HTTP 服务器主要操作是进行 HTTP 请求处理的。HTTP 属于超文本传输协议，是在整个 Web 开发过程之中使用最广泛的处理协议，同时也是所有的服务器必用的开发协议。HTTP 协议实际上是构建在 TCP 上的一种应用，在使用 HTTP 协议的时候本质上还是进行数据的传输与响应的。

HTTP 请求信息(request)

请求信息是由客户端发送给浏览器的，但是需要注意的只要遵从 HTTP 协议标准的请求都可以发送到服务器端，也就是说发送请求的不应只有浏览器，也可以通过客户端进行发送(eg:HttpClient、Netty)。

HTTP1.0：GET、 POST 、HEAD

HTTP1.1：OPTIONS 、 PUT 、 DELETE 、TRACE 、CONNECT

No.	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于 get 请求，只不过返回的响应中没有具体的内容，用于获取报头。
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中，POST 请求可能会导致新的资源的建立或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP1.1 协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	运行客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

在进行请求的时候除了数据本身之外，还会包含有头部的信息内容，而在请求处理之中常见的头部信息如下：

协议头	说明	示例	状态
Accept	可接受的响应内容类型（Content-Types）。	Accept: text/plain	固定
Accept-Charset	可接受的字符集	Accept-Charset: utf-8	固定
Accept-Encoding	可接受的响应内容的编码方式。	Accept-Encoding: gzip, deflate	固定
Accept-Language	可接受的响应内容语言列表。	Accept-Language: en-US	固定
Accept-Datetime	可接受的按照时间来表示的响应内容版本	Accept-Datetime: Sat, 26 Dec 2015 17:30:00 GMT	临时
Authorization	用于表示 HTTP 协议中需要认证资源的认证信息	Authorization: Basic OSdjJGRpbjpvcGVuIANlc2SdDE==	固定
Cache-Control	用来指定当前的请求/回复中的，是否使用缓存机制。	Cache-Control: no-cache	固定

协议头	说明	示例	状态
Connection	客户端（浏览器）想要优先使用的连接类型	Connection: keep-alive Connection: Upgrade	固定
Cookie	由之前服务器通过 Set-Cookie（见下文）设置的一个 HTTP 协议 Cookie	Cookie: $Version=1; Skin=new;	固定：标准
Content-Length	以 8 进制表示的请求体的长度	Content-Length: 348	固定
Content-MD5	请求体的内容的二进制 MD5 散列值（数字签名），以 Base64 编码的结果	Content-MD5: oD8dH2sgSW50ZWdyaIEd9D==	废弃
Content-Type	请求体的 MIME 类型 （用于 POST 和 PUT 请求中）	Content-Type: application/x-www-form-urlencoded	固定
Date	发送该消息的日期和时间（以 RFC 7231 中定义的"HTTP 日期"格式来发送）	Date: Dec, 26 Dec 2015 17:30:00 GMT	固定
Expect	表示客户端要求服务器做出特定的行为	Expect: 100-continue	固定
From	发起此请求的用户的邮件地址	From: user@itbilu.com	固定
Host	表示服务器的域名以及服务器所监听的端口号。如果所请求的端口是对应的服务的标准端口（80），则端口号可以省略。	Host: www.itbilu.com:80 Host: www.itbilu.com	固定
If-Match	仅当客户端提供的实体与服务器上对应的实体相匹配时，才进行对应的操作。主要用于像 PUT 这样的方法中，仅当从用户上次更新某个资源后，该资源未被修改的情况下，才更新该资源。	If-Match: "9jd00cdj34pss9ejqiw39d82f20d0ikd"	固定
If-Modified-Since	允许在对应的资源未被修改的情况下返回 304 未修改	If-Modified-Since: Dec, 26 Dec 2015 17:30:00 GMT	固定
If-None-Match	允许在对应的内容未被修改的情况下返回 304 未修改（ 304 Not Modified ），参考 超文本传输协议 的实体标记	If-None-Match: "9jd00cdj34pss9ejqiw39d82f20d0ikd"	固定
If-Range	如果该实体未被修改过，则向返回所缺少的那一个或多个部分。否则，返回整个新的实体	If-Range: "9jd00cdj34pss9ejqiw39d82f20d0ikd"	固定
If-Unmodified-Since	仅当该实体自某个特定时间以来未被修改的情况下，才发送回应。	If-Unmodified-Since: Dec, 26 Dec 2015 17:30:00 GMT	固定

