HTTP 协议

在 OSI 七层模型中，HTTP 协议位于最顶层的应用层中。通过浏览器访问网页就直接使用了 HTTP 协议。使用 HTTP 协议时，客户端首先与服务端的 80 端口建立一个 TCP 连接，然后在这个连接的基础上进行请求和应答，以及数据的交换。

HTTP 有两个常用版本，分别是 1.0 和 1.1。主要区别在于 HTTP 1.0 中每次请求和应答都会使用一个新的 TCP 连接，而从 HTTP 1.1 开始，运行在一个 TCP 连接上发送多个命令和应答。因此大幅度减少了 TCP 连接的建立和断开，提高了效率。

常用的 HTTP 方法有哪些？

- GET：用于请求访问已经被 URL（统一资源标识符）识别的资源，可以通过 URL 传参给服务器。
- POST：用于传输信息给服务器，主要功能与 Get 方法类似，但一般推荐 POST 方式。
- PUT：传输文件，报文主体包含文件内容，保存到对应 URL 位置。
- HEAD：获取报文首部，与 GET 方法类似，只是不返回报文主体，一般用于验证 URL 是否有效。
- DELET：删除文件，与 PUT 方法相反，删除对应 URL 位置的文件。
- OPTIONS：查询相应 URL 支持的 HTTP 方法。

get 和 post 的区别

1. get 是从服务器上获取数据，post 是向服务器传送数据。
2. get 是把参数数据队列加到提交表单的 ACTION 属性所指的 URL 中，值和表单内各个字段一一对应，在 URL 中可以看到。post 是通过 HTTP 的 post 机制，将表单内各个字段与其内容放置在 HTML header 内一起传送到 ACTION 属性所指的 URL 地址。用户看不到这个过程。
3. 对于 get 方式，服务器端用 Request.QueryString 获取变量的值，对于 post 方式，服务器端用 Request.Form 获取提交的数据。

4. get 传送的数据量较小，因为受 URL 限制，不能大于 2KB，但是效率高。
   post 传送的数据量较大，一般被默认为不受限制，所以上传文件时只能用 post。
   但理论上，IIS4 中最大量为 80KB，IIS5 中为 100KB。
5. get 安全性非常低，因为 URL 是可见的，可能会泄露私密信息，如密码等，
   post 安全性较高。但是执行效率却比 Post 方法好。
6. get 方式只能支持 ASCII 字符，向服务器传的中文字符可能会乱码。
   post 支持标准字符集，可以正确传递中文字符。
7. get 请求可以被缓存，可以被收藏为书签，但 post 不行。
8. get 请求会保留在浏览器的历史记录中，post 不会。

SO：

1、get 方式的安全性较 Post 方式要差些，包含机密信息的话，建议用 Post 数据提交方式；

2、在做数据查询时，建议用 Get 方式；而在做数据添加、修改或删除时，建议用 Post 方式

PS： POST 请求仅比 GET 请求略安全一点，它的数据不在 URL 中，但依然以明文的形式存放于 HTTP 的请求头中。

HTTP 请求报文与响应报文格式

请求报文包含三部分：

- 请求行：包含请求方法、URI、HTTP 版本信息
- 请求首部字段
- 请求内容实体

响应报文包含三部分：

- 状态行：包含 HTTP 版本、状态码、状态码的原因短语
- 响应首部字段
- 响应内容实体

常见的 HTTP 相应状态码

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

常见 HTTP 首部字段

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

一次完整的 HTTP 请求事务包含以下四个环节

1. 建立起客户机和服务器连接。
2. 建立连接后，客户机发送一个请求给服务器。
3. 服务器收到请求给予响应信息。
4. 客户端浏览器将返回的内容解析并呈现，断开连接。

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->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

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。

HTTP 优化

- 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 加密（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 的缺点

- 通信使用明文不加密，内容可能被窃听
- 不验证通信方身份，可能遭到伪装
- 无法验证报文完整性，可能被篡改

HTTP1.1 版本新特性

- 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 - 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
- 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）

Cookie 和 Session 的区别

HTTP 是一种无状态的连接，客户端每次读取 web 页面时，服务器都会认为这是一次新的会话。但有时候我们又需要持久保持某些信息，比如登录时的用户

名、密码，用户上一次连接时的信息等。这些信息就由 Cookie 和 Session 保存。

Cookie

cookie 实际上是一小段文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用 response 向客户端浏览器颁发一个 cookie，客户端浏览器会把 cookie 保存起来，当浏览器再次请求访问该网站时，浏览器把请求的网站连同该 cookie 一同提交给服务器，服务器检查该 cookie，以此来辨认用户状态。

1. 简单来说，cookie 的工作原理可总结如下：client 连接 server
2. client 生成 cookie（有效期）,再次访问时携带 cookie
3. server 根据 cookie 的信息识别用户身份

Session

Session 是服务器端使用的一种记录客户端状态的机制，使用上比 Cookie 简单一些。同一个客户端每次和服务端交互时，不需要每次都传回所有的 Cookie 值，而是只要传回一个 ID，这个 ID 是客户端第一次访问服务器的时候生成的，而且每个客户端是唯一的。这样每个客户端就有了一个唯一的 ID，客户端只要传回这个 ID 就行了，这个 ID 通常是 name 为 JSESIONID 的一个 Cookie。Session 依据这个 id 来识别是否为同一用户（只认 ID 不认人）。

区别：

1. cookie 数据存放在客户的浏览器上，session 数据放在服务器上。
2. cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗
   考虑到安全应当使用 session。
3. session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能
   考虑到减轻服务器性能方面，应当使用 COOKIE。

4. 单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。