

¿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

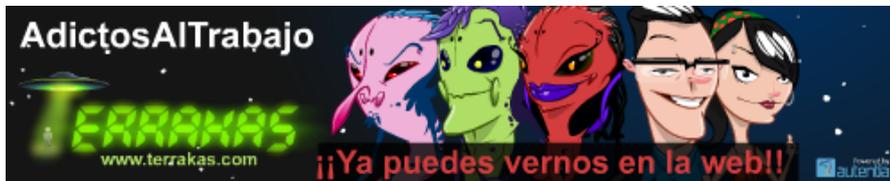
Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

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

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



E-mail:
Contraseña:

[Deseo registrarme](#)
[He olvidado mis datos de acceso](#)

[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#) [Charlas](#) [Más](#)

Estás en:

[Inicio](#) [Tutoriales](#) Realizando peticiones Cross-Domain con JQuery



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



Share |

0

[Regístrate para votar](#)

Realizando peticiones Cross-Domain con JQuery.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. El escenario.
- 4. El problema.
- 5. La solución.
- 6. Referencias.
- 7. Conclusiones.

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

Catálogo de servicios
Autentia

Últimas Noticias

 [Crónica del evento de Liferay en Madrid](#)

 [El primer capítulo de Terrakas ya está online](#)

 [Ya ha terminado la CAS 2011, ahora toca pensar cómo me gustaría que fuera la CAS 2012](#)

 [Restrospectiva, Carrera de las empresas 2011](#)

 [¿Qué ganan los demás con que tu](#)

- 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.

vayas a una conferencia?



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
18     response)
19         throws ServletException, IOException {
20
21         final String param1 = request.getParameter("parametro1");
22
23         response.setContentType("application/json");
24         response.setHeader("Cache-Control", "no-store");
25         final PrintWriter out = response.getWriter();
26         final String respuesta = generaJSON(param1);
27         out.write(respuesta);
28         out.flush();
29         out.close();
30     }
31
32     private String generaJSON (String parametro) {
33         JSONObject json = new JSONObject();
34         json.put("nombre", "prueba");
35
36         json.put("tipo", "Cross-domain");
37         json.put("parametro", parametro);
38         return json.toString();
39     }
40 }

```

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 mostrará los datos de respuesta. LLamaremos al documento prueba.html

```

01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
05 <title>Prueba Cross-Domain</title>
06 <script type="text/javascript" src="http://code.jquery.com/jquery-
07 1.7.min.js"></script>
08 </head>
09 <body>
10 <h1>PRUEBA CROSS-DOMAIN CON JQUERY</h1>
11
12 <script type="text/javascript">
13 $(document).ready(function () {
14     var URL = "http://localhost:8080/prueba-crossdomain/prueba?
15     parametro1=MIPARAMETRO";
16
17     $.getJSON(URL, function(datos){
18         var nombre = datos.nombre;
19         var tipo = datos.tipo;
20         var parametro = datos.parametro;
21         alert("DATOS\n\nNombre: " + nombre + "\nTipo: " + tipo +
22         "\nParametro: " + parametro);
23     });
24 }

```

Últimos Tutoriales

Crear gráficas de series con JFreeChart

Técnicas básicas con Mybatis

CSS3 Media Queries o cómo hacer un diseño adaptativo según el terminal

CSS Browser Selector o cómo olvidarnos de los hacks en CSS

Generar hojas de cálculo con fórmulas mediante Apache POI

Últimos Tutoriales del Autor

Crear gráficas de series con JFreeChart

Generar hojas de cálculo con fórmulas mediante Apache POI

El patrón de diseño Template Method

Clean Code: reglas y principios

Clean Code: Impresiones

Síguenos a través de:



Últimas ofertas de empleo

2011-09-08
 Comercial - Ventas - MADRID.

2011-09-03
 Comercial - Ventas - VALENCIA.

```

21     });
22     });
23     });
24     </script>
25     </body>
26     </html>

```

2011-08-19
Comercial -
Compras -
ALICANTE.

2011-07-12
Otras Sin catalogar
- MADRID.

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:

```

1 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:

URL	Estado	Dominio	Tamaño
GET prueba.html	200 OK	dominiocliente.com:8080	837 B
GET jquery-1.7.min.js	200 OK	code.jquery.com	32.5 KB
GET prueba?parametro1=MIPARAMETRO	Aborted	dominioservicio.com:8080	0
3 peticiones			33.4 KB

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:

```

01 <script type="text/javascript">
02
03 $(document).ready(function () {
04
05     var URL = "http://dominioservicio.com:8080/prueba-crossdomain/prueba?
parametro1=MIPARAMETRO";
06
07     URL += "&jsoncallback=?";
08
09     $.getJSON(URL, function(datos){
10         var nombre = datos.nombre;
11         var tipo = datos.tipo;
12         var parametro = datos.parametro;
13         alert("DATOS\n\nNombre: " + nombre + "\nTipo: " + tipo +
"\nParametro: " + parametro);
14     });
15
16 });
17
18 </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 devolver 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:

```

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         final String callback = request.getParameter("jsoncallback");
23
24         response.setContentType("text/javascript");
25         response.setHeader("Cache-Control", "no-store");
26         final PrintWriter out = response.getWriter();
27         String respuesta = generaJSON(param1);
28
29         respuesta = callback + "(" + respuesta + ");";
30
31         out.write(respuesta);
32         out.flush();
33         out.close();
34
35     }
36
37     private String generaJSON (String parametro) {
38         JSONObject json = new JSONObject();
39         json.put("nombre", "prueba");
40         json.put("tipo", "Cross-domain");
41         json.put("parametro", parametro);
42         return json.toString();
43     }
44 }

```

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"});

PRUEBA CROSS DOMAIN CON JQUERY

DATOS

Nombre: prueba
Tipo: Cross-domain
Parametro: MIPARAMETRO

Aceptar

Transfiriendo datos desde dominioservicio.com...

URL	Estado	Dominio	Tamaño	Remote IP	Línea de tiempo
GET prueba.ht	200 OK	dominiocliente.com:8080	835 B		105m
GET jquery-1.	200 OK	code.jquery.com	32.5 KB		2m
GET prueba?p.	200 OK	dominioservicio.com:8080	109 B		

Parámetros Cabeceras Respuesta

```
jQuery1709206964943693735_1322081023392({ "nombre": "prueba", "parametro": "MIPARAMETRO"
});
```

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

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Regístrate** y accede a esta y otras ventajas «

COMENTARIOS



Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

IMPULSA

Impulsores

Comunidad

[¿Ayuda?](#)

sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by [kamacracy](#)

Copyright 2003-2011 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

