Recogida y Verificación de Datos de un Formulario

PHP

Acceso a formularios HTML desde PHP

- Los Formularios, no forman parte del lenguaje PHP sino que son parte de la página del cliente, y están realizados con el lenguaje HTML.
- Pero sirven para que enviemos, junto con la solicitud de la página, valores que aporte el usuario escribiéndolos por el teclado, enviando parejas de datos en modo “variable- valor”.
- El script se ejecuta en el servidor cuando un navegador solicita una página.

- Desde PHP se puede acceder a los datos de un formulario HTML.
- Fichero **uno.html:**

```
<html><body>
<form action="dos.php" method="POST">
   Edad: <input type="text" name="edad">
<input type="submit" value="aceptar">
</form>
</body></html>
```

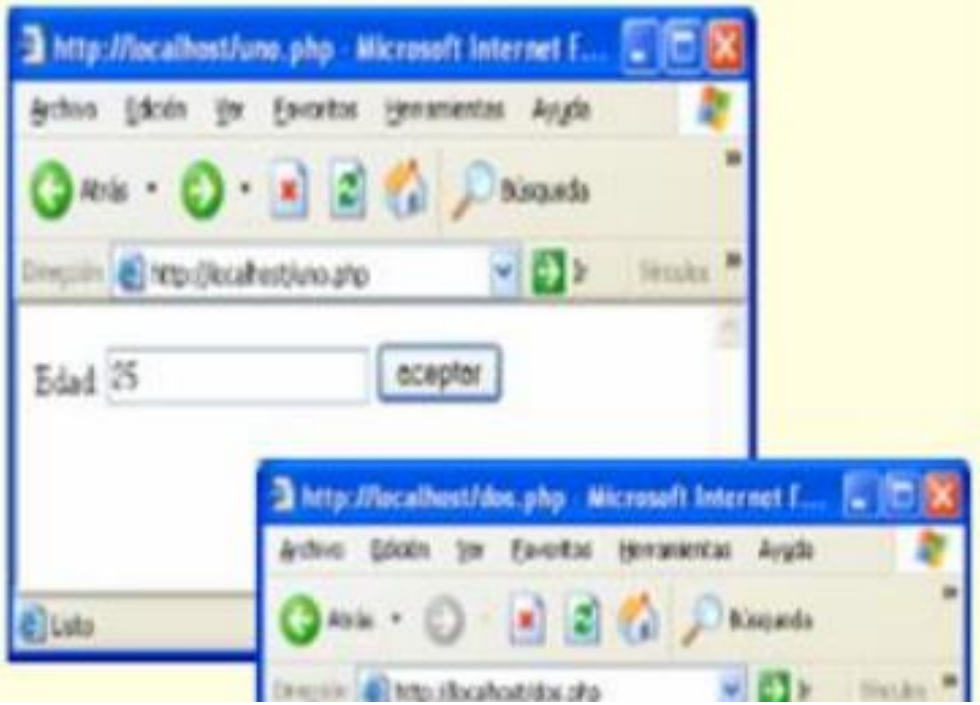
Página a la que se llamará al hacer click en submit

Estos valores deben ser iguales. Si no no podremos recuperar la información.

- Fichero **dos.php**
  - register_globals=off

```
<?php
echo "La edad es:". $_REQUEST['edad'];
?>
```

Podría ser $_POST



Las matrices $_POST, $_GET y $_REQUEST

- Una vez que estamos en el servidor, los datos son pasados del cliente al servidor usando las variables super-globales o matrices $_POST $_GET, $_REQUEST.
- Su uso, dependerá del modo en el que pasemos los datos del formularios desde el cliente.
- Como hemos visto cada control crea un elemento de la matriz $_REQUEST, que se identifica como:

  - $_REQUEST [‘valor_del_atributo_name’] : Contiene el valor entregado por el formulario .

- Para mostrar el contenido de cualquiera de estas matrices, se utiliza como en cualquier matriz print_r().

