HTTP协议（HyperText Transfer Protocol，超文本传输协议）是因特网上应用最为广泛的一种网络传输协议，所有的WWW文件都必须遵守这个标准。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

**HTTP 简介**

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

**HTTP 工作原理**

HTTP协议工作于客户端-服务端架构上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。

Web服务器有：Apache服务器，IIS服务器（Internet Information Services）等。

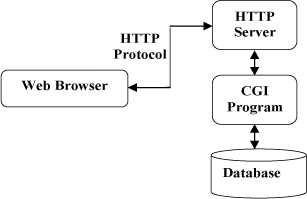
Web服务器根据接收到的请求后，向客户端发送响应信息。

HTTP默认端口号为80，但是你也可以改为8080或者其他端口。

**HTTP三点注意事项：**

* HTTP是无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
* HTTP是媒体独立的：这意味着，只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送。客户端以及服务器指定使用适合的MIME-type内容类型。
* HTTP是无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

以下图表展示了HTTP协议通信流程：



# HTTP 消息结构

HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

一个HTTP"客户端"是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。

一个HTTP"服务器"同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。

HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。

一旦建立连接后，数据消息就通过类似Internet邮件所使用的格式[RFC5322]和多用途Internet邮件扩展（MIME）[RFC2045]来传送。

## 客户端请求消息

**客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成，下图给出了请求报文的一般格式。**



## 服务器响应消息

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。



## 实例

下面实例是一点典型的使用GET来传递数据的实例：

客户端请求：

GET /hello.txt HTTP/1.1

User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3

Host: www.example.com

Accept-Language: en, mi

服务端响应:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: "34aa387-d-1568eb00"

Accept-Ranges: bytes

Content-Length: 51

Vary: Accept-Encoding

Content-Type: text/plain

输出结果：

Hello World! My payload includes a trailing CRLF.

# HTTP请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

|  |  |  |
| --- | --- | --- |
| **序号** | **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和/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 | HTTP/1.1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 | 允许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。 |

**HTTP 响应头信息**

HTTP请求头提供了关于请求，响应或者其他的发送实体的信息。

在本章节中我们将具体来介绍HTTP响应头信息。

|  |  |
| --- | --- |
| **应答头** | **说明** |
| Allow | 服务器支持哪些请求方法（如GET、POST等）。 |
| Content-Encoding | 文档的编码（Encode）方法。只有在解码之后才可以得到Content-Type头指定的内容类型。利用gzip压缩文档能够显著地减少HTML文档的下载时间。Java的GZIPOutputStream可以很方便地进行gzip压缩，但只有Unix上的Netscape和Windows上的IE 4、IE 5才支持它。因此，Servlet应该通过查看Accept-Encoding头（即request.getHeader("Accept-Encoding")）检查浏览器是否支持gzip，为支持gzip的浏览器返回经gzip压缩的HTML页面，为其他浏览器返回普通页面。 |
| Content-Length | 表示内容长度。只有当浏览器使用持久HTTP连接时才需要这个数据。如果你想要利用持久连接的优势，可以把输出文档写入 ByteArrayOutputStream，完成后查看其大小，然后把该值放入Content-Length头，最后通过byteArrayStream.writeTo(response.getOutputStream()发送内容。 |
| Content-Type | 表示后面的文档属于什么MIME类型。Servlet默认为text/plain，但通常需要显式地指定为text/html。由于经常要设置Content-Type，因此HttpServletResponse提供了一个专用的方法setContentType。 |
| Date | 当前的GMT时间。你可以用setDateHeader来设置这个头以避免转换时间格式的麻烦。 |
| Expires | 应该在什么时候认为文档已经过期，从而不再缓存它？ |
| Last-Modified | 文档的最后改动时间。客户可以通过If-Modified-Since请求头提供一个日期，该请求将被视为一个条件GET，只有改动时间迟于指定时间的文档才会返回，否则返回一个304（Not Modified）状态。Last-Modified也可用setDateHeader方法来设置。 |
| Location | 表示客户应当到哪里去提取文档。Location通常不是直接设置的，而是通过HttpServletResponse的sendRedirect方法，该方法同时设置状态代码为302。 |
| Refresh | 表示浏览器应该在多少时间之后刷新文档，以秒计。除了刷新当前文档之外，你还可以通过setHeader("Refresh", "5; URL=http://host/path")让浏览器读取指定的页面。  注意这种功能通常是通过设置HTML页面HEAD区的＜META HTTP-EQUIV="Refresh" CONTENT="5;URL=http://host/path"＞实现，这是因为，自动刷新或重定向对于那些不能使用CGI或Servlet的HTML编写者十分重要。但是，对于Servlet来说，直接设置Refresh头更加方便。   注意Refresh的意义是"N秒之后刷新本页面或访问指定页面"，而不是"每隔N秒刷新本页面或访问指定页面"。因此，连续刷新要求每次都发送一个Refresh头，而发送204状态代码则可以阻止浏览器继续刷新，不管是使用Refresh头还是＜META HTTP-EQUIV="Refresh" ...＞。   注意Refresh头不属于HTTP 1.1正式规范的一部分，而是一个扩展，但Netscape和IE都支持它。 |
| Server | 服务器名字。Servlet一般不设置这个值，而是由Web服务器自己设置。 |
| Set-Cookie | 设置和页面关联的Cookie。Servlet不应使用response.setHeader("Set-Cookie", ...)，而是应使用HttpServletResponse提供的专用方法addCookie。参见下文有关Cookie设置的讨论。 |
| WWW-Authenticate | 客户应该在Authorization头中提供什么类型的授权信息？在包含401（Unauthorized）状态行的应答中这个头是必需的。例如，response.setHeader("WWW-Authenticate", "BASIC realm=＼"executives＼"")。  注意Servlet一般不进行这方面的处理，而是让Web服务器的专门机制来控制受密码保护页面的访问（例如.htaccess）。 |

