

TEMA 6

Formularios amigables

Este tema va a ser eminentemente práctico.

Aplicando todo lo que sabemos hasta ahora, vamos a partir de un formulario de petición de datos a un usuario potencial de nuestra página.

Desarrollaremos toda la lógica básica para el funcionamiento del mismo y procesamiento de la información.

Pero como el usuario debe tener una grata experiencia al visitar nuestra página, vamos a ir modificando nuestro formulario, dando a lugar a diferentes versiones del mismo, hasta quedar satisfechos con el resultado.

Nuestra satisfacción no será total, pero con lo que tenemos hasta aquí es lo mejor que podemos hacer.

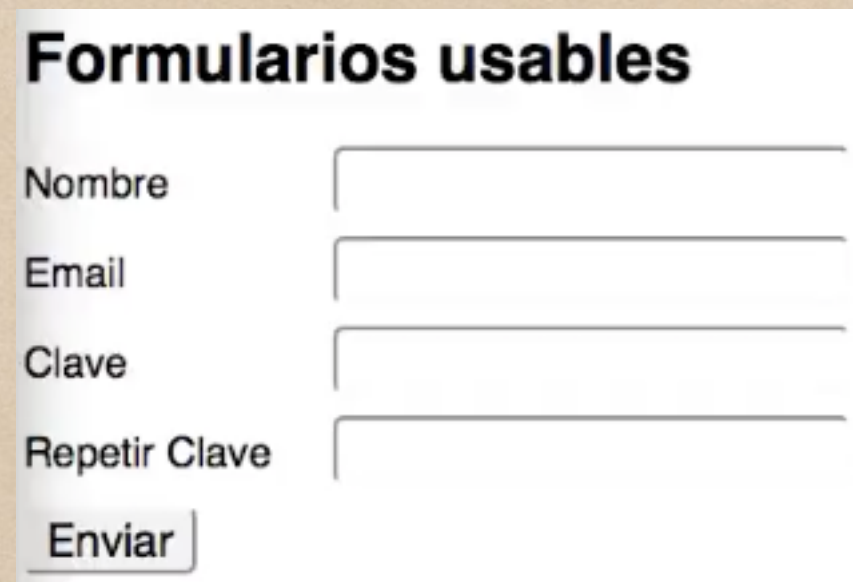
En próximos temas podremos todavía más mejorar este resultado.

Formulario inicial

```
<form action="<?=$_SERVER["PHP_SELF"]?> method="post">
  <p>
    <label for="nombre">Nombre</label>
    <input type="text" name="nombre" value="">
  </p>
  <p>
    <label for="email">Email</label>
    <input type="email" name="email">
  </p>
  <p>
    <label for="clave1">Clave</label>
    <input type="password" name="clave1">
  </p>
  <p>
    <label for="clave2">Repetir Clave</label>
    <input type="password" name="clave2">
  </p>
  <p>
    <label>
      <input type="submit" value="Enviar">
    </label>
  </p>
</form>
```


Início

Partiendo del formulario inicial anterior



Formularios usables

Nombre

Email

Clave

Repetir Clave

que debe ir entre las correspondientes etiquetas `<body>` y `</body>` del HTML, vamos a ir añadiendo poco a poco nuestra lógica de negocio. Es decir, la forma con la que debe interactuar con el usuario que está al otro lado.

Primeros pasos

A partir de ahora vamos a llamar al código del formulario anterior **formulario.html** pero sólo por resumir todas esas líneas ya que no está en un archivo independiente.

Lo primero que tenemos es un archivo index.php tal que así:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Formularios usables</title>
  <link rel="stylesheet" href="estilos.css">
</head>
<body>
  <h1>Formularios usables</h1>
  formulario.html //Aquí va nuestro formulario
</body>
</html>
```


Versión 1

En una primera aproximación, vamos a procesar las entradas del formulario con unas mínimas condiciones, que aunque no son suficientes, por algo empezamos. En el **body** escribimos:

