HTTP通信机制是在一次完整的HTTP通信过程中，Web浏览器与Web服务器之间将完成下列7个步骤：

**1. 建立TCP连接**  
在HTTP工作开始之前，Web浏览器首先要通过网络与Web服务器建立连接，该连接是通过TCP来完成的，该协议与IP协议共同构建Internet，即著名的TCP/IP协议族，因此Internet又被称作是TCP/IP网络。HTTP是比TCP更高层次的应用层协议，根据规则，只有低层协议建立之后才能进行更高层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。

**2. Web浏览器向Web服务器发送请求命令**   
一旦建立了TCP连接，Web浏览器就会向Web服务器发送请求命令。例如：GET/sample/hello.jsp HTTP/1.1。

**3. Web浏览器发送请求头信息**   
浏览器发送其请求命令之后，还要以头信息的形式向Web服务器发送一些别的信息，之后浏览器发送了一空白行来通知服务器，它已经结束了该头信息的发送。

**4. Web服务器应答**   
客户机向服务器发出请求后，服务器会客户机回送应答， HTTP/1.1 200 OK ，应答的第一部分是协议的版本号和应答状态码。

**5. Web服务器发送应答头信息**   
正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档。

**6. Web服务器向浏览器发送数据**   
Web服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据。

**7. Web服务器关闭TCP连接**   
一般情况下，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接，然后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。

HTTP请求信息由3**部分组成**：

1、请求方法（GET/POST）、URI、协议/版本

2、请求头(Request Header)

3、请求正文

以上图做例进行分析：

POST http://xg.mediportal.com.cn/health/sms/verify/telephone HTTP/1.1

User-Agent: DGroupPatient/1.052701.230/Dalvik/2.1.0 (Linux; U; Android 5.1.1; KIW-AL10 Build/HONORKIW-AL10)  
Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
Host: xg.mediportal.com.cn  
Connection: Keep-Alive  
Accept-Encoding: gzip  
Content-Length: 33  
  
telephone=15527177736&userType=1&

（1）请求方法、URI、协议/版本

请求的第一行是“方法、URL、协议/版本”：

POST http://xg.mediportal.com.cn/health/sms/verify/telephone HTTP/1.1

以上代码中“POST”代表请求方法，“http://xg.mediportal.com.cn/health/sms/verify/telephone”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。例如：HTTP1.1目前支持7种请求方法：GET、POST、HEAD、OPTIONS、PUT、DELETE和TARCE。

|  |  |
| --- | --- |
| GET | 请求获取由Request-URI所标识的资源 |
| POST | 在Request-URI所标识的资源后附加新的数据 |
| HEAD | 请求获取由Request-URI所标识的资源的响应消息报头 |
| OPTIONS | 请求查询服务器的性能，或查询与资源相关的选项和需求 |
| PUT | 请求服务器存储一个资源，并用Request-URI作为其标识 |
| DELETE | 请求服务器删除由Request-URI所标识的资源 |
| TRACE | 请求服务器回送收到的请求信息，主要用语测试或诊断 |

在Internet应用中，最常用的方法是GET和POST。最后，协议版本声明了通信过程中使用HTTP的版本。

（2）请求头(Request Header)

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。

User-Agent: DGroupPatient/1.052701.230/Dalvik/2.1.0 (Linux; U; Android 5.1.1; KIW-AL10 Build/HONORKIW-AL10)   //用户发送请求的客户端环境  
Content-Type: application/x-www-form-urlencoded; charset=UTF-8   //表单默认的提交数据的格式  
Host: xg.mediportal.com.cn   //请求资源的Intenet主机和端口号  
Connection: Keep-Alive   //持久连接  
Accept-Encoding: gzip   //浏览器能够进行解码的数据编码方式  
Content-Length: 33   //请求正文的长度

