HTTP y el objeto XMLHttpRequest

Introducción

Aunque no es reciente, el objeto XMLHttpRequest no ha sido muy utilizado antes de la aparición de Ajax (en comparación con otras tecnologías como CSS, DOM, XML, JavaScript común...), hemos decidido dedicar una parte del curso, con un capítulo completo y presentarlo antes de realizar aplicaciones prácticas con Ajax, para que se entienda mejor el funcionamiento de las comunicaciones asíncronas que caracterizan a las aplicaciones Ajax.

Recursos sobre tecnologías utilizadas por Ajax

Es importante consultar otras fuentes para encontrar la información pertinente de muchos recursos sobre las tecnologías subyacentes de Ajax (CSS, DOM, JavaScript, XML...).
Es complicado en un solo curso aprenderlas y conocerlas por completo, optamos por presentar algunos aspectos importantes de dichas tecnologías y dejar al estudiante su profundización.

Un recordatorio sobre el protocolo HTTP

Para las aplicaciones Ajax y como para la mayor parte de las aplicaciones Web tradicionales, las transferencias de los datos entre el servidor Web y el cliente (el navegador) utilizan el protocolo HTTP (HyperText Transfer Protocol). Para entender mejor el funcionamiento de los mecanismos del objeto XMLHttpRequest, es interesante hacer un pequeño recordatorio del modo de funcionamiento de este protocolo ampliamente utilizado en Internet.

Una transferencia HTTP consta de dos elementos: las solicitudes (GET o POST) enviadas por el navegador y la respuesta devuelta al navegador por el servidor.

Las solicitudes HTTP

Para enviar una petición (solicitud) HTTP al servidor desde un navegador, se pueden utilizar dos métodos: *GET* o *POST*.

Como indicar el método elegido en Ajax?
En el caso de un simple formulario, se indica el método elegido, especificando que el valor del atributo method de la etiqueta del formulario (<form method="get"> por ejemplo) mientras que en el caso de Ajax, se notificará en el primer parámetro del método open() de XMLHttpRequest (objetXHR.open('get','miArchivo.php?val='+i,true) por ejemplo. El método open() y los otros métodos y propiedades del objeto XMLHttpRequest se detallan más adelante.

Solicitud con el método GET

Una solicitud GET puede iniciarse mediante un formulario (el atributo method se configuraría con el valor de GET) pero también a partir de un simple vínculo hipertexto al cual se le asociarán con la URL en forma de un par variable/valor después de un signo de interrogación (ver el siguiente código 1-1).

Código 1-1:

miPrograma.php?message=hola

Si deben enviarse varias parejas de variable/valor, estas están separadas por el símbolo (ver el código 1-2).

Código 1-2:

miPrograma.php?message=hola&nombre=pepito

Entre los inconvenientes del método GET, cabe señalar que el tamaño de una consulta GET se limita a 255 bytes (o 1024 según los sistemas), la cantidad de información enviada por este método se reduce de tal modo. Además, los valores que se envían en el URL, por lo tanto son visibles por todos. Las consultas GET no podrán utilizarse para enviar informaciones voluminosas o si se debe preservar la confidencialidad de la información.

Las ventajas del método GET son principalmente su simplicidad y la posibilidad de construir dinámicamente los pseudo-URL en los que están integrados los valores para enviar (lo que es una buena práctica en los catálogos del sitio para mostrar el registro de un producto específico de una lista de hipervínculos, por ejemplo: registro.php?id=5). Tengamos en cuenta que el método GET se utiliza también con frecuencia en las consultas de los motores de búsqueda debido a la posibilidad de que el usuario guarde la consulta en sus favoritos.

Sea cual sea el origen de una solicitud GET (formulario o hipervínculo con parámetros), el servidor recibirá un conjunto de información que consiste en una línea de solicitud y de varias líneas de encabezado (para las solictudes GET, el cuerpo de la solicitud es vacío porque los valores se insertan en la línea de la solicitud, consultar el código 1-4).

Código 1-3: Ejemplo del URL absoluto con parámetros:
http://www.curso.com/miDirectorio/miPrograma.php?Message=Hola

Código 1-4: Ejemplo simplificado de una solicitud HTTP con el método GET iniciado por el URL del código 1-3:

1: Línea de solicitud
GET **/miDirectorio/?message=Hola** HTTP/1.1

2: Líneas de encabezado
Host: **www.curso.com**
User-Agent: **Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6) Gecko/20050225**
➥**Firefox/1.0.1**
Connection: Keep-Alive

3: Cuerpo de la solicitud (vacio en el caso de una solicitud GET)

Solicitud con el método POST

A diferencia del método GET, los valores enviados con el método POST no son visibles en la barra de direcciones (lo cual es importante si desean enviar contraseñas por ejemplo...) pero si están considerados en una variable de entorno insertada en el cuerpo de la solicitud (y por lo tanto invisible para el usuario). La cantidad de información no está más limitada como en el método GET y este método ofrece la posibilidad de transmitir flujos de datos con mayor cantidad de información (hasta 2 GB). Por otra parte, los datos enviados con el método POST no pueden transmitirse más que con formularios (o con el método open() de un objeto XMLHttpRequest para aplicaciones Ajax) y no por una URL como con el método GET.

