自：<https://blog.csdn.net/qq_31617637/article/details/72955239>

# 跨域问题出现原因和解决方案

2017年06月09日 11:09:38 [踏凌霄](https://me.csdn.net/qq_31617637) 阅读数：11472 标签： [跨域](http://so.csdn.net/so/search/s.do?q=%E8%B7%A8%E5%9F%9F&t=blog) 更多

个人分类： [跨域](https://blog.csdn.net/qq_31617637/article/category/6963552)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qq\_31617637/article/details/72955239

## 出现原因

* 【出现原因】什么是跨域以及产生原因

跨域是指a页面想获取b页面资源，如果a、b页面的协议、域名、端口、子域名不同，或是a页面为ip地址，b页面为域名地址，所进行的访问行动都是跨域的，而浏览器为了安全问题一般都限制了跨域访问，也就是不允许跨域请求资源。

| **Uri** | **说明** | **是否跨域** |
| --- | --- | --- |
| <http://www.cnblogs.com/a.js>  <http://www.a.com/b.js> | 不同域名 | 是 |
| <http://www.a.com/lab/a.js> <http://www.a.com/script/b.js> | 同域名下不同文件 | 否 |
| <http://www.a.com:8000/a.js>  <http://www.a.com/b.js> | 同域名下不同端口 | 是 |
| <http://www.a.com/a.js>  <https://www.a.com/b.js> | 同域名 不同协议 | 是 |
| <http://www.a.com/a.js>  <http://70.32.92.74/b.js> | 域名和域名对应ip | 是 |
| <http://www.a.com/a.js>  <http://script.a.com/b.js> | 主域名相同 子域名不同 | 是（cookie不可访问） |
| <http://www.a.com/a.js>  <http://a.com/b.js> | 同一域名，不同二级域名（同上） | 是 |

## 解决方案

* 【策略一】Jsonp 需要目标服务器配合一个callback函数

JSONP(JSON with Padding)是一个非官方的协议，它允许在服务器端集成Script tags返回至客户端，通过javascript callback的形式实现跨域访问（这仅仅是JSONP简单的实现形式）。

Json+padding（内填充），顾名思义，就是把JSON填充到一个盒子里，它的基本思想是，网页通过添加一个<script>元素，向服务器请求JSON数据，这种做法不受同源政策限制；服务器收到请求后，将数据放在一个指定名字的回调函数里传回来。   
首先，网页动态插入<script>元素，由它向跨源网址发出请求。

//Js 客户端 方法一

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

<script type="text/javascript">

function jsonpCallback(result) {

//alert(result);

for(var i in result) {

alert(i+":"+result[i]);//循环输出a:1,b:2,etc.

}

}

var JSONP=document.createElement("script");

JSONP.type="text/javascript";

JSONP.src="http://crossdomain.com/services.php?callback=jsonpCallback";

document.getElementsByTagName("head")[0].appendChild(JSONP);

</script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

//Js 客户端 方案二

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

<script type="text/javascript">

function jsonpCallback(result) {

alert(result.a);

alert(result.b);

alert(result.c);

for(var i in result) {

alert(i+":"+result[i]);//循环输出a:1,b:2,etc.

}

}

</script>

<script type="text/javascript" src="http://crossdomain.com/services.php?callback=jsonpCallback"></script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13

//Jq 客户端 方案一

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$.getJSON("http://crossdomain.com/services.php?callback=?",

function(result) {

for(var i in result) {

alert(i+":"+result[i]);//循环输出a:1,b:2,etc.

}

});

</script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

//Jq 客户端 方案二

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$.ajax({

url:"http://crossdomain.com/services.php",

dataType:'jsonp',

data:'',

jsonp:'callback',

success:function(result) {

for(var i in result) {

alert(i+":"+result[i]);//循环输出a:1,b:2,etc.

