PRÓLOGO

Cuando Jesse James Garret, en un artículo publicado en febrero de 2005, decidió dar el nombre de AJAX a la combinación de una serie de tecnologías Web existentes, no imaginó la repercusión que ese término iba a tener en la comunidad de desarrolladores.

Es incluso paradójico que una técnica de programación, basada en el uso conjunto de tecnologías existentes, haya tenido más éxito y difusión que algunas de las propias tecnologías en las que se apoya.

Pero la razón de este éxito es bien sencilla: con AJAX es posible trasladar la experiencia de usuario con las aplicaciones de escritorio a la Web, reduciendo enormemente los tiempos de espera y permitiendo que las páginas actualicen su contenido de manera casi inmediata como respuesta a las acciones de usuario.

Esta nueva filosofía de trabajo permite orientar los desarrollos Web hacia lo que se conoce como aplicaciones de una página. Hasta la llegada de AJAX, las aplicaciones Web consistían en una serie de recursos de servidor (páginas JSP, ASP, PHP, etc.), cuya función consistía en generar y enviar una nueva página al cliente como respuesta a una solicitud realizada por éste a la aplicación. Con AJAX se pretende cambiar ligeramente esa forma de trabajo, haciendo que las peticiones al servidor devuelvan únicamente datos al cliente para que éste los muestre sobre la vista actual, en vez de generar nuevas páginas completas.

OBJETIVO DEL LIBRO

El objetivo de este libro es acercar al lector esta técnica y todas las tecnologías en las que se basa, presentándole todos los elementos necesarios para que pueda desarrollar auténticas aplicaciones Web interactivas.

Dado que AJAX es una técnica de programación orientada a la capa cliente, puede ser utilizada en cualquier desarrollo Web, independientemente de la tecnología utilizada para la implementación de la capa intermedia. En nuestro caso, tal y como se refleja en el título del libro, centraremos nuestro estudio en el uso de AJAX dentro de aplicaciones Web JAVA EE, si bien los contenidos analizados en los tres primeros capítulos del libro son igualmente aplicables en cualquier otra arquitectura de desarrollo.

A QUIÉN VA DIRIGIDO

Este libro está dirigido fundamentalmente a programadores Web en entorno Java que deseen mejorar la funcionalidad de sus aplicaciones, añadiendo a sus desarrollos la potencia y capacidades que ofrece AJAX.

Puesto que el código de servidor de los ejemplos y prácticas presentados en el libro están escritos en Java, será necesario el conocimiento tanto de este lenguaje como de la tecnología JAVA EE para su comprensión y puesta en funcionamiento de los mismos. No obstante, un lector con conocimientos de otro lenguaje de programación podrá adaptar a éste la mayoría de los ejemplos y prácticas, manteniendo el código AJAX de cliente.

Así pues, cualquier persona con conocimientos de programación será capaz, utilizando este libro, de comprender la mecánica de funcionamiento AJAX y de adaptarla a su entorno particular.

ESTRUCTURA DEL LIBRO

El libro está organizado en cuatro capítulos y tres apéndices.

En el capítulo 1 se exponen los fundamentos básicos de AJAX y sus aplicaciones, analizando a través de un ejemplo el funcionamiento a nivel global de este tipo de aplicaciones.

Es durante el capítulo 2 donde se estudia con detenimiento el proceso de desarrollo y ejecución de la aplicaciones AJAX, así como las tecnologías en que se basa, abordando, además, la problemática de la compatibilidad del código para distintos tipos y versiones de navegadores. Se analizarán con detalle en este capítulo todas las propiedades y métodos que expone el objeto XMLHttpRequest.

Una vez completado el análisis del desarrollo con AJAX, el capítulo 3 nos presenta las distintas herramientas y librerías de libre distribución creadas por la comunidad de desarrolladores AJAX, orientadas a facilitar la construcción de este tipo de aplicaciones y mejorar su rendimiento.

Por último, el capítulo 4 aborda la integración de AJAX dentro de las aplicaciones JAVA EE, enfocando el análisis dentro del contexto de las aplicaciones basadas en arquitectura Modelo Vista Controlador. En esta línea, el capítulo finaliza presentando el marco de trabajo Java Server Faces y la utilización dentro del mismo de componentes gráficos con capacidades AJAX.

En cuanto a los tres apéndices finales, pretenden ser una guía de referencia para el usuario sobre las tecnologías básicas en la que se apoya AJAX, en este orden: XHTML-CSS, XML y JavaScript. Puede encontrar un desarrollo más amplio de estos temas en otros títulos publicados por esta misma editorial.

MATERIAL ADICIONAL

Los ejercicios prácticos desarrollados en los distintos capítulos del libro pueden ser descargados desde la Web de Ra-Ma. Debe dirigirse a la ficha correspondiente a esta obra, dentro de la ficha encontrará el enlace para poder realizar la descarga. Dicha descarga consiste en un fichero ZIP con una contraseña de este tipo: XXX-XX-XXXX-XXX-X la cual se corresponde con el ISBN de este libro.

