HTTP协议

HTTP协议概念及工作流程

1. HTTP协议

重要性：无论是以后用webservice，和爱是用Rest做大型架构，都离不开对HTTP协议的认识。

甚至可以简化的说：

webservice = HTTP 协议 + XML

Rest = HTTP 协议 + json

各种API，也一般是用HTTP + XML/json来实现的。

往小了说：做采集，小偷站也需要对HTTP协议有所理解，

以及Ajax，对HTTP有了解之后，学习Ajax是非常容易理解的。

HTTP协议学习目录

原理：

1. 形象理解HTTP协议
2. 动手试试HTTP协议
3. HTTP协议3部分介绍

实战：

4. PHP + socket编程发送HTTP请求
5. PHP批量发帖
6. HTTP协议防盗链

优化：

7. HTTP协议与缓存控制
8. HTTP协议与COOKIE
9. 持久连接

1.1 什么是协议

计算机中的协议和现实中的协议是一样的，一式双份/多份。双方/多方都遵从共同的一个规范，这个规范就可以称为协议。

计算机之所以能全世界互通，协议是功不可没的，如果美哟u协议，计算机各说各话，根本随都听不懂谁。

HTTP协议，ftp协议，SMTP协议，离婚协议....

协议就是按规矩说话

你来问我来答

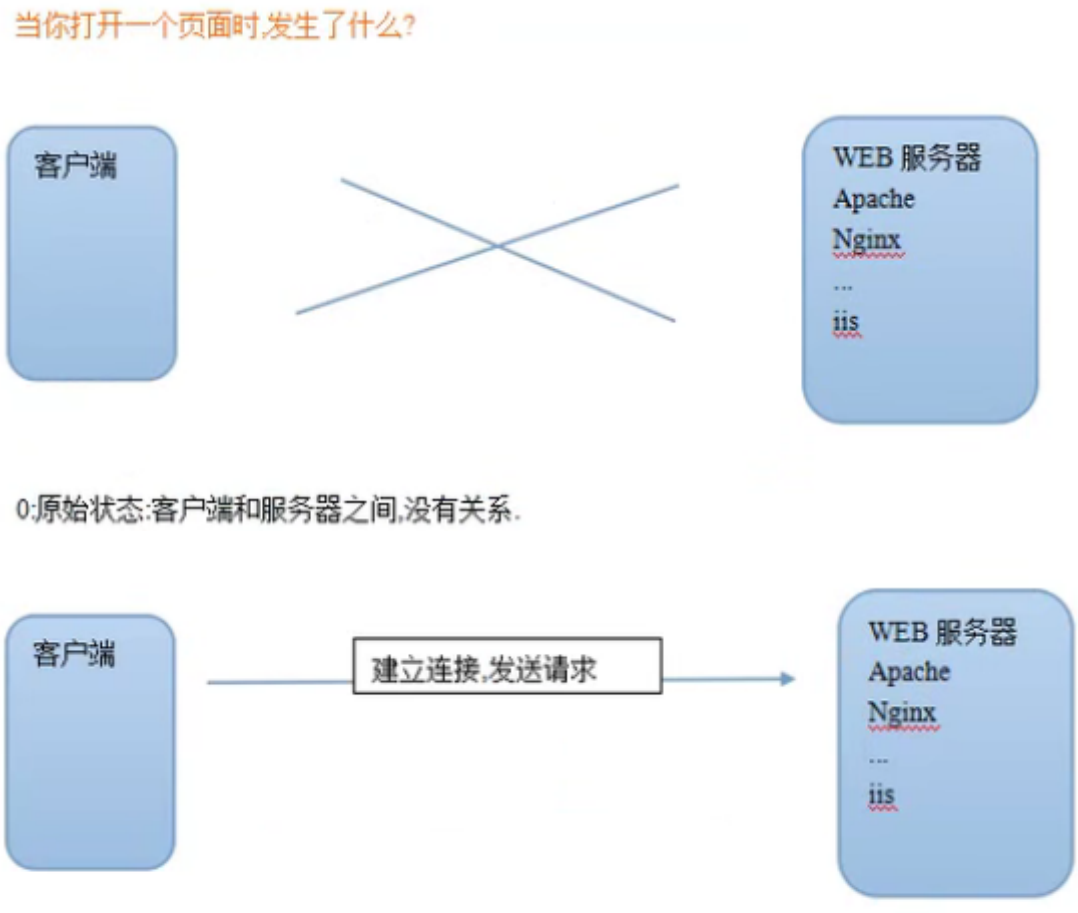
你怎么问，我怎么答

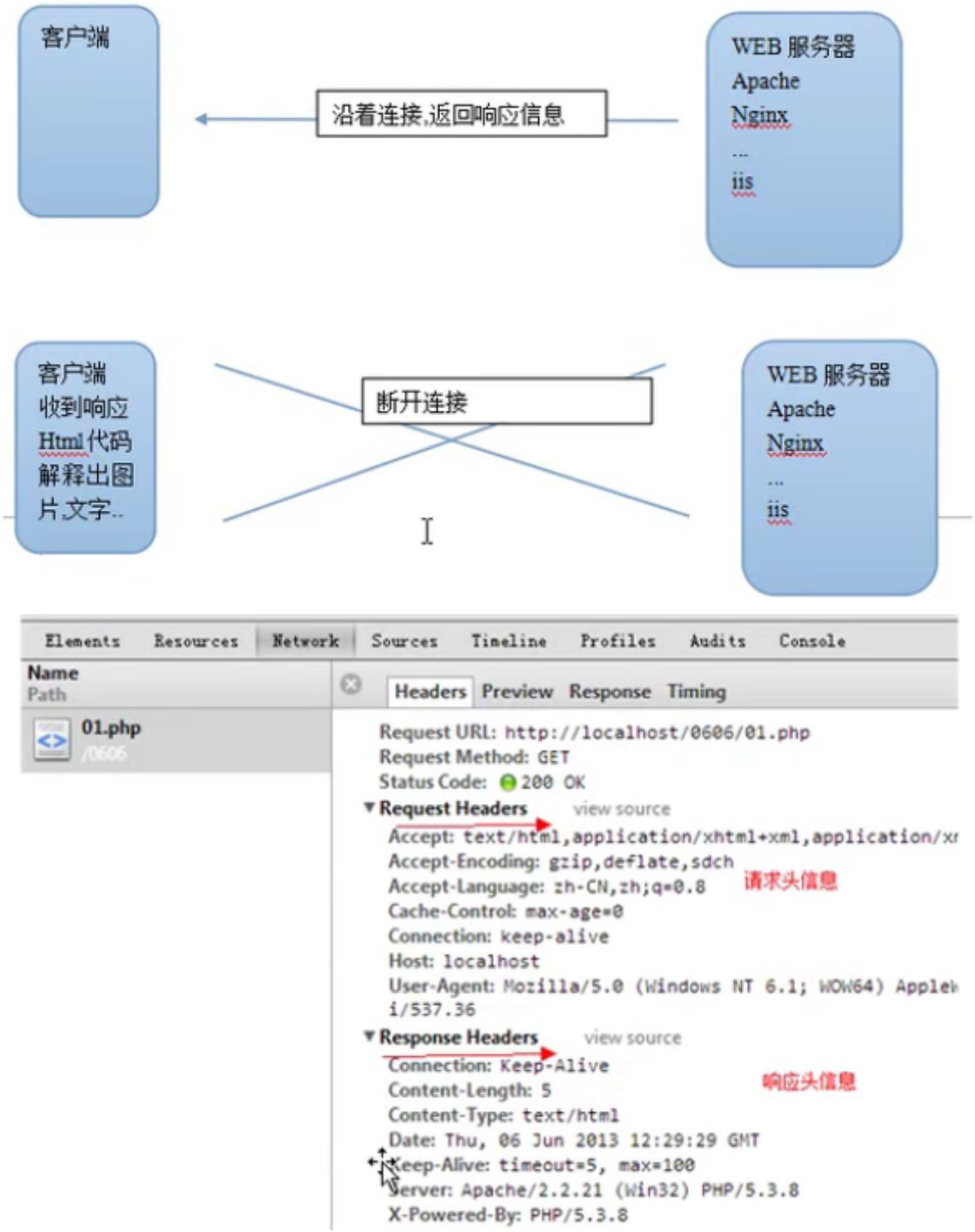
1.2 什么HTTP协议

HTTP 协议即按照一定规则，向服务器要数据，或发送数据。而服务器按一定规则，回应数据。

1.2.1 HTTP协议的工作流程

连接：就是网络上的虚拟电路





1.2.2 HTTP请求信息和响应信息的格式

1. **请求:**

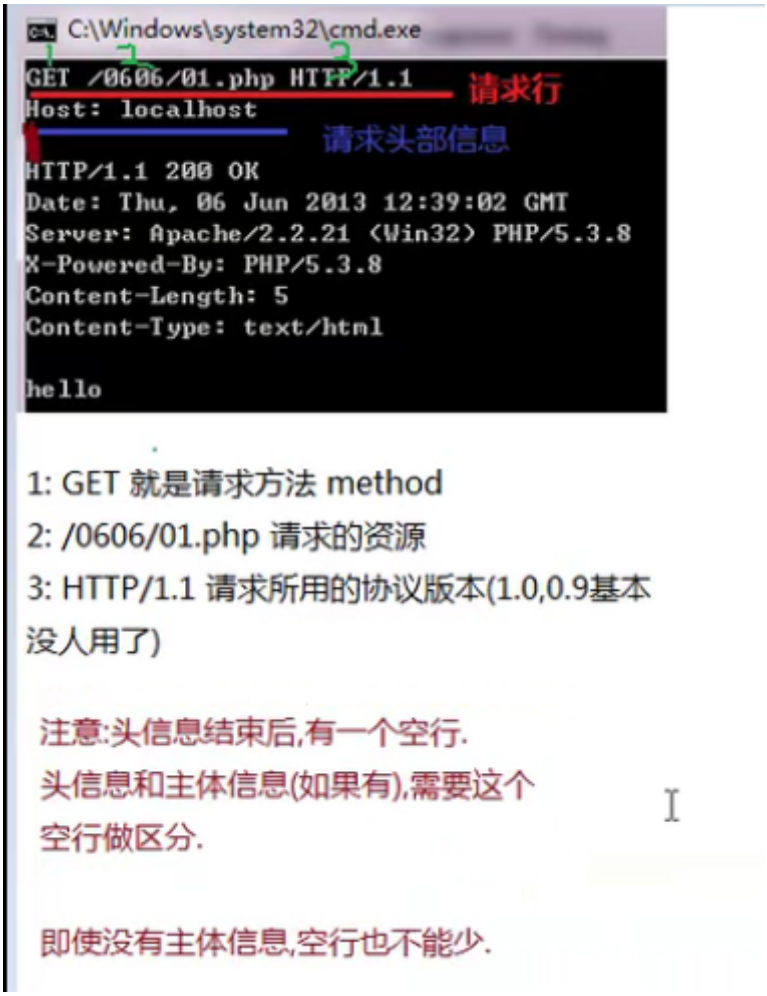
(1) 请求行

- 请求方法
  - GET、POST、PUT、DELETE、TRACE、OPTIONS
- 请求路径：就是URL的一部分
- 所用的协议：目前一般是HTTP1.1，0.9、1.0已经基本不用

