Desarrollo web en entorno servidor

Consulte nuestra página web: **www.sintesis.com**
En ella encontrará el catálogo completo y comentado

Queda prohibida, salvo excepción prevista en la ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de la propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sigs. Código Penal). El Centro Español de Derechos Reprográficos (www.cedro.org) vela por el respeto de los citados derechos.

Desarrollo web en entorno servidor

Xabier Ganzábal García

EDITORIAL SINTESIS

Asesor editorial:

Juan Carlos Moreno Pérez

© Xabier Ganzábal García

Asesor editorial: Juan Carlos Moreno Pérez

© EDITORIAL SÍNTESIS, S. A.
Vallehermoso, 34. 28015 Madrid
Teléfono 91 593 20 98
http://www.sintesis.com

ISBN: 978-84-9171-349-4
Depósito Legal: M-11.684-2019

Impreso en España - Printed in Spain

Reservados todos los derechos. Está prohibido, bajo las sanciones penales y el resarcimiento civil previstos en las leyes, reproducir, registrar o transmitir esta publicación, íntegra o parcialmente, por cualquier sistema de recuperación y por cualquier medio, sea mecánico, electrónico, magnético, electroóptico, por fotocopia o por cualquier otro, sin la autorización previa por escrito de Editorial Síntesis, S. A.

Índice

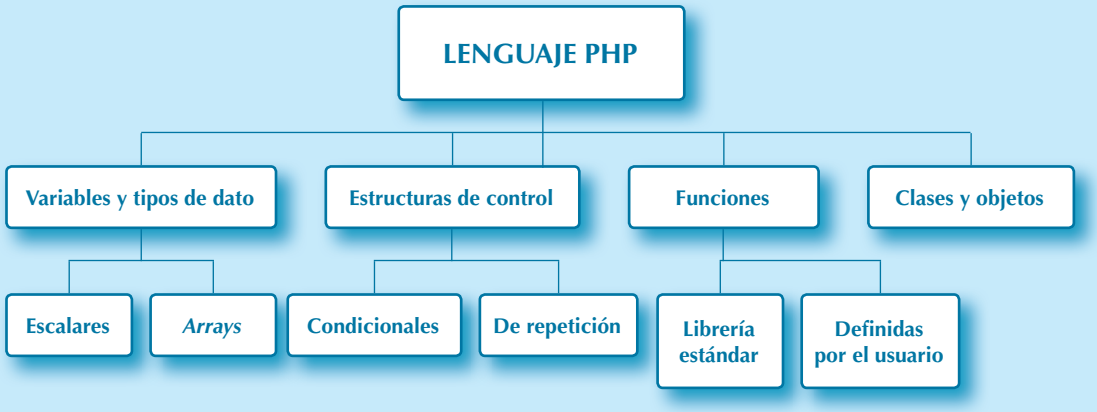
2

Introducción al lenguaje PHP

Objetivos

- ✓ Conocer la sintaxis básica de PHP.
- ✓ Entender cómo se integran PHP y HTML.
- ✓ Describir los tipos de datos existentes en PHP.
- ✓ Manejar las estructuras de control básicas.
- ✓ Aprender a utilizar los *arrays* asociativos.
- ✓ Presentar la notación de objetos en PHP.

Mapa conceptual



Glosario

Array. Es un tipo de dato compuesto habitual en los lenguajes de programación. Aunque los detalles varían entre lenguajes, en general, se parecen a vectores o listas ordenadas.

Bucle. Estructura de programación que permite repetir instrucciones.

Estructura condicional. Posibilita ejecutar o no una instrucción según se cumpla una condición.

Función. Conjunto de instrucciones que realiza una tarea concreta.

Librería. Las funciones relacionadas entre sí se agrupan en librerías o bibliotecas.

Programación orientada a objetos. Paradigma de programación basado en la idea de objetos, elementos que agrupan variables y funciones.

Script. Programa sencillo, habitualmente ejecutado por un intérprete en lugar de ser compilado.

Variable. Posición en la memoria del ordenador identificada por un nombre. La variable almacena datos. Estos datos son el valor de la variable.

2.1. PHP y HTML. Código incrustado

PHP es el lenguaje de programación para desarrollo web en el lado del servidor. Desde su aparición en 1994 ha tenido gran aceptación y se puede decir que es lenguaje más extendido para el desarrollo en el lado del servidor. Aunque no es la única opción, lo normal es que el intérprete de PHP sea un módulo del servidor web.

