

## 12 | 响应状态码该怎么用？

2019-06-24 Chrono

《透视HTTP协议》

[课程介绍 >](#)



讲述：Chrono

时长 12:16 大小 16.87M



前两讲中，我们学习了 HTTP 报文里请求行的组成部分，包括请求方法和 URI。有了请求行，加上后面的头字段就形成了请求头，可以通过 TCP/IP 协议发送给服务器。

服务器收到请求报文，解析后需要进行处理，具体的业务逻辑多种多样，但最后必定是拼出一个响应报文发回客户端。

响应报文由响应头加响应体数据组成，响应头又由状态行和头字段构成。

我们先来复习一下状态行的结构，有三部分：

状态行

Version

SP

Status Code

SP

Reason

CRLF

领资料



开头的 Version 部分是 HTTP 协议的版本号，通常是 HTTP/1.1，用处不是很大。

后面的 Reason 部分是原因短语，是状态码的简短文字描述，例如“OK”“Not Found”等等，也可以自定义。但它只是为了兼容早期的文本客户端而存在，提供的信息很有限，目前的大多数客户端都会忽略它。

所以，状态行里有用的就只剩下中间的**状态码**（Status Code）了。它是一个十进制数字，以代码的形式表示服务器对请求的处理结果，就像我们通常编写程序时函数返回的错误码一样。

不过你要注意，它的名字是“状态码”而不是“错误码”。也就是说，它的含义不仅是错误，更重要的意义在于表达 HTTP 数据处理的“状态”，客户端可以依据代码适时转换处理状态，例如继续发送请求、切换协议，重定向跳转等，有那么点 TCP 状态转换的意思。

## 状态码

目前 RFC 标准里规定的状态码是三位数，所以取值范围就是从 000 到 999。但如果把代码简单地从 000 开始顺序编下去就显得有点太“low”，不灵活、不利于扩展，所以状态码也被设计成有一定的格式。

RFC 标准把状态码分成了五类，用数字的第一位表示分类，而 0~99 不用，这样状态码的实际可用范围就大大缩小了，由 000~999 变成了 100~599。

这五类的具体含义是：

- 1xx：提示信息，表示目前是协议处理的中间状态，还需要后续的操作；
- 2xx：成功，报文已经收到并被正确处理；
- 3xx：重定向，资源位置发生变动，需要客户端重新发送请求；
- 4xx：客户端错误，请求报文有误，服务器无法处理；
- 5xx：服务器错误，服务器在处理请求时内部发生了错误。

在 HTTP 协议中，正确地理解并应用这些状态码不是客户端或服务器单方的责任，而是双方共同的责任。

领资料



客户端作为请求的发起方，获取响应报文后，需要通过状态码知道请求是否被正确处理，是否需要再次发送请求，如果出错了原因又是什么。这样才能进行下一步的动作，要么发送新请求，要么改正错误重发请求。

服务器端作为请求的接收方，也应该很好地运用状态码。在处理请求时，选择最恰当的状态码回复客户端，告知客户端处理的结果，指示客户端下一步应该如何行动。特别是在出错的时候，尽量不要简单地返 400、500 这样意思含糊不清的状态码。

