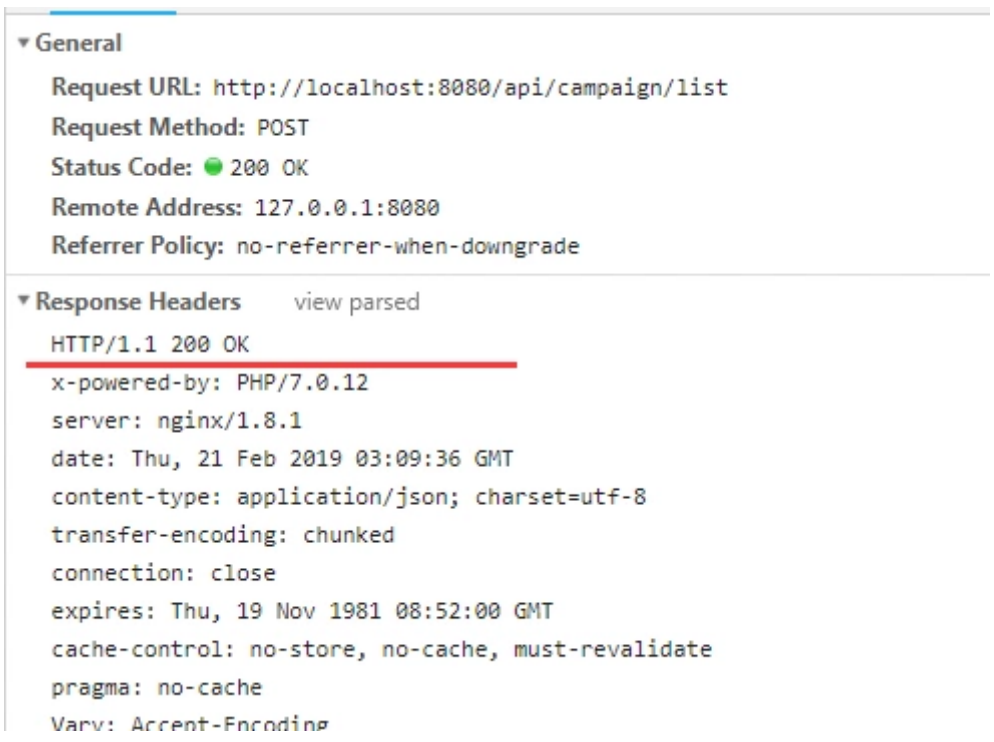
寒水寺一禅 1.7k

一文读懂Http Headers为何物(超详细)

http1.1 http缓存 http首部 http 发布于 2019-02-21

一、HTTP 请求内容

由于最新的http2，并没有被各大浏览器广泛使用，所以本文是基于http/1.1所编写的。
同时经过检测我们也发现，chrome等浏览器也正是使用http/1.1版本的。

