Basic PHP

Desarrollo Web Entorno Servidor
PHP Básico

- Embeding PHP
- Displaying information
- Including files
- Datatypes
- Variables
- Constants
- Expressions and operators
- Decision structures
- Iteration structures (Loops)
- Functions
- Arrays

Antonio Jesús Calvo Morales

Syntax

Case sensitive

- PHP es sensible a las mayúsculas
- ¿Cómo se incrusta en la página web?

PHP is embed in HTML

```
<?PHP ... ?>
```
recomendado, siempre disponible
```
<?= expresión ?>
```
equivale a <? echo expresión ?>

- Las instrucciones se separan con un ; como en java y C. La marca final ?> implica un ;
- Comentarios: como en C, /* ... */ (varias líneas) y // (una línea)

```
/* Comentario de
varias líneas */
print “hola”; // Comentario de una línea
```

Printing

- statements: **echo** y **print**

echo: muestra una o más cadenas

```
echo cadena1 [, cadena2...];

echo “Hola mundo”;
echo “Hola “, “mundo”;
```

print: muestra una cadena

```
print cadena;

print “Hola mundo”;
print  “Hola “ . “mundo”;
```

- Example:

```
<html>
<head>
<title>Mi primer programa en PHP</title>
</head>

<body>
<?php
   print (“<p>Hola mundo</p>”);
?>
</body>
</html>
```

Including files

- For including external files we'll use:
  - **include**()
  - **require**()
- They both include and evaluate the file.
- Difference: if error include() provokes a warning and require() a fatal error.
- We'll use require() if an error is generated we must break the page loading.
- Example:

```
<HTML>
<HEAD>
   <TITLE>Título</TITLE>
<?PHP
// Incluir bibliotecas de funciones
   require ("conecta.php");
?>
</HEAD>
<BODY>
<?PHP
   include ("cabecera.html");
?>
// Código HTML + PHP
. . .
<?PHP
   include ("pie.html");
?>
</BODY></HTML>
```

Coments

- Las líneas de comentario sirven para poder recordar en un futuro qué es lo que hemos hecho al escribir un script y por qué razón lo hemos hecho así.
- Tenemos dos tipos de comentarios.
  - De una sola línea // o #
  - De varias líneas /* ... */

Data types

- PHP soporta 8 **tipos de datos primitivos**:
  - Tipos escalares: boolean, integer, double, string.
  - Tipos compuestos: array, object.
  - Tipos especiales: resource, NULL.
- El tipo de una variable no se suele especificar. Se decide en tiempo de ejecución en función del contexto y puede variar.

runtime

Data types 2

Funciones de interés:

- La función gettype() devuelve el tipo de una variable
- Las funciones is_*type* comprueban si una variable es de un tipo dado:
  is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar(), is_string()
- La función var_dump() muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays

String

- Las cadenas se encierran entre comillas simples o dobles:
  - Simples ': admite los caracteres de escape \' (comilla simple) y \\ (barra). Las variables **NO** se expanden
  - "dobles": admite más caracteres de escape, como \n, \r, \t, \\, \$, \". Los nombres de variables **SI** se expanden. **Para que los \n se coviertan en saltos de línea finales tendremos que pasar a la cadena la función nl2br($cadena)**.
  - Ejemplos:

```
$a = 9;
print 'a vale $a\n'; // muestra a vale $a\n
print "a vale $a\n"; // muestra a vale 9 y
  avanza una línea
print "<IMG SRC='logo.gif'>";
          // muestra <IMG SRC='logo.gif'>
print "<IMG SRC=\"logo.gif\">";
          // muestra <IMG SRC="logo.gif">
```

  - Acceso a un carácter de la cadena:
    - La forma es $inicial = $nombre{0};

Variables

- Las variables siempre van precedidas de un $
- El nombre es sensible a las mayúsculas.
- Comienzan por letra o subrayado, seguido de letras, números o subrayado.
- Variables predefinidas:
  - $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIES, $_FILES, $_ENV, $_REQUEST, $_SESSION
- Ámbito: globales al fichero o locales a una función.
- Ejemplo:

```
$valor = 5;
print “El valor es: “ . $valor . “\n”;
print “El valor es: $valor\n”; // ojo: comillas dobles

Resultado:
  El valor es: 5
```

Variables 2

- Variables variables
  - Se pueden crear nombres de variables dinámicamente
  - La variable variable toma su nombre del valor de otra variable previamente declarada
  - Ejemplo:

```
$a = "hola";
$$a = "mundo";

print "$a $hola\n";
print "$a ${$a}";

Resultado:
    hola mundo
    hola mundo
```

Variables 3


```
<?PHP
   $mensaje_es="Hola";
   $mensaje_en="Hello";
   $idioma = "es";
   $mensaje = "mensaje_" .
$idioma;
   print $$mensaje;
?>
```

Hola

```
<?PHP
   $mensaje_es="Hola";
   $mensaje_en="Hello";
   $idioma = "en";
   $mensaje = "mensaje_" .
$idioma;
   print $$mensaje;
?>
```

Hello

Static Variables

- Para poder conservar el último valor de una variable definida dentro de una función basta con definirla como estática.
- Ejemplo:

```
function crearJugador() {
    static numero_jugadores=0;
    ++numero_jugadores;
    …
}
```

Server Variables

- Son variables que devuelven información sobre el servidor donde se está ejecutando tu script.

