APRENDERAPROGRAMAR.COM

DIFERENCIA ENTRE GET Y POST CON AJAX. SETREQUESTHEADER. ENCODEURICOMPONENT. VALIDAR FORMULARIO EN SERVIDOR (CU01212F)

Sección: Cursos
Categoría: Tutorial básico del programador web: Ajax desde cero

Fecha revisión: 2031

Resumen: Entrega nº12 del Tutorial básico "Ajax desde cero".

Autor: Alex Rodríguez

DIFERENCIAS ENTRE POST Y GET CON AJAX

En entregas anteriores del curso hemos visto ejemplos de código con los que recuperábamos información desde el servidor usando el método GET y Ajax. Vamos a explicar cómo podemos utilizar el método POST para recuperar información con Ajax.

REPASO RÁPIDO DE GET Y POST

GET y POST son dos métodos del protocolo HTTP que podemos ver como formas de envío de datos a través de internet. Tanto GET como POST son muy conocidos debido a su uso en formularios.

Por ejemplo: <form action="http://www.aprenderaprogramar.com/prog/newuser" method="get"> indica que los datos del formulario serán enviados a la url especificada usando el método get.

Un resultado usando el método GET, a modo de ejemplo, podría ser el siguiente:

http://www.aprenderaprogramar.com/newuser.php?nombre=Pepe&apellido=Flores&email=h52turam%40uco.es&sexo=mujer

El símbolo ? indica dónde empiezan los parámetros que se reciben desde el formulario que ha enviado los datos a la página.

Después del símbolo ? aparecen parejas de datos con su nombre y valor separadas por el símbolo &. Las parejas dato1=valor1, dato2=valor2, dato3=valor3... reflejan el nombre y el valor de los campos enviados por el formulario.

Si en un formulario usamos method="post" los datos se transmiten igualmente, pero no son visibles en la url y sólo pueden ser recuperados a través de código de programación.

REPASO RÁPIDO DE DIFERENCIAS ENTRE GET Y POST

En la siguiente tabla resumimos diferencias entre el uso de get y post:

Aspecto	Con GET	Con POST
Los datos son visibles en la url	Sí	No
Los datos pueden permanecer en el historial del navegador	Sí	No

Aspecto	Con GET	Con POST
Una url puede ser guardada conteniendo parámetros de un envío de datos	Sí	No
Existen restricciones en la longitud de los datos enviados	Sí (no se puede superar la longitud máxima de una url)	No
Se considera preferible para envío de datos sensibles (datos privados como contraseñas, números de tarjeta bancaria, etc.)	No (los datos además de ser visibles pueden quedar almacenados en logs)	Sí (sin que esto signifique que por usar post haya seguridad asegurada)
Codificación en formularios	application/x-www-form-urlencoded	application/x-www-form-urlencoded ó multipart/form-data. Se usa multipart para envío de datos binarios, por ejemplo ficheros.
Restricciones de tipos de datos	Sí (sólo admite caracteres ASC-II)	No (admite tanto texto como datos binarios p.ej. archivos)
Se considera preferible para disparar acciones	No (podría ser accedido por un robot que dispararía la acción)	Sí (sin que esto garantice que no pueda acceder un robot)
Riesgo de cacheado de datos recuperados en los navegadores	Sí	No
Posibles ataques e intentos de hackeo	Sí (con más facilidad)	Sí (con menos facilidad)

Por motivos de seguridad se recomienda usar GET a los efectos de únicamente **recuperar información** desde el servidor. Por el contrario, si se desea realizar operaciones como actualización, borrado, etc. no se recomienda usar GET.

Hemos de prestar atención a una cuestión en relación al uso de estos métodos con Ajax: los datos recuperados usando GET **pueden quedar cacheados** en los navegadores. Esto significa que si repetimos una petición para recuperar datos con GET que ha sido realizada anteriormente, podemos obtener los mismos resultados en el navegador incluso si los datos han cambiado en el servidor (debido al cacheado en el navegador).

En relación a Ajax, la cuestión de visibilidad en la url no resulta relevante (ya que en cualquier caso, la invocación de la url se producirá en segundo plano). Sin embargo, los aspectos de seguridad sí siguen siendo relevantes, por eso debemos tener en cuenta:

- Usaremos GET sólo para operaciones que impliquen **consulta/recuperación de datos** siempre que valoremos que el cacheado en el navegador no causará problemas.

- Usaremos POST para operaciones que impliquen **modificación en el servidor** (por ejemplo borrado de datos, actualización, etc.). También puede usarse POST para consulta/recuperación de datos.

