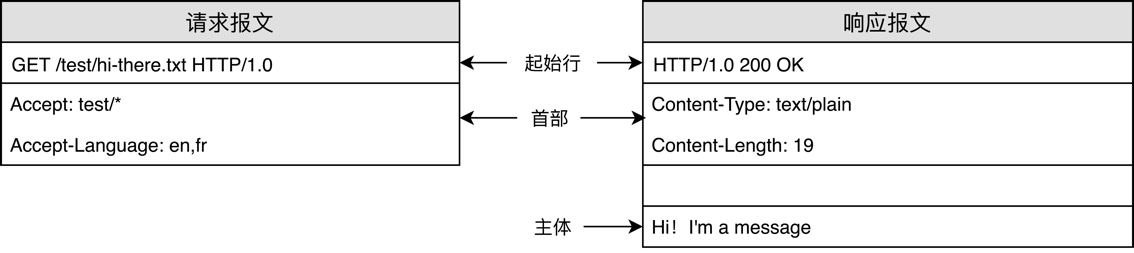
# HTTP概述

HTTP使用的是可靠的数据传输协议。所有的内容都是Web资源；HTTP协议中会在返回的数据上打上MIME（Multipurpose Internet Mail Extension）标签，用来表示媒体的类型，最初是用在电子邮件系统中；也就是Content-Type与Content-length；MIME由对象的类型/子类型构成，比如image/jpeg；服务器资源名被称为统一资源标识符，Uniform Resource Identifier（URI）URI有2种形式：URL与URN；URL是同一资源定位符，URL的构成：方案（协议）+服务器地址（地址+端口号）+资源地址；URN统一资源名。

http事务指的是一次请求及其响应。请求中包含命令也就是HTTP方法，返回中包含响应码与返回短语用于描述，一个web页面包含多个HTTP事务；HTTP报问请求的格式如下，HTTP报文是由简单的字符串组成，



HTTP基于TCP协议，TCP提供无差错的数据传输，按序传输，未分段的数据流；HTTP交互步骤：

* 浏览器从URL中解析出服务器的主机名；
* 根据主机名到DNS解析出IP地址；
* 解析端口号，没有就是80默认；
* 根据IP：Port建立TCP连接；
* 发送HTTP请求报文；
* 获取HTTP响应报文；
* 连接关闭，显示资源。

telnet可以建立TCP客户端，在里面可以模拟HTTP交互。

HTTP/0.9 HTTP/1.0 HTTP/1.0+ HTTP/1.1 HTTP/NG。

* 代理：客户端与服务器之间的中间实体，代替用户，作为转发所有web流量的可信任的中间节点使用；
* 缓存：代理缓存；
* 网关：相对代理来说的，是服务器端的代理，通常用于将HTTP请求转发成其他协议的请求，用户访问的真正的服务器可能就是网关，网关之后的服务器才是资源真正存储的地方；
* 隧道：没看懂，对数据进行盲转发；
* Agent代理：HTTP客户端程序。

