$_SERVER["REQUEST_METHOD"]

La expresión $_SERVER["REQUEST_METHOD"] == "POST" se utiliza en PHP para verificar si el formulario ha sido enviado utilizando el método HTTP POST

- **$_SERVER**: Es un array asociativo que se llena automáticamente por el servidor web y que contiene información sobre los encabezados, rutas y ubicaciones de scripts.

- **"REQUEST_METHOD"**: Es una clave en el array $_SERVER que contiene el método de solicitud utilizado para acceder a la página. Los métodos comunes son GET, POST, PUT, DELETE, etc.

- **"POST"**: Es el valor que se espera en $_SERVER["REQUEST_METHOD"] cuando el formulario ha sido enviado utilizando el método POST.

NOTACIÓN DE CORCHETES EN LOS FORMULARIOS

- **La notación de corchetes** es utilizada en formularios HTML para enviar conjuntos de datos que contienen múltiples valores. Esta notación permite que los datos se envíen como un array al servidor.

- **Acumulación de Valores**: PHP automáticamente acumula los valores en un array cuando se utiliza la notación de corchetes.

- **Acceso a Datos en PHP**: Utiliza la superglobal $_POST para acceder a los datos enviados y tratarlos como un array.

SITUACIONES COMUNES PARA USAR LA NOTACIÓN DE CORCHETES EN FORMULARIOS HTML

- **Select Múltiple:** Cuando se permite seleccionar múltiples opciones en un campo <select>.
- **Checkbox:** Cuando se tienen múltiples checkbox y se desea enviar todas las opciones seleccionadas.
- **File múltiple**: Cuando se permite la selección y envío de múltiples ficheros en un campo de tipo archivo.

EJEMPLOS

```html
<select name="opciones[]" multiple>
    <option value="opcion1">Opción 1</option>
    <option value="opcion2">Opción 2</option>
    <option value="opcion3">Opción 3</option>
</select>
```

El uso de `name="opciones[]"` permite que todas las opciones seleccionadas se envíen al servidor como un array llamado opciones.

```html
<input type="checkbox" name="preferencias[]" value="musica"> Música
<input type="checkbox" name="preferencias[]" value="deporte"> Deporte
<input type="checkbox" name="preferencias[]" value="cine"> Cine
```

Si el usuario selecciona más de una casilla, se enviarán como un array llamado `preferencias[]`.

```html
<input type="file" name="archivos[]" multiple>
```

Esto permite que el usuario seleccione múltiples archivos y los envíe al servidor, donde serán tratados como un array llamado `archivos[]`

¿CÓMO SE NUMERAN LOS ARRAYS?
SOLO EN CHECKBOX ES POSIBLE PONER ÍNDICES

Sin índices en los corchetes ([]): El array es numerado automáticamente por el servidor, comenzando desde 0.

Con índices dentro de los corchetes ([índice]): El array respetará los índices que se especifiquen en el nombre del campo.

EJEMPLO

```
<input type="checkbox" name="preferencias[1]" value="musica"> Música
<input type="checkbox" name="preferencias[3]" value="deporte"> Deporte
<input type="checkbox" name="preferencias[5]" value="cine"> Cine
```

Si el usuario selecciona deporte y cine. Se recibirá en el servidor:

```
$_POST['preferencias'] = array(
    3 => 'deporte',
    5 => 'cine'
);
```

SUBIR UN FICHERO A UN SERVIDOR WEB

SUPERGLOBAL $_FILES

EJEMPLO

```html
<h2>Subir un archivo (máximo 2 MB)</h2>

<form action="subir.php" method="post" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="2097152"> <!-- 2MB en bytes -->
    <label for="archivo">Selecciona un archivo:</label>
    <input type="file" name="archivo" id="archivo" accept=".jpg, .jpeg, .png, .gif">
    <input type="submit" value="Subir archivo">
</form>
```

EJEMPLO

```html
<h2>Subir varios archivos (máximo 2 MB cada uno)</h2>

<form action="subir.php" method="post" enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE" value="2097152"> <!-- 2MB en bytes -->
    <label for="archivo">Selecciona un archivo:</label>
    <input type="file" name="archivo[]" id="archivo" accept="image/*" multiple>
    <input type="submit" value="Subir archivo">
</form>
```

