Curso PHP Básico - Capítulo 1.B

Estructuras de Control

Las estructuras de control de flujo son instrucciones que permiten controlar y alterar el orden en que se ejecuta un programa. Sin estas instrucciones, un programa se ejecutará de principio a fin sin posibilidad de evaluar datos y condiciones para cambiar las cosas. Las estructuras de control permiten, entre otras cosas, ejecutar ciertas instrucciones basadas en una condición o el valor de una variable, repetir una acción mientras se cumpla otra, etc.

- Estructura IF

La forma de una estructura if en PHP no difiere demasiado en la forma en que esta estructura se presenta en otros lenguajes y es la siguiente:

```
<?php
if (expr)
        sentencia
?>
```

Básicamente lo que hace el IF es decirle al lenguaje: “si pasa o se cumple X condición/ situación, hacé esto (la sentencia).

La expresión es evaluada a su valor booleano. Si la expresión se evalúa como TRUE, PHP ejecutará la sentencia y si se evalúa como FALSE la ignorará.

Para obtener más información sobre transformación de expresiones a valores booleanos podemos recurrir a la documentación oficial de PHP:

http://www.php.net/manual/es/language.types.boolean.php

Recordemos que, como en otros lenguajes, si queremos ejecutar un conjunto de sentencias debemos agruparlas entre llaves de la siguiente manera:

```
<?php
if (expr)
{
        sentencia1;
        sentencia2;
```

```
        sentencia3;
}
?>
```

Para verlo en un ejemplo en concreto

```
<?php
$tipo_de_documento = 2;
if($tipo_de_documento == 2)
        echo “El tipo de documento es DNI”;
else
        echo “El tipo de documento es otro”;
?>
```

Como podemos ver si la sentencia a ejecutar es una sola en el ejemplo no hizo falta utilizar llaves. El else le indica a PHP qué hacer si NO se cumple la condición del IF.

También podemos utilizar expresiones anidadas de la siguiente manera (vemos el uso del operador AND (&&).

```
<?php
$tipo_de_documento = 2;
$nacionalidad = “ARG”;

if($tipo_de_documento ==  2 && $nacionalidad == “ARG”)
        echo “El tipo de documento es DNI y la nacionalidad es ARGENTINO”;
else
        echo “El tipo de documento es otro y/ó la nacionalidad no es ARGENTINO”;
?>
```

prestar particularmente atención en el “y/ó”.

Para ver todos los tipos de Operadores de comparación que nos ofrece el PHP podemos ingresar a la documentación oficial:
http://www.php.net/manual/es/language.operators.comparison.php

- Estructura elseif/else if

elseif, como su nombre lo sugiere, es una combinación de if y else. Del mismo modo que else, extiende una sentencia if para ejecutar una sentencia diferente en caso que la expresión if original se evalúe como FALSE. Sin embargo, a diferencia de else, esa expresión alternativa sólo se ejecutará si la expresión condicional del elseif se evalúa como TRUE. Por ejemplo, el siguiente código debe mostrar a es mayor que b, a es igual que b o a es menor que b:

```
<?php
if ($a > $b) {
        echo "a es mayor que b";
} elseif ($a == $b) {
        echo "a es igual que b";
} else {
        echo "a es menor que b";
}
?>
```

- Estructura SWITCH

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, es posible que se quiera comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a que valor es igual. Para esto es exactamente la expresión switch.

Podemos ver como se aplica en el siguiente ejemplo:

```
<?php
if ($i == 0) {
   echo "i es igual a 0";
} elseif ($i == 1) {
   echo "i es igual a 1";
} elseif ($i == 2) {
   echo "i es igual a 2";
}
```

Si usamos el switch, esto queda:

```
switch ($i) {
   case 0:
```

```
    echo "i es igual a 0";
    break;
  case 1:
    echo "i es igual a 1";
    break;
  case 2:
    echo "i es igual a 2";
    break;
}
?>
```

También la estructura nos permite el uso de strings para comparar:

```
<?php
switch ($i) {
  case "manzana":
    echo "i es una manzana";
    break;
  case "barra":
    echo "i es una barra";
    break;
  case "pastel":
    echo "i es un pastel";
    break;
}
?>
```

Es importante entender cómo la sentencia switch es ejecutada con el fin de evitar errores. La sentencia switch ejecuta línea por línea (en realidad, sentencia por sentencia) desde el momento en que una sentencia case es encontrada con un valor que coincide con el valor de la sentencia switch. PHP continúa ejecutando las sentencias hasta el final del bloque switch, o hasta la primera vez que vea una sentencia break. Si no se escribe una sentencia break al final de la lista de sentencias de un caso, PHP seguirá ejecutando las sentencias del caso siguiente. Por ejemplo:

```
<?php
switch ($i) {
  case 0:
    echo "i es igual a 0";
```

```
  case 1:
    echo "i es igual a 1";
  case 2:
    echo "i es igual a 2";
}
?>
```

Aquí, si $i es igual a 0, PHP ejecutaría todas las sentencias echo! Si $i es igual a 1, PHP ejecutaría las últimas dos sentencias echo. Se obtendría el comportamiento esperado (se mostraría 'i es igual a 2') sólo si $i es igual a 2. Por lo tanto, es importante no olvidar las sentencias break (aunque es posible que se desee evitar proporcionarles a propósito bajo determinadas circunstancias).

