

# Desarrollo Web en Entorno Servidor

---

Programación en PHP

José A. Lara

# Formularios y PHP

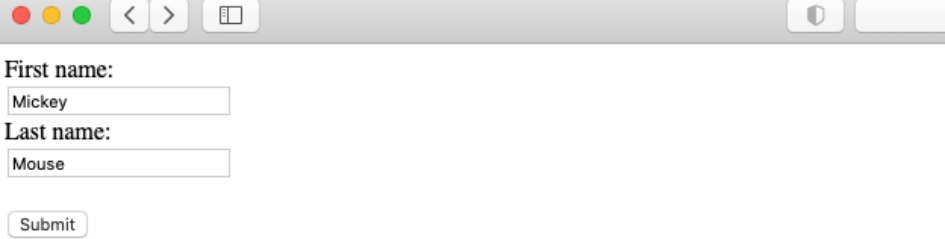
---

- El lenguaje PHP está inevitablemente unido al lenguaje HTML, hemos visto y utilizado el PHP para poder modificar el código final HTML, mediante estructuras de decisión o de repetición.
- Llega el momento de interactuar con el usuario, y para ello utilizaremos los formularios en concreto la etiqueta `<form>` justamente para recibir esta información.



# Formularios y PHP

- Resumen de formularios en HTML



A screenshot of a web browser window. The browser has a single tab and a standard address bar. The page content is a simple form. It starts with the label "First name:" followed by a text input field containing the text "Mickey". Below this is the label "Last name:" followed by a text input field containing the text "Mouse". At the bottom of the form is a button labeled "Submit".

If you click the "Submit" button, the form-data will be sent to a page called "/action\_page.php".

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <form action="/action_page.php"> First name:<br>
      <input type="text" name="firstname" value="Mickey">
      <br>
      Last name:
      <br>
      <input type="text" name="lastname" value="Mouse">
      <br><br>
      <input type="submit" value="Submit">
    </form>
    <p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>
  </body>
</html>
```

# Formularios y PHP

---

- De las etiquetas más importantes nos encontramos con la etiqueta `<input>`, que nos va a permitir definir los diferentes elementos de entrada y por lo tanto de interacción con el usuario. Dependiendo del atributo `type`, el elemento tendrá una funcionalidad u otra. Algunos ejemplos:

tipo	descripción
<code>&lt;input type="text"&gt;</code>	Define una caja de texto libre
<code>&lt;input type="radio"&gt;</code>	Define un tipo selector de tipo <i>radio button</i>
<code>&lt;input type="submit"&gt;</code>	Define un tipo botón

# Formularios y PHP

---

## Tipo text

- Este tipo define una caja de texto, alfanumérica de entrada libre. Esto es importante ya que después en el momento de recoger la información introducida por parte del usuario, tendremos que tener en cuenta que tiene esas dos características:
  - Es libre, y por lo tanto deberemos realizar una comprobación de errores y/o de datos.
  - Es alfanumérica y por lo tanto deberemos realizar conversiones. Tenemos la suerte de que el lenguaje php se encarga de forma automática de esto.



# Formularios y PHP

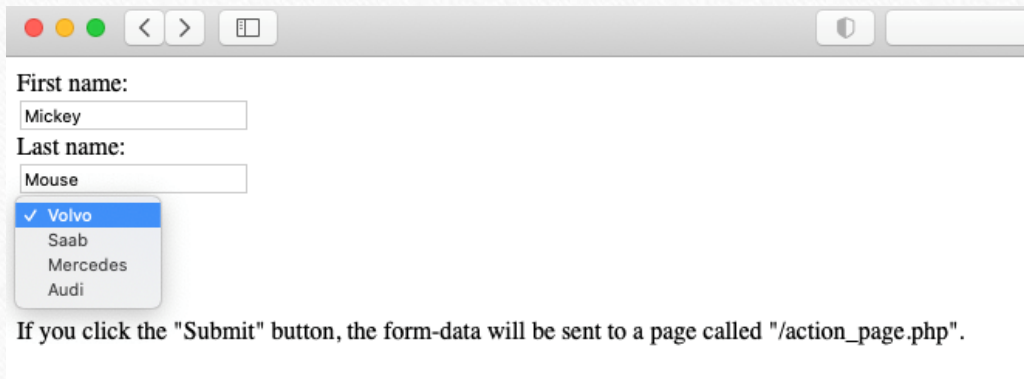
---

## Tipo text

- En segundo lugar debemos fijarnos en un atributo MUY IMPORTANTE para nuestras necesidades de recolección de información desde la parte de servidor. El atributo name. El atributo name va a permitir identificar una caja de texto con un identificador que debe ser ÚNICO, y que después utilizaremos dicho identificador para recoger dicha información.
- Cuando liguemos esta parte con la recogida de información en PHP veremos de la importancia de este atributo.

# Formularios y PHP

## Tipo select



First name:  
Mickey

Last name:  
Mouse

Volvo  
Saab  
Mercedes  
Audi

If you click the "Submit" button, the form-data will be sent to a page called "/action\_page.php".

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <form action="/action_page.php"> First name:<br>
      <input type="text" name="firstname" value="Mickey">
      <br>
      Last name:
      <br>
      <input type="text" name="lastname" value="Mouse">
      <br>
      <select name="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
      </select>
      <br><br>
      <input type="submit" value="Submit">
    </form>
    <p>If you click the "Submit" button, the form-data will be
    sent to a page called "/action_page.php".</p>
  </body>
</html>
```

