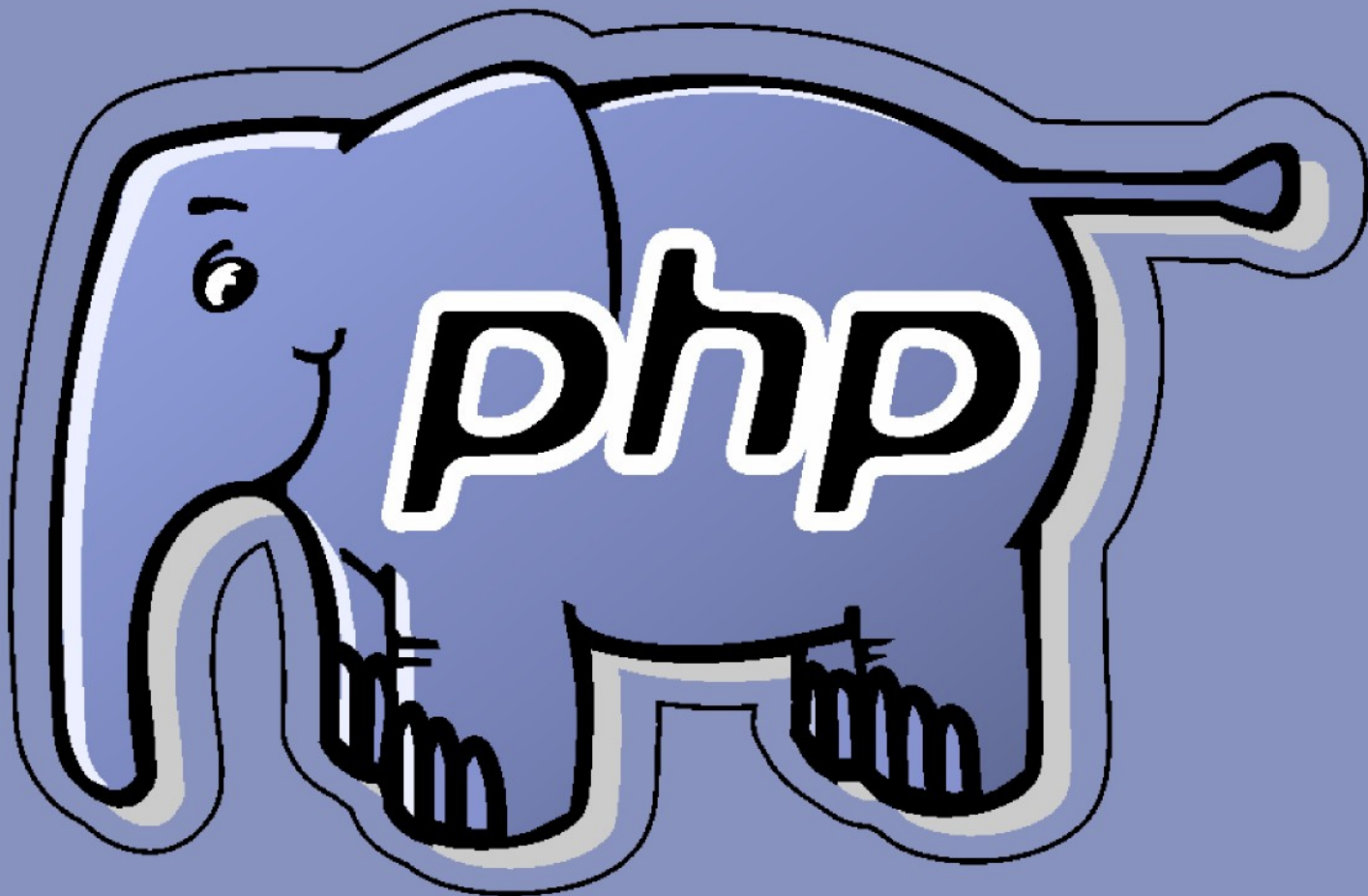




# Programación en



**Gugler**

Laboratorio de Investigación Gugler

**FCyT**

Facultad de Ciencia y Tecnología

**UADER**

Universidad Autónoma de Entre Ríos

## CONTENIDO

<b>Programación Web.....</b>	<b>5</b>
Introducción.....	5
Esquema cliente-servidor en la Web.....	5
Lenguaje HTML.....	7
Secciones de un documento HTML.....	8
Formularios.....	9
Componentes de los Formularios.....	10
Componente INPUT.....	10
Componente SELECT.....	16
Componente TEXTAREA.....	16
Métodos de envío de un Formulario.....	17
Procesando datos con PHP.....	18
Tratamiento de los componentes en PHP.....	20
Elementos INPUT.....	20
Elemento SELECT.....	21
Elemento TEXTAREA.....	22



