# Desarrollo Web en Entorno Servidor

2.- PHP Básico

IES Severo Ochoa

# Índice

# PHP

- *Personal Home Page*
- Lenguaje de propósito general
- Sintaxis similar a *C* / *Java*
- El código se ejecuta en el servidor (*Apache* con *mod_php*)
  - El cliente recibe el resultado

- Última versión: 7.4 (Nov 2019)
  - 7.0 - Dic 2015
  - 2x más rápido que PHP5

# Bloques de código

- **<?php** código **?>**

- Las sentencias se separan con **;**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>PHP fácil</title>
</head>
<body>
  <!-- Muestra una frase con HTML -->
  Hola mundo<br>
  <!-- Muestra una frase con PHP -->
  <?php echo "Es muy fácil programar en PHP."; ?>
</body>
</html>
```

# echo y print()

- **echo** *expresión;*
- **print**(*expresión*);
- **<?=** *expresión* **?>**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Echo y print</title>
</head>
<body>
  <p><?php echo "Este texto se mostrará en la página web." ?></p>
  <p><?= "Este texto se mostrará en la página web." ?></p>
  <p><?php print("Este texto se mostrará en la página web.") ?></p>
</body>
</html>
```

# Comentarios

```php
<?php
  // Este es un comentario de una sola línea
  /*
    Este es
    un comentario
    que ocupa
    varias líneas
  */
?>
```

# Errores

- Si hay un error de ejecución → `Fatal Error`

```
Fatal error: Uncaught Error: Call to undefined function
plint() in C:\xampp\htdocs\202echo.php:11
 Stack trace:
 #0 {main}
       thrown in C:\xampp\htdocs\202echo.php on line 11
```

- Desde PHP 5 se lanzan como una excepción.
  - Más adelante veremos el uso de `try / catch`

# Variables

- $nombre
- Empiezan por letras o guion bajo (_)
- $var != $VaR
- Conveniente inicializarlas, sino dan error.

```php
<?php
  $nombre = "Aitor";
  $nombre_completo = "Aitor Medrano";
  $numero = 123;
  $pi = 3.14;
  $suerte = true;
  $sin_valor;

  echo $sin_valor;
?>
```

# Constantes

- El valor no varía
- **define**(nombre, valor);
- **const** nombre; // PHP > 5.3

```php
define("PI", 3.1416);
const IVA = 0.21;

echo PI, " ", IVA; // No se pone el símbolo dolar
```

- Hay un conjunto de constantes ya predefinidas:
  https://www.php.net/manual/es/language.constants.predefined.php

# Operadores: aritméticos

ies severo ochoa

| Operador | Nombre | Descripción | Ejemplo | Result. |
|---|---|---|---|---|
| + | Suma | Devuelve la suma de dos operandos. | 2 + 2 | 4 |
| - | Resta | Devuelve la diferencia entre dos operandos. | 5 - 3 | 2 |
| * | Producto | Devuelve la multiplicación entre dos operandos. | 2 * 3 | 6 |
| / | Cociente | Devuelve el cociente de la división entre dos operandos. | 8 / 2 | 4 |
| % | Módulo | Devuelve el resto de la división entre dos operandos. | 7 % 4 | 3 |
| - | Cambio de signo | Devuelve el valor del operando con el signo cambiado. | $x=6;<br>$y=-x;<br>$y; | -6 |

# Operadores: comparación

ies severo ochoa

| Operador | Nombre | Descripción | Ejemplo | Result. |
|---|---|---|---|---|
| == | Es igual | Devuelve verdadero si los dos operandos son iguales. | 2 == 2<br>2 == 5 | TRUE<br>FALSE |
| != | No es igual | Devuelve verdadero si los dos operandos son diferentes. | 2 != 2<br>2 != 5 | FALSE<br>TRUE |
| < | Es menor | Devuelve verdadero si el primer operando es menor que el segundo. | 2 < 5<br>5 < 2 | TRUE<br>FALSE |
| > | Es mayor | Devuelve verdadero si el primer operando es mayor que el segundo. | 2 > 5<br>5 > 2 | FALSE<br>TRUE |
| <= | Es menor o igual | Devuelve verdadero si el primer operando es menor o igual que el segundo. | 2 <= 5<br>2 <= 2 | TRUE<br>TRUE |
| >= | Es mayor o igual | Devuelve verdadero si el primer operando es mayor o igual que el segundo. | 5 >= 2<br>2 >= 2 | TRUE<br>TRUE |