Podrá localizar el número de ISBN en la página IV (página de créditos). Para su correcta descompresión deberá introducir los dígitos y los guiones.

Cuando descomprima el fichero obtendrá los archivos que complementan al libro para que pueda continuar con su aprendizaje.

Estas aplicaciones han sido creadas con el entorno de desarrollo Java NetBeans, de modo que si el lector cuenta con dicho entorno instalado en su máquina podrá abrir directamente los proyectos y probar su funcionamiento. No

obstante, cada proyecto incluye una carpeta con los códigos fuente a fin de que puedan ser utilizados en otro entorno.

CONVENCIONES UTILIZADAS

Se han utilizado las siguientes convenciones a lo largo del libro:

- Uso de **negrita** para resaltar ciertas definiciones o puntos importantes a tener en cuenta.
- Utilización de *cursiva* para nombres y propiedades de objetos, así como para presentar el formato de utilización de algún elemento de código.
- Empleo de estilo `courier` para listados, tanto de código como de etiquetado HTML. Para destacar los comentarios dentro del código se utilizará el tipo *`courier en cursiva`*, mientras que para resaltar las instrucciones importantes se empleará **`courier en negrita`**.

DIRECCIÓN DE CONTACTO

Espero que este libro resulte de utilidad al lector y le ayude a integrar en sus desarrollos todas las capacidades que ofrece esta tecnología.

Si desea realizar alguna consulta u observación, puede contactar conmigo a través de la siguiente dirección de correo electrónico:

ajms66@gmail.com

Capítulo 1

INTRODUCCIÓN A AJAX

A pesar de que AJAX sea un término de reciente aparición en el mundo del desarrollo Web, no se trata realmente de una nueva tecnología.

Las siglas AJAX, Asynchronous JavaScript And XML, hacen referencia a un conjunto de tecnologías de uso común en la Web que, combinadas adecuadamente, permiten mejorar enormemente la experiencia del usuario con las aplicaciones de Internet.

Esto, sin duda, representa una gran noticia para un programador Web, pues, al estar basado en tecnologías y estándares utilizados habitualmente, la curva de aprendizaje para incorporar AJAX al desarrollo de aplicaciones se reduce notablemente, y le evita el coste, tanto en tiempo como en esfuerzo, de aprender nuevos lenguajes, librerías o herramientas de programación.

1.1 CONTEXTO DE UTILIZACIÓN DE AJAX

Hasta la llegada de AJAX, el proceso de generación de las páginas que forman la vista de la aplicación a partir de datos extraídos del backend, y su posterior envío al cliente, se realizaba desde algún componente del servidor como una operación única e indivisible.

Esto implicaba, por ejemplo, que una vez cargada la página en el navegador, si el usuario realizaba una operación sobre ésta que requiriera la actualización de tan sólo una parte de la misma con nuevos datos del servidor, el navegador se viera obligado a lanzar de nuevo una petición al componente del

servidor para que éste volviera a realizar de nuevo todo el proceso de obtención de datos, construcción de la vista completa y su envío posterior al cliente, proceso que es conocido habitualmente como **recarga de la página**.

En un escenario donde la vista está formada por un elevado número de datos y elementos gráficos, la cantidad de información que hay que transmitir por la red durante la recarga de la página puede llegar a ser muy elevada, lo cual se traduce en varios segundos de espera para el usuario; demasiado tiempo para actualizar una pequeña porción de la página.

Son precisamente este tipo de situaciones las que AJAX puede ayudarnos a solucionar, al permitir que se actualicen ciertas partes de una página Web sin necesidad de recargar todo el contenido de la misma.

1.2 ¿QUÉ ES AJAX?

El término AJAX hace referencia a un mecanismo de combinación de tecnologías y estándares de cliente, consistente en la solicitud asíncrona de datos al servidor desde una página Web y la utilización de éstos para actualizar una parte de la misma, **sin obligar al navegador a realizar una recarga completa de toda la página**.

Además de transmitirse una menor cantidad de información por la red, al enviarse en la respuesta únicamente los datos que hay que actualizar, la realización de la operación en modo asíncrono permite al usuario de la aplicación seguir trabajando sobre la interfaz mientras tiene lugar el proceso de recuperación de los datos actualizados; como se puede sospechar, esto supone una enorme reducción del tiempo de espera para el usuario, percibiendo éste la sensación de estar trabajando sobre una aplicación de escritorio.

Pensemos, por ejemplo, en el siguiente escenario:

Se dispone de una página Web en la que además de una serie de elementos gráficos, como imágenes y menús, tenemos un formulario mediante el que vamos a solicitar una serie de datos al usuario.

