网站与HTTP(RFC2616)

概述

当今为互联网/移动互联的时代，信息更新，咨询获取频繁，HTTP协议基于TCP/IP层之上，它成为现今最为广泛使用的互联网协议。它的标准约定在RFC2616中详细被描述。今天我们试着对HTTP/1.1简单了解，以此来提高对网站的开发。



对于HTTP我们首先明确的明确是，它是基于应用层的，在网络层传输层之上，我们通过网站的一个访问可以方便的获取到它的网络数据包：

telnet [www.baidu.com](http://www.baidu.com) 80

GET / HTTP/1.1

Accept:text/html, application/xhtml+xml, \*/\*

Accept-Language:zh-CN

User-Agent:Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)

Accept-Encoding:gzip, deflate

Host:www.baidu.com

Connection:Keep-Alive

Cookie:Hm\_lvt\_9f14aaa038bbba8b12ec2a4a3e51d254=1318342563718; BAIDUID=212DF6E1ED9CC1569D6297E583ECC9A6:FG=1; SFSSID=a4c52b0b6b299e0f240b4bd22415b297; SIGNIN\_UC=70a2711cf1d3d9b1a82d2f87d633bd8a00709070422

HTTP/1.1 200 OK

Date: Fri, 11 Nov 2011 06:41:35 GMT

Server: BWS/1.0

Content-Length: 6816

Content-Type: text/html;charset=gb2312

Cache-Control: private

Expires: Fri, 11 Nov 2011 06:41:35 GMT

Set-Cookie: BAIDUID=B13B517DFC9D8F82426466475BC48016:FG=1; expires=Fri, 11-Nov-41 06:41:35 GMT; path=/; domain=.baidu.com

P3P: CP=" OTI DSP COR IVA OUR IND COM "

Connection: Keep-Alive

….正文略….

我们可以通过telnet来测试网络应用，书写我们的请求之后，我们最后空行(两个回车换行\r\n\r\n)，空行之后HTTP的请求正常发出。然后收到baidu的一个HTTP数据返回，返回的数据，分为头和正文，也是通过\r\n\r\n作为区分。我们也可以看到，百度和GOOGLE之类的公司，我们的核心HTTP服务都是自己写的，没有直接用APACHE，NGINX，百度如果我们用POST提交它就会说：

HTTP/1.1 501 Not Implemented

Date: Fri, 11 Nov 2011 07:30:32 GMT

Server: BWS/1.0

Content-Length: 278

Content-Type: text/html

Cache-Control: private

Connection: Keep-Alive

<HTML><HEAD><TITLE>501 The requested method is not implemented by this server.

</TITLE></HEAD>

<BODY BGCOLOR="#cc9999" TEXT="#000000" LINK="#2020ff" VLINK="#4040cc">

<H2>501 The requested method is not implemented by this server.

</H2>

Not Implemented<HR>

</BODY>

</HTML>

HTTP请求方法

URL全称为资源描述符，一个URL它描述了网站上的一个资源．而HTTP中的请求方法有四种：GET，POST， PUT，DELETE，分别对应了，对这个资源的查，改，增，删．同时还包含了HEAD，CONNECT，TRACE，OPTIONS的请求。

我们平时最常用的还是GET，POST，所以我们需要对它有清晰的认识，通过它们之间的对比我们来进行认识。

本质上GET是用来获取资源，POST用于提交资料以完成添加修改等功能，由于本质上的区分，所以在HTTP的设计中，会加入一些特性，

GET的时候数据直接在附加在URL中提交到服务器，POST提交的数据放置在HTTP的正文部分。优劣对比：

GET方式提交长度受URL的长度限制，在IE的地址栏中，最大是2083字节的长度，也就是一次提交的数据最大不能超过2083字节，POST则不受该限制，理论上是任意长度都可以。这里的优劣比较明显，如果我们平台做黑名单处理，一次可以处理５万以上的黑名单，如果通过GET来实现是不可以的。

安全性上，GET更差，如果我们在登陆页面直接用GET提交(Get是Form的默认方法)，该用户名的密码就直接在URL地址上保留了，如果其他查看你的浏览历史记录就可以轻易的把密码拿到。当然POST也仅仅是相对的安全，只是规避了上面这个小漏洞，如果要真正实现安全，有两个措施可以去考虑，一个是将密码框用自定义的控件取代，另外就是使用HTTPS协议。我们会发现银行网站通常会采纳上述的方法。

GET在设计有缓存，POST没有缓存一说，我们可以看到在我们访问一些静态资源的时候：如图片，JS，CSS等，所有的请求都是以GET方式实现。如果想让你的访问更快，流量更小，静态资源一定要求是GET方式来请求。

POST vs. GET

|---> GET --> ETag/Expire -> Cache

|

`---> POST🡪NoCache

其他：

AJAX是一种异步的请求方式，如果使用GET它能快速响应，如果采用POST，它将先提交header出去，然后再提交post data，分成了两步来实现。我这里没有考证，大家有空可以测试一下。不过我们还是推荐在AJAX的时候使用GET，因为它有缓存机制，POST没有缓存机制。

<form action=”test.php?aa=tt”>….</form>这里注意一下，默认是GET方式提交，通过GET提交之后，这里的aa参数是不会被记录的，在form提交的时候直接给抛弃了，URL地址栏中不会出现aa=tt的参数，所以建议使用POST，或者使用<input type=’hidden’>来实现这个功能。

HTTP状态码

HTTP状态码(Http Status Code)是一个３数字组成的响应状态码。通常是1xx，2xx，3xx，4xx，5xx，分别是1,2,3,4,5打头的数字。如下列出了一些我们常见的HTTP状态码：

1xx消息

2xx定义为成功

200成功访问

3xx定义为重定向：

302为重定向到其他网页

304未修改，重定向到缓存

4xx请求错误

400参数错误或者语义错误

403被拒绝的请求

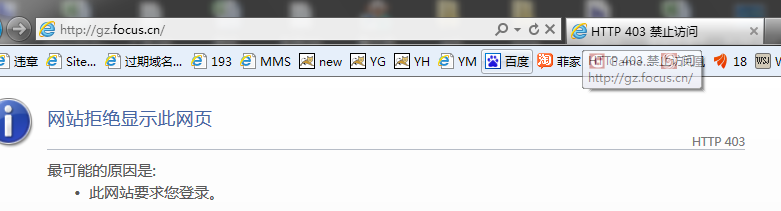
404请求的文件不存在

5xx服务器错误

501服务器没有实现，未被支持

503服务器过载或者在维护，前段使用QQ邮箱，就老是提示503。

505HTTP版本不支持



Http Head

Host

要请求资源目的服务器的主机和端口。HTTP/1.1请求必须包含Host，我们在与渠道进行数据交互的时候，我们自己实现的httpclient如果没有加Host有些服务器，有些服务器却不行就是这个原因，否则系统会以400状态码返回。400: Bad Request

Last-Modified/Expires/Cache-Control

基于7层的缓存或者代理如：varnish/haproxy，如果有cookies则不会缓存．

服务器很多的静态资源，如flash，图片以及javascript等，这些静态资源它是经常不会发生变化，但是它的流量经常占用了一个网页90%以上的流量，为让响应更快，流量更小，我们通常会对这部分静态资源进行缓存，在下次请求的时候只要做最后修改时间的判断，如果最后修改时间没有变，就直接提取缓存中的内容，如果发生变化再去服务器提取。

Expires、Cache-Control、Last-Modified、ETag是RFC 2616（HTTP/1.1）协议中和网页缓存相关的几个字段，前两个用来控制缓存的失效日期，后两个用来验证网页的有效性。

缓存的时效我们通过Expires/Cache-Control来控制，到指定时间这个缓存将自动失效，重新去服务器请求，而请求要用到的最后修改时间我们通过Last-Modified来控制。

