ALBERTO SILVESTRE MONTES LINARES 14275445-N

PROGRAMACIÓN WEB: CÓMO SE HIZO LA PRÁCTICA EVALUABLE 2

Para describir como he hecho esta práctica, voy a ir explicando como he hecho cada apartado de la práctica, destacando los aspectos más relevantes.

Pero antes, voy a comentar la base de datos, llamada centrodeportivoII, que he montado para esta práctica, exponiendo las tablas que he usado para ello, sus atributos y sus tipos de valor:

Usuario (username [varchar(15)], nombre [varchar(20)], apellidos [varchar(25)], correo [varchar(35)], contrasenia [varchar(32)], fechanac [date], photo [varchar(35)])

Hilo (codHilo [int(3)], titulo [varchar(50)], descripcion [varchar(100)], username [varchar(15)], dia [varchar(2)], mes [varchar(2)], anio [varchar(4)], hora [varchar(2)], minuto [varchar(2)]) con username como llave externa de Usuario(username)

Entrada (codEntrada [int(3)], codHilo [int(3)], descripcion [varchar(100)], username [varchar(15)], dia [varchar(2)], mes [varchar(2)], anio [varchar(4)], hora [varchar(2)], minuto [varchar(2)]) con username como llave externa de Usuario(username) y con codHilo como llave externa de Hilo(codHilo)

Con esta base de datos, podremos saber los datos de un usuario y cuándo y quién hizo una determinada entrada o hilo, cosas que nos harán falta para los siguientes apartados.

Nota (1): Yo he considerado como “hilo”, un tema del foro y como “entrada”, una respuesta a un tema (hilo) concreto.

Una vez contado esto, pasamos a describir los apartados:

1- A partir del formulario que se hizo en la primera práctica evaluable para que los usuarios del centro deportivo se den de alta incorporarlos como miembros del gimnasio

En este apartado, he creado 2 archivos php basados en el archivo “altausuario.html” de la práctica anterior llamados “altausuario.php” y otro llamado “procesar_formulario_alta.php”.

En “altausuario.php”, lo que he hecho es modificar el formulario poniéndole como action el archivo “procesar_formulario_alta.php” para que se procese el formulario y se de al usuario de alta en el centro deportivo y también he incorporado como aspecto innovador no visto en clase la sentencia ‘enctype=”multipart/form-data” ’ para encriptar los datos que se envíen en el formulario, sobre todo, se hace para la foto.

En el formulario, he puesto como campos obligatorios el nombre de usuario, el nombre, los apellidos, el email, la contraseña, la contraseña repetida (para certificar la contraseña introducida), la fecha de nacimiento y la foto (todos estos campos son obligatorios y van insertados a la base de datos, excepto la contraseña repetida, por lo que estos campos van a ser validados en su formato correspondiente, esto lo explicaré con más detalle en el apartado 6); el resto de campos son opcionales y no van metidos en la base de datos pero los he dejado ahí ya que los puse en la práctica anterior para ensayar los diferentes tipos de botones.

En “procesar_formulario_alta.php”, lo que hago primero es conectarme a la base de datos introduciendo lo siguiente:

```
$conexion = new PDO("mysql:host=localhost;dbname=db14275445_pw1718
" ,"x14275445","14275445");
$conexion->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
```

Luego, compruebo si el usuario está registrado en la base de datos seleccionando todos los usuarios registrados y si veo que el username introducido es igual al username de uno de los usuarios registrados, no lo registro en la base de datos mandándolo otra vez a la página “altausuarios.php” con la función ‘header("Location: altausuario.php")’, siendo esta otra innovación metida en mi página.

Despúes, cojo los campos introducidos en el formulario (todos los obligatorios) con la variable “$_POST [“nombre del campo en el formulario”]” menos el username que ya lo he cogido antes de la misma manera. Una vez cogidos los campos, encripto la contraseña usando el método de encriptación md5 con la función ‘md5($_POST["contra"])’, otra innovación más, y armo el archivo donde voy a guardar la imagen de la siguiente manera:

```
$tipo = str_replace("image/","",$_FILES["photo"]["type"]); // $_FILES["photo"]["type"]
// contiene lo siguiente image/jpeg si la foto es un jpeg, por lo que con esta función me cojo el tipo
// que en este caso sería jpeg
$nombre_foto = "imagenes/foto_".$username.".".$tipo; // aquí armo el nombre de la foto
// que va a ser de la siguiente forma: “imagenes/foto_username.jpeg”, por ejemplo
```

donde $_FILES es un array multidimensional que contiene los datos del archivo, en este caso, la foto que hemos pasado por el formulario, otra innovación más.

