**目录**

[1 绪论 0](#_Toc356563314)

[1.1 主要研究的目的及内容 0](#_Toc356563315)

[1.2 研究现状 1](#_Toc356563316)

[2 相关技术简介 2](#_Toc356563317)

[2.1 Git 2](#_Toc356563318)

[2.2 Node.js 2](#_Toc356563319)

[2.3 Express.js 3](#_Toc356563320)

[2.4 NoSQL 3](#_Toc356563321)

[2.5 NoSQL与关系型数据库设计理念比较 5](#_Toc356563322)

[2.6 MongoDB 6](#_Toc356563323)

[2.7 MarkDown 6](#_Toc356563324)

[2.8 EJS 8](#_Toc356563325)

[2.9 Bootstrap 10](#_Toc356563326)

[2.10 JQuery 10](#_Toc356563327)

[3 系统设计 11](#_Toc356563328)

[3.1 概要设计 11](#_Toc356563329)

[3.2 模型(Models)部分 11](#_Toc356563330)

[3.3 视图(Views)部分 14](#_Toc356563331)

[3.4 控制器(Controls)部分 17](#_Toc356563332)

[3.5 协作图 17](#_Toc356563333)

1. 绪论

## 主要研究的目的及内容

随着web2.0的到来，网站和用户不再是单纯的信息输出和信息接受者，用户慢慢转变为网站内容的生产者，网站慢慢成为用户活动的载体。因此，网站的响应速度，用户体验，被用来当做衡量一个网站是否合格的重要指标。

为了提升网站的响应速度，催生出各种相关技术来达到这个目的。

本人所在公司是一个专注于地图数据采集、测绘，并向用户提供准确的数字地图、相关的导航功能以及其它相关功能在内的公司。因此，对于网站的响应速度以及并发量、数据库使用技术有着极高的要求。需要网站在海量用户，超高并发量的访问请求下，能够快速做出响应。并且，由于海量的地图数据，因此需要一个高效的数据库方案来达到快速检索用户所需信息，并返回给用户。

目前，公司使用的是最新发展的 Node.js 作为服务器端，同时配合其它一些新型的数据库技术，来达到能够应对超高并发量的访问请求的目的。

本课题的主要目的是通过运用在公司中使用的Node.js服务器端,新型的数据库 MongoDB 以及Node.js的服务器框架Express.js 和其前端模板引擎 EJS,配合JavaScript框架jQuery、html解析方法MarkDown来完成一个多人博客系统。通过完成一个博客系统，来达到了解相关使用技术，并对技术使用过程中的一些注意事项做到初步掌握。以期在以后的工作中，当遇到相似的问题能够快速分析问题，找到问题原因，并提出合理的解决方案。

* 1. 研究现状

博客，又译为网络日志、部落格或部落阁等，是一种通常由个人管理、不定期张贴新的文章的网站。博客上的文章通常根据张贴时间，以倒序方式由新到旧排列。博客系统，是指使用计算机语言编写，并便于用户安装和使用，在互联网上建立个人博客的一整套系统。

**网上几种流行的博客系统**

* Wordpress 是一种使用 PHP 语言和 MySQL 数据库开发的开源、免费的 Blog（博客，网志）引擎。
* PJ-Blog 是由舜子（PuterJam）所开发的一套开源免费的中文个人博客系统程序，采用asp+Access的技术
* Z-Blog 是一款小巧而强大的基于Asp平台的Blog程序。著名的科技点评博客 – 月光博客 使用的博客程序就是Z-Blog
* EMLog 是一款基于PHP语言和MySQL数据库的开源、免费、功能强大的个人或多人联合撰写的博客系统(blog)。

虽然网上已经有多种比较成熟的博客系统，但是通过观察可以发现，多种博客系统基本都是采用了PHP+MySQL的技术组合，重合度太高。而且，随着使用人数的不断增加，博客系统开发者为了迎合大多数人的需要，不断增加一些新功能，趋向于转变成为一个内容管理系统（CMS），慢慢变得代码臃肿、维护困难、运行缓慢、资源消耗严重，而且慢慢偏离了博客系统的初衷—一个供个人或多人发表文章、分享经验的日志管理系统。

因此，基于以上原因，同时希望能够在编写系统的过程中，运用一些现今发展的新技术，而对这些新技术有一个初步的了解，所以提出了这个课题。

# 相关技术简介

* 1. Git

Git是一个分布式版本控制／软件配置管理软件，原来是linux内核开发者林纳斯•托瓦兹（Linus Torvalds）为了更好地管理linux内核开发而创立的版本控制工具.

