DWR: Easy Ajax for Java

Juan Fernández Rodríguez
uo67775@uniovi.es

Resumen. En este artículo describiremos brevemente que es Ajax de que tecnologías se compone y como funciona, haremos un breve resumen de un framework para Java DWR y daremos como conclusiones finales algunos pros y contras de utilizar este tipo de tecnologías.

Palabras clave: AJAX, XML, JavaScript, XHTML, CSS, DOM, ECMAScript, XSLT, XMLHttpRequest, DWR, Servlet

1 Ajax

Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones web interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta manera es posible realizar cambios sobre la interfaz sin necesidad de recargarla. Esto implica un aumento en la interactividad, velocidad y usabilidad de las aplicaciones web que usan Ajax.
Pero Ajax no constituye una tecnología en si, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente:

- XHTML y CSS para el diseño de la vista o presentación de la información
- DOM accedido con un lenguaje de scripting por parte del usuario, especialmente utilizando implementaciones de ECMAScript como JavaScript, para mostrar e interactuar dinámicamente con la información presentada
- XML y XSLT para el intercambio y manipulación de datos, aunque cualquier formato puede servir como HTML preformateado, texto plano, JSON y hasta EBML.
- XMLHttpRequest para el intercambio de datos de forma asíncrona

En las aplicaciones web clásicas los usuarios realizan peticiones HTTP esperando una respuesta por parte del servidor. El servidor recoge los datos, los procesa y finalmente devuelve la página HTTP al cliente.
En la figura siguiente podemos ver una comparativa de la interactividad del usuario con una aplicación web entre el modelo tradicional de aplicaciones web con el modelo Ajax

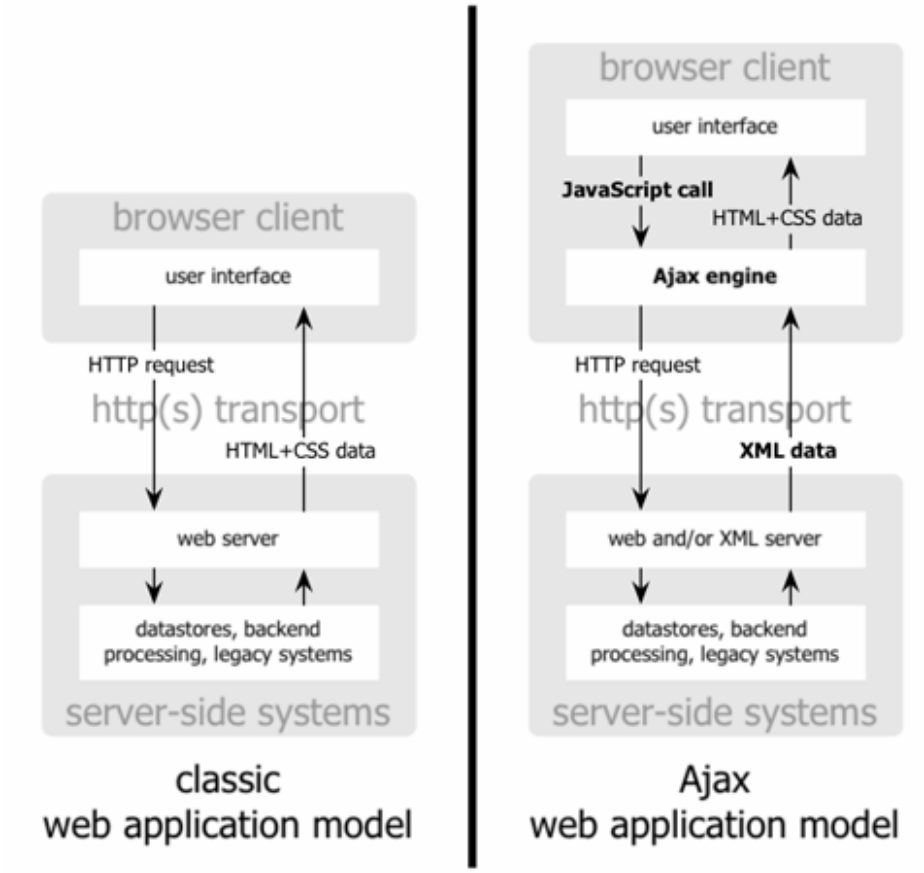

Figura 1. Modelo tradicional de aplicaciones Web (izquierda) comparado con el modelo Ajax (derecha).

