logo.png

ajax

H5 web前端

[www.bufanui.com](http://www.bufanui.com)

# 一、BS/CS

## 1.1 C/S架构

Client/Server这两个单词的首字母,指的是客户端,服务器架构的意思,很多常见的软件都是这种架构。

* **解释**

对于C/S架构,最为常见的例子就是网络游戏,比如LOL,WOW如果不联网无法使用,你在软件内所做的所有操作通过互联网能够传递到其他的的玩家身上

* **优点:**
  + 性能较高:可以将一部分的计算工作放在客户端上,这样服务器只需要处理数据即可
  + 界面酷炫: 只要客户端的显卡足够用，那么就可以在客户端进行高质量的画面渲染。
* **缺点:**
  + **更新软件:**如果推出了新版本,不更新客户端无法登陆使用(一部分)
  + **不同设备访问:**如果使用其他的电脑,没有安装客户端的话就无法登陆软件(比如收发邮件)
  + **安全性：**由于客户端安装在本地，很容易通过外挂修改内存数据。

## 1.2 B/S架构

Browser/Server的这两个单词的首字母,指的是浏览器,服务器,是WEB兴起之后的一种架构

当前主流架构，也可能是以后的一种趋势。

* **解释**

现在所有的网站都是B/S架构,较为常见的例子有网站、后台管理系统、移动缴费系统等等,所有只需要通过浏览器即可使用.

* **优点**
  + 更新简洁:如果需要更新内容了,对开发人员而言需要更改服务器的内容,但是对用户而言只需要刷新浏览器即可
  + 多设备同步:所有数据都在网上,只要能够使用浏览器即可登录使用
* **缺点:**
  + 性能较低:相比于客户端应用性能较低,但是随着硬件性能的提升,这个差距在缩小
  + 浏览器兼容:处理低版本的浏览器显示问题一直是前端开发人员头痛的问题之一,移动设备兼容性较好,ie6已经没人用了（政府单位除外）。

# 二、服务器

* **什么是服务器？**
* **服务器分类**
  + 按类型分:
    - 文件服务器、数据库服务器、（阿里oss存储）邮件服务器、Web 服务器等；
  + 按软件分:
    - Apache服务器、Nginx 服务器、IIS服务器、Tomcat服务器、Node服务器等。

以后工作中都会遇到

Apache 🡺 php

Nginx🡺目前最火的服务器

IIS 🡺微软 asp.net

Tomcat🡺java

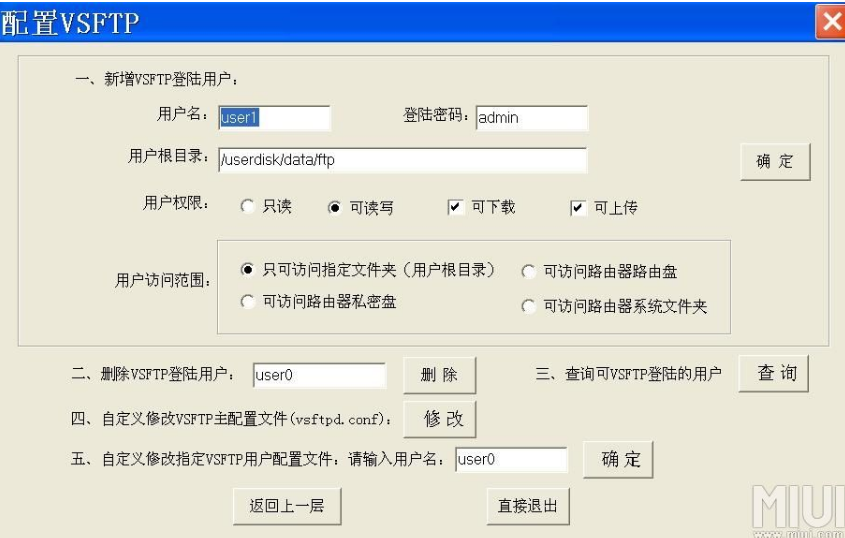
Node🡺node.js的服务

* + 按操作系统分:
    - Linux服务器、Windows服务器等、mac操作系统、ubantu 、麒麟。。。

### 服务器软件

提供了某种服务的计算器,我们称之为服务器,那么这些赋予计算器各种服务功能的软件主要有哪一些呢?

