# HTTP/1.1协议简介

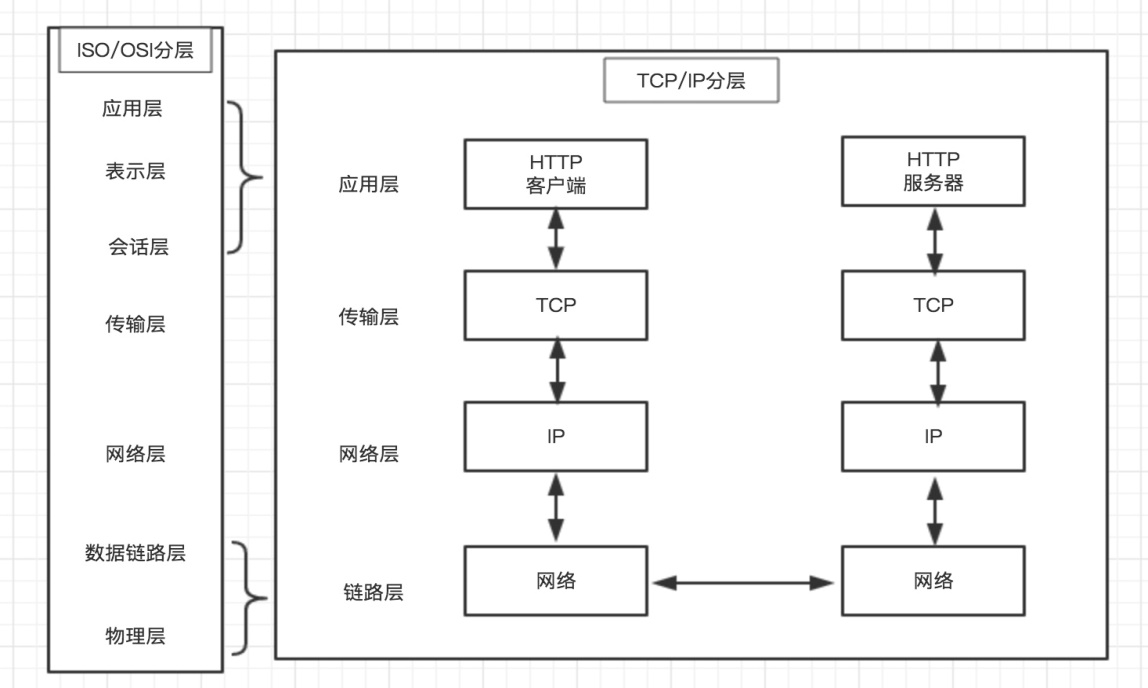
一、HTTP协议简介

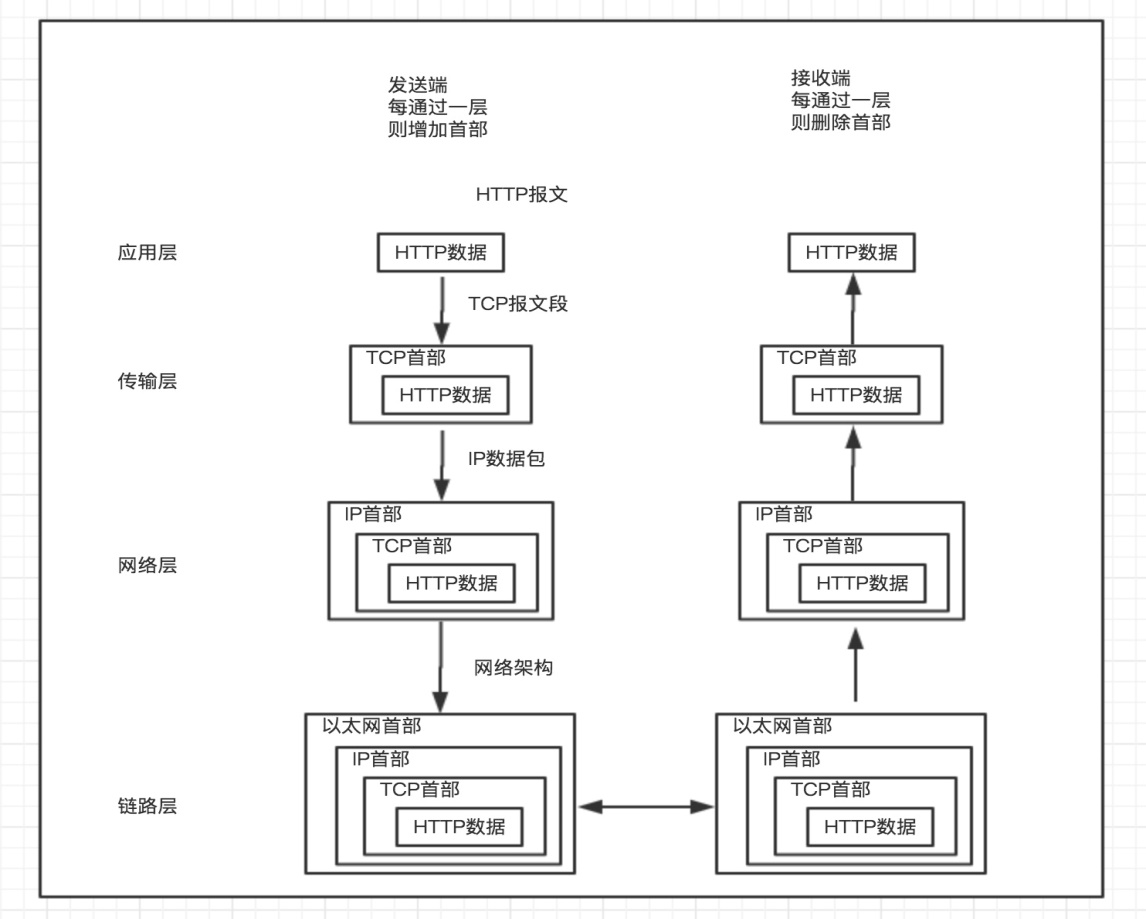
**网络协议**是指计算机通信网络中两台计算机之间进行通信所必须共同遵守的规定或规则。  
      
   **HTTP**（HyperText Transfer Protocol ，超文本传输协议)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。1960年美国人Ted Nelson构思了一种通过计算机处理文本信息的方法，并称之为超文本（hypertext）,这成为了HTTP超文本传输协议标准架构的发展根基。Ted Nelson组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force ）共同合作研究，最终发布了一系列的RFC，其中著名的RFC 2616定义了HTTP 1.1。  
      
    在我们如今的网络时代。我们每天浏览着万万千千的网页，图片，文字，这些都离不开Http，他是我们快速可靠访问世界各地的web服务器资源地基础。Http使用的是可靠地数据传输协议TCP协议。这样就是保证了我们所访问资源的万无一失，不会产生数据丢失或者损坏。这也可以使我们开发人员把更多的经历放在程序业务细节上得编写。避免了考虑一些数据传输途中的缺陷。

