TEMA -9- AJAX CON DATOS DISTINTOS DE HTML O TEXTO

Ajax con otros tipos de datos: XML, JSON y archivos

Hasta ahora, en todos los ejemplos de este capítulo nos hemos limitado a responder desde el servidor con fragmentos de código HTML o texto (si se fija en los encabezados de respuesta en Firebug verá que su Content-type es siempre text/html o text/plain).

Sin embargo, en la práctica es más frecuente **utilizar respuestas que contengan datos estructurados**, bien en **XML** o en **JSON**, pues JavaScript posee propiedades y métodos que permiten aprovechar la estructura de esos datos.

- **En el caso de XML**, recuerde que el DOM es válido para HTML y XML, de modo que podremos aprovechar todos los métodos que ya conocemos (como getElementsByTagName o getAttribute) para acceder directamente a los datos.
- **En el caso de JSON**, los datos recibidos son directamente interpretados como objetos de JavaScript, de modo que podremos acceder a sus propiedades con la notación de punto o la literal.

XML

XML es un lenguaje de marcas que sirve para estructurar datos; su sintaxis es muy similar a la de HTML. Por ejemplo:

alumno_1.xml

```
001  <?xml version="1.0" encoding="UTF-8" ?>
002  <estudiante>
003          <nombre>Isaac Newton</nombre>
004          <calificaciones curso="20/21">
005                  <calificacion materia="Matemáticas" nota="7.5"/>
006                  <calificacion materia="Lenguaje" nota="9"/>
007          </calificaciones>
008  </estudiante>
```

Inicialmente el objeto XMLHttpRequest fue diseñado para trabajar con datos XML, pero actualmente este formato está dejando paso a otros menos verbosos, como JSON. El hecho de que XML sea tan verboso hace que las transacciones se ralenticen.

Para trabajar con datos XML en Ajax utilizaremos la propiedad **`responseXML`** en lugar de **`responseText`**, pues **en ella dispondremos del documento DOM de los datos XML recibidos**, y podremos explotarlos con todo el arsenal de técnicas que aprendimos en el tema anterior.

Nota: Para que la propiedad **`responseXML`** contenga el documento XML, es decir, un objeto de la clase **Document** del DOM (recuerde que el DOM es válido para HTML y XML), es necesario que el **content-type** declarado por el servidor en la respuesta sea **text/xml**. Si no es así, la propiedad `responseXML` contendrá el valor null. Si el servidor no declara correctamente el content-type podemos suplantarlo en el cliente llamando al método **`overrideMimeType`** del objeto **XMLHttpRequest** (se recomienda hacerlo antes de llamar al método `send()`. Por ejemplo, si el servidor devuelve un fragmento de un documento XML pero declara su **content-type** como **text/plain**, podríamos usar en el cliente el método **`overrideMimeType('text/xml')`** para que la propiedad **`responseXML`** lo adquiera correctamente.

Por ejemplo, cree el archivo del listado anterior con el nombre alumno_1.xml y otro llamado alumno_2.xml con el código que se muestra a continuación.

alumno_2.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<estudiante>
	<nombre>Marie Courie</nombre>
	<calificaciones curso="20/21">
		<calificacion materia="Matemáticas" nota="8"/>
		<calificacion materia="Lenguaje" nota="9.5"/>
	</calificaciones>
</estudiante>
```

Y, a continuación, cree el archivo index.xml con el código que se muestra a continuación.

```
<!DOCTYPE html>
<html>
	<head>
		<title>AJAX</title>
		<style>
		</style>
		<script>
			var xhr;
			var alumno;
			var materia;
			var calificacion;
			var datos;
			function enviarPeticionAJAX(evento) {
				if (alumno.value !='' ){
					alumno.disabled = true;
					materia.disabled = true;
					calificacion.value = '';
					materia.selectedIndex = 0;
					datos = new FormData(document.forms[0]);
					xhr = new XMLHttpRequest();
					xhr.addEventListener('readystatechange', gestionarRespuesta, false);
					xhr.open('POST', 'alumno_' + alumno.value + '.xml', true);
					xhr.send(null);
				}else{
					calificacion.value = '';
					materia.selectedIndex = 0;
					materia.disabled = true;
				}
			}
			function gestionarRespuesta(evento) {
				if (evento.target.readyState == 4 && evento.target.status == 200) {
					alumno.disabled = false;
					materia.disabled = false;
					datos = evento.target.responseXML;
				}
			}
			function actualizarCalificacion(){
				if (materia.value !=''){
					var i;
					var calificaciones = datos.getElementsByTagName('calificacion');
					for (i=0;i<calificaciones.length;i++){
						if(calificaciones[i].getAttribute('materia') == materia.value){
							calificacion.value = calificaciones[i].getAttribute('nota');
						}
					}
				}
			}
			document.addEventListener('readystatechange', inicializar, false);
```

