50 ANIVERSARIO

IES ANTONIO SEQUEROS ALMORADÍ

TEMA 1

Php

2020/2021

Versión: 200913.2108

IES Antonio Sequeros

ÍNDICE

1. QUÉ ES PHP

PHP es un lenguaje de programación dirigido a la creación de páginas web. Es un lenguaje de programación procedural con una sintaxis similar a la del lenguaje C, aunque actualmente puede utilizarse una sintaxis de programación orientada a objetos similar a la de Java.

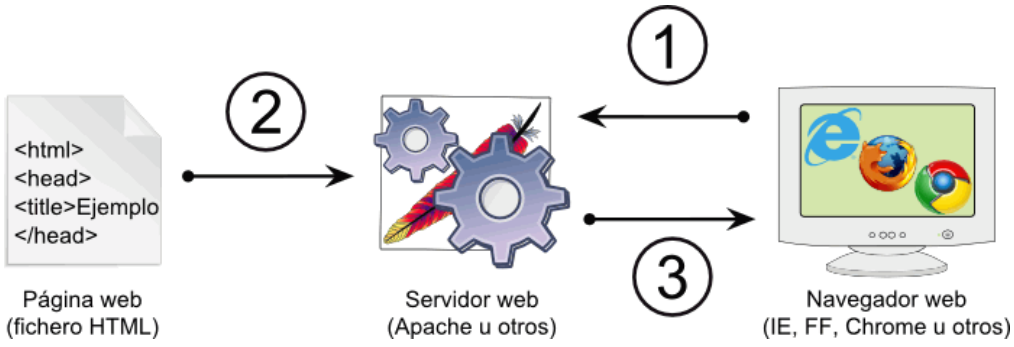
1.1. Páginas web, lenguajes de programación y bases de datos

La World Wide Web (o simplemente, la web) fue ideada por Tim Berners Lee en 1990, empezó a funcionar en el CERN en 1991 y se extendió rápidamente por las Universidades del mundo (en aquel entonces Internet era una red a la que sólo tenían acceso entidades gubernamentales, particularmente entidades educativas). En 1992 algunos proveedores comerciales empezaron a dar acceso a particulares y empresas, lo que convirtió la web en una red de comunicación universal que ha transformado nuestras vidas.

Páginas Estáticas o Dinámicas

En sus primeros años, las páginas web solían ser documentos de texto guardado en algún directorio de un servidor y a las que se accedía mediante los primeros navegadores web. Cada página web que se veía en el navegador correspondía a un fichero en el servidor.

La imagen siguiente ilustra de forma simplificada el esquema de funcionamiento:



El usuario escribe la dirección de la página web en su navegador

- (1) El navegador la solicita al servidor web correspondiente (este paso requiere la participación de máquinas intermedias que no se comentan aquí)
- (2) El servidor lee el fichero que corresponde a esa página web
- (3) El servidor envía el fichero al navegador
- El navegador muestra la página web al usuario

Este esquema de funcionamiento es suficiente para sitios web pequeños creados por una sola persona, pero en cuanto un sitio web empieza a crecer, empiezan a surgir los problemas. Por ejemplo:

- ✓ si el sitio contiene muchas páginas es necesario crear menús que permitan orientarse por el sitio. Como cada página debe contener el menú, cualquier cambio en el menú obliga a modificar todas las páginas.
- ✓ si el sitio modifica a menudo su contenido (por ejemplo, la web de un periódico), tener que editar manualmente los ficheros ralentiza el proceso.
- ✓ si el sitio es creado por varias personas, cualquiera puede modificar por error los ficheros de otras personas y si varias personas quieren modificar el mismo fichero se pueden producir conflictos.

