HTTP状态码

维基百科，自由的百科全书

HTTP状态码（英语：HTTP Status Code）是用以表示網頁伺服器超文本传输协议响应状态的3位数字代码。它由 RFC 2616 规范定义的，并得到 RFC 2518、RFC 2817、RFC 2295、RFC 2774 與 RFC 4918 等规范扩展。所有状态码的第一个数字代表了响应的五种状态之一。所示的消息短语是典型的，但是可以提供任何可读取的替代方案。 除非另有说明，状态码是HTTP / 1.1标准（RFC 7231）的一部分。[1]

HTTP状态码的官方注册表由互联网号码分配局（Internet Assigned Numbers Authority）维护。[2]

微软互联网信息服务（Microsoft Internet Information Services）有时会使用额外的十进制子代码来获取更多具体信息，[3]但是这些子代码仅出现在响应有效内容和文档中，而不是代替实际的HTTP状态代码。

目录

1xx消息

这一类型的状态码，代表请求已被接受，需要继续处理。这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束。由于HTTP/1.0协议中没有定义任何1xx状态码，所以除非在某些试验条件下，服务器禁止向此类客户端发送1xx响应。[4] 这些状态码代表的响应都是信息性的，标示客户应该采取的其他行动。

100 Continue

服务器已经接收到请求头，并且客户端应继续发送请求主体（在需要发送身体的请求的情况下：例如，POST请求），或者如果请求已经完成，忽略这个响应。服务器必须在请求完成后向客户端发送一个最终响应。要使服务器检查请求的头部，客户端必须在其初始请求中发送`Expect: 100-continue`作为头部，并在发送正文之前接收`100 Continue`状态代码。响应代码417期望失败表示请求不应继续。[2]

101 Switching Protocols

服务器已经理解了客户端的请求，并将通过Upgrade消息头通知客户端采用不同的协议来完成这个请求。在发送完这个响应最后的空行后，服务器将会切换到在Upgrade消息头中定义的那些协议。[5] 只有在切换新的协议更有好处的时候才应该采取类似措施。例如，切换到新的HTTP版本（如HTTP/2）比旧版本更有优势，或者切换到一个实时且同步的协议（如WebSocket）以传送利用此类特性的资源。

102 Processing（WebDAV；RFC 2518）

WebDAV请求可能包含许多涉及文件操作的子请求，需要很长时间才能完成请求。该代码表示服务器已经收到并正在处理请求，但无响应可用。[6]这样可以防止客户端超时，并假设请求丢失。

2xx成功

这一类型的状态码，代表请求已成功被服务器接收、理解、并接受。[2]

200 OK
请求已成功，请求所希望的响应头或数据体将随此响应返回。实际的响应将取决于所使用的请求方法。在GET请求中，响应将包含与请求的资源相对应的实体。在POST请求中，响应将包含描述或操作结果的实体。[7]

201 Created
请求已经被实现，而且有一个新的资源已经依据请求的需要而建立，且其URI已经随Location头信息返回。假如需要的资源无法及时建立的话，应当返回'202 Accepted'。[8]

202 Accepted
服务器已接受请求，但尚未处理。最终该请求可能会也可能不会被执行，并且可能在处理發生时被禁止。[9]

203 Non-Authoritative Information（自HTTP / 1.1起）
服务器是一个转换代理服务器（transforming proxy，例如网络加速器），以`200 OK`状态码为起源，但回应了原始响应的修改版本。[10][11]

204 No Content
服务器成功处理了请求，没有返回任何内容。[12]

205 Reset Content
服务器成功处理了请求，但没有返回任何内容。与204响应不同，此响应要求请求者重置文档视图。[13]

206 Partial Content（RFC 7233）
服务器已经成功处理了部分GET请求。类似于FlashGet或者迅雷这类的HTTP 下载工具都是使用此类响应实现断点续传或者将一个大文档分解为多个下载段同时下载。[14]

207 Multi-Status（WebDAV；RFC 4918）
代表之后的消息体将是一个XML消息，并且可能依照之前子请求数量的不同，包含一系列独立的响应代码。[15]

208 Already Reported （WebDAV；RFC 5842）
DAV绑定的成员已经在（多状态）响应之前的部分被列举，且未被再次包含。

226 IM Used （RFC 3229）
服务器已经满足了对资源的请求，对实体请求的一个或多个实体操作的结果表示。[16]

3xx重定向

这类状态码代表需要客户端采取进一步的操作才能完成请求。通常，这些状态码用来重定向，后续的请求地址（重定向目标）在本次响应的Location域中指明。[2]