Notas importantes

- **El atributo accept es una sugerencia para el navegador**: No es un mecanismo de seguridad. Aunque restringe la selección de archivos en el lado del cliente, el servidor **debe verificar el tipo y tamaño de los archivos subidos** para evitar subir archivos no permitidos.
- **No todos los navegadores validan este atributo de la misma manera**, por lo que el procesamiento del archivo en el servidor es siempre esencial.

```
<input type="hidden" name="MAX_FILE_SIZE" value="2097152">
```

- Este campo oculto establece el tamaño máximo de archivo que puede ser subido.
- En el ejemplo, está configurado a 2 MB (2 * 1024 * 1024 = 2097152 bytes).
- Este es un límite en el lado del cliente (navegador) y no garantiza la restricción del tamaño del archivo en el servidor.

CONFIGURACIONES DEL SERVIDOR
ARCHIVO `PHP.INI`

upload_max_filesize: Este parámetro limita el tamaño máximo de **cada archivo** que puede ser subido.

post_max_size: Este parámetro limita el tamaño máximo total de los datos enviados en una solicitud HTTP **POST** (incluyendo todos los archivos, así como cualquier otro dato del formulario).

<FORM ACTION="SUBIR.PHP" METHOD="POST" ENCTYPE="MULTIPART/FORM-DATA">

Cuando se envían datos desde un formulario HTML, el navegador codifica la información de acuerdo con el tipo de contenido especificado en el atributo `enctype` del formulario.

Cada tipo de codificación trata los datos de manera diferente, especialmente en cómo maneja espacios en blanco y caracteres especiales.

Hay tres valores principales que se pueden utilizar con `enctype`

- **application/x-www-form-urlencoded**: Valor predeterminado. Codifica los datos del formulario como pares clave-valor. Adecuado para formularios sin archivos.
- **multipart/form-data**: Utilizado para formularios que incluyen archivos. Divide los datos del formulario en partes separadas.
- **text/plain**: Envía los datos del formulario sin codificación. No es seguro ni eficiente para la mayoría de las aplicaciones web.

ENCTYPE="APPLICATION/X-WWW-FORM-URLENCODED"

Cuando se envían datos a través de URL o formularios, hay caracteres que pueden causar confusión o problemas en la interpretación de esos datos.

Espacios: Un espacio en blanco puede interpretarse como el final de una cadena. Por lo tanto, se convierte en + o %20.

Caracteres especiales: Algunos caracteres como &, =, %, ?, etc., tienen un significado especial en una URL y deben ser codificados para que se interpreten correctamente como parte de los datos.

¿Cómo funciona la codificación a porcentaje?

Cada carácter que necesita ser codificado se representa en un formato especial: se reemplaza por un símbolo de porcentaje (%) seguido de su valor en hexadecimal.

```
<FORM ACTION="SUBIR.PHP" METHOD="POST" ENCTYPE="MULTIPART/FORM-DATA">
```

Este tipo de contenido se utiliza cuando un formulario incluye archivos. Divide los datos del formulario en partes separadas, lo que permite enviar archivos junto con otros datos.

- **Codificación**:
  - Cada campo se envía como un bloque separado en el cuerpo de la solicitud.
  - No se realizan transformaciones en los caracteres, los espacios y otros caracteres especiales se mantienen como están.
- **Uso**: Necesario para formularios que permiten la subida de archivos.

$_FILES

- La superglobal $_FILES es un array asociativo que contiene información sobre los archivos subidos a través de un formulario HTML.

- Esta superglobal se llena automáticamente cuando se envía un formulario con el atributo enctype="multipart/form-data" y contiene detalles sobre cada archivo subido.