El lenguaje PHP es flexible y permite programar pequeños *scripts* con rapidez. Comparado con lenguajes como Java, requiere escribir menos código y, en general, resulta menos engorroso. La sintaxis de los elementos básicos es bastante parecida a la de lenguajes muy extendidos como Java y C. Por estos motivos, es un lenguaje rápido de aprender para las personas con alguna experiencia en programación.

En este capítulo se presentan la sintaxis y los elementos básicos del lenguaje PHP. Se espera que el lector esté familiarizado con los conceptos básicos de programación estructurada y orientada a objetos.

En el desarrollo web es muy habitual utilizar PHP incrustado dentro ficheros HTML. El código PHP se introduce dentro del HTML utilizando la etiqueta `<?php` para abrir el bloque de PHP y la etiqueta `?>` para cerrarlo.

El ejemplo **hola_mundo.php** muestra una página HTML completa con un bloque PHP incrustado en las líneas 7-10. El bloque tiene una única sentencia que sirve para mostrar la cadena "Hola mundo".

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Hola mundo</title>
5      </head>
6      <body>
7          <?php
8              echo "Hola mundo";
9          ?>
10     </body>
11 </html>
```

Al solicitar la página al servidor web, el resultado es:

Hola mundo

Si se consulta el código fuente de la página (pulsando Ctrl + u), se obtiene:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hola mundo</title>
    </head>
    <body>
        Hola mundo
    </body>
</html>
```

Se puede observar que el servidor web ha modificado una parte del fichero. El bloque PHP ha desaparecido y en su lugar aparece "Hola mundo". El servidor web procesa los bloques de PHP y los sustituye por su salida, es decir, por lo que muestran las sentencias del bloque. En este caso se utiliza `echo`, que muestra el valor de una variable o de una cadena de texto.

TOMA NOTA

Para probar los ejemplos del libro hay que instalar el entorno de trabajo como se explica en el capítulo 1. Para probar este ejemplo accede a http://localhost/cap2/hola_mundo.php. Para el resto de los ejemplos solo hay que cambiar el nombre del fichero. Recuerda arrancar el servidor web.

2.2. Sintaxis de PHP

El elemento básico en PHP es el bloque. Un bloque PHP está delimitado por las etiquetas correspondientes y contiene una serie de sentencias separadas por punto y coma.

```
<?php
    sentencia1;
    sentenciaN;
? >
```

RECUERDA

✓ Cuando un fichero contiene solo PHP se recomienda no cerrar la etiqueta del último bloque. Puede dar problemas si la respuesta del servidor involucra varios ficheros.

2.3. Variables y tipos de dato

Una de las características principales de PHP es que es un lenguaje *no fuertemente tipado*. Esto quiere decir que no es necesario indicar el tipo de dato al declarar una variable. De hecho, las variables no se declaran, se crean la primera vez que se les asigna un valor. El tipo de dato depende del valor con que se inicialicen.

Esto agiliza la escritura de programas, pero también tiene inconvenientes. Si no se presta atención, puede dar lugar a código de baja de calidad y, a medida que las aplicaciones crecen, pueden darse errores difíciles de depurar.

2.3.1. Declaración de variables

En PHP los identificadores de las variables van siempre precedidos por el carácter `'$'`. El identificador de la variable debe comenzar por una letra o un guion bajo (`'_'`), y puede estar formado por números, letras y guiones bajos.

Para declarar una variable, solo hay que asignarle un valor:

```
$nombre = valor;
```

Por ejemplo, esta sentencia declara la variable `$entero`, que será de tipo `integer` porque se inicializa con un entero.

```
$entero = 4;
```

También es posible cambiar el tipo de dato de una variable simplemente asignándole un valor de otro tipo de dato, como se puede ver en el ejemplo **tipos_dato.php** (líneas 8-12). El ejemplo utiliza la función `gettype()`, que devuelve el tipo de dato de una variable.

```
1  <?php
2  /* declaración de variables */
3  $entero = 4; // tipo integer
4  $numero = 4.5; // tipo coma flotante
5  $cadena = "cadena"; // tipo cadena de caracteres
6  $bool = TRUE; //tipo booleano
7  /* cambio de tipo de una variable */
8  $a = 5; // entero
9  echo gettype($a); // imprime el tipo de dato de a
10 echo "<br>";
11 $a = "Hola"; // cambia a cadena
12 echo gettype($a); // se comprueba que ha cambiado
```

La salida del ejemplo confirma que la variable cambia de tipo de dato.

```
integer
string
```

2.3.2. Asignación por copia y por referencia

En principio, la asignación de variables se realiza mediante copia. Es decir, si hacemos:

```
$a = $b;
```

se crea una nueva variable *a* y se le asigna el valor que tenga *b*. Las variables *a* y *b* representan posiciones diferentes de memoria, aunque tengan el mismo valor después de la asignación.