```
049                         function inicializar() {
050                                 if (document.readyState == 'complete') {
051                                         alumno = document.getElementById('alumno');
052                                         materia = document.getElementById('materia');
053                                         calificacion =
    document.getElementById('calificacion');
054                                         alumno.addEventListener('change', enviarPeticionAJAX,
    false);
055                                         materia.addEventListener('change',
    actualizarCalificacion, false);
056                                 }
057                         }
058                 </script>
059         </head>
060         <body>
061                 <form id='formulario'>
062                         <label for='alumno'>Alumno: </label>
063                         <select id='alumno' name='alumno'>
064                                 <option value='' selected='selected'>--Elija un alumno--
    </option>
065                                 <option value='1'>Isaac Newton</option>
066                                 <option value='2'>Marie Courie</option>
067                         </select>
068                         <label for='materia'>Materia: </label>
069                         <select id='materia' name='materia' disabled='disabled'>
070                                 <option value='' selected='selected'>--Elija una materia--
    </option>
071                                 <option>Lenguaje</option>
072                                 <option>Matem&aacute;ticas</option>
073                         </select>
074                         <label for='calificacion'>Calificaci&oacute;n: </label>
075                         <input type='text' readonly='readonly' id='calificacion'/>
076                 </form>
077         </body>
078 </html>
```

JSON (http://www.json.org/)

JSON es el acrónimo de *JavaScript Object Notation* y simplemente es una notación para expresar objetos JavaScript casi idéntica a la notación literal que aprendimos en el tema relativo a Arrays. La principal diferencia es que en JSON los nombres de las propiedades deben ser cadenas escritas entre comillas dobles (no está estandarizado el uso de comillas simples).

Por ejemplo:

- Si el objeto JavaScript expresado en notación literal como `{edad: 38, peso: 79}`
- En JSON se expresaría como `{"edad": 38, "peso": 79}`.

Actualmente JSON es uno de los formatos más utilizados en transacciones Ajax debido a que es muy compacto (frente a la verbosidad de XML).

ACLARACIONES SOBRE JSON

La especificación de JSON se encuentra recogida en el documento RFC 4627 dentro del organismo de normalización **Internet Engineering Task Force**, más conocido por sus siglas IETF (el documento puede consultarse en http://www.ietf.org/rfc/rfc4626.txt). La sintaxis guarda gran similitud con el lenguaje javascript, esto se debe al hecho, que JSON es un subconjunto de JavaScript definido en ECMA-262 novena edición.
El principio de representación en el que se basa JSON es la **serialización**. La **serialización** permite representar objetos mediante una secuencia de bits o cadenas de texto, lo cuál facilita su lectura.

Los **formatos utilizados al serializar** un objeto mediante una cadena de texto son **XML** y **JSON**.

Las ventajas de serializar un objeto es que

- nos permite almacenar el objeto en un archivo o
- almacenar el objeto directamente en memoria,
- facilitar su transmisión a través de la red o
- comunicación entre objetos distribuidos.

Actualmente JSON está creciendo en popularidad y aporta ventajas sustanciales frente a XML al ser más legible, fácil y rápido de procesar.
Por otro lado, las definiciones de **JSON** señalan que se trata de un **formato ligero de intercambio de información**, donde el término ligero se debe a las características de JSON, **fácil edición y ocupa poco espacio**.

JSON permite representar la información mediante una serie predefinida de tipos primitivos y dos tipos estructurados de datos:

- Tipos primitivos
  - Cadena
  - Numérico
  - Booleano
  - Nulo
- Tipos estructurados
  - Objeto: parejas clave-valor, donde la clave es una cadena de texto.
  - Array: secuencia ordenada de 0 a n elementos

Ejemplo JSON

