Estructura del lenguaje

3.1 Sintaxis del lenguaje

PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP necesitamos especificar cuáles son las partes constitutivas del código escritas en este lenguaje.

Esto se hace, como en otros casos, delimitando nuestro código por etiquetas.

Estos son los modos de abrir y cerrar las etiquetas son:

- <? ?>
- <% %>
- <?php ?>
- <script language="php">

Otra característica general de los scripts en PHP es la forma de separar las distintas instrucciones. Para hacerlo, hay que acabar cada instrucción con un punto y coma **";"**. Para la ultima expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

Si usamos doble barra (//) o el símbolo # podemos introducir comentarios de una línea. Mediante /* y */ creamos comentarios multilinea.

```
<?
$mensaje="Tengo hambre!!"; //Comentario de una linea
echo $mensaje; #Este comentario también es de una linea
/*En este caso
mi comentario ocupa
varias lineas, lo ves? */
?>
```

3.1.1 Declarando e inicializando variables

Variables

Una variable es un contenedor de información, en el que podemos meter números enteros, números decimales, caracteres, etc. El contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.

En PHP todas las variables comienzan con el símbolo del dólar $ y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres.

Existen 2 tipos de variables, las variables locales que solo pueden ser usadas dentro de funciones y las variables globales que tienen su ámbito de uso fuera de las funciones, podemos acceder a una variable global desde una función con la instrucción global nombre_variable;

3.1.2 Sentencias y expresiones

Operadores Aritméticos

Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Resta	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	$a++	Suma 1 al contenido de una variable.
--	Resta 1	$a--	Resta 1 al contenido de una variable.

Operadores de comparación

Los operadores de comparación son usados para comparar valores y así poder tomar decisiones.

Operador	Nombre	Ejemplo	Devuelve verdadero cuando:
==	Igual	$a == $b	$a es igual $b
===	Idéntico	$a === $b	$a es igual $b y ambos son del mismo tipo
!=	Distinto	$a != $b	$a es distinto $b
!==	No idéntico	$a !== $b	$a no es idéntico a $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

Operadores Lógicos

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas.

Operador	Nombre	Ejemplo	Devuelve verdadero cuando:
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

Sentencias condicionales

Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado de evaluar una condición. Las más frecuentes son la instrucción **if** y la instrucción **switch**.

La sentencia **if** ejecuta una serie de instrucciones u otras dependiendo de la condición que le pongamos.

Probablemente sea la instrucción más importante en cualquier lenguaje de programación.

```php
<?php
 $a = 10;
 $b = 20;
if($a > $b){
 echo $b;
} else {
    echo $a;
}
 ?>
```

Sentencia switch

Con la sentencia **switch** podemos ejecutar unas u otras instrucciones dependiendo del valor de una variable.

```php
<?php
$opcion = 3;
    switch($opcion){
        case 1:
            echo "El valor es 1";
            break;
        case 2:
            echo "El valor es 2";
            break;
        case 3:
            echo "El valor es 3";
            break;
        default:
            echo "El valor es diferente de 1,2 y 3";
            break;
    }
?>
```

3.1.3 Declarando y usando arreglos

El PHP ofrece la posibilidad de agrupar un conjunto de valores para almacenarlos juntos y referenciarlos por un índice.

```php
<?php
    $array[0] = "Tecgurus ";
    $array[1] = "PHP ";
    $array[2] = "Desde  ";
    $array[3] = "Cero ";

    echo $array[0];

?>
```

Los índices puedes ser del tipo numérico (entero) o una cadena de forma indistinta.

```php
<?php
    $array["tecgurus"] = "Tecgurus ";
    $array["PHP"] = "PHP ";
    $array["Desde"] = "Desde  ";
    $array["Cero"] = "Cero ";

    echo $array[0];

?>
```

3.1.4 Manejo de errores

Existen diferentes tipos de errores entre ellos existen:

- **Sintácticos:** como cuando escribimos mal una palabra o nos olvidamos el punto y coma.
- **De ejecución:** como cuando queremos acceder a un archivo que no existe.
- **Lógicos:** como cuando proveemos de datos incorrectos a una variable.

Existen varios mensajes que PHP nos puede emitir (errores, advertencias, etc.). Cada mensaje es representado por una constante que nos permite clasificarlo:

- **E_ERROR:** Error fatal. No es posible recuperarse de este error, finaliza la ejecución del script.
- **E_WARNING:** Advertencia en tiempo de ejecución. El script se sigue ejecutando.
- **E_NOTICE:** Aviso en tiempo de ejecución. Se encontró algo que podría ser un error pero no es seguro.
- **E_DEPRECATED:** Advertencia que indica que el código no funcionará en futuras versiones.

•**E_STRICT:** Aviso que sugiere un cambio al código para asegurar la interoperación y compatibilidad con futuras versiones.
•**E_USER_ERROR:** Mensaje de error generado por el usuario utilizando trigger_error().
•**E_USER_WARNING:** Mensaje de advertencia creado por el usuario utilizando trigger_error().
•**E_USER_NOTICE:** Aviso creado por el usuario utilizando trigger_error().
•**E_USER_DEPRECATED:** Aviso creado por el usuario utilizando trigger_error().
•**E_ALL:** Habilita todos los errores y advertencias PHP

Podemos modificar la configuración de PHP para indicar qué queremos que pase con los errores.

