HLC

2º DAM

I.E.S. POLITÉCNICO H.LANZ
JOSÉ MARÍA MOLINA

TEMA 1-2 – PROGRAMACIÓN BÁSICA EN PHP

TEMA 1 – INTRODUCCIÓN A PHP

Vamos a ver los conceptos básicos de programación en PHP.

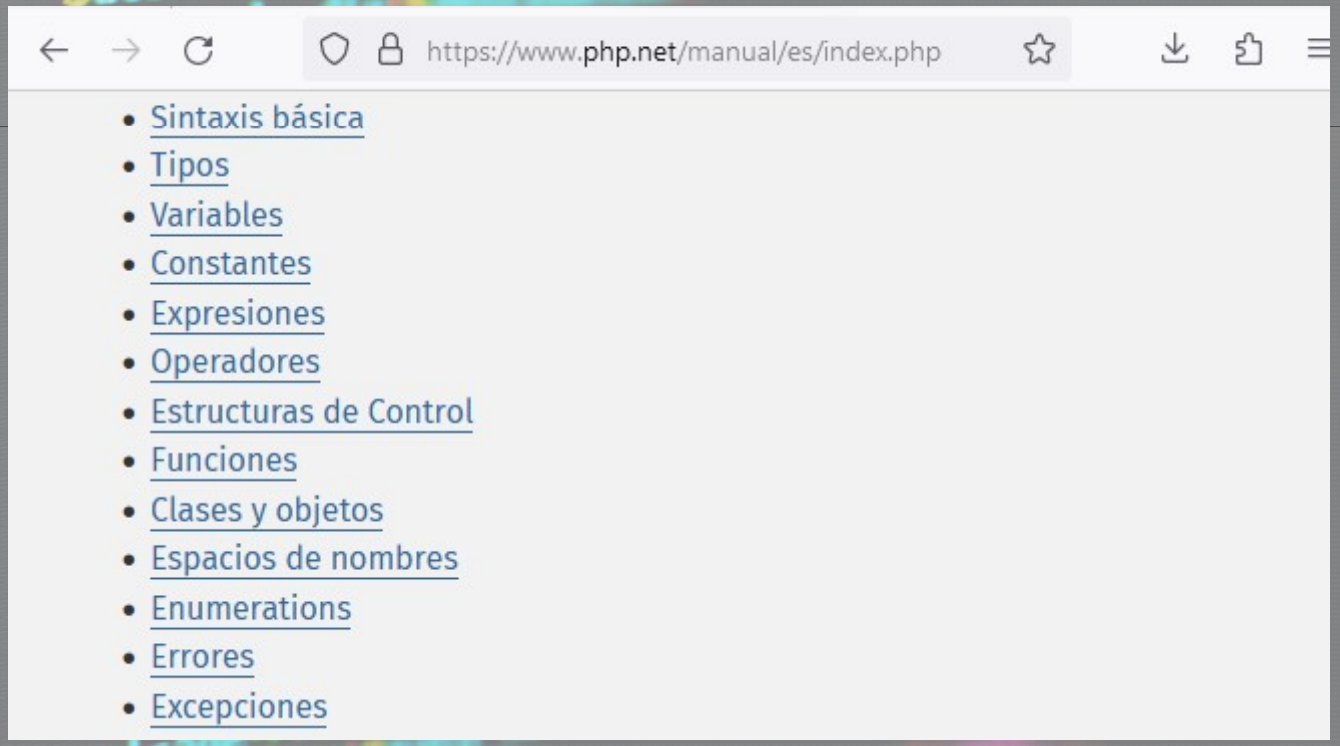
- ✓ 1 LENGUAJE PHP
- ✓ 2 SINTAXIS
- ✓ 3 OPERADORES
- ✓ 4 TIPOS DE VARIABLES
- ✓ 5 FUNCIONES, LIBRERÍAS Y MANEJO DE ERRORES
- ✓ 6 ESTRUCTURAS DE CONTROL