# Formularios y PHP

---

## Tipo select

Nuevamente toca analizar el código que hemos utilizado:

- En primer lugar tenemos el elemento `<select>` que define una estructura desplegable de datos
- Para poder introducir cada uno de los datos que aparecerán en el desplegable utilizaremos las etiquetas `<option>`
- Dentro de la etiqueta `<option>` tendremos el atributo `value` que almacenará el valor que después utilizaremos o recogeremos desde PHP.



# Formularios y PHP

---

## Tipo select

- Otro de los aspectos importantes que vemos en el anterior ejemplo es la diferenciación entre la información presentada, Volvo y el valor de dicha option “volvo”. Podría poner, por ejemplo, un valor numérico.

# Formularios y PHP

## Tipo submit

- Otro de los grandes y necesarios elementos, al menos desde el punto de vista de nuestras necesidades dentro de la programación de backend.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <form action="/action_page.php"> First name:
      <br>
      <input type="text" name="firstname" value="Mickey">
      <br> Last name:
      <br> <input type="text" name="lastname" value="Mouse">
      <br><br>
      <input type="submit" value="Submit">
    </form>
    <p>If you click the "Submit" button, the form-data will be
    sent to a page called "/action_page.php".</p>
  </body>
</html>
```



# Formularios y PHP

---

## Otros elementos

- Tenemos la suerte de contar actualmente con muchos elementos dentro de un formulario, que podemos dividir en dos grandes apartados:
  - Elementos que mejoran la visualización de la información, como la etiqueta `<label>`
