# TEMA 2. INSERCIÓN DE CÓDIGO EN PÁGINAS WEB.

## Objetivos

- Aprender a generar código de forma dinámica para ser mostrada en el cliente web.
- Conocer la sintaxis de las etiquetas propias del lenguaje PHP que permite insertar código en páginas web ejecutadas en el servidor.
- Dominar la declaración de variable, tipos simples y conversiones de tipos.
- Comprender la importancia de controlar el ámbito de declaración de ceda variable.

## Contenidos

### 2.1.- Lenguajes y tecnologías del Servidor.

El intercambio de datos entre cliente y servidor se realiza mediante un protocolo denominado HTTP.

Un protocolo es un conjunto de reglas que gobiernan el intercambio de información entre entidades de una red.

El servidor a través del puerto de comunicaciones (normalmente el 80) se mantiene a la espera de peticiones HTTP realizadas por el navegador del cliente.

Secuencia de comunicación:

1.- Navegador solicita la traducción de la URL al servidor DNS.
2.- Se realiza la petición a la IP obtenida.
3.- El servidor procesa la solicitud del cliente.
4.- Tras ejecutar el código solicitado responde al cliente el código HTML de la página.
5.- El cliente recibe el código y lo interpreta mostrándolo en la pantalla.

En función de como procesasn las peticiones podemos distinguir distintos tipos de servidores web:

- **Servidores basados en procesos**. Su funcionamiento se basa en la obtención de un paralelismo de ejecución mediante la duplicación del proceso de ejecución.
- **Servidores basados en hilos**. La creación de un hilo por parte de un proceso servidor no es tan costosa como la duplicación de un proceso completo.
- **Servidores dirigidos por eventos**. Utilización de sockets (espacios de memoria para la comunicación entre dos aplicaciones que permiten que un proceso intercambie información con otro proceso estando los dos en distintas máquinas, en este caso cliente y servidor).
- **Servidores implementados en el núcleo del sistema (kernel).** Sitúan el procesamiento de cada una de las ejecuciones del servidor web en un espacio de trabajo perteneciente al sistema operativo (kernel) y no en un nivel de usuario (sobre el sistema operativo).

Las tecnologías descritas anteriormente se implementan en diferente grado en algunos de los servidores más utilizados actualmente:

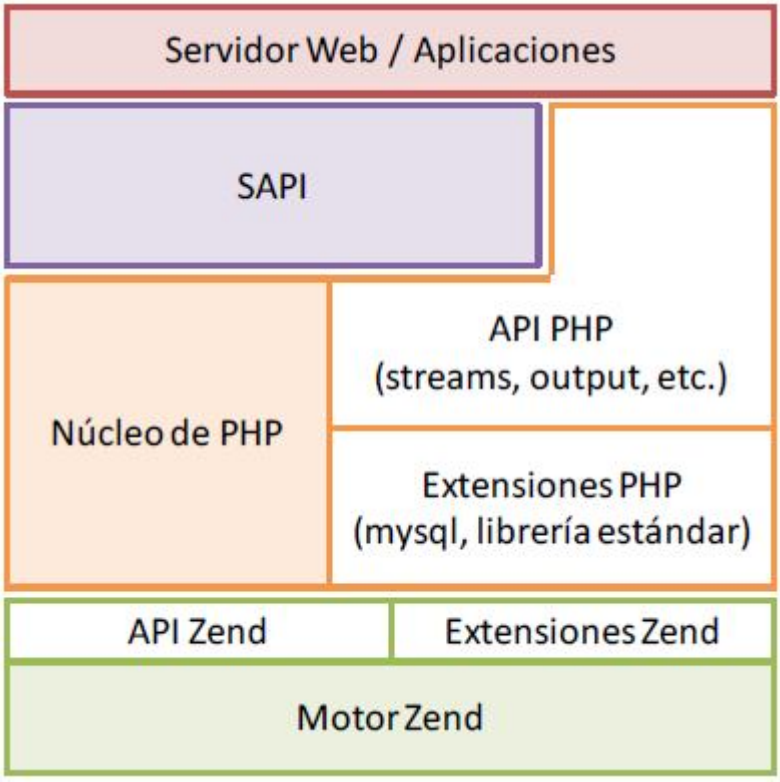
- **Apache Server.** Se trata de un servidor HTTP diseñado para ser utilizado en múltiples plataformas y sistemas operativos.
- **Microsoft IIS**. Es el servidor web de Microsoft©.
- **Sun Java System Web Server**. Se trata de un servidor web de alto rendimiento, de escalabilidad masiva y seguro que ofrece contenido dinámico y estático.
- **Ngnix**. Es un servidor HTTP que ha ganado cuota de mercado en los últimos años (de < 1% en 2007 a casi el 10% a finales de 2011).
- **Lighttp**. Es un servidor web especializado para entornos en los que se requieren respuestas rápidas.