(2) 请求头信息

(3) 请求主体信息（可以没有）

(4) 头信息结束后和主题信息之间要空一行

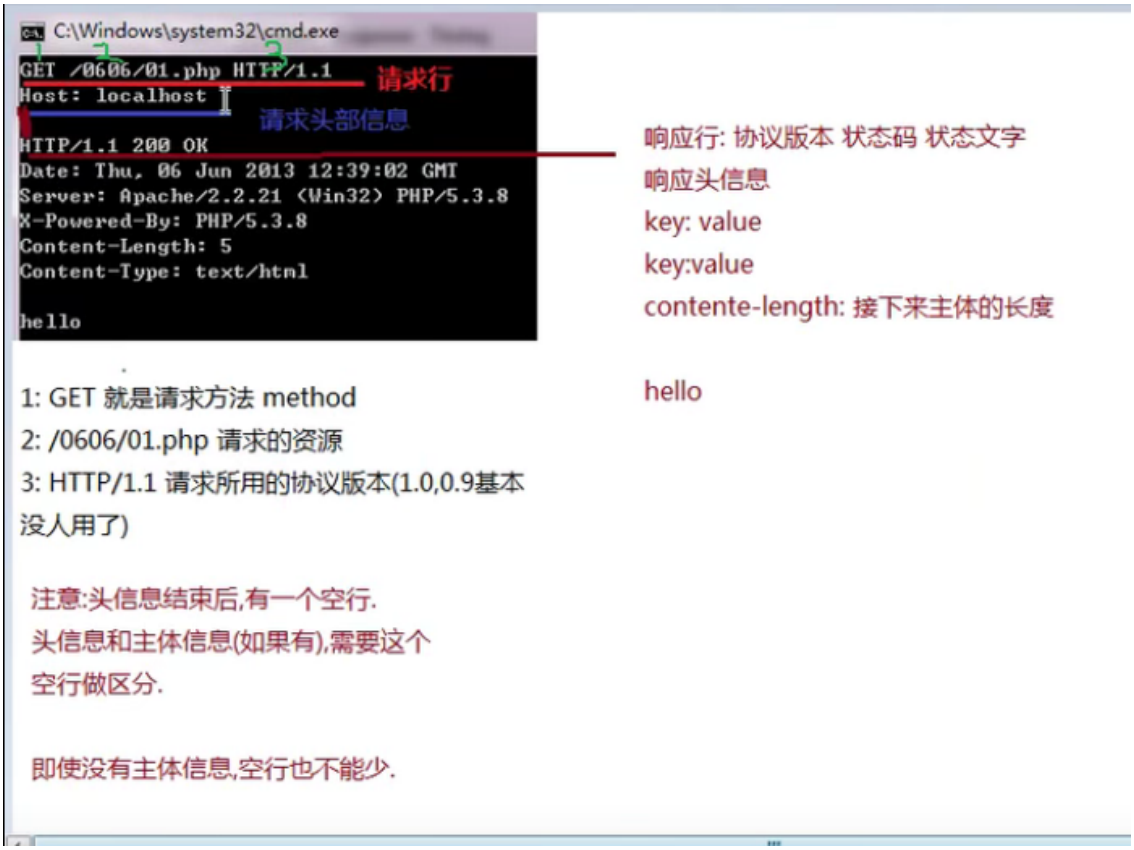