```
<h1>Formularios usables</h1>
<?php if (!$_POST) { ?>
    formulario.html //nuestro formulario nuevamente
<?php
    } else {
        //procesamos el formulario
        if (!isset($_POST["nombre"])) {
            echo "No he recibido el nombre";
        } elseif (strlen($_POST["nombre"]) < 3) {
            echo "Campo nombre demasiado corto";
        } elseif (!isset($_POST["email"])) {
            echo "No he recibido el email";
        } elseif (strlen($_POST["email"]) < 6) {
            echo "El email no puede ser válido";
        } elseif(!isset($_POST["clave1"]) || !isset($_POST["clave2"])) {
            echo "No he recibido ambas claves";
        } elseif (strlen($_POST["clave1"]) < 5) {
            echo "La clave debe tener 5 o más caracteres";
        } elseif ($_POST["clave1"] != $_POST["clave2"]) {
            echo "Las claves tienen que ser iguales";
        } else {
            echo "Todo bien, registro ese usuario";
        }
    }
?>
```


Problemas de esta versión

Aunque lo que acabamos de hacer es perfectamente funcional, el usuario no debe estar muy agradecido al desarrollador.

Cada vez que comete una equivocación, debe volver hacia atrás para corregir el error.

¿Y si el usuario es algo novato (que aunque parezca mentira, existen) y no sabe que tiene que darle hacia atrás a la página para iniciar de nuevo el proceso?

La experiencia del usuario se resiente y podría abandonar nuestra página sin ni siquiera haber entrado.

Mejora 1

Podemos solucionar ese problema añadiendo un enlace que nos permita regresar a nuestra página justo debajo del error mostrado.

```
//procesamos el formulario
if (!isset($_POST["nombre"])) {
    echo "No he recibido el nombre";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif (strlen($_POST["nombre"]) < 3) {
    echo "Campo nombre demasiado corto";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif (!isset($_POST["email"])) {
    echo "No he recibido el email";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif (strlen($_POST["email"]) < 6) {
    echo "El email no puede ser válido";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif (!isset($_POST["clave1"]) || !isset($_POST["clave2"])) {
    echo "No he recibido ambas claves";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif (strlen($_POST["clave1"]) < 5) {
    echo "La clave debe tener 5 o más caracteres";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} elseif ($_POST["clave1"] != $_POST["clave2"]) {
    echo "Las claves tienen que ser iguales";
    echo '<br><a href="index.php">Volver al Formulario</a>';
} else {
    echo "Todo bien, registro ese usuario";
}
```


Inconvenientes

- Cada vez que haya un error y pinchemos en el enlace para regresar, los datos introducidos se pierden, teniendo que volver a escribirlos.
- Estamos repitiendo una línea de código en cada uno de los **elseif**, lo cual no es nada bueno.
- Los errores los vemos de uno en uno y no los podemos solucionar todos de una sola vez.

Una posible solución al primero de los problemas es escribir:

```
<a href="javascript:history.go(-1) ;">Volver</a>
```

Pero si el usuario no tiene habilitado JS no funcionará, o puede que esté en navegadores antiguos y no dispongan de JS. Y además, eso no nos resuelve el segundo de los problemas.

Mejora 2

- Vamos a quitar los enlaces para poner uno una única vez al final.
- Crearemos un array **errores[]** donde iremos añadiendo los diferentes errores que nos vayamos encontrando.
- Cambiaremos el condicional. Todos los **elseif** los cambiamos por **if**, para que contabilice todos los posibles errores.
- Crearemos una función que nos va a permitir mostrar los errores.

Mejora 2

```
$errores = []
//procesamos el formulario
if (!isset($_POST["nombre"])) {
    $errores[] = "No he recibido el nombre";
}
if (strlen($_POST["nombre"]) < 3) {
    $errores[] = "Campo nombre demasiado corto";
}
if (!isset($_POST["email"])) {
    $errores[] = "No he recibido el email";
}
if (strlen($_POST["email"]) < 6) {
    $errores[] = "El email no puede ser válido";
}
if(!isset($_POST["clave1"]) || !isset($_POST["clave2"])) {
    $errores[] = "No he recibido ambas claves";
}
if (strlen($_POST["clave1"]) < 5) {
    $errores[] = "La clave debe tener 5 o más caracteres";
}
if ($_POST["clave1"] != $_POST["clave2"]) {
    $errores[] = "Las claves tienen que ser iguales";
}
if ($errores) {
    mostrar_errores($errores);
    echo '<br><a href="' . $_SERVER["PHP_SELF"] . '">Volver</a>';
} else {
    echo "Todo bien, registro ese usuario";
}
```


Mejora 2

Debajo de los condicionales y antes de cerrar PHP escribimos la siguiente función:

```
function mostrar_errores($errores) {  
    echo '<ul class="listaerrores">';  
    foreach ($errores as $error) {  
        echo "<li>$error</li>";  
    }  
    echo "</ul>";  
}
```


Inconvenientes

- Cada vez que haya errores y pinchemos en el enlace para regresar, los datos introducidos se pierden, teniendo que volver a escribirlos.
- Los errores se muestran pero el formulario ya no se ve, por lo que puede que no recordemos que hemos hecho para generarlos.
- Y lo más importante: el script "nuevo" está mal porque...

Inconvenientes

- Cada vez que haya errores y pinchemos en el enlace para regresar, los datos introducidos se pierden, teniendo que volver a escribirlos.
- Los errores se muestran pero el formulario ya no se ve, por lo que puede que no recordemos que hemos hecho para generarlos.
- Y lo más importante: el script "nuevo" está mal porque si no nos envían el campo nombre, el primer condicional es cierto y entra, anotando el error, pero el segundo condicional falla, generando el error. Y lo mismo con el tercero y cuarto.

Mejora 3

Vamos lo primero a corregir el error de los condicionales.

```
//procesamos el formulario
if (!isset($_POST["nombre"])) {
    $errores[] = "No he recibido el nombre";
}elseif (strlen($_POST["nombre"]) < 3) {
    $errores[] = "Campo nombre demasiado corto";
}
if (!isset($_POST["email"])) {
    $errores[] = "No he recibido el email";
}elseif (strlen($_POST["email"]) < 6) {
    $errores[] = "El email no puede ser válido";
}
if(!isset($_POST["clave1"]) || !isset($_POST["clave2"])) {
    $errores[] = "No he recibido ambas claves";
}else{
    if (strlen($_POST["clave1"]) < 5) {
        $errores[] = "La clave debe tener 5 o más caracteres";
    }
    if ($_POST["clave1"] != $_POST["clave2"]) {
        $errores[] = "Las claves tienen que ser iguales";
    }
}
if ($errores) {
    mostrar_errores($errores);
    echo '<br><a href="' . $_SERVER["PHP_SELF"] . '">Volver</a>';
} else {
    echo "Todo bien, registro ese usuario";
}
```


Mejora 3

Vamos a seguir realizando modificaciones en el script para que los errores se muestren junto al formulario y así resultar más fácil su corrección por parte del usuario. Para ello, los datos introducidos deben permanecer en el formulario.

Vamos a modificar el condicional último, quitando el enlace de volver y sustituyéndolo por "otra vez" el formulario.

```
if($errores) {
    mostrar_errores($errores);
?>
<form action="<?=$_SERVER["PHP_SELF"]?> method="post">
    <p>
        <label for="nombre">Nombre</label>
        <input type="text" name="nombre" value="<?=$_POST["nombre"]?>">
    </p>
    <p>
        <label for="email">Email</label>
        <input type="email" name="email" value="<?=$_POST["email"]?>">
    </p>
    <p>
        <label for="clave1">Clave</label>
        <input type="password" name="clave1">
    </p>
    <p>
        <label for="clave2">Repetir Clave</label>
        <input type="password" name="clave2">
    </p>
    <p>
        <label>
            <input type="submit" value="Enviar">
        </label>
    </p>
</form>
<?php
}else{
```


Inconvenientes

Lo que acabamos de hacer funciona según nuestras especificaciones.

Pero hemos realizado dos acciones que no debemos hacerlas nunca.

Y son...

Inconvenientes

Lo que acabamos de hacer funciona según nuestras especificaciones.

Pero hemos realizado dos acciones que no debemos hacerlas nunca.

- Hemos repetido código (famoso copiar y pegar). El formulario está escrito dos veces. Si lo quisiéramos modificar, habría que duplicar el cambio.
- Estamos usando en el **value** del nombre y email la variable **\$_POST** sin comprobar si existe, lo cual puede dar lugar a generarnos un error.

Mejora 4

- En primer lugar, vamos a escribir el código del formulario en un script aparte, y lo incluiremos donde estaba con un **require** o **include**. El nuevo archivo lo llamaremos **formulario.php**. Al formulario le quitamos los **values** que nos daban el otro inconveniente, dejándolo como al principio.
- El formulario lo vamos a modificar para que "tenga memoria". Debe recordar lo que el usuario ha escrito y así poder corregir el error. Pero cerciorándonos que las variables a las que vamos a acceder existen.

Mejora 4

Ahora nuestro script queda de la siguiente forma:

```
<h1>Formularios usables</h1>
<?php
    if (!$_POST) {
        include "formulario.php";
    } else {
        $errores = [];
        //Procesamos el formulario
        ...
        if ($errores) {
            mostrar_errores($errores);
            include "formulario.php";
        } else {
            echo "Todo bien, registro ese usuario";
        }
    }
    function mostrar_errores($errores) {
        ...
    }
?>
```


Ahora vamos a cambiar el formulario para que tenga memoria.