1.1.网络基础TCP/IP  
    为了理解HTTP,我们有必要先了解一下TCP/IP协议族。  
    通常使用的网络(包括互联网)是在TCP/IP协议族的基础上动作的。而HTTP属于它的内部的一个子集。

**1.1.1.TCP/IP协议族**  
    计算机与网络设备要相互通信，需要一种规则来满足各种情况，这种规则称为协议(protocol)  
    TCP/IP协议是互联网相关的各类协议族的总称。

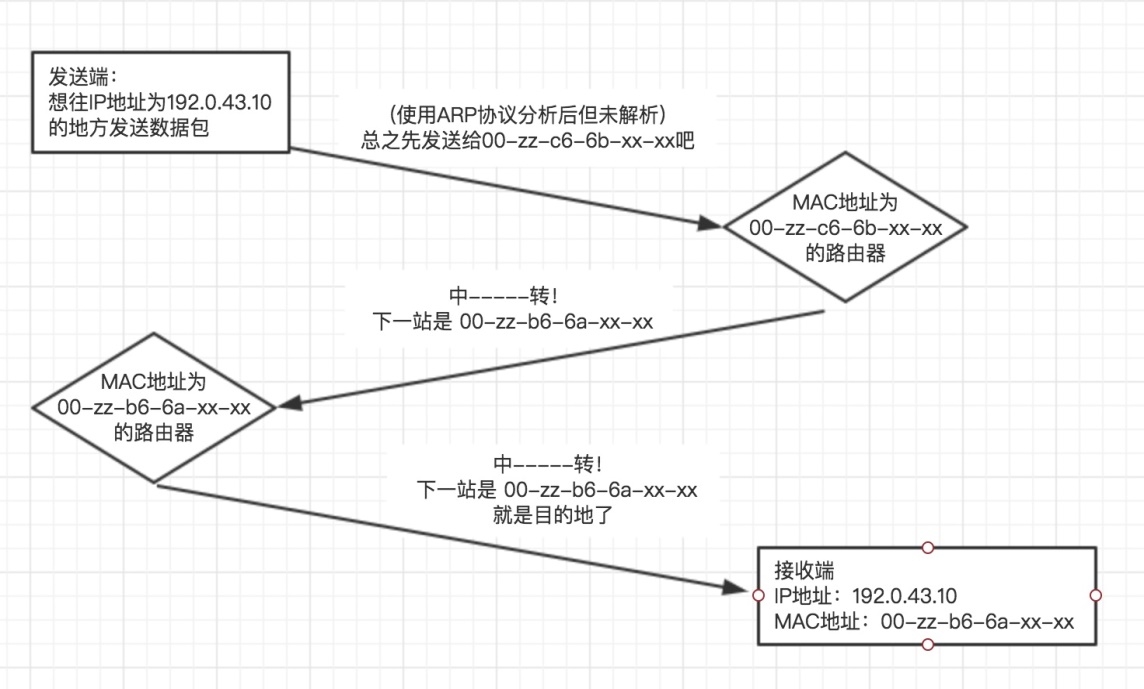
1.1.2.TCP/IP的分层管理  
    TCP/IP协议族里重要的一点就是分层，好处是解藕。TCP/IP协议族层次分别为以下4层：应用层、传输层、网络层和数据链路层。  
**应用层：**  
    应用层决定了向用户提供应用服务时通信的活动。  
    TCP/IP协议族内预存了各类通用的应用服务。比如，FTP(File Transfer Protocol, 文件传输协议)和DNS(Domain Name System, 域名系统)服务就是其中两类。  
    HTTP协议也处于该层。  
**传输层：**  
    传输层对上层应用层，提供处于网络连接中的两台计算机之间的数据传输。  
    在传输层有两个性质不同的协议：TCP(Transmission Control Protocol, 传输控制协议)和UDP(User Data Protocol, 用户数据报)  
**网络层(又名网络互连层)：**  
    网络层用来处理在网络上流动的数据包。数据包是网络传输的最小数据单位。该层规定了通过怎样的路径(所谓的传输路线)到达对方计算机，并把数据包传送给对方。  
    与对方计算机之间通过多台计算机或网络设备进行传输时，网络层所起的作用就是在众多的选项内选择一条传输路线。  
    IP协议处于该层。  
**链路层(又名数据链路层，网络接口层)：**  
    用来处理连接网络的硬件部分。包括控制操作系统、硬件的设备驱动、NIC(Network Interface Card, 网络适配器，即网卡)，及光纤等物理可见部分(还包括连接器等一切传输媒介)。硬件上的范畴均在链路层的作用范围之内。



