搞定所有的跨域请求问题：Jsonp & Core

2017-12-01 ImportNew

（点击上方公众号，可快速关注）

来源：JavaDoop，

javadoop.com/post/cross-domain

如有好文章投稿，请点击 → 这里了解详情

网上各种跨域教程，各种实践，各种问答，除了简单的 jsonp 以外，很多说 CORS 的都是行不通的，老是缺那么一两个关键的配置。本文只想解决问题，所有的代码经过亲自实践。

本文解决跨域中的 get、post、data、cookie 等这些问题。

本文只会说 get 请求和 post 请求，读者请把 post 请求理解成除 get 请求外的所有其他请求方式。

JSONP

jsonp 的原理很简单，利用了【前端请求静态资源的时候不存在跨域问题】这个思路。

但是 只支持 get，只支持 get，只支持 get。

注意一点，既然这个方法叫 jsonp，后端数据一定要使用 json 数据，不能随便的搞个字符串什么的，不然你会觉得结果莫名其妙的。

前端 jQuery 写法

```
$.ajax({

type: "get",
```

```
url: baseUrl + "/jsonp/get",

dataType: "jsonp",

success: function(response) {

$("#response").val(JSON.stringify(response));

}

});
```

dataType: “jsonp” 。除了这个，其他配置和普通的请求是一样的。

后端 SpringMVC 配置

如果你也使用 SpringMVC，那么配置一个 jsonp 的 Advice 就可以了，这样我们写的每一个 Controller 方法就完全不需要考虑客户端到底是不是 jsonp 请求了，Spring 会自动做相应的处理。

```
@ControllerAdvice

public class JsonpAdvice extends AbstractJsonpResponseBodyAdvice {

    public JsonpAdvice(){

        // 这样如果请求中带 callback 参数，Spring 就知道这个是 jsonp 的请求了

        super("callback");

    }

}
```

以上写法要求 SpringMVC 版本不低于 3.2，低于 3.2 的我只能说，你们该升级了。

后端非 SpringMVC 配置

以前刚工作的时候，Struts2 还红遍天，几年的光景，SpringMVC 就基本统治下来了国内市场。

偷懒一下，这里贴个伪代码吧，在我们的方法返回前端之前调一下 wrap 方法：

```
@ControllerAdvice
public class JsonpAdvice extends AbstractJsonpResponseBodyAdvice {
    public JsonpAdvice(){
        // 这样如果请求中带 callback 参数，Spring 就知道这个是 jsonp 的请求了
        super("callback");
    }
}
```

CORS

Cross-Origin Resource Sharing

毕竟 jsonp 只支持 get 请求，肯定不能满足我们的所有的请求需要，所以才需要搬出 CORS。

国内的 web 开发者还是比较苦逼的，用户死不升级浏览器，老板还死要开发者做兼容。

CORS 支持以下浏览器，目前来看，浏览器的问题已经越来越不重要了，连淘宝都不支持 IE7 了~~~

- Chrome 3+
- Firefox 3.5+
- Opera 12+
- Safari 4+

- Internet Explorer 8+

前端 jQuery 写法

直接看代码吧：

```
$.ajax({
    type: "POST",
    url: baseUrl + "/jsonp/post",
    dataType: 'json',
    crossDomain: true,
    xhrFields: {
        withCredentials: true
    },
    data: {
        name: "name_from_frontend"
    },
    success: function (response) {
        console.log(response)// 返回的 json 数据
        $("#response").val(JSON.stringify(response));
    }
});
```

dataType: “json” ，这里是 json，不是 jsonp，不是 jsonp，不是 jsonp。

crossDomain: true，这里代表使用跨域请求

xhrFields: {withCredentials: true}，这样配置就可以把 cookie 带过去了，不然我们连 session 都没法维护，很多人都栽在这里。当然，如果你没有这个需求，也就不需要配置这个了。

后端 SpringMVC 配置

对于大部分的 web 项目，一般都会有 mvc 相关的配置类，此类继承自 WebMvcConfigurerAdapter。如果你也使用 SpringMVC 4.2 以上的版本的话，直接像下面这样添加这个方法就可以了：

```
@Configuration
public class WebConfig extends WebMvcConfigurerAdapter {


    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**/*").allowedOrigins("*");
    }
}
```

如果很不幸你的项目中 SpringMVC 版本低于 4.2，那么需要「曲线救国」一下：

```
public class CrossDomainFilter extends OncePerRequestFilter {
    @Override
```

```
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {

        response.addHeader("Access-Control-Allow-Origin", "*");// 如果提示 * 不行，请往下看

        response.addHeader("Access-Control-Allow-Credentials", "true");

        response.addHeader("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");

        response.addHeader("Access-Control-Allow-Headers", "Content-Type");

        filterChain.doFilter(request, response);

    }

}
```

在 web.xml 中配置下 filter：

```
<filter>

    <filter-name>CrossDomainFilter</filter-name>

    <filter-class>com.javadoop.filters.CrossDomainFilter</filter-class>

</filter>

<filter-mapping>

    <filter-name>CrossDomainFilter</filter-name>

    <url-pattern>/*</url-pattern>

</filter-mapping>
```

有很多项目用 shiro 的，也可以通过配置 shiro 过滤器的方式，这里就不介绍了。

注意了，我说的是很笼统的配置，对于大部分项目是可以这么笼统地配置的。文中类似“*” 这种配置读者应该都能知道怎么配。

如果读者发现浏览器提示不能用 ‘*’ 符号，那读者可以在上面的 filter 中根据 request 对象拿到请求头中的 referer（request.getHeader(“referer”)），然后动态地设置“Access-Control-Allow-Origin” ：

```
String referer = request.getHeader("referer");
if (StringUtils.isNotBlank(referer)) {
    URL url = new URL(referer);
    String origin = url.getProtocol() + "://" + url.getHost();
    response.addHeader("Access-Control-Allow-Origin", origin);
} else {
    response.addHeader("Access-Control-Allow-Origin", "*");
}
```

前端非 jQuery 写法

jQuery 一招鲜吃遍天的日子是彻底不在了，这里就说说如果不使用 jQuery 的话，怎么解决 post 跨域的问题。

来一段原生 js 介绍下：

```
function createCORSRequest(method, url) {
    var xhr = new XMLHttpRequest();
    if ("withCredentials" in xhr) {
```

```
        // 如果有 withCredentials 这个属性，那么可以肯定是 XMLHTTPRequest2 对象。看第三个参数
        xhr.open(method, url, true);
    } else if (typeof XDomainRequest != "undefined") {
        // 此对象是 IE 用来跨域请求的
        xhr = new XDomainRequest();
        xhr.open(method, url);
    } else {
        // 如果是这样，很不幸，浏览器不支持 CORS
        xhr = null;
    }
    return xhr;
}

var xhr = createCORSRequest('GET', url);
if (!xhr) {
    throw new Error('CORS not supported');
}
```

其中，Chrome，Firefox，Opera，Safari 这些「程序员友好」的浏览器使用的是 XMLHTTPRequest2 对象。IE 使用的是 XDomainRequest。

我想，对于 95% 的读者来说，说到这里就够了，我就不往下说了，读者如果有需要补充的，请在评论区留言。

[阅读原文]