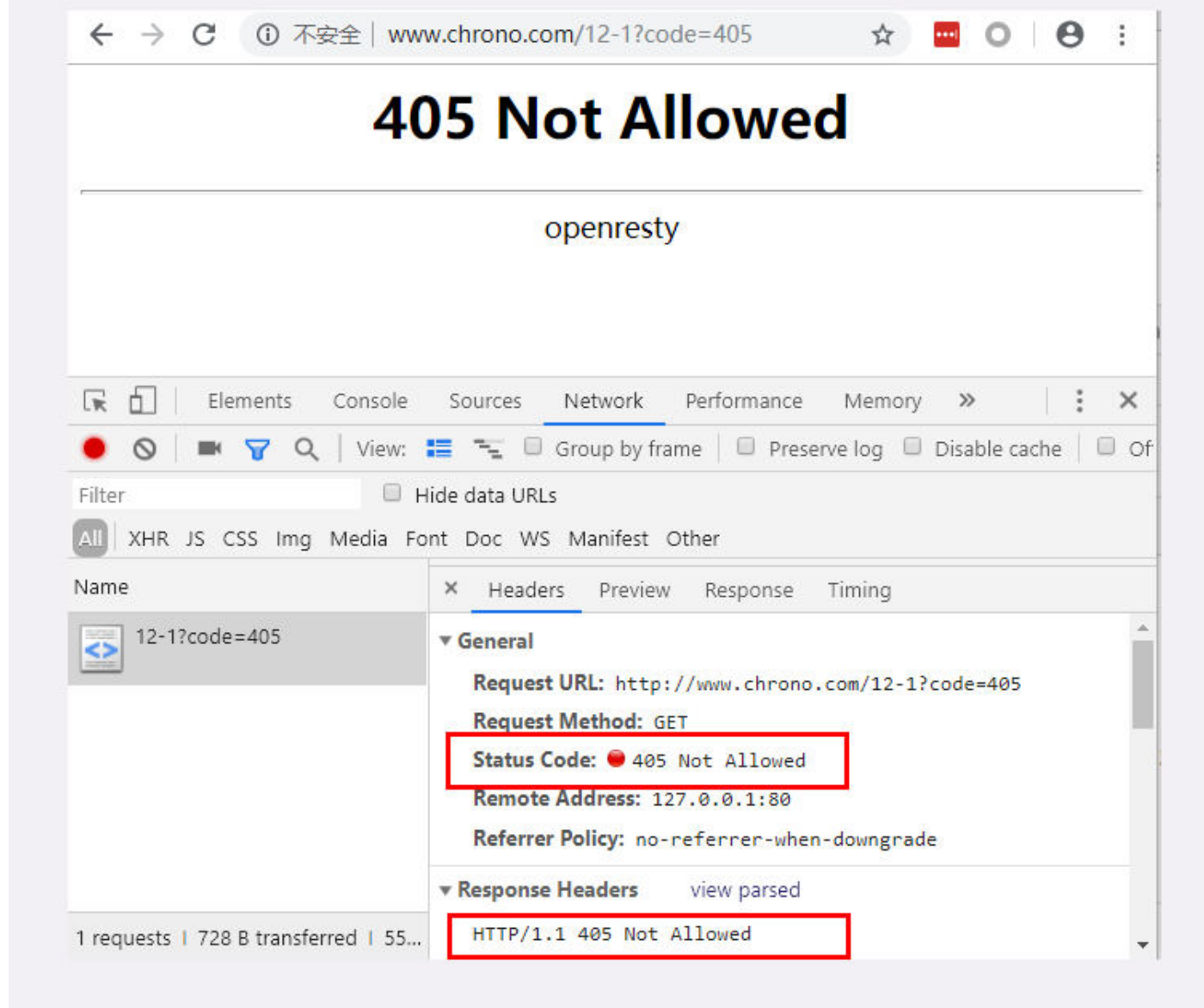
目前 RFC 标准里总共有 41 个状态码，但状态码的定义是开放的，允许自行扩展。所以 Apache、Nginx 等 Web 服务器都定义了一些专有的状态码。如果你自己开发 Web 应用，也完全可以在不冲突的前提下定义新的代码。

在我们的实验环境里也可以对这些状态码做测试验证，访问 URI“/12-1”，用查询参数“code=xxx”来检查这些状态码的效果，服务器不仅会在状态行里显示状态码，还会在响应头里用自定义的“Expect-Code”字段输出这个代码。

