UD-02c: Introducción a PHP

Desarrollo Web en Entorno Servidor

Curso 2020/2021

Características básicas del lenguaje

Estructuras de control

Al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de estructuras de control:

- Sentencias condicionales
- Sentencias de iteración (bucle)

Características básicas del lenguaje

Condicionales

```
<?php
    if (condicion) {
        Instruccion1
        Instruccion2
        …
    }
?>
```

```
<?php
    if ($a < $b)
        echo "a es menor que b";
    elseif ($a > $b)
        echo "a es mayor que b";
    else
        echo "a es igual a b";
?>
```

(Aquí no hay llaves, porque afectan a una única sentencia)

Características básicas del lenguaje

Condicionales

Sentencia **switch** con *case*, *break* y *default*

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

Características básicas del lenguaje

Estructuras iterativas

Sentencia **while**. El significado de una sentencia *while* es simple. Le dice a PHP que ejecute las sentencias anidadas, cuando la expresión se evalúe como TRUE.

Ejemplo

```
<?php
    $a = 1;
    while ($a < 8)
        $a += 3;
    echo $a;
?>
```

Ejemplo

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

Características básicas del lenguaje

Bucles

Sentencia do…while. Los bucles *do-while* son muy similares a los bucles *while*, excepto que la expresión verdadera es verificada al final de cada iteración en lugar que al principio.

Ejemplo

```
<?php
    $a = 5;
    do
        $a -= 3;
    while ($a > 10);
    echo $a;
    // el bucle se ejecuta una sola vez, con lo que el valor obtenido es 2
?>
```

Características básicas del lenguaje

Bucles

Sentencia **for**

```
<?php
    for ($a = 5; $a < 20; $a += 3)    {
        echo $a; // Se muestran los valores 5,8,11,14 y 17
        echo "<br />";
    }
?>
```

Características básicas del lenguaje

Inclusión de ficheros externos

- Conforme van creciendo los programas, resulta trabajoso encontrar la información que buscas dentro del código.
- En ocasiones, resulta útil agrupar ciertas funciones o bloques de código, y ponerlos en un fichero aparte.
- Posteriormente, puedes hacer referencia a esos ficheros para que PHP incluya su contenido como parte del programa actual.

include → Evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, en el mismo punto en que se realiza la llamada

Si el fichero que queremos incluir no se encuentra, continua la ejecución del script sin problemas, pero emite una advertencia (E_WARNING).

Ejemplo

definiciones.php

```php
<?php
    $modulo = 'DWES';
    $ciclo = 'DAW';
?>
```

programa.php

```php
<?php
    echo "Módulo $modulo del ciclo $ciclo<br />";
    //Solo muestra "Modulo del ciclo"
    include 'definiciones.php';      O include ("definiciones.php");
    echo " Módulo $modulo del ciclo $ciclo<br />";
    // muestra "Modulo DWES del ciclo DAW"
?>
```

Características básicas del lenguaje

include_once → Si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función). include_once funciona igual que include, pero sólo incluye aquellos ficheros que aún no se hayan incluido.

require → La diferencia más importante al usar require es que cuando no se puede incluir el fichero, se detiene la ejecución del guión. *E*s idéntico a include excepto que en caso de fallo producirá un error fatal de nivel E_COMPILE_ERROR y detendrá la ejecución del script.

require_once → Combinación de las dos anteriores. Asegura la inclusión del fichero indicado sólo una vez, y genera un error si no se puede llevar a cabo.

Características básicas del lenguaje

Funciones

El flujo de ejecución de cualquier programa es de arriba hacia abajo. Pero este orden se ve alterado cuando entran en juego las funciones, pues una función NO ejecuta el código que hay en su interior hasta que esa función no es llamada.

¿Cómo llamamos a una función? → nombre_funcion();

Recuerda que las funciones nos permiten…

- Asociar una etiqueta (nombre) a un bloque de código
- Estructurar mejor el código
- Evitar escribir varias veces la misma tarea
- …

Características básicas del lenguaje

Para crear una función, se usa la palabra **function**:

Ejemplo

El precio con IVA es 0 euros. ¿Por qué?

```
<?php
    function precio_con_iva()
    {
        $precio_iva = $precio * 1.18;
        echo "El precio con IVA es “ . $precio_iva . " euros";
    }
    $precio = 10;
    precio_con_iva();
?>
```

Características básicas del lenguaje

Ejemplo

```
<?php
    function precio_con_iva()
    {
        global $precio;
        $precio_iva = $precio * 1.18;
        echo "El precio con IVA es ".$precio_iva."euros";;
    }
    $precio = 10;
    precio_con_iva();
?>
```

Definimos la variable `$precio` como global pa**ra que se pueda sobreescribir.**

Si se define una variable dentro de una función, su ámbito se restringe a la función.
Para que el ámbito de la variable sobrepase la función se definirá como global.