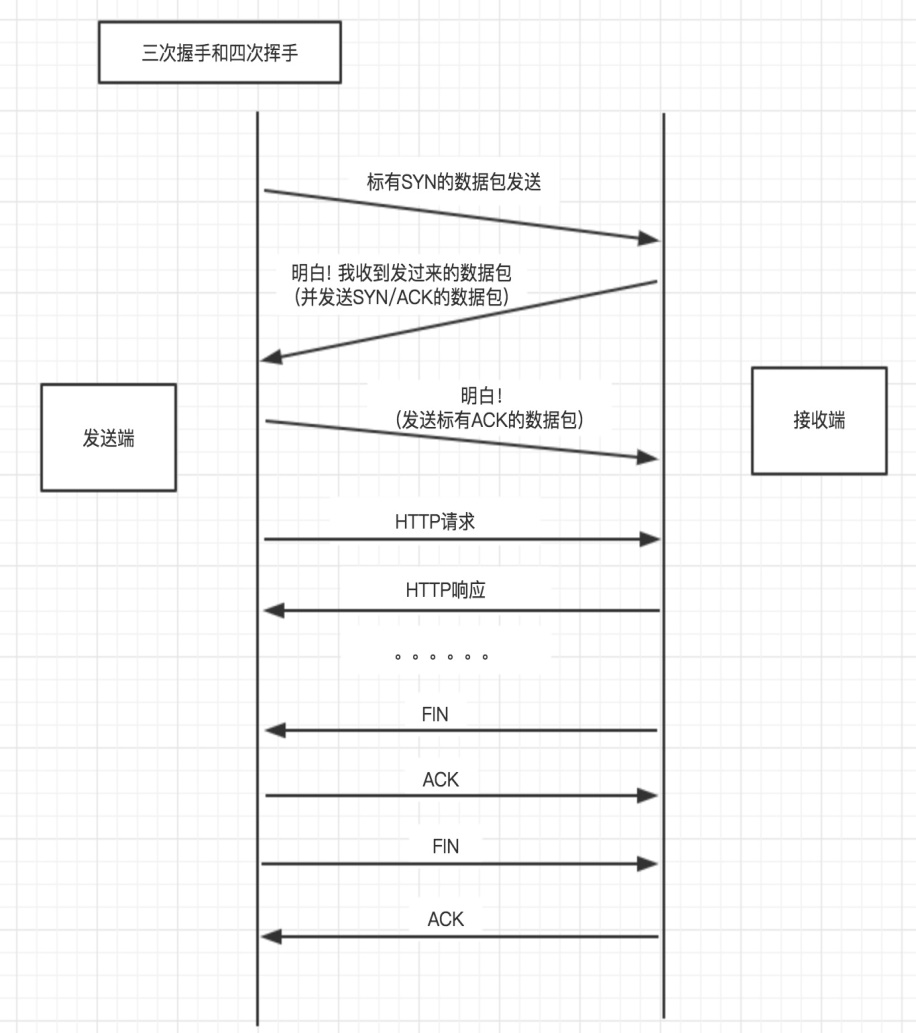


1.4.与HTTP关系密切的协议：IP、TCP和DNS  
下面分别针对在TCP/IP协议族与HTTP密不可分的3个协议IP、TCP和DNS。  
  
1.4.1 负责传输的IP协议  
    IP(Internet Protocol)网际协议位于网络层，也就是网际协议，几乎所有使用网络的系统都会用到IP协议。  
    IP协议的作用是把各种数据包传送给对方。而要保证确实传送到对方那里，则需要满足各类条件。其中两个重要的条件是IP地址和MAC地址(Media Access Control Address).  
    IP地址指明了节点被分配到的地址，MAC地址是指网卡所属的固定地址。IP地址可以和MAC地址进行配对。IP地址可变换，但MAC地址基本上不会更改。

**使用ARP协议凭借MAC地址进行通信**  
    IP间的通信依赖MAC地址。在网络上，通常是多台计算机和网络设备中转才能连接到对方。而在中转时，会利用下一站中转设备的MAC地址来收索下一个中转目标。这地，会采用**ARP协议**(Address Resolution Protocol).ARP是一种用以解析地址的协议，根据通信方的IP地址就可以反查出对应的MAC地址。  
 **没有人能全面掌握互联网中的传输状况**  
    在到达通信目标前的中转过程中，那些计算机和路由器等网络设备只能获悉很粗略的传输路线，这种机制称为**路由选择(routing),**有点像快递公司的送货过程。



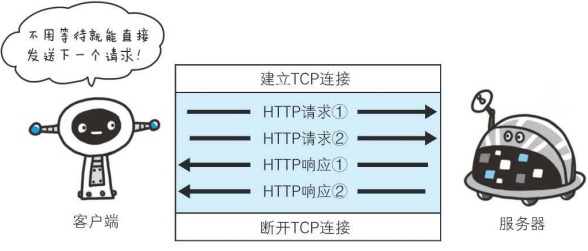
1.4.2.确保可靠性的TCP协议  
    TCP位于传输层，提供可靠的字节流服务。所谓的字节流服务是指，为了方便传输，将大块数据分割成以报文段(segment)为单位的数据包进行管理。而可靠的传输服务是指，能够把数据准确可靠地传给对方。也就是说，TCP协议为了更容易传送大数据才把数据分割，而且TCP协议能够确认数据最终是否送达到对方。  
**确保数据能到达目标**  
    为了准确无误地将数据送达目标处，TCP协议采用了三次握手策略(three-way handshaking)。用TCP协议把数据包送出去后，TCP一定会向对方确认是否成功送达。握手过程中使用了TCP的标志----SYN(synchronize)和ACK(acknowledgement)。  
    若在握手过程中某个阶段莫名中断，TCP协议会再次以相同的顺序发送相同的数据包。



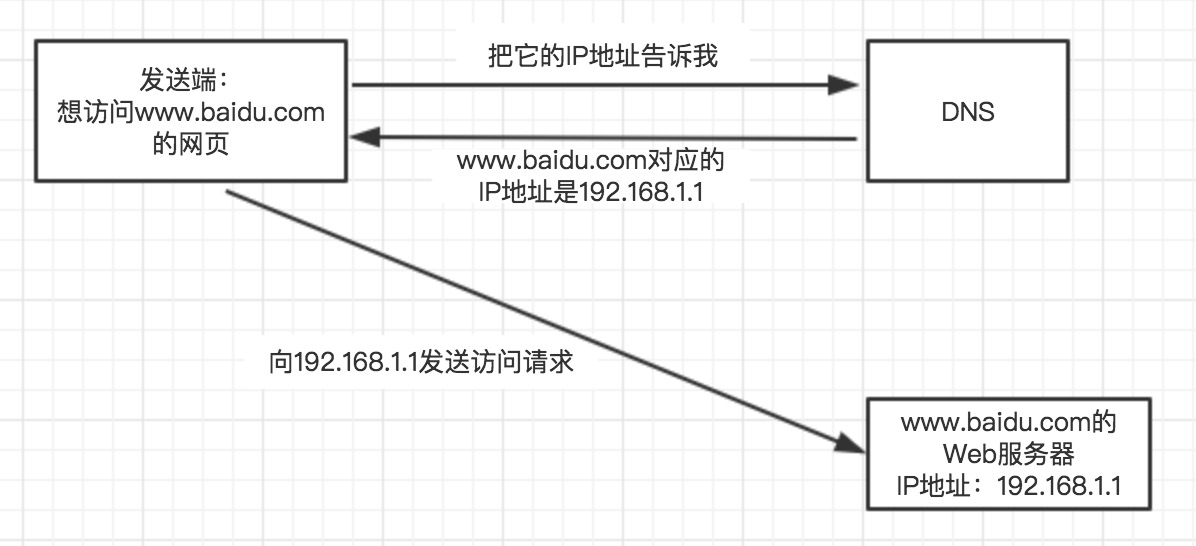
    除了三次握手，TCP协议还有其他各种手段来保证通信的可靠性。

**持久连接节省通信量**  
    HTTP协议的初始版本中，每次进行一次通信都要断开一次TCP连接。但是由于传输信息量的增大(如一个页面里有多张图片请求)，每次请求都造成的无谓TCP连接建立和断开，增加了通信量的开销。  
    因此，为解决上述TCP连接的问题，提出了**持久连接**（或HTTP keep-alive）方法。特点是只要任意一端没有明确提出断开连接，则保持TCP连接状态。  
    持久连接的**好处**在于减少了 TCP 连接的重复建立和断开所造成的额外开销，减轻了服务器端的负载。另外，减少开销的那部分时间，使 HTTP 请求和响应能够更早地结束，这样 Web 页面的显示速度也就相应提高了。   
    在 HTTP/1.1 中，所有的连接默认都是持久连接，但在 HTTP/1.0 内并未标准化。虽然有一部分服务器通过非标准的手段实现了持久连接，但服务器端不一定能够支持持久连接。毫无疑问，除了服务器端，客户端也需要支持持久连接。

