Universidad Politécnica de Tlaxcala Región Poniente

Ingeniería en Sistemas Computacionales

Materia: Programación WEB

Docente: Ing. Vanesa Tenopala Zavala

Alumno: Isaac Brandon Martínez Ramírez

Matricula: 22SIC008

Tema: Reporte de Práctica: Server Side Scripting (SSS) con PHP

Ciclo escolar: septiembre - diciembre 2024

Fecha: 11 de noviembre de 2024

Índice

Reporte de Práctica: Server Side Scripting (SSS) con PHP

Objetivo

Aprender los fundamentos de Server Side Scripting (SSS) creando una aplicación sencilla de formulario de contacto. El objetivo es comprender cómo un formulario HTML envía datos al servidor, donde se procesan con PHP, y cómo el servidor responde al cliente con un mensaje adecuado.

Descripción General

La práctica consiste en crear un formulario HTML que envía los datos al servidor. Al enviarse, un script en PHP procesa los datos y muestra un mensaje de confirmación o error al usuario, dependiendo de la información proporcionada.

Herramientas Utilizadas

- **Servidor local (XAMPP/WAMP/MAMP)**: Permite ejecutar un servidor web y un servidor de bases de datos en nuestra máquina. Para esta práctica, usamos el servidor web Apache, que ejecuta scripts PHP.
- **HTML**: Es el lenguaje de marcado que se utiliza para crear la estructura del formulario en la interfaz de usuario.
- **PHP**: Es un lenguaje de scripting de servidor que se usa para procesar los datos enviados por el formulario y generar respuestas dinámicas.

Paso a Paso y Explicación del Código

Paso 1: Configuración del Servidor Local

1. **Instalación de XAMPP o WAMP**:
   - Se recomienda XAMPP o WAMP como servidores locales para ejecutar Apache (necesario para ejecutar scripts PHP). También incluye otros servicios, como MySQL, que pueden ser útiles en otras prácticas.
   - **Razón**: La instalación de un servidor local permite simular un entorno de servidor real, donde se procesan los scripts en el lado del servidor.
2. **Ubicación de archivos en el servidor**:
   - En XAMPP, los archivos HTML y PHP se colocan en la carpeta htdocs, que es el directorio de trabajo principal para archivos accesibles desde http://localhost/.
   - **Razón**: Los servidores locales requieren que los archivos de proyecto estén en ubicaciones específicas para ejecutarse correctamente.

Paso 2: Creación del Formulario HTML (index.html)

index.html > ...

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Contacto</title>
</head>
<body>
    <h1>Formulario de Contacto</h1>
    <form action="procesar.php" method="POST">
        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre" name="nombre" required>
        <br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <br><br>

        <label for="mensaje">Mensaje:</label>
        <textarea id="mensaje" name="mensaje" rows="4" required></textarea>
        <br><br>

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

- **Estructura del formulario**: Se crean tres campos: nombre, email, y mensaje, con un botón para enviar la información.
- **Atributo action="procesar.php"**: Define que, al enviar el formulario, los datos se enviarán al archivo procesar.php.
- **Método POST**: Utiliza el método POST para enviar datos al servidor. Este método es más seguro que GET para el envío de datos sensibles.
- **Atributo required**: Hace que los campos sean obligatorios, de manera que el formulario no se pueda enviar si están vacíos.

La estructura del formulario HTML permite al usuario ingresar datos y enviarlos al servidor para su procesamiento. El uso del método POST ayuda a mantener los datos en secreto en comparación con GET, que muestra los datos en la URL.

Paso 3: Creación del Script PHP para Procesar el Formulario (procesar.php)

procesar.php > ...

```php
<?php
// Para verificar si se enviaron los datos del formulario
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Guardar y sanitizar los datos
    $nombre = htmlspecialchars(trim($_POST["nombre"]));
    $email = htmlspecialchars(trim($_POST["email"]));
    $mensaje = htmlspecialchars(trim($_POST["mensaje"]));

    // Validar que no haya campos vacíos
    if (!empty($nombre) && !empty($email) && !empty($mensaje)) {
        // Mostrar un mensaje de confirmación
        echo "<h1>Gracias, $nombre.</h1>";
        echo "<p>Se ha recibido tu mensaje:</p>";
        echo "<blockquote>$mensaje</blockquote>";
        echo "<p>Nos pondremos en contacto contigo en <strong>$email</strong> pronto.</p>";
    } else {
        // Mensaje de error en caso de existan campos vacíos
        echo "<h1>Error</h1>";
        echo "<p>Por favor, completa todos los campos del formulario.</p>";
    }
} else {
    // Mensaje de error si se intenta acceder al script directamente
    echo "<h1>Error</h1>";
    echo "<p>Acceso no autorizado.</p>";
}
?>
```

- **Verificación del método de solicitud**: $_SERVER["REQUEST_METHOD"] verifica que la solicitud sea POST, evitando accesos no deseados a este archivo.