Tengamos en cuenta que, a diferencia de una solicitud GET para la que algunos navegadores (IE por ejemplo) recuperan automáticamente los datos anteriores en la memoria caché, no es igual con el método POST que requiere la autorización del usuario para reenviar los parámetros (sin importar el navegador utilizado).
En el caso de una solicitud POST, el servidor recibirá un conjunto de información compuesta por una línea de solicitud, varias líneas de encabezado y un cuerpo de solicitud que contiene los valores a transmitir (ver el código 1-5).

Código 1-5: Ejemplo simplificado de una solicitud HTTP con el método POST iniciado por uno formulario que contiene un campo denominado «mensaje»:

1: Línea de solicitud
POST **/miDirectorio/** HTTP/1.1

2: Líneas de encabezado
Host: **www.curso.com**
User-Agent: **Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6) Gecko/20050225**
➥**Firefox/1.0.1**
Content-type: **application/x-www-form-urlencoded**
Content-Length: **15**
Connection: Keep-Alive

3: Cuerpo de la solicitud
message=Hola

Si se compara la información de una solicitud POST (ver el código 1-5) con la de la solicitud GET (ver el código 1-4), se destaca el hecho que los datos que se van a transmitir ahora se insertan en el cuerpo de la solicitud y que se han añadido dos nuevos encabezados. El encabezado Content-type que especifica la codificación utilizada (podemos notar que se trata siempre del mismo tipo MIME independientemente del navegador) para el cuerpo de la solicitud -que contiene ahora los datos- y el encabezado Content-Length que indica el tamaño en bytes de los datos contenidos en el cuerpo de la solicitud.

La respuesta HTTP

El tratamiento por el servidor de una solicitud HTTP reenvía un conjunto de información al navegador que inició la petición (ver el código 1-6). Si checamos esta información, se pueden distinguir tres partes diferentes:

La **línea de estado** (ver la parte 1 del código 1-6) que nos recuerda la versión del protocolo HTTP (HTTP/1.1 en el ejemplo) seguido del estado de la respuesta (información importante para Ajax como 200, que significa el éxito de la operación) y un mensaje complementario (OK por ejemplo).

Un número variable de **líneas de encabezado** (ver la parte 2 del código 1-6) que contienen entre otras cosas, el encabezado Content-Type que indica el tipo MIME del resultado (información importante para Ajax). Este tipo puede por ejemplo ser text/plain para el texto en bruto, text/html para una página HTML o text/xml para XML. Señalamos también el encabezado Cache-Control que es interesante para Ajax ya que permite interactuar con la caché del navegador.

El **contenido de la respuesta** reenviada (ver la parte 3 del código 1-6) puede ser una página HTML completa como en el ejemplo que se muestra abajo, pero en el caso de Ajax, el contenido de la respuesta consistirá principalmente de datos solos (en el formato texto o XML), o incluso de fragmentos HTML para las respuestas generadas por una consulta de una aplicación Ajax. De hecho, hemos visto que una de las ventajas de una aplicación Ajax es precisamente poder recuperar el mínimo útil del servidor y no la página completa como con las solicitudes Web tradicionales.

Código 1-6: Ejemplo de una respuesta HTTP del servidor:

1 : Linea de estado
HTTP/1.1 **200** OK

2 : Lineas de encabezado

Cache-Control: private
Content-type: **text/html**
Server: GWS/2.1

Date:Tue, 15 May 2013 13:20:24 GMT

3 : Contenido de la respuesta
<html>
<head>
<title>Respuesta</title>
</head>
<body>
Hola
</body>
</html>

Según la respuesta del servidor, el código del estado HTTP puede cambiar. Veremos a continuación que este código deberá probarse con el script del motor Ajax para asegurarse que la operación se ha efectuado con éxito. Es por lo tanto útil conocer los principales códigos del estado HTTP que el servidor puede reenviar. Estos códigos se detallan en la siguiente tabla.

Código del estado	Significado	Ejemplo/Mensaje
200 a 299	Éxito de la operación	200/OK : Solicitud completada con éxito
300 a 399	Redirección	301/Moved Permanently :Redirección permanente
400 a 499	Error cliente	404/Not Found: Documento no encontrado
500 a 599	Error servidor	503/Service Unavailable: Servicio no disponible

Tabla 1: Principales códigos del estado HTTP.