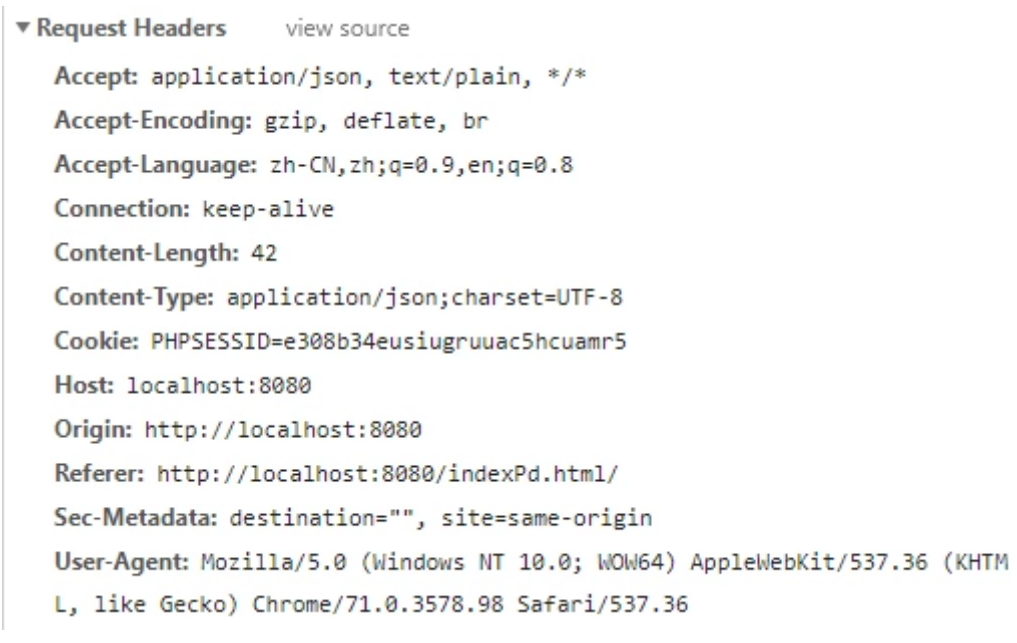


关于http/1.1协议的详情，可查看官方文档

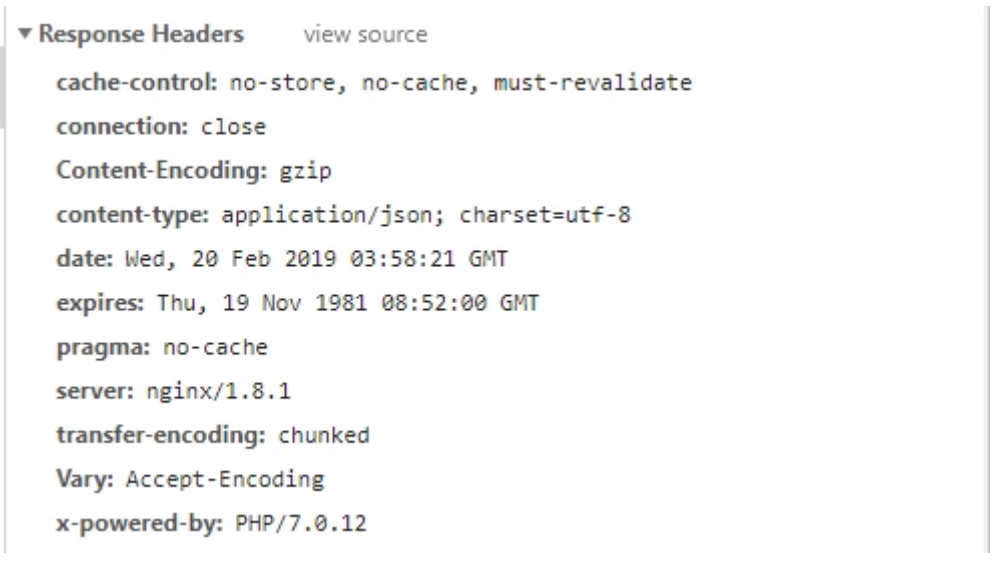
我们打开chrome的network，点击任何一条request请求，即可发现，每个http headers都包含以下部分:Genaral，Request Headers，Response Headers，Request Payload。

General(不属于headers，只用于收集请求url和响应的status等信息)

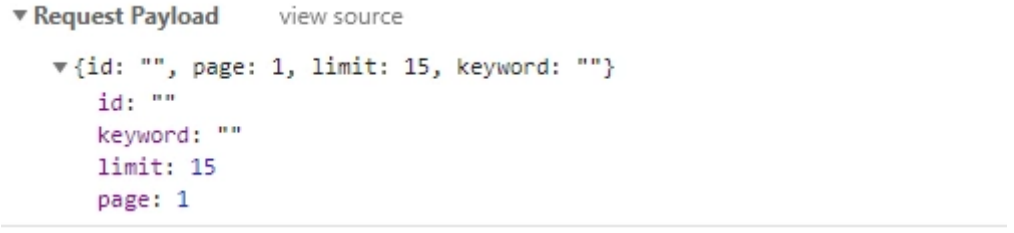
Request Headers(请求headers)



Response Headers(响应headers)



Request Payload(请求参数)



二、HTTP Headers分类

在http heanders中，为了方便，分为以下几类：Genaral headers(和上面说的General不同，这个只是为了方便统计)，Request Headers，Response Headers，Entity Headers(也是为了方便统计)。

所以，一个完整的请求头/响应头，应该除了自身，还包括 General Headers和Entity Headers。

1、Genaral headers: 同时适用于请求和响应消息，但与最终消息传输的数据无关的消息头。
2、Request Headers: 包含更多有关要获取的资源或客户端本身信息的消息头。
3、Response Headers：包含有关响应的补充信息，如其位置或服务器本身（名称和版本等）的消息头。
4、Entity Headers：包含有关实体主体的更多信息，比如主体长(Content-Length)度或其MIME类型。