## Capítulo 2

# Programación Web

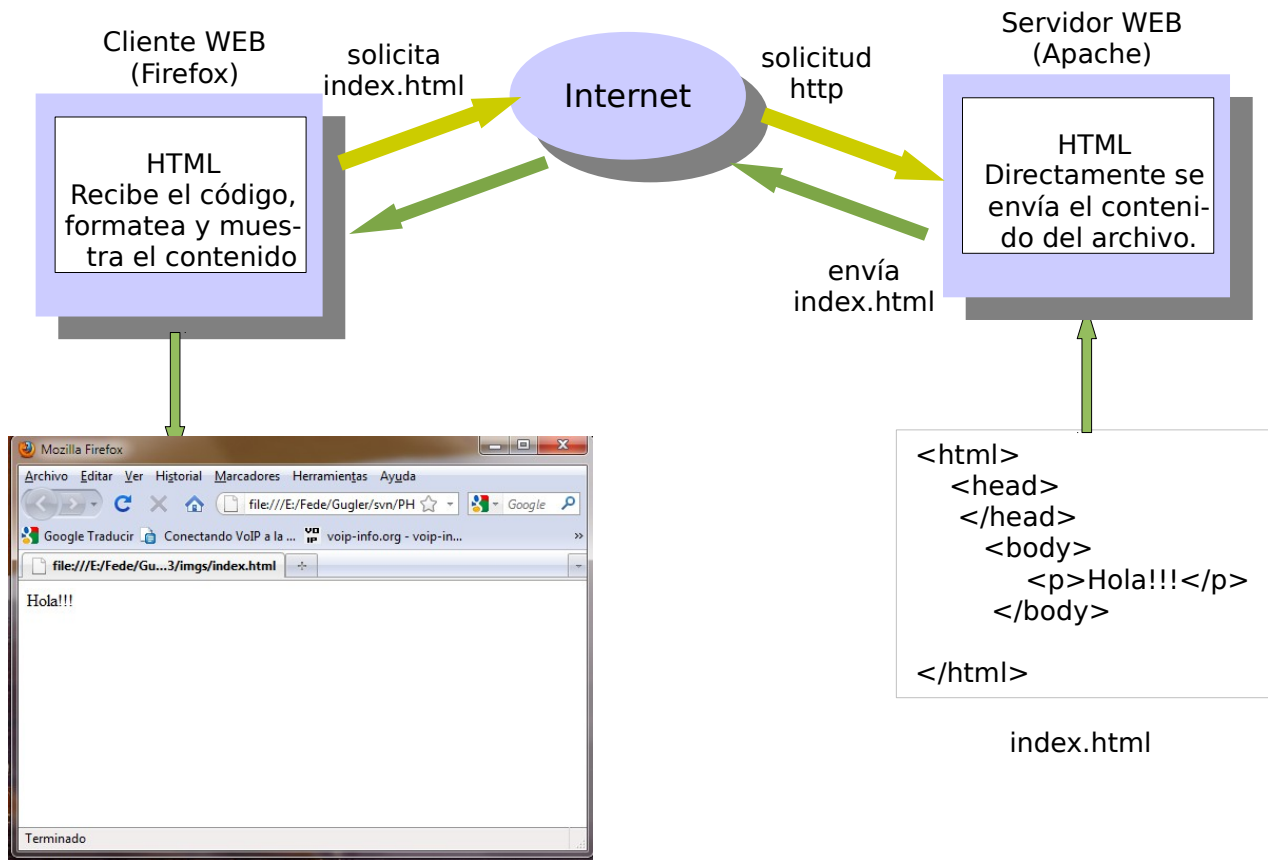
### Introducción

Si bien PHP es un lenguaje muy potente en cuanto a prestaciones (podemos hacer casi todo lo que nos imaginemos), fue diseñado en principio para el desarrollo de aplicaciones web. Es por esto que contamos con una gran cantidad de recursos que nos permitirán manipular y tratar lo que al desarrollo web involucra.