与CVS、Subversion一类的集中式版本控制工具不同，它采用了分布式版本库的作法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。Git的速度很快，这对于诸如Linux kernel这样的大项目来说自然很重要。

Git最为出色的是它的合并追踪（***merge tracing***）能力。

Git和其他版本控制系统（如CVS）有不少的差别，Git本身关心档案的整体性是否有改变，但多数的CVS，或Subversion系统则在乎档案内容的差异。因此Git更像一个档案系统，直接在本机上取得资料，不必连线到host端取资料回来。

由于svn需要有一个专门的代码托管服务器,而本人做的是一个个人项目,没有svn需要的统一的代码托管服务器,但是为了在不同机器之间代码的共享和快速迁移,所以本人选择了git作为代码版本控制工具,同时使用网上流行的基于Git的托管服务器[GitHub]。

-----------------维基百科

* 1. Node.js

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎。目的是为了提供撰写可扩充网络程序，如Web服务。第一个版本由Ryan Dahl于2009年发布，后来，Joyent公司雇用了Dahl，并协助发展Node.js。

与一般JavaScript不同的地方，Node.js并不是在Web浏览器上运行，而是一种在服务器上运行的服务端JavaScript。Node.js实现了部份CommonJS规格。

**Node.js的优点**

nodejs作为一个新兴的后台语言，有很多吸引人的地方：

* RESTful API
* 单线程 Node.js可以在不新增额外线程的情况下，依然可以对任务进行并行处理 —— Node.js是单线程的，它通过事件轮询（event loop）来实现并行操作.
* 非阻塞IO Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（寻址并检索文件），硬盘准备好的时候非阻塞接口会通知Node来处理完成事件。
* 事件驱动 简单来说即事先定义一系列事件处理操作,当服务器进行到某个事件时,会触发预先为这个事件定义（绑定）好的事件处理函数,但是这个函数什么时候执行却是不确定的。
* V8虚拟机

常见的JavaScript引擎分为几类，

* IE – Jscript
* Gogle Chrome – V8
* Mozilla Firefox – SpiderMonkey
* Opera - Carakan

Google的V8引擎本身使用了一些最新的编译技术。这使得用Javascript这类脚本语言编写出来的代码与用C这类高级语言写出来的代码性能相差无几，却节省了开发成本。对性能的苛求是Node的一个关键因素。

Javascript是一个事件驱动语言，Node利用了这个优点，编写出可扩展性高的服务器。Node采用了一个称为“事件循环(event loop）”的架构，使得编写可扩展性高的服务器变得既容易又安全

-------百度百科

* 1. Express.js

Express.js是一个小而强大的Node.js的Web程序框架,它提供了一系列强大的功能来创建各种web应用程序. 在大量原生的HTTP工具方法和Connect中间件的帮助下,用户可以快速又方便的创建强健且用户友好的API. Express仅仅是给应用程序包装了一层轻量级的功能函数,不会影响任何node.js特性.

----------express.js官网

* 1. NoSQL

传统的关系型数据库的代表是 MySQL、SQLServe、Oracle、SQLite等。

随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，例如

1. High performance - 对数据库高并发读写的需求

web2.0网站要根据用户个性化信息来实时生成动态页面和提供动态信息，所以基本上无法使用动态页面静态化技术，因此数据库并发负载非常高，往往要达到每秒上万次读写请求。关系数据库应付上万次SQL查询还勉强顶得住，但是应付上万次SQL写数据请求，硬盘IO就已经无法承受了。其实对于普通的BBS网站，往往也存在对高并发写请求的需求。

1. Huge Storage - 对海量数据的高效率存储和访问的需求

对于大型的SNS网站，每天用户产生海量的用户动态，以国外的Friendfeed为例，一个月就达到了2.5亿条用户动态，对于关系数据库来说，在一张2.5亿条记录的表里面进行SQL查询，效率是极其低下乃至不可忍受的。再例如大型web网站的用户登录系统，例如腾讯，盛大，动辄数以亿计的帐号，关系数据库也很难应付。