1.1 Diferencias con las aplicaciones Web tradicionales

Las aplicaciones Ajax eliminan la interacción natural en una aplicación web (petición-espera-petición-espera…) introduciendo un intermediario entre el usuario que realiza las peticiones y el servidor que las procesa, el motor Ajax.
En vez de cargar la página web, al inicio de la sesión, el navegador carga el motor Ajax, escrito en JavaScript (lo que repercute en que la descarga inicial de la página sea más lenta al tener que descargarse todo el código JavaScript). Este motor es responsable tanto de renderizar la interfaz que el usuario visualiza como de comunicarse con el servidor sustituyendo a el usuario. El motor Ajax permite que la interacción del usuario con la aplicación suceda de forma asíncrona, independiente de la comunicación con el servidor. De está manera se consigue que el usuario no esté siempre esperando a que el servidor haga algo.
Además en las aplicaciones web tradicionales cada petición del usuario hacía que el servidor respondiese enviando otra página web lo que conllevaba a un gran

desperdicio de ancho de banda. En aplicaciones AJAX se pueden enviar peticiones al servidor web para obtener únicamente la información necesaria.

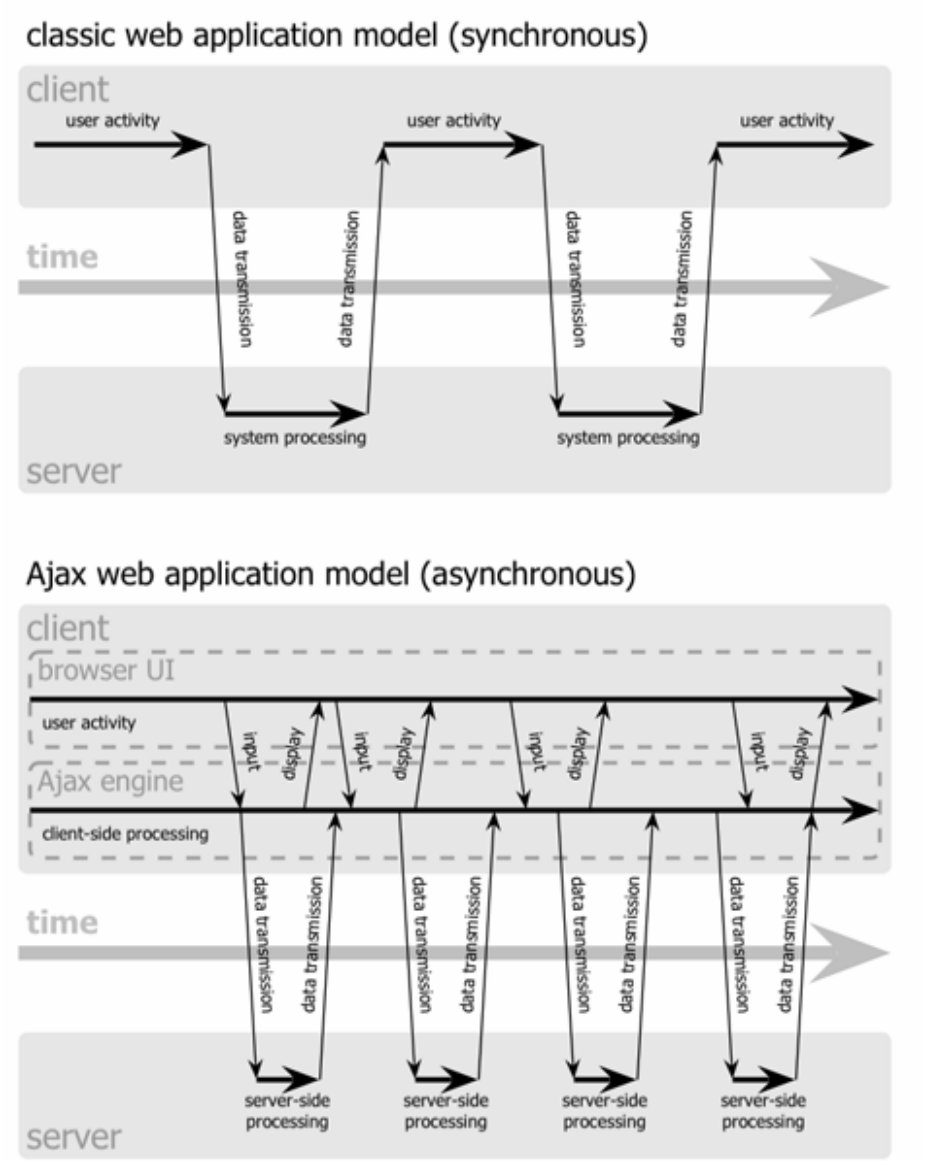

Figura 2. El patrón de interacción síncrona de una aplicación web tradicional (arriba) frente al patrón asíncrono de una aplicación Ajax (abajo).

Cada acción del usuario que normalmente generaba una petición HTTP se transforma en una llamada JavaScript al motor Ajax. Cada respuesta a la acción del usuario no

requiere una respuesta del servidor sino que el motor Ajax la maneja por si mismo. Si el motor necesita datos del servidor para responder a la solicitud del usuario, realiza dichas peticiones asíncronamente sin que ello detenga la interacción del usuario con la aplicación.