Por último, movemos la foto de su ubicación temporal (la sabemos mirando el valor de $_FILES[“photo”][“tmp_name”]) a la nueva ubicación que le hemos preparado en la variable $nombre_foto con la función ‘move_uploaded_file($_FILES["photo"]["tmp_name"], $nombre_foto)’, otra innovación más. Entonces, esta función devuelve un bool indicando si la foto se ha movido o no, si es así, se introduce el usuario en la base de datos y se muestra un formulario con el valor de cada dato, si no, dará un error y no se insertará en la base de datos mostrando un mensaje de error.

Finalmente, me desconecto de la base de datos introduciendo: $conexion = “”;

2- Modificar la página principal para permitir identificarse al usuario y abrir así una sesión, manteniéndola activa hasta que el usuario, de alguna forma, cierre la misma.

3- Una vez identificado, el usuario activo se quedará en el index.php pero aparecerá su nombre de usuario en la esquina superior derecha, así como un enlace o botón para cerrar sesión.

Como aclaración, tengo que decir que voy a juntar la explicación de los apartados 2 y 3 de esta práctica.

Para abrir sesión, he creado el siguiente formulario:

```
<form action="abrir_sesion.php" method="post" onsubmit="return validar_sesion()">
  <label for="username">Username</label> <br>
  <input type="text" id="username" size="13" name="username" /> <br>

  <label for="contra">Contraseña</label> <br>
  <input type="password" id="contra" size="13" name="contra" /> <br>

  <input type="submit" class="button" value="Enviar" />
</form>
```

que lo mando a la página que he creado llamada “abrir_sesion.php”, en la cual me conecto a la base de datos, cojo el username y la contraseña (que la encripto con md5). Luego, selecciono los datos de ese username y compruebo si las contraseña introducida (ya cifrada) y la contraseña introducida son iguales, si es así, abro la sesión introduciendo ‘$_SESSION["username"] = $username’, aquí lo que hago es asignarle una variable de sesión al array de sesiones ($_SESSION) a ese username, por lo que el usuario aquí ya habría iniciado sesión.

Una vez hecho todo esto, me desconecto de la base de datos y redirecciono al usuario a la página de inicio introduciendo ‘header("Location: index.php")’.

Nota (2): Este formulario lo he introducido en todas las páginas en las que se puede iniciar sesión, es decir, en todas excepto en la del perfil ya que en esta solo te puedes meter si tienes una sesión iniciada.

Finalmente, decir que para que la sesión del usuario se inicie en cada página, tengo que poner en todas las páginas donde se pueda tener la sesión iniciada la función ‘session_start()’, otra innovación más.

También, en la página “altausuario.php”, compruebo si el usuario tiene abierta una sesión, si es así, lo redirecciono a la página “index.php” ya que los usuarios no pueden a esta página con una sesión iniciada; y, en la página “perfil.php”, compruebo si el usuario no tiene abierta una sesión, si es así, lo redirecciono a la página “index.php” ya que los usuarios tienen que entrar a la página del perfil con una sesión iniciada.

Para ver que el usuario se ha identificado, pongo lo siguiente en todas las páginas en las que se pueda tener una sesión iniciada:

```
<section>
  Conectado como <strong><?php echo $_SESSION["username"]; ?></strong> <br>
  <a href="cerrar_sesion.php">Desconectar</a>
</section>
```

donde muestro el nombre de usuario que ha iniciado la sesión (el valor de $_SESSION[“username”]) y también tengo un enlace de desconectar que te manda a la página “cerrar_sesion.php”, la página donde cierro las sesiones.

En la página “cerrar_sesion.php”, primero establezco la sesión del usuario a desconectar con la función ‘session_start()’, luego, le borro todas sus variables de sesión con la función ‘session_unset()’, después, le destruyo la sesión con la función ‘session_destroy()’ y, finalmente, lo redirecciono a la página de inicio con la función header; aquí también introduzco más innovaciones.