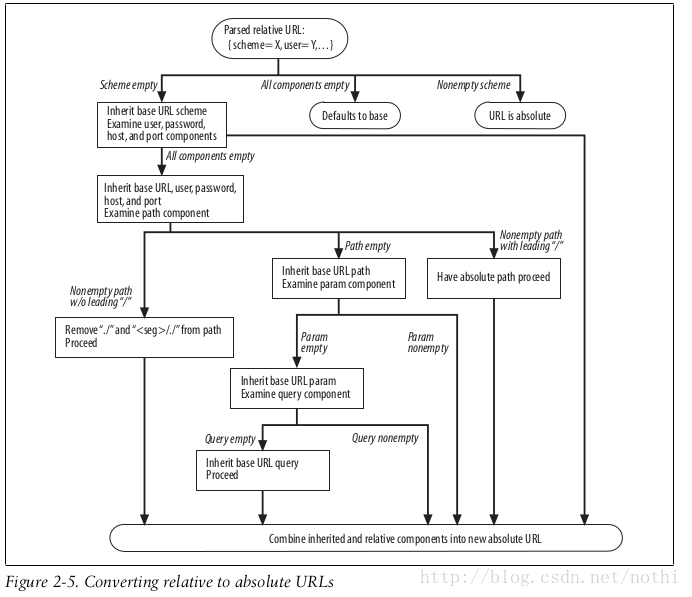
# URL与资源

URL就是因特网资源的标准化名称，URL指向一条电子信息片段。URI包含URL与URN，URL表示的是资源的位置，URN表示的是资源的名字；URL提供了统一的资源命名方式，可以通过http/ftp等获得资源；结构：方案://服务器地址/路径；URL格式：

<schema>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>;

Params是名/值对，使用“;”分隔，每一个路径块内都可以有参数，query也是参数，使用&分隔的名/值对，frag片段指定返回资源的某部分。其余部分不说；（ftp协议中，如果没有指定用户名/密码，用户名=anonymous，密码看各个浏览器）。

URL有绝对URL预相对URL，相对URL有一个基础URL，比如出现在HTML中的”./xxx.html”样式的URL是基于地址栏的URL的；浏览器负责解析相对URL；<BASE>或者本身资源的URL都可以作为基础URL；转换算法如下：



URL包含关键字的处理机制是转义，转移符号是%；当碰到不安全的字符后，使用%XX表示字符的16进制形式的字节，就是安全的编码机制，表示的就是URL，而不是什么特殊的字符等；包括很多复杂的字符。



上图中是关键字的转义规则。

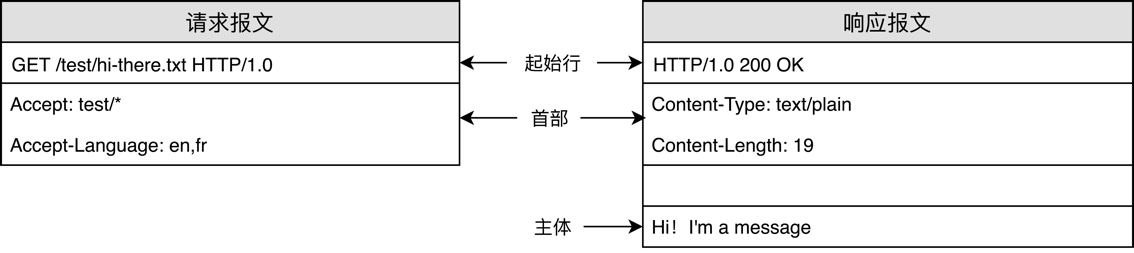
# HTTP报文

## 3.1 报文流

HTTP报文是HTTP应用程序之间发送的数据块，这些数据库以文本形式的元信息开头，描述了报文的内容与含义，后面跟着可选的数据部分。

## 3.2 报文的组成部分

HTTP报文是简单的格式化数据块；起始行与首行是行分隔的ASCII文本，每行通过回车换行符终止；主体是可选的，可以是文本也可以是二进制数据。



请求报文的语法：

<method> <request-URL> <version>

<headers>

<entity-body>

响应报文的语法：

<version> <status> <reason-phrase>

<headers>

<entity-body>

* 起始行：请求报文的起始行说明要做什么，响应报文：发生了什么；
* 响应行：状态信息；
* 方法：告知服务器要做什么。

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 是否包含主体 |
| GET | 获取文档 | 否 |
| HEAD | 获取文档的首部 | 否 |
| POST | 发送需要处理的数据 | 是 |
| PUT | 将请求的主体部分存储在服务器上 | 是 |
| TRACE | 对报文跟踪 | 否 |
| OPTIONS | 服务器上可以执行的方法 | 否 |
| DELETE | 删除一份文档 | 否 |