另：头信息是非常丰富的，虽然我们图中只写了一个。而且丰富的头信息，也是我们的一个学子重点。

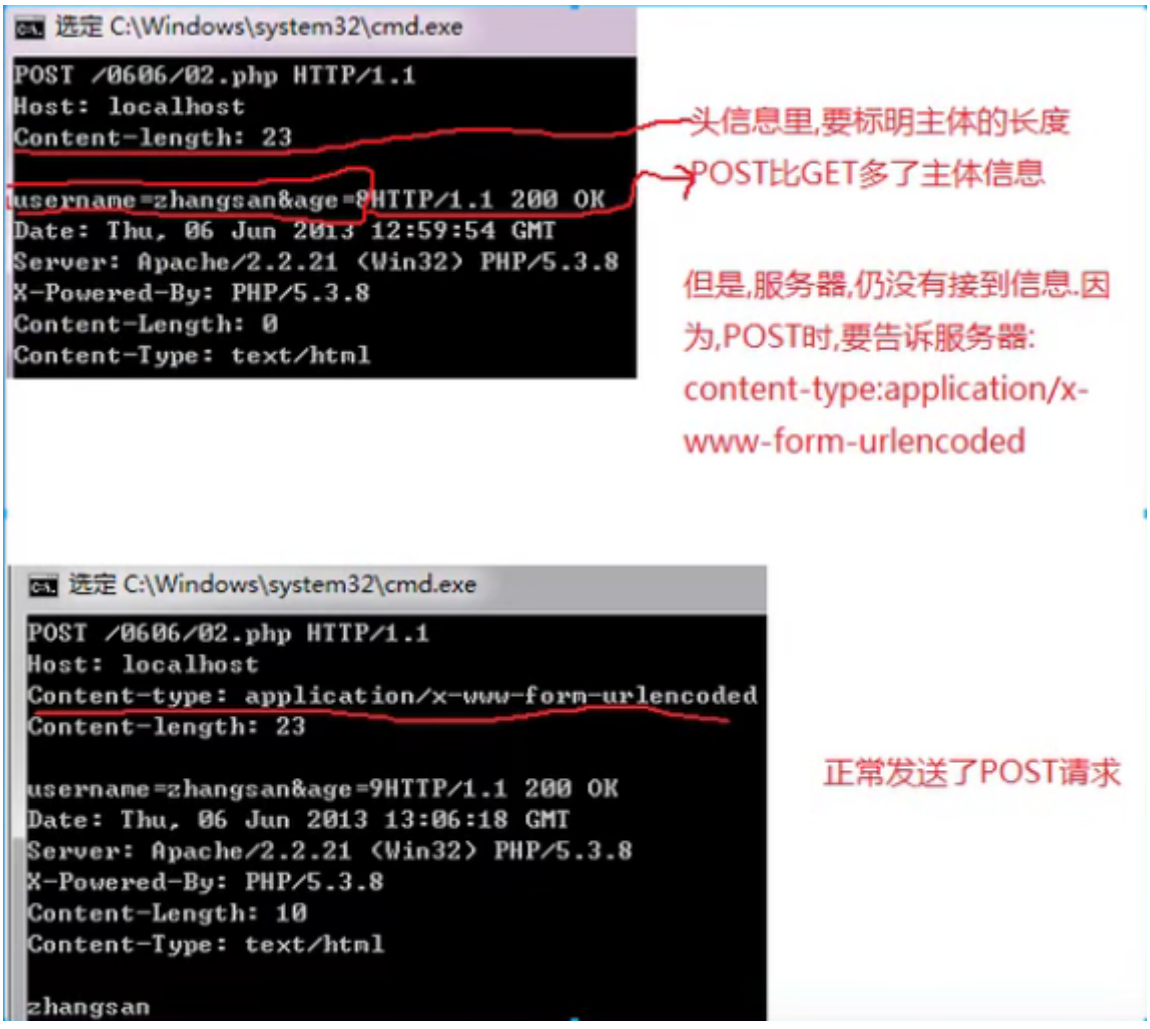
问：浏览器能发送HTTP协议，那么HTTP协议一定要浏览器来发送吗?

