INSTITUTO SAN IGNACIO DE LOYOLA

Estructuras de Control

1. Introducción

Las estructuras de control permiten bifurcar el flujo del programa y así ejecutar unas partes u otras del código según ciertas condiciones. PHP dispone de todas las estructuras clásicas de los lenguajes de alto nivel, con la sintaxis de C, C++ o Java, y además algunas otras estructuras más típicas de lenguajes interpretados como Perl o Bash.

En todos los casos, las estructuras de control contienen una expresión cuya evaluación como verdadero ó falso determinará el flujo a seguir dentro de la estructura. Estas expresiones pueden ser una variable, una función (el valor que devuelve), una constante, o cualquier combinación de éstas con los operadores.

2. if ... elseif ... else

La instrucción **if** es la estructura de control más básica de todas. En su forma más simple, su sintaxis es esta:

```
if (expresión) {
   instrucciones
}
```

Si expresión se evalúa como verdadero, se ejecutan las *instrucciones* y después se sigue ejecutando el resto del programa. Si se evalúa como falso, no se ejecutan las *instrucciones* y continúa con el resto del programa.

Ejemplo 01:

Archivo: php0401.html

```
<form method = "post" action = "php0402.php">
   <b>Tu Nombre:</b>
   <input type="text" name="nombre" size="20">
   <br><input type="submit" value="Enviar">
</form>
```

Archivo: php0402.php

```
<?php
   if ( isset($nombre) ){
      if ($nombre){
         echo "Hola $nombre<br>";
      }
   }
?>
<input type="button" value="Back" onClick="history.back()">
```

A una instrucción **if** le podemos añadir instrucciones que se ejecuten cuando la condición es falsa mediante la cláusula **else**:

```
if (expresión) {
   instrucciones_verdadero
} else {
   instrucciones_falso
}
```

Si *expresión* se evalúa como *verdadero*, se ejecutan *instrucciones_verdadero*. Si se evalúa como *falso*, se ejecuta *instrucciones_falso*. En ambos casos luego se ejecuta el resto de instrucciones que sigan a la instrucción **if**.

Ejemplo 02

Archivo: php0403.html

```
<h1>Identificación de Usuario</h1>
<form method = "post" action = "php0404.php">
   <b>Usuario:</b>
   <input type="text" name="usuario" size="20">
   <br><b>Clave:</b>
   <input type="text" name="clave" size="20">
   <br><input type="submit" value="Enviar">
</form>
```

Archivo: php0404.php

```
<?php
   if (!isset($usuario) ){
      die("<h1>Tiene que usar el formulario.</h1>");
   }
   if ($usuario=="claudia" and $clave=="gatita"){
      echo "datos correctos";
   }else{
      echo "error en su identificacion";
   }
?>
<br><input type="button" value="Back" onClick="history.back()">
```

Por último, podemos encadenar varias condiciones con la cláusula **elseif**, de la siguiente forma:

```
if (expresion1) {
   instrucciones1
} elseif (expresion2) {
   instrucciones2
} elseif (expresion3) {
   instrucciones3
}
...
elseif (expresionN) {
   instruccionesN
} else {
   instruccionesElse
}
```

Ejemplo 03

Archivo: php0405.html

```
<h1>Conoce el Perú</h1>
<form method = "post" action = "php0406.php">
  <b>Ciudad a visitar:</b>
  <select size="1" name="ciudad">
    <option value="1">Chiclayo</option>
    <option value="2">Trujillo</option>
    <option value="3">Cajamarca</option>
    <option value="4">Iquitos</option>
    <option value="5">Huaraz</option>
    <option value="6">Huancayo</option>
    <option value="7">Arequipa</option>
    <option value="8">Cuzco</option>
  </select>
  <br><input type="submit" value="Enviar">
</form>
```

Archivo: php0406.html

```
<?php
  echo "Ciudad seleccionada: $ciudad<br>";
  if($ciudad=="1"){
    echo "Chiclayo, ciudad de la amistad.";
  }elseif($ciudad=="2"){
    echo "Trujillo, ciudad de la eterna primavera.";
  }elseif($ciudad=="3"){
    echo "Cajamarca, simplemente una ciudad espectacular.";
  }elseif($ciudad=="4"){
    echo "Iquitos, no te podras olvidar de sus encantos";
  }elseif($ciudad=="5"){
    echo "Huaraz, Sus nevados son impresionantes.";
  }elseif($ciudad=="6"){
    echo "Huancayo, ciudad INCONTRASTABLE.";
  }elseif($ciudad=="7"){
    echo "Arequipa, lo mejor es su gente.";
  }elseif($ciudad=="8"){
    echo "Cuzco, quedaras encantado.";
  }
?>
<br><input type="button" value="Back" onClick="history.back()">
```

3. switch

Su sintaxis es:

```
switch (variable) {
   case valor1:
        instrucciones1
   case valor2:
        instrucciones2
   ...
   case valorN:
        instruccionesN
   default:
        instruccionesDefault
}
```

Cuando se termina de ejecutar el código de un **case**, si no se finaliza el **switch** explícitamente con una instrucción **break**, se continúa ejecutando el código del siguiente case aunque no se cumpla la condición, hasta que se llegue al final del bloque **switch** o se finalice este con un **break**.