1. High Scalability && High Availability- 对数据库的高可扩展性和高可用性的需求

在基于web的架构当中，数据库是最难进行横向扩展的，当一个应用系统的用户量和访问量与日俱增的时候，你的数据库却没有办法像web server和app server那样简单的通过添加更多的硬件和服务节点来扩展性能和负载能力。对于很多需要提供24小时不间断服务的网站来说，对数据库系统进行升级和扩展是非常痛苦的事情，往往需要停机维护和数据迁移，

在上面提到的“三高”需求面前，关系数据库遇到了难以克服的障碍，而对于web2.0网站来说，关系数据库的很多主要特性却往往无用武之地，例如：

1. 数据库事务一致性需求

很多web实时系统并不要求严格的数据库事务，对读一致性的要求很低，有些场合对写一致性要求也不高。因此数据库事务管理成了数据库高负载下一个沉重的负担。

1. 数据库的写实时性和读实时性需求

对关系数据库来说，插入一条数据之后立刻查询，是肯定可以读出来这条数据的，但是对于很多web应用来说，并不要求这么高的实时性。

1. 对复杂的SQL查询，特别是多表关联查询的需求

任何大数据量的web系统，都非常忌讳多个大表的关联查询，以及复杂的数据分析类型的复杂SQL报表查询，特别是SNS类型的网站，从需求以及产品设计角度，就避免了这种情况的产生。往往更多的只是单表的主键查询，以及单表的简单条件分页查询，SQL的功能被极大的弱化了。

因此，关系数据库在这些越来越多的应用场景下显得不那么合适了，为了解决这类问题的非关系数据库应运而生，并且由于其本身的特点得到了非常迅速的发展。NoSQL，指的就是非关系型的数据库。

* 1. NoSQL与关系型数据库设计理念比较

关系型数据库中的表都是存储一些格式化的数据结构，每个元组字段的组成都一样，即使不是每个元组都需要所有的字段，但数据库会为每个元组分配所有的字段，这样的结构可以便于表与表之间进行连接等操作，但从另一个角度来说它也是关系型数据库性能瓶颈的一个因素。

而非关系型数据库以键值对存储，如下面代码所示

Data1 = { "key1" : "value1" , "key2" : "value2"}

Data2 = { "key1" : "value1", "key3" : "value2", "key2": "value3"}

它的结构不固定，每一个元组可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对，这样就不会局限于固定的结构，可以减少一些时间和空间的开销

**NoSQL数据库的特点:**

* 它们可以处理超大量的数据。
* 它们运行在便宜的PC服务器集群上。
* PC集群扩充起来非常方便并且成本很低，避免了“sharding”操作的复杂性和成本。
* 它们击碎了性能瓶颈。

NoSQL的支持者称，通过NoSQL架构可以省去将Web或Java应用和数据转换成SQL友好格式的时间，执行速度变得更快。

“SQL并非适用于所有的程序代码，” 对于那些繁重的重复操作的数据，SQL值得花钱。但是当数据库结构非常简单时，SQL可能没有太大用处。

* 没有过多的操作。

虽然NoSQL的支持者也承认关系数据库提供了无可比拟的功能集合，而且在数据完整性上也发挥绝对稳定，他们同时也表示，企业的具体需求可能没有那么多。

**NoSQL数据库的优点：**

* 易扩展
* 大数据量
* 高性能
* 灵活的数据模型
* 高可用性
  1. MongoDB

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似json的bjson格式，因此可以存储比较复杂的数据类型。

Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。它的特点是高性能、易部署、易使用，存储数据非常方便。

**主要功能特性：**

* 面向集合存储，易存储对象类型的数据
* 模式自由
* 支持动态查询
* 支持完全索引，包含内部对象
* 支持查询
* 支持复制和故障恢复
* 使用高效的二进制数据存储，包括大型对象（如视频等）
* 自动处理碎片，以支持云计算层次的扩展性
* 支持RUBY，PYTHON，JAVA，C++，PHP等多种语言
* 文件存储格式为BSON（一种JSON的扩展）
* 可通过网络访问

------------------百度百科

* 1. MarkDown

Markdown 是一种轻量级标记语言，创始人为约翰•格鲁伯（John Gruber）和亚伦•斯沃茨（Aaron Swartz）。它允许人们“使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML(或者HTML)文档”。这种语言吸收了很多在电子邮件中已有的纯文本标记的特性。使其成为可读性最大并可再发行的可输入输出的格式。

