2010

CN-CNIC-CSDB

Tan Fei

[Js 跨域访问解决方案总结]

在 js 中 XMLHttpRequest 对象是 AJAX-web 应用的核心，我们在编写 web 应用的客户端脚本的时候，可以利用该对象来进行异步请求,以提高 web 应用的用户体验。但是 XMLHttpRequest 对象进行跨域的请求的时候，会受到浏览器的限制（浏览器本身的安全考虑）。本文总结了现有的两大类解决 js 跨域访问的方法：代理和动态 script 方式，简单阐述其原理并通过相应的实验来说明。

目录

1、什么是跨域访问

浏览器对于网络上的链接会有一些限制，这些限制包括对 XMLHttpRequest 的限制。网络浏览器不会允许脚本去对它本身所在的 server 之外的 server 进行连接。

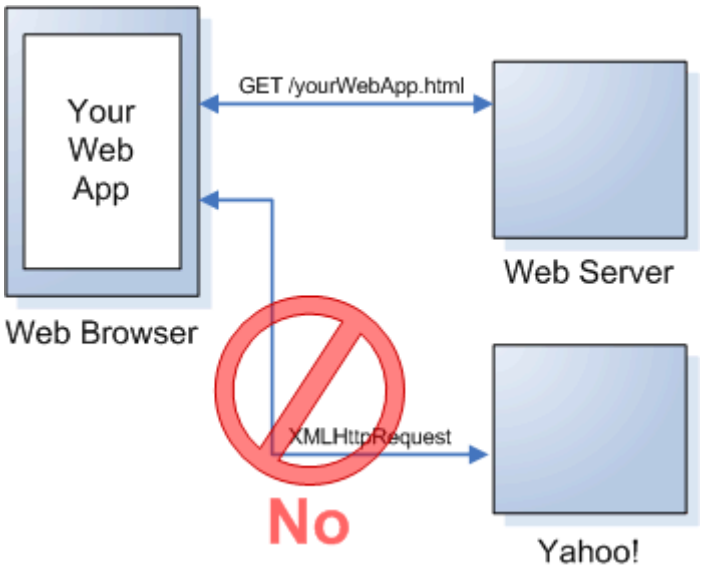

图 1 跨域访问示意图[1]

在图 1 中，来自 Web Server 的脚本在客户端的 Web Browser 中需要对其它（Yahoo）域下的 service 进行请求，那么就构成了跨域访问（对脚本所在的 server 之外的 server—Yahoo Web Service 进行了访问），浏览器会拒绝对 Yahoo Web Service 的请求。这就是 js 脚本的跨域访问问题。

这样的问题在 web 开发中并不少见，比如在使用 Yahoo 提供的各种 web service 的时候。通常有两大类方法用于解决跨域访问的问题：一类是采用代理的方式，一类是采用动态 script 标签的方式。

2、通过代理的方式来解决

2.1、原理

该方式是在服务器上安装配置一个代理，不用 XMLHttpRequest 去直接请求需要的 Web Service，而是让其先请求代理，由代理去请求 Web Service，然后再将结果返回到客户端的脚本中去处理。这样就能保重 XMLHttpRequest 请求的是

[1] 图片来源于 Yahoo Developer(http://developer.yahoo.com/javascript/howto-proxy.html)

脚本所在的 Web Server，数据也是从脚本所在的 Web Server 中获取的。整个过程如图 2 所描述的那样，如果客户端的 js 需要请求不在 Web Server 域下的第三方的 Web Service，其过程如下：

1、 由 Client 向 Web Server 发起异步请求；
2、 Web Server 收到请求之后，交给代理（Proxy）；
3、 代理（Proxy）向 Web Service 发起请求；
4、 Web Service 将接收、处理请求，并将结果返回给代理（Proxy）
5、 代理（Proxy）将结果返回给 Client