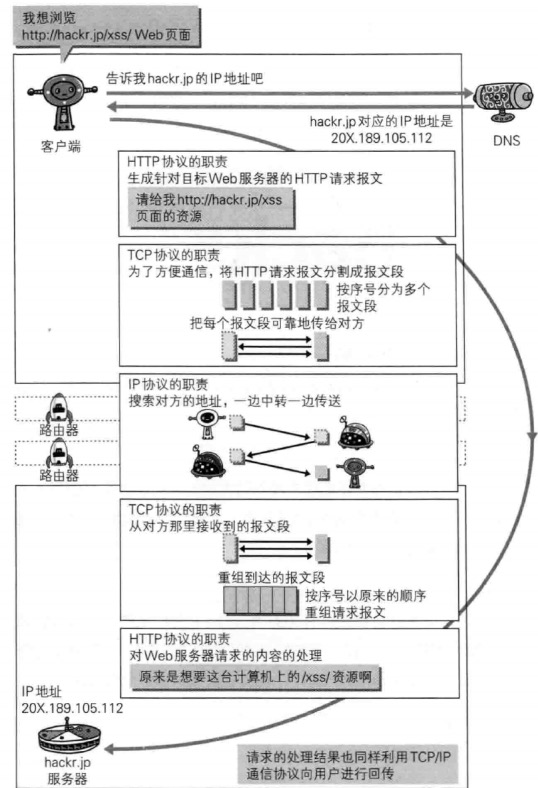
**管线化**  
    背景：持久连接技术使得管线化方式发送成为可能  
    特点：管线化技术出现后使得**不用等待响应即可直接发送下一个请求**（从前发送请求后需要等待并接收响应），这样就可以做到同时并行发送多个请求，而不需要一个接一个等待响应。



1.5.负责域名解析的DNS服务  
    DNS(Domain Name System)服务是和HTTP协议一样位于应用层的协议。它提供域名到IP地址之间的解析服务。

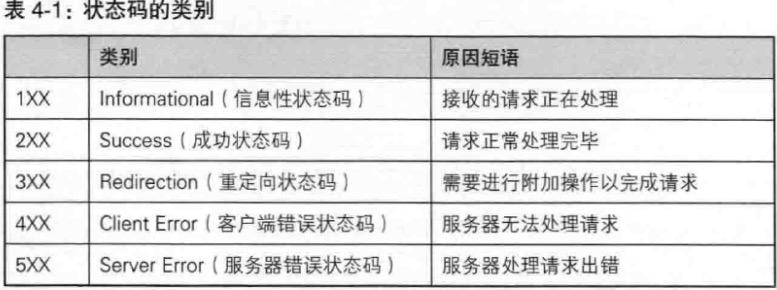


各种协议与HTTP协议的关系



二、URL详解

   统一资源定位符(Uniform Resource Locator， URL)，又叫做网页地址，是互联网上标准的资源的地址（Address），用于描述一个网络上的资源。统一资源定位符的开始，一般会标志着一个计算机网络所使用的网络协议。URI（Uniform Resource Identifier，统一资源标识符）用于标识某一资源   
   基本格式为：  
   schema://host[:port]/path/[;url-params][?query-string][#anchor]  
  
**格式说明**　  
  scheme：指定底层使用的协议(例如：http, https, ftp， mailto，file,telnet)  
  host：  HTTP服务器的IP地址或者域名  
  port：  HTTP服务器的默认端口是80，这种情况下端口号可以省略。如果使用了别的端口，必须指明。例如http://www.cnblogs.com:8080/  
  path：  由零或多个“/”符号隔开的字符串，一般用来表示主机上的一个目录或文件地址  
  url-params：用于指定特殊参数的可选项  
  query-string：可选，用于给动态网页（如使用CGI、ISAPI、PHP、JSP、ASP、ASP.NET等技术制作的网页）传递参数，多个参数用“&”符号隔开，每个参数的名和值用“=”符号隔开。    
  anchor(锚用)：于指定网络资源中的片断。例如一个网页中有多个名词解释，可使用fragment直接定位到某一名词解释  
  
**URL的一个例子（使用http协议）**  
  http://www.mywebsite.com/sj/test;id=8079?name=sviergn&x=true#stuff  
  
  schema: http  
  host:   www.mywebsite.com  
  port:   80,没有端口号时默认为80  
  path:   /sj/test  
  url-params:   id=8079  
  query-string: name=sviergn&x=true  
  anchor:       stuff

状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：  
1xx：指示信息--表示请求已接收，继续处理  
2xx：成功--表示请求已被成功接收、理解、接受

**[html]** [view plain](http://blog.csdn.net/oChangWen/article/details/51626969) [copy](http://blog.csdn.net/oChangWen/article/details/51626969)

1. 1）200 OK：表示从客户端发来的请求在服务器端被正常处理
3. 2）204 No Content：请求处理成功，但无资源可返回。代表服务器接收的请求已成功处理，但在返回的响应报文中不含实体的主体部分。另外，也不允许返回任何实体的主体。比如，当从浏览器发出请求处理后，返回 204 响应，那么浏览器显示的页面不发生更新。
5. 3）206 Partial Content：对资源某一部分的请求。表示客户端进行了范围请求，而服务器成功执行了这部分的 GET 请求。响应报文中包含由 Content-Range 指定范围的实体内容。

3xx：重定向--要完成请求必须进行更进一步的操作

**[html]** [view plain](http://blog.csdn.net/oChangWen/article/details/51626969) [copy](http://blog.csdn.net/oChangWen/article/details/51626969)

1. 1）301 Moved Permanently
2. 永久性重定向。该状态码表示请求的资源已被分配了新的URI，以后应使用资源现在所指的URI。
3. 也就是说，如果已经把资源对应的URI保存为书签了，这时应该按Location 首部字段提示的 URI 重新保存。
5. 2）302 Found
6. 临时性重定向。该状态码表示请求的资源已被分配了新的 URI，希望用户（本次）能使用新的 URI 访问。
7. 和 301 Mov ed Permanently 状态码相似，但 302 状态码代表的资源不是被永久移动，只是临时性质的。换句话说，已移动的资源对应的 URI 将来还有可能发生改变。比如，用户把 URI 保存成书签，但不会像 301 状态码出现时那样去更新书签，而是仍旧保留返回 302 状态码的页面对应的 URI。
9. 3）303 See Other
10. 表示由于请求对应的资源存在着另一个 URI，应使用 GET 方法定向获取请求的资源。
11. 303 状态码和 302 Found 状态码有着相同的功能，但 303 状态码明确表示客户端应当采用 GET 方法获取资源，这点与 302 状态码有区别。
13. 4）304 Not Modified
14. 资源已找到，但未复合条件请求。该状态码表示客户端发送附带条件的请求时，服务器端允许请求访问资源，但未满足条件的情况。304 状态码返回时，不包含任何响应的主体部分。304 虽然被划分在 3XX 类别中，但是和重定向没有关系。
16. 5）307 Temporary Redirect
17. 临时重定向。该状态码与 302 Found 有着相同的含义。尽管 302 标准禁止 POST 变换成 GET，但实际使用时大家并不遵守。307 会遵照浏览器标准，不会从 POST 变成 GET。但是，对于处理响应时的行为，每种浏览器有可能出现不同的情况。

