

# UT3 Guiones de servidor PHP

Desarrollo en entorno servidor

# Sanear y validar formularios

- Un formulario permite al usuario introducir datos para enviar al servidor que, a su vez, puede trasladarlos a una base de datos para consultar, modificar, ...
- Un formulario es un punto débil en un sitio web y presenta vulnerabilidades ya que:
  - a) un usuario puede incluir cualquier código HTML, desde código inocuo (unas simples etiquetas de marcado en negrita) hasta código malicioso (sofisticados script Javascript que efectúan acciones peligrosas \*) - **falta de seguridad**
  - b) un usuario puede introducir datos erróneos, no se ajustan a determinados requerimientos (una edad con valor negativo, un email con caracteres incorrectos, una nota con caracteres alfabéticos y no numéricos) – **datos no válidos**

# Sanear y validar formularios

- Un formulario es, por tanto, una fuente de amenazas de seguridad y de datos erróneos
- Tipos de ataques maliciosos
  - robo y destrucción de datos
  - redireccionar usuarios a otras páginas
  - insertar links a otros sitios web
  - spam
  - cuando se insertan datos para interactuar con un BD se puede inyectar SQL

# Sanear y validar formularios

**Cross-Site Scripting (XSS)**- una amenaza de seguridad que se basa en explotar vulnerabilidades de un sitio para inyectar HTML/JavaScript no deseado.

[http://es.wikipedia.org/wiki/Cross-site\\_scripting](http://es.wikipedia.org/wiki/Cross-site_scripting)

<https://www.youtube.com/watch?v=PNjrbA-8Rj8> Busca otros ejemplos

<http://pressroom.hostalia.com/white-papers/ataque-cross-site-scripting>

**Inyección SQL (SQL injection)** - método de infiltración de código intruso dentro del código SQL programado, a fin de alterar el funcionamiento normal del programa

<http://php.net/manual/es/security.database.sql-injection.php>

# Sanear y validar formularios

- Hay que minimizar estos riesgos, por lo tanto se debe:
  - saneando los datos
  - validando los datos
- Sanear los datos
  - filtrando todas las entradas recibidas
    - con htmlspecialchars()
    - strip\_tags()
    - str\_replace()
    - eliminaremos también los espacios en blanco a ambos lados de un string recibido

# Sanear y validar formularios

- Validar los datos
  - comprobar tipos de datos (`if (!is_numeric()) ...`)
  - longitud de las cadenas (`strlen()` )
  - evitar campos vacíos (`if (empty(..))` )
  - formato de los datos (nº teléfono correcto, email correcto, ...)
  - valores lógicamente correctos (edad no negativa, sueldo no negativo, ...)
  - además de funciones PHP para validaciones complejas se utilizan **expresiones regulares**

# Sanear y validar formularios

## Ejercicios de PHP

Nombre

Apellido

Edad

Los datos de los campos correctos se mantienen en el formulario

Nombre

Apellido

Edad

El programa los reescribe

**Bienvenido/a Julia Merino, tienes 22 años**

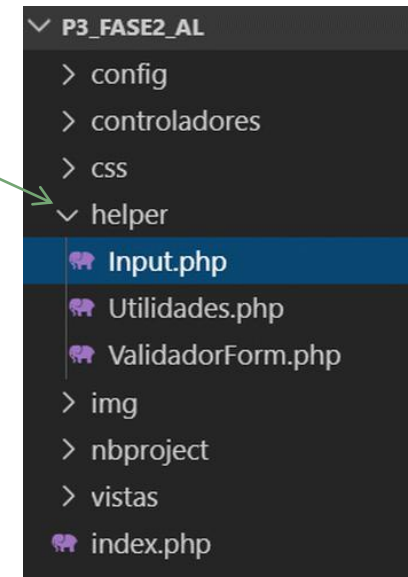
# Sanear y validar formularios

Se crea un carpeta **helper** donde iremos incluyendo las clases con los métodos que nos permitirán realizar la validación, saneamiento, filtración de los datos.

En la carpeta **helper** creamos una clase **Input** con un método estático **get(\$dato)** . El método **get** devolverá el dato si este está definido o 'sino devolverá ""

Los métodos de las clases de la carpeta **helper** serán estáticos para poder utilizarlos sin necesidad de crear un objeto de la clase.

Puedes aplicarlo en los ejercicios de las presentaciones anteriores.





