完全忘记不久之前所做过的事一样。因为 HTTP 服务器并不保存关于客户的任何信息，所以我们说 HTTP 是一个**无状态协议**（stateless protocol）。我们同时也注意到 Web 使用了客户 - 服务器应用程序体系结构（如 2.1 节所述）。Web 服务器总是打开的，具有一个固定的 IP 地址，且它服务于可能来自数以百万计的不同浏览器的请求。

2.2.2 非持续连接和持续连接

在许多因特网应用程序中，客户和服务器在一个相当长的时间范围内通信，其中客户发出一系列请求并且服务器对每个请求进行响应。依据应用程序以及该应用程序的使用方式，这一系列请求可以以规则的间隔周期性地或者间断性地一个接一个发出。当这种客户 - 服务器的交互是经 TCP 进行的，应用程序的研制者就需要做一个重要决定，即每个请求/响应对是经一个单独的 TCP 连接发送，还是所有的请求及其响应经相同的 TCP 连接发送呢？采用前一种方法，该应用程序被称为使用**非持续连接**（non-persistent connection）；采用后一种方法，该应用程序被称为使用**持续连接**（persistent connection）。为了深入地理解该设计问题，我们研究在特定的应用程序即 HTTP 的情况下持续连接的优点和缺点，HTTP 既能够使用非持续连接，也能够使用持续连接。尽管 HTTP 在其默认方式下使用持续连接，HTTP 客户和服务器也能配置成使用非持续连接。

1. 采用非持续连接的 HTTP

我们看看在非持续连接情况下，从服务器向客户传送一个 Web 页面的步骤。假设该页面含有一个 HTML 基本文件和 10 个 JPEG 图形，并且这 11 个对象位于同一台服务器上。该 HTML 文件的 URL 为：http://www.someSchool.edu/someDepartment/home.index。

我们看看发生了什么情况：

- HTTP 客户进程在端口号 80 发起一个到服务器 www.someSchool.edu 的 TCP 连接，该端口号是 HTTP 的默认端口。在客户和服务器上分别有一个套接字与该连接相关联。
- HTTP 客户经它的套接字向该服务器发送一个 HTTP 请求报文。请求报文中包含了路径名/someDepartment/home.index（后面我们会详细讨论 HTTP 报文）。
- HTTP 服务器进程经它的套接字接收该请求报文，从其存储器（RAM 或磁盘）中检索出对象 www.someSchool.edu/someDepartment/home.index，在一个 HTTP 响应报文中封装对象，并通过其套接字向客户发送响应报文。
- HTTP 服务器进程通知 TCP 断开该 TCP 连接。（但是直到 TCP 确认客户已经完整地收到响应报文为止，它才会实际中断连接。）
- HTTP 客户接收响应报文，TCP 连接关闭。该报文指出封装的对象是一个 HTML 文件，客户从响应报文中提取出该文件，检查该 HTML 文件，得到对 10 个 JPEG 图形的引用。
- 对每个引用的 JPEG 图形对象重复前 4 个步骤。

当浏览器收到 Web 页面后，显示给用户。两个不同的浏览器也许会以不同的方式解释（即向用户显示）该页面。HTTP 与客户如何解释一个 Web 页面毫无关系。HTTP 规范（[RFC 1945] 和 [RFC 2616]）仅定义了在 HTTP 客户程序与 HTTP 服务器程序之间的通信协议。