4xx：客户端错误--请求有语法错误或请求无法实现

**[html]** [view plain](http://blog.csdn.net/oChangWen/article/details/51626969) [copy](http://blog.csdn.net/oChangWen/article/details/51626969)

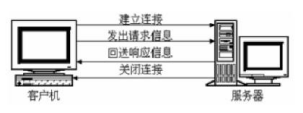
1. 1）400 Bad Request
2. 该状态码表示请求报文中存在语法错误。当错误发生时，需修改请求的内容后再次发送请求。另外，浏览器会像 200 OK 一样对待该状态码。
4. 2）401 Unauthorized
5. 该状态码表示发送的请求需要有通过HTTP认证(BASIC认证、DIGEST认证)的认证信息。另外，若之前已进行过一次请求，则表示用户认证失败。返回含有401的响应必须包含一个适用于被请求资源的WWW-Authenticate首部用以质询用户信息。当浏览器初次接收到401响应，会弹出认证用的对话窗口。
7. 3）403 Forbidden
8. 表示请求资源的访问被服务器拒绝。未获得文件系统的访问授权，访问权限出现某些问题（从未授权的发送源 IP 地址试图访问）等列举的情况都可能是发生 403 的原因。
10. 4）404 Not Found
11. 表示服务器上无法找到请求的资源。除此之外，也可以在服务器端拒绝请求且不想说明理由时使用。

5xx：服务器端错误--服务器未能实现合法的请求  
1)500 Internal Sever Error 表示服务器端执行请求时发生了错误。也有可能是 Web 应用存在的 bug或某些临时的故障。2）503 Service Unavailable 表明服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。如果事先得知解除以上状况需要的时间，最好写入 Retry After 首部字段再返回给客户端。

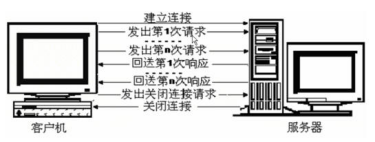
　常见状态代码、状态描述、说明：  
　　200 OK //客户端请求成功  
　　400 Bad Request //客户端请求有语法错误，不能被服务器所理解  
　　401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用   
　　403 Forbidden //服务器收到请求，但是拒绝提供服务  
　　404 Not Found //请求资源不存在，eg：输入了错误的URL  
　　500 Internal Server Error //服务器发生不可预期的错误  
　　503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

三、HTTP1.1请求处理流程

    首先，http属于TCP/IP模型中的应用层协议，而两个应用程序（我们这里指的就是浏览器与服务器）之间要进行互相通信，首先得建立TCP连接，然后浏览器才能向服务器发送请求信息，服务器在接受到请求信息后，返回相应的应答信息，浏览器接收到来自服务器的应答信息后，对这些数据进行解释执行。  
    **在http 1.0的版本中**，浏览器的每次请求（也就是对每一个页面的访问）都要求建立一次单独的TCP连接，在处理完每一次的请求后，就自动释放连接。如下图



    http 1.1则可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再发送下一个请求。



**HTTP工作过程**  
  1).客户机（Web浏览器）与服务器建立TCP连接（TCP的三次握手）  
  2).Web浏览器向Web服务器发送请求HTTP命令  
  3).Wed服务器接受请求并返回HTTP响应  
  4).[服务器关闭TCP连接(TCP的四次挥手)]  
  5).客户端浏览器解析HTML内容  
**注意**：如果浏览器或者服务器在其头信息加入了这行代码 Connection:keep-alive  
TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽

四、HTTP1.1请求方法

HTTP是不保存状态的协议  
    HTTP是一种不保存状态，即无状态(stateless)协议。HTTP协议自身不对请求和响应之间的通信状态进行保存。也就是说在HTTP这个级别，协议对于发送过的请求或响应都不做持久化处理。这是为了更快的处理大量事务，确保协议的可伸缩性，而特意把HTTP协议设计成如此简单的。  
    但是无状态也会导致业务处理变得有时棘手：比如，用户登录到一家购物网站，即使他跳转到该站的其他页面后，也需要能继续保持登录状态。针对这个实例，网站为了能够掌握是谁送出的请求，需要保存用户的状态。  
   因此，引入了Cookie技术用以实现保持状态功能。

**使用Cookie的状态管理**  
1、引入原因：  
    1）无状态优点：减少服务器cpu及内存资源的消耗  
    2）缺点：HTTP是无状态协议，无法根据之前的状态进行本次的请求处理。登录认证的web页面不能管理状态，每次跳转需要在请求报文添加参数来管理信息  
    3）如果让服务器管理全部客户端状态则会成为负担，保留无状态协议这个特征的同时又要解决类似的矛盾问题，于是引入了 Cookie 技术。  
**2、Cookie特点：**  
    Cookie技术通过在请求和响应报文中写入Cookie信息来控制客户端的状态，Cookie会根据从服务器端发送的响应报文内的一个叫Set-Cookie的首部字段，通知客户端保存Cookie，当下次客户端再次往该服务器发送请求时，客户端会自动在请求报文中加入Cookie值后发送出去。服务器会去检查是从哪个客户端发来的请求，然后对比服务器上的记录，得到之前的状态信息。

    在 HTTP 协议中，HTTP 请求可以使用多种请求方法，这些方法指明了要以何种方式来访问 Request-URI 所标识的资源。HTTP1.1 支持的请求方法如下  
 **（1）GET      
    GET 方法用于获取由 Request-URI 所标识的资源的信息**  
    GET方法是默认的HTTP请求方法，例如当我们通过在浏览器的地址栏中直接输入网址的方式去访问网页的时候，浏览器采用的就是 GET 方法向服务器获取资源。    
    我们可以使用GET方法来提交表单数据，用GET方法提交的表单数据只经过了简单的编码，同时它将作为URL的一部分向服务器发送，因此，如果使用GET方法来提交表单数据就存在着安全隐患上。例如：http://localhost/login.php?username=aa&password=1234  
从上面的URL请求中，很容易就可以辩认出表单提交的内容（？之后的内容）。另外由于GET方法提交的数据是作为URL请求的一部分所以提交的数据量不能太，这是因为浏览器对url的长度有限制。      
   该方法的响应是可缓存的（如果此响应满足HTTP缓存的要求)   
  