**服务器返回的14种常见HTTP状态码**

当我们从客户端向服务器发送请求时   
服务器向我们返回状态码   
状态码就是告诉我们服务器响应的状态   
通过它，我们就可以知道当前请求是成功了还是出现了什么问题   
状态码是由3位数字和原因短语组成的（比如最常见的：200 OK）   
其中第一位数字表示响应类别，响应类别从1到5分为五种   
add：其实我查阅资料发现还有一个6打头的一个 600 Unparseable Response Headers   
表示服务器没有返回响应头部，只返回实体内容，也算做服务器错误状态码吧，不过绝对不常见

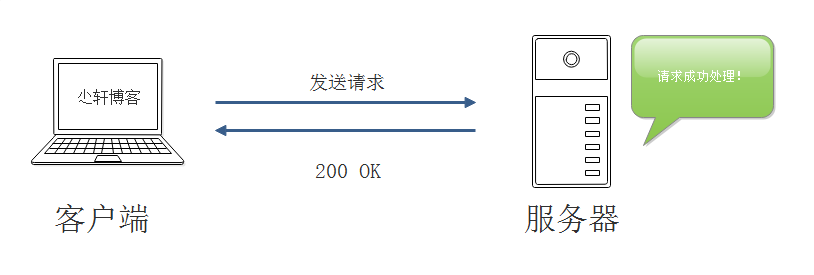
| **状态码** | **响应类别** | **原因短语** |
| --- | --- | --- |
| 1XX | 信息性状态码（Informational） | 服务器正在处理请求 |
| 2XX | 成功状态码（Success） | 请求已正常处理完毕 |
| 3XX | 重定向状态码（Redirection） | 需要进行额外操作以完成请求 |
| 4XX | 客户端错误状态码（Client Error） | 客户端原因导致服务器无法处理请求 |
| 5XX | 服务器错误状态码（Server Error） | 服务器原因导致处理请求出错 |

状态码响应类别的的定义是必须遵守的，后面的两个数字服务器自己DIY也没什么问题   
RFC2616（用来指定HTTP协议标准的文档）标准定义的就有四十多种，加上扩展能达到六十种   
不过常见的大概只有十四种，下面我来谈谈这些有代表性的14种HTTP状态码

**2XX 成功**

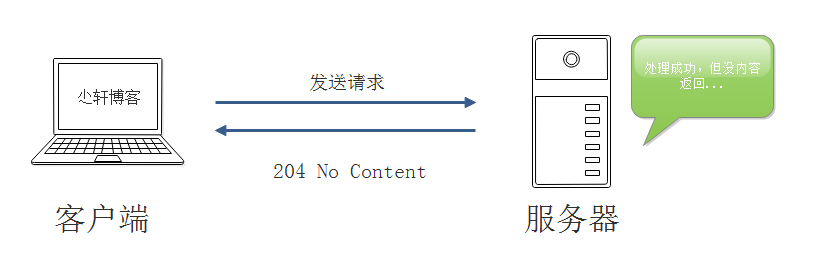
**200 OK**

表示请求被服务器正常处理   
最常见的就是这个了   
随着这个状态码返回的信息跟你的请求方法有关   
比如GET请求，请求的资源会作为响应实体返回   
而HEAD请求，信息只存在于响应报文首部，因为它不会返回报文实体，只返回报文首部



