Guía Básica del lenguaje PHP

Objetivo: Comprender el funcionamiento básico del lenguaje PHP.
Sitio oficial: http://www.php.net

Introducción

Fue creado originalmente por Rasmus Lerdorf en 1994, PHP es un lenguaje de programación interpretado, el nombre proviene del acrónimo recursivo “PHP” que significa PHP: Hypertext Preprocessor, es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. El lenguaje se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores.

PHP es uno de los lenguajes que sirven para la programación de scripts del lado del servidor, otros lenguajes muy utilizados son ASP o JSP, que tienen características similares.

Lenguajes de lado Cliente y Servidor

Se denomina lenguaje de lado cliente a aquel lenguaje de programación utilizado para controlar las operaciones de los clientes web (navegadores). Los más populares son el HTML (maquetación y presentación de la información en el navegador) y JavaScript (programación de acciones sobre el navegador).

Por otro lado llamamos lenguaje de lado servidor a aquel lenguaje de programación que es interpretado por el servidor web para realizar operaciones antes de brindar un resultado al navegador. Como lenguajes de lado servidor podemos nombrar: PHP, ASP, .net, entre otros.

Licencia de PHP

La licencia PHP es la licencia bajo la que se publica el lenguaje de programación PHP. El sitio oficial “www.php.net” de PHP la declara:

PHP 4, PHP 5 PHP 7 y 8 se distribuye bajo la licencia PHP v3.01 , derechos de autor (c) el Grupo de PHP.

- Se trata de una licencia de código abierto , certificado por la Open Source Initiative.

- La licencia de PHP es una licencia tipo BSD, que no tiene el "copyleft" restricciones asociadas con GPL.

De acuerdo a la Free Software Foundation es una licencia de software libre no copyleft y una licencia de código abierto según la Open Source Initiative. Debido a la restricción en el uso del término "PHP", no es compatible con la licencia GPL.

Licencia completa: https://www.php.net/license/index.php

Sintaxis

La sintaxis de PHP deriva de muchos lenguajes, con predominio de *C*, pero *Perl* también ha tenido mucha influencia en su sintaxis. Con la inclusión de orientación a objetos, la sintaxis Java se hizo presente también. A pesar de la incorporación de elementos varios lenguajes, PHP sigue sencillo y fácil de entender.

Código fuente PHP y Etiquetas

Desde sus orígenes PHP fue diseñado principalmente como un procesador de texto (de ahí su nombre). Para facilitar esta función, el código puede ser insertado directamente en archivos (casi siempre archivos con contenido HTML) mediante un conjunto especial de etiquetas. De esta forma, el intérprete enviará tal como esta el texto fuera de las etiquetas y ejecutará el código que se encuentra dentro de las mismas.

Hay cuatro tipos de etiquetas disponibles:

Etiquetas estándar	**<?php** echo “Método recomendado para incrustar código”; **?>**

Etiquetas disponibles

Encerradas entre las etiquetas encontraremos varias instrucciones a ser ejecutadas por el intérprete. Estas instrucciones podrán ser escritas juntas en la misma fila o separadas una por fila. En cualquiera de los casos, el carácter que separará cada instrucción sera el **; (punto y coma)**.

Separador	<?php echo “El Punto y coma separa una instrucción de otra”; ?>
	<?php echo “Una ”; echo “ o varias instrucciones por fila”; ?>
Cierre de etiqueta	<?php echo “El cierre de un bloque implica un punto y coma” ?>

Separador de instrucciones

Para realizar comentarios en nuestro código, PHP nos varias posibilidades dependiendo de las necesidades que requiramos.

Comentarios	<?php echo “Método 1”; **//comentario de una sola linea** ?>
	<?php echo “Método 2”; **/* podemos hacer comentarios de múltiples líneas, haciendo uso de “/* */” */** ?>
Cierre de etiqueta	<?php echo “Método 3”; **# también comentarios al estilo Shell** ?>

Comentarios

Tipos de Datos

PHP soporta muchos tipos de datos diferentes, pero en general podemos agruparlos dentro de dos grandes categorías: *escalares y compuestos.*

Un ***dato escalar*** puede contener un solo valor en un momento dado, mientras que un ***dato compuesto*** es contenedor de datos escalares.

Datos Escalares

Entre los datos escalares, PHP soporta cuatro:

Booblean	El valor solamente podrá ser **True** o **False**.
Int	Un valor numérico entero.
Float	Un valor numérico con signo y punto flotante.
String	Una colección de datos binarios.

Datos escalares

Boolean

Los datos booleanos son tipos de datos que pueden contener dos valores **True** o **False.** Se utilizan generalmente para realizar las operaciones booleanas básicas. Los valores true y false son en sí, la representación de los valores 1 y 0 respectivamente.

