Unión Europea

Fondo Social Europeo

El FSE invierte en tu futuro

PHP

Trabajando con Formularios

Pepe

GENERALITAT VALENCIANA

Conselleria d'Educació, Investigació, Cultura i Esport

GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

Continguts

1 Trabajando con formularios

Como PHP se ejecuta dentro de HTML, sólo puede recibir datos del usuario de la aplicación a través del navegador web.

Y sólo hay una forma de introducir datos en una página web: a través de un formulario.

Veámoslo con un ejemplo. Supongamos que hemos definido en HTML este sencillo formulario:

```
<body>
    <form method="post" action="destino.php">
        Nombre<br/>
        <input type="text" name="nombre"><br/>

        Apellidos<br>
        <input type="text" name="apellido"><br/>

        <input type="submit">
    </form>
</body>
```

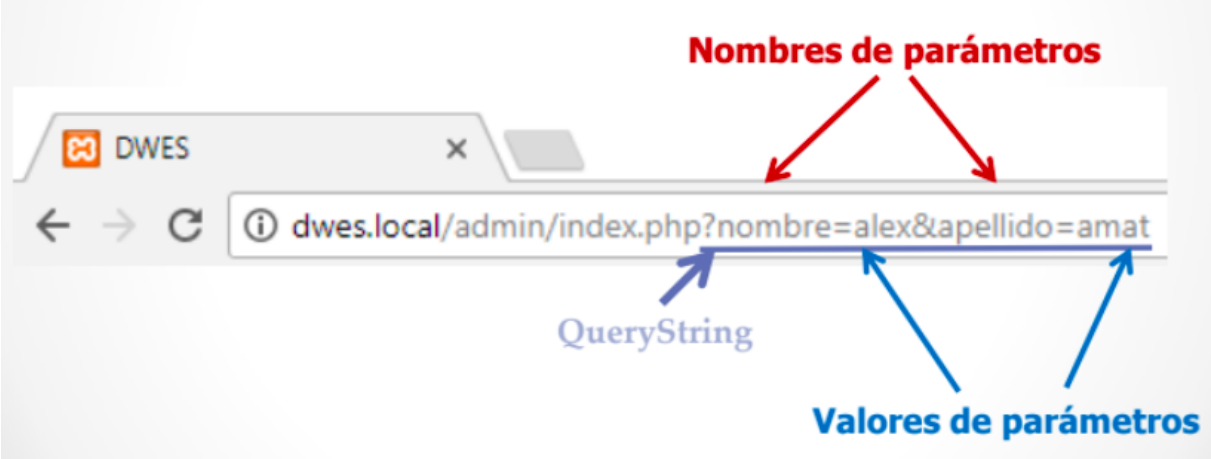
Al pulsar el botón **Enviar**, se cargará el script **destino.php** en el servidor.

Ese script recibirá dos variables HTML llamadas **nombre y apellido**, con el valor que el usuario haya introducido en el formulario.

Para acceder a las variables HTML, se usa el array del sistema `$_POST` *(En el siguiente epígrafe profundizaremos con los métodos)*, indexándolo con el nombre de la variable:

```
<?php
     echo "La variable nombre vale".$_POST['nombre']."<br>"
?>
```

1.1 Las variables $_GET, $_POST y $_REQUEST



(a) evio de datos

Para recoger los datos que envían los clientes, tenemos predefinidas diferentes variables en PHP. Por ejemplo:

- si los datos se envían por método `GET` (cuando vienen de un enlace, o de un formulario con method="get"), podemos recogerlos en una variable llamada `$_GET`. Por el contrario,
- si se envían por método `POST` (para formularios con method="post"), podremos obtenerlos con la variable `$_POST`.
- Y tenemos una tercera variable, llamada `$_REQUEST`, que nos servirá para tomar los datos de cualquiera de estos dos métodos (y de otros más que no veremos aquí). **Así, lo más habitual será utilizar esta tercera variable**, a no ser que queramos restringir la recepción de ciertos datos sólo a métodos GET o POST.
- Mostrar todos los datos recibidos:

```php
var_dump($_POST);
```

Cualquiera de estas tres variables es un array asociativo, es decir, un conjunto de datos enviados a los que se accede por el nombre de cada campo. Para acceder a un dato concreto de ese conjunto, en general, usaremos los corchetes y dentro, entre comillas, el mismo nombre que hayamos puesto en el atributo name del formulario que lo envió. Por ejemplo, si tenemos un formulario como este:

Y recogemos los datos en `destino.php`:

```php
<?php
    $nombre = $_GET["nombre"];
```

```
    $apellido1 = $_GET["apellido"];
   // Si cambiamos $_GET por $_POST o $_REQUEST, el resultado es el mismo.
       solo debéis fijaros en la URL.
    echo "Hola $nombre $apellido";
?>
```

Siempre es interesante comprobar el estado de una variable antes de procesarla, para evitar algún disgusto. Los métodos se estudiaron en el partado *2.3 Comprobar el estado de las variables* del tema anterior.

