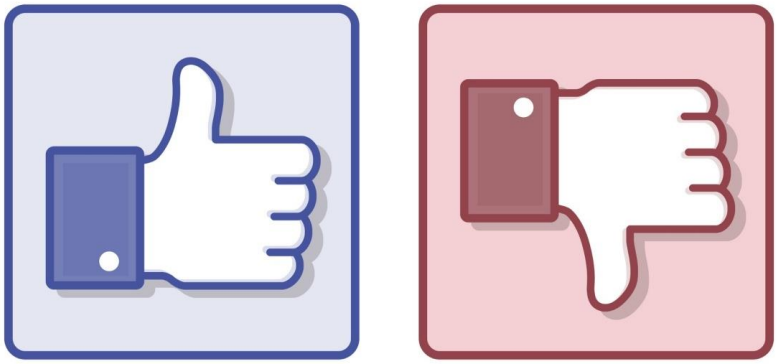
Sistema Like y Dislike con PHP y MySQL



Teléfono: 900 103 253
www.hostalia.com

Hoy en día es habitual encontrarse páginas web que ofrecen un sistema de votación donde los usuarios pueden indicar si les gusta o no esa publicación. Un claro ejemplo lo tenemos en algunas redes sociales como Facebook. Como pensamos que resulta muy útil este tipo de sistemas de votaciones, en nuestro White Paper os vamos a mostrar cómo crear esto utilizando PHP, MySQL y jQuery.

Creación de la base de datos

Lo primero de todo, será crear la estructura de base de datos. Para nuestro ejemplo, utilizaremos dos tablas: **post** y **post_votes**. En la primera de ellas se almacenará las publicaciones a las que podemos votar. En la segunda de ellas, será donde se registren los votos realizados por los usuarios.

La estructura de estas tablas serán las siguientes:

SQL

```sql
CREATE TABLE `posts` (
  `id` int(10) NOT NULL,
  `post_id` int(10) NOT NULL,
  `user_id` int(10) NOT NULL,
  `message` text NOT NULL,
  `date` date NOT NULL,
  `vote_up` int(10) NOT NULL DEFAULT '0',
  `vote_down` int(10) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

En la tabla “post” es donde se guarda la información sobre el artículo, así como el número de votos en positivo y en negativo que tiene. Será este dato el que iremos modificando.

SQL

```sql
CREATE TABLE `post_votes` (
  `id` int(10) NOT NULL,
  `post_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

En la tabla “**posts_votes**” lo que tendremos son los registros de las diferentes votaciones. Por cada votación se guardará el ID del artículo, el ID del usuario y si ha votado en positivo o en negativo.

El siguiente paso es mostrar el listado de artículos donde poder votar y que lo haremos en nuestro archivo **index.php**.

Archivo index.php

El archivo index.php será donde encontremos todo el código encargado de mostrar los diferentes posts sobre los que poder votar. La imagen final sería algo parecido a la que os dejamos a continuación.

El código encargado de generar esto es el siguiente:

HTML y PHP

```
<div class="container">
   <h2>Sistema Like y Dislike con PHP</h2>
      <div class="row" id="post-list">
         <?php
         include ('Posts.php');
         $posts = new Posts();
         $postsData = $posts->getPosts();

         foreach($postsData as $post) {
         ?>

          <div class="col-sm-4 col-lg-4 col-md-4 post-body">
             <div class="post-content"><?php echo $post['message']; ?></div>
             <div class="post-options pull-right">
                <a class="options" data-vote-type="1" id="post_vote_up_<?php echo
$post['post_id']; ?>"><i class="glyphicon glyphicon-thumbs-up" data-original-title="Like
this post"></i></a>
                 <span class="counter" id="vote_up_count_<?php echo $post['post_id'];
?>">  <?php echo $post['vote_up']; ?></span>   
                <a class="options"  data-vote-type="0" id="post_vote_down_<?php echo
$post['post_id']; ?>"><i class="glyphicon glyphicon-thumbs-down" data-original-
title="Dislike this post"></i></a>
                <span class="counter" id="vote_down_count_<?php echo $post['post_id'];
?>">  <?php echo $post['vote_down']; ?></span>
              </div>
            </div>
            <?php } ?>
         </div>
   </div>
```

Aquí lo primero que hacemos es importar el archive “**Posts.php**” que es el archivo que contiene todas las funciones para interactuar con la base de datos y que veremos más tarde.

PHP

```
include ('Posts.php');
```

Después creamos un objeto de esa clase y llamamos al método "**getPosts()**" que se encargará de devolver un array con toda la información de la base de datos.

PHP

```
$posts = new Posts();
$postsData = $posts->getPosts();
```

Este array lo recorreremos mediante un bucle "**foreach**" de PHP, ejecutándose tantas veces como artículos haya en la base de datos.