* 常见的服务器软件有：
  + 文件服务器：Server-U、FileZilla、VsFTP等；

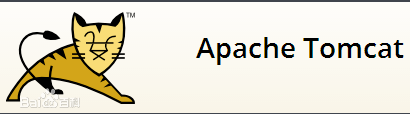


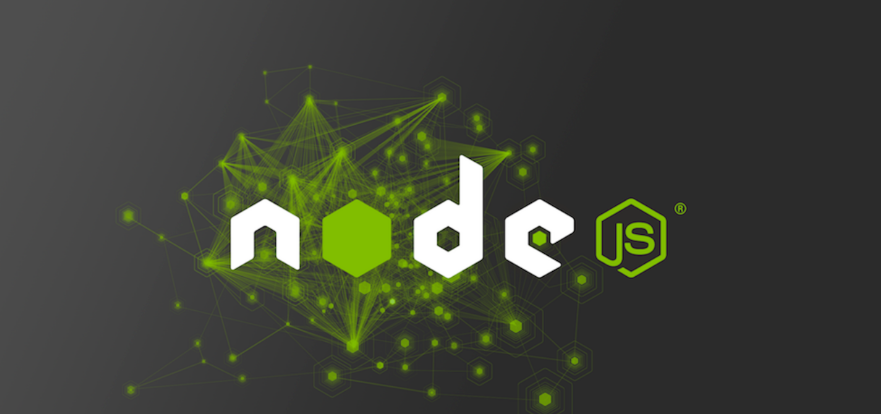
* + 数据库服务器：Oracle、MySQL、mongodb、sqlserver等；



**关系型数据库和非关系型数据库**

* + 邮件服务器：Postfix、Sendmail等；
  + HTTP 服务器：Apache、Nginx、IIS、Tomcat、NodeJS等； 这里我们主要介绍HTTP服务器



**Apache HTTP [Server](https://baike.baidu.com/item/Server" \t "_blank)**（简称**[Apache](https://baike.baidu.com/item/Apache/6265" \t "_blank)**）是[Apache软件基金会](https://baike.baidu.com/item/Apache%E8%BD%AF%E4%BB%B6%E5%9F%BA%E9%87%91%E4%BC%9A" \t "_blank)的一个开放源码的网页服务器， Apache是世界使用排名第一的Web[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)软件。它可以运行在几乎所有广泛使用的[计算机平台](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%B9%B3%E5%8F%B0" \t "_blank)上，由于其[跨平台](https://baike.baidu.com/item/%E8%B7%A8%E5%B9%B3%E5%8F%B0" \t "_blank)和安全性被广泛使用，是最流行的Web服务器端软件之一。

**Nginx**是一款[轻量级](https://baike.baidu.com/item/%E8%BD%BB%E9%87%8F%E7%BA%A7/10002835" \t "_blank)的[Web](https://baike.baidu.com/item/Web/150564" \t "_blank) 服务器/[反向代理](https://baike.baidu.com/item/%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86" \t "_blank)服务器及[电子邮件](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6" \t "_blank)（IMAP/POP3）代理服务器，并在一个BSD-like 协议下发行。其特点是占有内存少，[并发](https://baike.baidu.com/item/%E5%B9%B6%E5%8F%91" \t "_blank)能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、[京东](https://baike.baidu.com/item/%E4%BA%AC%E4%B8%9C" \t "_blank)、[新浪](https://baike.baidu.com/item/%E6%96%B0%E6%B5%AA" \t "_blank)、[网易](https://baike.baidu.com/item/%E7%BD%91%E6%98%93" \t "_blank)、[腾讯](https://baike.baidu.com/item/%E8%85%BE%E8%AE%AF" \t "_blank)、[淘宝](https://baike.baidu.com/item/%E6%B7%98%E5%AE%9D" \t "_blank)、不凡等。