协议头	说明	示例	状态
Max-Forwards	限制该消息可被代理及网关转发的次数。	Max-Forwards: 10	固定
Origin	发起一个针对跨域资源共享的请求（该请求要求服务器在响应中加入一个Access-Control-Allow-Origin 的消息头，表示访问控制所允许的来源）。	Origin: http://www.itbilu.com	固定：标准
Pragma	与具体的实现相关，这些字段可能在请求/回应链中的任何时候产生。	Pragma: no-cache	固定
Proxy-Authorization	用于向代理进行认证的认证信息。	Proxy-Authorization: Basic IOoDZRgDOi0vcGVuIHNlNidJi2==	固定
Range	表示请求某个实体的一部分，字节偏移以 0 开始。	Range: bytes=500-999	固定
Referer	表示浏览器所访问的前一个页面，可以认为是之前访问页面的链接将浏览器带到了当前页面。Referer 其实是 Referrer 这个单词，但 RFC 制作标准时给拼错了，后来也就将错就错使用 Referer 了。	Referer: http://itbilu.com/nodejs	固定
TE	浏览器预期接受的传输时的编码方式：可使用回应协议头 Transfer-Encoding 中的值（还可以使用"trailers"表示数据传输时的分块方式）用来表示浏览器希望在最后一个大小为 0 的块之后还接收到一些额外的字段。	TE: trailers,deflate	固定
User-Agent	浏览器的身份标识字符串	User-Agent: Mozilla/……	固定
Upgrade	要求服务器升级到一个高版本协议。	Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11	固定
Via	告诉服务器，这个请求是由哪些代理发出的。	Via: 1.0 fred, 1.1 itbilu.com.com (Apache/1.1)	固定
Warning	一个一般性的警告，表示在实体内容体中可能存在错误。	Warning: 199 Miscellaneous warning	固定

HTTP 响应(response)

在进行 HTTP 响应过程之中首先要注意的就是一个请求编码，如果请求编码为 200 则表示请求成功可以进行数据的返回，常见的 HTTP 状态码如下：

No.	分类	描述
1	1xx	服务器收到请求，需要请求者继续执行操作
2	2xx	成功，操作被成功接收并处理
3	3xx	重定向，需要进一步的操作以完成请求
4	4xx	客户端错误，请求包含语法错误或无法完成请求
5	5xx	服务器端错误，服务器在处理请求的过程之中发生了错误

状态码	含义
100	客户端应当继续发送请求。这个临时响应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，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。
101	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。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 版本比旧版本更有优势，或者切换到一个实时且同步的协议以传送利用此类特性的资源。
102	由 WebDAV（RFC 2518）扩展的状态码，代表处理将被继续执行。
200	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。
201	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'。
202	服务器已接受请求，但尚未处理。正如它可能被拒绝一样，最终该请求可能会也可能不会被执行。在异步操作的场合下，没有比发送这个状态码更方便的做法了。 返回 202 状态码的响应的目的是允许服务器接受其他过程的请求（例如某个每天只执行一次的基于批处理的操作），而不必让客户端一直保持与服务器的连接直到批处理操作全部完成。在接受请求处理并返回 202 状态码的响应应当在返回的实体中包含一些指示处理当前状态的信息，以及指向处理状态监视器或状态预测的指针，以便用户能够估计操作是否已经完成。
203	服务器已成功处理了请求，但返回的实体头部元信息不是在原始服务器上有效的确定集合，而是来自本地或者第三方的拷贝。当前的信息可能是原始版本的子集或者超集。例如，包含资源的元数据可能导致原始服务器知道元信息的超级。使用此状态码不是必须的，而且只有在响应不使用此状态码便会返回 200 OK 的情况下才是合适的。
204	服务器成功处理了请求，但不需要返回任何实体内容，并且希望返回更新了的元信息。响应可能通过实体头部的形式，返回新的或更新后的元信息。如果存在这些头部信息，则应当与所请求的变量

