户端使用 gzip 解码器对实体进行解压缩。

下面给出的响应片段是另一个编码响应的例子（一个压缩的图像）：

```
HTTP/1.1 200 OK
Date: Fri, 05 Nov 1999 22:35:15 GMT
Server: Apache/1.2.4
Content-Length: 6096
Content-Type: image/gif
Content-Encoding: gzip
[...]
```

注意，Content-Type 首部可以且还应当出现在报文中。它说明了实体的原始格式，一旦实体被解码，要显示的时候，可能还是需要该信息才行的。记住，Content-Length 首部现在代表的是*编码之后*的主体长度。

15.5.2 内容编码类型

HTTP 定义了一些标准的内容编码类型，并允许用扩展编码的形式增添更多的编码。由互联网号码分配机构（IANA）对各种编码进行标准化，它给每个内容编码算法分配了唯一的代号。Content-Encoding 首部就用这些标准化的代号来说明编码时使用的算法。

表 15-2 列出了一些常用的内容编码代号。

352

表15-2 内容编码代号

Content-Encoding值	描　述
gzip	表明实体采用 GNU zip 编码[a]
compress	表明实体采用 Unix 的文件压缩程序
deflate	表明实体是用 zlib 的格式压缩的[b]
identity	表明没有对实体进行编码。当没有 Content-Encoding 首部时，就默认为这种情况

a：RFC 1952 中说明了 gzip 编码。

b：RFC 1950 和 1951 中讲解了 zlib 格式和 deflate 压缩算法。

gzip、compress 以及 deflate 编码都是无损压缩算法，用于减少传输报文的大小，不会导致信息损失。这些算法中，gzip 通常是效率最高的，使用最为广泛。

15.5.3 Accept-Encoding首部

毫无疑问，我们不希望服务器用客户端无法解码的方式来对内容进行编码。为了避免服务器使用客户端不支持的编码方式，客户端就把自己支持的内容编码方式