### 2.2.- Obtención de código enviado al cliente.

Las principales características del lenguaje PHP (Hypertext Preprocessor) son:

- Lenguaje de scripting.
- De propósito general y de código abierto.
- Especialmente diseñado para el desarrollo de aplicaciones web.
- Puede ser embebido (intercalado) en código HTML.

Componentes de la Arquitectura genérica de PHP:




- **Servidor web.** Recibe las peticiones que el cliente hace a una dirección determinada.
- **Capa SAPI (Server AbstractionAPI).** Donde PHP interactúa con el servidor u otras aplicaciones.

- **Núcleo de PHP**. Se encarga de la configuración del entono de ejecución php.ini) y carga de extensiones.
- **Motor Zend.** Componente encargado del análisis y ejecución del código PHP.

Proceso de ejecución de PHP con Zend:

1.- El script pasa por un analizador léxico (lexer) y un analizador sintáctico (parser).
2.- Se genera un conjunto de instrucciones (o código intermedio).
3.- El Motor Zend ejecuta una por una las instrucciones del código intermedio.

## 2.3.- Etiquetas para la inserción de código.

La programación en entorno servidor, el caso de los lenguajes embebidos, se centra en la inserción de código propio en páginas HTML.

Uso del PHP dentro de una página.

```
<?php
    Instrucciones;
?>
```

Terminación de las líneas de código PHP.
Las líneas de código PHP terminan en punto y coma ;

Comentarios.
Una línea //
Multilinea /* */

## 2.4.- Variables.

Permiten almacenar datos temporalmente. Empiezan con $ seguida de un guión bajo o una letra y después números, letras y guiones bajos. No pueden llevar espacio y distingue entre mayúsculas y minúsculas.

Variables predefinidas. $_SERVER['variable']

| Variable | Significado |
|---|---|
| SERVER_NAME | Indica el nombre del equipo servidor sobre el que se ejecuta el *script*. |
| SERVER_PORT | Indica el puerto del equipo servidor que usa el servidor Web para la comunicación. |

| Variable | Significado |
|---|---|
| SERVER_SOFTWARE | Indica qué *software* está siendo utilizado en el equipo servidor. |
| REMOTE_PORT | Contiene el puerto que utiliza el peticionario para comunicarse con el servidor Web. |
| REMOTE_ADDR | Contiene la dirección remota desde la que se realiza la petición. |
| DOCUMENT_ROOT | Indica el directorio raíz del documento bajo el que se ejecuta el *script*. |
| HTTP_REFERER | Contiene la dirección de la página (en caso de haberla) desde la que el navegador saltó a la página actual. |

Tipos de Datos.

- Enteros $edad = 35;
- Decimales $sueldo = 1245.56;
- Cadenas. Entre comillas simples ‘ o comillas dobles “ $lenguaje="PHP";
  Para mostrar comillas o $ dentro de una cadena utilizar \ o utilizar el tipo de comillas contrario.
- Array. Puede contener datos de diferentes tipos y empiezan en la posición 0 del array. Se puede crear una matriz asociativa enumerando los índices de la misma.

