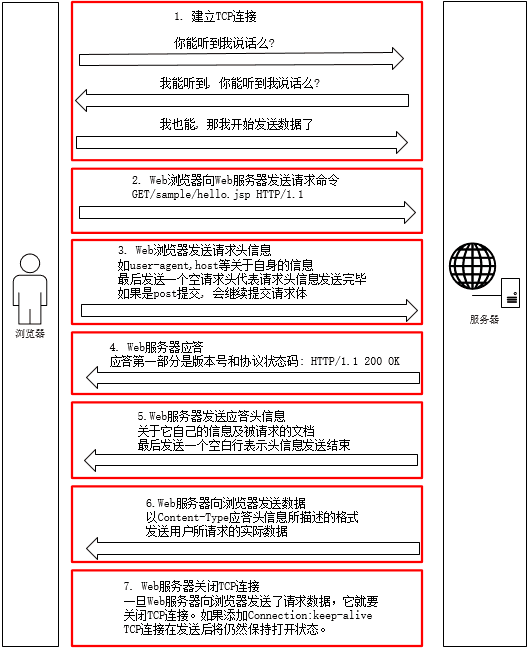
**HTTP请求和响应：**



**100 Continue**

HTTP **100 Continue** 信息型状态响应码表示目前为止一切正常, 客户端应该继续请求, 如果已完成请求则忽略.

为了让服务器检查请求的首部, 客户端必须在发送请求实体前, 在初始化请求中发送 Expect: 100-continue 首部并接收 100 Continue 响应状态码.

**101 Switching Protocol**

HTTP  **101 Switching Protocol**（协议切换）状态码表示服务器应客户端升级协议的请求（[Upgrade](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Upgrade)请求头）正在进行协议切换。

服务器会发送一个[Upgrade](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Upgrade)响应头来表示其正在切换过去的协议。  
该过程在协议升级机制([Protocol upgrade mechanism](https://developer.mozilla.org/en-US/docs/Web/HTTP/Protocol_upgrade_mechanism))中详细描述 .

**（Upgrade** 请求服务端升级协议

Upgrade: HTTP/2.0, HTTPS/1.3, IRC/6.9, RTA/x11, websocket）

**200 OK**

状态码 **200 OK** 表明请求已经成功. 默认情况下状态码为200的响应可以被缓存。

不同请求方式对于请求成功的意义如下:

* [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET): 已经取得资源，并将资源添加到响应的消息体中。
* [HEAD](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/HEAD): 响应的消息体为头部信息。
* [POST](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/POST): 响应的消息体中包含此次请求的结果。
* [TRACE](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/TRACE): 响应的消息体中包含服务器接收到的请求信息。

**201 Created**

在HTTP协议中，**201 Created** 是一个代表成功的应答状态码，表示请求已经被成功处理，并且创建了新的资源。新的资源在应答返回之前已经被创建。同时新增的资源会在应答消息体中返回，其地址或者是原始请求的路径，或者是 [Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 首部的值。

这个状态码的常规使用场景是作为 [POST](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/POST) 请求的返回值。

**202 Accepted**

响应状态码**202 Accepted** 表示服务器端已经收到请求消息，但是尚未进行处理。这个状态码被设计用来将请求交由另外一个进程或者服务器来进行处理，或者是对请求进行批处理的情形。（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。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](http://baike.baidu.com/view/80110.htm)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](http://baike.baidu.com/view/159417.htm)，以便用户能够估计操作是否已经完成。

）

**203 Non-Authoritative Information(未授权信息)**

在 HTTP 协议中，响应状态码 **203 Non-Authoritative Information**  表示请求已经成功被响应，但是获得的负载与源头服务器的状态码为 [200](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/200) (OK)的响应相比，经过了拥有转换功能的 [proxy](https://developer.mozilla.org/en-US/docs/Glossary/Proxy_server) （代理服务器）的修改。

The 203 状态码有点类似于 [Warning](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Warning) 首部的 [214](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Warning#Warning_codes)（Transformation Applied）警告码，后者的优势在于可以应用于任何状态码的响应之中。

**204 No Content**

HTTP协议中 **204 No Content**成功状态响应码表示目前请求成功，但客户端不需要更新其现有页面。204 响应默认是可以被缓存的。在响应中需要包含头信息 [ETag](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/ETag)。