# Operadores: lógicos

| Operador | Nombre | Descripción | Ejemplo | Result. |
|---|---|---|---|---|
| &&<br>And | AND (y) | Devuelve verdadero si ambos operandos son verdaderos. | TRUE && FALSE | FALSE |
| \|\|<br>Or | OR (o) | Devuelve verdadero si al menos uno de los operandos es verdadero. | TRUE && FALSE | TRUE |
| Xor | XOR<br>(o exclusivo) | Devuelve verdadero si uno de los operandos es verdadero pero no ambos. | TRUE &&TRUE<br>TRUE && FALSE | FALSE<br>TRUE |
| ! | NOT<br>(negación) | Devuelve verdadero si el operando es falso y falso si es verdadero | TRUE<br>FALSE | FALSE<br>TRUE |

# Concatenar cadenas

ies severo ochoa

| Op. | Nombre | Descripción | Ejemplo | Result. |
|---|---|---|---|---|
| . | Concatenar | Une dos cadenas de texto. | "Hola" . "Mundo" | "Hola Mundo" |

```php
<?php
  $x = 2;
  $y = 3;
  $z = $x + $y;
  echo "El resultado de sumar 2 y 3 es " . 5 . "<br>";
  echo "El resultado de sumar 2 y 3 es " . (2 + 3) . "<br>";
  echo "El resultado de sumar 2 y 3 es " . $z . "<br>";
?>
```

# Operadores: asignación

ies severo ochoa

| Operador | Ejemplo | Equivalencia |
|---|---|---|
| = | $x = 4 | $x = 4 |
| += | $x += 4 | $x = $x + 4 |
| -= | $x -= 4 | $x = $x – 4 |
| *= | $x *= 4 | $x = $x * 4 |
| /= | $x /= 4 | $x = $x / 4 |
| %= | $x %= 4 | $x = $x % 4 |
| .= | $x .= "mundo" | $x = $x . "mundo" |
| ++ | $x++ | $x = $x + 1 |
| -- | $x-- | $x = $x - 1 |

# Precedencia de operadores

| | |
|---|---|
| Operador lógico de negación | ! |
| Operadores multiplicativos | * / % |
| Operadores aditivos | + - . |
| Operadores de comparación | < <= > >= |
| Operadores de comparación (igualdad) | == != |
| Operadores lógicos (salvo la negación) | && \| \| xor |
| Operadores de asignación | = (y operadores comprimidos) |

```php
<?php
  echo "El resultado de (2 + 5) * 4 es " . ((2 + 5) * 4);
  echo "<br>";
  echo "El resultado de 2 + 5 * 4 es " . (2 + 5 * 4);
  echo "<br>";
  // El último caso resulta más legible si ponemos
  echo "El resultado de 2 + 5 * 4 es " . (2 + (5 * 4));
?>
```

# PHP 7: Nave espacial

- **<=>**

- Compara dos expresiones
- Devuelve -1, 0 o 1 cuando `$a` es respectivamente menor, igual, o mayor que `$b`

```php
<?php
  // Números enteros
  echo 1 <=> 1; // 0
  echo 1 <=> 2; // -1
  echo 2 <=> 1; // 1

  // Números decimales
  echo 1.5 <=> 1.5; // 0
  echo 1.5 <=> 2.5; // -1
  echo 2.5 <=> 1.5; // 1

  // Cadenas de caracteres
  echo "a" <=> "a"; // 0
  echo "a" <=> "b"; // -1
  echo "b" <=> "a"; // 1
?>
```

# Leyendo datos desde un Form

ies severo ochoa

- Los datos se envían via URL
  - `var1=valor1&var2=valor2…`
  - `ejemplo.php?nombre=Bruce+apellido1=Wayne`

- Paso 1: generar un formulario con
  - **`action`**`=archivo.php`
  - **`method=GET`**
- Paso 2: en el `archivo.php` leer los datos con `$_GET['nombreVar']`

# Ejemplo lectura datos

```html
<form action="saluda.php" method="get">
    <p><label for="nombre">Nombre: </label>
       <input type="text" name="nombre" id="nombre"></p>
    <p><label for="apellido1">Primer apellido:</label>
       <input type="text" name="apellido1" id="apellido1"></p>
    <p><input type="submit" value="enviar"></p>
</form>
```

saluda.html

```php
<?php
  $nombre = $_GET["nombre"];
  $apellido1 = $_GET["apellido1"];

  echo "Hola $nombre $apellido1";
?>
```

saluda.php

# Todo en uno

```
<form action="" method="get">
  <p><label for="nombre">Nombre: </label>
     <input type="text" name="nombre" id="nombre"></p>
  <p><label for="apellido1">Primer apellido:</label>
     <input type="text" name="apellido1" id="apellido1"></p>
  <input type="submit" value="enviar">
</form>
<p>
<?php
  if(isset($_GET['nombre'])) {
    $nombre = $_GET["nombre"];
    $apellido1 = $_GET["apellido1"];

    echo "Hola $nombre $apellido1";
  }
?>
```