Características del objeto XMLHttpRequest

La comunicación de las aplicaciones Ajax con el servidor se basa principalmente en el objeto XMLHttpRequest. Esta clase JavaScript permite ejecutar las solicitudes HTTP del navegador hacia el servidor de una forma asíncrona (respuesta diferida sin bloquear la aplicación cliente) sin tener que recargar la página HTML como es el caso de una solicitud HTTP tradicional.

Ajax ha estado en funcionamiento desde 1998

El objeto XMLHttpRequest no es reciente ya que la clase que permite instanciarlo fue creada en 1998. En ese momento, Microsoft la había desarrollado para integrarla como un control ActiveX en su navegador Internet Explorer 5.0.

Muy poco utilizado hasta la llegada de Ajax, sin embargo, se implementará gradualmente en la mayoría de navegadores e incluso será el objeto de estudio para un proyecto del W3C desde 2006.

Año de implementación	Navegador(es)
1998	Internet Explorer
2002	Mozilla
2004	Safari
2005	Konqueror y Opera

Tabla 2: Progresión de la implementación de la clase XMLHttpRequest en los diferentes navegadores.

Una instancia en espera

Si esta clase está implementada en los principales navegadores, el proceso de instanciación de un objeto es desafortunadamente un aspecto que no se ha estandarizado para todos (aunque la homogeneización de su uso pronto será toda una realidad).

De hecho, como la clase fue creada originalmente como un control ActiveX, el procedimiento de instanciación con Internet Explorer (utilización del constructor ActiveXObject()) es diferente de los otros navegadores (usando el constructor XMLHttpRequest()).

Además, el valor que se pasa como parámetro en la instanciación de un objeto en un navegador Microsoft es diferente entre la primera versión de Internet Explorer, en el que se implemento (la versión 5.0 del IE) y las versiones posteriores. Por lo tanto encontramos 3 diferentes tipos de sintaxis para crear una instanciación según el navegador utilizado!

Para solucionar este problema, un método ingenioso consiste en desarrollar una función de creación genérica del objeto XMLHttpRequest, teniendo en cuenta la detección del navegador (y también la versión para el IE) con el fin de utilizar la sintaxis correcta para la instanciación según el contexto.

Esta función puede integrarse convenientemente en un archivo JavaScript externo que está ligado a las páginas HTML usando Ajax. Tengamos en cuenta que si se utilizan frameworks (por ejemplo, jQuery o Prototype por ejemplo) la funcionalidad de la detección del navegador se integra generalmente en su biblioteca JavaScript.

Navegador(es)	Sintaxis de instanciación de un objeto XMLHttpRequest
Internet Explorer 5.0	new ActiveXObject(“Microsoft.XMLHttp”)
Internet Explorer versión posterior a 5.0	new ActiveXObject(“Msxml2.XMLHttp”)
Otros navegadores (Firefox,…)	new XMLHttpRequest ( )

Tabla 3: Sintaxis utilizadas para instanciar un objeto XMLHttpRequest según los navegadores y sus versiones.

Para ilustrar la creación de un objeto XMLHttpRequest en JavaScript, presentamos a continuación un ejemplo de una función genérica de creación de un objeto XMLHttpRequest según el navegador utilizado.

Código 1-7: Función genérica de creación de un objeto XMLHttpRequest:

```
function creationXHR() {
var resultado=null;
try {//prueba para los navegadores : Mozilla, Opera...
  resultado= new XMLHttpRequest();
  }
catch (Error) {
  try {//prueba para los navegadores Internet Explorer > 5.0
  resultado= new ActiveXObject("Msxml2.XMLHTTP");
  }
  catch (Error) {
    try {//prueba para el navegador Internet Explorer 5.0
    resultado= new ActiveXObject("Microsoft.XMLHTTP");
    }
      catch (Error) {
      resultado= null;
    }
  }
}
return resultado;
}
```

La función JavaScript anterior puede ser incorporada directamente en la página HTML pero tiene más sentido colocarla en un archivo JS externo y vincularla a continuación en la correspondiente página HTML con la finalidad de poder disponer de esta fácilmente en las otras páginas del sitio.

```
var objectXHR = creationXHR();
```

La línea de código anterior es un ejemplo de utilización de la función genérica para crear un objeto XMLHttpRequest.

La función del código 1-7 prueba sucesivamente la creación del objeto XMLHttpRequest con las diferentes sintaxis de instanciación que corresponden a los tres tipos de navegadores. La estructura de gestión de las excepciones (try) permite probar una función sin generar incluso error. Si la prueba es positiva, el objeto se crea y, en el caso contrario, esta pasar a la siguiente prueba y se realiza el mismo proceso (catch).