答：不是，HTTP既然是一种协议，那么只要满足这种协议，什么工具都可以发。

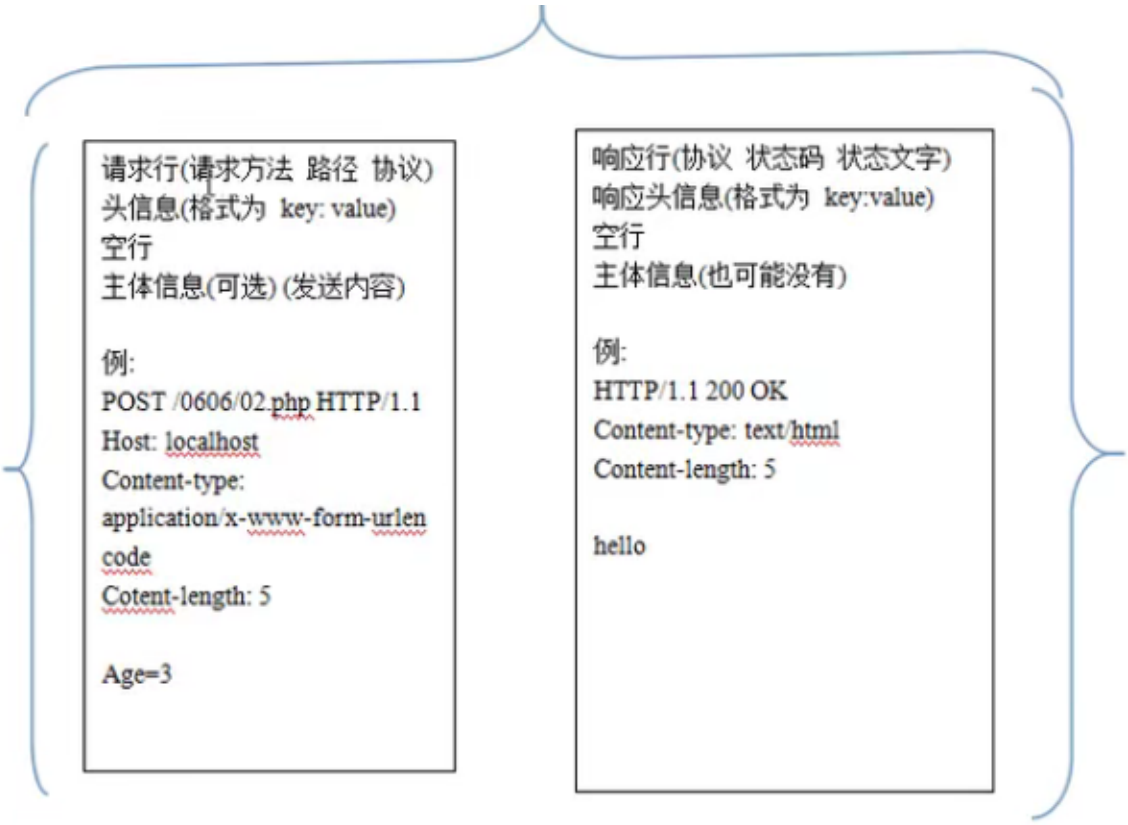
2. **响应：**


C:\Windows\system32\cmd.exe
GET /0606/01.php HTTP/1.1
Host: localhost
请求行
请求头部信息
HTTP/1.1 200 OK
Date: Thu, 06 Jun 2013 12:39:02 GMT
Server: Apache/2.2.21 (Win32) PHP/5.3.8
X-Powered-By: PHP/5.3.8
Content-Length: 5
Content-Type: text/html
hello
响应行: 协议版本 状态码 状态文字
响应头信息
key: value
key:value
contente-length: 接下来主体的长度
hello
1: GET 就是请求方法 method
2: /0606/01.php 请求的资源
3: HTTP/1.1 请求所用的协议版本(1.0,0.9基本没人用了)
注意:头信息结束后,有一个空行.
头信息和主体信息(如果有),需要这个空行做区分.
即使没有主体信息,空行也不能少.

POST请求:


选定 C:\Windows\system32\cmd.exe
POST /0606/02.php HTTP/1.1
Host: localhost
Content-length: 23
username=zhangsan&age=9HTTP/1.1 200 OK
Date: Thu, 06 Jun 2013 12:59:54 GMT
Server: Apache/2.2.21 (Win32) PHP/5.3.8
X-Powered-By: PHP/5.3.8
Content-Length: 0
Content-Type: text/html
头信息里,要标明主体的长度
POST比GET多了主体信息
但是,服务器,仍没有接到信息.因为,POST时,要告诉服务器:content-type:application/x-www-form-urlencoded
选定 C:\Windows\system32\cmd.exe
POST /0606/02.php HTTP/1.1
Host: localhost
Content-type: application/x-www-form-urlencoded
Content-length: 23
username=zhangsan&age=9HTTP/1.1 200 OK
Date: Thu, 06 Jun 2013 13:06:18 GMT
Server: Apache/2.2.21 (Win32) PHP/5.3.8
X-Powered-By: PHP/5.3.8
Content-Length: 10
Content-Type: text/html
zhangsan
正常发送了POST请求

2. HTTP协议之方法与状态码



2.1 请求方法

请求方法有哪些?

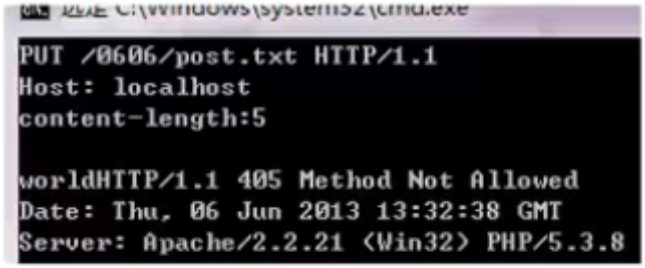
GET POST HEAD PUT TRACE DELETE OPTIONS..

