15.2.2 错误的Content-Length

错误的 Content-Length 比缺少 Content-Length 还要糟糕。因为某些早期的客户端和服务器在 Content-Length 计算上存在一些众所周知的错误，有些客户端、服务器以及代理中就包含了特别的算法，用来检测和纠正与有缺陷服务器的交互过程。HTTP/1.1 规定用户 Agent 代理应该在接收且检测到无效长度时通知用户。

15.2.3 Content-Length与持久连接

Content-Length 首部对于持久连接是必不可少的。如果响应通过持久连接传送，就可能有另一条 HTTP 响应紧随其后。客户端通过 Content-Length 首部就可以知道报文在何处结束，下一条报文从何处开始。因为连接是持久的，客户端无法依赖连接关闭来判别报文的结束。如果没有 Content-Length 首部，HTTP 应用程序就不知道某个实体主体在哪里结束，下一条报文从哪里开始。

我们将在 15.6 节看到，有一种情况下，使用持久连接时可以没有 Content-Length 首部，即采用*分块编码*（chunked encoding）时。在分块编码的情况下，数据是分为一系列的块来发送的，每块都有大小说明。哪怕服务器在生成首部的时候不知道整个实体的大小（通常是因为实体是动态生成的），仍然可以使用分块编码传输若干已知大小的块。

15.2.4 内容编码

HTTP 允许对实体主体的内容进行编码，比如可以使之更安全或进行压缩以节省空间（本章稍后将详细解释压缩的问题）。如果主体进行了内容编码，Content-Length 首部说明的就是*编码后*（encoded）的主体的字节长度，而*不是*未编码的原始主体的长度。

某些 HTTP 应用程序在这方面搞错了，发送的是数据编码之前的大小，这会导致严
345 重的错误，尤其是用在持久连接上。不幸的是，HTTP/1.1 规范中没有首部可以用来说明原始的、未编码的主体的长度，这就让客户端难以验证*解码过程*的完整性。[2]

15.2.5 确定实体主体长度的规则

下面列出的规则说明了在若干不同的情况下如何正确计算主体的长度和结束位置。这些规则应当按顺序应用，谁先匹配就用谁。

注 2：Content-MD5 首部也不行，它用来说明文档的 128 位 MD5 值，但这也是针对编码后的文档的。本章后面会说明 Content-MD5 首部。