当且仅当后续的请求所使用的方法是GET或者HEAD时，用户浏览器才可以在没有用户介入的情况下自动提交所需要的后续请求。客户端应当自动监测无限循环重定向（例如：A→B→C→……→A或A→A），因为这会导致服务器和客户端大量不必要的资源消耗。按照HTTP/1.0版规范的建议，浏览器不应自动访问超过5次的重定向。[17]

300 Multiple Choices

被请求的资源有一系列可供选择的回馈信息，每个都有自己特定的地址和浏览器驱动的商议信息。用户或浏览器能够自行选择一个首选的地址进行重定向。[18]
除非这是一个HEAD请求，否则该响应应当包括一个资源特性及地址的列表的实体，以便用户或浏览器从中选择最合适的重定向地址。这个实体的格式由Content-Type定义的格式所决定。浏览器可能根据响应的格式以及浏览器自身能力，自动作出最合适的选择。当然，RFC 2616规范并没有规定这样的自动选择该如何进行。
如果服务器本身已经有了首选的回馈选择，那么在Location中应当指明这个回馈的URI；浏览器可能会将这个Location值作为自动重定向的地址。此外，除非额外指定，否则这个响应也是可缓存的。

301 Moved Permanently

被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个URI之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。[19]除非额外指定，否则这个响应也是可缓存的。
新的永久性的URI应当在响应的Location域中返回。除非这是一个HEAD请求，否则响应的实体中应当包含指向新的URI的超链接及简短说明。
如果这不是一个GET或者HEAD请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。
注意：对于某些使用HTTP/1.0协议的浏览器，当它们发送的POST请求得到了一个301响应的话，接下来的重定向请求将会变成GET方式。

302 Found

要求客户端执行临时重定向（原始描述短语为“Moved Temporarily”）。[20]由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况下，这个响应才是可缓存的。
新的临时性的URI应当在响应的Location域中返回。除非这是一个HEAD请求，否则响应的实体中应当包含指向新的URI的超链接及简短说明。
如果这不是一个GET或者HEAD请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。
注意：虽然RFC 1945和RFC 2068规范不允许客户端在重定向时改变请求的方法，但是很多现存的浏览器将302响应视作为303响应，并且使用GET方式访问在Location中规定的URI，而无视原先请求的方法。[21]因此状态码303和307被添加了进来，用以明确服务器期待客户端进行何种反应。[22]

303 See Other

对应当前请求的响应可以在另一个URI上被找到，当响应于POST（或PUT / DELETE）接收到响应时，客户端应该假定服务器已经收到数据，并且应该使用单独的GET消息发出重定向。[23]这个方法的存在主要是为了允许由脚本激活的POST请求输出重定向到一个新的资源。这个新的URI不是原始资源的替代引用。同时，303响应禁止被缓存。当然，第二个请求（重定向）可能被缓存。
新的URI应当在响应的Location域中返回。除非这是一个HEAD请求，否则响应的实体中应当包含指向新的URI的超链接及简短说明。
注意：许多HTTP/1.1版以前的浏览器不能正确理解303状态。如果需要考虑与这些浏览器之间的互动，302状态码应该可以胜任，因为大多数的浏览器处理302响应时的方式恰恰就是上述规范要求客户端处理303响应时应当做的。

304 Not Modified

表示资源在由请求头中的If-Modified-Since或If-None-Match参数指定的这一版本之后，未曾被修改。在这种情况下，由于客户端仍然具有以前下载的副本，因此不需要重新传输资源。[24]