- **error_reporting:** Indicamos el nivel de reporte de los errores
- **display_errors:** Indicamos si los errores deberían mostrarse por pantalla (en producción conviene desactivar este parámetro)
- **log_errors:** Indicamos si los errores deberían mostrarse en el log de errores del servidor.

La configuración la podemos establecer en el archivo php.ini, en la configuración del servidor

3.2.2 Ciclos

Los ciclos o también conocidos como bucles, nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

Sentencia for

La instrucción **for** es la instrucción de bucles más completa. En una sola instrucción nos permite controlar todo el funcionamiento del bucle.

El primer parámetro del **for**, es ejecutado la primera vez y sirve para inicializar la variable del bucle, el segundo parámetro indica la condición que se debe cumplir para que el bucle siga ejecutándose y el tercer parámetro es una instrucción que se ejecuta al final de cada iteración y sirve para modificar el valor de la variable de iteración.

```php
<?php

for($i = 6;$i < 10;$i++){
 echo "El valor de i es menor de 10";
}
?>
```

Sentencia While

Mientras la condición sea cierta se reiterará la ejecución de las instrucciones que están dentro del **while**.

```php
<?php
    $i = 6;
    while($i < 10){
        echo "El valor de i es menor de 10";
        $i++;
    }
?>
```

3.2.3 Manejando excepciones

¿Que es una excepción?

Una excepción es un evento que ocurre durante la ejecución de un programa y requiere de la ejecución controlada de un bloque de código fuera del flujo de normal de ejecución.

El manejo de excepciones es una herramienta muy potente a la hora de realizar una gestión de una situación. Lo primero que hay que comprender es que una excepción no es un error. Es una situación que se experimenta un bloque de código y que este no es capaz de manejar.

PHP cuenta con dos tipos de excepciones predefinidas, **Exception** y **ErrorException**, y un conjunto de excepciones definidas en la **SPL** (standard PHP library).

Podemos lanzar excepciones manualmente utilizando la palabra clave **throw**:

```php
<?php
thrown new Exception('Mensaje');
?>
```

El constructor de la clase **Exception** adopta una serie de parámetros opcionales, un mensaje y un código de error.

Para manejar las excepciones que lanzamos utilizamos la estructura de control **try/catch.** Todo el código que pueda lanzar una excepción debe incluirse dentro del bloque try ya que este bloque es el encargado de parar la ejecución del script y pasar el control al bloque catch cuando se produzca una excepción.

```php
<?php
try {

// Codigo que puede lanzar excepciones.

}
catch ( Exception $excepcion ){

// Codigo para controlar la excepcion.

}
?>
```

3.3 Definiendo clases

Una clase es una plantilla (molde), que define atributos (lo que conocemos como variables) y métodos (lo que conocemos como funciones).

La clase define los atributos y métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

Debemos crear una clase antes de poder crear objetos (instancias) de esa clase.

Al crear un objeto de una clase, se dice que se crea una instancia de la clase o un objeto propiamente dicho.

```php
<?php
    class ejemplo{
        //Funciones y objetos a declarar
    }
?>
```

3.3.1 Atributos

Éstas se definen usando una de las palabras reservadas *public*, *protected*, o *private*, seguido de una declaración normal de variable. Esta declaración puede incluir una inicialización, pero esta inicialización debe ser un valor constante, es decir, debe poder ser evaluada durante la compilación y no depender de información generada durante la ejecución.

```php
<?php
    class ejemplo{
        public $a = 10;
        private $b = 12;
        protected $c = 14;
    }
?>
```

Dentro de los métodos de una clase, se puede acceder a las propiedades no estáticas utilizando -> (el operador de objeto): *$this->propiedad* (donde *propiedad* es el nombre de la propiedad). A las propiedades estáticas se puede acceder utilizando *::* (doble dos puntos): *self::$propiedad*.

$this está disponible dentro de cualquier método de clase cuando éste es invocado dentro del contexto de un objeto. *$this* es una referencia al objeto invocador (usualmente el objeto al cual pertenece el método, aunque puede que sea otro objeto si el método es llamado estáticamente desde el contexto de un objeto secundario).

```php
<?php
    class ejemplo{
        private $tecgurus = FALSE;

        public function __construct(){
            $this->$tecgurus = TRUE;
            echo($this->$tecgurus);
        }
    }
?>
```

3.3.2 Funciones

uso de funciones nos da la capacidad de agrupar varias instrucciones bajo un solo nombre y poder llamarlas a estas varias veces desde diferentes sitios, ahorrándonos la necesidad de escribirlas de nuevo.

Opcionalmente podemos pasarle parámetros a las funciones que se trataran como variable locales y así mismo podemos devolver un resultado con la instrucción return valor; Esto produce la terminación de la función retornando un valor.

