Desarrollo Full Stack

Fundación Telefónica Movistar

CURSO DE DESARROLLO FULL STACK.

Operadores relacionales.

Cuando definimos una condición que debe ser evaluada, damos por sentado que debe ser booleana (verdadera o falsa), no podemos poner condiciones que no cumplan esta regla.

Una condición es una afirmación, y esa afirmación por lo general toma la forma de una comparación no es más que comparar el valor real de una variable contra los posibles valores que puede tomar.

Para estas comparaciones, debemos hacer uso de los operadores de comparación (hasta ahora solo vimos algunos).

- == (Igual a)
- ! = (Distinto de)
- <> (Distinto de)
- < (Menor que)
- > (Mayor que)
- <= (Menor o igual)
- >= (Mayor o igual)
- === Idéntico
- !== No idéntico

Por ejemplo, una estructura IF que aún no vimos, pero cuya sintaxis no difiere mucho de lo visto en otros lenguajes, con 'distinto de':

```
$apellido = 'Alfaro';

if ($apellido != 'Alfaro'){

    echo 'Usted no es Guillermo';

}

else echo 'Bienvenido Guillermo';
```

Veremos por pantalla el mensaje "Bienvenido Guillermo".

Operadores lógicos.

Los operadores lógicos nos permiten unir dos o más condiciones en una sola.

Por ejemplo, si queremos comparar los valores de dos variables.

```
$var1 = 12;

$var2 = 23;

if ($var1 < $var2 and $var1 > 10){

    echo 'Se cumplen ambas condiciones';

}

else echo 'Una, o ambas condiciones, no se han cumplido';
```

A continuación, vemos los operadores lógicos que usamos en PHP:

- **And (conjunción, también puede escribirse como &&)**
- **Or (disyunción, también se escribe como ||)**
- **Xor (disyunción excluyente, también conocida como or excluyente)**
- **! (negación)**

La diferencia entre OR y XOR es que con OR, no importa cuál de las dos condiciones se cumpla, o si se cumplen ambas, mi resultado es VERDADERO.

En cambio, con XOR, mi resultado será VERDADERO solo si alguno de las dos condiciones se cumple, no ambas.

Por ejemplo:

```php
$var1 = 9;
$var2 = 23;
if ($var1 < $var2 or $var1 > 10){
    echo 'Condicion VERDADERA';
}
else echo 'Condicion FALSA';
```

Un ejemplo de XOR:

```php
$var1 = 12;
$var2 = 23;

if ($var1 < $var2 xor $var1 > 10){
    echo 'Condicion VERDADERA';
}
else echo 'Condicion FALSA';
```

Estructura IF.

Muchas veces vamos a querer que nuestro código decida automáticamente si ejecuta una parte de nuestro código u otra, por lo general en base a un determinado valor o a una acción realizada de parte del usuario.

Este tipo de decisiones son muy similares en su lógica a las que enfrentamos en la vida diaria, por ejemplo, 'si hace frío, entonces llevo una campera' o 'si me duele la cabeza, entonces tomo una aspirina'.

La estructura que nos permite que la ejecución de nuestro código sea condicional, es IF.

Su forma más básica es:

```php
$a =23;


if ($a == 23){
    // Si el resultado de la condicion es VERDADERO
    // se ejecuta esta parte del codigo
    echo 'Condicion VERDADERA';
}
```

En el ejemplo, asignamos el valor 23 a una variable, y después, consultamos, dentro de la estructura IF, si el valor es 23, en cuyo caso mostramos por pantalla un determinado mensaje.

Es decir, si la condición es VERDADERA, se ejecuta el código que pusimos dentro de la estructura.

Si la condición resultase FALSA, el programa sale de la estructura IF sin hacer nada.

Importante: para las comparaciones utilizamos el operador == (que se usa solo para comparar) a diferencia del operador = (que es un operador de asignación).

Estructura IF – ELSE.

A una estructura de decisión, en forma opcional, le podemos agregar un bloque de código que se ejecute si la condición resulta falta.