Client 的 js 只需要按照一定的规则将请求发给 Proxy 就可以了，对响应数据的接收和处理跟没有使用代理是完全一样的。

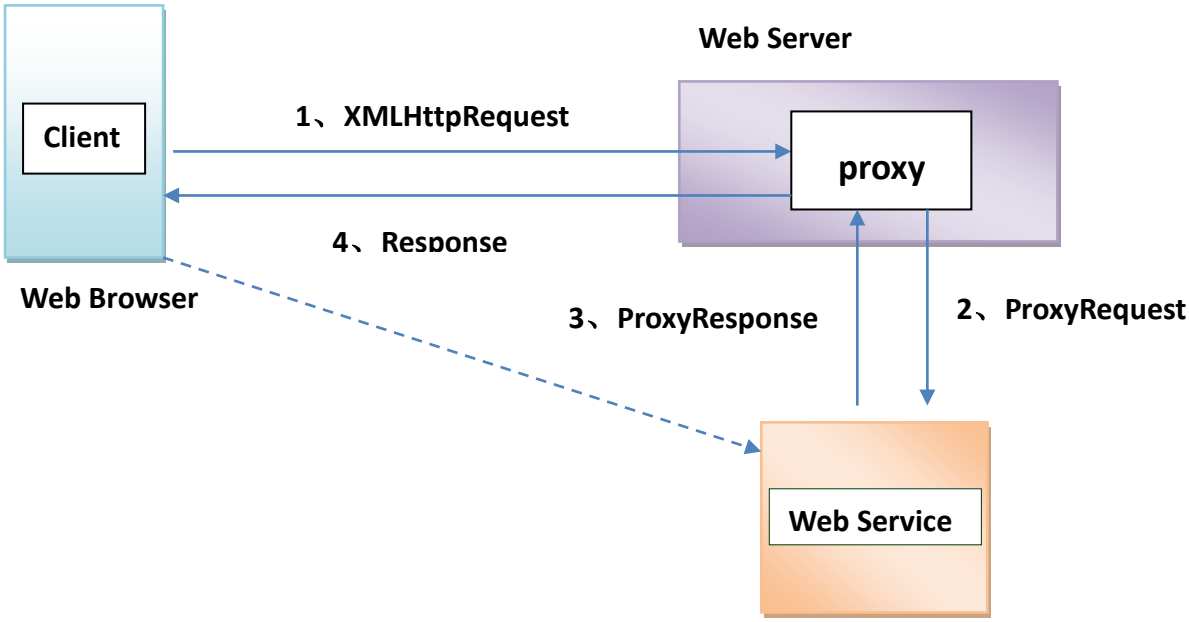

图 2 基于 proxy 的跨域访问示意图

如何在 Web Server 上构建 Proxy？通过利用某些 web 服务器和脚本语言就能构建简单实用的 Proxy。下面分别介绍利用 Apache、PHP、JSP 这三种方式，搭建一个满足 AJAX 跨域访的 Proxy。

2.2、利用 Apache 实现 JS 跨域访问

利用 Apache 的重写(mod_rewrite)或者代理模块(mod_proxy)来将跨请求转发到你需要请求的 Web Service 上面。对 Apache 进行简单的配置之后，客户端只需安照一定的方式来填写请求的 url，Apache 将自动处理好请求转发任务。

【接下来的一段内容对于非 Tomcat 用户可以忽略】

我们这边主要使用的是 Tomcat，有必要说明下 Apache 和 Tomcat 的区别：Apache 是一个 web 服务器，更是一个通用的应用平台，可以在该平台的基础之上再运行支持其它服务器端脚本语言的解释器，比如：PHP、ASP 等；Tomcat 是