Si se quiere usar get y asegurar evitar posibles problemas de cacheado se puede usar un pequeño truco: añadir un parámetro siempre distinto a la url. La idea podría basarse en un pequeño código similar a este:

```
var bustCache = '?' + new Date().getTime();
oReq.open('GET', url + bustCache, true);
```

REALIZAR PETICIONES AJAX CON POST

En una entrega anterior del curso hemos estudiado que cuando se producen peticiones y respuestas entre servidor y cliente los mensajes llevan cabeceras con cierta información.

Para realizar una petición GET con Ajax no hemos de establecer ninguna cabecera específicamente. En cambio con POST sí hemos de hacerlo. Los parámetros a enviar mediante POST con Ajax se construyen como una cadena de texto dato1=valor1&dato2=valor2& ... &daton=valorN

El esquema para usar post con Ajax sería el siguiente:

```
// Creamos el objeto XmlHttpRequest
// Creamos la cadena de parámetros a pasar: cadenaParams = " dato1=valor1&dato2=valor2& ..."
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send(cadenaParams);
```

Para que el servidor procese la petición mediante POST **es obligatorio escribir el setRequestHeader** tal y como hemos indicado.

Además hemos de construir la cadena con los parámetros a enviar, y enviar dicha cadena como argumento del método send del objeto XmlHttpRequest.

Recordar que cuando usábamos el método GET la invocación Ajax era simplemente xmlhttp.send(); que es como si escribiéramos send(null);

Sin embargo ahora si usamos POST la invocación Ajax será xmlhttp.send(parametros);

MÉTODO ENCODEURICOMPONENT

Para evitar problemas con espacios o caracteres especiales se recomienda construir la cadena de parámetros aplicando previamente el método JavaScript encodeURIComponent. El uso de este método se basa en la sintaxis: encodeURIComponent(str);

Donde str es el parámetro a pasar.

Ejemplo de uso:

str = "name=" + encodeURIComponent(name.value) + "&email=" + encodeURIComponent(email.value);

EJEMPLO RECUPERACIÓN DATOS POR POST CON AJAX

Crea un archivo denominado datosCU01212F.php y súbelo al servidor con este contenido:

```
<meta charset = "utf-8"/>
<?php // Datos
$pais[0]="spain"; $pais[1]="mexico"; $pais[2]="argentina"; $pais[3]="colombia";
$ciudad[0][0]="Madrid"; $ciudad[0][1]="Barcelona"; $ciudad[0][2]="Valencia"; $ciudad[0][3]="Sevilla";
$ciudad[0][4]="Zaragoza"; $ciudad[0][5]="Málaga"; $ciudad[0][6]="Murcia";
$ciudad[1][0]="México D.F."; $ciudad[1][1]="Ecatepec"; $ciudad[1][2]="Guadalajara"; $ciudad[1][3]="Puebla";
$ciudad[1][4]="Juárez"; $ciudad[1][5]="Tijuana"; $ciudad[1][6]="León"; $ciudad[1][7]="Zapopan";
$ciudad[2][0]="Buenos Aires"; $ciudad[2][1]="Córdoba"; $ciudad[2][2]="Rosario"; $ciudad[2][3]="La Plata";
$ciudad[2][4]="Mar del Plata"; $ciudad[2][5]="San Miguel de Tucumán"; $ciudad[2][6]="Ciudad de Salta";
$ciudad[3][0]="Bogotá"; $ciudad[3][1]="Medellín"; $ciudad[3][2]="Cali"; $ciudad[3][3]="Barranquilla";
$ciudad[3][4]="Cartagena"; $ciudad[3][5]="Cúcuta"; $ciudad[3][6]="Soledad"; $ciudad[3][7]="Ibagué";

// Rescatamos el parámetro pais que nos llega mediante post
$paisRecibido=$_POST["pais"]; $ciudadesDevueltas="";
$existePais = false;
for ($i = 0; $i<count($pais) ; $i++) {     if ($paisRecibido == $pais[$i]) {$indicePais = $i; $existePais=true;} }

$msg = 'El pais recibido por post en segundo plano es '.$paisRecibido ;
if ($existePais) {$msg = $msg. ' y tiene indice '.$indicePais;}
$ciudadesRespuesta = "";

//  Creamos el array a devolver
for ($i = $indicePais; $i<count($ciudad[$indicePais]) ; $i++) {
   $ciudadesRespuesta .= ",".$ciudad[$indicePais][$i];
}

echo $msg.$ciudadesRespuesta;
?>
```