**Markdown优点**

* 纯文本，所以兼容性极强，可以用所有文本编辑器编辑。
* 可以专注写作而不是排版。用Word写作的时候，经常浪费大量时间去思考排版，而用Markdown，写作完成后可以自己css定义样式或使用别人的css样式。
* Markdown 语法简单(语法说明,见附件Markdown 语法说明 (简体中文版).html)，很快就可以学会。
* Markdown 的标记语法有极好的可读性。
* 格式转换方便，Markdown 的文本你可以轻松可以通过各种工具（如http://johnmacfarlane.net/pandoc/）转换为html、pdf、epub、mobi等格式。
* 可以使用git、svn进行版本管理，协作也更方便。

**简单的MarkDown和HTML写法对比：**

比如要实现下面的无序列表效果

无序列表1，**加重**效果

无序列表2，*倾斜*效果

无序列表3，普通效果

无序列表4，[链接1](http://isayme.com)状态

无序列表5，[链接2](http://isayme.com)效果

HTML代码需要使用下面的写法

<ul>

<li>无序列表1，<strong>加重</strong>效果</li>

<li>无序列表2，<em>倾斜</em>效果</li>

<li>无序列表3，普通效果</li>

<li>无序列表4，<a href="http://ISayme.com">链接1</a>效果</li>

<li>无序列表5，<a href="http://ISayme.com">链接2</a>效果</li>

</ul>

而使用MarkDown书写,则只需要写成如下格式

- 无序列表1，\*\*加重\*\*效果

- 无序列表2，\_倾斜\_效果

- 无序列表3，普通效果

- 无序列表4，[链接1]效果

- 无序列表5，[链接2]效果

[链接1]: http://isayme.com

[链接2]: http://isayme.com

当使用Html写作方式时,需要自己注意书写开始闭合标签.当写作的结构很复杂时,就需要很多标签来格式化文本,但是标签数目过多的时候,满满的文章中就会到处是开始闭合标签,而把真正需要写作的内容淹没,可读性非常差.但是使用markdown写作方式时,markdown本身的语法就非常具有可读性。

* 1. EJS

Ejs是一个express.js的前端框架,配合现在流行的MVC编程模式,将模板和数据分开,使得数据层和模板层耦合度大降低,提高开发效率,而且可读性很强.

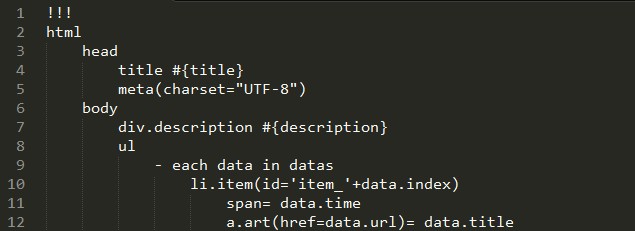
**Ejs和jade对比**

1. 解释效率比较

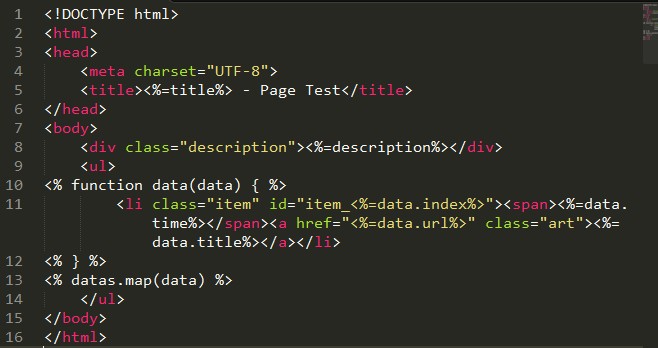
WEB模板jade、ejs、handlebars 万行代码解释效率比较，jade完败