**iis**是Internet Information Services的[缩写](https://baike.baidu.com/item/%E7%BC%A9%E5%86%99" \t "_blank)，意为[互联网](https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91/199186" \t "_blank)信息服务，是由微软公司提供的基于运行Microsoft Windows的互联网基本服务。最初是Windows NT版本的可选包，随后内置在Windows 2000、Windows XP Professional和Windows Server 2003一起发行，但在Windows XP Home版本上并没有IIS。IIS是一种Web（网页）服务组件，其中包括Web服务器、FTP服务器、NNTP服务器和SMTP服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送等方面，它使得在网络（包括互联网和局域网）上发布信息成了一件很容易的事。

**Tomcat**是Apache 软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由[Apache](https://baike.baidu.com/item/Apache/6265" \t "_blank)、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5支持最新的Servlet 2.4 和JSP 2.0 规范。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

**Node.js**是一个Javascript运行环境(runtime)，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装。Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。

V8引擎执行Javascript的速度非常快，性能非常好。[1]  Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用[事件驱动](https://baike.baidu.com/item/%E4%BA%8B%E4%BB%B6%E9%A9%B1%E5%8A%A8" \t "_blank)， 非阻塞[I/O](https://baike.baidu.com/item/I%2FO/84718" \t "_blank) 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。

# 三、HTTP协议

## 3.1 什么是Http协议

### 超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。

## 3.2 TCP/IP HTTP的区别

**TPC/IP协议是传输层协议，主要解决数据如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。**关于TCP/IP和HTTP协议的关系，网络有一段比较容易理解的介绍：“**我们在传输数据时，可以只使用（传输层）TCP/IP协议，但是那样的话，如果没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用到应用层协议，应用层协议有很多，比如HTTP、FTP、TELNET等，也可以自己定义应用层协议。WEB使用HTTP协议作应用层协议，以封装HTTP 文本信息，然后使用TCP/IP做传输层协议将它发到网络上。**”

　　术语TCP/IP代表传输控制协议/网际协议，指的是一系列协议。“IP”代表网际协议，TCP和UDP使用该协议从一个网络传送数据包到另一个网络。把**IP想像成一种高速公路**，它允许其它协议在上面行驶并找到到其它电脑的出口。**TCP和UDP是高速公路上的“卡车”，它们携带的货物就是像HTTP**，文件传输协议FTP这样的协议等。

    你应该能理解，TCP和UDP是FTP，HTTP和SMTP之类使用的传输层协议。虽然TCP和UDP都是用来传输其他协议的，它们却有一个显著的不同：TCP提供有保证的数据传输，而UDP不提供。这意味着TCP有一个特殊的机制来确保数据安全的不出错的从一个端点传到另一个端点，而UDP不提供任何这样的保证。

    HTTP(超文本传输协议)是利用TCP在两台电脑(通常是Web服务器和客户端)之间传输信息的协议。客户端使用Web浏览器发起HTTP请求给Web服务器，Web服务器发送被请求的信息给客户端。

