Iniciación en Ajax

Sumario

- 1 Introducción a AJAX
- 2 Ajax Nativo
- 3 Iniciando a solicitude
- 4 **A ter en conta** Unha restricción moi importante en AJAX é que na url soamente poderemos escribir recursos que se atopen no mesmo dominio onde estamos executando a aplicación ajax. Polo tanto non poderemos facer peticións a recursos externos ó noso dominio.
    - 4.1 Xestionando o progreso da petición
    - 4.2 Obtendo a resposta

Introducción a AJAX

A historia de AJAX está intimamente relacionada cun obxecto de programación chamado XMLHttpRequest. A orixe deste obxecto remóntase ao ano 2000, con produtos como Exchange 2000, Internet Explorer 5 e Outlook Web Access.

Todo comezou en 1998, cando Alex Hopmann e o seu equipo atopábanse desenvolvendo a entón futura versión de Exchange 2000. O punto débil do servidor de correo electrónico era o seu cliente vía web, chamado OWA (Outlook Web Access).

Durante o desenvolvemento de OWA, avaliáronse dúas opcións: un cliente formado só por páxinas HTML estáticas que se recargaban constantemente e un cliente realizado completamente con HTML dinámico ou DHTML. Alex Hopmann puido ver as dúas opcións e decantouse pola baseada en DHTML. Con todo, para ser realmente útil a esta última faltáballe un compoñente esencial: "algo" que evitase ter que enviar continuamente os formularios con datos ao servidor.

Motivado polas posibilidades futuras de OWA, Alex creou nun só fin de semana a primeira versión do que denominou XMLHTTP. A primeira demostración das posibilidades da nova tecnoloxía foi un éxito, pero faltaba o máis difícil: incluír esa tecnoloxía no navegador Internet Explorer.

Se o navegador non incluía XMLHTTP de forma nativa, o éxito do OWA reduciuse enormemente. O maior problema é que faltaban poucas semanas para que se lanzase a última beta de Internet Explorer 5 previa ao seu lanzamento final. Grazas aos seus contactos na empresa, Alex conseguiu que a súa tecnoloxía se incluíse na libraría MSXML que inclúe Internet Explorer.

De feito, o nome do obxecto (XMLHTTP) elixiuse para ter unha boa escusa que xustificase a súa inclusión na libraría XML de Internet Explorer, xa que este obxecto está moito máis relacionado con HTTP que con XML.

Uns anos máis tarde presentouse o termo AJAX por primeira vez no artigo "Ajax: A New Approach to Web Applications" publicado por Jesse James Garrett o 18 de Febreiro de 2005. Até ese momento, non existía un termo normalizado que fixese referencia a un novo tipo de aplicación web que estaba a aparecer.

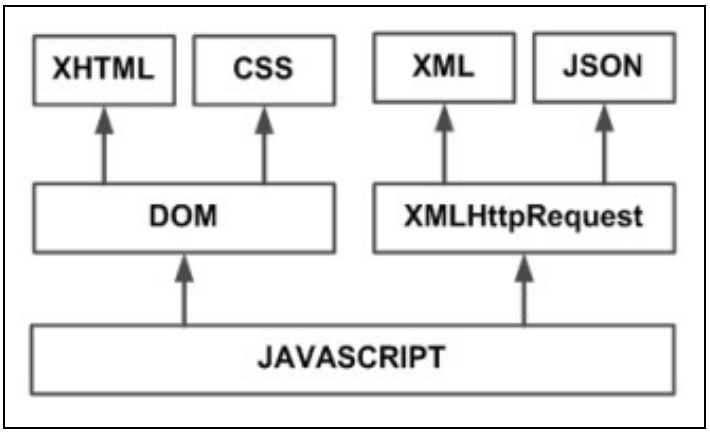
En realidade, o termo AJAX é un acrónimo de Asynchronous JavaScript XML, que se pode traducir como "JavaScript asíncrono XML".