使用惯例是，在 [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 请求中进行资源更新，但是不需要改变当前展示给用户的页面，那么返回 204 No Content。如果新创建了资源，那么返回 [201](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/201)Created 。如果页面需要更新以反映更新后的资源，那么需要返回 [200](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/200) 。(返回的Response中只有一些Header和一个状态行， 没有实体的主题内容（没有response body）

204状态码的作用在于： 1. 在不获取资源的情况下了解资源的情况（比如判断其类型）

2. 通过查看Response中的状态码， 看看某个对象是否存在

3. 通过查看Header, 测试资源是否被修改了。)

**205 Reset Content**

(在 HTTP 协议中，响应状态码 **205 Reset Content** 用来通知客户端重置文档视图，比如清空表单内容、重置 canvas 状态或者刷新用户界面。)

**206 Partial Content**

HTTP **206 Partial Content** 成功状态响应代码表示请求已成功，并且主体包含所请求的数据区间，该数据区间是在请求的 [Range](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Range) 首部指定的。

如果只包含一个数据区间，那么整个响应的 [Content-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type) 首部的值为所请求的文件的类型，同时包含  [Content-Range](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Range) 首部。

如果包含多个数据区间，那么整个响应的  [Content-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type)  首部的值为 multipart/byteranges ，其中一个片段对应一个数据区间，并提供  [Content-Range](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Range) 和 [Content-Type](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type)  描述信息。（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

(206状态码代表服务器已经成功处理了部分GET请求（只有发送GET 方法的request, web服务器才可能返回206），应用场景：

1. FlashGet, 迅雷或者HTTP下载工具都是使用206状态码来实现断点续传

2. 将一个大文档分解为多个下载段同时下载 比如，在线看视频

 实例：　一些流媒体技术比如在线视频，可以边看边下载。　就是使用206来实现的。)

**300 Multiple Choices**

**300 Multiple Choices** 是一个用来表示重定向的响应状态码，表示该请求拥有多种可能的响应。用户代理或者用户自身应该从中选择一个。由于没有如何进行选择的标准方法，这个状态码极少使用。

假如服务器可以提供一个优先选择，那么它应该生成一个  [Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 首部。

**301 Moved Permanently**

HTTP **301 永久重定向** 说明请求的资源已经被移动到了由 [Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 头部指定的url上，是固定的不会再改变。搜索引擎会根据该响应修正。

尽管标准要求浏览器在收到该响应并进行重定向时不应该修改http method和body，但是有一些浏览器可能会有问题。所以最好是在应对[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) 方法时使用301，其他情况使用[308](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/308) 来替代301。（(请求的URL已移走。Response中应该包含一个Location URL, 说明资源现在所处的位置

例如：  1.  浏览器客户端访问 http://map.google.cn

2. Web服务器返回Response 301，Location=http://titu.google.cn  （告诉客户端我们的资源位于这里， ）

3. 浏览器客户端会自动再发送一个Request 去访问http://titu.google.cn

**302 Found**

HTTP **302 Found** 重定向状态码表明请求的资源被暂时的移动到了由[Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 头部指定的 URL 上。浏览器会重定向到这个URL， 但是搜索引擎不会对该资源的链接进行更新 (In SEO-speak, it is said that the link-juice is not sent to the new URL)。

即使规范要求浏览器在重定向时保证请求方法和请求主体不变，但并不是所有的用户代理都会遵循这一点，你依然可以看到有缺陷的软件的存在。所以推荐仅在响应 [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) 方法时采用 302 状态码，而在其他时候使用 [307](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/307) Temporary Redirect 来替代，因为在这些场景下方法变换是明确禁止的。

在确实需要将重定向请求的方法转换为 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET)的场景下，可以使用 [303](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/303) See Other。例如在使用 [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 方法进行文件上传操作时，需要返回确认信息（例如“你已经成功上传了xyz”）而不是上传的资源本身，就可以使用这个状态码。

**303 See Other**

HTTP **303 See Other** 重定向状态码，通常作为 [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 或 [POST](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/POST) 操作的返回结果，它表示重定向链接指向的不是新上传的资源，而是另外一个页面，比如消息确认页面或上传进度页面。而请求重定向页面的方法要总是使用 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET)。

**304 Not Modified**

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)。

如果是 [200](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/200) OK ，响应会带有头部 [Cache-Control](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Cache-Control), [Content-Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Location), [Date](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Date), [ETag](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/ETag), [Expires](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Expires)，和 [Vary](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Vary).