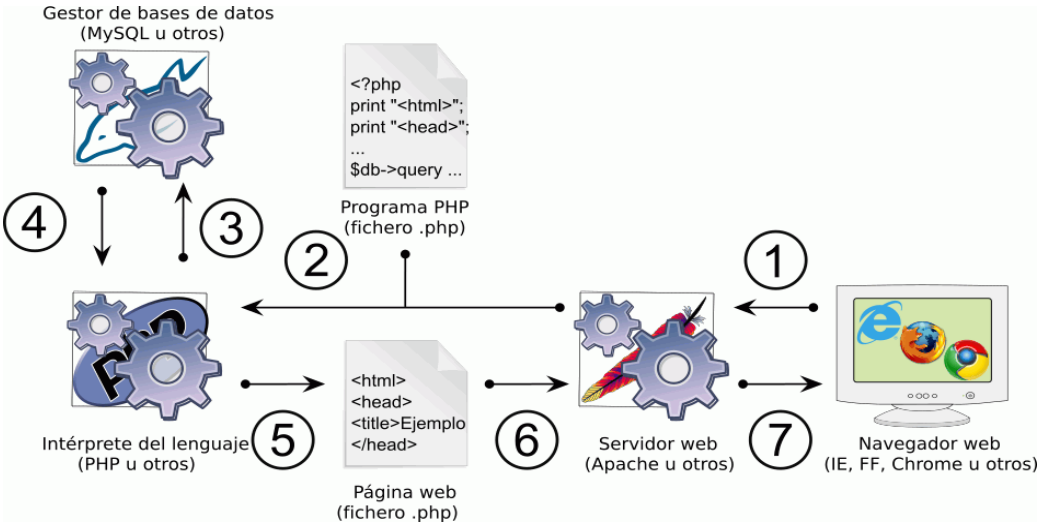
- ✓ si se quiere permitir la participación del público (foros, comentarios en blogs, etc), esta participación depende de que uno de los autores modifique los ficheros, lo que ralentiza el proceso.
- ✓ si personas sin conocimientos técnicos quieren participar en la creación del sitio, pueden cometer errores editando los ficheros.

La solución es que las páginas no sean ficheros estáticos guardados en el disco, sino que las páginas se generen automáticamente cuando el usuario las solicite.

Por ejemplo, para resolver el problema de los menús comentado anteriormente, una solución sería que el menú estuviera en un fichero aparte y cuando el usuario solicitara una página, el menú se añadiera al principio de cada página (nota: no me refiero a la composición de páginas mediante frames, una solución desaconsejada hace muchos años, sino a componer la página web a partir de varios ficheros).

Esa generación de las páginas se puede hacer de varias maneras. Una de ellas es recurrir a lenguajes de programación generales o específicos (como PHP). Desde sus inicios, los servidores web permiten recurrir a lenguajes de programación para generar las páginas web.

La imagen siguiente ilustra de forma simplificada el esquema de funcionamiento:



El usuario escribe la dirección de la página web en su navegador

- (1) El navegador la solicita al servidor web correspondiente (este paso requiere la participación de máquinas intermedias que no se comentan aquí)
- (2) El servidor detecta que es un programa PHP y lo envía al intérprete del lenguaje
- (3) El intérprete del lenguaje ejecuta el programa. Si el programa incluye consultas a la base de datos, estas se realizan.
- (4) La base de datos entrega al intérprete el resultado de las consultas
- (5) El intérprete del lenguaje completa la ejecución del programa.
- (6) El resultado final del programa (por ejemplo, el código fuente de una página web) se envía al servidor
- (7) El servidor envía el fichero al navegador
- El navegador muestra la página web al usuario

Como antes, el usuario no puede saber si se ha accedido o no a un sistema gestor de bases de datos. El navegador recibe un documento de texto que contiene etiquetas html y lo muestra al usuario.

2. EMPEZANDO CON PHP

2.1. Estructura de una página PHP

Una página PHP es un archivo de texto que contiene uno o varios fragmentos de código PHP y que también puede contener fragmentos de código HTML.

Los fragmentos de código PHP están delimitados por las etiquetas <?php (etiqueta inicial) y ?> (etiqueta final) y contienen las instrucciones de PHP. Más adelante se comentan otros posibles delimitadores de fragmentos de código PHP.

