Desarrollo web en entorno servidor - DWES

DWES03 - Formularios PHP.

Profesor: Pepe Lluyot
pepe.lluyot@iescristobaldemonroy.es

Formularios PHP - Relación de ejercicios 3

Para poder realizar los siguientes ejercicios, vamos a elaborar una web con html5 con una estructura básica que nos servirá de base para incluir el código PHP.

Contenido:

Ejercicios (bloque III)

Formularios

1. Crea un **formulario** que permita al usuario ingresar una temperatura en grados Celsius. Al enviar el formulario mediante el método **GET**, el sistema debe convertir la temperatura a Fahrenheit y mostrar el resultado en una nueva página. Incluye un botón de "Limpiar" para restablecer el formulario.
   Fórmula de conversión: °F = (°C × 9/5) + 32
2. Crea un formulario que simule una calculadora en la que se solicite dos números y una operación matemática (+, -, *, /). Al enviar el formulario mediante **POST**, el sistema debe realizar la operación y mostrar el resultado. Asegúrate de validar los datos: no se debe permitir la división por cero y los campos no pueden estar vacíos.
3. Crea un formulario sencillo donde se envíen los datos de un empleado (nombre y apellidos). La misma página deberá recoger la información almacenada mediante el método GET.
   - Mediante php, validar que los campos son obligatorios y que tienen una longitud menor a 25 y 35 respectivamente cada uno. Mostrar un mensaje de validación.
   - Validar que ambos campos son requeridos mediante la propiedad html required.
   - Recuperar el nombre y los apellidos enviados y mostrarlos en los controles de tipo texto correspondientes.
4. En el ejercicio anterior hacer uso de:
   - **`$_SERVER['REQUEST_METHOD']`** para comprobar que los datos se han recibido por el método GET.
   - Realiza una inyección en cualquiera de los campos nombre o apellidos y posteriormente haz uso de la función **htmlspecialchars** para solventar el problema
   - Mirar el uso del operador **??** y aplicarlo en la solución
5. Crea un formulario en el que el usuario pueda seleccionar las siguientes opciones:
   - Color de fondo: Una lista de colores predefinidos (por ejemplo: rojo, azul, verde).
   - Tipo de fuente: Una lista de fuentes (por ejemplo: Arial, Verdana, Courier).

- Estilo literario: Una lista de estilos literarios (por ejemplo: Narrativo, Poético, Ensayo).

Al hacer clic en el botón "Enviar", justo debajo del formulario, se debe mostrar un mensaje con las siguientes características:

- El mensaje tendrá el color de fondo seleccionado.
- El tipo de fuente será el que se haya elegido.
- El contenido del mensaje será un texto seleccionado al azar de un array, de acuerdo con el estilo literario elegido.

Utiliza el método POST para recuperar los datos y para que el formulario se reenvíe a la misma página sin perder los valores.

Usa la función **array_rand** para tomar una clave aleatoria de un array

6. Tabla de multiplicar: Crea un formulario que solicite al usuario un número del 1 al 10. Al enviar el formulario, muestra la tabla de multiplicar de ese número.
7. Crea un formulario que permita al usuario *registrar tareas*. El formulario debe incluir un campo de texto para ingresar la tarea y dos botones: uno para agregar la tarea y otro para borrar todas las tareas registradas. Al enviar el formulario mediante el método POST, el sistema debe manejar las siguientes acciones:
   - Agregar Tarea: Si el usuario ingresa una tarea y hace clic en "Agregar Tarea", la tarea se debe almacenar en un array. Utiliza un campo oculto para mantener el estado de las tareas serializadas y asegurarte de que se mantengan después de enviar el formulario. Las tareas deben ser mostradas en la misma página después de agregar una nueva.
   - Borrar Tareas: Si el usuario hace clic en "Borrar Tareas", el array debe vaciarse y se debe mostrar un mensaje indicando que no hay tareas registradas.

   Asegúrate de validar que el campo de tarea no esté vacío antes de agregarla al array. Las tareas registradas deben mostrarse en un listado debajo del formulario.

   Usa **htmlspecialchars()** para evitar problemas de inyección de HTML al mostrar las tareas.

