Acceso eficiente al servidor (AJAX)

Realizado por A.Garay (Dpto. de Informática)

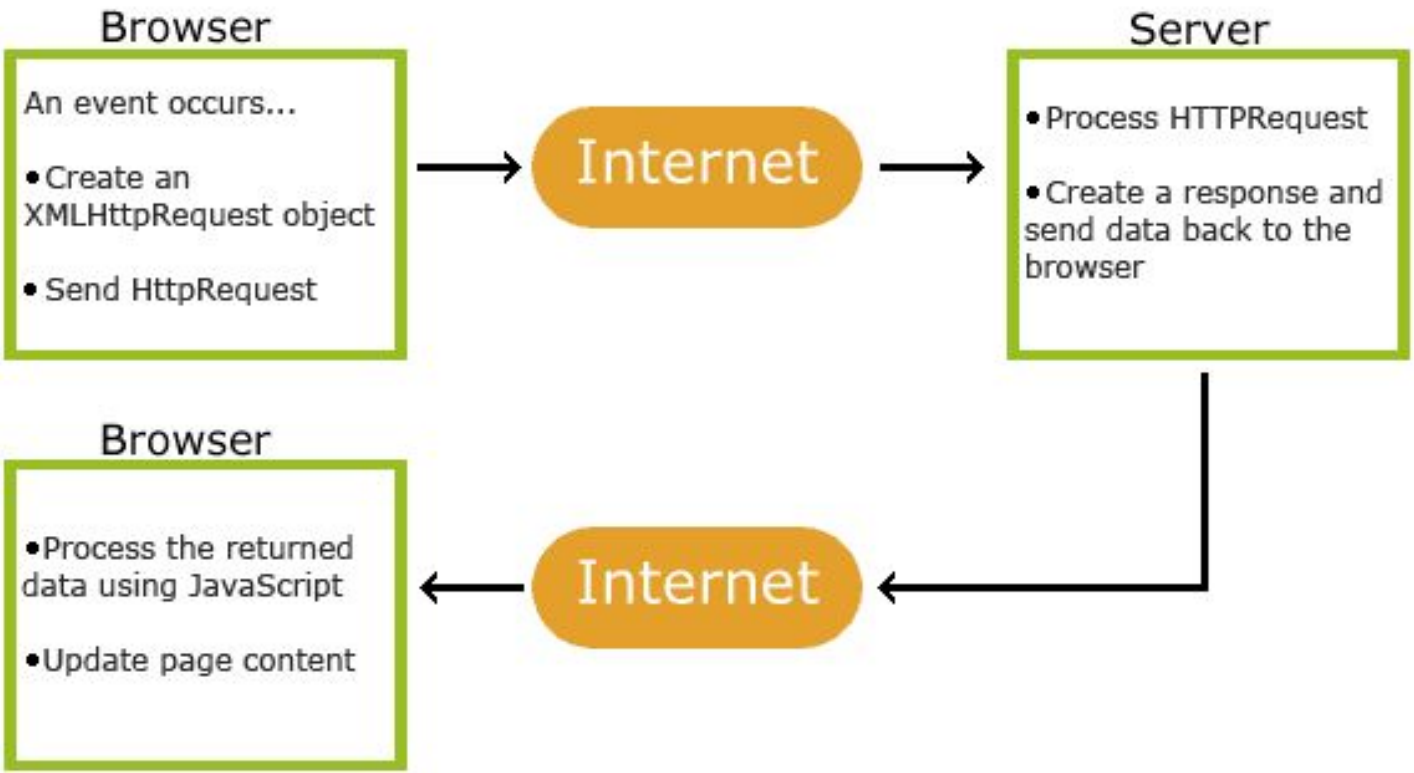
Resumen de contenidos

- Conceptos y funcionamiento
- Ejemplo sencillo
- Protección ante ejecuciones no-Ajax
- Enviando más información al servidor
- Recibiendo información en XML

AJAX: conceptos

- AJAX: Asynchronous Javascript And Xml
- Son técnicas del lado cliente, utilizando **javascript** para conseguir que el contenido de una página cambie, con información proveniente del servidor, pero sin necesidad de recargarse completamente de nuevo.
- Es **asíncrono** porque se pueden actualizar varias partes de la página simultáneamente, sin necesidad de que acabe la actualización de una para que empiece la siguiente.
- Y es **XML** porque la información que proviene del servidor suele formatearse en XML, para que el cliente la pueda entender, aunque podría venir en texto plano, HTML o en otros formatos como JSON