	相呼应。 如果客户端是浏览器的话，那么用户浏览器应保留发送了该请求的页面，而不产生任何文档视图上的变化，即使按照规范新的或更新后的元信息应当被应用到用户浏览器活动视图中的文档。 由于 204 响应被禁止包含任何消息体，因此它始终以消息头后的第一个空行结尾。
205	服务器成功处理了请求，且没有返回任何内容。但是与 204 响应不同，返回此状态码的响应要求请求者重置文档视图。该响应主要是被用于接受用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。 与 204 响应一样，该响应也被禁止包含任何消息体，且以消息头后的第一个空行结束。
206	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。 该请求必须包含 Range 头信息来指示客户端希望得到的内容范围，并且可能包含 If-Range 来作为请求条件。 响应必须包含如下的头部域： Content-Range 用以指示本次响应中返回的内容的范围；如果是 Content-Type 为 multipart/byteranges 的多段下载，则每一 multipart 段中都应包含 Content-Range 域用以指示本段的内容范围。假如响应中包含 Content-Length，那么它的数值必须匹配它返回的内容范围的真实字节数。 Date ETag 和/或 Content-Location，假如同样的请求本应该返回 200 响应。 Expires, Cache-Control，和/或 Vary，假如其值可能与之前相同变量的其他响应对应的值不同的话。 假如本响应请求使用了 If-Range 强缓存验证，那么本次响应不应该包含其他实体头；假如本响应的请求使用了 If-Range 弱缓存验证，那么本次响应禁止包含其他实体头；这避免了缓存的实体内容和更新了的实体头信息之间的不一致。否则，本响应就应当包含所有本应该返回 200 响应中应当返回的所有实体头部域。 假如 ETag 或 Last-Modified 头部不能精确匹配的话，则客户端缓存应禁止将 206 响应返回的内容与之前任何缓存过的内容组合在一起。 任何不支持 Range 以及 Content-Range 头的缓存都禁止缓存 206 响应返回的内容。
207	由 WebDAV(RFC 2518)扩展的状态码，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。
300	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。 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	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。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，因此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	请求的资源现在临时从不同的 URI 响应请求。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。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。
303	对应当前请求的响应可以在另一个 URI 上被找到，而且客户端应当采用 GET 的方式访问那个资源。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	如果客户端发送了一个带条件的 GET 请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变，则服务器应当返回这个状态码。304 响应禁止包含消息体，因此始终以消息头后的第一个空行结尾。　　该响应必须包含以下的头信息：　　Date，除非这个服务器没有时钟。假如没有时钟的服务器也遵守这些规则，那么代理服务器以及客户端可以自行将 Date 字段添加到接收到的响应头中去(正如 RFC 2068 中规定的一样)，缓存机制将会正常工作。　　ETag 和/或 Content-Location，假如同样的请求本应返回 200 响应。　　Expires, Cache-Control，和/或 Vary，假如其值可能与之前相同变量的其他响应对应的值不同的话。　　假如本响应请求使用了强缓存验证，那么本次响应不应该包含其他实体头；否则（例如，某个带条件的 GET 请求使用了弱缓存验证），本次响应禁止包含其他实体头；这避免了缓存了的实体内容和更新了的实体头信息之间的不一致。　　假如某个 304 响应指明了当前某个实体没有缓存，那么缓存系统必须忽视这个响应，并且重复发送不包含限制条件的请求。　　假如接收到一个要求更新某个缓存条目的 304 响应，那么缓存系统必须更新整个条目以反映所有在响应中被更新的字段的值。
305	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。　　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	在最新版的规范中，306 状态码已经不再被使用。
307	请求的资源现在临时从不同的 URI 响应请求。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。因为部分浏览器不能识别 307 响应，因此需要添加上述必要信息以便用户能够理解并向新的 URI 发出访问请求。　　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。
400	1、语义有误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。 2、请求参数有误。
401	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。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。参见 RFC 2617。
402	该状态码是为了将来可能的需求而预留的。
403	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	请求失败，请求所希望得到的资源未被在服务器上发现。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	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。 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	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。 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	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。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。参见 RFC 2617。
408	请求超时。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。
409	由于和被请求的资源的当前状态之间存在冲突，请求无法完成。这个代码只允许用在这样的情况下才能被使用：用户被认为能够解决冲突，并且会重新提交新的请求。该响应应当包含足够的信息以便用户发现冲突的源头。 冲突通常发生于对 PUT 请求的处理中。例如，在采用版本检查的环境下，某次 PUT 提交的对特定资源的修改请求所附带的版本信息与之前的某个（第三方）请求向冲突，那么此时服务器就应该返回一个 409 错误，告知用户请求无法完成。此时，响应实体中很可