Expires是HTTP/1.0的缓存头， Cache-Control: max-age 是HTTP/1.1是用来进行HTTP缓存的头。Expires指定了资源过期的绝对时间，GMT格式，如果本地时间不对就会出现缓存过期问题，Cache-Control: max-age指定了资源过期的相对时间，单位是秒。在支持HTTP/1.1的浏览器上，如果发送两个头，那么Cache-Control: max-age会覆盖掉Expires;在支持HTTP/1.0的浏览器上，即使发送了两个头，但是只有Expires会起作用，所以为了兼容老的浏览 器，还是要同时发送这两个头。

Expires/Max-age生效后，会在有效期内，如果在浏览器再次请求该资源（如在浏览录入URL回车行为或者转到，还有一种就是浏览器后退．），就直接会从浏览器的Cache中提取数据，而不用去服务器获取数据．这里需要注意一下的是，如果是GET资源才会从浏览器Cache中提取缓存，如果是POST则不会．

Public：

设置 Cache-Control: public 以指定响应能由客户端和共享（代理）缓存进行缓存。

Private(默认为private)：

设置 Cache-Control: private 以指定响应只能缓存在客户端，而不能由共享（代理服务器）缓存进行缓存。一些个人信息，我们建议private

no-cache：

设置 Cache-Control: no-cache 标头。则浏览器不会留下缓存．写操作时，建议no-cache

在PHP session\_start()会自动设置：

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

GET /content/templates/wp-themes-blogger/images/bg\_cal\_today.gif HTTP/1.1

Accept: image/png,.image/svg+xml,.image/\*;q=0.8,.\*/\*;q=0.5

Referer: http://dev.jizhiinfo.net/t/

Accept-Language: zh-CN

User-Agent: Mozilla/5.0.(compatible;.MSIE.9.0;.Windows.NT.6.1;.Trident/5.0)

Accept-Encoding: gzip,.deflate

Host: dev.jizhiinfo.net

If-Modified-Since: Thu,.08.Sep.2011.06:59:57 GMT

Connection: Keep-Alive

Cookie: Hm\_lpvt\_593485456f0132e2c3a1cbd773b7ba8f=1318395769825;.\_\_utmc=81173250;.jiathis\_rdc=%7B%22http%3A//dev.jizhiinfo.net/%3Fpost%3D32%22%3A%221%7C1318379064472%22%7D;.PHPSESSID=58a9967364a9529551dcdbd29493ffcd;.EM\_AUTHCOOKIE\_8UgKuYjaH74xQddG8EBwfkN5pzScXNeM=gzjzjsb%7C%7C7a974b1e2e08b780698aa936a4820834;.Hm\_lvt\_593485456f0132e2c3a1cbd773b7ba8f=1318395599523;.\_\_utma=81173250.1755484502.1318332089.1318342184.1318373329.3;.\_\_utmz=81173250.1318373329.3.3.utmcsr=tongji.baidu.com|utmccn=(referral)|utmcmd=referral|utmcct=/hm-web/2866543/home/welcome

HTTP/1.1.

304.Not.Modified

Server: nginx/1.0.5

Date: Wed,.12.Oct.2011.04:59:00 GMT

Last-Modified: Thu,.08.Sep.2011.06:59:57.GMT

Connection: keep-alive

        过程如下:  
                1. 客户端请求一个页面（A）。   
                2. 服务器返回页面A，并在给A加上一个Last-Modified/ETag。   
                3. 客户端展现该页面，并将页面连同Last-Modified/ETag一起缓存。   
                4. 客户再次请求页面A，并将上次请求时服务器返回的Last-Modified/ETag一起传递给服务器。   
                5. 服务器检查该Last-Modified或ETag，并判断出该页面自上次客户端请求之后还未被修改，直接返回响应304和一个空的响应体。

ETag

首先说明ETag这是一个有争议的head，它在HTTP/1.1中被实现。它主要目的是实现客户端的缓存，它是1.0版本中Expires的补充。在Apache中ETag的值是由文件的索引节点（INode），文件大小（Size）和最后修改时间（MTime）进行Hash后得到的。

在网站的使用中，它的流程如下：

第一请求：

C🡪S：Get /pp.jpg

S🡪C：HTTP/1.1 200 OK

ETag：W/"4481-1291274210000"

Last-Modified：Sat, 19 Mar 2011 21:56:39 GMT

Expires：Thu, 01 Dec 2011 09:12:16 GMT

Cache-Control：public, max-age=2592000

第二请求：

C🡪S：GET /images/sy\_r14\_c27.jpg HTTP/1.1

If-Modified-Since：Sat, 19 Mar 2011 21:56:39 GMT

If-None-Match ：W/"4481-1291274210000"

S🡪C：HTTP/1.1 304 Not Modified

ETag：W/"4481-1291274210000"

ETAG解决了什么问题呢？对文件索引节点，文件大小及最后修改时间进行Hash值，通过这个组合得到的值更精确，相对Last-Modified它更精确，Last-Modified的精确时间是秒，Linux的文件系统中的MTime只能支持到秒单位，不能精确到微秒毫秒，如果一个文件在一秒内被多次修改将不会被Last-Modified发现，而ETag则解决了这个问题。

但是ETag也是存在缺陷的，并且这个缺陷很要命，现在大多数的网站都会部署多台WEB服务器，更多的静态资源都会部署在CDN上面，如果我们通过Etag来访问，就会存在一个问题，如果这次请求的WEB服务器与上次的WEB服务器不同，那它的文件索引节点(iNode)就会不同，那么就会再次发生下载，相同的域名，如果交错访问不同的WEB服务器，每次得到的ETag将都会不同，都将发生流量。

ETag的缺陷修复：

HTTP/1.1 标准并没有规定Etag的内容是什么或者说要怎么实现，唯一规定的是Etag需要放在""内。Apache中目前的ETag使用了弱ETag，它以W/开始，比如:W/"2e681a"

弱Etag(WeakEtag),这个Etag仅仅基于MTime来生成（待考证）

我在测试tomcat的时候发现：W-ETAG不会因为inode的变化而生成新的Etag

Nginx默认不支持ETAG.

Nginx中默认没有添加对Etag标识.Igor Sysoev的观点”在对静态文件处理上看不出如何Etag好于Last-Modified标识。”

Note:Yes, it's addition,and it's easy to add, however, I do not see howETag is better than Last-Modified for static files. -Igor Sysoev

经测试baixin.com，news.sina.com.cn没有使用ETag，都是使用了Last-Modified。原因比较清楚。多台WEB Server时，不会出现重复请求的问题。同时ETAG的计算量也相对较大。百姓网在这方面吃亏，2年内浪费了很多的流量

不用ETAG的实现

请求 GET /uploads/user/69/138715\_50.jpg HTTP/1.1

If-Modified-Since Mon, 29 Aug 2011 09:24:02 GMT

响应 HTTP/1.1 304 Not Modified

Last-Modified Mon, 29 Aug 2011 09:24:02 GMT

Expires Wed, 02 Nov 2011 09:15:13 GMT

其他：

另外大家一定要注意的是，通过SFTP上传的文件会将文件MTime做修改。而我测试通过wget从其他服务器下载文件，文件的MTime就没有发生变化。

也存在问题，就是MTIME只能精确到秒，如果文件一秒内修改多次，就不能实时拿到最新的文件。

Connection/keepalive

1.1:默认为keepalive

1.0:默认为close

Keepalive是一把双刃剑，要使用好，参数设置好，同时要分清楚场景。

Keepalive会大大增加socket的连接数目。

http的keep-alive是看情况的

如果一个访问会对同一个服务器请求多次那是有用的。不然就是没用的。

比如在完全分离动态页面和静态文件的时候，跑动态页面的服务器就可以关闭keep-alive，以节省连接数。

在用户不是太多的时候，长连接还是很有用的，可以节约分组，提升响应速度，但是一旦超出某个平衡点，由于为了保持很多长连接，创建了太多的进程，导致系统不堪重负，内存不够了，开始换入换出，cpu也被很多进程吃光了,load上去了。这种情况下，对apache来说，每次请求重新建立连接要比保持这么多长连接和进程更划算。

对于静态服务器来说，由于一个域名会请求N多资源，比较宜开启Keep-Alive，但是对于动态服务器，不宜开启Keep-Alive，因为这样会造 成很多的空闲进程，浪费内存空间。

　　配置：

　　KeepAlive On #开启KeepAlive

　　KeepAliveTimeout 2 #保持连接2秒，不宜太长

如果有多个小文件要从一个站点提取，我们就建议keepalive=true.