PHP

```
foreach($postsData as $post) {
```

Por cada repetición se pintará uno de los artículos junto con los iconos de votar en positivo o en negativo, además de indicar el número de votos recibidos.

HTML y PHP

```
<div class="col-sm-4 col-lg-4 col-md-4 post-body">
     <div class="post-content"><?php echo $post['message']; ?></div>
     <div class="post-options pull-right">
         <a class="options" data-vote-type="1" id="post_vote_up_<?php echo
$post['post_id']; ?>"><i class="glyphicon glyphicon-thumbs-up" data-original-title="Like
this post"></i></a>
         <span class="counter" id="vote_up_count_<?php echo $post['post_id'];
?>">  <?php echo $post['vote_up']; ?></span>   
         <a class="options"  data-vote-type="0" id="post_vote_down_<?php echo
$post['post_id']; ?>"><i class="glyphicon glyphicon-thumbs-down" data-original-
title="Dislike this post"></i></a>
          <span class="counter" id="vote_down_count_<?php echo $post['post_id'];
?>">  <?php echo $post['vote_down']; ?></span>
       </div>
</div>
```

A los iconos de las votaciones, se le asigna el "**id**" del artículo ya que será el que utilizaremos en la llamada Ajax para actualizar dicha información.

Archivo Posts.php

Este archivo, lo que contiene es el código encargado de realizar las consultas a la base de datos del servidor MySQL. En su interior, lo primero que nos encontramos es la declaración de una serie de variables de configuración.

PHP

```
private $host  = 'localhost';
private $user  = 'root';
```

```
private $password   = '';
private $database  = 'likedislike';
private $postTable = 'posts';
private $postVotesTable = 'post_votes';
private $dbConnect = false;
```

El significado de cada una de estas variables es el siguiente:

- **$host**: Para indicar el host de conexión a la base de datos. Normalmente es "localhost" aunque puede cambiar dependiendo del proveedor de alojamiento web.
- **$user**: Usuario de la base de datos.
- **$password**: Contraseña asociada al usuario de la base de datos. En nuestro ejemplo es vacía, pero es necesario utilizar una compleja para evitar problemas de seguridad.
- **$database**: Nombre de la base de datos.
- **$postTable**: Nombre de la tabla de la base de datos donde se guardarán las publicaciones.
- **$postVotesTable**: Nombre de la tabla de la base de datos donde se guardarán los registros de los usuarios que han votado.
- **$dbConnect**: Toma el valor "false" mientras que no se haya establecido una conexión con la base de datos.

Lo siguiente que nos encontraremos será el código que se encarga de realizar la conexión con la base de datos.

PHP

```
public function __construct(){
    if(!$this->dbConnect){
      $conn = new mysqli($this->host, $this->user, $this->password, $this->database);
      if($conn->connect_error){
         die("Error failed to connect to MySQL: " . $conn->connect_error);
      }else{
         $this->dbConnect = $conn;
      }
    }
}
```

Mediante la función "**mysqli**" de PHP, se realiza la conexión a la base de datos. A esta función hay que pasarle el host, el usuario, la contraseña y el nombre de la base de datos. Si todo va bien, se establece la conexión y se guarda en la variable $dbConnect.

Lo siguiente que nos encontramos son los métodos que se utilizan para recuperar, insertar o modificar la información en la base de datos. Veamos a continuación cada uno de ellos.

Método getPosts

PHP

```
public function getPosts(){
  $sqlQuery = 'SELECT post_id, user_id, message, date, vote_up, vote_down FROM '.$this-
>postTable;
```

```
  $resultado = mysqli_query($this->dbConnect, $sqlQuery);
  $row = $resultado->fetch_all(MYSQLI_ASSOC);
  return  $row;
}
```

Como se puede ver, se trata de un método muy sencillo que lo único que hace es devolver todos los artículos de la entrada "**posts**".

Método isUserAlreadyVoted

PHP

```
public function isUserAlreadyVoted($user_id, $post_id){
        $sqlQuery = 'SELECT post_id, user_id, vote FROM '.$this->postVotesTable." WHERE
user_id = '".$user_id."' AND post_id = '".$post_id."'";
        $result = mysqli_query($this->dbConnect, $sqlQuery);
        return $result->num_rows;
    }
```

En esta ocasión, el método hará una consulta a base de datos para comprobar si el usuario ha votado ya al artículo que se le pasa por parámetros. Devuelve el número de filas afectadas. Si el valor es 0, significa que no ha votado aún.