# Sanear y validar formularios

```
public static function get($campo) //SI
{
    if (isset($_POST[$campo])) {
        $campo = $_POST[$campo];
    } else {
        $campo= "";
    }

    return Input::filtrar($campo);
}
```

En la clase **Input**  
creamos el método  
estático **get()**

También se debe crear  
el método **filtrar()**

Se utiliza el método  
en el formulario,  
para volver a mostrar  
el valor correcto introducido

<label>Nombre</label>

<input type="text" name="nombre" value="<?php echo Input::get('nombre') ?>" />

<br />

# Sanear y validar formularios

Revisa también con las presentaciones, el UML y el Proyecto facilitado.  
Para validar los errores seguiremos los siguientes pasos:

1.- En **Controlador.php** crearemos :

- ☐ un método **crearReglasDevalidacion()**, para crear las reglas de validación que se consideren.
- ☐ Un método **validar()** que realizará el proceso de validación utilizando un objeto de la clase y el método anterior

2.- Crearemos una clase **ValidadorForm** en la carpeta **helper**. En el caso de haber errores, estos se guardarán en un array **\$errores**.

3.- En el **formulario** comprobaremos si el array **\$errores** contiene datos (hay errores) o no, antes de mostrar el formulario y/o resultado.

# Sanear y validar formularios

## La clase ValidadorForm

Creamos la clase ValidadorForm-parte1

**ValidadorForm**

```
*-$errores: array()
*-$reglasValidacion = null
*-$valido: boolean = false

*+construct ()
*+validar ($fuente:array, $reglasValidacion:array)
Comprueba si los
campos saneados
de $fuente
cumplen las
reglas de
validacion
establecidas en
$reglasValidacion
Asigna a su
propiedad
$valido el
resultado de
comparar el n°
de elementos de
$errores con 0
```

Sanear/Filtar significa:  
eliminar caracteres  
no válidos(peligrosos),  
aplicando funciones  
tipo strip\_tags(),  
**htmlspecialchars()**  
stripslashes(), trim()...

# Sanear y validar formularios

## La clase ValidadorForm

- Creamos la clase ValidadorForm-parte2

```
*+addError($nombreCampo:string,$error:string)
Añade $error al
array $errores
en la clave
asignada $campo

*+esValido()
Devuelve el
valor de la
propiedad
$valido

*+getErrores()
Devuelve el
array $errores

*+getMensajeError($campo)
Si está definido
en el array
$errores la
clave $campo
devuelve el
valor del array
de esa clave
```

# Sanear y Validar Formularios

- 2.- En el controlador el método run() ahora debe controlar la acción que se está realizando y lo hará de la siguiente manera :

```
public function run()
{
    if (!isset($_POST['oper']))//no se ha enviado el formulario
    { // primera petición
        //se llama al método para mostrar el formulario inicial
        $this->mostrarFormulario("validar", null, null);
        exit();
    }
    if (isset($_POST['oper']) && ($_POST['oper']) == 'validar')//se ha enviado el formulario
    { //se valida el formulario
        $this->validar();
        exit();
    }if (isset($_POST['oper']) && ($_POST['oper']) == 'continuar')//se ha enviado el formulario
    {
        //Terminar
        unset($_POST); //Dejo limpio $_POST como la primera vez
        //echo 'Programa Finalizado';
        $this->mostrarFormulario("validar", null, null);
        exit();
    }
}
```

En el caso de  
que se quiera  
limpiar el formulario

# Sanear y Validar Formularios

2.- El método `mostrarFormulario` cambiará debido a la necesidad de más argumentos:

```
private function mostrarFormulario($fase, $validador, $resultado)
{
    //se muestra la vista del formulario (la plantilla form_bienvenida.php)
    include 'vistas/form_bienvenida.php';
}
```

En el controlador crearemos :

un método **`crearReglasDevalidacion()`**, para crear las reglas de validación que se consideren.

```
private function crearReglasDeValidacion()
{
    $reglasValidacion = array(
        "apellido" => array("required" => true),
        "nombre" => array("required" => true),
        "edad" => array("min" => 16, "required" => true)
    );

    return $reglasValidacion;
}
```

# Sanear y Validar Formularios

2.- En el controlador crearemos :

- ❑ Un método **validar()** que realizará el proceso de validación utilizando un objeto de la clase y el método anterior

```
private function validar()
{
    $validador = new ValidadorForm();
    $reglasValidacion = $this->crearReglasDeValidacion();
    $validador->validar($_POST, $reglasValidacion);
    if ($validador->esValido()) {
        //Formulario correcto, recoger datos y
        //volver a mostrar formulario con el resultado correcto
        //COMPLETA
        //ESTUDIA y ANALIZA los parámetros
        $this->mostrarFormulario("continuar", $validador, $resultado);
        exit();
    }

    // formulario no correcto, mostrarlo nuevamente con los errores
    $this->mostrarFormulario("validar", $validador, null);
    exit();
}
```