简单的以1万行数据，进行解释效率比较：

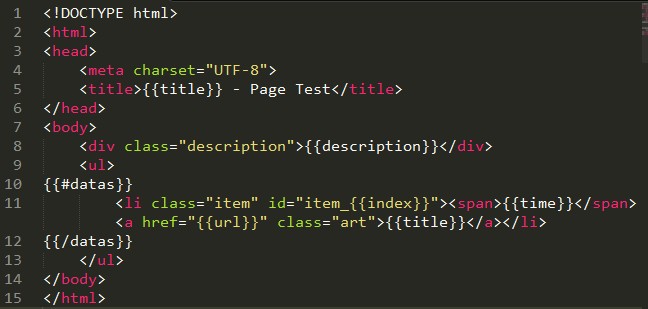
Jade 模板：



ejs 模板：



handlebars 模板：



效率比较结果（平均消耗时间，约数）

jade 287ms > ejs 43ms > handlebars 28ms

jade因为采用了类似zen code的语法，比较新奇，但效率极其低下。如果只保留部分的1万行数据解释，则约为245ms。

- 可视化太弱，甚至可以说是毫无可视化可言，学习成本高，维护与团队合作成本高，语法过于晦涩、复杂。

因此,经过一番比较之后,最终决定选用EJS模板

----------------------http://cnodejs.org/topic/50e70edfa7e6c6171a1d70fa

* 1. Bootstrap

Bootstrap是Twitter推出的一个开源的用于前端开发的工具包。它由Twitter的设计师Mark Otto和Jacob Thornton合作开发，是一个CSS/HTML框架。Bootstrap提供了优雅的HTML和CSS规范，它即是由动态CSS语言Less写成。Bootstrap一经推出后颇受欢迎，一直是GitHub上的热门开源项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。

* 1. JQuery

jQuery是一个兼容多浏览器的javascript库，核心理念是write less,do more(写的更少,做的更多)。

jQuery在2006年1月由美国人John Resig在纽约的barcamp发布，吸引了来自世界各地的众多JavaScript高手加入，现在由Dave Methvin率领团队进行开发。

如今，jQuery已经成为最流行的javascript库，目前全世界57.3%的网站使用它。也就是说，10个网站里面，有6个使用jQuery。如果只考察使用工具库的网站，这个比例就会上升到惊人的91.7%。

jQuery是免费、开源的，使用MIT许可协议。jQuery的语法设计可以使开发者更加便捷，例如操作文档对象、选择DOM元素、制作动画效果、事件处理、使用Ajax(Asynchronous JavaScript and XML, 即异步JavaScript和XML)以及其他功能。除此以外，jQuery提供API让开发者编写插件。其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

**jQuery包含以下特点：**

1. 动态特效

2. AJAX

3. 通过插件来扩展

4. 方便的工具 - 例如浏览器版本判断

5. 渐进增强

6. 链式调用

7. 多浏览器支持，支持Internet Explorer6.0+、Opera9.0+、Firefox2+、Safari2.0+、Chrome1.0+（在2.0.0中取消了对Internet Explorer6,7,8的支持）

-----------------百度百科

1. 系统设计
   1. 概要设计

在系统需求分析阶段，搞清楚了软件“做什么”的问题，形成了目标系统的逻辑模型。现在我们所要做的就是要把软件“做什么”的逻辑模型变换为“怎么做”的物理模型，即着手实现系统的需求。首先，我们需要描述的是系统的总的体系结构。

系统的概要设计中最重要的就是系统的模块化。模块化是指解决一个复杂问题时自项向下逐层把软件系统划分成若干个模块的过程。每个模块完成一个特定的功能，所有的模块按某种方法组织起来，成为一个整体，完成整个系统所要求的功能。

将系统划分为多个模块是为了降低软件系统的复杂性，提高可读性、可维护性，但模块的划分不能是任意的，应尽量保持其独立性。也就是说，每个模块只完成系统要求的独立的子功能，并且与其他模块的联系最少且接口简单，即尽量做到高内聚低耦合，提高模块的独立性，为设计高质量的软件结构奠定基础。

在系统的概要设计中采用结构化设计以需求分析阶段产生的数据流图为基础，按一定的步骤映射成软件结构。首先将整个系统化分为几个小问题，小模块，在系统中，设计了用户管理、好友管理、博客管理、博客分类等管理模块。然后，进一步细分模块，添加细节。

按照面向对象以及MVC编程的模式,将整个博客系统分为三个部分:

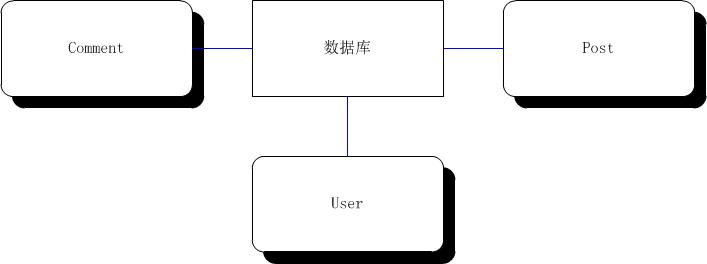
1. 模型(Models)部分
2. 视图(Views)部分
3. 控制器(Controls)部分
   1. 模型(Models)部分

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

1. 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.

