http消息中的请求头详解

概览：

HTTP请求方式	主要使用到“GET”和“POST”
Host	请求的web服务器域名地址
User-Agent	请求的web服务器域名地址
Accept	指定客户端能够接收的内容类型
Accept-Language	指定HTTP客户端浏览器用来展示返回信息所优先选择的语言
Accept-Encoding	指定客户端浏览器可以支持的web服务器返回内容压缩编码类型
Accept-Charset	浏览器可以接受的字符编码集
Content-Type	显示此HTTP请求提交的内容类型
Connection	表示是否需要持久连接
Keep-Alive	显示此HTTP连接的Keep-Alive时间
cookie	请求时，会把保存在该请求域名下的所有cookie值一起发送给服务器
Referer	包含一个URL，用户从该URL代表的页面出发访问当前请求的页面

1、HTTP请求方式

如下表：

GET	向Web服务器请求一个文件
POST	向Web服务器发送数据让Web服务器进行处理
PUT	向Web服务器发送数据并存储在Web服务器内部
HEAD	检查一个对象是否存在
DELETE	从Web服务器上删除一个文件
CONNECT	对通道提供支持
TRACE	跟踪到服务器的路径
OPTIONS	查询Web服务器的性能

说明：

主要使用到“GET”和“POST”。

实例：

POST /test/tupian/cm HTTP/1.1

分成三部分：

（1）POST：HTTP请求方式

（2）/test/tupian/cm：请求Web服务器的目录地址（或者指令）

（3）HTTP/1.1: URI（Uniform Resource Identifier，统一资源标识符）及其版本

备注：

在Ajax中，对应method属性设置。

2、Host

说明：

请求的web服务器域名地址

3、User-Agent

说明：

HTTP客户端运行的浏览器类型的详细信息。通过该头部信息，web服务器可以判断到当前HTTP请求的客户端浏览器类别。

实例：

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11

4、Accept

说明：

指定客户端能够接收的内容类型，内容类型中的先后次序表示客户端接收的先后次序。

例如：

Accept:text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

备注：

在Prototyp（1.5）的Ajax代码封装中，将Accept默认设置为“text/javascript, text/html, application/xml, text/xml, */*”。这是因为Ajax默认获取服务器返回的Json数据模式。

在Ajax代码中，可以使用XMLHttpRequest 对象中setRequestHeader函数方法来动态设置这些Header信息。

5、Accept-Language

说明：

指定HTTP客户端浏览器用来展示返回信息所优先选择的语言。

实例：

Accept-Language: zh-cn,zh;q=0.5

这里默认为中文。

6、Accept-Encoding

说明：

指定客户端浏览器可以支持的web服务器返回内容压缩编码类型。表示允许服务器在将输出内容发送到客户端以前进行压缩，以节约带宽。而这里设置的就是客户端浏览器所能够支持的返回压缩格式。

实例：

Accept-Encoding: gzip,deflate

备注：

其实在百度很多产品线中，apache在给客户端返回页面数据之前，将数据以gzip格式进行压缩。

7、Accept-Charset

说明：

浏览器可以接受的字符编码集。

实例：

Accept-Charset: gb2312,utf-8;q=0.7,*;q=0.7

8、Content-Type

说明：

显示此HTTP请求提交的内容类型。一般只有post提交时才需要设置该属性。

实例：

Content-type: application/x-www-form-urlencoded;charset:UTF-8

有关Content-Type属性值可以如下两种编码类型：

（1）“application/x-www-form-urlencoded”： 表单数据向服务器提交时所采用的编码类型，默认的缺省值就是“application/x-www-form-urlencoded”。 然而，在向服务器发送大量的文本、包含非ASCII字符的文本或二进制数据时这种编码方式效率很低。

（2）“multipart/form-data”： 在文件上载时，所使用的编码类型应当是“multipart/form-data”，它既可以发送文本数据，也支持二进制数据上载。

当提交为单单数据时，可以使用“application/x-www-form-urlencoded”；当提交的是文件时，就需要使用“multipart/form-data”编码类型。

在Content-Type属性当中还是指定提交内容的charset字符编码。一般不进行设置，它只是告诉web服务器post提交的数据采用的何种字符编码。

一般在开发过程，是由前端工程与后端UI工程师商量好使用什么字符编码格式来post提交的，然后后端ui工程师按照固定的字符编码来解析提交的数据。所以这里设置的charset没有多大作用。

9、Connection

说明：

表示是否需要持久连接。如果web服务器端看到这里的值为“Keep-Alive”，或者看到请求使用的是HTTP 1.1（HTTP 1.1默认进行持久连接），它就可以利用持久连接的优点，当页面包含多个元素时（例如Applet，图片），显著地减少下载所需要的时间。要实现这一点， web服务器需要在返回给客户端HTTP头信息中发送一个Content-Length（返回信息正文的长度）头，最简单的实现方法是：先把内容写入ByteArrayOutputStream，然 后在正式写出内容之前计算它的大小。

实例：

Connection: keep-alive

10、Keep-Alive

说明：

显示此HTTP连接的Keep-Alive时间。使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，Keep-Alive功能避免了建立或者重新建立连接。

以前HTTP请求是一站式连接，从HTTP/1.1协议之后，就有了长连接，即在规定的Keep-Alive时间内，连接是不会断开的。

实例：

Keep-Alive: 300

11、cookie(max-age)

说明：

HTTP请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。

12、Referer

说明：

包含一个URL，用户从该URL代表的页面出发访问当前请求的页面