

Cap.2- Fundamentos PHP

- 4) Comentarios
- 5) Variables
- 6) Constantes
- 7) Superglobales

3 Comentarios

Comentarios:

- Forma de mejorar la legibilidad del código y que se recomienda siempre como una buena práctica a la hora de programar.
- Comentarios de una línea:
 - `//` esto es un comentario de una línea
 - `#` esto es otra forma de comentario de una línea
- Comentarios de múltiples líneas:
 - `/*` esta es la forma de escribir comentarios de varias líneas `*/`

4 Variables

- Definición y uso:
 - “Almacenes temporales de datos que permiten gestionar los datos utilizados por la aplicación web durante el flujo de ejecución de una página determinada”.
 - Se identifican por el símbolo del dólar (\$) seguido del nombre de una variable.
 - En PHP, las variables no necesitan ser declaradas explícitamente.
 - Además, no tienen un tipo definido hasta que no se les asigna un valor. (loosely type)
 - `$var = 15;`
 - `$var = "cambio de tipo";`

4 Variables

- Reglas de nombrado:
 - debe comenzar con una letra o con un guión bajo (“_”).
 - únicamente puede contener caracteres alfanuméricos y guiones bajos (a-z, A-Z, 0-9 y _).
 - no debe contener espacios en blanco.
 - PHP es case-sensitive → las variables *\$Variable*, *\$variable* y *\$VARIABLE* son variables completamente diferentes.

Recomendación

- Usar solo minúsculas.
- En nombres multipalabra separar cada palabra con guión bajo.

4 Variables

- Tipos de datos:
 - Tipos escalares: boolean, integer, float y string.
 - Tipos compuestos: array y object.
 - Tipos especiales: NULL y resource.
 - Pseudo-tipos: mixed, number y callback.

4 Variables

Expresiones y operadores

- Operadores:
 - `$var = (($var - 3) * 4) / 2;`
- Concatenación:
 - `$var = "cadena" . " unida";`
- Varias formas de añadir 1 a la variable `$var`:
 - `$var = $var + 1;`
 - `$var += 1;`
 - `$var++;`
- Varias formas de multiplicar o dividir:
 - `$var = $var * 2; → $var *= 2;`
 - `$var = $var / 2; → $var /= 2;`

4 Variables

- Conversiones entre tipos de datos:
 - Funciones específicas (algunas):
 - `string strval` (mixed variable) → transforma a string.
 - `integer intval` (mixed variable) → transforma a integer.
 - `float floatval` (mixed variable) → transforma a float.
 - Genérica:
 - `Settype` (mixed variable, string type).
 - Parámetro “variable”: valores de tipo *array*, *boolean*, *float*, *integer*, *object* o *string*.
 - Parámetro “type”: cadena de caracteres que indica el tipo al que queremos transformar el parámetro “variable”.

4 Variables

- Conversiones entre tipos de datos:

Sentencia	Resultado
(int) \$var (integer) \$var	Conversión a tipo integer.
(bool) \$var (boolean) \$var	Conversión a tipo boolean.
(float) \$var (double) \$var (real) \$var	Conversión a tipo float.
(string) \$var	Conversión a tipo string.
(array) \$var	Conversión a tipo array.
(object) \$var	Conversión a tipo object.

4 Variables

- Conversiones entre tipos de datos (Ejemplos):

Valor de \$var	(int) \$var	(bool) \$var	(string) \$var	(float) \$var
null	0	false	""	0
true	1	true	"1"	1
false	0	false	""	0
0	0	false	"0"	0
3.8	3	true	"3.8"	3.8
"0"	0	false	"0"	0
"10"	10	true	"10"	10
"6 metros"	6	true	"6 metros"	6
"hola"	0	true	"hola"	0

4 Variables

Conversión automática:

- Cuando combinamos en una misma expresión dos variables que inicialmente tienen tipos diferentes o cuando pasamos una variable como argumento a una función que espera un tipo de dato diferente.
- \$var se convierte a tipo integer con valor 35:
 - \$var = "20" + 15;
- \$var se convierte a string con valor = "20 años":
 - \$var = 20 . " años";
- \$var se convierte a tipo integer con valor = 20:
 - \$var = 20 + " años";
- \$var se convierte a tipo integer con valor = 42:
 - \$var = 40 + "2 razones";

4 Variables

- Comprobación del tipo de una variable:
 - `boolean is_int` (mixed variable).
 - `boolean is_float` (mixed variable).
 - `boolean is_bool` (mixed variable).
 - `boolean is_string` (mixed variable).
 - `boolean is_array` (mixed variable).
 - `boolean is_object` (mixed variable).
 - `Boolean is_null` (mixed variable).

4 Variables

- Estado de una variable:
 - Una variable puede estar en un estado indeterminado (no tener un valor asignado) e incluso puede no haber sido definida (estado indefinido).
 - Funciones para comprobar el estado de una variable:
 - *boolean isset (mixed var)* → comprueba si a una variable se le ha asignado un valor no nulo.
 - *boolean empty (mixed var)* → comprueba si esa variable tiene un valor.
 - Función para pasar una variable a estado “indefinido”:
 - *unset()*.