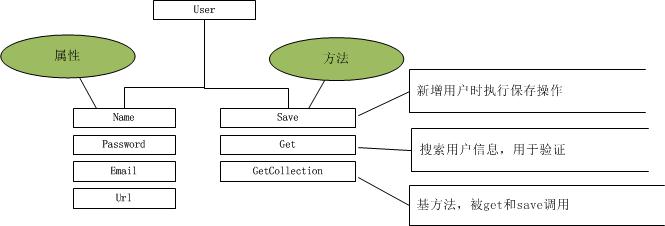
由于使用的是非关系型数据库，在前面的介绍中也已说过，非关系型数据库和关系型数据库的区别。因此不需要提前为每个部分设计好数据库，只需要在使用的过程中，将需要使用到的字段添加进来即可。

所以简单的将数据库分为三个部分



1. 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，主要操作对象为数据库中的User数据。

主要结构如下图所示



属性详解：

Name – 用户昵称

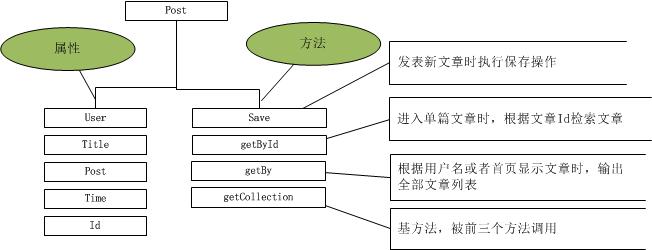
Password – 用户密码

Email – 用户邮箱

Url – 用户个人网站网址

1. models/post.js 文章处理类, 主要操作对象为数据库中的Post数据，利用db.js中的接口,负责
2. 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.
3. 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

主要结构如下图所示



属性详解：

User – 发表文章作者

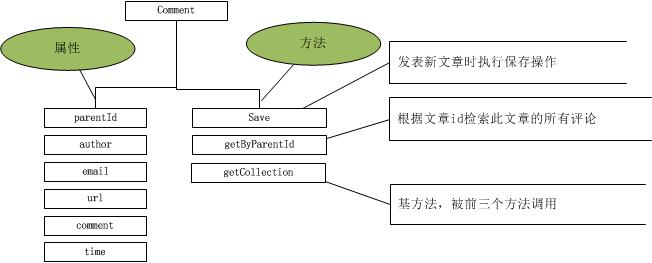
Title – 文章标题

Post – 文章内容

Title – 文章发表时间

Id – 文章唯一Id

1. models/comment.js评论处理类，主要操作对象为数据库中的comment数据。 负责
2. 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.
3. 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.



属性详解：

ParentId – 所在文章id

Author – 评论作者

Email – 评论者邮箱地址

url – 评论者网址

comment – 评论内容

time – 评论时间

* 1. 视图(Views)部分

文件位置 views/\*.ejs

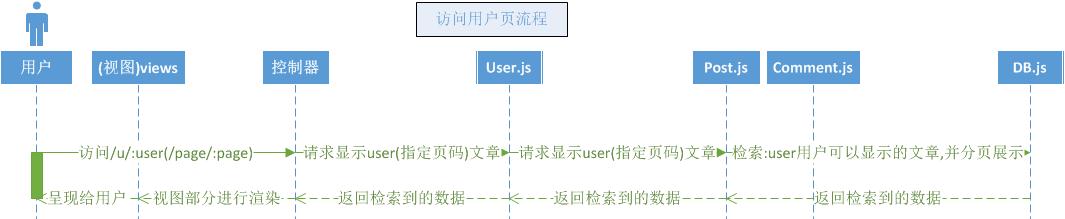
主要作用为，将models部分的处理程序从数据库中检索到的数据组装成HTML片段,然后发送给控制器,再经控制器发送到用户浏览器前端呈现给用户显示. 主要有以下几个部分

