

U.T. 3: Formularios II

Validación de datos externos (formularios)

Validación datos externos

Además de la validación de los datos que se realiza en el lado cliente es importante realizarla también en el lado servidor ya que con esto garantizaremos la seguridad y calidad de los datos.

Validación datos externos

Algo importante a tener en cuenta es que los datos que vienen del exterior por POST o GET siempre son tipo string.

Validación datos externos

Contamos con varias posibilidades:

- Funciones validación números y letras.
(`is_numeric`, `ctype_digit`, `is_string`, `ctype_Alpha`)
- Filtros de validación `filter_var`

<https://www.php.net/manual/es/filter.filters.validate.php>

- Funciones propias de validación con expresiones regulares.

Aspectos básicos de seguridad en PHP

Balancear riesgo y usabilidad

- Asumir que algunas medidas de seguridad pueden influir en la rapidez de acceso.
- Utilizar medidas transparentes al usuario y que afecten lo menos posible a la usabilidad.

Rastrear la información

- Saber de donde vienen los datos.
Conocer origen.
 - Origen interno → Datos seguros
 - Origen externo → Datos no seguros. NUNCA confíes de datos que vienen de usuarios o de cualquier otra fuente externa
- Conocer cual va a ser la salida de los datos.

Ataque XSS (**Cross-site scripting**)

Un ataque XSS consiste en ejecutar cualquier código Javascript en el navegador de un usuario.

Objetivos que se quieren conseguir con ataques XSS son:

- Robar cookies y sesiones de usuarios
- Modificar el sitio web
- Redireccionar a usuarios a sitios dañinos
- Instalar malware
- Reescribir o manipular extensiones de navegador, etc.

Saber más sobre XSS

En los siguientes enlaces encontrarás más información sobre XSS, cómo prevenirlo, qué consecuencias puede tener y algunos ejemplos:

- <https://ciberseguridad.com/amenazas/vulnerabilidades/cross-site-scripting/>
- https://es.wikipedia.org/wiki/Cross-site_scripting#AJAX
- <https://es.myservername.com/cross-site-scripting-attack-tutorial-with-examples#Conclusion>

Prevenir ataques XSS

Partimos de que no tenemos que confiar en los datos externos.

Algunas medidas a tomar para garantizar la seguridad en nuestros sitios web con respecto a los datos externos.

Podemos agruparlas de la siguiente manera:

- Data sanitization (eliminar entrada de código malintencionado)
- Data validation (validación de los datos de entrada)
- Output escaping (escapado de los datos de salida)

Data sanitization

Consiste en limpiar las entradas de datos externos y eliminar el posible código malintencionado.

Dos opciones:

- Utilizar filtros de saneamiento propios del lenguaje, en nuestro caso PHP.

<https://www.php.net/manual/es/filter.filters.sanitize.php>

- .
- Crear nuestras propias funciones de saneamiento.

Funciones de saneamiento de texto personalizadas

Si es texto, no necesitamos almacenar ninguna etiqueta.

- Al recoger el dato al menos tenemos que:
 - Comprobar si existe, sino sustituirlo por ""
 - Eliminar espacios sobrantes **trim** y espacios intermedios sobrantes **preg_replace**
 - Limpiar posibles etiquetas html en el texto **strip_tags**

Funciones de saneamiento de texto personalizadas

Esta podría ser una función para recoger datos

```
function recoge($var)
{
    if (isset($_REQUEST[$var]))
        $tmp=strip_tags(sinEspacios($_REQUEST[$var]));
    else
        $tmp= "";

    return $tmp;
}
```

La función **sinEspacios** elimina los espacios sobrantes de la cadena

Otras funciones de saneamiento de texto personalizadas

Hay casos especiales, a tener en cuenta, por ejemplo:

- Entradas que deben mantener etiquetas HTML.
- Entradas tipo array.
- Entradas de radio button y checkbox

Escapado de la salida

Os remito a la página de Diego Lázaro donde explica de forma resumida el “escapado” de la salida de datos.