4- Permitir que el usuario activo pueda crear una entrada nueva en el foro del centro, o que pueda responder a cualquier entrada, incluida la suya.

Para este apartado he creado 2 páginas, una llamada “procesar_nuevo_hilo.php” y “procesar_nueva_entrada.php”.

Pero primeramente, voy a explicar como he hecho la página del foro, en mi caso, la he llamado “foro.php”. Aquí lo que hago es recorrerme todos los hilos, luego, para cada hilo selecciono el usuario que realizó ese hilo cogiendo su foto, y, a su vez cogemos todos los títulos de ese usuario de cara al apartado 7, una vez hecho esto, realizamos el apartado 7 que luego lo comentaré con más detalle y, posteriormente, imprimo los datos del usuario que abrió ese hilo. Lo siguiente que hago es seleccionar todos las entradas correspondientes a ese hilo y repito el mismo proceso de imprimir sus datos que con el hilo. Finalmente, en cada hilo, compruebo si el usuario tiene una sesión iniciada, si es así, imprimo un formulario para que el usuario pueda responder a ese hilo, o lo que es lo mismo, pueda generar una entrada; por último, verifico si el usuario está conectado, en caso afirmativo, se le mostrará a este un formulario para que pueda crear un hilo.

También, decir que en esta página antes de hacer todo el proceso que he comentado, se realiza una conexión a la base de datos y al finalizar me desconecto de la base de datos; ya en el apartado aclaré como hacía eso.

Ahora, voy a explicar como he hecho la página “procesar_nuevo_hilo.php”; en esta página lo que hago primero es conectarme a la base de datos, luego cojo la fecha actual usando la función que nos proporciona PHP llamada “getdate()”, que devuelve un array con la fecha y hora del momento (una de las innovaciones no vistas en clase que meto en la web), luego cojo el titulo y la descripcion del formulario para crear un hilo, el username lo cojo de la variable $_SESSION[“username”] y aseguro que va a tener un valor porque el formulario para crear el hilo solo se lo muestro a los usuarios con la sesión iniciada y, por último cojo los datos de la hora actual y transformo los minutos de la siguiente manera:

Ya para terminar, inserto el hilo en la tabla “Hilo”, me desconecto de la base de datos y redirecciono al usuario a la página “foro.php” para mostrarle el nuevo hilo (y los demás) con sus correspondientes entradas.

Para terminar este apartado, voy a explicar como he hecho la página “procesar_nueva_entrada.php”; en esta página hago lo mismo que en la página “procesar_nuevo_hilo.php”, la única diferencia que tiene es que en vez de coger un título ya que ahora no hace falta, cojo el código del hilo asociado a esa entrada que se lo paso por el formulario de creación de entradas a través de un ‘input type=”hidden”’ para ocultar el código del hilo al usuario.

5- Que el usuario activo pueda modificar sus datos personales. Para ello se añadirá un elemento adicional en el menú que se denominará “Perfil”.

Para hacer este apartado, he añadido la página “perfil.php”, inspirada en la página “altausuarios.php”.

Lo que hago en esta página es conectarme primero a la base de datos, cojo el nombre de usuario de la variable $_SESSION[“username”], por lo que para poder entrar a esta página tienes que haber iniciado sesión (ya aclaré en la explicación conjunta de los apartados 2 y 3 como hacía la comprobación); luego, selecciono todos los datos del usuario y, después, compruebo para cada campo del usuario si el usuario ha modificado ese campo con la condición ‘isset($_POST["nombre"]) && $_POST["nombre"]!=$nombre’ en el caso del nombre pero así se haría con todos los campos excepto con la foto que lo hago de la siguiente manera:

```php
if(!empty($_FILES["photo"]["name"])) {
  $tipo = str_replace("image/","",$_FILES["photo"]["type"]);
  $nombre_foto = "imagenes/foto_".$_SESSION["username"].".".$tipo;
  unlink($photo); // Elimino del servidor la foto anterior
  if(move_uploaded_file($_FILES["photo"]["tmp_name"],$nombre_foto)) {
     if($nombre_foto!=$photo) $conexion->query("update Usuario set photo='$nombre_foto' where username='$username';");
  }
}
```

