

[2.1基本需求 3](#_Toc8734)

[2.1.1功能需求 3](#_Toc19868)

[2.1.2性能需求 3](#_Toc2403)

[2.2运行环境需求 4](#_Toc2360)

[2.2.1软件环境 4](#_Toc26225)

[2.2.2硬件环境 4](#_Toc5644)

[2.3 E-r图 4](#_Toc22807)

[2.3.1用户实体 4](#_Toc25422)

[2.3.2 通知实体 5](#_Toc12243)

[2.3.3文章实体 5](#_Toc6455)

[2.3.4评论实体 5](#_Toc32480)

[第三章 系统设计 7](#_Toc31831)

[3.1流程图及流程说明 7](#_Toc31945)

[.3.1.1.登录注册 7](#_Toc17504)

[3.1.2查看用户信息 7](#_Toc14466)

[3.1.3日志查看/编辑 8](#_Toc19033)

[3.1.4通知 9](#_Toc7507)

[3.1.5首页投稿 9](#_Toc20824)

[第四章 算法设计 11](#_Toc6941)

[4.1 MVC设计模式 11](#_Toc1597)

[4.2 MVVM设计模式 11](#_Toc15808)

[4.3系统交互模式设计 11](#_Toc30205)

[4.4技术选型 12](#_Toc5287)

[4.4.1 Vue 12](#_Toc30585)

[4.4.2 Webpack 12](#_Toc5062)

[4.4.3 Ajax 12](#_Toc11521)

[4.4.4 Apache+php+mysql 12](#_Toc8600)

[4.5 开发工具 13](#_Toc8940)

[4.5.1 Webstorm 13](#_Toc25042)

[4.5.2 Phpstorm 13](#_Toc31725)

[4.5.3 Sublime text3 13](#_Toc2791)

[第五章 数据库设计 14](#_Toc24841)

[第六章 接口设计 16](#_Toc17565)

[第七章 系统实现 21](#_Toc23141)

[7.1 开发步骤 21](#_Toc16438)

[7.2 主要配置文件的核心代码 21](#_Toc7992)

[第八章 程序/算法测试 22](#_Toc32512)

[第九章 总结 23](#_Toc7853)

[附录 24](#_Toc8778)

# 

# 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，进行添加、删除、编辑等操作
8. 注册用户可以管理自己的文章分类列表，进行添加、删除、编辑等操作

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。使用该系统不需要任何的计算机知识，只需要掌握markdown基础语法，就可以任意书写博客了。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；已注册用户享有更多权限，可以对自己的博客进行管理等；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。

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

### 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

### 3.1.5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

# 第四章 算法设计

## 4.1 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。

## 4.2 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。

## 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

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。

### 4.4.2 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。

### 4.4.3 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。

### 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。

## 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轮子的配置，自动纠错，代码高亮。

### 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。

### 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。

# 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

见附录

# 第八章 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。

# 第九章 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！

# 附录

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')

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"

}

}