Para los casos en que la condición resultó falsa, disponemos de la instrucción ELSE que podemos incorporar dentro de la estructura de decisión:

```
$a =23;


if ($a == 23){
    // Si el resultado de la  ódigo ón es VERDADERO
    // se ejecuta esta parte del  ódigo
    echo 'Condicion VERDADERA';
}
    // Si el resultado de la  ódigo ón es FALSO
    // se ejecuta esta parte del  ódigo
else echo 'Condicion FALSA';
```

Si ejecutamos el código anterior, veremos por pantalla el mensaje "Condición VERDADERA".

La estructura IF termina siempre con un punto y coma.

El código encerrado dentro del IF y del ELSE, lleva corchetes si tiene más de una línea, también se puede usar para una sola línea, es una buena práctica.

Estructura ELSEIF.

Si necesitamos evaluar varias condiciones diferentes (siempre recordando que solo una puede ser verdadera) recurrimos al ElseIf.

```php
$a =23;


if ($a <= 10){
    echo 'El valor se encuentra entre 1 y 10.';
}
elseif ($a > 10 and $a <= 100){
    echo 'El valor se encuentra entre 11 y 100.';
}
else echo 'El valor es mayor a 100.';
```

Con el código anterior veremos por pantalla:

El valor se encuentra entre 11 y 100.

Podemos usar tantos elseif como sean necesarios, pero si nuestras condiciones son muchas, tal vez nos resulte más útil el uso de switch-case.

Estructura switch case.

Es una estructura condicional, como la del IF, pero a menudo la vamos a encontrar como 'estructura selectiva' ya que nos permite seleccionar uno de varios valores conocidos de una variable.

```php
$variable = 2;


// Estructura SWITCH CASE
switch($variable){
    case 1: $valor = 'Uno';break;
    case 2: $valor = 'Dos';break;
    case 3: $valor = 'Tres';break;
    default: $valor = 'El numero ingresado NO es valido';
}


echo $valor;
```

A continuación de la palabra switch, entre paréntesis, colocamos la variable que queremos evaluar y dentro de corchetes, el código que queremos ejecutar en base al valor obtenido.

Dentro de este código, vamos a incluir tantos CASE (cada valor posible) como sea necesario.

Por ejemplo, si recibimos el número del día almacenado en una variable, pero lo que en realidad necesitamos es el nombre, podemos hacerlo de la siguiente manera:

```php
<?php
    $dia = 13;
    switch ($dia) {
        case 1: $nombre_dia = 'Lunes';break;
        case 2: $nombre_dia = 'Martes';break;
        case 3: $nombre_dia = 'Miercoles';break;
        case 4: $nombre_dia = 'Jueves';break;
        case 5: $nombre_dia = 'Viernes';break;
        case 6: $nombre_dia = 'Sabado';break;
        case 7: $nombre_dia = 'Domingo';break;
        default: $nombre_dia = 'Valor incorrecto';break;
        };
    echo $nombre_dia;
?>
```

La estructura debería terminar con un punto y coma, pero algunos servidores admiten no colocarlo.

El break es porque, una vez ejecutado el código por la condición que suponemos verdadera, no tiene sentido seguir recorriendo la estructura, mediante esta instrucción, salimos.

Podría darse el caso que ninguno de los valores que incluimos en nuestra estructura cumpla con la condición, para evitar este dilema lógico, incluimos la sentencia DEFAULT, que se ejecutará solo en caso de que ninguna de las condiciones anteriores resulte verdadera.

Estructuras de repetición.

Un tipo muy útil de estructuras, son las de repetición.

En PHP tenemos tres tipos de estructuras:

- FOR
- WHILE
- DO WHILE

Estructura FOR.

Este tipo de estructura se utiliza por lo general, cuando tenemos exactamente determinadas el número de repeticiones que necesitamos hacer.

Podemos incluir una variable de control dentro de la estructura que se incrementa o disminuye en forma automática.

La sintaxis básica es la siguiente:

```
<?php
    for ($variable = valor_inicial; condición; incremento)
        { ... bloque de código; }
?>
```

Mientras la condición se verifique, se cumplirá con el bloque de código dentro de los corchetes.

