ajax请求总是不成功？浏览器的同源策略和跨域问题详解

木酱 2016-04-13 16:41:35 浏览11413 评论3 发表于： 阿里云 DataV 数据可视化 >> 前端

互联网产品及应用 大数据 前端与交互设计 javascript html5 数据可视化

摘要： XMLHttpRequest cannot load http://oldwang.com/isdad. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://xiao

场景

码农小明要做一个展示业务数据的大屏给老板看，里面包含了来自自己网站的数据和来自隔壁老王的数据。

那么自己网站的数据提供了 http://xiaoming.com/whoami 这样的数据接口

隔壁老王提供了 http://oldwang.com/isdad 这样的数据接口

单独点开都是没有问题的。但是一使用 js 的 ajax 请求就无法收到来自 oldwang.com 的数据了。

点开浏览器控制台一看，红字标出（Chrome）：

```
XMLHttpRequest cannot load http://oldwang.com/isdad. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://xiaoming.com/' is therefore not allowed access.
```

这就是遇到了跨域问题

为什么会有这样的问题？

想象一下如果隔壁老王根本不认识你，他的网站自己有各种用户接口、订单接口、文章接口，那么谁都可以把这些接口返回的数据直接放在自己的网站上了，还是实时的。

> 所以浏览器制定了一个 **同源策略** ，限制了从一个源（origin）中的脚本获取来自其他源（origin）中的资源。

什么是同源

如果两个页面拥有相同的协议（protocol: http），端口（port: 80），和主机（host: xiaoming.com），那么这两个页面就属于同一个源（origin）。

解决方案

这里就不讲多年前的iframe、flash等方式了，只讲几个最常用到的方案

A.x.com 和 B.x.com 间的跨域

子域名不同也是会受到跨域限制的。这种问题最简单，只需要将页面声明为更高级的域就可以了。

```
<script>
  document.domain = "x.com";
</script>
```

最经典、高效、浏览器兼容最好的解决方案：JSONP

但是有一个致命的缺点：非常高的跨站脚本攻击风险，所以 DataV 是不支持这种方式的

看到JSONP这个名字很多人以为它是和JSON密切相关的一种用来跨域的黑科技，但实际上从跨域的视角看，跟JSON并没

看到JSONP这个名字很多人以为这是和JSON密切相关的一种用来跨域的黑科技，但实际上从跨域的视角看，跟JSON并没有一毛钱关系，他是利用了浏览器允许跨域加载 js 等资源来获取数据。

因为浏览器支持跨域加载 js 如 `<script src="http://aliyun.com/....."></script>` ，所以很简单，可以把数据包装成 js 就可以了。

这是数据，通过 script 加载到数据无法“执行”，更无法传给 ajax 的回调函数：

```
{
  "data": 123
}
```

这是js脚本，只要将 `callback` 与 ajax 的回调函数做关联，就可以讲数据传给回调函数：

```
callback({
  "data": 123
})
```

这可以看到四点：

一、需要 callback 与 ajax 回调函数绑定；

二、需要数据服务器 **配合** 的。

三、只支持GET请求

四、数据服务器可以随意插入危险的脚本

前端如果用 jquery，jquery已经完成了整个取值过程的封装，逻辑是：

1. 随机生成不重复的 callback 函数名，并与 ajax 回调函数 绑定。

1. 将 callback 函数名放入 URL 的 query string 中，如

`http://oldwang.com/api?callback= jQuery21401555797592329124 5_1460532274390`

1. 生成一个 `<script>` 标签，将上述 URL 作为 src

1. 等待数据加载，并把数据传入 ajax 的回调函数

后端以 php 为例，逻辑是获取浏览器传来一个参数作为callback包装数据：

```
<?php
  echo $_GET['callback']."(". $data .")";
?>
```

大部分新浏览器都兼容的 CORS（Cross Origin Resource Sharing）

他的原理是隔壁老王主动告诉浏览器“别拦着小明，我们是亲戚……”

所以最简单的例子，就是在数据服务器返回的头信息中包含：

```
Access-Control-Allow-Origin: http://xiaming.com
```

然而这个头信息并不支持枚举，那如果隔壁老王的亲戚太多就只能通过程序来动态得生成这个头信息了，以PHP为例：

```
<?php
if (is_my_bastard($_SERVER['HTTP_ORIGIN'])) {
 header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
}
?>
```

如果老王作为一个好人，来者不拒。那么可以直接使用 *

```
```

```
Access-Control-Allow-Origin: *
```

Cookies

CORS默认是不带 cookie 信息的，如果要带上 cookie 需要添加 withCredentials 参数，以 jquery 为例：

```
$.ajax({
    url: "http://laowang.com/isdad",
    xhrFields: {
      withCredentials: true
    }
  });
```

而服务器还需要加上允许 Credentials 的头信息以及不允许用通配符“*”，如下面的代码

```
<?php
if (is_my_bastard($_SERVER['HTTP_ORIGIN'])) {
  header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}"); // 不允许用 *
  header("Access-Control-Allow-Credentials:true");
}
?>
```

这就是隔壁老王的故事

其他参数可以参阅：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Access_control_CORS