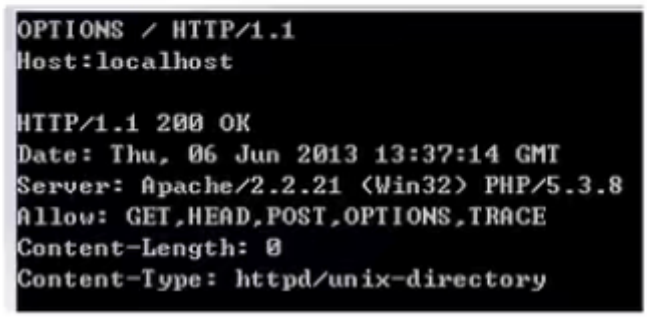
HEAD：和GET基本一致，知识返回内容。比如我们知识确认一个内容（比如照片）还正常存在，不需要返回照片内容。这时用HEAD比较合适。

TRACE：是你用了dialing上网，比如用代理方位news.163.com，你想看看代理哟就没有修改你的HTTP请求，可以用TRACE来测试一下，163.com的服务器就会把最后收到的请求返回给你。

OPTIONS：返回服务器可用的请求方法。

注意：这些请求方法虽然HTTP协议规定的，但是Web server 未必允许或支持这些方法。





2.2 状态码，状态文字

状态码：是用来反应服务器相应情况的。

最常见的如：200 OK，404 NOT FOUND

状态文字：是用来描述状态码的，便于人观察。

状态码	定义	说明
1XX	信息	收到请求，继续处理
2XX	成功	操作成功地收到，理解和接受
3XX	重定向	为了完成请求，必须采取进一步措施
4XX	客户端错误	请求的语法有错误或不能完全被满足。
5XX	服务端错误	服务器无法完成明显有效的请求。

状态码	说明
200	服务器成功返回网页
301/302	永久/临时重定向
304 Not Modified	未修改（表示取的缓存）
307	重定向中保持原有的请求数据

失败的状态码：

状态码	说明
404	请求的网页不存在
503	服务器暂时不可用
500	服务器内部错误

3. Socket编程发送请求

3.1 PHP + socket请求原理

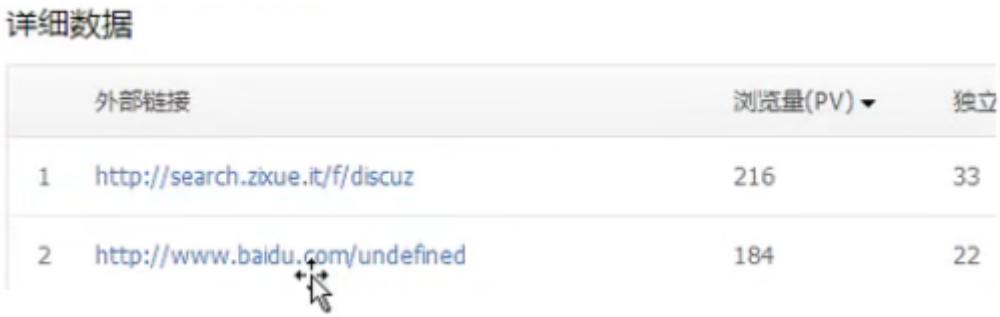
4. referer头与防盗链

像上图中的这个效果，当我们在网页里引入站外图片时，常出现这样的情况。

？？？服务器是怎么样知道这个图片实在站外被引用的呢?

？？？还有在网站的统计结果，统计用户从何而来?

？？？统计时，是如何得知用户从哪儿来到的本站呢?



)

在HTTP协议中，头信息里有一个重要的选项：Referer

Referer：代表网页的俩元，即上一页的网址

如果是直接在浏览器上输入地址，回来进来，则没有Referer头。

这也是：为什么服务器知道我们的图片是从哪儿引用的，也知道我们的客户是从哪个网站链接点寄过来的。

问题：如何配置Apache服务器，用于图片防盗链?

原理：在web服务器层面，根据HTTP协议的referer头信息，来判断。如果来自站外，则统一重写到一个很小的防盗链提醒图片上去。

具体步骤：

1. 打开Apache 重写模块 mode_rewrite

```
LoadModule rewrite_module modules/mod_rewrite.so
```

前面的"#"去掉，并重启Apache

2. 在需要放到的网站或目录，写 .htaccess文件，并指定防盗链规则。

如何指定?

自然是分析referer信息，如果不是来自本站，则重写

重写规则：

哪种情况 重写：

- 是jpeg/jpg/gif/png图片时
- 是referer头与localhost不匹配时

重写

怎么重写?

