

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación



Contenido

Variables súper globales	3
Variables \$_SERVER	5
Variables \$_GET	7
Variables \$_POST	9
Variables \$_COOKIE	12
Variables \$_SESSION.....	18
Variables \$_FILE	20
Variables \$_REQUEST	24
Variables \$_GLOBAL	26
Bibliografía	29
Para ampliar la información	29

Clase 5



¡Te damos la bienvenida a la materia
Programación Web II!

En esta clase vamos a ver los siguientes temas:

- Variables súper globales
- `$_SERVER`
- `$_GET`
- `$_POST`
- `$_COOKIE`
- `$_SESSION`
- `$_FILES`
- `$_REQUEST`
- `$_GLOBAL`

Variables súper globales

En programación, las variables superglobales son variables predefinidas que se pueden acceder desde cualquier parte del código, incluyendo funciones, clases, archivos y scripts.

Las variables superglobales son conocidas como "super" porque están disponibles en todos los ámbitos, lo que significa que no tienen un alcance limitado como las variables regulares.

En PHP, hay varias variables superglobales disponibles, como `$_SERVER`, `$_GET`, `$_POST`, `$_COOKIE`, `$_SESSION` y `$_FILES`. Cada una de estas variables tiene un propósito específico y se utiliza para recopilar información del entorno de ejecución del script PHP o del usuario que accede al sitio web.

Por ejemplo, `$_SERVER` contiene información sobre la solicitud de HTTP, incluyendo la dirección IP del cliente, el agente de usuario, el nombre del servidor y la URL actual. `$_GET` y `$_POST` se utilizan para recopilar información enviada a través de un formulario web, mientras que `$_COOKIE` y `$_SESSION` se utilizan para almacenar información del usuario en una sesión o en una cookie.

Las variables superglobales son útiles porque permiten que los datos se compartan y se accedan en todo el código sin tener que pasarlas manualmente entre funciones y clases. Sin embargo, es importante tener en cuenta que las variables superglobales pueden ser modificadas desde cualquier lugar del código, lo que significa que pueden ser vulnerables a ataques de seguridad si no se manejan adecuadamente. Por lo tanto, es importante utilizarlas con precaución y validar siempre los datos que se recopilan a través de ellas.

A continuación, se presenta una lista completa de las variables superglobales en PHP junto con un ejemplo y su respectiva salida:

- **\$_SERVER:** Contiene información del servidor y la solicitud del cliente. Algunos de los valores incluyen el nombre del servidor, la dirección IP del cliente, el agente de usuario, la ruta del script y la solicitud actual.
- **\$_GET:** Contiene datos enviados a través del método HTTP GET, como los valores de los parámetros de la URL.
- **\$_POST:** Contiene datos enviados a través del método HTTP POST, como los valores de los campos de un formulario HTML.
- **\$_COOKIE:** Contiene los valores de las cookies que se han configurado en el navegador del usuario.
- **\$_SESSION:** Contiene los valores de las variables de sesión que se han configurado en el servidor.
- **\$_FILES:** Contiene información sobre los archivos que se han subido al servidor a través de un formulario HTML con el atributo "enctype" establecido en "multipart/form-data".
- **\$_REQUEST:** Contiene los valores de `$_GET`, `$_POST` y `$_COOKIE` combinados.
- **\$_ENV:** Contiene información sobre las variables de entorno del sistema.
- **\$_GLOBALS:** Contiene una referencia a todas las variables globales definidas en el script.

En PHP, las variables superglobales son variables predefinidas que están disponibles en todos los ámbitos (lo que significa que pueden ser accedidas desde cualquier parte del código) y contienen información útil sobre la solicitud HTTP actual, el entorno de ejecución y otros datos globales.

Variables `$_SERVER`

La variable superglobal `$_SERVER` en PHP se utiliza para acceder a información sobre el entorno del servidor y detalles de la solicitud HTTP actual.

