Desarrollo web en entorno servidor

UT 1.1 – Arquitectura y herramientas web

2 – PHP – Tipos – Variables – Operadores – Estructuras – Arrays

IES Clara del Rey – Madrid

PHP – Sintaxis

El código PHP siempre tiene que estar incluido dentro de los tags PHP:

- Apertura: *<?php*
- Cierre: *?>*

PHP es sensible a mayúsculas y minúsculas.

Todas las sentencias de PHP tienen que terminar con ;

Los comentarios se pueden escribir de forma muy similar a Java:

- Con almohadilla: *# Esto es un comentario de una línea*
- Con doble barra: // *Esto es un comentario de una línea*
- Multilínea, con /* … */

PHP – Mostrando o "pintando" la salida

Se pueden usar varias instrucciones y funciones para generar la salida:

- Instrucción "echo": *echo 'Hola';*
- Función "print": *print('Hola');*
- La forma abreviada de echo:

<? = $variable ?>

- Lo que esté fuera de los tags php se mostrará en la página:

 # Acaba un bloque de php ?>
Esto se muestra porque no es php
<?php # Inicia otro bloque de php

PHP – Mostrando o "pintando" la salida

Se pueden usar varias instrucciones y funciones para generar la salida:

- Función "var_dump": *var_dump($variable);*
  Vuelca la variable "$variable" en la salida
- Función "print_r": *print_r($variable);*
  Vuelca la variable "$variable" de una forma más legible.
- Función "printf": *printf("formato", ...$variables);*
  Similar al printf de Java. Se puede ver referencia de cadenas de formato en https://www.php.net/manual/en/function.printf.php
- Función "sprintf": *$var = sprintf("formato", ...$variables);*
  Similar a String.format en Java. Devuelve un string en lugar de escribirlo en la salida estándar.

PHP – Tipos y variables

Las variables siempre comienzan con el símbolo dólar:

- Deben empezar, tras el $, con una letra o un guion bajo. No se admite que empiece por números u otros caracteres especiales. Ejemplos: $nombre, $fecha, $numTelefono.
- No se tienen que declarar las variables. Cuando se asigna valor a una variable se crea con el tipo adecuado. Ejemplo, al ejecutar esta línea:

  $nombre = "Juan";

  La variable "$nombre" cambia su valor si ya existe.

  Si no existe se crea en este momento y se asigna el valor.

  En este caso, el tipo de la variable sería "String".

PHP – Tipos y variables

En PHP se usa un tipado dinámico y débil:

- Dinámico:
  - No se puede declarar los tipos de las variables. Se infiere del contexto, de lo que se está asignando a las variables.
  - En algunos sitios sí se puede poner tipo, pero son limitados: atributos de clases, parámetros en métodos, y poco más.
  - El tipo de una variable puede cambiar en tiempo de ejecución.
- Débil: PHP intentará convertir los tipos de datos siempre que pueda. De nuevo, infiere los tipos del contexto. Sobre todo, en función de los operadores que se estén usando.

PHP – Tipos y variables

Entero (distintas bases):

```
$edad = 30;         // Decimal (base 10)
$edad = 012;        // 10 en octal (base 8). Empiezan con cero.
$edad = 0x0F;       // 15 en hexadecimal (base 16). Empiezan con 0x.
$edad = 0b0011; // 3 en binario. Empiezan con 0b.
```

Reales (con decimales)

```
$precio = 1.567          // Se usa el punto para separar decimales.
$precio = 1.567e2        // Científica: 1.567 * 10^2 = 156.7
```

Boolean

```
$todoOk = false   // Puede usarse False o FALSE
$todoOk = true    // Puede usarse True o TRUE
```

PHP – Tipos y variables

String:

```
$apellidos = 'Esto también es cadena, pero con comilla simple";
$nombre = "Esto es una cadena, con comillas dobles";
```

En general se recomienda que para string se use comilla simple, porque así se pueden poner dentro comillas dobles. Se usan dobles cuando:

- Dentro del string necesitamos poner comillas simples.
- Usamos caracteres especiales (\t, \n, etc.)
- se quieren embeber (interpolar) otras variables:

```
echo "El valor de la variable $edad es {$edad} años.";
```

PHP – Tipos y variables

Para escapar caracteres especiales se usa la contrabarra "\":

```
$mensaje = "A continuación tenemos un tab \t";
```

Se puede comprobar si una variable tiene valor con la función isset:

```
if (isset($valor)) {
    // Hacer algo
}
```

Y se puede "vaciar" o "borrar" una variable con la función unset:

```
unset($valor);
```

PHP – Operadores

Aritméticos y sus atajos:

```
$numero = 3 + 5;
$numero = 3 - '5';      // PHP convierte '5' al número 5.
$numero += 4;           // $numero debe tener un valor previo.
echo $numero++;         // Incrementa tras evaluar
echo ++$numero;         // Incrementa antes de evaluar
```

Operador ternario:

```
$msg = $edad >= 18 ? "Mayor de edad" : "Menor de edad";
```

Concatenación de cadenas: se usa "." en lugar de "+".

```
$string = "Hola " . "otra cadena " . 33; // El 33 se convierte a String.
```

PHP – Operadores

Comparación. Convierten operandos si es necesario.

```
$a == $b;          // Igualdad
$a != $b;          // Desigualdad
$a <> $b;          // Forma alternativa para desigualdad
$a > $b;           // Mayor que. < para menor que
$a >= $b;          // Mayor o igual que. <= para menor o igual
```

Igualdad y desigualdad estrictas. Compara valor y tipo, sin convertir.

```
$a === $b;         // Igualdad estricta
$a !== $b;         // Desigualdad estricta
```