También es posible definir una referencia a una variable utilizando el operador *ampersand*:

```
$var2 = &$var1;
```

En este caso `$var2` no es una nueva variable con el valor de `$var1`. Por el contrario, `$var2` apunta a la misma dirección de memoria que `$var1`, de manera que `$var1` y `$var2` son en realidad dos nombres para el mismo dato. En el ejemplo **copia.php** se muestra que, al cambiar el valor de una referencia, se modifica también el de la variable referenciada.

```
<?php
    $var1 = 100;
    $var2 = &$var1; // asignación por referencia
```

```
$var3 = $var1; // asignación por copia
echo "$var2<br>";// muestra 100
$var2 = 300; // cambia el valor de $var2
echo "$var1<br>";// $var1 también cambia
$var3 = 400; // este cambio no afecta a $var1
echo $var1;
```

La salida de este ejemplo será:

```
100
300
300
```

2.3.3. Variables no inicializadas

Si se intenta utilizar una variable antes de asignarle un valor, se genera un error de tipo E_NOTICE, pero no se interrumpe la ejecución del *script*. Si una variable no inicializada aparece dentro de una expresión, dicha expresión se calcula tomando el valor por defecto para ese tipo de dato. En el ejemplo **no_init.php**, se puede ver lo que ocurre al utilizar una variable no inicializada dentro de una expresión.

```
1  <?php
2  $var1 = 100;
3  $var3 = 100 + $var2;//$var2 no existe, se toma como 0
4  echo "$var3 <br>"; // muestra 100
5  $var3 = 100 * $var2; // $var2 no existe, se toma como 0
6  echo "$var3 <br>"; // muestra 0
```

En la línea 3, se intenta sumar una variable no inicializada. Como el valor por defecto para un entero es 0, el resultado de la suma es 100. En la línea 5, se intenta multiplicar por una variable no inicializada y, al multiplicar por 0, el resultado es 0.

La salida muestra un mensaje de error por cada intento de utilización de una variable no inicializada.

```
Notice: Undefined variable: var2 in C:\xampp\htdocs\cap2\ejeinicializaciones.php on line 3
100
Notice: Undefined variable: var2 in C:\xampp\htdocs\cap2\ejeinicializaciones.php on line 5
0
```

2.3.4. Constantes

Para definir constantes se utiliza la función define(), que recibe el nombre de la constante y el valor que queremos darle.

```
define("LIMITE", 1000);
```

Es habitual utilizar identificadores en mayúsculas para las constantes.

2.3.5. Tipos de datos escalares

PHP ofrece cuatro tipos de datos escalares: *integer*, *float*, *boolean* y *string*.

A) Números

Para representar números enteros se usa el tipo de dato *integer*. El tamaño y los valores máximo y mínimo de un entero dependen de la plataforma, y se pueden conocer mediante los constantes PHP_INT_SIZE, PHP_INT_MAX y PHP_INT_MIN, respectivamente.

Para números reales, se utiliza el tipo *float*. El tamaño también depende de la plataforma, pero suele ofrecer una precisión de 14 decimales. En cualquier caso, la precisión de los *float* presenta los mismos problemas que en otros lenguajes y los redondeos pueden dar sorpresas.

La conversión entre *integer* y *float* es automática. Si se recibe un *float* cuando se espera un *integer*, se trunca. Si al realizar una operación sobre un entero el resultado supera los valores límite o tiene decimales, se convierte a *float*. También se pueden utilizar los operadores de conversión, `(int)` o `(float)`.

El ejemplo **tipos_numericos.php** muestra las diferentes opciones disponibles para literales *integer* o *float* y algunas operaciones entre ellos.

```
<?php
    echo PHP_INT_SIZE.'<br>';
    echo PHP_INT_MIN.'<br>';
    echo PHP_INT_MAX.'<br>';
    $a = 0b100; // en binario
    $a = 0100; // octal
    $a = 0x100; // hexadecimal
    $a = 3/2; // la división entre enteros no da problemas
    echo $a.'<br>'; // 1.5
    $b = 7.5;
    $a = (int)$b; //casting a int
    echo $a.'<br>'; // 7, se trunca
    $b = 7e2; // notación científica
    $b = 7E2;
```