1、Genaral headers

Cache-Control——控制缓存的行为； 详情
Connection——决定当前的事务完成后，是否会关闭网络连接； 详情
Date——创建报文的日期时间； 详情
Keep-Alive——用来设置超时时长和最大请求数；详情
Via——代理服务器的相关信息；详情
Warning——错误通知；详情
Trailer——允许发送方在分块发送的消息后面添加额外的元信息； 详情
Transfer-Encoding——指定报文主体的传输编码方式；详情
Upgrade——升级为其他协议；

2、Request headers

Accept——客户端可以处理的内容类型；详情
Accept-Charset——客户端可以处理的字符集类型；详情
Accept-Encoding——客户端能够理解的内容编码方式；详情
Accept-Language——客户端可以理解的自然语言；详情
Authorization——Web 认证信息；详情
Cookie——通过Set-Cookie设置的值；详情
DNT——表明用户对于网站追踪的偏好；详情
From——用户的电子邮箱地址；详情
Host——请求资源所在服务器；详情
If-Match——比较实体标记（ETag）；详情
If-Modified-Since——比较资源的更新时间；详情
If-None-Match——比较实体标记（与 If-Match 相反）；详情
If-Range——资源未更新时发送实体 Byte 的范围请求；详情
If-Unmodified-Since——比较资源的更新时间（与 If-Modified-Since 相反）；详情
Origin——表明了请求来自于哪个站点；详情
Proxy-Authorization——代理服务器要求客户端的认证信息；详情
Range——实体的字节范围请求；详情
Referer——对请求中 URI 的原始获取方；详情
TE——指定用户代理希望使用的传输编码类型；详情
Upgrade-Insecure-Requests——表示客户端优先选择加密及带有身份验证的响应；详情
User-Agent——浏览器信息；详情

3、Response Headers

Accept-Ranges——是否接受字节范围请求；详情
Age——消息对象在缓存代理中存贮的时长，以秒为单位；详情
Clear-Site-Data——表示清除当前请求网站有关的浏览器数据（cookie，存储，缓存）；详情
Content-Security-Policy——允许站点管理者在指定的页面控制用户代理的资源；详情
Content-Security-Policy-Report-Only—— 详情
ETag——资源的匹配信息；链接描述
Location——令客户端重定向至指定 URI；详情
Proxy-Authenticate——代理服务器对客户端的认证信息；详情
Public-Key-Pins——包含该Web 服务器用来进行加密的 public key （公钥）信息；详情
Public-Key-Pins-Report-Only——设置在公钥固定不匹配时，发送错误信息到report-uri；详情
Referrer-Policy——用来监管哪些访问来源信息——会在 Referer 中发送；详情
Server——HTTP 服务器的安装信息；详情
Set-Cookie——服务器端向客户端发送 cookie；详情
Strict-Transport-Security——它告诉浏览器只能通过HTTPS访问当前资源；详情
Timing-Allow-Origin——用于指定特定站点，以允许其访问Resource Timing API提供的相关信息；详情
Tk——显示了对相应请求的跟踪情况；详情
Vary——服务器缓存的管理信息；详情
WWW-Authenticate——定义了使用何种验证方式去获取对资源的连接；详情
X-XSS-Protection——当检测到跨站脚本攻击 (XSS)时，浏览器将停止加载页面；详情

4、Entity Headers

Allow——客户端可以处理的内容类型，这种内容类型用MIME类型来表示；详情
Content-Encoding——用于对特定媒体类型的数据进行压缩；详情
Content-Language——访问者希望采用的语言或语言组合；详情
Content-Length——发送给接收方的消息主体的大小；详情
Content-Location——替代对应资源的 URI；详情
Content-Range——实体主体的位置范围；详情
Content-Type——告诉客户端实际返回的内容的内容类型；详情
Expires——包含日期/时间， 即在此时候之后，响应过期；详情
Last-Modified——资源的最后修改日期时间；详情

