Control de Flujo

Estructuras de Control

Las estructuras de control determinan el flujo del código dentro de una aplicación. Nuestro código de programación deberá estar sometido a diferentes estructuras que PHP soporta como así también la mayoría de los lenguajes de programación. En este módulo veremos todas las estructuras de control disponibles para PHP, tales como bucles repetitivos, sentencias condicionales, etc.

Operadores de Comparación

Expresiones

Algunos operadores de expresión en PHP

> mayor que

< menor que

<= menor igual que

>= mayor igual que

== igual que

!= distinto de

! negación

Operadores de Igualdad

Los operadores de igualdad se utilizan para comparar dos valores. Vean las siguientes pruebas de equivalencia.

Ejemplo	Descripcion	Resultado
$a == $b	es equivalente a	Verdadero si $a y $b son equivalentes
$a != $b	no es equivalente a	Verdadero si $a no es equivalente a $b
$a === $b	es identico	Verdadero si $a y $b son equivalente del mismo tipo de dato
$a !== $b	no es idéntico	Verdadero si $a y $b no son equivalentes y del mismo tipo de dato

Uno de los errores más comunes que se generan con los operadores de comparación en PHP, es utilizar el signo = (igual) para comparar, ya que en algunos lenguajes funciona correctamente. En PHP el símbolo igual no compara

sino que asigna. Para comparar se utiliza el == (doble igual) o el === (triple igual) tal como lo explica el cuadro anterior.

Ejemplo de equivalencia y comparación estricta

$x = (1 == ‘1’); // $x vale TRUE

$x = (1 == true); // $x vale TRUE

$x = (1 == 1); // $x vale TRUE

$x = (1 === ‘1’); // $x vale FALSE, ya que no coincide el tipo de dato

$x = (1 === true); // $x vale FALSE, ya que no coincide el tipo de dato

$x = (1 === 1); // $x vale TRUE

Operadores Lógicos

Los operadores de comparación (operadores lógicos) proporcionan un método de flujo de programa directo a través de un examen de los valores comparativos de dos o más variables.

Ejemplo	Descripcion	Resultado
$a < $b	menor a	Verdadero si $a es menor que $b
$a > $b	mayor a	Verdadero si $a es mayor que $b
$a <= $b	menor o igual a	Verdadero si $a es menor o igual que $b
$a >= $b	mayor o igual a	Verdadero si $a es mayor o igual que $b
($a == 12) ? 5 : -1	ternario	Si $a es igual a 12, el valor de retorno es 5; de lo contrario, el valor devuelto es -1

Ejemplo de la utilización de operadores

// Esta expresión devuelve Verdadero. A la variable $x se le asigna el valor TRUE

$x = (7 > 5)

// Esta expresión devuelve falso. A la variable $x se le asigna el valor FALSE

$x = !(7 > 5) // el operador not ! evalúa la negación de una expresión, equivale a (7 > 5 == false)

Atención: Los operadores generalmente se utilizan dentro de sentencias condicionales que explicaremos a continuación.

Sentencias Condicionales

Introducción

Las sentencias condicionales hacen posible que el programa pueda responder en consecuencia a una amplia variedad de entradas, usando la lógica para discernir entre diversas condiciones de valor. Las sentencias condicionales son un elemento básico de todos los lenguajes de programación, incluido PHP.

Características

- Son aquellas sentencias que colocando una condición producen distintas alternativas.
- Cada alternativa depende del resultado de la condición o expresión lógica.
- Permite la ejecución condicional de fragmentos de código.

Sentencia IF

La estructura de control “if” permite decidir entre dos opciones resultantes de la evaluación de una sentencia. O sea, se ejecuta un bloque de código delimitado por llaves: **{** y **}** si y solo si la evaluación de la condición (expresión Booleana) es positiva (osea da como resultado TRUE). También podemos especificar acciones para realizar en caso de que la evaluación sea negativa.