4 Variables

- Estado de una variable:

Contenido de \$var	isset(\$var)	empty(\$var)	(bool) \$var
\$var = null;	false	true	false
\$var = 0;	true	true	false
\$var = true	true	false	true
\$var = false	true	true	false
\$var = "0";	true	true	false
\$var = "";	true	true	false
\$var = "foo";	true	false	true
\$var = array();	true	true	false
unset (\$var);	false	true	false

4 Variables

- **Ámbito de una variable:**
 - *“Contexto dentro del que la variable está definida, es decir, la zona del programa en la que puede ser referenciada/usada”.*
 - En PHP hay tres ámbitos diferentes:
 - local
 - global
 - static

4 Variables

Ámbito local:

- las variables internas a una función, única y exclusivamente pueden ser utilizadas dentro de dicha función.

```
function duplicar($var) {  
    $temp = $var * 2;  
}
```

```
$variable = 5;
```

```
duplicar($variable);
```

```
echo "El valor de la variable \ $temp es: $temp";
```

→ Salida: El valor de la variable \$temp es: ... y ningún valor para \$temp.

4 Variables

Ámbito global:

- 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.

```
function duplicar($var) {  
    global $temp;  
    $temp = $var * 2;  
}  
$variable = 5;  
duplicar($variable);  
echo "El valor de la variable \ $temp es: $temp";
```

- **Salida:** El valor de la variable \$temp es: 10.
- La utilización de variables globales sin control puede resultar en un código difícil de mantener y propenso a dar errores.

4 Variables

- **Ámbito global:**
 - PHP almacena todas la variables globales en una matriz llamada `$GLOBALS[index]`. El índice lo constituye el nombre de la variable. Esta matriz también es accesible desde dentro de las funciones y puede ser usada para actualizar variables globales directamente.

```
<?php
$x=5;
$y=10;

function myTest()
{
    $GLOBALS['y']= $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

4 Variables

- Ámbito global:
 - Una variable global declarada dentro de una función solo existirá si se ejecuta dicha función.

¿Cuál es el resultado de ejecutar el siguiente código?

```
<?php
function variableGlobal(){
    global $probando;

    $probando="Hola Mundo cruel";
}

//variableGlobal();
echo $probando;
?>
```

4 Variables

- **Ámbito estático:**

Si una variable interna a una función es declarada *static*, conservará el valor de la última vez que fue llamada. La variable sigue siendo local.

```
<?php  
  
function myTest()  
{  
    static $x=0;  
    echo $x;  
    $x++;  
}  
  
myTest();  
myTest();  
myTest();  
  
?>
```

→ La salida será: 012

5 Constantes

■ Constantes

- PHP también permite la definición de valores constantes cuyo contenido no puede ser cambiado durante la ejecución del código.
 - `define ("PI", 3.14159);`
- El nombre de una constante empieza por una letra o un '_' y son case-sensitive.
- Las constantes son siempre globales a lo largo del script
- Solo pueden ser valores boolean, integer, float o string.

Nota: PHP ofrece un largo número de constantes predefinidas a cualquier script en ejecución. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y sólo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque han sido compiladas.

5 Constantes

Hay ocho constantes predefinidas que cambian dependiendo de donde son usadas. Por ejemplo el valor de `__LINE__` depende en la línea que se use en el script. Estas constantes especiales son sensibles a mayúsculas y son las siguientes:

Varias constantes PHP "mágicas"

Nombre	Descripción
<code>__LINE__</code>	Línea actual en el fichero.
<code>__FILE__</code>	Ruta completa y nombre del fichero. Si se usa dentro de un include, devolverá el nombre del fichero del include. Desde PHP 4.0.2, <code>__FILE__</code> siempre contiene la ruta absoluta con symlinks resueltos, en otras versiones contenía la ruta relativa según las circunstancias.
<code>__DIR__</code>	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluido. Esta constante es igual que <code>dirname(__FILE__)</code> . El nombre del directorio no lleva la barra inicial a no ser que esté en el directorio root. (Fue añadida en PHP 5.3.0)
<code>__FUNCTION__</code>	Nombre de la función. (Añadida en PHP 4.3.0) Desde PHP 5 esta constante devuelve el nombre de la función donde fue declarada (sensible a mayúsculas). En PHP 4 su valor siempre es en minúsculas.
<code>__CLASS__</code>	Nombre de la clase. (Añadida en PHP 4.3.0) Desde PHP 5 esta constante devuelve el nombre de la clase donde fue declarada (sensible a mayúsculas). En PHP 4 su valor siempre es en minúsculas. El nombre de la clase incluye el namespace declarado en (p.e.j. <code>Foo\Bar</code>). Tenga en cuenta que a partir de PHP 5.4 <code>__CLASS__</code> también funciona con traits. Cuando es usado en un método trait, <code>__CLASS__</code> es el nombre de la clase del trait que está siendo utilizado.
<code>__TRAIT__</code>	El nombre de el trait. (Añadido en PHP 5.4.0) A partir de PHP 5.4 esta constante devuelve el trait que fué declarado (sensible a mayúsculas y minúsculas). El nombre de el trait incluye el namespace si alguno fué declarado en (p.e.j. <code>Foo\Bar</code>).
<code>__METHOD__</code>	Nombre del método de la clase. (Añadida en PHP 5.0.0.) Nombre del método devuelto donde fue declarada. (sensible a mayúsculas).
<code>__NAMESPACE__</code>	Nombre del espacio de nombres actual (sensible a mayúsculas). Esta constante se define en tiempo de compilación (Añadida en PHP 5.3.0) El nombre del namespace actual (sensible a mayúsculas).

Véase también [get_class\(\)](#), [get_object_vars\(\)](#), [file_exists\(\)](#) y [function_exists\(\)](#).

6 Superglobales

Las variables superglobales son:

`$GLOBALS`

`$_SERVER`

`$_GET`

`$_POST`

`$_FILES`

`$_COOKIE`

`$_SESSION`

`$_REQUEST`

`$_ENV`