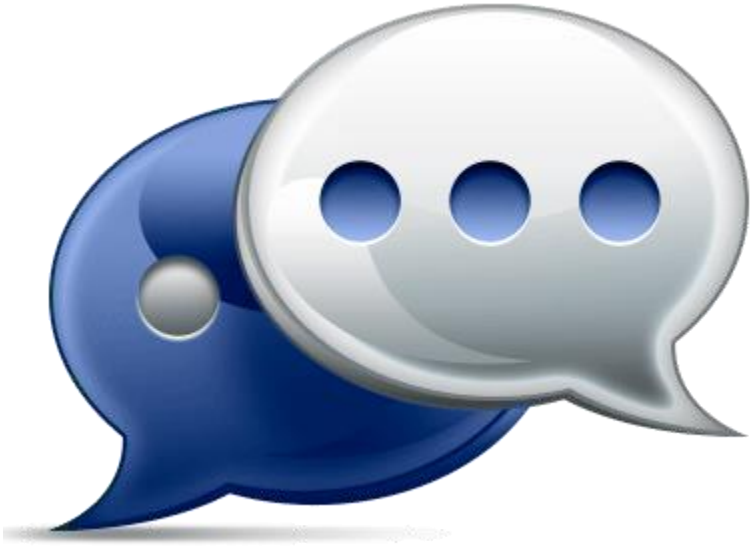
Sistema de comentarios con PHP y jQuery



Internet se ha convertido en una poderosa herramienta donde todo el mundo puede opinar de cualquier cosa, compartiendo sus ideas o sus impresiones sobre algún producto de una tienda online o sobre los servicios que ofrece una determinada empresa, por poner algunos ejemplos. Debido a esto, es habitual que las páginas web ofrezcan a sus visitantes la posibilidad de dejar comentarios en sus publicaciones. Desde acens creemos que esta funcionalidad es importante, por este motivo, en nuestro White Paper de este mes os vamos a explicar cómo crear un sencillo sistema de comentarios con la opción de borrar esa publicación, utilizando PHP y jQuery.

Estructura de la base de datos

Lo primero que veremos será la estructura que tendrá nuestra base de datos, la cual la hemos llamado "**comentarios**". Esta tendrá una única tabla donde se almacenarán los comentarios de los usuarios. La estructura de esta tabla se puede ver en el siguiente código.

SQL

```sql
CREATE TABLE `usuario` (
  `id` int(50) NOT NULL,
  `username` varchar(50) NOT NULL,
  `message` varchar(500) NOT NULL,
  `fecha` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE `usuario`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `usuario`
  MODIFY `id` int(50) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
COMMIT;
```

Como se puede ver en el código anterior, se trata de una tabla que contendrá cuatro campos.

- **id**: Será la clave primaria y su valor será un entero que irá incrementando con cada inserción.
- **username**: Campo donde se guardará el nombre del usuario
- **message**: En este campo se guardará el texto enviado en el comentario.
- **fecha**: Un campo que contendrá la fecha de cuando se hizo el comentario por parte del usuario y que cogerá del sistema de forma automática.

Archivo db.php

El siguiente paso será crear el archivo que se encargue de realizar la conexión a la base de datos. El código encargado de esta acción es el siguiente.

PHP

```php
$conn = mysqli_connect("localhost","root","","comentarios");
if ($conn->connect_error) {
    die("Fallo en la conexión: " . $conn->connect_error);
}
```