一个 Web 服务器，是一个 Servlet（jsp 最终也会被编译成 Servlet）容器。通常可以将 Apache 和 Tomcat 整合到一起，由 Apache 负责处理响应的静态页面，由 Tomcat 负责处理 jsp、servlet 等动态页面，因为它们各自在处理对应的方面都具有优势。它们都是 web 服务器，不同的是 Apache 更多的是作为一个 Web 服务器的平台，而 Tomcat 侧重是作为一个 Servlet 的容器。Apache 的 APR（可移植运行时库，Apache Portable Runtime）提供了跨平台的操作系统抽象层和功能函数，提供了良好的 API，可以在此基础上开发相应的模块，来完成 web 服务器处理过程中的功能，比如 mod_mono 模块，就是在 APR 的基础上开发出来的 APS.NET 页面的解析器。因为需要用 Apache 做 Proxy，但是我们这边主要使用的是 Tomcat，但是可以通过 mod_jk 模块将它们整合到一起，具体整合方法请参照后面的参考资料【1】。连接之后的 Apache 和 Tomcat 仍然可以看着两个单独的 Web 服务器单独地在服务器上运行，Apache 在监听 80 端口，Tomcat 在监听 8080 端口，它们之间通过 Socket 通信（进程之间通信比较常用的一种）。

【穿插内容结束】

接下来阐述如何用 mod_rewrite 或者 mod_proxy 来实现 js 的跨域访问：

2.2.1 编译安装配置 Apache

到网站：http://httpd.apache.org/下载 Apache httpd 的源码，然后参照文档【2】进行编译，需要注意的是：默认的编译过程是没有将 mod_rewrite 和 mod_proxy 编译进来，需要在 configure 的时候指定需要编译的模块：

```
./configure --prefix=/usr/local/apache2 \
--enable-rewrite=shared \
--enable-speling=shared \
--enable-proxy \
--enable-proxy-http \
```

上面的命令指定了 Apache 的安装路径为：/usr/local/apache2；将 rewrite、speling 模块等编译到 apache 中。

2.2.2 利用 mod_rewrite 来实现跨域访问

在 apache 的安装目录下的 conf/httpd.conf 文件的后面添上如下几句：

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule rewrite_module modules/mod_rewrite.so
RewriteEngine On
RewriteRule ^/proxy/(.*)$ http://$1 [P,L]
```

第 1、2、3 句：告诉 Apache 在载入的时候装入相应的共享模块

第 4 句：开启重写功能

第 5 句：通过正则表达式指定重写规则。例如如下的 url：

http://localhost/proxy/192.168.56.129/OtherDomainServlet?p1=para1

将被映射为：

http:// 192.168.56.129/OtherDomainServlet?p1=para1

然后服务器就会自动去请求另外一个域下的 servlet。关于正在表达式请参考【3】，关于 RewriteRule 的参数说明请参照【4】。

2.2.3 利用 mod_proxy 来实现跨域访问

mod_proxy 是用来将一个远端服务器映射到本地服务器的 URL 空间，这种方式没有重写方式灵活，同时该方式容易受到文件.htaccess 的阻止，不建议使用。同样是在配置文件 httpd.conf 的后面加上如下：

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#Pass the call from http://www.yourserver.com/call to
#http://api.local.yahoo.com
ProxyPass /call/ http://api.local.yahoo.com/
# Handle any redirects that yahoo might respond with
ProxyPassReverse /call/ http://api.local.yahoo.com/
```

需要说明的是，如果在 Apache-2.2 上启用 mod_proxy 模块，还需要载入名为 mod_proxy_http 的模块，因为 Apache-2.2 将代理功能拆成了多个小模块。

上面的代理配置能够将如下路径

```
http://www.yourserver.com/call/js
```

映射到

```
http://api.local.yahoo.com/js
```

关于 ProxPass 和 ProxyPassReverse 的详细说明请参照【5】

2.2.4 测试

在宿主机(windows)和虚拟机(ubuntu)之间进行响应的测试：

Windows 的 web 服务器上放置一个 CrossDomainServlet 的 Servlet，其对应本地 IP 地址为：192.168.56.1，CrossDomainServlet 对应的 URL 为。

```
http://localhost:8080/CrossDomainTest/servlet/CrossDomainServlet
```

虚拟机的 IP 地址为：192.168.56.129， 虚拟机中的 Apache 已经被配置了代理，然后通过 XMLHttpRequest 向如下地址请求宿主机中的 CrossDomainServlet:

```
http://192.168.56.129/proxy/CrossDomainTest/servlet/CrossDomainSe
rvlet
```

```
http://192.168.56.129/proxy/CrossDomainTest/servlet/CrossDomainServl
et
```

