Li Dong's Blog

跨源资源共享 Cross Origin Resource Sharing(CORS)

什么是跨域？

JavaScript出于安全方面的考虑，不允许跨域调用其他页面的对象。通常来说,跨域分为以下几类：

URL	说明	是否允许
http://www.a.com/a.js http://www.a.com/b.js	同一域名下	允许
http://www.a.com/lab/a.js http://www.a.com/script/b.js	同一域名下不同文件夹	允许
http://www.a.com:8000/a.js http://www.a.com/b.js	同一域名，不同端口	不允许
http://www.a.com/a.js https://www.a.com/b.js	同一域名，不同协议	不允许
http://www.a.com/a.js http://70.32.92.74/b.js	域名和域名对应ip	不允许
http://www.a.com/a.js http://script.a.com/b.js	主域相同，子域不同	不允许
http://www.a.com/a.js http://a.com/b.js	同一域名，不同二级域名（同上）	不允许（cookie这种情况下也不允许访问）
http://www.cnblogs.com/a.js http://www.a.com/b.js	不同域名	不允许

在跨域问题上，域仅仅是通过“URL的首部”来识别而不会去尝试判断相同的ip地址对应着两个域或两个域是否在同一个ip上。

CORS

通过XHR实现Ajax通信的一个主要限制，来源于跨域安全策略。默认情况下，XHR对象只能访问与包含它的页面位于同一个域中的资源。但是合理的跨域请求对开发某些浏览器应用程序至关重要。 CORS的背后基本思想就是使用自定义的HTTP头部让浏览器与服务器进行沟通，从而决定请

求响应是应该成功还是应该失败。

比如：一个请求附加了一个额外的Origin头部，其中包含请求页面的源信息（协议、域名和端口），以便服务器根据这个头部信息来决定是否给予响应。

```
1 Origin: http://www.example.com
```

如果服务器认为这个请求可以接受，就在Access-Control-Allow-Origin头部中回发相同的源信息（如果是公共资源可以回发“*”），例如：

```
1 Access-Control-Allow-Origin: http://www.example.com
```

如果是没有这个头部，或者有这个头部但是源信息不匹配，浏览器就会驳回请求。

1. 处理一个简单请求

```
1   var url = 'http://api.bob.com/cors';
2   var xhr = createCORSRequest('GET', url);
3   xhr.send();
```

这段javascirpt代码会发送一个GET请求，会伴随一个真实的浏览器HTTP请求被发出

```
1 HTTP request:
2 GET /cors HTTP/1.1
3 Origin: http://api.bob.com
4 Host: api.alice.com
5 Accept-Language: en-US
6 Connection: keep-alive
7 User-Agent: Mozilla/5.0…
```

CORS请求总是会包含由浏览器添加的“origin” header，它的值由scheme(e.g. http/https), domain(e.g. bob.com), 和端口号（只被包含如果非default 80端口 e.g.81）组成。例如：www.bob.com 但是origin header的存在与否并不能说明它是不是一个跨域请求，因为有一些同域的请求也会包含这个header。比如，在firefox上同域请求是不包含origin的，但是在chrome和safari中put/post/delete同域请求包含origin header。浏览器并不期望同域请求的response里包含CORS response header。

所有和CORS相关的response header都是以“Access-Control-“为前缀的：

- Access-Control-Allow-Origin（必须） 这个必须包含在所有合法的跨域请求的response中，其值要么是Origin header中的值，要么就是”*“允许任何域的请求。
- Access-Control-Allow-Credentials(可选)，默认情况下cookie是不包含在CORS请求中的，使用这个header将会指明要在CORS请求中包含cookie，它的有效值是true, 如果不需要cookie, 正确的做法不是将其值设为false, 而是根本就不要这个包含header.
- Access-Control-Expose-Header(可选)，XMLHttpRequest 2 object中有一个getResponseHeader()方法，用以返回特定的response header，但是它只能得到简单的响应header,如果想让客户端访问到其他的一些header, 必须设定这个 Access-Control-Expose-Header，它的值是以逗号分隔的你想暴漏给客户端的header。

2. 处理一个不简单的请求

除了GET之外，如果想使得请求具备譬如HTTP PUT/DELETE动作,或者支持Content-Type: application/json, 就需要去处理被我们称做是not-so-simple的请求。

```
1 var url = 'http://api.bob.com';
2 var xhr = createCORSRequest('PUT', url);
3 xhr.setRequestHeader('X-Custom-Header', 'value');
4 xhr.send();
```

Preflight Request

```
1 OPTIONS /cors HTTP/1.1
2 Origin: http://api.bob.com
3 Access-Control-Request-Method: PUT
4 Access-Control-Request-Headers: X-Custom-Header
5 Host: api.alice.com
6 Accept-Language: en-US
7 Connection: keep-alive
8 User-Agent: Mozilla/5.0...
```

CORS允许使用自定义header以及出了GET和POST以外的其他方法，不同body内容类型通过一种透明的服务器验证机制被称做是Preflighted Request，当试图发起真实请求的时候，”preflight“请求就已经先发送给服务器端。这种请求使用OPTIONS方法，同时也通常会附带一些额外的header,

- Origin
- Access-Control-Request-Method 请求希望使用的http方法
- Access-Control-Request-Headers （可选）用逗号分隔的自定义header列表 例如：一个带有自定义header X-Custom-Header的PUT请求： Origin: http://api.bob.com
- Access-Control-Request-Method: PUT
- Access-Control-Request-Headers: X-Custom-Header 通过这个请求，服务器可以判断是否支持这种类型请求，服务器通过发送以下header的response进行响应：

```
1 Access-Control-Allow-Origin: http://api.bob.com
2 Access-Control-Allow-Methods: POST, GET
3 Access-Control-Allow-Headers: X-Custom-Header
4 Access-Control-Max-Age: 1728000
```

- Access-Control-Allow-Origin 同一般请求一样.
- Access-Control-Allow-Methods 以逗号分隔的允许使用的方法列表
- Access-Control-Allow-Headers 以逗号分隔的允许使用的header列表
- Access-Control-Max-Age 以秒为单位的preflight的缓存时间

一旦**Preflight Request**给予许可，浏览器将会发起真实请求。

```
1 PUT /cors HTTP/1.1
2 Origin: http://api.bob.com
3 User-Agent: Mozilla/5.0…
4 Connection: keep-alive
5 Accept-Language: en-US
6 X-Custom-Header: value
```

```
7 Host: api.alice.com
```

真实的响应：

```
1 Access-Control-Allow-Origin: http://api.bob.com
2 Content-Type: text/html; charset=utf-8
```

如果服务器想拒绝一个CORS请求，可以只返回一个不带CORS header的响应（HTTP 200）。服务器可能会拒绝不合法的preflight请求，因为没有专门的CORS header在响应中，浏览器会假定请求是不合法的，所以就不会发真实的请求。 *Preflight Request*

```
1 OPTIONS /cors HTTP/1.1
2 Connection: keep-alive
3 Accept-Language: en-US
4 Host: api.alice.com
5 Access-Control-Request-Headers: X-Custom-Header
6 Access-Control-Request-Method: PUT
7 Origin: http://api.bob.com
```

Preflight Response:

```
1 // ERROR - No CORS headers, this is an invalid request!
2 Content-Type: text/html; charset=utf-8
```

备注：$.support.cors将会被设置为true如果浏览器支持CORS。JQuery $.ajax()方法可以用于通常的XHR请求也可以用于CORS请求，但是它的实现当中不支持IE的XDomainRequest对象（IE8）,但是有JQuery的插件去补充它[http://bugs.jquery.com/ticket/8283]，