请求 GET / HTTP/1.1

Connection Keep-Alive

响应 HTTP/1.1 200 OK

Connection Keep-Alive

Keep-Alive timeout=2, max=100

Connection：请求：close（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，断开连接，不要等待本次连接的后续请求了）。   
keepalive（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，保持连接，等待本次连接的后续请求）。   
响应：close（连接已经关闭）。   
keepalive（连接保持着，在等待本次连接的后续请求）。   
Keep-Alive：如果浏览器请求保持连接，则该头部表明希望 WEB 服务器保持连接多长时间（秒）。例如：Keep-Alive：300

Referer

原资源地址，用于跟踪用户的行为，另外登陆页面可通过Referer回退到之前页面。

Range

User-Agent

请求用户的信息

Ms user-agent

<http://msdn.microsoft.com/en-us/library/ms537503.aspx>

Spaw:

User-Agent:Mozilla/3.0 (compatible; Indy Library)

正常：

User-Agent: Mozilla/5.0.(compatible;.MSIE.9.0;.Windows.NT.6.1;.Trident/5.0)..

User-Agent:Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)

User-Agent:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

User-Agent:.Mozilla/5.0.(Windows.NT.6.1).AppleWebKit/535.1.(KHTML,.like.Gecko).Chrome/13.0.782.220.Safari/535.1

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)

"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)"   
"Mozilla/5.0 (Windows; U; Windows NT 6.0; zh-CN; rv:1.8.1) Gecko/20061010 Firefox/2.0"  
"Opera/9.10 (X11; Linux i686; U; zh-cn)"  
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"  
"Mozilla/5.0 (X11; U; Linux i686; zh-CN; rv:1.8.0.4) Gecko/20060608 Ubuntu/dapper-security Firefox/1.5.0.4"  
"Mozilla/5.0 (X11; U; Linux i686; zh-CN; rv:1.8.1) Gecko/20060601 Firefox/2.0 (Ubuntu-edgy)"

排版引擎(Layout Engine/Readering Engine)

排版引擎的提出由Mozilla提出来，它的Gecko排版引擎提出来之后，市场上陆续出来其他的排版引擎。

网页浏览器排版引擎：

Trident/Tasman(IE内核，或者叫MSHTML)：1997年在微软的IE4里面首次被提出来，在以后的IE里面不断升级，一直到现在Trident一直存在IE. Tasman则是在mac机上面的IE排版引擎。现在市场上有很多的浏览器都使用了Trident的排版引擎，如Maxthon，QQ浏览器等，同时也包含了outlook，另外RealPlayer有使用它的mini-brower。Trident由于win的原因也长期处于市场领先地位。

Gecko：开源，C++编写的排版引擎，该引擎原本是由网景通信公司开发的，现则由Mozilla基金会维护。使用了Gecko的浏览有：Firefox，Mozilla等，这的特点就是小，类似OS的微内核，扩展性好。

KHTML/WebKit：WebKit排版引擎下面也有两个重要产品：Google Chrome，Safari。WebKit的前身为开源的Linux项目组KDE设计开发的KHTML，后来Apple与KDE分歧，由Apple发展到现在的WebKit，为开源项目。Nokia的S60浏览器，还有现在市场上流行的Google Chrome，都是基于WebKit开发实现的。它的优势在于WebKit拥有清晰的源码结构、极快的渲染速度。

另外国内的360浏览器，百度浏览器，搜狗浏览器都采用了双内核，如360的极速模式为WebKit引擎，IE兼容则是Trident引擎。

|  |  |
| --- | --- |
| Token | Description |
| Mozilla/5.0 | Application name and version. For historical reasons, Internet Explorer identifies itself as a Mozilla browser. |
| compatible | Compatibility flag token. It indicates that Internet Explorer is compatible with a set of common features. |
| MSIE 9.0 | The Version token identifies the browser and contains the version number, as reported by the current [browser mode](http://msdn.microsoft.com/en-us/library/gg589500.aspx). The example value indicates Internet Explorer 9. |
| Windows NT 6.1 | The Platform token identifies the operating system and version. The example token indicates Windows 7. |
| Trident/5.0 | The Trident token identifies the version of MSHTML (Trident) and can be used to determine whether or not the webpage is displayed in Compatibility View. |

攻击

今天突然的发现网站的响应速度非常慢, 登录到服务器上发现CPU高达90%

查看系统的访问日志, 一个叫Mozilla/3.0 (compatible; Indy Library)的东东访问网站异常的频繁

查了一下,据说这是一种叫做Chinese Spam Bot的机器人, 有人认为是一种蠕虫病毒

这种机器人的特点,是快速读取能够找到的内容, 但是,只读文字不会读取其他的资源,比如,图片等

Indy Library最初是一个关于网络操作的开源程序库，不过这个程序库被强制破解并被滥用于一些中国的spam bots中。

也就是说，这个“家伙”很可能是一个自动发送垃圾广告的程序。

将其屏蔽掉的方法有两种：

（1）修改.htaccess文件

SetEnvIfNoCase User-Agent "Indy Library" bad\_bot

Order Allow,Deny

Allow from all

Deny from env=bad\_bot

改后系统访问日志,所有的访问都返回403, 机器人或许会没完没了地访问同一个页面

（2）修改网站代码, 返回空的访问

例如

if( "Mozilla/3.0 (compatible; Indy Library)" == $user\_agent)

{

exit();

}

这样服务器会主动的返回200的状态,但是不返回任何的内容

gzip/deflate压缩

通过压缩，让浏览器的响应更快速，缺点就是要加多一些ＣＰＵ的计算。

同时：当你的带宽成为了你的瓶颈的时候，你就需要将HTTP的内容进行压缩。

压缩与否测试: <http://gzip.zzbaike.com/>

C🡪S

Accept-Encoding: gzip, deflate(zlib格式)

一般情况下只要在nginx.conf的http段中加入下面几行配置即可

   gzip  on;  
   gzip\_min\_length  1000;  
   gzip\_buffers     4 8k;

   gzip\_comp\_level 6; #0-9 默认值为1，值越大压缩率越高，消耗的cpu资源越多，传输量减小。  
   gzip\_types text/plain text/html text/css application/json application/x-javascript text/xmlapplication/xml application/xml+rss text/javascript;

重启nginx  
可以通过[网页gzip检测工具](http://gzip.zzbaike.com/" \t "_blank)来检测网页是否启用了gzip

Deflate(zlib格式)

gzip响应数据:

HTTP/1.1.200 OK

Server:.nginx/1.1.7

Date:Thu, 08 Dec 2011.07:58:01.GMT..

Content-Type:.text/html..

Transfer-Encoding:.chunked..

Connection:.keep-alive..

X-Powered-By:.PHP/5.3.8..

Content-Encoding:.gzip

....46.............V.-NW.Rr..u..)5.4O.)575.P.Q\*..M-.I.-P.246266.432.....".2.....0....