	能会包含两个冲突版本之间的差异比较，以便用户重新提交归并以后的新版本。
410	被请求的资源在服务器上已经不再可用，而且没有任何已知的转发地址。这样的状况应当被认为是永久性的。如果可能，拥有链接编辑功能的客户端应当在获得用户许可后删除所有指向这个地址的引用。如果服务器不知道或者无法确定这个状况是否是永久的，那么就应该使用 404 状态码。除非额外说明，否则这个响应是可缓存的。 410 响应的目的主要是帮助网站管理员维护网站，通知用户该资源已经不再可用，并且服务器拥有者希望所有指向这个资源的远端连接也被删除。这类事件在限时、增值服务中很普遍。同样，410 响应也被用于通知客户端在当前服务器站点上，原本属于某个个人的资源已经不再可用。当然，是否需要把所有永久不可用的资源标记为'410 Gone'，以及是否需要保持此标记多长时间，完全取决于服务器拥有者。
411	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。
412	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。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	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。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	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。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（Query String）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[1]。没有此类漏洞的服务器，应当返回 414 状态码。
415	对于当前请求的方法和所请求的资源，请求中提交的实体并不是服务器中所支持的格式，因此请求被拒绝。
416	如果请求中包含了 Range 请求头，并且 Range 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中又没有定义 If-Range 请求头，那么服务器就应当返回 416 状态码。 假如 Range 使用的是字节范围，那么这种情况就是指请求指定的所有数据范围的首字节位置都超过了当前资源的长度。服务器也应当在返回 416 状态码的同时，包含一个 Content-Range 实体头，用以指明当前资源的长度。这个响应也被禁止使用 multipart/byteranges 作为其 Content-Type。
417	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务器，它有明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。
421	从当前客户端所在的 IP 地址到服务器的连接数超过了服务器许可的最大范围。通常，这里的 IP 地址指的是从服务器上看到的客户端地址（比如用户的网关或者代理服务器地址）。在这种情况下，连接数的计算可能涉及到不止一个终端用户。

422	从当前客户端所在的 IP 地址到服务器的连接数超过了服务器许可的最大范围。通常，这里的 IP 地址指的是从服务器上看到的客户端地址（比如用户的网关或者代理服务器地址）。在这种情况下，连接数的计算可能涉及到不止一个终端用户。
422	请求格式正确，但是由于含有语义错误，无法响应。（RFC 4918 WebDAV）423 Locked 当前资源被锁定。（RFC 4918 WebDAV）
424	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。（RFC 4918 WebDAV）
425	在 WebDav Advanced Collections 草案中定义，但是未出现在《WebDAV 顺序集协议》（RFC 3658）中。
426	客户端应当切换到 TLS/1.0。（RFC 2817）
449	由微软扩展，代表请求应当在执行完适当的操作后进行重试。
500	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。一般来说，这个问题都会在服务器的程序码出错时出现。
501	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。
502	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。
503	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。这个状况是临时的，并且将在一段时间以后恢复。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。 注意：503 状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。
504	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。 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	服务器不支持，或者拒绝支持在请求中使用的 HTTP 版本。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	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：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	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。WebDAV (RFC 4918)
509	服务器达到带宽限制。这不是一个官方的状态码，但是仍被广泛使用。
510	获取资源所需要的策略并没有没满足。（RFC 2774）

响应头信息：

响应头	说明	示例	状态
Access-Control-Allow-Origin	指定哪些网站可以跨域源资源共享	Access-Control-Allow-Origin: *	临时
Accept-Patch	指定服务器所支持的文档补丁格式	Accept-Patch: text/example;charset=utf-8	固定
Accept-Ranges	服务器所支持的内容范围	Accept-Ranges: bytes	固定
Age	响应对象在代理缓存中存在的时间，以秒为单位	Age: 12	固定
Allow	对于特定资源的有效动作;	Allow: GET, HEAD	固定
Cache-Control	通知从服务器到客户端内的所有缓存机制，表示它们是否可以缓存这个对象及缓存有效时间。其单位为秒	Cache-Control: max-age=3600	固定
Connection	针对该连接所预期的选项	Connection: close	固定
Content-Disposition	对已知 MIME 类型资源的描述，浏览器可以根据这个响应头决定是对返回资源的动作，如：将其下载或是打开。	Content-Disposition: attachment; filename="fname.ext"	固定
Content-Encoding	响应资源所使用的编码类型。	Content-Encoding: gzip	固定
Content-Language	响就内容所使用的语言	Content-Language: zh-cn	固定
Content-Length	响应消息体的长度，用 8 进制字节表示	Content-Length: 348	固定
Content-Location	所返回的数据的一个候选位置	Content-Location: /index.htm	固定
Content-MD5	响应内容的二进制 MD5 散列值，以 Base64 方式编码	Content-MD5: IDK0iSsgSW50ZWd0DiJUi==	已淘汰
Content-Range	如果是响应部分消息，表示属于完整消息的哪个部分	Content-Range: bytes 21010-47021/47022	固定

