PHP

LUIS JOSÉ SÁNCHEZ GONZÁLEZ

1. INTRODUCCIÓN
2. VARIABLES
3. OPERADORES ARITMÉTICOS
4. OPERADORES DE COMPARACIÓN
5. OPERADORES LÓGICOS
6. SENTENCIAS CONDICIONALES
7. BUCLES
8. MANEJO DE CADENAS DE CARACTERES
9. ENVÍO Y RECEPCIÓN DE DATOS
10. ENVÍO DE CORREOS ELECTRÓNICOS
11. EJERCICIOS
12. CONEXIÓN A UNA BASE DE DATOS DE MySQL DESDE PHP
13. MOSTRAR UNA CONSULTA COMPLETA
14. BUSCAR UN VALOR
15. EJERCICIOS

1 INTRODUCCIÓN

El lenguaje PHP es un lenguaje de programación estructurado, es decir, un lenguaje con variables, sentencias condicionales, bucles, funciones.... No es un lenguaje de etiquetas como HTML, XML o WML. Está mas cercano a otros lenguajes como JavaScript o a C.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP se ejecuta en el servidor y el resultado se envía al navegador. El resultado es normalmente una página HTML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario el navegador lo soporte, es independiente del navegador, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

La ventaja que tiene PHP sobre otros lenguajes de programación que se ejecutan en el servidor es que nos permite intercalar las sentencias PHP en las páginas HTML.

Vamos a ver un ejemplo sencillo para comprenderlo mejor.

```
<html>

<body>

Parte de HTML normal.
<BR><BR>

<?php
     echo "Parte de PHP<br>";

     for($i=0;$i<10;$i++)
     {
          echo "Línea ".$i."<br>";
     }
?>

</body>
</html>
```

El código PHP ejecutado tiene dos partes: la primera imprime "Parte de PHP" y la segunda es un bucle que se ejecuta 10 veces de 0 a 9, por cada vez que se ejecuta se escribe una línea, la variable $i contiene el número de línea que se está escribiendo.

2 VARIABLES

Una variable es un contenedor de información, en el que podemos meter números enteros, números decimales, caracteres, etc. El contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.

En PHP, los nombres de las variables comienzan con el símbolo del dólar $ y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres.

```
<html>

<body>

<?php
   $a = 1;
   $b = 3.34;
   $c = "Hola mundo.";
   echo $a,"<br>",$b,"<br>",$c;
?>

</body>

</html>
```

En este ejemplo hemos definido tres variables, $a, $b y $c y con la instrucción echo hemos mostrado el valor que contienen, insertando un salto de línea entre ellas.

3 OPERADORES ARITMÉTICOS

Los operadores de PHP son muy parecidos a los de C y JavaScript, si usted conoce estos lenguajes le resultaran familiares y fáciles de reconocer.
Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Resta	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	$a++	Suma 1 al contenido de una variable.
--	Resta 1	$a--	Resta 1 al contenido de una variable.

```
<html>

<body>

<?php
   $a = 8;
   $b = 3;
   echo $a + $b,"<br>";
   echo $a - $b,"<br>";
   echo $a * $b,"<br>";
   echo $a / $b,"<br>";
   $a++;
   echo $a,"<br>";
   $b--;
   echo $b,"<br>";
?>

</body>

</html>
```

4 OPERADORES DE COMPARACIÓN

Los operadores de comparación se usan para comparar valores y así poder tomar decisiones que hagan que el programa PHP tome un camino u otro.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
==	Igual	$a == $b	$a es igual $b
!=	Distinto	$a != $b	$a es distinto $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

```
<html>

<body>

<?php
   $a = 8;
   $b = 3;
   $c = 3;
   echo $a == $b,"<br>";
   echo $a != $b,"<br>";
   echo $a < $b,"<br>";
   echo $a > $b,"<br>";
   echo $a >= $c,"<br>";
   echo $b <= $c,"<br>";
?>

</body>

</html>
```

5 OPERADORES LÓGICOS

Los operadores lógicos se utilizan para combinar varias comparaciones.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

```
<html>

<body>

<?php
   $a = 8;
   $b = 3;
   $c = 3;
   echo ($a == $b) && ($c > $b),"<br>";
   echo ($a == $b) || ($b == $c),"<br>";
   echo !($b <= $c),"<br>";
?>

</body>

</html>
```

6 SENTENCIAS CONDICIONALES

Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado de evaluar una condición.