1.2 Aplicaciones que usan Ajax

Tradicionalmente se ha considerado la primera aplicación AJAX al cliente Web que tiene la herramienta de trabajo en grupo Microsoft Exchange Server aunque sin lugar a dudas Google es uno de los grandes responsables de la popularización de AJAX, al usarla en varias de sus aplicaciones, entre las que se cuentan Google Groups, Google Suggest, Google Maps y el servicio de correo electrónico gratuito Gmail. Así como también empresas en crecimiento que actualmente están desarrollando aplicaciones basadas en AJAX.

- A9, buscador de Amazon
- Flickr. álbumes de fotos online.
- Oddpost, servicio avanzado de webmail de Yahoo!
- Basecamp, servicio de gestión de proyectos diseñado por 37Signals sobre plataforma Rails.
- 24SevenOffice, ERP/CRM
- Panoramio.com, Comunidad de fotos sobre Google Maps
- Meebo, Sistema de mensajería web que utiliza los protocolos conocidos de hotmail, yahoo, jabber, etc. En una sola web
- Trabber.com, Buscador de vuelos
- Writely, Un procesador de texto web, estilo Word
- gOFFICE, Un paquete de oficina como Open office o Microsoft office, ademas libre
- Kiko, Un calendario web basado en AJAX
- Gmail, el archiconocido correo web de Google
- Openomy, Un sistema de ficheros online

2 DWR

DWR es una librería de código abierto escrita en Java que permite escribir aplicaciones web con Ajax.
DWR esta formado por dos partes:

- Un Servlet corriendo en el servidor que procesa las peticiones y envía las respuestas de retorno al navegador.
- JavaScript corriendo en el navegador que envía peticiones y puede actualizar dinámicamente el contenido de las páginas.