1. LENGUAJE PHP

- PHP → **P**repreprocesador de **H**i**P**ertexto
- El código se ejecuta en el servidor (donde hay un módulo de PHP)
- Procesa una página (php) → devuelve otra (html). Este proceso es transparente para el usuario. El usuario piensa que sigue viendo una página php, pero lo que ve es una página html creada de forma dinámica mediante php. La extensión de los ficheros será **.php**
- Con los script de servidor puedo utilizar casi cualquier tipo de herramienta que esté instalada en el servidor y se pueda conectar con PHP: bases de datos, hojas de cálculo, imágenes, ficheros, pdf, ftp,.....
- Vamos a aprender a manejar:
  - **Apartado 2, 3 y 4:** Sintaxis , operadores y tipos de variables (conversiones)
  - **Apartado 5:** Funciones , Librerías y Manejo de Errores
  - **Apartado 6:** Estructuras de Control
  - **Siguientes partes:** PHP Avanzado : Paso de variables, bases de datos mediante API, etc…

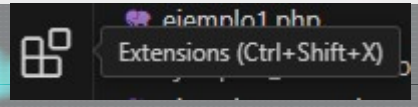
2 SINTAXIS

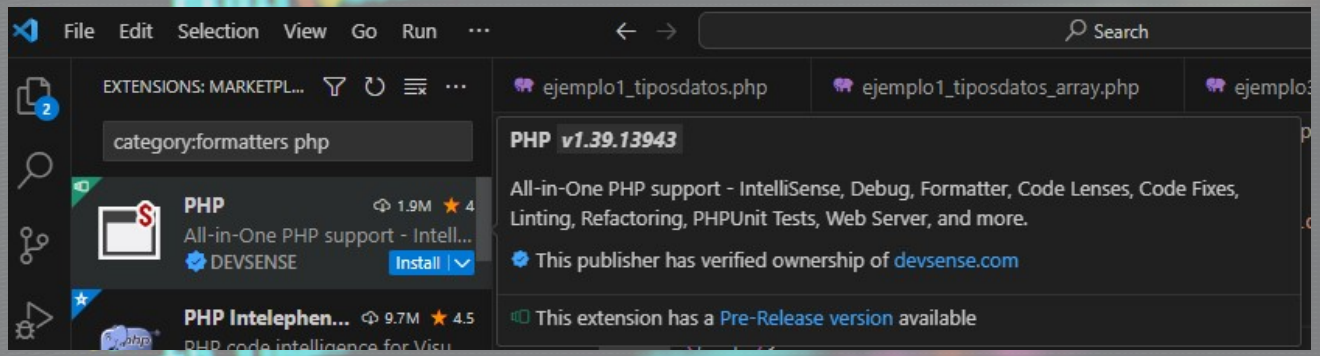
- **AYUDA**
- https://www.php.net/manual/es/index.php

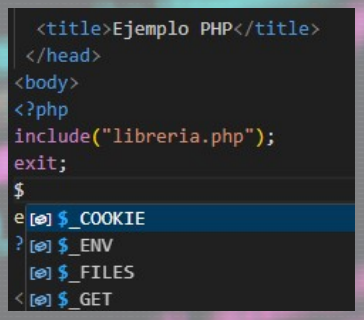


2 SINTAXIS

- **AYUDA**
- Extensión para Visual Studio: Barra lateral izquierda (Extensions)
- **All-in-One PHP Support**: sugerencias de código, autoformateo (Alt+May+F), detección y arreglo de errores, etc…







2 SINTAXIS

- **Inserción de PHP en HTML**
- <? .... ?> Sólo si se activa la función **short_tags()** o la bandera de configuración *short_open_tag.*
- **<?php ....**

  ?> USAREMOS ESTOS
- <script language="php"> .... </script>
- Los tags se pueden abrir y cerrar siempre que lo necesite, por lo tanto puedo introducir código html en el sitio que quiera de la página
- **Separación de instrucciones**
- Las instrucciones se separan con *';'*, en el caso de ser la última instrucción no es necesario el punto y coma.
- **Comentarios**
- Los comentarios en PHP pueden ser:
- Como en C o C++, **/*...*/ o //**
- Otro tipo de comentario de una línea es #.