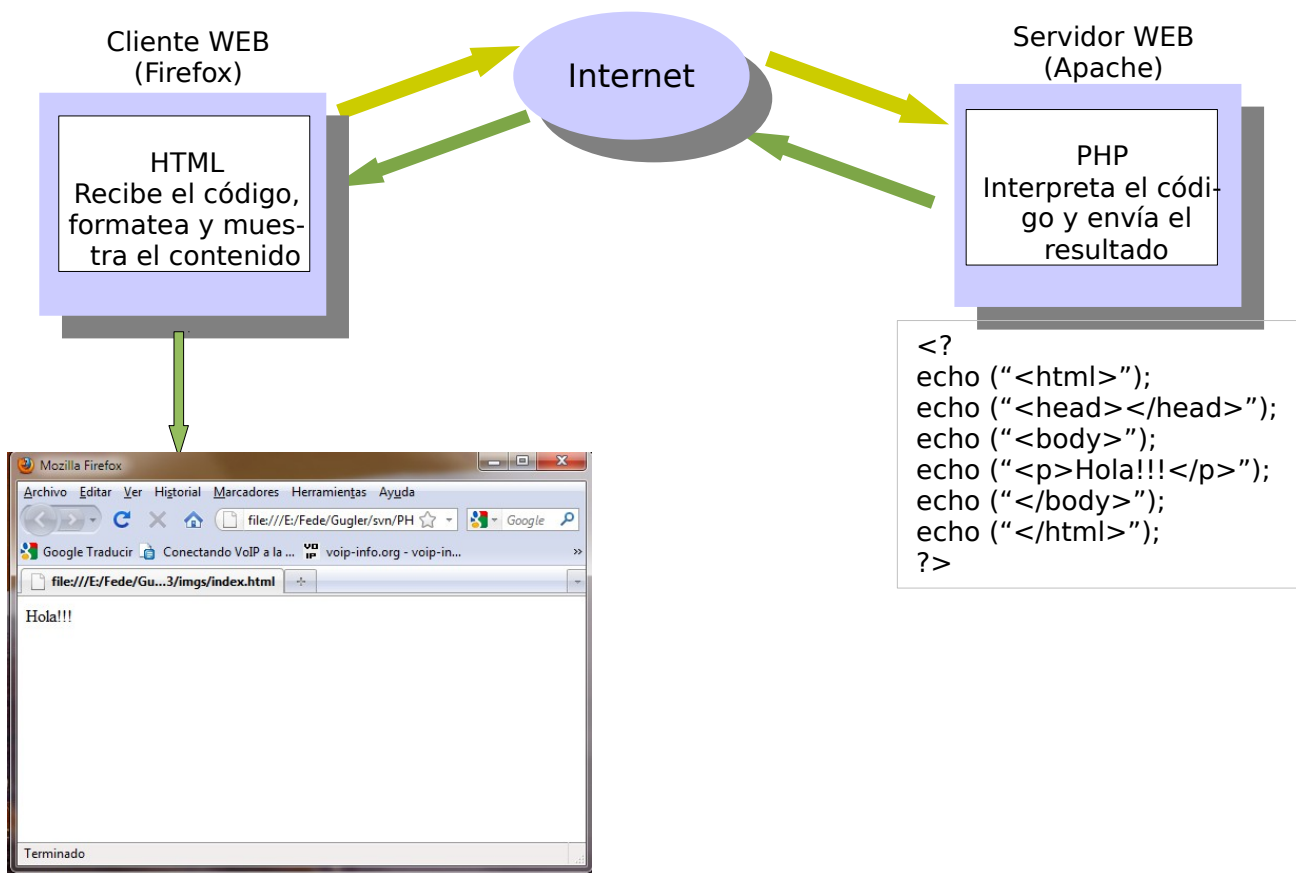
### Esquema cliente-servidor en la Web

La mayoría de las veces pensamos en una página web como un archivo ascii con código HTML dentro. Eso está bien en un principio, pero si queremos darle más dinamismo a nuestro sitio, hacer que éste gestione información ingresada por un usuario o almacenada en una base de datos, etc, deberemos hacer uso de un lenguaje de programación del lado servidor como lo es PHP.

Desde el punto de vista de un servidor la Web, la generación de un documento comienza con una petición HTTP en la que el cliente (navegador) solicita acceso a un recurso alojado en el mismo (documento html o script PHP). Si el recurso solicitado es un documento HTML completo, el servidor envía directamente el contenido de dicho archivo al navegador.



Si por el contrario, el documento tuviese fracciones de código PHP o fuese en su totalidad un script PHP, previo a enviar cualquier tipo de información al cliente, el servidor interpretará y ejecutará las acciones en el descriptas.