Nota: Se aconseja revisar los recursos de JavaScript (gestión de excepciones: try, catch).

Propiedades y métodos del objeto XMLHttpRequest

Como todas las clases, XLMHttpRequest posee propiedades y métodos. Antes de utilizarlos para implementar una solicitud al servidor, presentamos las tablas 4 y 5 donde se resumen sus propiedades y métodos.

Propiedades	Descripción
onreadystatechange	Designa una función de retrollamada que se llamará cada vez que el estado de tratamiento de una solicitud asíncrona (readyState) cambiará de estado.
readyState	Estado del tratamiento de una solicitud asíncrona (valor numérico de 0 a 4, ver la tabla 6 para conocer los diferentes estados y su significado).
responseText	Contiene el resultado del servidor en formato texto.
responseXml	Contiene el resultado del servidor en formato XML.
status	Contiene el código del estado HTTP (ejemplo: 200, ver la tabla 1). Este código está disponible en modo lectura únicamente cuando se ha transmitido la respuesta.
statusText	Contiene el mensaje del estado HTTP (ejemplo: OK, ver tabla 1). Este texto está disponible en modo lectura únicamente cuando se ha transmitido la respuesta.

Tabla 4: Propiedades del objeto XMLHttpRequest .

Métodos	Descripción
abort()	Anula la solicitud en curso.
getAllResponseHeaders()	Retorna en una cadena de caracteres todos los encabezados HTTP contenidos en la respuesta del servidor. Este método es útil solo cuando el estado del tratamiento (readyState) es igual a 3 o 4 (Interactive o Completed).
getResponseHeader(“nombreEncabezado”)	Retorna el valor del encabezado cuyo nombre está indicado en el parámetro (nombreEncabezado). Este método es útil sólo cuando el estado del tratamiento (readyState) es igual a 3 o 4 (Interactive o Completed).
open(“metodo”,”url”,”async”)	Inicializa el objeto mediante la especificación de algunos parámetros de la solicitud: el método utilizado (método: GET o POST), el URL del script del lado del servidor (url: archivo.php por ejemplo) y un tercer parámetro opcional para indicar si la comunicación debería ser asíncrona (async: true) o síncrona (async: false).
send(“contenido”)	Envía la solicitud. El parámetro (contenido) puede ser nulo en el caso de una solicitud inicializada con el método GET o una cadena de solicitud en el caso de un método POST. Obviamente este método puede usarse sólo cuando el método open() ya se ha configurado y aun cuando el estado del tratamiento (readyState) es igual a 1 (Loading).
setRequestHeader(“nombre”, “valor”)	Asigna un valor (valor) al encabezado de la solicitud, cuyo nombre se especifica en el primer parámetro (nombre). Este método es útil sólo cuando el estado del tratamiento (readyState) es igual a 1 (Loading).

Tabla 5: Métodos del objeto XMLHttpRequest .

Valor	Estado	Significado
0	Unitialized	El objeto aun no se ha inicializado y por lo tanto el método open() aún no se ha llamado.
1	Loading	El objeto se ha inicializado pero la solicitud aun no se envía utilizando el método send().
2	Loaded	La solicitud se ha enviado con el método send().
3	Interactive	La respuesta está en proceso de recepción.
4	Completed	La respuesta del servidor se recibe totalmente. Los datos están disponibles en la propiedad responseText (se trata de los datos en formato texto) o en responseXML (se trata de los datos en formato XML).

Tabla 6: Significado de los diferentes estados de tratamiento de una solicitud asíncrona (accesible gracias a la propiedad <<readyState>>.

Restricción de acceso a dominios externos
Para evitar el mal uso de una aplicación Ajax que recupera datos de otro sitio Web sin el consentimiento de su propietario, los navegadores han integrado un sistema de seguridad que prohíbe las llamadas Ajax a diferentes áreas en las cuales se encuentra la aplicación Ajax.
Sin embargo, en algunos casos es necesario acceder a los recursos situados en un servidor externo o proporcionado por un proveedor de información (caso de los flujos RSS por ejemplo). Para desviar esta restricción, entonces es posible implementar un servidor proxy que tendrá la función de transmitir las solicitudes entre el servidor Web y el servidor del proveedor de información.

Creación de los motores básicos de Ajax

Envío de una solicitud síncrona sin parámetros

Incluso si el objeto XMLHttpRequest se utiliza muy a menudo en modo asíncrono, este también es capaz de manejar solicitudes síncronas. En este caso, el tercer parámetro del método open() debe configurarse con el valor false y el tratamiento será secuencial (ver el código 1-8). Este modo de funcionamiento es más parecido al proceso de una solicitud de una aplicación Web tradicional (bloqueo de la aplicación cliente al esperar la respuesta del servidor) y la utilización del objeto XMLHttpRequest tiene mucho menos interés que en el modo asíncrono que trataremos a continuación.