3.3.3 Parámetros

Cualquier información puede ser pasada a las funciones mediante la lista de argumentos, la cual es una lista de expresiones delimitadas por comas. Los argumentos son evaluados de izquierda a derecha.

```php
<?php
    class ejemplo{
        public function valorA($a){
            $a = 57;
            echo $a //Esto imprime 57
        }
    }
?>
```

3.3.4 Ventajas de crear clases

- Reutilización de código
- Aceleran el desarrollo
- Permite crear sistemas programados más complejos.
- Relacionar el sistema al mundo real.
- Facilita la creación de programas visuales.
- Permiten la construcción de prototipos.
- Agilizan el desarrollo de software.

3.4.1 Código del lado del cliente

Los lenguajes de lado cliente (entre los cuales no sólo se encuentra el HTML sino también el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento.

Así, por ejemplo, un lenguaje de lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas.

3.4.2 Código del lado del servidor

Los lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

Un lenguaje de lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo.

3.4.4 Estilos

CSS, es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores... CSS son las siglas de Cascading Style Sheets, en español Hojas de estilo en Cascada. En este reportaje vamos a ver algunos de los efectos que se pueden crear con las CSS sin necesidad de conocer la tecnología entera.

Las Hojas de Estilo en Cascada se escriben dentro del código HTML de la página web, solo en casos avanzados se pueden escribir en un archivo a parte y enlazar la página con ese archivo. También se puede utilizar la manera más directa de aplicar estilos a los elementos de la página. Para ello, y esto es la primera lección de este artículo, vamos a conocer un nuevo atributo que se puede utilizar en casi todas las etiquetas HTML: style.

** El párrafo saldrá con color verde y en negrita **

Dentro del atributo style se deben indicar los atributos de estilos CSS separados por punto y coma (;).

3.4.5 Integrando librerías externas (JQuery)

jQuery es una librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

```
<script src="//code.jquery.com/jquery-latest.js"></script>
```

Formularios

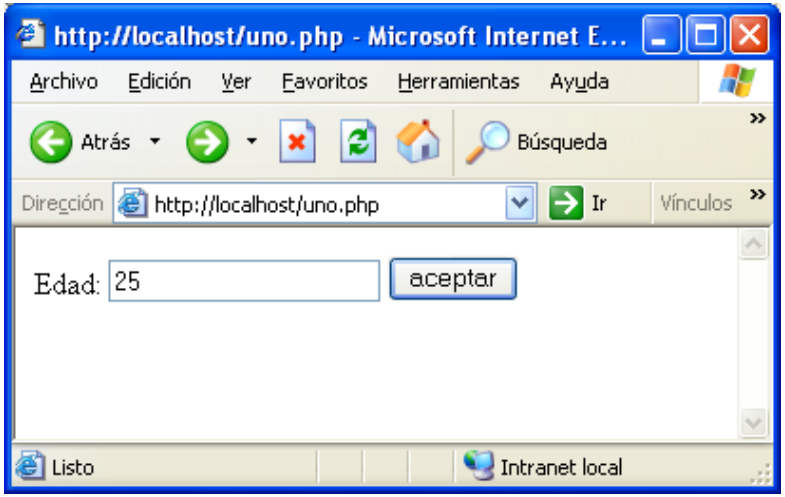
Desde PHP se puede acceder fácilmente a los datos introducidos desde un formulario HTML.

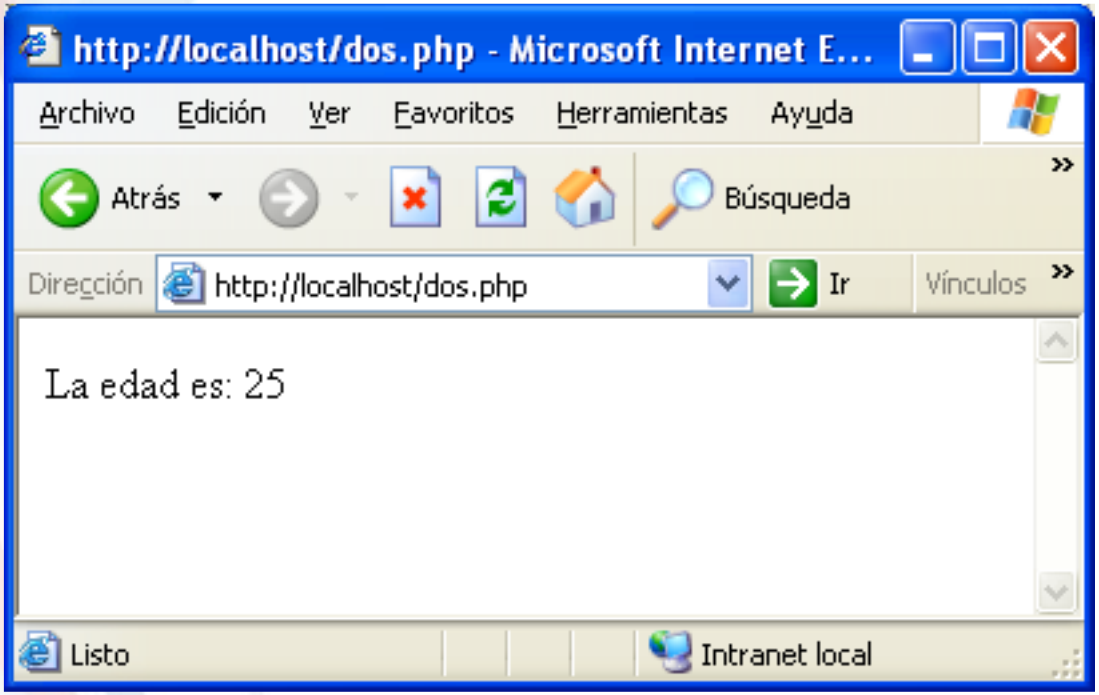
Ejemplo:
Fichero uno.php

```
<HTML>
<BODY>
<FORM ACTION=”dos.php” METHOD=”POST”>
Edad: <INPUT TYPE=”text” NAME=”edad”>
<INPUT TYPE=”submit” VALUE=”aceptar”>
</FORM>
</BODY>
</HTML>
```