GZip＋Chunked是如何编码的，每个Chunk块进行一次GZip独立压缩。

Chunk传输编码(3.6.1/19.4.6)

Transfer-Encoding: chunked

接受完Content-Length中定义的长度字节后开始解析页面，但如果服务端有部分数据延迟发送吗，则会出现浏览器白屏，造成比较糟糕的用户体验。

一般HTTP通信时会使用是Content-Length头信息性来指定小，但是有时候无法确定信息大小，就要使用trunked编码动态的提供body内容的长度。进行Chunked编码传输的HTTP数据要在消息头部设置：Transfer-Encoding: chunked表示Content Body将用chunked编码传输内容。Chunked编码一般使用若干个chunk串连而成，最后由一个标明长度为0的chunk标示结束。每个chunk分为头部和正文两部分，头部内容指定下一段正文的字符总数（非零开头的十六进制的数字）和数量单位（一般不写,表示字节）.正文部分就是指定长度的实际内容，两部分之间用回车换行(CRLF)隔开。在最后一个长度为0的chunk中的内容是称为footer的内容，是一些附加的Header信息（通常可以直接忽略）。

[Chunk大小][ CRLF][Chunk数据体][ CRLF][Chunk大小][ CRLF][Chunk数据体][ CRLF][0][ CRLF]

   chunked编码的基本方法是将大块数据分解成多块小数据，每块都可以自指定长度，其具体格式如下（BNF文法）:  
    Chunked-Body   =

\*chunk            /\*0或者n个chunk\*/

                     last-chunk         /\*最后一个chunk \*/

                     trailer            /\*尾部\*/

                     CRLF               /\*结束标记符\*/

   chunk          = chunk-size [ chunk-extension ] CRLF

                        chunk-data CRLF

   chunk-size     = 1\*HEX

last-chunk     = 1\*("0") [ chunk-extension ] CRLF

chunk-extension= \*( ";" chunk-ext-name [ "=" chunk-ext-val ] )

chunk-ext-name = token

chunk-ext-val  = token | quoted-string

chunk-data     = chunk-size(OCTET)

trailer        = \*(entity-header CRLF)

解释：

Chunked-Body表示经过chunked编码后的报文体。报文体可以分为chunk, last-chunk，trailer和结束符四部分。chunk的数量在报文体中最少可以为0，无上限；

每个chunk的长度是自指定的，即，起始的数据必然是16进制数字的字符串，代表后面chunk-data的长度（字节数）。这个16进制的字符串第一个字符如果是“0”，则表示chunk-size为0，该chunk为last-chunk,无chunk-data部分。

可选的chunk-extension由通信双方自行确定，如果接收者不理解它的意义，可以忽略。

trailer是附加的在尾部的额外头域，通常包含一些元数据（metadata, meta means "about information"），这些头域可以在解码后附加在现有头域之后

gzip包头:1F 8B

如:

0x00f0: 6f64 696e 673a 2067 7a69 700d 0a0d 0a34 oding:.gzip....4

0x0100: 360d 0a1f 8b08 0000 0000 0000 03ab 56ca 6.............V.

0x0110: 2d4e 57b2 5272 f6f7 750d 8929 35b5 344f -NW.Rr..u..)5.4O

0x0120: 8929 3537 35b2 50d2 512a c9cc 4d2d 2e49 .)575.P.Q\*..M-.I

0x0130: cc2d 50b2 3234 3632 3636 b234 3332 a805 .-P.246266.432..

0x0140: 0014 0822 cc32 0000 000d 0a30 0d0a 0d0a ...".2.....0....

JAVA实现例子:

StringBuilder sb = new StringBuilder();

sb.append("HTTP/1.1 200 OK\r\n");

sb.append("Content-Type: text/plain\r\n");

sb.append("Transfer-Encoding: chunked\r\n\r\n");

sb.append("25\r\n");

sb.append("This is the data in the first chunk\r\n"); // 37 bytes

sb.append("\r\n1A\r\n");

sb.append("and this is the second one"); // 26 bytes

sb.append("\r\n0\r\n\r\n");

十六进制包长+\r\n+报文包+\r\n 为一个传输单元

0+\r\n+\r\n 当遇到这种空传输单元时结束

Cookies

负载均衡时，服务器之间跳转保证了session的一致性。

可以基于Cookies实现的Session。

像论坛上的贴图有些[IMG]标签没过滤好,cookie很容易被获取.

session\_id的泄露最大的可能是嗅探，如果你处于广播型的以太网内就可以被嗅探

基于cookie所以特别不安全.只要这个session\_id被知晓或者说cookie内容被盗(这很容易办到), 就很有可能被盗窃身份.我在作程序时session数据都加入了IP验证.   非根本性的解决方案.

用P3P header解决iframe跨域处理cookie，在Ａ域也可以Ｂ域的cookies

<http://blog.csdn.net/wonder4/article/details/2125804>

httponly：不被js访问的cookie

security

会话cookie

持久cookie:

Set-Cookie: uuauth=19218fcc8f5894fcd95531f269978f8a; expires=Thu, 10-Apr-2014 13:37:38 GMT; path=/; domain=.99yx.com; httponly

会话cookie:

Set-Cookie: JSESSIONID=49201DD5D31567F786A63DAA51B0A5D1; Path=/new

Expire:If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).

有效时间0表示该cookie为会话cookie,会话cookie是指cookie的生命周期为从会话开始到会话结束(关闭浏览器)

setcookies();

GET response head:

Set-Cookie uuid=ttxie; expires=Wed, 01-Jan-2014 08:21:51 GMT; path=/; domain=uu1758.com

Set-Cookie pwd=123456

GET request head:

Cookie pwd=123456; uuid=ttxie

文件上传MINE-HTML

application/x-www-form-urlencoded、multipart/form-data

Form 的POST通常是Content-Type：application/x-www-form-urlencoded

Content-Type

MIME邮件格式分析及信息提取：<http://wezly.iteye.com/blog/1045215>

HTTP协议和MIME协议是两个概念。多用途网际邮件扩充协议（MIME）是 Multipurpose Internet Mail Extensions 的缩写。

multipart类型，是MIME邮件的精髓。邮件体被分为多个段，每个段又包含段头和段体两部分，这两部分之间也以空行分隔。常见的multipart类型有三种：multipart/mixed, multipart/related和multipart/alternative。如果在邮件中要添加附件，必须定义multipart/mixed段；如果存在内嵌资源，至少要定义multipart/related段；如果纯文本与超文本共存，至少要定义multipart/alternative段。

什么是“至少”？举个例子说，如果只有纯文本与超文本正文，那么在邮件头中将类型扩大化，定义为multipart/related，甚至multipart/mixed，都是允许的。

1.http协议的主体信息部分是使用html来包装的,http协议刚开始是用来传送html的，只是和MIME协议结合才可以传送别的格式数据。

2.http与MIME应该分别是两个此协议,MIME可以附属在其他许多协议上,例如SMTP协议,利用MIME协议可以在协议中包含特殊格式的信息文本.例如:在http协议中包含pdf格式的数据,可以在浏览器中打开,在邮件中可以利用MIME协议来包含pdf附件,其余其他可以包含MIME数据的协议,我就不清楚了,我想,这应该可以有很多,因为从原理上说,这仅仅是按一定格式包含一段特殊数据。

3.浏览器可以认为是用来解读http协议的客户端,最终显示的内容是http协议数据体部分html格式的数据。

注意Form表单中enctype="multipart/form-data", method=post, type="file"  这三个属性是必须的。multipart/form-data 是新增的编码类型，以提高二进制文件的传输效率。

POST /index.php?m=att&a=swfupload&dosubmit=1 HTTP/1.1  
Content-Type: multipart/form-data; boundary=----------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5  
User-Agent: Shockwave Flash  
Host: 109.106.40.25  
Content-Length: 101  
Connection: Keep-Alive  
Cache-Control: no-cache  
Cookie: PHPSESSID=3s4csitc6gt503fbis6i6c3ei4