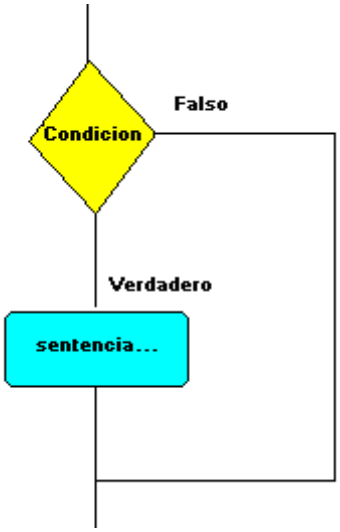
Cuando en un caso se procesa la declaración, la expresión de las pruebas son evaluadas, y el resultado se interpreta como un valor booleano. Si la prueba es verdadera, se ejecuta la sentencia contenida dentro de ella. Si la prueba es falsa, simplemente se procede con la ejecución de la siguiente declaración. Se debe tener en cuenta que una declaración en esta sintaxis puede ser una sola que termina con un punto y coma, o un bloque de instrucciones entre llaves u otro constructor condicional (que en sí se considera como una declaración individual).

```
if (expresionBooleana)

{

sentencias1;

}
```



Características:

- Es una sentencia condicional porque la ejecución de un bloque de código esta condicionada.
- Colocando una condición lógica se producen dos alternativas (Verdadero o Falso).
- Permite la ejecución condicional de un bloque de código delimitado entre llaves **{ }**
- Su característica es similar a la del lenguaje C.
- Los condicionales pueden anidarse entre sí a una profundidad arbitraria.

Estructura de la Sentencia

```
<?php

if (condición _ lógica)

{

   fragmento de código PHP

}

?>
```

Ejemplo Completo de la Sentencia if

```
<?php

$tiempo = "llueve";
```

```
if ( $tiempo == "llueve" ) echo "Ir a buscar paraguas" ;

?>
```

Ejemplo Sentencia if con operador OR

```
<?php

$usuario = "Carlos" ;

if ( $usuario == "Carlos" or $usuario == "Jose" )
{
      echo "Bienvenido ". $usuario . "<BR> \n"  ;
}
else
{
      echo "Usuario inválido <BR> \n"  ;
}

?>
```

Ejemplo Sentencia if con operador AND

```
<?php

$usuario = "Carlos" ;
```

```
$password = "123" ;

if ( $usuario == "Carlos" and $password == "123" )
{
	echo "Bienvenido ". $usuario . "<BR> \n"  ;
}
else
{
	echo "Usuario inválido <BR> \n"  ;
}

?>
```

Ejemplo completo de la utilización de operadores

```
<?php
$titulo = "Extensiones" ;
$encabezado = "Ejemplo de Extensiones" ;
?>
<html>
<head>
<title> <?php echo $titulo; ?> </title>
</head>

<body>
<h1 align="center"> <? echo $encabezado; ?>  </h1>
<BR>

<?php
$num1 = 10;
```

```
$num1 = 15;

if ( $num1 == $num2 )
{
    echo "Los números son iguales" . "<br>\n" ;
}

if ( $num1 < $num2 )
{
    echo "El primer número es menor que el segundo número" . "<br>\n" ;
}
else
{
    echo "El segundo número es mayor que el primer número" . "<br>\n" ;
}

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

Sentencia ELSE

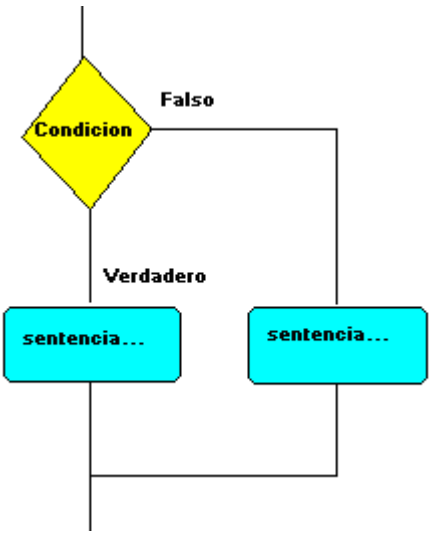
Si la prueba no es cierta y hay una cláusula más, la segunda declaración se ejecuta. La sentencia else sólo es ejecutada si la expresión if es evaluada como falsa (FALSE).

Características:

- Es la extensión de la sentencia IF
- Significa “de lo contrario” y permite la ejecución de un bloque de código si la condición de la sentencia IF fue falsa.
- La sentencia ELSE se ejecuta solamente si la expresión IF se evalúa como falsa.

Ejemplo de la sentencia ELSE

```
<?php
 if ($x > $y) {
    print $x . " es mayor que " . $y;
 } else {
    print $x . " NO es mayor que " . $y;
 }
?>
```



Ejemplo Completo de la sentencia ELSE

