PHP-4

Estructuras de control

INTRODUCCIÓN

- Un programa PHP se ejecuta en principio de forma secuencial, desde la primera instrucción hasta la última y de una en una. Las estructuras de control permiten modificar este flujo, eligiendo entre instrucciones alternativas o repitiendo instrucciones.

Bloques de instrucciones

- Para indicar a PHP que varias instrucciones forman un bloque de instrucciones que se ejecutarán secuencialmente (aunque un bloque pueda a su vez contener estructuras de control), se utilizan las llaves({}) .
- No es necesario escribir un punto y coma (;) después de las llaves y, en general, se recomienda no hacerlo.

Bloques de instrucciones

- Los bloques de instrucciones se emplean en las estructuras de control que se comentan en este tema (if ... elseif ... else, for, while, do ... while o foreach).
- No se suelen utilizar bloques cuando no hay estructuras de control. Si se quiere indicar visualmente al programador que unas instrucciones están relacionadas entre sí, basta con dejar una línea en blanco en el código fuente, o añadir líneas de comentarios.

```php
<?php
print "<p>Hola</p>\n";
print "<p>¿Qué tal?</p>\n";

print "<p>Recuerdos a la familia</p>\n";
print "<p>Adios</p>\n";
?>
```

```php
<?php
// Saludo
print "<p>Hola</p>\n";
print "<p>¿Qué tal?</p>\n";

// Despedida
print "<p>Recuerdos a la familia</p>\n";
print "<p>Adios</p>\n";
?>
```

Sentencia condicional if ... elseif ... else ...

- permite condicionar la ejecución de un bloque de sentencias al cumplimiento de una condición.
- La sintaxis de la construcción if más sencilla es la siguiente:

if (condición) { bloque_de_sentencias }

La condición se evalúa siempre, si el resultado es **true** se ejecuta el bloque de sentencias y si el resultado es **false** no se ejecuta el bloque de sentencias.

Sentencia condicional if ... elseif ... else ...

- La construcción if se puede complicar añadiendo la instrucción else:

if (condición) { bloque_de_sentencias_1 } else { bloque_de_sentencias_2 }

- La condición se evalúa siempre, si el resultado es **true** se ejecuta solamente el bloque de sentencias 1 y si el resultado es **false** se ejecuta solamente el bloque de sentencias 2.

Sentencia condicional if ... elseif ... else ...

- La construcción if ... else se puede extender añadiendo la instrucción elseif:

```
if (condición_1)
{ bloque_de_sentencias_1 }
elseif (condición_2)
{ bloque_de_sentencias_2 }
else { bloque_de_sentencias_3 }
```

Sentencia condicional if ... elseif ... else ...

- La condición 1 se evalúa siempre, si el resultado es **true** se ejecuta solamente el bloque de sentencias 1 y solamente si el resultado es **false** se evalúa la condición 2, si el resultado de esta es **true** se ejecuta solamente el bloque de sentencias 2 y si el resultado es **false** se ejecuta solamente el bloque de sentencias 3. En cualquier caso solamente se ejecuta uno de los tres bloques de sentencias.
- Se pueden añadir tantas instrucciones elseif como se desee, teniendo en cuenta que en cualquier caso solamente se ejecuta uno de los bloques de sentencias.

Notación abreviada para algunos if else: .. ? … : …

- Si la construcción if ... else ... simplemente asigna valor a una variable

if (condición_1) { $variable = expresión_1; }

else $variable = expresión_2; }

- Esta expresión se puede sustituir por la construcción:

$variable = (condición_1)? expresión_1 : expresión_2;

- que se suele escribir en varias líneas para facilitar la legibilidad:

```
$variable = (condición_1)
? expresión_1
: expresión_2;
```

Sentencia condicional switch

- La sentencia switch es equivalente a una construcción if ... elseif ... en las que las expresiones son comparaciones de igualdad de la misma condición con valores distintos.
- La sintaxis de la sentencia switch es la siguiente:

```
switch (expresión_1)
{
case valor_1: bloque_de_sentencias_1; break;
case valor_2: bloque_de_sentencias_2; break;
...
case valor_n: bloque_de_sentencias_n; break;
 }
```

Bucle for

- La sintaxis del bucle for es la siguiente:

for (expresión_inicial; condición_continuación; expresión_paso)

{ bloque_de_sentencias }

- La expresión inicial se evalúa siempre. La condición de continuación se evalúa al principio de cada iteración: si el resultado es **true** se ejecuta primero el bloque de sentencias.

Bucle for

- A continuación se evalúa la expresión de paso y finalmente se evalúa nuevamente la condición de continuación; si el resultado es **false** el bucle se termina.
- Cuando se programa un bucle for hay que tener cuidado en que la condición de continuación vaya a dejar de cumplirse en algún momento, porque si no es así, el bucle no terminaría nunca.

Bucle for

- En los bucles for más sencillos, en la expresión inicial se inicializa una variable que se evalúa en la expresión final y que se modifica en la expresión de paso, como muestra el ejemplo siguiente:

```
<?php
for ($i = 0; $i < 5; $i++)
 { print "<p>$i</p>\n"; }
 ?>
```

Bucle while

- La sintaxis del bucle while es la siguiente:

while (condición_entrada)

{ bloque_de_sentencias }

La condición se evalúa al principio de cada iteración: si el resultado es **true** se ejecuta el bloque de sentencias; si el resultado es **false** el bucle se termina.