响应头	说明	示例	状态
Content-Type	当前内容的 MIME 类型	Content-Type: text/html; charset=utf-8	固定
Date	此条消息被发送时的日期和时间(以 RFC 7231 中定义的"HTTP 日期"格式来表示)	Date: Tue, 15 Nov 1994 08:12:31 GMT	固定
ETag	对于某个资源的某个特定版本的一个标识符，通常是一个 消息散列	ETag: "737060cd8c284d8af7ad3082f209582d"	固定
Expires	指定一个日期/时间，超过该时间则认为此回应已经过期	Expires: Thu, 01 Dec 1994 16:00:00 GMT	固定:标准
Last-Modified	所请求的对象的最后修改日期(按照 RFC 7231 中定义的“超文本传输协议日期”格式来表示)	Last-Modified: Dec, 26 Dec 2015 17:30:00 GMT	固定
Link	用来表示与另一个资源之间的类型关系，此类型关系是在 RFC 5988 中定义	Link: ; rel="alternate"	固定
Location	用于在进行重定向,或在创建了某个新资源时使用。	Location: http://www.itbilu.com/nodejs	固定
P3P	P3P 策略相关设置	P3P: CP="This is not a P3P policy!	固定
Pragma	与具体的实现相关,这些响应头可能在请求/回应链中的不同时候产生不同的效果	Pragma: no-cache	固定
Proxy-Authenticate	要求在访问代理时提供身份认证信息。	Proxy-Authenticate: Basic	固定
Public-Key-Pins	用于防止中间攻击,声明网站认证中传输层安全协议的证书散列值	Public-Key-Pins: max-age=2592000; pin-sha256="……";	固定
Refresh	用于重定向,或者当一个新的资源被创建时。默认会在 5 秒后刷新重定向。	Refresh: 5; url=http://itbilu.com	
Retry-After	如果某个实体临时不可用,那么此协议头用于告知客户端稍后重试。其值可以是一个特定的时间段(以秒为单位)或一个超文本传输协议日期。	• 示例 1:Retry-After: 120 • 示例 2: Retry-After: Dec, 26 Dec 2015 17:30:00 GMT	固定
Server	服务器的名称	Server: nginx/1.6.3	固定

响应头	说明	示例	状态
Set-Cookie	设置 HTTP cookie	Set-Cookie: UserID=itbilu; Max-Age=3600; Version=1	固定:标准
Status	通用网关接口的响应头字段，用来说明当前 HTTP 连接的响应状态。	Status: 200 OK	
Trailer	Trailer 用户说明传输中分块编码的编码信息	Trailer: Max-Forwards	固定
Transfer-Encoding	用表示实体传输给用户的编码形式。包括：chunked、compress、deflate、gzip、identity。	Transfer-Encoding: chunked	固定
Upgrade	要求客户端升级到另一个高版本协议。	Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11	固定
Vary	告知下游的代理服务器，应当如何对以后的请求协议头进行匹配，以决定是否可使用已缓存的响应内容而不是重新从原服务器请求新的内容。	Vary: *	固定
Via	告知代理服务器的客户端，当前响应是通过什么途径发送的。	Via: 1.0 fred, 1.1 itbilu.com (nginx/1.6.3)	固定
Warning	一般性警告，告知在实体内容体中可能存在错误。	Warning: 199 Miscellaneous warning	固定
WWW-Authenticate	表示在请求获取这个实体时应当使用的认证模式。	WWW-Authenticate: Basic	固定

部分内容 via: https://www.cnblogs.com/honghong87/articles/6941436.html