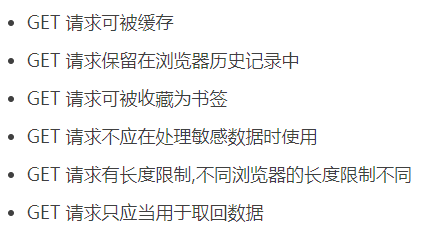
HTTP是超文本传输协议，其定义了客户端与服务器端之间文本传输的规范。HTTP默认使用80端口，这个端口指的是服务端的端口，而客户端使用的端口是动态分配的。当我们没有指定端口访问时，浏览器会默认帮我们添加80端口。我们也可以自己指定访问端口如：http://www.ip138.com:80。 需要注意的是，现在大多数访问都使用了HTTPS协议，而HTTPS的默认端口为443，如果使用80端口访问HTTPS协议的服务器可能会被拒绝。

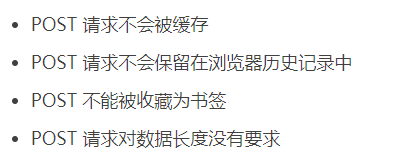
根据 HTTP 标准，HTTP 请求可以使用多种请求方法。  
HTTP1.0 定义了三种请求方法： GET, POST 和 HEAD方法。  
HTTP1.1 新增了六种请求方法：OPTIONS、PUT、PATCH、DELETE、TRACE 和 CONNECT 方法。



1. GET方法：向特定的路径资源发出请求，数据暴露在url中



1. POST方法：向指定路径资源提交数据进行处理请求（一般用于上传表单或者文件），数据包含在请求体中。理论上，POST传递的数据量没有限制。



1. PUT方法：PUT方法用来传输文件，就像FTP协议的文件上传一样，要求在请求报文的主体中包含文件内容，然后保存在请求URL指定的位置。但是HTTP/1.1的PUT方法自身不带验证机制，任何人都可以上传文件，存在安全问题，故一般不用。（如果服务器有请求的文件就更新，如果不存在就新建一个）
2. HEAD方法：HEAD方法跟GET方法相同，只不过服务器响应时不会返回消息体。一个HEAD请求的响应中，HTTP头中包含的元信息应该和一个GET请求的响应消息相同。这种方法可以用来获取请求中隐含的元信息，而不用传输实体本身。也经常用来测试超链接的有效性、可用性和最近的修改。

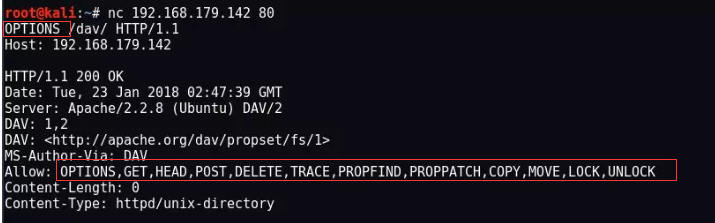
一个HEAD请求的响应可被缓存，也就是说，响应中的信息可能用来更新之前缓存的实体。如果当前实体跟缓存实体的阈值不同（可通过Content-Length、Content-MD5、ETag或Last-Modified的变化来表明），那么这个缓存就被视为过期了。

HEAD方法和GET方法的区别： GET方法有实体，HEAD方法无实体。

5、DELETE方法： 指明客户端想让服务器删除某个资源，与PUT方法相反，按URL删除指定资源

DELETE方法唯一有趣的地方在于当你接收了一个标识为200 OK的响应的时候，那并不意味着指定的资源已经被删除了。那仅仅说明服务器接收到了删除资源的命令。

6、OPTIONS方法：OPTIONS方法用来查询针对请求URL指定资源服务器支持的方法（客户端询问服务器可以提交哪些请求方法）



1. CONNECT方法：CONNECT这个方法的作用就是把服务器作为跳板，让服务器代替用户去访问其它网页，之后把数据原原本本的返回给用户。这样用户就可以访问到一些只有服务器上才能访问到的网站了，这就是HTTP代理。

CONNECT方法是需要使用TCP直接去连接的，所以不适合在网页开发中使用，不过网页开发中也用不到这玩意儿。要是使用CONNECT方法，首先要让服务器监听一个端口来接收CONNECT方法的请求。这个是服务器软件做的事情，我们只要配置好它就可以了，除非你闲着无聊想自己实现一个这样的服务器。在服务器监听了端口以后就是客户端的请求，我们必须告诉代理服务器我们想要访问哪个Internet服务器。假如我想通过代理访问这个博客（www.web-tinker.com），我就需要建立一个TCP连接，连接到服务器监听的那个端口，然后给服务器发送一个HTTP头。下面就是这个HTTP头的内容：

CONNECT www.web-tinker.com:80 HTTP/1.1

Host: www.web-tinker.com:80

Proxy-Connection: Keep-Alive

Proxy-Authorization: Basic \*

Content-Length: 0

所有的HTTP头都是类似的，第一行是方法名、主要参数、HTTP版本。接着一行一个参数，最后用两个换行来结束。这个HTTP头其实也没什么好介绍的，唯一一个重点的地方就是星号的部分，这个地方应该填写验证的用户名和密码。而且，用户名和密码也是有固定格式的。要把用户名和密码用冒号连接起来，再经过BASE64的编码后才可以使用。假如用户名是abc密码是123，那么星号的地方就应该换上YWJjOjEyMw==，也就是abc:123经过BASE64编码的结果。

　　发送完这个请求之后，就是服务器端响应请求了。如果用户名和密码验证通过，就会返回一个状态码为200的响应信息。虽然状态码是200，但是这个状态描述不是OK，而是Connection Established。