•Fichero dos.php

```
<HTML>
<BODY>
<?PHP
print (“La edad es: $edad”);
?>
</BODY>
</HTML>
```





A partir de PHP 4.2.0, el valor por defecto de la directiva de PHP **register_globals** es off.

•Esto tiene una gran importancia sobre los formularios, ya que no es posible acceder a las variables enviadas de la manera anterior (como variables globales).

En su lugar hay que utilizar la variable predefinida de PHP **$_REQUEST**, escribiendo **$_REQUEST[‘edad’]** en lugar de **$edad.**

•Se puede poner register_globals = on en el fichero de configuración php.ini, pero no es recomendable por motivos de seguridad. Una alternativa que permite hacer mínimos cambios en el código ya existente es la siguiente:

$edad = $_REQUEST[‘edad’];

Acceso a los diferentes tipos de elementos de entrada de formulario

–Elementos de tipo INPUT

- TEXT
- RADIO
- CHECKBOX
- BUTTON
- FILE
- HIDDEN
- PASSWORD
- SUBMIT

–Elemento SELECT

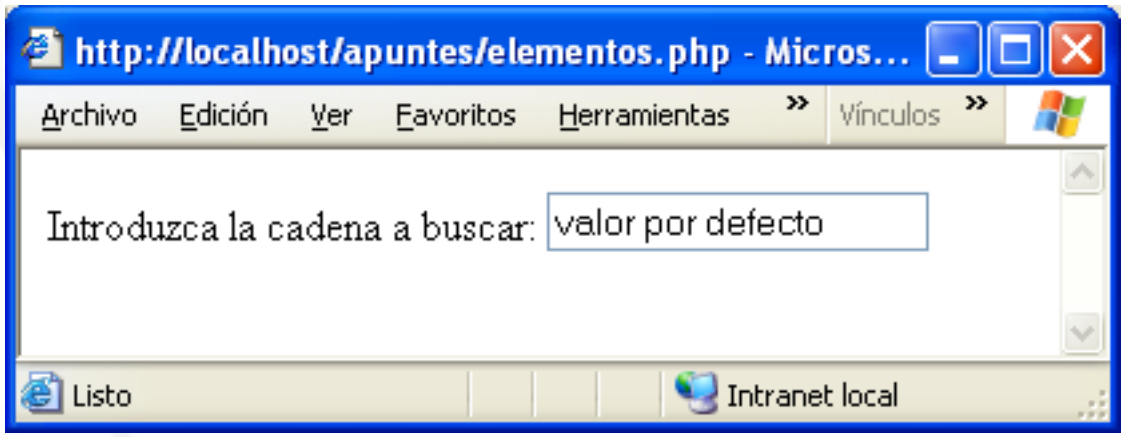
•Simple / múltiple

–Elemento TEXTAREA

•TEXT

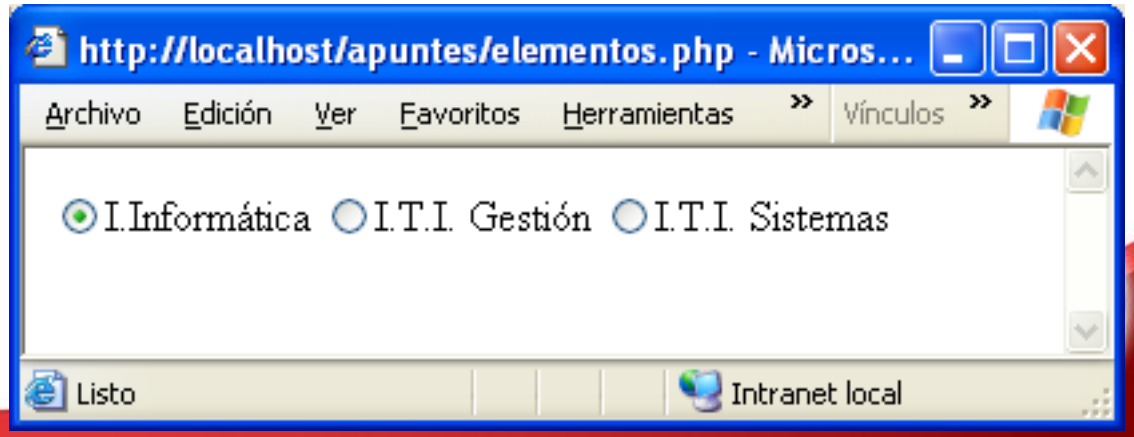
Introduzca la cadena a buscar:

```
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto" SIZE="20">
    <?PHP
        print ($cadena);
        //print ($_REQUEST ['cadena']);
    ?>
```



