跨域访问设置

一、什么是跨域?

跨域是指一个域下的文档或脚本试图去请求另一个域下的资源。广义的跨域包括：

1. 资源跳转：A链接、重定向、表单提交
2. 资源嵌入：`<link>`、`<script>`、`<img>`、`<video>`等Dom标签以及样式表中的background:url( )、@font-face( )等文件外链
3. 脚本请求：Ajax请求、Dom和JavaScript对象的跨域操作等

而我们通常所说的跨域是狭义的，是由浏览器同源策略限制的一类请求场景。**同源策略**（Same-origin policy）是浏览器最核心也最基本的一个安全策略，它限制了从同一个源加载的文档或脚本如何与另一个源的资源进行交互。它能帮助阻隔恶意文档，减少可能被攻击的媒介。所谓同源是指协议、域名、端口三者相同，当一个请求URL的协议、域名、端口三者中任意一个与当前页面URL不同即为跨域。同源策略的限制如下：

1. 无法读取非同源网页的Cookie、LocalStorage和IndexDB
2. 无法获得非同源网页的Dom和Js对象
3. 无法向非同源地址发送Ajax请求

二、跨域解决方案

常见的跨域解决方案有：

- 通过jsonp跨域
- 跨域资源共享（CORS）
- Nginx代理跨域
- postMessage跨域
- WebSocket协议跨域
- ...

开发者应结合实际情况选择最适合的方案，下面将对与保利威视频播放服务相关的跨域问题解决方案进行介绍。

1、通过jsonp跨域

`<script>`、`<img>`等一些获取资源的HTML标签是没有跨域限制的，基于此原理，网页可通过添加一个`<script>`，向服务器请求 JSON 数据，服务器收到请求后将数据放在一个指定名字的回调函数的参数位置传回来。

JSONP 是服务器与客户端跨源通信的常用方法。最大特点就是简单适用，兼容性好（兼容低版本IE），缺点是只支持GET请求。

原生实现：

```
<script src="http://www.domain2.com:8080/login?user=admin&callback=handleCallback"></script>
// 向服务器test.com发出请求，该请求的查询字符串有一个callback参数，用来指定回调函数的名字

// 处理服务器返回回调函数的数据
<script type="text/javascript">
    function handleCallback(res){
        // 处理获得的数据
        console.log(res);
    }
</script>
```

服务端返回数据如下：

```
handleCallback({"status":true,"user":"admin"})
```

jQuery Ajax：

```
$.ajax({
    url: 'http://www.domain2.com:8080/login',
    type: 'get',
    dataType: 'jsonp',  // 请求方式为jsonp
    jsonpCallback: "handleCallback",    // 自定义回调函数名
    data: {}
});
```

Vue.js：

```
this.$http.jsonp('http://www.domain2.com:8080/login', {
    params: {},
    jsonp: 'handleCallback'
}).then((res) => {
    console.log(res);
})
```

2、跨域资源共享（CORS）

跨域资源共享（Cross-Origin Resource Sharing）是W3C标准，目前所有浏览器都支持该功能（IE8/9需要使用XDomainRequest对象来支持CORS），CORS也已经成为主流的跨域解决方案。详见Http访问控制CORS的详解。

- 普通跨域请求：只需服务端设置**Access-Control-Allow-Origin**即可，前端无需设置
- 带cookie跨域请求：前后端都需要进行设置

前端设置：

1.) 原生Ajax

```
var xhr = new XMLHttpRequest(); // IE8/9需用window.XDomainRequest兼容

// 前端设置是否带cookie
xhr.withCredentials = true;

xhr.open('post', 'http://www.domain2.com:8080/login', true);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.send('user=admin');

xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        alert(xhr.responseText);
    }
};
```

2.) jQuery Ajax

```
$.ajax({
   url: 'http://www.domain2.com:8080/login',
   type: 'get',
   data: {},
   xhrFields: {
       withCredentials: true    // 前端设置是否带cookie
   },
   crossDomain: true,   // 会让请求头中包含跨域的额外信息，但不会含cookie
});
```

3.) vue-resource

```
Vue.http.options.credentials = true
```

4.) axios

```
axios.defaults.withCredentials = true
```

服务端设置：

服务器端对于CORS的支持，主要是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。

1.) Java后台

```
/*
 * 导入包：import javax.servlet.http.HttpServletResponse;
 * 接口参数中定义：HttpServletResponse response
 */

// 允许跨域访问的域名：若有端口需写全（协议+域名+端口），若没有端口末尾不用加'/'
response.setHeader("Access-Control-Allow-Origin", "http://www.domain1.com");

// 允许前端带认证cookie：启用此项后，上面的域名不能为'*'，必须指定具体的域名，否则浏览器会提示
response.setHeader("Access-Control-Allow-Credentials", "true");

// 提示OPTIONS预检时，后端需要设置的两个常用自定义头
response.setHeader("Access-Control-Allow-Headers", "Content-Type,X-Requested-With");
```

2.) PHP后台

```
<?php
 header("Access-Control-Allow-Origin:*");
```

3、Flash播放器的跨域配置

Flash播放器在请求授权播放和跑马灯接口时，需要配置允许跨域请求。配置方式为：添加crossdomain.xml文件到播放域名的根目录下。

crossdomain.xml文件内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<cross-domain-policy>
    <allow-access-from domain="*"/>
    <allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
```