Lo veremos en mayor profundidad en la unidad 4

# Condición simple

```
if (condición) {
   // código a ejecutar si se cumple la condición
}
```

```php
<?php
   $hora = 8; // La hora en formato de 24 horas
   if ($hora == 8) {
      echo "Suena el despertador.";
   }
   echo "<br>";
   if ($hora == 8)
      echo "Suena el despertador.";
?>
```

# Condición compuesta

```
if (condición) {
   // código a ejecutar si se cumple la condición
} else {
   // código a ejecutar si no se cumple la condición
}
```

```php
<?php
  $hora = 17; // La hora en formato de 24 horas
  if ($hora <= 12) {
    echo "Son las " . $hora . " de la mañana";
  } else {
    echo "Son las " . ($hora - 12) . " de la tarde";
  }
?>
```

# Condición anidadas

```
if (condición1) {
   // código a ejecutar si se cumple la condición1
} else if (condición2){
   // código a ejecutar si no se cumple condición pero sí lo
   hace condición2
} else {
   // código a ejecutar si no se cumple ninguna condición
}
```

```php
<?php
   $hora = 14; // La hora en formato de 24 horas
   if ($hora == 8) { echo "Es la hora de desayunar.";
   } else if ($hora == 14) { echo "Es la hora de la comida.";
   } else if ($hora == 21) { echo "Es la hora de la cena.";
   } else { echo "Ahora no toca comer.";
   }
?>
```

# switch

```
switch (expresión) {
   case etiqueta1:
      // Código a ejecutar si expresión == etiqueta1
      break;
   case etiqueta2:
      // Código a ejecutar si expresión == etiqueta2
      break;
   default:
      // Código a ejecutar si no se verifica ninguna
      etiqueta
      break;
}
```

# Ejemplos `switch`

ies severo ochoa

```php
<?php
  $hora = 14; // La hora en formato de 24 horas
  switch ($hora) {
    case 9:
      echo "Es la hora de desayunar.";
      break;
    case 14:
      echo "Es la hora de comer.";
      break;
    case 21:
      echo "Es la hora de merendar.";
      break;
    default:
      echo "Ahora no toca comer.";
      break;
  }
?>
```

```php
<?php
  $hora = 19; // La hora en formato de 24 horas
  switch($hora) {
     case
      24:
     case
      23:
     case
      22:
     echo "Ya he cenado.";
     case
      21:
     case
      20:
     case
      19:
     case
      18:
     case
      17:
     case
      16:
     case
      15:
     echo "Ya he comido.";
     case
      14:
     case
      13:
     case
      12:
     case
      11:
     case
      10:
     echo "Ya he desayunado.";
     break;
   default:
     echo "Tengo hambre";
     break
       ;
  }
?>
```

# Operador ternario

- **`condición ? valorVerdadero : valorFalso;`**

```php
<?php
  $hora = 14;
  $formato = ($hora > 12) ? 24 : 12;
  echo "El formato es de $formato horas"
?>
```

- Si queremos comprobar si una variable tiene valor y si no, darle otro:

  **`expresión ?: valorSiVacio;`**

```php
<?php
  $nombre = $_GET['nombre'] ?: "desconocido"
?>
```

# Bucles - while

```
while (condición) {
  // código a ejecutar mientras la
  condición sea verdadera
}
```

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

# Bucles - do...while

```
do {
  // código a ejecutar
} while (condición);
```

```php
<?php
  do {
    $dado = rand(1, 6);
    // rand() devuelve un valor aleatorio
    echo "Tirando el dado... ";
    echo "ha salido un " . $dado . ".";
    echo "<br>";
  } while ($dado != 5);
  echo "¡Bien! Saco una ficha de casa.";
?>
```

# Bucles - for

```
for (inicialización; condición; incremento){
   // código a ejecutar
}
```

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

  for ($i = 10; $i >= 0; $i--) {
    echo "Línea " . $i;
    echo "<br>";
  }
?>
```

# Rompiendo un bucle

ies severo ochoa

- PHP, del mismo modo que Java y C, permite romper los bucles mediante la instrucción `break`.
- A su vez, `continue` permite saltar a la siguiente iteración.

- Su uso no es recomendable
  - Aunque se pueden usar en algunos casos.

# ¿Alguna pregunta?