Guía para Formularios en HTML y PHP

1. ¿Qué es un formulario en HTML?

Un formulario en HTML es una estructura que permite a los usuarios ingresar información (como texto, números, correos, etc.) y enviarla al servidor para su procesamiento. Es muy común en aplicaciones web, por ejemplo, para registrarse, iniciar sesión o enviar comentarios.

2. Estructura básica de un formulario en HTML

```
<form action="procesar.php" method="POST">
   Nombre: <input type="text" name="nombre">
   Edad: <input type="number" name="edad">
   <input type="submit" value="Enviar">
</form>
```

1. <form>: La etiqueta principal del formulario.
   - action="procesar.php": Indica la página o archivo PHP donde se enviarán los datos cuando se envíe el formulario.
   - method="POST": Especifica cómo se envían los datos. POST es el más común, pero también se puede usar GET. Explicaremos la diferencia más adelante.

2. Campos de entrada:
   - <input type="text" name="nombre">: Un campo de texto para ingresar el nombre. El atributo name="nombre" es importante porque identifica el campo cuando enviamos los datos.
   - <input type="number" name="edad">: Un campo numérico para ingresar la edad.

3. Botón de envío:
   - <input type="submit" value="Enviar">: Un botón que envía los datos del formulario al archivo PHP cuando se hace clic.

3. Métodos GET vs POST

- GET: Los datos se envían en la URL del navegador. Este método es útil cuando los datos no son sensibles (como una búsqueda en Google). El tamaño de los datos es limitado.

Ejemplo de URL con GET: http://miweb.com/procesar.php?nombre=Juan&edad=25

- POST: Los datos se envían "escondidos" en el cuerpo de la solicitud HTTP, lo que es más seguro para información sensible (como contraseñas o correos). Este método no tiene un límite práctico en la cantidad de datos que puedes enviar.

4. Cómo se procesan los datos en PHP

Cuando enviamos un formulario, los datos se trasladan al archivo especificado en el atributo action del formulario. En PHP, podemos acceder a esos datos utilizando las superglobales $_GET o $_POST, dependiendo del método usado.

Ejemplo en PHP para recibir los datos:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // Recogemos los datos del formulario usando $_POST
   $nombre = $_POST['nombre'];
   $edad = $_POST['edad'];

   // Mostrar los datos
   echo "Nombre: " . htmlspecialchars($nombre) . "<br>";
   echo "Edad: " . htmlspecialchars($edad) . "<br>";
}
?>
```

5. Ejemplo completo: HTML + PHP

Archivo HTML (formulario.html):

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Formulario de Ejemplo</title>
</head>
<body>
   <h1>Formulario de Registro</h1>
   <form action="procesar.php" method="POST">
      Nombre: <input type="text" name="nombre"><br><br>
      Edad: <input type="number" name="edad"><br><br>
```

```
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

Archivo PHP (procesar.php):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $nombre = $_POST['nombre'];
  $edad = $_POST['edad'];

  // Verificamos que no estén vacíos
  if (!empty($nombre) && !empty($edad)) {
    echo "Nombre: " . htmlspecialchars($nombre) . "<br>";
    echo "Edad: " . htmlspecialchars($edad) . "<br>";
  } else {
    echo "Por favor, completa todos los campos.";
  }
}
?>
```

6. Diferencias clave entre GET y POST

- GET: Los datos se pasan en la URL, por lo que no es adecuado para información sensible. Ejemplo:

```
<form action="procesar.php" method="GET">
  Nombre: <input type="text" name="nombre">
  Edad: <input type="number" name="edad">
  <input type="submit" value="Enviar">
</form>
```

Para acceder a los datos en PHP:

```
$nombre = $_GET['nombre'];
$edad = $_GET['edad'];
```

- POST: Es más seguro porque los datos no se ven en la URL y puede manejar grandes cantidades de datos. Se recomienda para enviar datos sensibles como contraseñas,

formularios de registro, etc.

7. Errores comunes que debemos evitar

- Olvidar el atributo name en los campos: Si un campo no tiene name, no podrás acceder a su valor en PHP.

Ejemplo incorrecto:

<input type="text"> <!-- No tiene name -->

- Confundir GET y POST: Si usas el método POST en el formulario, debes asegurarte de acceder a los datos con $_POST[], no con $_GET[].

8. Ejercicios

1. Ejercicio básico con GET: Crea un formulario que pida el nombre y la edad y los envíe usando GET. Luego muestra los datos en la misma página.

2. Ejercicio con POST y validación: Crea un formulario que pida nombre, edad y correo. Valida que los campos no estén vacíos y que la edad sea un número mayor que 0.

3. Formulario de comentarios: Crea un formulario con un área de texto donde los usuarios puedan escribir un comentario. Valida que el comentario tenga al menos 10 caracteres antes de procesarlo.

IMPORTANTE: El objetivo es que para cada formulario referencieis un fichero php diferente. Una vez tengáis los tres ejercicios resueltos, uniremos los tres ficheros php en uno, controlando cada formulario.