Funcionamiento de AJAX



Ejemplo (1/3): HTML cliente

```
<head>
<script type="text/javascript">
function usarAJAX()
{
    // Código javascript
    // para actualizar datos de
    // la página utilizando AJAX
}
</script>
</head>
<body>
    <h2>Texto fijo</h2>
    <div id="idDiv">
        <h2>Texto cambiante (AJAX)</h2>
    </div>
    <button type="button" onclick="usarAJAX()">Cambiar</button>
</body>
```

Ejemplo (2/3): JavaScript cliente

```
function usarAJAX()
{
    //Objeto javascript que manejará la petición
    var xmlhttp=new XMLHttpRequest();

    // Se envía la petición AJAX al script "ajax.php" vía GET
    xmlhttp.open("GET","ajax.php",true);
    xmlhttp.send();

    // Cuando se tenga la respuesta del servidor,
    // asíncronamente se ejecutará este código.
    xmlhttp.onreadystatechange=function()  {
      if (xmlhttp.readyState==4 && xmlhttp.status==200) {
        //responseText contiene la salida del servidor
        document.getElementById("idDiv")
            .innerHTML=xmlhttp.responseText;
      }
    }
}
```

Creación del objeto XMLHttpRequest

Envío de la petición al servidor

Manejo de la respuesta del servidor

En este ejemplo hemos cambiado el contenido de un , pero lo más habitual hubiera sido analizar datos recibidos en algún formato: XML, JSON, etc., y actuar en consecuencia...

Ejemplo (3/3) PHP servidor

```php
<?php
    echo "<h1>VIVA AJAX</h1>";
?>
```

- En este caso el servidor envía código HTML “normal y corriente”, pero podría haber enviado texto aún más simple, sin marcas, o texto más complejo: XML, etc.
- El cliente es el responsable de entender en qué formato está escrita y qué hacer con la respuesta del servidor, que siempre es texto plano.
- Como lo habitual es que lo que envía el servidor son datos que ha consultado de una Base de Datos, el formato más habitual de envío es XML, ya que hay funciones en javascript del lado cliente que son capaces de navegar por esos datos en XML.
- Últimamente se utiliza más el formato JSON porque para encapsular datos de objetos, ocupa mucho menos que XML, que es demasiado genérico.

¿Cómo impedir ejecución de código en el servidor pensado para servir peticiones AJAX?

CLIENTE JAVASCRIPT: Enviando una cabecera en la petición, identificándose como una petición AJAX (algunos frameworks y navegadores la envían automáticamente)

```
xmlhttp.open("GET","ajax.php",true);
xmlhttp.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
xmlhttp.send();
```

SERVIDOR (PHP): Comprobando si existe esa cabecera, como requisito previo a la ejecución del script

```
<?php
$esAjax = isset (
        $_SERVER ['HTTP_X_REQUESTED_WITH'] ) ?
        strtolower ( $_SERVER ['HTTP_X_REQUESTED_WITH'] ) == 'xmlhttprequest' :
        false;
if ($esAjax) {
    echo "<h1>VIVA AJAX</h1>";
} else {
    echo "Sólo para ejecuciones AJAX";
}
?>
```

Enviando más información al servidor

- Vía GET
  - xmlhttp.open("GET", "ajax.php?dato1=valor1&dato2=valor2", true);
- Vía POST
  - xmlhttp.open("POST", "ajax.php", true);
  - xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
  - xmlhttp.send("dato1=valor1&dato2=valor2");

Serialización de formularios

- Utilizando “serialize.min.js”

```
<script type="text/javascript"
src="https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/form-serialize/serialize-0.2.min.js" ></script>
...
var datosSerializados = serialize(document.getElementById("idFormulario")));
xmlhttp.open("GET", "ajax.php?" + datosSerializados, true);
xmlhttp.send();
...
```

- Utilizando el objeto FormData (*)

```
...
var datosSerializados = new FormData(document.getElementById("idFormulario"));
xmlhttp.open("POST", "ajax.php", true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send(datosSerializados);
```

Recibiendo información en XML

- xmlDoc = (new DOMParser()).
  parseFromString(xmlhttp.responseText, "application/xml");
  - A veces basta con
    - xmlDoc = xmlhttp.responseXML;
  - xmlDoc es un nodo también (el nodo raíz)
- miNodo.**getElementsByTagName("etiquetaXML")**
  - Devuelve un array de nodos (hijos de "miNodo") con ese tag.
  - Se accede a cada nodo del array por su posición.
  - Se puede iterar con un "*for + arrayNodos.length*" o un "*for (v of arrayNodos)*"
- miNodo.**childNodes**
  - Contiene un array de nodos hijos de miNodo
- miNodo.**nodeValue**
  - Devuelve el valor de "miNodo" siempre que éste sea un nodo de tipo "texto"
- miNodo.**getAttribute("nombreAtributo")**
  - Devuelve el valor de un atributo de un nodo