B) Cadenas

El tipo de dato *string* permite almacenar cadenas de caracteres. Para delimitar una cadena es posible utilizar comillas simples o dobles, pero hay una diferencia. Si se utilizan comillas dobles (también llamadas *comillas mágicas*), las variables que aparezcan dentro de la cadena se sustituirán por su valor. Las comillas dobles son muy prácticas, ya que es más rápido insertar directamente las variables que montar las cadenas con varias concatenaciones.

```
<?php
    $var = "Paco";
    $a = "Hola $var <br>";
    $b = 'Hola $var';
    echo $a;
    echo $b;
```

La salida será:

```
Hola Paco
Hola $var
Hola Paco
```

En el primer caso, `$var` se sustituye por su valor, "Paco". En el segundo, se interpreta como texto normal. La última línea del *script* muestra el operador de concatenación entre cadenas, `"."`.

TOMA NOTA

A la hora de formatear la salida hay que tener en cuenta que esta va a ser procesada como HTML por un navegador web, es decir, los saltos de línea se ignoran y los espacios en blanco consecutivos colapsan. Los caracteres de escape como '\n', '\r' y '\t' son ignorados por el navegador. Para introducir un salto de línea la opción más sencilla es incluir la etiqueta de salto de línea de HTML ("
") en la salida, como se puede ver en los ejemplos anteriores.

C) Booleanos

Este tipo de dato es para variables *booleanas*. Solo pueden tomar los valores TRUE y FALSE, verdadero y falso. Este es el tipo de dato que se obtiene, entre otros casos, como resultado de los operadores de comparación y se utiliza en sentencias condicionales y bucles.

Cuando se espera un valor *booleano* y se recibe otro tipo de dato, se aplican las reglas de conversión recogidas en el cuadro 2.1.

CUADRO 2.1
Conversión implícita a *boolean*

Tipo	Valor como *boolean*
integer	Si es 0 se toma FALSE, en otro caso como TRUE
float	Si es 0.0 se toma FALSE, en otro caso como TRUE
string	Si es una cadena vacía o "0", se toma como FALSE, en otro caso como TRUE
variables no inicializadas	FALSE
null	FALSE
array	Si no tiene elementos se toma FALSE, en otro caso como TRUE

D) Otros tipos de dato

Además de los tipos de datos dato escalares en PHP también existen los siguientes tipos de datos:

1. *array*. Para representar colecciones de elementos. Los *arrays* de PHP son muy potentes y se explican con detalle en el apartado 2.2.5.
2. *object*, PHP tiene soporte completo para la programación orientada a objetos. Se explica en el apartado 2.2.8.
3. *callable*. Un tipo de dato especial para representar funciones de *callback*, funciones que se pasan a otras funciones.
4. *null*. El tipo de dato *null* representa una variable que no ha sido asignada. Solo puede tomar un único valor, NULL. Se considera que una variable es de tipo *null* si se le asigna el valor NULL o no está inicializada.
5. *resource*. Este tipo de dato representa recursos externos, como una conexión a una base de datos.

Cuadro 2.2
Tipos de dato en PHP

Tipo	Descripción
integer	Números enteros
float	Números reales en coma flotante
string	Cadenas de caracteres
boolean	Booleanos, TRUE o FALSE
array	Colección de elementos identificados
object	Un objeto es una instancia de una clase
callable	Para las funciones de *callback*
null	Para representar variables no asignadas
resource	Representa recursos externos

2.3.6. Ámbito de las variables

El ámbito de una variable es la parte del código en que esta es visible. Una variable declarada en un fichero PHP está disponible en ese fichero y en los ficheros que se incluyan desde este.

Por otro lado, las funciones definen un ámbito local, de manera que las variables que se declaran en las mismas solo son accesibles desde la propia función. Además, desde la función no se puede acceder a otras variables que no sean las locales o sus argumentos.

Para definir variables globales hay dos opciones. La palabra reservada `global` y la variable predefinida `$_GLOBALS`. Las variables globales son accesibles desde cualquier función o fichero de la aplicación.

2.3.7. Variables predefinidas

En PHP hay muchas variables predefinidas disponibles. Contienen información sobre el servidor, datos enviados por el cliente o variables de entorno. Dentro de las variables predefinidas

hay un grupo de ellas, las *superglobales*, que están disponibles en cualquier ámbito. Cada una de ellas guarda información de un tipo.

