

morethink

经史子集，了然于胸。谈笑之间，代码写就。

博客园 首页 联系 管理

随笔 - 64 文章 - 0 评论 - 28

SpringMVC解决跨域问题

有个朋友在写扇贝插件的时候遇到了跨域问题。
于是我对解决跨域问题的方式进行了一番探讨。

问题

API: 查询单词

URL: <https://api.shanbay.com/bdc/search/?word={word}>

请求方式: GET

参数: {word}, 必须, 要查询的单词

报错为

```
XMLHttpRequest cannot load http://localhost/home/saveCandidate. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access. The response had HTTP status code 404.
```

这就是典型的跨域问题。

但是我在浏览器里输入URL是可以进行查询单词的操作的, 有什么不同, 即下面两个问题

1. 为什么在浏览器地址栏输入URL不会出现跨域问题。
2. 不在服务器运行的html是否可以完成一次http请求

经过Google和自己测试

1. 跨域限制是浏览器行为, 不是服务器行为。浏览器认为地址栏输入时安全的, 所以不限制认为是跨域。
2. 可以, 只要服务器配置为所有域都可以进行请求, 那么不在服务器运行的HTML就可以完成http请求。

什么是跨域问题

同源策略:

2

0

同源指的是域名（或IP），协议，端口都相同，不同源的客户端脚本(javascript、ActionScript)在没明确授权的情况下，不能读写对方的资源。

URL	解释	是否跨域
http://www.morethink.cn	原来的URL	
http://www.image.morethink.cn	子域名	跨域(cookie也无法访问)
http://morethink.cn	不加www	跨域
https://www.morethink.cn	更改协议	跨域
http://www.morethink.cn:8080	更改端口号	跨域

原因：

同源政策的目的是为了保证用户信息的安全，防止恶意的网站窃取数据。
设想这样一种情况：**A**网站是一家银行，用户登录以后，又去浏览其他网站。如果其他网站可以读取**A**网站的**Cookie**，会发生什么？
很显然，如果**Cookie**包含隐私（比如存款总额），这些信息就会泄漏。更可怕的是，**Cookie**往往用来保存用户的登录状态，如果用户没有退出登录，其他网站就可以冒充用户，为所欲为。因为浏览器同时还规定，提交表单不受同源政策的限制。
由此可见，"同源政策"是必需的，否则 **Cookie** 可以共享，互联网就毫无安全可言了。

同源策略限制以下几种行为：

- 1. Cookie、LocalStorage 和 IndexDB 无法读取
- 2. DOM 和 Js对象无法获得
- 3. AJAX 请求不能发送

模拟跨域问题

测试URL为 http://localhost:80/home/allProductions

可以直接在浏览器 `console` 中执行

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:80/home/allProductions', true);
xhr.send();
xhr.onreadystatechange=function() {
  if(xhr.readyState == 4) {
    if(xhr.status == 200) {
      console.log(JSON.parse(xhr.responseText));
    }
  }
}
```

20

在任意网站打开控制台，执行此段代码可以模拟跨域请求。

在知乎控制台打开报错如下

Mixed Content: The page at 'https://www.zhihu.com/question/26376773' was loaded over HTTPS, but requested an insecure XMLHttpRequest endpoint 'http://localhost/home/allProductions'. This request has been blocked; the content must be served over HTTPS.



模拟跨域请求

因为知乎是https，报错与普通的http协议不同。

再澄清一下跨域问题：

- 1. 并非浏览器限制了发起跨站请求，而是跨站请求可以正常发起，但是返回结果被浏览器拦截了。最好的例子是CRSF跨站攻击原理，无论是否跨域，请求已经发送到了后端服务器！
- 2. 但是，有些浏览器不允许从HTTPS的域跨域访问HTTP，比如Chrome和Firefox，这些浏览器在请求还未发出的时候就会拦截请求，这是一个特例。

在博客园控制台打开报错如下

XMLHttpRequest cannot load http://localhost/home/allProductions. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.cnblogs.com' is therefore not allowed access.

怎么解决跨域问题

解决方案有很多

- 1. 通过jsonp跨域
- 2. document.domain + iframe跨域
- 3. location.hash + iframe
- 4. window.name + iframe跨域
- 5. postMessage跨域
- 6. 跨域资源共享（CORS）
- 7. [前端通过Nginx解决跨域问题](#)
- 8. nodejs中间件代理跨域
- 9. WebSocket协议跨域

1. JSONP
2. CORS
3. WebSocket

可以直接参考[Spring MVC 4.1 支持jsonp](#)进行配置你的SpringMVC注解

我虽然请求不了json数据，但是我可以请求一个 `Content-Type` 为 `application/javascript` 的JavaScript对象，这样就可以避免浏览器的同源策略。

就是当服务器接受到名为 `jsonp` 或者 `callback` 的参数时，返回 `Content-Type: application/javascript` 的结果，从而避免浏览器的同源策略检测。