三、HTTP 具体应用

1、Cookie

HTTP 协议是无状态的，主要是为了让 HTTP 协议尽可能简单，使得它能够处理大量事务。HTTP/1.1 引入 Cookie 来保存状态信息。

Cookie 是服务器发送到用户浏览器并保存在本地的一小块数据，它会在浏览器之后向同一服务器再次发起请求时**自动被携带**上，用于告知服务端两个请求是否来自同一浏览器。由于之后每次请求都会需要携带 Cookie 数据，因此会带来额外的性能开销（尤其是在移动环境下）。

Cookie 曾一度用于客户端数据的存储，因为当时并没有其它合适的存储办法而作为唯一的存储手段，但现在随着现代浏览器开始支持各种各样的存储方式，Cookie 渐渐被淘汰。新的浏览器 API 已经允许开发者直接将数据存储到本地，如使用 Web storage API（本地存储和会话存储）或 IndexedDB。

a、创建过程

服务器发送的响应报文包含Set-Cookie首部字段，客户端得到响应报文后把 Cookie 内容保存到浏览器中。

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: PHPSESSID=kq8v6iujarsgflkeq7shmai9c7
```

客户端之后对同一个服务器发送请求时，会从浏览器中取出 Cookie 信息并通过 Cookie 请求首部字段发送给服务器。

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: PHPSESSID=kq8v6iujarsgflkeq7shmai9c7
```

b、分类

会话期 Cookie：浏览器关闭之后它会被自动删除，也就是说它仅在会话期内有效。
持久性 Cookie：指定一个特定的过期时间（Expires）或有效期（max-age）之后就成为了持久性的 Cookie。
安全 Cookie：指定HttpOnly，这样cookie不能使用 JavaScript 经由 Document.cookie 属性，来防范跨站脚本攻击（XSS）。
HTTPS Cookie: 指定Secure，只有在请求使用SSL和HTTPS协议的时候才会被发送到服务器。

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;HttpOnly
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;Secure
```

2、缓存

降低客户端获取资源的延迟：缓存通常位于内存中，读取缓存的速度更快。并且缓存在地理位置上也有可能比源服务器来得近，例如浏览器缓存(但是只能缓存get，不能缓存其他类型请求)。
cache FAQ MDN

流程图:

a、判断cache-control或者expires是否有效

max-age值为缓存的毫秒数。

可以看到response-headers中设置了cache-control，并大于0，则下次直接从缓存(from disk cache)中获取

b、Etag判断

当发现Cache-Control设置的毫秒数过期了，则直接发送请求:
——如果响应中包含Etag(服务器生成的值)，则浏览器再次向web服务器发送请求并带上头If-None-Match（Etag的值），web服务器收到请求后发现有头If-None-Match 则与被请求资源的相应校验串进行比对，决定返回200或304(浏览器会从缓存中获取)。

——如果响应中不包含Etag，则进行Last-Modified判断

c、Last-Modified判断

当发现response header中含有Last-Modified，则再次向web服务器请求时带上头 If-Modified-Since，web服务器收到请求后发现有头If-Modified-Since 则与被请求资源的最后修改时间进行比对，然后服务器决定返回200或者304(缓存)

浏览器强制告诉服务器不缓存资源:

```
//request headers
Cache-Control:max-age=0, no-cache
```

四、HTTP 请求 method

五、HTTP 状态码

1、1XX

- 100(continue) 表明到目前为止都很正常，客户端可以继续发送请求或者忽略这个响应。

2、2XX

- 200(OK) 表示从客户端发来的请求在服务器端被正常处理了。
- 204(No Content) 该状态码代表服务器接收的请求已成功处理，但在返回的响应报文中不含实体的主体部分。
- 206(Partial Content) 该状态码表示客户端进行了范围请求，而服务器成功执行了这部分的 GET 请求。响应报文中包含由 Content-Range 指定范围的实体内容。

3、3XX

- 301(Moved Permanently) 永久性重定向。该状态码表示请求的资源已被分配了新的 URI，以后应使用资源现在所指的 URI。
- 302(Found) 临时性重定向。比如在没有登录情况下访问网站"个人中心"，会重定向到登录页，但是你登录后，访问个人中心时，它又不会重定向到其他地方了。
- 303(See Other) 和 302 有着相同的功能，但是 303 明确要求客户端应该采用 GET 方法获取资源。
- 304(Not Modified) 如果请求报文首部包含一些条件，例如：If-Match，If-Modified-Since，If-None-Match，If-Range，If-Unmodified-Since，如果不满足条件，则服务器会返回 304 状态码。

4、4XX

- 400(Bad Request) 该状态码表示请求报文中存在语法错误。当错误发生时，需修改请求的内容后再次发送请求。
- 401 Unauthorized 该状态码表示发送的请求需要有认证信息。返回含有 401 的响应必须包含一个适用于被请求资源的 WWW-Authenticate 首部用以询问用户信息。当浏览器初次接收到 401 响应，会弹出认证用的对话窗口。第二次接收到，则不弹出，直接表示认证失败。
- 403(Forbidden) 对请求资源的访问被服务器拒绝了，一般是未获得文件系统的访问授权，问权限出现某些问题。
- 404(Not Found) 浏览器地址错误。服务器找不到对应资源。

5、5XX

- 500(Internal Server Error) 服务器在执行时报错。
- 503(Service Unavailable) 服务器暂时处于超负载或正在进行停机维护，无响应。一般需要重启服务器即可。

六、MIME类型

浏览器通常使用MIME类型（而不是文件扩展名）来确定如何处理文档；因此设置正确的MIME类型附加到headers是非常重要的。

除了上面的基本的5中类型外，还有一种类型，即multipart类型。

> multipart/form-data 在表单中通过post上传
> multipart/byteranges (不常用，不知道)

一般的，如果没有显示的在request headers的Allow上设置类型，浏览器的MIME 嗅探会推测出来对应的类型，如果发现找不到特定的子类型，则给出默认类型，比如对于text文件类型若没有特定的subtype，就使用 text/plain。类似的，二进制文件没有特定或已知的 subtype，即使用 application/octet-stream。

七、HTTP 使用的认证方式

> 1、BASIC 认证（基本认证）
> 2、DIGEST 认证（摘要认证）
> 3、SSL 客户端认证
> 4、FormBase 认证（基于表单认证）

1、BASIC 认证

当在浏览器端输入一个url时，会自动弹出一个框，要求输入用户名和密码。此种方式为basic认证。
下面是认证执行过程：
第一步:在浏览器地址栏中输入 http://localhost:8080/auth
第二步: 服务器执行，发现需要认证，返回这个请求的响应。并在response headers中添加WWW-Authenticate，将http请求状态设置为401.

浏览器检测到WWW-Authenticate为basic后，自动弹出框。

第三步: 当用户看到框后，输入 用户名和密码，浏览器会将用户名和密码通过base64方式编码，然后添加到 request headers的 Authorization 中发送给服务器，浏览器验证通过，返回200状态码

如果验证不通过，则继续返回状态码401，提示验证失败。

缺点:
BASIC 认证虽然采用 Base64 编码方式，但这不是加密处理。不需要任何附加信息即可对其解码。换言之，由于明文解码后就是用户ID和密码,在 HTTP 等非加密通信的线路上进行 BASIC 认证的过程中，如果被人窃听，被盗的可能性极高。

2、DIGEST 认证

为了弥补Basic认证没有加密所带来的不安全性，出现了DIGEST 认证。
过程如下:
第一步:在浏览器地址栏中输入 http://localhost:8080/auth
第二步: 服务器执行，发现需要认证，返回这个请求的响应。并在response headers中添加WWW-Authenticate（包含有随机码nonce），将http请求状态设置为401.

浏览器检测到WWW-Authenticate为 digest 后，自动弹出框。

第三步: 当用户看到框后，输入 用户名和密码，浏览器会将用户名和上步返回的nouce，添加到 request headers的 Authorization 中，同时也将经过 MD5 运算后的密码字符串，生成key为response，一并添加到Authorization 中。至此请求的request headers的 Authorization 包含有如下信息。