<https://diego.com.es/escape-de-datos-de-salida-en-php>

FORMULARIOS QUE SE AUTOPROCESAN

FORMULARIOS: Procesar formularios desde PHP

- ❑ Una forma de trabajar con formularios en PHP es utilizar un **único programa** que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
- ❑ Ventajas:
 - ❑ Disminuye el número de ficheros
 - ❑ Permite validar los datos del formulario en el propio formulario
- ❑ Procedimiento:

```
si se ha enviado el formulario
    Procesar formulario
si no
    Mostrar formulario
finsi
```

FORMULARIOS: Procesar formularios desde PHP

❑ Esquema de funcionamiento

si se ha enviado el formulario
Procesar formulario

si no
Mostrar formulario
finsi

- ❑ La 1ª vez que se carga la página se muestra el formulario
 - ❑ La 2ª vez se procesa el formulario
-

FORMULARIOS: Procesar formularios desde PHP

- ❑ Para saber si se ha enviado el formulario se acude a la variable correspondiente al botón de envío. Si este botón aparece de la siguiente forma en el formulario HTML:

`<INPUT TYPE=SUBMIT NAME="enviar" VALUE="procesar">`

- ❑ entonces la condición anterior se transforma en:

`if (isset($_POST['enviar']))`

- ❑ o bien

`if ($_POST['enviar'] == "procesar")`

FORMULARIOS: Validación de los datos de un formulario

❑ Esquema de funcionamiento:

si se ha enviado el formulario
validar datos
finsi

si se ha enviado y no hay errores
Procesar formulario

si no
Mostrar formulario
finsi

FORMULARIOS

☐ Esquema de funcionamiento:

- ☐ La 1ª vez que se carga la página se muestra el formulario
 - ☐ La 2ª y sucesivas veces se validan los datos
 - ☐ Si hay errores, se muestra de nuevo el formulario con los errores
 - ☐ Si no hay, se procesa el formulario
-

ADJUNTAR FICHEROS A FORMULARIOS

FORMULARIOS: Subida de ficheros al servidor

- ❑ Para subir un fichero al servidor se utiliza el elemento de entrada FILE
 - ❑ Hay que tener en cuenta una serie de consideraciones importantes:
 - ❑ El elemento FORM debe tener el atributo `ENCTYPE="multipart/form-data"`
 - ❑ El fichero tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes y complementarias:
 - ❑ En el fichero de configuración php.ini
 - ❑ En el propio formulario
 - ❑ Comprobándolo directamente en el código, es otra posibilidad.
-

FORMULARIOS: Subida de ficheros al servidor

❑ php.ini

```
.....  
,,,,,,,,,,,,,  
; File Uploads ;  
.....  
,,,,,,,,,,,,,  
; Si se permite o no subir archivos mediante HTTP  
file_uploads = On  
;  
; Tamaño máximo de cada archivo subido.  
upload_max_filesize = 2M  
; Tamaño máximo de los datos mandados por POST  
;(incluidos los que no sean archivos)  
post_max_size = 8M
```

❑ formulario

se tiene que llamar así y es un entero con el valor en bytes

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE='102400'>  
<INPUT TYPE="FILE" NAME="fichero">
```

FORMULARIOS: Subida de ficheros al servidor

```
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- ❑ La variable `$_FILES` contiene toda la información del fichero subido:
 - ❑ `$_FILES['imagen']['name']`
 - ❑ Nombre original del fichero en el cliente
 - ❑ `$_FILES['imagen']['type']`
 - ❑ Tipo MIME del fichero. Por ejemplo, "image/gif"
 - ❑ `$_FILES['imagen']['size']`
 - ❑ Tamaño en bytes del fichero subido
 - ❑ `$_FILES['imagen']['tmp_name']`
 - ❑ Nombre temporal del fichero que se genera para guardar el fichero subido
 - ❑ `$_FILES['imagen']['error']`
 - ❑ Código de error asociado a la subida del fichero
-