------------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5  
Content-Disposition: form-data; name="thumb\_width"

0  
------------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5  
Content-Disposition: form-data; name="Filedata"; filename="aa.php;"  
Content-Type: application/octet-stream

<?phpinfo()?>  
------------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5  
Content-Disposition: form-data; name="Upload"

Submit Query  
------------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5--

注意这一行：

Content-Type: multipart/form-data; boundary=----------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5

------------GI3GI3GI3Ef1KM7KM7gL6Ef1Ef1Ij5

消息头中的boundary比数据包中的boundary少 --

而最后一个boundary最后多出个 --

提交数据实际上的格式是这样的：

假设HTTP头中我们指定的分割付是xxxxxx，那么提交数据的格式是：

\r\n--xxxxxx\r\n<数据>\r\n--xxxxxx\r\n<数据>\r\n--xxxxxx--\n

|  |  |  |
| --- | --- | --- |
| Multipart Types (mostly email) |  |  |
| Messages with multiple parts |  | multipart/mixed |
| Messages with multiple, alternative parts |  | multipart/alternative |
| Message with multiple, related parts |  | multipart/related |
| Multiple parts are digests |  | multipart/digest |
| For reporting of email status (admin.) |  | multipart/report |
| Order of parts does not matter |  | multipart/parallel |
| Macintosh file data |  | multipart/appledouble |
| Aggregate messages; descriptor as header |  | multipart/header-set |
| Container for voice-mail |  | multipart/voice-message |
| HTML FORM data (see Ch. 9 and App. B) |  | multipart/form-data |
| Infinite multiparts - See Chapter 9 (Netscape) |  | multipart/x-mixed-replace |
|  |  |  |
| Message Types (mostly email) |  |  |
| MIME message |  | message/rfc822 |
| Partial message |  | message/partial |
| Message containing external references |  | message/external-body |
| Message containing USENET news |  | message/news |
| HTTP message |  | message/http |

邮件：

From: "xielijun" <xielijun@jizhiinfo.net>

To: <xielijun@jizhiinfo.net>

Subject: =?gb2312?B?16q3ojoguau45jIwMTEuMTEuOS4=?=

Date: Wed, 9 Nov 2011 21:28:30 +0800

Message-ID: <000001cc9ee3$78f32ff0$6ad98fd0$@net>

MIME-Version: 1.0..Content-Type: multipart/mixed;

boundary="----=\_NextPart\_000\_0001\_01CC9F26.87166FF0"

X-Mailer: Microsoft Office Outlook 12.0

Thread-Index: AcyezwNwotdjz3ptQGW9LrjqkJ5sQQAFGW2g

Content-Language: zh-cn

MIME

------=\_NextPart\_000\_0001\_01CC9F26.87166FF0

Content-Type: multipart/alternative;

boundary="----=\_NextPart\_001\_0002\_01CC9F26.87181DA0"

------=\_NextPart\_001\_0002\_01CC9F26.87181DA0

Content-Type: text/plain; charset="gb2312"

Content-Transfer-Encoding: base64

IA0KDQogDQoNCreivP7IyzogeHV5YW5mZWkgW21haWx0bzp4dXlhbmZlaUBqaXpoaWluZm8ubmV0XSANCreiy83KsbzkOiAyMDExxOoxMdTCOcjVIDE4OjUxDQrK1bz+yMs6IL2quuzWvjsg0LvA6L78DQqzrcvNOiC1y7qjwcE7IMDotqvD9zsgzrrV8dDLDQrW98ziOiC5q7jmMjAxMS4xMS45Lg0KDQogDQoNCsDNt7O908rVuL28/qOuDQoNCiANCg0KMjAxMS0xMS0wOSANCg0KICBfX19fXyAgDQoNCnh1eWFuZmVpIA0KDQo=

Base64在线解码：<http://blbear.com/tools/base64-utf8.php>

------=\_NextPart\_001\_0002\_01CC9F26.87181DA0

Content-Type: text/html;...charset="gb2312"

Content-Transfer-Encoding: quoted-printable

<html xmlns:v=3D"urn:schemas-microsoft-com:vml"

--boundary分段标识

Content-Type： application/msword;

          name="readme.doc"

Content-Transfer-Encoding： base64

Content-Disposition： attachment;

          filename=" readme.doc "

……

文件内容的Base64编码

……

--boundary分段标识

彩信包：

POST /vas HTTP/1.1

Host:218.200.243.150:80

Content-Type:multipart/related;boundary="--NextPart\_0\_2817\_24856";type="text/xml";start="</tnn-200102/mm7-vasp>"

Content-Transfer-Encoding:8bit

Authorization:Basic.enhtZTp6eG1l

SOAPAction:""

MM7APIVersion:V1.5.3\_20040621

Connection:Close

Content-Length:1697

Mime-Version:1.0

this.is.a.multi-part.message.in.MIME.format

----NextPart\_0\_2817\_24856

Content-Type:text/xml;charset="GB2312"

Content-Transfer-Encoding:8bit

Content-ID:</tnn-200102/mm7-vasp>

<?xml.version="1.0".encoding="GB2312"?>

<env:Envelope.xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Header>

<mm7:TransactionID.xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23\_series/23.140/schema/REL-6-MM7-1-0".env:mustUnderstand="1">

104586954</mm7:TransactionID></env:Header><env:Body>

<SubmitReq.xmlns="http://www.3gpp.org/ftp/Specs/archive/23\_series/23.140/schema/REL-6-MM7-1-0">

<MM7Version>6.3.0</MM7Version><SenderIdentification>

<VASPID>801203</VASPID><VASID>10668002</VASID>

<SenderAddress>1066800232</SenderAddress></SenderIdentification>

<Recipients><To><Number>15043338382</Number></To></Recipients>

<ServiceCode>127</ServiceCode><LinkedID>19464520079152212805</LinkedID>

<ExpiryDate>2011-11-09T22:56:53+08:00</ExpiryDate>

<DeliveryReport>true</DeliveryReport><ReadReply>true</ReadReply>

<Subject>................</Subject><ChargedParty>ThirdParty</ChargedParty>

<ChargedPartyID>15043338382</ChargedPartyID></SubmitReq></env:Body>

</env:Envelope>

----NextPart\_0\_2817\_24856

Content-Type:multipart/related;start="10044-.smil";

type="application/smil";boundary="SubPart\_7452684322002\_77645"

Content-ID:smil.multipart

Content-Transfer-Encoding:8bit

--SubPart\_7452684322002\_77645

Content-Type:application/smil

Content-Transfer-Encoding:8bit

Content-ID:10044-.smil

Content-Location:10044-.smil

<smil.xmlns="http://www.w3.org/2000/SMIL20/CR/Language">

<head>

<meta.name="title".content=""/>

</head>

<body>

</body>

</smil>

--SubPart\_7452684322002\_77645--

----NextPart\_0\_2817\_24856--

缓存机制

HTTP WEB服务

<http://woodpecker.org.cn/diveintopython3/http-web-services.html>

网站带宽调优及用户体验度量：

带宽统计:

base:5分种

95计费，峰值计费

流量：http access.log bit\*3%=带宽带

用户体验度量：

一、

时间：发起连接 到，用户收到数据。

HTTP的access日志：

（加入建立连接之后，给到用户的时长）

apache nginx

%d(微秒), $request\_time

%t(秒)

nginx:

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" $http\_x\_forwarded\_for 【"$request\_time"】';

【"$request\_time"】默认情况下是没有的。

二、

也可以通过客户端来统计，通过javsscript来发送时间节点

也可以通过FLASH来度量，时间

三、

三层交换机：mirror到另外的服务器。

或者在proxy来统计，如varnish...