Con el campo de la foto, primero verifico si se ha enviado alguna foto, monto el nombre de la foto al igual que lo hice en la página “altausuario.php” (ya expliqué como lo hacía en el apartado 1), luego, elimino la foto que tenía antes ya que la voy a reemplazar por otra con la función unlink (otra innovación que añado a la página) y, por último, muevo la foto al igual que en el apartado 1 y comprobamos si la extensión de la imagen es la misma, si no es así la actualizamos ya que el nombre del archivo va a ser distinto, si es así no actualizamos el nombre puesto que el nombre de la foto va a ser el mismo pero el contenido va a ser distinto.

Finalmente, muestro al usuario un formulario con los valores de ese username actualizados, excepto el de la contraseña y la contraseña repetida ya que esta cifrada con el algoritmo md5, y el de la foto ya que el ‘type=”file”’ no me permite mostrar el valor de la foto.

6- Validar la corrección de todos los formularios (con JavaScript). En relación a los formularios, deben existir campos obligatorios, los comentarios a las entradas deben tener una longitud máxima, deben existir en los formularios que correspondan valores numéricos, números de teléfonos y direcciones de correo electrónico o URLs (todos validados según su formato correspondiente).

Todas las funciones para validar la correción de todos los formularios los he puesto en un archivo llamado “funciones.js” que lo exporto al archivo “.php” que las necesite introduciendo la siguiente sentencia en el “head”:

```
<script type="text/javascript" src="funciones.js"></script>
```

Nota (3): En cada formulario se introduce en el <form> la sentencia ‘onsubmit="return nombre_funcion()"’, por ejemplo, validar_sesion(), que es una de mis funciones. Esto se hace para que cuando el usuario le de al botón de tipo “submit”, el formulario pase a validarse a esa función que te devuelve “true” si el formulario es válido pasando este formulario redireccionándose al archivo introducido en action o “false” si el formulario no es válido por lo que se mostrará un mensaje de error con la función alert() y el formulario tendrá que completarse de nuevo.

Una vez dicho esto, procedemos a explicar cada una de las funciones de validación:

Primero, he creado la función validar_alta() para validar el formulario de alta de usuario. Esta función lo que hace es extraer el valor de cada campo utilizando la sentencia ‘document.getElementById(nombre_del_id).value’, una vez hecho esto, compruebo si ese valor es nulo, tiene longitud 0 o esta longitud es mayor que la correspondiente en la base de datos, si ocurren

una de estas 3 cosas en alguno de los campos se mostrará un mensaje de error y devolverá “false”, en caso contrario, devolverá “true”.

En esta función, hago comprobaciones adicionales para evitar ataques de inyección de código SQL al introducir el username, para ello chequeamos que no se introducen sentencias SQL en el username de la siguiente manera:

```
// Evito ataques de inyección de código SQL al introducir el nombre de usuario
// comprobando que no se introducen sentencias SQL
var exp1 = new RegExp(";");
var exp2 = new RegExp("=");
var exp3 = new RegExp("'");
if(exp1.test(username) || exp2.test(username) || exp3.test(username)) {
    alert("ERROR: Nombre de usuario no válido");
    return false;
}
```

Creamos 3 objetos de tipo RegExp los cuales contienen expresiones regulares usadas en sentencias SQL y comprobamos con la función test() si en el username aparece algunas de estas 3 expresiones, si esto pasa, no daremos por válido ese nombre de usuario por lo que evitaríamos un ataque por inyección de código SQL con JavaScript, esta es otra novedad no vista en clase introducida en la práctica.

También, compruebo que el email tenga un formato válido con la siguiente expresión: /\w+@+\w+\.+[a-z]/ , es decir, primero introduce caracteres, luego un @, después más caracteres, posteriormente un punto y por ultimo una secuencia de letras de la ‘a’ a la ‘z’. Una vez hecho esto, chequeamos con la función test si el email es válido, si no lo es envío un mensaje de error.