Proporciona una amplia gama de datos que son útiles en diferentes contextos. Algunos usos comunes de la variable `$_SERVER` son:

- Obtener información sobre la URL y la solicitud HTTP: Puedes acceder a la URL completa de la página actual utilizando `$_SERVER['REQUEST_URI']`. Además, `$_SERVER['HTTP_HOST']` te dará el nombre del host de la solicitud y `$_SERVER['REQUEST_METHOD']` te dará el método HTTP utilizado (GET, POST, etc.).
- Obtener información sobre la dirección IP del cliente: Podés utilizar `$_SERVER['REMOTE_ADDR']` para obtener la dirección IP del cliente que hizo la solicitud. Tené en cuenta que esta información puede ser manipulada o estar detrás de un proxy, por lo que no siempre es 100% precisa.
- Obtener información sobre el servidor: Podés acceder a información sobre el servidor donde se ejecuta el script PHP utilizando `$_SERVER['SERVER_SOFTWARE']` para obtener el software del servidor, `$_SERVER['SERVER_NAME']` para obtener el nombre del servidor y `$_SERVER['SERVER_PORT']` para obtener el número de puerto en el que se está ejecutando.
- Obtener información sobre el cliente y el navegador: La variable `$_SERVER` también proporciona detalles sobre el cliente que realizó la solicitud, como `$_SERVER['HTTP_USER_AGENT']`, que contiene el agente de usuario del navegador utilizado para acceder a la página.
- Obtener información sobre el esquema de la solicitud (HTTP o HTTPS): Podés verificar si la solicitud se realizó a través de HTTP o HTTPS utilizando `$_SERVER['HTTPS']`. Si su valor es "on", significa que se utilizó HTTPS.

Ejemplo de algunos usos de la variable `$_SERVER`:

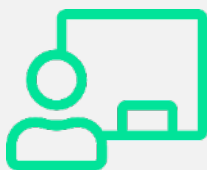
superglobales.php

```
<?php

echo $_SERVER['SERVER_SOFTWARE'];
echo "<br>";
echo $_SERVER['SERVER_ADDR'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_PROTOCOL'];
echo "<br>";
echo $_SERVER['SCRIPT_FILENAME'];
echo "<br>";
echo $_SERVER['REQUEST_METHOD'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";

?>
```

Vista en el Navegador



Para más información sobre las variables súper globales de PHP, podés leer la sección Variables Predefinidas en la documentación oficial del lenguaje.

Variables \$_GET

Ejemplo de la variable `$_GET`:

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Método Get</title>
</head>
<body>
    <a href="listado.php/?id=3">Enviar ID por URL</a>
</body>
</html>
```

listado.php

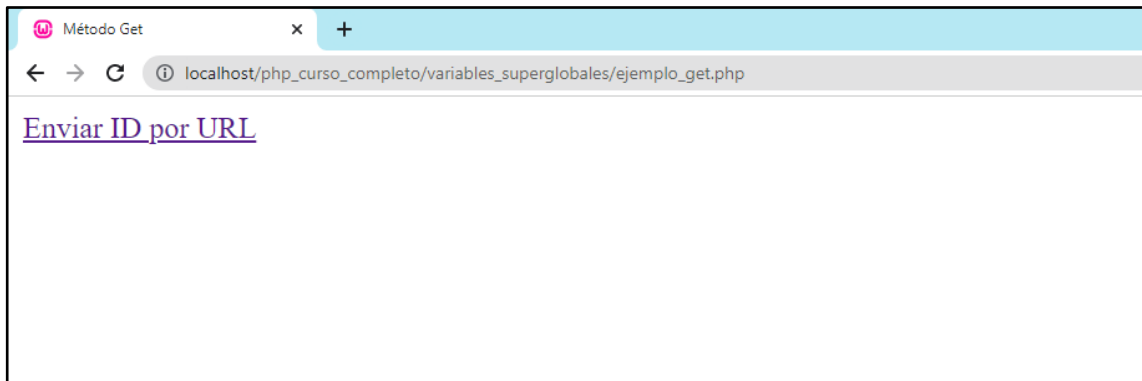
```
<?php

// recupera el dato enviado a través de la URL
$id = $_GET['id'];