Sentencia if ... else

```
<?php
  if (condición)
  {
    Sentencias a ejecutar cuando la
    condición es cierta.
  }
  else
  {
    Sentecias a ejecutar cuando la
    condición es falsa.
  }
?>
```

La sentencia if ejecuta una serie de instrucciones u otras dependiendo de la condición que le pongamos. Probablemente sea la instrucción más importante en cualquier lenguaje de programación.

```
<html>

<body>

<?php
   $a = 8;
   $b = 3;
   if ($a < $b)
   {
      echo "a es menor que b";
   }
   else
   {
      echo "a no es menor que b";
   }
?>

</body>

</html>
```

En este ejemplo la condición no es verdadera por lo que se ejecuta la parte de código correspondiente al else.

Sentencia switch ... case

```
switch(variable) {

  case valor1:  instrucciones
    break;

  case valor2:  instrucciones
    break;

    .
    .
    .

  default:    instrucciones

}
```

```
<html>
<body>

<?php
   $posicion = "arriba";

   switch($posicion) {

      case "arriba":   // Bloque 1
         echo "La variable contiene";
         echo " el valor arriba";
         break;

      case "abajo":   // Bloque 2
         echo "La variable contiene";
         echo " el valor abajo";
         break;

      default:   // Bloque 3
         echo "La variable contiene otro valor";
         echo " distinto de arriba y abajo";
   }
?>

</body>
</html>
```

Con la sentencia switch podemos ejecutar unas u otras instrucciones dependiendo del valor de una variable, en el ejemplo anterior, dependiendo del valor de la variable $posicion se ejecuta el bloque 1 cuando el valor es "arriba", el bloque 2 cuando el valor es "abajo" y el bloque 3 si no es ninguno de los valores anteriores.

7 BUCLES

Los bucles nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

Sentencia while

```
<?php
  while (condición)
  {
    intrucciones a ejecutar.
  }
?>
```

Mientras la condición sea cierta se reiterará la ejecución de las instrucciones que están dentro del while.

```
<html>

<body>

Inicio<BR>

<?php
   $i=0;
   while ($i<10)
   {
      echo "El valor de i es ", $i,"<br>";
      $i++;
   }
?>

Final<BR>

</body>

</html>
```

En el ejemplo, el valor de $i al comienzo es 0, durante la ejecución del bucle, se va sumando 1 al valor de $i de manera que cuando $i vale 10 ya no se cumple la condición y se termina la ejecución del bucle.

Sentencia for

```
<?php
  for (inicial ; condición ; ejecutar en iteración)
  {
    intrucciones a ejecutar.
  }
?>
```

```
<html>

<body>

Inicio<BR>

<?php
   for($i=0 ; $i<10 ; $i++)
   {
      echo "El valor de i es ", $i,"<br>";
   }
?>

Final<BR>

</body>

</html>
```

El primer parámetro del for se ejecuta la primera vez y sirve para inicializar la variable del bucle, el segundo parámetro indica la condición que se debe cumplir para que el bucle siga ejecutándose y el tercer parámetro es una instrucción que se ejecuta al final de cada iteración y sirve para modificar el valor de la variable de iteración.

8 MANEJO DE CADENAS DE CARACTERES

El tratamiento de cadenas de caracteres es muy importante en PHP. Existen muchas funciones para el manejo de cadenas, a continuación se explican las más usadas.

Nombre y parámetros	Descripción
strlen(cadena)	Nos devuelve el número de caracteres de una cadena.
split(separador,cadena)	Divide una cadena en varias usando un carácter separador.
substr(cadena, inicio, longitud)	Devuelve una subcadena de otra, empezando por inicio y de longitud longitud.
chop(cadena)	Elimina los saltos de línea y los espacios finales de una cadena.
strpos(cadena1, cadena2)	Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.
str_replace(cadena1, cadena2, texto)	Reemplaza la cadena1 por la cadena2 en el texto.

```
<html>
<body>

<?php
   echo strlen("12345"),"<br>";

   $palabras=split(" ","Esto es una prueba");
   for($i=0;$palabras[$i];$i++)
      echo $palabras[$i],"<br>";

   echo substr("Devuelve una subcadena de otra",9,3),"<br><br>";

   if (chop("Cadena \n\n ") == "Cadena")
      echo "Iguales<br><br>";

   echo strpos("Busca la palabra dentro de la frase",
"palabra"),"<br><br>";

   echo str_replace("verde","rojo","Un pez de color verde, como
verde es la hierba."),"<br>";
?>

</body>
</html>
```

9 ENVÍO Y RECEPCIÓN DE DATOS

El lenguaje PHP nos proporciona una manera sencilla de manejar formularios, permitiéndonos de esta manera procesar la información que el usuario introduce.