$_FILES
VARIABLE SUPERGLOBAL

$_FILES

```html
<!-- El usuario elige un archivo llamado galleta.png -->
<h2>Subir un archivo (máximo 2 MB)</h2>
    <form action="subir.php" method="post" enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="2097152"> <!-- 2MB en bytes -->
        <label for="archivo">Selecciona un archivo:</label>
        <input type="file" name="nombreCampo" id="archivo" accept=".jpg, .jpeg, .png, .gif">
        <input type="submit" value="Subir archivo">
    </form>
```

```
C:\xampp\htdocs\xClase\asd\pruebas\subir.php:2:
array (size=1)
  'nombreCampo' =>
    array (size=6)
      'name' => string 'galleta.png' (length=11)
      'full_path' => string 'galleta.png' (length=11)
      'type' => string 'image/png' (length=9)
      'tmp_name' => string 'C:\xampp\tmp\php4663.tmp' (length=24)
      'error' => int 0
      'size' => int 204020
```

$_FILES

$_FILES['nombreCampo']['name']	El nombre original del fichero en la máquina del cliente.
$_FILES['nombreCampo']['full_path']	Si se sube un directorio, contiene las rutas relativas en el sistema de archivos del usuario.
$_FILES['nombreCampo']'type']	El tipo MIME del fichero, si el navegador proporcionó esta información.
$_FILES['nombreCampo’]['size']	El tamaño, en bytes, del fichero subido.
$_FILES['nombreCampo']'tmp_name']	El nombre temporal del fichero donde se almacena el fichero en el servidor.
$_FILES['nombreCampo']['error']	El código de error asociado a esta subida.

$_FILES SUBIDA MÚLTIPLE

```html
<!-- El usuario elige dos archivos llamados galleta.png y calamar.png -->
    <h2>Subir varios archivos</h2>
    <form action="subir.php" method="post" enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="10097152"> <!-- 9MB en bytes -->
        <label for="archivo">Selecciona un archivo:</label>
        <input type="file" name="nombreCampo[]" id="archivo" accept="image/*" multiple>
        <input type="submit" value="Subir archivo">
    </form>
```

```
C:\xampp\htdocs\xClase\asd\pruebas\subir.php:2:
array (size=1)
  'nombreCampo' =>
    array (size=6)
      'name' =>
        array (size=2)
          0 => string 'calamar.png' (length=11)
          1 => string 'galleta.png' (length=11)
      'full_path' =>
        array (size=2)
          0 => string 'calamar.png' (length=11)
          1 => string 'galleta.png' (length=11)
      'type' =>
        array (size=2)
          0 => string 'image/png' (length=9)
          1 => string 'image/png' (length=9)
      'tmp_name' =>
        array (size=2)
          0 => string 'C:\xampp\tmp\php370F.tmp' (length=24)
          1 => string 'C:\xampp\tmp\php3710.tmp' (length=24)
      'error' =>
        array (size=2)
          0 => int 0
          1 => int 0
      'size' =>
        array (size=2)
          0 => int 42344
          1 => int 204020
```

PROCESAMIENTO DE FICHERO EN EL SERVIDOR

Cuando un usuario sube un archivo a través de un formulario HTML, PHP **lo almacena temporalmente en un directorio específico del servidor antes de que el archivo sea procesado**.

Por defecto, esta ubicación es el directorio temporal del sistema, que puede variar según la configuración del servidor y el sistema operativo.

ELIMINACIÓN DE ARCHIVOS TEMPORALES

Eliminación automática: Los archivos almacenados en el directorio temporal se eliminan automáticamente al final de la solicitud, siempre y cuando no hayan sido movidos a otra ubicación en el servidor.

Esto significa que, **si el archivo no se procesa y se mueve a un directorio permanente**, se perderá después de que el script PHP termine su ejecución.

ELIMINACIÓN DE ARCHIVOS TEMPORALES

Es crucial que los scripts que manejan la carga de archivos:

- verifiquen los errores
- procesen el archivo
- lo muevan a su destino final utilizando la función **move_uploaded_file()** para evitar la pérdida de datos.

ELIMINACIÓN DE ARCHIVOS TEMPORALES

PHP devuelve un código de error junto con el array de ficheros ($_FILES). El código de error se puede encontrar en el segmento error del array de ficheros que PHP crea durante la subida de ficheros. En otras palabras, el error puede encontrarse en $_FILES['nombre']['error']: (no hay error 5)

UPLOAD_ERR_OK
Valor: 0; No hay error, fichero subido con éxito.

UPLOAD_ERR_INI_SIZE
Valor: 1; El fichero subido excede la directiva upload_max_filesize de php.ini

UPLOAD_ERR_FORM_SIZE
Valor: 2; El fichero subido excede la directiva MAX_FILE_SIZE especificada en el formulario HTML.