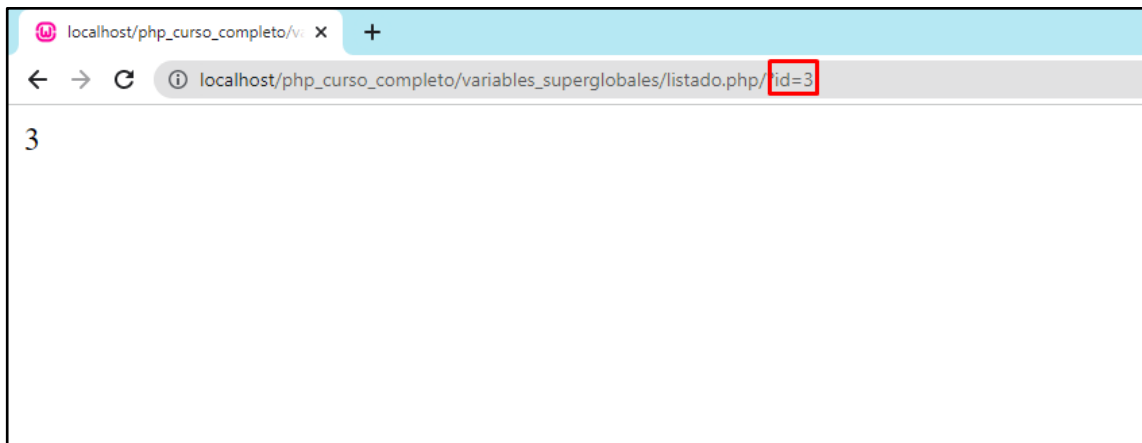
// muestra el número 3 en el navegador
echo $id;

?>
```


Vista en el Navegador antes de clicar sobre el enlace



Vista en el navegador luego de haber clickeado sobre el enlace



Variables \$_POST

Ejemplo de la variable \$_POST:

formulario.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <style>
        .etiquetas{
            text-align: right;
        }
    </style>

    <title>Método Post</title>
</head>
<body>

    <h1>Formulario de Contacto</h1>

    <form action="validar_datos.php" method="POST">
        <table>
            <tr>
                <td class="etiquetas"><label for="">Nombre:</label></td>
                <td><input type="text" name="nombre" id=""><br></td>
            </tr>
            <tr>
                <td class="etiquetas"><label for="">Apellido:</label></td>
                <td><input type="text" name="apellido" id=""><br></td>
            </tr>
            <tr>
                <td class="etiquetas"><label for="">Correo:</label></td>
                <td><input type="text" name="correo" id=""><br></td>
            </tr>
```

```

        <tr>
            <td class="etiquetas"><label for="">Usuario:</label></td>
            <td><input type="text" name="usuario" id=""><br></td>
        </tr>
        <tr>
            <td class="etiquetas"><label for="">Clave:</label></td>
            <td><input type="password" name="clave" id=""><br></td>
        </tr>
        <tr>
            <td><br></td>
        </tr>
        <tr>
            <td><input type="submit" value="Enviar formulario"></td>
        </tr>
    </table>
</form>

</body>
</html>

```

validar_datos.php

```

<?php

$nombre = $_POST['nombre'];
$apellido = $_POST['apellido'];
$correo = $_POST['correo'];
$usuario = $_POST['usuario'];

echo '<b>Nombre: </b>' . $nombre . '<br>';
echo '<b>Apellido: </b>' . $apellido . '<br>';
echo '<b>Correo: </b>' . $correo . '<br>';
echo '<b>Usuario: </b>' . $usuario . '<br>';

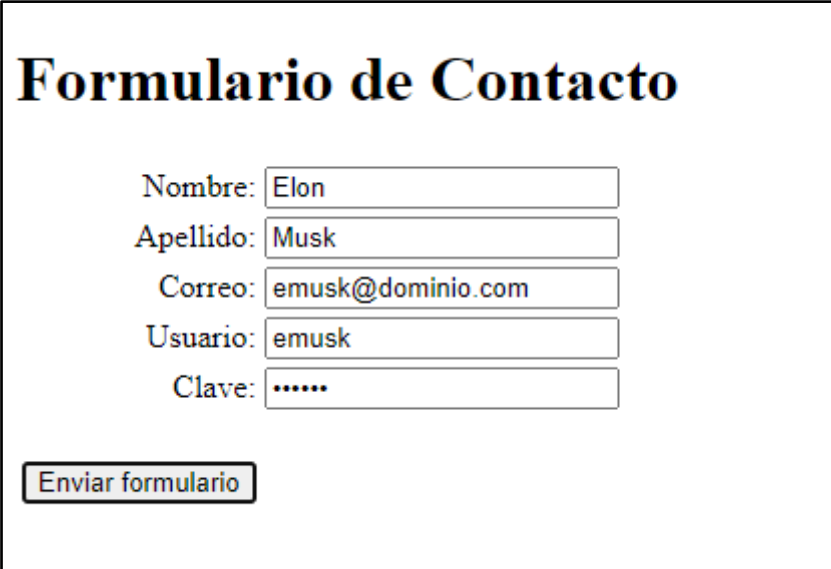
?>