Al diseñar un formulario debemos indicar la página PHP que procesará la información que se introduce en él, así como en método por el que se le pasará la información a la página.

```
<html>

<body>

<H1>Ejemplo de procesado de formularios</H1>

Introduzca su nombre:
<FORM ACTION="procesa.php" METHOD="GET">
<INPUT TYPE="text" NAME="nombre"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>

</body>

</html>
```

Al pulsar el botón **Enviar**, el contenido del formulario se envía a la página que indicamos en el atributo ACTION de la etiqueta FORM.

En versiones anteriores a la 4.2.0, PHP creaba una variable por cada elemento del FORM, esta variable creada tenía el mismo nombre que el cuadro de texto de la página anterior y el valor que habíamos introducido.

Este método sigue funcionando bien, pero si queremos mejorar la seguridad, para acceder a las variables del formulario utilizaremos el array de parámetros $_POST[] o $_GET[] dependiendo del método usado para enviar los parámetros.

En este ejemplo se ha creado una entrada en el array $_GET[] con el índice 'nombre' y con el valor que haya introducido el usuario.

```
<html>

<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido es: <?php echo $_GET['nombre'] ?>
<br>
</body>

</html>
```

Métodos GET y POST

Hemos comentado anteriormente que los datos de un formulario se envía mediante el método indicado en el atributo METHOD de la etiqueta FORM, los dos métodos posibles son GET y POST.

El resultado final es el mismo, solo que con el método GET podemos ver los parámetros pasados ya que están codificados en la URL.

10 ENVÍO DE CORREOS ELECTRÓNICOS

PHP nos ofrece la posibilidad de enviar correos electrónicos de una manera sencilla y fácil, para ello el lenguaje nos proporciona la instrucción mail( ).

```
<?php
  mail(destinatario, asunto, texto del mensaje);
?>
```

En el parámetro **destinatario** pondremos la dirección de correo a donde se enviará el mensaje, en el parámetro **asunto**, el tema o subject del mensaje y el parámetro texto del mensaje el cuerpo del mensaje en formato texto plano.

Existe una sintaxis extendida de la instrucción mail( ) que nos permite añadir información adicional a la cabecera del mensaje.

```
<?php
    mail(destinatario, asunto, texto del mensaje, información adicional de cabecera);
?>
```

En la información de cabecera podremos incluir parámetros adicionales al mensaje como Reply-To:, From:, Content-type:... que nos permiten tener un mayor control sobre el mensaje.

```
<html>

<body>

Introduzca su dirección de correo:
<FORM ACTION="email.php" METHOD="GET">
<INPUT TYPE="text" NAME="direccion"><BR><BR>
Formato: <BR>
<INPUT TYPE="radio" NAME="tipo" VALUE="plano" CHECKED> Texto
plano<BR>
<INPUT TYPE="radio" NAME="tipo" VALUE="html"> HTML<BR><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>

<html>
```

```
<body>

<?
   $direccion=$_GET['direccion'];
   $tipo=$_GET['tipo'];

   if ($direccion!=""){
   if ($tipo=="plano"){

      // Envío en formato texto plano

   mail($direccion,"Ejemplo de envío de email","Ejemplo de envío
de email de texto plano\nUsando PHP\n.\n","FROM: Pruebas
<periquillo@yahoo.com>\n");
   } else {

      // Envio en formato HTML

    mail($direccion,"Ejemplo de envío de email", "<html> <body>
Ejemplo de envío de email en HTML<br><br>usando <b>PHP</p>.
</body></html>","Content-type: text/html\n", "FROM: Pruebas
<periquillo@yahoo.com>\n");
   }

echo "Se ha enviado un email a la direccion: ",$direccion," en
formato <b>",$tipo,"</b>.";
}
?>
<br>
</FORM>

</body>

</html>
```

11 EJERCICIOS

1. Escribir un programa en PHP que sume dos números introducidos por teclado.

2. Escribir un programa en PHP que sume, reste, multiplique y divida dos números introducidos por teclado.

3. Escribir un programa en PHP que calcule el área de un rectángulo.

4. Escribir un programa en PHP que calcule el área de un triángulo.

5. Escribir un programa en PHP que calcule las soluciones de una ecuación de segundo grado.

6. Realiza un conversor de euros a pesetas y de pesetas a euros.

7. Escribir un programa en PHP que detecte si se han introducido en orden creciente tres números introducidos por el usuario.