$_SERVER['DOCUMENT_ROOT']	$HTTP_SERVER_VARS['DOCUMENT_ROOT']
C:\apache\htdocs (windows) o /var/www/html (linux)	
$_SERVER['HTTP_HOST']	$HTTP_SERVER_VARS['HTTP_HOST']
localhost	
$_SERVER['SERVER_PORT']	$HTTP_SERVER_VARS['SERVER_PORT']
80	
$_SERVER['SERVER_ADDR']	$HTTP_SERVER_VARS['SERVER_ADDR']
127.0.0.1	

Constants

- Definición de constantes:

```
define ("CONSTANTE", "hola");
print CONSTANTE;
```

No llevan $ delante

Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)

- Constantes predefinidas.

__LINE__ : devuelve el nº de línea que se está interpretando.

__FILE__ : nombre de fichero que se está ejecutando.

PHP_VERSION : versión de PHP.

PHP_OS: información sobre el sistema operativo.

Operators

- Operadores aritméticos:
  +, -, *, /, %, ++, --
- Operador de asignación:
  =
  operadores combinados: .=, +=, etc
  $a = 3; $a += 5; → a vale 8
  $b = “hola ”; $b .= “mundo”; → b vale “hola mundo”
  → Equivale a $b = $b . “mundo”;
- Operadores de comparación:
  ==, !=, <, >, <=, >= y otros
- Operador de control de error: @. Antepuesto a una expresión, evita cualquier mensaje de error que pueda ser generado por la expresión
- Operadores lógicos:
  and (&&), or (||), !, xor
  &&/and y ||/or tienen diferentes prioridades
- Operadores de cadena:
  concatenación: . (punto)
  asignación con concatenación: .=

Control structures

- Estructuras selectivas:
  - if-else
  - switch
- Estructuras repetitivas:
  - while
  - for
  - foreach

Decision structures

- Estructura selectiva **if-else**

```
if (condición)
   sentencia
```

```
if (condición)
   sentencia 1
else
   sentencia 2
```

```
if (condición1)
   sentencia 1
else if (condición2)
   sentencia 2
...
else if (condición n)
   sentencia n
else
   sentencia n+1
```

- Mismo comportamiento que en Java y C.
- Las sentencias compuestas se encierran entre llaves.
- elseif puede ir todo junto.

Decision structures 2

- Ejemplo de estructura if-else

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

Decision structures 3

- Estructura selectiva **switch**

```
switch (expresión)
{
   case valor_1:
        sentencia 1
        break;
   case valor_2:
        sentencia 2
        break;
   …
   case valor_n:
        sentencia n
         break;
   default
        sentencia n+1
}
```

- Mismo comportamiento que en Java y C, sólo que la expresión del case puede ser integer, float o string

Decision structures 4

- Ejemplo de estructura selectiva switch:

```
switch ($extension)
{
   case ("PDF"):
      $tipo = "Documento Adobe PDF";
      break;
   case ("TXT"):
      $tipo = "Documento de texto";
      break;
   case ("HTML"):
   case ("HTM"):
      $tipo = "Documento HTML";
      break;
  default:
      $tipo = "Archivo " . $extension;
}
print ($tipo);
```

Loops

- Las estructuras iterativas while,do ... while, for tienen el mismo comportamiento que en Java y C.
- Ejemplo de estructura while:

```
<?PHP
print ("<ul>\n");
$i=1;
while ($i <= 5) {
   print ("<li>Elemento $i</li>\n");
   $i++ };
print ("</ul>\n");
?>
```

- Ejemplo de estructura for:

```
<?PHP
print ("<ul>\n");
for ($i=1; $i<=5; $i++)
   print ("<li>Elemento $i</li>\n");
print ("</ul>\n");
?>
```

Functions

- El comportamiento de las funciones es similar al de otros lenguajes. Los parámetros pueden ser pasados por valor o por referencia. Tenemos una sentencia return para devolver un valor.
- Veamos un ejemplo:

```
function suma ($x, $y)
{
   $s = $x + $y;
   return $s;
}
$a=1;
$b=2;
$c=suma ($a, $b);
print $c;
```

Functions 2

Por defecto los parámetros se pasan por valor

- Paso por referencia:

```
function incrementa (&$a)
{
   $a = $a + 1;
}

$a=1;
incrementa ($a);
print $a; // Muestra un 2
```

Functions 3

- Argumentos por defecto. Deben ser siempre el último parámetro si existiese más de uno.

```
function muestranombre ($titulo = "Sr.")
{
    print "Estimado $titulo:\n";
}
muestranombre ();
muestranombre ("Prof.");
```

- Salida:

```
Estimado Sr.:
Estimado Prof.:
```

Functions

- Interesting functions
  - ctype_alpha($cadena)
    - Return true or false depending on $cadena contains only letters or not.
  - is_numeric($cadena)
    - Return true or false depending on $cadena contains only numbers or not.

