**HTTP协议**

**HTTP协议简介**

超文本传输协议（英文：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。

HTTP的发展是由蒂姆·伯纳斯-李于1989年在欧洲核子研究组织（CERN）所发起。HTTP的标准制定由万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，最终发布了一系列的RFC，其中最著名的是1999年6月公布的 RFC 2616，定义了HTTP协议中现今广泛使用的一个版本——HTTP 1.1。

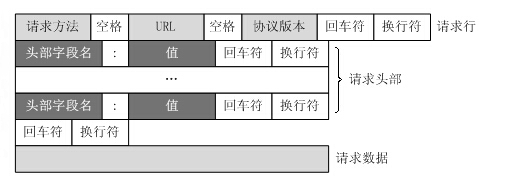
2014年12月，互联网工程任务组（IETF）的Hypertext Transfer Protocol Bis（httpbis）工作小组将HTTP/2标准提议递交至IESG进行讨论，于2015年2月17日被批准。 HTTP/2标准于2015年5月以RFC 7540正式发表，取代HTTP 1.1成为HTTP的实现标准。

**HTTP协议概述**

HTTP是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。通过使用网页浏览器、网络爬虫或者其它的工具，客户端发起一个HTTP请求到服务器上指定端口（默认端口为80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如HTML文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

尽管TCP/IP协议是互联网上最流行的应用，HTTP协议中，并没有规定必须使用它或它支持的层。事实上，HTTP可以在任何互联网协议上，或其他网络上实现。HTTP假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。因此也就是其在TCP/IP协议族使用TCP作为其传输层。

通常，由HTTP客户端发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。HTTP服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。



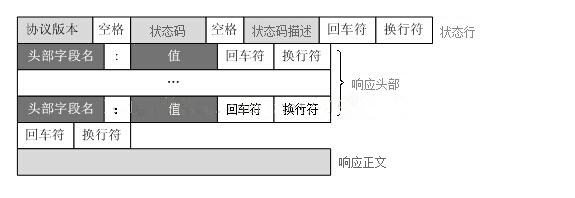
**客户端发送HTTP请求到服务器的请求消息包括以下格式**

**1.请求行（Request Line）**

**2.请求头部（Header）Request Headers**

**3.空行**

**4.请求数据**



**响应头Response Headers信息**

**Allow**服务器的请求方式（GET/POST等）

**Content-Encoding**文档的编码（Encode）方法，只有在解码之后才可以得到Content- Type头指定的内容类型

**Content-Length**内容长度，只在当浏览器使用持久HTTP连接时才需要这个数据

**Content-Type**文档MLME类型

**Date**当前请求时间

**Expires**什么时候文档过期

**Last-Modified**文档最后的改动时间

**Location**表示客户到哪里去提取文档

**Refresh**多少时间刷新文档

**Server**服务器名字

**Set-Cookie**是否关联Cookie

**WWW-Authenticat**客户在Authenticat头中提供什么类型的授权信息

**HTTP工作原理**

HTTP协议定义Web客户端如何从Web服务器请求Web页面，以及服务器如何把Web页面传送给客户端。HTTP协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

**以下是 HTTP 请求/响应的步骤：**

**1. 客户端连接到Web服务器**

一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。例如，http://www.luffycity.com。

**2. 发送HTTP请求**

通过TCP套接字，客户端向Web服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据4部分组成。

**3. 服务器接受请求并返回HTTP响应**

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据4部分组成。

**4. 释放连接TCP连接**

若connection 模式为close，则服务器主动关闭TCP连接，客户端被动关闭连接，释放TCP连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;

**5. 客户端浏览器解析HTML内容**

客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

例如：在浏览器地址栏键入URL，按下回车之后会经历以下流程：

浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;

解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立TCP连接;

浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求，该请求报文作为 TCP 三次握手的第三个报文的数据发送给服务器;

服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;

释放 TCP连接;

浏览器将该 html 文本并显示内容;