```
<?php
$edad = 22;

if ( $edad < 18 )
{
      echo "Usted es menor de edad" ;
```

```
}

else

{

      echo "Usted es ya es mayor de edad" ;

}


?>
```

Sentencia ELSEIF

Es una combinación entre if y else. Es una sentencia con su propia condición lógica en caso de que la sentencia IF anterior no haya sido válida.

La sentencia **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.

Se pueden anidar sucesivos elseif

Estructura de la Sentencia elseif

```
<?php

if (expresión)

{

   fragmento de código PHP

}

elseif (expresión)

{

   fragmento de código PHP

}

elseif (expresión)
```

```
{
   fragmento de código PHP
}
else
{
   fragmento de código PHP
}

?>
```

Sentencia Switch

Para un determinado tipo de ramificación de múltiples caminos, el Switch puede ser útil. La expresión puede ser una variable o cualquier otro tipo de expresión, siempre y cuando se evalúa a un valor simple (es decir, un número entero, un doble, o una cadena). La construcción se ejecuta mediante la evaluación de la expresión y luego se prueba el resultado para cada valor de la igualdad según sea el caso. Tan pronto como se encuentra un valor coincidente, las declaraciones posteriores se ejecutan en secuencia hasta la declaración (break;) o hasta el final del interruptor de la construcción (break también se puede utilizar para salir de un bucle construcciones).

Características:

- Es una sentencia condicional.
- Permite la ejecución condicional de un bloque de código.
- Su principal característica es que permite establecer muchas opciones según el valor de una variable (o expresión).
- A diferencia de la sentencia IF que solo hay dos opciones (Verdadero – Falso).
- Su característica es similar a la del lenguaje C.

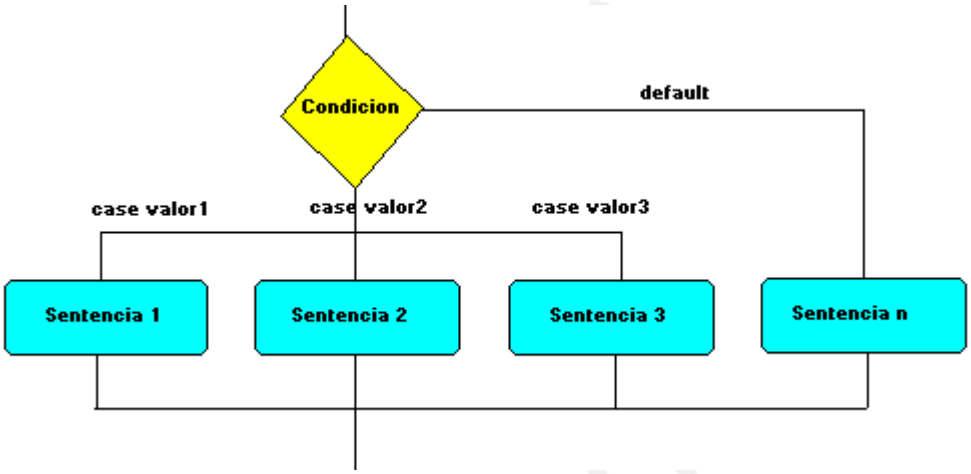
Estructura de la Sentencia

```
<?php
switch (variable)
{
```

```
    case valor1:
        sentencia;
        break;

    case valor2:
        sentencia;
        break;

    case valor3:
        sentencia;
        break;
}
?>
```



La condición se evalúa sólo una vez y el resultado se compara a cada sentencia case.

Switch puede ser más rápido que utilizar “elseif”

Ejemplo Completo de la sentencia SWITCH

```
<?php
 $x = 1;
```

```
 switch ($x) {
    case 0:
        echo "El valor de x = 0";
        break;
    case 1:
        echo "El valor de x = 1";
        break;
    case 2:
        echo "El valor de x = 2";
        break;
 }
?>
```

Ejemplo Completo de la sentencia SWITCH con DEFAULT

```
<?php
 $nombre = "Juan";

 switch ($nombre) {
    case "Miguel":
        echo "El nombre es Miguel";
        break;
    case "Martin":
        echo "El nombre es Martin";
        break;
    default:
```

```
        echo "El nombre no es Miguel ni Martin";
        break;
 }
?>
```

Ejemplo Completo de la sentencia SWITCH con DEFAULT II