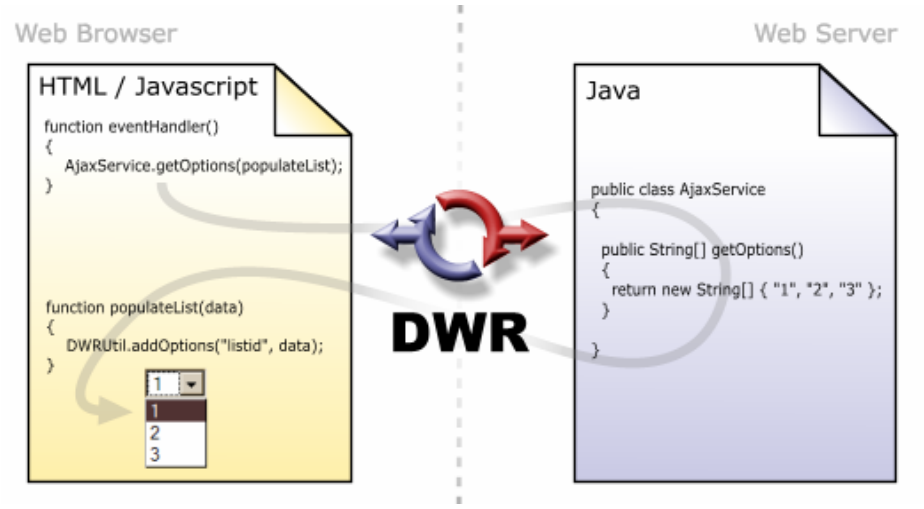

Figura 3. Esquema de funcionamiento de DWR

2.1 El lado del Servidor

DWR proporciona llamadas a métodos o funciones de invocación remota y traducción de tipos de manera similar a RPC ó RMI.
Implementa un servlet (uk.ltd.getahead.dwr.DWRServlet) que corre en el contenedor de servlets del servidor que recoge todas las peticiones HTTP del cliente, el cual se encarga de instanciar los objetos necesarios y realizar la llamada al método solicitado pasándole los parámetros enviados en la petición HTTP.
Además DWR puede realizar traducciones de varios tipos de datos: tipos básicos, colecciones, arrays y beans. DWR siempre intenta traducir el tipo Java al tipo JavaScript más parecido. Por ejemplo las colecciones se pueden traducir a arrays y los beans a arrays asociativos, siendo los nombres de las propiedades del bean los índices del array.

2.2 Configuración del servidor

El motor de DWR es el objeto DWRServlet. En esta clase se centralizan todas las posibles funcionalidades que ofrece la librería DWR: desde la generación del código

JavaScript a utilizar en el cliente, hasta el marshalling de tipos (pasando, por supuesto, por la invocación a los métodos remotos).
Como cualquier otro servlet, DWRServlet se declara y mapea en el fichero WEBINF/web.xml de la aplicación web.

```
<servlet>
<servlet-name>dwr-invoker</servlet-name>
<display-name>DWR Servlet</display-name>
<servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-
class>
<init-param>
<param-name>debug</param-name>
<param-value>true</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>dwr-invoker</servlet-name>
<url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

2.3 Exportar objetos Java

Cualquier clase puede exportar métodos utilizando DWR. Todas las declaraciones necesarias, tanto para exportar métodos, como para realizar traducciones de tipos se realizan en el fichero: dwr.xml. En este fichero se realizará toda la configuración necesaria, tanto para el lado servidor como para el cliente.

El fichero dwr tiene dos partes:

1. **Declaración de mapeo de tipos**. Se definen con la etiqueta <convert>. Cada una de las entradas, en el fichero dwr.xml, etiquetada como <convert> define un mapeo de tipo. Para cada tipo "no estándar" (siendo tipos estándar: int, boolean, float, String, Collections, etc.) se debe definir un mapeo.
2. **Declaración de clases a exportar**. Las clases (controladores DWR) cuyos métodos podrán ser llamados desde el cliente, se definen con la etiqueta <create>. En esta etiqueta se indica cómo se deben crear y manejar las instancias de objetos exportados