```

En este ejemplo, el archivo HTML contiene un formulario de Contacto.

- Al enviar el formulario, los datos se envían al archivo "validar_datos.php" mediante el método POST.
- Se accede a los valores enviados desde el formulario utilizando la variable superglobal `$_POST`. Los campos del formulario se identifican por los nombres "nombre", "apellido", "correo" y "usuario".
- Luego, se utilizan las variables para mostrar los datos ingresados por el usuario utilizando `echo`.

Vista en el Navegador antes de clicar sobre el botón "Enviar formulario"



Formulario de Contacto

Nombre:

Apellido:

Correo:

Usuario:

Clave:

Vista en el Navegador antes de clicar sobre el botón "Enviar formulario"



Nombre: Elon
Apellido: Musk
Correo: emusk@dominio.com
Usuario: emusk

Variables \$_COOKIE

Ejemplo de la variable \$_COOKIE:

formulario2.php

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulario de inicio de sesión</title>
  <!-- Agrega los enlaces CSS de Bootstrap -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap
.min.css">

  <style>
    .container {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .login-form {
      width: 300px;
    }
  </style>

</head>
<body>

  <div class="container">
    <div class="login-form">
      <h1 class="mt-5">Inicio de sesión</h1>
      <form action="validar_ingreso.php" method="POST"
class="mt-4">
        <div class="form-group">
          <label for="usuario">Usuario:</label>
          <input type="text" name="usuario" id="usuario"
value="emusk" class="form-control" required autocomplete="off">
        </div>
        <div class="form-group">
```

```

        <label for="clave">Contraseña:</label>
        <input type="password" name="clave" id="clave"
value="123456" class="form-control" required autocomplete="off">
    </div>
    <button type="submit" class="btn btn-primary">Iniciar
sesión</button>
</form>
</div>
</div>

    <!-- Agrega los scripts de Bootstrap al final del cuerpo del
documento -->
    <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@1.16.0/dist/umd/popper
r.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.m
in.js"></script>
</body>
</html>

```

validar_ingreso.php

```

<?php

session_start();

if($_SERVER["REQUEST_METHOD"] == "POST"){

    if(isset($_POST["usuario"], $_POST["clave"])){

        if(!empty($_POST["usuario"] && !empty($_POST["clave"]))) {

            if($_POST["usuario"] == 'emusk' && $_POST["clave"] ==
123456){

                $usuario = $_POST["usuario"];

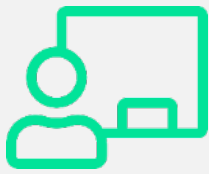
                // Establecemos la cookie "usuario" con el nombre del usuario, esta
expira en 30 días

```



```
        setcookie('usuario', $usuario, time() + (30 * 24 *  
60 * 60));  
  
        header('Location: welcome.php');  
  
        exit();  
  
    }  
  
    else{  
  
        echo 'Usuario y/o clave inválido.';  
  
    }  
  
}  
  
else{  
  
    echo 'Error, los campos Usuario y Clave son  
requeridos.';  
  
}  
  
}  
  
else{  
  
    echo 'Error, variables no definidas o nulas.';  
  