Veamos un ejemplo sencillo:

```
// Estructura FOR
for ($x = 1; $x <= 10; $x = $x+1){
    echo 'Valor de X = '.$x.'<br>';
}
```

Al ejecutarlo, esto es lo que veremos por pantalla:

Valor de X = 1
Valor de X = 2
Valor de X = 3
Valor de X = 4
Valor de X = 5
Valor de X = 6
Valor de X = 7
Valor de X = 8
Valor de X = 9
Valor de X = 10

Esta estructura de repetición inicializa la variable $variable en 1, pone como condición para ejecutar el código que $variable sea menor o igual a 10, y por último el incremento, con cada ejecución del bucle, le aumento su valor en una unidad.

El resultado mostrado por pantalla serán los números del 1 al 10.

** El incremento $variable++ es lo mismo que poner $variable = $variable + 1

```
// Estructura FOR
for ($x = 1; $x <= 10; $x++){
    echo 'Valor de X = '.$x.'<br>';
}
```

Estructura While.

La instrucción While ejecuta un bloque de código mientras se cumpla una determinada condición. Cuando la condición deja de ser verdadera, se interrumpe la instrucción y se sigue ejecutando el resto del código de nuestra aplicación.

La condición es evaluada al inicio de la estructura.

La siguiente es la sintaxis básica:

```
<?php
    while (condición) {
        ... bloque de código;
    }
?>
```

Veamos un ejemplo sencillo:

```
// Estructura WHILE
$valor = 1;


while ($valor <= 6){
    echo 'Valor de X = '.$valor.'<br>';
    $valor++;
}
```

Esta estructura, nos muestra por pantalla los valores de la variable $valor siempre y cuando dicho valor sea menor o igual a 6.

Probemos de darle un valor mayor a la variable $valor, por ejemplo, le asignamos el valor 7, como la condición se evalúa antes de ejecutar el código, no se mostrará por pantalla ningún valor.

Estructura Do While.

La estructura Do While es similar a la estructura While, solo que en este caso, la condición se evalúa al final de la estructura en lugar de hacerse al principio.

Sintaxis básica:

```
<?php
    do {
        …sentencias
    } while(condición);
?>
```

Veamos un ejemplo sencillo:

```
// Estructura DO WHILE
$valor = 7;


do {
    echo 'Valor de X = '.$valor.'<br>';
    $valor++;
}while ($valor <= 6)
```

Le damos a la variable un valor igual a 7.

Luego se ejecuta el bloque de código, que muestra la variable por pantalla y luego le incrementa su valor.

El último paso es verificar la condición, si es falsa, se termina la ejecución.

Habíamos dicho que, en este tipo particular de estructura, primero se ejecuta el código y al final se valida la condición.

Vemos que, en este caso, si partimos de un valor del argumento que hace falsa a la condición, el código de todas formas se ejecuta, al menos una vez, antes de abandonar la estructura.

Expresiones.

En PHP disponemos del incremento o decremento, previo y posterior.

Veamos los siguientes ejemplos:

```
<?php
    // Primero muestra y después incrementa.
    $variable = 12;
    echo $variable++; // 12
    echo $variable; // 13
?>
```

En este ejemplo, asignamos un valor (12) a la variable, luego la mostramos por pantalla y acto seguido, la incrementamos en una unidad.

Veamos a continuación, un ejemplo similar, pero utilizando el decremento:

```
<?php
    // Primero muestra y después incrementa.
    $variable = 12;
    echo $variable--; // 12
    echo $variable; // 11
?>
```

Veamos a continuación, como podemos aplicar los cambios antes de mostrar el valor por pantalla:

```php
<?php
    // Primero incrementa y después muestra.
    $variable = 12;
    echo ++$variable; // 13
    echo $variable; // 13
?>
```

```php
<?php
    // Primero incrementa y después muestra.
    $variable = 12;
    echo --$variable; // 11
    echo $variable; // 11
?>
```

También podemos aplicar estas variaciones en valores diferentes a la unidad y con otros operadores aritméticos, por ejemplo:

```php
<?php
    $variable = 10;
    echo $variable+=5; // 15
    echo $variable-=5; // 10
    echo $variable*=3; // 30
    echo $variable/=2; // 15
?>
```

Operador condicional ternario.

Permite asignar una condición a una determinada variable, personalizando la información a la que accedemos cada vez que verificamos dicha variable.