**307 Temporary Redirect**

在 HTTP 协议中， 307 Temporary Redirect（临时重定向）是表示重定向的响应状态码，说明请求的资源暂时地被移动到  [Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 首部所指向的 URL 上。

原始请求中的请求方法和消息主体会在重定向请求中被重用。在确实需要将重定向请求的方法转换为  [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET) 的场景下，可以考虑使用 [303](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/303) See Also 状态码。例如在使用 [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 方法进行文件上传操作时，需要返回确认信息（例如“你已经成功上传了xyz”）而不是上传的资源本身，就可以使用这个状态码。

状态码 307 与 [302](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/302) 之间的唯一区别在于，当发送重定向请求的时候，307 状态码可以确保请求方法和消息主体不会发生变化。当响应状态码为 302 的时候，一些旧有的用户代理会错误地将请求方法转换为 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET)：使用非 GET 请求方法而返回 302 状态码，Web 应用的运行状况是不可预测的；而返回 307 状态码时则是可预测的。对于 GET 请求来说，两种情况没有区别。

**308 Permanent Redirect**

在 HTTP 协议中， **308 Permanent Redirect**（永久重定向）是表示重定向的响应状态码，说明请求的资源已经被永久的移动到了由 [Location](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Location) 首部指定的 URL 上。浏览器会进行重定向，同时搜索引擎也会更新其链接（用 SEO 的行话来说，意思是链接汁被传递到了新的 URL）。

在重定向过程中，请求方法和消息主体不会发生改变，然而在返回 [301](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/301) 状态码的情况下，请求方法有时候会被客户端错误地修改为 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/GET) 方法。

**400 Bad Request**

HTTP **400 Bad Request** 响应状态码表示由于语法无效，服务器无法理解该请求。 客户端不应该在未经修改的情况下重复此请求。（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

(发送的Request中的数据有错误(比如：表单有错误，Cookie有错误)，  这个我们也经常见到。)

**401 Unauthorized**

状态码 **401 Unauthorized** 代表客户端错误，指的是由于缺乏目标资源要求的身份验证凭证，发送的请求未得到满足。

这个状态码会与   [WWW-Authenticate](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/WWW-Authenticate) 首部一起发送，其中包含有如何进行验证的信息。

这个状态类似于 [403](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/403)， 但是在该情况下，依然可以进行身份验证。

**403 Forbidden**

状态码 **403 Forbidden** 代表客户端错误，指的是服务器端有能力处理该请求，但是拒绝授权访问。

这个状态类似于 [401](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/401)，但是进入该状态后，不能再继续进行验证。该访问是永久禁止的，并且与应用逻辑密切相关（例如不正确的密码）。

**404 Not Found**

状态码 **404 Not Found** 代表客户端错误，指的是服务器端无法找到所请求的资源。返回该响应的链接通常称为坏链（broken link）或死链（dead link），它们会导向链接出错处理([link rot](https://en.wikipedia.org/wiki/Link_rot))页面。

404 状态码并不能说明请求的资源是临时还是永久丢失。如果服务器知道该资源是永久丢失，那么应该返回 [410](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/410) (Gone) 而不是 404 。

**405 Method Not Allowed(不允许使用的方法)**

状态码 **405 Method Not Allowed** 表明服务器禁止了使用当前 HTTP 方法的请求。需要注意的是，[GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET) 与 [HEAD](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/HEAD) 两个方法不得被禁止，当然也不得返回状态码 405。（405是指Web服务器不支持Request中的方法。实例:  发送一个是trace方法的Request 给www.google.com）

**406 Not Acceptable**

HTTP 协议中的 **406 Not Acceptable** 状态码表示客户端错误，指代服务器端无法提供与  [Accept-Charset](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Accept-Charset) 以及 [Accept-Language](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Accept-Language) 消息头指定的值相匹配的响应。

**407 Proxy Authentication Required（要求代理身份验证）**

状态码 **407 Proxy Authentication Required**代表客户端错误，指的是由于缺乏位于浏览器与可以访问所请求资源的服务器之间的代理服务器（[proxy server](https://developer.mozilla.org/en-US/docs/Glossary/proxy_server) ）要求的身份验证凭证，发送的请求尚未得到满足。

这个状态码会与 [Proxy-Authenticate](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Proxy-Authenticate) 首部一起发送，其中包含有如何进行验证的信息。

**408 Request Timeout**

响应状态码 **408 Request Timeout** 表示服务器想要将没有在使用的连接关闭。一些服务器会在空闲连接上发送此信息，**即是在客户端没有发送任何请求的情况下**。