305 Use Proxy

被请求的资源必须通过指定的代理才能被访问。Location域中将给出指定的代理所在的URI信息，接收者需要重复发送一个单独的请求，通过这个代理才能访问相应资源。只有原始服务器才能建立305响应。许多HTTP客户端（像是Mozilla[25]和Internet Explorer）都没有正确处理这种状态代码的响应，主要是出于安全考虑。[26]
注意：RFC 2068中没有明确305响应是为了重定向一个单独的请求，而且只能被原始服务器建立。忽视这些限制可能导致严重的安全后果。

306 Switch Proxy

在最新版的规范中，306状态码已经不再被使用。最初是指“后续请求应使用指定的代理”。[27]

307 Temporary Redirect

在这种情况下，请求应该与另一个URI重复，但后续的请求应仍使用原始的URI。 与302相反，当重新发出原始请求时，不允许更改请求方法。 例如，应该使用另一个POST请求来重复POST请求。[28]

308 Permanent Redirect (RFC 7538)

请求和所有将来的请求应该使用另一个URI重复。 307和308重复302和301的行为，但不允许HTTP方法更改。 例如，将表单提交给永久重定向的资源可能会顺利进行。[29]

4xx客户端错误

这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理。除非响应的是一个HEAD请求，否则服务器就应该返回一个解释当前错误状况的实体，以及这是临时的还是永久性的状况。这些状态码适用于任何请求方法。浏览器应当向用户显示任何包含在此类错误响应中的实体内容。[30]

如果错误发生时客户端正在传送数据，那么使用TCP的服务器实现应当仔细确保在关闭客户端与服务器之间的连接之前，客户端已经收到了包含错误信息的数据包。如果客户端在收到错误信息后继续向服务器发送数据，服务器的TCP栈将向客户端发送一个重置数据包，以清除该客户端所有还未识别的输入缓冲，以免这些数据被服务器上的应用程序读取并干扰后者。

400 Bad Request

由于明显的客户端错误（例如，格式错误的请求语法，太大的大小，无效的请求消息或欺骗性路由请求），服务器不能或不会处理该请求。[31]

401 Unauthorized（RFC 7235）

参见：HTTP基本认证、HTTP摘要认证

类似于403 Forbidden，401语义即“未认证”，即用户没有必要的凭据。[32]该状态码表示当前请求需要用户验证。该响应必须包含一个适用于被请求资源的WWW-Authenticate信息头用以询问用户信息。客户端可以重复提交一个包含恰当的Authorization头信息的请求。[33]如果当前请求已经包含了Authorization证书，那么401响应代表着服务器验证已经拒绝了那些证书。如果401响应包含了与前一个响应相同的身份验证询问，且浏览器已经至少尝试了一次验证，那么浏览器应当向用户展示响应中包含的实体信息，因为这个实体信息中可能包含了相关诊断信息。
注意：当网站（通常是网站域名）禁止IP地址时，有些网站状态码显示的401，表示该特定地址被拒绝访问网站。

402 Payment Required

该状态码是为了将来可能的需求而预留的。该状态码最初的意图可能被用作某种形式的数字现金或在线支付方案的一部分，但几乎没有哪家服务商使用，而且这个状态码通常不被使用。如果特定开发人员已超过请求的每日限制，Google Developers API会使用此状态码。[34]

403 Forbidden

服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个HEAD请求，而且服务器希望能够讲清楚为何请求不能被执行，那么就应该在实体内描述拒绝的原因。当然服务器也可以返回一个404响应，假如它不希望让客户端获得任何信息。

404 Not Found

请求失败，请求所希望得到的资源未被在服务器上发现，但允许用户的后续请求。[35]没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使用410状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404这个

状态码被广泛应用于当服务器不想揭示到底为何请求被拒绝或者没有其他适合的响应可用的情况下。

405 Method Not Allowed

请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个Allow头信息用以表示出当前资源能够接受的请求方法的列表。例如，需要通过POST呈现数据的表单上的GET请求，或只读资源上的PUT请求。