String fuctions

- string chr(int ascii)
  - Devuelve el carácter en forma de cadena del ascii especificado.
- String sprintf(string formateado)
  - Devuelve una cadena aplicando el formato especificado
  - Cada especificación de conversión consiste en un signo de porcentaje (%), seguido por uno o más de estos elementos, en orden:
    - Especificador de signo en caso numérico (+/-)
    - Especificador de relleno (espacio/0)
    - Especificador de alineación (- izquierda)
    - Especificador de ancho (caracteres mínimos)
    - Especificador de precisión (.decimales)
    - Especificador de tipo: d (entero signo), u (entero sin signo), e (notación científica), f (punto flotante), s (string).

String functions2

- Ejemplos

```
$money1 = 68.75455;
$money2 = 54.35344;
$money = $money1 + $money2;
$formatted = sprintf("%08.2f", $money);
echo $formatted; //producirá "00123.11"
```

- printf(cadena formateada)
  - Imprime por pantalla la cadena formateada.
- trim, rtrim y ltrim.
  - Elimina espacios en blanco.
- int strlen(string)
  - Determina la longitud de la cadena.
- strpos($cadena,$buscada)
  - Devuelve false si la cadena buscada no está dentro de la cadena. En caso contrario devuelve la posición en cadena a partir de la cual está la buscada.
- String substr(string cad,int start,int len)
  - Devuelve una subcadena de cad que comienza por la posición 'start' y de 'len' longitud.
- explode and implode

Arrays

Desarrollo Web Entorno Servidor
PHP Básico

- Un array es sencillamente una tabla de valores. Cada uno de los elementos de esa tabla se identifica por medio de un nombre (común para todos) y un índice (que diferenciaría a cada uno de ellos). La sintaxis que permite definir elementos en un array es esta: $nombre[indice]
- Cuando los índices de un array son números se dice que es *escalar* mientras que si fueran cadenas se le llamaría array *asociativo*.

Tablas unidimendionales					
Array escalar			Array asociativo		
Variable	Indice	Valor	Variable	Indice	valor
$a[0]	0	Domingo	$a['primero']	primero	Domingo
$a[1]	1	Lunes	$a['segundo']	segundo	Lunes

Arrays 2

- En php no se especifican las dimensiones del array antes de su uso. Va creciendo dinámicamente.
- En un array escalar, si no especificamos un índice se asignaría el valor al siguiente del último índice asignado.
  - `$a[0]=2;`
  - `$a[]=7; #Lo asignaría al índice 1.`
  - `$a[5]=5; #los valores para los índices 2,3,4 estarían vacíos`
- Arrays Bidimensionales.
  - Se definen mediante dos corchetes.
    - `$a[][]=3; #Es el elemento (0,0)`
    - `$partido['Barça']['Madrid']=”2-1”;`

array function

- Para asignar valores a una matriz puede usarse la función array(), que tiene la siguiente sintaxis:

```
$a= array (índice 0 => valor,
..... ,
índice n => valor,);
```

- Por ejemplo:

```
$z=array (0 =>2,
1 =>”Pepe”,2 =>34.7,3 =>”34Ambrosio”,);
```

producirá igual resultado que:

```
$z[0]=2;
$z[1]="Pepe";
$z[2]=34.7;
$z[3]="34Ambrosio";
```

- Some array functions
- Arrays with keys
- Sorting arrays

Foreach loop

- El bucle foreach es específico de los array y aplicable a ellos tanto si son escalares como si son de tipo asociativo. Tiene dos posibles opciones. En una de ellas lee únicamente los valores contenidos en cada elemento del array. En el otro caso lee además los índices del array.
- Ejemplos:

```
$a=array("a","b","c","d","e");
$b=array(
"uno" =>"Primer valor",
"dos" =>"Segundo valor",
"tres" =>"Tecer valor",);
foreach($a as $valor) {echo $valor,"<br>";}
foreach($b as $valor) {echo $valor,"<br>";}
```

foreach loop 2

- Con una sintaxis como la que sigue se pueden leer no sólo los valores de un array sino también sus índices.

```
foreach( array as ind => val ) {
...instrucciones...}
```

donde array es el nombre del vector o la tabla, as es una palabra obligatoria, ind es el nombre de la variable que recogerán los índices, los caracteres => (son obligatorios) son el separador entre ambas variables y, por último, val es el nombre de la variable que recoge el valor de cada uno de los elementos del array.