`<?php`
`print "<p>Hola</p>\n";`
`?>`
	`<p>Hola</p>`

Los fragmentos de PHP no pueden anidarse, es decir, no puede haber un fragmento dentro de otro:

`<?php`
`print "<p>Hola</p>\n";`
 	
`<?php`
`print "<p>Adios</p>\n";`
`?>`
	`Parse error: syntax error, unexpected '<'`
	`in ejemplo.php on line 4`

Si en una misma página hay varios fragmentos PHP, se tratan como un único programa. Los siguientes programas son equivalentes:

`<?php`
`$saludo = "Hola";  // Se define una variable`
`print "<p>$saludo</p>";  // Se escribe el valor de la variable`
`?>`
	`<p>Hola</p>`

`<?php`
`$saludo = "Hola";  // Se define una variable`
`?>`
`<?php`
`print "<pre>$saludo<pre>";  // Se escribe el valor de la variable`
`?>`
	`<p>Hola</p>`

Si el programa termina con un fragmento PHP, ese último fragmento PHP no necesita cerrarse. Los siguientes programas son correctos:

`<?php` `print "<p>Hola</p>\n";`	`<p>Hola</p>`

`<?php`
`  print "<p>Hola</p>\n";`
`?>`
`<?php`
`print "<p>Adios</p>\n";`
	`<p>Hola</p>`
	`<p>Adios</p>`

Cuando un navegador solicita una página PHP a un servidor, el servidor lee el archivo secuencialmente y va generando el resultado:

- Si la página contiene fragmentos HTML, el código HTML se incluye sin modificados en el resultado.
- Si la página contiene fragmentos PHP, se ejecutan las instrucciones que se encuentran en los fragmentos de código PHP. Si esas instrucciones generan texto, el texto se incluye en el resultado.

Cuando el servidor termina de leer el archivo, el servidor envía al navegador el resultado.

Es importante señalar que:

El navegador recibe una página web (es decir, únicamente código HTML), **no recibe código PHP**.

- Los fragmentos PHP tiene que generar las etiquetas HTML que se necesiten para una correcta visualización de la página web en el navegador.
- El navegador no puede distinguir en la página recibida, qué parte ha sido generada en un fragmento PHP y qué parte se encontraba ya en un fragmento HTML.
- Como la página PHP se lee secuencialmente, el código HTML generado por los fragmentos PHP y el incluido en los fragmentos HTML se encuentran en el mismo orden en que se encontraban los fragmentos en la página PHP.

Los ejemplos siguientes muestran diferentes programas PHP que generan el mismo resultado:

- Programa con un único fragmento PHP

```
<?php
print "<p>Hola</p>\n";
print "<p>¿Cómo estás?</p>\n";
print "<p>Adios</p>\n";
?>
```

```
<p>Hola</p>
<p>¿Cómo estás?</p>
<p>Adios</p>
```

- Programa con un fragmento PHP y un fragmento HTML

```
<p>Hola</p>
<?php
print "<p>¿Cómo estás?</p>\n";
print "<p>Adios</p>\n";
?>
```

```
<p>Hola</p>
<p>¿Cómo estás?</p>
<p>Adios</p>
```

- Programa con dos fragmentos PHP y un fragmento HTML

```
<?php
print "<p>Hola</p>\n";
?>
<p>¿Cómo estás?</p>
<?php
print "<p>Adios</p>\n";
?>
```

```
<p>Hola</p>
<p>¿Cómo estás?</p>
<p>Adios</p>
```

En un fragmento PHP no pueden escribirse etiquetas HTML sueltas; el código HTML debe generarse siempre con instrucciones de PHP. El programa siguiente no puede ni siquiera ejecutarse y produce un error de sintaxis:

```
<?php
<p>Hola</p>
?>
```

```
Parse error: syntax error, unexpected '<' in
     ejemplo.php on line 2
```