}

},

timeout:3000

});

</script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

//Jq 客户端 方案三

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$.get('http://crossdomain.com/services.php?callback=?', {name: encodeURIComponent('tester')}, function (json) { for(var i in json) alert(i+":"+json[i]); }, 'jsonp');

</script>

* 1
* 2
* 3
* 4
* 5

<?php

//服务端返回JSON数据

$arr=array('a'=>1,'b'=>2,'c'=>3,'d'=>4,'e'=>5);

$result=json\_encode($arr);

//echo $\_GET['callback'].'("Hello,World!")';

//echo $\_GET['callback']."($result)";

//动态执行回调函数

$callback=$\_GET['callback'];

echo $callback."($result)";

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 【策略二】通过修改document.domain来跨子域

将子域和主域的document.domain设为同一个主域.前提条件：这两个域名必须属于同一个基础域名!而且所用的协议，端口都要一致，否则无法利用document.domain进行跨域

主域相同的使用document.domain

* 【策略三】使用window.name来进行跨域

.window.name+iframe 需要目标服务器响应window.name,window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的!

javascript   
window.name = data;   
//接着，子窗口跳回一个与主窗口同域的网址。   
location = 'http://parent.url.com/xxx.html';   
//然后，主窗口就可以读取子窗口的window.name了   
var data = document.getElementById('iframe').contentWindow.name;

优点：window.name容量很大，可以防止非常长的字符串；   
缺点：必须监听子窗口window.name属性的变化，会影响网页性能。   
- 【策略四】跨文档消息传输window.postMessage

上面两种方法都属于破解，HTML5为解决这个问题，引入一个全新的API：跨文档消息传输Cross Document Messaging。   
下一代浏览器都将支持这个功能:Chrome 2.0+、Internet Explorer 8.0+, Firefox 3.0+, Opera 9.6+, 和 Safari 4.0+ 。   
Facebook已经使用了这个功能，用postMessage支持基于web的实时消息传递。

使用方法：otherWindow.postMessage(message, targetOrigin);   
otherWindow: 对接收信息页面的window的引用。可以是页面中iframe的contentWindow属性;window.open的返回值；   
通过name或下标从window.frames取到的值。   
message: 具体的信息内容，string类型。   
targetOrigin: 接受消息的窗口的源(origin)，即“协议+域名+端口”。也可以设为“\*”，表示不限制域名，向所有窗口发送。   
message事件的事件对象event，提供一下三个属性:   
(1).event.source:发送消息的窗口   
(2).event.origin:消息发向的网站   
(3).event.data:消息内容

// a.com/index.html中的代码

<iframe id="ifr" src="b.com/index.html"></iframe>

<script type="text/javascript">

window.onload = function() {

var ifr = document.getElementById('ifr');

var targetOrigin = 'http://b.com'; // 若写成'http://b.com/c/proxy.html'效果一样

// 若写成'http://c.com'就不会执行postMessage了

ifr.contentWindow.postMessage('I was there!', targetOrigin);

};

</script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

// b.com/index.html中的代码

<script type="text/javascript">

window.addEventListener('message', function(event){

// 通过origin属性判断消息来源地址

if (event.origin == 'http://a.com') {

alert(event.data); // 弹出"I was there!"

alert(event.source); // 对a.com、index.html中window对象的引用

// 但由于同源策略，这里event.source不可以访问window对象

}

}, false);

</script>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

* 【策略五】通过CORS解决AJAX跨域

CORS是跨源资源分享（Cross-Origin Resource Sharing）的缩写。它是W3C标准，是跨源AJAX请求的根本解决方法。相比JSONP只能发GET请求，CORS允许任何类型的请求。   
定义：CORS其实出现时间不短了，它在维基百科上的定义是：跨域资源共享（CORS）是一种网络浏览器的技术规范，它为Web服务器定义了一种方式，允许网页从不同的域访问其资源。而这种访问是被同源策略所禁止的。CORS系统定义了一种浏览器和服务器交互的方式来确定是否允许跨域请求。 它是一个妥协，有更大的灵活性，但比起简单地允许所有这些的要求来说更加安全。而W3C的官方文档目前还是工作草案，但是正在朝着W3C推荐的方向前进。