```
<dwr>
  <allow>
  <convert converter="bean"
   match="es.princast.framework.core.vo.PropertyBean"/>
  <create  creator="session" javascript="MunicipiosController"
class="es.princast.sampleapp.web.dwr.MunicipiosController">
  </create>
  </allow>
</dwr>
```

Para convertir un tipo Java a JavaScript se deben indicar dos parámetros (atributos de la etiqueta <convert>):
1. La clase Java a convertir.

2. El nombre del "convertidor" (Converter) que se va a encargar de realizar el marshalling. Existen varios convertidores incluidos en DWR (el más habitual es el converter bean, que transforma beans a arrays asociativos JavaScript). También es posible implementar convertidores propios.

Mediante la etiqueta <create> se definen los objetos que se van a exportar. Los atributos necesarios son:
1. El nombre completamente cualificado de la clase Java a exportar (atributo class).
2. El nombre identificativo en JavaScript de la clase (atributo javascript). Todos los métodos de la clase se exportarán a JavaScript como funciones, con el convenio de nombrado: <Nombre JavaScript>.<nombre del método>
3. El ámbito (scope) del objeto (atributo creator)

2.4 DWR en el cliente

La parte de DWR que se ejecuta en el navegador, tiene dos funciones: por un lado, sirve como stub para la realización de llamadas a los objetos del servidor, y por otro, proporciona un conjunto de funciones que facilitan la operación sobre el código DHTML de la página web.

1. **Stub del lado cliente**: Para poder utilizar las facilidades de llamada a métodos remotos en el servidor, es necesario importar, en el cliente, el motor DWR, escrito en el fichero engine.js. Este fichero está ubicado en el path: <servlet-dwr-path>/engine.js. Además, DWR proporciona para cada uno de los métodos exportados por el servidor, una función JavaScript que actúa como stub del cliente. Los stubs de cliente ocultan al desarrollador las complejidades de la comunicación cliente-servidor (especialmente, el manejo del objeto XMLHttpRequest). Las funciones JavaScript, que hacen referencia a los métodos de una misma clase, se agrupan en un mismo fichero .js. Cada uno de estos fichero se puede obtener siempre de la URL: <servlet-dwr-path>/interface/<nombreclase>.js. No es necesario escribir los ficheros stub (.js) el servlet DWRServlet se encarga de generarlos "al vuelo" utilizando la configuración que obtiene del fichero dwr.xml. Por ejemplo, si se exporta la clase:

```
public class Foo {
         public String doFoo() {
           return "foo";
         }
}
```

   En la URL: <servlet-dwr-path>/interface/foo.js se encuentra un fichero de script que contiene funciones que permitirán acceder a todos los métodos de la clase Foo. En concreto, la función " function Foo.doFoo()" (en el cliente), permite acceder al método doFoo() de la clase Foo (en el servidor).
   Para realizar llamadas a métodos del servidor, basta con utilizar las funciones de las librerias "interface". No es necesario invocar ninguna función del fichero engine.js.

2. **Utilidades para actualizar dinámicamente la página HTML**: Además de facilitar la comunicación con el servidor, DWR incluye una biblioteca de funciones que permiten manipular el código DHTML de la página para actualizar sus contenidos, de forma dinámica. Algunas de las operaciones que se incluyen son: actualizar el contenido de un DIV, cargar un "combo", actualizar determinados campos de texto, etc.

2.5 JavaScript Asíncrono

La filosofía AJAX supone que la actualización de la vista se realiza de forma asíncrona. Es decir, el usuario puede estar centrado (leyendo, operando, escribiendo, etc.) en una parte de la página y, simulatáneamente, JavaScript está actualizando otra parte (o la misma!!). Ya que el trabajo sobre la vista (página HTML) se realiza de forma asíncrona (multihilo), se debe tener en cuenta que:

1. Puede haber conflictos al actualizar simultáneamente los componentes de la página. Por ejemplo, si se envía un formulario al servidor y, antes de que este conteste, se realiza otra operación (un listado), se pueden producir inconsistencias en el estado de la aplicación.
2. Las llamadas a métodos remotos deben ser asíncronas. Es decir, no se puede llamar directamente a un método y obtener un resultado.