例如，在 Chrome 里访问“<http://www.chrono.com/12-1?code=405>”的结果如下图。

领资料





接下来我就挑一些实际开发中比较有价值的状态码逐个详细介绍。

## 1xx

1xx类状态码属于提示信息，是协议处理的中间状态，实际能够用到的时候很少。

我们偶尔能够见到的是“**101 Switching Protocols**”。它的意思是客户端使用 Upgrade 头字段，要求在 HTTP 协议的基础上改成其他的协议继续通信，比如 WebSocket。而如果服务器也同意变更协议，就会发送状态码 101，但这之后的数据传输就不会再使用 HTTP 了。

## 2xx

2xx类状态码表示服务器收到并成功处理了客户端的请求，这也是客户端最愿意看到的状态码。

领资料



“**200 OK**”是最常见的成功状态码，表示一切正常，服务器如客户端所期望的那样返回了处理结果，如果是非 HEAD 请求，通常在响应头后都会有 body 数据。

“**204 No Content**”是另一个很常见的成功状态码，它的含义与“200 OK”基本相同，但响应头后没有 body 数据。所以对于 Web 服务器来说，正确地区分 200 和 204 是很必要的。

“**206 Partial Content**”是 HTTP 分块下载或断点续传的基础，在客户端发送“范围请求”、要求获取资源的部分数据时出现，它与 200 一样，也是服务器成功处理了请求，但 body 里的数据不是资源的全部，而是其中的一部分。

状态码 206 通常还会伴随着头字段“**Content-Range**”，表示响应报文里 body 数据的具体范围，供客户端确认，例如“Content-Range: bytes 0-99/2000”，意思是此次获取的是总计 2000 个字节的前 100 个字节。

### 3xx

3xx 类状态码表示客户端请求的资源发生了变动，客户端必须用新的 URI 重新发送请求获取资源，也就是通常所说的“重定向”，包括著名的 301、302 跳转。

“**301 Moved Permanently**”俗称“永久重定向”，含义是此次请求的资源已经不存在了，需要改用新的 URI 再次访问。

与它类似的是“**302 Found**”，曾经的描述短语是“**Moved Temporarily**”，俗称“临时重定向”，意思是请求的资源还在，但需要暂时用另一个 URI 来访问。

301 和 302 都会在响应头里使用字段 **Location** 指明后续要跳转的 URI，最终的效果很相似，浏览器都会重定向到新的 URI。两者的根本区别在于语义，一个是“永久”，一个是“临时”，所以在场景、用法上差距很大。

比如，你的网站升级到了 HTTPS，原来的 HTTP 不打算用了，这就是“永久”的，所以要配置 301 跳转，把所有的 HTTP 流量都切换到 HTTPS。

再比如，今天夜里网站后台要系统维护，服务暂时不可用，这就属于“临时”的，可以配置成 302 跳转，把流量临时切换到一个静态通知页面，浏览器看到这个 302 就知道这只是暂时的情况，不会做缓存优化，第二天还会访问原来的地址。



“**304 Not Modified**”是一个比较有意思的状态码，它用于 If-Modified-Since 等条件请求，表示资源未修改，用于缓存控制。它不具有通常的跳转含义，但可以理解成“重定向已到缓存的文件”（即“缓存重定向”）。

301、302 和 304 分别涉及了 HTTP 协议里重要的“重定向跳转”和“缓存控制”，在之后的课程中我还会细讲。

## 4xx

4xx类状态码表示客户端发送的请求报文有误，服务器无法处理，它就是真正的“错误码”含义了。

“**400 Bad Request**”是一个通用的错误码，表示请求报文有错误，但具体是数据格式错误、缺少请求头还是 URI 超长它没有明确说，只是一个笼统的错误，客户端看到 400 只会是“一头雾水”“不知所措”。所以，在开发 Web 应用时应当尽量避免给客户端返回 400，而是要用其他更有明确含义的状态码。

“**403 Forbidden**”实际上不是客户端的请求出错，而是表示服务器禁止访问资源。原因可能多种多样，例如信息敏感、法律禁止等，如果服务器友好一点，可以在 body 里详细说明拒绝请求的原因，不过现实中通常都是直接给一个“闭门羹”。