  Ejemplos:

```
$matriz[0] = 10;
$matriz[1] = ‘Hola’;
$matriz[2] = 324.34;

$matriz2[‘nombre’] = ‘Ana’;
$matriz2[‘edad’] = 25;
$matriz2[‘peso’] = 60.7;
```

Funciones para Variables.

`gettype (variable)`: Devuelve el tipo de dato pasado como parámetro, pudiendo devolver como valor: `integer`, `float`, `string`, `array`, `class`, `object` y `unknown type`.

`settype (variable, tipo)`: Establece el tipo de dato a guardar en una variable. Tiene dos argumentos: el nombre de la variable y el tipo que se quiere establecer. Con esta función podemos, pues, realizar conversiones de un tipo de datos a otro. Devolverá valor `true` si ha tenido éxito; en caso contrario, devolverá `false`.

`isset (variable)`: Indica si una variable ha sido inicializada con un valor, en cuyo caso devuelve `true` y, en caso contrario, devuelve `false`.

`unset(variable)`: Destruye una variable liberando los recursos dedicados a dicha variable. Es necesario indicar como parámetro el nombre de la variable a destruir.

`empty(variable)`: Devuelve valor `true` si la variable aún no ha sido inicializada, o bien, tiene un valor igual a 0 o es una cadena vacía y, en caso contrario, devuelve `false`.

`is_int(variable)`, `is_integer(variable)`, `is_long(variable)`: Estas funciones devuelven `true` si la variable pasada como argumento es de tipo `integer`; en caso contrario, devuelven `false`.

`is_float(variable)`, `is_real(variable)`, `is_double(variable)`: Estas funciones devuelven `true` si la variable pasada como argumento es de tipo `float`; en caso contrario, devuelven `false`.

`is_numeric(variable)`: Esta función devuelve `true` si la variable pasada como argumento es un número o una cadena numérica; en caso contrario, devuelve `false`.

`is_bool(variable)`: Esta función devuelve `true` si la variable pasada como argumento es de tipo lógico; en caso contrario, devuelve `false`.

`is_array(variable)`: Esta función devuelve `true` si la variable pasada como argumento es de tipo *array*; en caso contrario, devuelve `false`.

`is_string(variable)`: Esta función devuelve `true` si la variable pasada como argumento es de tipo *string*; en caso contrario, devuelve `false`.

`is_object(variable)`: Esta función devuelve `true` si la variable pasada como argumento es de tipo `object`; en caso contrario, devuelve `false`.

**Constantes.**

Una constante es una variable que mantiene el mismo valor durante toda la ejecución del programa. Para hacer referencia a la constante no es necesario anteponer el símbolo $.

`define(constante, valor)`: Esta función nos permite crear una constante asignándole su nombre y su valor a través de los parámetros que recibe.

`defined(constante)`: Esta función devuelve `true` si la constante pasada como argumento está definida y, por tanto, existe; en caso contrario, devuelve `false`.

| Constante | Significado |
|---|---|
| PHP_VERSION | Cadena que representa la versión del intérprete de PHP en uso. |
| PHP_OS | Cadena con el nombre del sistema operativo en el que se está ejecutando el intérprete de PHP. |

| Constante | Significado |
|---|---|
| TRUE | Verdadero. |
| FALSE | Falso. |
| E_ERROR | Información sobre errores distintos a los de interpretación del cual no es posible recuperarse. |
| E_PARSE | Informa que el intérprete encontró una sintaxis inválida en el archivo de comandos. Finaliza la ejecución. |
| E_NOTICE | Informa que se produjo algo incorrecto que puede provenir o no de un error. La ejecución continúa. |
| E_WARNING | Denota un error que no impide que continúe la ejecución. |
| E_ALL | Conjunto con todos los errores que se han producido. |