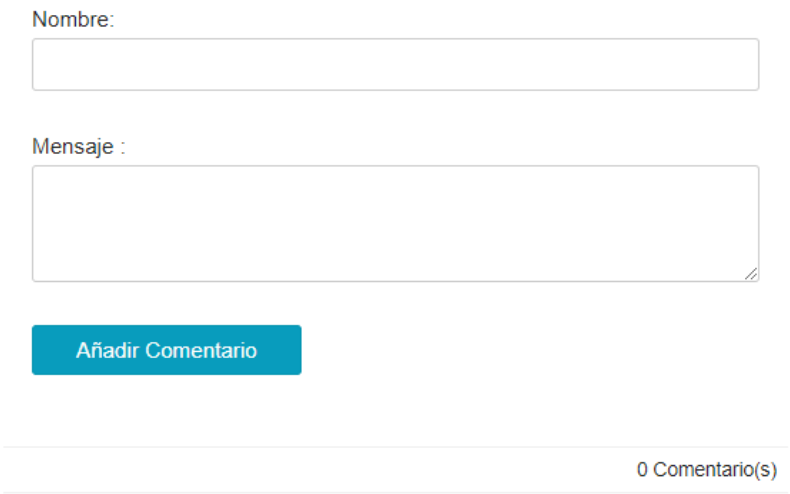
Aquí lo que hacemos es llamar a la función "**mysqli_connect**" de PHP que se encarga de conectar con la base de datos. A esta función se le pasa cuatro parámetros.

- **Host de la base de datos**. En nuestro caso "**localhost**".
- **Usuario de la base de datos**. En nuestro ejemplo nuestro usuario es "**root**".
- **Contraseña del usuario**. Para este ejemplo, el usuario "**root**" no tiene asignada ninguna contraseña.
- **Nombre de la base de datos**. Como hemos dicho anteriormente, nuestra base de datos se llama "**comentarios**".

Archivo index.php

El archivo "**index.php**" realizará varias tareas diferentes que veremos de forma desglosada. A modo de resumen, será el encargado de pintar el formulario de envío de comentarios, mostrará los comentarios que haya en base de datos y también lanzará las funcionalidades de inserción y eliminación de comentarios. Veamos poco a poco todo su contenido.

Formulario de recogida de comentarios



La apariencia que tendrá el formulario es la que se ve en la imagen superior. El código encargado de pintar este formulario es el siguiente.

HTML

```
<form action=" " id="frmComment" method="post">
    <div class="row">
        <label> Nombre: </label> <span id="name-info"></span><input class="form-field"
id="name" type="text" name="user">
    </div>
    <div class="row">
        <label for="mesg"> Mensaje : <span id="message-info"></span></label>
        <textarea class="form-field" id="message" name="message" rows="4"></textarea>
    </div>
    <div class="row">
        <input type="hidden" name="add" value="post" />
```

```
      <button type="submit" name="submit" id="submit" class="btn-add-comment">Añadir
Comentario</button>
        <img src="LoaderIcon.gif" id="loader" />
     </div>
  </form>
```

Como se puede ver, se trata de un formulario que está compuesto por dos campos. El primero de ellos de tipo "**text**" y que hace referencia al nombre del usuario. El segundo de los campos es de tipo "**textarea**" y aquí el usuario es donde puede indicar el texto a publicar. Además de estos dos campos, también se ha colocado un botón que se encargará de lanzar el JavaScript encargado de hacer la llamada al archivo PHP que guardará la información en base de datos.

Recuperar los comentarios de la base de datos

Otra de las acciones que se llevan a cabo dentro del archivo "**index.php**", es la de recuperar los comentarios que hay almacenados en la base de datos cuando se accede a la página. El código encargado de realizar esta tarea es el siguiente.

PHP

```
<?php
include_once 'db.php';

$sql_sel = "SELECT * FROM usuario ORDER BY id DESC";
$result = $conn->query($sql_sel);
$count = $result->num_rows;

    if($count > 0) {
?>
        <div id="comment-count">
        <span id="count-number"><?php echo $count;?></span> Comentario(s)
        </div>
<?php } ?>
        <div id="response">
<?php
while ($row = $result->fetch_array(MYSQLI_ASSOC)){
?>
            <div id="comment-<?php echo $row["id"];?>" class="comment-row">
                <div class="comment-user"><?php echo $row["username"];?></div>
                <div class="comment-msg" id="msgdiv-<?php echo $row["id"];?>"><?php echo
$row["message"];?></div>
                <div class="delete" name="delete"
                    id="delete-<?php echo $row["id"];?>"
                    onclick="deletecomment(<?php echo $row["id"];?>)">Eliminar</div>
            </div>
<?php
}
?>
```

Lo primero que hacemos es llamar al archivo que contiene el código que realiza la conexión a la base de datos y que hemos visto anteriormente.