Crea un archivo denominado cursoAjaxCU01212F.html con el contenido que indicamos a continuación y súbelo al servidor:

```
<!DOCTYPE html><html><head><title>Cursos aprende a programar</title><meta charset="utf-8">
 <style type="text/css">
 *{font-family:sans-serif;} a:link {text-decoration:none;} select{font-size:18px;}
 div div {color: blue; background-color:#F1FEC6; font-size: 20px; float:left; border: solid; margin: 20px; padding:15px;}
 </style>
 <script>
function mostrarSugerencia(str) {
var xmlhttp;
var contenidosRecibidos = new Array();
var nodoMostrarResultados = document.getElementById('listaCiudades');
var contenidosAMostrar = '';

if (str.length==0) { document.getElementById("txtInformacion").innerHTML=""; nodoMostrarResultados.innerHTML = '';
return; }

xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
    contenidosRecibidos = xmlhttp.responseText.split(",");
    document.getElementById("txtInformacion").innerHTML=contenidosRecibidos[0];
    for (var i=1; i<contenidosRecibidos.length;i++) {
    contenidosAMostrar = contenidosAMostrar+'<div id="ciudades'+i+'"> <a href="http://aprenderaprogramar.com">' +
contenidosRecibidos[i]+ '</a></div>';
    }
    nodoMostrarResultados.innerHTML = contenidosAMostrar;
    }
}
var cadenaParametros = 'pais='+encodeURIComponent(str);
xmlhttp.open('POST', 'datosCU01212F.php'); // Método post y url invocada
xmlhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded'); // Establecer cabeceras de petición
xmlhttp.send(cadenaParametros); // Envio de parámetros usando POST
}

</script>
</head>
<body style="margin:20px;">
<h2>Elige un país:</h2>
<form action="">
 <select onchange="mostrarSugerencia(this.value)">
  <option value="none">Elige</option>
  <option value="spain">España</option>
  <option value="mexico">México</option>
  <option value="argentina">Argentina</option>
  <option value="colombia">Colombia</option>
</select>
</form>
<br/>
<p>Informacion sobre operacion en segundo plano con POST y Ajax: <span style="color:brown;"
id="txtInformacion"></span></p>
<div id="listaCiudades"> </div>
</body>
</html>
```

Este código es muy similar a uno que utilizamos en entregas anteriores del curso como ejemplo de uso de Ajax con GET. Fíjate que las diferencias son pocas:

- Invocamos una url con método POST y la url carece de parámetros.

- Establecemos la cabecera con setRequestHeader

- Creamos la cadena de paso de parámetros post que añadimos como parámetro en send.

- En la url invocada los datos enviados se recuperan por POST.

Recuerda siempre que uses POST no olvidar incluir el establecimiento de cabecera de petición, ya que en caso de no hacerlo no obtendrás resultado alguno al ignorar el servidor la petición.

RESULTADOS

Al invocar la ruta donde se encuentre el fichero html, que será del tipo http://aprenderaprogramar.com/cursoAjaxCU01212F.html, debes ser capaz de elegir opciones del combobox desplegable y visualizar resultados por pantalla. Por ejemplo, si eliges la opción "Argentina", el resultado esperado será que por pantalla se visualice:

Informacion sobre operacion en segundo plano con Ajax: El pais recibido por post en segundo plano es argentina y tiene indice 2
Rosario La Plata Mar del Plata San Miguel de Tucumán Ciudad de Salta

Visualmente lo que obtenemos es análogo a lo que obteníamos usando get:

Si cambias la selección del país por otro país, debes observar un refresco "casi instantáneo" de modo que se modifica el texto y las ciudades que se muestran, viéndose en cada caso las correspondientes al país seleccionado.

Si eres capaz de visualizar esto, todo ha ido correctamente.

EJERCICIO

Muchas validaciones se realizan del lado del cliente usando JavaScript, pero otras se realizan del lado del servidor (por ejemplo comprobar si un nombre de usuario está libre para dar de alta a un usuario, o comprobar si un correo electrónico está libre para registrar a un usuario en un foro).

Crea un documento HTML que conste de:

a) Un título H1 con el texto “Alta de usuarios”

b) Un formulario con un campo Nombre, otro Apellidos y otro Nombre de Usuario.

c) Una utilidad Ajax que informe si el nombre de usuario ya existe cada vez que el campo correspondiente a nombre del usuario pierda el foco. En ese momento deberá mostrarse un mensaje “Nombre de usuario libre” ó si no está libre, “Nombre de usuario no disponible”.

Para comprobar si ya existe el nombre de usuario debes realizar una invocación al archivo comprobarUsuarios.php donde deberás definir la lista de nombres de usuario existentes como un array php (por ejemplo puedes usar el array nombreExistente[0]='juan'; nombreExistente[1]='pedro'; nombreExistente[2]='alfredo'; nombreExistente[3]='luis';) y añadir el código necesario para obtener la funcionalidad deseada.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01213F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=83&Itemid=212