**Operadores.**

- **Operadores Aritméticos.**

| Operador | Ejemplo | Descripción |
|---|---|---|
| + | $a + $b | Suma dos operandos |
| - | $a - $b | Resta dos operandos |
| * | $a * $b | Multiplica dos operandos |
| / | $a / $b | Divide dos operandos |
| % | $a % $b | Resto de la división entera |

La prioridad de los operadores es similar a cualquier lenguaje de programación

% / * + –

- **Operadores de Asignación.**

| Operador | Ejemplo | Descripción |
|---|---|---|
| = | $a = $b | $a toma el valor de $b |
| += | $a += $b | Equivale a $a = $a + $b |
| -= | $a -= $b | Equivale a $a = $a - $b |
| *= | $a *= $b | Equivale a $a = $a * $b |
| /= | $a /= $b | Equivale a $a = $a / $b |
| %= | $a %= $b | Equivale a $a = $a % $b |
| .= | $a .= $b | Equivale a $a = $a . $b |

- **Operadores con cadenas.**

Operador **.** permite concatenar cadenas de texto.

- **Operadores de Incremento y Decremento**

| Operador | Ejemplo | Descripción |
|---|---|---|
| ++ | ++$a | **Preincremento**: Incrementa $a en uno y después devuelve $a |
| | $a++ | **Postincremento**: Devuelve $a y después incrementa en uno $a |
| -- | --$a | **Predecremento**: Decrementa $a en uno y después devuelve $a |
| | $a-- | **Posdecremento**: Devuelve $a y después decrementa en uno $a |

- **Operadores Lógicos**

| Oper. | Ejemplo | Devuelve TRUE cuando |
|---|---|---|
| && | $a && $b | $a y $b son ambos true. |
| and | $a and $b | |
| \|\| | $a \|\| $b | $a o $b son true. |
| or | $a or $b | |
| ! | ! $a | $a es false, niega el valor lógico de la variable. |
| xor | $a xor $b | $a es true o $b es true, pero no lo son los dos a la vez. |

**Sentencia echo.**

Como en la mayoría de los lenguajes de scripting, en php tenemos una sentencia que nos permite "mostrar" algo en pantalla. Imprimir algo. Probemos el siguiente código en un archivo php...

```
<?php

   echo "Bienvenido al tutorial de programación en php";

?>
```

Imprime en la pagina, o en el explorador la frase "Bienvenido al tutorial de programación en php". Así sin más, simplemente una frase, sin formato o color.

```
<?php

        echo "<font color=red><b>Bienvenidos al tutorial de php</b></font>";

?>
```

Esto mostraría la frase "Bienvenidos al tutorial de php" en color rojo y en negrita. Esto quiere decir que podemos usar html para mostrar lo que hacemos con nuestro código php.

La sentencia **var_dump** imprime el tipo de variable y el contenido.

Ejemplo:

```
$variable = 15;
var_dump($variable);
```

mostrará **int 15**

### 2.5.- Ámbito de las variables

El ámbito de las variables es la zona del programa en la que la variable puede ser accedida.

- **Ámbito local:** las variables internas a una función única y exclusivamente pueden ser utilizadas dentro de dicha función.
- **Ámbito global:** la Sentencia "global" declara que una variable dentro de una función es la misma que la variable que hemos utilizado (o utilizaremos) fuera de esa función. La utilización de variables globales sin control puede resultar en un código difícil de mantener y propenso a dar errores.

  Ejemplo:

```
global  $nombre;
$nombre = "Luis";
```

### 2.6.- Dar Formato a valores numéricos.

Podemos dar formato a valores numéricos utilizando la función **number_format**:

```
number_format(variable,pos_decimales,"separ_decimal"," separ_miles")
```

**Ejemplo:**

```
$sueldo = 1200.7678;
echo number_format($sueldo,2,",",".")." &euro;<br>";
```