En ambos casos, el servidor generará unos datos de cabecera los cuáles indicarán al cliente el tipo de información le va a adjuntar (texto html, un documento pdf, de word, etc) y seguido de estos enviará el contenido obtenido de los documentos. Según el tipo de información que reciba el navegador, mostrara el contenido (html) o ejecutará algún plugin para visualizarlo (pdf, etc).

## Lenguaje HTML

Siglas correspondientes a *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto) es el lenguaje utilizado para la generación de páginas web. Mediante este podemos definir la la estructura y el contenido de las páginas webs, ademas de dar formato al texto, incluir imágenes, audio, etc en un navegador.

Todos los documentos HTML siguen una cierta estructura estandarizada ([www.w3c.org](http://www.w3c.org)) la cual divide los mismos en distintas secciones. Dentro de cada sección se incluirá información de distinta índole la que conformará la página web.

Los documentos HTML, sus distintas secciones y los componentes que este posee (tablas, formularios, etc) son declarados por medio de etiquetas. Las etiquetas son palabras especiales encerradas entre los signos < y >.

<etiqueta>

Existen etiquetas de apertura y de cierre, las cuales se utilizarán para abrir o cerrar una sección dentro del documento. Las etiquetas de apertura comienzan con el < , mientras que las de cierre con </ .

<etiqueta>texto entre etiquetas</etiqueta>

## Secciones de un documento HTML

Un documento está dividido en dos secciones. A continuación definiremos la estructura básica y explicaremos cada una de ellas:

```
<HTML>
  <HEAD>
    // información de cabecera.
  </HEAD>

  <BODY>
    // información de contenido del documento.
  </BODY>
</HTML>
```

Las etiquetas <HTML> definen que el código que se encuentra entre ellas corresponde a un documento HTML. Dentro de las mismas existen dos bloques: HEAD y BODY.

La sección <HEAD> (cabecera) es la sección que primero recibe el navegador. Dentro de ella enviamos todo los datos que sirven al navegador para identificar el tipo de información contenida en el cuerpo (*body*). Entre estos datos podemos mencionar: el lenguaje en el que se debe mostrar el documento (ingles, español, etc), la codificación que deberá utilizar el navegador para mostrar el documento (ISO8859-1, UTF-8), etc.

Por último la sección <BODY> contiene el contenido de la pagina web en si. Aquí dentro definiremos las tablas, el texto, los colores y todo lo que nos será visible en la pagina que estamos programando.

Ejemplo:

```
<html>
  <head>
```



```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Content-Language" content="es-ar">
<title>Ejemplo de una Página Web</title>
</head>

<body>
  <table border="0" cellpadding="0" cellspacing="0" align="center">
    <tr>
      <td>Hola, este es el cuerpo de la página. Estoy dentro de el Body.</td>
    </tr>
  </table>
</body>
</html>
```

Dentro de esta última sección es donde nosotros deberemos de escribir o definir nuestra página web.

## Formularios

Los formulario son la herramientas que nos provee HTML para enviar datos desde un navegador web a un script alojado en el servidor. Básicamente esta interacción se da utilizando alguno de éstos dos métodos: *GET* y *POST*. La utilización de uno u otro presenta diferencias las cuales estudiaremos más adelante en este capítulo.

Para declarar un formulario en HTML deberemos de hacer uso de las etiquetas *<form>* *</form>* e incluir dentro de las mismas los componentes que queramos. Además encontraremos distintos atributos para trabajar con ellos:

- **action=URL:** especifica el script que procesará los datos enviados por el formulario.
- **method=get/post:** el método utilizado para enviar los datos.
- **enctype=tipo de contenido:** este atributo especifica el tipo de contenido usado para enviar el formulario al servidor (siempre y cuando se envíe por *post*). El valor por defecto de este atributo es "application/x-www-form-urlencoded". El valor "multipart/form-data" debería usarse en combinación con el elemento INPUT, type="file".
- **name=nombre:** este atributo da nombre al formulario de manera que se pueda hacer referencia a él desde CSS o scripts.

```
<form action="procesoFormulario.php" method="POST">
  // los componentes que posea
</form>
```

**Nota:** cabe aclarar que no se han incluido todos los atributos, sino los que creemos relevante para el dictado de este curso.