Códigos educativos
Los distintos códigos de estas notas se han desarrollado para que puedas entender los mecanismos de una aplicación Ajax de manera progresiva. A menudo están incompletos y se aconseja no tratar de probarlos en la práctica, sino más bien concentrarse en su análisis. Más adelante se realizarán prácticas en las que se implementarán algunas aplicaciones Ajax.

Código 1-8: Ejemplo de tratamiento de una solicitud síncrona (este código necesita de la declaración previa de la función creationXHR() definida en el código 1-7):

```
###############-MOTOR AJAX-#################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//----Tratamiento SINCRONO
//Configuración del método utilizado, del script servidor objetivo
//y finalmente la activación del tipo síncrono como último parámetro
objetoXHR.open("get","script.php",false);
//envío de la solicitud
objetoXHR.send(null);
//recuperación de la respuesta del servidor
var resultado = objetoXHR.responseText ;
….
//El resultado puede ahora insertarse en la página HTML
//si el script servidor corresponde a aquel del cuadro, el valor
//de la variable "resultado" sería entonces igual a "Buenos días"
###############-FIN DEL MOTOR AJAX-#################
```

La primera línea del código crea el objeto XMLHttpRequest por instanciación de su clase. Una vez creado el objeto, es entonces posible inicializar la solicitud mediante el método open(). En el caso de una solicitud síncrona, el tercer parámetro de este método deberá configurarse con el valor false. Es suficiente con sólo enviar la solicitud con el método send(null) (tengamos en cuenta que el argumento del método send() es igual a null ya que el método de la solicitud se inicializa con GET) y esperar la respuesta en la propiedad responseText del objeto.

Respuesta HTTP del servidor a una solicitud sin parámetro
Para que entiendas completamente todos los aspectos de este proceso de comunicación HTTP, proponemos un ejemplo del script del servidor muy simple que se puede utilizar en los ejemplos del motor de Ajax sin parámetros que se discutirán en estas notas. Este script es voluntariamente minimalista con el fin de que puedas concentrarte en el motor Ajax y no en el procesamiento del lado del servidor.

Código 1-9: Ejemplo de código PHP del archivo script.php:

```
<?php
//indica que el tipo de la respuesta reenviada al cliente será en formato texto
header("Content-Type: text/plain");
//regresa la respuesta al cliente con la ayuda del comando echo
echo "Buenos días";
?>
```

En este contexto, la respuesta HTTP del servidor a una solicitud simple (sin parámetro) podría parecerse al siguiente ejemplo:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-type: text/plain
Server: GWS/2.1
Date:Tue, 15 May 2013 13:20:24 GMT
Buenos días
```

Envío de una solicitud asíncrona sin parámetro

A diferencia del modo síncrono, el modo asíncrono requiere el uso de una función de retrollamada (callback). Esta función de retrollamada debe declararse y señalarse antes del envío de la solicitud para que el motor de Ajax entonces pueda llamar a la recepción de la respuesta del servidor. Para designar la función de retrollamada, su nombre debe almacenarse en la propiedad onreadystatechange del objeto XHR (ver la parte 1 en el código 1-10).
Así, la función designada como función de retrollamada será llamada en cada cambio de la propiedad readyState. La propiedad readyState corresponde a los diferentes estados del tratamiento de una solicitud Ajax (ver la tabla 6), así esta función cambiará varias veces de estado durante el ciclo de cada solicitud. Como queremos recuperar el resultado sólo cuando esté completamente asegurada la recepción, es por lo tanto necesario agregar una prueba en la función de retrollamada (ver la parte 2 del código 4-10) con el fin de asegurar

que la propiedad readyState es igual al valor 4 (estado Completed) antes de copiar el resultado en una variable para su futura operación (ver el código 1-10).
Código 1-10: Ejemplo del tratamiento de una solicitud asíncrona (este código requiere la declaración previa de la función creationXHR() definida en el código 1-7).

