AJAX

Antonio Espín Herranz

Introducción

- AJAX (JavaScript Asíncrono y XML) es un nuevo término que se introdujo para describir dos capacidades de los navegadores que de forma independiente ya han estado presentes durante muchos años.

- No es una nueva tecnología si no la unión de varias.

- El concepto consiste en cargar y renderizar una página, mantenerse en esa página mientras scripts ejecutados en 2º plano consultan al servidor buscado, los datos que usados para actualizar la página y sólo renderizando la página y mostrando y ocultando porciones de la misma.

Introducción

- Posibilidad de hacer peticiones al servidor sin tener que volver a cargar la página (XMLHttpRequest).
- Posibilidad de analizar y trabajar con documentos XML (DOM).
- La presentación de la página basada en los estándares xHTML y CSS.
- JavaScript se encarga de unificar todo y será nuestro lenguaje de programación.

Introducción

- En lugar de cargar una página Web completa; al inicio de la sesión, el navegador carga el motor AJAX (escrito en JavaScript).

- Este motor es el responsable de renderizar la interfaz que el usuario ve y de comunicarse con el servidor en nombre del usuario.

- El motor AJAX permite que la interacción del usuario con la aplicación sucesa asíncronamente, de modo que el usuario nunca está mirando una ventana en blanco del navegador y un icono de reloj de arena esperando que el Servidor responda.

XMLHttpRequest

- Para poder realizar la petición al Servidor se utiliza el objeto XMLHttpRequest.

- El nombre del objeto podría llevar a cierta confusión:
  - Se soporta cualquier formato de texto, no sólo XML.
  - Puede ser utilizado sobre https, no sólo http.
  - No sólo puede lanzar peticiones si no que puede capturar las respuestas.

XMLHttpRequest

- Toda implementación de XMLHttpRequest ha de tener soporte para:
  - DOM: Se ha de soportar Document Object Model.
  - HTTP: Protocolo de transmisión HTTP.
  - XML: Se ha de soportar alguna versión de XML, en caso contrario las respuestas siempre serán vacías.

XMLHttpRequest

- En cuanto a las implementaciones indicar que inicialmente fue implementada por IE mediante un control ActiveX: XMLHTTP.

- Posteriormente Mozilla y el resto de navegadores implementaron dicha clase en el API según la especificaci´ón W3C, ofreciendo los métodos y propiedades del objeto original.

- Hoy en día tenemos que detectar el navegador:

  - Si es IE, tendremos que cargar el control: **Microsoft.XMLHTTP.**
    - **http_request = new ActiveXObject(“Microsoft.XMLHTTPRequest)**

  - Si se trata de Mozilla, Opera, Safari, etc. Se ha de instanciar el constructor de la clase **XMLHTTPRequest**.
    - **var http_request = new XMLHTTPRequest();**

Proceso a realizar

- Se ha de instanciar el objeto XMLHTTPRequest, teniendo en cuenta si se respeta o no la especificación del W3C.
- Si es posible instanciar el objeto, se ha indicar la función encargada de manejar las respuestas del Servidor.
- Se crea la petición y se envía al Servidor.
- La función manejadora monitorizará el estado de las Respuestas.
- Si se obtiene respuesta sin error, dicha función analizará la respuesta.
- Se utiliza la respuesta, normalmente para interactuar con el árbol del documento.

Ejemplo

- Queremos cambiar el titulo de nuestra página mediante AJAX.
  - El titulo se encuentra en un fichero XML almacenado en el Servidor.
- Disponemos del fichero XML, de la página y mediante javascript instanciamos el objeto XMLHTTPRequest, le hacemos la petición.
  - Recuperamos el fichero XML, capturamos el nuevo título mediante el DOM y actualizamos el de nuestra página.
- A la función AJAX la llamamos desde un botón de nuestra página.

La página

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>EJEMPLO BASICO DE AJAX</title>
<script src="ajax.js"></script>
</head>
<body>
<p>Pulse en el siguiente enlace para cambiar de forma dinámica el titulo de la página</p>
<input type="button" name="pulsa" value = "pulsa" onclick = "cargarAJAX('titulo.xml')" />

</body>
</html>
```

El fichero XML

```
<?xml version="1.0" encoding="UTF-8"?>
<titulo>Titulo obtenido mediante
   AJAX</titulo>
```

Código JavaScript

```
// Petición HTTP:
var http_request = false;

/* Código para cargar el Motor AJAX */
function cargarAJAX(url){

// 1. Comprobamos si podemos cargar el
      objeto de la especificación W3C.
// Será para navegadores <> de IE.

if (window.XMLHttpRequest){
// Creamos el objeto:
http_request = new XMLHttpRequest();

} else if (window.ActiveXObject){
// Instanciamos el ActiveX, estamos con IE.
http_request = new
      ActiveXObject("Microsoft.XMLHTTP");

}

// 2. ¿Ha ido todo bien?
if (http_request == null){
// No.
window.alert("Error no se puede cargar el objeto
      XMLHttRequest ...");
return false;

}

// 3. Hemos conseguido instanciar el objeto,
      le asignamos la función manejadora:
http_request.onreadystatechange =
      funcionProcesamiento;

// 4. Se inicia la petición asíncrona:
http_request.open('GET', url, true);

// 5. Se envía la petición:
http_request.send(null);
}
```

Código JavaScript II

```
function funcionProcesamiento(){

    if (http_request.readyState == 4){ // Carga completada:

        if (http_request.status == 200){ // OK

            var nuevoTitulo;

            // Respuesta del Servidor:
            var xmldoc = http_request.responseXML;


            // Obtengo la información deseada:
            var nodoTitulo = xmldoc.getElementsByTagName('titulo').item(0);
            nuevoTitulo = nodoTitulo.firstChild.nodeValue;

            // Se accede al árbol DOM del documento (mi página) y se cambia el título:
            document.title = nuevoTitulo;

        } else
            window.alert("respuesta del Servidor: " + http_request.status);
        }
}
```

Métodos

- **open(“HTTP method”, “URL”, syn/asyn)**
  - Asigna método HTTP (POST/GET), URL y el modo (true = asíncrono, false = síncrono)
- **send(content)**
  - Envía la petición incluyendo opcionalmente un contenido
- **abort()**
  - Cancela la petición en curso
- **getAllResponseHeaders()**
  - Devuelve el conjunto de cabeceras HTTP como una cadena
- **getResponseHeader(“header”)**
  - Devuelve el valor de la cabecera especificada
- **setRequestHeader(“label”,”value”)**
  - Añade un par etiqueta/valor a la cabecera HTTP a enviar

Propiedades

- **onreadystatechange**
  - Función JavaScript que se activa con cada cambio de estado
- **readyState**
  - devuelve el estado del objeto actual como sigue:
  - 0 = sin inicializar, 1 = cargando, 2 = cargado, 3 = interactivo y 4 = completado
- **status**
  - Estado HTTP devuelto por el servidor: **200 = OK**, 404 = Not Found, …
- **responseText**
  - Devuelve la respuesta como una cadena
- **responseXML**
  - Devuelve la respuesta como XML
- **statusText**
  - Devuelve el estado como una cadena (e.g. “OK”, “Not Found”, …)

PRACTICA:

AJAX