8. Escribir un programa en PHP que determine si un número leído desde el teclado es par o impar.

9. Escribir un programa en PHP que dado un número del 1 a 7 escriba el correspondiente nombre del día de la semana.

10. Escribir un programa en PHP que dada una letra (A,B,C,D,...) indique su posición en el alfabeto(1, 2, 3, 4,...).

11. Escribir un programa en PHP que una vez leída una hora en formato (horas, minutos, segundos) indique cual será la hora dentro de un segundo.

12. Escribir un programa en PHP que calcule el salario semanal de un trabajador en base a las horas trabajadas y el pago por hora trabajada. Horas ordinarias (40 primeras horas de trabajo) 12 euros/hora. A partir de la hora 41, se pagan 16 euros/hora .

13. Mostrar los números múltiplos de 5 de 0 a 100 utilizando un bucle for.

14. Mostrar los números múltiplos de 5 de 0 a 100 utilizando un bucle while.

15. Escribir un programa en PHP que diga si un número introducido por teclado es o no primo.

12 CONEXIÓN A UNA BASE DE DATOS DE MySQL DESDE PHP

Desde una página con código escrito en PHP nos podemos conectar a una base de datos de MySQL y ejecutar comandos en lenguaje SQL (para hacer consultas, insertar o modificar registros, crear tablas, dar permisos, etc).

En el siguiente ejemplo vemos cómo nos podemos conectar a la base de datos banco y extraemos los datos de un registro que cumple una condición.

```
 <html>
 <body>
 <?php
 $conexion = mysql_connect("localhost", "root");
 mysql_select_db("banco", $conexion);
 $consulta = mysql_query("SELECT * FROM empleado WHERE
dni=\"13579\"", $conexion);
 echo "Nombre: ".mysql_result($consulta, 0, "nombre")."<br>";
 echo "Cargo: ".mysql_result($consulta, 0, "cargo")."<br>";
 echo "Sueldo :".mysql_result($consulta, 0, "sueldo")."<br>";
 ?>
 </body>
 </html>
```

En primer lugar nos encontramos con la función mysql_connect(), que abre una conexión con el servidor MySQL en el Host especificado (en este caso la misma máquina en la que está alojada el servidor MySQL, localhost). También debemos especificar un usuario (root, personal, etc.), y si fuera necesario una contraseña para el usuario indicado (mysql_connect("localhost", "root", "clave_del_root")). Si la conexión ha tenido éxito, la función mysql_connect() devuelve un identificador de dicha conexión (un número) que es almacenado en la variable $conexion, sino ha tenido éxito, devuelve 0 (FALSE).

Con mysql_select_db(), PHP le dice al servidor que en la conexión $conexion nos queremos conectar a la base de datos restaurante. Podríamos establecer distintas conexiones a la BD en diferentes servidores.

La siguiente función mysql_query(), es la que ejecuta la sentencia propiamente dicha, usando el identificador de la conexión ($conexion), envía una sentencia SQL al servidor MySQL para que éste la procese. El resultado de ésta operación se almacena en la variable $consulta.

Finalmente, mysql_result() se usa para mostrar los valores de los campos devueltos por la consulta ($consulta). En este ejemplo mostramos los valores del registro 0, que es el primer registro, y mostramos el valor de los campos especificados.

13 MOSTRAR UNA CONSULTA COMPLETA

Ahora mostraremos una consulta completa, es decir, todas las filas que la componen. Para ello, deberemos recorrer con un bucle todos los valores de la variable $consulta.

```
<html>
<body>
<center>
<h2>
Base de datos <u>banco</u><br>
Tabla <u>cliente</u><br>
</h2>

<?php

$conexion = mysql_connect("localhost", "root");
mysql_select_db("banco", $conexion);
$consulta = mysql_query("SELECT dni, nombre, direccion, telefono
FROM cliente", $conexion);

echo "<table border = '1'> \n";
echo "<tr> \n";
echo "<td><b>DNI</b></td> \n";
echo "<td><b>Nombre</b></td> \n";
echo "<td><b>Dirección</b></td> \n";
echo "<td><b>Teléfono</b></td> \n";
echo "</tr> \n";

while ($registro = mysql_fetch_array($consulta)){
echo "<tr> \n";
echo "<td>".$registro[dni]."</td> \n";
echo "<td>".$registro[nombre]."</td> \n";
echo "<td>".$registro[direccion]."</td> \n";
echo "<td>".$registro[telefono]."</td> \n";
echo "</tr> \n";
}

echo "</table> \n";

?>
</center>
</body>
</html>
```