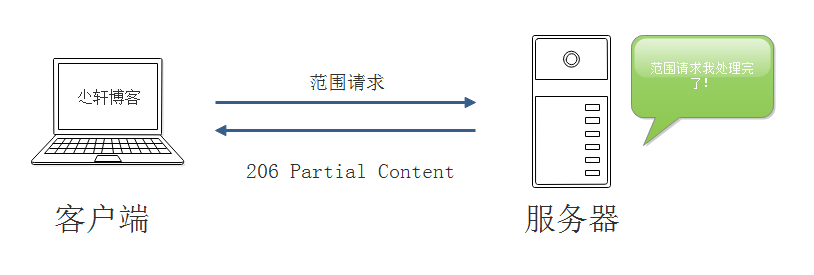
**204 No Content**

表示请求已成功处理，但是没有内容返回（就应该没有内容返回的状况）   
也就是返回的响应报文中没有报文实体（其实是没有报文实体的主体部分）   
浏览器向服务器发送请求后收到了204，那么浏览器页面不会发生更新   
一般用在只是客户端向服务器发送信息，而服务器不用向客户端返回什么信息的情况



**206 Partial Content**

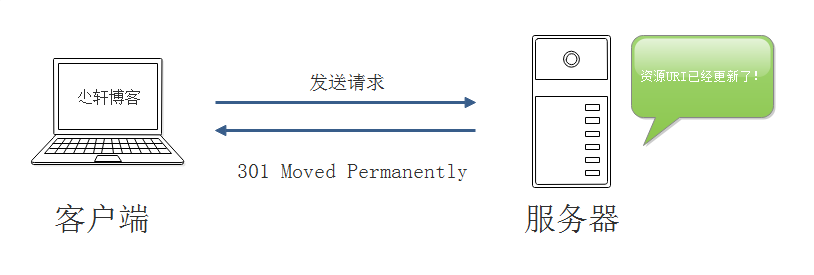
表示服务器已经完成了部分GET请求（客户端进行了范围请求）   
响应报文中包含Content-Range指定范围的实体内容



**3XX 重定向**

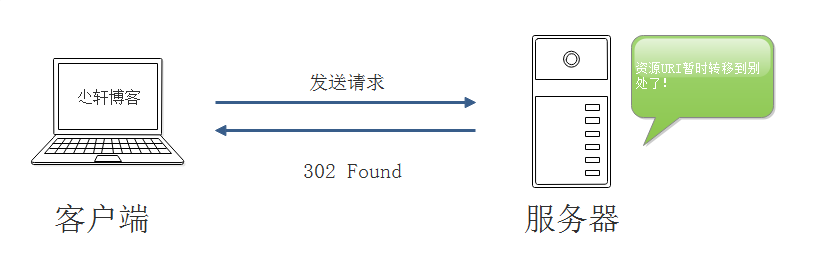
**301 Moved Permanently**

永久重定向，表示请求的资源已经永久的搬到了其他位置   
就是说资源已经被分配了新的URI   
新的URI应该提示在响应报文的Location首部字段   
只要不是HEAD请求，响应实体应该包含新URI的超链接和简短的说明



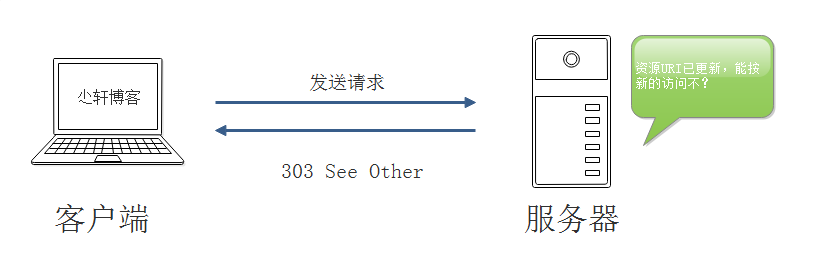
**302 Found**

临时重定向，表示请求的资源临时搬到了其他位置   
请求的资源暂时被配到到了新的URI   
和301很像，只不过资源是临时移动，资源在将来可能还会改变   
同样地，新的临时URI应该提示在响应报文的Location首部字段   
只要不是HEAD请求，响应实体应该包含新URI的超链接和简短的说明



**303 See Other**

表示请求资源存在另一个URI，应使用GET定向获取请求资源   
303功能与302一样，区别只是303明确客户端应该使用GET访问   
（很多HTTP/1.1之前的浏览器不能理解303，但是大家都把302当303对待，使用GET请求新URI）