O artigo define AJAX da seguinte forma:

"Ajax non é unha tecnoloxía en si mesmo. En realidade, trátase de varias tecnoloxías independentes que se unen de formas novas e sorprendentes."

As tecnoloxías que forman AJAX son:

- XHTML e CSS, para crear unha presentación baseada en estándares.
- DOM, para a interacción e manipulación dinámica da presentación.
- XML, XSLT e JSON, para o intercambio e a manipulación de información.
- XMLHttpRequest, para o intercambio asíncrono de información.
- JavaScript, para unir todas as demais tecnoloxías.



O corazón de AJAX é o obxecto XMLHttpRequest que nos permite realizar unha conexión ao servidor, enviarlle unha petición e recibir a resposta que logo procesaremos no noso código Javascript. Estamos entón a falar do verdadeiro motor de Ajax; por exemplo grazas a este obxecto podemos dende unha páxina HTML ler datos dunha Web ou enviar datos dun formulario sen necesidade de recargar a páxina.

10 razóns para usar AJAX:

1. Baseado nos estándares abertos
2. Usabilidade
3. Válido en calquera plataforma e navegador
4. Beneficia as aplicacións Web
5. Non é difícil a súa utilización
6. Compatible con Flash
7. Adoptado polos "grandes compañías" da tecnoloxía Web
8. Web 2.0
9. É independente do tipo de tecnoloxía de servidor que se utilice
10. Mellora a estética da Web

A maneira máis fácil para comprender realmente a funcionalidade de Ajax é ver como funciona unha aplicación Web con Ajax e como unha sen Ajax.

Sen Ajax

Crearíase unha páxina cun formulario, cando o usuario envía os datos do formulario prodúcese unha conexión á base de datos e móstrase por pantalla a páxina que o servidor devolve, todo isto fai que se recargue a páxina xa sexa saltando a unha diferente ou a ela mesma, o usuario debe esperar unha nova carga de páxina despois de cada envío. Este proceso é lento porque debe descargar a información HTML por duplicado.

Con Ajax

Utilizariamos un código Javascript que crearía o mencionado obxecto XMLHttpRequest. Cando enviamos o formulario, esta chamada prodúcese de forma asíncrona o que significa que se envían os datos e non se recarga a páxina. Cando o servidor responde unha función Javascript é a que valora a resposta do servidor, se esta resposta é a desexada imprimiremos o texto que indique ao usuario que os seus datos foron enviados correctamente.

O navegador non recarga a páxina, a experiencia desde o punto de vista do usuario é moi satisfactoria posto que se asemella á resposta do típico software de escritorio, xa non temos que enlazar páxinas senón enviar e recibir datos nunha mesma páxina que mediante funcións avaliará as diferentes respostas.

É bastante máis rápido xa que non ten que descargar de novo o código HTML da páxina de confirmación do formulario.

Ajax Nativo

Nesta sección imos a ver unha pequena introducción o que é a programación en Ajax sin empregar ningún tipo de librería ou axuda.

Como se comentou na sección anterior Ajax está baseado no obxecto XMLHttp, o cal nos permite facer peticións en background a un servidor e xestionar os resultados devoltos polo mesmo, e todo elo sin ter que recargar a páxina do noso navegador.

Exemplo de creación dun obxecto XMLHttpRequest:

```
<script language="javascript">
var xhr=false;
try
{
xhr= new ActiveXObject("Msxml2.XMLHTTP");
}
catch (a)
{
try {
xhr= new ActiveXObject("Microsoft.XMLHTTP");
}
catch (b) {
xhr= false;
}
}
if (!xhr && typeof XMLHttpRequest!='undefined')
{
xhr= new XMLHttpRequest();
}
</script>
```

Todo o código anterior o único que fai é intentar crear un obxecto XMLHttp.

