HTTP 是一个属于应用层的面向对象的协议，HTTP 协议一共有五大特点：1、支持客户/服务器模式；2、简单快速；3、灵活；**4、无连接；5、无状态**。

**无连接**

**无连接的含义**是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

早期这么做的原因是 HTTP 协议产生于互联网，因此服务器需要处理同时面向全世界数十万、上百万客户端的网页访问，但每个客户端（即浏览器）与服务器之间交换数据的间歇性较大（即传输具有突发性、瞬时性），并且网页浏览的联想性、发散性导致两次传送的数据关联性很低，大部分通道实际上会很空闲、无端占用资源。因此 HTTP 的设计者有意利用这种特点将协议设计为**请求时建连接、请求完释放连接，以尽快将资源释放出来服务其他客户端**。

随着时间的推移，网页变得越来越复杂，里面可能嵌入了很多图片，这时候每次访问图片都需要建立一次 TCP 连接就显得很低效。后来，Keep-Alive 被提出用来解决这效率低的问题。

**Keep-Alive 功能使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，Keep-Alive 功能避免了建立或者重新建立连接**。市场上的大部分 Web 服务器，包括 iPlanet、IIS 和 Apache，都支持 HTTP Keep-Alive。对于提供静态内容的网站来说，这个功能通常很有用。但是，对于负担较重的网站来说，这里存在另外一个问题：虽然为客户保留打开的连接有一定的好处，但它同样影响了性能，因为在处理暂停期间，本来可以释放的资源仍旧被占用。当Web服务器和应用服务器在同一台机器上运行时，Keep-Alive 功能对资源利用的影响尤其突出。 

这样一来，客户端和服务器之间的 HTTP 连接就会被保持，不会断开（超过 Keep-Alive 规定的时间，意外断电等情况除外），当客户端发送另外一个请求时，就使用这条已经建立的连接。

**无状态**

**无状态**是指协议对于事务处理没有记忆能力，服务器不知道客户端是什么状态。即我们给服务器发送 HTTP 请求之后，服务器根据请求，会给我们发送数据过来，但是，发送完，不会记录任何信息。  
  
HTTP 是一个无状态协议，这意味着每个请求都是独立的，Keep-Alive 没能改变这个结果。  
  
缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。  
  
HTTP 协议这种特性有优点也有缺点，优点在于解放了服务器，每一次请求“点到为止”不会造成不必要连接占用，缺点在于每次请求会传输大量重复的内容信息。  
  
客户端与服务器进行动态交互的 Web 应用程序出现之后，HTTP 无状态的特性严重阻碍了这些应用程序的实现，毕竟交互是需要承前启后的，简单的购物车程序也要知道用户到底在之前选择了什么商品。于是，两种用于保持 HTTP 连接状态的技术就应运而生了，一个是 Cookie，而另一个则是 Session。

**Cookie可以保持登录信息到用户下次与服务器的会话，换句话说，下次访问同一网站时，用户会发现不必输入用户名和密码就已经登录了**（当然，不排除用户手工删除Cookie）。而还有一些Cookie在用户退出会话的时候就被删除了，这样可以有效保护个人隐私。  
  
Cookies 最典型的应用是判定注册用户是否已经登录网站，用户可能会得到提示，是否在下一次进入此网站时保留用户信息以便简化登录手续，这些都是 Cookies 的功用。另一个重要应用场合是“购物车”之类处理。用户可能会在一段时间内在同一家网站的不同页面中选择不同的商品，这些信息都会写入 Cookies，以便在最后付款时提取信息。

**与 Cookie 相对的一个解决方案是 Session，它是通过服务器来保持状态的。**  
  
当客户端访问服务器时，服务器根据需求设置 Session，将会话信息保存在服务器上，同时将标示 Session 的 SessionId 传递给客户端浏览器，浏览器将这个 SessionId 保存在内存中，我们称之为无过期时间的 Cookie。浏览器关闭后，这个 Cookie 就会被清掉，它不会存在于用户的 Cookie 临时文件。  
  
以后浏览器每次请求都会额外加上这个参数值，服务器会根据这个 SessionId，就能取得客户端的数据信息。  
  
如果客户端浏览器意外关闭，服务器保存的 Session 数据不是立即释放，此时数据还会存在，只要我们知道那个 SessionId，就可以继续通过请求获得此 Session 的信息，因为此时后台的 Session 还存在，当然我们可以设置一个 Session 超时时间，一旦超过规定时间没有客户端请求时，服务器就会清除对应 SessionId 的 Session 信息。