**基于Node.js和MongoDB的博客系统编写思路**

代码托管地址: <https://github.com/Vinntoe/meblog/>

代码提交历史活动可以<https://github.com/Vinntoe/meblog/commits/master>看到

已经完成及TODO情况 <https://github.com/Vinntoe/meblog#todo>

**按照面向对象以及MVC编程的模式,将整个博客系统分为几个部分**

**1 控制器部分(control) ,或者叫路由部分(routes)**

文件位置 routes/index.js

负责将从前端接受到的请求 经过鉴权 判断等,将请求的动作分配到适合的处理程序中进行处理,同时将结果返回给前端界面

**2 模板视图部分(views)**

文件位置 views/\*.ejs

将models部分的处理程序从数据库中检索到的数据组装成HTML片段,然后发送给控制器,再经控制器发送到前端. 主要有以下几个部分

*注: 冒号开头的为变量,在访客进行访问的时候根据访客请求进行变化*

**A** /index 负责访客直接请求首页时进行展示文章列表

**B** /index/page/:page 负责当访客点击页码时展示请求页码的文章列表

**C** /u/:user 负责访客点击某个用户名称时进行展示该用户的文章列表

**D** /u/:user/page/:page 负责当访客点击某个用户文章列表中的页码时展示该页码的文章列表

**E** /reg GET方法请求该页面时负责展示注册页面, POST方法请求该页面时进行字段审核,成功后进行数据库写入操作,增加新用户.

**F** /login GET方法请求该页面时负责展示访客登录界面, POST方法请求该页面时进行用户鉴权操作,鉴权成功后重定向网页到/index 页面

**G** /logout 负责清除用户登录信息,清除成功后,引导访客重新到达/index页面

**H** /post 当用户登录后,GET方法请求该页面时负责展示发表文章页面,让已登录用户进行发表文章操作,POST方法请求该页面时进行新文章的数据库写入操作,添加新文章

**I** /p/:id/comment/page/:page 当用户点击某个文章的标题时进入单篇文章显示模式,如果请求页码有评论数据的话,显示请求页码的评论,以及显示评论框允许访客评论.

**J** /comment 在单篇文章模式下,当用户输入评论后,使用POST方法请求该地址,进行评论数据的数据库写入操作,写入成功后重定向网页到访客提交评论的/p/:id页面,因此此页面对访客不可见.

**3 模型部分(models)**

主要作用是进行数据库的一系列操作. 主要分为以下几个模型

**A** models/db.js 数据库处理类,负责处理和数据库有关的操作.定义一些操作,然后将接口暴露出去,供其它类调用..

**B** models/user.js 用户处理类,负责处理在用户 注册 登录 退出 过程中的一些检索 验证 保存操作

**C** models/post.js 文章处理类,利用db.js中的接口,负责

**1)** 在浏览过程中将文章从数据库中检索出来,并进行前期处理,然后将文章数据交给模板进行解析,组装成浏览器可以识别的HTML代码,然后展现给用户.

**2)** 在已登录用户进行发表文章的过程中,对评论数据进行包装,然后利用db.js中的接口,将文章数据存储到数据库中

**D** models/comment.js评论处理类 负责

**1)** 在用户进行浏览文章的时候,利用db.js中的接口,从数据库中将评论数据检索出来,然后交给模板进行解析,在前端展示.

**2)** 在用户进行评论的时候,对评论数据进行包装,然后利用db.js中的接口,将评论数据存储到数据库中,以便下次可以正确从数据库中进行检索出来.

**视完成进度,想要添加的页面及功能**

**页面及功能:**

**1** /setting/:user 用户设置页面,对个人信息进行设置或更改

**2** /tags/:tag 标签页面,用于检索已设置某标签的所有文章,并分页展示

**功能:**

**1** 已登录用户发表文章时,添加文章的标签

**2** 访客第一次评论成功后,将访客邮箱对应的Gravatar头像图片缓存到本地目录,以后再次展示头像图片时直接从本地目录读取

**3** /index页面, /u/:user页面 及/tags:tag页面,在每篇文章下显示已有评论数目,允许访客点击评论数目后跳转到单篇文章页面的评论部分进行评论