**HTTP请求方法**

HTTP/1.1协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源：

**GET**

向指定的资源发出“显示”请求。使用GET方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在Web Application中。其中一个原因是GET可能会被网络蜘蛛等随意访问。（请求的数据包含在URL地址中，数据不会大）

HEAD

与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

**POST**

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。（涉及机密，或者大大数据，属于隐式提交）

PUT

向指定资源位置上传其最新内容。

DELETE

请求服务器删除Request-URI所标识的资源。

TRACE

回显服务器收到的请求，主要用于测试或诊断。

OPTIONS

这个方法可使服务器传回该资源所支持的所有HTTP请求方法。用'\*'来代替资源名称，向Web服务器发送OPTIONS请求，可以测试服务器功能是否正常运作。

CONNECT

HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。通常用于SSL加密服务器的链接（经由非加密的HTTP代理服务器）。

**注意事项：**

方法名称是**区分大小写**的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码501（Not Implemented）。

HTTP服务器至少应该实现GET和HEAD方法，其他方法都是可选的。当然，所有的方法支持的实现都应当匹配下述的方法各自的语义定义。此外，除了上述方法，特定的HTTP服务器还能够扩展自定义的方法。例如PATCH（由 RFC 5789 指定的方法）用于将局部修改应用到资源。

**HTTP状态码General里的Status Code的值**

所有HTTP响应的第一行都是状态行，依次是当前HTTP版本号，3位数字组成的状态代码，以及描述状态的短语，彼此由空格分隔。

**状态代码的第一个数字代表当前响应的类型：**

1\*\*消息——————请求已被服务器接收，继续处理

2\*\*成功——————请求已成功被服务器接收、理解、并接受

3\*\*重定向—————需要后续操作才能完成这一请求

4\*\*请求错误————请求含有词法错误或者无法被执行（发不出去）

5\*\*服务器错误———服务器在处理某个正确请求时发生错误（对方出错）

虽然 RFC 2616 中已经推荐了描述状态的短语，例如"200 OK"，"404 Not Found"，但是WEB开发者仍然能够自行决定采用何种短语，用以显示本地化的状态描述或者自定义信息。

**1开头的状态码**

100 Continue 继续。客户端应继续其请求

101 Switching Protocols 切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议

**2开头的状态码**

200 OK 请求成功。一般用于GET与POST请求

201 Created 已创建。成功请求并创建了新的资源

202 Accepted 已接受。已经接受请求，但未处理完成

203 Non-Authoritative Information 非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本

204 No Content 无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档

205 Reset Content 重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206 Partial Content 部分内容。服务器成功处理了部分GET请求

**3开头的状态码**

300 Multiple Choices 多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择

301 Moved Permanently 永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302 Found 临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