“**404 Not Found**”可能是我们最常看见也是最不愿意看到的一个状态码，它的原意是资源在本服务器上未找到，所以无法提供给客户端。但现在已经被“用滥了”，只要服务器“不高兴”就可以给出个 404，而我们也无从得知后面到底是真的未找到，还是有什么别的原因，某种程度上它比 403 还要令人讨厌。

4xx里剩下的一些代码较明确地说明了错误的原因，都很好理解，开发中常用的有：

- 405 Method Not Allowed：不允许使用某些方法操作资源，例如不允许 POST 只能 GET；
- 406 Not Acceptable：资源无法满足客户端请求的条件，例如请求中文但只有英文；
- 408 Request Timeout：请求超时，服务器等待了过长的时间；
- 409 Conflict：多个请求发生了冲突，可以理解为多线程并发时的竞态；
- 413 Request Entity Too Large：请求报文里的 body 太大；

领资料





- 414 Request-URI Too Long: 请求行里的 URI 太大;
- 429 Too Many Requests: 客户端发送了太多的请求, 通常是由于服务器的限连策略;
- 431 Request Header Fields Too Large: 请求头某个字段或总体太大;

## 5xx

5xx类状态码表示客户端请求报文正确, 但服务器在处理时内部发生了错误, 无法返回应有的响应数据, 是服务器端的“错误码”。

“**500 Internal Server Error**”与 400 类似, 也是一个通用的错误码, 服务器究竟发生了什么错误我们是不知道的。不过对于服务器来说这应该算是好事, 通常不应该把服务器内部的详细信息, 例如出错的函数调用栈告诉外界。虽然不利于调试, 但能够防止黑客的窥探或者分析。

“**501 Not Implemented**”表示客户端请求的功能还不支持, 这个错误码比 500 要“温和”一些, 和“即将开业, 敬请期待”的意思差不多, 不过具体什么时候“开业”就不好说了。

“**502 Bad Gateway**”通常是服务器作为网关或者代理时返回的错误码, 表示服务器自身工作正常, 访问后端服务器时发生了错误, 但具体的错误原因也是不知道的。

“**503 Service Unavailable**”表示服务器当前很忙, 暂时无法响应服务, 我们上网时有时候遇到的“网络服务正忙, 请稍后重试”的提示信息就是状态码 503。

503 是一个“临时”的状态, 很可能过几秒钟后服务器就不那么忙了, 可以继续提供服务, 所以 503 响应报文里通常还会有一个“**Retry-After**”字段, 指示客户端可以在多久以后再次尝试发送请求。

## 小结

1. 状态码在响应报文里表示了服务器对请求的处理结果;
2. 状态码后的原因短语是简单的文字描述, 可以自定义;
3. 状态码是十进制的三位数, 分为五类, 从 100 到 599;
4. 2xx类状态码表示成功, 常用的有 200、204、206;
5. 3xx类状态码表示重定向, 常用的有 301、302、304;

领资料



6. 4xx类状态码表示客户端错误，常用的有 400、403、404；
7. 5xx类状态码表示服务器错误，常用的有 500、501、502、503。

## 课下作业

1. 你在开发 HTTP 客户端，收到了一个非标准的状态码，比如 4xx、5xx，应当如何应对呢？
2. 你在开发 HTTP 服务器，处理请求时发现报文里缺了一个必需的 query 参数，应该如何告知客户端错误原因呢？

欢迎你把自己的答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，欢迎你把文章分享给你的朋友。

领资料







## == 课外小贴士 ==

- 01 301 和 302 还另有两个等价的状态码 “308 Permanent Redirect” 和 “307 Temporary Redirect”，但这两个状态码不允许后续的请求更改请求方法。
  