负载均衡：

3层IP

4层TCP

7层HTTP应用层

数据统计：

IP，区域，运营商，流量，处理时长

流量数据：

统计你的流量图，5分钟一次，绘图

IP，区域，运营商，流量，处理时长

请求响应数据：

IP，区域，运营商，流量，处理时长

网站访问统计

百度的统计缺少：IP地址显示，网站的流量(具体多少字节)

用户单个行为分析很好，清楚显示访问的每一个页面及在页面上停留的时间和先后顺序。用户的轨迹清晰，便于分析用户的行为。

Google统计也有试用，可是速度方面跟不上，具体功能没有深入。



Alexa免费排名服务

Alexa排名是根据对用户下载并安装了Alexa Tools Bar嵌入到IE、FireFox等浏览器，从而监控其访问的网站数据进行统计的，因此，其排名数据并不具有绝对的权威性，ALEXA这种统计的方式数据精确度存在问题，特别是在中国。1999年Alexa被亚马逊收购，亚马逊对加入了用户网上购物行为分析。

Alexa的网站世界排名主要分综合排名和分类排名两种。综合排名也叫绝对排名，即特定的一个网站在所有网站中的名次。Alexa每三个月公布一次新的网站综合排名。此排名的依据是用户链接数（Users Reach）和页面浏览数（Page Views）三个月累积的几何平均值。



PV量：

UV量：

网站推广SEO

中国社会科学院发布的2011《新媒体蓝皮书》中显示，中国互联网站总数为191万个，比2009年的323万个减少了132万个。这是中国互联网站年度总数第一次出现下降，降幅高达41%，主要原因为09年底推行了CN域名实名制和网站拍照备案新政。但同时，去年中国网页数量达到600亿个，比2009年增长78.6%。

在如此众多的网站和网页中，如何才能脱颖而出，需要网络的营销方式。

SEO(Search Engine Optimization)搜索引擎优化，是较为流行的网络营销方式，主要目的是增加特定关键字的曝光率以增加网站的能见度，进而增加销售的机会。

SEO技术建议：

网页标题<title><title>在这里加入最重要的目标关键字。

网页的meta信息(keywords,description), 搜索和排名更靠前。

图片alt信息，让图片更容易找到

不采用frame/frameset/iframe标签

使用H1标签，标题标签从最大到最小依次为H1到H6，H1填写主要关键字。

网站导航文字形式，让spider更方便。

加入网站地图

百度收录：加入到百度的站长统计之后，第二天早上便见到百度的robot spider.

Google PageRank：

提交给google，方便googlebot过来：

<https://www.google.com/webmasters/tools/submit-url?hl=zh-CN>

网络爬虫

二, 百姓网全面升级防爬取系统

在上个星期，百姓网全面升级了防爬取系统，来避免竞争对手爬取百姓网的用户信息。这个举动导致竞争对手无法再爬取百姓网的用户信息，这让靠着爬信息生存的竞争对手失去了重要的信息来源。百姓网这个做法，破坏了业内的潜规则，甚至导致赶集网杨总在微博上说：“百姓网成为整个行业的威胁”。 我们相信这个声明也能够让大家理解为啥最近我们的同行突然这么关注百姓网。

接下来我们来说明大家最关注的问题： 凭什么说爬取信息是行业内潜规则？

对于这个事情，我们不希望说太多，我们只用2张截图来说明这个事情。

截图1，在今年3月底，百姓网发现赶集网上出现了百姓网的用户信息，因此发信给赶集网的负责人询问原因，得到的答复如下：

[](http://blog.baixing.com/wp-content/uploads/2011/04/11.jpg)

截图2：赶集网技术团队招聘微博，请注意画红框的地方。

[](http://blog.baixing.com/wp-content/uploads/2011/04/22.jpg)

百姓网：速度为先

(联合创始人)

每天5K万左右动态请求

高峰每秒钟1300次

50多台服务器，分上海，北京两个数据中心

页面打开速度，alexa数据：

baidu.com 2.157''

google.com 1.577''

baixing.com 0.627''

ETag(inode-size-timestamp):

If-Noe-Mathc: "Etag"

304 Not Modified

如果在两台web server(不同的数据中心)，inode不同，ETag也会不同

百姓网在这方面吃亏，2年内浪费了很多的流量

一个HTTP请求，90%时间在css，图片，js

css放页面最上面

网站速度：

前端:

两本书：Hight Performance Web Sites, Even Fast Web Sites

YSlow是Yahoo发布的一款基于FireFox的插件。

<http://www.oschina.net/p/yslow>

YSlow可以对网站的页面进行分析，并告诉你为了提高网站性能，如何基于某些规则而进行优化。

IE/chrome浏览器也可以查看速度。

测试环境：IE6 800\*600，最坏的环境，很多插件

监控最好办法，让你的网页打开更快。

JavaScript + Google Analytics报表数据

网页中拿数据，在页头和页尾加一个js, endtime-starttime。将数据拿到。

单击事件，到完成页面打开的一个时间差.

后端，

每一次的SQL做时间统计，做执行时间数据分析，如果大于多少就记录去某个数据文件

监控系统：图表形式体现出来

核心负载：搜索+数据库

负载均衡+Web+memcache+ Mysql

solar

CDN，静态加速，动态基本无效。

网络，

机房非常重要，速度

local site(<http://guangzhou.baixing.com/>, shanghai.baixing.com)

南北机房

双机房问题：

数据同步不稳定

权限控制

数据的cache

日志数据需要回传分析，监控数据报表的依据

远程维护

如果不在乎100ms，可能会考虑一个好机房来处理，而不要双机房。

一定要监控，这个最为重要，才能保证速度的持续性。

10K, 2’’

7\*24

大众点评网

大众点评网

---------------------------------

每天7000W动态请求

活跃用户3000万/月，600万手机用户

手机占50%

asp+access->asp.net+sqlserver->asp.net+mysql+nosql[memcached]+MogileFS+Lucene

8名，17台服务器，17万访问量/天

dingping.com（5万）

静态动态内容分离,短域名用于静态资源

\*CSS/JS文本资源进行压缩，IE缓存

YUI Compressor

合并JS文件

启用GZIP压缩

图片优化：jpegoptim/optipng

Expire和Cache-Control的区别和优先级

\*负载均衡实现：

DNS轮询：免费，无成本，无法进行健康检查，策略调整麻烦。

Nginx/LVS,免费，有一定瓶颈

Hardware，可控，稳定性

但需要注意Session保持问题

(DNS+LVS)

图片多，第三方CDN，分布式文件系统(MogileFS)

css静态文件，文件名加版本号，控制文件更新

文件：g.base.v22.css

memcacheD：制定缓存策略(缓存对象可能无法删除)

DAO->数据访问+AOP

缓存对象加版本

\*多台memcached, 哈希算法

序列化反序列化CPU占用CPU高

雪崩现象：异步操作

消息队列：MemcacheQ/Redis

下阶段：数据库数据路由

经验：

尽早实现动静分离

Memcached好用，用好不容易

NoSQL Join，业务设计到技术实现上都贯彻

成熟的第三方平台：DNS，CDN

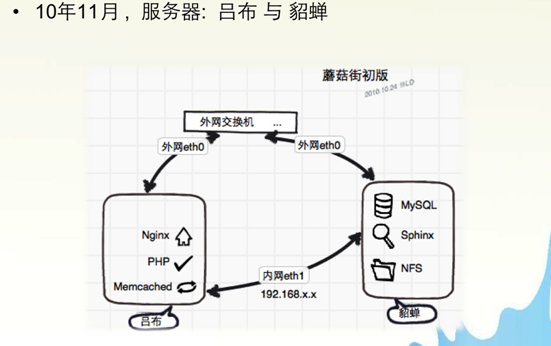
有效的监控系统，能够快速有效的Debug

[Trunk.ly CTO董洵谈架构](http://www.infoq.com/cn/articles/dx-on-architecture)