}  
  
}
```

?>



Aquí se presenta un ejemplo de un formulario de inicio de sesión básico

- Se inicia la sesión utilizando `session_start()`. Esto permite mantener y acceder a variables de sesión a lo largo de múltiples páginas en el sitio web.
- Se verifica si la solicitud es de tipo POST utilizando `$_SERVER["REQUEST_METHOD"] == "POST"`. Esto asegura que el código se ejecute sólo cuando se envíe el formulario.
- Se verifica si las variables `$_POST["usuario"]` y `$_POST["clave"]` están definidas utilizando `isset()`. Esto se realiza para asegurarse de que se hayan enviado los campos de usuario y clave desde el formulario.
- Se verifica si los campos de usuario y clave no están vacíos utilizando `!empty()`. Esto se realiza para asegurarse de que los campos no estén en blanco.
- Si el usuario ingresado es "emusk" y la clave es "123456", se establece una cookie llamada "usuario" con el valor del usuario. La cookie tiene una expiración de 30 días.
- Si el inicio de sesión es exitoso, se redirige al usuario a la página "welcome.php" utilizando `header('Location: welcome.php')`. La función `exit()` se utiliza para detener la ejecución del script después de redireccionar al usuario.
- Si el usuario y la clave no coinciden con los valores esperados, se muestra el mensaje "Usuario y/o clave inválido".
- Si los campos de usuario y clave están vacíos, se muestra el mensaje "Error, los campos Usuario y Clave son requeridos".
- Si las variables `$_POST["usuario"]` y `$_POST["clave"]` no están definidas, se muestra el mensaje "Error, variables no definidas o nulas".

Vista en el Navegador antes de clicar sobre el botón “Iniciar sesión”

Inicio de sesión

Usuario:

Contraseña:

Iniciar sesión

Vista en el Navegador luego de clicar sobre el botón “Iniciar sesión”

Aplicación					
Filtrar					
Aplicación	Nombre	Valor	Domain	Path	Expires / Max-Age
	PHPSESSID	umd69I7h29dacobf1b26fuophi	localhost	/	Sesión
	usuario	emusk	localhost	/cookies	2023-08-09T14:35:04.153Z
Almacenamiento					
▶ Almacenamiento local					
▶ Almacenamiento de la sesión					
Base de datos indexada					
Web SQL					
▼ Cookies					
http://localhost					
Tokens de estado privado					
Grupos de interés					
▶ Almacenamiento compartido					
Almacenamiento en caché					

Más ejemplos de la variable \$_COOKIE:

```
<?php

// Nombre de usuario (expira en un mes)

setcookie('usuario', $usuario, time() + (30 * 24 * 60 * 60));

// Idioma (expira en 6 meses)

setcookie('idioma', $idioma, time() + (6 * 30 * 24 * 60 * 60)); //
Ejemplo: idioma establecido en "es" (español)

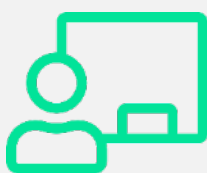
// Tema (expira en 1 año)

setcookie('tema', $tema, time() + (365 * 24 * 60 * 60)); // Ejemplo:
tema establecido en "dark" (oscuro)

?>
```

En este ejemplo, se utilizan las funciones `setcookie()` para establecer las cookies correspondientes.

- La cookie 'username' almacena el nombre de usuario y tiene una duración de un mes (30 días * 24 horas * 60 minutos * 60 segundos).
- La cookie 'language' almacena el idioma y tiene una duración de 6 meses (6 meses * 30 días * 24 horas * 60 minutos * 60 segundos).
- La cookie 'theme' almacena el tema y tiene una duración de 1 año (365 días * 24 horas * 60 minutos * 60 segundos).



Recordá que la función **setcookie()** debe llamarse antes de cualquier salida al navegador y que se recomienda colocarla al comienzo del archivo PHP, antes de cualquier etiqueta HTML o salida de texto

Variables \$_SESSION

Ejemplo de la variable `$_SESSION`:

sesion.php

```
<?php

// Iniciar la sesión
session_start();

// Establecer variables de sesión
$_SESSION['nombre'] = 'Juan';
$_SESSION['edad'] = 30;