# Sanear y Validar Formularios

3.- La página SPA se divide en tres partes

ZONA de ERRORES

ZONA de FORMULARIO

ZONA de MENSAJES/RESULTADOS



# Sanear y Validar Formularios

3.- En la ZONA de ERRORES comprobaremos si el array **\$errores** contiene datos (hay errores) o no, antes de mostrar el formulario y/o resultado. Para ello se necesitar un objeto **\$validador**.

```
<?php
include "cabecera.php";

if (Input::siEnviado("post")) // ha habido envío //también if (isset($validador))
{

    $errores = $validador->getErrores();
    if (!empty($errores))
    {
        echo "<div class='errores'>";
        foreach ($errores as $campo => $mensajeError)
        {
            echo "<p>$mensajeError</p>\n";
        }
        echo "</div>";
    }
}

?>
<form id="form" action="index.php?" method="post">
```

# Sanear y Validar Formularios

Se debe añadir en la clase Input el método static siEnviado(\$tipo)

```
public static function siEnviado($tipo = 'post')
{
    switch ($tipo) {
        case 'post':
            return !empty($_POST);
            break;
        default:
            return false;
            break;
    }
}
```

Se puede  
incluir el case 'get'

# Sanear y Validar Formularios

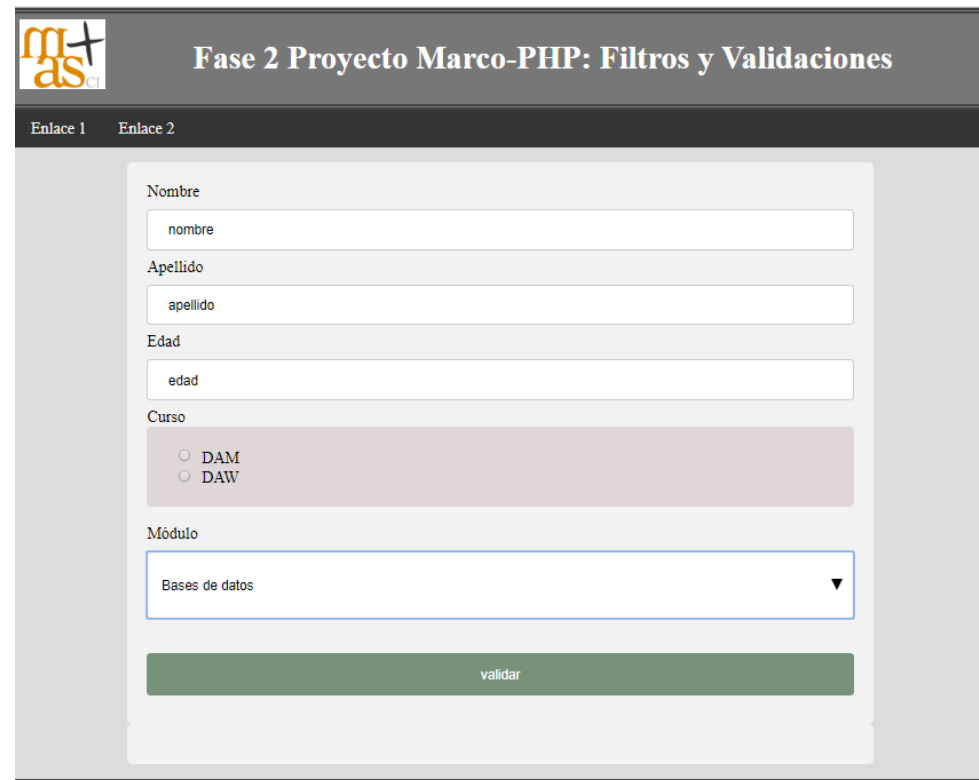
- Se debe tener definidas las clases que se utilicen.
- Se puede hacer la lista de archivos include al comienzo del script o definir una función `__autoload()`:
- Mostrar config.php de PEV2\_1

<http://php.net/manual/es/language.oop5.autoload.php>

# Sanear y Validar Formularios

Haz una copia del Proyecto y nombra al proyecto P3\_Fase2\_1, podéis, si así lo consideráis, completar este proyecto como práctica, antes de pasar a vuestro proyecto, con las indicaciones siguientes:

Para practicar con otros tipos de controles añade al formulario un botón de opción con el curso DAW o DAM y una lista desplegable con nombres de módulos.