Ejemplo 4

```
<?php
   switch ($i) {
     case 1:
         echo "Código del 1";
     case 2:
         echo "Código del 2";
     case 3:
         echo "Código del 3";
         break;
     case 4:
         echo "Código del 4";
   }
?>
```

Si **$i** vale 1, se imprimirán las tres primeras cadenas; Si vale 2, la segunda y la tercera; Si vale 3, sólo la tercera; Y si vale 4, sólo la última.

Ejemplo 5

```
<?php
   switch ($i) {
     case 0:
     case 1:
     case 2:
     case 3:
         echo "i es menor que 4, pero no negativa";
         break;

     case 4:
         echo "i vale 4";
         break;

     default:
         echo "i mayor que 4 o negativa";
         break;
   }
?>
```

4. while y do .. while

Las estructuras de control **while** y **do .. while** representan bucles que se ejecutan mientras se verifique una determinada condición.

La estructura de un bucle **while**:

```
while (expresión) {
    instrucciones
}
```

La estructura **do .. while** puede ser vista como una variante de **while** en la que la comprobación de la condición se realiza al final de cada iteración del bucle en lugar de al principio. Con esto lo que se consigue es que al menos la iteración se realice siempre una vez, aunque *expresión* se evalúe como *falso*.

La estructura de **do .. while** es:

```
do {
    instrucciones
} while (expresión);
```

5. break y continue

La sentencia **break** nos permite salir inmediatamente de una estructura de control **while**, **for** o **switch**.

Ejemplo 6

```
<?php
   $a = 0;
   while ($a < 10) {
      if ($a == 5) {
         break;
   }
   $a++;
}
?>
```

El bucle **while** finaliza en la iteración *$a = 5*, pese a que la condición del **while** es *$a < 10*.

Después del **break** podemos especificar un parámetro, el número de niveles de bucles anidados de los que queremos salir. Por defecto es uno (salir del bucle más interno).

Ejemplo 7

```
<?php
   $a = 0;
   while ($a < 10) {
      $b = 0;

      while ($b < 5) {
         if ($b == 2) {
            break; // Equivale a "break 1", sale del while b
         }
      }

      while ($b < 5) {
         if ($a == 3 && $b == 3) {
            break 2; // Saldría de los DOS bucles.
         }
      }
   }
?>
```

Por su parte, la instrucción **continue** lo que hace es saltarse el resto de la iteración actual, y pasar directamente a la siguiente:

Ejemplo 8

```
<?php
   $a = 0;
   while ($a < 5) {
     if ($a == 2) {
         continue;
     }
     echo "\$a vale $a.";
   }
?>
```

En el ejemplo se saltaría la instrucción **echo** de la iteración *$a* = *2*, y el resultado sería:

```
$a vale 0
$a vale 1
$a vale 3
$a vale 4
```

6. for

Los bucles **for** son los más complejos de que dispone PHP. Su estructura es la misma que en C:

```
for (expresión1; expresión2; expresión3) {
   instrucciones
}
```

Donde:

expresion1 Es la iniciación del bucle. Generalmente da un valor inicial a una o varias variables (separadas por comas). Sólo se ejecuta una vez, al principio, cuando el flujo del programa llega al bucle.

expresión2 Es la condición. Mientras que expresión2 se evalúe como verdadera, el bucle estará iterando. Se evalúa al inicio de cada iteración, y si no se verifica la condición la siguiente iteración ya no se realiza y finaliza el bucle, continuando la ejecución del programa con el resto del código de después del **for**.

expresión3 Es el paso de iteración. Se ejecuta después de cada iteración, y generalmente modifica el valor de alguna variable (separadas por comas si hay más de una).

Cualquiera de las tres expresiones puede estar vacía, aunque en este caso debemos tener cuidado de realizar su función en el cuerpo del bucle.

Ejemplo 09

```
<?php
   $factorial5 = 1;
   for ($i = 2; $i <= 5; $i++ ) {
     $factorial5 *= $i;
   };
   echo $factorial5;
?>
```

Este ejemplo calcula la factorial de 5 (5!).

Ejemplo 10

```
<?php
   for ($factorial5 = 1, $i = 2; $i <= 5; $i++ ) {
     $factorial5 *= $i;
   }
   echo $factorial5;
?>
```

En este ejemplo podemos apreciar como en *expresion1* podemos inicializar varias variables separándolas con comas.

Ejemplo 11

```
<?php
   for ($factorial5=1, $i=2; $i<=5; $factorial5*=$i, $i++);
   echo $factorial5;
?>
```

En este ejemplo podemos apreciar como en *expresión3* también podemos operar sobre varias variables separándolas con comas, con lo que podremos encerrar todo el código del bucle en la línea **for**.

7. foreach

El bucle **foreach** representa una estructura de control típica de lenguajes interpretados como *Perl* y *Bash*, en la que a una variable se le van asignando todos los valores de una lista. Su sintaxis es esta:

```
foreach (array as $variable) {
   comandos
}
```

Ejemplo 12

```
<?php
   $a = array (1, 2, 3, 17);
   foreach ($a as $v) {
     print "Valor actual de \$a: $v.\n";
   }
?>
```

El resultado sería:

```
Valor actual de $a: 1
Valor actual de $a: 2
Valor actual de $a: 3
Valor actual de $a: 17
```