PHP

UT2: FUNDAMENTOS DEL LENGUAJE PHP

1. Sintaxis general
2. Variables
3. Cadenas de caracteres
4. Operadores básicos
5. Función: echo
6. Constantes
7. Tipos de datos

www.php.net/manual/es/

1. Sintaxis general

Las normas sintácticas esenciales del lenguaje PHP son:

- Cualquier script PHP que integremos en una página web debe ir comprendido entre los elementos `<?php` y `?>`. Estos elementos son los que indican al intérprete PHP sobre qué parte de la página debe actual.
- El final de cada instrucción PHP se designa con un punto y coma (;).
- Los bloques de instrucciones correspondientes a bucles, sentencias condicionales, y funciones deben ir encerrados entre llaves {}.
- El intérprete PHP ignora los espacios, los tabuladores y los saltos de línea; en otras palabras: podemos poner tantos espacios, tabuladores y saltos de línea como queramos para mejorar la legibilidad de nuestro código porque el intérprete considerará que todo lo que escribamos corresponde a una misma instrucción hasta que encuentre un punto y coma;.
- PHP es parcialmente insensible al uso de mayúsculas y minúsculas, pero no totalmente insensible. Al intérprete PHP le da igual que escribamos los nombres de las instrucciones y de las funciones en mayúsculas o minúsculas o mezclando mayúsculas o minúsculas; por ejemplo, echo funciona igual de bien que ECHO o eChO. Sin embargo, el intérprete PHP es mucho más exigente con los nombres de variables y constantes; en este caso debemos ser coherentes y escribirlos siempre del mismo modo. Una norma no escrita aconseja utilizar mayúsculas para los nombres de las constantes y minúsculas para los nombres de las variables.
- Cualquier contenido de una línea situado a continuación de una doble barra // será ignorado por el intérprete PHP. Es habitual escribir // al principio de una línea para dejarla sin efecto cuando se está depurando código. // sirve para crear comentarios de una sola línea.
- De forma similar, todo lo escrito entre los símbolos /* y */ será ignorado por el intérprete PHP, incluso aunque ocupe varias líneas. /* y */ sirven para crear comentarios de varias líneas.
- Es aconsejable indentar las instrucciones porque así será más fácil e intuitivo localizar un bloque de líneas al depurar o estudiar un script.

2. Variables

Una variable es un identificativo que nos permite referirnos a un dato que puede variar a lo largo de la ejecución del script.

En PHP todas las variables deben empezar con el signo $, seguido de una letra o un signo de subrayado, seguido de cualquier combinación de letras, números y signos de subrayado.

PHP es sensible al uso de mayúsculas/minúsculas en las variables; por ejemplo, $edad y $Edad se interpretan como dos variables diferentes.

PHP es muy "laxo" en el manejo de las variables:

- No nos obliga a indicar a qué tipo de datos va a estar referida la variable. Por ejemplo, una misma variable puede contener un número en un punto del script y una cadena de caracteres en otro punto diferente.
- No nos obliga a inicializar explícitamente las variables, es decir, no tenemos que asignar un valor inicial a las variables, sino que podemos usarlas directamente en el código aunque aún no representen a ningún dato concreto; en esta situación PHP aplica unas reglas de asignación de dato predeterminado que básicamente consisten en asignar el dato 0 a las variables que intervienen en operaciones matemáticas, y una cadena vacía a las variables que intervienen en operaciones con cadenas de caracteres.

Para asignar un dato a una variable se utiliza el operador =. Por ejemplo:

```
001   $miedad=38;
```

Por defecto PHP asigna los datos a las variables por valor; esto quiere decir que si asignamos a una variable otra variable, aunque esta segunda cambiase de dato no cambiaría el dato de la primera.

```
001   $precio_kw=0.15;
002   $consumo_julio=3200;
003   $factura_julio=$consumo_julio*$precio_kw;
004   echo $factura_julio;//El dato al que se refiere la variable es 480
005   echo '<br/>';
006   $precio_kw=0.17;
007   echo $factura_julio;//El dato al que se refiere la variable sigue siendo 480
```

