Get和post 的区别：<https://www.cnblogs.com/logsharing/p/8448446.html>

Http请求头：<https://www.cnblogs.com/printN/p/6534529.html>

Uri , url , urn ：<https://www.cnblogs.com/yplong/p/5572033.html>

# HTTP

## TTP的全称是什么？

超文本传输协议，HyperText Transfter Protocol，这几个单词可别发走音了。所谓的超文本就是带标记的文本，刚开始的时候是指HTML。现在HTTP协议传输的东西可不只是HTML了，什么表单啊JSON啊XML啊文件啊都可以传输。

## HTTP常用的状态码有哪些？

200：请求被正常处理

204：请求被受理但没有资源可以返回

206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。

301：永久性重定向

302：临时重定向

303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上

304：发送附带条件的请求时，条件不满足时返回，与重定向无关

307：临时重定向，与302类似，只是强制要求使用POST方法

400：请求报文语法有误，服务器无法识别

401：请求需要认证

403：请求的对应资源禁止被访问

404：服务器无法找到对应资源

500：服务器内部错误

503：服务器正忙

## HTTP协议的8种请求？

HTTP 协议中共定义了八种方法或者叫“动作”来表明对 Request-URI 指定的资源的不同操作方式，具体介绍如下：

OPTIONS：返回服务器针对特定资源所支持的HTTP请求方法。也可以利用向Web服务器发送'\*'的请求来测试服务器的功能性。

HEAD：向服务器索要与GET请求相一致的响应，只不过响应体将不会被返回。这一方法可以在不必传输整个响应内容的情况下，就可以获取包含在响应消息头中的元信息。

GET：向特定的资源发出请求。

POST：向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的创建和/或已有资源的修改。

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。

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。

虽然 HTTP 的请求方式有 8 种，但是我们在实际应用中常用的也就是 get 和 post，其他请求方式也都可以通过这两种方式间接的来实现。

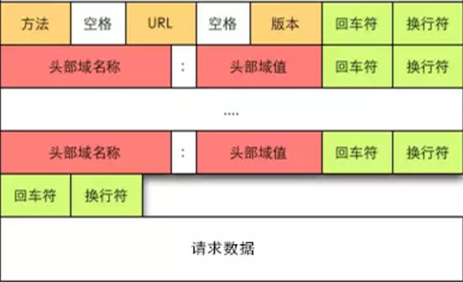


## HTTP协议结构

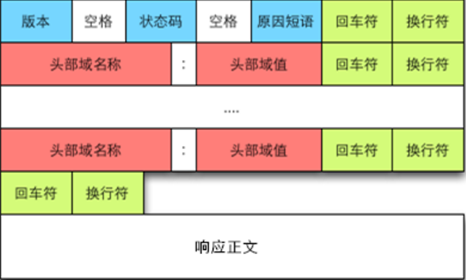
HTTP的请求和响应的消息协议是一样的，分为三个部分，起始行、消息头和消息体。这三个部分以CRLF作为分隔符。最后一个消息头有两个CRLF，用来表示消息头部的结束。

1.请求行 2.请求头 3.空行 4.消息主体

下图为http请求的报文结构



下图为http响应报文结构



**通用首部字段（请求报文与响应报文都会使用的首部字段）**

Date：创建报文时间

Connection：连接的管理

Cache-Control：缓存的控制

Transfer-Encoding：报文主体的传输编码方式

**请求首部字段（请求报文会使用的首部字段）**

Host：请求资源所在服务器

Accept：可处理的媒体类型

Accept-Charset：可接收的字符集

Accept-Encoding：可接受的内容编码

Accept-Language：可接受的自然语言

**响应首部字段（响应报文会使用的首部字段）**

Accept-Ranges：可接受的字节范围

Location：令客户端重新定向到的URI

Server：HTTP服务器的安装信息

**实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）**

Allow：资源可支持的HTTP方法

Content-Type：实体主类的类型

Content-Encoding：实体主体适用的编码方式

Content-Language：实体主体的自然语言

Content-Length：实体主体的的字节数

Content-Range：实体主体的位置范围，一般用于发出部分请求时使用

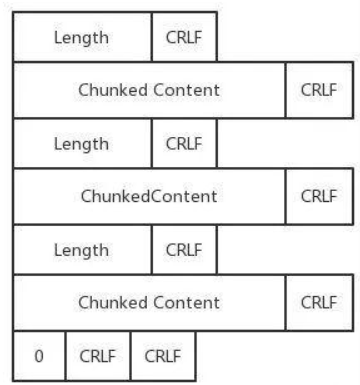
## 什么是分块传送

当浏览器向服务器请求一个资源时，这个资源是一个动态资源，服务器无法提前预知资源的大小，这个时候就可以使用分块传输。

服务器先生成一个chunk，发送这个chunk，再生成一个chunk，再发送一个chunk，直到全部资源传送完成。

分块传送需要在请求头增加一个特殊的键值对transfer-encoding: chunked，那么消息体的内容便是分块传送的。

chunked传输格式如图所示，由一段一段的分块组合而成，每个块由一个长度行和一个分块体组成，最后一个分块长度为0表示结束。



## 持久连接的机制是怎样的？

HTTP早期版本中每个请求都会发起一个连接，一个网页除了页面的HTML之外还会有很多静态资源以及诸多的API调用，如果每个请求都一个连接，势必网页的一次加载就会和服务器创建多次连接，这是非常浪费服务器资源的，同时也让客户端的访问速度慢了不少。HTTP1.0之后引入了Keep-Alive持久连接，在HTTP1.1版本中成为默认选项。它使得HTTP的一个连接可以连续服务多个请求，有效节省了资源，增加了客户端页面的加载速度。