PHP

```
include_once 'db.php';
```

A continuación se ejecuta una sentencia SQL que se trae todos los registros de la tabla "usuarios" de la base de datos.

PHP
```
$sql_sel = "SELECT * FROM usuario ORDER BY id DESC";
$result = $conn->query($sql_sel);
$count = $result->num_rows;
```

En la variable **"$count**", se guarda el número de registros que hay en base de datos. Si el valor es mayor que cero, entonces se pinta cada uno de los comentarios añadiéndole la función de borrado. Para ello hay que recorrer la variable **"$result**" que contiene todos los registros. Esto lo llevamos a cabo con la siguiente línea.

PHP
```
while ($row = $result->fetch_array(MYSQLI_ASSOC)){
```

En cada pasada del bucle, en la variable "**$row**" se guardará un registro de la base de datos que luego utilizaremos para pintar el comentario. El bloque que se encarga de mostrar en pantalla cada comentario y que está dentro del bucle anterior es el siguiente.

PHP y HTML
```
<div id="comment-<?php echo $row["id"];?>" class="comment-row">
   <div class="comment-user"><?php echo $row["username"];?></div>
   <div class="comment-msg" id="msgdiv-<?php echo $row["id"];?>"><?php echo
$row["message"];?></div>
   <div class="delete" name="delete" id="delete-<?php echo $row["id"];?>"
onclick="deletecomment(<?php echo $row["id"];?>)">Eliminar</div>
</div>
```

En el código anterior, podemos destacar el evento "**onclick**" asignado a la opción eliminar. Este evento será el encargado de hacer la llamada al método "**deletecomment**" que a su vez, mediante AJAX, lanza la llamada al PHP que borrará el comentario de la base de datos.

Eliminar comentario

Otra de las secciones que nos encontramos dentro del "**index.php**" es el código JavaScript que se encarga de lanzar el borrado de un comentario mediante la llamada a un PHP utilizando tecnología AJAX. El código que se encarga de realizar esto es el siguiente.

JS
```
function deletecomment(id) {

        if(confirm("¿Estás seguro de que quieres borrar este comentario?")) {

             $.ajax({
             url: "comment-delete.php",
             type: "POST",
             data: 'comment_id='+id,
             success: function(data){
                 if (data)
                 {
                     $("#comment-"+id).remove();
```

```
                    if($("#count-number").length > 0) {
                        var currentCount = parseInt($("#count-number").text());
                        var newCount = currentCount - 1;
                        $("#count-number").text(newCount)
                    }
                }
            }
            });
        }
    }
```

Dentro de esta función, lo primero que se hace es preguntar al usuario si está seguro de borrar el comentario.

JS

```
if(confirm("¿Estás seguro de que quieres borrar este comentario?")) {
```

Si la respuesta es afirmativa, se realiza la llamada AJAX al archivo "**comment-delete.php**".

JS

```
$.ajax({
    url: "comment-delete.php",
    type: "POST",
    data: 'comment_id='+id,
```

Además de indicar el archivo, también se indica que los parámetros enviados serán mediante "**POST**" y el "**id**" del comentario que lo asignaremos a la variable "**comment_id**". Como ya veremos, esta variable será recuperada dentro del archivo "**comment-delete.php**" para realizar el borrado del registro de la base de datos de nuestro servidor.

A continuación verificamos si la eliminación ha sido satisfactoria.

JS

```
success: function(data){
   if (data) {
```

Si todo ha ido bien, haremos dos cosas. Por un lado eliminamos el contenedor "**div**" que hace referencia al comentario, para que este no se muestre en pantalla.

JS

```
$("#comment-"+id).remove();
```

Y por otro lado, descontamos una unidad al texto que nos indica el número de comentarios que hay publicados, siempre y cuando sea mayor a 0.