SINTAXIS:

$variable = condición ? valor si es verdadera : valor si es falsa

Por ejemplo:

```
$variable1 = -3;


$signo = $variable1 <0 ? 'Signo negativo' : 'Signo positivo';


echo $signo;
```

En este caso, dependiendo del valor que le demos a la variable, será el valor del mensaje mostrado por pantalla.

Tipo de datos.

En PHP tenemos distintos tipos de datos a nuestra disposición.

- BOOLEANO: puede tomar dos valores diferentes, verdadero o falso. En PHP se analizan las variables según el contexto, y los criterios para definir un tipo FALSE son los siguientes:

```
// Definicion explicita
$variable = false;


if ($variable){
    echo 'Condicion VERDADERA';
} else echo 'Condicion FALSA'
```

```php
// Valor numerico cero
  $variable = -12;


  if ($variable){
      echo 'Condicion VERDADERA';
  } else echo 'Condicion FALSA';
```

```php
  // Cadena vacia
  $variable = '';


  if ($variable){
      echo 'Condicion VERDADERA';
  } else echo 'Condicion FALSA';
```

Otros valores que hacen FALSE a la variable son: cadena conteniendo un valor cero, valor NULL y un ARRAY vacío.

- NUMÉRICOS: entre los tipos de datos numéricos, tenemos dos subtipos, los enteros y los valores de coma flotante (INT y FLOAT).
  Para los valores de tipo entero, podemos optar entre tres bases diferentes: los enteros de base 10 (decimal), los enteros de base 8 (octal) y los enteros de base 16 (hexadecimal).
  Para saber con qué tipo de datos está tratando, el intérprete PHP verifica el comienzo del valor.

Los enteros negativos se escriben anteponiendo el signo –

Los numéricos con parte entera y decimal, se escriben separando ambas con un punto.

Funciones.

Las funciones se declaran mediante la palabra FUNCTION, al igual que las variables, los nombres de función pueden comenzar con un guion bajo o una letra y a continuación cualquier cadena alfanumérica.

```php
<?php
    // Defino una función. function
    function saludo($arg1){
        $saludo = 'Hola '.$arg1;
        return $saludo;
    }

    // Invoco la función y le paso el argumento 'Pedro'
    echo saludo('Pedro');
?>
```

Para el ejemplo anterior, veremos por pantalla el mensaje “Hola Pedro”.

Veamos un ejemplo con dos argumentos:

```php
<?php
    // Defino una función. function
    function sumar($v1,$v2){
        $resultado = $v1+$v2;
        echo 'El resultado es: '.$resultado;
    }

    // Invoco la función
    sumar(12,7);
?>
```

Al invocar la función y pasarle los dos valores por argumento, se muestra por pantalla el valor de la suma de ambos valores.

Dentro de una función podemos poner todo tipo de sentencias o estructuras como las que vimos hasta ahora, incluso otras funciones que serán invocadas dentro de la función original.

Veamos un ejemplo con una estructura de decisión.

Definimos una función que analiza un valor pasado por argumento y me devuelve el resultado 'par' o 'impar' dependiendo del valor ingresado.

```php
<?php
    // Defino una función. function
    function par($valor){
        if ($valor%2 == 0)
            echo 'Es PAR';
        else
            echo 'No es PAR';
    }

    // Invoco la función
    par(7);
?>
```

Al invocar la función, le pasamos un número que se guarda en la variable $valor y dentro de la función, se analiza el resto de dividir dicho argumento sobre el número 2.

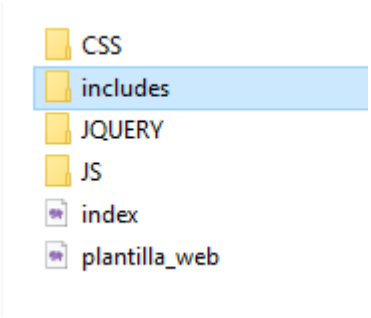
Mediante una estructura IF / ELSE verificamos si el resto de dicha división es cero, en cuyo caso el número será par, y si la condición resultase falsa, entonces nos devuelve el valor 'impar'.

Práctica.