- 02 愚人节玩笑协议 HTCPCP 里也定义了一个特殊的错误码：“418 I'm a teapot”，表示服务器拒绝冲咖啡，因为“我是茶壶”。



# 透视 HTTP 协议

深入理解 HTTP 协议本质与应用

罗剑锋

奇虎360技术专家


Nginx/OpenResty 开源项目贡献者



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

分享给需要的人，Ta订阅超级会员，你将得 **50** 元

Ta单独购买本课程，你将得 **20** 元

 生成海报并分享

 赞 14     提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 11 | 你能写出正确的网址吗？

下一篇 13 | HTTP有哪些特点？

领资料



# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



## 精选留言 (47)

写留言



肥low

2019-06-25

1、4xx一般是服务端业务状态码，看业务上怎么解决的，5xx就比较多，500、502比如是在Inmp架构下，500是php代码写的很烂，导致nginx的upstream接受错误，抛异常了。502就是php-fpm挂了。504说明有慢查询php-fpm可能还在运行着，可是ngx由于本身的超时设置已经主动断开了，entity too large就是上传发文件超过ngx本身设置，其它的好像还没遇到过，404大多是浏览器请求facon.ico导致的，现在前后端分离根本没这个东西

2、参数错误一般是通过接口返回具体的业务代码来表明，http响应报文一般都是200，也有400的，但是我的经验中一般是返回的body为json格式，然后里面通过一个errno来标识一下，具体怎么规定我觉得在接口文档中写清楚就好

作者回复: 写的很好。

共 8 条评论 >

30

领资料



ly

2019-08-11

个人对状态码使用的理解：

就目前项目开发的过程中，状态码的使用对业务程序来说，基本上很少去利用，大致都是nginx、tomcat、spring等这些框架、中间件会返回一些状态码，例如接口不存在，框架会返回40



4, nginx配置有问题或者拒绝访问返回403或者502等状态码, 一旦正确进入业务接口的时候, 不管请求的用户id是否在用户库中存在、以及请求参数中是否缺少必要参数, 都是返回200状态码, 只是在body的json里面会返回自己特有的业务错误码, 例如: {"code":-1,"message":"userid参数必须填写"}。

整体来讲, 开发业务的程序员很少会去思考该返回什么http状态码, 不知道这样的开发过程是否标准?

作者回复: 这也是web服务里的一种常见做法。

因为http状态码很有限, 而且含义不是很清楚, 所以通常需要在body里返回额外的数据来说明更详细的错误原因, 常用的就是json了。

共 2 条评论 >

👍 10



壹笙👉漂泊

2019-06-25

课后题:

- 1、给个错误页面、或者弹窗报个错误, 然后做跳转
- 2、在返回body里面写明错误原因, 状态码一般给500

总结:

1、状态行结构:

version 【SP】 status code 【SP】 reason 【CRLF】

version: 是HTTP协议的版本号, 用处不大

reason: 是原因短语, 简短的文字描述, 如果“OK”“Not Found”等等, 也可以自定义, 用处也不大, 很多客户端会忽略, 为了兼容早期的文本客户端

status code: 状态码, 不是错误码, 表达状态。以代码的形式表示服务器对请求的处理结果。

2、状态码

RFC标准规定状态码是三位数, 把状态码分成了五类, 用数字的第一位表示分类。

- \* 1xx:提示信息, 表示目前是协议处理的中间状态, 还需要后续的操作;
- \* 2xx: 成功, 报文已经收到并被正确处理;
- \* 3xx:重定向, 资源位置发生变动, 需要客户端重新发送请求;
- \* 4xx:客户端错误, 请求报文有误, 服务器无法处理;
- \* 5xx:服务器错误, 服务器在处理请求时内部发生了错误。

作者回复: 总结的不错。

共 2 条评论 >

👍 6

领资料



瑞

2019-06-24

第一个问题: 目前客户端基本都是解析成失败的情况, 大部分给个失败错误友好界面

第二个问题: 目前服务器很多也是返回4xx的错误码, 感觉被烂用了, 应该要返回5xx, 然后我们还会定义一个msg字段, 用来说明具体失败的原因

作者回复: √

共 2 条评论 >

👍 5



许童童

2019-06-24

1: 收到了一个非标准的状态码, 一般约定返回message字段, 然后弹窗提示

2: 用406状态码



👍 5



马哲富

2019-06-24

希望老师在后续的过程中讲一下206这个断点续传, 是不是类似于百度网盘那种下载软件暂停后再继续下载也是类似的原理?