UPLOAD_ERR_PARTIAL
Valor: 3; El fichero fue sólo parcialmente subido.

UPLOAD_ERR_NO_FILE
Valor: 4; No se subió ningún fichero.

UPLOAD_ERR_NO_TMP_DIR
Valor: 6; Falta la carpeta temporal.

UPLOAD_ERR_CANT_WRITE
Valor: 7; No se pudo escribir el fichero en el disco.

UPLOAD_ERR_EXTENSION
Valor: 8; Una extensión de PHP detuvo la subida de ficheros. PHP no proporciona una forma de determinar la extensión que causó la parada de la subida de ficheros.

MOVE_UPLOADED_FILE (STRING $FILENAME ,STRING $DESTINATION) : BOOL

Esta función intenta asegurarse de que el archivo designado por filename es un archivo subido válido (lo que significa que fue subido mediante el mecanismo de subida HTTP POST de PHP).

Si el archivo es válido, será movido al nombre de archivo dado por $destination. DEBE contener la ruta relativa y el nombre del fichero con el que se quiere guardar.

```
move_uploaded_file($archivoTemporal, $rutaDestino);
```

Para un funcionamiento apropiado, la función **move_uploaded_file** () necesita un argumento como $_FILES['archivo_usuario']['tmp_name'], - **el nombre del archivo subido de la máquina del cliente $_FILES['archivo_usuario']['name'] no funciona.**

Si filename no es un archivo válido subido, no sucederá ninguna acción, y move_uploaded_file() devolverá FALSE.

Si filename es un archivo subido válido, pero no puede ser movido por alguna razón (no tenemos permiso para escribir en el directorio de destino…), no sucederá ninguna acción, y move_uploaded_file() devolverá FALSE.

```
mime_content_type(string $filename): string
```

Detecta el MIME para un fichero determinado.

Devuelve un string que contiene el tipo MIME para un fichero determinado.

$filename: La ruta al archivo cuyo tipo MIME deseas determinar. Este parámetro debe ser una cadena que contenga la ruta absoluta o relativa del archivo.

Valor de retorno: La función devuelve una cadena que representa el tipo MIME del archivo. Si el archivo no existe o no se puede determinar su tipo MIME, puede devolver un valor false.

`mime_content_type(string $filename): string`

Tipos MIME Comunes

Algunos tipos MIME comunes son:

- **Imágenes**:
  - image/jpeg: Para archivos JPEG.
  - image/png: Para archivos PNG.
  - image/gif: Para archivos GIF.
- **Documentos**:
  - application/pdf: Para archivos PDF.
  - application/msword: Para documentos de Microsoft Word.
  - text/plain: Para archivos de texto sin formato.

https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

mime_content_type(string $filename): string

```php
// Ruta del archivo que deseas verificar
$archivo = 'ruta/al/archivo.gif';

// Verificar si el archivo existe
if (file_exists($archivo)) {
    // Obtener el tipo MIME utilizando mime_content_type()
    $tipoMime = mime_content_type($archivo);
    echo "Tipo MIME usando mime_content_type: $tipoMime<br>";

    // Comparar si el tipo MIME es 'image/gif'
    if ($tipoMime === 'image/gif') {
        echo "El archivo es un GIF.<br>";
    } else {
        echo "El archivo NO es un GIF.<br>";
    }
}
```

FILESIZE(STRING $FILENAME):INT
FILE_EXISTS(STRING $FILENAME):BOOL

```php
// Ruta del archivo que deseas medir
$rutaArchivo = 'ruta/al/archivo.txt';

// Verificar si el archivo existe
if (file_exists($rutaArchivo)) {
    // Obtener el tamaño del archivo
    $tamaño = filesize($rutaArchivo); // El tamaño se devuelve en bytes

    // Convertir el tamaño a kilobytes (KB) y megabytes (MB) para mostrar
    $tamañoKB = $tamaño / 1024; // Convertir a KB
    $tamañoMB = $tamaño / (1024 * 1024); // Convertir a MB

    // Mostrar el tamaño del archivo
    echo "El tamaño del archivo es: " . $tamaño . " bytes<br>";
    echo "El tamaño del archivo es: " . round($tamañoKB, 2) . " KB<br>";
    echo "El tamaño del archivo es: " . round($tamañoMB, 2) . " MB<br>";
} else {
    echo "El archivo no existe.";
}
```