Vamos a escribir el código de un menú de navegación sencillo, le vamos a aplicar algunas clases de Bootstrap para darle estilo y lo vamos a guardar en una función PHP que incluiremos en todos los archivos de nuestra aplicación.

PASO 1- Creamos la carpeta INCLUDES.

Dentro de nuestra aplicación PRACTICA2020, creamos una nueva carpeta donde guardaremos los archivos a incluir que no sean de estilo ni archivos JS.



PASO 2 – Creamos un archivo MENU.PHP

Este archivo será el que contenga la función de PHP con el código HTML necesario para generar un menú de navegación.

La función se llamará MENU.

```
function menu(){


}
```

PASO 3 – Creamos las etiquetas NAV.

La etiqueta para hacer un menú de navegación es la etiqueta NAV.

Dentro de la misma vamos a crear un DIV del tipo CONTAINER para que los elementos que vayamos incorporando, estén todos contenidos dentro del mismo espacio.

```php
<?php
    function menu(){ ?>
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top">
            <div class="container">


            </div>
        </nav>
    <?php
    }
?>
```

*** IMPORTANTE.

Es importante prestar atención a como mezclamos código PHP y HTML.

Cada vez que empecemos a escribir en PHP, debemos abrir la correspondiente etiqueta, y antes de empezar a escribir HTML, debemos cerrarla.

PASO 4 – Incluimos el archivo del menú, dentro de nuestra plantilla.

```php
    <!-- Archivos a incluir -->
    <?php include("includes/menu.php"); ?>
```

PASO 5 – Invocamos la función MENU dentro de la plantilla.

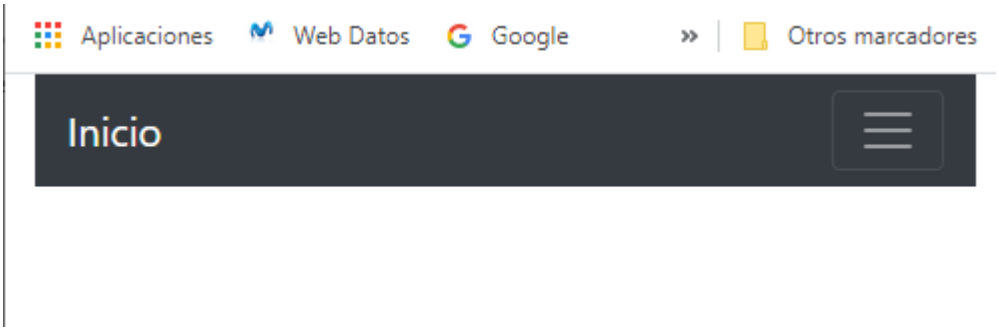
Invocamos la función como vimos en la clase de teoría, llamándola por su nombre.

Esto lo hacemos en la primera línea del BODY.

```
<body class="container">
    <?php menu(); ?>
```

PASO 6 – Agregamos enlace al inicio y el botón RESPONSIVE.

```
<a class="navbar-brand" href="#">Inicio</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>
```



PASO 7 – Creamos un DIV colapsable y una lista.

```
<div class="collapse navbar-collapse" id="navbarResponsive">
    <ul class="navbar-nav ml-auto">


    </ul>
</div>
```

PASO 8 – Agregamos elementos.

Agregamos un par de elementos simples a la lista, para ver el funcionamiento del menú.

```
<!-- Elementos de la lista del menu -->
<li class="nav-item active">
    <a class="nav-link" href="pagina1.php">Página 1</a>
</li>


<li class="nav-item active">
    <a class="nav-link" href="pagina2.php">Página 2</a>
</li>
```

PASO 9 – Creamos páginas y las enlazamos al menú.

Vamos a crear tres páginas a partir de nuestra plantilla.

- INDEX
- Página 1
- Página 2

Abrimos el archivo PLANTILLA_WEB.PHP y, mediante la opción guardar como, lo guardamos con el nombre INDEX.PHP.

Repetimos la operación, guardándolo como PAGINA1.PHP y PAGINA2.PHP.

Fin de la práctica, ya tenemos un menú de navegación RESPONSIVE funcionando en nuestra aplicación.

Este es el INDEX.