服务器应该在此类响应中将 [Connection](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Connection) 首部的值设置为 "close"，因为  408意味着服务器已经决定将连接关闭，而不是继续等待。

这类响应出现的比较频繁，源于一些浏览器——例如  Chrome, Firefox 27+, 或者 IE9 等——使用 HTTP 协议中的预连接机制来加速上网体验。同时应该注意到，某些服务器会直接关闭连接，而不发送此类消息。

**409 Conflict**

响应状态码 **409 Conflict** 表示请求与当前服务器端的状态相冲突。

冲突最有可能发生在对 [PUT](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/PUT) 请求的响应中。例如，当上传文件的版本比服务器上已存在的要旧，从而导致版本冲突的时候，那么就有可能收到状态码为 409 的响应。

**410 Gone**

HTTP **410 丢失** 说明请求的内容在服务器上不存在了，同时是永久性的丢失。如果不清楚是否为永久或临时的丢失，应该使用[404](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/404)

**500 Internal Server Error(内部服务器错误)**

在 HTTP 协议中，**500 Internal Server Error** 是表示服务器端错误的响应状态码，意味着所请求的服务器遇到意外的情况并阻止其执行请求。

这个错误代码是一个通用的“全方位”响应代码。通常服务器管理员对于类似于 500 这样的错误会更加详细地记录相关的请求信息来防止以后同样错误的出现。（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…）

**501 Not Implemented（未实现）**

HTTP **501 Not Implemented** 服务器错误响应码表示请求的方法不被服务器支持，因此无法被处理。服务器必须支持的方法（即不会返回这个状态码的方法）只有 [GET](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/GET) 和 [HEAD](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Methods/HEAD)。请注意，你无法修复 501 错误，需要被访问的 web 服务器去修复该问题。（客户端发起的请求超出服务器的能力范围(比如，使用了服务器不支持的请求方法)时，使用此状态码）。

实例： 使用Fiddler Composer 给www.qq.com,  发送一个OPTIONS 方法的Request.   服务器就能返回501了。）

**502 Bad Gateway**

**502 Bad Gateway** 表示作为网关或代理角色的服务器，从上游服务器（如tomcat、php-fpm）中接收到的响应是无效的。

[Gateway](https://zh.wikipedia.org/wiki/Gateway_(telecommunications)) （网关）在计算机网络体系中可以指代不同的设备，502 错误通常不是客户端能够修复的，而是需要由途径的Web服务器或者代理服务器对其进行修复。（代理使用的服务器遇到了上游的无效响应。  Fiddler本身就是代理服务器。  当我们访问www.facebook.com，这网站被我们天朝屏蔽了。 所以我们可以得到502）

**503 Service Unavailable（服务不可用）**

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 Gateway Timeout**

**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。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**

**505 HTTP Version Not Supported** 表示服务器不支持请求所使用的 HTTP 版本。

（表示Web服务器不支持此HTTP协议的版本。  众所周知我们现在使用的HTTP协议版本是HTTP/1.1， 如果我们发送一个HTTP/2.0 的request 给博客园， 博客园肯定不能支持HTTP/2.0，所以会返回505 ）

**511 Network Authentication Required**

**511 Network Authentication Required** 表示客户端需要通过验证才能使用该网络。

该状态码不是由源头服务器生成的，而是由控制网络访问的拦截代理服务器生成的。

网络运营商们有时候会在准许使用网络之前要求用户进行身份验证、接受某些条款，或者进行其他形式的与用户之间的互动（例如在网络咖啡厅或者机场）。他们通常用用户设备的  [MAC](https://developer.mozilla.org/zh-CN/docs/Glossary/MAC) 地址来进行识别。

# 常用标准请求头字段

请求头：用于在请求消息中向服务器传递附加信息,主要包括客户机可以接受的数据类型,压缩方法,语言,以及客户计算机上保留的信息和发出该请求的超链接源地址等。

**主要: Accept , Accept-Encoding , Accept-Language , Host 。**

**Accept** 设置接受的内容类型，**表示浏览器支持的 MIME 类型。**

Accept: text/plain

**Accept-Charset** 设置接受的字符编码

Accept-Charset: utf-8

**Accept-Encoding表示浏览器有能力解码的编码类型。**

Accept-Encoding: gzip, deflate

**Accept-Datetime** 设置接受的版本时间

Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT

**Accept-Language** 表示浏览器所支持的语言类型。

Accept-Language: en-US

**Authorization** 设置HTTP身份验证的凭证

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

**Cache-Control** 设置请求响应链上所有的缓存机制必须遵守的指令

Cache-Control: no-cache

如果选中了no cache,则请求资源时，请求头中的Cache-Control为no-cache，表明不使用缓存，则会直接获取服务器资源。另外，若没选中no cache，Cache-Control有二种情况：

       1、max-age>0 时直接从游览器缓存中提取

       2、max-age<=0 时向服务器发送http请求,该资源是否有修改有的话返回200 ,无的话返回304.