*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*

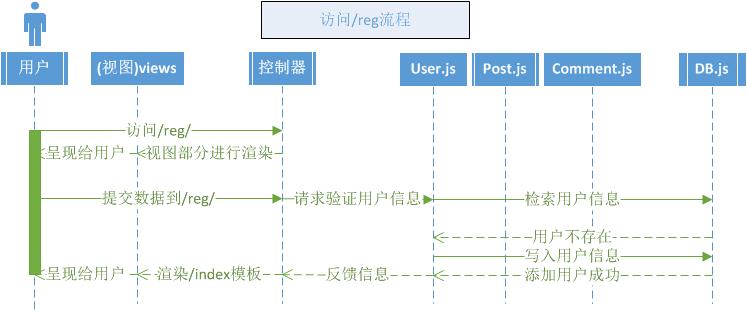
1. /index 负责访客直接请求首页时进行展示文章列表
2. /index/page/:page 负责当访客点击页码时展示请求页码的文章列表



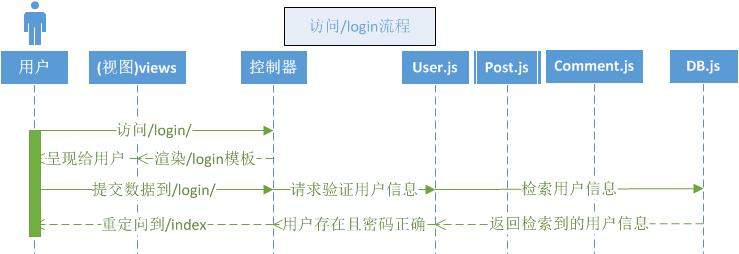
1. /u/:user 负责访客点击某个用户名称时进行展示该用户的文章列表
2. /u/:user/page/:page 负责当访客点击某个用户文章列表中的页码时展示该页码的文章列表



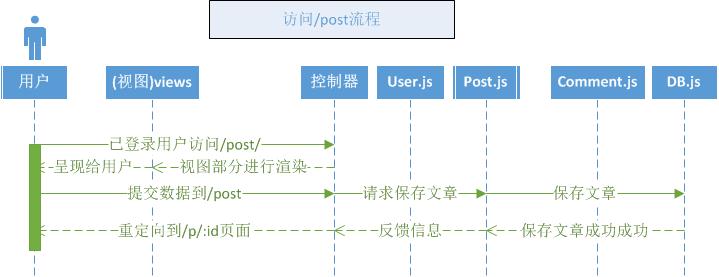
1. /reg GET方法请求该页面时负责展示注册页面, POST方法请求该页面时进行字段审核,成功后进行数据库写入操作,增加新用户.



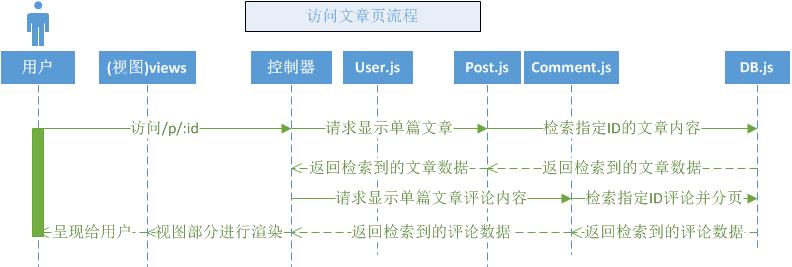
1. /login GET方法请求该页面时负责展示访客登录界面, POST方法请求该页面时进行用户鉴权操作,鉴权成功后重定向网页到/index 页面



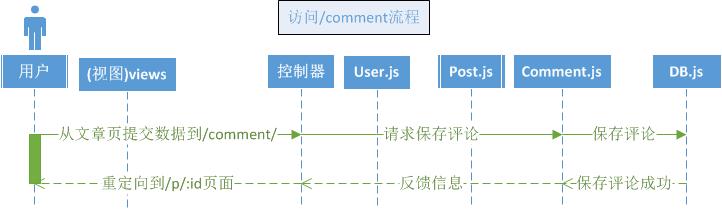
1. /logout 负责清除用户登录信息,清除成功后,引导访客重新到达/index页面
2. /post 当用户登录后,GET方法请求该页面时负责展示发表文章页面,让已登录用户进行发表文章操作,POST方法请求该页面时进行新文章的数据库写入操作,添加新文章



1. /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.