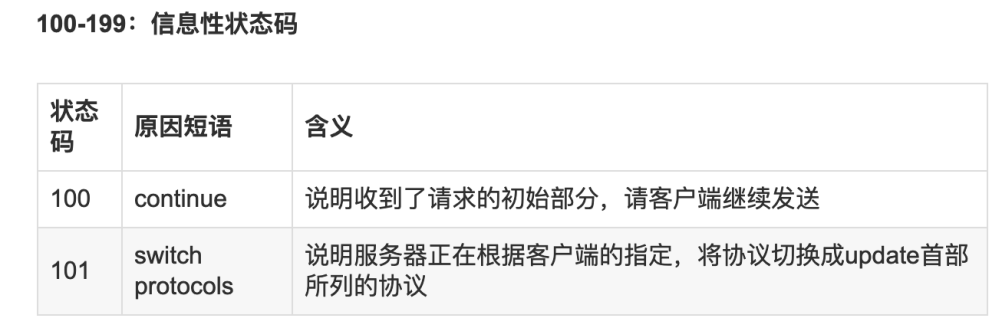
* 状态码：发生了什么事情；
* 原因短语：
* 首部：通用首部、请求首部、响应首部、实体首部、扩展首部；
* 主体：

## 3.3 方法

安全方法：get/head就是请求不会在HTTP服务器上产生什么结果；get请求资源，head只返回get的首部（了解资源情况、查看对象是否存在、资源是否修改了），put方法写入或者更新资源，POST常用与提交表单；http请求会经过中间节点，请求会被篡改，trace会将最终送到服务器的请求原文返回，以观察篡改的情况；扩展方法：对所发送的内容严一些，对所接收的内容宽容一些。

## 3.4 状态码

* 100-199：信息性状态码



100必须用在请求有实体的情况下，此时客户端使用带有100 continue的Expect首部发送，服务端返回100，此时客户端可以发送请求实体；

* 200-299：成功状态码，



* 300-399: 重定向状态码；



* 400-499：客户端错误状态码；



* 500-599：服务器错误状态码；



## 3.5 首部

通用首部：Connection、Date、MIME-Version、Trailer、Transfer-Encoding、Update、Via、Cache-Control、Pragma；

请求首部：给出客户端信息，Client-IP、From、Host、Referer、UA-Color、UA-CPU、UA-Disp、UA-OS、UA-Pixels、User-Agent、Accept、Accept-Charset、Accept-Encoding、Accept-Language、TE、条件请求首部：Expect、If-Match、If-Modified-Since、If-None-Match、If-Range、If-Unmodified-Since、Range、安全请求首部：Authorization、Cookie、Cookie2、代理请求首部：Max-Forward、Proxy-Authorization；

响应首部：用于返回服务端信息并告知客户端一些命令信息，以便客户端与服务器进行后续的良好交互；Age、Public、Retry-After、Server、Title、Warning、协商首部：vary、Accept-Ranges、安全响应首部：Proxy-Authenticate、Set-Cookie、WWW-Authenticate；

实体首部：Allow、Location、Content-Base、Content-Encoding、Content-Languages、Content-Length、Content-Location、Content-Type、ETag、Expires、Last-Modified。

# 第4章 连接管理

## 4.1 TCP链接

HTTP协议基于TCP连接，提供可靠的byte传输通道；TCP流是分段的，由IP数据报传送。

## 4.2 对TCP性能的考虑

HTTP事务时延主要是由TCP网络时延构成的，比如解析DNS、建立连接、传输数据等；

## 4.3 HTTP连接的处理

Connecttion首部的作用（关闭连接，删除一些首部信息）。

一个TCP连接传输HTTP报文，尽量保证HTTP事务的并行化，有几种方式：1.并行连接（多条TCP连接发起并发的HTTP请求）；2.持久连接（重用TCP连接，消除连接与关闭时延）；3.管道化连接（通过一个TCP发起并发的HTTP请求）等。

## 4.4 并行连接

HTTP并发连接，重用TCP，提高页面的加载速度。

## 4.5 持久连接

HTTP资源具有站点局部性，HTTP事务处理结束后仍保持打开的TCP连接称为持久连接；现在都会打开几个并行连接，每个连接都是持久连接；持久连接的2种类型：HTTP/1.0的keep-alive与HTTP/1.1的persistent；