　　下面的图表试图显示不同的TCP/IP和其他的协议在最初OSI(国际标准化组织制定的不同计算机互联的标准，是设计和描述计算机通信的基本模型)模型中的位置：

|  |  |  |
| --- | --- | --- |
| 7 | **应用层** | 例如[HTTP](http://zh.wikipedia.org/wiki/HTTP" \o "HTTP)、[SMTP](http://zh.wikipedia.org/wiki/SMTP" \o "SMTP)、[SNMP](http://zh.wikipedia.org/wiki/SNMP" \o "SNMP)、[FTP](http://zh.wikipedia.org/wiki/FTP" \o "FTP)、[Telnet](http://zh.wikipedia.org/wiki/Telnet" \o "Telnet)、[SIP](http://zh.wikipedia.org/wiki/SIP" \o "SIP)、[SSH](http://zh.wikipedia.org/wiki/SSH" \o "SSH)、[NFS](http://zh.wikipedia.org/wiki/NFS" \o "NFS)、[RTSP](http://zh.wikipedia.org/wiki/RTSP" \o "RTSP)、[XMPP](http://zh.wikipedia.org/wiki/XMPP" \o "XMPP)、[Whois](http://zh.wikipedia.org/wiki/Whois" \o "Whois)、[ENRP](http://zh.wikipedia.org/w/index.php?title=ENRP&action=edit&redlink=1" \o "ENRP) |
| 6 | **表示层** | 例如[XDR](http://zh.wikipedia.org/w/index.php?title=External_Data_Representation&action=edit&redlink=1" \o "External Data Representation)、[ASN.1](http://zh.wikipedia.org/w/index.php?title=Abstract_Syntax_Notation_1&action=edit&redlink=1" \o "Abstract Syntax Notation 1)、[SMB](http://zh.wikipedia.org/w/index.php?title=Server_message_block&action=edit&redlink=1" \o "Server message block)、[AFP](http://zh.wikipedia.org/w/index.php?title=Apple_Filing_Protocol&action=edit&redlink=1" \o "Apple Filing Protocol)、[NCP](http://zh.wikipedia.org/w/index.php?title=NetWare_Core_Protocol&action=edit&redlink=1" \o "NetWare Core Protocol) |
| 5 | **会话层** | 例如[ASAP](http://zh.wikipedia.org/w/index.php?title=Aggregate_Server_Access_Protocol&action=edit&redlink=1" \o "Aggregate Server Access Protocol)、[TLS](http://zh.wikipedia.org/wiki/Transport_Layer_Security" \o "Transport Layer Security)、[SSH](http://zh.wikipedia.org/wiki/SSH" \o "SSH)、ISO 8327 / CCITT X.225、[RPC](http://zh.wikipedia.org/w/index.php?title=Remote_procedure_call&action=edit&redlink=1" \o "Remote procedure call)、[NetBIOS](http://zh.wikipedia.org/w/index.php?title=NetBIOS&action=edit&redlink=1" \o "NetBIOS)、[ASP](http://zh.wikipedia.org/w/index.php?title=AppleTalk&action=edit&redlink=1" \o "AppleTalk)、[Winsock](http://zh.wikipedia.org/w/index.php?title=Winsock&action=edit&redlink=1" \o "Winsock)、[BSD sockets](http://zh.wikipedia.org/wiki/Berkeley_sockets" \o "Berkeley sockets) |
| 4 | **传输层** | 例如[TCP](http://zh.wikipedia.org/wiki/TCP" \o "TCP)、[UDP](http://zh.wikipedia.org/wiki/User_Datagram_Protocol" \o "User Datagram Protocol)、[RTP](http://zh.wikipedia.org/w/index.php?title=Real-time_Transport_Protocol&action=edit&redlink=1" \o "Real-time Transport Protocol)、[SCTP](http://zh.wikipedia.org/w/index.php?title=Stream_Control_Transmission_Protocol&action=edit&redlink=1" \o "Stream Control Transmission Protocol)、[SPX](http://zh.wikipedia.org/w/index.php?title=Sequenced_packet_exchange&action=edit&redlink=1" \o "Sequenced packet exchange)、[ATP](http://zh.wikipedia.org/w/index.php?title=AppleTalk&action=edit&redlink=1" \o "AppleTalk)、[IL](http://zh.wikipedia.org/w/index.php?title=IL_Protocol&action=edit&redlink=1" \o "IL Protocol) |
| 3 | **网络层** | 例如[IP](http://zh.wikipedia.org/wiki/%E7%BD%91%E9%99%85%E5%8D%8F%E8%AE%AE" \o "网际协议)、[ICMP](http://zh.wikipedia.org/wiki/ICMP" \o "ICMP)、[IGMP](http://zh.wikipedia.org/wiki/IGMP" \o "IGMP)、[IPX](http://zh.wikipedia.org/wiki/IPX" \o "IPX)、[BGP](http://zh.wikipedia.org/wiki/BGP" \o "BGP)、[OSPF](http://zh.wikipedia.org/wiki/OSPF" \o "OSPF)、[RIP](http://zh.wikipedia.org/wiki/RIP" \o "RIP)、[IGRP](http://zh.wikipedia.org/wiki/IGRP" \o "IGRP)、[EIGRP](http://zh.wikipedia.org/wiki/EIGRP" \o "EIGRP)、[ARP](http://zh.wikipedia.org/wiki/ARP" \o "ARP)、[RARP](http://zh.wikipedia.org/wiki/RARP" \o "RARP)、 [X.25](http://zh.wikipedia.org/wiki/X.25" \o "X.25) |
| 2 | **数据链路层** | 例如[以太网](http://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E7%BD%91" \o "以太网)、[令牌环](http://zh.wikipedia.org/wiki/%E4%BB%A4%E7%89%8C%E7%8E%AF" \o "令牌环)、[HDLC](http://zh.wikipedia.org/wiki/HDLC" \o "HDLC)、[帧中继](http://zh.wikipedia.org/wiki/%E5%B8%A7%E4%B8%AD%E7%BB%A7" \o "帧中继)、[ISDN](http://zh.wikipedia.org/wiki/ISDN" \o "ISDN)、[ATM](http://zh.wikipedia.org/wiki/%E5%BC%82%E6%AD%A5%E4%BC%A0%E8%BE%93%E6%A8%A1%E5%BC%8F" \o "异步传输模式)、[IEEE 802.11](http://zh.wikipedia.org/wiki/IEEE_802.11" \o "IEEE 802.11)、[FDDI](http://zh.wikipedia.org/wiki/FDDI" \o "FDDI)、[PPP](http://zh.wikipedia.org/wiki/PPP" \o "PPP) |
| 1 | **物理层** | 例如[线路](http://zh.wikipedia.org/w/index.php?title=%E7%BA%BF%E8%B7%AF&action=edit&redlink=1" \o "线路)、[无线电](http://zh.wikipedia.org/wiki/%E6%97%A0%E7%BA%BF%E7%94%B5" \o "无线电)、[光纤](http://zh.wikipedia.org/wiki/%E5%85%89%E7%BA%A4" \o "光纤) |