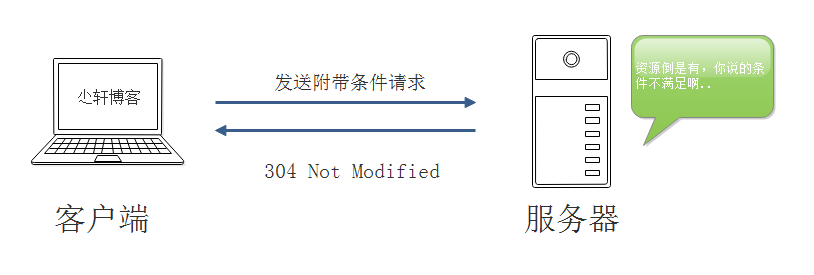


**304 Not Modified**

表示客户端发送附带条件的请求（GET方法请求报文中的IF…）时，条件不满足

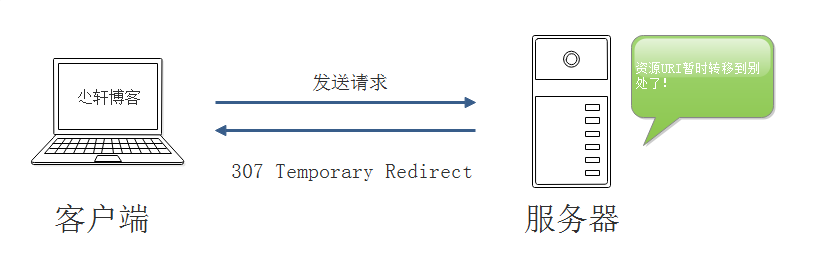
HTTP **304 未改变**说明无需再次传输请求的内容，也就是说可以使用缓存的内容。这通常是在一些安全的方法（[safe](https://developer.mozilla.org/en-US/docs/Glossary/safe)），例如[GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET) 或[HEAD](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/HEAD) 或在请求中附带了头部信息： [If-None-Match](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/If-None-Match) 或[If-Modified-Since](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/If-Modified-Since)。

返回304时，不包含任何响应主体   
虽然304被划分在3XX，但和重定向一毛钱关系都没有



**307 Temporary Redirect**

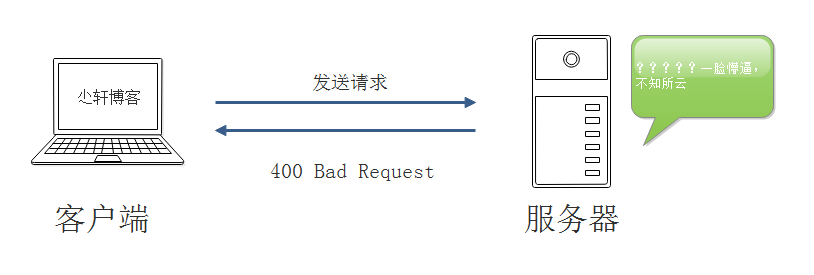
临时重定向，和302有着相同含义   
尽管302标准禁止POST变为GET，但没人听他的   
而307就会遵照标准，不会从POST变为GET   
但处理响应行为，各个浏览器可能不同



**4XX 客户端错误**

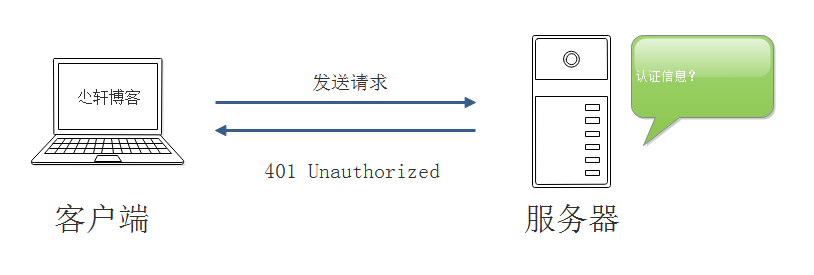
**400 Bad Request**

表示请求报文存在语法错误或参数错误，服务器不理解   
服务器不应该重复提交这个请求   
需要修改请求内容后再次发送



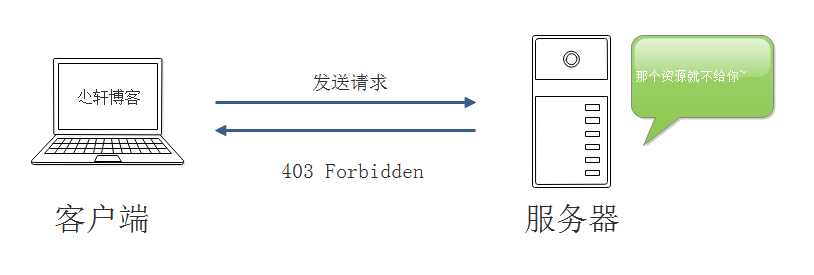
**401 Unauthorized**

表示发送的请求需要有HTTP认证信息或者是认证失败了   
返回401的响应必须包含一个适用于被请求资源的WWW-Authenticate首部以质询用户信息   
浏览器初次接受401时，会弹出认证窗口