De la misma manera, no se deben escribir instrucciones de PHP fuera de fragmentos PHP. No se producen errores, pero el resultado no es el esperado. Como el texto fuera de fragmentos PHP se envía tal cual

al navegador, el navegador recibirá las instrucciones de PHP, que el navegador no sabe ejecutar y se mostrarán en la pantalla.

```
<?php
print "<p>Hola</p>\n";
?>
<p>¿Cómo estás?</p>

print "<p>Adios</p>\n";
```

```
<p>Hola</p>
<p>¿Cómo estás?</p>

print "<p>Adios</p>\n";
```

2.2. Comentarios en páginas PHP

Comentarios en PHP

En un fragmento PHP se pueden comentar líneas de código utilizando:

- // para comentar el resto de la línea (como en C++)
- # para comentar el resto de la línea (como en la shell de Unix o en Perl)
- /* */ para delimitar varias líneas (como en C)

Estos comentarios no se añaden al código HTML generado por la página, por lo que no pueden verse en el navegador.

```
<p><strong>
<?php
// La instrucción print escribe texto en la página
      web
print "Hola"; // El comentario se puede escribir al
      final de la línea
?>
</strong></p>
```

```
<p><strong>Hola</strong></p>
```

```
<p><strong>
<?php
# La instrucción print escribe texto en la página web
print "Hola"; # El comentario se puede escribir al
      final de la línea
?>
</strong></p>
```

```
<p><strong>Hola</strong></p>
```

```
<?php
print "<p><strong>";
/* Dentro de un fragmento PHP no se pueden escribir
   etiquetas html sueltas, tienen que estar siempre
   incluidas en instrucciones print
*/
?>
Hola
<?php
print "</strong></p>";
?>
```

```
<p><strong>Hola</strong></p>
```

Comentarios en HTML

Si se quieren escribir comentarios en los fragmentos HTML, hay que utilizar la etiqueta de comentarios de HTML <!-- .... -->.

Estos comentarios, como todo el código HTML situado en los fragmentos HTML, se incluyen sin modificaciones en el resultado, por lo que pueden verse en el navegador.

```
<p>
<?php
print "Hola";
?>
</p>
<!-- El texto anterior ha sido generado por PHP -->
```

```
<p>Hola</p>
<!-- El texto anterior ha sido generado por PHP -->
```

Comentarios en CSS

Si se quieren escribir comentarios en las hojas de estilo CSS, hay que utilizar la etiqueta de comentarios de C /* .... */.

Estos comentarios, como todo el código CSS situado en las hojas de estilo, se incluyen sin modificaciones en el resultado, por lo que pueden verse en el navegador.

```
html {
  background-color: #FF0000; /* #FF000000 es rojo */
}
```

```
html {
  background-color: #FF0000; /* #FF000000 es rojo */
}
```

2.3. Cadenas

En PHP, las cadenas de texto se delimitan por comillas (dobles o simples).

```
<?php
print "<p>Esto es una cadena.</p>\n";
?>
```

```
<p>Esto es una cadena.</p>
```

```
<?php
print '<p>Esto es otra cadena.</p>';
?>
```

```
<p>Esto es otra cadena.</p>
```

Si las comillas de apertura y cierre no son del mismo tipo (una de ellas doble y otra simple), se produce un error de sintaxis.

```
<?php
print "<p>Esto es una cadena.</p>\n';
?>
```

```
Parse error: syntax error, unexpected end of file,
expecting variable (T_VARIABLE) or

${ (T_DOLLAR_OPEN_CURLY_BRACES)
or {$ (T_CURLY_OPEN) in ejemplo.php on line 3
```

Comillas dentro de cadenas

Si una cadena está delimitada por comillas dobles, en su interior puede haber cualquier número de comillas simples, y viceversa.

```
<?php
print "<p>Esto es una comilla simple: '</p>";
?>
```

```
<p>Esto es una comilla simple: '</p>
```

```
<?php
print '<p>Esto es una comilla doble: "</p>';
?>
```

```
<p>Esto es una comilla doble: "</p>
```

Lo que no puede haber en una cadena es una comilla del mismo tipo que las que delimitan la cadena.