2 SINTAXIS

- **Ejemplos básicos**

- El siguiente ejemplo llama a la función phpinfo() que muestra información sobre el módulo php (**ejemplo0_phpinfo().php**)

```
<?php
phpinfo();
?>
```

- El siguiente ejemplo saca un mensaje por pantalla php (**ejemplo1.php**)

```
<html>
 <head>
  <title>Ejemplo PHP</title>
 </head>
<body>
 <?php echo "Hola, este es un ejemplo con PHP"; ?>
</body>
</html>
```

2 SINTAXIS

- **Ejemplos básicos**
- El siguiente ejemplo NO SACA NADA ya que al detectar un solo fallo "CORTA" la ejecución del php (**ejemplo1_errores.php**)

```
<html>
 <head>
  <title>Ejemplo PHP</title>
 </head>
<body>
<?php echo "Hola, este es un ejemplo con PHP que no debería sacar NADA por pantalla porque contiene errores"    //Error 1: Falta el ;
echo "PRUEBA;  //Error 2: No cierra comillas
?>
</body>
</html>
```

3 OPERADORES

- **OPERADORES**
- Los operadores más utilizados son de 4 tipos:
  - ARITMÉTICOS: +, - , *, / , %
  - ASIGNACIÓN: =
  - ARITMÉTICOS DE ASIGNACIÓN: usan los operadores antes de la asignación
  - RELACIONALES: > , >=, < , <=, ==, !=
  - LÓGICOS: and (&&) y or (||). Podemos usar las dos notaciones
  - No confundir el operador de asignación = con el de igualdad ==

Operadores de Comparación:

`$a < $b`	$a **menor que** $b
`$a > $b`	$a **mayor que** $b
`$a <= $b`	$a **menor o igual que** $b
`$a >= $b`	$a **mayor o igual que** $b
`$a == $b`	$a **igual que** $b
`$a != $b`	$a **distinto que** $b

Operadores Lógicos:

`$a AND $b`	Verdadero si ambos son verdadero
`$a && $b`	Verdadero si ambos son verdadero
`$a OR $b`	Verdadero si alguno de los dos es verdadero
`$a !! $b`	Verdadero si alguno de los dos es verdadero
`$a XOR $b`	Verdadero si sólo uno de los dos es verdadero
`!$a`	Verdadero si $a es falso

Operadores de Asignación:

`$a = $b`	Asigna a $a el contenido de $b
`$a += $b`	Le suma a $b a $a
`$a -= $b`	Le resta a $b a $a
`$a *= $b`	Multiplica $a por $b y lo asigna a $a
`$a /= $b`	Divide $a por $b y lo asigna a $a
`$a .= $b`	Añade la cadena $b a la cadena $a

3 OPERADORES

- **FUNCIONES MÁS ÚTILES**
- **FUNCION ECHO**

Es una de las funciones más utilizadas. Saca cadenas de texto por pantalla. Lo vemos ahora ya que se va a utilizar en posteriores ejemplos.

Formato: echo "cadena a imprimir";

Puedo incluir variables dentro de ella y **concatenar cadenas** mediante el operador "**.**"

Ejemplo: echo "hola "**.**" mundo";

- **FUNCION EXIT**

Por cualquier motivo, podemos salirnos del script mediante la orden exit;

- **FUNCION HEADER**

Hace una redirección (automática) a otra página → header("location:http://www.google.es")

4 TIPOS DE VARIABLES

- **Tipos de datos**
- Las variables en PHP empiezan por un signo $. Así, la variable valor se pondrá: **$valor**
- Igual que en C o JavaScript, no podrán empezar por número y son sensibles a las mayúsculas.
- Los tipos de cada variable en PHP no están tan claros como en C. El intérprete ***asigna el tipo de una variable según el uso que se esté haciendo de ella*** (pasa algo parecido a JavaScript). Para asignar un tipo fijo a una variable se utiliza la función settype(), aunque no es obligatorio ya que si le asigno un real el interprete entenderá que esa variable es real. Los tipos son:

- Enteros (int)
- Flotantes (float)
- String (string)
- Arrays
- Objetos (ejemplo: al utilizar una clase cualquiera desde php)

4 TIPOS DE VARIABLES

- **Tipos de datos ENTEROS** (ver **ejemplo1_tiposdatos.php**)
- Este tipo de dato permite operaciones aritméticas
- $a=4+1;

```
echo "La variable a vale $a y es de tipo ";
echo gettype($a);
```

```
echo "La variable a es ";
settype($a, int); //indico el tipo de dato que va a contener
echo gettype($a);
echo "<br>";
```

4 TIPOS DE VARIABLES

- **Tipos de datos REALES** (ver **ejemplo1_tiposdatos.php**)
- Este tipo de dato permite operaciones aritméticas
- $a=4+0.01;

```
echo "La variable a vale $a y es de tipo ";
echo gettype($a);
```

```
echo "La variable a es ";
settype($a, float); //indico el tipo de dato que va a contener
echo gettype($a);
echo "<br>";
```

4 TIPOS DE VARIABLES

- **Tipos de datos CADENA (STRING)** (ver **ejemplo1_tiposdatos.php**)
- Este tipo de dato permite operaciones aritméticas
- $a="2";

```
echo "La variable a vale $a y es de tipo ";
echo gettype($a);
```

```
echo "La variable a es ";
settype($a, string); //indico el tipo de dato que va a contener
echo gettype($a);
echo "<br>";
```

4 TIPOS DE VARIABLES

- **Tipos de datos CADENA (STRING)**
- **CARACTERÍSTICAS Y FUNCIONES**
- El operador para concatenar cadenas es el operador "." : es equivalente al operador **+** utilizado en JavaScript.
- Ejemplo: echo "hola "."adios"; → Une las dos cadenas.
- En este punto hay que hacer una distinción **muy importante** : **php distingue entre comillas simples y comillas dobles**. Las dobles interpretan el código que hay dentro, las simples no lo hacen. Ejemplo:

$a="Mundo"

echo "Hola $a"; → Sacará el mensaje : Hola Mundo

echo 'Hola $a'; → Sacará el mensaje : Hola $a

- Códigos especiales (utilizando comillas dobles):

\\	Barra invertida
\$	Signo del dólar
\"	Comillas dobles

4 TIPOS DE VARIABLES

•**Tipo de datos ARRAY** (ver **ejemplo1_tiposdatos_array.php**)

•Las tablas (o array en inglés), son muy importantes en PHP, ya que generalmente las funciones que devuelven varios valores, como las funciones ligadas a las bases de datos, lo hacen en forma de tabla.

•En PHP disponemos de dos tipos de tablas. El primero sería el clásico, utilizando índices:

```
<?php
$ciudad[] = "París";
$ciudad[] = "Jaén";
$ciudad[] = "Roma";
$ciudad[] = "Sevilla";
$ciudad[] = "Londres";
print ("yo vivo en " . $ciudad[1] . "<BR>\n"); //print es similar a echo
?>
```

Esta es una forma de asignar elementos a una tabla, pero una forma más formal es utilizando la función **array**

```
<?php
$ciudad = array("París", "Roma", "Sevilla", "Londres"); //Declaración, con array siempre en minúsculas
$numelentos = count($ciudad); //contamos el número de elementos de la tabla
for ($i=0; $i < $numelentos; $i++){//imprimimos todos los elementos de la tabla
print ("La ciudad $i es $ciudad[$i] <BR>\n"); }
?>
```

4 TIPOS DE VARIABLES

•**Tipo de datos ARRAY** (ver **ejemplo1_tiposdatos_array.php** y la carpeta arrays)

