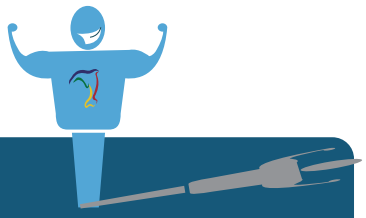
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

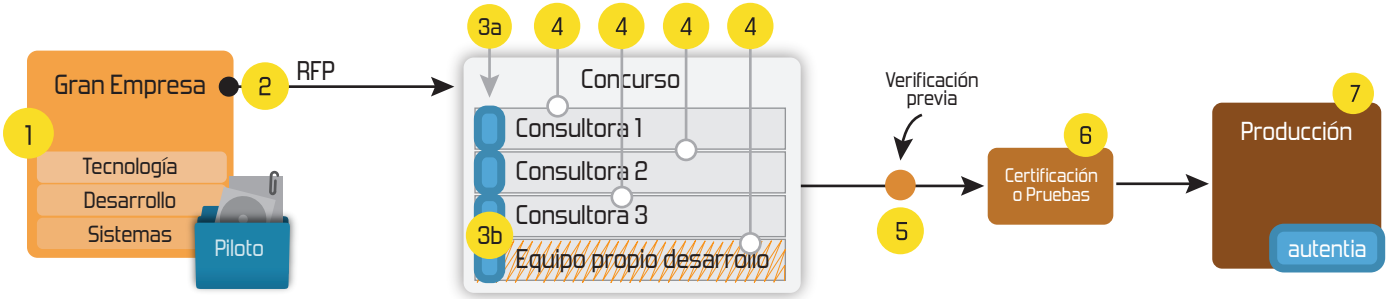
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

DESARROLLADO POR:
Miguel Arlandy Rodríguez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Ver tutoriales de Miguel Arlandy Rodríguez

Fecha de publicación del tutorial: 2011-11-28

Realizando peticiones Cross-Domain con JQuery.

0. Índice de contenidos.

1. Introducción

Los navegadores actuales contienen un mecanismo de seguridad que evita que se produzcan peticiones AJAX entre aplicaciones que residen en diferentes dominios. Sin embargo, existen ocasiones en que esta medida de seguridad puede convertirse en un problema, por ejemplo si queremos tener un servicio abierto que pueda ser consumido por diferentes aplicaciones (API abierta).

En este tutorial vamos a ver cómo realizar peticiones AJAX a diferentes dominios con ayuda de JQuery.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.2 Ghz Intel Core I7, 8GB DDR3).
- Sistema Operativo: Mac OS Snow Leopard 10.6.7

- Entorno de desarrollo: Eclipse 3.7 Indigo.
- Apache Tomcat 7.0.
- JQuery 1.7
- VirtualBox 4.0.8.
- Mozilla Firefox 8.
- Internet Explorer 8.
- Google Chrome.
- Safari 5.1.

3. El escenario.

En nuestro ejemplo vamos a suponer que necesitamos tener un servicio que recibá una petición GET y que devuelva una respuesta JSON con tres elementos:

- Nombre: una constante.
- Tipo: una constante.
- Parámetro: un parámetro recibido en la petición.

Para implementar dicho servicio construiremos un servlet que atenderá a las peticiones y devolverá el JSON de respuesta.

```
01 package com.autentia.tutoriales.prueba_crossdomain;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
05
06 import javax.servlet.ServletException;
07 import javax.servlet.http.HttpServlet;
08 import javax.servlet.http.HttpServletRequest;
09 import javax.servlet.http.HttpServletResponse;
10
11 import org.json.JSONObject;
12
13 public class SimpleServlet extends HttpServlet {
14
15     private static final long serialVersionUID = 202637587045782767L;
16
17     protected void service(HttpServletRequest request, HttpServletResponse
   response)
18             throws ServletException, IOException {
19
20         final String param1 = request.getParameter("parametro1");
21
22         response.setContentType("application/json");
23         response.setHeader("Cache-Control", "no-store");
24         final PrintWriter out = response.getWriter();
25         final String respuesta = generaJSON(param1);
26         out.write(respuesta);
27         out.flush();
28         out.close();
29     }
30
31     private String generaJSON (String parametro) {
32         JSONObject json = new JSONObject();
33         json.put("nombre", "prueba");
34         json.put("tipo", "Cross-domain");
35         json.put("parametro", parametro);
36         return json.toString();
37     }
38 }
```

Y para consumir este servicio vamos a crear un documento html, que colgaremos en el raíz de la aplicación, que realizará una petición AJAX a nuestro servicio (colgado en el contexto /prueba de la aplicación) y que muestrará los datos de respuesta. LLamaremos al documento prueba.html