简言之，CORS就是为了让AJAX可以实现可控的跨域访问而生的。

以往的解决方案：   
以前要实现跨域访问，可以通过JSONP、Flash或者服务器中转的方式来实现，但是现在我们有了CORS。   
CORS与JSONP相比，无疑更为先进、方便和可靠。   
1、 JSONP只能实现GET请求，而CORS支持所有类型的HTTP请求。   
2、 使用CORS，开发者可以使用普通的XMLHttpRequest发起请求和获得数据，比起JSONP有更好的错误处理。   
3、 JSONP主要被老的浏览器支持，它们往往不支持CORS，而绝大多数现代浏览器都已经支持了CORS（这部分会在后文浏览器支持部分介绍）。

 【策略六】通过设置Access-Control-Allow-Origin

在被请求的Response header中加入 :

// 指定允许其他域名访问

header('Access-Control-Allow-Origin:\*');

// 响应类型

header('Access-Control-Allow-Methods:POST');

// 响应头设置

header('Access-Control-Allow-Headers:x-requested-with,content-type');

* 1
* 2
* 3
* 4
* 5
* 6

就可以实现ajax POST跨域访问了。

客户端请求：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=utf-8">

<title> 跨域测试 </title>

<script src="//code.jquery.com/jquery-1.11.3.min.js"></script>

</head>

<body>

<div id="show"></div>

<script type="text/javascript">

$.post("http://www.server.com/server.php",{name:"fdipzone",gender:"male"})

.done(function(data){

document.getElementById("show").innerHTML = data.name + ' ' + data.gender;

});

</script>

</body>

</html>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17

服务器端处理：

<?php

$ret = array(

'name' => isset($\_POST['name'])? $\_POST['name'] : '',

'gender' => isset($\_POST['gender'])? $\_POST['gender'] : ''

);

header('content-type:application:json;charset=utf8');

header('Access-Control-Allow-Origin:\*');

header('Access-Control-Allow-Methods:POST');

header('Access-Control-Allow-Headers:x-requested-with,content-type');

echo json\_encode($ret);

?>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

Access-Control-Allow-Origin:\* 表示允许任何域名跨域访问   
如果需要指定某域名才允许跨域访问，只需把Access-Control-Allow-Origin:\*改为Access-Control-Allow-Origin:允许的域名   
例如：header(‘Access-Control-Allow-Origin:[http://www.client.com](http://www.client.com/)‘);

如果需要设置多个域名允许访问，这里需要用PHP处理一下   
例如允许 www.client.com 与 www.client2.com 可以跨域访问

服务器端处理：

<?php

$ret = array(

'name' => isset($\_POST['name'])? $\_POST['name'] : '',

'gender' => isset($\_POST['gender'])? $\_POST['gender'] : ''

);

header('content-type:application:json;charset=utf8');

$origin = isset($\_SERVER['HTTP\_ORIGIN'])? $\_SERVER['HTTP\_ORIGIN'] : '';

$allow\_origin = array(

'http://www.client.com',

'http://www.client2.com'

);

if(in\_array($origin, $allow\_origin)){

header('Access-Control-Allow-Origin:'.$origin);

header('Access-Control-Allow-Methods:POST');

header('Access-Control-Allow-Headers:x-requested-with,content-type');

}

echo json\_encode($ret);

?>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21

 【策略七】通过Nginx反向代理 

之前我们跨域是借助了浏览器对 Access-Control-Allow-Origin 的支持。但有些浏览器是不支持的，所以这并非是最佳方案,现在我们来利用nginx 通过反向代理 满足浏览器的同源策略实现跨域!   
  
然后我们回到nginx.conf 配置一个反向代理路径（location /apis部分）:

server {

listen 8094;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

}

location /apis {

rewrite ^.+apis/?(.\*)$ /$1 break;

include uwsgi\_params;

proxy\_pass http://localhost:1894;

}

}