```
var ObjetoJSON=
{
"procesador":"AMD",
"memoria": "16GB",
"disco": "1TB"
}
```

Como vemos, la creación de un objeto en formato JSON es muy similar a la forma utilizada hasta ahora al crear objetos en JavaScript. La principal diferencia es la acotación entre comillas dobles del par **clave:valor**, como el caso del procesador. Existen librerías y utilidades que nos pueden ayudar en el proceso de formatear objetos según la sintaxis de JSON.
Una librería con la que se pueden convertir objetos de JavaScript a JSON de manera muy sencilla. La librería está disponible en http://www.json.org/json2.js.

JavaScript nos ofrece el **objeto JSON** con los métodos:

- **`stringify(objeto)`**, para convertir un objeto en una cadena expresada en JSON, y
- **`parse(cadena)`**, para interpretar una cadena expresada en JSON como un objeto de JavaScript.

Por ejemplo, en el siguiente código enviamos al servidor una cadena expresada en JSON creada a partir de un objeto con **`JSON.stringify`**, y recibimos de él una cadena que convertimos en un objeto mediante **`JSON.parse`**. En el archivo ajax.php simplemente se accede al cuerpo de la petición, en el que se encuentra la cadena enviada desde JavaScript, **se convierte esa cadena en un array** haciendo uso de la instrucción **`json_decode`**, y se devuelve una cadena expresada en JSON que JavaScript puede convertir en un objeto mediante **`JSON.parse()`**.

index.html

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>AJAX</title>
005                  <style>
006                  </style>
007                  <script>
008                          var xhr;
009                          var alumno;
010                          var materia;
011                          var calificacion;
012                          var objetoPeticion = new Object();
013                          var objetoRespuesta;
014                          function enviarPeticionAJAX(evento) {
015                                  if (alumno.value !='' && materia.value !=''){
016                                          objetoPeticion.alumno = alumno.value;
017                                          objetoPeticion.materia = materia.value;
018                                          alumno.disabled = true;
019                                          materia.disabled = true;
020                                          xhr = new XMLHttpRequest();
021                                          xhr.addEventListener('readystatechange',
     gestionarRespuesta, false);
022                                          xhr.open('POST', 'ajax.php', true);
023                                          xhr.setRequestHeader("Content-Type",
     "application/json");
024                                          xhr.send(JSON.stringify(objetoPeticion));
025                                  }else{
026                                          calificacion.value = '';
027                                  }
028                          }
029                          function gestionarRespuesta(evento) {
030                                  if (evento.target.readyState == 4 && evento.target.status ==
     200) {
031                                          alumno.disabled = false;
032                                          materia.disabled = false;
033                                          objetoRespuesta =
     JSON.parse(evento.target.responseText);
034                                          calificacion.value = objetoRespuesta.calificacion;
035                                  }
036                          }
037                          document.addEventListener('readystatechange', inicializar, false);
038                          function inicializar() {
039                                  if (document.readyState == 'complete') {
040                                          alumno = document.getElementById('alumno');
041                                          materia = document.getElementById('materia');
042                                          calificacion =
     document.getElementById('calificacion');
043                                          alumno.addEventListener('change', enviarPeticionAJAX,
     false);
044                                          materia.addEventListener('change',
     enviarPeticionAJAX, false);
045                                  }
046                          }
047                  </script>
048          </head>
049          <body>
050                  <form id='formulario'>
051                          <label for='alumno'>Alumno: </label>
052                          <select id='alumno' name='alumno'>
053                                  <option value='' selected='selected'>--Elija un alumno--
     </option>
054                                  <option>Isaac Newton</option>
055                                  <option>Marie Courie</option>
056                          </select>
057                          <label for='materia'>Materia: </label>
058                          <select id='materia' name='materia'>
059                                  <option value='' selected='selected'>--Elija una materia--
     </option>
060                                  <option>Lenguaje</option>
061                                  <option>Matem&aacute;ticas</option>
062                          </select>
063                          <label for='calificacion'>Calificaci&oacute;n: </label>
064                          <input type='text' readonly='readonly' id='calificacion'/>
065                  </form>
066          </body>
067  </html>
```

ajax.php

```
001  <?php
002          $entrada = fopen('php://input','r');
003          $datos = fgets($entrada);
004          $datos = json_decode($datos,true);
005          switch ($datos['alumno']){
006                  case 'Isaac Newton':
007                          switch ($datos['materia']){
008                                  case 'Matemáticas':
009                                          echo '{"calificacion":10.5}';
010                                          break;
011                                  case 'Lenguaje':
012                                          echo '{"calificacion":9.5}';
013                                          break;
014                          }
015                          break;
016                  case 'Marie Courie':
017                          switch ($datos['materia']){
018                                  case 'Matemáticas':
019                                          echo '{"calificacion":10.5}';
020                                          break;
021                                  case 'Lenguaje':
022                                          echo '{"calificacion":7.5}';
023                                          break;
024                          }
025                          break;
026          }
027  ?>
```

Archivos

Tradicionalmente, para enviar archivos a un servidor mediante Ajax se ha recurrido a técnicas muy poco ortodoxas basadas en el uso de Flash o iframes ocultos. Sin embargo, gracias al objeto **FormData** ahora podemos enviar archivos con la misma facilidad que el resto de los datos de un formulario; los controles **`input`** de tipo **`file`** se codifican en el FormData de forma transparente.