Entre estos datos, necesitamos que nos proporcione la provincia y localidad a la que pertenece, para lo cual utilizaremos unos controles de tipo lista donde se podrá elegir estos datos entre un conjunto de valores dados. La idea es que, una vez elegida la provincia en la primera lista, se cargue la lista de localidades con los nombres de todas aquellas pertenecientes a la provincia elegida, facilitando así al usuario la elección de ese dato y se evitan incoherencias (figura 1).

En una aplicación Web tradicional, la carga de las localidades en la segunda lista supondría tener que volver a solicitar la página de nuevo al servidor y volver a cargar en el navegador las imágenes, menús y demás elementos que componen la vista. Aplicando la solución AJAX, un bloque de código JavaScript que se ejecute en el navegador al producirse la selección de un elemento de la lista podría encargarse de solicitar en segundo plano al servidor los datos de las localidades mientras el usuario continúa trabajando sobre la página.

Una vez recibidos los datos, otro bloque de código JavaScript se encargaría de actualizar el contenido de la lista a partir de la información recibida, manteniendo intactos el resto de los componentes de la página, lo que sin duda alguna aumentará notablemente la sensación de velocidad de la aplicación.

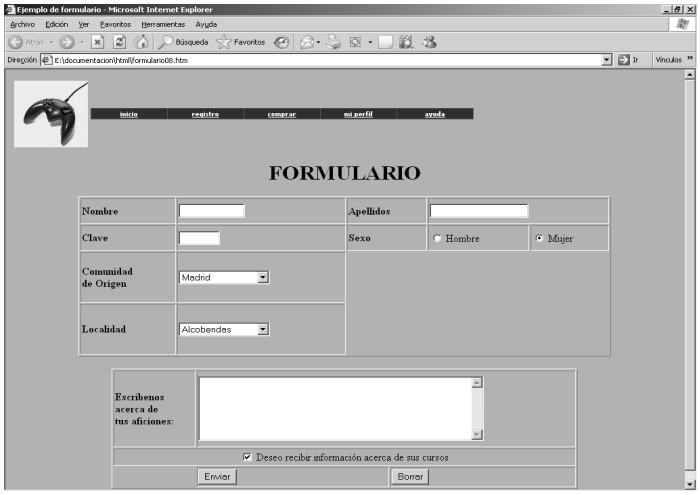

Fig. 1. *Formulario para solicitud de datos*

Otro ejemplo clásico de aplicación basada en AJAX es el Google maps. Se trata de una aplicación de tipo callejero que permite al usuario desplazarse por el plano de una localidad, ampliando y reduciendo las distintas vistas del mismo (figura 2). La imagen que forma la vista actual es realmente un puzzle de pequeñas imágenes de información distribuidas a modo de matriz bidimensional.

Cuando el usuario se desplaza por el mapa, se ejecuta en el cliente una función de JavaScript que solicita de forma asíncrona al servidor el subconjunto de imágenes necesario para actualizar la vista, evitando la recarga de todas las imágenes en la página y permitiendo al usuario seguir interactuando con la misma mientras se recuperan los datos.

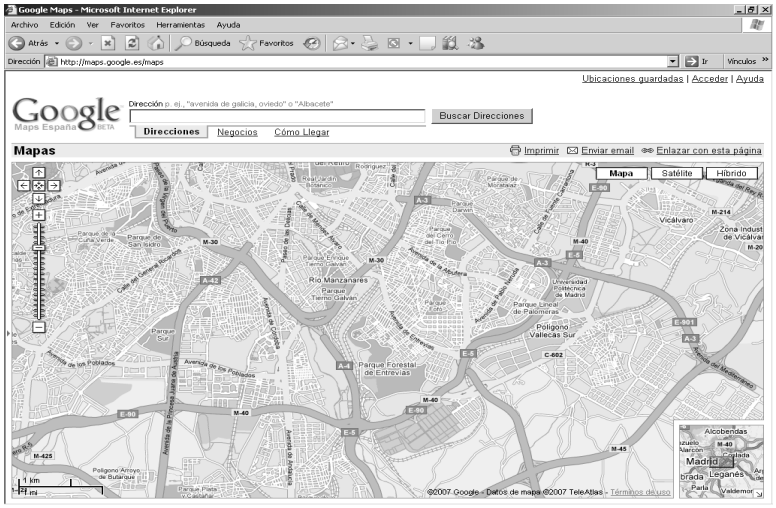

Fig. 2. *Aspecto de Google maps*

1.3 LAS TECNOLOGÍAS AJAX

Básicamente, el mecanismo de funcionamiento de una aplicación AJAX consiste en realizar peticiones HTTP de forma asíncrona al servidor desde la página Web cliente, utilizando los datos recibidos en la respuesta para actualizar determinadas partes de la página.

