Estructuras de control PHP

1. if (Simple)

La estructura de control **if** permite la ejecución condicional de fragmentos de código.

```php
<?php
$x=7;
$y=5;
if ($x > $y) {
    echo "$x es mayor que $y";
}
```

Las sentencias if se pueden incluir unas dentro de otras indefinidamente.

2. else (Dobles)

Sirve para ejecutar una sentencia cuando otra no se cumple. ***else*** extiende una sentencia ***if***, y se ejecuta cuando ésta es **false**. Siguiendo el ejemplo anterior:

```php
<?php
if ($x > $y) {
    echo "$x es mayor que $y";
} else {
    echo "$y es mayor que $x";
}
```

3. elseif/else if (compuestas)

Es una combinación entre ***if*** y ***else***. Se ejecuta cuando if es **false**, pero sólo si la expresión condicional del ***elseif*** es **true**.

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

Puede haber varios *elseif* dentro de una sentencia *if* (aunque en ese caso suele ser más recomendable usar *switch*). No hay diferencias de comportamiento entre *elseif* y *else if*.

elseif sólo se ejecuta si el *if* precedente o cualquier *elseif* anterior son evaluadas como **false**.

4. switch(Decisión Multiple)

switch es como una serie de sentencias *if*. Es útil para comparar una misma **variable o expresión** con valores direrentes y ejecutar un código diferente a otro dependiendo de esos valores.

```
<?php
switch ($i) {
    case "perro":
        echo "\$i es un perro";
        break;
    case "gato":
        echo "\$i es un gato";
        break;
    case "avestruz":
        echo "\$i es un avestruz";
        break;
}
```

Cuando una sentencia *case* coincide con el valor de la sentencia *switch*, PHP ejecuta el código dentro del *case*. PHP sigue ejecutando las sentencias hasta el final o hasta que choca con un *break*, que entonces finaliza la iteración. Si se omite *break*, swith ejecutará todos los *cases* restantes cuando encuentra uno que cumpla con la condición:

```
<?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";
}
```

En el ejemplo anterior, si se define a *$i* como 0, se mostrarán todos los *echo* restantes porque no hay ningún *break*. Si se define a *$i*como 1, se mostrarán los *echo* del *case* 1 y *case* 2. Si se define a *$i* como 2, sólo se mostrará el *case* 2.

Con ***switch*** la condición sólo se evalúa una vez, y su valor es comparado con cada uno de los *case*, a diferencia de lo que ocurre con ***elseif***, que la condión se va evaluando continuamente con el loop.

case puede no ejecutar ningún código, pero hace que se activen los cases posteriores hasta que se encuentre con un break:

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

En caso de que no haya ningún *case* válido, puede utilizarse ***default***, para ejecutar algo cuando no se cumplen los case:

```php
<?php
switch($i) {
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    default:
        echo "i no es ni 0 ni 1";
}
```

- En los case sólo se permiten **tipos simples**: *int*, *float* y *string*. Los **arrays** y **objetos** pueden utilizarse si se muestran como un tipo simple.
- Es posible escribir punto y coma ";" en lugar de dos puntos ":" después de un *case*.

5. while (Repita Mientras)

Es el tipo más sencillo de **loop-Lazo** en **PHP**. Se ejecutan las sentencias dentro del ***while*** siempre y cuando se evalúen como **true-verdadera**. El valor

de la expresión se comprueba cada vez al inicio del **loop**, y la ejecución no se detendrá hasta que finalice la **iteración** (cada vez que PHP ejecuta las sentencias en un loop es una iteración). Si la expresión ***while*** se evalúa como **false**, las sentencias no se ejecutarán ni siquiera una vez.

También es posible agrupar varias instrucciones ***while*** dentro de una.

```php
<?php
$i = 1;
while($i <= 10){
    echo $i;
    $i++;
}
```

6. do-while (Repita Hasta)

Muy similares a los loops ***while***, simplemente aquí la expresión para el loop se verifica al final en lugar de al principio, esto garantiza que el código se ejecute por lo menos la primera vez.

```php
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

Este loop se ejecutaría sólo una vez, ya que después no cumple la condición.

7. for (Repita Para)

Los loops ***for*** son los más complejos en PHP.

```php
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
} // Devuelve 123456789
```

- Las expresiones o conjunto de expresiones van separadas por punto y coma **;** y sólo hay 3.
- La primera expresión, **$i = 1**, se ejecuta una vez incondicionalmente al comienzo del bucle.
- La segunda expresión, **$i <= 10**, es una **condición**, si es true, se ejecutará la tercera expresión.

- La tercera expresión, **$i++**, es la acción a realizar si se cumple la segunda expresión.

Cada una de las expresiones pueden estar **vacías** o contener **múltiples expresiones**, lo que resulta útil en ciertas ocasiones. Si la expresión 2 está vacía, el loop será definido como **true**.

```php
<?php
// Todos los siguientes ejemplos son válidos y devuelven lo mismo,
123456789
// EJEMPLO 1
for($i = 1; $i <= 10; $i++) {
    echo $i;
}

// EJEMPLO 2
for ($i = 1; ; $i++){
    if($i > 10) {
        break;
    }
    echo $i;
}
```

Referecnias Digitales:

Diego Lazaro. Estructuras de control en PHP, 2016. [On line]: https://diego.com.es/estructuras-de-control-en-php