

Eusko Jaurlaritzaren Informatika Elkartea Sociedad Informática del Gobierno Vasco

UDA - Utilidades de desarrollo de aplicaciones

Emulación de peticiones XHR mediante iframes

Fecha: 18/09/2012 Referencia:

EJIE S.A.

Mediterráneo, 14

Tel. 945 01 73 00*

Fax. 945 01 73 01

01010 Vitoria-Gasteiz

Posta-kutxatila / Apartado: 809

01080 Vitoria-Gasteiz

www.ejie.es



<u>UDA – Utilidades de desarrollo de aplicaciones</u> by <u>EJIE</u> is licensed under a <u>Creative Commons Reconocimiento-NoComercial-Compartirlgual 3.0 Unported License.</u>



Control de documentación Título de documento: Gestión de validaciones Histórico de versiones Código: Versión: Fecha: Resumen de cambios: 1.0.0 18/09/2012 Primera versión. Cambios producidos desde la última versión Control de difusión Responsable: Ander Martínez Aprobado por: Firma: Fecha: Distribución: Referencias de archivo Autor: Nombre archivo: Localización:



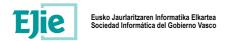
Contenido Capítulo/sección Página I Introducción Arquitectura 2 1.1 Soporte completo de operaciones HTTP 2 Soporte a la gestión de códigos de error HTTP 3



1 Introducción

En el presente documento se van a exponer un conjunto de soluciones necesarias para integrar correctamente el uso de iframes con la infraestructura de UDA.

Cabe indicar que mediante este documento no se promueve el uso de iframes, sino que se describen los componentes y configuraciones de los mismos que se deben de realizar para que su integración con el resto de componentes sea correcta.



2 Arquitectura

Aunque el uso de iframes está desaconsejado existen determinadas situaciones en las que su uso se vuelve necesario:

- Sustituir las peticiones AJAX para navegadores que no permiten realizar subida de ficheros mediante peticiones XHR.
- Realizar peticiones cross-domain.

El uso de iframes puede solventarnos estos problemas pero su uso no está exento de particularidades y incompatibilidades con la infraestructura proporcionada por UDA. Los problemas que se han de solventar son los siguientes:

- Imposibilidad de realizar las peticiones mediante métodos http diferentes a GET y POST. En los
 envíos de formularios mediante un submit (modo en el que se realiza desde un iframe), únicamente se
 puede realizar mediante uno de estos dos métodos http. Esto representa una limitación a la hora de
 utilizar el conjunto de operaciones GET, POST, PUT y DELETE utilizado en UDA en el mapeo de las
 peticiones en los controller.
- No se puede determinar el código de error de la respuesta http. Una vez obtenida la respuesta del servidor no se puede determinar el código de error http que se ha incluido en la misma. Esto presenta un problema a la hora de resolver errores como los de validación, seguridad o producidos durante el envío de ficheros, los cuales hacen uso de los códigos de estado http.

Como problema añadido, en Internet Explorer 8 no se puede acceder al contenido de la respuesta en caso de recibir un código de error http.

Para solventar estos problemas se han incluido una serie de componentes que mediante su correcta configuración permiten la correcta integración de los iframes con el resto de la infraestructura.

1.1 Soporte completo de operaciones HTTP

A continuación se va a indicar la configuración que se ha de llevar a cabo para permitir el uso de operaciones diferentes a GET y POST para el envío de formularios.

Para lograr ese objetivo haremos uso del filtro HiddenHttpMethodFilter proporcionado por *Spring*. El funcionamiento del filtro es el siguiente: La petición http se realiza de manera normal mediante GET o POST. En la petición se incluye un parámetro cuyo valor es el método http que se desea utilizar para realizar el mapeo de la petición. El filtro modifica el método de la petición para que el mapeo se realice con el indicado en el parámetro.

La configuración se debe de realizar una vez en cada unos de los War de que disponga la aplicación.