303 See Other 查看其它地址。与301类似。使用GET和POST请求查看

304 Not Modified 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305 Use Proxy 使用代理。所请求的资源必须通过代理访问

306 Unused 已经被废弃的HTTP状态码

307 Temporary Redirect 临时重定向。与302类似。使用GET请求重定向

**4开头的状态码**

400 Bad Request 客户端请求的语法错误，服务器无法理解

401 Unauthorized 请求要求用户的身份认证

402 Payment Required 保留，将来使用

403 Forbidden 服务器理解请求客户端的请求，但是拒绝执行此请求

404 Not Found 服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面

405 Method Not Allowed 客户端请求中的方法被禁止

406 Not Acceptable 服务器无法根据客户端请求的内容特性完成请求

407 Proxy Authentication Required 请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权

408 Request Time-out 服务器等待客户端发送的请求时间过长，超时

409 Conflict 服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突

410 Gone 客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置

411 Length Required 服务器无法处理客户端发送的不带Content-Length的请求信息

412 Precondition Failed 客户端请求信息的先决条件错误

413 Request Entity Too Large 由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息

414 Request-URI Too Large 请求的URI过长（URI通常为网址），服务器无法处理

415 Unsupported Media Type 服务器无法处理请求附带的媒体格式

416 Requested range not satisfiable 客户端请求的范围无效

417 Expectation Failed 服务器无法满足Expect的请求头信息

**5开头的状态码**

500 Internal Server Error 服务器内部错误，无法完成请求

501 Not Implemented 服务器不支持请求的功能，无法完成请求

502 Bad Gateway 充当网关或代理的服务器，从远端服务器接收到了一个无效的请求

503 Service Unavailable 由于超载或系统维护，服务器暂时的无法处理客户端的请求。延 时的长度可包含在服务器的Retry-After头信息中

504 Gateway Time-out 充当网关或代理的服务器，未及时从远端服务器获取请求

505 HTTP Version not supported 服务器不支持请求的HTTP协议的版本，无法完成处理

**URL**

超文本传输协议（HTTP）的统一资源定位符将从因特网获取信息的五个基本元素包括在一个简单的地址中：

**https://www.baidu.com：80/search/img/ddd.html#bottom?name=淘宝&sss=123456**

协议：http:// https://

域名：www.baidu.com

资源：search/img/ddd.html

锚点：#bottom

参数：?name=淘宝&sss=123456

**传送协议。**

层级URL标记符号(为[//],固定不变)

访问资源需要的凭证信息（可省略）

服务器.（通常为域名，有时为IP地址）

端口号.（以数字方式表示，若为HTTP的默认值“:80”可省略）

路径.（以“/”字符区别路径中的每一个目录名称）

查询.（GET模式的窗体参数，以“?”字符为起点，每个参数以“&”隔开，再以“=”分开参数名称与数据，通常以UTF8的URL编码，避开字符冲突的问题）

片段。以“#”字符为起点

**以http://www.luffycity.com:80/news/index.html?id=250&page=1** 为例, 其中：

**http**是协议；

**www.luffycity.com**是服务器；

**80**是服务器上的网络端口号；

**/news/index.html**是路径；

**?id=250&page=1**是查询。

大多数网页浏览器不要求用户输入网页中“http://”的部分，因为绝大多数网页内容是超文本传输协议文件。同样，“80”是超文本传输协议文件的常用端口号，因此一般也不必写明。一般来说用户只要键入统一资源定位符的一部分（www.luffycity.com:80/news/index.html?id=250&page=1）就可以了。

由于超文本传输协议允许服务器将浏览器重定向到另一个网页地址，因此许多服务器允许用户省略网页地址中的部分，比如 www。从技术上来说这样省略后的网页地址实际上是一个不同的网页地址，浏览器本身无法决定这个新地址是否通，服务器必须完成重定向的任务。

get和post的区别

1. GET在浏览器回退时是无害的，而POST会再次提交请求。

2. GET产生的URL地址可以被Bookmark（书签），而POST不可以

3. GET请求会被浏览器主动Cache，而POST不会，除非手动设置。

4. GET请求只能进行url编码，而POST支持多种编码方式。

5. GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

6. GET请求在URL中传送的参数是有长度限制的，而POST没有。

7. 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。

8. GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。

9. GET参数通过URL传递，POST放在Request body中。

**Cache**

Cache存储器，电脑中为高速缓冲存储器，是位于CPU和主存储器DRAM（Dynamic Random Access Memory）之间，规模较小，但速度很高的存储器，通常由SRAM（Static Random Access Memory 静态存储器）组成。**它是位于CPU与内存间的一种容量较小但速度很高的存储器。CPU的速度远高于内存，当CPU直接从内存中存取数据时要等待一定时间周期，而Cache则可以保存CPU刚用过或循环使用的一部分数据，如果CPU需要再次使用该部分数据时可从Cache中直接调用，这样就避免了重复存取数据，减少了CPU的等待时间，因而提高了系统的效率。**Cache又分为L1Cache（一级缓存）和L2Cache（二级缓存），L1Cache主要是集成在CPU内部，C而L2Cache集成在主板上或是CPU上