$a = true;	Valor verdadero. Correspondiente al valor 1.
$b = false;	Valor verdadero. Correspondiente al valor 0.

Datos tipo *boolean*.

Int

Los datos tipo ***int*** son utilizados para representar enteros con signo (valores tanto positivos como negativos).

$a = 16543	Valor entero.
$a = - 16543	Valor entero negativo.
$a = 0123	Valor octal (equivalente al 83 decimal).
$a = 0x1A	Valor exadecimal (equivalente al 26).

Datos tipo *int*.

Float

Los ***float*** en cambio poseen un componente fraccional el cual nos permite trabajar a nivel fracción.

$a = 1,2654	Valor decimal.
$a = 1.2e3	Valor octal.
$a = 7E-10	Valor exadecimal.

Datos tipo *float*.

String

Mientras que en algunos lenguajes los datos tipo **string** son equivalentes a texto, en muchos otros, como en el caso de PHP, ésta es una incorrecta descripción de este tipo de datos. Los string son colecciones ordenadas de datos binarios, los cuales pueden ser texto pero podrían ser también el contenido de un archivo de imagen, una hoja de cálculo, etc. Más adelante se extenderán estos conceptos.

Datos Compuestos

Entre los datos compuestos PHP ofrece:

Array	Los arreglos son contenedores ordenados de elementos de datos. Pueden ser utilizados para almacenar otro tipo de datos, incluyendo números, valores booleanos, texto, objetos y otros arreglos.
Object	Los objetos son contenedores tanto de datos como de código fuente. Ellos forman la base de la Programación Orientada a Objetos que veremos más adelante en otro capítulo.

Datos Compuestos

Datos Especiales

Esto son dos tipos de datos que PHP permite manipular y son utilizados en situaciones especiales.

Null	Indica que una variable no tiene valor. Una variable se considera NULL si se le ha asignado el valor especial NULL o si aún no ha sido asignado un valor en absoluto.
Resource	Se utiliza para indicar los recursos externos que no se encuentran representados de forma nativa por PHP, como por ejemplo, referencias a archivos, base de datos, etc.

Datos Especiales

Variables

Son contenedores de almacenamiento de datos temporales. Una variable puede contener cualquiera de los tipos de datos definidos en el apartado anterior.

En PHP las variables se identifican con un signo de dólar **$**, seguido del nombre del identificador. Pueden utilizarse letras (az, AZ), números y el carácter de subrayado. Su nombre puede comenzar con una letra o un guión bajo pero no con un número. Además, el nombre de una variable es sensible a mayúsculas y minúsculas .

```
$valor = “valor1”;      //nombre válido.
$_valor = “valor1”;     //nombre válido.
$1valor = “valor1”;     //nombre inválido. Comienza con un número.
```

Cada variable tiene un ámbito sobre el cual tiene inferencia. El ámbito de una variable puede ser *global* o *local*.

Global	La variable tendrá inferencia dentro de todo el script.
Local	La variable tendrá inferencia dentro de una función.

Ámbito de una variable

Por último, PHP nos provee de un conjunto de variables especiales predefinidas las cuales nos permitirán recuperar datos externos a una aplicación o script. Algunas de ellas son:

$GLOBALS - $_SERVER - $_GET - $_POST
$_COOKIES - $_FILES - $_ENV - $_REQUEST - $_SESSION

Variables variables

Una variable variable es aquella en la cual su nombre es definido a partir del valor de otra variable. Ejemplo:

```
$idioma = “es”;
$saludo_es = “Hola!!!”;
$saludo_en = “Hello!!!”;
$saludar = “saludo_” . $idioma;
print $$saludar;
```

Hola!!!

```
$idioma = “en”;
$saludo_es = “Hola!!!”;
$saludo_en = “Hello!!!”;
$saludar = “saludo_”.$idioma;
print $$saludar;
```

Hello!!!

En el ejemplo se puede ver que comenzamos con una variable $idioma cuyo valor es *“es” o “en”*. A continuación definimos la variable *$saludo_es* y *$saludo_en.* Se concatenan los valores en la variable *$saludar* y por medio de la instrucción $$saludar generamos una nueva variable cuyo valor queda “saludo_es” o “saludo_en”.

Funciones de variables

Nos periten determinar el tipo de dato que contiene una variable. Entre ellas definimos:

gettype()	Devuelve el tipo de una variable.
is_array()	Devuelve true o false si la variable es un arreglo.
is_bool()	Devuelve true o false si la variable es boobleana.
is_integer()	Devuelve true o false si la variable es un entero.
is_float()	Devuelve true o false si la variable es float.
is_object()	Devuelve true o false si la variable es un objeto.
is_string()	Devuelve true o false si la variable es un string.
is_resource()	Devuelve true o false si la variable es algún tipo de recurso.
var_dump()	Muestra el tipo y valor de una variable.

