**HTTP 协议中的Content-Encoding**

转载[想跑步丶小胖子](https://me.csdn.net/m0_37542889" \t "_blank) 最后发布于2018-09-28 20:37:12 阅读数 1959  收藏

展开

**HTTP 协议中的Content-Encoding**

主要内容

1. Content-Encoding 是什么
2. 内容编码格式gzip 和 deflate

**Content-Encoding 是什么**

Accept-Encoding 和Content-Encoding是HTTP中用来对采用哪种编码格式传输正文进行协定的一对头部字段。

工作原理如下:

1. 首先浏览器（也就是客户端）发送请求时，通过Accept-Encoding带上自己支持的内容编码格式列表；
2. 服务端在接收到请求后，从中挑选出一种用来对响应信息进行编码，并通过Content-Encoding来说明服务端选定的编码信息
3. 浏览器在拿到响应正文后，依据Content-Encoding进行解压。

服务端也可以返回未压缩的正文，但这种情况不允许返回Content-Encoding

* + 1

内容编码的目的是优化传输内容的大小，通俗的讲就是尽心压缩。一般经过gzip压缩过的文本响应，只有原始大小的1/4（这个数据我现在还不确定）。对于文本类响应是否开启了内容压缩，是我们做性能优化时首先要检查的重要项目；而对于JPG/PNG这类本身已经高度压缩过的二进制文件，不推介开启内容压缩，效果几乎微乎其微，还浪费cpu

内容的编码针对的只是传输正文。在HTTP/1中，头部始终是以ASCII文本传输，没有经过任何压缩。不过这个问题在HTTP/2中已经解决，详见[HTTP/2 头部压缩技术介绍](https://blog.csdn.net/u014569188/article/details/78912446)

内容编码使用特别广泛，理解起来也特别简单，但是要注意的是不要把它与HTPP中的另外一个概念：传输编码[Transfer-Encoding](https://blog.csdn.net/u014569188/article/details/78912446)搞混即可。

内容编码格式gzip 和deflate

开始之前，先来介绍三种数据压缩格式：

* DEFLATE，是一种使用 Lempel-Ziv 压缩算法（LZ77）和哈夫曼编码的数据压缩格式。定义于 [RFC 1951 : DEFLATE Compressed Data Format Specification](http://tools.ietf.org/html/rfc1951)；
* ZLIB，是一种使用 DEFLATE 的数据压缩格式。定义于 [RFC 1950 : ZLIB Compressed Data Format Specification](http://tools.ietf.org/html/rfc1950)；
* GZIP，是一种使用 DEFLATE 的文件格式。定义于 [RFC 1952 : GZIP file format specification](http://tools.ietf.org/html/rfc1952)

在 HTTP/1.1 的初始规范 RFC 2616 的「[3.5 Content Codings](http://tools.ietf.org/html/rfc2616" \l "section-3.5)」这一节中，这样定义了 Content-Encoding 中的 gzip 和 deflate：

* gzip，一种由文件压缩程序「Gzip，GUN zip」产生的编码格式，描述于 RFC 1952。这种编码格式是一种具有 32 位 CRC 的 Lempel-Ziv 编码（LZ77）；
* deflate，由定义于 RFC 1950 的「ZLIB」编码格式与 RFC 1951 中描述的「DEFLATE」压缩机制组合而成的产物；

RFC 2616 对 Content-Encoding 中的 gzip 的定义很清晰，它就是指在 RFC 1952 中定义的 GZIP 编码格式；但对 deflate 的定义含糊不清，实际上它指的是 RFC 1950 中定义的 ZLIB 编码格式，但 deflate 这个名字特别容易产生误会。

在 Zlib 库的官方网站，有这么一条 FAQ：[What’s the difference between the “gzip” and “deflate” HTTP 1.1 encodings?](http://www.gzip.org/zlib/zlib_faq.html" \l "faq38) 就是在讨论 HTTP/1.1 对 deflate 的错误命名：

Q：在 HTTP/1.1 的 Content-Encoding 中，gzip 和 deflate 的区别是什么？

A：gzip 是指 GZIP 格式，deflate 是指 ZLIB 格式。HTTP/1.1 的作者或许应该将后者称之为 zlib，从而避免与原始的 DEFLATE 数据格式产生混淆。虽然 HTTP/1.1 RFC 2016 正确指出，Content-Encoding 中的 deflate 就是 RFC 1950 描述的 ZLIB，但仍然有报告显示部分服务器及浏览器错误地生成或期望收到原始的 DEFLATE 格式，特别是微软。所以虽然使用 ZLIB 更为高效（实际上这正是 ZLIB 的设计目标），但使用 GZIP 格式可能更为可靠，这一切都是因为 HTTP/1.1 的作者不幸地选择了错误的命名。

结论：在 HTTP/1.1 的 Content-Encoding 中，请使用 gzip。

在 HTTP/1.1 的修订版 RFC 7230 的 [4.2 Compression Codings](https://tools.ietf.org/html/rfc7230#section-4.2) 这一节中，彻底明确了 deflate 的含义，对 gzip 也做了补充：

* deflate，包含「使用 Lempel-Ziv 压缩算法（LZ77）和哈夫曼编码的 DEFLATE 压缩数据流（RFC 1951）」的 ZLIB 数据格式（RFC 1950）。

注：一些不符合规范的实现会发送没有经过 ZLIB 包装的 DEFLATE 压缩数据；

* gzip，具有 32 位循环冗余检查（CRC）的 LZ77 编码，通常由 Gzip 文件压缩程序（RFC 1952）产生。接受方应该将 x-gzip 视为 gzip；

总结一下，HTTP标准中定义的Content-Encoding：deflate，实际上指的是ZLIB编码(RFC 1950).但由于RFC2616中含糊不清的定义，导致IE错误地实现为只接受原始DEFLATE(rfc 1951)。为空兼容IE，我们只能使用Content-Encoding：gzip进行内容编码，他指的是GZIP编码（RFC 1952）

其实上，ZLIB 和 DEFLATE 的差别很小：ZLIB 数据去掉 2 字节的 ZLIB 头，再忽略最后 4 字节的校验和，就变成了 DEFLATE 数据。

在 Fiddler 增加以下处理，就可以让 IE 支持标准的 Content-Encoding: deflate（ZLIB 编码）:

1. if ((compressedData.Length > 2) &&
2. ((compressedData[0] & 0xF) == 0x8) && *// Low 4-bits must be 8*
3. ((compressedData[0] & 0x80) == 0) && *// High-bit must be clear*
4. ((((compressedData[0] << 8) + compressedData[1]) % 31) == 0))
5. *// Validate checksum*
6. {
7. Debug.Write("Fiddler: Ignoring RFC1950 Header bytes for DEFLATE");
8. iStartOffset = 2;
9. }

由于其它浏览器也能解析原始 DEFLATE，所以有些 WEB 应用干脆为了迁就 IE 直接输出原始 DEFLATE。