鉴于PUT，DELETE方法会对服务器上的资源进行写操作，因而绝大部分的网页服务器都不支持或者在默认配置下不允许上述请求方法，对于此类请求均会返回405错误。

406 Not Acceptable

参见：内容协商

请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体，该请求不可接受。[36]除非这是一个HEAD请求，否则该响应就应当返回一个包含可以让用户或者浏览器从中选择最合适的实体特性以及地址列表的实体。实体的格式由Content-Type头中定义的媒体类型决定。浏览器可以根据格式及自身能力自行作出最佳选择。但是，规范中并没有定义任何作出此类自动选择的标准。

407 Proxy Authentication Required（RFC 2617）

与401响应类似，只不过客户端必须在代理服务器上进行身份验证。[37]代理服务器必须返回一个Proxy-Authenticate用以进行身份询问。客户端可以返回一个Proxy-Authorization信息头用以验证。

408 Request Timeout

请求超时。根据HTTP规范，客户端没有在服务器预备等待的时间内完成一个请求的发送，客户端可以随时再次提交这一请求而无需进行任何更改。[38]

409 Conflict

表示因为请求存在冲突无法处理该请求，例如多个同步更新之间的编辑冲突。

410 Gone

表示所请求的资源不再可用，将不再可用。当资源被有意地删除并且资源应被清除时，应该使用这个。在收到410状态码后，用户应停止再次请求资源。[39]但大多数服务端不会使用此状态码，而是直接使用404状态码。

411 Length Required

服务器拒绝在没有定义Content-Length头的情况下接受请求。在添加了表明请求消息体长度的有效Content-Length头之后，客户端可以再次提交该请求。[40]

412 Precondition Failed（RFC 7232）

服务器在验证在请求的头字段中给出先决条件时，没能满足其中的一个或多个。[41]这个状态码允许客户端在获取资源时在请求的元信息（请求头字段数据）中设置先决条件，以此避免该请求方法被应用到其希望的内容以外的资源上。

413 Request Entity Too Large（RFC 7231）

前称“Request Entity Too Large”，表示服务器拒绝处理当前请求，因为该请求提交的实体数据大小超过了服务器愿意或者能够处理的范围。[42]此种情况下，服务器可以关闭连接以免客户端继续发送此请求。

如果这个状况是临时的，服务器应当返回一个Retry-After的响应头，以告知客户端可以在多少时间以后重新尝试。

414 Request-URI Too Long（RFC 7231）

前称“Request-URI Too Long”，[43]表示请求的URI长度超过了服务器能够解释的长度，因此服务器拒绝对该请求提供服务。通常将太多数据的结果编码为GET请求的查询字符串，在这种情况下，应将其转换为POST请求。[44]这比较少见，通常的情况包括：

- 本应使用POST方法的表单提交变成了GET方法，导致查询字符串过长。

- 重定向URI“黑洞”，例如每次重定向把旧的URI作为新的URI的一部分，导致在若干次重定向后URI超长。
- 客户端正在尝试利用某些服务器中存在的安全漏洞攻击服务器。这类服务器使用固定长度的缓冲读取或操作请求的URI，当GET后的参数超过某个数值后，可能会产生缓冲区溢出，导致任意代码被执行[45]。没有此类漏洞的服务器，应当返回414状态码。

415 Unsupported Media Type

对于当前请求的方法和所请求的资源，请求中提交的互联网媒体类型并不是服务器中所支持的格式，因此请求被拒绝。例如，客户端将图像上传格式为svg，但服务器要求图像使用上传格式为jpg。

416 Requested Range Not Satisfiable（RFC 7233）

前称“Requested Range Not Satisfiable”。[46]客户端已经要求文件的一部分（Byte serving），但服务器不能提供该部分。例如，如果客户端要求文件的一部分超出文件尾端。[47]

417 Expectation Failed

在请求头Expect中指定的预期内容无法被服务器满足，或者这个服务器是一个代理服显的证据证明在当前路由的下一个节点上，Expect的内容无法被满足。[48]