## Componentes de los Formularios

En HTML existe una gran diversidad de componentes que podremos incluir dentro de los formularios para recoger la información a ser enviada. Podemos a grandes rasgos hacer una división de estos en:

- elementos INPUT (text, hiddend, file, password, checkbox, radio, reset, submit, button)
- elemento SELECT
- elemento TEXTAREA

Los elementos de tipo INPUT presentan distintos tipos, los cuales definiremos a través de su atributo *type*. Cada tipo de elemento presentará una funcionalidad distinta dentro del formulario. Sin embargo, todos estos ellos compartirán un conjunto de atributos:

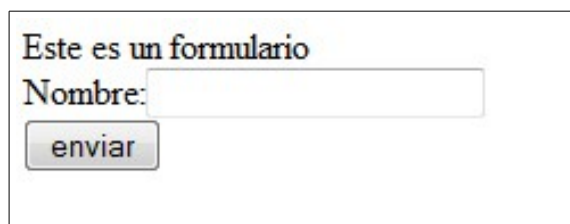
- **type=tipo:** definirá el tipo de elemento.
- **name=nombre:** el nombre del elemento mediante el cual lo reconoceremos en script o css.
- **value=valor:** el valor inicial que tomará el elemento cuando sea cargado el formulario.

### Componente INPUT

Como vimos anteriormente, dentro del componente INPUT encontraremos distintos tipos de elementos. A continuación describiremos cada uno de ellos.

- **text:** permite el ingreso de una cadena de texto, número y caracteres especiales. Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Nombre: <input type="text" value="" name="nombre" /><br />
      <input type="submit" value="enviar">
    </form>
  </body>
</html>
```

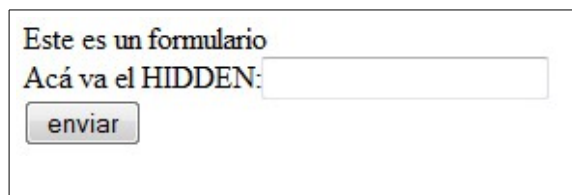
The image shows a visual rendering of the HTML code provided. It features a text input field with the label "Nombre:" and a submit button labeled "enviar". The text "Este es un formulario" is displayed above the input field. The entire form is enclosed in a rectangular border.

*Figura1: elemento TEXT*

- **hidden:** permite el ingreso de caracteres de la misma forma que *text* pero posee la particularidad de que no es visible dentro del formulario. Es utilizado para enviar información de manera *"invisible"* al usuario.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Acá va el Hidden
      <input type="hidden" value="5" name="oculto" /><br />
      <input type="submit" value="enviar">
    </form>
  </body>
</html>
```



*Figura2: elemento HIDDEN*

- **file:** permite el ingreso de un archivo a ser enviado al servidor. Este componente nos presentará un cuadro mediante el cual podremos seleccionar el archivo a subir.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoForm.php" method="POST">
      Seleccione su foto:
      <input type="file" value="" name="foto" />
    </form>
  </body>
</html>
```

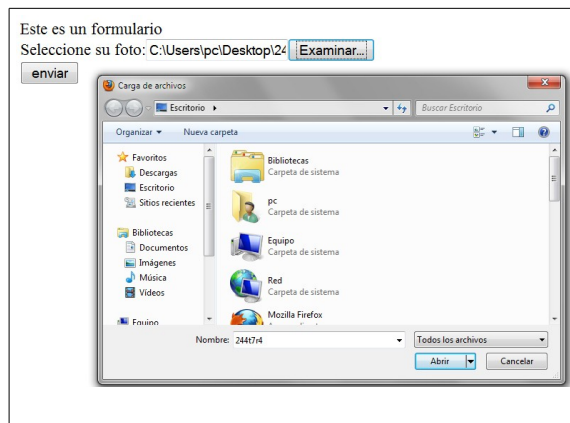


Figura 3: elemento FILE

- **password:** permite el ingreso de contraseñas. Presenta los valores en forma de asteriscos.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />Contraseña:
      <input type="password" value="5" name="contrasenia" />
      <br /><input type="submit" value="enviar">
    </form>
  </body>
</html>
```

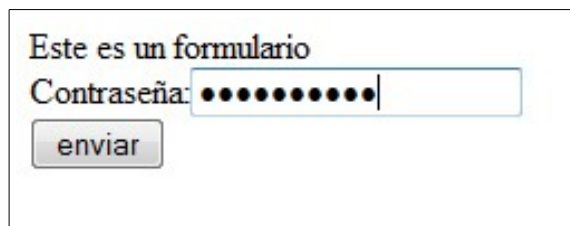


Figura4: elemento PASSWORD

- **checkbox:** nos presentan casillas de chequeo mediante los cuales podremos seleccionar uno o más valores de una lista de valores.