A razón de empregar tantas liñas é por que dependendo da versión dos diferentes navegadores de Microsoft hai que crear o obxecto dun xeito ou outro (new ActiveXObject). Por iso o código intenta crear o obxecto con *new ActiveXObject("Msxml2.XMLHTTP")* se esa creación produce un erro, por que o navegador non a soporta entón intenta facelo con *new ActiveXObject("Microsoft.XMLHTTP");*.

Se non puido crear o obxecto mediante new ActiveXObject entón trátase dun navegador que non é Internet Explorer e intenta facelo coa instrucción *new XMLHttpRequest();*

Outro exemplo de creación de obxecto XMLHttp:

```
var xhr;
if (window.XMLHttpRequest) {
  xhr = new XMLHttpRequest();
}
else if (window.ActiveXObject) {
  xhr = new ActiveXObject("Msxml2.XMLHTTP");
}
else {
  throw new Error("Ajax non está soportado neste navegador.");
}
```

Unha vez creado o obxecto XMLHttp (xhr)teremos á nosa disposición unha serie de propiedades e métodos que nos permitirán facer as peticións ó servidor.

Tabla de métodos e propiedades do obxecto XMLHttp:

Métodos:

- abort() - Detén a petición en curso.
- getAllResponseHeaders() - Devolve todas as cabeceiras da resposta (etiquetas e valores) como unha cadea.
- getResponseHeader(etiqueta) - Devolve o valor da etiqueta nas cabeceiras da resposta.
- open(método,URL,asíncrona,nome,password) - Abre unha conexión con esa URL mediante ese método (GET ou POST).
- send(contido) - Envía o contido ao servidor.
- setRequestHeader(etiqueta,valor) - Establece o valor dunha etiqueta das cabeceiras de petición.

Propiedades:

- onreadystatechange - Contén o nome da función que se executa cada vez que o estado da conexión cambie.
- readyState - Estado da conexión, pode valer desde 0 (non iniciada), 1 (cargando), 2(cargado), 3(interactivo) e 4 (rematado).
- responseText - Datos devoltos polo servidor en formato cadea.

- responseXML - Datos devoltos polo servidor en forma de documento XML que pode ser percorrido mediante as funcións do DOM (getEementsByTagName, etc).
- status - Código enviado polo servidor, do tipo 404 (documento non atopado) ou 200 (OK). Máis información sobre os diferentes estados.
- statusText - Mensaxe de texto enviado polo servidor xunto ao código (status), para o caso de código 200 conterá "OK".

Coñecer estas propiedades e métodos é algo moi útil á hora de desenvolver aplicacións utilizando Ajax debido á gran axuda que moitas delas ofrecen á hora de depurar erros. E dános unha maior idea acerca da potencia desta conxunción de tecnoloxías.

De está lista deterémonos no método open() que é un dos máis utilizados e o que nos permitirá utilizar a mellor característica de Ajax que é a carga de datos externos á páxina sen necesidade de recargar a mesma.

Iniciando a solicitude

Antes de que podamos enviar unha petición ao servidor, necesitamos facer o seguintes pasos previos de configuración:

1. Especificar o método HTTP de envío de datos (POST ou GET)
2. Proporcionar a URL do servidor onde enviaremos eses datos
3. Programar como nos informará o obxecto XHR (XmlHttpRequest) do estado da petición en curso
4. Facer a petición ó servidor

As **opcións 1 e 2** as faremos coa seguinte instrucción:

```
xhr.open('GET','/algun/recurso/url');
```

A ter en conta

Unha restricción moi importante en AJAX é que na url soamente poderemos escribir recursos que se atopen no mesmo dominio onde estamos executando a aplicación ajax. Polo tanto non poderemos facer peticións a recursos externos ó noso dominio.

Atención este método non causa que a petición sexa enviada ó servidor. O único que realiza e configurar os parámetros da petición.

Ó método tamén se lle pode pasar un terceiro parámetro de tipo boolean que especifica se a petición vai ser asíncrona (true por defecto, para uso con ajax) ou síncrona (false - funcionaría do xeito tradicional).