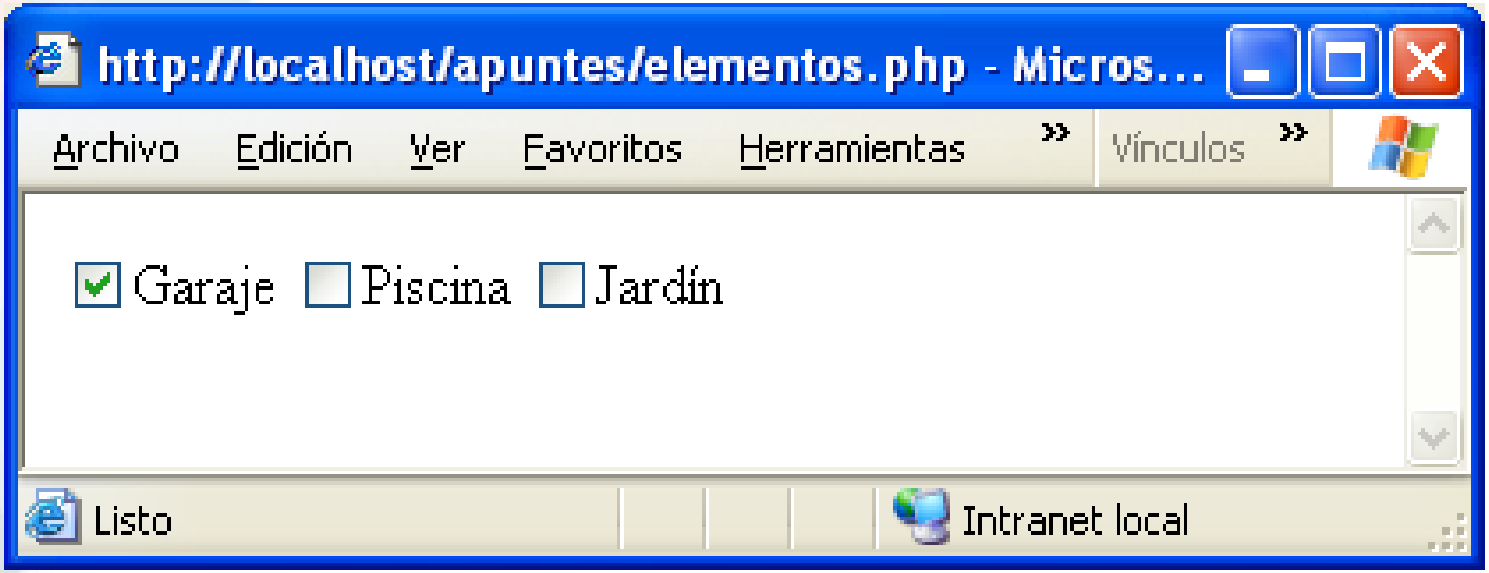
RADIO

```
<INPUT TYPE="radio" NAME="titulacion" VALUE="II"
CHECKED>I.Informática
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIG">I.T.I.
Gestión
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIS">I.T.I.
Sistemas

<?PHP
print ($titulacion);
//print ($_REQUEST ['titulacion']);
?>
```



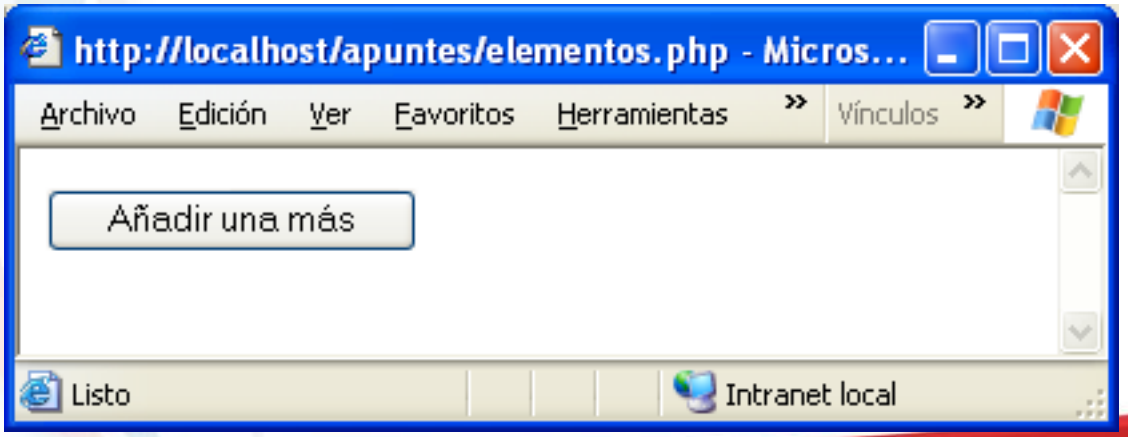
CHECKBOX

```
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="garaje"
CHECKED>Garaje <INPUT TYPE="checkbox" NAME="extras[]"
VALUE="piscina">Piscina <INPUT TYPE="checkbox" NAME="extras[]"
VALUE="jardin">Jardín

<?PHP
    $n = count ($extras);
    for ($i=0; $i<$n; $i++)
    print ("$extras[$i]<BR>\n");

//foreach ($_REQUEST['extras'] as $extra)
//print ("$extra<BR>\n"); ?>
```