Tomando como referencia el ejemplo anterior:

```
<?php
$nombre = $_REQUEST["nombre"];
$apellido =$_REQUEST["apellido"];

    if(empty($nombre)){
      $nombre="Anónimo";
      $apellido="";
    }
   // Si cambiamos $_GET por $_POST o $_REQUEST, el resultado es el mismo.
       solo debéis fijaros en la URL.
    echo "Hola $nombre $apellido";
?>
```

1.1.1 Si lo quisiéramos realizar todo en un único archivo (lo cual no es recomendable), podemos hacerlo así:

```
<form action="" method="get">
    <p><label for="nombre">Nombre: </label>
    <input type="text" name="nombre" id="nombre"></p>
    <p><label for="apellido1">Primer apellido:</label>
    <input type="text" name="apellido1" id="apellido1"></p>
    <input type="submit" value="enviar">
</form>
<p>
    <?php
    if(isset($_GET['nombre'])) {
        $nombre = $_GET["nombre"];
        $apellido1 = $_GET["apellido1"];

        echo "Hola $nombre $apellido1";
    }
    ?>
</p>
```

2 Algunos envíos especiales

En esta sección veremos cómo recoger datos que se envían de algún modo especial, como campos múltiples, ficheros, o datos a través de enlaces.

2.1 Recogida de campos múltiples

En el caso de campos con múltiples valores enviados (por ejemplo, una lista de selección múltiple), el campo del formulario tendrá este aspecto:

```
<form...>
...
    <select multiple name="personas[]" size="5">
        <option value="Juan Rodríguez">Juan Rodríguez</option>
    ...
    </select>
...
```

Al enviarse, recogeremos los elementos enviados en una variable que podremos recorrer con un *foreach* o una estructura similar:

```
$personas = $_REQUEST['personas'];

foreach ($personas as $persona) {
    ...
    }
...
```

- Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<form action="" method="get">

<select name="lenguajes[]" multiple="true">
    <option value="c">C</option>
    <option value="java">Java</option>
    <option value="php">PHP</option>
    <option value="python">Python</option>
</select>
<br/>
```

```
<input type="checkbox" name="lenguajes[]" value="c" /> C<br />
<input type="checkbox" name="lenguajes[]" value="java" /> Java<br />
<input type="checkbox" name="lenguajes[]" value="php" /> Php<br />
<input type="checkbox" name="lenguajes[]" value="python" /> Python<br />
<input type="submit" value="enviar">
</form>
<?php

if(isset($_GET["lenguajes"])) {
    $lenguajes = $_GET["lenguajes"];

    foreach ($lenguajes as $lenguaje)
        echo "$lenguaje <br />";
}
?>
</body>
</html>
```

2.2 Envío de datos mediante enlaces

Hemos dicho que a través de los enlaces también podemos enviar datos al servidor. Normalmente, los enlaces no envían nada más que la página o recurso que queremos ver (por ejemplo, http://www.google.es). Pero también podemos añadir, al final de la URL, nombres de parámetros y sus respectivos valores, separados por el símbolo &, como si los enviáramos desde un formulario. Así, si quisiéramos **"simular"** el envío del formulario anterior para un usuario con login **"usu1"** y e-mail **"usu1@gmail.com"**, pondríamos un enlace así:

```
<a href="mipagina.php?login=usu1&email=usu1@gmail.com">
    Enviar datos
</a>
```

Estos datos podremos **recogerlos** a través de las variables `$_GET` o `$_REQUEST` **(el método POST no se emplea en los enlaces)**:

```
    $mailUsuario = $_GET["email"];
```

2.3 Envíos a la propia página

Es habitual también encontrarnos con formularios cuyo action apunta a la misma página del formulario.

Después, con código PHP (con instrucciones if..else), podemos distinguir si queremos cargar una página normal, o si hemos recibido datos del formulario y hay que procesarlos.

En cualquier caso, para poder hacer que un formulario se envíe a su propia página, podemos directamente poner el nombre de la misma página en el action (por ejemplo, `action="mipagina.php"`), pero si más adelante decidimos cambiar el nombre del archivo, corremos el riesgo de olvidarnos cambiar el formulario también. Para evitar este problema, contamos en PHP con una variable llamada `$_SERVER['PHP_SELF']` con lo que bastaría con poner esa variable en el action del formulario :

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    ...
</form>
```

Si nos centramos en el array `$_SERVER` podemos consultar las siguientes propiedades:

- PHP_SELF: nombre del script ejecutado, relativo al document root (p.ej: /tienda/carrito.php)
- SERVER_SOFTWARE: (p.ej: Apache)
- SERVER_NAME: dominio, alias DNS (p.ej: www.elche.es)
- REQUEST_METHOD: GET
- REQUEST_URI: URI, sin el dominio
- QUERY_STRING: todo lo que va después de ? en la URL (p.ej: heroe=Batman&nombre=Bruce)
- REMOTE_ADDR: Determina y muestra que ordenaor hace la petición.

Más información en https://www.php.net/manual/es/reserved.variables.server.php

3 Validación

Respecto a la validación, es conveniente siempre hacer validación doble:

- En el cliente mediante JS
- En servidor, antes de llamar a negocio, es conveniente volver a validar los datos.

```
<?php

    if(isset($_REQUEST["nombre"])){
    if(empty($_REQUEST["nombre"])){
     ...

?>
```

- En `action` se debe escribir `<?php echo htmlespecialchars($_SERVER["PHPSELF"]); ?>`.

- $_SERVER[“PHP_SELF”] es una superglobal que devuelve el nombre del archivo en el que se encuentra el formulario, lo que hace que los datos se envíen al mismo archivo, en lugar de llevarlos a otro archivo para tratarlos.
- Si se emplea esta forma de indicar el archivo para action, es necesario usar la función htmlespecialchars(), que convierte caracteres especiales en entidades HTML previniendo posibles ataques `Cross-site Scripting`.

4 Redirecciones e inclusiones

PHP también dispone de instrucciones para permitir incluir el contenido de un documento en otro como parte de la respuesta a un cliente, o redirigir a otra página si es necesario.

4.1 Redirigir a otra página

La redirección de una página a otra se hace aprovechando el protocolo HTTP, mediante una de sus cabeceras, llamada `Location`. Usaremos el comando `header` para acceder a las cabeceras HTTP, y dentro pondremos la cabecera Location, junto con la página a la que queremos redirigir. Por ejemplo, si queremos redirigir a la página login.php de nuestra web, pondríamos algo como:

```
    header("Location:login.php");
```

El siguiente código redirige a index.php pasados 5 segundos, mostrando un mensaje de redirección.

```
header("Refresh:5; url=index.php");
echo '<p>En breve le redirigiremos a la página principal.</p>';
```

Una de las utilidades prácticas que tiene esta característica es la de poder enviar al usuario a otro recurso si no se cumplen ciertas condiciones. Por ejemplo, si los datos de un formulario no son correctos. Así, podríamos utilizar funciones como `isset` o `empty` (entre otras, como la comprobación de expresiones regulares) para determinar si los datos recibidos son correctos y, en caso contrario, redirigir a una página de error o al propio formulario de nuevo:

```
if (!isset($_REQUEST['login']) || empty($_REQUEST['login']))
{
    header("Location:login.php");
    exit();
}
...
```

Además, es recomendable también hacer una llamada a `die()` o `exit()` tras una redirección, para evitar que se siga ejecutando el resto de la página.

