Práctica 4

Sesiones 5, 6, 7 y 8

(Continuación)

Programación web dinámica con PHP

Captura de datos de un formulario

Cuando se escribe un formulario, el elemento *form* debe indicar qué guión se encargará de procesar los datos introducidos en él, así como el método que se emplea para enviarle dichos datos de entrada.

```
<form action="gestor_del_formulario.php" method="post">
```

Recordar en este punto que existen dos métodos para tal fin: *get* (todos los datos se envían a través del URL y es utilizado para enviar pequeñas cantidades de datos, haciendo más fácil al usuario reenviar el formulario) y *post* (envía los datos a través de las cabeceras de HTTP, empleado cuando hay una cantidad de datos más grande).

PHP toma los datos de los formularios de tres arrays globales (superglobal variables):

- *$_GET*: que contiene una lista de nombres de campos junto con sus valores correspondientes enviados por el formulario por medio del método get.
- *$_POST*: que contiene una lista de nombres de campos junto con sus valores correspondientes enviados por el formulario por medio del método post.
- $_REQUEST: que contiene los valores de *$_GET* y *$_POST*, combinados, junto con los valores de otro array global denominado *$_COOKIE*.

Así, a partir de un campo de texto como *<input type="text" name="correo" value="" />* , un guión PHP podría acceder al valor introducido en el formulario utilizando *$_GET* ó *$_REQUEST* de la siguiente forma:

```
$email = $_GET["correo"];
$email = $_REQUEST["correo"];
```

Ejercicio 1:

Dado el siguiente documento HTML que implementa un formulario:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Formulario</title>
	</head>
<body>
	<h1>Ejemplo de formulario</h1>
	<p>Por favor, rellene los siguientes datos y haga click en el botón Enviar.</p>
	<form action="procesar_formulario.php" method="post">

	<label for="nombre">Nombre</label>
	<input type="text" name="nombre" id="nombre" value="" />
	<label for="apellido">Apellido</label>
	<input type="text" name="apellido" id="apellido" value="" />
	<label for="password1">Contrase&ntilde;a</label>
	<input type="password" name="password1" id="password1" value="" />
	<label for="password2">Repita la contrase&ntilde;a</label>
	<input type="password" name="password2" id="password2" value="" />
	<label for="generoHombre">&iquest;Eres hombre</label>
	<input type="radio" name="genero" id="generoHombre" value="H" />
	<label for="genero">... o mujer&#63;</label>
	<input type="radio" name="genero" id="generoMujer" value="M" />
	<label for="rangoEdad">Rango de edad</label>
	<select name="rangoEdad" id="rangoEdad" size="1">
		<option value="infante">Infante</option>
		<option value="adolescente">Adolescente</option>
		<option value="adulto">Adulto</option>
		<option value="mayor">Mayor</option>
	</select>
	<label for="deporte">&iquest;Practicas deporte&#63</label>
	<input type="checkbox" name="deporte" id="deporte" value="S&iacute;" />
	<label for="comentarios">&iquest;Alg&uacute;n comentario?</label>
```

```
            <textarea name="comentarios" id="comentarios" rows="4" cols="50"> </textarea>
            <input type="submit" name="botonDeEnvio" id="botonDeEnvio"
                value="Enviar datos" />
            <input type="reset" name="bontonDeReset" id="botonDeReset"
                value="Vaciar formulario" />
        </form>
</body>
</html>
```

Escribe el guión procesar_formulario.php, basado en POO, tal que, en un documento HTML, muestre los valores introducidos por el usuario (menos la contraseña) en el formulario anterior como una lista de definiciones. Antes de mostrar el contenido, nos deberemos asegurar de que hay un dato asociado a cada componente del formulario, en cuyo caso, podemos emplear las funciones isset o array_key_exit. En caso de no haberlo, entonces pondremos el mensaje "Campo no asignado".

En el caso de tener componentes de formulario que puedan enviar más de un valor (elecciones múltiples), como los siguientes elementos:

```
        <label for="consolasFavoritas">&iquest;Cu&aacute;les son tus consolas favoritas?</label>
        <select name="consolasFavoritas" id="consolasFavoritas" size="3" multiple="multiple">
            <option value="xbox">XBox</option>
            <option value="psp">PSP</option>
            <option value="wii">Wii</option>
        </select>
        <label for="comunicacionesTelefonicas">
                &iquest;Quieres recibir comunicaciones telef&oacute;nicas? </label>
        <input type="checkbox" name="comunicaciones" id="comunicacionesTelefonicas"
                value="comunicacionesTelefonicas" />
        <label for="comunicacionesCorreo">&iquest;Quieres recibir comunicaciones por correo?</label>
        <input type="checkbox" name="comunicaciones" id="comunicacionesCorreo"
                value="comunicacionesCorreo" />
```

El truco para manejar campos multivaluados en los guiones PHP es añadir corchetes justo después del nombre del campo correspondiente:

```
        <select name="consolasFavoritas[]" id="consolasFavoritas" size="3" multiple="multiple">
            ...
        </select>
        <input type="checkbox" name="comunicaciones[]" id="comunicacionesTelefonicas"
                value="comunicacionesTelefonicas" />
```

```
<input type="checkbox" name="comunicaciones[]" id="comunicacionesCorreo"
                value="comunicacionesCorreo" />
```

Y para procesarlo, el código sería algo así:

```
<?php
        $consolasFavoritas = "";
        $comunicaciones = "";
        if ( isset( $_POST["consolasFavoritas"] ) ) {
                foreach ( $_POST["consolasFavoritas"] as $consola ) {
                        $consolasFavoritas .= $consola . ", ";
                }
        // También se podría acceder mediante $_POST["consolasFavoritas"][0],
        // $_POST["consolasFavoritas"][1] y $_POST["consolasFavoritas"][2].
        }
        if ( isset( $_POST["comunicaciones"] ) ) {
                foreach ( $_POST["comunicaciones"] as $comunicacion ) {
                        $comunicaciones .= $comunicacion . ", ";
                }
        }
        echo "$consolasFavoritas <br/> $comunicaciones <br/>";
?>
```

Para comprobar si un campo está vacío se puede emplear la función *empty*:

```
if(empty($_POST['nombreDeCampo']))
{
        echo "El campo está vacío";
        /* Sentencias para mostrar de nuevo el formulario. */
}
```

También se puede emplear la función *isset*.

Se pueden emplear expresiones regulares para comprobar la entrada de un usuario. Echa un vistazo a http://www.php.net/manual/es/refs.basic.text.php.

Ejercicio 2:

Escribe un guión PHP basado en POO, procesar_formulario2.php, que:

1) Muestre el formulario del Ejercicio 1. Algunos campos son requeridos (nombre, apellidos, password1, password2 y genero) y deben de estar marcados con un asterisco en el formulario.

2) Cuando se envíe el formulario, el guión debe comprobar que los campos requeridos han sido

rellenados.

3) Si todos lo han sido correctamente, entonces el guión mostrará un mensaje de agradecimiento.

4) Si alguno de los campos requeridos no ha sido introducido, el guión mostrará de nuevo un mensaje de error y señalará los campos que necesitan ser rellenados, mostrando el formulario completo. El guión recordará aquellos otros que sí se rellenaron y sus correspondientes valores.

Paso de datos entre guiones PHP

En algunas ocasiones puede ser interesante almacenar el valor de una variable entre ejecuciones de un guión de PHP. Una de las técnicas que se emplean es guardarlas en formularios, concretamente en componentes que está ocultos al usuario (propiedad *hidden*):

```
<input type="hidden" name="consulaSeleccionada" value="<?php echo $consolaSeleccionada?>" />
```

No es buena idea utilizar esta técnica para almacenar y transmitir datos sensibles (como contraseñas, identificadores de usuarios, etc).

Una de las utilidades de este método es la creación de series de formularios que vayan guiando al usuario a través de un proceso de entrada de datos.

Ejercicio 3:

Escribe un guión en PHP que realice la entrada de un formulario en cinco pasos. En cada paso se mostrará un componente del formulario y al hacer clic en el botón siguiente, se pasará al siguiente campo del formulario. Recuerda que hay que almacenar los valores que se van guardando en cada etapa. Finalmente, se mostrarán todos los valores introducidos.

Otra alternativa para enviar datos de una página a otra es mediante el enlazado a un URL, en el que se añaden las variables y valores siguiendo el formato: formulario.php?var1=10&var2=30. Así, se podría hacer de estas tres formas:

```
<a href="paginaSiguiente.php?edad=14">Ir a la siguiente pagina</a>

header("Location: paginaSiguiente.php?edad=14");

<form action="paginaSiguiente.php?edad=14" method="POST">
```

De esta forma, cualquier dato pasado estará disponible en el array *$_GET*.

Ojo, que todos los valores de las variables que se pasan por medio de los URLs se pueden ver. Otro problema es que los usuarios pueden teclear el URL completo, junto con las variables pasadas y sus valores.

Otra forma de pasar datos es mediante el envío de cookies, que son pares variable=valor que el navegador almacena en el ordenador. Antes de emplear esta alternativa, tenemos que ser conscientes que los usuarios tienen el control de rechazar o borrar las cookies.

Para almacenar cookies se emplea la función *setcookie("variable", "valor",tiempoDeExpiracion)*. Las cookies quedarán accesibles en el array $_COOKIE, pero no en el guión donde se han asignado, sino otra página, salvo que recargue la que se encuentra.

Veamos algún ejemplo de su uso:

```
setcookie("comunidad","andalucia");

// En otra página se podría consultar su valor:
```

```
echo "Tu comunidad es ".$_COOKIE["comunidad"];
setcookie("Nombre",$nombre,time()+(3*86400)); #expira en 3 días.
setcookie("comunidad","andalucia",mktime(3,0,0,4,1,2013));
#expira a las 3:00 AM del 1 de abril de 2013 .
setcookie("Nombre"); / / Borra la cookie.
setcookie("comunidad", ""); Borra la cookie.
```

Hay que tener en cuenta que las cookies sólo se pueden emplear ANTES de que se haya enviado algún valor a la salida. No se puede crear una cookie en mitad del guión una vez que se haya realizado alguna salida a una página web.

Como alternativa a las cookies tenemos las sesiones de PHP y almacenar valores en una base de datos.

Redirección

Para redirigir el navegador a una nueva página como consecuencia de una introducción de datos, por ejemplo, se emplea la función header:

```
if ( isset( $_POST["botonDeEnvio"] ) ) {
/ / (Gestión de los datos del formulario aquí)
header( "Location: gracias.html" );
exit;
} else displayForm();
}
```

Otras cuestiones interesantes

- El proceso para subir archivos en el servidor y gestionarlos se puede encontrar en http://www.php.net/manual/es/features.file-upload.post-method.php.
- Sesiones: http://www.php.net/manual/es/book.session.php.

Bibliografía

– http://phpexercises.com/