Peticións AJAX cross-domain con JSONP

Sumario

A problemática de seguridade coas peticións AJAX

Unha das limitacións das peticións AJAX é a de que non se poder facer peticións AJAX a dominios alleos ó dominio no que está aloxada a nosa aplicación, é o que se coñece como "Same-origin policy", para evitar que se carguen mediante JavaScript datos potencialmente inseguros por estar aloxados noutros servidores. É dicir que os datos que cargamos no noso propio dominio son considerados seguros, pero os que están noutros dominios de Internet non.

Esta restricción pode evitarse si o dominio ó cal te conectas está configurado para aceptar conexións dende outros dominios. Isto consíguese activando a configuración "Cross-origin resource sharing" pero tamén pode solucionarse independentemente da configuración do servidor a través do uso de JSONP.

Aceptar por defecto peticións AJAX dende outros dominios en Apache (Autorización CORS)

- Si queremos que o noso dominio acepte peticións AJAX dende outros dominios teremos que configurar o servidor web para responder a esas peticións:
- Simplemente teremos que engadir a seguinte liña dentro de das seccións **<Directory>, <Location>, <Files> ou <VirtualHost>** que estarán no arquivo *.conf (httpd.conf ou apache.conf).

- Outra opción tamén sería engadi-lo dentro dun arquivo **.htaccess**.

```
Header set Access-Control-Allow-Origin "*"
```

- Para asegurarnos que está correcto executaremos os seguintes comandos:

```
# Activamos o módulo de cabeceiras en Apache (por defecto debería vir activado)
a2enmod headers

# Comprobamos o arquivo de configuración de Apache
apachectl -t

# Reiniciamos o servizo
service apache2 restart
```

Cómo é JSONP

JSON e JSONP son basicamente o mesmo, un formato lixeiro para intercambiar datos con notación de obxecto de Javascript. A notación non cambia pero si cambia o "envoltorio". En JSONP en lugar de viaxar os datos a secas como en JSON, o que viaxa é unha función xeralmente chamada "**callback**". Esa función é como un JavaScript que engloba os datos que solicitamos. De ahí que o JSONP sexa coñecido como JSON con Padding.

Exemplo de petición que devolve un JSONP: http://api.openbeerdatabase.com/v1/beers.json?callback=mifuncion&query=beer

O que estamos recibindo é unha función como a seguinte:

```
mifuncion({
  "page":1,
  "pages":1,
  "total":1,
  "beers":[
    {
      "id":87,
```

```
      "name":"Ginger Beer",
      "description":"A beer with authenticity and flavour, rich in ginger and citrus notes, spice with a malty sweetness.",
      "abv":2.5,
      "created_at":"2013-01-18T15:01:22Z",
      "updated_at":"2013-01-18T15:19:22Z",
      "brewery":{
        "id":51,
        "name":"Adnams"
      }
    }
  ]
})
```


Por qué funciona o JSONP en diferentes dominios e non JSON

Para facer posible que JSONP funcione en situaciones cross-domain emprégase unha alternativa permitida á carga de scripts nunha páxina. Os navegadores sí que aceptan a carga de código JavaScript que se incorpore coa etiqueta <script> e o atributo SRC. É o método que empregamos para cargar por exemplo unha librería de jquery dende un CDN.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

Debido a que iso non provoca ningún problema de seguridade, JSONP aproveita ese formato para acceder dese modo ó recurso solicitado inxectando ese script na páxina e executando o código para extraer os datos.

Atención: Si nos chegase un JSON normal non poderíamos executa-lo tal cual, xa que o JSON e un obxecto literal e non unha sentencia de Javascript que sexa capaz de executar o navegador, por iso non se podería meter entre as etiquetas SCRIPT da páxina.

Cómo se procesa un JSONP para extraer os datos

JQuery facilítanos esa tarefa. Simplemente teremos que indicarlle que o que estamos a recibir en un JSONP en lugar dun JSON.

Se empregamos o método $.ajax teremos que indicarlle na variable de configuración **dataType: "jsonp"** en lugar de "json".

Por exemplo empregando o método **$.ajax()** de jQuery:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery.getJSON demo JSONP</title>
<style>
img {
height: 100px;
float: left;
}
</style>
</head>
<body>
<div id="images"></div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script>
$.ajax({
url: "http://api.openbeerdatabase.com/v1/beers.json",
data: {
query: "beer"
},
type: "GET",
dataType: "jsonp",
success: function(respuesta){
console.log("Datos recibidos: ", respuesta);
$("<pre>").text(JSON.stringify(respuesta)).appendTo("body");
}
});

</script>
```

```
</body>
</html>
```

Na opción success a función que se executa contén un obxecto JavaScript nativo xa convertido para o seu uso inmediato.

Podemos facer o mesmo exemplo anterior empregando o método **$.getJSON()** de jQuery moito máis sinxelo:

Atención: é necesario engadir un novo parámetro á URL. Ese novo parámetro é **callback**. Para facer iso pódese facer con **?callback=?** se é o único parámetro da URL ou ben **&callback=?** se é o último parámetro (xa temos máis na URL) que queremos engadir. **É necesario engadi-lo para que xestione correctamente o JSONP**.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery.getJSON JSONP</title>
<style>
img {
height: 100px;
float: left;
}
</style>
</head>
<body>
<div id="images"></div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script>
// Hay que engadir á URL o parámetro ?callback=? e jQuery encárgase do resto.
$.getJSON("http://api.openbeerdatabase.com/v1/beers.json?callback=?",{query:"beer"},function(respuesta)
{
console.log("Recibes: ", respuesta);
$("<pre>").text(JSON.stringify(respuesta)).appendTo("body");
});
</script>
</body>
</html>
```

Programación no lado do servidor para responder a peticións JSONP

Os servidores tamén teñen que estar preparados para dar respostas a peticións de este tipo e para iso terán que facer algo como o seguinte exemplo en PHP que devolverá un JSONP cando se lle pasa como parámetro GET **callback**.

O parámetro a pasar na petición dependerá do servidor ó que nos conectemos, pero si todos seguen o estandar deberían usar como parámetro GET **callback**:

```
if (isset($_GET['callback']))
  echo $_GET['callback'] . '(' .json_encode($datos). ')';
else
  echo json_encode($datos);
```

--Veiga (discusión) 19:18 20 ene 2016 (CET)