La sentencia de control while(), que tiene un funcionamiento similar al de otros lenguajes, ejecuta una cosa mientras la condición sea verdadera. En esta ocasión while() evalúa la función mysql_fetch_array(), que devuelve un array con el contenido del registro actual (que se almacena en $registro) y avanza una posición en la lista de registros devueltos en la consulta SQL.

Hay que destacar la utilización del punto (.), como operador para concatenar cadenas.

14 BUSCAR UN VALOR

Veremos ahora un script que sirve para buscar una determinada cadena (que recibiremos de un formulario, y la almacenamos en la variable $buscar), dentro de nuestra base de datos, concretamente dentro del campo "nombre".
campo "nombre".

En primer lugar escribiremos el texto HTML de la página web que nos servirá como formulario de entrada, la llamaremos formulario.html

```
<html>
<body>
<form method = "POST" action = "http://localhost/buscador.php">
<strong>Texto a buscar dentro del campo nombre:</strong>
<input type="text" name="buscar" size="20"><br><br>
<input type="submit" value="Buscar">
</form>
</body>
</html>
```

El siguiente script de búsqueda lo llamaremos buscador.php, y será el encargado de hacer la búsqueda en la base de datos, y de devolver por pantalla los registros encontrados.

```
<html>
<body>
<?php
$conexion = mysql_connect("localhost", "root");
mysql_select_db("banco", $conexion);
$sql = "SELECT * FROM cliente WHERE nombre LIKE '%$buscar%'
ORDER BY nombre";
$resultado = mysql_query($sql, $conexion);

if ($registro = mysql_fetch_array($resultado)){
echo "<table border = '1'> \n";
echo "<tr> \n";
echo "<td><b>DNI</b></td> \n";
echo "<td><b>Nombre</b></td> \n";
echo "<td><b>Dirección</b></td> \n";
echo "<td><b>Teléfono</b></td> \n";
echo "</tr> \n";

do {
echo "<tr> \n";
echo "<td>".$registro[dni]."</td> \n";
echo "<td>".$registro[nombre]."</td> \n";
echo "<td>".$registro[direccion]."</td> \n";
echo "<td>".$registro[telefono]."</td> \n";
echo "</tr> \n";
} while ($registro = mysql_fetch_array($resultado));
echo "<p><a href=formulario.html>Volver</p> \n";
```

```
echo "</table> \n";
} else {
echo "<p>¡No se ha encontrado ningún registro!</p>\n";
echo "<p><a href=formulario.html>Volver</p> \n";
}
?>
</body>
</html>
```

Lo más importante de este script, es sin duda la sentencia SQL que le enviamos al servidor MySQL, y más concretamente la condición que le imponemos, WHERE nombre LIKE '%$buscar%'. Con la sentencia LIKE buscamos cualquier ocurrencia de la cadena contenida en $buscar, mientras que con los signos de porcentaje (%) indicamos el lugar de la coincidencia, por ejemplo, si hubiesemos puesto nombre LIKE '%$buscar', buscaríamos cualquier ocurrencia al final del campo"nombre", mientras que si hubiesemos puesto nombre LIKE '$buscar%', buscariamos cualquier ocurrencia al principio del campo "nombre".

15 EJERCICIO

Crea una aplicación web (con varias páginas en html y/o php) que permita hacer listado, alta, baja y modificación sobre la tabla **cliente** de la base de datos **banco**.

- Para realizar el **listado** bastará un SELECT, tal y como se ha visto en los ejemplos anteriores.

- El **alta** se realizará mediante un formulario donde se especificarán todos los campos del nuevo registro. Luego esos datos se enviarán a una página que ejecutará un INSERT.

- Para realizar una **baja**, se pedirá el DNI del cliente mediante un formulario y a continuación se ejecutará un DELETE en otra página.

- La **modificación** se realiza mediante un UPDATE. Se pedirá previamente en un formulario el DNI del cliente para el que queremos modificar los datos.