<?php print " Esto es una comilla doble: " "; ?>	**Parse error: syntax error, unexpected '/' in ejemplo.php on line 2**
<?php print ' Esto es una comilla simple: ' '; ?>	**Parse error:** syntax error, unexpected '/' in **ejemplo.php** on line **2**

Para poder escribir en una cadena una comilla del mismo tipo que las que delimitan la cadena, se debe utilizar los caracteres especiales \' o \".

<?php print " Esto es una comilla simple: ' y esto una comilla doble: \" "; ?>	Esto es una comilla simple: ' y esto una comilla doble: "
<?php print ' Esto es una comilla simple: \' y esto una comilla doble: " '; ?>	Esto es una comilla simple: ' y esto una comilla doble: "

Pero los caracteres especiales \" y \' no se pueden utilizar para delimitar cadenas.

Diferencias entre comillas simples y dobles

Aunque en los ejemplos anteriores las comillas simples o dobles son equivalentes, en otras situaciones no lo son. Por ejemplo, PHP no sustituye las variables que se encuentran dentro de cadenas delimitadas con comillas simples, mientras que sí que lo hace (pero no siempre) si se utilizan comillas dobles, como se ve en el siguiente ejemplo:

<?php $cadena = "Hola"; print " La variable contiene el valor: $cadena "; ?>	La variable contiene el valor: Hola
<?php $cadena = "Hola"; print ' La variable contiene el valor: $cadena '; ?>	La variable contiene el valor: $cadena

Concatenar cadenas

El operador . (punto) permite concatenar dos o más cadenas.

3. VARIABLES

3.1. ¿Qué es una variable?

En los lenguajes de programación, una variable es un elemento que permite almacenar información. Las variables se identifican por su nombre. Para facilitar la comprensión del programa, se aconseja que los nombres de las variables hagan referencia a la información que contienen. Por ejemplo, si se va a guardar en una variable la edad del usuario, un nombre adecuado de la variable sería por ejemplo $edad.

En PHP el programador puede dar el nombre que quiera a las variables, con algunas restricciones:

- Los nombres de las variables tienen que empezar por el carácter $.
- A continuación tiene que haber una letra (mayúscula o minúscula) o un guión bajo (_).
- El resto de caracteres del nombre pueden ser números, letras o guiones bajos.

En los nombres de las variables, PHP distingue entre mayúsculas y minúsculas, es decir, si se cambia algún carácter de mayúscula a minúscula o viceversa, para PHP se tratará de variables distintas.

Si el nombre de la variable contiene varias palabras, se aconseja utilizar la notación "camel case", es decir, escribir la primera palabra en minúsculas y la primera letra de las siguientes palabras en mayúsculas.

PHP define automáticamente una serie de variables (son las llamadas variables predefinidas). Los nombres de estas variables siguen siempre el mismo patrón: empiezan por un guión bajo y se escriben en mayúsculas (por ejemplo, $_REQUEST, $_SERVER, $_SESSION, etc.). Para evitar conflictos con variables predefinidas que se creen en el futuro, se recomienda no crear en los programas variables con nombres que sigan ese mismo patrón.

3.2. Usar variables

✓ Para guardar un valor en una variable se utiliza el operador de asignación (=) escribiendo a la izquierda únicamente el nombre de la variable y a la derecha el valor que queremos guardar. Si queremos guardar un número, no hace falta poner comillas, pero si queremos guardar una cadena de texto hay que poner comillas (dobles o simples).

```
$edad = 15;                    // La variable $edad tiene ahora el valor 15
$nombre = "Pepito Conejo"      // La variable $nombre tiene ahora el valor Pepito Conejo
```

✓ En el lado izquierdo de la asignación (=) no se puede escribir más que el nombre de una variable. El programa siguiente no es sintácticamente correcto por lo que no se ejecutará y PHP mostrará un mensaje de error.

```
$edad + 1 = 16;
```

✓ En la parte derecha de la asignación se pueden escribir expresiones matemáticas:

```
$perimetro = 2 * 3.14 * 15;
```