Referencia a las matrices $_POST, $_GET y $_REQUEST dentro y fuera de cadenas

- Si se hace referencia **dentro de una cadena**, el índice se debe escribir **sin comillas**.
  - Si se escribe cualquier tipo de comillas (simples o dobles) se produce un error.
- Si se hace referencia **fuera de una cadena**, el índice se debe escribir **con comillas** dobles o simples. Si se escribe sin comillas, se produce un aviso.

- **Formulario:**

```
<form action="respuesta.php?valor=10" method="post">
<input type="text" name="nombre">
</form>
```

- **PHP (¿todos correctos?):**

```
<?
echo 'valor: '.$_GET['valor'].'<br>';
echo 'valor: '.$_POST['valor'].'<br>';
echo 'valor: '.$_REQUEST['valor'].'<br>';
echo 'nombre: '.$_GET['nombre'].'<br>';
echo 'nombre: '.$_POST['nombre'].'<br>';
echo 'nombre: '.$_REQUEST['nombre'].'<br>';
?>
```

- **Formulario:**

```
<form action="respuesta.php?valor=10" method="post">
<input type="text" name="nombre">
</form>
```

- **PHP (¿todos correctos?):**

```
<?
echo 'valor: '.$_GET['valor'].'<br>';
echo 'valor: '.$_POST['valor'].'<br>';→ Vacío
echo 'valor: '.$_REQUEST['valor'].'<br>';
echo 'nombre: '.$_GET['nombre'].'<br>';→ Vacío
echo 'nombre: '.$_POST['nombre'].'<br>';
echo 'nombre: '.$_REQUEST['nombre'].'<br>';
?>
```

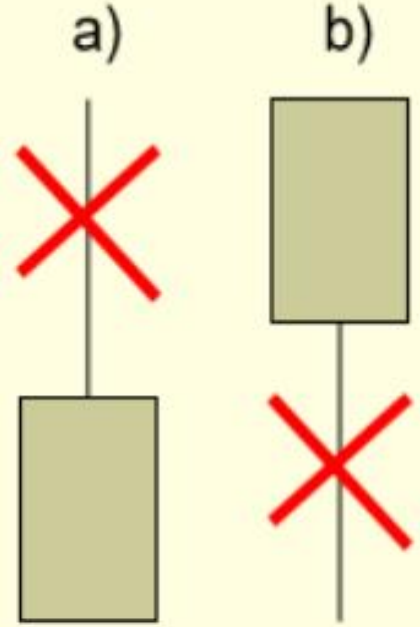
- Procesar formularios
  - La forma habitual de trabajar con formularios en PHP es utilizar un único programa que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
  - Ventajas:
    - Disminuye el número de ficheros
    - Permite validar los datos del formulario en el propio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

```
si se ha enviado el formulario:
   Procesar formulario
si no:
   Mostrar formulario
fsi
```

Esquema de funcionamiento

```
si se ha enviado el formulario:
   Procesar formulario
```

```
si no:
   Mostrar formulario
fsi
```



- La 1ª vez que se carga la página se muestra el formulario (a)
- La 2ª vez se procesa el formulario (b)

Procesar formularios en PHP

- Para saber si se ha enviado el formulario se acude a la variable correspondiente, es decir, al botón de envío.
- Supongamos un botón definido en el formulario de la siguiente forma:

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

- La condición que debemos controlar sería:

```
if($_REQUEST[‘enviar] == "procesar")
```

- o bien :

if(isset($_REQUEST[‘enviar’])) → como veremos enseguida

Una vez enviado: comprobación de existencia de los controles

- **Controles vacíos:**

➢ La primera situación que los programas deben tener en cuenta es que los controles no contengan ningún valor.

➢Este problema se puede resolver incluyendo una estructura if ... else que considere la posibilidad de que no se haya escrito nada en el formulario.

➢Ya que el hecho de que aparezca una variable no implica que la variable exista, esto en php puede crear una confusión por su naturaleza dinámica.

Si en el código aparece esta línea:

```
$edad;
```

Pregunta

¿A partir de esa línea la variable existe?

- La respuesta es que existe pero tiene un valor null

¿A partir de esa línea la variable ha sido declarada?

- En realidad la respuesta es si, como null

¿Tiene algún valor?