通过上面的实验就能看到访问 Web Server(192.168.56.129)的同时，也能够访问 192.168.56.1 上面提供的服务。

2.3、利用 PHP 实现 JS 跨域访问

利用 PHP 脚本语言来实现一个对请求的转发和接收，该方式的详细代码请参照【6】。通过一个全局的变量(HOSTNAME)来设置转发的地址，该脚本就是将传递过来的参数组成一个新的 HTTP 请求，发送到 HOSTNAME，等待返回数据之后，再将结果返回给客户端。

```
// The full path to the PHP proxy
var url = 'http://localhost/php_proxy_simple.php?yws_path=' +
encodeURIComponent(path);
... // core xmlhttp code
xmlhttp.open('GET', url, true);

// The web services request minus the domain name
var path= 'VideoSearchService/V1/videoSearch?appid=YahooDemo&
query=madonna&results=2';
var path = HOSTNAME + path;//this is the new request url
```

该脚本支持 POST 和 GET 两种请求数据的方式。

2.4、利用 JSP 实现 JS 跨域访问

其实现原理同 PHP 的实现类似，该 JSP 的处理过程是：

1、 接收客户端请求传递的参数
2、 将参数重新组合
3、 通过 java.net.URL 构造新的 HTTP 请求
4、 读取请求返回的内容
5、 组合返回的内容再发回给客户端

下面是一个简单的实现：

```
<%@page import="java.net.*, java.util.*, java.io.*"%>
<%@page contentType="text/html;charset=gb2312"%>
<%
    final int MAX_URL_LEN = 2048;//the max url of IE

    //parameters'names
    Enumeration enu= request.getParameterNames();
    //the request url and parameters
    StringBuilder requestPara = new StringBuilder(MAX_URL_LEN);
    String requestUrl = ""; //the request url

    while(enu.hasMoreElements()){
        String token = (String)enu.nextElement();
```

```
        if(token.equals("url")){
            requestUrl = request.getParameter(token) + '?';
        }else{
            requestPara.append(token);
            requestPara.append("=");

    requestPara.append(URLEncoder.encode(request.getParameter(token)
, "UTF-8"));
            requestPara.append("&");
        }
    }

    requestPara.insert(0, requestUrl);

    URL url = new URL(requestPara.toString());
    BufferedReader br = new BufferedReader(new
                    InputStreamReader(url.openStream()));
    String line;

    //get the response and send this to the client
    while(((line=br.readLine())!=null)){
        if(line.length()>0)
            out.println(line+"a");
    }
    br.close();
%>
```

上面的代码不支持 POST，读者可以自行实行对 POST 的支持。

3、动态 script 标签的方式

3.1、原理

前面提到的跨域是因为浏览器对 XMLHttpRequest 的限制而引起的。script 标签的 src 属性是没有跨域这一说法。也就是该属性可以指向任意的 URL 地址，只要该地址是存在的。

本解决方案利用上面提到的性质，用 js 脚本向页面动态插入 script 标签，利用该标签的 src 属性向服务器端发出请求（可以带参数），src 属性指向的是一个服务器端的处理程序或脚本，例如 php、jsp 等。服务器端返回的是一段 js 代码，

当返回给浏览器的时候，js 代码就会被执行，从而带到异步通讯的目的。

通常情况下，服务器端返回的一个回调函数，该函数的参数已经在服务器端被填充(类似于 XMLHttpRequest 向服务器端请求之后，服务器端返回的 xml 流)，该回调函数（带着被填充的参数）就会被执行。

由于方式需要使用 document 对象，所以使用该方式的页面需是标准 HTML 页面。

表格 1 动态 script 方式同 Ajax 的比较[2]

	采用 script 标签	Ajax
数据获取方式	用 script 标签的 src 引用外部脚本，数据处理比采用 XMLHttpRequest 对象要高。	采用 XMLHttpRequest 对象，不同浏览器，获取该对象的方式不一样。
是否支持跨域	支持	不支持
请求方式	支持 get 方式，post 支持较为复杂，但 get 方式就能满足大部分需求。	支持 get 和 post 方式