Coalescencia de nulos

```
$a = $b ?? "Valor si $b es nulo o no definido";
$a = $b ?? $c ?? "Valor si $b Y $c son nulos o no definidos";
```

PHP – Operadores

Lógicos.

```
$a && $b;          // Y
$a and $b;         // Y de baja precedencia
$a || $b;          // O
$a or $b;          // O de baja precedencia
!$a;               // Not
$a xor $b;         // XOR (O exclusivo)
```

Los operadores de baja precedencia se evalúan después que los de alta precedencia. Así estas dos expresiones no son lo mismo:

```
$a = false; $b = true; $c = true;
$a && $b || $c;          // ($a && $b) || $c - true
$a and $b || $c;         // $a and ($b || $c) - false
```

PHP – if / elseif / else

If sin else o elseif:

```
if ($variable == 3) {
        print 'Es un tres';
}
```

If con elseif y else:

```
if ($edad < 18) {
        print 'No puedes beber';
} elseif  ($edad >= 18 && $edad < 50 {
        print 'Puedes beber';
} else {
        print 'No bebas, que no estás ya para estas cosas.';
}
```

PHP – if / elseif / else – En HTML

Sintaxis alternativa para usar en HTML, para que sea más breve y más fácil de leer:

```
<?php if ($x): ?>
    Se muestra si $x se evalua a true.
<?php else: ?>
    Se muestra en otro caso.
<?php endif; ?>
```

Ojo, como PHP hace conversión de tipos, aunque lo que evaluemos no sea un boolean, intentará convertirlo a boolean.

PHP – while y do...while

While:

```
$i = 0;
while ($i < 5) {
    print($i++);
} // Muestra "01234"
```

Do ... while:

```
$i = 0;
do {
    print($i++);
} while ($i < 5); // Muestra "01234"
```

Ojo, el do ... while siempre entra al menos una vez en el bucle. En el ejemplo, si $i comienza con valor 100, mostraría el 100. Con while no.

PHP – for y foreach

For:

```
for ($x = 0; $x < 10; $x++) {
        print ($x);
} // Muestra "0123456789"
```

Foreach – Para recorrer arrays:

```
$personas  = ['Juan', 'María'];
foreach ($personas as $nombre) {
        echo $nombre . " - ";
} // Muestra "Juan – María - "
```

PHP también tiene break y continue.

PHP – switch

Switch:

```
switch ($edad) {
    case 0:
        print('Recien nacido';
        break; // break impide que se siga al siguente caso
    case 1:
    case 2:
        print('Menor de tres años');
        break;
    default:
        print('Tres o más años');
```

PHP – match (PHP 8.0+)

Match:

```
$opcion = 2;
$resultado = match ($opcion) {
        1           => "Opción 1 seleccionada",
        2           => "Opción 2 seleccionada",
        3, 4        => "Opción 3 o 4 seleccionada",
        default     => "Opción no válida",
};
echo $resultado;
```

Match devuelve un resultado. Las condiciones se evalúan en orden, y cuando una se cumple dejan de evaluarse el resto.

PHP – match (PHP 8.0+)

Match:

```
$edad = 18;
$salida = match (true) {
        $edad < 2          => "Bebé",
        $edad < 13         => "Niño",
        $edad <= 19        => "Adolescente",
        $edad < 40         => "Joven adulto",
        $edad >= 40        => "Adulto mayor"
};
```

Se pueden evaluar condiciones en lugar de valores. Se usa "true" en la expresión a evaluar, y la primera de las condiciones que se cumpla, que devuelva true, es la que se selecciona.

PHP – Arrays

En PHP los arrays son arrays asociativos, como en JavaScript, similares a los Map en Java, y hay varias formas de definirlos.

```
$diasA = array('Lunes' => 1, 'Martes' => 2, ... , 'Domingo' => 7);
$diasB = ['Lunes' => 1, 'Martes' => 2, ... , 'Domingo' => 7];
```

Las dos sintaxis son equivalentes. Los puntos suspensivos no son parte del lenguaje. En el ejemplo indican que faltan días, por abreviar.

Al acceder a arrays asociativos, se accede por la "clave", sensible a mayúsculas y minúsculas:

```
print(diasA['Martes']);          // Muestra 2
print(diasB['Domingo']);         // Muestra 7
```

En un array asociativo, si accedemos a él por posición, puede no devolver nada. Si no existe una clave con ese valor no devuelve nada.

PHP – Arrays

Si declaramos los arrays como lista de literales se asigna una clave implícita, que es un entero, y que comienza en cero.

```
$diasA = array('Lunes', 'Martes', 'Miércoles', ... , 'Domingo');
$diasB = ['Lunes', 'Martes', 'Miércoles', ... , 'Domingo'];
```

Las dos sintaxis son equivalentes. Los puntos suspensivos no son parte del lenguaje. En el ejemplo indican que faltan días, por abreviar.

Al acceder a arrays asociativos declarados como literales, se usa el índice:

```
print(diasA[0]);    // Muestra Lunes
print(diasB[2]);    // Muestra Miércoles
```

PHP – Arrays

Se puede recorrer un array asociativo con foreach.

```
$dias = ["Lunes" => 1, "Martes" => 2, "Miércoles" => 3,
        "Jueves" => 4, "Viernes" => 5,  "Sábado" => 6, "Domingo" => 7];
```

Se pueden recorrer sólo los valores:

```
foreach ($dias as $valor) {
        echo "Valor: {$valor}\n";
}
```

O las parejas de clave / valor

```
foreach ($dias as $clave => $valor) {
        echo "Clave: {$clave} - Valor: {$valor}\n";
}
```