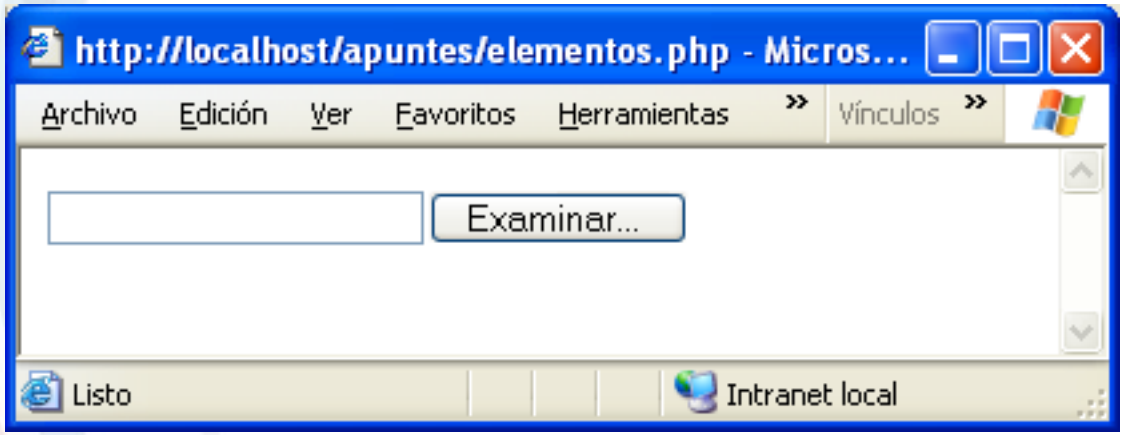
Button

```
<INPUT TYPE="button" NAME="nueva" VALUE="Añadir una más">

<?PHP if ($nueva)
        print ("Se va a añadir una nueva");
        //if ($_REQUEST [‘nueva’])
        //print ("Se va a añadir una nueva");
?>
```



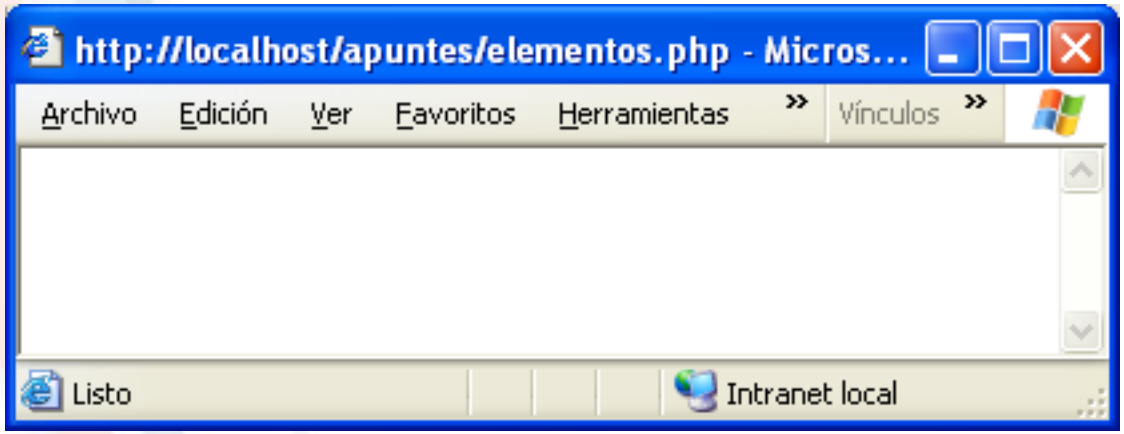
File

```
<FORM ACTION="procesa.php" METHOD="post“
      ENCTYPE="multipart/form-data">

      <INPUT TYPE="file" NAME="fichero">
</FORM>
```



Hidden

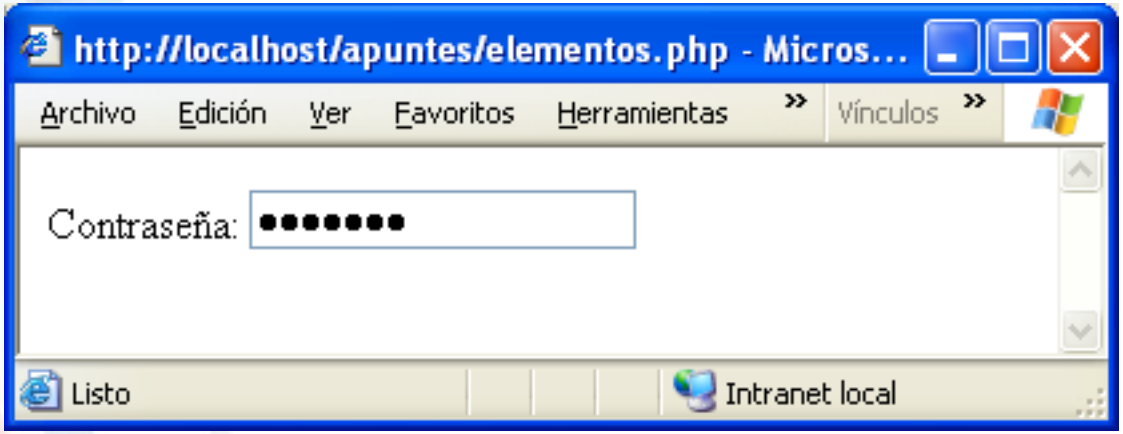
```
<?PHP
    print("<INPUT TYPE='hidden' NAME='username' VALUE='$usuario'>\n");
?>

<?PHP
    print ($username);
    //print ($_REQUEST ['username']);
?>
```