**HTTP/1.1 200 Connection Established**

　　如果用户名和密码验证不通过。会返回一个407的状态码，状态表述是Unauthorized。表示没有权限访问代理服务器。

**HTTP/1.1 407 Unauthorized**

验证失败的情况有时候还会带上一堆HTML，这是有些服务器为了让网页上在连接失败时显示用的，如果不是通过浏览器来连接的话无视就好了。无论验证成功还是验证失败，这些服务器返回的信息在不同的服务器软件上会有一些差异。比如有些服务器软件返回这些代码会使用HTTP/1.0，有些则会在后面加上个代表服务器版本的字段。这些信息都无所谓，对于服务器返回的数据，我们关键是看状态码。

验证通过之后，我们就可以做普通的HTTP操作了。完全可以把现在的代理服务器看作是请求连接的Internet服务器，也就是说可以像直接访问普通的服务器一样，使用GET、POST等方法来请求Internet服务器上的页面了。我们在发送CONNECT请求的时候就已经告诉了服务器我们需要访问的Internet服务器，上面我用了这个博客的网址。现在我们要访问这个博客的主页就可以发送一个简单的GET请求。

GET / HTTP/1.1

Host: www.web-tinker.com

Content-Length: 0

这个就是普通的GET请求的操作了。

参考链接（<https://www.cnblogs.com/niie9/p/6727077.html>）

1. TRACE方法：TRACE方法是HTTP（超文本传输）协议定义的一种协议调试方法，回显服务器收到的请求，主要用于测试或诊断。该方法使得服务器原样返回任何客户端请求的内容（可能会附加路由中间的代理服务器的信息），由于该方法原样返回客户端提交的任意数据，因此，可用来进行跨站脚本（XSS）攻击，这种攻击方式又称为跨站跟踪攻击（XST）。

**启用TRACE方法存在如下风险**：

1、恶意攻击者可以通过TRACE方法返回的信息了解到网站前端的某些信息，如缓存服务器等，从而为进一步的攻击提供便利。

2、恶意攻击者可以通过TRACE方法进行XSS攻击。

3、即使网站对关键页面启用了HttpOnly头标记和禁止脚本读取cookie信息，但是通过TRACE 方法恶意攻击者还是可以绕过这个限制读取到cookie信息。

**补充：**

1. HTTP协议的交互过程

HTTP的交互流程简单来讲就是客户端与服务器端的通信，包括客户端对服务器端的请求以及服务器端对客户端的响应。

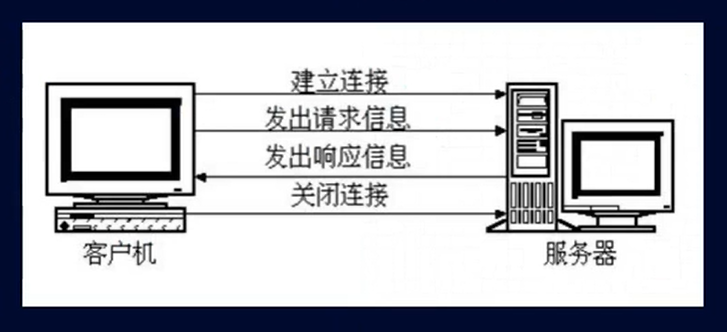
首先客户端与服务器端建立一个连接，三次握手经历完成之后才能建立一个稳定可靠的连接。

“三次握手”。第一次握手：客户端给服务器端发送一个syn的标志位；服务器端接收到syn后会返回一个ack(相当于一个回调的机制)，同时还有一个服务器端的syn；客户端接收服务器端发送的syn后会再次给服务器端发送一个ack，这样才算完成三次握手。

然后客户端就可以向服务器端发送请求并且服务器端也会给客户端响应了。在HTTP1.1后，他们之间的连接就是可持续的连接，也叫作常连接。客户端可以向服务器端发送多个请求并且得到多个响应。

当客户端不再发送请求给服务器端并且服务器端没有响应发送给客户端时，就可以断开连接了。

这里面有一个四次分手的原则。客户端向服务器端发送断开连接的请求；服务器端接收到请求后，返回可以断开连接的请求，客户端断开连接并且释放资源；服务器端向客户端发送断开连接的信息；客户端向服务器端发送同意断开连接的信息，服务器端断开连接释放资源。



1. HTTP 请求/响应的步骤

客户端连接到Web服务器->发送Http请求->服务器接受请求并返回HTTP响应->释放TCP连接->客户端浏览器解析HTML内容

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.baidu.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。

1. GET和POST的区别

首先GET和POST是什么？他们是HTTP协议中两种发送请求的方式。HTTP是什么？HTTP是基于TCP与IP的关于数据在万维网中如何通信的协议。HTTP的底层是TCP/IP，也就是说GET与POST都是TCP链接。GET与POST做的事是一样的，都可以传输数据。因此GET与POST在本质上没有区别，而真正的区别在于TCP链接的不同，由于在万维网中各个浏览器以及服务器的限制，导致他们在引用过程中体现的不同（比如传输数据大小的限制）。

那么真正的区别在哪呢？我认为真正的区别在于TCP数据包，GET方式产生一个TCP数据包，而POST方式会产生两个TCP数据包。详细的说，对于GET请求，浏览器会把http header和data一并发送出去，服务器响应200ms后返回数据。而POST请求，浏览器会先发送http header服务器响应100 continue，浏览器再发送data，服务器响应200ms后再返回数据。

但是呢，在网络条件好的情况下，发送一次和发送两次数据包的时间差是可以直接忽略无视的。只有在网络条件差的时候，发送两次数据包在TCP的验证数据上会更加的稳定。