•FUNCIONES: Las siguientes funciones hacen tareas relacionadas con los array. Ejemplo

```
<?php
$semana = array("lunes", "martes", "miércoles", "jueves","viernes", "sábado", "domingo");
echo count($semana); //7
//Ponemos puntero en el 1º elemento
reset($semana);
echo current($semana); //lunes
next($semana);
echo pos($semana); //martes
end($semana);
echo pos($semana); //domingo
prev($semana);
echo current($semana); //sábado
?>
```

current	- devuelve el valor del elemento que indica el puntero
pos	- realiza la misma función que **current**
reset	- mueve el puntero al **primer** elemento de la tabla
end	- mueve el puntero al **último** elemento de la tabla
next	- mueve el puntero al elemento **siguiente**
prev	- mueve el puntero al elemento **anterior**
count	- devuelve el número de elementos de una tabla.

Otra función que saca el contenido de un array → print_r($semana)

4 TIPOS DE VARIABLES

•**Tipo de datos ARRAY** (ver **ejemplo1_tiposdatos_array.php**)

•En PHP se utiliza mucho un tipo de array especial llamado **Array Asociativo**, es una especie de array con 2 filas, donde la primera fila tiene un elemento y en la otra fila hay un **valor clave** asociado a ese dato. Se llama Asociativo ya que asocia un valor a otro.

•Ya veremos los arrays más importantes de PHP al recoger variables de una página a otra y cuando veamos las bases de datos.

•Ejemplo: vamos a suponer que trabajamos por horas y que guardamos en un array el número de horas trabajadas de cada día de la semana para llevar el control. Guardaremos los valores "lunes", "martes", "miercoles" ... Y en cada casilla meteremos el valor de las horas.

•Si quisiera guardar el nombre de los días y el número de horas tendría que usar dos vectores en vez de uno

•dias[0]="lunes"

•dias[1]="martes"

•...

•horas[0]=10;

•horas[1]=8;

•...

•Si lo hago de esta forma tengo varios problemas: Tengo que gestionar dos arrays y tengo que encargarme de asociar la posición **X** de uno con la posición **X** de otro para que cuadren (siendo X el número de días desde 0 hasta 6)

4 TIPOS DE VARIABLES

•**Tipo de datos ARRAY** (ver **ejemplo1_tiposdatos_array.php**)

•La solución : arrays asociativos, su declaración se hace así:

$visitas = array("lunes"=>**10**, "martes"=>**8**, "miércoles"=>**9**, "jueves"=>**11**);

•De esta forma accedo al dato (número de horas) por su campo asociado (campo clave) que son los nombres de los días.

•Los arrays asociativos se pueden recorrer de forma normal o de una forma especial mucho más efectiva → comando **foreach.** Ejemplo:

```
<?php
    $horas = array("lunes"=>10, "martes"=>8,"miércoles"=>9, "jueves"=>11);
    foreach ($visitas as $clave => $valor) {
                        echo "el día $clave he trabajado $valor horas<br>";
}
?>
```

Clave	valor
lunes	10
martes	8
Miércoles	9
jueves	11

•A foreach le paso el array como parámetro, **as** digamos que divide los valores asociados en dos partes, guardándolas en dos variables ($clave y $valor). De esta forma puedo separar el array asociativo para manejar los valores. Todo debe ir dentro de un while ya que no se el nº de elementos que tiene el array.

5 FUNCIONES. LIBRERÍAS Y ERRORES

- **FUNCIONES** (ver **ejemplo3_funciones_librerias.php**)
- Se declaran de la misma forma que en JavaScript (respetar minúsculas):
  - **function** cuadrado($numero){

    return ($numero*$numero);

    }
- Lo normal es que una función siempre devuelva un tipo de dato, aunque puede haber casos en los que no sea necesario (ej: una función con mensaje de bienvenida).