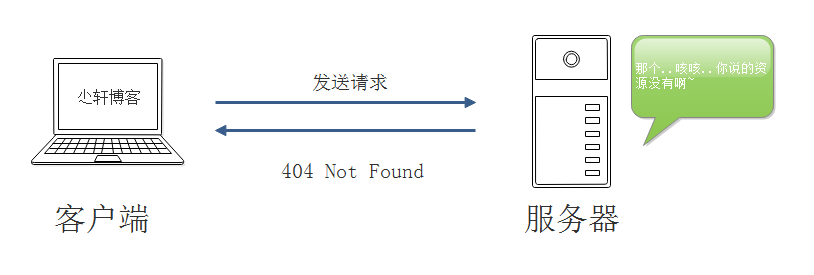
**403 Forbidden**

表示对请求资源的访问被服务器拒绝了   
服务器可以对此作出解释，也可以不解释   
想说明的话可以在响应实体的主体部分描述原因   
比如说你可能没有访问权限



**404 Not Found**

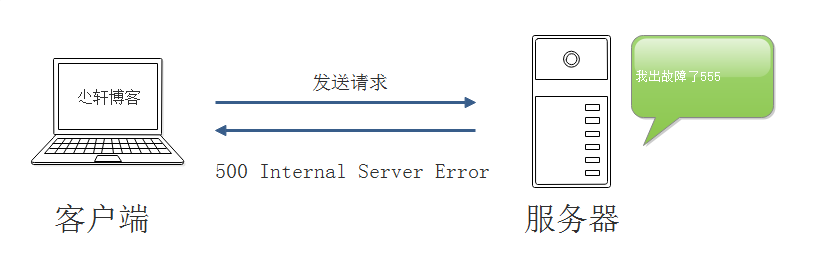
表示服务器找不到你请求的资源   
也有可能服务器就是不想给你然后骗你找不到(⊙ˍ⊙)   
而且大多服务器都是这么玩这个状态码的



**5XX 服务器错误**

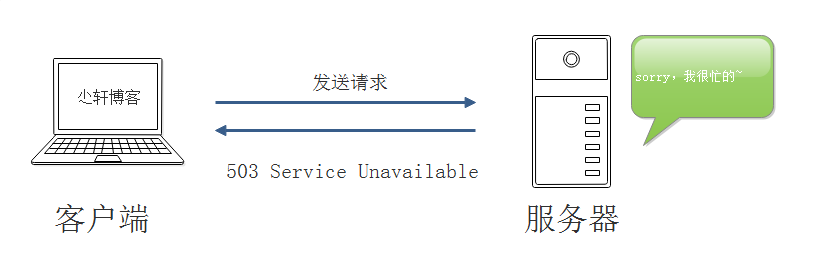
**500 Internal Server Error**

表示服务器执行请求的时候出错了   
可能是Web应用有bug或临时故障   
更有可能是服务器源代码有bug…



**503 Service Unavailable**

表示服务器超负载或正停机维护，无法处理请求   
如果服务器知道还需要多长时间，就写入Retry-After首部字段返回



**总结**

返回的状态码和状态不一致的情况是有可能发生得   
比如Web应用程序内部错误，但仍然返回 200 OK

* 200 OK   
  请求正常处理完毕
* 204 No Content   
  请求成功处理，没有实体的主体返回
* 206 Partial Content   
  GET范围请求已成功处理
* 301 Moved Permanently   
  永久重定向，资源已永久分配新URI
* 302 Found   
  临时重定向，资源已临时分配新URI
* 303 See Other   
  临时重定向，期望使用GET定向获取
* 304 Not Modified   
  发送的附带条件请求未满足
* 307 Temporary Redirect   
  临时重定向，POST不会变成GET
* 400 Bad Request   
  请求报文语法错误或参数错误
* 401 Unauthorized   
  需要通过HTTP认证，或认证失败
* 403 Forbidden   
  请求资源被拒绝
* 404 Not Found   
  无法找到请求资源（服务器无理由拒绝）
* 500 Internal Server Error   
  服务器故障或Web应用故障

|  |  |
| --- | --- |
| * 502（错误网关） | 服务器作为网关或代理，从上游服务器收到无效响应。 |

* 503 Service Unavailable   
  服务器超负载或停机维护