  - Elementos utilizados para recoger información como por ejemplo el elemento `type=radio`

# Formularios y PHP

---

## Ejercicio

- Tras iniciarnos en la creación de los formularios vamos a generar un formulario que permita recoger la información de inscripción a nuestra web de un usuario:
  1. Generamos la estructura de ficheros dentro con una nueva carpeta y un html que denominaremos inscripción.html. Crearemos la estructura básica con html.
  2. Crearemos un nuevo formulario con la etiqueta form.
  3. Introduciremos dos campos del tipo input text para recoger el Nombre y los Apellidos
  4. Añadiremos una nueva caja seleccionable para recoger el rango de edad (15 a 25 años, 26 a 35 años, 36 a 45 años, mas de 46 años)
  5. Incluiremos un botón de submit para enviar la información.



# Métodos HTTP

---

Llega el momento de unir los formularios HTML a nuestro código PHP en servidor. En todo lenguaje de programación existe la forma y los mecanismos de interactuar con el usuario:

- Recogiendo información y/o estados del usuario, esto lo realizaremos a través de los formularios vistos en el apartado anterior.
- Realizando una acción a partir de la información recibida.

# Protocolo HTTP y modelo cliente-servidor

---

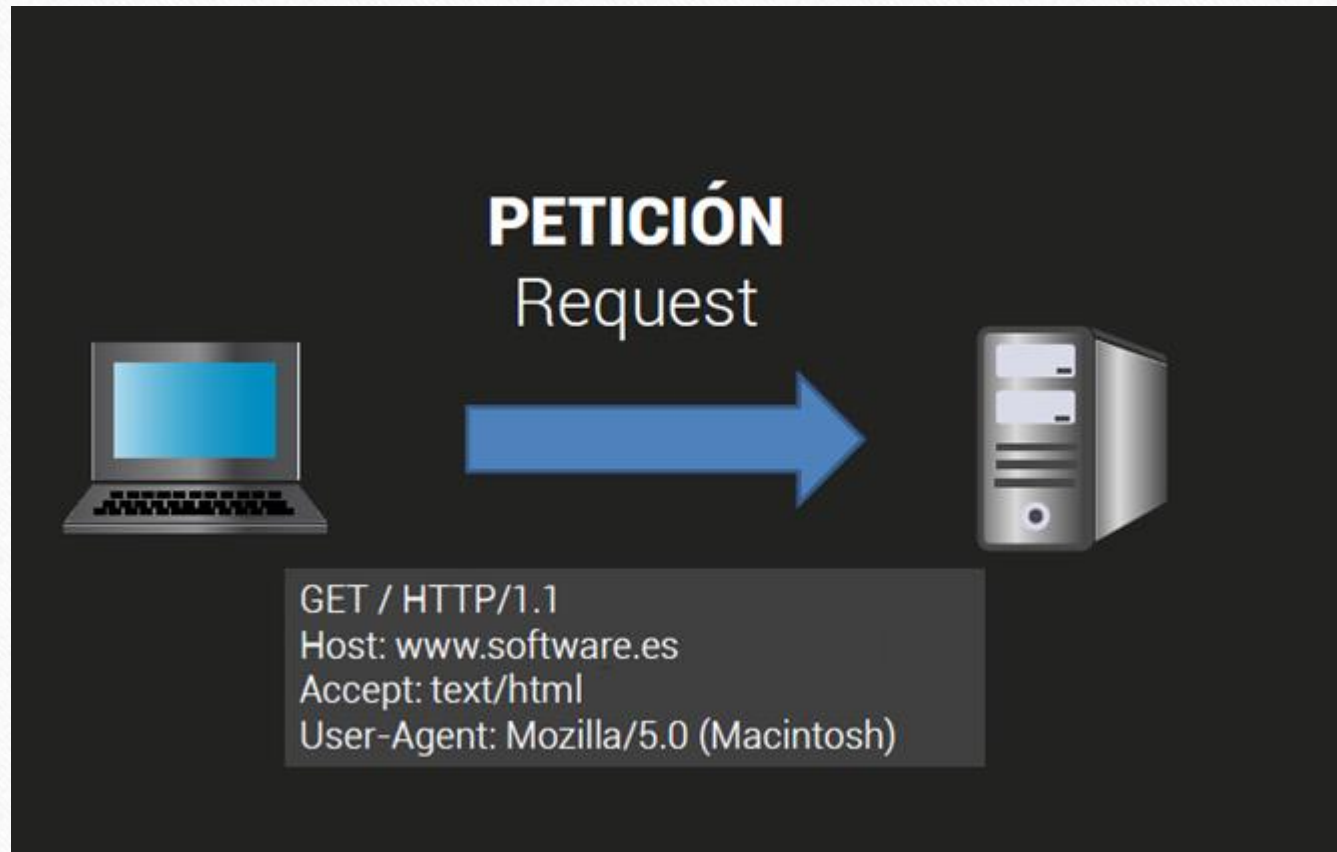
El protocolo HTTP permite la comunicación entre clientes y servidores. En esta comunicación se puede intercambiar información, de hecho el mecanismo habitual al cual el estándar WWW funciona es en el que el cliente realiza una petición mediante una URL y el servidor contesta con datos que habitualmente es un código HTML que el navegador sabe interpretar.