Password

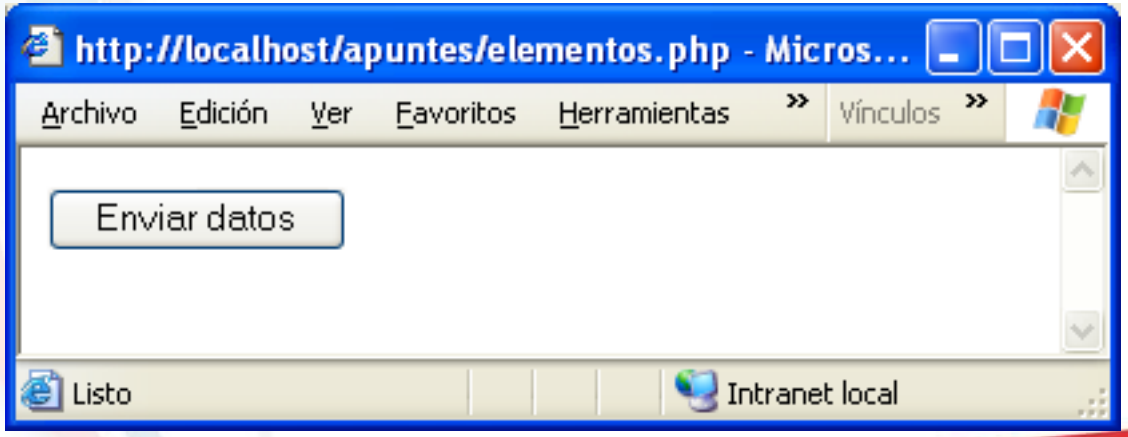
Contraseña: <INPUT TYPE="password" NAME="clave">

```
<?PHP print ($clave);
      //print ($_REQUEST [‘clave’]);
?>
```



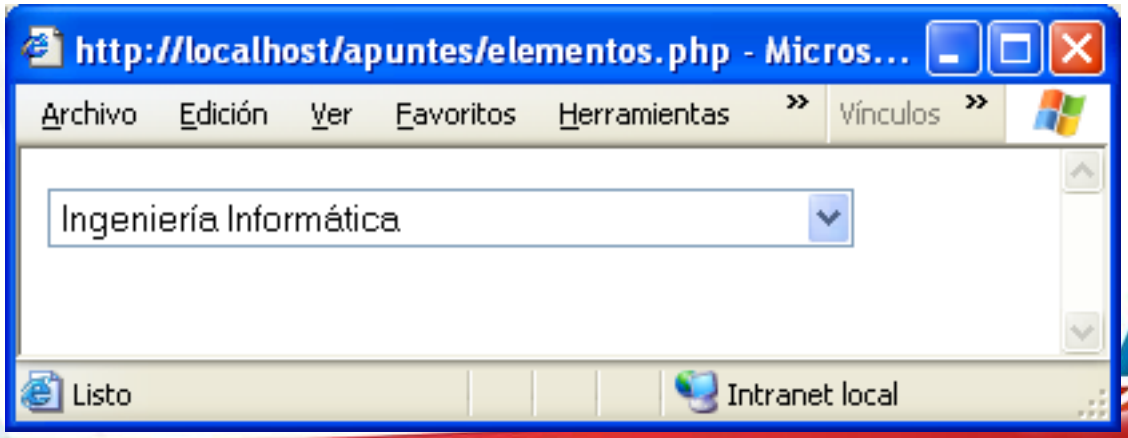
Submit

```
<INPUT TYPE="submit" NAME="enviar" VALUE="Enviar datos">

<?PHP if ($enviar)
      print ("Se ha pulsado el botón de enviar");
      //if ($_REQUEST [‘enviar’])
      //print ("Se ha pulsado el botón de enviar");
?>
```