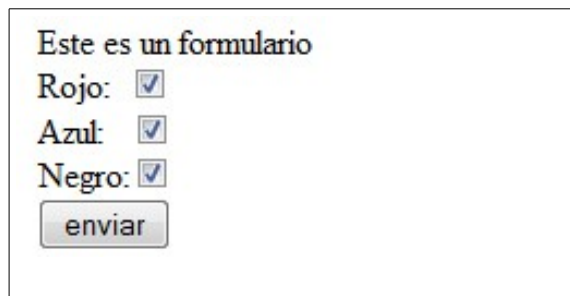
Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
```

```

Este es un formulario <br />
Rojo:<input type="checkbox" name="color" value="rojo"/>
<br />
Azul:<input type="checkbox" name="color" value="azul"/>
<br />
Negro:<input type="checkbox" name="color" value="negro"/>
<br />
<input type="submit" value="enviar">
</form>
</body>
</html>

```



*Figura5: elemento checkbox*

- **radio:** nos permite por medio de botones la selección de un valor de un conjunto de valores.

Ejemplo:

```

<html>
<head><title>Programación en PHP</title></head>
<body>
  <form action="procesoFormulario.php" method="POST">
    Este es un formulario <br />
    Rojo:<input type="radio" name="color" value="rojo"/>
    <br />
    Azul:<input type="radio" name="color" value="azul"/>
    <br />
    Negro:<input type="radio" name="color" value="negro"/>
    <br />
    <input type="submit" value="enviar">
  </form>
</body>
</html>

```

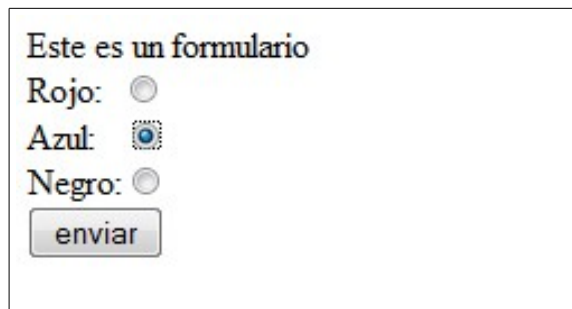


Figura5: elemento radio

- **reset:** presenta un botón mediante el cual borraremos todos los datos que contengan los componentes del formulario.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Nombre: <input type="text" value="" name="nombre" /><br />
      <input type="reset" />
    </form>
  </body>
</html>
```

- **submit:** presenta un botón mediante el cual podremos enviar el formulario para que sea procesado mediante el script definido en el atributo *action* del mismo.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Nombre: <input type="text" value="" name="nombre" /><br />
      <input type="submit" />
    </form>
  </body>
</html>
```

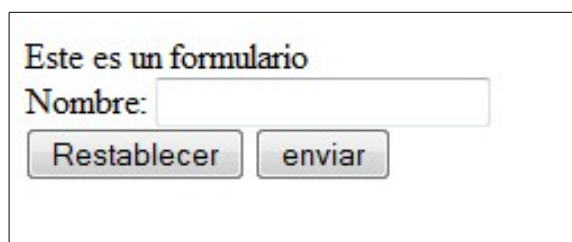


Figura 6: elementos reset y submit

- **button:** presenta un botón mediante pero el mismo no trae ninguna funcionalidad asociada. Presenta un componente a ser utilizado, por ejemplo, en llamadas a funciones javascript.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Nombre: <input type="text" value="" name="nombre" /><br />
      <input type="button" value="boton">
    </form>
  </body>
</html>
```

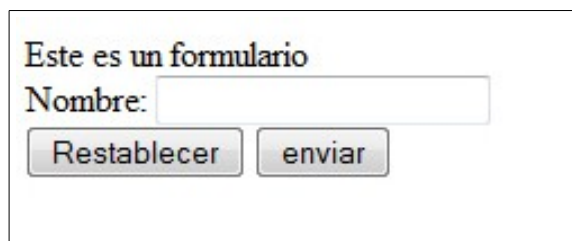


Figura 6: elemento button

## Componente SELECT

Presenta una lista desplegable con un conjunto finito de valores de entre los cuales podremos elegir alguno.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />
      Selecciones un Color:
      <select name="color">
        <option value="rojo">rojo</option>
        <option value="azul">azul</option>
        <option value="negro">negro</option>
      </select>
      <input type="submit" value="enviar">
    </form>
  </body>
```