// Acceder a las variables de sesión
$nombre = $_SESSION['nombre'];
$edad = $_SESSION['edad'];

// Mostrar las variables de sesión
echo "Nombre: $nombre<br>";
echo "Edad: $edad<br>";

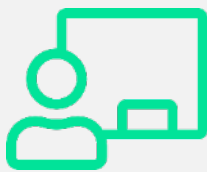
// Eliminar una variable de sesión
unset($_SESSION['edad']);

// Destruir la sesión
session_destroy();

?>
```

En este ejemplo, se inicia la sesión utilizando `session_start()`.

- Luego, se establecen variables de sesión como `$_SESSION['nombre']` y `$_SESSION['edad']`. Estas variables se pueden utilizar para almacenar información relevante para el usuario durante su sesión en la aplicación.
- Después, se accede a las variables de sesión y se asignan a variables locales (`$nombre` y `$edad` en este caso) para su uso posterior.
- Se muestran las variables de sesión utilizando `echo`, lo que imprimirá el nombre y la edad en el navegador.
- A continuación, se utiliza `unset($_SESSION['edad'])` para eliminar la variable de sesión 'edad'. Esto es opcional y se puede utilizar para eliminar una variable de sesión específica si ya no es necesaria.
- Finalmente, se destruye la sesión utilizando `session_destroy()`. Esto elimina todas las variables de sesión y cierra la sesión del usuario.



Recordá que es necesario llamar a **`session_start()`** al comienzo de cada página donde quieras utilizar las sesiones y que las variables de sesión estarán disponibles mientras la sesión esté activa

Variables \$_FILE

Ejemplo de la variable \$_FILE:

formulario3.php

```
<!DOCTYPE html>

<html>

<head>

    <title>Formulario de carga de archivo</title>

</head>

<body>

    <h1>Subida de Archivos</h1>

    <form method="POST" action="procesar.php" enctype="multipart/form-
data">

        <input type="file" name="archivo" required>

        <br />

        <br />

        <button type="submit"> Cargar archivo</button>

    </form>

</body>

</html>
```

procesar.php

```
<?php

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (isset($_FILES['archivo'])) {

        $archivo = $_FILES['archivo'];

        // Información del archivo

        $nombreArchivo = $archivo['name'];
        $tipoArchivo = $archivo['type'];
        $tamanoArchivo = $archivo['size'];
        $ubicacionTemporal = $archivo['tmp_name'];

        // Directorio de destino para guardar el archivo

        $directorioDestino = 'C:/wamp64/www/cookies/uploads/';
        $rutaArchivoDestino = $directorioDestino . $nombreArchivo;

        // Mover el archivo a su ubicación definitiva

        move_uploaded_file($ubicacionTemporal, $rutaArchivoDestino);

        echo 'El archivo se ha cargado correctamente.';
    } else {

        echo 'No se ha seleccionado ningún archivo.';
    }
} else {

    echo 'Método de solicitud no válido.';
}

?>
```

Vista en el Navegador antes de subir el archivo

Subida de Archivos

Sin archivos seleccionados

Vista en el Navegador luego de seleccionar un archivo

Subida de Archivos

bikepacking.jpg

Vista en el Navegador luego de clicar sobre el botón “Cargar archivo”

El archivo se ha cargado correctamente.

En este ejemplo, el archivo HTML contiene un formulario de carga de archivo.

- Al enviar el formulario, los datos se envían al archivo "procesar.php" mediante el método POST y el atributo "enctype" se establece en "multipart/form-data" para permitir la carga de archivos.
- En el archivo PHP "procesar.php", se verifica si se recibió una solicitud *POST* (`$_SERVER['REQUEST_METHOD'] === 'POST'`) y si se ha enviado un archivo (si `$_FILES['archivo']` está definido). Si se cumplen estas condiciones, se procede a procesar el archivo.
- La información del archivo se extrae de la variable `$_FILES['archivo']`. Luego, se define un directorio de destino donde se guardará el archivo. El archivo se mueve desde su ubicación temporal (proporcionada por `$_FILES['archivo']['tmp_name']`) al directorio de destino utilizando la función `move_uploaded_file()`.
- Finalmente, se muestra un mensaje indicando si el archivo se ha cargado correctamente o si no se ha seleccionado ningún archivo.

Variables \$_REQUEST

Ejemplo de la variable `$_REQUEST`:

formulario4.php