Funciones de variables

Constantes

Son contenedores para almacenar datos inmutables. Pueden contener cualquiera de los tipos de datos antes visto. Para definirlas, haremos uso de la palabra reservada *define* en base a la siguiente sintaxis:

bool define (nombre, valor [, bool $case_insensitive =false])

en donde:

nombre: es el nombre de la constante a definir.
valor: el valor que se le va a asignar a dicha constante.
[, **bool $case_insensitive =false**]: si se define sensible a mayúsculas y minúsculas.

Las constantes no pueden comenzar con un signo **$**. Ejemplo:

```
define (“Nombre”,”Esteban”,”true”);
print Nombre;
```

Esteban

Operadores

En PHP existe una gran variedad de operadores los cuales nos permitirán manipular y trabajar con valores o variables. A continuación presentaremos los mismos:

Aritméticos	+, -, *, / , %, ++, --
De Comparación	=, ==, !=, <, >, <=, >=
Lógicos	AND (&&), OR (||), NOT (!)
De cadena	. (concatenación), = (asignación)
De Asignación	=

Operadores

De Asignación

Nos permitirán asignar un valor a una variable.

Ejemplo:

```
$a= 5;
$b= 4;
```

Operadores Aritméticos

Nos permitirán realizar todas las operaciones aritméticas existentes.

Ejemplo:

```
$a= 5; $b= 4;
print "Suma: ". $a + $b;          → Suma: 9
print "Resta: ". $a - $b;         → Resta: 1
print "Producto: ". $a * $b;     → Producto: 20
```

```
$a= 1;
        → Salida = 1; Valor de $a= 1
$a++
        → Salida = 2; Valor de $a= 2 (incrementa el valor en 1)
++$a
        → Salida = 3; Valor de $a= 3 (incrementa el valor en 1)
--$a
        → Salida = 2; Valor de $a= 2 (decrementa el valor en 1)
$a--
        → Salida = 2; Valor de $a= 1 (decrementa el valor en 1)
```

Operadores de Comparación

Nos permitirán realizar todo tipo de comparaciones.

Ejemplo:

```
$a= 4; $b= 5;
```

$a == $b → false
$a != $b → true
$a > $b → false

Operadores Lógicos

Nos permitirán realizar todo tipo de operaciones lógicas.

Ejemplo:

$a= 4; $b= 5;
(($a == 4) && ($b == 5)) → true
(($a == 4) || ($b == 12)) → true
(($a == 4) and ($b == 12)) → true

Operadores de Cadena

Nos permitirán realizar asignaciones y concatenaciones de cadenas.

Ejemplo:

$a= “No voy a usar”;
$b= ” Hola mundo!!!!!!”;

$mensaje= $a.$b;
print $mensaje; → No voy a usar Hola mundo!!!!!

$a= “No voy a usar”; $a.= ” Hola mundo!!!!!!”;
print $a; → No voy a usar Hola mundo!!!!!

Estructuras de Control

Las estructuras de control permiten controlar el flujo de datos a lo largo de la ejecución de un script. PHP dispone de una serie de estructuras de control diferentes y aunque algunas de ellas pueden parecer redundantes, simplifican notablemente el desarrollo de los scripts. Es importante familiarizarse con ellas ya que son uno de los elementos fundamentales del lenguaje.

Estructuras Condicionales

Son estructuras que nos permiten cambiar el flujo de ejecución de un script basándose en una o mas condiciones, las cuales son ejecutadas y evaluadas.

Estructura Selectiva if – else

```
if (condición){
        opción o sentencia verdadera;
}else{
        opción o sentencia falsa;
}
```

Estructura Selectiva if – else anidada

```
 if (condicion1){
        opción o sentencia verdadera;
}else if(condicion2){
                opción o sentencia verdadera (condición2);
          }else{
                opción o sentencia falsa (condición1);
          }
}
```

Ejemplos:

```
<?php
        $a = 5;
        if ($a < 5){
                echo “El valor es menor que cinco”;
        }else if ($a = 5){
                        echo “El valor es igual a cinco.”;
                }else{
                        echo “El valor es mayor a cinco.”;
                }
        }
?>
```

Estructura Selectiva switch - case

Este tipo de estructura nos permitirán comparar el valor de una variable contra distintos valores y ejecutar alguna acción en particular según la coincidencia dada.
La estructura es la siguiente:

```
<?php
        switch ($variable){
                case condicion1:
                        //ejecución de la acción
                break;
                case condicion2:
                        //ejecución de la acción
                break;
                default:
                        //la acción que se ejecutará por defecto
                break;
        }
?>
```