4.2 Subir ficheros.

Se almacenan en el servidor en el array `$_FILES` con el nombre del campo del tipo *file* del formulario.

```
<form enctype="multipart/form-data" action="pagina.php"  method="POST">
    Archivo: <input name="archivoEnviado" type="file" />
    <br />
    <input type="submit" name="btnSubir" value="Subir" />
</form>
```

Configuración en `php.ini`

- `file_uploads`: on / off
- `upload_max_filesize`: 2M
- `upload_tmp_dir`: directorio temporal. No es necesario configurarlo, cogerá el predeterminado del sistema
- `post_max_size`: tamaño máximo de los datos POST. Debe ser mayor a upload_max_filesize.
- `max_file_uploads`: número máximo de archivos que se pueden cargar a la vez.
- `max_input_time`: tiempo máximo empleado en la carga (GET/POST y upload → normalmente se configura en 60)
- `memory_limit`: 128M
- `max_execution_time`: tiempo de ejecución de un script (no tiene en cuenta el upload)

El tamaño máximo permitido del fichero a subir se puede establecer en el archivo `php.ini` (propiedad upload_max_filesize), o añadiendo un campo oculto **(type="hidden")** en el formulario, con nombre *MAX_FILE_SIZE* y el máximo tamaño permitido, en *bytes*.

```
<form action="pagina.php" method="post" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="20000" />
    <input type="file" name="archivoEnviado" />
```

La información sobre un archivo subido la proporciona el array multidimensional $_FILES. Este array se crea con el key que se indique en el input del formulario, en este caso `archivoEnviado`:

- `$_FILES["archivoEnviado"]["name"]`. Guarda el nombre original del archivo del cliente.
- `$_FILES["archivoEnviado"]["type"]`. Guarda el MIME type del archivo.
- `$_FILES["archivoEnviado"]["size"]`. Guarda el tamaño del archivo en bytes.
- `$_FILES["archivoEnviado"]["tmp_name"]`. Guarda el nombre del archivo temporal.

- `$_FILES["archivoEnviado"]["error"]`. Guarda cualquier código de error que pueda provocar la subida del archivo.

El array $_FILES['imagen']['error'] especifica por qué no se ha podido subir el archivo, lo que permite especificar un mensaje de vuelta para cada tipo de error. Devuelve un integer con el número de error:

```php
<?php
if (isset($_POST['btnSubir']) && $_POST['btnSubir'] == 'Subir') {
    if (is_uploaded_file($_FILES['archivoEnviado']['tmp_name'])) {
        // subido con éxito
        $nombre = $_FILES['archivoEnviado']['name'];
        move_uploaded_file($_FILES['archivoEnviado']['tmp_name'], "uploads
            /{$nombre}");

        echo "<p>Archivo $nombre subido con éxito</p>";
    }
}
?>
```

La función `move_uploaded_file()` mueve un archivo subido del directorio temporal al directorio que se indique.

5 Ejercicios

5.0.1 calculadora.php

Escribe un programa `calculadora.php` que acepte por la dirección las variables `$x` y `$y` y que:

Muestra por pantalla:

- El valor del array `$_GET` (utiliza la función print_r())
- La suma, resto, multiplicación y división de x e y.
- El valores de la variable `$_SERVER`.
- ¿Cual es el ordenador que hace la petición?
- En qué variable están los parámetros de la petición.
- ¿Qué es la ruta del sitio web en el ordenador local ?
- Utilitza una vista para mostrar el resultado. `calculadora.view.php`

5.0.2 formulario.html y formulario.php

Crea un formulario(`utiliza bootstrap`) que solicite:

- Nombre y apellidos.
- Email.
- URL página personal.
- Sexo (radio).
- Número de convivientes en el domicilio.
- Aficiones (checkboxes) => poner mínimo 4 valores.
- Menú favorito (lista selección múltiple) => poner mínimo 4 valores.
- **Muestra los valores cargados en una tabla-resumen.**

5.0.3 login.php

Vamos a simular un formulario de acceso:

`login.php`: el formulario de entrada, que solicita el usuario y contraseña. `compruebaLogin.php`: recibe los datos y comprueba si son correctos (los usuarios se guardan en un array asociativo) pasando el control mediante el uso de include a:

ok.php: El usuario introducido es correcto

ko.php: El usuario es incorrecto. Informar si ambos están mal o solo la contraseña. Volver a mostrar el formulario de acceso.

5.0.4 subidaImagen.php

(`utiliza bootstrap`)

Crea un formulario que permita subir unicamente imágenes (comprueba la propiedad type del archivo subido). Si el usuario selecciona otro tipo de archivos, se le debe informar del error y permitir que suba un nuevo archivo. En el caso de subir el tipo correcto, visualizar la imagen durante 5 segundos,con la ruta y nombre, tamaño de anchura y altura y redirecciona al formulario. También hay que crear un enlace para mostrar el listado de todas las imagenes subidas.(analiza/estudia el método `scandir()`).