## 3.3 request/Response

Request 和 Response 对象起到了服务器与客户机之间的信息传递作用。Request 对象用于接收客户端浏览器提交的数据，而 Response 对象的功能则是将服务器端的数据发送到客户端浏览器。

一、Request对象的五个集合：

QueryString：用以获取客户端附在url地址后的查询字符串中的信息。

例如：stra=Request.QueryString ["strUserld"]

Form：用以获取客户端在FORM表单中所输入的信息。（表单的method属性值需要为POST）

例如：stra=Request.Form["strUserld"]

Cookies：用以获取客户端的Cookie信息。 每一次请求 request对象都会带上cookies

例如：stra=Request.Cookies["strUserld"]

ServerVariables：用以获取客户端发出的HTTP请求信息中的头信息及服务器端环境变量信息。

例如：stra=Request.ServerVariables["REMOTE\_ADDR"],返回客户端IP地址

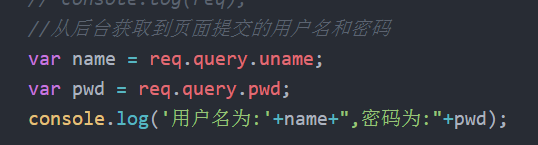
ClientCertificate：用以获取客户端的身份验证信息

例如：stra=Request.ClientCertificate["VALIDFORM"],对于要求安全验证的网站，返回有效起始日期。

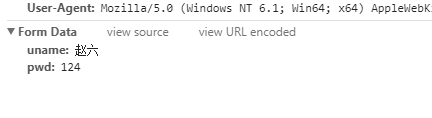
**案例：**get方式 提交

Get方式 数据从QueryString 中获取

### req.query 取地址栏参数



post方式 提交



## 3.3 404与500

404 文件没找到 请求没找到

500 服务器代码出错 常见的是模板中变量不存在

200

二、Response对象

Response对象用于动态响应客户端请示，控制发送给用户的信息，并将动态生成响应。Response对象提供了一个数据集合cookie，它用于在客户端写入cookie值。若指定的cookie不存在，则创建它。若存在，则将自动进行更新。结果返回给客户端浏览器。