```
</html>
```

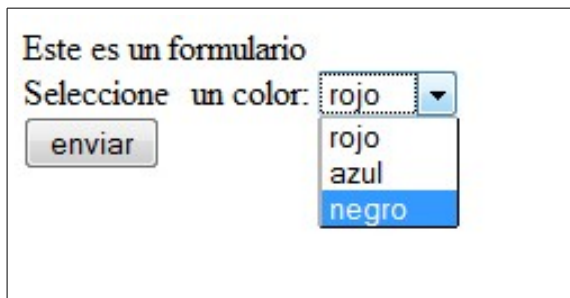


Figura 7: componente select

## Componente TEXTAREA

Presenta un área de texto para ingresar párrafos de caracteres.

Ejemplo:

```
<html>
  <head><title>Programación en PHP</title></head>
  <body>
    <form action="procesoFormulario.php" method="POST">
      Este es un formulario <br />Ingrese su comentario:
      <textarea name="areatexto"></textarea><br />
      <input type="submit" value="enviar">
    </form>
  </body>
</html>
```

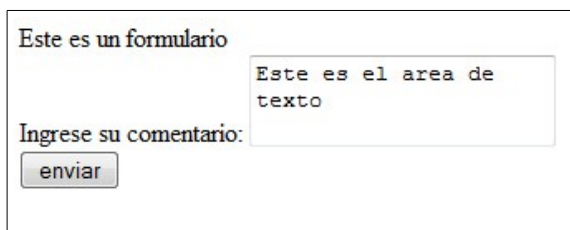


Figura 8: componente y texarea

## Métodos de envío de un Formulario

Como vimos en el apartado anterior, los formularios son la herramienta que el HTML nos ofrece para el envío de datos desde un navegador a nuestro servidor web.

Existen dos métodos para enviar formularios: GET y POST. En cualquiera de los métodos, los datos del formulario serán enviados en pares *nombre/valor*, es decir conjuntos de datos representados por el nombre del elemento formulario, un signo igual ("=") y luego el valor asociado. Estos pares nombre/valor se separan unos de otros mediante el símbolo de



unión ("&"). Ejemplo:

```
campo1=valor1&campo2;=valor2&campo3;=valor3
```

La diferencia en los dos métodos radica en la manera en que se envían los datos. A continuación veremos la diferencia entre ellos.

- **GET:** es el método por defecto que toman los formularios para enviar los datos. En este tipo de envío, los datos del formulario se adjuntan a la URL definida en el atributo *action* luego de anteponer un signo ? al final de la misma. De esta manera, la información a enviar viaja a través de la URL de destino. Ejemplo:

```
<form action="procesoFormulario.php" method="get">  
  Nombre: <input type="text" name="nombre" />  
  <input type="submit" name="bt_enviar" value="Enviar" />  
</form>
```

Al enviar el formulario, la URL formada quedará de la siguiente manera:

```
http://localhost/procesoFormulario.php?nombre=María&bt_enviar=Enviar
```

- **POST:** envía los datos haciendo uso de las cabeceras HTTP existentes en la comunicación. Es una forma un poco más segura de enviar los datos y se aconseja usarla cuando los datos deban interactuar con una base de datos. Ejemplo:

```
<form action="procesoFormulario.php" method="post">  
  Nombre: <input type="text" name="nombre" />  
  <input type="submit" name="bt_enviar" value="Enviar" />  
</form>
```

Al enviar el formulario, la URL formada quedará de la siguiente manera:

```
http://localhost/procesoFormulario.php
```

Pero si hiciésemos una captura de datos, en la cabecera HTTP encontraríamos algo similar a esto:

```
POST /index.php HTTP/1.1  
Host: localhost  
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.5) Gecko/20091102  
Firefox/3.5.5  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
Accept-Language: en-us,en;q=0.5  
Accept-Encoding: gzip,deflate  
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7  
Keep-Alive: 300  
Connection: keep-alive  
Referer: http://google.com  
Content-Type: application/x-www-form-urlencoded
```

Content-Length: 23

**nombre=María&apellido=La del Barrio**

Como podemos ver, al final de este fragmento encontramos los datos que formaban parte del formulario.

## Procesando datos con PHP

Siempre que enviamos datos a través de un formulario éstos deberán ser procesados por algún script en el servidor, en este caso un script de PHP. Es por esto que necesitaremos de algún tipo de herramienta para recuperar los mismos. A continuación daremos una breve explicación de como PHP realiza el tratamiento de estos datos dada la forma de envío que declaramos en el formulario.

Según el método de envío (get o post), PHP definirá un arreglo asociativo con todos los datos enviados a través del formulario. Si el formulario fue enviado a través de *get*, el arreglo se denominará `$_GET[]`, de otra forma, `$_POST[]`. En ambos casos, los *índices* de cada arreglo estarán formados por los campos enviados desde el formulario y los *elementos* por los valores.