```
function println(data) {
    console.log(data);
}

var url = "http://localhost:80/home/allProductions?&callback=println";
// 创建script标签，设置其属性
var script = document.createElement('script');
script.setAttribute('src', url);
// 把script标签加入head，此时调用开始
document.getElementsByTagName('head')[0].appendChild(script);
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <script src="http://cdn.bootcss.com/jquery/1.10.2/jquery.min.js"></script>
  <script type="text/javascript">
    function println(data) {
      console.log(data);
```

```

        console.log(print');
    }
    function jsonp_test() {
        $.ajax({
            type: "get",
            url: "http://localhost:80/home/allProductions",
            dataType: "jsonp",
            jsonp: "callback", //传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(一般默认为:callback)

            jsonpCallback: "println", //返回后调用的处理函数
            error: function () { //请求出错的处理
                alert("请求出错");
            }
        });
    }
}
</script>
</head>
<body onload=jsonp_test()>
</body>
</html>

```

CORS

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

CORS需要浏览器和服务器同时支持。

1. 所有浏览器都支持该功能，IE浏览器不能低于IE10。
整个**CORS**通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一个附加的请求，但用户不会有感觉。
2. 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。

即**CORS**与普通请求代码一样。

CORS与JSONP相比

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。

@CrossOrigin 注解

此注解既可用于方法也可用于类

源码如下：

```

@CrossOrigin(origins = "http://www.zhihu.com")
@RequestMapping(value = "/allProductions", method = RequestMethod.GET)
public Result getAllOldProductions() {

}

```

2

0

@CrossOrigin 注解既可注解在方法上，也可注解在类上。

完成配置之后

XML全局配置

所有跨域请求都可以访问

```
<mvc:cors>
  <mvc:mapping path="/**" />
</mvc:cors>
```

更加细粒度的配置：

```
<mvc:cors>

  <mvc:mapping path="/api/**"
    allowed-origins="http://domain1.com, http://domain2.com"
    allowed-methods="GET, PUT"
    allowed-headers="header1, header2, header3"
    exposed-headers="header1, header2" allow-credentials="false"
    max-age="123" />