语法格式：Response.Cookies(CookieName)[(key)|.attribute]=value。这里的CookiesName是指定的Cookie的名称，如果指定了Key，则该Cookie就是一个字典，Attribute属性包括Domain，Expires，HasKeys，Path，Secure。

response的方法:

Write：向客户端发送浏览器能够处理的各种数据,包括:html代码,脚本程序等。

Redirect：response.redirect("url")的作用是在服务器端重定向于另一个网页。

End：用来终止脚本程序。

Clear：要说到Clear方法，就必须提到response的Buffer属性，Buffer属性用来设置服务器端是否将页面先输出到缓冲区。语法为：Response.Buffer=True/False

Flush：当Buffer的值为True时，Flush方法用于将缓冲区中的当前页面内容立刻输出到客户端。

**案例：**使用res.send 向客户端输出html

使用res.render 向客户端输出内容

# 四、数据库(DBA数据库管理员)

试想一下,如果一个网站的注册用户有1千万个那么应该使用什么格式,什么方式来保存这些数据呢?

* **保存的格式**: 这个很简单:比如 用户名,密码,性别,出生,爱好,头像 等等 这样的数据排列方式,每一个用户就是一行
* **常规保存方式**:
  + .txt文件:因为内容是文本,所以直接使用文本文件保存肯定可以,但是当一个文本文件很大很大时,打开是异常缓慢的
  + .excel文件:可以用来进行数据的统计,分析等操作,但是当很多个人需要访问同一个.excel文件时,性能也很差
  + 当然也可能想到一些其他的保存方式,但如果出现被删除,被胡乱修改所带来的后果都是灾难性的
* **数据库**:
  + 按照数据结构来组织,存储和管理数据的仓库,软件开发行业一般指的是数据库软件,常见的有:Oracl(甲骨文)e、MySQL、MSSQL等

Oracle 甲骨文 （收费的，而且非常贵！）

适用大型企业

收购了sun🡺 java

Mysql 免费的

一般适用于中小企业

性能比不上oracle

阿里 在mysql的基础上改版 达到现在的应用级别，而且准备开源免费使用

能够支持支付宝高频操作和双11高并发。

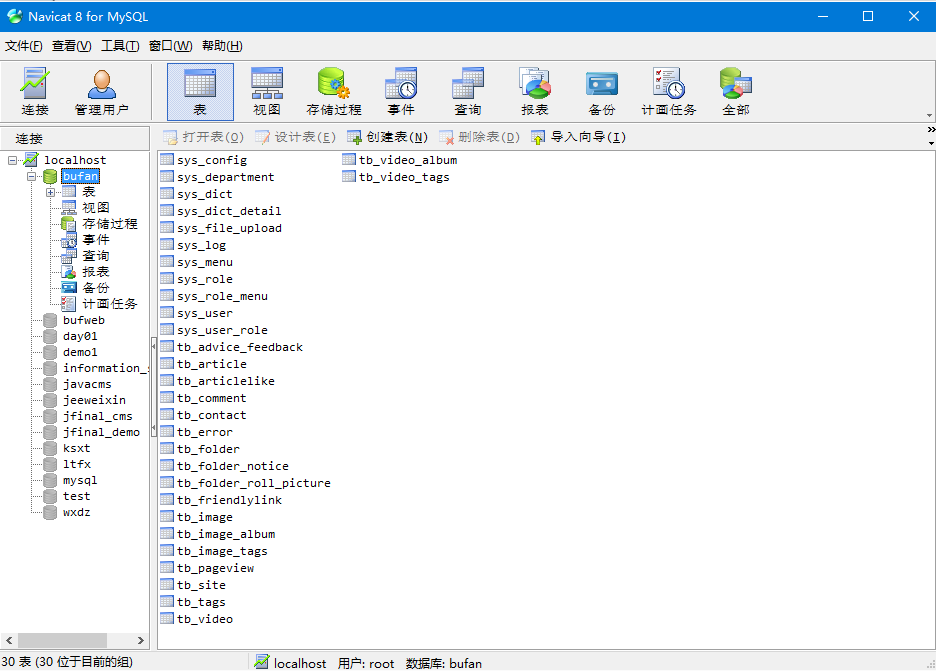
* + 特点:
    - 数据共享:多用户同时访问数据的稳定性
    - 故障恢复:数据库软件提供了一套的方法,可以用来发现错误并且修复错误
    - 减少数据冗余:由于大家都可以使用同一套数据,没有必要重复创建了
* **DBA数据库管理员**: db 🡺 database dba 🡺 database administrator
  + 从事管理和维护数据库管理系统(DBMS)的相关工作人员的统称，他的工作目的是:保证数据库的稳定性,安全性,完整性和高性能.