Bucle while

- Cuando se programa un bucle while hay que tener cuidado en que la condición deje de cumplirse en algún momento, porque si no es así, el bucle no terminaría nunca.
- En los bucles while más sencillos, antes del bucle se inicializa una variable que se evalúa en la condición y dentro del bucle se modifica la variable, como muestra el ejemplo siguiente:

```
<?php
 $i = 0; while ($i < 5)
{ print "<p>$i</p>\n"; $i++; }
?>
```

Bucle do ... while

- La sintaxis del bucle do ... While es la siguiente:
- do { bloque_de_sentencias } while (condición)
- La condición se evalúa al final de cada iteración: si el resultado es **true** se ejecuta el bloque de sentencias; si el resultado es **false** el bucle se termina.
- El bloque de sentencias se ejecuta por lo menos una vez mientras que en el bucle while depende de si la condición es cierta o no la primera vez que se evalúa.

Bucle do ... while

```
<?php
 $i = 0;
do
{ print "<p>$i</p>\n"; $i++; }
 while ($i < 5)
?>
```

- antes del bucle se inicializa una variable que se evalúa en la condición y dentro del bucle se modifica la variable, como muestra el ejemplo siguiente:

Bucle foreach

- El bucle foreach es muy útil para recorrer matrices cuyo tamaño se desconoce o matrices cuyos índices no son correlativos o numéricos (matrices asociativas).
- La sintaxis del bucle foreach más simple es la siguiente:

foreach ($matriz as $valor)

{ bloque_de_sentencias }

- El bucle ejecuta el bloque de sentencias tantas veces como elementos contenga la tabla $matriz y, en cada iteración, la variable $valor toma uno de los valores de la matriz.

Bucle foreach

```
<?php
 $matriz = [0, 1, 10, 100, 1000];
foreach ($matriz as $valor)
{ print "<p>$valor</p>\n"; }
?>
```

Bucle foreach

- La sintaxis del bucle foreach puede también ser la siguiente:

foreach ($matriz as $indice => $valor)

{ bloque_de_sentencias }

- El bucle ejecuta el bloque de sentencias tantas veces como elementos contenga la tabla $matriz y, en cada iteración, la variable $valor toma uno de los valores de la matriz y la variable $indice toma como valor el índice correspondiente.

Bucle foreach

- En el ejemplo siguiente se define una matriz asociativa y se imprimen sus valores:

```
<?php
$matriz = ["red" => "rojo", "green" => "verde",
"blue" => "azul"];
 foreach ($matriz as $indice => $valor)
{ print "<p>$indice : $valor</p>\n"; }
 ?>
```

ESTRUCTURAS DE CONTROL: BREAK

- Sentencia break:
  - break n;
- Nos permite salir de una estructura de control o bucle de manera directa.
- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.

ESTRUCTURA DE CONTROL: CONTINUE

- Sentencia continue:

  –continue [n];

- Nos permite abandonar la iteración vigente de una estructura de control.

- Si la acompañamos de un número entero (opcional), indicamos el número de niveles de la estructura anidada que queremos saltar.

Diferencias entre break y continue en PHP

- Son dos de las sentencias más utilizadas en PHP para **manipular el flujo de las iteraciones en las estructuras de control .**
- Ambas cortan el ciclo actual pero con una importante diferencia:
  - break finaliza la ejecución de la estructura control en curso.
  - continue finaliza la iteración actual de la estructura control y se inicia una nueva iteración.

Diferencias entre break y continue en PHP

- En otras palabras, continue le dice a PHP que la iteración actual se ha acabado y debe empezar en la evaluación de la condición que abre la estructura de control.
- Por su parte, break le dice a PHP que la evaluación de la estructura control actual ha terminado y que no siga haciendo iteraciones

```php
$letters = [ 'A', 'B', 'C' ];
foreach ( $letters as $letter) {
    if( 'A' == $letter ) {
        continue;
        echo 'Esto nunca se imprimirá';
    }
    echo $letter;
}
```

- Se imprimirá la cadena BC ya que cuándo $letter es igual a A la iteración no alcanza la sentencia echo $letter; sino que vuelve al principio del foreach.

```php
$letters = [ 'A', 'B', 'C' ];
foreach ( $letters as $letter) {
    if( 'A' == $letter ) {
        break;
        echo 'Esto nunca se imprimirá';
    }
    echo $letter;
}
```

- No se imprimirá nada ya que en la primera iteración, cuándo $letter es igual a A, se finaliza la ejecución de la estructura foreach y ninguno de los echo es alcanzado.

EJEMPLOS-ANIDADOS

- Cuándo break o continue se utilizan en una estructura de control anidada en otra se puede **especificar el número de estructuras a las que afectan**. El número por omisión es 1 y afecta sólo a la estructura actual

```php
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {  <--------------------┐
        continue;                 --- vuelve aquí -----┘
    }
}

// Esta estructura es igual a la anterior
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {  <--------------------┐
        continue 1;               --- vuelve aquí -----┘
    }
}
```

Pero podemos hacer que vuelva a la estructura `while` externa con `c`

```php
while ( $foo ) {                  <--------------------┐
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {
        continue 2;               --- vuelve aquí ----┘
    }
}
```

EJEMPLOS ANIDADOS

O con `break`:

```
while ( $foo ) {
    $items = [ 1, 2, 3 ];
    foreach( $items as $item ) {
        break 2;                    --- salta aquí -----┐
    }
}
                                <--------------------┘
```