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <title>Prueba Cross-Domain</title>
05 <script type="text/javascript" src="http://code.jquery.com/jquery-
   1.7.min.js"></script>
06 </head>
07 <body>
08 <h1>PRUEBA CROSS-DOMAIN CON JQUERY</h1>
09
10 <script type="text/javascript">
11
12 $(document).ready(function () {
13
14     var URL = "http://localhost:8080/prueba-crossdomain/prueba?
   parametro1=MIPARAMETRO";
15
16     $.getJSON(URL, function(datos){
17         var nombre = datos.nombre;
18         var tipo = datos.tipo;
19         var parametro = datos.parametro;
20         alert("DATOS\n\nNombre: " + nombre + "\nTipo: " + tipo +
   "\nParametro: " + parametro);
```

```
    });
});

</script>

</body>
</html>
```


A continuación, arrancamos el servidor y ejecutamos el ejemplo:

Pues como vemos, de momento, todo perfecto aunque pronto se complicará... ;)

4. El problema.

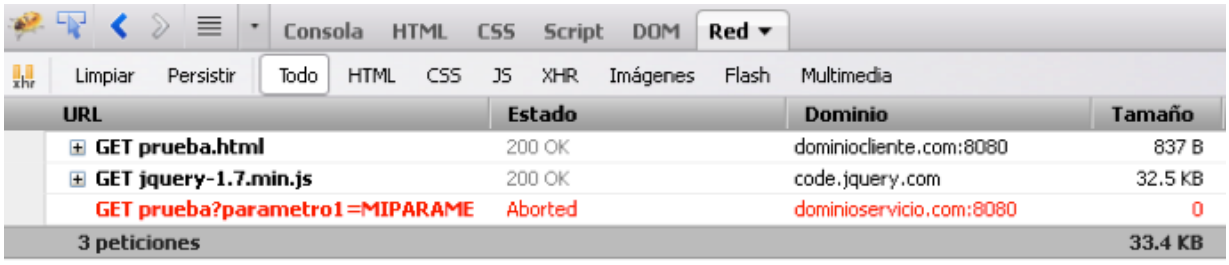
Ahora vamos a ver qué pasa si invocamos al servicio de la aplicación desde otro dominio diferente. Para ello lo primero que haremos será cambiar el dominio localhost por dominioservicio.com (fichero hosts), por tanto las peticiones que quieran consumir el servicio deberán enviarse a http://dominioservicio.com:8080/prueba-crossdomain/prueba.

Además desde una máquina virtual donde tendremos desplegada esta misma aplicación en el dominio dominiocliente.com (fichero hosts) intentaremos conectarnos a dominioservicio.com. Para ello cambiaremos en el archivo prueba.html (de la aplicación que está corriendo en dominiocliente.com) la URL a donde realizaremos la petición AJAX para que apunte a dominioservicio.com.

El cambio de la URL a la que haremos la petición desde la aplicación que está en dominiocliente.com quedaría así:

```
var URL = "http://dominioservicio.com:8080/prueba-crossdomain/prueba?
parametro1=MIPARAMETRO";
```

Arrancamos las dos aplicaciones e intentamos realizar la petición Ajax desde la aplicación cliente, que corre en dominiocliente.com, a la aplicación que corre en dominioservicio.com y observamos que no obtenemos respuesta. Abriendo Firebug observamos lo siguiente:



La petición ha sido abortada, debido a la medida de seguridad que tienen los navegadores para este tipo de peticiones Cross-Domain.

Pues bien, vamos a intentar solucionar esto... :)

5. La solución.

Como hemos visto no nos va a ser posible devolver la respuesta JSON desde el servidor ya que el

navegador bloqueará esta comunicación. Ahora bien, ¿y si en vez de devolver los datos devolvemos una invocación a una función javascript a la que se le pasen estos datos?. Con JQuery es muy sencillo realizar funciones de callback, funciones que son llamadas después de que se ejecute otra.

Vamos a modificar ligeramente la petición AJAX:

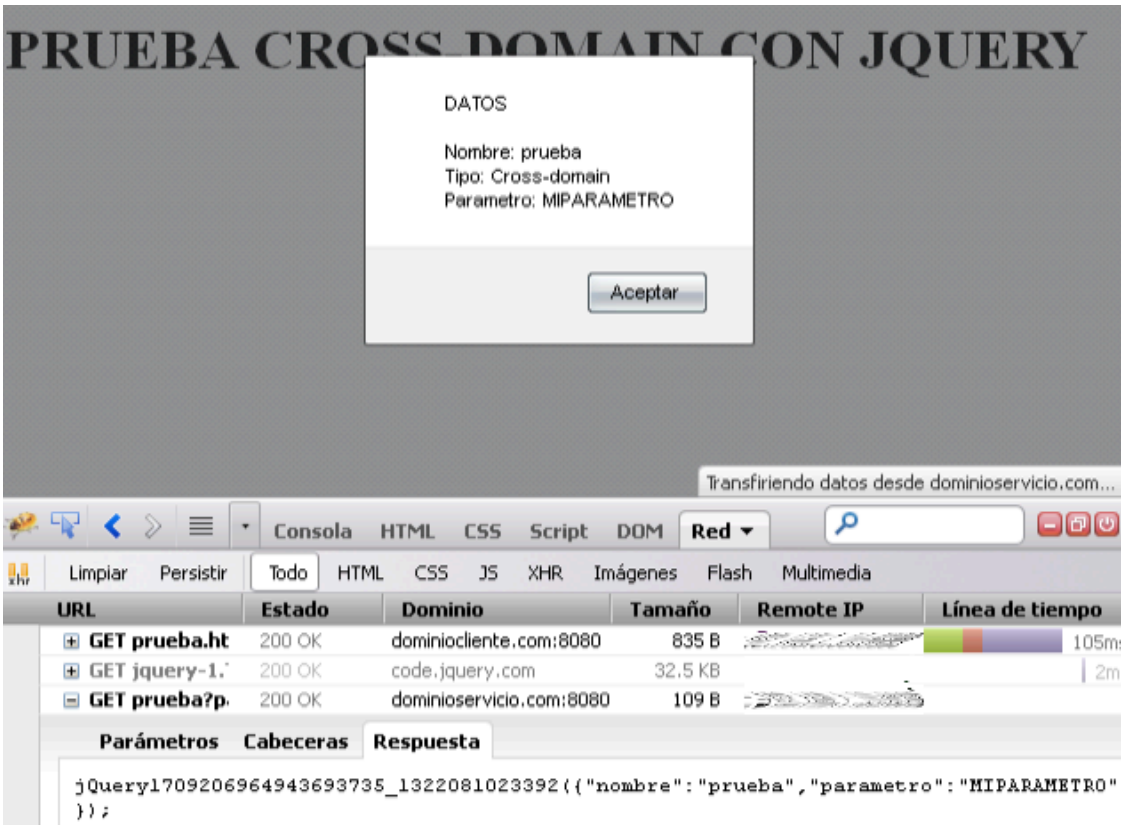
```
<script type="text/javascript">