Para poder realizar estas operaciones, las aplicaciones AJAX se apoyan en las siguientes tecnologías:

- **XHTML y CSS**. La interfaz gráfica de una aplicación AJAX es una página Web cargada en un navegador. XHTML y CSS son los dos estándares definidos por el W3C para la construcción de páginas Web; mientras que XHTML se basa en la utilización de un conjunto de marcas o etiquetas para la construcción de la página, el estándar CSS define una serie de propiedades de estilo que pueden aplicarse sobre las etiquetas XHTML, a fin de mejorar sus capacidades de presentación.

- **JavaScript**. Las peticiones HTTP que lanza la página Web en modo asíncrono al servidor son realizadas por un bloque de script implementado con cualquier lenguaje capaz de ser interpretado por el navegador; este lenguaje es, en la gran mayoría de los casos, JavaScript.

Los bloques de código JavaScript que forman la aplicación AJAX se encuentran embebidos dentro de la propia página Web, o bien en archivos independientes con extensión .js que son referenciados desde ésta. Estos bloques de código son traducidos y ejecutados por el navegador , bien cuando se produce la carga de la página en él o bien cuando tiene lugar alguna acción del usuario sobre la misma. Mediante código JavaScript las aplicaciones AJAX pueden realizar solicitudes al servidor desde la página Web, recuperar los datos enviados en la respuesta y modificar el aspecto de la interfaz.

- **XML**. Aunque podría utilizarse cualquier otro formato basado en texto, cuando la aplicación del servidor tiene que enviar una serie de datos de forma estructurada al cliente, XML resulta la solución más práctica y sencilla, ya que podemos manipular fácilmente un documento de estas características y extraer sus datos desde el código JavaScript cliente.

- **DOM**. El Modelo de Objeto de Documento (DOM) proporciona un mecanismo estándar para acceder desde código a la información contenida en un documento de texto basado en etiquetas.

  Mediante el DOM, tanto páginas Web como documentos XML pueden ser tratados como un conjunto de objetos organizados de forma jerárquica, cuyas propiedades y métodos pueden ser utilizados desde código JavaScript para modificar el contenido de la página en un caso, o para leer los datos de la respuesta en otro.

- **El objeto XMLHttpRequest**. Se trata del componente fundamental de una aplicación AJAX. **A través de sus propiedades y métodos es posible lanzar peticiones en modo asíncrono al servidor y acceder a la cadena de texto enviada en la respuesta**. Para poderlo utilizar, deberá ser previamente instanciado desde la aplicación.

Las anteriores tecnologías y componentes constituyen el núcleo fundamental de AJAX, sin embargo, estas aplicaciones se apoyan también para su funcionamiento en los siguientes estándares y tecnologías:

- **HTTP**. Al igual que el propio navegador, el objeto XMLHttpRequest utiliza el protocolo HTTP para realizar las solicitudes al servidor, manejando también las respuestas recibidas mediante este protocolo.

- **Tecnologías de servidor**. Una aplicación AJAX no tendría sentido sin la existencia de un programa en el servidor que atendiera las peticiones enviadas desde la aplicación y devolviera resultados al cliente.

  Las mismas tecnologías que se utilizan para crear procesos en el servidor en una aplicación Web convencional, como Java EE, ASP.NET o PHP, pueden utilizarse igualmente en el caso de que la capa cliente esté basada en AJAX. Dado que éste es un libro enfocado a la utilización de AJAX con aplicaciones Java EE, todos los ejemplos de código de servidor que se presenten serán implementados con esta tecnología.

1.4 PRIMERA APLICACIÓN AJAX

Seguidamente, vamos a desarrollar una sencilla aplicación AJAX de ejemplo que nos va a servir para comprender la mecánica de funcionamiento de este tipo de programas y comprender el papel que juegan las distintas tecnologías mencionadas anteriormente.

La aplicación consistirá en una página Web, mediante la que se solicitará al usuario la elección del título de un libro entre una serie de valores presentados dentro de una lista desplegable. Cuando el usuario seleccione uno de estos títulos, se deberá mostrar en la parte central de la página un texto descriptivo sobre el libro elegido (figura 3).

Fig. 3. *Aspecto de la página de ejemplo*

1.4.1 Descripción de la aplicación

La vista de la aplicación está formada por una página XHTML. A fin de simplificar el desarrollo, los títulos de los libros serán incluidos en el propio código XHTML de la página.

Por otro lado, la lógica de servidor será implementada mediante un servlet que mantendrá en un array los textos asociados a cada libro. Cuando el usuario seleccione un título en la lista que aparece en la página XHTML, se invocará al servlet pasándole como dato el índice del libro seleccionado, valor que será utilizado para localizar el texto asociado en el array y enviárselo al cliente.

A continuación, vamos a analizar los distintos bloques que forman la aplicación a fin de comprender su funcionamiento.