Na **opcion 3** proporcionamos un medio para que o obxecto XHR nos indique que está ocurrindo coa petición. Eso faise asignando á propiedade **onreadystatechange** do obxecto, unha función de retorno. Esa función de retorno será invocada polo obxecto XHR cada vez que se produza un cambio de estado na solicitude feita ó servidor.

Na **opcion 4** completaremos o proceso para facer a petición de envío. Esto faise empregando o método **send()**. Este método terá como parámetro **null** si estamos empregando o **método GET** (paso 1) para envío de datos (xa que estes son enviando formando parte da url (exemplo: paxina.php?nome=rafa&apelidos=veiga).

Si estamos empregando o **método POST** (paso 1) para envío de datos, entón no método **send()** poremos como **parámetros** os datos que queremos enviar (exemplo: send('nome=rafa&apelidos=veiga'))

Xestionando o progreso da petición

O obxecto XHR infórmanos do seu progreso a través da función de retorno que indicamos na configuración anterior.

Cando enviamos a petición ó servidor con send(), ésta función será chamada varias veces, xustamente cada vez que se produza un cambio de estado (**onreadystatechange**) na solicitude. O estado actual da solicitude estará dispoñible na propiedade **readyState** (os 4 estados citados na sección anterior).

Exemplo de programación do progreso da petición:

```
xhr.onreadystatechange = function()
{
```

```
   if (xhr.readyState == 4)  //soamente terá en conta cando chegue ó estado 4 (rematado).
       if (xhr.status == 200)
     // (si a resposta do servidor é 200 e que rematou a execución sin erros no servidor.
          {
          //Aquí xestionaríamos a resposta recibida do servidor
          }
       else
          {
          //Produciuse un erro no servidor, como páxina non atopada, error sintáctico php, etc...
          }
}
```

Obtendo a resposta

Dependendo de cómo o servidor envíe a resposta teremos diferentes propiedades para acceder a dita resposta.

Por exemplo se o servidor envía a resposta en formato XML empregaremos a propiedade **responseXML** do obxecto XHR.

Para calquera outro tipo de resposta podemos empregar a propiedade **responseText** do obxecto XHR.

Isto quere dicir que en calquera desas dúas propiedades teremos o contido que nos devolveu o servidor.

Exemplo completo:

```
// Creamos o obxecto xhr que será do tipo XMLHttpRequest

var xhr;
if (window.XMLHttpRequest) {
  xhr = new XMLHttpRequest();
}
else if (window.ActiveXObject) {
  xhr = new ActiveXObject("Msxml2.XMLHTTP");
}
else {
  throw new Error("Ajax non está soportado neste navegador.");
}

// Programamos a función que se executará en cada cambio de estado da solicitude ó servidor.

xhr.onreadystatechange = function()
{
   if (xhr.readyState == 4)  //soamente terá en conta cando chegue ó estado 4 (rematado).
       if (xhr.status == 200)
       //  (si a resposta do servidor é 200 e que rematou a execución sin erros no servidor.
          {
                 document.getElementById("contido").innerHTML=xhr.responseText;
                 // Asignamos a un DIV con id "contido" o texto que recibimos do servidor.
          }
       else
          {
        //Produciuse un erro no servidor, como páxina non atopada, error sintáctico no php, etc...
          }
}

// Facemos o envio

xhr.open('GET','/paxina.php?nome=Rafa&apelidos=Veiga');
xhr.send(null);
```

Pois ben, con jQuery todas as liñas que programamos no exemplo anterior quedarían reducidas a 1 liña:

```
$('#contido').load('/paxina.php?nome=Rafa&apelidos=Veiga');
```

Aquí si que observamos a grande potencia dos comandos para empregar Ajax con jQuery.

--Veiga (discusión) 23:08 26 ene 2015 (CET)