持久连接也不宜一直保持，毕竟每个连接都会占用服务器资源，如果打开网页的人太多，那服务器资源也会紧张，所以一般服务器都会配置一个KeepAlive Timeout参数和KeepAlive Requests参数限制单个连接持续时长和最多服务的请求次数。

如果服务器设置的timeout时长为0，就退化到非持久连接。非持久连接会在响应头部增加一个头信息Connection: Close通知客户端在接受完当前响应后连接需要立即关闭。

同样浏览器也不会因为服务器将KeepAlive Timeout配置了无限长就不管不问一直持续保持连接。每个浏览器都有它自己的内置限制，具体限制浏览器厂商各有不同。

## 什么叫Pipeline管线化？

HTTP1.0不支持管线化，同一个连接处理请求的顺序是逐个应答模式，处理一个请求就需要耗费一个TTL，也就是客户端到服务器的往返时间，处理N个请求就是N个TTL时长。当页面的请求非常多时，页面加载速度就会非常缓慢。

从HTTP1.1开始要求服务器支持管线化，可以同时将多个请求发送到服务器，然后逐个读取响应。这个管线化和Redis的管线化原理是一样的，响应的顺序必须和请求的顺序保持一致。

## 如何理解HTTP协议的无状态性？

所谓HTTP协议的无状态性是指服务器的协议层无需为不同的请求之间建立任何相关关系，它特指的是协议层的无状态性。但是这并不代表建立在HTTP协议之上的应用程序就无法维持状态。应用层可以通过会话Session来跟踪用户请求之间的相关性，服务器会为每个会话对象绑定一个唯一的会话ID，浏览器可以将会话ID记录在本地缓存LocalStorage或者Cookie，在后续的请求都带上这个会话ID，服务器就可以为每个请求找到相应的会话状态。

## 什么是Http协议无状态协议?怎么解决Http协议无状态协议?

无状态协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息

也就是说，当客户端一次HTTP请求完成以后，客户端再发送一次HTTP请求，HTTP并不知道当前客户端是一个”老用户“。

可以使用Cookie来解决无状态的问题，Cookie就相当于一个通行证，第一次访问的时候给客户端发送一个Cookie，当客户端再次来的时候，拿着Cookie(通行证)，那么服务器就知道这个是”老用户“。

## HTTP1.1版本新特性

a、默认持久连接节省通信量，只要客户端服务端任意一端没有明确提出断开TCP连接，就一直保持连接，可以发送多次HTTP请求

b、管线化，客户端可以同时发出多个HTTP请求，而不用一个个等待响应

c、断点续传，实际上就是利用HTTP消息头使用分块传输编码，将实体主体分块传输。

## HTTP优化方案

TCP复用：TCP连接复用是将多个客户端的HTTP请求复用到一个服务器端TCP连接上，而HTTP复用则是一个客户端的多个HTTP请求通过一个TCP连接进行处理。前者是负载均衡设备的独特功能；而后者是HTTP 1.1协议所支持的新功能，目前被大多数浏览器所支持。

内容缓存：将经常用到的内容进行缓存起来，那么客户端就可以直接在内存中获取相应的数据了。

压缩：将文本数据进行压缩，减少带宽

SSL加速（SSL Acceleration）：使用SSL协议对HTTP协议进行加密，在通道内加密并加速

TCP缓冲：通过采用TCP缓冲技术，可以提高服务器端响应时间和处理效率，减少由于通信链路问题给服务器造成的连接负担。

## Http与Https的区别：

HTTP 的URL 以http:// 开头，而HTTPS 的URL 以https:// 开头

HTTP 是不安全的，而 HTTPS 是安全的

HTTP 标准端口是80 ，而 HTTPS 的标准端口是443

在OSI 网络模型中，HTTP工作于应用层，而HTTPS 的安全传输机制工作在传输层

HTTP 无法加密，而HTTPS 对传输的数据进行加密

HTTP无需证书，而HTTPS 需要CA机构wosign的颁发的SSL证书

## HTTPS工作原理

一、首先HTTP请求服务端生成证书，客户端对证书的有效期、合法性、域名是否与请求的域名一致、证书的公钥（RSA加密）等进行校验；

二、客户端如果校验通过后，就根据证书的公钥的有效， 生成随机数，随机数使用公钥进行加密（RSA加密）；

三、消息体产生的后，对它的摘要进行MD5（或者SHA1）算法加密，此时就得到了RSA签名；

四、发送给服务端，此时只有服务端（RSA私钥）能解密。

五、解密得到的随机数，再用AES加密，作为密钥（此时的密钥只有客户端和服务端知道）。

## 一次完整的HTTP请求所经历的7个步骤

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：

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，才能进行更层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。

Web浏览器向Web服务器发送请求行

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。

Web浏览器发送请求头

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。

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。

Web服务器发送应答头

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。

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。

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。

建立TCP连接->发送请求行->发送请求头->（到达服务器）发送状态行->发送响应头->发送响应数据->断TCP连接