Por ejemplo, en `$_SERVER` hay información sobre el servidor en el que está alojada la página. El *script* **global_server.php** muestra algunos de los datos disponibles.

```
<?php
    echo "Ruta dentro de htdocs: ". $_SERVER['PHP_SELF'];
    echo "Nombre del servidor: ". $_SERVER['SERVER_NAME'];
    echo "Software del servidor: ". $_SERVER['SERVER_SOFTWARE'];
    echo "Protocolo: ". $_SERVER['SERVER_PROTOCOL'];
    echo "Método de la petición: ". $_SERVER['REQUEST_METHOD'];
```

Las variables *superglobales* (cuadro 2.3) son muy relevantes para el desarrollo de aplicaciones web y se irán poniendo en práctica a lo largo del libro.

CUADRO 2.3
Variables *superglobales*

Nombre	Descripción
$GLOBALS	Variables globales definidas en la aplicación
$_SERVER	Información sobre el servidor
$_GET	Parámetros enviados con el método GET (en la URL)
$_POST	Parámetros enviados con el método POST (formularios)
$_FILES	Ficheros subidos al servidor
$_COOKIE	*Cookies* enviadas por el cliente
$_SESSION	Información de sesión
$_REQUEST	Contiene la información de $_GET, $_POST y $_COOKIE
$_ENV	Variables de entorno

2.4. Comentarios

En PHP se pueden utilizar comentarios:

- De bloque, encerrados entre "`/*`" y "`*/`".
- De línea, comenzando por "`//`" o por "`#`".

```
<?php
    //comentario de línea
    $a = 0; // comentario de línea
    /* comentario
    de bloque
    */
    # comentario de una sola línea
```

Como en cualquier lenguaje, comentar adecuadamente el código se considera una buena práctica de programación.

2.5. Estructuras de control

PHP cuenta con las estructuras de control habituales en la programación estructurada para realizar sentencias condicionales y de repetición. La sintaxis es muy parecida a la de Java o C.

2.5.1. Estructuras condicionales

Las estructuras condicionales de PHP son `if`, `if-else`, `if-elseif` y `switch`.

La sentencia condicional más sencilla es la estructura `if`. La sintaxis general es:

```
if (condición)
    instrucción
```

Si se cumple la condición, se ejecuta la instrucción. Si no se cumple, no se ejecuta. Para agrupar dentro del *if* más de una sentencia, se encierran entre llaves.

```
if (condición){
    instrucción 1
    instrucción n
}
```

La condición es una expresión que se evalúa a verdadero a falso, siguiendo las normas de conversión a *boolean* si es necesario.

En el siguiente ejemplo solo se cumple el segundo `if` y, por tanto, la salida del programa sería "Es mayor que cero".

```
<?php
    $var = 3;
    if($var < 0) echo "Es menor que cero";
    if ($var > 0){
        echo "Es mayor que cero";
    }
```

Cuando se utiliza un `if` se puede añadir un `else`. Las instrucciones dentro del `else` solo se ejecutan cuando la condición del `if` no se cumple.

```
<?php
    $var = 3;
    if($var < 0){
        echo "Es menor que cero";
    }else{
        echo "Es mayor o igual que cero";
    }
```

Si se anidan varias sentencias condicionales, se puede usar `elseif`, que es equivalente a `else if`.

```
<?php
    $var = 3;
    if ($var == 1) {
        echo "Es un uno";
    }elseif ($var == 2) {
        echo "Es un dos";
    }elseif ($var == 3) {
        echo "Es un tres";
    }else{
        echo "No es un uno, ni un dos, ni un tres"
    }
```

La primera condición que se cumpla es la que se ejecuta. Si no se cumple ninguna, se ejecuta el `else` final (si lo hay).

Para agrupar varios `if` puede ser útil la estructura `switch`, también habitual en otros lenguajes. El ejemplo switch.php es equivalente al anterior, pero más fácil de leer.

```
<?php
    $var = 3;
    switch($var){
        case 1:
            echo "Es un 1";
        break;
        case 2:
            echo "Es un 2";
        break;
        case 3:
            echo "Es un 3";
        break;
        default:
            echo "No es un 1, ni un 2, ni un 3";
    }
```

Según el valor de `$var`, se ejecutará un `case` u otro. La sección `default` se ejecuta cuando no se da ninguno de los `case`.

RECUERDA

✓ Si no hay un *break* al final de un *case*, la ejecución continúa con el siguiente.

2.5.2. Estructuras de repetición

Las estructuras de repetición o bucles sirven para repetir un conjunto de instrucción mientras se dé una condición. PHP cuenta con las estructuras habituales: `for`, `while` y `do-while`, que tienen la misma sintaxis que en Java o C.