   Haz uso de las funciones **serialize** y **unserialize** para serializar los arrays y enviarlos a través de un campo del formulario
8. Crea un formulario que permita al usuario ingresar un texto en un campo textarea. Al enviar el formulario mediante el método POST, el sistema debe validar el

contenido del textarea asegurándose de que no esté vacío y que no exceda los 500 caracteres. Si el texto es válido, se deben mostrar las siguientes estadísticas:

- Longitud del texto (número total de caracteres).
- Número de palabras.
- Número de líneas.
- Frecuencia de las vocales (cantidad de veces que aparecen las letras a, e, i, o, u).
- Porcentaje de espacios en blanco respecto al total de caracteres.

Si el formulario contiene errores, muestra los mensajes de error correspondientes debajo del textarea. Asegúrate de mantener el texto introducido por el usuario si hay errores en la validación.

Recuerda:

- implementar la sanitización del texto antes de procesarlo.
- Mostrar un mensaje de error si el campo textarea está vacío o si el texto excede los 500 caracteres.

9. Crea un formulario que permita al usuario generar una contraseña aleatoria. El formulario debe tener las siguientes opciones:
   - Longitud de la contraseña (8 a 16 caracteres).
   - Incluir letras minúsculas.
   - Incluir letras mayúsculas.
   - Incluir números.
   - Incluir caracteres especiales.

   Se debe validar que al menos incluya alguno de los tipos de caracteres y que la longitud sea la correcta. Los caracteres seleccionados deben aparecer al menos una vez en la contraseña. Por ejemplo si marcamos una longitud de 8, minúsculas y número, en la contraseña generada debe haber al menos un número y una letra minúscula.

   Al enviar, muestra la contraseña generada.

   funciones nuevas que se pueden usar para la resolución **str_split**

Archivos

Para poder trabajar con ficheros en un formulario debes tener en cuenta los siguientes puntos:

- configurar el archivo ***php.ini*** para permitir la subida de archivos:

i. Establecer ***file_uploads = On***.
ii. Ajustar ***upload_max_filesize*** y ***post_max_size*** según sea necesario.

- Funciones que vamos a usar: **fopen**, **fclose**, **feof**, **fread**, **fgets**, **fgetc**, **basename**, **move_uploaded_file**, **pathinfo**, **file_get_contents**, **file_put_contents**, **file_exists**, **uniqid**, **unlink**
- Array **$_FILES**

10. **Archivos**: Crea un formulario que permita al usuario subir un archivo al servidor, en un directorio específico. Valida el tipo de archivo (sólo imágenes: jpeg, jpg o png) y el tamaño (máximo 2MB). Si todo es correcto, muestra la imagen subida y un mensaje de éxito.
11. **Archivos**. Crea un formulario con los campos nombre y fichero, en nombre se deberá introducir el nombre que queremos que tenga el fichero y en fichero se deberá subir un archivo que contenga 5 palabras separadas por una coma. Hay que verificar las siguientes condiciones:
    - El nombre debe tener entre 3 y 30 caracteres.
    - El archivo debe tener la extensión txt
    - No superar 100bytes
    - Debe contener 5 palabras separadas por una coma

    Si se verifican estas condiciones, se debe almacenar en un directorio llamado "*palabras*" y con el nombre recogido en el campo nombre del formulario. Se debe verificar que ese fichero no exista previamente.
    Mostrar mensajes de validación y de errores, y de éxito en el caso de que todo se realice correctamente.
    Si todo es correcto, se deberá mostrar en forma de lista las palabras incluidas en el fichero ordenadas alfabéticamente.
12. Crea un formulario que muestre 3 preguntas seleccionadas aleatoriamente de un archivo que contiene un conjunto mayor de preguntas y opciones de respuesta. El archivo tendrá la siguiente estructura:
    - Pregunta
    - Opciones de respuesta (múltiple opción)
    - Respuesta correcta

Al cargar la página:

- Se deben seleccionar 3 preguntas de manera aleatoria y presentarse en el formulario.
- Al enviar el formulario, se debe mostrar la puntuación obtenida (número de respuestas correctas).

Hay que tener en cuenta:

- Las respuestas correctas deben almacenarse de manera **temporal** en el servidor (usa archivos) y no deben incluirse en el código HTML del formulario.
- Al procesar el formulario, el sistema debe comparar las respuestas del usuario con las respuestas correctas y mostrar la puntuación.
- Después de procesar el formulario, elimina el archivo temporal con las respuestas correctas.
- Asegúrate de que los campos del formulario sean obligatorios.

