La instrucción continue

Si la instrucción *break* permite interrumpir el desarrollo de un bucle, mediante *continue* se puede *impedir* que, bajo unas condiciones determinadas, se ejecuten *algunas* o *todas* las instrucciones de un bucle sin que por ello se interrumpa la ejecución de las iteraciones siguientes.

Esta instrucción es aplicable tanto a bucles **for** como a los de tipo **while** o **do while**.

Seguramente los ejemplos nos ayudarán a aclarar un poquito más la idea. En todos ellos hay un **condicional** que contiene la función **continue**.

El primero de ellos (*un bucle* **for**) tiene como condición: **$i % 2 == 0**, que, como recordarás, significa que *el resto de la división de $i entre dos*($i % 2) sea igual (==) a cero.

En este supuesto (*condición de múltiplo de dos*) se activará la opción **continue** y por lo tanto en esa iteración no se ejecuta la instrucción **echo** o, lo que es lo mismo, *no se imprimirán en pantalla los múltiplos de 2*.

En el segundo ejemplo (caso de *bucle* **while**) la condición establecida para que se ejecute **continue** es que el valor de la variable **no sea múltiplo de tres**, en cuyo caso **echo** sólo imprimirá los **múltiplos de 3**.

El tercer ejemplo utiliza *un bucle* **do ... while** para presentar en pantalla los **múltiplos de 11**.

La instrucción continue *n*

La instrucción **continue** puede utilizar un parámetro **n** con valor **entero positivo** que cuando no se indica toma por defecto el valor **1**.

La idea es la siguiente. Cuando tenemos **bucles anidados** el intérprete de PHP los considera **numerados** correlativamente –**de dentro hacia fuera**– a partir de **UNO**.

Cuando es *ejecutada* **continue n** se *redirecciona la iteración* **al bucle**, cuyo **número coincide** con el valor de **n**.

Obviamente, el valor de n no puede ser nunca mayor que el número de bucles anidados en el script.

Analicemos los ejemplos que tenemos a la derecha.

En el primer caso el bucle **for** sería el **UNO** y el **while** sería el **DOS**.

Cuando se cumpla la condición que *activa* **continue 2**, se redirecciona la **iteración** al **paso siguiente del bucle DOS**, en el caso del ejemplo al paso siguiente de **while**.

En el segundo ejemplo, como puedes ver, hemos anidado a tres niveles y hemos escrito *continue 3*, aunque a la hora de ejecutar los ejemplos podrás ver las *tres variantes* posibles de ese script modificando los valores del **n** de **continue**.

Ejemplos de continue

```
<?
for ($i=0;$i<=10;$i++){

            #condicion de  multiplo de 2
        if ($i % 2 ==0 ) {
           continue ;
          }

        echo "La variable I vale ",$i,"<br>";
}
   ?>
```

ejemplo53.php

```
<?
$i = 0;
while ($i++ < 14) {

     #condicion de no multiplo de 3 usando para distinto la sintaxis !=

    if ($i % 3 !=0){
            continue ;
    }

    echo "El valor de i es: ",$i,"<br>";
}

  ?>
```

ejemplo54.php

```
<?
$i = 0;
do {

      # condicion de no multiplo de 11. fijate en la sintaxis alternativa
        # observa que aquí distinto lo hemos escrito <>

    if ($i % 11 <>0 ){
            continue ;
    }

    echo "El valor de i es: ",$i,"<br>";

}while ($i++ < 100)
?>
```

ejemplo55.php

Ejemplos de *continue n*

```
<?
$j=0;
while (++$j <5) {
        for($i=1;$i<5;$i++){

                if ($i==3){
                  continue 2;
                   }
              echo "El valor de j es: ",$j, " y el de i es: ",$i,"<br>";


        }
}
   ?>
```

Caso continue 1	Caso continue 2

```
<?
$j=0;$k=0;
```

Fíjate en un **matiz** importante. Cuando el intérprete lee la instrucción **for** por primera vez lo hace a partir del valor **inicial** de la variable que controla las iteraciones, pero ni **do... while** ni **while** tienen esa opción dado que por sí mismos no modifican las variables de control.
Estos trabajan con condiciones mientras que for lo hacen con su variable de control.

Esa es la razón por la que en los ejemplos de los casos **continue 1** y **continue 2** la variable **k** no pasa del valor **0**, ya que al sobrepasar **j** el valor **5**, el bucle **while** no se ejecuta.

Si quieres que esas variables se reinicien al modo de **for** tendrás que añadir –dentro del **if** que contiene el *continue* correspondiente y antes de *continue*– una línea donde asignes a esas variables su valor inicial.

```
do {
while (++$j <=5) {
        for($i=1;$i<=5;$i++){

        if ($i==2){
              continue 3;
                    }
        echo "El valor de k es: ",$k,
         " y el valor de j es: ",$j, " y el de i es: ",$i,"<br>";


                }
        }
}while ($k++ <=5);
   ?>
```

Caso continue 1	Caso continue 2	Caso continue 3