Keep-alive：客户端发送请求中包含Connection: keep-alive，服务器接收到后如果愿意保持打开状态，则返回首部，没有，客户端接收到后主动关闭连接；可以通过keep-alive首部来调节keep-alive的行为，比如timeout、max等。

Keep-alive与哑代理：主要的问题原因是代理不理解Connection首部。解决的方案是代理不要转发connnection首部及首部中列出的首部（Connection首部是一个逐跳首部，只适用单条传输链路，不应该向下传输）。

Proxy-Connection首部可以解决哑代理的问题，但是还是没办法解决多个代理同时作用的问题；在HTTP/1.1版本中，任何连接缺省都是长连接，除非显示的添加报文Connection:close。

## 4.6 管道化连接

可以在持久连接上使用请求管道；在响应达到前，将多条请求放入队列；依次发出，相当于并发发出请求，但是请求的返回顺序必须与发送的顺序一致；且必须要求是持久连接。

## 4.7 关闭连接的奥秘

Content-Length描述主体尺寸，如果没有指定此header，需要连接关闭来说明数据的真实末尾；客户端不应该以管道化的方式传输非幂等的请求。

# 第5章 Web服务器

## 5.1 各种形状和尺寸的Web服务器

Web服务器就是对HTTP请求进行处理并返回响应；通用Web服务器软件，Web服务器设备，嵌入式web服务器。

## 5.2 最小的Perl Web服务器

## 5.3 实际的web服务器会做些什么

建立连接、接收请求、处理请求、访问资源、构建响应、发送响应、记录事务处理过程。

## 5.4 接受客户端连接

TCP建立连接。

## 5.5 接收请求报文

## 5.8 构建响应

Web服务器可能会返回重定向返回码3XX，Location首部包含了内容的新地址；重定向用于：永久搬离的资源；临时搬离的资源、URL增强、复杂均衡等；

# 第6章 代理

## 6.1 Web的中间实体

Web上的代理服务器是代表客户端完成事务处理的中间人，代理既是客户端优势服务端；公有代理是众多客户端共享的，私有代理是某个客户端专用的，代理代理的是相同协议的请求，网关则是不同协议的端点，负责进行协议转换。

## 6.2 为什么使用代理

代理可以改善安全性，提高性能，监视流量并修改；具体的作用有：1.过滤URL；2.访问控制；3.防火墙；4.Web缓存；5.反向代理（）；6.内容路由器；7.转码器；8.匿名者；

## 6.3 代理会去往何处

部署代理服务器的几种方式：1.出口代理；2.入口代理；3.反向代理；4.网络交换代理。

代理可以含有层次，越靠近服务器的代理被称为父代理，靠近客户端的是子代理，根据代理的内容路由的特性，可以子代理可以根据请求的特性发送给一些特定的父代理或者直接到服务器；

内容路由的使用场景有：负载均衡，地理位置附近的代理，协议/类型路由。

代理获取客户端流量的方式：修改客户端，让其直接发送到代理服务器；通过路由器拦截；修改DNS的命名空间，Web服务器重定向。

## 6.4 客户端的代理设置

* 显式的设置要使用的代理：浏览器特性；比如firefox浏览器



* 预先配置浏览器：
* 代理的自动配置（Proxy Auto-Configuration,PAC）：URI，指向的是PAC文件，根据PAC中的函数计算结果决定代理设置；
* WPAD的代理发现协议（Web Proxy Autodicovery Protocol），自动检测出浏览器可以从哪个配置服务器下载到一个自动配置文件；根据WPAD协议找到PAC文件。

## 6.5 与代理请求有关的一些棘手问题

* 代理URI与服务器URI的不同：直接向服务器发与向代理发，只有URI不同，其他一样，向服务器发，只有URI，向代理发，要包括完整URI，因为代理需要知道目标服务器的信息；
* 虚拟主机web服务器，也需要知道目标服务器的信息，此时使用Host首部获得；