418 I'm a teapot（RFC 2324）

本操作码是在1998年作为IETF的传统愚人节笑话, 在RFC 2324超文本咖啡壶控制协议'中定义的，并不需要在真实的HTTP服务器中定义。當一個控制茶壺的HTCPCP收到BREW或POST指令要求其煮咖啡時應當回傳此錯誤。[49]这个HTTP状态码在某些网站（包括Google.com）與項目（如Node.js、ASP.NET和Go語言）中用作彩蛋。[50]

420 Enhance Your Caim

Twitter Search与Trends API在客户端被限速的情况下返回。

421 Misdirected Request （RFC 7540）

该请求针对的是无法产生响应的服务器（例如因为连接重用）。[51]

422 Unprocessable Entity（WebDAV；RFC 4918）

请求格式正确，但是由于含有语义错误，无法响应。[15]

423 Locked（WebDAV；RFC 4918）

当前资源被锁定。[15]

424 Failed Dependency（WebDAV；RFC 4918）

由于之前的某个请求发生的错误，导致当前请求失败，例如PROPPATCH。[15]

425 Unordered Collection

在WebDAV Advanced Collections Protocol中定义，但Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol中并不存在。

426 Upgrade Required（RFC 2817）

客户端应当切换到TLS/1.0，并在HTTP/1.1 Upgrade header中给出。[15]

428 Precondition Required (RFC 6585)

原服务器要求该请求满足一定条件。这是为了防止“‘未更新’问题，即客户端读取（GET）一个资源的状态，更改它，并将它写（PUT）回服务器，但这期间第三方已经在服务器上更改了该资源的状态，因此导致了冲突。”[52]

429 Too Many Requests （RFC 6585）

用户在给定的时间内发送了太多的请求。旨在用于网络限速。[52]

431 Request Header Fields Too Large （RFC 6585）

服务器不愿处理请求，因为一个或多个头字段过大。[52]

444 No Response
Nginx上HTTP服務器擴展。服務器不向客戶端返回任何信息，並關閉連接（有助於阻止惡意軟體）。

450 Blocked by Windows Parental Controls
這是一個由Windows家庭控制（Microsoft）HTTP阻止的450狀態代碼的示例，用於信息和測試。

451 Unavailable For Legal Reasons

该访问因法律的要求而被拒絕，由IETF在2015核准后新增加。[53][54][55]

494 Request Header Too Large
在错误代码431提出之前Nginx上使用的扩展HTTP代码。

5xx服务器错误

表示服务器无法完成明显有效的请求。[56]这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。除非这是一个HEAD请求，否则服务器应当包含一个解释当前错误状态以及这个状况是临时的还是永久的解释信息实体。浏览器应当向用户展示任何在当前响应中被包含的实体。这些状态码适用于任何响应方法。[57]

500 Internal Server Error
通用错误消息，服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。没有给出具体错误信息。[58]

501 Not Implemented
服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求。[59]（例如，网络服务API的新功能）

502 Bad Gateway
作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。[60]

503 Service Unavailable
由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是暂时的，并且将在一段时间以后恢复。[61]如果能够预计延迟时间，那么响应中可以包含一个Retry-After头用以标明这个延迟时间。如果没有给出这个Retry-After信息，那么客户端应当以处理500响应的方式处理它。

504 Gateway Timeout
作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。[62]
注意：某些代理服务器在DNS查询超时时会返回400或者500错误。

505 HTTP Version Not Supported
服务器不支持，或者拒绝支持在请求中使用的HTTP版本。[63]这暗示着服务器不能或不愿使用与客户端相同的版本。响应中应当包含一个描述了为何版本不被支持以及服务器支持哪些协议的实体。

506 Variant Also Negotiates（RFC 2295）
由《透明内容协商协议》（RFC 2295）扩展，代表服务器存在内部配置错误，[64]被请求的协商变元资源被配置为在透明内容协商中使用自己，因此在一个协商处理中不是一个合适的重点。

507 Insufficient Storage（WebDAV；RFC 4918）
服务器无法存储完成请求所必须的内容。这个状况被认为是临时的。[15]

508 Loop Detected （WebDAV；RFC 5842）
服务器在处理请求时陷入死循环。（可代替 208状态码）

510 Not Extended（RFC 2774）

获取资源所需要的策略并没有被满足。[65]