- La respuesta es que tiene el valor null

Comprobación de existencia

- Las comprobaciones que debo hacer son las siguientes:


```php
<?php
print "<pre>"; print_r($_REQUEST); print "</pre>\n";

if ($_REQUEST["nombre"] == "") {
    print "<p>No ha escrito ningún nombre</p>";
} else {
    print "<p>Su nombre es $_REQUEST[nombre]</p>\n";
}
?>
```

```
Array
(
    [nombre] =>
)
No ha escrito ningún nombre
```

Comprobación de existencia

- **Controles inexistentes: isset().**

➢ Un problema más grave es que el programa suponga que existe un control que en realidad no le ha llegado.

➢ Eso puede ocurrir, por ejemplo, en el caso de las casillas de verificación y los botones radio, ya que los formularios envían el control **solamente** cuando se han marcado.

Deseo recibir información: ☑

Enviar

```php
<?php
print "<pre>"; print_r($_REQUEST); print "</pre>\n";

if ($_REQUEST["acepto"] == "on") {
    print "<p>Desea recibir información</p>\n";
} else {
    print "<p>No desea recibir información</p>\n";
}
?>
```

```
Array
(
    [acepto] => on
)
Desea recibir información
```

Deseo recibir información: ☐

Enviar

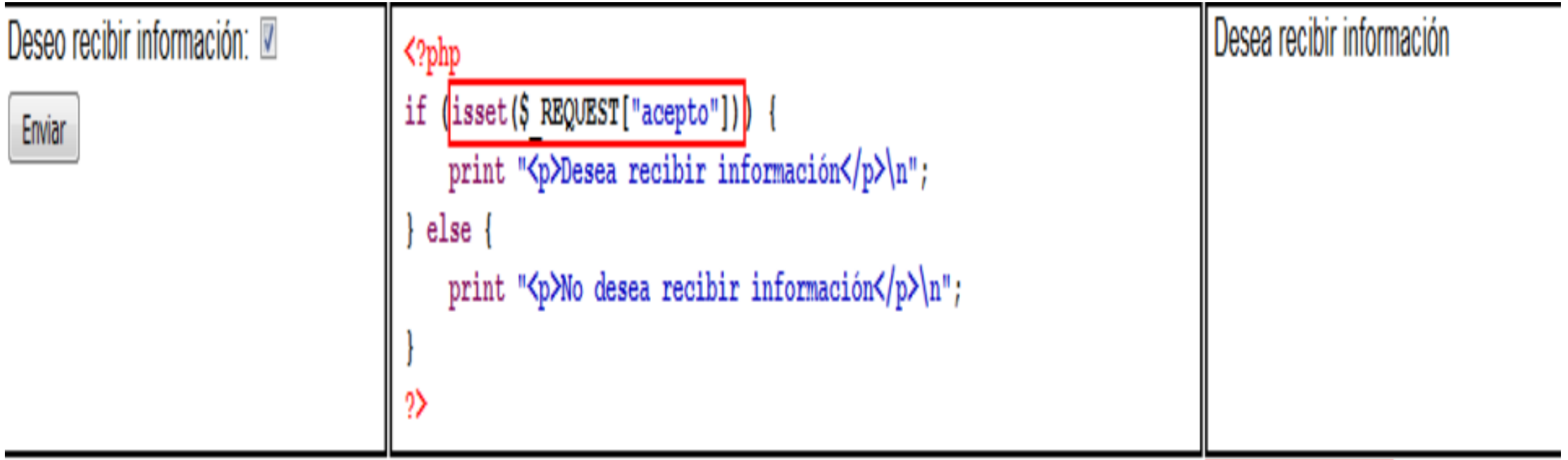
```php
<?php
print "<pre>"; print_r($_REQUEST); print "</pre>\n";

if ($_REQUEST["acepto"] == "on") {
    print "<p>Desea recibir información</p>\n";
} else {
    print "<p>No desea recibir información</p>\n";
}
?>
```