LA INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

- La inyección de código XSS (Cross-Site Scripting) es un tipo de vulnerabilidad en aplicaciones web que permite a los atacantes inyectar scripts maliciosos en las páginas web que son vistas por otros usuarios.
- Esto sucede cuando una aplicación no valida o filtra adecuadamente los datos que recibe antes de mostrarlos en el navegador.
- Estos scripts pueden ser utilizados para robar información sensible (como cookies), manipular el contenido de la página, redirigir a los usuarios a sitios peligrosos, entre otras acciones.

INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

- Para prevenir ataques XSS (Cross-Site Scripting) en PHP, es esencial **validar, filtrar y escapar** correctamente cualquier entrada de usuario que se renderice en una página web.

1. **Escapar la salida de datos** (output escaping). Cuando los datos del usuario se muestran en el navegador, debes asegurarte de escaparlos correctamente para evitar la inyección de scripts maliciosos.

   - htmlspecialchars(): Convierte caracteres especiales en entidades HTML. Esto es crucial al mostrar datos que podrían contener caracteres peligrosos como <, >, ", y '.

```php
// La función htmlspecialchars convierte los caracteres especiales en entidades HTML, de modo que el texto se muestra en la página web en lugar de ejecutarse como código JavaScript.

    $nombre = '<script>alert("Hola: inyección XSS!!!");</script>';
    echo htmlspecialchars($nombre, ENT_QUOTES, 'UTF-8');

// La constante ENT_QUOTES en la función htmlspecialchars especifica que tanto las comillas dobles (") como las comillas simples (') deben ser convertidas en entidades HTML.
```

INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

2. **Filtrar y validar** las entradas del usuario (input validation):

Nunca confíes en los datos que vienen del usuario. Siempre filtra y valida antes de utilizarlos. PHP ofrece varias funciones y filtros para validar los datos de entrada.

Función filter_input() o filter_var() se utiliza para filtrar y validar los datos de entrada, ya sean provenientes de formularios, cookies, variables de sesión o incluso entradas del servidor.

Esta función es crucial para la seguridad en aplicaciones web, ya que permite procesar y validar la información recibida del usuario antes de utilizarla.

INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

```
filter_input($tipo_de_dato, $nombre_de_variable, $filtro, $opciones=0): mixed
filter_var($nombre_de_variable, $filter,  $options = 0): mixed
```

Las funciones filter_input() y filter_var() en PHP son similares en cuanto a la aplicación de filtros:

- filter_input() está especialmente diseñada para trabajar con variables de entrada provenientes de formularios (a través de $_POST), parámetros de URL (a través de $_GET), $_COOKIES, $_SERVER o $_ENV. Si la variable no existe o no está definida en la entrada especificada (por ejemplo, si no existe en $_POST), filter_input() devolverá null. NO genera warning
- filter_var() es más flexible, ya que puedes usarla para filtrar cualquier dato, sin importar de dónde provenga. Si la variable que pasas está vacía ‘’ o NULL, simplemente devolverá el valor tal como lo recibe. Si no existe, genera un warning.

INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

- `filter_input($tipo_de_dato, $nombre_de_variable, $filtro, $opciones=0): mixed`
- `filter_var($nombre_de_variable, $filtro,  $opciones = 0): mixed`

1. tipo_de_dato: El tipo de entrada de donde proviene la variable. Los valores posibles son:
   - INPUT_GET (para variables $_GET)
   - INPUT_POST (para variables $_POST)
   - INPUT_COOKIE (para variables $_COOKIE)
   - INPUT_SERVER (para variables $_SERVER)
   - INPUT_ENV (para variables $_ENV)

2. nombre_de_variable: El nombre de la variable que quieres filtrar (una cadena que corresponde a la clave de la matriz de datos, como 'nombre' en $_POST['nombre']) o bien una variable cualquiera en el caso de filter_var().
3. filtro: El filtro a aplicar a los datos. Puedes usar una amplia gama de filtros nativos de PHP o definir tus propios filtros. Los filtros más comunes se detallan a continuación.
4. opciones: (opcional) Un array asociativo de opciones para modificar el comportamiento del filtro.