$(document).ready(function () {

    var URL = "http://dominioservicio.com:8080/prueba-crossdomain/prueba?parametro1=MIPARAMETRO";

    URL += "&jsoncallback=?";

    $.getJSON(URL, function(datos){
        var nombre = datos.nombre;
        var tipo = datos.tipo;
        var parametro = datos.parametro;
        alert("DATOS\n\nNombre: " + nombre + "\nTipo: " + tipo + "\nParametro: " + parametro);
    });

});

</script>
```

Como vemos, hemos añadido a la petición un parámetro llamado jsoncallback con un valor igual a ?. Con esto estamos enviando información adicional al servidor para indicarle que debe delvolver los datos invocando a una función javascript pasándole como parámetros la información que queremos obtener. El nombre de esa función llegará como valor del parámetro jsoncallback y será JQuery quien se ocupe de asignarlo.

Vamos a modificar el servicio para que devuelva los datos de la forma deseada:

```
package com.autentia.tutoriales.prueba_crossdomain;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.JSONObject;

public class SimpleServlet extends HttpServlet {

    private static final long serialVersionUID = 202637587045782767L;

    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        final String param1 = request.getParameter("parametro1");

        final String callback = request.getParameter("jsoncallback");

        response.setContentType("text/javascript");
        response.setHeader("Cache-Control", "no-store");
        final PrintWriter out = response.getWriter();
        String respuesta = generaJSON(param1);

        respuesta = callback + "(" + respuesta +");";

        out.write(respuesta);
        out.flush();
        out.close();

    }

    private String generaJSON (String parametro) {
        JSONObject json = new JSONObject();
        json.put("nombre", "prueba");
        json.put("tipo", "Cross-domain");
        json.put("parametro", parametro);
        return json.toString();
    }
}
```

Como vemos, ahora tenemos en cuenta este nuevo parámetro jsoncallback. El tipo de respuesta ya no es JSON, sino javascript. En la respuesta devolvemos la invocación a una función cuyo nombre es el que hemos recibido como parámetro jsoncallback asignándole como parámetros los datos de respuesta. La invocación a esta función sería del tipo: nombredefuncion({"nombre": "prueba", "parametro" : "MIPARAMETRO", "tipo" : "Cross-domain"});



Como se puede ver en la imagen ahora si que obtenemos los datos desde un dominio distinto al que realiza la llamada. ¿Cosa de magia?. Ni mucho menos, lo que ha pasado es que desde el servidor se ha devuelto una invocación a una función (función de callback) que ya estaba preparada en el cliente (JQuery se ocupa de ello), que recibe como parámetros los datos calculados en el servidor.

6. Referencias.

- JSON con JQuery.

7. Conclusiones.

En este tutorial hemos visto que es posible realizar peticiones con AJAX a diferentes dominios sin que el navegador bloquee la comunicación. Este tipo de peticiones pueden ser de mucha utilidad para consumir servicios de terceros como API´s abiertas de determinados proveedores (arquitectura SOA).

Espero que este tutorial os haya sido de ayuda. Un saludo.

Miguel Arlandy

marlandy@autentia.com

COMENTARIOS