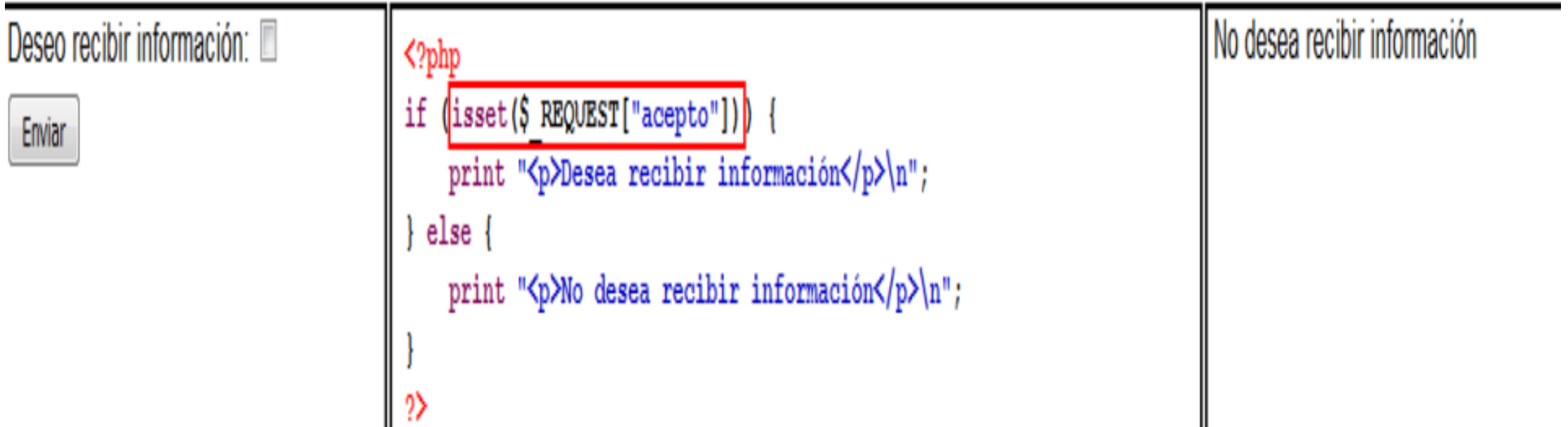
```
Array
(
)
Notice: Undefined index: acepto in ejemplo.php on line 4
No desea recibir información
```

Comprobación de existencia

- Estos problemas se pueden resolver comprobando que el índice está definido antes de hacer referencia a él, para ello se utiliza la función **isset($variable**), esta función admite como argumento una variable devolviendo **true si existe y false si no existe**.
- Por tanto **determina** si una **variable está declarada y su valor interno no sea** desconocido **(NULL).**



ejemplo.html ejemplo.php http:// ... /ejemplo.php



Ejemplo

- Realiza en un solo script de php un formulario donde se envíe un valor introducido en una caja de texto.
- Desde el servidor se debe informar si no ha insertado ningún valor que lo inserte, y si lo ha insertado lo visualizarás:

hemos recogido tu valor y el valor que has insertado es: -Elena-

Debes de aportar un valor válido

```
 if (isset($_POST['enviar'])) {
            $valor = $_POST['valor'];
            if ($valor!='')
                        echo " hemos recogido tu valor y el valor que has insertado es: -$valor-<br />";
            else
                        echo "Debes de aportar un valor válido<br />";
          }
 else {
?>
  <!doctype html>
 <html lang="en">
 <head>
    <meta charset="UTF-8">
    <title>un solo fichero</title>
 </head>
 <body>
     <form action="ejemploformulario-1.php" method="POST">
         <input type="text" name="valor" id="">
         <input type="submit" name="enviar" value="Enviar">
    </form>
</body>
</html>
<?php
;}
?>
```

VALIDAR DATOS

- Consiste en comprobar que los datos recogidos y saneados son los esperados.
- Esto se puede hacer mediante:
  - Validación con expresiones regulares
  - Especificar la codificación de caracteres
  - La extensión de filtrado de PHP, estas son un conjunto de funciones booleanas que devuelven si el argumento es o no de un tipo de datos determinado.

Funciones filter

- La función filter, filtra una variable con el filtro que se indique
- filter_var($valor[,$filtro[,opciones]]) devuelve los datos filtrados o false si el filtro falla.