INYECCIÓN DE CÓDIGO XSS (CROSS-SITE SCRIPTING)

FILTER_VALIDATE_INT: Valida que el valor es un número entero.

FILTER_VALIDATE_FLOAT: Valida si el valor es un número de punto flotante.

Opciones: min_range (mínimo permitido) y max_range (máximo permitido).

FILTER_VALIDATE_BOOLEAN: Valida si el valor es un booleano (true, false, 1, 0, "yes", "no").

```php
$edad = filter_input(INPUT_POST, 'edad', FILTER_VALIDATE_INT);

$edad = filter_input(INPUT_POST, 'edad', FILTER_VALIDATE_INT, ['options' => ['min_range' => 1, 'max_range' => 120]]);

$precio = filter_input(INPUT_POST, 'precio', FILTER_VALIDATE_FLOAT);

$precio = filter_input(INPUT_POST, 'precio', FILTER_VALIDATE_FLOAT, ['options' => ['min_range' => 0, 'max_range' => 1000]]);

$acepto = filter_input(INPUT_POST, 'acepto', FILTER_VALIDATE_BOOLEAN);
```

Filtros de Validación

FILTER_VALIDATE_URL:Valida si el valor es una URL bien formada.

FILTER_VALIDATE_EMAIL: Valida si el valor es un correo electrónico válido.

FILTER_VALIDATE_IP:Valida si el valor es una dirección IP válida. Soporta tanto IPv4 como IPv6.

```php
$sitio_web = filter_input(INPUT_POST, 'sitio_web', FILTER_VALIDATE_URL);

$correo = filter_input(INPUT_POST, 'correo', FILTER_VALIDATE_EMAIL);

$ip = filter_input(INPUT_POST, 'ip', FILTER_VALIDATE_IP);
```

Filtros de Validación

```php
//Elimina todos los caracteres excepto letras, dígitos y los caracteres !#$%&'*+-/=?^_{|}~@.[]`.
$correo = filter_input(INPUT_POST, 'correo', FILTER_SANITIZE_EMAIL);

//Elimina todos los caracteres excepto letras, dígitos y los caracteres ~!$&'()*+,;=:/?@.
$sitio_web = filter_input(INPUT_POST, 'sitio_web', FILTER_SANITIZE_URL);

//Escapa caracteres especiales en HTML (<, >, &, y ").
$comentario = filter_input(INPUT_POST, 'comentario', FILTER_SANITIZE_FULL_SPECIAL_CHARS);

// Sanitizar diferentes tipos de entrada. Se diferencia de la anterior porque también elimina las comillas simples.
$nombre = htmlspecialchars($_POST['nombre'], ENT_QUOTES, 'UTF-8');

//Elimina todos los caracteres excepto los dígitos, el signo más y el signo menos.
$edad = filter_input(INPUT_POST, 'edad', FILTER_SANITIZE_NUMBER_INT);

//Elimina todos los caracteres excepto los dígitos, el signo más, el signo menos, el punto y la coma.
$precio = filter_input(INPUT_POST, 'precio', FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRACTION);
//obsoleta en php 8.1 utilizar htmlspecialchars
//$nombre = filter_input(INPUT_POST, 'nombre', FILTER_SANITIZE_STRING);
```

FILTROS DE SANITIZACIÓN SE UTILIZAN PARA LIMPIAR O MODIFICAR LOS DATOS DE ENTRADA. ESTOS FILTROS ELIMINAN O ESCAPAN CARACTERES NO DESEADOS PARA ASEGURAR QUE LOS DATOS SEAN SEGUROS PARA SU USO EN LA APLICACIÓN.

FILTRAR vs *SANITIZAR (LIMPIAR-SANEAR)*

Filtrar (validar) se usa para asegurarse de que los datos tienen el **formato o valor correcto** (por ejemplo, si un campo es un correo electrónico válido o un número dentro de un rango).

Sanitizar (limpiar) se utiliza para eliminar o neutralizar cualquier contenido que pueda ser dañino o peligroso (por ejemplo, para prevenir XSS o inyecciones de código).