# Protocolo HTTP y modelo cliente-servidor

## HTTP Request

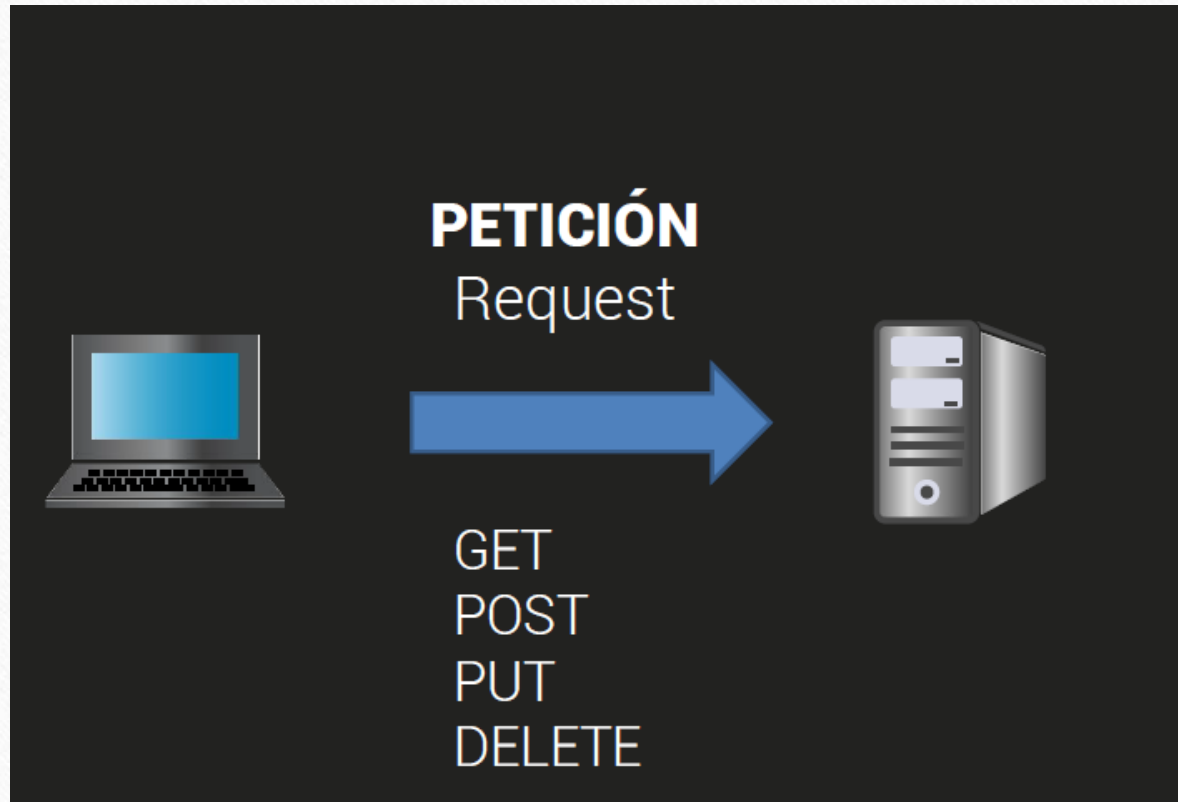
Una petición por parte de un cliente, un navegador, una aplicación, un dispositivo, ... utilizará el protocolo HTTP tal y como hemos visto.



# Protocolo HTTP y modelo cliente-servidor

---

## Métodos dentro de HTTP





# Protocolo HTTP y modelo cliente-servidor

---

## Formularios. Método GET y POST

En el momento en el que utilizamos formularios, el usuario no sólo solicita datos al servidor sino que envía información mediante una petición que puede ser GET o POST:

- El método GET, habitualmente utilizamos este método en todas las peticiones, solicitamos información o un/os recurso/s al servidor
- El método POST sirve para enviar datos al servidor

# Protocolo HTTP y modelo cliente-servidor

---

1. Tenemos el anterior formulario. El usuario rellenará la información de los dos campos propuestos “firstname” y “lastname”.
2. El usuario pulsa el botón y el formulario reacciona enviando la información a la página que aparece en el atributo action, en este caso la página action\_page.php
3. Dependiendo del valor method, la información se enviará mediante el método get o post

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <form action="/action_page.php"> First name:
      <br>
      <input type="text" name="firstname" value="Mickey">
      <br> Last name:
      <br> <input type="text" name="lastname" value="Mouse">
      <br><br>
      <input type="submit" value="Submit">
    </form>
    <p>If you click the "Submit" button, the form-data will be
    sent to a page called "/action_page.php".</p>
  </body>
</html>
```