Formato del Archivo de Preguntas (**JSON**):

- Utiliza un archivo de texto (preguntas.json) para almacenar las preguntas con sus opciones y la respuesta correcta.
- El formato del archivo será similar a este ejemplo:

```
[
 {
  "pregunta": "¿Cuál es la capital de Francia?",
  "opciones": ["París", "Roma", "Londres", "Madrid"],
  "respuesta_correcta": "París"
 },
 {
  "pregunta": "¿Cuál es el resultado de 5 + 5?",
  "opciones": ["8", "9", "10", "11"],
  "respuesta_correcta": "10"
 },
 {
  "pregunta": "¿Cuál es el idioma oficial de Brasil?",
  "opciones": ["Español", "Portugués", "Inglés", "Francés"],
  "respuesta_correcta": "Portugués"
 }
```

]

Pistas:

- Usa las funciones **file_get_contents**(), **array_rand**() y **file_put_contents**() para gestionar el archivo de preguntas y las respuestas correctas.
- Puedes usar serialize() y unserialize() para guardar y recuperar las respuestas correctas en el archivo temporal.
- Usa unlink() para eliminar el archivo temporal después de procesar el formulario.

Pasos:

```
   //abrimos el fichero e12_preguntas.json y cargamos su contenido en una
variable
   //transformamos la variable en un array asociativo (jsondecode)
   //elegimos 3 preguntas aleatorias y la cargamos en un nuevo array.
   //hacemos uso de print_r para mostrar los resultados
   //El siguiente paso es generar un formulario para mostrar esas tres
preguntas.
```

Ejemplo de archivo JSON con preguntas y respuestas:

```
[

   {
     "pregunta": "¿Qué función se usa para mostrar texto en PHP?",
     "opciones": ["echo", "print", "display", "show"],
     "respuesta_correcta": "echo"
   },
   {
     "pregunta": "¿Qué símbolo se utiliza para indicar una variable en PHP?",
     "opciones": ["$", "@", "#", "&"],
     "respuesta_correcta": "$"
   },
   {
     "pregunta": "¿Cómo se declara una constante en PHP?",
     "opciones": ["define()", "const()", "constant()", "setconst()"],
     "respuesta_correcta": "define()"
   },
   {
     "pregunta": "¿Cuál es la extensión estándar de un archivo PHP?",
     "opciones": [".php", ".html", ".js", ".py"],
     "respuesta_correcta": ".php"
   },
   {
     "pregunta": "¿Qué función en PHP se usa para contar el número de elementos en un array?",
     "opciones": ["count()", "sizeof()", "length()", "array_count()"],
     "respuesta_correcta": "count()"
   },
```

```
  {
    "pregunta": "¿Qué función se utiliza para incluir un archivo en PHP?",
    "opciones": ["include()", "require()", "import()", "add_file()"],
    "respuesta_correcta": "include()"
  },
  {
    "pregunta": "¿Cuál es la versión más reciente de PHP?",
    "opciones": ["7.4", "8.0", "8.1", "8.3"],
    "respuesta_correcta": "8.3"
  },
  {
    "pregunta": "¿Cómo se inicia una sesión en PHP?",
    "opciones": ["session_start()", "session_init()", "session()", "start_session()"],
    "respuesta_correcta": "session_start()"
  },
  {
    "pregunta": "¿Qué superglobal contiene los datos enviados por un formulario mediante el método
POST?",
    "opciones": ["$_POST", "$_REQUEST", "$_GET", "$_FORM"],
    "respuesta_correcta": "$_POST"
  },
  {
    "pregunta": "¿Cómo se conecta a una base de datos MySQL en PHP?",
    "opciones": ["mysqli_connect()", "mysql_connect()", "pdo_connect()", "db_connect()"],
    "respuesta_correcta": "mysqli_connect()"
  },
  {
    "pregunta": "¿Qué función destruye una variable en PHP?",
    "opciones": ["unset()", "destroy()", "delete()", "unset_var()"],
    "respuesta_correcta": "unset()"
  },
  {
    "pregunta": "¿Cómo se realiza una redirección a otra página en PHP?",
    "opciones": ["header()", "redirect()", "goto()", "send_to()"],
    "respuesta_correcta": "header()"
  },
  {
    "pregunta": "¿Qué función se usa para obtener la longitud de una cadena en PHP?",
    "opciones": ["strlen()", "count()", "strlength()", "string_size()"],
    "respuesta_correcta": "strlen()"
  },
  {
    "pregunta": "¿Qué función se utiliza para filtrar y sanitizar una entrada en PHP?",
    "opciones": ["filter_var()", "sanitize_input()", "clean_var()", "validate_input()"],
    "respuesta_correcta": "filter_var()"
  },
  {
    "pregunta": "¿Qué función se utiliza para incluir un archivo sólo si aún no ha sido incluido?",
    "opciones": ["include_once()", "require_once()", "include()", "load_once()"],
    "respuesta_correcta": "include_once()"
  },
  {
    "pregunta": "¿Qué función se usa para detener la ejecución de un script en PHP?",
    "opciones": ["exit()", "stop()", "halt()", "break()"],
    "respuesta_correcta": "exit()"
  },
  {
    "pregunta": "¿Qué operador se utiliza para la concatenación de cadenas en PHP?",
    "opciones": [".", "+", "&", "&&"],
    "respuesta_correcta": "."
  },
  {
    "pregunta": "¿Qué tipo de error devuelve PHP cuando falta un archivo requerido?",
    "opciones": ["Fatal Error", "Warning", "Notice", "Parse Error"],
    "respuesta_correcta": "Fatal Error"
  },
  {
    "pregunta": "¿Cómo se define una función en PHP?",
    "opciones": ["function nombreFuncion()", "def nombreFuncion()", "define nombreFuncion()", "fn
nombreFuncion()"],
    "respuesta_correcta": "function nombreFuncion()"
  },
```

```
  {
    "pregunta": "¿Cómo se puede convertir un string a entero en PHP?",
    "opciones": ["(int)", "intval()", "(integer)", "cast_int()"],
    "respuesta_correcta": "intval()"
  }
]
```