# 缓存

缓存，本地保存副本，减少冗余数据传输，解决带宽不足，降低服务器要求，降低距离时延；

## 7.1 冗余的数据传输

有些数据只需要加载一次就够了，重复加载耗尽带宽，影响服务器。

## 7.2 带宽瓶颈

解决带宽不足，因为距离最近的速度最快，网络传输总是很慢的。

## 7.3 瞬间拥塞

热点事件文档的瞬发请求会特别多，造成服务器崩溃，此时缓存非常有效。

## 7.4 距离时延

资源距离太长，时延也会增大，缓存比较近。

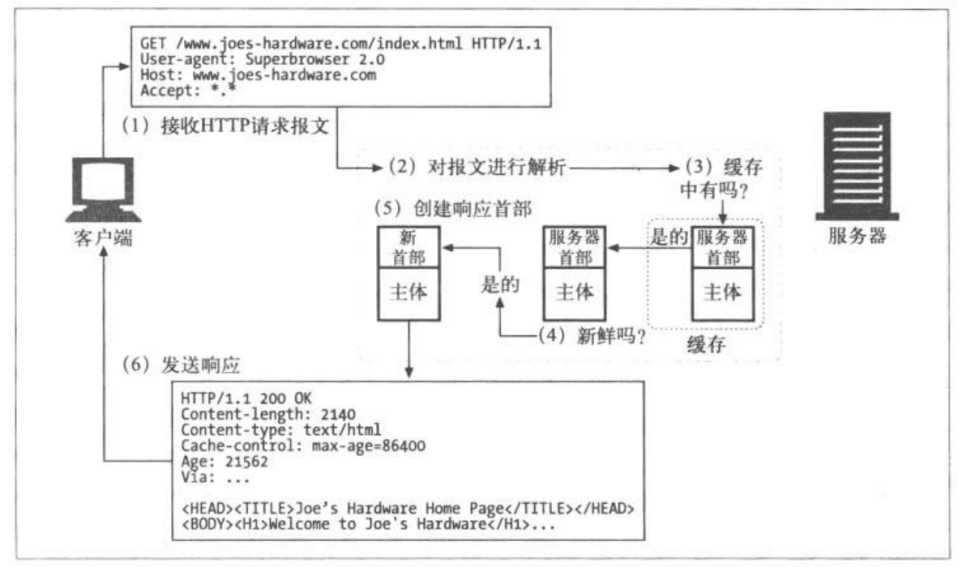
## 7.5 命中和未命中的

缓存分为命中与未命中；还需要对缓存进行新鲜度检测；向服务器发起验证，服务器会返回304或者最新额网络资源；称为再验证命中。

## 7.6 缓存的拓扑结构

私有缓存、共有缓存(代理服务器缓存，减少流量)，层次缓存，网状缓存。

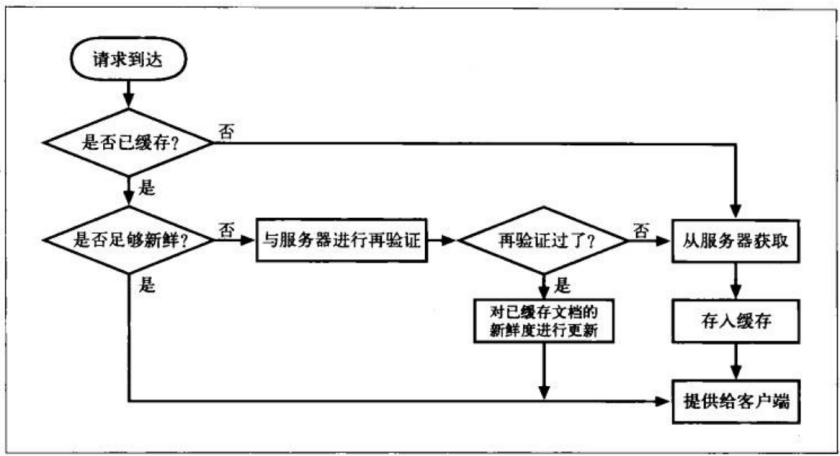
## 7.7 处理步骤



检测网络输入；解析HTTP输入；根据URL查看缓存信息；新鲜度检测；创建响应并发送。

## 7.8 保持副本的新鲜

文档过期与服务器再验证；Expires（1.0）与Cache-Control（1.1）首部中的max-age；可以指定缓存的新鲜度，在这个日期内，都会使用缓存；过期后，会要求进行新鲜度检测；