# Protocolo HTTP y modelo cliente-servidor

---

## Método GET

La información del formulario viajará a través de la URL en el navegador. Veremos entonces que la dirección de nuestra URL cambia a

`action_page.php?firstname=value1&lastname=value2`

Como se observa todos los campos del formulario aparecen en dicha URL, comienzan los valores a partir del símbolo “?”, cada dupla viene representada por nombre=valor, y además las duplas se separan con el símbolo &.

# Protocolo HTTP y modelo cliente-servidor

---

## Método POST

En este caso la información no viaja a través de la URL, sino a través del cuerpo de la petición HTTP:

### Cuerpo de la petición

```
POST action_page.php HTTP/1.1  
Host: w3schools.com  
firstname=value1&lastname=value2
```

# Protocolo HTTP y modelo cliente-servidor

---

## **Método POST**

Como se puede observar, los parámetros recogidos se encapsulan dentro de la cabecera HTTP, y por lo tanto no tendremos las limitaciones de tamaño que en el caso del método GET teníamos, sin embargo también perdemos otras ventajas como la de poder utilizar en llamadas con la etiqueta `<a>` mediante el método GET.



# Protocolo HTTP y modelo cliente-servidor

---

## GET vs POST

	GET	POST
Historial	Los parámetros se guardan en el historial del buscador (browser) porque son parte del enlace (URL)	No se guardan los parámetros en el historial del buscador
Archivado (bookmark)	Puede ser archivado	No puede ser archivado

# Protocolo HTTP y modelo cliente-servidor

---

## GET vs POST

	GET	POST
Parámetros	Puede enviarlos pero los datos de parámetros está limitado a lo que cabe en el URL. Es más seguro usar menos de 2K de los parámetros (algunos servidores llegan a 64K)	Puede enviar parámetros, incluyendo subir archivos al servidor
Vulnerabilidad	Más fácil de atacar	Más difícil de atacar

# Protocolo HTTP y modelo cliente-servidor

---

## GET vs POST

	GET	POST
Restricciones en el tipo de datos en el formulario	Sí, sólo se permiten caracteres ASCII	No tiene restricciones. Se permiten datos binarios.
Restricciones en la cantidad de datos en formularios	Sí porque la data del formulario está en el URL y hay un límite para URLs de 2,048 caracteres, aunque puede variar	No tiene restricciones.



# Protocolo HTTP y modelo cliente-servidor

---

## GET vs POST

	GET	POST
Visibilidad	GET está visible para todo el mundo (en la barra de URLs del buscador) y tiene límites a la cantidad de datos que puede enviar	Las variables en POST no se ven en los URLs
Valores de variables grandes	Límite de 7,607 caracteres	Límite de 8MB

# Protocolo HTTP y modelo cliente-servidor

---

## **GET vs POST**

Utilizaremos en la gran mayoría de ocasiones el método POST, debido a que no tiene limitaciones en el tamaño de información, y tiene mayor seguridad.

Llegamos por fin al punto en el que unimos lo aprendido sobre la creación sobre formularios, sobre la utilización de métodos HTTP y recogida de dicha información mediante código PHP.

# Formularios PHP

---

## SuperGlobal `$_POST`, `$_GET`

Recoger la información enviada por un formulario HTML a través de un documento PHP es supersencillo, ya que nos proporciona un array asociativo que recogerá de forma automática todo el contenido de un formulario.

PHP nos facilita enormemente el trabajo ya que nos proporciona dos arrays asociativos denominados superglobals, en los cuales tendremos toda la información de los formularios HTML.