作者回复: 后面会专门讲的。



👍 3



彧豪

2019-06-24

1.一般接口报错的话返回的body中都会封装有错误消息, 显示消息即可, 401一般都不作处理, 比如获取登录用户的信息的接口, 未登录的时候调是401, 是因为没有带相应token过去, 登录之后就能正常返回登录用户的信息了, 自己处理的情况比如SPA, 用户访问了一个不存在的路由, 此时前端自己返回一个404的页面, 里面是一张图片: “么么哒, 网页找不到了耶”之类的😂😂, 5xx这个时候需要找服务端沟通了, 正式上线一般不会出现5xx的错误, 一般在开发时出现

2.返回的body中告知前端错误信息

另外就是老师貌似忘了讲401了, 401也挺常见/用的, 默认原因短语是"Unauthorized", 比如调登录接口没带授权的时候

作者回复: 401在rfc7235中, 我用的比较少, 所以就不说了。

感谢你的补充。

领资料





Leolee

2021-04-20

作业：

- 1、统一跳转到一个报错页面
- 2、自定义一个4XX代码，告知客户端缺少了一个query。

状态码

目前的RFC标准里规定状态码是三位数，取值范围从100~599，用数字第一位表示分类，共五类，这五类具体含义是：

- 1XX：提示信息，表示目前是协议处理的中间状态，还需要后续的操作；
- 2XX：成功，报文已经收到并被正确处理；
- 3XX：重定向，资源位置发生变动，需要客户端重新发送请求；
- 4XX：客户端错误，请求报文有误，服务器无法处理；
- 5XX：服务器错误，服务器在处理请求时内部发生了错误。

1xx

此类属于提示信息，是协议处理的中间状态，实际能够用到的时候很少。

101 Switching Protocols 意思是客户端使用Upgrade头字段，要求在HTTP协议基础上改成其他协议继续通信，比如WebSocket，如果服务器统一变更协议，就会返回101，后续数据传输就不会再使用HTTP了。

2xx

此类表示服务器收到并成功处理了客户端的请求，客户端最喜欢的状态码。

200 ok 浏览器最喜欢的成功了；

202 Accepted 浏览器收到请求，但暂缓处理，暂时无法给出处理结果；

204 No Content 含义与200 OK基本相同，但响应头后没有body数据；

206 Particl Content 意思是服务器成功处理了请求，但body里的数据不是资源的全部，而是一部分，这个是HTTP分块下载或断点续传的基础，在客户端放松了“范围请求”、要求获取资源的部分数据时出现。一般206后会跟着头字段“Content-Range”，表示body里数据的具体范围，例如“Content-Range: bytes 0-99/2000”。