```
//request header
Authorization: Digest username="my
name",realm="DIGEST",nounce="xxxxxxxxxxx",algorithm="MD5",response="xxxxxxxxxxxxx"
```

然后发送给服务器，浏览器验证通过，返回200状态码。

如果验证不通过，则继续返回状态码401，提示验证失败。

缺点: 虽然 DIGEST 认证提供了高于 BASIC 认证的安全等级， DIGEST 认证提供防止密码被窃听的保护机制，但并不存在防止用户伪装的保护机制，仍达不到多数 Web 网站对高度安全等级的追求标准。

3、SSL 客户端认证

对于 BASIC 认证和 DIGEST 认证来说，只要输入的用户名和密码正确，即可认证是本人的行为。但如果用户名和密码被盗，就很有可能被第三者冒充。

而利用 SSL客户端认证则可以避免该情况的发生。在SSL认证时，必须使用https协议。

由于SSL中的各种加密和秘钥算法过于复杂，有兴趣的可以直接阅读SSL相关书籍，本文忽略详细过程。

4、FormBase 认证

基于表单认证的标准规范尚未有定论，一般会使用 Cookie 来管理。
认证过程:
第一步：当用户在浏览器的登录页面，输入用户名和密码，通过http请求发送给后端。
第二步：后端保存用户的信息到session中，并返回sessionId, 通过http添加到response headers的Set-cookie中

```
//response headers
Set-cookie: PHPSESSID=kq8v6iujarsgflkeq7shmai9c7
```

然后浏览器成功登录，并跳转页面。
第三步：当用户访问个人中心或者其他页面时。http请求的request header中会自动携带cookie

```
//request headers
Cookie: PHPSESSID=kq8v6iujarsgflkeq7shmai9c7
```

这样，服务端会认为是你本人在操作。

但是如果攻击者通过"跨站脚本攻击（XSS）"，通过docuemnt.cookie来获取cookie，则sessionID很容易被盗。
为减轻XSS造成的损失，可以事先在 Set-cookie内加上 httponly 属性，这样就禁止了docuemnt.cookie操作。

```
Set-cookie: PHPSESSID=kq8v6iujarsgflkeq7shmai9c7, httponly
```

八、跨域资源共享(CORS)

是一种机制，它使用额外的 HTTP 头来告诉浏览器 让运行在一个 origin (domain) 上的Web应用被准许访问来自不同源服务器上的指定的资源。当一个资源从与该资源本身所在的服务器不同的域、协议或端口请求一个资源时，资源会发起一个跨域 HTTP 请求。

1、简单请求（不会触发CORS预检请求）

> 需要满足下列**所有**条件：
> 第一条: 请求方式必须为 GET | HEAD | POST
> 第二条: Content-Type 的值必须属于下列之一:application/x-www-form-urlencoded | multipart/form-data | text/plain
> 第三条: 在请求中，不会发送自定义的头部（如X-Modified）

例如: 在http://foo.exmaple上要访问 http://bar.other上的资源。则

```
//request headers上添加
Origin: http://foo.example
```

```
//response headers返回
Access-Control-Allow-Origin: *
```

由于在 http://foo.example 上要访问 http://bar.other，所以http://bar.other必须要告诉其...，能不能访问我。*号表示该资源可以被任意外域访问。

如果返回

```
Access-Control-Allow-Origin: http://foo.example
```

表示，http://bar.other的资源只能被http://foo.example访问，其他网站不能访问我。

2、非“简单请求”（触发CORS预检请求）

满足下列条件**之一**:
第一条： http请求方式为下列：PUT | DELETE | CONNECT | OPTIONS | TRACE | PATCH
第二条： Content-Type 的值不属于下列之一: application/x-www-form-urlencoded | multipart/form-data | text/plain
第三条: 在请求中，发送自定义的头部（如X-Modified）

如果在 http://foo.exmaple 上要访问 http://bar.other/resources/po... 上的资源。且 request headers 中 Content-Type为application/xml，请求method为post。
那么此请求是个“非简单请求”。首先浏览器会自动发送带有options选项的预检请求，然后发送实际请求

```
//预检请求request headers
OPTIONS /resources/post-here/ HTTP/1.1（自动，不需要设置）
Access-Control-Request-Method: POST
Access-Control-Request-Headers: Content-Type
```

```
//预检请求返回response headers
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
```

Access-Control-Allow-Origin 表明服务器允许任何其他服务器访问自己。
Access-Control-Allow-Methods 表明服务器允许客户端使用 POST, GET, OPTIONS 方法发起请求。
Access-Control-Allow-Headers 表明服务器允许请求中携带字段 X-PINGOTHER, Content-Type。
Access-Control-Max-Age 表明在86400内，不会再发送预检请求。

然后浏览器接着发送实际请求