# Formularios PHP

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Formulario</title>
  </head>
  <body>
    <form action="formulario.php" method="post">
      Nombre:<br>
      <input type="text" name="nombre" value="Paco"><br>
      Apellidos:<br>
      <input type="text" name="apellidos" value="Gomez"><br>
      <input type="radio" name="sexo" value="h" checked> Hombre<br>
      <input type="radio" name="sexo" value="m"> Mujer<br>
      <input type="hidden" name="version" value="1.3"><br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Recogida de informacion</title>
  </head>
  <body>
    La informacion recibida es:
    <br>
    <?php
      print_r($_POST);
    ?>
    <br>
    <?php
      echo "El nombre recibido es: " . $_POST["nombre"];
    ?>
  </body>
</html>
```

# Formularios PHP

---

En el caso del uso de método GET, la recepción será exactamente igual solo que utilizaremos la superglobal `$_GET` para recoger toda la información.

Pero lo mejor es utilizar ejemplos y ejercicios para ir comprendiendo el funcionamiento de dichas superglobals.

# Formularios PHP

---

## Ejercicio

Añadiremos funcionalidad a nuestro formulario de inscripción en web:

- 1) Abriremos el documento inscripción.html
- 2) Añadiremos el método post al formulario
- 3) Añadiremos como action el fichero que crearemos en el siguiente punto inscripción.php
- 4) Dentro del mismo directorio crearemos un nuevo documento denominado inscripción.php
- 5) Mostraremos la información enviada por el formulario mediante la función `var_dump`



# Formularios PHP

---

## **Funciones de comprobación y verificación**

Cuando realizamos envío de información a través de formularios mediante cualquiera de los métodos propuestos anteriormente necesitamos de métodos de control y verificación antes de realizar cualquier acción, ya que pueden darnos errores no controlados.

Las tres funciones más utilizadas cuando realizamos comprobaciones de los parámetros enviados a través de formularios son: `isset`, `is_null` y `empty`

# Formularios PHP

- **is\_null:** Comprueba si la variable dada es NULL. Por tanto, esta función devolverá true cuándo la variable comprobada tenga un valor NULL (nulo). En PHP NULL se considera un valor especial que representa a una variable sin valor, lo que puede ocurrir cuándo la variable no haya sido definida, cuándo esté definida pero sin valor asignado, cuándo se le asigne el valor NULL o cuándo haya sido destruida con unset().

```
<?php
$var = NULL;
if (is_null($var)) {
    echo "Esta variable está definida pero su valor es nulo";
}
?>
```

← → ↻ ⓘ localhost:8080/php/getpost.php

Esta variable está definida pero su valor es nulo



# Formularios PHP

- **isset:** Determina si una variable está definida y no es NULL. Es importante fijarse bien: si una variable tiene valor nulo, aunque haya sido declarada, `isset()` devolverá false. En todos los ejemplos anteriores para `is_null()`, `isset()` devolvería false mientras que `is_null()` devolvió true, por eso se consideran funciones opuestas. También es importante fijarse que `isset()` devuelve true para variables con valores vacíos

```
<?php
$var = "HOLA";
// Esto evaluará a TRUE así que el texto se imprimirá.
if (isset($var)) {
    echo "Esta variable está definida, así que se imprimirá";
}
?>
```

← → ↻ ⓘ localhost:8080/php/getpost.php

Esta variable está definida, así que se imprimirá



# Formularios PHP

- **empty:** Determina si una variable es considerada vacía. Una variable se considera vacía si no existe o si su valor es igual a FALSE. `empty()` no genera una advertencia si la variable no existe. La definición es muy sencilla pero abarca muchos posibles escenarios. Una variable se considera vacía si no existe (aquí coincide con `isset()` y con `is_null()`) 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, aquí la peculiaridad, el string "0" (no como valor numérico, sino como cadena de texto) también dará positivo en la función `empty()`.

```
<?php
$var = 0; // Se evalúa a true ya que $var está vacía
if (empty($var)) {
    echo '$var es o bien 0, vacía, o no se encuentra definida en absoluto';
} // Se evalúa como true ya que $var está definida
if (isset($var)) { echo '$var está definida a pesar que está vacía'; }
?>
```

← → ↻ ⓘ localhost:8080/php/getpost.php  
\$var es o bien 0, vacía, o no se encuentra definida en absoluto \$var está definida a pesar que está vacía

# Formularios PHP

---

Además de las funciones antes descritas pueden resultar interesantes

- `is_bool()` - Comprueba si una variable es de tipo booleano
- `is_numeric()` - Comprueba si una variable es un número o un string numérico
- `is_float()` - Comprueba si el tipo de una variable es float
- `is_int()` - Comprueba si el tipo de una variable es integer
- `is_string()` - Comprueba si una variable es de tipo string
- `is_object()` - Comprueba si una variable es un objeto
- `is_array()` - Comprueba si una variable es un array



# Formularios PHP

---

En el caso de formularios, las funciones de comprobación más utilizadas son tanto `isset` como `empty`. Si tenemos el formulario:

