HTTP 将来的扩展可能会推动对更多传输编码的需求。如果真的如此，那分块编码仍应始终作用在其他传输编码之上，这样就保证数据可以像隧道那样“穿透”那些只理解分块编码但不理解其他传输编码的 HTTP/1.1 应用程序。

15.6.3 分块编码

分块编码把报文分割为若干个大小已知的块。块之间是紧挨着发送的，这样就不需要在发送之前知道整个报文的大小了。

要注意的是，分块编码是一种传输编码，因此是报文的属性，而不是主体的属性。本章前面部分讨论过的多部分编码，就是主体的属性，它和分块编码是完全独立的。

1. 分块与持久连接

若客户端和服务器之间不是持久连接，客户端就不需要知道它正在读取的主体的长度，而只需要读到服务器关闭主体连接为止。 356

当使用持久连接时，在服务器写主体之前，必须知道它的大小并在 Content-Length 首部中发送。如果服务器动态创建内容，就可能在发送之前无法知道主体的长度。

分块编码为这种困难提供了解决方案，只要允许服务器把主体逐块发送，说明每块的大小就可以了。因为主体是动态创建的，服务器可以缓冲它的一部分，发送其大小和相应的块，然后在主体发送完之前重复这个过程。服务器可以用大小为 0 的块作为主体结束的信号，这样就可以继续保持连接，为下一个响应做准备。

分块编码是相当简单的。图 15-6 展示了一个分块编码报文的基本结构。它由起始的 HTTP 响应首部块开始，随后就是一系列分块。每个分块包含一个长度值和该分块的数据。长度值是十六进制形式并将 CRLF 与数据分隔开。分块中数据的大小以字节计算，不包括长度值与数据之间的 CRLF 序列以及分块结尾的 CRLF 序列。最后一个块有点特别，它的长度值为 0，表示“主体结束”。

客户端也可以发送分块的数据给服务器。因为客户端事先不知道服务器是否接受分块编码（这是因为服务器不会在给客户端的响应中发送 TE 首部），所以客户端必须做好服务器用 411 Length Required（需要 Content-Length 首部）响应来拒绝分块请求的准备。