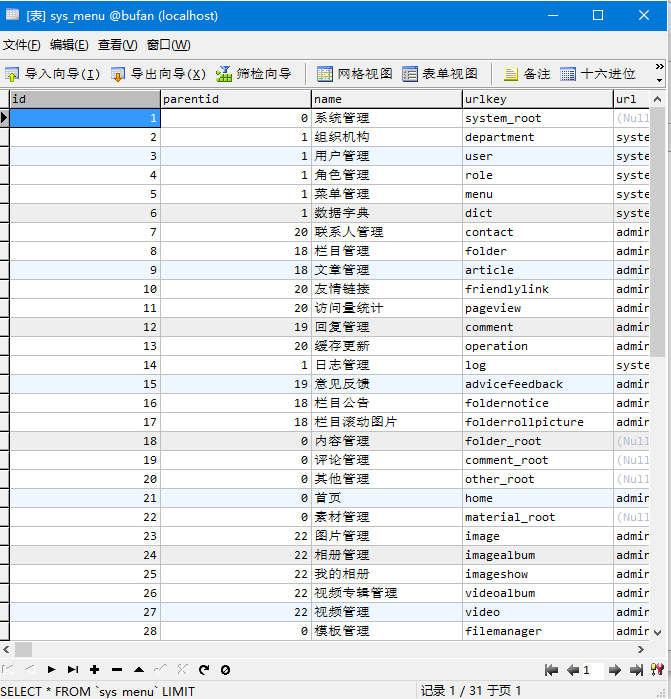
数据库的结构：

Bufan(bufanui网站的数据库)

50张表组成，每张表保存一类数据。

一张表可以保存任意多条数据，每条数据由多个字段组成，至少要有一个id标明该数据的唯一性。





# 五、node基本原理

## 5.1 什么是node.js

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。

Node.js 的包管理器 npm (node package manager)，是全球最大的开源库生态系统。

## 5.2 使用node执行js文件

## 5.3 使用node开启http服务

|  |
| --- |
| var http = require('http');  var fs = require('fs');  var url = require('url');  http.createServer(function(request,response){  // 解析请求，包括文件名  var pathname = url.parse(request.url).pathname;  response.writeHead(200, {'Content-Type': 'text/html;charset=utf-8'});  fs.readFile('demo.html',function(err,data){  var rs = data.toString();  response.write(rs);  response.end();  })    }).listen(8080,function(){  console.info('server is start in 8080...');  }); |

## 5.4 模板原理

|  |
| --- |
| // 思考 ：如何把相应的符号解析出来并替换为相应的内容？  var reg = /<%=[^%>]+%>/g; // .是任意字符 能包含 %>的组合吗？如果包含 <%=xx%>xxxx%>  // 共找到 3 处匹配：  // <%=uname%>  // <%= day %>  // <%= weather%>  // replace(reg,fn) 这个方法 可以通过匹配正则来替换  var res = str.replace(reg,function($1){  // 解析 匹配到的内容 <%=uname%> ==> uname;  var reg2 = /^(<%=\s\*)|(\s\*%>)/g;  var key = $1.replace(reg2,'');  var val = data[1][key];  console.log("匹配到的内容为："+$1);  console.log("替换结果为："+key);  // 返回值 会把匹配到的内容替换  return val;  }) |