服务器再验证只会发生在文档过期之后，如果验证没改变，则获取新的过期时间设置的等首部；但没有文档；否则一次请求；http通过向头部里面放入缓存条件形成条件GET完成缓存验证与获取对象；条件为真，则直接获取资源；条件为假，则返回304，并更新过期日期。

* If-Modified-Since:<date>：如果指定日期文档被修改，则获取资源与Last-Modified首部配合使用；
* If-None-Match:<tags>：标签不同，则获取资源；文档周期性重写，修改不重要，无法判定最后日期，亚秒内发生变化，导致根据日期验证不能发挥作用；使用ETag（实体标签）标示文档的版本；弱验证器、强验证器；

## 7.9 控制缓存的能力

* Pragma:no-cache, HTTP/1.0G规范，禁止缓存；
* Cache-Control:no-store:禁止缓存，响应后删除对象；
* Cache-Control:no-cache:每次使用缓存都要进行新鲜度检测；
* Cache-Control:max-age=3600:响应存在的最大的秒数；
* Cache-Control:s-maxage=3600:共享缓存响应存在的最大的秒数；
* Expires:<date>:指定过期日期；
* Must-revalidate:过期后必须验证，不能提供陈旧的副本；
* 如果响应没有带有任何的缓存信息，浏览器或者web会给相应设定一个默认的过期间隔，这个间隔叫做试探性过期时间，不会超过24小时，通常取获取相应的与文档的最后修改事件的差值的一部分；也就是LM-factor算法；
* 浏览器也能在GET请求中加入一些缓存首部，控制是否获取缓存等；首部见书上；

## 7.10 设置缓存控制

## 7.11 详细算法

缓存是否新鲜：使用期<新鲜生存期；

## 7.12 缓存与广告

# 第8章 集成点：网关、隧道及中继

## 8.1 网关

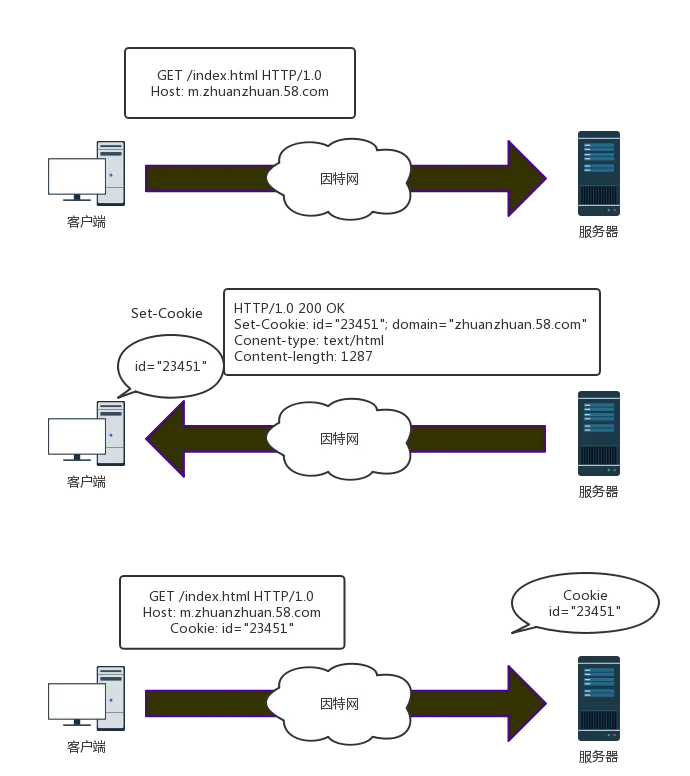
# 第11章 客户端识别与cookie机制

## 11.6 cookie

cookie是识别当前用户，实现持久会话的最好方式，非常重要，通常要与一些技术联合使用。cookie的内容不允许缓存，cookie分为2类:

* + 会话cookie，临时cookie，生命周期是浏览器关闭
  + 持久cookie，存储在硬盘上，关闭或者重启都在，频繁访问的站点可以用持久cookie

区别就是过期时间，如果设置了Discard参数或者没有设置Expires/Max-Age参数的cookie就是会话cookie。



cookie的工作过程如上图。cookie中可以包含任意的信息，浏览器会记住从服务器返回的Set-Cookie/Set-Cookie2首部的内容，把cookie存储在浏览器的cookie数据库中，然后再次发起同一个站点的请求时通过cookie首部携带这些信息。

cookie的基本思想就是让浏览器积累一组服务器特有的信息，每次访问服务器时将这些信息提供给它。因为时浏览器存储所以也叫客户端侧状态，cookie规范的正式名称是HTTP状态管理机制(HTTP state management machanism)。

浏览器只向服务器发送服务器产生的那些cookie，这是因为如果都发送cookie内容比较多、多余的cookie可能没什么用也不会倍识别，同时还涉及到隐私问题，cookie的domain属性用于控制哪些站点可以看到这个cookie，Path用来规定cookie匹配的路径前缀，路径前缀匹配的cookie才是可以看到的。

cookie分2个版本:

* 版本0也就是Netscape版本，定义的cookie格式如下:

Set-Cookie: name=value[; expires=date][; path=path][; domain=domain][; secure]

Cookie: name1=value1[; name2=value2]

name=value是强制的，特殊字符要包含在双引号内，否则特殊字符会被忽略， expires=date指定过期日期，日期的格式是: Weekday, DD-Mon-YY HH:MM:SS GMT如果没有指定，则cookie就会在用户会话结束时过期，domain就是服务器指定的cookie属于的域，没有设置则是产生Set-Cookie响应的服务器的主机名，Path就是为服务器上特定的文档分配cookie，Path是URL路径前缀，如果没有指定则是设置Set-Cookie响应的URL的路径，secure指定只用使用https的时候才会发送cookie。

* 版本1也是版本0的扩展版本，引入了Set-Cookie2与Cookie2首部，

改动包括：cookie添加解释性文本，允许浏览器退出时不考虑过期时间强制销毁cookie，用相对秒数而不是绝对日期来表示cookie的Max-Age，通过URL端口号而不仅仅是域和路径来控制cookie，Cookie首部添加域、端口与路径信息，版本号信息，$关键字。比cookie0多出来的字段：

* Version，版本号，新的是1
* Comment，
* CommentURL
* Discard，指示是会话cookie
* Max-Age，设置以秒为单位的cookie生存周期，超过这个值，则应该将cookie丢弃，值为0说明应该立即将cookie丢弃;
* Port，端口号

Cookie2首部负责在能够理解不同cookie规范版本的客户端和服务器之间进行互操作性的协商。Cookie2告知服务器，客户端支持cookie新的版本，如果服务器支持新的cookie版本协议，识别出Cookie2首部，返回的响应中就会包含Set-Cookie2首部而不是Set-Cookie，如果客户端接收到Set-Cookie与Set-Cookie2，则忽略Set-Cookie。

可以使用cookie对用户进行跟踪，电子商务网站用会话cookie在用户浏览时记录下用户的购物车信息。缓存cookie相关的首部信息要特别注意。Cookie与缓存并没有特定的关系，如果除了Set-Cookie首部之外的文档是可缓存的，那么可以设置

Cache-Control: no-cache=”Set\_Cookie”

缓存Set-Cookie要特别注意是否会破坏用户跟踪，或者有些缓存只会缓存主体响应内容不会缓存首部字段，此时从新请求不会有cookie内容，强制向服务器验证缓存可以返回响应的Cookie新的内容，可缓存的响应需要添加下面的首部:

Cache-Control: must-revalidate, max-age=0

也可以不用cookie是实现跟踪记录，使用持久cookie跟踪用户就是一种最大的滥用。