The screenshot shows a web form titled "Fase 2 Proyecto Marco-PHP: Filtros y Validaciones". The form is divided into two sections: "Enlace 1" and "Enlace 2". The "Enlace 1" section contains the following fields:

- Nombre: A text input field with the placeholder text "nombre".
- Apellido: A text input field with the placeholder text "apellido".
- Edad: A text input field with the placeholder text "edad".
- Curso: A section containing two radio buttons labeled "DAM" and "DAW".
- Módulo: A dropdown menu with the selected option "Bases de datos".

At the bottom of the form is a green button labeled "validar".

# Formularios

## Recuperar valores introducidos

- El menú de los módulos en un array: (select envía siempre, al menos el primer valor)

```
<label>Módulo</label>
<select name="modulo" >
    <?php
        $modulos = array("Bases de datos", "PHP", "Lenguajes de marcas", "Programación");
        foreach ($modulos as $m)
        {
            echo "<option value='" . $m . "'";
            echo Utilidades::verificarLista(Input::get('modulo'), $m);
            if (isset($modulo))
            {
                verificarLista($m, $modulo);
            }
            echo " >$m</option>\n";
        }
    ?>
</select><br />
```

Se crea en un archivo (**Utilidades/Input**) en la carpeta **helper** el método static **verificarlista (\$e,\$l)** en dicha clase

# Formularios

## Recuperar valores introducidos

- Para volver a escribir el elemento ya elegido, debemos buscar si el **\$valor del menú** que se está revisando coincide con el **\$valor** introducido() y si esto es así, se **marca** :

```
public static function verificarLista($valor, $valormenu)
{
    if ($valor== $valormenu) {
        echo '    selected = "selected"';
    }
}
```

Se crea en la carpeta **helper** , en la clase **(Utilidades o Input)** el método **static verificarlista (\$e,\$l)**

# Formularios

## Recuperar valores introducidos

Estas funciones estáticas lo que hacen es poner el atributo 'selected' o 'checked' dependiendo del control, lista, botón...y de esa manera aparece el valor elegido marcado.

Se indican nombres descriptivos *verificarLista()*, *verificarBotones()*, *verificar Casillas()*, .....

```
public static function verificarLista($valor, $valormenu)
{
    if ($valor== $valormenu) {
        echo '    selected = "selected"';
    }
}
```

Es una lista,  
luego selected

# Formularios

# Recuperar valores introducidos

Se pueden añadir otros controles (completa):

```

<label>Curso</label>

<input type="radio" name="curso" value = "DAM"
    <?php echo Utilidades::verificarBotones(Input::get('curso'), "DAM") ?>/>DAM
<input type="radio" name="curso" value = "DAW"
    <?php echo Utilidades::verificarBotones(Input::get('curso'), "DAW") ?> />DAW
<br />
<label>&nbsp;</label>
<input type="submit" name="oper" value="<?php echo $fase ?>" />

```



# Formularios

## Recuperar valores introducidos

- Lo mismo para los botones, se marca el elegido. La misma idea.

```
public static function verificarBotones($valor, $valorbot)
{
    if ($valor== $valorbot)
    {
        echo 'checked = "checked"';
    }
}
```

En el mismo archivo  
en el que se encuentra  
**verificarlista (\$e,\$l)**  
se incluye  
**verificarbotones (\$e,\$l)**

## ATENCIÓN con los controles ARRAYS

# Formularios

## Recuperar valores introducidos

Se pueden añadir otros controles (En este caso la nota):

El valor apellido es requerido

El valor nombre es requerido

El valor edad es requerido

El valor curso es requerido

Nombre

Apellido

Edad

Módulo

PHP

Nota

7

Curso

☐ DAM ☐ DAW

validar

# Formularios

## Recuperar valores introducidos

y otras reglas de validación, el curso requeriso (id aplicando a vuestro proyecto):

El valor apellido es requerido

El valor nombre es requerido

La edad es menor de 18

Nombre	<input type="text"/>
Apellido	<input type="text"/>
Edad	<input type="text" value="14"/>
Módulo	<input type="text" value="PHP"/>
Nota	<input type="text" value="7"/>
Curso	<input type="radio"/> DAM <input checked="" type="radio"/> DAW
<input type="button" value="validar"/>	

# Formularios

## Recuperar valores introducidos

Nombre	<input type="text" value="Ane"/>
Apellido	<input type="text" value="Elizalde"/>
Edad	<input type="text" value="22"/>
Módulo	<input type="text" value="Lenguajes de marcas"/>
Nota	<input type="text" value="7"/>
Curso	<input type="radio"/> DAM <input checked="" type="radio"/> DAW
<input type="button" value="continuar"/>	

**Bienvenido/a Ane Elizalde, tienes 22 años  
estás en el curso DAW.  
Tu nota es: 7 en el módulo: Lenguajes de marcas**