**Connection** 设置当前连接和hop-by-hop协议请求字段列表的控制选项

Connection: keep-alive  
Connection: Upgrade

**Content-Length** 设置请求体的字节长度

Content-Length: 348

**Content-MD5** 设置基于MD5算法对请求体内容进行Base64二进制编码

Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==

**Content-Type** 设置请求体的MIME类型（适用POST和PUT请求）

Content-Type: application/x-www-form-urlencoded

**Cookie** 设置服务器使用Set-Cookie发送的http cookie

Cookie: $Version=1; Skin=new;

**Date** 设置消息发送的日期和时间

Date: Tue, 15 Nov 1994 08:12:31 GMT

**Forwarded** 披露客户端通过http代理连接web服务的源信息

Forwarded: for=192.0.2.60;proto=http;by=203.0.113.43  
Forwarded: for=192.0.2.43, for=198.51.100.17

**From** 设置发送请求的用户的email地址

From: [user@example.com](https://link.jianshu.com?t=mailto:user@example.com)

**Host** 设置服务器域名和TCP端口号，如果使用的是服务请求标准端口号，端口号可以省略

Host: en.wikipedia.org:8080  
Host: en.wikipedia.org

**If-Match** 设置客户端的ETag,当时客户端ETag和服务器生成的ETag一致才执行，适用于更新自从上次更新之后没有改变的资源

If-Match: "737060cd8c284d8af7ad3082f209582d

**If-Modified-Since** 设置更新时间，从更新时间到服务端接受请求这段时间内如果资源没有改变，允许服务端返回304 Not Modified

If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

**If-None-Match** 设置客户端ETag，如果和服务端接受请求生成的ETage相同，允许服务端返回304 Not Modified

If-None-Match: "737060cd8c284d8af7ad3082f209582d"

**If-Range** 设置客户端ETag，如果和服务端接受请求生成的ETage相同，返回缺失的实体部分；否则返回整个新的实体

If-Range: "737060cd8c284d8af7ad3082f209582d"

**If-Unmodified-Since** 设置更新时间，只有从更新时间到服务端接受请求这段时间内实体没有改变，服务端才会发送响应

If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT

**Max-Forwards** 限制代理或网关转发消息的次数

Max-Forwards: 10

**Origin** 标识跨域资源请求（请求服务端设置Access-Control-Allow-Origin响应字段）

Origin: http://www.example-social-network.com

**Pragma** 设置特殊实现字段，可能会对请求响应链有多种影响

Pragma: no-cache

**Proxy-Authorization** 为连接代理授权认证信息

Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

**Range** 请求部分实体，设置请求实体的字节数范围，具体可以参见HTTP/1.1中的Byte serving

Range: bytes=500-999

**Referer** 设置前一个页面的地址，并且前一个页面中的连接指向当前请求，意思就是如果当前请求是在A页面中发送的，那么referer就是A页面的url地址（轶事：这个单词正确的拼法应该是"referrer",但是在很多规范中都拼成了"referer"，所以这个单词也就成为标准用法）

Referer: http://en.wikipedia.org/wiki/Main\_Page

**TE** 设置用户代理期望接受的传输编码格式，和响应头中的Transfer-Encoding字段一样

TE: trailers, deflate

**Upgrade** 请求服务端升级协议

Upgrade: HTTP/2.0, HTTPS/1.3, IRC/6.9, RTA/x11, websocket

**User-Agent** 用户代理的字符串值

User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:12.0) Gecko/20100101 Firefox/21.0

**Via** 通知服务器代理请求

Via: 1.0 fred, 1.1 example.com (Apache/1.1)