```
##############-MOTOR AJAX-#################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//Declaración de la función de retrollamada
function tratamientoResultado() {
if(objetoXHR.readyState==4) {❷
var resultado = objetoXHR.responseText ;
...
//####El resultado puede ahora insertarse en la página HTML
//si le script del servidor corresponde al del cuadro
//el valor de la variable "resultado" será
//entonces igual a " Buenos días "
}
}
//----Tratamiento ASINCRONO
//Designación de la función de retrollamada
objetoXHR.onreadystatechange = tratamientoResultado; ❶
//Configuración del método utilizado, de script servidor objetivo
//y finalmente la activación de tipo asíncrono al final
objetoXHR.open("get","script.php",true);
//envío de la solicitud
objetoXHR.send(null);
###############-FIN DEL MOTOR AJAX-#################
```

Agrega un tratamiento de los errores HTTP del servidor

Las anteriores scripts funcionan muy bien siempre y cuando no haya ningún problema en la transferencia HTTP. Ahora imaginemos que el script del servidor se elimina o mueve.
En este caso los scripts ya no funcionan y nada indicará la naturaleza del problema. Sin embargo, el servidor tiene una variable que indica los errores HTTP, se trata por supuesto del código de estado (ver la tabla 1). Asimismo, el objeto XMLHttpRequest tiene una propiedad estado (status) que permite leer fácilmente esta información. Así que vamos a modificar el script de la función de retrollamada para agregar una segunda prueba y asegurarse de que todo ha sido correcto antes de usar el resultado (en este caso, la propiedad status debe ser igual a 200). En caso contrario, mostrará un mensaje de error para informar a los usuarios.

Código 1-11: Ejemplo de una solicitud asíncrona con el tratamiento de posibles errores HTTP (este código requiere la declaración previa de la función creationXHR() definida en el código 1-7):

```
################-MOTO AJAX-##################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//Declaración de la función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    var resultado = objetoXHR.responseText ;
...
   //####El resultado puede ahora insertarse en la página HTML
   //si el script del servidor corresponde al del cuadro,
   //el valor de la variable "resultado" será
   //entonces igual a "Buenos días"
    }else {
    alert("Error HTTP N°"+ objetoXHR.status);
   }
  }
}
//----Tratamiento ASINCRONO
//Designación de la function de retrollamada
objetoXHR.onreadystatechange = tratamientoResultado;
//Configuración del método utilizado, de script del servidor objetivo
//y finalmente la activación del tipo asíncrono en último parámetro
objetoXHR.open("get","script.php",true);
//envío de la solicitud
objetoXHR.send(null);
################-FIN DEL MOTOR AJAX-##################
```

Envío de una solicitud asíncrona con un parámetro GET

En general las solicitudes Ajax están acompañadas de parámetros que indican al script del servidor los datos de retorno.

La manera más simple de enviar un parámetro es utilizar el método GET y agregar un parámetro del URL después de la dirección del script objetivo como por ejemplo: scriptConParametro.php?id=2 (para el uso de este parámetro del lado del servidor, ver el código 1-12).

Este método también tiene la ventaja de estimular el desarrollo del script del lado del servidor, ya que sólo lo llama individualmente en el navegador añadiendo el parámetro del URL manualmente, seguido su nombre (para ello, introduce la dirección URL completa en la barra del navegador como por ejemplo http://localhost/scriptConParametro.php? id = 2). El script entonces llamado con este parámetro debe mostrar en la pantalla los mismos datos que luego reenviará enseguida al objeto XMLHttpRequest cuando el sistema estará operativo (es decir, en nuestro ejemplo "Buenos días señor Smith"). Esta técnica permite verificar que el script del servidor funciona correctamente y de direccionar posiblemente las búsquedas hacia el programa JavaScript del lado del cliente si el problema persiste.

Del lado del servidor, el parámetro (?id=2, por ejemplo) se recupera a través de la variable $_REQUEST['id'] antes de ser utilizado en el programa de construcción del resultado (ver el código 1-12). Entonces este se mostrará en la pantalla para ser recuperado en modo retardado en la propiedad responseText de la función de llamada del motor Ajax del lado del cliente (ver la función tratamientoResultado() del código 1-13).

Respuesta HTTP del servidor a un parámetro de la solicitud

Para los motores Ajax que incluyen un parámetro en el envío de la solicitud, hemos modificado ligeramente el script del servidor para que envíe el mensaje correspondiente al número de identificación del usuario en cuestión.
Código 1-12: Ejemplo del código PHP del archivo scriptConParametro.php:

```
<?php
//indica que el tipo de la respuesta reenviada al cliente estará en modo texto
header("Content-Type: text/plain");
//recuperación de la variable GET
if(isset($_REQUEST['id'])) $id=$_REQUEST['id']; else $id=0;
//asignación del nombre que corresponde al identificador
if($id==1) $nom="Díaz";
elseif($id==2) $nom="Smith";
elseif($id==3) $nom="Mariscal";
else $nom="Desconocido";
//formateado y reenvío de la respuesta al cliente
echo "Buenos días señor ".$nom;
?>
```

En este contexto, la respuesta HTTP del servidor podría parecerse al siguiente ejemplo, si el parámetro de la solicitud es id=2.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-type: text/plain
Server: GWS/2.1
Date:Tue, 15 May 2013 13:20:24 GMT
Buenos días señor Smith
```