Se deberá de incluir lo siguiente en el fichero web.xml:



```
<param-name>multipartResolverBeanName</param-name>
       <param-value>multipartResolver</param-value>
    </init-param>
</filter>
<filter-mapping>
<filter-name>multipartFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
<!-- Filtro encargado de mapear correctamente peticiones que no sean GET
      y POST, al tipo indicado en el parametro _method -->
<filter>
       <filter-name>httpMethodFilter</filter-name>
       <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
       <filter-name>httpMethodFilter</filter-name>
       <url-pattern>/*</url-pattern>
</filter-mapping>
```

Se incluyen los siguientes filtros:

- <u>HiddenHttpMethodFilter</u>: Este filtro es el que posibilita realizar el mapeo de operaciones diferentes a
 GET y POST. Realiza la trasformación del método http de la petición al indicado en el parámetro
 correspondiente que se incluye en la request. El nombre del parámetro por defecto es _method pero
 puede ser personalizado mediante el parámetro methodParam en la configuración del filtro.
- <u>MultipartFilter</u>: En el caso de realizarse envío de ficheros es necesario definir este filtro **antes** que el HiddenHttpMethodFilter. Esto es debido a que este último no puede procesar peticiones *multipart* por lo que deben ser resueltas previamente.

El valor del parámetro multipartResolverBeanName corresponde al especificado en la configuración del fichero myc-config.xml

Los componentes RUP *formulario* y *upload* sd realizan internamente la gestión del parámetro *_method* en caso de que sea necesario el uso del mismo. En caso de que se desee utilizar esta solución para formularios propios de la aplicación que no hagan uso de los componentes RUP, se deberá de incluir de manera manual este parámetro en la petición. Un ejemplo sería la inclusión de un campo *hidden* como el siguiente:

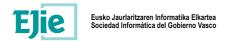
```
<input type="hidden" id="_method" name="_method" value="PUT" />
```

1.2 Soporte a la gestión de códigos de error HTTP

La siguiente configuración va a permitir la gestión de los códigos de error http enviados en la respuesta del servidor.

Como ya se ha comentado anteriormente, el uso de iframes no permite el acceso al código de estado enviado en la respuesta. Los componentes RUP de UDA interactuan con el servidor mediante estos códigos de error en los siguientes casos:

403 Forbidden: Error de seguridad producido a partir de un intento de acceso no autorizado.



- 406 Not aceptable: Errores de validación de los datos enviados.
- 413 Request Entity Too Large: Error producido al producirse un envío de un fichero que supera el límite de tamaño máximo establecido en la configuración del servidor.

La solución a esta problemática pasa por el envío de una estructura que arrope el contenido original de la respuesta junto con la información relevante del código de error http. La estructura utilizada por la mayoría de componentes *jquery* es la siguiente (como ejemplo con el envío de un código de error 406):

```
<textarea status="406" statusText="NotAcceptable">
    ["Contenido de la respuesta"]
</textarea>
```

El contenido de la respuesta se arropa mediante un tag <textarea>...</textarea> en cuyos atributos status y status Text se incluye la información del código de error.

Es importante destacar que para permitir el acceso al contenido de la respuesta cuando se utiliza el navegador IE8, se sustituye el estado http por el de 200 OK.

Para activar esta característica se debe de incluir en el fichero de configuración *mvc-config.xml* la siguiente declaración de bean:

```
<!-- Filtro utilizado para emular el comportamiento de los mensajes de error http en peticiones
realizadas desde iframes -->
<bean id="iframeXHREmulationFilter" class="com.ejie.x38.IframeXHREmulationFilter" />
```

Y en el fichero web.xml se deberá de incluir lo siguiente:

Al implementar la solución se ha tratado de lograr una que sea lo menos intrusiva con el código propio de la aplicación. De este modo con solo incluir o retirar el filtro se activa o desactiva esta característica.

Para indicar al filtro las peticiones que debe de procesar para realizar la transformación en la respuesta, se utiliza un parámetro enviado en la petición que determina el procesado de la misma. El parámetro utilizado para tal función es _emulate_iframe_http_status.



Los componentes RUP gestionan de manera automática el uso de este parámetro detectando los casos en los que es necesario para el correcto funcionamiento de la aplicación. En el caso de que sea necesario su uso por parte de los desarrolladores para utilizarlo en los desarrollos propios, se puede incluir en la petición del mismo modo que con el parámetro _method.

```
<input type="hidden" id="_emulate_iframe_http_status" name="_ emulate_iframe_http_status"
value="true" />
```