```
<form action="<?=$_SERVER["PHP_SELF"]?> method="post">
  <p>
    <label for="nombre">Nombre</label>
    <input type="text" name="nombre"
    <?php
      if(isset($_POST["nombre"])) {
        echo ` value="` . $_POST["nombre"] . `";
      }
    ?>
  >
</p>
<p>
  <label for="email">Email</label>
  <input type="email" name="email"
  <?php
    if(isset($_POST["email"])) {
      echo ` value="` . $_POST["email"] . `";
    }
  ?>
  >
</p>
...
</form>
```


Mejora 4

Con las modificaciones realizadas, hemos solucionado los dos problemas planteados, quedando nuestra aplicación más adecuada para el manejo de los usuarios.

Mejora 4

Con las modificaciones realizadas, hemos solucionado los dos problemas planteados, quedando nuestra aplicación más adecuada para el manejo de los usuarios.

Inconvenientes

Si nos fijamos con detalle, vemos que los dos trozos de código que hemos escrito para "recordar" el nombre y el email (y que debemos repetir para clave1 y clave2) son prácticamente el mismo, y de lo que se trata es de no repetir código.

Hemos de solucionar ese problema.

Mejora 5

Vamos a cambiar el código del formulario:

```
<?php
    if(isset($_POST["nombre"])) {
        echo ` value="" . $_POST["nombre"] . `"";
    }
?>
```

Por este otro:

```
<?php
    mostrar_campo("nombre");
?>
```

Y eso lo hacemos con el resto de campos (email, clavel y clave2)

Y la función "mostrar_campo" la definimos a continuación de la otra función que tenemos, "mostrar_errores".

```
function mostrar_campo($campo) {
    if(isset($_POST[$campo])) {
        echo ` value="" . $_POST[$campo] . `"";
    }
}
```


Mejora 6

Ahora, nuestro formulario ya está bien. El usuario tendrá una experiencia agradable al acceder a nuestra página y tratar de darse de alta. Pero nos quedan cosas por hacer.

La primera, y con el objetivo de hacer nuestro código más mantenible por nosotros mismos o por otros desarrolladores, vamos a crearnos un nuevo archivo llamado `funciones.php`, el cual va a contener las dos funciones que hasta ahora tenemos creadas.

Ese archivo, y mediante un `include "funciones.php"` lo añadiremos al principio de nuestro script, justo antes de la cabecera HTML.

Mejora 6

Lo último que vamos a hacer es mejorar la forma de mostrar los errores.

En lugar de mostrarlos todos juntos, vamos a presentarlos al lado del campo del formulario al que se refieren.

Así, de un sólo vistazo, el usuario ve cual es y donde está el problema.

Para ello, el array `$errores` debe ser asociativo, donde la clave será el campo y el valor el error que estamos guardando. Y debe ser declarado antes del `if` y no en el **`else`** como hasta ahora, porque vamos a acceder a él desde el formulario, que está en los dos lados del condicional.

Para no repetir código, crearemos una nueva función en `funciones.php` a la que llamaremos **`mostrar_error_campo`**

En funciones.php añadimos:

```
function mostrar_errores_campo($campo, $errores){  
    if(isset($errores[$campo])){  
        echo '<span class="errorf">' . $errores[$campo] . '</span>';  
    }  
}
```

El script `index.php` queda casi igual, sólo cambiamos de sitio la declaración del array `$errores`, como ya habíamos dicho. Y borramos la llamada y la declaración de la función `mostrar_errores`, que ahora no vamos a usar.

Y el formulario queda de la siguiente forma:

```
<p>  
    <label for="nombre">Nombre</label>  
    <input type="text" name="nombre"  
        <?php  
            mostrar_campo("nombre");  
        ?>  
    >  
    <?php  
        mostrar_error_campo("nombre", $errores);  
    ?>  
</p>
```

E igual hacemos en `email` y `clave1` (en `clave2` no se muestran errores) pero sí se muestra el campo.

Final

Aunque con lo que tenemos ya es suficiente, nos quedaría todavía algo que es fundamental y que lo hemos pasado de puntillas.

Es la validación de los campos. Más adelante nos encontraremos en condiciones de mejorar la validación de los campos con lo nuevo que vayamos viendo a partir de aquí.

Si todo ha ido bien, registraremos nuestro nuevo usuario en la base de datos de la aplicación.