```
method="get" → $_GET[]  
method="post" → $_POST[]
```

Ejemplo:

Dado el siguiente formulario enviado por **get**:

```
<form action="procesoFormulario.php" method="get">  
  Nombre: <input type="text" name="nombre" value="María"/>  
  Apellido: <input type="text" name="apellido" value="Sosa"/>  
  <input type="submit" name="bt_enviar" value="Enviar" />  
</form>
```

Para recuperar los datos del mismo dentro del script procesoFormulario.php:

```
<?php  
  $nombre=$_GET['nombre'];  
  $apellido=$_GET['apellido'];  
  echo "Hola, ".$nombre." ".$apellido  
?>
```

Ahora por **post**:

```
<form action="procesoFormulario.php" method="post">
```

```
Nombre: <input type="text" name="nombre" value="María"/>
Apellido: <input type="text" name="apellido" value="Sosa"/>
<input type="submit" name="bt_enviar" value="Enviar" />
</form>
```

Para recuperar los datos del mismo dentro del script procesoFormulario.php:

```
<?php
    $nombre=$_POST['nombre'];
    $apellido=$_POST['apellido'];
    echo "Hola, ".$nombre." ".$apellido;
?>
```

Para saber si un formulario fue enviado, podemos hacer uso de la variable que representa al botón submit, en nuestro caso de nombre *bt\_enviar*.

```
<?php
    If (isset($_POST['bt_enviar'])){
        print_r ($_POST['nombre']);
    }
?>
```

En el caso de que fuesen enviados por get:

```
<?php
    If (isset($_GET['bt_enviar'])){
        print_r ($_GET['nombre']);
    }
?>
```

## Tratamiento de los componentes en PHP

A continuación veremos de que manera obtener los datos según el tipo de componente definido en el formulario.

### Elementos INPUT

- TEXT: siempre contiene un solo valor al momento de enviar el formulario. Ejemplo:

```
<input type="text" name="nombre">
```

En PHP:

```
<?php
    $nombre = $_GET['nombre'];
    $nombre = $_POST['nombre'];
?>
```

- **HIDDEN:** siempre contiene un solo valor al momento de enviar el formulario. Ejemplo:

```
<input type="hidden" name="id_producto">
```

En PHP:

```
<?php
    $id_producto = $_GET['id_producto'];
    $id_producto = $_POST['id_producto'];
?>
```

- **PASSWORD:** siempre contiene un solo valor al momento de enviar el formulario. Ejemplo:

```
<input type="password" name="contrasenia">
```

En PHP:

```
<?php
    $contrasenia = $_GET['contrasenia'];
    $contrasenia = $_POST['contrasenia'];
?>
```

- **RADIO:** se elije un único valor de una lista de posibles valores. Ejemplo:

```
<input type="radio" name="sexo" value="femenino">
<input type="radio" name="sexo" value="masculino">
```

En PHP:

```
<?php
    $sexo = $_GET['sexo'];
?>
```

**Nota:** se envía únicamente el componente chequeado.

- **CHECKBOX:** permite enviar uno o más valores asociados a un solo nombre. Todos los campos check que queramos que se relacionen, deben llevar el mismo nombre, de la forma: name="colores[]". Solamente se enviarán los valores chequeados. PHP tratará a los valores como un array. Ejemplo:

```
Rojo<input type="checkbox" name="colores[]" value="rojo"><br />
Azul<input type="checkbox" name="colores[]" value="azul"><br />
Verde<input type="checkbox" name="colores[]" value="verde"><br />
```

En PHP:

```
<?php
    $colores = $_POST["colores"];
    foreach ($colores as $color){
        print ("Color seleccionado: ".$color."<br>");
    }
?>
```

## Elemento SELECT

Nos permite seleccionar y enviar un único valor seleccionado de una lista desplegable de valores. Ejemplo:

```
<select name="departamento">
  <option value="parana">Paraná</option>
  <option value="nogoya">Nogoyá</option>
  <option value="galeguay">Galegay</option>
  <option value="victoria">Victoria</option>
</select>
```

En PHP:

```
<?php
    print("El Departamento Seleccionado es : ".$_POST["departamento"]);
?>
```

## Elemento TEXTAREA

Nos permite enviar una cadena caracteres extensa a través de un único nombre de componente. Ejemplo:

Escribanos algo: <textarea rows="5" cols="20" name="texto"></textarea>

En PHP:

```
<?php
    print ("Esta es una copia del texto enviado: " . "<br>");
    print ($_POST["texto"]);
```