```
<form action="conversion.php" method="post">
    <input type="text" name="nombre" value="">
    <br><br>
    <input type="submit" name="enviar" value="ENVIAR">
</form>
```

La combinación que deberíamos utilizar para asegurarnos que el valor nos llega correctamente en creación y contenido sería:

```
<?php
if( isset($_POST["nombre"]) && !empty($_POST["nombre"]) ) {
?>
```



# Formularios y objetos

---

Hasta este punto, hemos utilizado los formularios escritos en HTML para poder recibir información del usuario y poder realizar cálculos, sacar por pantalla la información o realizar cambios en la página visible. También hemos visto que podíamos utilizar dos métodos de comunicación por debajo del protocolo HTTP, como eran el método POST y el método GET para el envío de la información.

Es el turno de dar un paso más en nuestro conocimiento del PHP y unir la programación orientada a objetos que hemos estado viendo en anteriores lecciones para añadírsela al envío de información a través de formularios.

# Formularios y objetos

---

Pasaremos por lo tanto de utilizar dos ficheros a utilizar 3 y más ficheros que se relacionarán entre ellos para dar un resultado. Este proceso hará que nuestro desarrollo sea mucho más modular, mantenible y ampliable en un futuro.

En realidad no vamos a aprender o realizar nada nuevo, sino que vamos a integrar lo que sabíamos de clases y objetos para añadirlo a la recogida de información con formularios



# Formularios y objetos

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Formulario</title>
  </head>
  <body>
    <form action="formulario.php" method="post">
      Base:<br>
      <input type="text" name="base"><br>
      Altura:<br>
      <input type="text" name="altura"><br>
      <br>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

```
<?php
// Incluimos las librerias necesarias
include "area.php";
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Recogida de informacion</title>
  </head>
  <body>
    <?php
      $areaFormulario=new area();
      $areaFormulario->setBase($_POST["base"]);
      $areaFormulario->setAltura($_POST["altura"]);
      $areaCalculado=$areaFormulario->calculoArea();

      echo "El area del rectangulo: ".$areaCalculado;
    ?>
  </body>
</html>
```



# Formularios y objetos

```
<?php
class area
{

    //atributos privados
    private $base=0;
    private $altura=0;

    public function getBase()
    {
        return $this->base;
    }

    public function setBase($base)
    {
        $this->base = $base;
    }

    public function getAltura()
    {
        return $this->altura;
    }
}
```

```
        public function setAltura($altura)
        {
            $this->altura = $altura;
        }

        public function calculoArea(){
            return $this->altura*$this->base;
        }
    }

?>
```

# Formularios y objetos

---

Como hemos visto en este ejemplo, el uso de los objetos nos permite tener el código mucho más claro, dividido en varios documentos y poder ser reutilizables.

Expresión	gettype()	empty()	is_null()	isset()	boolean: if(\$x)
\$x = «»;	string	TRUE	FALSE	TRUE	FALSE
\$x = » «; (espacio)	string	FALSE	FALSE	TRUE	TRUE
\$x = null;	NULL	TRUE	TRUE	FALSE	FALSE
\$x = «\0»; (NULL byte)	string	FALSE	FALSE	TRUE	TRUE
var \$x;	NULL	TRUE	TRUE	FALSE	FALSE
\$x is indefinida	NULL	TRUE	TRUE	FALSE	FALSE
\$x = array(); (array vacío)	array	TRUE	FALSE	TRUE	FALSE
\$x = false;	boolean	TRUE	FALSE	TRUE	FALSE
\$x = true;	boolean	FALSE	FALSE	TRUE	TRUE
\$x = 1;	integer	FALSE	FALSE	TRUE	TRUE
\$x = 42;	integer	FALSE	FALSE	TRUE	TRUE
\$x = 0;	integer	TRUE	FALSE	TRUE	FALSE
\$x = -1;	integer	FALSE	FALSE	TRUE	TRUE
\$x = «1»;	string	FALSE	FALSE	TRUE	TRUE
\$x = «0»;	string	TRUE	FALSE	TRUE	FALSE
\$x = «-1»;	string	FALSE	FALSE	TRUE	TRUE
\$x = «Hola!»;	string	FALSE	FALSE	TRUE	TRUE
\$x = «true»;	string	FALSE	FALSE	TRUE	TRUE
\$x = «false»;	string	FALSE	FALSE	TRUE	TRUE