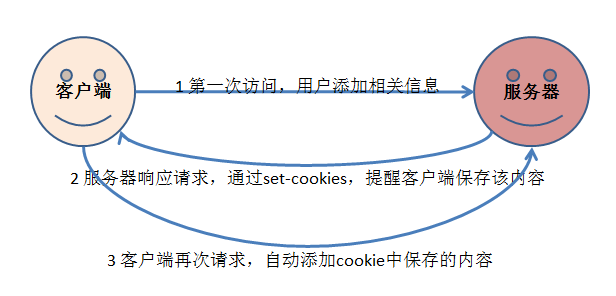
**无状态**

由于http是一种无状态的协议，因此无论是客户端还是服务器都不记录http的相关信息。

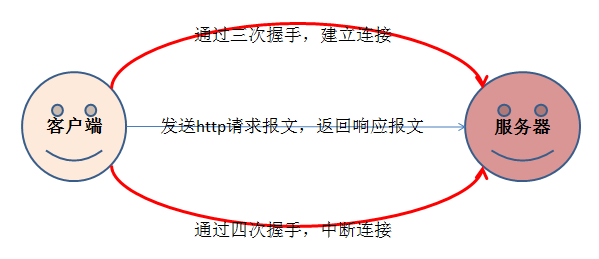
这样设计一方面减轻了服务器端的负载，另一方面减小了http请求的开销。

但是针对某些特殊的场景，需要时刻记录用户的相关信息，这该如何处理呢?

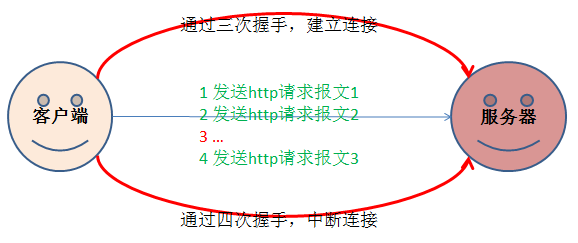
Cookie恰好可以解决这个问题,Cookie的运行机制如下：  
Cookie是一种由服务器端确定，并保存在客户端浏览器中的内容。这样，就不需要每次都添加用户的相关信息，请求会自动添加cookie中对应的内容。

**持久化**

正常在发送http时，都需要建立TCP的连接，再发送报文。

如果每次想要发送http报文都需要经过这个过程，那么时间大部分都会消耗在建立和断开连接的过程中。

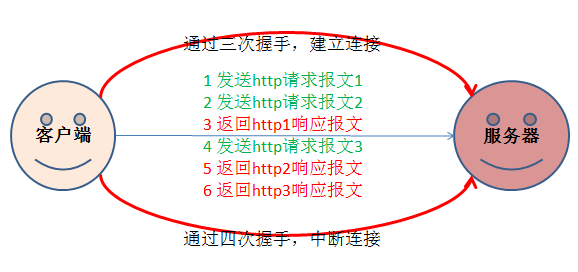
因此http中使用了connection属性，用于指定连接的方式。

**当设置成keep-alive，就会建立一条持久化的连接。**

不需要每次都建立连接，再中断。

**管道化**

如果一个http请求，请求了大量的图片等大文件，那么其他的http请求怎么办呢？

**不用怕，http可以一次发送多个http请求，然后等待响应连接。不需要排队等候，这样就加快了http的响应时间。**

内容编码

由于某些报文的内容过大，因此在传输时，为了减少传输的时间，会**采取一些压缩的措施。**

例如上面的报文信息中，Accept-Encoding就定义了内容编码的格式：gzip

有下面几种方式：

**gzip：GNU压缩格式**

**compress：UNIX系统的标准压缩格式**

**deflate：是一种同时使用了LZ77和哈弗曼编码的无损压缩格式**

**identity：不进行压缩**

多部分对象集合

有的时候传输的内容，不仅仅是一些字符串，还有可能是一些图片，字符，音乐二进制等混杂的内容。

这就需要使用多部分对象集合**，multipart，**例如在使用java编写web上传文件的代码时，需要在form中指定form的编码格式。

设置form的enctype属性的值为**multipart/form-data。**

这是因为默认的情况下form使用的编码格式是：applicatin/x-www-form-urlencoded，这种编码格式会把所有的内容进行编码，不适合上传文件这种情况。

这两种编码格式的区别主要是：

multipart/form-data 会以控件为基准，编码form中的内容。

application/x-www-form-urlencoded 会把form中的内容编码成键值对的形式。

范围请求

有些场景下，http报文请求一些很大的图片，但是加载过程很慢。

比如我们登录一些大图片的网址，会发现有时候图片是一块一块加载的。

这就是因为**设置了http请求的长度，这样就可以分块的加载资源文件。**

在请求报文中使用Range属性，在响应报文中使用Content-Type属性都可以指定一定字节范围的http请求。