Código 1-13: Ejemplo de una solicitud asíncrona acompañada de un parámetro transmitido con el método GET (este código necesita la declaración previa de la función creationXHR() definida en el código 1-7):

```
###############-MOTOR AJAX-#################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//Declaración de la función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    var resultado = objetoXHR.responseText ;
    …
    //####El resultado pude ahora insertarse en la página HTML
    //si el script del servidor corresponde al del cuadro,
    //el valor de la variable "resultado" será
    //entonces igual a "Buenos días señor Smith"
```

```
   }else {
   alert("Error HTTP N°"+ objetoXHR.status);
   }
  }
}
//----Tratamiento ASINCRONO - GET
//Designación de la function de retrollamada
objetoXHR.onreadystatechange = tratamientoResultado ;
//Configuración del método utilizado, del script servidor con
//los parámetros que se transmiten al servidor
//y finalmente la activación del tipo asíncrono como último parámetro
var numero=2; //simulación de la elección de un número de identificador
objetoXHR.open("get","scriptConParametro.php?id="+numero,true);
//envío de la solicitud
objetoXHR.send(null);
################-FIN DEL MOTOR AJAX-#################
```

Envío de una solicitud asíncrona con un parámetro POST

Otra alternativa para enviar un parámetro al servidor consiste en configurar la solicitud con el método POST. Esta técnica se utiliza generalmente cuando los datos de los parámetros son importantes (más de 512 bytes) que es bastante raro en la práctica.
Por otro lado, este método es menos fácil de implementar porque si desear verificar el buen funcionamiento del script PHP solo, es necesario crear un formulario POST para simular el envío de la solicitud Ajax.

Sin embargo, hay que tener en cuenta que si deseas enviar tus parámetros al servidor en XML y no en formato texto bajo la forma de par variable/valor, será necesario utilizar el método POST para implementar tu aplicación.

El funcionamiento es al menos parecido (ver el código 1-41), aparte del hecho de que el parámetro se pasa como argumento del método send() (como por ejemplo: send(id=2)) y no como un resultado más del URL del archivo del servidor en el segundo parámetro del método open() (para la memoria: scriptConParametro.php?id=2). En el método POST, se debe indicar el tipo de datos enviados a través del método setRequestHeader que afectará el tipo correspondiente al encabezado Content-Type antes de enviar la solicitud (ver el código 1-14).
De hecho, cuando un formulario tradicional en un navegador envía datos con el método POST, el navegador se encarga de afectar automáticamente al valor application/x-www-form-urlencoded en el encabezado Content-type de la solicitud. En el caso de una solicitud Ajax, se debe manualmente llevar a cabo esta afectación, ya que cuando los datos POST llegan al servidor, PHP buscará su tipo de codificación en este encabezado con el fin de analizarlos correctamente.

En el dado del servidor no hay diferencia (el archivo del servidor será por lo tanto como el código 1-12) ya que para la recuperación del parámetro que utilizamos, la variable HTTP $_REQUEST['id'] sirve para recuperar variables enviadas por un cliente, independientemente de su método (a diferencia de $_GET['id'] y $_POST['id'] que se dedican a cada uno de los métodos).

Código 1-14: Ejemplo de una solicitud asíncrona acompañada de un parámetro transmitido con el método POST (este código requiere la declaración previa de la función creationXHR() definida en el código 1-7):

```
################-MOTOR AJAX-##################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//Declaración de la función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    var resultado = objetoXHR.responseText ;
    ....
    //El resultado puede ahora insertarse en la página HTML
    //si el script del servidor corresponde al del cuadro,
    //el valor de la variable "resultado" será
    //entonces igual a "Buenos días señor Smith"
    }else {
    alert("Error HTTP N°"+ objetoXHR.status);
    }
  }
}
//----Tratamiento ASINCRONO - POST
//Designación de la function de retrollamada
objetoXHR.onreadystatechange = tratamientoResultado ;
//Configuración del método utilizado, del script del servidor objetivo
//y finalmente la activación del tipo asíncrono como último parámetro
objetoXHR.open("post","scriptConParametro.php",true);
//Asignación del tipo de codificado de la solicitud enviada
objetoXHR.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
//simulación de la elección del número de identificador
var numero=2;
//envío de la solicitud con un parámetro
objetoXHR.send(id=numero);
################-FIN DEL MOTOR AJAX##################
```

Recuperación del resultado de la solicitud con el responseText o el responseXML

Para recuperar el resultado reenviado por el servidor, hemos utilizado hasta ahora (en la función de retrollamada tratamientoResultado ()) la propiedad responseText del objeto XMLHttpRequest (ver el código 1-15) si la respuesta está en formato texto.