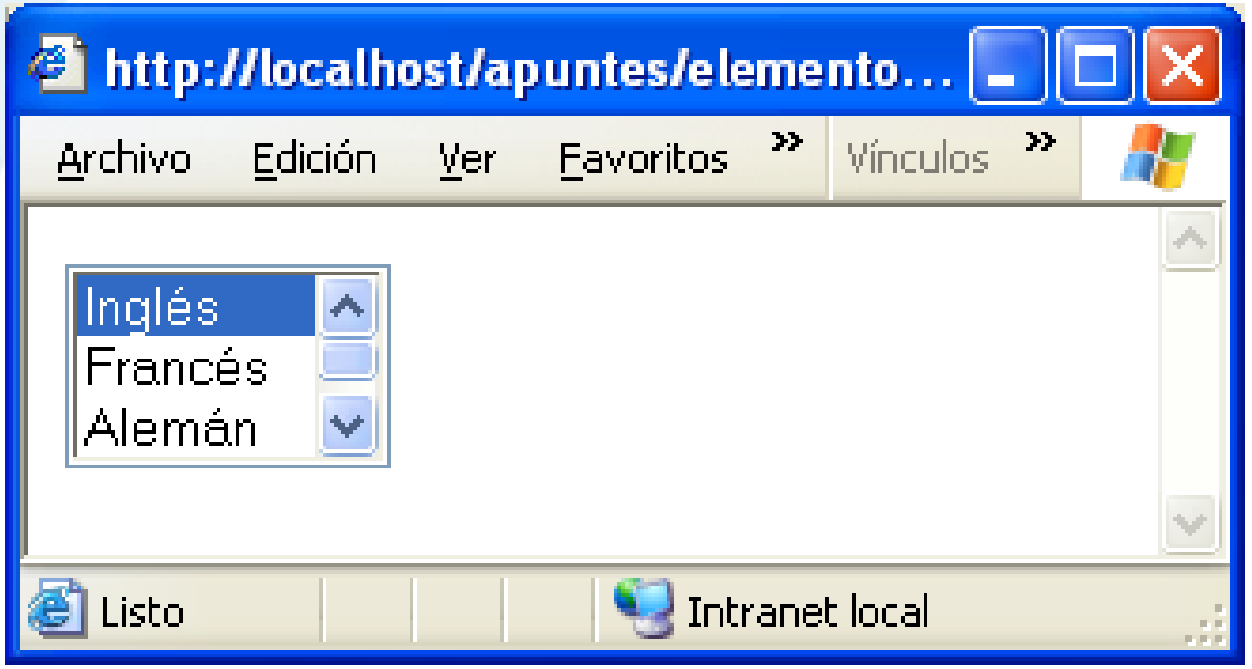
SELECT Simple

```
<SELECT NAME="titulacion">
    <OPTION VALUE="II" SELECTED>Ingeniería Informática
    <OPTION VALUE="ITIG">Ingeniería Técnica en Informática de Gestión
    <OPTION VALUE="ITIS">Ingeniería Técnica en Informática de Sistemas
</SELECT>

<?PHP print ($titulacion);
    //print ($_REQUEST ['titulacion']);
?>
```



SELECT Múltiple

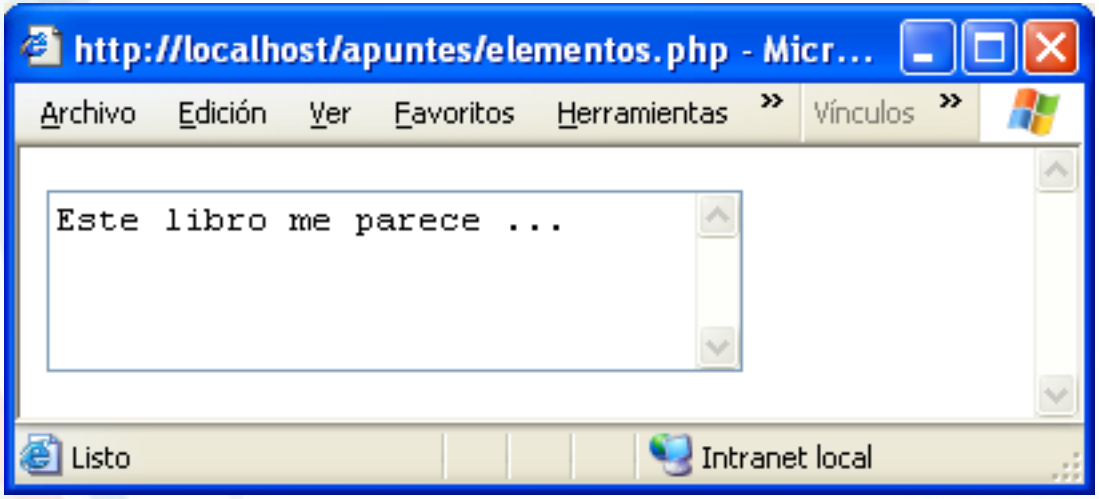
```
<SELECT MULTIPLE SIZE="3" NAME="idiomas[]">
      <OPTION VALUE="ingles" SELECTED>Inglés
      <OPTION VALUE="frances">Francés
      <OPTION VALUE="aleman">Alemán
      <OPTION VALUE="holandes">Holandés
</SELECT>

<?PHP
      $n = count ($idiomas);
      for ($i=0; $i<$n; $i++)
            print (“$idiomas[$i]<BR>\n”);
            //foreach ($_REQUEST[‘idiomas’] as $idioma)
            //print (“$idioma<BR>\n”);
?>
```



TEXTAREA

<TEXTAREA COLS=“30" ROWS=“4" NAME="comentario"> Este libro me parece ... </TEXTAREA>

```
<?PHP
      print ($comentario);
      //print ($_REQUEST [‘comentario’]);
?>
```



- 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

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($enviar))
o bien
if ($enviar == “procesar”)