**（2）POST  
    请求服务器，让其接受请求中的实体，以作为被请求资源的一个新的从属物**  
    POST方法是GET方法的一个替代方法，它主要是向Web服务器提交表单数据，尤其是大批量的数据。通过POST方法提交表单数据时，数据不是作为URL请求的一部分而是作为标准数据传送给Web服务器，这就克服了GET方法中的信息无法保密和数据量太小的缺点。因此，出于安全的考虑以及对用户隐私的尊重，通常表单提交时采用POST方法。  
    该方法的响应是不可缓存的，除非响应里有合适的Cache-Control或者Expires头域  
  
**（3）PUT  
     PUT方法请求服务器去把请求里的实体存储在请求URI（Request-URI）标识下**  
     如果请求URI（Request-URI）指定的的资源已经在源服务器上存在，那么此请求里的实体应该被当作是源服务器关于此URI所指定资源实体的最新修改版本。如果请求URI（Request-URI）指定的资源不存在，并且此URI被用户代理定义为一个新资源，那么源服务器就应该根据请求里的实体创建一个此URI所标识下的资源。  
    该方法的响应是不可缓存的。  
   
**（4）DELETE      
    DELETE方法请求源服务器删除请求URI指定的资源**  
    该方法的响应是不能被缓存的。  
  
**（5）HEAD  
    获取请求实体的头域信息而不需要传输实体主体**  
    HEAD 方法与 GET 方法几乎是相同的，它们的区别在于 HEAD 方法只是请求消息报头，而不是完整的内容。对于 HEAD 请求的回应部分来说，它的 HTTP 头部中包含的信息与通过 GET 请求所得到的信息是相同的。利用这个方法，不必传输整个资源内容，就可以得到 Request-URI 所标识的资源的信息。这个方法通常被用于测试超链接的有效性，是否可以访问，以及最近是否更新。  
    该方法的响应是可缓存的。  
  
 **（6）OPTIONS  
    用来查询针对请求URI指定的资源支持的方法。(响应Allow:GET,POST)**  
    该方法的响应是不能缓存的。  
  
**（7）TRACE  
    用于激发一个远程的，应用层的请求消息回路**  
    TRACE方法让客户端测试到服务器的网络通路，回路的意思如发送一个请求返回一个响应，这就是一个请求响应回路  
    该方法的响应是不能被缓存的。  
   
**（8）CONNECT   
   动态切换到隧道的代理   
   注意**：这是HTTP 1.1协议规范保留的一个方法。

**GET和POST的区别**  
     1. GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456. POST方法是把提交的数据放在HTTP包的Body中.  
    2. GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制.  
    3. GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值。  
    4. GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码.

五、HTTP1.1消息结构

1、HTTP报文  
    用于HTTP协议交互的信息被称为HTTP报文，请求端的称之为请求报文，响应端的称之为响应报文。  
    组成：HTTP报文本身是由多行（用CR+LF作换行符）数据结构构成的字符串文本，大致可以分为报文首部和报文主体两块。

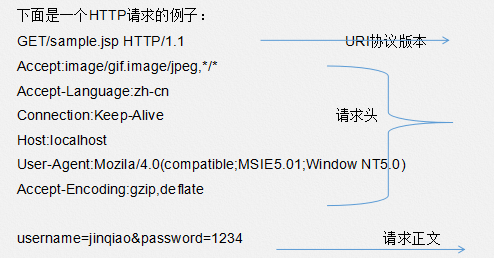


**1.1.请求报文和响应报文的首部内容由以下数据组成：**  
    1）请求行：包含用于请求的方法，请求URI和HTTP版本  
    2）状态行：包含表明响应结果的状态码，原因短语和HTTP版本  
    3）首部字段：包含表示请求和响应的各种条件和属性的各类首部，一般分为4种首部，通用首部、请求首部、响应首部、实体首部。  
    4）其他：可能是包含HTTP的RFC里未定义的首部（Cookie 等）

2、编码提升传输速率  
    HTTP在传输数据时可以按照数据原貌直接传输，也可以在传输过程中通过编码提升传输速率。通过在传输时编码，能有效地处理大量的访问请求。但是，编码的操作需要计算机来完成，因此会消耗更多的CPU等资源。  
**2.1.1.报文主体和实体主体的差别**  
    1）报文：是HTTP通信汇总的基本单位，由8位组字节流组成，通过HTTP通信传输。   
    2）实体：作为请求或者响应的有效载荷数据（补充项）被传输，其内容由实体首部和实体主体组成。  
    报文的主体用于传输请求或响应的实体主体。通常报文主体等于实体主体，只有当传输中进行编码操作时，实体主体的内容发生变化，才与报文主体产生差异。  
**2.1.2.压缩传输的内容编码**  
     向待发送邮件内增加附件时，为了使邮件容量变小，我们会先用zip压缩文件之后再添加附件发送。  
     HTTP协议中有一种被称为内容编码的功能也能进行类似的操作。内容编码指明应用在实体内容上的编码格式，并保持实体信息原样压缩。内容编码后的实体由客户端接收并负责解码。   
     常见的内容编码有以下几种：gzip(GUN zip)，compress(UNIX 系统的标准压缩)，deflate(zlib), identity(不进行编码)  
**2.1.3.分割发送的分块传输编码**  
    在HTTP通信过程中，请求的编码实体资源未全部传输完成之前，浏览器无法显示请求页面。但传输大容量数据时，通过把数据分割成多块，能够让浏览器逐步显示页面。这种把实体主体分块的功能称为分块传输编码。  
    分块传输编码会将实体主体分成多个部分(块)。每一块都会用16进制来标记块的大小，而实体主体的最后一块会使用"0(CR+LF)"来标记。使用分块传输编码的实体主体由接受的客户端负责解码，恢复编码前的实体主体。  
    HTTP/1.1中存在一种称为传输编码的机制，它可以在通信时按某种编码方式传输，但只定义作用于分块传输编码中。   
  
3、发送多种数据的多部分对象集合  
    发送邮件时，我们可以在邮件里写入文字并添加多份附件，这是因为采用MIME(Multipurpose Internet Mail Extensions, 多用途因特网邮件扩展)机制，它允许邮件处理文本、图片、视频等多个不同类型的数据。在MIME扩展中会使用一种称为多部分对象集合(Multipart)的方法，来容纳多份不同类型的数据。  
    相应地，HTTP协议中也采纳了多部分对象集合，发送的一份报文主体内可含有多类型实体。通常是在图片或文本文件等上传时使用。  
    多部分对象集合包含的对象如下：  
      1).multipart/form-data：在Web表单文件上传时使用。  
      2).multipart/byteranges：状态码206(Partial Content, 部分内容)响应报文包含了多个范围的内部时使用。  
      在HTTP报文中使用多部分对象集合时，需要在首部字段里加上Content-type。  
    多部分对象集合的每个部分类型中，都可以含有首部字段。另外，可以在某个部分中嵌套多部分对象集合。  
      