Características básicas del lenguaje

```php
<?php
    $nombre="ANA";
    function cambiar_nombre () {
        $nombre="EVA";
    }
    cambiar_nombre();
    echo $nombre;
?>
```

¿Cuál es el resultado?
¿Por qué?

```php
<?php
    $nombre="ANA";
    function cambiar_nombre () {
        global $nombre;
        $nombre="EVA";
    }
    cambiar_nombre();
    echo $nombre;
?>
```

¿Cuál es el resultado?
¿Por qué?

Características básicas del lenguaje

En PHP no es necesario definir una función antes de utilizarla...a no ser que esté condicionada, como este ejemplo:

```
<?php
	$iva = true;
	$precio = 10;
	precio_con_iva(); // Error, aún no está definida la función

	if ($iva)
	{
		function precio_con_iva()
		{
			global $precio;
			$precio_iva = $precio * 1.18;
			echo "El precio con IVA es ".$precio_iva;
		}
	}
	precio_con_iva(); // Aquí ya no da error
?>
```

Es lo mismo que… if ($iva == true)

Características básicas del lenguaje

Argumentos

- En el ejemplo anterior se usaba una variable global en la función, lo cual no es una buena práctica.
- Siempre es mejor utilizar **argumentos o parámetros** al hacer la llamada. *(Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, es mejor devolver un valor usando la sentencia return).*

```
<?php
    function precio_con_iva($precio)
    {
        return $precio * 1.18;
    }
    $precio = 10;
    echo "El precio con IVA es ". precio_con_iva($precio);
?>
```

Características básicas del lenguaje

- Los *argumentos* se indican en la definición de la función como una lista de variables *separada por comas*. *No se indica el tipo* de cada argumento, *ni se indica si la función va a devolver o no un valor* (si una función no tiene sentencia return, devuelve null al finalizar).
- Al definir la función, *pueden indicarse valores por defecto para los argumentos*, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado. *Deben estar a la derecha*.
- *Si no se especifica, los argumentos se pasan por valor* y cualquier cambio que se haga dentro de la función a los valores de los argumentos, no se reflejará fuera de la función. Para que ocurra lo contrario, se debe pasar *por referencia, añadiendo el símbolo & antes de su nombre*.

Características básicas del lenguaje

```php
<?php
    function precio_con_iva($precio, $iva = 0.18) {
        return $precio * (1 + $iva);
    }
    $precio = 10;
    $precio_iva = precio_con_iva($precio);
    echo "El precio con IVA (18%) es ". $precio_iva;
    echo "<br><br>";
    $precio_iva2 = precio_con_iva($precio, 0.15);
    echo "El precio con IVA (15%) es ". $precio_iva2;
?>
```

11,8

11,5

EJERCICIOS

- Crea 2 guiones (scripts). El primero se llamará ***funciones.php*** y el segundo ***ejer04.php*** *y con título Ejercicio04*

En el **primer archivo** tendrás dos funciones:

- Una que se llame ***fechaCastellano*** (no recibe parámetros), que muestra por pantalla la fecha actual así: **Martes, 25 de Septiembre de 2020**
  Obligatorio usar la estructura *switch* para obtener, al menos, el mes.
- Otra llamada ***factorial*** (recibe un número como argumento), que calcula el factorial de dicho número, mostrando algo así: **4! = 4 x 3 x 2 x 1 = 24**
  [suponiendo que ha recibido un 4]

En el **segundo archivo** se incluirá el primer archivo y se llamará a ambas funciones (a fechaCastellano sin parámentros y a factorial, pasándole un 5).

- Escribe un programa que genere una tabla de multiplicar según una variable **$n**, cuyo valor será 7. Llama al ejercicio **ejer05.php** y dale el título Ejercicio05.

Tabla de multiplicar

La tabla de multiplicar del 7 es:

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

- Realiza un programa de nombre **ejer06.php** y dale el título Ejercicio06 que cumpla las siguientes especificaciones:
  - Un comercio realiza descuentos con la siguiente regla:
    - ✓ Si la compra no alcanza los 100 euros, no se realiza descuento.

Precio inicial: 58
No se aplica ningún descuento
Precio final: 58

- ✓ Si la compra está entre 100 euros y 500 euros, se descuenta un 10%.

```
Precio inicial: 500
Se aplica un descuento del 10%
Precio final: 450
```

- ✓ Si la compra supera los 500 euros, se descuenta un 15%.

- La cantidad **precio inicial** será un variable con valor dada en el programa principal. Esta cantidad será pasada como parámetro a la función **calcularPrecio(cantidad)** la cual muestra por pantalla un mensaje según el descuento aplicado y además devuelve el precio final de la compra. Esta función estará ubicada en el fichero ***funciones2.php***.

- Finalmente el programa principal (***ejer06.php***) debe indicar cuánto debe pagar el cliente.