1.4.2 Código de servidor

Como acabamos de indicar, el código de servidor de esta aplicación está implementado mediante un servlet que genera una respuesta a partir de la información contenida en un array. Esta respuesta estará formada por un texto plano que se corresponderá con la descripción asociada al título cuyo índice ha sido enviado en la petición.

El siguiente listado corresponde al código de este servlet:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ComentarioLibro extends HttpServlet {
    protected void service (HttpServletRequest request,
                            HttpServletResponse response)
                                 throws ServletException,
                                  IOException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        //array de descripciones
        String [] comentarios=
        {"Requiere conocimientos básicos de programación
                                       orientada a objetos",
        "Puede construir fácilmente aplicaciones para
                                       la Web",
        "Aprenderá rápidamente los principales trucos
```

```
                                        de Ajax",
            "Introduce las principales tecnologías de
                                        la plataforma"};
            int sel;
            //tit es el nombre del parámetro enviado en
            //la petición
             sel=Integer.parseInt(request.getParameter("tit"));
            out.println(comentarios[sel]);
            out.close();
        }
}
```

1.4.3 La vista cliente

La vista cliente está formada por una página XHTML estática. Además de las etiquetas, en esta página se incluye el código JavaScript que implementa la funcionalidad cliente de la aplicación AJAX.

Para facilitar la comprensión de la vista cliente, el siguiente listado muestra el contenido HTML de la página, omitiéndose el código JavaScript:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
                                Transitional//EN">
<html>
  <head>
    <title>Página de información sobre libros</title>
    <script language="JavaScript">
        //aquí iría el código JavaScript para AJAX
    </script>
  </head>
  <body bgcolor="#8beff1">
     <center>
         <table cellspacing="30">
             <tr>
                 <td><img src="libros.jpg" /></td>
                 <td><h1>Información sobre libros</h1></td>
             </tr>
         </table>
      <table width="30%">
          <tr><td><b>Elija título de libro:</b></td></tr>
          <tr>
              <td >
```

```
                    <select id="titulo"
                            onchange="buscarComentario();">
                        <option>- Seleccione título -</option>
                        <option value="0">Programación con C#
                        </option>
                        <option value="1">ASP.NET</option>
                        <option value="2">AJAX en un día
                        </option>
                        <option value="3">JAVA EE</option>
                    </select>
                </td>
            </tr>
            <tr>
                <td>
                    <br/><br/>
                    <div id="comentario"></div>
                </td>
            </tr>
        </table>
       </center>
    </body>
</html>
```

Como se puede ver, la lista con los títulos de los libros es generada mediante un control XHTML de tipo *select*. Cada elemento <option> del control incluye en su atributo *value* un número que se corresponde con la posición dentro del array del servidor donde se encuentra el texto descriptivo asociado a ese título.

Así mismo, podemos comprobar la existencia de una etiqueta de tipo , cuyo identificador es "comentario", que será utilizada para mostrar el texto descriptivo del título seleccionado.

En cuanto a la conexión entre la vista XHTML y el código script de cliente, observamos que el control <select> incluye un manejador de evento *onchange* con una llamada a la función de JavaScript *buscarComentario()*. Esta función estará definida dentro del bloque <script> de la página y será invocada en el momento en que se produzca la selección de un elemento dentro de la lista, debiendo ser la encargada de poner en marcha todo el mecanismo de ejecución AJAX en el cliente.

1.4.4 Código de script de cliente

Todo el proceso de ejecución de una aplicación AJAX está controlado por código JavaScript que se ejecuta en el cliente. A grandes rasgos, este código debe encargarse de solicitar unos datos al servidor y, una vez obtenidos, insertarlos en la parte de la página destinada a mostrar los resultados.

Esta sencilla aplicación, pero práctica a efectos didácticos, está estructurada en tres funciones cuyo código se muestra en el siguiente listado y que serán explicadas en los siguientes subapartados:

```
var xhr;
function buscarComentario(){
      //creación del objeto XMLHttpRequest
      try{
           xhr=new ActiveXObject("Microsoft.XMLHttp");
           enviaPeticion();
      }
      catch(ex){
           alert("Su navegador no soporta este objeto");
      }
}
function enviaPeticion(){
      //configuración de los parámetros de la
      //petición y envío de la misma
      var lista=document.getElementById("titulo");
      var tit=lista.options[lista.selectedIndex].value;
      xhr.open("GET","comentariolibro?tit="+tit,true);
      xhr.onreadystatechange=procesaDatos;
      xhr.send(null);
}
function procesaDatos(){
      //tratamiento de la respuesta
      if(xhr.readyState==4)
                document.getElementById("comentario").
                    innerHTML="<i>"+xhr.responseText+"</i>";
}
```

1.4.4.1 CREACIÓN DEL OBJETO XMLHTTPREQUEST

La función *buscarComentario()* representa el punto de arranque de la aplicación AJAX. La primera tarea que realiza esta función es la creación del

objeto XMLHttpRequest, cuya instancia es almacenada en la variable pública "xhr" a fin de que pueda ser utilizada por el resto de las funciones del programa.