  <mvc:mapping path="/resources/**"
    allowed-origins="http://domain1.com" />

</mvc:cors>
```

WebSocket

WebSocket是一种通信协议，使用ws://（非加密）和wss://（加密）作为协议前缀，在2008年诞生，2011年成为国际标准。所有浏览器都已经支持了。

它的最大特点就是，服务器可以主动向客户端推送信息，客户端也可以主动向服务器发送信息，是真正双向平等对话，属于服务器推送技术的一种。

该协议不实行同源政策，只要服务器支持，就可以通过它进行跨源通信。

请求头信息：(多了个 origin)

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

响应头：(如果origin在白名单内)

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

2

0

相比于HTTP/2

HTTP/2只是对HTML、CSS等JS资源的传输方式进行了优化，并没有提供新的JS API，不能用于实时传输消息，也无法推送指定的信息。

参考文档：

- 1. 跨域
 - [浏览器同源政策及其规避方法](#)
 - [跨域资源共享 CORS 详解](#)
- 2. SpringMVC 跨域解决方法
 - [Spring MVC 4.2 增加 CORS 支持](#)
- 3. [前端常见跨域解决方案（全）](#)

本文作者： 李文浩

本文链接：<https://www.cnblogs.com/morethink/p/6525216.html>

版权声明： 本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 3.0](#) 许可协议。转载请注明出处！

分类： SpringMVC

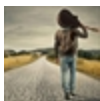
好文要顶

关注我

收藏该文







[morethink](#)
[关注 - 24](#)
[粉丝 - 41](#)

[+加关注](#)

« 上一篇：[SpringMVC空字符串转为null](#)
» 下一篇：[前端通过Nginx反向代理解决跨域问题](#)

posted @ 2017-03-09 13:24 morethink 阅读(24616) 评论(3) 编辑 收藏

评论列表

#1楼 2017-03-17 18:31 叫我小红依吧

还是不懂

支持(0) 反对(0)

#2楼 2017-03-20 19:00 税晓豪

这位同学很是棒棒哦

2

0

谢谢博主的详解

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

- 【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码
- 【推荐】百度云“猪”你开年行大运，红包疯狂拿，低至1折
- 【推荐】专业便捷的企业级代码托管服务 - Gitee 码云
- 【活动】2019第四届全球人工技术大会解码“智能+时代”

相关博文：

- SpringMVC
- SpringMVC
- SpringMVC
- springmvc乱码解决
- SpringMVC:解决406 not acceptable

最新新闻：

- 外国专家：中国客机或需10年时间才能匹敌波音空客
 - 小米美团上市半年后，再论互联网创投模式的落幕
 - 波音答澎湃16问：737MAX系统问题在哪?坠机能否避免
 - 科技股收盘|美科技股周五全线上涨 拼多多股价涨幅超6%
 - AirPower无线充电板没了，苹果宣布取消这个项目
- » 更多新闻...

公告

个人博客：<https://www.morethink.cn/>

昵称：morethink

园龄：2年7个月

粉丝：41

关注：24

2

0

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

随笔分类

- IDEA(1)
- java(3)
- Linux(3)
- MyBatis
- Programing-Thinking(3)
- Python(2)
- Spring(4)
- SpringMVC(5)
- Struts2
- Web安全(2)
- Weekly(7)
- 服务器(9)
- 工具(4)
- 数据库(3)
- 算法(10)
- 网络(4)

随笔档案

- 2019年2月 (1)
- 2019年1月 (2)
- 2018年12月 (1)
- 2018年10月 (2)
- 2018年9月 (2)
- 2018年8月 (2)
- 2018年7月 (1)
- 2018年5月 (2)
- 2018年4月 (5)
- 2018年3月 (1)
- 2018年2月 (2)
- 2018年1月 (4)
- 2017年12月 (9)
- 2017年11月 (4)
- 2017年10月 (2)
- 2017年9月 (2)
- 2017年8月 (2)
- 2017年7月 (3)
- 2017年6月 (1)

2017年5月 (1)
2017年4月 (3)
2017年3月 (2)
2017年2月 (4)
2016年12月 (1)
2016年10月 (4)
2016年9月 (1)

积分与排名

积分 - 44588
排名 - 12261

最新评论

1. Re:学以致用:Python爬取廖大Python教程制作pdf
mark
- 瀧思葉
2. Re:puppeteer截图
mark
- 瀧思葉
3. Re:puppeteer截图
赞！
- sparkdev
4. Re:GitHub更新已经fork的项目
学习.
- NewSea
5. Re:常见web攻击总结
@牛腩真是个悲伤的故事...
- morethink

阅读排行榜

1. SpringMVC解决跨域问题(24616)
2. 常见web攻击总结(5832)
3. JSP与HTML及前后分离(5116)
4. SpringMVC实现PUT请求上传文件(4454)
5. Java解决CSRF问题(4305)
6. Java使用Openoffice将word、ppt转换为PDF(3604)
7. Java实现八大排序算法(3383)
8. Java实现单链表的快速排序和归并排序(2606)
9. Java实现二分查找算法(2067)
10. Netty接收HTTP文件上传及文件下载(1239)

评论排行榜

1. Java实现八大排序算法(11)
2. SpringMVC实现PUT请求上传文件(4)
3. SpringMVC解决跨域问题(3)
4. 常见web攻击总结(3)

2

0

- 5. puppeteer截图(2)
- 6. Java实现单链表的快速排序和归并排序(2)
- 7. 面试中的Java链表(1)
- 8. 学以致用:Python爬取廖大Python教程制作pdf(1)
- 9. GitHub更新已经fork的项目(1)

推荐排行榜

- 1. Java实现八大排序算法(60)
- 2. 常见web攻击总结(10)
- 3. Java实现单链表的快速排序和归并排序(5)
- 4. Python 豆瓣顶帖(4)
- 5. 给你的博客园图片添加标题(3)
- 6. 学以致用:Python爬取廖大Python教程制作pdf(2)
- 7. SpringMVC实现PUT请求上传文件(2)
- 8. 前端通过Nginx反向代理解决跨域问题(2)
- 9. SpringMVC解决跨域问题(2)
- 10. SpringMVC空字符串转为null(2)