RESUMEN DE PHP III – Operadores

Los operadores sirven para realizar operaciones entre variables.

Operador de asignación:

El más básico es el operador de asignación **(=),** utilizado para dar valores a las variables que usamos en nuestro código.

```
<?php
        $variable = 5;
        $variable2 = "Asignación de valores";
?>
```

Operador unario:

Este operador **(-)** tiene la propiedad de hacer a los números, negativos o positivos, dependiendo del signo actual.

```
<?php
        $entero = 100;
        $entero_negativo = -$entero; // El valor ahora es -100
        $entero_positivo = -$entero_negativo; // El valorahora es 100
?>
```

Operadores aritméticos:

Este tipo de operadores forman parte de la aritmética básica.

Operador	Operación	Descripción	Ejemplo	Resultado
+	Suma	Suma entre valores	$x=2+2	4
-	Resta	Resta entre valores	$x=5-2	3
*	Multiplicación	Multiplica distintos valores	$x=5*5	25
/	División	Divide entre valores	$x=10/2	5
%	Módulo	Devuelve el resto de una división	$x=10%2	0
++	Incremento	Suma una unidad a determinado valor	$x=5; $x=++;	6
–	Decremento	Resta una unidad a determinado valor	$x=5; $x=–;	4

Operadores condicionales:

Los operadores condicionales se utilizan para comparar valores. Devuelven *true o false.*

Operador	Descripción	Ejemplo	Resultado
==	Igual: Compara dos valores (no compara el tipo de variable)	$a == $b	Verdadero si $a es igual a $b
===	Igual: Compara los dos valores si son iguales y del mismo tipo	$a === $b	Verdadero si $a es igual a $b y son el mismo tipo de variable
!=	Distinto: Compara si las variables son diferentes	$a != $b	Verdadero si $a es diferente de $b

<>	Distinto: Compara si las variables son diferentes	$a <> $b	Verdadero si $a es diferente de $b
!==	Distinto: Compara que las dos variables sean diferentes y de diferentes tipos	$a !== $b	Verdadero si $a y $b son diferentes y son diferentes tipos
<	Menor: Compara que la primera variable sea más pequeña que la segunda	$a < $b	Verdadero si $a es menor que $b
>	Mayor: Compara que la primera variable sea mayor que la segunda	$a > $b	Verdadero si $a es mayor a $b
<=	Menor o igual: Compara que la primera variable sea menor o igual que la segunda	$a <= $b	Verdadero si $a es menor o igual a $b
>=	Mayor o igual: Compara que la primera variable sea mayor o igual que la segunda	$a >= $b	Verdadero si $a es mayor o igual a $b

Operadores lógicos:

Durante el desarrollo de las aplicaciones podemos encontrar situaciones en las que necesite hacer varias comparaciones seguidas para que se cumpla una determinada condición. Para esto sirven los operadores lógicos.

Operador	Descripción	Ejemplo	Resultado
And	Compara que las dos variables sean verdaderas	$a && $b	Verdadero si $a y $b son verdaderos
Or	Compara que alguna de las dos variables sean verdaderas	$a || $b	Verdadero si $a o $b son verdaderos
Xor	Compara que $a o $b sean verdaderos pero no ambos	$a Xor $b	Verdadero si $a o $b son verdaderos pero no ambos
Not	Niega una condición	!$a	Verdadero si $a es falso

Operador ternario:

Los operadores que hemos visto hasta ahora son capaces de manejar un operando (unarios) o dos operandos (binarios). El operador ternario, o de comparación, evalúa un operando y, dependiendo de si es falso o verdadero, evalúa el segundo operando o el tercero.

```php
<?php
        $valor = false;
        $valor == true ? $resultado = "OK" : $resultado = "FALLO";

        // Si $value es true $resultado será OK
        // Si es false $re sultado será FALLO
        echo $resultado;
?>
```

Operadores de bit a bit:

Este tipo de operadores no vamos a verlos porque no suelen utilizarse mucho en las aplicaciones webs.

Operador de ejecución:

Mediante el operador de ejecución podemos ejecutar comandos del sistema en nuestras aplicaciones PHP. El operador es las comillas invertidas **(` `)**

```
<?php
        $listado_archivos = `ls-la`;
        echo $listado_archivos;
?>
```

Operador de supresión de errores:

Cuando algo falla en nuestras aplicaciones PHP el navegador informa de una serie de errores. Esta no es una buena práctica porque un atacante podría aprovecharse de los códigos de error para recabar información de como esta diseñado el interior de nuestras aplicaciones. Para suprimir los errores se utiliza el símbolo **(@)**.

Hace tiempo leí en un artículo de mejora de rendimiento de las aplicaciones, que este operador consumía bastantes recursos del servidor. Así que es bueno no utilizarlo muy a menudo. Si lo que queremos es no mostrar errores, podemos configfurar PHP para que no muestre ninguno mediante **error_reporting(0).**

```
<?php
        /*abrimos un fichero en modo lectura.
        Si el fichero no existe,  el navegador da un error de acceso.
        Mediante el operador @ no lo mostramos en el navegador. */
        $fichero = @fopen("prueba.txt","r");
?>
```