```
<html>

<body>

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

    Nombre: <input type="text" name="nombre" placeholder="Ingrese
nombre">

    <input type="submit" value="Enviar">

</form>

<?php

    if ($_SERVER["REQUEST_METHOD"] == "POST") {

        $nombre = $_REQUEST['nombre'];

        if (empty($nombre)) {

            echo "El campo nombre es requerido.";

        } else {

            echo $nombre;

        }

    }

?>

</body>

</html>
```

En este ejemplo, el archivo HTML contiene un formulario que permite al usuario ingresar su nombre y enviarlo.

- Luego, en el bloque de código PHP, se procesa el formulario y se muestra el nombre ingresado o un mensaje de error si el campo está vacío.
- Cuando el usuario envía el formulario, el código PHP verifica si se ha enviado una solicitud POST. Si es así, se accede al valor del campo de nombre proporcionado por el usuario. Si el campo está vacío, se muestra un mensaje de error indicando que es obligatorio. Si el campo no está vacío, se muestra el nombre ingresado por el usuario.
- El objetivo de este código es recibir los datos del formulario, validarlos y proporcionar una respuesta adecuada en función de los valores ingresados por el usuario.

Variables \$_GLOBAL

Ejemplo de la variable `$_GLOBAL`:

global.php

```
<?php

$mensaje = 'Hola, mundo!';

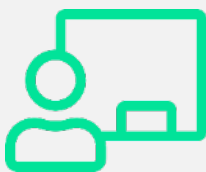
function imprimir_mensaje_global() {
    // Acceder a la variable global utilizando $GLOBALS
    echo $GLOBALS['mensaje'];
}

imprimir_mensaje_global();

?>
```

En este ejemplo, se define una variable llamada `$mensaje` con el valor 'Hola, mundo!' en el ámbito global del script.

- Luego, se crea una función llamada `imprimir_mensaje_global()` que accede a la variable global utilizando `$GLOBALS['mensaje']` y la imprime utilizando `echo`.
- Al llamar a la función `imprimir_mensaje_global()`, se mostrará el mensaje 'Hola, mundo!' que se encuentra en la variable global `$mensaje`.



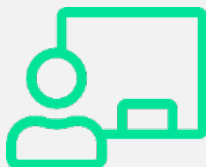
Recordá que es importante utilizar las variables globales con precaución y preferir el uso de parámetros y valores de retorno en las funciones para evitar posibles efectos secundarios y facilitar el mantenimiento del código.

Las variables superglobales en PHP, como \$_SERVER, \$_GET, \$_POST, \$_SESSION, \$_COOKIE, entre otras, son importantes porque proporcionan acceso a información crítica y contextual en diferentes áreas de desarrollo web. Al ser superglobales, están disponibles en todos los ámbitos y se pueden acceder desde cualquier parte del script sin necesidad de declararlas como globales explícitamente.



Hemos llegado así al final de esta clase en la que vimos:

- Variables súper globales
- `$_SERVER`
- `$_GET`
- `$_POST`
- `$_COOKIE`
- `$_SESSION`
- `$_FILES`
- `$_REQUEST`
- `$_GLOBAL`



Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.
¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP:

<https://www.php.net/manual/es/index.php>

W3Schools: PHP Tutorial:

<https://www.w3schools.com/php/default.asp>

Cabezas Granado, L., González Lozano, F., (2018). Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O., (2016). Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, L., Thompson, L., (2017). Desarrollo Web con PHP y MySQL. Quinta Edición. Editorial Anaya.

Para ampliar la información

[PHP: Variables predefinidas](#)

[PHP Global Variables - Superglobals](#)

[PHP: The Right Way](#)

[Todo sobre PHP](#)

[PHP Programming Language Tutorial - Full Course](#)

[PHP Full Course for non-haters](#)

[CURSO de php desde cero](#)

[MDN: Introducción al lado Servidor](#)

[Guía de HTML](#)