3xx

此类表示客户端请求的资源发生了改动，客户端必须用新的URI创新发送请求获取资源，也就是通常所说的“重定向”，包括注明的301、302跳转。

301 Moved Permanently 俗称“永久重定向”，含义是此次请求的资源已经不存在了，需要用新的URI再次访问。

302 Found 俗称“临时重定向”，意思是请求的资源还在，但需要暂时用了一个URI来访问。

304 Not Modified 即“缓存重定向”，用于If-Modified-Since等条件请求，表示资源未修改，用于缓存控制。它不具有通常的跳转含义。

4xx

此类表示客户端发送的请求报文有无，服务器无法处理，含义就是“错误码”。





400 Bad Request 含义就是数据格式有误，具体哪里错误没有明说，会让客户端一头雾水，开发WEB应用时应该尽量避免；

403 Forbidden 这个不是客户的请求出错，而是服务器禁止访问资源。愿意多种多样，可能是信息敏感、法律禁止等；

404 Not Found 被滥用的状态码，愿意是请求的资源在服务器上找不到，但很多服务器动不动就给你来个404；

405 Method Not Allowed 不允许使用某些方法操作资源，例如不允许POST只允许GET；

406 Not Acceptable 资源无法满足客户端请求的条件，例如请求中文但只有英文；

408 Request Timeout 请求超时，服务器等待了过长的时间；

409 Conflict 多个请求发生了冲突，可以理解为多线程并发时的竞争状态；

413 Request Entity Too Large 请求报文里的body太大；

414 Request-URI Too Long 请求行里的URI太长；

429 Too Many Requests 客户端发送了太多请求，通常是由于服务器的限制连接策略；

431 Request Header Fields Too Large 请求头某个字段或总体太大。

5xx

此类表示客户端请求报文正确，但服务器在处理时内部发生了错误，无法返回应有的响应数据，是服务器端的“错误码”。

500 Internal Server Error 相当于服务器端的400,属于通用错误码，不利于调试，但能够防止黑客窥探或分析；

501 Not Implemented 表示客户端请求的功能还不支持；

502 Bad Gateway 通常是服务器作为王冠或者代理是返回的错误码，表示服务器自身工作正常，访问后端服务器时发生错误，具体错误原因未知；

503 Service Unavailable 服务器繁忙，当前不可用，这是一个临时的状态，通常503的响应报文里会有一个“Retry-After”字段，意思是多久后再重试。

作者回复: 学习态度非常认真，课下问题可以再参考其他同学的回答。

共 2 条评论 >

👍 1



Geek\_Maggie

2021-03-15

### 【课后思考题】

1. 你在开发 HTTP 客户端，收到了一个非标准的状态码，比如 4xx、5xx，应当如何应对呢？

我个人感觉4XX和5XX还是要区分开提示。4XX主要是客户端错误，5XX主要是服务端处理错误。如果是我开发客户端遇到这个问题，我会和业务人员沟通要如何进行客户提示，如：4XX一律提示为“请求有误，请稍后再试” 5XX一律提示为“服务器繁忙，请稍后重试”；

2. 你在开发 HTTP 服务器，处理请求时发现报文里缺了一个必需的 query 参数，应该如何告知客户端错误原因呢？

领资料





返回4xx错误，并在返回json报文中定义详细错误代号，如：{code: "-1"; msg: QUERY\_ERROR; .....}

作者回复: 说的挺好，4xx和5xx目前的错误还是比较少，不能准确描述所有的错误，所以最好再添加其他的信息，方便排错。



1



哈德韦

2020-10-07

419 和 503 的区别是什么？都是服务器繁忙而且只是暂时的

作者回复: 应该是429吧。

两者的原因不同，429是客户端请求太多，处理不过来，而503是服务器内部的问题，不一定是客户端请求太多。



1



钱

2020-03-28

这节很有借鉴意义，尤其对于开发RPC接口而言。之前的项目组针对web的服务会回传错误原因，这个越清楚越好，这样操作者就知道怎么调整了，也都是内部系统，内部使用。

如果是RPC接口基本模式是结果码、结果描述、调用链码、响应信息、是否建议重试等，我个人感觉这种方式比较好，只是因业务和系统不同，状态码的归类和业务含义不好提前定义全，还有会用0表示OK，<0表示各种异常，>0特殊情况下的正常，比如：降级处理。