Sin embargo, también permite almacenar en una variable una referencia a otra variable, de modo que si esta segunda cambia, también lo haga la primera, o (¡atención!) si cambia la primera, también cambie la segunda. Si anteponemos el signo & al nombre de una variable, PHP entenderá que queremos referirnos a su referencia en lugar de a su valor. Y si asignamos esa referencia a otra variable, a partir de ese momento ambas variables se referirán a un mismo dato.

```
001  $salario_base=2000;
002  $salario_juan=&$salario_base;//Ahora ambas variables apuntan a la misma posición de
     memoria
003  echo $salario_juan;
004  echo '<br/>';
005  $subida_ipc=2;
006  $salario_base=$salario_base*(100+$subida_ipc)/100;//Ahora el salario base es 2040
007  echo $salario_juan;//El salario de Juan se actualiza automáticamente al salario
     base
008  echo '<br/>';
009  $incentivo_juan=5;
010  $salario_juan=$salario_juan*(100+$incentivo_juan)/100;//Ahora el salario de Juan es
     2142
011  echo $salario_base;//Y el salario base también
012  //$salario_base y $salario_juan son referencias a una misma posición de memoria
```

3. Cadenas de caracteres

En PHP un dato de tipo cadena de caracteres debe ir escrito (preferiblemente) entre comillas simples.

Nota: Para incluir en una cadena de caracteres el carácter de delimitación ' tendremos que escribirlo precedido por el carácter \, es decir, escribiremos \'. La anteposición del carácter \ a otro carácter se denomina escapado. Para incluir el carácter \ tendremos que escribirlo también escapado, es decir, escribiremos \\.

Anteriormente se indicaba que en PHP lo preferible es identificar las cadenas de caracteres colocándolas entre comillas simples, pero también se pueden usar comillas dobles, aunque tienen una connotación especial: al utilizar comillas dobles el intérprete busca dentro de la cadena nombres de variables y, si los encuentra, los sustituye por sus correspondientes valores. Por ejemplo:

```
001  <?php
002          $variable=10;
003          echo "En el mundo s&oacute;lo hay $variable de
     personas";
004  ?>
```

Al usar comillas dobles, los caracteres que requieren escapado para poder mostrarse correctamente son `\\`, `\"` y `\$`.

Si no necesitamos esta funcionalidad de sustitución de variables dentro de una cadena de caracteres es preferible usar comillas simples porque se interpretan más rápido.

Nota: A diferencia de otros lenguajes de programación, cuando en PHP se asigna una cadena de caracteres a una variable no es necesario declarar a priori el tamaño de esa variable (la longitud de la cadena de caracteres). En PHP las cadenas de caracteres pueden tener cualquier limitación, sin ninguna limitación más que la memoria del ordenador.

Para concatenar dos cadenas de caracteres se utiliza el operador punto (.), por ejemplo:

```
001  <?php
002          echo 'cadena 1'. ' y
     '.'cadena 2';
003  ?>
```

4. Operadores básicos

Aparte del operador de concatenación de cadenas de caracteres (.) y el operador de asignación (=) que ya conocemos, en PHP pueden utilizarse:

- Los operadores aritméticos clásicos:
  +: Suma
  -: Resta
  *: Multiplicación
  /: División
  %: Cociente o módulo
  ++: Incremento unitario
  --: Decremento unitario

- Los operadores de comparación clásicos:
  ==: Igualdad
  !=: Desigualdad
  <: Menor que
  >: Mayor que
  <=: Menor o igual que
  >=: Mayor o igual que
  ===: Devuelve TRUE si los dos valores que compara son idénticos en contenido y tipo y FALSE si no lo son.
- Los operadores lógicos clásicos:
  !: No, negación
  && ó and: y
  || ó or: o

Nota: Las operaciones de comparación y lógicos dan como resultado datos booleanos (TRUE o FALSE; verdadero o falso). El operador de igualdad flexible (==) considera TRUE cualquier número distinto de cero, cualquier cadena de caracteres no vacía excepto la cadena '0', y cualquier array no vacío.

Por ejemplo:

```
001  <?php
002          echo "<br/>".(5+4);           //9
003          echo "<br/>".(5-4);           //1
004          echo "<br/>".(5*4);           //20
005          echo "<br/>".(10/3);   //3.33333333
006          echo "<br/>".(5%4);           //1
007
008          echo "<br/>".(5==4);          //
009          echo "<br/>".(-1==TRUE);      //1
010          echo "<br/>".(array(2,3,4)==TRUE);
             //1
011          echo "<br/>".(''==TRUE);      //
012          echo "<br/>".('0'==TRUE);     //
013
014          echo "<br/>".(5<4);           //
015
016          echo "<br/>".((3-2==1) and
     (3*2==5));
017          echo "<br/>".(TRUE or FALSE);  //1
018
019  ?>
```

En la siguiente tabla se recogen los principales operadores de PHP en orden de precedencia descendente (recuerde que se puede alterar este orden utilizando paréntesis). Tras la tabla encontrará algunas notas de interés sobre estos operadores.