JS

```
if($("#count-number").length > 0) {
   var currentCount = parseInt($("#count-number").text());
   var newCount = currentCount - 1;
   $("#count-number").text(newCount)
}
```

Insertar comentarios en la base de datos

La última de las acciones que se llevan a cabo dentro del "**index.php**", es la ejecución del código JavaScript que realiza la llamada al archivo PHP que se encarga de insertar los comentarios.

JS

```
$(document).ready(function() {

        $("#frmComment").on("submit", function(e) {
                $(".error").text("");
            $('#name-info').removeClass("error");
            $('#message-info').removeClass("error");
            e.preventDefault();
            var name = $('#name').val();
            var message = $('#message').val();

            if(name == ""){
                    $('#name-info').addClass("error");
            }
            if(message == ""){
                    $('#message-info').addClass("error");
            }
            $(".error").text("required");
            if(name && message){
                    $("#loader").show();
                    $("#submit").hide();
                    $.ajax({

                 type:'POST',
                 url: 'comment-add.php',
                 data: $(this).serialize(),
                 success: function(response)
                    {
                         $("#frmComment input").val("");
                         $("#frmComment textarea").val("");
                         $('#response').prepend(response);

                          if($("#count-number").length > 0) {
                              var currentCount = parseInt($("#count-number").text());
                              var newCount = currentCount + 1;
                              $("#count-number").text(newCount)
                          }
                          $("#loader").hide();
                         $("#submit").show();
                     }
                    });
            }
        });
    });
```

Lo primero que hacemos con este código, es detectar cuando se pulsa el botón de "**añadir comentario**" mediante la primera línea de código.

JS

```
$("#frmComment").on("submit", function(e) {
```

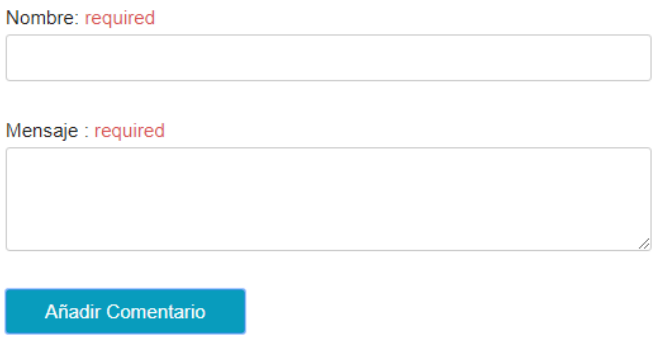
Las siguientes líneas son las encargadas de revisar que el usuario cumplimenta todos los campos antes de enviar la petición al servidor para que guarde la información. Lo primero que hacemos es borrar el texto que pueda tener el bloque que contiene la clase "**error**", además de eliminar esa clase de los campos de

recogida de información. También anulamos el envío del formulario para que la petición se haga mediante AJAX

JS

```
$(".error").text("");
$('#name-info').removeClass("error");
$('#message-info').removeClass("error");
e.preventDefault();
```

Lo siguiente es recuperar la información enviada para verificar si están vacíos o no. En el caso de que haya campos que están vacíos, se le añade la clase "**error**" y se le asigna el texto "**required**" para que informe a los usuarios de que son campos obligatorios.



JS

```
var name = $('#name').val();
var message = $('#message').val();

if(name == ""){
    $('#name-info').addClass("error");
}
if(message == ""){
    $('#message-info').addClass("error");
}
$(".error").text("required");
```

Si los campos de texto contienen información entonces ocultamos el botón de "**añadir comentario**" para que no se pueda pulsar repetidamente.

JS

```
if(name && message){
     $("#loader").show();
     $("#submit").hide();
```

Lo siguiente sería hacer la llamada AJAX al archivo PHP "**comment-add.php**" al que se le pasa los valores del formulario mediante el método POST.

JS

```
$.ajax({
    type:'POST',
    url: 'comment-add.php',
    data: $(this).serialize(),
```

Si la inserción ha sido correcta, limpiamos la información de los campos de texto del formulario y añadimos el comentario a la lista que se muestra por pantalla. Por último actualizamos el valor que nos indica el número de comentarios que hay publicados.