Ejercicio de repaso:

Crear un **formulario de login** en PHP que solicite el nombre de usuario y la contraseña. La autenticación debe realizarse leyendo un archivo usuarios.csv donde cada línea tiene el formato: usuario,password,contador.

- El formulario tendrá campos para introducir el nombre de usuario y la contraseña, y un botón para enviar.
- Validar que ambos campos no estén vacíos.
- Si los campos están completos, verificar que el usuario y la contraseña coincidan con alguna entrada del archivo usuarios.csv.
- Si la autenticación es exitosa, incrementar en 1 el campo "contador" (número de logins) en el archivo usuarios.csv y mostrar un mensaje de éxito.
- Si no es correcta, mostrar un mensaje de error indicando que el usuario o la contraseña son incorrectos.

Aunque no es muy eficiente, vamos a reescribir el archivo csv por completo.

Información adicional sobre formularios:

¿Por qué validar en PHP si ya usamos `required` en HTML?

Seguridad: Aunque en HTML pongamos `required`, esa validación se puede desactivar o manipular fácilmente. Por ejemplo, alguien podría modificar el código del formulario directamente en el navegador o enviar la solicitud sin pasar por el formulario. Por eso, no podemos confiar únicamente en la validación del lado del cliente.

Consistencia: No todos los navegadores interpretan la validación HTML5 de la misma manera. Aunque el atributo `required` suele ser fiable, hay otros atributos de validación que pueden variar según el navegador. Al validar con PHP, te aseguras de que la validación sea siempre la misma, sin importar el navegador.

Control total: En PHP, puedes hacer validaciones mucho más específicas. Por ejemplo, podrías comprobar que un nombre no incluya caracteres no permitidos o que un email tenga el formato adecuado. PHP te permite tener control absoluto sobre los datos que recibes.

En resumen:

- La validación en el cliente (HTML) mejora la experiencia del usuario porque evita que se envíen formularios vacíos sin necesidad de recargar la página.
- La validación en el servidor (PHP) es imprescindible para garantizar la seguridad, consistencia y control de los datos.

Aunque `required` en HTML es útil, siempre es mejor hacer una validación adicional en PHP para tener una capa extra de seguridad.

—----------

Flujo recomendado para validar en PHP:

1. **Comprobar si el formulario ha sido enviado:** Utiliza isset($_GET['enviar']) o isset($_POST['enviar']), según el método de tu formulario. De esta manera, te aseguras de que el código PHP solo se ejecuta cuando realmente se ha enviado el formulario.
2. **Validar los datos enviados:** A pesar de haber verificado que el formulario ha sido enviado, es importante validar cada campo de forma individual con PHP. Utiliza isset() y !empty() para asegurarte de que los campos necesarios existen y tienen contenido. Esto es crucial, ya que no debes confiar al 100% en la validación del lado del cliente, que es fácil de saltar o modificar.