返回的数据格式	回调函数名(json 或者 字符串)
	_onreadystatechange({"name":"zzy","phone":"8732"});
	或者
	_onreadystatechange("name==zzy!!phone==8732");
		XML 或者 json
		<Data><name>zzy</name>…</Data>
		或者
		{"name":"zzy","phone":"8732"}

3.2、举例

客户端的脚本

```
<script language="javascript">
function callback(p){
    alert("In the callback function, the arguments[0] is: " +
arguments[0]);
}
function getRequest(){
    var scriptObj = document.createElement("script");
    scriptObj.src = "http://192.168.23.67:8080/test.jsp?p1=2";
    document.body.appendChild(scriptObj);
}
</script>
```

[2] 该表格来源于搜狗实验室技术交流文档《利用 Ajaj 实现跨域访问》

服务器端的 test.jsp 内容如下:

```
<%@page contentType="text;charset=GBK"%>
<%
    String para1 = request.getParameter("p1");
    out.print("callback("+para1+");");
%>
```

当然，可以对该实现方式进行封装，让其开起来跟使用 XMLHttpRequest 没差别。在文章【7】中，就实现了类似的库。

4、其它方式

除了上面提到的方式之外，还有一些其他的可选方法，下面简单的列出，有兴趣的读者可以去进一步的了解。

- 采用数字签名的 script。如果客户端使用的是 Firefox 浏览器，你可以对你的 script 脚本进行数字签名，这样的话 Firefox 就会认为该 script 脚本是受信任的，在该受信任的脚本中，你可以让 XMLHttpRequest 去访问任何域。但是目前只有 Firefox 支持签名的脚本，所以该方式在具体使用的时候仍然会有限制。具体如何使用签名的脚本，可以查看【8】;
- 使用能够进行跨域访问的库 Sarissa。Sarissa 是一个能够进行跨域访问的 ECMAScript 库，它提供一些用于解决浏览器对 XMLHttpRequest/XML/XSLT/Xpath 的限制的解决方法。具体可参见 Sarissa 的主页【9】;
- 使用 Mixendo cross-domain XHR。通过引用一个 Mixendo cross-domain XHR 提供的 script 脚本文件，就可以用 XMLHttpRequest 对象进行跨域请求了。具体请查看【10】。

5、总结和建议

通常以下两个使用场景中，开发人员会考虑到跨域访问的问题：

场景一：在我们的 web server 上去调用第三方的服务；

场景二：我们自己写的服务，希望能让别人方便的使用该服务而不受跨域访问的影响。

在场景一中，我们能够通过在自己的 web server 中通过代理的方式来实现跨域的访问。在场景二中，我们应该考虑的是使用动态 script 的方式。

6、参考资料

【1】Linux 下 Apache 与 Tomcat 整合调试
http://blog.chinaunix.net/u1/40226/showart_1889521.html
【2】Compiling and Installing Apache：
http://httpd.apache.org/docs/2.2/install.html
【3】正则表达式 30 分钟入门教程
http://deerchao.net/tutorials/regex/regex.htm
【4】Apache 中 RewriteRule 规则参数介绍
http://www.52web.com/52article/?view-80.html
【5】apache 模块 mod_proxy 中的 ProxyPass 和 ProxyPassReverse
http://www.blogjava.net/bukebushuo/archive/2009/07/05/285557.html
【6】A simple web proxy written in PHP
http://developer.yahoo.com/javascript/samples/proxy/php_proxy_simple.txt
【7】利用 Ajaj 实现跨域访问
http://www.sogou.com/labs/report/3-1.pdf
【8】Signed Scripts in Mozilla
http://www.mozilla.org/projects/security/components/signed-scripts.html
【9】Sarissa
http://dev.abiss.gr/sarissa/
【10】Mixendo cross-domain XHR
http://dev.mixendo.com/wiki/Mixendo_XHR/