**4、获取部分内容的范围请求**  
    以前，用户不能使用现在这种高速的带宽访问互联网，如果下载过程中遇到网络中断的情况，那就必须重头开始。为了解决上述问题，需要一种可恢复的机制。所谓恢复是指能从之前下载中断处恢复下载。  
    要实现这种功能需要指定下载的实体范围，这种指定范围发送的请求叫做范围请求。  
    执行范围请求时，会用到首部字段Range来指定资源的byte范围。byte范围的指定形式如下：  
    Range: bytes=5001-10000  # 5001~10000字节  
    Range: bytes=5001-  # 从5001字节之后全部的  
    Range: bytes=-3000， 5000-7000  # 从一开始到3000字节和5000~7000字节的多重范围  
    针对范围请求，响应会返回状态码206(Partial Content, 部分内容)的响应报文。另外，对于多重范围的范围请求，响应会在首部字段Content-Type标明multipart/byteranges后返回响应报文  
  
**5、内容协商返回最适合的内容**  
    同一个web网站可能存在多份相同内容的页面，比如英文/中文，它们内容上虽相同，但使用的语言却不同。  
    当浏览器的默认语言为英文/中文，访问相同URI的WEB页面则会显示对应的英文版/中文版的web页面，这样的机制称为内容协商（Content Negotiation）。  
    内容协商机制是指客户端和服务器端就响应的资源内容进行交涉，然后提供给客户端最为适合的资源。内容协商会以响应资源的语言、字符集、编码方式等作为判断的基准。  
    包含在请求报文中的某些首部字段(如下)就是判断的基准。  
    Accept, Accept-Charset, Accept-Encoding, Accept-Language, Content-Language  
    内部协商技术有以下3种类型：  
    1).服务器驱动协商(Server-driven Negotiation)  
    由服务器端进行内容协商。以请求的首部字段为参考，在服务器端自动处理。但对用户来说，以浏览器发送的信息作为判定的依据，并不一定能筛选出最优内容。  
    2),客户端驱动协商(Agent-driven Negotiation)  
    由客户端进行内容协商的方式。用户从浏览器显示的可选项列表中手动选择。还可以利用JavaScript脚本在Web页面上自动进行上述选择。比如按OS的类型或浏览器类型，自行切换成PC版页面或手机版页面。  
    3).透明协商(Transparent Negotiation)  
    是服务器驱动和客户端驱动的结合体，是由服务器端和客户端各自进行内容协商的一种方法。

  5.1、HTTP请求格式

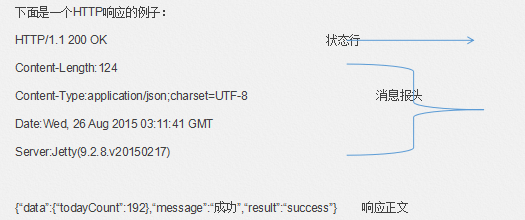
    当浏览器向Web服务器发出请求时，它向服务器传递了一个数据块，也就是请求信息，HTTP请求信息由3部分组成：请求方法URI协议/版本、请求头(Request Header)、请求正文



**（1）请求方法URI协议/版本**  
    请求的第一行是“方法URL议/版本”：GET /sample.jsp HTTP/1.1  
    以上代码中“GET”代表请求方法，“/sample.jsp”表示URI，“HTTP/1.1代表协议和协议的版本。     
    根据HTTP标准，HTTP请求可以使用多种请求方法，详情见“HTTP1.1请求方法”章节  
  
 **(3）请求正文**   
    请求头和请求正文之间是一个空行，这个行非常重要，它表示请求头已经结束，接下来的是请求正文。请求正文中可以包含客户提交的查询字符串信息：      
    username=jinqiao&password=1234      
    在以上的例子的HTTP请求中，请求的正文只有一行内容。当然，在实际应用中，HTTP请求正文可以包含更多的内容。  
  
**（2） 请求头(Request Header)**  
    每个头域由一个域名，冒号和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。请求头包含许多有关的客户端环境和请求正文的有用信息。例如，请求头可以声明浏览器所用的语言，请求正文的长度等。  
  
  **HTTP最常见的请求头如下：   
  1.Transport 头域**  
    ①.Connection  
    作用：表示是否需要持久连接。  
    Connection: keep-alive 当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接  
    Connection: close  代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭，当客户端再次发送Request，需要重新建立TCP连接。  
  
    ②.Host  
    作用：发送请求时，该报头域是必需的，主要用于指定被请求资源的Internet主机和端口号，它通常从HTTP URL中提取出来的  
    例如： Host: 192.168.42.237:8181  
  
 **2.Client 头域**  
    ①.Accept  
    作用：浏览器可以接受的媒体类型（MIME类型）。通配符 \* 代表任意类型。例如  Accept: \*/\*  代表浏览器可以处理所有类型，(一般浏览器发给服务器都是发这个)  
    ②.Accept-Encoding  
    作用： 浏览器申明自己接收的编码方法，通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate）  
    例如： Accept-Encoding: gzip, deflate。Server能够向支持gzip/deflate的浏览器返回经gzip或者deflate编码的HTML页面。许多情形下这可以减少5到10倍的下载时间，也节省带宽。  
    ③.Accept-Language  
    作用： 浏览器申明自己接收的语言。  
    例如： Accept-Language:zh-cn 。如果请求消息中没有设置这个报头域，服务器假定客户端对各种语言都可以接受。  
    ④.User-Agent  
    User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.114 Safari/537.36  
    作用：告诉HTTP服务器， 客户端使用的操作系统和浏览器的名称和版本等信息  
    ⑤.Accept-Charset  
    作用：浏览器申明自己接收的字符集，这就是本文前面介绍的各种字符集和字符编码，如gb2312，utf-8（通常我们说Charset包括了相应的字符编码方案）  
  
 **3.Cookie/Login 头域**    
    ①.Cookie  
    作用： 最重要的header, 将cookie的值发送给HTTP 服务器  
  
  **4.Entity头域**  
    ①.Content-Length  
    作用：发送给HTTP服务器数据的长度，即请求消息正文的长度；  
    例如： Content-Length: 38  
    ②.Content-Type  
    作用：指明发送给接收者的实体正文的媒体类型（MIME TYPE）  
  
 **5.Miscellaneous 头域**  
    ①.Referer  
    作用： 提供了Request的上下文信息的服务器，告诉服务器我是从哪个链接过来的，比如从我主页上链接到一个朋友那里，他的服务器就能够从HTTP Referer中统计出每天有多少用户点击我主页上的链接访问他的网站。  
    例如: Referer:http://translate.google.cn/?hl=zh-cn&tab=wT  
  