La instrucción de creación del objeto es encerrada dentro de un bloque *try*, de modo que si se produce un error en esta línea se mostrará un mensaje de aviso al usuario y no se ejecutará el resto del código.

Tras la creación del objeto XMLHttpRequest se invoca a la función *enviaPeticion()* para continuar con el proceso de ejecución de la aplicación.

1.4.4.2 ENVÍO DE LA PETICIÓN HTTP

La función *enviaPeticion()* se encarga de realizar la petición de datos al servidor utilizando el objeto XMLHttpRequest. Aunque más adelante analizaremos con detenimiento este objeto, vamos a comentar brevemente las propiedades y métodos que se utilizan en esta función, ya que representan la parte más importante de la aplicación:

- **open()**. Mediante este método el cliente se prepara para realizar la petición al servidor. Puede recibir hasta cinco parámetros, siendo los tres primeros los más importantes; en este orden:
  - **método de envío**. Método de envío de datos utilizado por la petición HTTP. Puede ser "GET" o "POST".
  - **url**. Este parámetro contiene una cadena de caracteres que representa la URL del recurso que se solicita en la petición. En este caso se trata de la URL del servlet "comentariolibro", al que se pasa un parámetro llamado "tit" que contiene el código asociado al título del libro seleccionado y que es obtenido con las dos primeras instrucciones de la función.
  - **modo asíncrono**. Indica si la operación se realizará en modo síncrono (*false*) o asíncrono (*true*).
- **onreadystatechange**. Propiedad en la que se indica la función JavaScript que va a procesar los datos de la respuesta.
- **send()**. El método *open()* no realiza la petición al servidor, simplemente prepara al cliente para ello, es la llamada al método *send()* la que lleva a cabo el proceso según los parámetros

configurados en *open()*. Posteriormente, veremos el significado del parámetro que se pasa al método, en este ejemplo su valor es *null*.

1.4.4.3 PROCESAMIENTO DE LA RESPUESTA

La última de las funciones de la aplicación de ejemplo es *procesaDatos()*. Esta es la función indicada en la propiedad *onreadystatechange*, y es la encargada de la recepción y procesamiento de los datos enviados por el servidor en la respuesta.

Cuando la petición se realiza en modo asíncrono, como es el caso de esta aplicación, la función *procesaDatos()* es invocada varias veces por el objeto XMLHttpRequest desde el momento en que se realiza la petición hasta que la respuesta ha sido recibida en el cliente.

Normalmente, al programador sólo le interesará que se ejecute su código una vez que se haya completado la recepción de los datos en cliente. Para comprobar en qué momento estarán disponibles estos datos, utilizamos la propiedad *readyState* del objeto XMLHttpRequest.

Una vez disponibles, el acceso a los datos de la respuesta enviada por el servlet se lleva a cabo utilizando las distintas propiedades del objeto XMLHttpRequest, según sea el formato que se haya dado a los datos de dicha respuesta. En este caso, dado que el servlet envía los datos como texto plano, debemos utilizar la propiedad *responseText* para recuperar la información, información que es presentada dentro del elemento de la página XHTML utilizando las propiedades y métodos DOM:

```
document.getElementById("comentario").
              innerHTML="<i>"+xhr.responseText+"</i>";
```

Como se puede apreciar, el método DOM *getElementById()* nos devuelve el objeto HTML cuyo identificador se le proporciona como parámetro. A partir de la propiedad *innerHTML* del objeto podemos acceder a su contenido HTML, pudiendo así añadir información en su interior. Además de esta propiedad, las etiquetas HTML proporcionan otras propiedades que permiten alterar las características de la etiqueta, como, por ejemplo, *style*, que nos da acceso a sus propiedades de estilo.

Este Capítulo ha servido para ilustrarnos los fundamentos de la implementación de una aplicación AJAX básica. En posteriores capítulos profundizaremos en el estudio de las diferentes tecnologías y componentes implicados en su desarrollo.

1.5 APLICACIONES AJAX MULTINAVEGADOR

Si la aplicación de ejemplo que hemos creado anteriormente la ejecutamos con un navegador distinto a Internet Explorer, nos encontraremos con una desagradable sorpresa y es que la aplicación no funcionará, mostrándose un cuadro de diálogo con el mensaje "Su navegador no soporta este objeto" al realizar la selección de un elemento en la lista (figura 4).

Fig. 4. *Error al ejecutar la aplicación en Firefox*

El problema se encuentra en la función que crea el objeto XMLHttpRequest, pues es precisamente esta instrucción la que provoca el error, dado que el objeto **ActiveXObject únicamente es reconocido por el navegador Internet Explorer**.