```
...
data = FooRemoteClass.fooMethod(); //Llamada remota con DWR
alert("Datos recibidos: "+data);
...
```

El código anterior no funcionaría, ya que se trata de una llamada síncrona. Para que una llamada remota con DWR funcione, es necesario pasarle, como parámetro, una función de callback que se encargue de procesar la respuesta del servidor. Por ejemplo:

```
...
data = FooRemoteClass.fooMethod(processData);
...
function processData(data) {
   alert("Datos recibidos: "+data);
}
```

Este código realiza, de forma asíncrona, las operaciones que se pretendían ejecutar, en el primer ejemplo, de forma síncrona.

2.6 Actualización dinámica de la vista

Aparte de simplificar la invocación de métodos remotos (en el servidor) desde clientes ligeros, DWR proporciona un conjunto de funciones que facilitarán la actualización de la página web activa (que está viendo el usuario), de forma dinámica, utilizando JavaScript.

Estas funciones se agrupan en un fichero de scripting, que se puede obtener de la URL: <servlet-dwr>/util.js. Al igual que ocurre con otros ficheros JS servidos por el Servlet DWR, el fichero util.js se genera de forma dinámica al ser solicitado (no es necesario incluir este fichero en la aplicación web).
Las funciones de utilidad van a permitir la actualización del árbol DOM de la página web activa, pero ocultando, en la medida de lo posible, las complejidades del modelo de objetos DOM. Algunas de las operaciones más comunes van a permitir:

1. Establecer un valor (o conjunto de valores) para un componente. Estas operaciones (getValue(),getValues(), setValue() y setValues()), permitirán actualizar el "valor" de un componente de la página. El significado del valor (o valores) vendrá determinado por el tipo de componente sobre el que se apliquen. Este conjunto de operaciones es aplicable a: campos de texto, radiobuttons, divs, etc. Prácticamente se puede utilizar con cualquier tipo de componente salvo tablas, imágenes y listas.
2. Añadir / Eliminar filas de una tabla, con las funciones addRow() y removeAllRows().
3. Añadir / Eliminar opciones de una lista, con las funciones addOptions() y removeAllOptions().

El siguiente fragmento de código permite actualizar el contenido de un DIV con el resultado de la invocación a un método remoto:

```
<script languaje="JavaScript">
...
FooRemoteClass.getContenidoDiv(loadDiv);
...
function loadDiv(data){
DWRUtil.setValue("divId", data);
}
</script>
<body>
...
<div id="divId"></div>
</body>
```

El código del servidor que devuelve el contenido para el DIV será:

```
public String getContenidoDIv() throws ServletException,
IOException{
return
ExecutionContext.get().forwardToString("/contenidoDiv.jsp");
}
```

4. Conclusiones

La ventaja más importante del uso de Ajax no es otro que su Interactividad: las aplicaciones se ejecutan en la máquina cliente, manipulando la página actual usando métodos DOM. Puede ser usado para multitud de tareas como actualizar y eliminar registros, expandir formularios, devolver peticiones simples de búsqueda, o editar árboles de categorías; todo sin necesidad de recargar toda la página de HTML cada vez que se realiza un cambio. Generalmente solo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas.
Entre algunas críticas que ha recibido destacan:
Usabilidad: acaba con el comportamiento normal del botón atrás del navegador. Los usuarios esperan que haciendo clic en el botón del navegador les lleve a la última página cargada, y en aplicaciones Ajax lo más seguro que esto no ocurra. Un problema relacionado es que las actualizaciones dinámicas hacen difícil el empleo de marcadores/favoritos.
Tiempos de respuesta: el intervalo entre la petición del usuario y la respuesta del servidor debe tenerse en cuenta. Sin el feedback claro al usuario, carga de datos rápida, y dirección apropiada del objeto XMLHttpRequest los usuarios pueden experimentar esperas en la interfaz, algo que pueden no esperar o comprender. Como solución se recomienda el uso de un feedback visual.
JavaScript: aunque Ajax no necesita ningún tipo de plug-in, requiere que los usuarios tengan JavaScript activado.
Accesibilidad: Los desarrolladores necesitan proporcionar opciones fallback para usuarios de otras plataformas o navegadores, ya que la mayoría de los métodos usados en AJAX utilizan rasgos solo presentes en navegadores gráficos.

Referencias

1. http://en.wikipedia.org/wiki/AJAX

2. http://www.adaptivepath.com/publications/essays/archives/000385.php

3. http://www.ajaxhispano.com/

4. http://en.wikipedia.org/wiki/Ajax_framework

5. http://getahead.ltd.uk/dwr/

6. http://weblogs.javahispano.org/page/retamar