++ y --
(integer), (float), (string), (boolean) y (array)
!
*, / y %
+, - y .
<, <=, > y >=
== y !=
&&
||
=, +=, -=, *=, /=, .= y %=
and
or

Los operadores de incremento y decremento unitario pueden usarse en modo prefijo o sufijo. Si se usan en modo prefijo devuelven el valor incrementado/decrementado; pero si se usan en modo sufijo devuelven el valor sin incrementar/decrementar y luego lo incrementan.

El operador % convierte los operandos a enteros antes de averiguar el cociente, y utiliza para el resultado el signo de numerador.

Los operadores && y || realizan exactamente la misma operación que los operadores and y or, respectivamente, pero observe que su orden de precedencia es superior al de ellos y al de los operadores de asignación (=, +=, -=, *=, /=, .= y %=).

Por ejemplo:

```
<?php
        $premisa1=TRUE;
        $premisa2=FALSE;
        $resultado=$premisa1 and  $premisa2;        //= precede a and
        echo (integer)$resultado;
        echo '<br />';
        $resultado=$premisa1 &&  $premisa2;        //&& precede a =
        echo (integer)$resultado;
?>
```

5. Función: echo

La función echo muestra información en la página web.

Tiene esta sintaxis:

echo *argumento*, donde la información que se muestra (*argumento*) puede ser una cadena de caracteres entre comillas o una variable.

Si el argumento es una cadena entre comillas dobles, debemos tener en cuenta que las variables incluidas en ese texto se expanden, es decir, muestran su contenido, no su nombre.

```
<?php
        $sueldo=400;
        echo "La variable \$sueldo tiene valor $sueldo <br/>";
        echo "La variable \$sueldo \"tiene\" valor $sueldo <br/>";
        echo 'La variable $sueldo tiene valor '. "$sueldo".' <br/>';
        echo 'La variable $sueldo "tiene" valor '." $sueldo".' <br/>';
?>
```

Sin embargo, las funciones no se expanden.

```
<?php
    $x = 'módulo iaweb';
    echo " La variable \$x contiene el texto ". strtoupper($x)." en letras mayúsculas";
    echo ' La variable $x contiene el texto '. strtoupper($x).' en letras mayúsculas';
?>
```

6. Constantes

Las constantes son nombres que se refieren a un dato que no cambia a lo largo de la ejecución del script. En cierta forma son como variables de ámbito superglobal, pues son reconocidas en el script en el que son definidas tanto dentro como fuera de las funciones. Sin embargo, tienen dos diferencias importantes respecto a las variables (además de la fundamental que es que no pueden alterar su dato durante la ejecución del script):

- Las constantes no se asignan con el operador =, sino con la función `define` de acuerdo a la sintaxis `define(nombre_constante_entre_comillas,dato_constante);`.
- Los nombres de las constantes no van precedidos del signo $, y por convenio se escriben completamente en mayúsculas.

Por ejemplo:

```
001  <?php

002  define('PI',3.1416);

003  echo 'PI='.PI;

004  echo '<br/><u>&Uacute;nica circunferencia cuyo per&iacute;metro y su &aacute;rea
     coinciden</u><br />';

005  echo 'Radio=2; Per&iacute;metro=&Aacute;rea=';

006  echo perimetro_circunferencia(2);

007  function perimetro_circunferencia($radio){

008  return 2*PI*$radio;

009  }

010  ?>
```

PHP incluye algunas constantes predefinidas, y entre ellas destacan algunas denominadas popularmente *constantes mágicas*, cuyo valor es asignado por el propio intérprete y varía en función de dónde sean utilizadas (no son verdaderas constantes), como:

- FILE__: Contiene la ruta completa del archivo en el que se encuentra el script.
- __DIR__: Contiene el nombre de la carpeta en la que se encuentra el archivo del script.