El objeto ActiveXObject es utilizado por Internet Explorer para crear instancias de componentes COM registrados en la máquina cliente, a partir de la cadena de registro de los mismos. En el caso de XMLHttpRequest, el componente COM que lo implementa es Microsoft.XMLHttp, aunque es muy posible que el cliente disponga además de versiones más modernas de éste, como la MSXM2.XMLHttp, la MSXM2.XMLHttp.3.0 o incluso la MSXM2.XMLHttp.5.0.

El resto de los navegadores más comúnmente utilizados por los usuarios de Internet, como Opera, Firefox, Safari e incluso Internet Explorer a partir de la versión 7, implementan XMLHttpRequest como un objeto nativo, lo que significa que su creación se debe llevar a cabo instanciando directamente la clase mediante el operador new:

```
xhr=new XMLHttpRequest();
```

Ésta es la manera en que el W3C, organismo encargado de regular las especificaciones para la Web, recomienda que debe crearse este objeto. Así pues, es de suponer que todas las futuras versiones de navegadores que se desarrollen en adelante soporten este mecanismo de creación.

1.5.1 Compatibilidad de código en todos los navegadores

Mientras sigan utilizándose navegadores Internet Explorer versión 6 y anteriores, las aplicaciones AJAX están obligadas a incluir cierta lógica que garantice la compatibilidad del código en todos los tipos de navegadores, lo que implica combinar en una misma función las dos formas analizadas de crear el objeto XMLHttpRequest.

Para ello, bastará con comprobar qué tipo de objeto nativo soporta el navegador donde se está ejecutando el código, ActiveXObject o XMLHttpRequest. Esto puede realizarse consultando las propiedades del mismo nombre del objeto window, ya que cuando un navegador soporta uno de estos objetos lo expone como una propiedad de window.

En el caso de IE6 y versiones anteriores la expresión:

window.ActiveXObject

contendrá un valor distinto de *null* y *undefined*, por lo que al ser utilizada como condición en una instrucción condicional de tipo *if* **el resultado de la evaluación será *true***. Lo mismo puede decirse para el resto de navegadores de la expresión:

window.XMLHttpRequest

Así pues, la estructura de código que habría que utilizar para garantizar la compatibilidad de la aplicación AJAX en todos los navegadores debería ser:

```
if(window.ActiveXObject){

    //Navegador IE o versiones anteriores

}

else if (window.XMLHttpRequest){

    //Resto de navegadores

}
```

else{

//Navegador sin soporte AJAX

}

En ciertas versiones del navegador, pudiera ocurrir que, aun implementando XMLHttpRequest como objeto nativo, no estuviera expuesto como propiedad del objeto window. En este caso habría que comprobar si el navegador soporta el objeto a través del operador *typeof* de JavaScript, en cuyo caso la expresión:

typeof XMLHttpRequest

devolverá un valor distinto de *undefined*.

Ahora podemos reescribir la función *buscarComentario()* del ejercicio anterior para garantizar la compatibilidad del código con prácticamente todas las versiones de navegadores utilizados por los usuarios de Internet, quedando como se indica en el siguiente listado:

```
function buscarComentario(){
     if(window.ActiveXObject){
         //navegador IE
          xhr=new ActiveXObject("Microsoft.XMLHttp");
     }
     else if((window.XMLHttpRequest) ||
                    (typeof XMLHttpRequest)!=undefined){
         //navegadores Firefox, Opera y Safari
          xhr=new XMLHttpRequest();
     }
     else{
         //navegadores sin soporte AJAX
          alert("Su navegador no tiene soporte para AJAX");
          return;
     }
     enviaPeticion();
}
```

1.6 MANIPULAR DATOS EN FORMATO XML

En el ejercicio AJAX de ejemplo que acabamos de desarrollar se ha utilizado la propiedad *responseText* del objeto XMLHttpRequest para acceder a la cadena de texto recibida en la respuesta del servidor.

Cuando estos datos van a ser tratados como texto plano el uso de esta propiedad resulta adecuado, sin embargo, hay muchas otras circunstancias en las que la información enviada en la respuesta debe tener cierta estructura, a fin de que los datos que la componen puedan ser recuperados de una forma individualizada por el código de script del cliente. En estos casos, el servidor **debe enviar los datos al cliente en un formato que permita su manipulación, este formato es XML**.

Cuando el servidor envía los datos como un documento XML bien formado, puede utilizarse la propiedad *responseXML* del objeto XMLHttpRequest para recuperar dicho documento y poder manipularlo desde el código JavaScript cliente utilizando el **Modelo de Objeto Documento (DOM)**, cuyo estudio será abordado en el próximo capítulo.

Como ejemplo de aplicación de esto, vamos a realizar una nueva versión de la aplicación de libros en la que incluiremos una mejora, consistente en obtener a partir del título seleccionado en la lista no sólo un comentario asociado al libro sino también su precio, mostrándose los datos en una tabla HTML tal y como se ve reflejado en la figura 5.

