# 跨域资源共享 CORS 详解

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。

CORS需要浏览器和服务器同时支持。而整个CORS通信过程，都是浏览器自动完成。因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。

# 1、简单请求和非简单请求

浏览器将CORS请求分成两类：简单请求（simple request）和非简单请求（not-so-simple request）。

只要同时满足以下两大条件，就属于简单请求。

（1) 请求方法是以下三种方法之一：

* HEAD
* GET
* POST

（2）HTTP的头信息不超出以下几种字段：

* Accept
* Accept-Language
* Content-Language
* Last-Event-ID
* Content-Type：只限于三个值application/x-www-form-urlencoded、multipart/form-data、text/plain

凡是不同时满足上面两个条件，就属于非简单请求。

浏览器对这两种请求的处理，是不一样的。

## 2、浏览器对简单请求的处理

对于简单请求，浏览器直接发出CORS请求。在头信息之中，增加一个Origin字段。Origin字段用来说明，本次请求来自哪个源（协议 + 域名 + 端口）。服务器根据这个值，决定是否同意这次请求。

### Origin不被允许的情况

如果Origin指定的源，不在许可范围内，服务器会返回一个正常的HTTP回应。浏览器发现，这个回应的头信息没有包含Access-Control-Allow-Origin字段（详见下文），就知道出错了，从而抛出一个错误。

### Origin被允许的情况

如果Origin指定的域名在许可范围内，服务器返回的响应，会多出几个头信息字段。

如：

Access-Control-Allow-Origin: http://api.bob.com   
//该字段是必须的。它的值要么是请求时Origin字段的值，要么是一个\*，表示接受任意域名的请求。  
  
Access-Control-Allow-Credentials: true  
//表示是否允许发送Cookie。默认情况下，Cookie不包括在CORS请求之中。  
设为true，即表示服务器明确许可，Cookie可以包含在请求中，一起发给服务器。这个值也只能设为true，如果服务器不要浏览器发送Cookie，删除该字段即可。  
  
Access-Control-Expose-Headers: FooBar  
// 该字段可选。CORS请求时，XMLHttpRequest对象的getResponseHeader()方法只能拿到6个基本字段：Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma。如果想拿到其他字段，就必须在Access-Control-Expose-Headers里面指定。上面的例子指定，getResponseHeader('FooBar')可以返回FooBar字段的值。  
  
Content-Type: text/html; charset=utf-8

### withCredentials 属性

如果要把Cookie发到服务器，一方面要服务器同意，指定Access-Control-Allow-Credentials字段。

开发者必须在AJAX请求中打开withCredentials属性。

否则，即使服务器同意发送Cookie，浏览器也不会发送。或者，服务器要求设置Cookie，浏览器也不会处理。

## 3、浏览器对非简单请求的处理

### 预检请求

非简单请求是那种对服务器有特殊要求的请求，比如请求方法是PUT或DELETE，或者Content-Type字段的类型是application/json。

非简单请求的CORS请求，会在正式通信之前，增加一次HTTP查询请求，称为"预检"请求（preflight）。

浏览器先询问服务器，当前网页所在的域名是否在服务器的许可名单之中，以及可以使用哪些HTTP动词和头信息字段。只有得到肯定答复，浏览器才会发出正式的XMLHttpRequest请求，否则就报错。

"预检"请求用的请求方法是OPTIONS

如

OPTIONS /cors HTTP/1.1  
Origin: http://api.bob.com  
Access-Control-Request-Method: PUT  
Access-Control-Request-Headers: X-Custom-Header  
Host: api.alice.com  
Accept-Language: en-US  
Connection: keep-alive  
User-Agent: Mozilla/5.0...

其中：

* Origin，表示请求来自哪个源。
* **Access-Control-Request-Method**：该字段是必须的，用来列出浏览器的CORS请求会用到哪些HTTP方法，如put
* **Access-Control-Request-Headers**：指定浏览器CORS请求会额外发送的头信息字段

### 服务器回应预检请求

服务器收到"预检"请求以后，检查了Origin、Access-Control-Request-Method和Access-Control-Request-Headers字段以后，确认允许跨源请求，就可以做出回应。

如

HTTP/1.1 200 OK  
Date: Mon, 01 Dec 2008 01:15:39 GMT  
Server: Apache/2.0.61 (Unix)  
Access-Control-Allow-Origin: http://api.bob.com （表示同意跨源请求）  
Access-Control-Allow-Methods: GET, POST, PUT  
Access-Control-Allow-Headers: X-Custom-Header  
Access-Control-Allow-Credentials: true  
Access-Control-Max-Age: 1728000  
Content-Type: text/html; charset=utf-8  
Content-Encoding: gzip  
Content-Length: 0  
Keep-Alive: timeout=2, max=100  
Connection: Keep-Alive  
Content-Type: text/plain

**Access-Control-Max-Age**

该字段可选，用来指定本次预检请求的有效期，单位为秒。上面结果中，有效期是20天（1728000秒），即允许缓存该条回应1728000秒（即20天），在此期间，不用发出另一条预检请求。

如果服务器否定了"预检"请求，会返回一个正常的HTTP回应，但是没有任何CORS相关的头信息字段。这时，浏览器就会认定，服务器不同意预检请求。控制台会打印出如下的报错信息。

### 预检请求之后浏览器的正常请求和回应

一旦服务器通过了"预检"请求，以后每次浏览器正常的CORS请求，就都跟简单请求一样，会有一个Origin头信息字段。服务器的回应，也都会有一个Access-Control-Allow-Origin头信息字段。

## 4、CORS与JSONP的比较

JSONP只支持GET请求，CORS支持所有类型的HTTP请求。JSONP的优势在于支持老式浏览器，以及可以向不支持CORS的网站请求数据。