Más aún, los mismos eventos que se explicaron anteriormente para controlar el progreso de la recepción de una respuesta Ajax (`progress`, `load`, `timeout`, …), pueden utilizarse también sobre la propiedad **`upload`** del objeto XMLHttpRequest para controlar el progreso del envío de archivos al servidor (pues esta propiedad es un objeto de tipo XMLHttpRequestUpload).

Nota: La propiedad `timeout` se establece sobre el objeto XMLHttpRequest y afecta a la transacción completa. No se puede asignar un timeout independiente para la propiedad `upload`, pero sí un oyente para el evento timeout sobre `upload`, de modo que podamos indicar al usuario que su velocidad de subida es demasiado lenta.

Por ejemplo:

1. En el siguiente código utilizamos un control **`input`** de tipo **`file`** para solicitar al usuario que elija una imagen de avatar. Esa imagen se envía al servidor mediante Ajax al pulsar el botón Enviar, y podemos seguir el avance de la transacción mediante el **control** de tipo **`progress`**, que se va actualizando con los **eventos** `progress` y `load` de la propiedad **`upload`** del objeto XMLHttpRequest.
2. Debajo del **control** **`progress`** hay una imagen vacía, que se utilizará para mostrar la imagen enviada una vez concluida la transacción, evidenciando visualmente que se ha transmitido correctamente.
3. El archivo ajax.php simplemente recibe el archivo de la imagen, lo mueve de la carpeta de recepción temporal de PHP a su misma carpeta, y devuelve el nombre del archivo como respuesta de la petición para que el documento index.html pueda utilizarlo como valor del atributo **`src`** de la imagen que estaba inicialmente vacío, demostrándose así que la imagen se ha transmitido correctamente.

index.html

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>AJAX</title>
005                  <meta charset="utf-8" />
006                  <style>
007                  </style>
008                  <script>
009                          var xhr;
010                          function enviarPeticionAJAX(evento) {
011                                  var datos = new FormData(document.forms[0]);
012                                  xhr = new XMLHttpRequest();
013                                  xhr.timeout = 2000;
014                                  xhr.upload.addEventListener('progress',gestionarProgreso);
015                                  xhr.upload.addEventListener('load',cargaCompletada);
016                                  xhr.upload.addEventListener('timeout',subidaLenta);
017                                  xhr.addEventListener('readystatechange',
     gestionarRespuesta);
018                                  xhr.open('POST', 'ajax.php', true);
019                                  xhr.send(datos);

020                          }
021                          function gestionarRespuesta(evento) {
022                                  if (evento.target.readyState == 4 && evento.target.status ==
     200) {
023                                          var imagenAvatar=
     document.getElementById('imagenAvatar');
024                                          imagenAvatar.src = evento.target.responseText;
025                                  }
026                          }
027                          function gestionarProgreso(evento){
028                                  document.getElementById('progreso').max = evento.total;
029                                  document.getElementById('progreso').value = evento.loaded;
030                          }
031                          function cargaCompletada(evento){
032                                  document.getElementById('progreso').max = 1;
033                                  document.getElementById('progreso').value = 1;
034                          }
035                          function subidaLenta(evento){
036                                  alert('Cancelado.Su velocidad de subida es demasiado
     lenta.')
037                          }
038                  </script>
039          </head>
040          <body>
041                  <form id='formulario'>
042                          <label for='avatar'>Avatar: </label>
043                          <input type='file' id='avatar' name='avatar' accept='image/*'/>
044                          <br />
045                          <button type='button'
     onclick='enviarPeticionAJAX(event);'>ENVIAR</button>
046                          <br />
047                          <progress id="progreso" value="0"></progress>
048                  </form>
049                  <img width='300' height='300' src='' id='imagenAvatar' />
050          </body>
051  </html>
```

ajax.php

```
001  <?php
002          header("Access-Control-Allow-Origin: *");
003          if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
004                  exit;
005          }
006          $ext = strtolower(substr(strrchr($_FILES['avatar']['name'], "."), 1));
007          if (array_search($ext,array('jpg','gif','png'))  === false){
008                  exit; //Evitar que se suban otros archivos.
009          }
010          $target_path = "./";
011          $target_path = $target_path . $_FILES['avatar']['name'];
012          if (move_uploaded_file($_FILES['avatar']['tmp_name'], $target_path)) {
013                  echo utf8_encode($_FILES['avatar']['name']);
014          }
015  ?>
```

Nota: Tenga en cuenta que, por defecto, muchos servidores w eb tienen limitado el tamaño máximo de los archivos que pueden recibir por HTTP a 2 megabytes. En Apache puede aumentar este límite a través de las directivas upload_max_filesize y post_max_size del archivo php.ini.

Nota: Si lo desea puede comprobar el efecto del evento progress ejecutando el ejemplo anterior contra un servidor remoto php disponible.