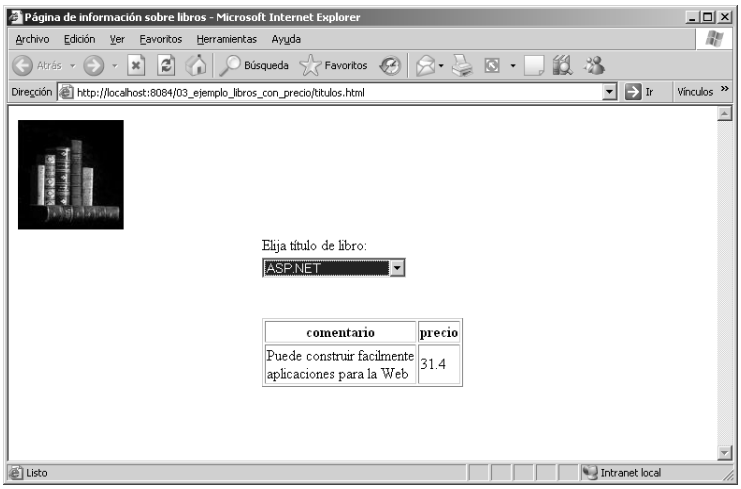

Fig. 5. *Nuevo aspecto de la aplicación de libros*

Para poder mantener esta información, el servlet incluirá un nuevo array en el que se almacenen los precios asociados a los libros.

A fin de poder enviar los datos al cliente de una forma estructurada, serán formateados como una cadena de texto XML que tendrá la siguiente estructura:

```
<?xml version="1.0"?>
<libro>
       <comentario>....</comentario>
       <precio>...</precio>
</libro>
```

El código del servlet se muestra en el siguiente listado:

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Comentariolibro extends HttpServlet {
    protected void service (HttpServletRequest request,
              HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/xml;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //textos de los comentarios
        String [] comentarios=
        {"Requiere conocimientos básicos de programación
                                    orientada a objetos",
        "Puede construir fácilmente aplicaciones
                                    para la Web",
        "Aprenderá rápidamente los principales
                                    trucos de Ajax",
        "Introduce las principales tecnologías
                                    de la plataforma"};
        //precios de cada libro
        String [] precios={"23.5","31.4","32.0","27.5"};
        int seleccionado=
             Integer.parseInt(request.getParameter("tit"));
```

```
            //formación del documento XML de respuesta
             String textoXML="<?xml version='1.0'?>";
             textoXML+="<libro>";
             textoXML+="<comentario>"+comentarios[seleccionado]+
                                      "</comentario>";
             textoXML+="<precio>"+
                        precios[seleccionado]+"</precio>";
             textoXML+="</libro>";
            //inserción de los datos XML en la respuesta
             out.println(textoXML);
             out.close();
        }
    }
```

Desde el punto de vista del cliente, tendremos que añadir una serie de instrucciones a la función *procesaDatos()* que, a partir del documento XML recuperado, extraigan la información del mismo y generen una tabla HTML con los datos para después mostrarla en la página, operación que se realizará, como se ha comentado anteriormente, utilizando el DOM.

El código de la función quedará como se indica en el siguiente listado:

```
function procesaDatos(){
     if(xhr.readyState==4){
          var resp=xhr.responseXML;
          var libro=resp.getElementsByTagName("libro").
                                   item(0);
          //recupera la colección de elementos
          //hijos de libro
          var elementos=libro.childNodes;
          var textoHTML="<table border='1'>";
          textoHTML+="<tr>";
          //genera la fila con los nombres
          //de los elementos
          for(var i=0;i<elementos.length;i++){
              textoHTML+="<th>"+elementos.item(i).nodeName+
                    "</th>";
          }
          textoHTML+="</tr>";
```

```
                textoHTML+="<tr>";
                //genera la fila con los datos del
                //libro incluidos en el documento
                 for(var i=0;i<elementos.length;i++){
                     textoHTML+="<td>"+
                           elementos.item(i).firstChild.nodeValue+
                                              "</td>";
                 }
                 textoHTML+="</tr></table>";
                //muestra la tabla en la página
                 document.getElementById("info").innerHTML=
                                                    textoHTML;
        }
}
```

Más adelante, analizaremos el Modelo de Objeto Documento y comprenderemos mejor algunas de las instrucciones que aparecen en el listado anterior. Por ahora, hay que comentar que el código de la función *procesaDatos()* se basa en generar una tabla con dos filas; en la primera de ellas se muestran los nombres de todos los subelementos de <libro>, mientras que en la segunda se cargan los valores contenidos en dichos elementos.

Para realizar estas operaciones, el código recorre la colección de elementos hijos de <libro>, por lo que la estructura podría servir no sólo para obtener la información de un documento de estas características en el que cada libro tiene un comentario y un precio, sino que, en general, podría ser aplicado para cualquier número de elementos hijos.