```
<?php
 $nombre = "Miguel";

 switch ($nombre) {
    case "Miguel":
    case "Martin":
        echo "El nombre es Martin o Miguel";
        break;
    default:
        echo "El nombre no es Miguel ni Martin";
        break;
 }
?>
```


Estructuras de Control (Bucles)

Introducción

Muchas veces necesitamos ejecutar un mismo bloque de código una repetida cantidad de veces. Un bucle repite la

ejecución de un conjunto de sentencias en un número determinado de veces. Se puede determinar la cantidad de iteraciones y el lenguaje garantiza que no se repita más que eso.

Características

- Permiten repetir la ejecución de un bloque determinado de código
- Son para ejecutar n cantidad de veces una porción de código de programa.
- Se puede ejecutar n cantidad de veces o hasta que se cumpla una condición especifica.
- Existen tres tipos de Bucles Repetitivos

Bucle FOR

La sentencia FOR realiza un bucle determinado número de veces. La construcción del bucle tiene la siguiente sintaxis:

Si sintaxis es: “for (expresión1; expresión2; expresión3)”

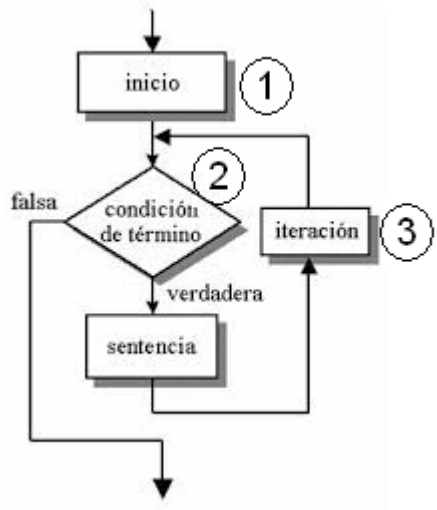
Está compuesta por tres partes:

1) Iniciación: Inicialización de la variable que utilizaremos en la condición (incondicionalmente se ejecuta solo una vez al principio del ciclo).

2) Condición: Indica la condición por la cual finalizará el bucle (se evalúa esta expresión al comienzo de cada iteración).

3) Incremento: Modificación de la variable inicializada en el paso 1). Es fundamental el incremento para no entrar en un bucle infinito (se ejecuta al final de cada iteración).

En primer lugar la iniciación de la expresión se evalúa una sola vez (o sea el punto 1). Luego por cada iteración se evalúa la condición (o sea el punto 2), si es falso, la instrucción FOR llega al fin, y si es verdadero, la instrucción continua su ejecución. Por último, se ejecuta el punto 3) al final del bucle y el ciclo comienza de nuevo.



Ejemplo Completo del bucle FOR

```
<?php

 for ($i=0; $i<10; $i++ )

 {

        $cuadrado = $i * $i;

        echo "El numero es: " . $i . " y su cuadrado es: " . $cuadrado ." <BR>\n";

 }

?>
```

Bucle While

Permite ejecutar un bloque de código determinado en forma repetitiva “mientras” (While en inglés) se cumpla la condición. La condición se evalúa al inicio del ciclo.

El bucle while evalúa una expresión de condición lógica (osea Booleano), si es cierto, se ejecuta la declaración y luego comienza de nuevo mediante la evaluación de la condición. Si la condición es falsa, el ciclo termina.

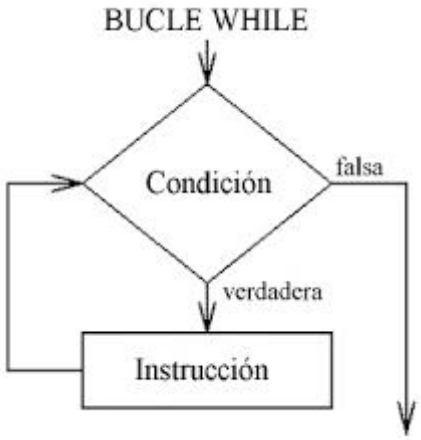
Si la condición es válida, ingresa en el ciclo y ejecuta el código que pusimos en el mismo delimitado por llaves: { }

Cuando deja de cumplirse la condición, el programa sale del while y continúa con el resto del programa.

Su sintaxis:

```
while( condición )