Código 1-15: Recuperación de la respuesta del servidor en una variable resultado.

var resultado = objetoXHR.responseText;

Más adelante veremos que también es posible recuperar resultados más complejos en formato XML. En este caso, se tendrá que utilizar la propiedad responseXML del mismo objeto.

Finalmente, recordemos que el resultado del servidor no está disponible en las propiedades responseText o responseXML después que el ciclo de transferencia HTTP ha terminado y que el estado de la propiedad readyState es igual a 4 (Completed). Es también por esta razón que hemos añadido la prueba del código 1-16 en la función de retrollamada ya que esta se llama en cada una de las 4 modificaciones de la propiedad readyState de un ciclo de transferencia HTTP y queremos recuperar la respuesta del servidor sólo al final de este ciclo.

Código 1-16: Prueba del estado de la propiedad readyState:

```
//---Función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    var resultado = objetoXHR.responseText ;
    }
  }
}
```

Utilización del innerHTML para mostrar el resultado de la solicitud

En los scripts anteriores, hemos recuperado la respuesta del servidor en una variable resultado (ver el código 1-15). Si ahora queremos mostrar en la página, el método más sencillo consiste en utilizar el atributo innerHTML. Este atributo permite reemplazar el contenido de un elemento por una cadena de caracteres que se asigna.
Para hacer esto, primero tenemos que identificar un elemento de la página (por ejemplo, ver el código 1-17) en referencia a su identificador con el método getElementById () y a continuación utilizar el atributo innerHTML como en el siguiente ejemplo:

document.getElementById ("MiMensaje").innerHTML

Esto reemplazará su contenido con la respuesta del servidor mediante la asignación de la propiedad responseText del objeto XMLHttpRequest (ver el código 1-17). Por lo tanto, la respuesta del servidor ("Buenos días", por ejemplo) aparecerá en la misma etiqueta span.

Código 1-17: Visualización del resultado en una marca de la página (código parcial):

```
<script language="JavaScript">
################-MOTOR AJAX-#################
//instanciación del objeto XMLHttpRequest
var objetoXHR = creationXHR();
//Declariación de la función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    document.getElementById("miMensaje").innerHTML=objetoXHR.responseText ;
    }
  }
}
//----Tratamiento ASINCRONO - GET
//Designación de la función de retrollamada
objetoXHR.onreadystatechange = tratamientoResultado ;
//Configuración del objeto XHR
objetoXHR.open("get","script.php",true);
//envío de la solicitud
objetoXHR.send(null);
################-FIN DEL MOTOR AJAX-#################
</script>
…
//antes de la asignación :
<span id="mensaje"></span>
//después de la asignación :
<span id="mensaje"> Buenos días </span>
```

Debemos notar, sin embargo, que la propiedad innerHTML no está estandarizada por el W3C y que es mejor utilizar los métodos DOM incluso si son más complejos de implementar.

Utilización de un controlador de eventos para activar el envío de la solicitud

Para que se ejecute el motor Ajax, es necesario activar (desencadenar) su llamado en la página HTML de la aplicación. Para ilustrar esto, proponemos establecer un controlador de eventos muy simple que llamará a una función que contiene el motor Ajax. En nuestro ejemplo, usaremos el controlador de eventos onkeypress para que la función muestraMensaje() (reagrupando el motor Ajax) se invoca cuando el usuario presiona un botón cualquier del teclado.

Obviamente en futuras aplicaciones, se puede activar de la misma manera la llamada del motor Ajax con uno de los muchos controladores de eventos que JavaScript pone a tu disposición.

Código 1-18: Activación del motor Ajax usando el controlador de eventos onekeypress:

```
<script language="JavaScript">
//-----Controlado de eventos
window.onkeypress = muestraMensaje;
###############-MOTOR AJAX-#################
//Declaración del objeto
var objetoXHR;
//Declaración de la función de retrollamada
function tratamientoResultado() {
  if(objetoXHR.readyState==4) {
    if(objetoXHR.status==200) {
    document.getElementById("miMensaje").innerHTML=objetoXHR.responseText ;
    }
  }
}
//Declaración de la función de llamada del motor Ajax
function muestraMensaje() {
  //Instanciación del objeto XMLHttpRequest
  objetoXHR = creationXHR();
  //Designación de la función de retrollamada
  objetoXHR.onreadystatechange = tratamientoResultado ;
  //Configuración del objeto XHR
  objetoXHR.open("get","script.php",true);
  //Envío de la solicitud
  objetoXHR.send(null);
}
###############-FIN DEL MOTOR AJAX-#################
</script>
…
//antes de que se haya presionado una tecla :
<span id="mensaje"></span>
//después de haber presionado una tecla :
<span id="mensaje"> Buenos días </span>
…
```