<http://www.infoq.com/cn/articles/dx-on-architecture>

蘑菇街：

http://www.infoq.com/cn/presentations/the-entrepreneurial-team-architecture





网线一定要好。监控系统排查问题。

DDOS攻击：

Distributed Denial of service (分布式拒绝服务攻击)，挟持肉机进行攻击

攻击手段：

全连接

SYN洪水

TCP复合包

UDP碎片

AJAX：

ajax实现的核心技术:

Document Object Model(DOM)

通过DOM找到要更新的元素如：

document.getElementById("").value

xmlhttp:

XMLHTTPRequest对象，无需刷新页面

只同服务器进行数据层的交换

最应用XMLHTTP是的微软，IE5以上

Mozilla创建：XMLHttpRequest

XMLHttpRequest:

var xmlhttp = new XMLHttpRequest();

微软的XMLHTTP:

var xmlhttp = new ActiveObject(msxml2.XMLHTTP);

var xmlhttp = new ActiveObject(Microsoft.XMLHTTP);

var xmlhttp = new XMLHttpRequest();

xmlhttp.open("GET", "http://localhost/addr?cardID=32", /\*async\*/true);

xmlhttp.onreadystatechange = myHandler;

xmlhttp.send(/\*no params\*/null);

/\*

json轻量级、快速，如果xml的话，需要使用req.responseXML中的DOM，一层层下去.

JSON是JavaScript的一个子集，你可以使用JavaScript自己的编译器来解析它，调用eval函数解析JSON仅需要一行

JSON是JavaScript和Python的子集

有人测试解析JSON比XML要快10倍！

\*/

function myHandler() {

if (xmlhttp.readyState == 4 /\*complete\*/) {

var addrField = document.getElementById('addr');

var card = eval('(' + req.responseText + ')');

addrField.value = card.addresses[0].value;

}

}

{

"fullname": "Sean Kelly",

"org": "SK Consulting",

"emailaddrs": [

{"type": "work", "value": "kelly@seankelly.biz"},

{"type": "home", "pref": 1, "value": "kelly@seankelly.tv"}

],

"telephones": [

{"type": "work", "pref": 1, "value": "+1 214 555 1212"},

{"type": "fax", "value": "+1 214 555 1213"},

{"type": "mobile", "value": "+1 214 555 1214"}

],

"addresses": [

{"type": "work", "format": "us",

"value": "1234 Main StnSpringfield, TX 78080-1216"},

{"type": "home", "format": "us",

"value": "5678 Main StnSpringfield, TX 78080-1316"}

],

"urls": [

{"type": "work", "value": "http://seankelly.biz/"},

{"type": "home", "value": "http://seankelly.tv/"}

]

}

<input type="test" onkeyup="sendrequet">

AJAX应用场景：

用户登陆，注册

级联菜单

树结构

数据过滤

直播室、聊天室

动静分离时，静态页面中动态实现：如新浪新闻的评论，奇艺视频播放时的评论.

实例：http://www.colr.org/

http://maps.google.com/

AJAX数据包：

POST./UserRegisterAction.do?action=countrySelectAction&countryCode=460 HTTP/1.1

Host:.www.shileke.com..

Connection:.keep-alive..

Content-Length:.0..

X-Requested-With: XMLHttpRequest /\*有些会加这个做区分\*/

Origin:.http://www.shileke.com..

User-Agent:.Mozilla/5.0.(Windows.NT.6.1).AppleWebKit/535.1.(KHTML,.like.Gecko).Chrome/14.0.835.202.Safari/535.1..

Accept:.\*/\*..

Referer:.http://www.shileke.com/UserRegisterAction.do?action=intoUserRegisterAction..

Accept-Encoding:.gzip,deflate,sdch..

Accept-Language:.zh-CN,zh;q=0.8..

Accept-Charset:.GBK,utf-8;q=0.7,\*;q=0.3..

Cookie:.JSESSIONID=F2D7F0C7727817F0803D4E20B68F4D57

HTTP/1.1.200.OK

Content-Type:.text/xml;charset=utf-8..

Transfer-Encoding:.chunked..

Date:.Mon,.31.Oct.2011.23:38:32.GMT..

Server:.Apache-Coyote/1.1

65

<response>

<city>

<citycodevalue>--....../...--</citycodevalue>

<citycode>0</citycode>

</city>

</response>

在Ajax请求中使用GET方法 (Use GET for AJAX Requests)

　　Yahoo! Mail 团队发现进行XMLHttpRequest的时候，POST方法在浏览器中分两步执行：先发送头部，然后发送数据。所以最好使用只发送一个TCP包(除非你有很多的cookie)的GET方法。IE中URL的最大长度是2000，所以如果你发送超过2000的数据就不能使用GET方法。

　　一个有趣的现象是，POST方法并不像GET那样实际发送数据(而Get则名副其实)。基于HTTP规范，GET方法意味着取回数据，所以当你只是请求数据时使用GET方法更为有意义(从语义上来说)，而在发送需要储存在服务器端的数据时则相反使用POST。

让你的AJAX可缓存，哪怕只有一个用户。

参考：

json让ajax加速

<http://4u4e.blog.163.com/blog/static/681452520076294288627/>

JQuery/Prototype

从 jQuery.com 下载。

应用最广泛的JavaScript框架，jQuery插件非常之多，涉及[LightBox灯箱插件](http://www.oschina.net/news/16416/30-best-jquery-lightbox-plugins)、日期插件、图表插件等各种类型的插件不计其数，OsChina就收录了1000多款jQuery插件。

使用 Google 的 CDN

<head>

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs

/jquery/1.4.0/jquery.min.js"></script>

</head>

Now:<http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js> 90K

使用 Microsoft 的 CDN

<head>

<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery

/jquery-1.4.min.js"></script>

</head>

$() 方法是在DOM中使用过于频繁的 document.getElementById() 方法的一个便利的简写，就像这个DOM方法一样，这个方法返回参数传入的id的那个元素。

jQuery 语法实例

$(this).hide()

演示 jQuery hide() 函数，隐藏当前的 HTML 元素。

$("#test").hide()

演示 jQuery hide() 函数，隐藏 id="test" 的元素。

#通过ID来获取元素

$("p").hide()

演示 jQuery hide() 函数，隐藏所有 <p> 元素。

$(".test").hide()

演示 jQuery hide() 函数，隐藏所有 class="test" 的元素。

所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

更多的选择器实例

语法 描述

$(this) 当前 HTML 元素

$("p") 所有 <p> 元素

$("p.intro") 所有 class="intro" 的 <p> 元素

$(".intro") 所有 class="intro" 的元素

$("#intro") id="intro" 的第一个元素

$("ul li:first") 每个 <ul> 的第一个 <li> 元素

$("[href$='.jpg']") 所有带有以 ".jpg" 结尾的属性值的 href 属性

$("div#intro .head") id="intro" 的 <div> 元素中的所有 class="head" 的元素

使用 jQuery 简化 Ajax 开发

<http://www.ibm.com/developerworks/cn/xml/x-ajaxjquery.html>

<http://www.w3school.com.cn/jquery/index.asp>

如果你的应用需要高性能，那么你要小心这些JavaScript框架。Prototype和jQuery在使用大的数据表和网格的时候会死的很难看。  
  
我对我目前正在工作的项目做了大量的基准测试，该项目大量使用AJAX技术，并且用在项目核心的大数据表上。  
  
我试验过jQuery和Prototype，它们的性能总是不行。问题在于document.getElementById()方法。DOM的查找操作是非常耗时的。实际上，我们的测试似乎表明DOM查找不是通过哈希来完成的。

[淘宝Kissy](http://www.cnblogs.com/bluedream2009/archive/2010/06/10/1755923.html)