{        sentencias;        }
```



Ejemplo Completo del bucle WHILE

```
<?php

 $numero = 1;

 while ( $numero <= 10 )

 {

       echo "El numero es: " . $numero . " es menor que 10 <BR>\n";

       $numero++;

 }

?>
```

Bucle Do While

Esta sentencia es igual a la anterior con la diferencia que se evalúa al final. La condición se evalúa al final del ciclo.

La declaración se ejecuta una vez, y luego se evalúa la expresión. Si la expresión es verdadera, se repite la declaración hasta que la expresión se convierte en falsa. La única diferencia práctica entre el anterior y do-while es que éste siempre ejecutará su declaración, al menos, una vez.

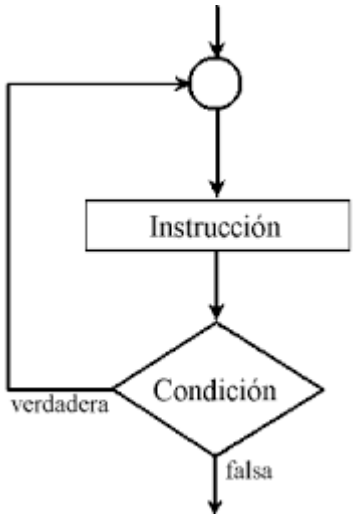
Características:

El bloque de código se ejecuta al menos una vez.

Cuando deja de cumplirse la condición, el programa sale del do while y continúa con el resto del programa.

Su sintaxis:

```
do
{       sentencias;
}
while( condición )
```



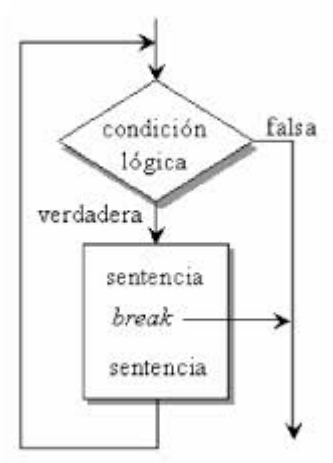
Ejemplo Completo del bucle DO WHILE

```
<?php
 $numero = 1;
 do
 {
      echo "El numero es: " . $numero . " es menor que 10 <BR>\n";
```

```
        $numero++;
 }
 while ( $numero <= 10 )
?>
```

Break y Continue

La manera estándar para salir de una estructura de bucle es que la condición de prueba principal se convierta en falsa. Los comandos especiales de break y continue ofrecen una salida lateral opcional de todos los bucles de construcciones, incluyendo al mismo tiempo: for, while y do-while.



El comando break sale del bucle interno de la construcción que lo contiene.

El comando continue salta hasta el final de la iteración actual del bucle más interno que lo contiene.

```
<?php
 $numero = 1;
 do
 {
        echo "El numero es: " . $numero . " es menor que 10 <BR>\n";
        if ($numero == 5 ) {
```

```
            echo "Mitad del ciclo";

            break;

        }


        $numero++;

 }

 while ( $numero <= 10 );

?>
```

Bucles Infinitos

Un bucle infinito es un bucle sin límites que se repite hasta que alguna condición se convierte en verdadera (o falsa), ya que la condición que depende de la acción tiene una sentencia break donde corta la iteración ilimitada. Los límites de los bucles son predecibles, mientras que los bucles sin límites puede ser tan complicado como se desee.

Resumen

La sentencia IF...ELSE permite ejecutar un bloque de instrucciones si la condición es verdadera y otro bloque de instrucciones si ésta es falsa. La condición que evalúa ha de estar encerrada entre paréntesis.

La sentencia IF...ELSEIF...ELSE permite ejecutar varias condiciones en cascada.

Otra alternativa es la sentencia SWITCH, la cuál evalúa y compara cada expresión de la sentencia CASE con la expresión. Si se llega al final de la lista de CASE y se encuentra una condición Verdadera, ejecuta el código de bloque que haya en DEFAULT. Si encontramos una condición verdadera debemos ejecutar un BREAK para que la sentencia SWITCH no siga buscando en la lista de CASE.

La sentencia WHILE ejecuta un bloque de código mientras se cumpla una determinada condición.

El bucle FOR no es estrictamente necesario, cualquier bucle FOR puede ser sustituido fácilmente por otro WHILE. Sin embargo, el bucle FOR resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. El bucle FOR también se puede romper mediante la sentencia BREAK.