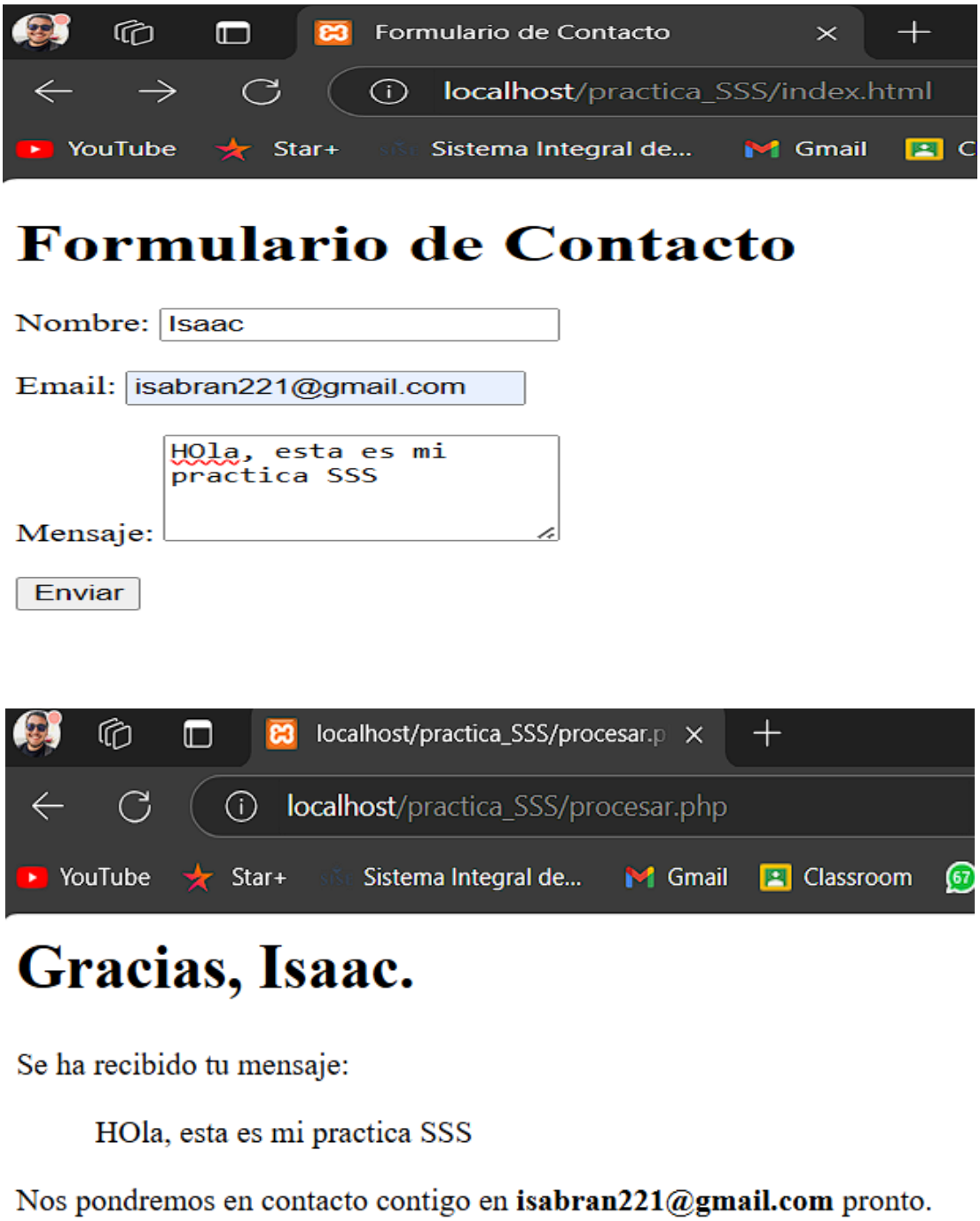
- **Sanitización de datos**:
  - trim(): Elimina espacios en blanco al inicio y al final de cada valor, lo que evita entradas no intencionadas.
  - htmlspecialchars(): Convierte caracteres especiales en entidades HTML, protegiendo contra ataques de tipo XSS (Cross-Site Scripting).
- **Validación de campos**: Si todos los campos tienen datos, se muestra un mensaje de confirmación; de lo contrario, se muestra un mensaje de error.
- **Respuesta personalizada**: Usamos echo para generar HTML con mensajes personalizados, como el nombre y mensaje del usuario, que le da una experiencia más interactiva.

Este script es esencial para procesar los datos en el servidor. La validación y sanitización de los datos aseguran que el servidor solo reciba información limpia y protegida contra ataques de inyección.

Paso 4: Prueba de la Aplicación

1. **Abrir el formulario en el navegador**: Visitar http://localhost/practica_SSS/index.html para ver el formulario.
2. **Prueba de envío**: Completar el formulario y enviar la información.
3. **Visualización de la respuesta**: El servidor muestra un mensaje de confirmación o error, basado en los datos ingresados.

Resultado



Conclusión

Esta práctica muestra cómo interactúan el lado del cliente (HTML) y el lado del servidor (PHP) en una aplicación web. Este flujo de trabajo permite familiarizarse con la estructura de una aplicación SSS y entender el ciclo completo de solicitud-respuesta.