FORMULARIOS: Subida de ficheros al servidor

❑ Consideraciones (cont)

- ❑ Debe darse al fichero un **nombre único**. Por ello, y como norma general, debe descartarse el nombre original del fichero y crear uno nuevo que sea único, p.e añadiéndole la fecha y hora
- ❑ El fichero subido se almacena en un directorio temporal y hemos de moverlo al directorio de destino usando la función

move_upload_file()

❑ Procedimiento:

si se ha subido correctamente el fichero

Asignar un nombre al fichero

Mover el fichero a su ubicación definitiva

si no

Mostrar un mensaje de error

finsi

Validación de datos

No es suficiente la validación del lado cliente

Podemos realizarla de dos maneras:

- Utilizando filtros de validación de PHP.
<http://php.net/manual/es/filter.filters.validate.php>
- Utilizando expresiones regulares.
<http://php.net/manual/es/book.pcre.php>

FORMULARIOS: Subida de ficheros al servidor

☐ Procedimiento:

si se ha subido correctamente el fichero

- ☐ Lo comprobamos con **is_uploaded_file("nombre temporal de \$_FILES")**
- ☐ Devuelve **true** si el archivo que se le pasa se ha subido por HTTP POST. Evita que el usuario intente usar archivos del servidor /etc/passwd

Asignar un nombre al fichero

- ☐ Añadir marca de tiempo

Mover el fichero a su ubicación definitiva

- ☐ **move_uploaded_file (\$_FILES['archivo'] ['tmp_name'], \$destino)**
- ☐ Lo mueve y si no puede da error




si no

Mostrar un mensaje de error

finsi

FORMULARIOS: Subida de ficheros al servidor

is_uploaded_file (\$_FILES['imagen']['tmp_name'])

-  Devuelve TRUE si el archivo que se pasa fue cargado a través de HTTP POST. Evita que un usuario intente que se manejen ficheros no cargados por POST. p.e /etc/passwd
-  Necesita como argumento \$_FILES['archivo_usuario']['tmp_name']
-  Si se le pasa \$_FILES['archivo_usuario']['name'] **no funciona.**

```
<?php
if (is_uploaded_file($_FILES['archivo_usuario']['tmp_name'])) {
    echo "El archivo ". $_FILES['archivo_usuario']['name'] ." fue cargado correctamente.\n";
    echo "Mostrando su contenido\n";
    readfile($_FILES['archivo_usuario']['tmp_name']);
} else {
    echo "Posible ataque de carga de archivo: ";
    echo "nombre de archivo ". $_FILES['archivo_usuario']['tmp_name'] . " . " .";}
?>
```

FORMULARIOS: Subida de ficheros al servidor

❑ **move_uploaded_file (\$_FILES['imagen']['tmp_name'], \$destino)**



nombre_temporal_archivo

- ❑ Esta función realiza un chequeo para asegurar que el archivo indicado por el primer parámetro sea un archivo cargado a través de HTTP POST.
 - ❑ Si el archivo es válido, será movido al nombre de archivo dado por **destino**.
 - ❑ Si *nombre_temporal_archivo* no es un archivo cargado válido, no hará nada, y devolverá FALSE.
 - ❑ Si *nombre_temporal_archivo* es un archivo cargado válido, pero no puede ser movido por alguna razón, no hará nada, devolverá FALSE y dará una advertencia.
-

FORMULARIOS: Subida de ficheros al servidor

☐ Variable predefinida \$_FILES

☐ Precauciones:

- ☐ Permisos de escritura en el directorio temporal
- ☐ Permisos de escritura en el directorio de destino
- ☐ Atención con los ficheros que puedan subir los usuarios
- ☐ Troyanos, scripts, ejecutables, etc.

☐ Ejercicio 4: subida de un fichero al servidor

- ☐ Ilustra cómo subir ficheros a un servidor, cómo controlar su tamaño, cómo crear un nombre único para el fichero y cómo almacenarlo en el lugar deseado.
-