FILTRAR vs *SANITIZAR (SANEAR-LIMPIAR)*

```php
$edad = "15años";
$edad = filter_var($edad, FILTER_VALIDATE_INT);
var_dump($edad);
```

```
boolean false
```

```php
$edad = 15;
$edad = "15";
$edad = filter_var($edad, FILTER_VALIDATE_INT);
var_dump($edad);
```

```
int 15
```

```php
$edad = "15años";
$edad = filter_var($edad, FILTER_SANITIZE_NUMBER_INT);
var_dump($edad);
```

```
string '15'
```

```php
$edad = 15;
$edad = filter_var($edad, FILTER_SANITIZE_NUMBER_INT);
var_dump($edad);
```

```
string '15'
```

FILTRAR vs *SANITIZAR (SANEAR-LIMPIAR)*

```php
$edad = "15años";
$edad = filter_var($edad, FILTER_SANITIZE_NUMBER_INT);
var_dump($edad);
```

```
string '15'
```

```php
$edad = 15;
$edad = filter_var($edad, FILTER_SANITIZE_NUMBER_INT);
var_dump($edad);
```

```
string '15'
```

El propósito de FILTER_SANITIZE_NUMBER_INT es sanitizar cadenas que contienen números, eliminando cualquier carácter no válido. Como PHP no puede estar seguro de que lo que está sanitizando es un número entero o parte de una cadena, devuelve siempre una cadena como resultado de la sanitización.

```php
$edad = (int) filter_var($edad, FILTER_SANITIZE_NUMBER_INT);
```

FILTRAR vs *SANITIZAR (SANEAR-LIMPIAR)*

Característica	Filtrar (Validar)	Sanitizar (Limpiar)
Propósito	Verificar si los datos son válidos	Limpiar los datos de caracteres peligrosos
Acción	Aceptar o rechazar la entrada	Modificar la entrada para hacerla segura
Resultado	En caso de éxito, valor de la variable pedida, false si el filtro falla o null si la variable no está definida	Devuelve los datos "limpios"
Ejemplo de uso	Validar que un correo electrónico esté bien formado	Eliminar etiquetas HTML o scripts de una entrada
Tipo de riesgo abordado	Prevención de datos inválidos (formato o estructura)	Prevención de código malicioso (XSS, inyecciones)

```
PATHINFO(STRING $PATH, INT $OPTIONS = PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME): MIXED
```

La función pathinfo() se usa para obtener información sobre una ruta de archivo. Devuelve un array asociativo que puede contener el directorio, el nombre del archivo y la extensión del archivo, según lo que se especifique.

```php
<?php
    $archivo = '/var/www/html/index.php';
    //$info es un array asociativo
    $info = pathinfo($archivo);
    echo "<pre>";
    print_r($info);
?>
```

```
Array
(
    [dirname] => /var/www/html
    [basename] => index.php
    [extension] => php
    [filename] => index
)
```

```php
<?php
$archivo = '/var/www/html/index.php';
//$info es un string con la extensión del archivo
$info = pathinfo($archivo, PATHINFO_EXTENSION);
echo $info;
?>
```

```
php
```

```
EXPLODE ( STRING $DELIMITER , STRING $STRING , INT $LIMIT = PHP_INT_MAX):ARRAY
```

La función explode() divide una cadena y la convierte en un array utilizando un delimitador especificado.

- **$delimiter**: El delimitador que se utiliza para dividir la cadena.
- **$string**: La cadena de texto que se va a dividir.
- **$limit** (opcional): Si se proporciona, limita el número de elementos en el array resultante.

```php
<?php
$cadena = "uno,dos,tres,cuatro";
$array = explode(",", $cadena);
echo "<pre>";
print_r($array);
?>
```

```
Array
(
    [0] => uno
    [1] => dos
    [2] => tres
    [3] => cuatro
)
```

IMPLODE(STRING $SEPARATOR, ARRAY $ARRAY): STRING

La función implode() une los elementos de un array en una cadena de texto utilizando un delimitador especificado.

- **$separator**: Delimitador que se utiliza para unir los elementos del array
- **$array**: Array de elementos que se va a unir en un string.

```php
<?php
$array = ["uno", "dos", "tres", "cuatro"];
$cadena = implode("-", $array);
echo $cadena;
?>
```

uno-dos-tres-cuatro