Método getPostVotes

PHP

```
public function getPostVotes($post_id){
        $sqlQuery = 'SELECT post_id, user_id, vote_up, vote_down FROM '.$this-
>postTable." WHERE post_id = '".$post_id."'";
        $result = mysqli_query($this->dbConnect, $sqlQuery);
        $row = $result->fetch_array(MYSQLI_ASSOC);
        return  $row;
    }
```

Este otro método se utiliza para obtener el número de votos positivos y negativos de un determinado artículo. Lo que devuelve es un array con toda la información.

Método updatePostVote

PHP

```
public function updatePostVote($postVoteData) {
        $sqlQuery = "UPDATE ".$this->postTable." SET vote_up =
'".$postVoteData['vote_up']."' , vote_down = '".$postVoteData['vote_down']."' WHERE
post_id = '".$postVoteData['post_id']."'";
        mysqli_query($this->dbConnect, $sqlQuery);

        $sqlVoteQuery = "INSERT INTO ".$this->postVotesTable." (id, post_id,
user_id,date) VALUES ('', '".$postVoteData['post_id']."',
'".$postVoteData['user_id']."',now())";
```

```
        if($sqlVoteQuery) {
            mysqli_query($this->dbConnect, $sqlVoteQuery);
            return true;
        }
    }
```

El último de los métodos que forma parte de este archivo se encarga de actualizar los valores de los votos del artículo, además de insertar un registro que certifique un usuario ha votado sobre ese artículo con la intención de que no pueda volver a votar más.

El código completo de esta clase es el siguiente.

PHP

```
<?php
class Posts{
    private $host  = 'localhost';
    private $user  = 'root';
    private $password   = '';
    private $database  = 'likedislike';
    private $postTable = 'posts';
    private $postVotesTable = 'post_votes';
    private $dbConnect = false;

    public function __construct(){
        if(!$this->dbConnect){
            $conn = new mysqli($this->host, $this->user, $this->password, $this-
>database);
            if($conn->connect_error){
                die("Error failed to connect to MySQL: " . $conn->connect_error);
            }else{
                $this->dbConnect = $conn;
            }
        }
    }

    public function getPosts(){
        $sqlQuery = 'SELECT post_id, user_id, message, date, vote_up, vote_down FROM
'.$this->postTable;
        $resultado = mysqli_query($this->dbConnect, $sqlQuery);
        $row = $resultado->fetch_all(MYSQLI_ASSOC);
        return  $row;
    }

    public function isUserAlreadyVoted($user_id, $post_id){
        $sqlQuery = 'SELECT post_id, user_id, vote FROM '.$this->postVotesTable." WHERE
user_id = '".$user_id."' AND post_id = '".$post_id."'";
        $result = mysqli_query($this->dbConnect, $sqlQuery);
        return $result->num_rows;
    }

    public function getPostVotes($post_id){
        $sqlQuery = 'SELECT post_id, user_id, vote_up, vote_down FROM '.$this-
>postTable." WHERE post_id = '".$post_id."'";
        $result = mysqli_query($this->dbConnect, $sqlQuery);
        $row = $result->fetch_array(MYSQLI_ASSOC);
        return  $row;
    }

    public function updatePostVote($postVoteData) {
        $sqlQuery = "UPDATE ".$this->postTable." SET vote_up =
'".$postVoteData['vote_up']."' , vote_down = '".$postVoteData['vote_down']."' WHERE
post_id = '".$postVoteData['post_id']."'";
```

```
        mysqli_query($this->dbConnect, $sqlQuery);

        $sqlVoteQuery = "INSERT INTO ".$this->postVotesTable." (id, post_id,
user_id,date) VALUES ('', '".$postVoteData['post_id']."',
'".$postVoteData['user_id']."',now())";
        if($sqlVoteQuery) {
            mysqli_query($this->dbConnect, $sqlVoteQuery);
            return true;
        }
    }
}
?>
```

Archivo options.js

En este archivo nos encontramos el código encargado de hacer la llamada Ajax para actualizar la información de la votación. El código que nos encontramos en su interior es el siguiente.

JAVASCRIPT

```
$(document).ready(function() {
    $(".options").on("click", function(){
        var post_id = $(this).attr("id");
        post_id = post_id.replace(/\D/g,'');
        var vote_type = $(this).data("vote-type");
        $.ajax({
            type : 'POST',
            url  : 'vote.php',
            dataType:'json',
            data : {post_id:post_id, vote_type:vote_type},
            success : function(response){

                    $("#vote_up_count_"+response.post_id).html("  "+response.vo
                    te_up);
                    $("#vote_down_count_"+response.post_id).html("  "+response.
                    vote_down);
            }
        });
    });
});
```

Esta función de JavaScript se ejecutará cuando el usuario pulse sobre alguno de los iconos de votaciones que aparece en el listado. Para ello se utiliza el evento "**on**" de jQuery para capturar la pulsación sobre aquellos elementos que utilicen la clase ".options".

