CORS

(Intercambio de recursos entre orígenes cruzados)

Motivación

Las APIs son elementos que permiten construir una rica experiencia en la red, partiendo de la creación de servicios que puedan ser consumidos por cualquier cliente. Sin embargo, su diseño suele presentar dificultades al momento de definir un mecanismo que permita limitar el acceso a los recursos por parte de clientes no autorizados.

Imagine que *unsitio.com* desea acceder a un recurso de *otrositio.com*. Esto plantea diferentes cuestiones a resolver:

- ¿Como puede *otrositio.com* determinar si el acceso por parte de *unsitio.com* es legítimo?
- Si el acceso no es legítimo ¿Como puede *otrositio.com* impedirlo?

A tal efecto, los agentes de usuario (navegadores) modernos aplican restricciones *del mismo origen* a las solicitudes de red. Estas restricciones **impiden** a una aplicación web del lado del cliente que se ejecuta **desde un origen** acceder a recursos **de otro origen**, y también limitan las peticiones HTTP no seguras que pueden ser lanzadas automáticamente hacia destinos que difieren de origen de la aplicación en ejecución. En resumen, si está navegando en *unsitio.com* y un fragmento de su código del lado de cliente desea acceder a un recurso de *otrositio.com*, simplemente no podrá hacerlo.

Definición

CORS es un modelo de seguridad definido como recomendación del W3C que permite la comunicación entre dominios desde el navegador. Construído sobre el objeto *XMLHttpRequest*, CORS brinda a los desarrolladores una forma de trabajar con los mismos modismos que se utilizan en las peticiones del mismo dominio.

Compatibilidad

Los siguientes agentes de usuario son compatibles con CORS:

- Chrome 3+
- Firefox 3.5+
- Opera 12+
- Safari 4+
- Internet Explorer 8+

En este enlace podrá consultar una lista completa de los navegadores compatibles.

Funcionamiento

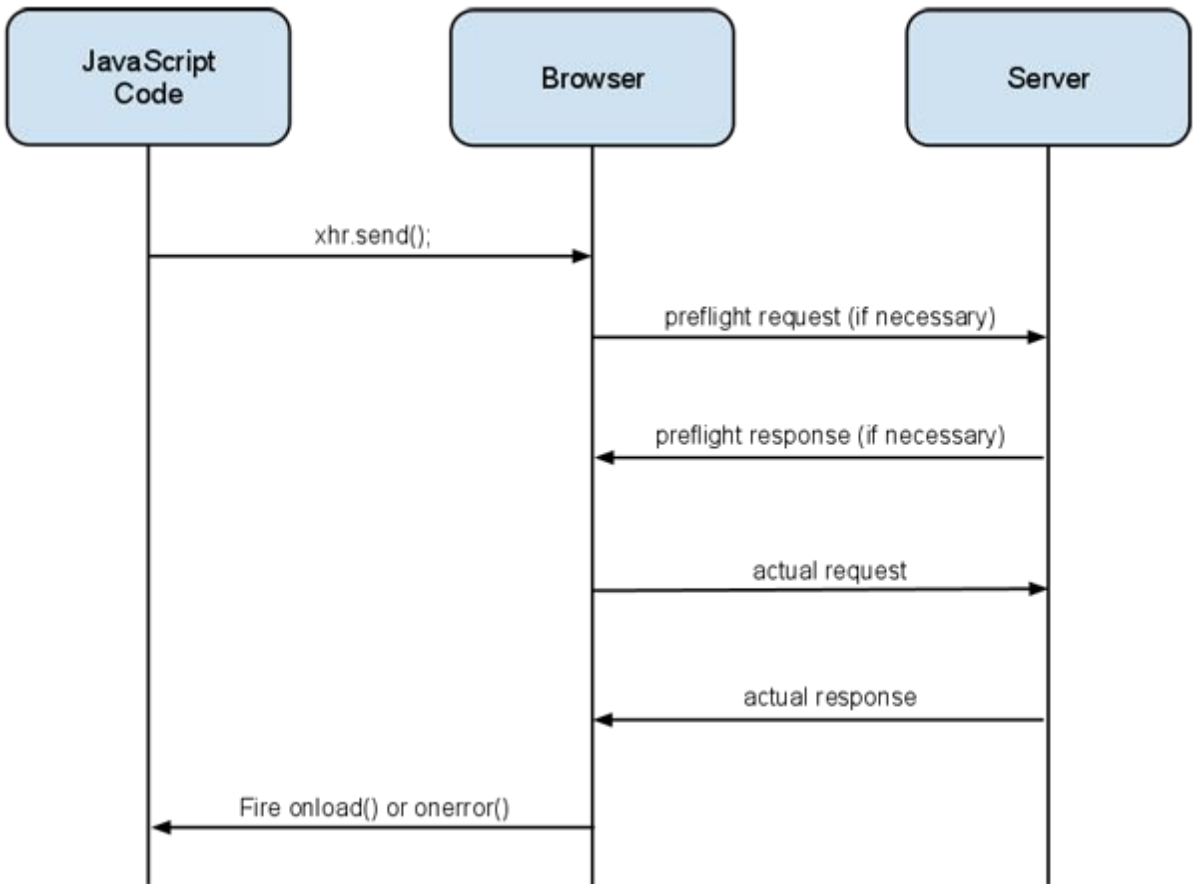
El soporte CORS requiere coordinación entre el servidor y el cliente, y su especificación extiende el modelo de comunicación entre ellos de varias maneras:

El servidor:

- Podrá incluir en la respuesta un encabezado de *Access-Control-Allow-Origin*, indicando como valor el *origen* que generó la petición, para permitir a este el acceso a los contenidos del recurso.
- Podrá determinar si una solicitud HTTP se consideró una solicitud de origen cruzado por el agente de usuario, a través de la cabecera de *Origin*.

El agente de usuario:

- Podrá descubrir a través de una solicitud previa al vuelo (*preflight*) si un recurso de origen cruzado está preparado para aceptar las solicitudes a partir de un origen determinado.
- Deberá verificar el valor con el nombre del origen que generó la solicitud.



Implementación

Cliente:

a. Utilizando Javascript puro:

```
function createCORSRequest(method, url) {
  var xhr = new XMLHttpRequest();
  if ("withCredentials" in xhr) { // el objeto es XMLHTTPRequest2 (Chrome, Firefox, ...)
    xhr.open(method, url, true);
  } else if (typeof XDomainRequest != "undefined") { // o quizá XDomainRequest (MSIE)
    xhr = new XDomainRequest();
    xhr.open(method, url);
  } else { // O ninguno de los anteriores: el navegador no soporta CORS!
    xhr = null;
  }
  return xhr;
}

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

b. Utilizando JQuery:

```
$.ajax({
  type: /* "GET", "POST", ... */, url: "http://otrositio.com",
  data: { /*Los datos que desee enviar, si los hay*/ },
  // he aquí el secreto de CORS en JQuery
  xhrFields: {withCredentials: true},
  crossDomain: true,
  success: function () { alert('¡Exito!'); },
  error: function (xhr) { alert('Error'); }
});
```

Servidor:

El siguiente algoritmo habilita el acceso desde cualquier origen. Es deseable que su aplicación verifique la legitimidad del origen y según el caso, permita o impida el acceso al recurso.

```
$origin = $_SERVER['HTTP_ORIGIN'];
header("Access-Control-Allow-Origin: $origin");
```

Recursos:

http://www.w3.org/TR/cors/
http://enable-cors.org/