- 统一rewrite到某个防盗链图片

5.HTTP 协议与缓存控制

我们观察图片的下载，往往：

第1次请求时：200 ok

第2次请求时：304 Not Modified 未修改状态

解释：在网络上，有一些缓存服务器，另：浏览器自身也有缓存功能。

当我们第一次访问某图片时在，正常下载图片，返回值 200

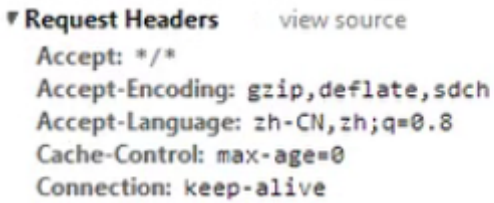
基于一个前提——图片不会经常改动，服务器在返回 200 的同时，还返回该图片的 ”签名“ ——Etag（也可以理解为图片的 ”指纹“）。

当浏览器再次访问该图片时，去服务器校验 ”指纹“，

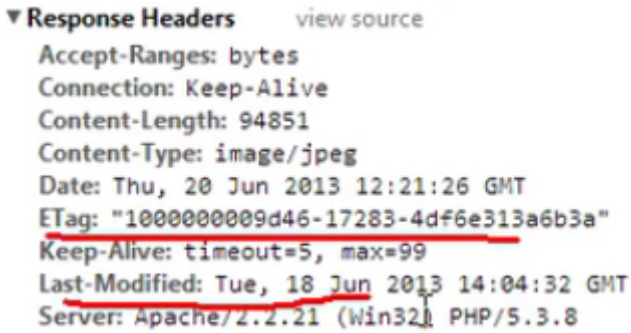
如果图片没有变化，直接使用缓存中的图片，这样减轻了服务器负担。

抓包观察

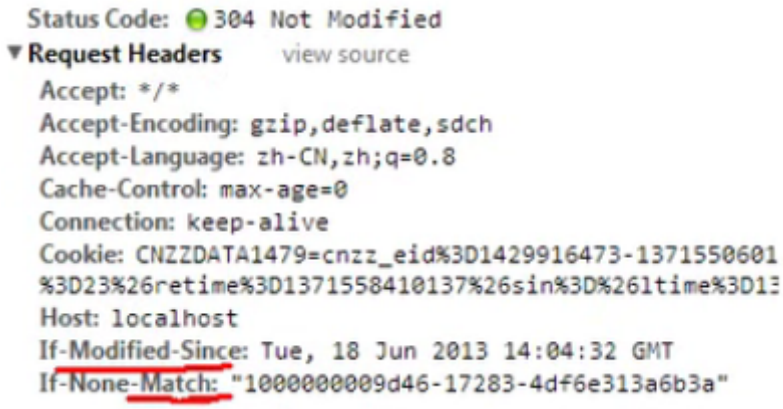
第一次请求头:



第一次响应头：



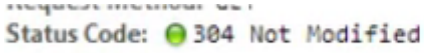
第二次请求头：



如果自"Tue, 18 Jun 2013 14:04:32 GMT"这个时间点以后，图片修改过，则重新请求。

如果该图片最新的Etag的值 和 If-None-Match的值不匹配，则重新请求。

第二次响应信息：



如果是304的话，就意味着浏览器从本地取缓存，节省了图片在网络上传输的时间。

选学：

如果网站比较大，有N台缓存服务器，那么这N台缓存服务器，如何处理主服务器上文件？

1. 要不要缓存？
2. 缓存多久？

思考：这说明缓存服务器与主服务器之间，应该有一些协议来说明这2个问题？

追问：用什么协议来说明这2个问题？

答：还是HTTP协议，用头信息 cache-control 来控制

具体用法：

在主服务器，打开 Apache 的 expires 扩展，利用该扩展来控制图片、css、html等文件，控制是否缓存及缓存生命周期。

在 .htaccess中，具体语法如下：

ExpiresDefault ”<base> [plus] {<num> <type>}*"

ExpiresByType type/encoding "<base> [plus] {<num> <type>}*"

ExpiresDefault 是设置默认的缓存参数

ExpiresByType 是按照文件类型来设计独特的缓存参数

我们用第二种来做测试，给jpg图片设置1个月的生存周期。

后面4个参数怎么理解?

1. Base：基于哪个时间点来计算缓存有效期
    - Access/now：基于请求响应的那一瞬间，比如从此瞬间到1个月之后。
    - Modification：基于被请求文件的最后修改日期来计算。比如 最后修改日期的后一周内仍然有效。
2. Num：缓存时间的大小（30）
3. Type：缓存时间的单位：（天）

实例：

```
ExpiresActive On
ExpiresByType image/jpeg "access plus 30 days"
```

如果这是在集群环境里，缓存服务器得到此图片，将会认为一个月内有效。减轻了主服务器的负担

我们能否设置服务器，不让用缓存呢?