JS

```
success: function(response) {
      $("#frmComment input").val("");
      $("#frmComment textarea").val("");
      $('#response').prepend(response);

      if($("#count-number").length > 0) {
          var currentCount = parseInt($("#count-number").text());
          var newCount = currentCount + 1;
          $("#count-number").text(newCount)
      }
      $("#loader").hide();
      $("#submit").show();
}
```

Archivo comment-delete.php

El código incluido en este archivo, será el encargado de eliminar el comentario de la base de datos. El código es el siguiente.

PHP

```
include_once 'db.php';

$comment_id = $_POST['comment_id'];

$sql_del = "DELETE FROM usuario WHERE id = $comment_id";
$stmt = $conn->prepare($sql_del);
$stmt->execute();

if (! empty($stmt)) {
    echo true;
}
```

En este caso, lo primero de todo es realizar la llamada al archivo que se encarga de conectar con la base de datos.

PHP

```
include_once 'db.php';
```

Después se recupera el "**id**" del comentario a eliminar y que se ha pasado en la llamada AJAX que hemos visto en el archivo "**index.php**".

PHP

```
$comment_id = $_POST['comment_id'];
```

Por último, ejecutamos la instrucción SQL encargada de borrar el registro. Si todo va bien, devolvemos un valor "**true**".

PHP
```
$sql_del = "DELETE FROM usuario WHERE id = $comment_id";
$stmt = $conn->prepare($sql_del);
$stmt->execute();

if (! empty($stmt)) {
    echo true;
}
```

Archivo comment-add.php

Veamos el último de los archivos utilizados en este sistema de comentarios. Es el encargado de insertar en base de datos los nuevos comentarios enviados, además de retornar el código HTML que se encarga de mostrar el comentario en pantalla.

PHP y HTML
```
<?php
if (isset($_POST['add'])) {
    include_once 'db.php';

    $username = $_POST['user'];
    $message = $_POST['message'];

    $sql = "INSERT INTO usuario (username, message) VALUES ('$username', '$message')";
    $stmt = $conn->prepare($sql);

    $stmt->execute();
    $comment_id = $stmt->insert_id;
    ?>


<div id="comment-<?php echo $comment_id;?>" class="comment-row">
    <div class="comment-user"><?php echo $username;?></div>
    <div class="comment-msg" id="msgdiv"><?php echo $message;?></div>

    <div class="delete" name="delete" id="delete"
        onclick="deletecomment(<?php echo $comment_id;?>)">Eliminar</button>
</div>

<?php
}
?>
```

En este código, lo primero que hacemos es que la petición se ha realizado mediante POST.
PHP
```
if (isset($_POST['add'])) {
```

Lo siguiente es realizar la llamada al archivo "**db.php**" que realiza la conexión a la base de datos.

PHP
```
include_once 'db.php';
```

A continuación, recuperamos la información de los campos del formulario y se ejecuta la instrucción SQL que inserta el registro.

PHP

```php
$username = $_POST['user'];
$message = $_POST['message'];

$sql = "INSERT INTO usuario (username, message) VALUES ('$username', '$message')";
$stmt = $conn->prepare($sql);

$stmt->execute();
$comment_id = $stmt->insert_id;
```

Por último, creamos el bloque HTML que se encarga de mostrar el nuevo mensaje insertado por pantalla.

PHP y HTML

```php
<div id="comment-<?php echo $comment_id;?>" class="comment-row">
    <div class="comment-user"><?php echo $username;?></div>
    <div class="comment-msg" id="msgdiv"><?php echo $message;?></div>

    <div class="delete" name="delete" id="delete"
        onclick="deletecomment(<?php echo $comment_id;?>)">Eliminar</button>
</div>
```

Aunque pueda parecer mucho código, a la hora de la verdad son instrucciones muy sencillas. Lo único que faltaría sería adaptarlo a las necesidades de cada página web o tienda online.