## 5.5 增加路由

# 六、node.js+express

## 6.1 express

Express 基于 Node.js 平台，快速、开放、极简的 web 开发框架。

根据链接创建express 项目 <http://www.expressjs.com.cn/starter/generator.html>

* + - 1. 安装node.js
      2. Express 是node.js的一个框架

作用：全局安装 express生成器 不要忘了-g

* $ npm install express-generator -g

作用：创建自己的项目名文件夹,也可以手动创建 myapp文件夹

* $ mkdir myapp

进入myapp文件夹,在myapp文件夹执行express -e

作用：通过修改配置 使express支持 ejs模板(跟html一样)，默认支持 jade

$ express -e

作用：在项目文件夹执行命令，安装package.json中配置的依赖库

$ npm install

作用：开启服务

$ npm start

最后通过浏览器 访问 localhost:3000 可以看到首页内容，

修改index.ejs 可以修改首页内容

项目发布后访问的几种形式

1. 访问 localhost:8080/index.html 🡺 可以认为服务器发布了一个index.html的静态网站

静态网站：比如之前用 node.js 的http-server 发布的内容就是静态网站，这种网站不需要数据库，因为没有前后台数据交互，只有html+css+js

2. 访问 localhost:8080/index || …/index.php || …/index.jsp || …/index.asp

|| …/index.action || …/index.do || …/index.htm || …/index.html

动态网站：发送的请求xxx.do/action/asp…都是一个url请求，没有直接指向静态资源（实际上并不代表服务器上的某个文件），当服务器接收到该请求之后，根据之前定义的需求，从数据库获取相应的数据并组装，返回给客户端。

3. 所有的b/s架构 都是这样的

.php 指的就是php网站

.asp asp.net

.jsp .action .do .htm 一般指的都是java的

特殊 .html 🡺 伪装url，把url设定为html结尾，给用户一种直接访问网页的感觉。

已经分清楚了 动态网站和静态网站的区别，到工作中一般需要做的是

根据ui设计师的设计稿 🡺 xx.html

后端程序员(node.js)用 xx.html🡺 xx.ejs

Php xx.html 🡺 xx.php

Java xx.html 🡺 xx.jsp

.Ejs .jade . php .jsp 都是模板，用户动态的替换部分内容，并生成html返回给客户端。

**Express 目录结构**

项目名

bin

www --- 创建http服务，并设置端口

**Public --- 用于发布静态资源 ，如果使用html,../public**

**Img 项目中直接用/img，项目发布后页面的请求都是相对路径**

**Css**

**Js**

**lib**

**Routes --- 路由，拦截客户端的url请求，并根据请求找相应的处理逻辑。**

**路由 url ‘/’ ‘/test’ ‘/login’ ‘/doLogin’**

**Index.js --- 在app.js中设置路由的命名空间**

**User.js**

**Views --- 前端模板**

**Index.ejs**

**error.ejs**

**App.js --- 1.引入项目必须的模块 2.定义项目的路由 3.设置项目的模**

**Node\_modules --- 项目的依赖库，该项目启动 必须安装的第三方库。根据package.json安装的库。**

**板类型（ejs/jade）**

**Package.json --- 项目依赖的库及项目基本信息的配置文件，当在项目目录执行npm install的时候 ，npm会解析该目录，并 从网上安装所有项目依赖的中间件。**

案例：用户登录注册

* + - 1. 注册页面的完成

注册的用户信息 包含用户名 密码 电话 性别 头像保存

头像选择

成功之后跳转登录页面

* + - 1. 登录页面

要求实现记住密码功能，如果选择记住密码，下次进入当前页面不用重新输入密码，直接把值设置在表单内容

技术： localStroage

如果有能力，要求实现图片验证功能。

* + - 1. 登录后台要求进行用户名和密码验证。
      2. 登录成功后跳转成功页面，显示个人所有信息。
      3. 失败跳转失败页面重新登录。