Filtro	Tipo de datos
FILTER_VALIDATE_INT	entero
FILTER_VALIDATE_BOOLEAN	booleano
FILTER_VALIDATE_FLOAT	float
FILTER_VALIDATE_REGEXP	expresión regular
FILTER_VALIDATE_URL	URL
FILTER_VALIDATE_EMAIL	dirección de correo
FILTER_VALIDATE_IP	IP

FUNCIONES IS_

	Función	Tipo de datos	alias (funciones equivalentes)
existencia	isset($valor)	definida	
	is null($valor)	null	
números	is bool($valor)	booleano	
	is numeric($valor)	número (puede tener signo, parte decimal y estar expresado en notación decimal, exponencial o hexadecimal).	
	is int($valor)	entero	is integer($valor), is long($valor)
	is float($valor)	float	is double($valor), is real($valor)
cadenas	is string($valor)	cadena	

Funciones ctype_

- Las funciones ctype_ son un conjunto de funciones booleanas que devuelven si todos los caracteres de una cadena son de un tipo determinado, de acuerdo con el juego de caracteres local. Estas funciones son las mismas que las que proporciona la biblioteca estándar de C ctype.

Función	Tipo de datos
ctype_alnum($valor)	alfanuméricos
ctype_alpha($valor)	alfabéticos (mayúsculas o minúsculas, con acentos, ñ, ç, etc)
ctype_cntrl($valor)	caracteres de control (salto de línea, tabulador, etc)
ctype_digit($valor)	dígitos
ctype_graph($valor)	caracteres imprimibles (excepto espacios)
ctype_lower($valor)	minúsculas
ctype_print($valor)	caracteres imprimibles
ctype_punct($valor)	signos de puntuación (caracteres imprimibles que no son alfanuméricos ni espacios en blanco)
ctype_space($valor)	espacios en blanco (espacios, tabuladores, saltos de línea, etc)
ctype_upper($valor)	mayúsculas
ctype_xdigit($valor)	dígitos hexadecimales

isset(), is_null() y empty(): Diferencias y ejemplos de su uso

- **is_null($var)**

devolverá true cuándo la variable comprobada tenga un valor NULL. Lo que puede ocurrir cuándo la variable no haya sido definida, cuándo esté definida pero sin valor asignado.

isset(), is_null() y empty(): Diferencias y ejemplos de su uso

```php
//La variable no está definida
// Devuelve true y lanza un aviso tipo Notice: undefined variable
var_dump( is_null($var) );

//Se asigna la constante NULL
$var = NULL;
var_dump( is_null($var) );

//Se declara la variable pero no se le asigna ningún valor
$var;
var_dump( is_null($var) );

//La variable es destruida con unset()
$var = "Hola";
unset($var);
var_dump( is_null($var) );
```

isset(), is_null() y empty(): Diferencias y ejemplos de su uso

isset($var)

Devuelve true si una variable ha sido declarada y su valor no es NULO.

```php
//isset() devuelve false por que $var no ha sido definida
var_dump( isset($var) );

//isset() devuelve false aunque la variable haya sido declarada pues su valor es NULL
$var;
var_dump( isset($var) );

//isset() devuelve true. El valor ya no es nulo aunque esté vacío
$var = "";
var_dump( isset($var) );
```

isset(), is_null() y empty(): Diferencias y ejemplos de su uso

empty($var)

Devuelve true si a si una variable está vacía

Esto ocurre si no existe, o si su valor es false, El valor false para una variable puede ser un valor lógico false, una cadena vacía, el número 0 (cero), un array vacío, el valor NULL y para el string "0"

```php
//La variable no está definida. empty() devuelve true
var_dump( empty($var) );

//empty() devuelve true. El valor de $var es nulo
$var;
var_dump( empty($var) );

//empty() devuelve true para un string vacío
$var = "";
var_dump( empty($var) );

//empty() devuelve true para un array vacío
$var = array();
var_dump( empty($var) );

//empty() devuelve true para el string "0"
$var = "0";
var_dump( empty($var) );

//empty() devuelve true para el número entero 0
$var = 0;
var_dump( empty($var) );

//empty() devuelve true para el valor lógico false
$var = false;
var_dump( empty($var) );
```