Por ejemplo:

```
001  <?php

002  echo __FILE__;

003  echo '<br />';

004  echo __DIR__;

005  ?>
```

7. *Tipos de datos*

Los tipos de datos de un lenguaje de programación son las clases de información que es capaz de manejar (entender, o utilizar en operaciones). Por ejemplo, cuando los humanos leemos 1-julio-2011 lo identificamos automáticamente como una fecha, y sabemos utilizar esta fecha en diversas operaciones (como calcular los días transcurridos desde otra fecha, o saber si es mayor o menor que otra fecha, …). Diríamos entonces que uno de los tipos de datos que manejan los humanos son fechas.

Exactamente igual ocurre con los lenguajes de programación, aunque generalmente sus tipos de datos son más primitivos (esenciales) que los que utilizamos los humanos. Los tipos de datos fundamentales de PHP son:

- **Booleanos** o lógicos (bool): sólo pueden tener el valor TRUE o FALSE
- **Números enteros** (integer): Cualquier número que pertenezca al grupo de los enteros (no simplemente natural; es decir, cualquier número positivo o negativo incluido el cero), dentro de los límites de representación que permita la arquitectura del ordenador.
- **Números de coma flotante** (float): Cualquier número que pertenezca al grupo de los reales. En las versiones iniciales de PHP este tipo de datos se llamaba número doble.
- **Cadenas de caracteres** (string).
- **Matrices** o arrays (array).
- **NULL**. Es un tipo de datos que sólo admite el valor NULL. ¿Un tipo de datos con un único valor? Suena raro ¿verdad? Cuando en el tema anterior indicamos que PHP permitía utilizar variables sin asignarles un valor explícito, realmente lo que ocurre es que PHP les asigna implícitamente este valor NULL. En otras palabras: el tipo de datos NULL es el que tienen todas las variables a las que aún no se ha asignado un dato concreto.

PHP no permite definir el tipo de datos de las variables, ni nos obliga a almacenar siempre el mismo tipo de datos en una variable. Pero entonces ¿cómo sabe cuál es el tipo de dato de cada variable? Lo establece en función del dato que tenga asignado en cada instante. Por ejemplo, si escribimos `$variable=5;`, la variable será de tipo entero, pero si luego escribimos `$variable='cadena';` pasará a ser de tipo cadena de caracteres.

PHP permite combinar en una misma operación datos de diferente tipo. ¿Podemos olvidarnos entonces del principio que nos enseñaron en la infancia 'no se pueden sumar peras con manzanas'? No. Para poder operar con datos de distinto tipo PHP los convierte internamente a un mismo tipo aplicando ciertas reglas. No obstante, estas reglas dependen de cada operación concreta y de los tipos de los operandos, de modo que son numerosas y difíciles de recordar. Por estos motivos, aunque posteriormente (en la sección dedicada a los operadores) citaremos cuáles son las principales reglas de conversión automática, es más cómodo ser disciplinado y procurar no mezclar operandos de diferente tipo en las operaciones (como en tantas otras facetas de la vida, a veces hacer bien las cosas cuesta menos que hacerlas mal).

PHP nos ofrece la función intrínseca `var_dump` para obtener información sobre una variable, dato o expresión; esta función devuelve el tipo del dato y su valor. Por ejemplo:

```
001  <?php
002  echo var_dump(TRUE); //boolean
003  echo var_dump(5); //integer
004  echo var_dump(5.5); //float
005  echo var_dump('5'); //string
006  echo var_dump(array(5)); //array
007  echo var_dump(NULL); //null
008  echo var_dump('TRUE'); //string
009  echo var_dump('TRUE' and 1); //boolean
010  ?>
```

Otra posibilidad alternativa a dejar en manos de PHP la conversión de datos, es forzarla nosotros mismo mediante una técnica que se denomina conversión explícita (*explicit casting* o *explicit type conversion*). Esta técnica consiste en escribir entre paréntesis delante del dato, variable o expresión el tipo al que queremos forzarlo. Por ejemplo:

```
001  <?php
002  echo var_dump(2.5+5.5);
003  echo var_dump((integer)2.5 + 5.5);
004  echo var_dump((integer)(2.5 + 5.5));
005  ?>
```

Los identificativos de conversión explícita son:

- `(boolean)`
- `(integer)`
- `(float)`
- `(string)`
- `(array)`
- `(unset)` para NULL.