JAVASCRIPT

```
$(".options").on("click", function(){
```

Lo siguiente es sacar el "**id**" del artículo y el tipo de voto: positivo o negativo. Para el primero de los casos se le asigna el valor 1 y para el segundo el valor 0.

JAVASCRIPT

```
var post_id = $(this).attr("id");
```

```
post_id = post_id.replace(/\D/g,'');
var vote_type = $(this).data("vote-type");
```

Una vez que tenemos toda esa información, se pasa hace la llamada Ajax utilizando jQuery.

JAVASCRIPT

```
$.ajax({
   type : 'POST',
   url  : 'vote.php',
   dataType:'json',
   data : {post_id:post_id, vote_type:vote_type},
   success : function(response){

       $("#vote_up_count_"+response.post_id).html("  "+response.vote_up);
       $("#vote_down_count_"+response.post_id).html("  "+response.vote_down);
   }
});
```

Si todo ha ido bien, se actualiza el contador que se muestra al usuario en la noticia sobre la que se ha votado. Esta parte corresponde al siguiente código.

JAVASCRIPT

```
success : function(response){

       $("#vote_up_count_"+response.post_id).html("  "+response.vote_up);
       $("#vote_down_count_"+response.post_id).html("  "+response.vote_down);
   }
```

Archivo vote.php

Por último, tendríamos el archivo "**vote.php**" que se ejecutaría mediante la llamada Ajax vista en el punto anterior y que se encarga de actualizar los datos en la base de datos y también en la parte pública. El código completo de este archivo es el siguiente.

PHP

```
<?php
include ('Posts.php');
$posts = new Posts();

if($_POST['post_id'] && $user_id) {
    $postVote = $posts->getPostVotes($_POST['post_id']);

    if($_POST['vote_type'] == 1) { //voto positivo
        if (!$posts->isUserAlreadyVoted($user_id, $_POST['post_id'])){
            $postVote['vote_up'] += 1;
        }
    } else if($_POST['vote_type'] == 0) { //voto negativo
        if(!$posts->isUserAlreadyVoted($user_id, $_POST['post_id'])) {
```

```
            $postVote['vote_down'] += 1;
        }
    }

    $postVoteData = array(
        'post_id' => $_POST['post_id'],
        'user_id' => $user_id,
        'vote_up' => $postVote['vote_up'],
        'vote_down' => $postVote['vote_down'],
    );

    $postVoted = $posts->updatePostVote($postVoteData);
    if($postVoted) {
        $response = array(
            'vote_up' => $postVote['vote_up'],
            'vote_down' => $postVote['vote_down'],
            'post_id' => $_POST['post_id']
        );
        echo json_encode($response);
    }
}
?>
```

Lo primero que se hace en este archivo es importar la clase "**Posts.php**" y crear un objeto para poder hacer uso de sus funciones para trabajar con la base de datos.

PHP

```
include ('Posts.php');
$posts = new Posts();
```

El siguiente paso es comprobar que en la llamada Ajax se ha pasado mediante "**post**" el identificador del artículo. De ser así, se recupera toda su información.

PHP

```
if($_POST['post_id'] && $user_id) {
    $postVote = $posts->getPostVotes($_POST['post_id']);
```

Una vez recuperados estos datos, se comprueba si hay que aumentar sus votos positivos o sus votos negativos. Para determinar la acción, hay que tener en cuenta el valor de la variable "**vote_type**" que se pasa en la llamada Ajax.

PHP

```
if($_POST['vote_type'] == 1) { //voto positivo
        if (!$posts->isUserAlreadyVoted($user_id, $_POST['post_id'])){
            $postVote['vote_up'] += 1;
        }
    } else if($_POST['vote_type'] == 0) { //voto negativo
        if(!$posts->isUserAlreadyVoted($user_id, $_POST['post_id'])) {
            $postVote['vote_down'] += 1;
        }
    }
```

Creamos un array con los datos de la votación del artículo actualizado.

PHP

```
$postVoteData = array(
        'post_id' => $_POST['post_id'],
        'user_id' => $user_id,
        'vote_up' => $postVote['vote_up'],
        'vote_down' => $postVote['vote_down'],
    );
```

Por último llamamos al método "**updatePostVote**" de la clase "Post" que se encargará de realizar la actualización de la información.

PHP

```
$postVoted = $posts->updatePostVote($postVoteData);
    if($postVoted) {
        $response = array(
            'vote_up' => $postVote['vote_up'],
            'vote_down' => $postVote['vote_down'],
            'post_id' => $_POST['post_id']
        );
        echo json_encode($response);
    }
```

Si todo ha ido bien, se devuelve un objeto en formato JSON a la llamada Ajax que contiene la información para actualizar los valores de la votación de la vista.