HTTP 协议构建于 TCP/IP 协议之上，是一个应用层协议，默认端口号是 80

HTTP 是无连接无状态的

http报文

http请求报文

计算机生成了可选文字:
空袼
URL
回车换行
版本CRLF请求行
字段名
字段名
CRLF
主体
值CRLF
请求头部
值CRLF

请求行：

方法：GET，POST，…

URL:资源描述符

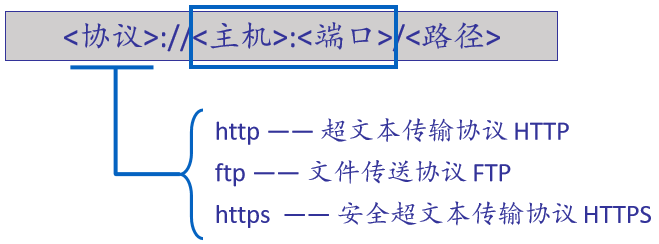
HTTP版本

HTTP请求头部

变长、可读字符串

包含：主机、认证、缓存、提交者、用户代理、连接管理

URL统一资源定位符: 对资源的位置和访问方法描述s



http应答报文

计算机生成了可选文字:
版本状态码短语CRLF
状态行
首部字段名
首部字段名
CRLF
主体
值CRLF
应答头部
值CRLF

状态行：

协议版本：

状态码：

状态描述：

响应头：

响应正文：

状态码：

计算机生成了可选文字:
状态码
IXX
2XX
4XX
5XX
成功
重定向
客户端
讠日1天
服务端
错误
接收到请求，继续处理
操作成功地收到，理解和接受
为了完成请求，必须采取进一
步措施
请求的语法有错误或不能完全
被满足。
服务器无法完成明显有效的请
求。
示伍
100Continue
200OK
301MovedPermanently
404NotFound
500InternalServerError

1. GET 用于信息获取，而且应该是安全的 和 幂等的。

所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET 请求一般不应产生副作用。就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改，增加数据，不会影响资源的状态。

幂等的意味着对同一 URL 的多个请求应该返回同样的结果。

GET 请求报文示例：

GET /books/?sex=man&name=Professional HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)

Gecko/20050225 Firefox/1.0.1

Connection: Keep-Alive

POST 表示可能修改变服务器上的资源的请求。

POST / HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)

Gecko/20050225 Firefox/1.0.1

Content-Type: application/x-www-form-urlencoded

Content-Length: 40

Connection: Keep-Alive

sex=man&name=Professional

注意:

* GET 可提交的数据量受到URL长度的限制，HTTP 协议规范没有对 URL 长度进行限制。这个限制是特定的浏览器及服务器对它的限制
* 理论上讲，POST 是没有大小限制的，HTTP 协议规范也没有进行大小限制，出于安全考虑，服务器软件在实现时会做一定限制
* 参考上面的报文示例，可以发现 GET 和 POST 数据内容是一模一样的，只是位置不同，一个在 URL 里，一个在 HTTP 包的包体里

### POST 提交数据的方式

HTTP 协议中规定 POST 提交的数据必须在 body 部分中，但是协议中没有规定数据使用哪种编码方式或者数据格式。实际上，开发者完全可以自己决定消息主体的格式，只要最后发送的 HTTP 请求满足上面的格式就可以。

但是，数据发送出去，还要服务端解析成功才有意义。一般服务端语言如 PHP、Python 等，以及它们的 framework，都内置了自动解析常见数据格式的功能。服务端通常是根据请求头（headers）中的 Content-Type 字段来获知请求中的消息主体是用何种方式编码，再对主体进行解析。所以说到 POST 提交数据方案，包含了 Content-Type 和消息主体编码方式两部分。下面就正式开始介绍它们：

* application/x-www-form-urlencoded

这是最常见的 POST 数据提交方式。浏览器的原生 <form> 表单，如果不设置 enctype 属性，那么最终就会以 application/x-www-form-urlencoded 方式提交数据。上个小节当中的例子便是使用了这种提交方式。可以看到 body 当中的内容和 GET 请求是完全相同的。

* multipart/form-data

这又是一个常见的 POST 数据提交的方式。我们使用表单上传文件时，必须让 <form> 表单的 enctype 等于 multipart/form-data。直接来看一个请求示例：

POST http://www.example.com HTTP/1.1

Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryrGKCBY7qhFd3TrwA

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="text"

title

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="file"; filename="chrome.png"

Content-Type: image/png

PNG ... content of chrome.png ...

------WebKitFormBoundaryrGKCBY7qhFd3TrwA--

这个例子稍微复杂点。首先生成了一个 boundary 用于分割不同的字段，为了避免与正文内容重复，boundary 很长很复杂。然后 Content-Type 里指明了数据是以 multipart/form-data 来编码，本次请求的 boundary 是什么内容。消息主体里按照字段个数又分为多个结构类似的部分，每部分都是以 --boundary 开始，紧接着是内容描述信息，然后是回车，最后是字段具体内容（文本或二进制）。如果传输的是文件，还要包含文件名和文件类型信息。消息主体最后以 --boundary-- 标示结束。关于 multipart/form-data 的详细定义，请前往 [RFC1867](http://www.ietf.org/rfc/rfc1867.txt) 查看（或者相对友好一点的 [MDN 文档](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Disposition)）。

这种方式一般用来上传文件，各大服务端语言对它也有着良好的支持。

上面提到的这两种 POST 数据的方式，都是浏览器原生支持的，而且现阶段标准中原生 <form> 表单也只支持这两种方式（通过 <form> 元素的 enctype 属性指定，默认为 application/x-www-form-urlencoded。其实 enctype 还支持 text/plain，不过用得非常少）。

随着越来越多的 Web 站点，尤其是 WebApp，全部使用 Ajax 进行数据交互之后，我们完全可以定义新的数据提交方式，例如 application/json，text/xml，乃至 application/x-protobuf 这种二进制格式，只要服务器可以根据 Content-Type 和 Content-Encoding 正确地解析出请求，都是没有问题的。

### 条件 GET

HTTP 条件 GET 是 HTTP 协议为了减少不必要的带宽浪费，提出的一种方案。详见 [RFC2616](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html) 。

1. HTTP 条件 GET 使用的时机？

客户端之前已经访问过某网站，并打算再次访问该网站。

1. HTTP 条件 GET 使用的方法？

客户端向服务器发送一个包询问是否在上一次访问网站的时间后是否更改了页面，如果服务器没有更新，显然不需要把整个网页传给客户端，客户端只要使用本地缓存即可，如果服务器对照客户端给出的时间已经更新了客户端请求的网页，则发送这个更新了的网页给用户。

下面是一个具体的发送接受报文示例：

客户端发送请求：

GET / HTTP/1.1

Host: www.sina.com.cn:80

If-Modified-Since:Thu, 4 Feb 2010 20:39:13 GMT

Connection: Close

第一次请求时，服务器端返回请求数据，之后的请求，服务器根据请求中的 If-Modified-Since 字段判断响应文件没有更新，如果没有更新，服务器返回一个 304 Not Modified响应，告诉浏览器请求的资源在浏览器上没有更新，可以使用已缓存的上次获取的文件。

HTTP/1.0 304 Not Modified

Date: Thu, 04 Feb 2010 12:38:41 GMT

Content-Type: text/html

Expires: Thu, 04 Feb 2010 12:39:41 GMT

Last-Modified: Thu, 04 Feb 2010 12:29:04 GMT

Age: 28

X-Cache: HIT from sy32-21.sina.com.cn

Connection: close

如果服务器端资源已经更新的话，就返回正常的响应。