**6.Cache 头域**  
    ①.If-Modified-Since  
    作用： 把浏览器端缓存页面的最后修改时间发送到服务器去，服务器会把这个时间与服务器上实际文件的最后修改时间进行对比。如果时间一致，那么返回304，客户端就直接使用本地缓存文件。如果时间不一致，就会返回200和新的文件内容。客户端接到之后，会丢弃旧文件，把新文件缓存起来，并显示在浏览器中。  
    例如：If-Modified-Since: Thu, 09 Feb 2012 09:07:57 GMT。  
    ②.If-None-Match  
    作用: If-None-Match和ETag一起工作，工作原理是在HTTP Response中添加ETag信息。 当用户再次请求该资源时，将在HTTP Request 中加入If-None-Match信息(ETag的值)。如果服务器验证资源的ETag没有改变（该资源没有更新），将返回一个304状态告诉客户端使用本地缓存文件。否则将返回200状态和新的资源和Etag。使用这样的机制将提高网站的性能  
    例如: If-None-Match: "03f2b33c0bfcc1:0"  
    ③.Pragma  
    作用： 防止页面被缓存， 在HTTP/1.1版本中，它和Cache-Control:no-cache作用一模一样  
    Pargma只有一个用法， Pragma: no-cache  
    注意: 在HTTP/1.0版本中，只实现了Pragema:no-cache, 没有实现  
    ④.Cache-Control  
    作用: 这个是非常重要的规则。 这个用来指定Response-Request遵循的缓存机制。各个指令含义如下  
    Cache-Control:Public    可以被任何缓存所缓存,多用户间共享  
    Cache-Control:Private   内容只缓存到私有缓存中,不能在用户间共享  
    Cache-Control:no-cache   所有内容都不会被缓存  
    Cache-Control: max-age=x：缓存时间 以秒为单位

  5.2、HTTP响应格式

    在接收和解释请求消息后，服务器会返回一个 HTTP 响应消息。与 HTTP 请求类似，HTTP响应也是由三个部分组成，分别是：状态行、消息报头和响应正文。



**（1）状态行**  
    状态行由协议版本、数字形式的状态代码，及相应的状态描述组成，各元素之间以空格分隔，结尾时回车换行符，格式如下：      
    HTTP-Version Status-Code Reason-Phrase CRLF  
      
    HTTP-Version 表示服务器 HTTP 协议的版本，  
    Status-Code 表示服务器发回的响应代码，  
    Reason-Phrase 表示状态代码的文本描述，  
    CRLF 表示回车换行。例如：HTTP/1.1 200 OK (CRLF)  
  
**（3）响应正文**  
    响应正文就是服务器返回的资源的内容，响应头和正文之间也必须用空行分隔。  
  
**（2）响应头信息**  
     HTTP最常见的响应头如下所示：  
  
**1.Cache头域**  
    ①.Date  
    作用：生成消息的具体时间和日期，即当前的GMT时间。  
    例如：　Date: Sun, 17 Mar 2013 08:12:54 GMT  
    ②.Expires  
    作用: Expires字段声明了一个网页或URL地址不再被浏览器缓存的时间，一旦超过了这个时间，浏览器都应该联系原始服务器。  
    例如: Expires: Thu, 19 Nov 1981 08:52:00 GMT（指定Expires值为一个早已过去的时间，那么访问此网时若重复在地址栏按回车，那么每次都会重复访问。有时候仅仅设置Pragma: no-cache Cache-Control: no-cache 还是不保险，需要将过期时间设置成过去的时间就确保了对象不被缓存）  
  
**2.Cookie/Login 头域**  
    ①.P3P  
    作用: 用于跨域设置Cookie, 这样可以解决iframe跨域访问cookie的问题  
    例如: P3P: CP=CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR  
    ②.Set-Cookie  
    作用： 非常重要的header, 用于把cookie 发送到客户端浏览器， 每一个写入cookie都会生成一个Set-Cookie.  
    例如: Set-Cookie: PORT-8181-JSESSIONID=x5vixfyb0r8r1payckxoieww3;Path=/  
  
**3.Entity实体头域**  
    ①.Etag  
    作用: 和If-None-Match 配合使用。响应中资源的校验值，在服务器上某个时段是唯一标识的。ETag是一个可以 与Web资源关联的记号（token），和Last-Modified功能差不多，也是一个标识符，一般和Last-Modified一起使用，加强服务器判断的准确度。  
    例如: ETag: "03f2b33c0bfcc1:0"  
    ②.Last-Modified  
    作用： 用于指示资源的最后修改日期和时间。（实例请看上节的If-Modified-Since的实例）  
    例如: Last-Modified: Wed, 21 Dec 2011 09:09:10 GMT  
    ③.Content-Type  
    作用：WEB服务器告诉浏览器自己响应的对象的类型和字符集,  
    例如:Content-Type:application/json;charset=UTF-8  
    ④.Content-Length  
    指明实体正文的长度，以字节方式存储的十进制数字来表示。在数据下行的过程中，Content-Length的方式要预先在服务器中缓存所有数据，然后所有数据再一股脑儿地发给客户端。  
    例如:  Content-Length:124  
    ⑤.Content-Encoding  
    作用：文档的编码（Encode）方法。一般是压缩方式。  
    WEB服务器表明自己使用了什么压缩方法（gzip，deflate）压缩响应中的对象。利用gzip压缩文档能够显著地减少HTML文档的下载时间。  
    例如：Content-Encoding：gzip  
    ⑥.Content-Language  
    作用： WEB服务器告诉浏览器自己响应对象的语言  
    例如： Content-Language: en  
  
**4.Miscellaneous 头域**  
    ①.Server  
    作用：指明HTTP服务器的软件信息  
    例如: Server: Jetty(9.2.8.v20150217)  
    ②.X-Powered-By  
    作用：表示网站是用什么技术开发的  
    例如： X-Powered-By: PHP/5.2.5  
  
**5.Transport头域**  
    ①.Connection     
    Connection: keep-alive 当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接     
    Connection: close 代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭， 当客户端再次发送Request，需要重新建立TCP连接。  
  
**6.Location头域**  
    ①.Location  
    作用： 用于重定向一个新的位置， 包含新的URL地址

六、HTTP1.1浏览器缓存

  浏览器缓存：包括页面html，image，js，css等资源的缓存  
  **6.1、缓存工作原理**  
    1）第一次请求：浏览器通过http的header报头，附带Expires，Cache-Control，Last-Modified/Etag向服务器请求，此时服务器记录第一次请求的Last-Modified/Etag                       
    2）再次请求：当浏览器再次请求的时候，请求头附带Expires，Cache-Control，If-Modified-Since/Etag向服务器请求     
    3）服务器根据第一次记录的Last-Modified/Etag和再次请求的If-Modified-Since/Etag做对比，判断是否需要更新，服务器通过这两个头判断本地资源未发生变化，客户端不需要重新下载，返回304响应  
  
**6.2、缓存的优点：**    
    1）服务器响应更快：因为请求从缓存服务器（离客户端更近）而不是源服务器被相应，这个过程耗时更少，让服务器看上去响应更快。      
    2）减少网络带宽消耗：当副本被重用时会减低客户端的带宽消耗；客户可以节省带宽费用，控制带宽的需求的增长并更易于管理。