|  |  |
| --- | --- |
| Content-Type | 是返回消息中非常重要的内容，表示后面的文档属于什么MIME类型。Content-Type: [type]/[subtype]; parameter。例如最常见的就是text/html，它的意思是说返回的内容是文本类型，这个文本又是HTML格式的。原则上浏览器会根据Content-Type来决定如何显示返回的消息体内容 |
| Host | 指定请求资源的Intenet主机和端口号，必须表示请求url的原始服务器或网关的位置。HTTP/1.1请求必须包含主机头域，否则系统会以400状态码返回 |
| Accept | 浏览器可接受的MIME类型 |
| Accept-Charset | 浏览器可接受的字符集 |
| Accept-Encoding | 浏览器能够进行解码的数据编码方式，比如gzip。Servlet能够向支持gzip的浏览器返回经gzip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 |
| Accept-Language | 浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到 |
| Authorization | 授权信息，通常出现在对服务器发送的WWW-Authenticate头的应答中 |
| Connection | 表示是否需要持久连接。如果Servlet看到这里的值为“Keep- Alive”，或者看到请求使用的是HTTP1.1（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，Servlet需要在应答中发送一个Content-Length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 |
| Content-Length | 表示请求消息正文的长度 |
| Cookie | 这是最重要的请求头信息之一 |
| From | 请求发送者的email地址，由一些特殊的Web客户程序使用，浏览器不会用到它 |
| Host | 初始URL中的主机和端口 |
| If-Modified-Since | 只有当所请求的内容在指定的日期之后又经过修改才返回它，否则返回304“Not Modified”应答 |
| Pragma | 指定“no-cache”值表示服务器必须返回一个刷新后的文档，即使它是代理服务器而且已经有了页面的本地拷贝 |
| 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 |
| User-Agent | 浏览器类型，如果Servlet返回的内容与浏览器类型有关则该值非常有用 |
| UA-Pixels，UA-Color，UA-OS，UA-CPU | 由某些版本的IE浏览器所发送的非标准的请求头，表示屏幕大小、颜色深度、操作系统和CPU类型 |

 常见的MIME类型如下：

* text/html ： HTML格式
* text/plain ：纯文本格式
* text/xml ：  XML格式
* image/gif ：gif图片格式
* image/jpeg ：jpg图片格式
* image/png：png图片格式

以application开头的媒体格式类型：

* application/xhtml+xml ：XHTML格式
* application/xml     ： XML数据格式
* application/atom+xml  ：Atom XML聚合格式
* application/json    ： JSON数据格式
* application/pdf       ：pdf格式
* application/msword  ： Word文档格式
* application/octet-stream ： 二进制流数据（如常见的文件下载）
* application/x-www-form-urlencoded ： <form encType=””>中默认的encType，form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）

   另外一种常见的媒体格式是上传文件之时使用的：

* multipart/form-data ： 需要在表单中进行文件上传时，就需要使用该格式

（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：

telephone=15527177736&userType=1&

**http响应格式**

HTTP应答与HTTP请求相似，HTTP响应也由3个部分构成，分别是：

1、状态行

2、响应头(Response Header)

3、响应正文

HTTP/1.1 200 OK   //状态行  
Server: nginx  
Date: Tue, 31 May 2016 02:09:24 GMT  
Content-Type: application/json;charset=UTF-8  
Connection: keep-alive  
Vary: Accept-Encoding  
Access-Control-Allow-Origin: \*  
Access-Control-Allow-Headers: X-Requested-With,access\_token,access-token,content-type,multipart/form-data,application/x-www-form-urlencoded  
Access-Control-Allow-Methods: GET,POST,OPTIONS  
Content-Length: 49  
  