5 FUNCIONES. LIBRERÍAS Y ERRORES

- **LIBRERÍAS** (ver **ejemplo3_funciones_librerias.php**)
- Una agrupación de funciones en un fichero forma una librería de funciones. Se puede hacer una librería de la misma forma que creábamos los ficheros **js.** Meto las funciones dentro de un fichero con extensión **php** y lo llamo desde mí código.
- ¿Cómo se hacen las llamadas? Hay 2 posibilidades:
  - include("fichero.php");
  - require("fichero.php");
- Diferencia: Si en un bucle se ejecutan varias veces una instrucción require(), el intérprete lo incluirá una sola vez, sin embargo si es include(), se incluirá el fichero cada vez que se ejecute la instrucción.

5 FUNCIONES. LIBRERÍAS Y ERRORES

•**MANEJO DE ERRORES** (ver **ejemplo_control_errores.php**)

•A menudo se producen errores en PHP , estos errores pueden ser debidos a muchos factores: no está el fichero al que llamo, la base de datos no existe o la consulta no devuelve nada,.....

• Se pueden controlar los errores *(siempre que no sean errores sintácticos)* mediante la función **or die.** Ejemplo:

```
$f = file_get_contents("noestoy.php");
echo "<br>";
$f = file_get_contents("noestoy.php") or die ("No se pudo");
```

6 ESTRUCTURAS DE CONTROL

•**ESTRUCTURAS DE CONTROL** (ver **ejemplo3_sentencias_control.php**)

•Las estructuras de control se manejan de igual forma que en JavaScript, con la diferencia del $ delante de cada variable

•**IF..ELSE**

```
if (condición) {
      Este bloque se ejecuta si la condición es VERDADERA
} else {
      Este boque se ejecuta si la condición es FALSA
}
```

•**IF..ELSEIF..ELSE :** Permite ejecutar condiciones en cascada (parecido a switch)

```
<?php
$nombre="";
if ($nombre == ""){
   echo "Tú no tienes nombre";
                } elseif (($nombre=="paco") OR ($nombre=="PACO")) {
                   echo "Tu nombre es PACO";
                 }
 else {
   echo "Tu nombre es " . $nombre;
}
```

6 ESTRUCTURAS DE CONTROL

- **SWITCH**

```
$dia="Jueves";
switch ($dia) {
case "Lunes": echo "Hoy es Lunes";break;
case "Martes":echo "Hoy es Martes";break;
.
.
.
default: echo "Esa cadena no corresponde a ningún día de la semana";
}
```

6 ESTRUCTURAS DE CONTROL

- **WHILE**
- Al igual que cualquier bucle WHILE, se puede romper mediante la orden **break** (aunque no sea correcto desde el punto de vista de la programación)

```
<?php
echo "<br><br>EJEMPLO WHILE CON BREAK<br>";
$num = 1;
while ($num < 5) {
        echo $num;
        if ($num == 3){
          echo "<br>Aquí nos salimos \n";
          break;
        }
$num++;
}
?>
```

6 ESTRUCTURAS DE CONTROL

- **DO-WHILE**
- Al igual que cualquier bucle WHILE, se puede romper mediante la orden **break**

```
<?php
echo "<br><br>EJEMPLO DO-WHILE CON BREAK<br>";
$num = 1;
do {
        echo $num;
        if ($num == 3){
        echo "Aquí nos salimos \n";
        break;
}
$num++;
} while ($num < 5);
?>
```

6 ESTRUCTURAS DE CONTROL

- **FOR**
- Se usa cuando se sabe exactamente el número de repeticiones que tiene que hacer el bucle.
- Podemos saltar a la siguiente iteración mediante **continue**, o salirnos mediante **break**

```
for ($num = 1; $num <=5; $num++){
  if ($num == 2){
            echo "Saltamos esta iteración \n";
            continue;
  }
  if ($num == 3){
            echo "Aquí nos salimos \n";
            break;
 }
echo $num;
}
```

6 ESTRUCTURAS DE CONTROL

- **OTROS EJEMPLOS**
- ejemplo4_prog_area
- ejemplo4_prog_calculos
- ejemplo4_prog1