✓ Esas expresiones pueden contener variables:

```
$perimetro = 2 * 3.14 * $radio;
```

✓ En PHP una igualdad no es una ecuación matemática, sino una asignación (el resultado de la derecha se almacena en la variable de la izquierda). Por ello, una asignación pueden utilizar en el lado derecho la variable en la que se va a guardar el resultado:

```
$edad = 2 * $edad;
```

✓ En el caso de números, cadenas o matrices de una dimensión, las variables se puede insertar directamente:

```
$seEscribe = ["separado", "junto"];
print "<p>El número $numero se escribe $seEscribe[0]: $texto</p>\n";
```

✓ En el caso de matrices de dos o más dimensiones, las variables no se puede insertar directamente:

```
<?php
$nombre = "Don Pepito";
$saludos = [["Hello", "Hola"], ["Goodbye", "Adios"]];
print "<p>¡$saludos[0][1], $nombre! ¿Cómo está
      usted?</p>\n";
?>
```

```
// dos dimensiones, no mostraría el valor esperado
<b>Notice</b>:  Array to string conversion in
<b>ejemplo.php</b> on line <b>5</b><br />
<p>¡Array[1], Don Pepito! ¿Cómo está usted?</p>
```

✓ Una solución a este problema es sacar la matriz de la cadena. Otra solución sin necesidad de sacar la matriz de la cadena es rodear la variable con llaves ({ }):

```
<?php
$nombre = "Don Pepito";
$saludos = [["Hello", "Hola"], ["Goodbye", "Adios"]];
print "<p>¡" . $saludos[0][1] . ", $nombre! ¿Cómo
      está usted?</p>\n";
?>
```

```
<p>¡Hola, Don Pepito! ¿Cómo está usted?</p>
```

```
<?php
$nombre = "Don Pepito";
$saludos = [["Hello", "Hola"], ["Goodbye", "Adios"]];
print "<p>¡{$saludos[0][1]}, $nombre! ¿Cómo está
      usted?</p>\n";
?>
```

```
<p>¡Hola, Don Pepito! ¿Cómo está usted?</p>
```

✓ En html/css a veces es necesario juntar números y caracteres, como en el ejemplo siguiente en el que se establece el tamaño del párrafo en 30px. Para ello se pueden utilizar llaves o sacar la variable de la cadena:

```
<?php
$x = 30;
print "<p style=\"font-size: {$x}px\">Texto grande</
p>\n";
?>
```

```
<p style="font-size: 30px">Texto grande</p>
```

```
<?php
$x = 30;
print "<p style=\"font-size: " . $x . "px\">Texto
grande</p>\n";
?>
```

```
<p style="font-size: 30px">Texto grande</p>
```

Cuestiones

1. Hola.php → Escriba el típico primer programa que salude al mundo en un único fragmento PHP.

2. Dado.php → Escriba un programa que cada vez que se ejecute muestre un valor entre 1 y 6, al azar. Utilizamos la función rand(1,6).

3. DadoGrafico.php → Escriba un programa que cada vez que se ejecute muestre un dado con el valor entre 1 y 6, al azar.

3.3. VARIABLES PREDEFINIDAS

PHP genera automáticamente una serie de variables con diversa información sobre el cliente y el servidor.

Estas variables predefinidas en PHP son "superglobales", lo que significa que están disponibles en todos los ámbitos a lo largo del script. No es necesario emplear **global $variable;** para acceder a ellas dentro de las funciones o métodos.

$_REQUEST

$_REQUEST es una matriz asociativa que contiene los datos enviado por los formularios y las cookies guardadas en el ordenador del cliente.

$_SERVER

$_SERVER es una matriz asociativa que contiene información sobre cabeceras, rutas y ubicaciones de scripts suministrada por el servidor (pero hay que tener en cuenta que no todos los servidores suministran todos los datos).

$_SERVER[PHP_SELF]

$_SERVER[PHP_SELF] contiene la dirección de la página (relativo a la raíz, es decir, sin el nombre del servidor).