作者回复: good。



1



Wr

2020-01-04

1. 目前遇到的情况都是跳转一个空白页面，页面上方标注错误码和错误信息
2. 返回一个4\*\*类型的错误码吧，返回错误信息为请求错误之类的（这几天处理开发写的自动化脚本，由于协议变更，请求出现很多问题，比如请求里的json字段不符合新的协议规范，返回的错误一般都是403，应该和这个问题类型，但是我看很多同学评论的都是5\*\*错误码，这是问什么呢？）

作者回复:

1. 这样也可以，但比较简单，对用户不太友好。

领资料



2.我认为返回4xx错误比较好，说明这是客户端的错误，但因为http很灵活，只要双方约定好，5xx也可以。



1



风翱

2019-06-24

对于问题二，如果缺少的参数，服务端采用友好的方式提示。调试方面确实方便了，但是对于安全性呢？

作者回复：这里应该不涉及安全的问题，如果比较在意，可以约定用一些代码来表示。

共 2 条评论 >

1



我行我素

2019-06-24

1.4\*\*一般都是弹窗提示，将错误显示出来，5\*\*有单独的页面显示错误信息；  
2.在自己的项目中都是返回json直接明确告诉缺少什么参数

作者回复：√



1



无名

2019-06-24

问题一：4xx，5xx，跳转到单独页面提示。

问题二：我们一般交互的都说json格式，会内部再定义一个code和msg，然后定义一个参数错误code，在msg中提示错误具体原因。

作者回复：不错。



1



刘敏

2022-01-03

1、你在开发 HTTP 客户端，收到了一个非标准的状态码，比如 4xx、5xx，应当如何应对呢？

4xx与5xx代表的错误原因不一致，4xx代表客户端错误，5xx代表服务器在处理过程中遇到了错误。

对于两者，处理方式都是尽量把错误原因通过body中明确出来方便调试。

领资料



2、你在开发 HTTP 服务器，处理请求时发现报文里缺了一个必需的 query 参数，应该如何告知客户端错误原因呢？

首先参数错误应该属于客户端参数传值的原因导致的，会思考采用400错误码并在响应中给出错误原因描述。

作者回复: great，可以再参考其他同学的回答。



HILLIEX

2021-11-05

Q1: 你在开发 HTTP 客户端，收到了一个非标准的状态码，比如 4xx、5xx，应当如何应对呢？

A1: 首先我会按照4、5先定义一个default的错误提示信息，4开头前端错了，就未知错误，5开头服务器错的，就服务器错误后面再试。

Q2: 你在开发 HTTP 服务器，处理请求时发现报文里缺了一个必需的 query 参数，应该如何告知客户端错误原因呢？

A2: 首先，报文错误，肯定是客户端的错误，所以是4开头的，具体错误信息，在body里面带上就好了吧，前端处理。

评论区收获：

1、约定body带json返回例如 {"code":-1,"message:"userid参数必须填写"}。

2、错误也可以是返回200，在body返回

作者回复: great。

不过200再返回错误body有点不太妥当。



领资料



程序猿石头

2021-08-14

401 其实也挺常见的 🤔

作者回复: 401 Unauthorized，我这里用的比较少。





Aibo

2021-07-04

对于问题二

我们项目用的grpc，对于客户端的参数错误会返回

code.invalidArgument，也就对应http code的400，在body里会携带具体的errorcode，message，标识错误。

作者回复: grpc是在http/2之上了，所以就允许不用http那套规矩，而是自己定义，其实和restful的json差不多。



zhangdroid

2021-04-07

通常在前端或移动端开发中对于诸如参数错误、缺失这类的错误请求处理方式是：请求成功，返回状态码200 Ok，响应体body中返回具体的错误信息。一般做法是定义接口json文档，其中一个字段表示请求结果如result，表示请求是否成功，若错误会有一个相应的错误提示字段errorMsg，最后还是一个字段是真正的业务上的body信息，一般是data字段，也是一个json。

作者回复: 对，这是目前最普遍的做法。因为http的状态码和描述短语现在实在是够用，只能在body上扩展了。



领资料