比如有些个人信息不允许缓存服务器缓存，必须到主服务器去请求。

Control-cache:no-store，must-revalidate; //这意味着不允许缓存，必须要去主服务器验证。

可以利用Apache 的header模块

```
ExpiresActive On
ExpiresByType image/jpeg "access plus 30 days"

<FilesMatch "\.(gif)$">
header set Cache-Control "no-store,must-revalidate"
</FilesMatch>
```

针对gif图，专门不缓存

▼ Response Headers view source
Accept-Ranges: bytes
Cache-Control: no-store,must-revalidate

多次刷新页面，发现

上例可以看出，girl.gif不允许缓存，因此每次都重新请求。

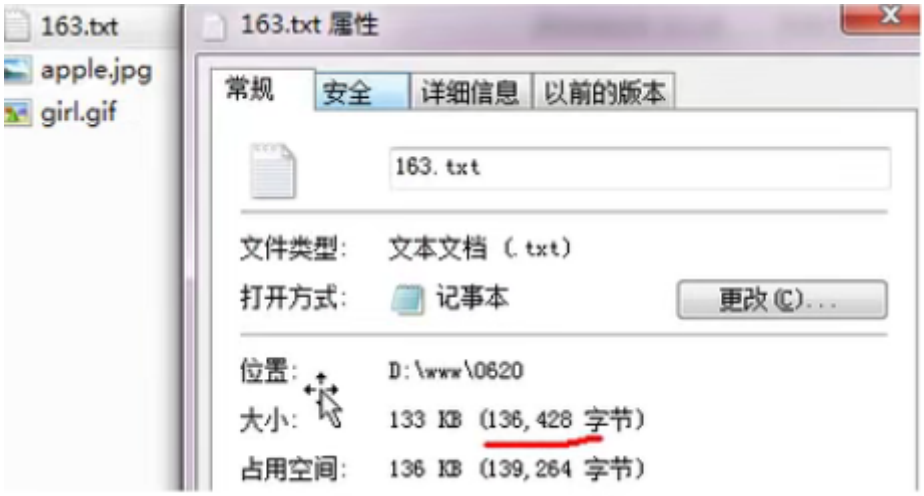
6. HTTP协议与内容压缩

观察：我们打开163的一篇新闻

看到如下响应头信息，注意 Content-Length:

▼ Response Headers view source
Age: 176
Cache-Control: max-age=240
Content-Encoding: gzip
Content-Length: 36187

同时，我们点击右键保存其源码，得到的文本文件大小



思考：Content-Length在之前的学习中，，代表返回的主体的长度，但此处，为什么返回的主体长度和content-length不一致呢?

原因在于：Content-Encoding：gzip 这个响应头信息在作用。

原理：为了提高网页在网络上的传输速度，服务器对主体信息进行压缩。

如常见的 gzip压缩、deflate压缩、compress压缩以及Google chrome 正在推的sdch压缩。

压缩过程：

刚才那个情况的原因——服务器对页面进行了压缩 content-length 是"压缩后"的长度。

如何在Apache启用压缩功能?

```
1开启 deflate 模块, 或 gzip 模块
2: 在 conf 文件中,写如下代码
<ifmodule mod_deflate.c>
DeflateCompressionLevel 6   # 压缩级别为 6, 可选 1-9,推荐为 6
AddOutputFilterByType DEFLATE text/plain # 压缩文本文件
AddOutputFilterByType DEFLATE text/html # 压缩 html 文件
AddOutputFilterByType DEFLATE text/xml # 压缩 xml
</ifmodule>
```

为什么要指定文件类型来压缩?

答：压缩也是要耗CPU资源的，图片/视频等文件压缩效果也不好，一半压缩文本格式。

压缩前:

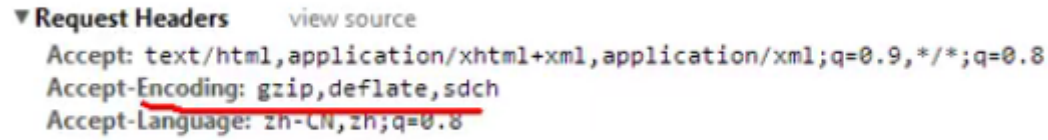
压缩后:

通过上面对比,节省了 40%的流量.
设某大型门户 10 亿 PV,平均页面大小 10000 字节

每天节省流量 10 亿*10000*0.4 === 4000G

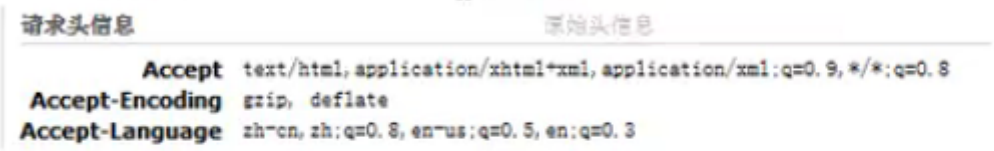
问：服务器怎么知道我们的浏览器支持gzip的?

答: 客户端允许发一个 Accept-Encoding 头信息,与服务器协商.
如:



这个例子可以看出,chrome 浏览器支持 gzip,deflate,sdch3 种压缩方式

再用 firefox 器做测试 ,可以看出 ff 只支持 gzip,deflate 压缩方式.



小技巧: 当我们在采集时,可以不发送 Accept-Encoding 信息,这样采集直接是源码.
当然,也可以采集 gzip(提高速度), 再用 PHP 解压 gzip 的内容.