URL	$_SERVER[PHP_SELF]
http://www.example.com/ejemplo.php	/ejemplo.php
http://www.example.com/ejercicios/ejemplo.php	/ejercicios/ejemplo.php

```
<?php
print "<p>\$_SERVER[PHP_SELF]: $_SERVER[PHP_SELF]
      </p>\n";
?>
```

```
<p>$_SERVER[PHP_SELF]:
      /consultar/php/lecciones/ejemplos/variables-
      predefinidas-server-phpself-1.php </p>
```

- ✓ Esta variable se puede utilizar en las páginas que enlazan consigo mismas. Por ejemplo, una página puede contener un formulario que envíe los datos a esa misma página. Para ello, el atributo action del formulario debe contener el nombre de la página. En vez de poner el nombre de la página, se puede utilizar $_SERVER[PHP_SELF]. La ventaja es que aunque se cambie el nombre del fichero, el enlace seguirá funcionando.
- ✓ El ejemplo siguiente muestra cómo se escribiría un enlace que apunta a la misma página que contiene el enlace:

```
<?php
print "<p><a href=\"$_SERVER[PHP_SELF]\">Enlace a esta misma página</a>\n";
?>
```

3.4. Tipos de variables

Los tipos de variables básicos son los siguientes:

- lógicas o booleanas (boolean)
- enteros (integer)
- decimales (float)
- cadenas (string)
- matrices (arrays)

Existen además los tipos:

- objetos (object)
- recursos (resource)
- nulo (null)

Variables lógicas (boolean)

Las variables de tipo lógico sólo pueden tener el valor true (verdadero) o false (falso). Se suelen utilizar en las estructuras de control.

- **Nota**: La estructura de selección if se verá con las estructuras de control. Para entender este ejemplo, es suficiente saber que if significa si (si como condición, no sí como afirmación) y va seguida de una operanción lógica que en caso de que su resultado de cierta se ejecuta la instrucción siguiente (normalmente entre corchetes { }).

```php
<?php
$autorizado = true;

if ($autorizado) {           // equivale a $autorizado == true
    print "<p>Usted está autorizado.</p>\n";
}

if (!$autorizado) {          // equivale a $autorizado == false
    print "<p>Usted no está autorizado.</p>\n";
}
?>
```

```
<p>Usted está autorizado.</p>
```

```php
<?php
$autorizado = false;

if ($autorizado) {           // equivale a $autorizado == true
    print "<p>Usted está autorizado.</p>\n";
}

if (!$autorizado) {          // equivale a $autorizado == false
    print "<p>Usted no está autorizado.</p>\n";
}
?>
```

```
<p>Usted no está autorizado.</p>
```

Variables enteras (integer)

Las variables de tipo entero pueden guardar números enteros (positivos o negativos).

Variables decimales (float)

Las variables de tipo decimal (float) pueden guardar números decimales (positivos o negativos). Como en las calculadoras, el separador de la parte entera y la parte decimal es el punto (.), no la coma (,)..

Variables de cadenas (string)

Las variables de tipo cadena pueden guardar caracteres.

- PHP no impone ningún límite al tamaño de las cadenas. Las cadenas pueden ser todo lo largas que permita la memoria del servidor.
- El juego de caracteres que utiliza PHP viene determinado en principio por el juego de caracteres que utiliza el fichero fuente del programa. Pero hay que tener en cuenta que las funciones de tratamiento de cadenas no están preparadas para tratar la diversidad de juegos de caracteres: muchas suponen que cada carácter ocupa solamente un byte, otras suponen un juego de caracteres determinado (UTF-8, por ejemplo), otras utilizan el juego de caracteres definido localmente, etc.
- Se puede acceder a caracteres individuales indicando la posición del carácter, como si se tratara de una matriz de una dimensión en la que el primer carácter ocupa la posición 0.
  - Si se indica una posición mayor que la longitud de la cadena, la cadena se alarga con espacios hasta llegar a ese valor
  - Si en una posición se guarda una cadena vacía, la cadena se acorta eliminando el carácter de esa posición

4. CONSTANTES

Las constantes son elementos de PHP que guardan un valor fijo que no se puede modificar a lo largo del programa. Las constantes pueden ser definidas por el programa o estar predefinidas por el propio PHP o por algún módulo. Los nombres de las constantes siguen las mismas reglas que los nombres de las variables, pero sin el dólar ($) inicial. La costumbre es escribir los nombres de las constantes en mayúsculas.