```
POST /resources/post-here/ HTTP/1.1
Content-Type: application/xml; charset=UTF-8
```

实际请求返回

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

3、附带身份凭证的请求

一般而言，对于跨域 XMLHttpRequest 或 Fetch 请求，浏览器不会发送身份凭证信息。如果要发送凭证信息，需要设置 XMLHttpRequest 的某个特殊标志位。

```
var xhr = new XMLHttpRequest();
var url = 'http://xxxxxxxxx';

xhr.open('GET', url, true);
xhr.withCredentials = true;  // 设置发送人请求时 携带 cookie凭证
xhr.send();
```

请求发送

```
GET /resources/access-control-with-credentials/ HTTP/1.1
Cookie: pageAccess=2
```

请求返回

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example  //在携带凭证的请求中，返回不得设置为*，必须设置为具体域名
Access-Control-Allow-Credentials: true  //必须携带这个，否则响应内容不会返回给请求的发起者
```

4、CORS几个response header解释

4.1 Access-Control-Allow-Origin

它的值只有两种，要么*， 好么具体的域名 <origin>

4.2 Access-Control-Expose-Headers

在跨域访问时，XMLHttpRequest对象的getResponseHeader()方法只能拿到一些最基本的响应头：
Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma

无法访问其他的响应头（甚至在控制台network中看不到），如 set-cookie等，如果要访问其他头，则需要服务器设置，将能返回的响应头放入白名单

```
Access-Control-Expose-Headers: set-cookie
```

这样浏览器就能够通过getResponseHeader访问set-cookie响应头了

4.3 Access-Control-Max-Age

指定了预检请求的结果能够被缓存多久，在此时间内，不会再发起预检请求。

```
Access-Control-Max-Age: <seconds>
```

4.4 Access-Control-Allow-Credentials

指定了当浏览器的credentials设置为true时，是否允许浏览器读取response的内容。

寒水寺一禅
1.7k

关注作者

0 条评论

得票 •时间

撰写评论 ...

提交评论

推荐阅读

连肝7个晚上，总结了计算机网络的知识点！（共66条）

前言 计算机网络知识，是面试常考的内容，在实际工作中也常常会涉及到。 最近总结了66条计算机网络相关的知识点，大家一起...

达达前端 • 阅读 4.1k • 85 赞 • 3 评论

前端优化之 Http 相关优化总结

学习和总结文章同步发布于 [链接]，有兴趣可以关注一下，一起学习和进步。 Http 优化方式是前端性能优化的重要部分，也是前...

Samon • 阅读 916 • 17 赞

HTTP 缓存的四种风味与缓存策略

本文从属于笔者的HTTP 理解与实践系列文章，对于HTTP的学习主要包含HTTP 基础、HTTP 请求头与请求体、HTTP 响应头与状态...

王下邀月熊_Chevalier • 阅读 14.5k • 17 赞 • 2 评论

前端应该知道的http

作为互联网通信协议的一员老将，HTTP 协议走到今天已经经历了三次版本的变动，现在最新的版本是 HTTP2.0，相信大家早已耳...

Alan • 阅读 1.2k • 12 赞 • 2 评论

《图解 HTTP》 阅读摘要

这次做了一些笔记，方便自己和其他人翻阅和复习，因为这本书是 2014 年出的初版，所以有一些不怎么常用的技术，笔记中就省...

SHERlocked93 • 阅读 790 • 11 赞

前端必须知道的http缓存

相关字段简述 RFC2616规定的47种http报文首部字段中与缓存相关的字段。 通用头部字段 请求头部字段 响应头部字段 实体头部字...

Pines_Cheng • 阅读 3.5k • 9 赞 • 2 评论

HTTP——https、http缓存、get与post、web安全、跨域

HTTP诞生 1989年为知识共享而诞生的Web，提出了3项WWW构建技术： 标准通用标记语言设为HTML（HyperText Markup Langu...

Lmagic16 • 阅读 2k • 6 赞

谈谈HTTP1.0,HTTP1.1和HTTP2.0区别

>>>点击获取更多文章<<< HTTP定义 HTTP协议（HyperTextTransferProtocol，超文本传输协议）是用于从WWW服务器传输超文...

张炳 • 阅读 3.5k • 3 赞

小李子的前端

用户专栏

热爱前端的菜鸟，怀揣梦想的小白

25 人关注　**15** 篇文章

关注专栏　专栏主页