En este tipo de estructura, el valor de *$variable* será evaluado una sola vez contra cada una de las condiciones de las clausulas *case*. Al encontrar una coincidencia, se ejecutarán las acciones declaradas debajo hasta encontrar la clausula *break*. A modo de ejemplo:

```
<?php
        $a = 3;
         switch ($a){

                case (1):
                                print (“El valor es 1”);
                                break;
                case (2):
                                 print (“El valor es 2”);
                                break;
                case (3):
                                 print (“El valor es 3”);
                                 break;
                default :
                                print (“No es ninguno de los valores.”)
        }

        Resultado:
                El valor es 3
?>
```

Estructuras Iterativas

Este tipo de estructuras hacen posible la ejecución de una o mas líneas de código un número determinado de veces. Como construcciones iterativas podemos encontrar **while()**, **do....while()**, **for()** y **foreach()**. En este apartado estudiaremos las tres primera, dejando la última

para el apartado de *Array()*. La estructura que siguen es la siguiente:

Estructuras while() do...while()

La forma de definir estas estructuras es la siguiente:

```
while(condicion){
        sentencia_1
        sentencia_2
        sentencia_n
}

do{
        sentencia_1
        sentencia_2
        sentencia_n
}while{condicion}
```

Ambas estructuras ejecutarán cada una de las sentencias hasta que la condición se evalúe como verdadera.

Ejemplo:

```
<?php
        $i = 0;
        while ($i < 10) {
                echo “Esta es la línea ”.$i;
                $i++;
        }

        $i = 0;
        do {
                echo “Esta es la línea ”.$i;
```

```
            echo " Acá ejecutamos dos sentencias.".$i;
            $i++;
        } while ($i < 10);
?>
```

Estos dos tipos de circuito son muy similares, la única diferencia es cuando la condición se comprueba para determinar si el código dentro de la construcción debe ser ejecutado o no. En un bucle *while ()*, la condición se chequea al *iniciar* el bucle, lo que significa que si la condición no es *true*, el código dentro del bucle no se ejecutará. En un *do ... while ()*, la comprobación se realizará al *finalizar* la iteración, por lo que siempre se ejecutará una vez el bucle.

Estructuras for()

Las otras dos estructuras iterativas que veremos son el *for()* y *foreach()*. Ambas aparecen como una alternativa a las anteriores, de manera de hacer más legible el código.

```
<?php
    for (inicio, condición, incremento){
        sentencia_1
        sentencia_2
        sentencia_n
    }
?>
```

En el *for*, la condición se encuentra formada por tres partes separadas por punto y coma *(inicio, condición, incremento)*. Inicio se utilizará para dar el valor de referencia a la variable a utilizar como control. Condición indicará la condición lógica propiamente dicha que debe evaluarse como verdadera para que se ejecute las sentencias y, por último, el incremento que indicará de cuanto en cuanto se deberá *incrementar/decrementar* el valor de la variable.
A modo de ejemplo:

```
<?php
    for ($i=0, $i <=10, $i++){
        print ("Fila ".$i);
```

```
                print (“<br>);
        }
          for  ($i=10, $i =10, $i--){
                print (“Fila “.$i);
                print (“<br>);
        }
?>
```

Detener y continuar

Como acabamos de ver en cualquiera de las estructuras iterativas anteriores, la sentencia o el conjunto de sentencia se ejecutan hasta que se evalúe una condición como verdadera. Sin embargo, podemos encontrarnos ante la situación de tener que interrumpir la iteración antes de que se termine de ejecutar la condición. Para esto, podemos hacer uso del **break**. Este tipo de manejo se puede volver útil en el caso de tener sentencias for anidadas.
A modo de ejemplo:

```
<?php
        for ($i = 0; $i < 10; $i++) {
                for ($j = 0; $j < 3; $j++) {
                        if (($j + $i) % 5 == 0) {
                                break 2; // sale de este ciclo y se saltea el siguiente en 1.
                        }
                }
        }
?>
```

Pueden aparecer situaciones en las cuales, en vez de terminar un ciclo, desee saltearse un número de interacciones. Esto puede realizare mediante la instrucción **continue**.

```
<?php
        for ($i = 0; $i < 10; $i++) {
                if ($i > 3 && $i < 6) {
```

```
                    continue;
                }
            echo “Línea “.$i;
            }
    ?>
```

En el ejemplo, la salida generada serán únicamente los números entre 0 y 3 y entre 6 y 9, dejando de lado o salteando todos los demás.

***Bibliografía*:**

Sitio oficial: http://www.php.net