Luego, para la contaseña introducida primeramente comprueba que la contraseña no tenga un tamaño mayor que 15 caracteres (aunque en la base de datos tengo puesto que la contraseña es un varchar(32), aquí pongo este límite de 15 porque a la contraseña luego le aplico una función hash de 32 caracteres después de cogerla con el algoritmo md5) y para la contraseña repetida solo compruebo que esa igual a la contraseña introducida anteriormente. Para la fecha de nacimiento, si hemos introducido una fecha, extraemos el año, lo pasamos a entero y comprobamos que el año este entre 1900 y 2000, en caso negativo, mostramos un mensaje de error; aquí no compruebo que la longitud del campo sea mayor que el valor en la base de datos ya que no se cuál es.

Por último, para la imagen compruebo que hay foto, en caso negativo, muestro mensaje de error, en caso afirmativo, chequeo que la imagen de extensión “.gif”, ”.jpeg”, “.jpg”, “.png” o “.bmp” creando objetos RegExp(extension,”i”), en el segundo argumento del constructor del objeto indico que no sea sensible a mayúsculas ni minúsuculas; ya para terminar, compruebo con la función test() que la extensión de la imagen sea una de las mencionadas anteriormente, si no es así, muestro un mensaje de error.

En segundo lugar, he creado la función validar_perfil(), que es similar a la función detallada anteriormente en primer lugar, con las siguientes diferencias:

- Elimino las validaciones del username, ya que no se va a introducir.

- A la contraseña introducida por primera vez, solo compruebo que la nueva contraseña que se introduce no es mayor que 15 por la razón que he mencionado antes. He quitado la comprobación que no se introduce porque el usuario puede no querer actualizar su contraseña, sino otro dato.

- En la foto, solo compruebo si se introduce la imagen, en caso afirmativo compruebo la extensión de la foto al igual que he explicado antes en la función validar_alta().

En tercer lugar, he añadido la función validar_sesion(), que comprueba el username y la contraseña al igual que en la función validar_alta().

En cuatro lugar, he añadido la función validar_hilo() que valida el título de la forma habitual y valida la descripción de la forma habitual con la diferencia de que compruebo si el usuario ha introducido números o letras en mayúscula u minúscula con la función /[a-z]|[0-9]/i.test(descripcion), en caso negativo, dará un mensaje de error.

En quinto y último lugar, he creado la función validar_entrada() donde valido la descripción al igual que lo he hecho con la descripción en la función validar_hilo().

7- Al pasar el ratón por la foto de un usuario en el foro, aparecerá una ventana emergente con los títulos de las entradas que ha realizado.

Para resolver este apartado, he introducido el siguiente código en la página “foro.php”:

```
$nombre_funcion = "mostrarTitulos".$i;
echo "<script type=\"text/javascript\">";
echo "function $nombre_funcion() {";
echo "  var array = ".json_encode($a)."; ";
echo "  if(array == null || array.length == 0) {";
echo "    alert(\"Este usuario no ha realizado ningún hilo.\");";
echo "  }";
echo "  else {";
echo "    var mensaje = \"Los titulos de las entradas que ha realizado el usuario son: \";";
echo "    for(i in array) { mensaje += \" - \" + array[i] + \" \"; }";
echo "    alert(mensaje);";
echo "  }";
echo "}";
echo "</script>";

echo "<img src=\"$imagen\" alt=\"foto del usuario\" onmouseover=\"$nombre_funcion()\" />";
$i++;
```

En esta página inserto una función Javascript por cada hilo y por cada entrada, y lo que hace es pasar el array PHP con los títulos de hilo que ha creado un determinado usuario a un array Javascript (JSON) con la función json_encode(array) en PHP, esta es otra novedad no vista en clase que he introducido; luego, comprueba el tamaño del array y si es nulo o de longitud 0, muestro un mensaje indicando que no ha creado ningún hilo, si es mayor que 0, muestra los títulos del array.

Por último, decir que ejecuto esta función cuando paso el ratón sobre la foto usando el evento ‘onmouseover’ en la imagen (etiqueta img).

Nota (4): He incluido en el directorio “centrodeportivoII”, un script SQL llamado ‘crear_tablas.sql’ para crear tablas que he usado en la práctica. Luego, en el terminal de mysql, me voy a la base de datos ‘db14275445_pw1718’ y ejecuto el script con la orden ‘source crear_tablas.sql’; para ejecutar este script tengo iniciar mysql en el directorio “centrodeportivoII”.