{"resultCode":1,"resultMsg":"手机号未注册"}   //正文

（1）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，各元素之间以空格分隔。

状态代码：

状态代码由3位数字组成，表示请求是否被理解或被满足。

状态描述：

状态描述给出了关于状态代码的简短的文字描述。

状态代码的第一个数字定义了响应的类别，后面两位没有具体的分类。

第一个数字有五种可能的取值：

- 1xx:   指示信息—表示请求已接收，继续处理。

- 2xx:   成功—表示请求已经被成功接收、理解、接受。

- 3xx:   重定向—要完成请求必须进行更进一步的操作。

- 4xx:   客户端错误—请求有语法错误或请求无法实现。

- 5xx: 服务器端错误—服务器未能实现合法的请求。

状态代码 状态描述    说明

   200  OK    客户端请求成功

   400  Bad Request   由于客户端请求有语法错误，不能被服务器所理解。

   401  Unauthonzed   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。服务器通常会在响应正文中给出不提供服务的原因

   404   Not Found   请求的资源不存在，例如，输入了错误的URL。

   500  Internal Server Error 服务器发生不可预期的错误，导致无法完成客户端的请求。

  503  Service Unavailable   服务器当前不能够处理客户端的请求，在一段时间之后，服务器可能会恢复正常

**（2）响应头**

响应头可能包括：

**Location：**

Location响应报头域用于重定向接受者到一个新的位置。例如：客户端所请求的页面已不存在原先的位置，为了让客户端重定向到这个页面新的位置，服务 器端可以发回Location响应报头后使用重定向语句，让客户端去访问新的域名所对应的服务器上的资源。当我们在JSP中使用重定向语句的时候，服务器 端向客户端发回的响应报头中，就会有Location响应报头域。

**Server：**

Server响应报头域包含了服务器用来处理请求的软件信息。它和User-Agent请求报头域是相对应的，前者发送服务器端软件的信息，后者发送客户 端软件(浏览器)和操作系统的信息。下面是Server响应报头域的一个例子：Server: Apache-Coyote/1.1

**WWW-Authenticate：**

WWW-Authenticate响应报头域必须被包含在401(未授权的)响应消息中，这个报头域和前面讲到的Authorization请求报头域是 相关的，当客户端收到401响应消息，就要决定是否请求服务器对其进行验证。如果要求服务器对其进行验证，就可以发送一个包含了 Authorization报头域的请求，下面是WWW-Authenticate响应报头域的一个例子：WWW-Authenticate: Basic realm="Basic Auth Test!"

从这个响应报头域，可以知道服务器端对我们所请求的资源采用的是基本验证机制。

**Content-Encoding：**

Content-Encoding实体报头域被使用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容编码，因而要获得Content- Type报头域中所引用的媒体类型，必须采用相应的解码机制。Content-Encoding主要用语记录文档的压缩方法，下面是它的一个例子： Content-Encoding: gzip。如果一个实体正文采用了编码方式存储，在使用之前就必须进行解码。

**Content-Language：**

Content-Language实体报头域描述了资源所用的自然语言。Content-Language允许用户遵照自身的首选语言来识别和区分实体。 如果这个实体内容仅仅打算提供给丹麦的阅读者，那么可以按照如下的方式设置这个实体报头域：Content-Language: da。

如果没有指定Content-Language报头域，那么实体内容将提供给所以语言的阅读者。

**Content-Length：**

Content-Length实体报头域用于指明正文的长度，以字节方式存储的十进制数字来表示，也就是一个数字字符占一个字节，用其对应的ASCII码存储传输。

       要注意的是：这个长度仅仅是表示实体正文的长度，没有包括实体报头的长度。

**Content-Type :**

     Content-Type实体报头域用语指明发送给接收者的实体正文的媒体类型。例如：

Content-Type: text/html;charset=ISO-8859-1

   Content-Type: text/html;charset=GB2312

**Last-Modified :**

     Last-Modified实体报头域用于指示资源最后的修改日期及时间。

**Expires :**

     Expires实体报头域给出响应过期的日期和时间。通常，代理服务器或浏览器会缓存一些页面。当用户再次访问这些页面时，直接从缓存中加载并显示给用 户，这样缩短了响应的时间，减少服务器的负载。为了让代理服务器或浏览器在一段时间后更新页面，我们可以使用Expires实体报头域指定页面过期的时 间。当用户又一次访问页面时，如果Expires报头域给出的日期和时间比Date普通报头域给出的日期和时间要早(或相同)，那么代理服务器或浏览器就 不会再使用缓存的页面而是从服务器上请求更新的页面。不过要注意，即使页面过期了，也并不意味着服务器上的原始资源在此时间之前或之后发生了改变。

      Expires实体报头域使用的日期和时间必须是RFC 1123中的日期格式，例如：

 Expires: Thu, 15 Sep 2005 16:00:00 GMT

       HTTP1.1的客户端和缓存必须将其他非法的日期格式(也包括0)看作已过期。例如，为了让浏览器不要缓存页面，我们也可以利用Expires实体报头 域，设置它的值为0，如下(JSP)：response.setDateHeader("Expires",0);