- Estructura WHILE

La sentencia while tiene la siguiente forma

```
<?php
while (expr)
  sentencia
?>
```

Este tipo de sentencias hace las veces de un bucle mientras la expresión que se ejecuta en while se evalúa como TRUE, es decir, la sentencia (o grupo de sentencias) es ejecutada tantas veces como la expresión dentro de los paréntesis del while sea evaluada como TRUE.

```
<?php
/* ejemplo 1 */

$i = 1;
while ($i <= 5) {
        echo $i;
        echo '<br/>';
        $i++;
}
?>
```

En el ejemplo anterior la salida en pantalla es la siguiente:

1
2
3
4
5

Cuando se utilicen este tipo de sentencias es importante prestar particular atención en no caer en bucles infinitos como es el siguiente:

```
<?php
/* ejemplo 1 */

$i = 10;
while ($i != 8) {
        echo “este bucle es infinito“;
}
?>
```

- Estructura FOR

La estructura for en PHP tiene la siguiente forma:

```
<?php
for (expr1; expr2; expr3)
   sentencia
?>
```

La primer expresión es ejecutada al principio del bucle, luego al comienzo de cada iteración se ejecuta la segunda expresión y si se evalúa como TRUE el bucle itera una vez más, de lo contrario finaliza su ejecución. Luego, al final de cada iteración se ejecuta la expresión 3.

Para ver en un ejemplo práctico:

```
<?php
for ($i = 1; $i <= 10; $i++) {
   echo “El valor de la variable $i es: “;
   echo $i;
   echo “<br/>”;
}
```

```
?>
```

Analicemos que pasa con el ejemplo anterior, al comienzo del bucle en la variable $i se asigna el valor 1 (esto se ejecuta una única vez). Luego, se evalúa la expresión 2 y como es verdadera se ejecutan las dos sentencias encerradas entre llaves y cuando termina la ejecución de estas sentencias se evalúa la tercer expresión lo que hace que $i tome el valor 2. Luego se ejecuta la expresión 2 y se vuelve a iterar una vez más el bucle ya que $i sigue siendo menor o igual que 10 y así sucesivamente hasta que $i llega al valor 11 y corta la iteración.

El resultado en pantalla es el siguiente:

```
El valor de la variable $i es: 1
El valor de la variable $i es: 2
El valor de la variable $i es: 3
El valor de la variable $i es: 4
El valor de la variable $i es: 5
El valor de la variable $i es: 6
El valor de la variable $i es: 7
El valor de la variable $i es: 8
El valor de la variable $i es: 9
El valor de la variable $i es: 10
```

- Estructura FOREACH

Es un modo simple de iterar para recorrer un array, sirve solamente para arrays y objetos. Si no es de este tipo lanzará un error.

Puede ser utilizado de dos maneras:

```
<?php
foreach ($array as $valor)
   sentencias

?>
```

y otra forma más completa:

```
<?php
foreach ($array2 as $clave => $valor)
   sentencias
```

```
?>
```

En ambas definiciones se recorre el array, se asigna en $valor el elemento actual. En la segunda, es un poco más completa, devuelve la clave en donde se aloja dicho elemento dentro del array (La cual puede ser vacía o numérica autogenerada).

Palabras reservadas

Php tiene una serie de “palabras reservadas”. No se puede usar ninguna de las siguientes palabras como constantes, nombres de clases, nombres de funciones o métodos. Se pueden usar como nombres de variables, pero podría dar lugar a confusiones. Estas palabras son las palabras que se reservan para el lenguaje:

endfor
endforeach
endif
endswitch
endwhile
eval()
exit()
extends
final
for
foreach
function
global
goto
if
implements
include
include_once
instanceof
insteadof
interface
isset()
list()
namespace
new
or
print
private
protected
public
require

require_once
return
static
switch
throw
trait
try
unset()
use
var
while
xor
yield

Anexo - Operadores Lógicos

Nombre	Ejemplo	Respuesta
Y	$a && $b	true si ambos son true
Y	$a and $b	true si ambos son true
O	$a || $b	true si uno de los dos es true
O	$a or $b	true si uno de los dos es true
O exclusivo	$a xor $b	true si sólo uno de los dos es true, no si ambos lo son
Negado	!$b	true si $b es false, false si $b es true

Cuando combinamos operaciones es conveniente utilizar paréntesis como una buena práctica.

Los operadores and / && y los operadores or / || no son completamente equivalentes, la diferencia es que no tienen la misma precedencia.
Los operadores && y || tienen mayor prioridad que and y or.
A su vez el operador de asignación = tiene una prioridad menor, se pueden producir situaciones inesperadas...

Por ejemplo:

```
<?php
   $x = true;
   $y = false;
   $res = $x && $y;
   if ($res) {
      echo “Verdadero”;
   } else {
      echo “Falso”;
   }
?>
```

```
// Esto devuelve Falso
```

Pero…

```
<?php
   $x = true;
   $y = false;
   $res = $x and $y;
   if ($res) {
      echo “Verdadero”;
   } else {
      echo “Falso”;
   }
?>


// Esto devuelve Verdadero
```

Esto ocurre porque el operador de asignación = tiene precedencia sobre el operador and. Esto quiere decir que PHP realiza antes la asignación que la operación lógica, es como si la expresión estuviese escrita: `$res = ($x) and $y;  // Verdadero`

Entonces, si queremos que se de el mismo resultado con && y con and, debemos encerrar la operación lógica completa entre paréntesis.

De esta manera: `$res = ($x and $y);` es igual a `$res = ($x && $y);`