1. /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.



* 1. 控制器(Controls)部分

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

为什么要进行前端优化

---

随着WEB2.0时代的来临，给网络带来了空前的发展。前端用户体验变得越来越显的重要，各个互联网公司纷纷成立用户体验部,如腾讯的 用户研究与体验设计中心(腾讯CDC)、腾讯社交用户体验设计(腾讯ISUX)、腾讯电商用户体验设计部(腾讯ECD)、淘宝用户体验(淘宝UED)、支付宝用户体验部(支付宝UED)、网易用户体验设计中心(网易UEDC)、百度泛用户体验，从而来弥补B/S结构的用户交互型差的一些弊端。可是这样会带来一个问题就是会增加客户端的压力，比如大量运用JS代码。

JS代码是运行在客户端的，会影响到整个网页的在浏览器的解析效率，这样也可能暗示着会增加客户端的流量，所以不管是从服务器负载角度还是站在用户的角度来看，对客户端的代码进行优化都显得尤为重要！

为什么要从前端开始着手有三个主要原因：

1. 这里有提升和改进的潜力。如果能减少一半的体积，就能减少40%的响应时间。

2. 改进前端比改进后端需要的时间和资源更少。（改进后端要重新设计应用程序规划，代码，寻找优化代码的方法，添加或改变硬件配置，分布式数据库，等等）

3. 黄金规则是：首先优化前端表现，这些东西耗费了用户端响应时间中的80%。

常见的优化方法 :

- 把样式表放到顶部

我们发现把css放到文档头部会让网页加载得更快。因为这样可以让页面逐渐加载。把样式表放到接近底部的问题是它阻止了页面元素的逐渐显示。这样还会导致“flash of unstyled content“，即在样式表加载之前页面内容是以没有样式的形式显示出来的，待加载完样式后，页面重绘，内容一闪即改变了样式表现。

- 把脚本放到底部

把脚本放到尽可能底部的地方，一个原因是让页面逐渐渲染，另一个是实现更好的并行下载。对于脚本，脚本以下的内容被阻止逐渐加载了，因为只有当下载完脚本以后才会下载下面的内容，第二个脚本引起的问题是阻止平行下载。 “HTTP/1.1 specification”建议浏览器对一个域名， 同一时间下载数不超过2个（按：实际监测发现一般有超过2个），当脚本正在下载的时候，浏览器不会开始下载任何东西

- 避免css expressions

- 让脚本和样式外延

这里引出一个问题， Javascript和CSS应该是外部调用还是内嵌呢？

用外部调用文件的方式更快，因为他们是可以被缓存的，如果是内嵌在页面中他们就无法被缓存了！如果用户要在某个网站看很多很多的页面，如果都是使用同一个外部脚本和样式，那么他们一旦被缓存，就再也不需要下载了，这样会给用户带来很大的潜在好处。

- 减小脚本体积

有两个比较流行的工具是用来减小脚本的体积的–JSMin和YUI Compressor

优化方法，主要是对 js代码和CSS代码 还有图片 进行优化,辅助进行的是对服务器发来的html代码进行压缩,去掉空格 回车等等

Js代码优化:

1. 压缩js代码为一个文件,,减少http请求次数,去掉空格回车符

2. 服务器端Gzip压缩 减小传输体积

3. 服务器端对js文件加一个长时间过期的头部,便利客户端能够缓存js文件

4. 运用cdn（Content Delivery Network，即内容分发网络）技术,提高客户端js文件加载速度

Css代码优化

1. 压缩css代码,去掉空格回车符

2. CSS代码简写。这里说的简写主要解决的问题是，网站代码冗余。如果可以用一句表达的语言，就不要多出几行代码，符合W3C标准。

3. 同属性提出

如果2个样式带有同样的属性，可以把同属性单独提出处理。例如A样式和B样式的字体颜色、大小、边框都一样，那么就可以考虑提出。

图片优化

1. 压缩图片体积

2. css图片使用CSS Sprit技术减少服务器请求次数，将多个图片放到一张大图片中，在代码编写过程中，使用CSS控制图片位置来显示不同的内容。

3. 尽量少使用大图片,防止网络原因造成效果显示缓慢,影响用户体验