CDN加速

五款免费智能CDN服务加速你的网站速度

<http://bbs.cfan.com.cn/thread-1386243-1-1.html>

DNSPOD: DNSPod建立于2006年3月份，是国内最早提供免费智能DNS产品的网站，致力于为各类网站提供高质量的多线智能DNS免费解析。

请马上到域名注册的地方将DNS修改为：  
f1g1ns1.dnspod.net  
f1g1ns2.dnspod.net

还可以做URL转向



<https://www.dnspod.cn/>

Webluker是China Cache旗下的一家免费提供CDN、域名管理、智能解析、网站监控的网站(这个是确定的，因为Nginx日志里留下的标志是China Cache)

<http://www.webluker.com/>

Webluker完全开放网站申请，只要你的网站符合两点要求：1、正规站点，2、有备案(大家懂的)。

整体上来看，yottaa的效果还是不错的，至少比我之前使用的webluker要快，要稳定，对wordpress支持更好。

1、cdn只需做cname指向，无需修改dns，这点很方便。

2、但是cdn服务并不算稳定。有一段时间在晚上访问出奇的慢，其他时间里加速效果也不算明显，只是在ping值上小很多。

3、就在前晚，cdn加速竟然出现主机名无效的访问错误，我只好把域名解析回源站。

4、后台系统还存在bug，比如删除不了添加过但已经失效的设备监控。

5、对动态网页没做到真正的缓存，对header含有set-cookis的页面直接不缓存。

6、cdn节点有时会失效，我就遇到过一个教育网节点失效的情况，还是我跟他们联系之后他们才发现的。

网站优化

Yahoo!网站性能最佳体验的34条黄金守则——内容

http://dudo.org/archives/2008051211216.html

YSLOW评分

尽可能少的HTTP请求（其中N个JS请求，N个CSS请求，N个CSS background images请求）

每次的HTTP头都占用了大量的网络。

Css在页前js在页后

网站测速

<http://www.webkaka.com/webCheck.aspx>

做好用户体验我把报告发给各位SEOer 用户最满意的打开网页时间，是在2秒以下。用户能够忍受的最长等待时间的中位数，在6～8秒之间。这就是说，8秒是一个临界值，如果你的网站打开速度在8秒以上，那么很可能，大部分访问者最终都会离你而去。如果等待12秒以后，网页还是没有载入，那么99%以上的用户会关闭这个网页，不再等待。

有一种经常被网页开发者忽略却往往十分浪费响应时间的跳转现象。这种现象发生在当URL本该有斜杠（/）却被忽略掉时。例如，当我们要访问http://astrology.yahoo.com/astrology 时，实际上返回的是一个包含301代码的跳转，它指向的是http://astrology.yahoo.com/astrology/  （注意末尾的斜杠）。在Apache服务器中可以使用Alias 或者 mod\_rewrite或者the DirectorySlash来避免。

加快网页下载速度的30种方法

<http://www.fastweb.com.cn/China_CDN_Newsletter/201108/Newsletter_ways_34.html>

Best Practices for Speeding Up Your Web Site

<http://developer.yahoo.com/performance/rules.html>

小项：

URL: <http://dev.jizhiinfo.net/favicon.ico>

Email地址用图片。

404的实现，让每一次的连接都可以拿有效的数据。

Nginx状态：

<http://www.9enjoy.com/nginx-status/>

http://wiki.nginx.org/NginxHttpStubStatusModule

http://hostingfu.com/article/quick-nginx-status-script

<http://kovyrin.net/2006/04/29/monitoring-nginx-with-rrdtool/>

CGI/FastCGI

通常HTML解析的时候，碰到文件是直接将文件sendfile的方式推送给客户端．CGI的不同之处，就是它不直接接内容送给客户端，而将CGI程序执行完成之后，将结果返回给客户端．CGI可以用任何一种语言编写，只要这种语言具有标准输入、输出和环境变量。它独立于WEB服务器．

通常情况下，WEB服务器与CGI服务器通过socket的方式来通信，实现数据的交互：

用户完成输入，向WEB服务器发起HTTP请求

WEB服务器接收到该请求后， FastCGI将CGI的有关数据设置成环境变量．FastCGI进程管理器启动/选择并连接到URL指定的外部CGI程序。

WEB服务器与外部CGI程序建立数据通道：**标准输入，标准输出**．同时保持同步．

外部CGI完成处理后将标准输出和错误信息从(同一连接)返回Web Server．

CGI标准将这个接口定义的非常简单 (即: WEB服务器收到客户端的请求后通过环境变量和标准输入(stdin)将数据传递给CGI程序, CGI程序通过标准输出(stdout)将数据返回给客户端).所以只要能操作标准输入/输出的程序语言都可以CGI程序, Perl/C++/JAVA/VB等.

ASP/JSP/PHP/PERL与CGI程序的关系?   
严格来讲,它们全都是CGI的变种,因为它们的操作原理都是CGI.而在又在CGI的基础上作了进一步的包装,   屏蔽了CGI的与程序语言相关的接口.

C/C++编写CGI还具有如下优势和原因:   
(1) 高效率,   C/C++不像PERL/VBS/JS等解释执行语言运行时解释执行源文件中的语句.   同时这一点仍非JAVA/PHP等所能及.   所以C/C++仍是许多WEB应用的首选,特点是大型WEB应用中.   
(2) 嵌入式设备(如PDA/数码产品/通信产品)WEB应用的首选,目前几乎所有的嵌入式设备都直接用C语言开发,而CPU/内存/外存等的限制几乎根本不可能安装如PERL/ASP/JSP的运行环境,所以嵌入式设备上C开发CGI几乎仍是唯一选择.

CGI，HelloWorld，新建文件hello.c：

#include <stdio.h>   
main() {   
    printf("Content-type:text/html\n\n");   
    printf("Hello,World!");   
}

用命令$gcc –o hello hello.c 生成 hello

然后查看apache配置文件httpd.conf, 设置为：[默认的差不多就是这样]

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"   
<Directory "/var/www/cgi-bin">   
    AllowOverride All   
    Options ExecCGI   
    Order allow,deny   
    Allow from all   
</Directory>

然后把生成的hello复制到 /var/www/cgi-bin/下，如果修改了配置的话，就先重起apache, 这时输入<http://localhost/cgi-bin/hello>,就应该可以看到结果了

FastCGI像是一个常驻(long-live)型的CGI，它可以一直执行着，只要激活后，不会每次都要花费时间去fork一次(这是CGI最为人诟病的fork-and-execute 模式)。

面前市面上的一些PHPCGI有：

PHP-FPM是一个PHP FastCGI管理器，目前已集成在php5.3.3之后版本里面．

Spawn-FCGI是一个通用的FastCGI管理服务器，它是lighttpd中的一部份，很多人都用Lighttpd的Spawn-FCGI进行FastCGI模式下的管理工作

php-cgi是被调用的进程，php-fpm是配置和管理进程的

cgi效率低，每次来了PHP请求，新建立一个PHP进程来解析，解析完毕进程销毁，再来请求再起进程。。。

fpm=fastcgi process manage，维持一定数量的进程数，供nginx调用，不用每次都新建进程，速度更快。原理上有点类似于数据库连接池吧。

|  |  |
| --- | --- |
| |  | | --- | |  | |
|  |

Libfcgi开发fastcgi

隆重推出C写的FastCGI Blog2.0

<http://www.cellphp.com/article-read-opensource-15-c-fastcgi-blog-2.0.html>

https://code.google.com/p/php-tokyocabinet/downloads/detail?name=blog.tar.bz2&can=2&q=#makechanges

什么是CGI、FastCGI、PHP-CGI、PHP-FPM、Spawn-FCGI？

<http://www.mike.org.cn/articles/what-is-cgi-fastcgi-php-fpm-spawn-fcgi/>