En principio, se puede no utilizar constantes nunca, puesto que las constantes definidas por el programa podrían reemplazarse por variables. La ventaja de usar constantes y variables es que se puede distinguir a simple vista si a lo largo de un programa algo va a permanecer constante (si es una constante) o puede cambiar (si es una variable). El inconveniente de usar constantes es que las constantes no se sustituyen dentro de las cadenas y es necesario sacarlas fuera de las cadenas, haciendo el código un poco más incómodo de escribir y leer. Desde el punto de vista del rendimiento, la diferencia es inapreciable.

✓ La función define(nombre_constante, valor_constante) permite definir constantes.

```
<?php
define("PI", 3.14);
print "<p>El valor de pi es " . PI . "</p>\n";
?>
```

```
<p>El valor de pi es 3.14</p>
```

✓ La costumbre es escribir los nombres de las constantes en mayúsculas. Los nombres de las constantes deben empezar por una letra y tener sólo letras (acentos, etc), números y guiones bajos.

✓ Las constantes no se sustituyen dentro de una cadena por lo que es necesario sacarlas para mostrar su valor.

```
<?php
define("PI", 3.14);
print "<p>El valor de pi es PI</p>\n";
print "<p>El valor de pi es " . PI . "</p>\n";
?>
```

```
<p>El valor de pi es PI</p>
<p>El valor de pi es 3.14</p>
```

✓ En las constantes, PHP distingue entre minúsculas y mayúsculas:

```
<?php
define("PI", 3.14);
define("pi", 3.141592);
print "<p>El valor de pi es " . PI . "</p>";
print "<p>El valor de pi es " . pi . "</p>";
?>
```

```
<p>El valor de pi es 3.14</p>
<p>El valor de pi es 3.141592</
```

4.1. Constantes predefinidas

Tanto PHP como los módulos cargados definen automáticamente una serie de constantes predefinidas, de las que se comentan algunas en esta lección:

✓ El número de constantes predefinidas depende de los módulos cargados en php.ini. La función get_defined_constants() devuelve las constantes predefinidas en el servidor que estemos utilizando:

```
<?php
print "<pre>";
print_r(get_defined_constants());
print "</pre>\n";
?>
```

```
<pre>Array
(
    [E_ERROR] => 1
    [E_RECOVERABLE_ERROR] => 4096
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_STRICT] => 2048
    ...
)
</pre>
```

Veamos como ejemplo estas dos: PHP_INT_MAX y PHP_INT_SIZE

- ✓ PHP_INT_MAX es el valor del mayor entero que se puede guardar en una variable de tipo entero.
- ✓ PHP_INT_SIZE es el tamaño en bytes de las variables de tipo entero, que depende del ordenador, del sistema operativo y de la versión de PHP. En sistemas de 32 bits suele ser 4 bytes y en sistema de 64 bits suele ser 8 bytes.

```
<?php
$maximo = PHP_INT_MAX;
print "<p>El mayor entero que se puede guardar \n";
print "en una variable entera es $maximo<p>\n";
$demasiado = (int)($maximo + 1);
print "<p>Si se intenta guardar 1 más, \n";
print "el resultado es $demasiado</p>\n";
?>
```

```
<p>El mayor entero que se puede guardar
en una variable entera es 2147483647</p>
<p>Si se intenta guardar 1 más,
el resultado es -2147483648</p>
```

```
<?php
$tamano = PHP_INT_SIZE;
print "<p>Los enteros se guardan utilizando $tamano
      bytes</p>";
?>
```

```
<p>Los enteros se guardan utilizando 4 bytes</p>
```