**Warning** 实体可能会发生的问题的通用警告

Warning: 199 Miscellaneous warning

# 常用标准响应头字段

**响应头：**用于在响应消息中向客户端传递附加信息,包括服务程序的名称,要求客户端进行认证的方式,请求的资源已移动到新地址等。

**主要: Location , Server , WWW-Authenticate(认证头)。**

**Access-Control-Allow-Origin** 指定哪些站点可以参与跨站资源共享

Access-Control-Allow-Origin: \*

**Accept-Patch** 指定服务器支持的补丁文档格式，适用于http的patch方法

Accept-Patch: text/example;charset=utf-8

**Accept-Ranges** 服务器通过byte serving支持的部分内容范围类型

Accept-Ranges: bytes

**Age** 对象在代理缓存中暂存的秒数

Age: 12

**Allow** 设置特定资源的有效行为，适用方法不被允许的http 405错误

Allow: GET, HEAD

**Cache-Control** 告诉服务端到客户端所有的缓存机制是否可以缓存这个对象，单位是秒

Cache-Control: max-age=3600

**Connection** 设置当前连接和hop-by-hop协议请求字段列表的控制选项

Connection: close

**Content-Disposition** 告诉客户端弹出一个文件下载框，并且可以指定下载文件名

Content-Disposition: attachment; filename="fname.ext"

**Content-Encoding** 设置数据使用的编码类型

Content-Encoding: gzip

**Content-Language** 为封闭内容设置自然语言或者目标用户语言

Content-Language: en

**Content-Length** 响应体的字节长度

Content-Length: 348

**Content-Location** 设置返回数据的另一个位置

Content-Location: /index.htm

**Content-MD5** 设置基于MD5算法对响应体内容进行Base64二进制编码

Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==

**Content-Range** 标识响应体内容属于完整消息体中的那一部分

Content-Range: bytes 21010-47021/47022

**Content-Type** 设置响应体的MIME类型

Content-Type: text/html; charset=utf-8

**Date** 设置消息发送的日期和时间

Date: Tue, 15 Nov 1994 08:12:31 GMT

**ETag** 特定版本资源的标识符，通常是消息摘要

ETag: "737060cd8c284d8af7ad3082f209582d"

**Expires** 设置响应体的过期时间

Expires: Thu, 01 Dec 1994 16:00:00 GMT

**Last-Modified** 设置请求对象最后一次的修改日期

Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT

**Location** 在重定向中或者创建新资源时使用

Location: http://www.w3.org/pub/WWW/People.html

**Pragma** 设置特殊实现字段，可能会对请求响应链有多种影响

Pragma: no-cache

**Proxy-Authenticate** 设置访问代理的请求权限

Proxy-Authenticate: Basic

**Public-Key-Pins** 设置站点的授权TLS证书

Public-Key-Pins: max-age=2592000; pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";

**Refresh** "重定向或者新资源创建时使用，在页面的头部有个扩展可以实现相似的功能，并且大部分浏览器都支持  
<meta http-equiv="refresh" content="5; url=http://example.com/">

Refresh: 5; url=http://www.w3.org/pub/WWW/People.html

**Retry-After** 如果实体暂时不可用，可以设置这个值让客户端重试，可以使用时间段（单位是秒）或者HTTP时间

Example 1: Retry-After: 120  
Example 2: Retry-After: Fri, 07 Nov 2014 23:59:59 GMT

**Server** 服务器名称

Server: Apache/2.4.1 (Unix)

**Set-Cookie** 设置HTTP Cookie

Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1

**Status** 设置HTTP响应状态

Status: 200 OK

**Strict-Transport-Security** 一种HSTS策略通知HTTP客户端缓存HTTPS策略多长时间以及是否应用到子域

Strict-Transport-Security: max-age=16070400; includeSubDomains

**Trailer** 标识给定的header字段将展示在后续的chunked编码的消息中

Trailer: Max-Forwards

**Transfer-Encoding** 设置传输实体的编码格式，目前支持的格式： chunked, compress, deflate, gzip, identity

Transfer-Encoding: chunked

**Upgrade** 请求客户端升级协议

Upgrade: HTTP/2.0, HTTPS/1.3, IRC/6.9, RTA/x11, websocket

**Vary** 通知下级代理如何匹配未来的请求头已让其决定缓存的响应是否可用而不是重新从源主机请求新的

Example 1: Vary: \*  
Example 2: Vary: Accept-Language

**Via** 通知客户端代理，通过其要发送什么响应

Via: 1.0 fred, 1.1 example.com (Apache/1.1)

**Warning** 实体可能会发生的问题的通用警告

Warning: 199 Miscellaneous warning

**WWW-Authenticate** 标识访问请求实体的身份验证方案

WWW-Authenticate: Basic