511 Network Authentication Required （RFC 6585）

客户端需要进行身份验证才能获得网络访问权限，旨在限制用户群访问特定网络。（例如连接WiFi热点时的强制网络门户）[52]

非官方状态码

420 Enhance Your Calm

據說早期 Twitter API 會在短期內送出太多需求的時候回傳這個 Status Code，不過在新版 API 改為使用 429 Too Many Requests。

498 Invalid Token
499 Token Required

這兩個是以前一個叫做 ArcGIS for Server 的系統會回應的 Status Code。一般來說驗證資訊錯誤還是會回傳 401 Unathorized。

520 Unknown Error

Cloudflare 會用的未知錯誤。

521 Web Server Is Down

指目標伺服器掛了

参见

- 超文本传输协议
- HTTP头字段列表
- FTP服务器返回码列表

参考文献

1. Hypertext Transfer Protocol ---HTTP/1.1 IETF. [2015-10-16].
2. Hypertext Transfer Protocol (HTTP) Status Code Registry. Iana.org. [January 8, 2015].
3. The HTTP status codes in IIS 7.0 Microsoft. July 14, 2009 [April 1, 2009].
4. 10 Status Code Definitions W3. [16 October 2015].
5. 101. httpstatus. [16 October 2015].
6. Goland, Yaronn; Whitehead, Jim; Faizi, Asad; Carter, Steve R.; Jensen, Del. HTTP Extensions for Distributed Authoring – WEBDAV (https://tools.ietf.org/html/rfc2518) IETF. February 1999 [October 24, 2009] RFC 2518.
7. 200 OK (https://tools.ietf.org/html/rfc2616#section-10.2.1) Hypertext Transfer Protocol -- HTTP/1.1 (https://tools.ietf.org/html/rfc2616) IETF. June 1999: sec. 10.2.1 [August 30, 2016] RFC 2616.
8. Stewart, Mark; djna. Create request with POST, which response codes 200 or 201 and content Stack Overflow. [16 October 2015].
9. 202. httpstatus. [16 October 2015].
10. RFC 7231, Section 6.3.4.
11. RFC 7230, Section 5.7.2.
12. Simmance, Chris. Server Response Codes And What They Mean koozai. [16 October 2015].
13. ikitommi; Deraen. metosin/ring-http-response GitHub. [16 October 2015].
14. diff --git a/linkchecker.module b/linkchecker.module. Drupal. [16 October 2015].
15. Dusseault, Lisa. HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV) (https://tools.ietf.org/html/rfc4918) IETF. June 2007 [October 24, 2009] RFC 4918.
16. Delta encoding in HTTP (https://tools.ietf.org/html/rfc3229) IETF. January 2002 [February 25, 2011] RFC 3229.

17. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. IETF. [13 February 2016].
18. 300. httpstatus. [16 October 2015].
19. 301. httpstatus. [16 October 2015].
20. Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk. Hypertext Transfer Protocol – HTTP/1.0 (https://tools.ietf.org/html/rfc1945). IETF. May 1996 [October 24, 2009]. RFC 1945.
21. Reference of method redirect_to in Ruby Web Framework "Ruby on Rails". It states: The redirection happens as a "302 Moved" header unless otherwise specified. [June 30, 2012].
22. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 6.4. IETF. [June 12, 2014].
23. 303. httpstatus. [16 October 2015].
24. Brown, Kevin; CRS... getting 304 response even with django-cors-headers. Stack Overflow. [16 October 2015].
25. Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect. March 3, 2003 [May 21, 2009].
26. Mozilla Bugzilla Bug 187996: Strange behavior on 305 redirect, comment 13. March 3, 2003 [May 21, 2009].
27. Cohen, Josh. HTTP/1.1 305 and 306 Response Codes. HTTP Working Group.
28. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 6.4.7 307 Temporary Redirect. IETF. 2014 [September 20, 2014].
29. The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect) Internet Engineering Task Force. April 2015 [2015-04-06].
30. E Explanation of Failure Codes. Oracle. [16 October 2015].
31. RFC7231 on code 400. Tools.ietf.org. [January 8, 2015].
32. RFC7235 on code 401. Tools.ietf.org. [February 8, 2015].
33. 401. httpstatus. [16 October 2015].
34. Google API Standard Error Responses. Google. 2015 [September 24, 2015].
35. Introduction. QAS. [16 October 2015].
36. Singh, Prabhat; user1740567. Spring 3.x JSON status 406 "characteristics not acceptable according to the request "accept" headers ()". Stack Overflow. [16 October 2015].
37. 407. httpstatus. [16 October 2015].
38. 408. httpstatus. [16 October 2015].
39. Does Google treat 404 and 410 status codes differently? (Youtube). Google. 2014 [February 4, 2015].
40. List of HTTP status codes. Google Books. [16 October 2015].
41. Kowser; Patel, Amit. REST response code for invalid data. Stack Overflow. [16 October 2015].
42. RFC2616 on status 413. Tools.ietf.org. [November 11, 2015].
43. RFC2616 on status 414. Tools.ietf.org. [November 11, 2015].
44. user27828. GET Request - Why is my URI so long? Stack Overflow. [16 October 2015].
45. HTTP Get存在缓冲溢出漏洞. [2008-05-05]. （原始内容存档于2009-03-18）.
46. RFC2616 on status 416. Tools.ietf.org. [November 11, 2015].
47. Sigler, Chris. 416 Requested Range Not Satisfiable. GetStatusCode. [16 October 2015]. （原始内容存档于2015年10月22日）.
48. TheDeadLike. HTTP/1.1 Status Codes 400 and 417, cannot choose which. serverFault. [16 October 2015].
49. Larry Masinter. Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0) (https://tools.ietf.org/html/rfc2324). 1 April 1998. RFC 2324.
50. Barry Schwartz. New Google Easter Egg For SEO Geeks: Server Status 418, I'm A Teapot. Search Engine Land. 26 August 2014.
51. Hypertext Transfer Protocol version 2. March 2015 [April 25, 2015].
52. Nottingham, M.; Fielding, R. RFC 6585 – Additional HTTP Status Codes. Request for Comments. Internet Engineering Task Force. April 2012 [May 1, 2012].
53. Bray, T. An HTTP Status Code to Report Legal Obstacles. ietf.org. February 2016 [7 March 2015].
54. Brian Chen. IESG 批准全新 HTTP 狀態碼「451」. 科技新報. 2015-12-21 [2015-12-22]. （原始内容存档于2015-12-25） （中文（繁體））.
55. 林妍溱. 未來上網碰到錯誤代碼451，你瀏覽的網頁被封鎖了!. iThome. 2015-12-22 [2015-12-22].
56. Server Error Codes. CSGNetwork.com. [16 October 2015].
57. mrGott. HTTP Status Codes To Handle Errors In Your API. mrGott. [16 October 2015]. （原始内容存档于2015年9月30日）.
58. Fisher, Tim. 500 Internal Server Error. Lifewire. [22 February 2017].
59. HTTP Error 501 Not implemented. Check Up Down. [22 February 2017].
60. Fisher, Tim. 502 Bad Gateway. Lifewire. [22 February 2017].

61. alex. What is the correct HTTP status code to send when a site is down for maintenance? Stack Overflow. [16 October 2015].
62. HTTP Error 504 Gateway timeout. Check Up Down. [16 October 2015].
63. HTTP Error 505 - HTTP version not supported. Check Up Down. [16 October 2015].
64. Holtman, Koen; Mutz, Andrew H.. Transparent Content Negotiation in HTTP (https://tools.ietf.org/html/rfc2295). IETF. March 1998 [October 24, 2009]. RFC 2295.
65. Nielsen, Henrik Frystyk; Leach, Paul; Lawrence, Scott. An HTTP Extension Framework (https://tools.ietf.org/html/rfc2774). IETF. February 2000 [October 24, 2009]. RFC 2774.

外部链接

- Hypertext Transfer Protocol (HTTP) Status Code Registry
- 微软知识库：MSKB943891: IIS 7.0、IIS 7.5 和 IIS 8.0 中的 HTTP 状态代码
