Lenguaje PHP

DESARROLLO BACK-END

¿ Que es PHP ?

PHP es el acrónimo de **P**HP (*Personal Home Page*) **H**ypertext **P**reprocessor

Es un Lenguaje de Scripting del lado del Servidor

Fue creado en 1994 por Rasmus Lerdorf

Actualmente mantenido por The PHP Group (www.php.net)

Publicado bajo licencia PHP – Software de Código Libre

Lenguaje Multiplataforma – Independiente de la plataforma

Corre en los Servidores Web mas comunes (Apache / IIS)

Existen IDE de productores de Software como Zend Studio, Delphi for PHP, Eclipse

Lenguaje PHP básico

1. Sintaxis básica
2. Tipos de datos
3. Variables
4. Constantes
5. Expresiones y operadores
6. Estructuras de control
7. Funciones
8. Tablas
9. Bibliotecas de funciones

Sintaxis básica

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

  `<?PHP ... ?>`

  recomendado, siempre disponible

  `<?= expresión ?>`

  equivale a <? echo expresión ?>
- Las instrucciones se separan con un ; como en 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
```

Sintaxis básica

- Para imprimir: **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”;
```

Sintaxis básica

- Ejemplo:

```
<HTML>
<HEAD>
<TITLE>Mi primer programa en PHP</TITLE>
</HEAD>

<BODY>

<?PHP
   print (“<P>Hola mundo</P>”);
?>

</BODY>
</HTML>
```

Sintaxis básica

- Uso de \n para generar código HTML legible
- a) Sin \n

Código PHP

```
print ("<P>Párrafo 1</P>");
print ("<P>Párrafo 2</P>");
```

Código HTML

```
<P>Párrafo 1</P><P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica

- Uso de \n para generar código HTML legible
- b) Con \n

Código PHP

```
print ("<P>Párrafo 1</P>\n");
print ("<P>Párrafo 2</P>\n");
```

Código HTML

```
<P>Párrafo 1</P>
<P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica

- Inclusión de ficheros externos:

include() / **include_once()**

require() / **require_once()**

- Ambos incluyen y evalúan el fichero especificado
- Diferencia: en caso de error include() produce un warning y require() un error fatal
- Se usará require() si al producirse un error debe interrumpirse la carga de la página

Sintaxis básica

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

Tipos de datos

- PHP soporta 8 **tipos de datos primitivos**:
  - Tipos comunes: 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

Tipos de datos

- 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
  - isset() verifica si un variable existe
  - unset() Destruye una variable
  - settype() Setea el tipo de una variable
  - empty() Verifica si una variable esta vacia
  - intval(), doubleval(), strval() Convierte el tipo de varible

Tipos de datos

- Tipo **integer** (números enteros)
  - 27, -5, 0

```
$entero_base10 = 1234;
$entero_base8 = 01234;
$entero_base16 = 0x1234;
$entero_negativo = -1234;
```

- Tipo **double** (números reales)

1.234, -5.33

- Tipo **boolean** (lógico)
  - Valores: *true*, *false* (insensibles a las mayúsculas)
  - El 0 y la cadena vacía tienen valor *false*

Tipos de datos

- Tipo 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 **SÍ** se expanden
    - 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]; o $nombre{0};
    -

Tipos de datos

- Tipo string:
- Ingreso de Variables de Multiples lineas:

```
<?php
$formulario = <<<INICIO
<form>
<ínput type="text" name="Nombre" value="Luís Miguel
   Cabezas">
<br>
<input type="submit" name="submit" value="Enviar" >
</form>
INICIO;
echo $formulario;
?>
```

Variables

- El Nombre del las variables siempre van precedidas de un $
- El nombre es sensible a las mayúsculas
- Debe comenzar por letra o subrayado, seguido de letras, números o subrayado
- Variables predefinidas:

$GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIES, $_FILES,

$_ENV, $_REQUEST, $_SESSION

- Ejemplo:

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

Resultado:
   El valor es: 5
```

Variables

Ámbito:

Globales:Toda variable definida dentro de un documento php que no este dentro de una funcion, tambien se pueden llamar con el array $GLOBALS

Locales: Toda funcion definida dentro de una funcion es local a ella

Estaticas: Definidas con la clave *static*, son variables locales que permanecen luego de cerrada la funcion

Parametros: Es una variable local pasada a la funcion por el procedimiento que la llama. Son locales a la funcion

Variables

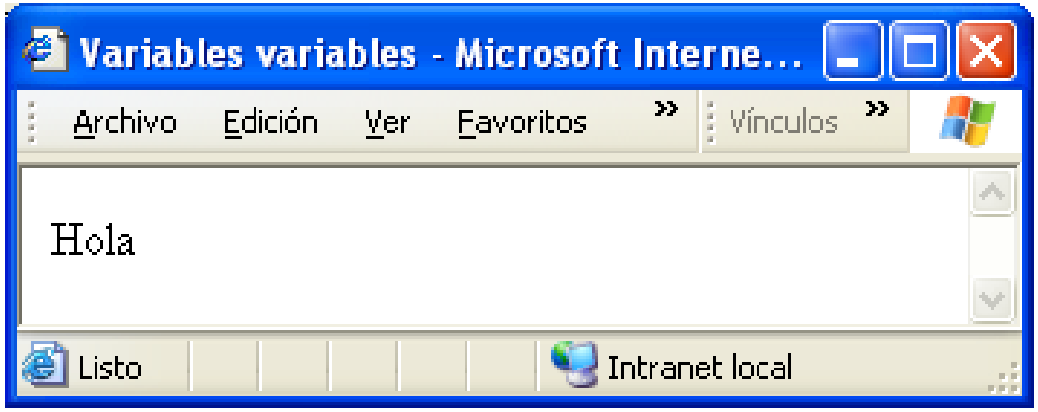
- Variables *variables*
  - Se pueden crear nombres de variables dinámicamente
  - La variable *variable* toma su nombre del valor de otra variable previamente declarada
  - Ejemplo:

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

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

Resultado:
 hola mundo
 hola mundo
```

Variables

- Ejemplo de variables *variables*: página internacionalizada

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



Constantes

Las constantes son tipos de datos que no varían en el desarrollo de un programa. En la vida real existen muchos tipos de constantes, el número pi, la temperatura de congelación del agua, el nombre de la Empresa, etcétera.

- Definición de constantes:

```
define (“CONSTANTE”, “hola”);
print CONSTANTE;
```

- No llevan $ delante
- Sólo se pueden definir constantes de los tipos comunes (boolean, integer, double, string)
- Funcion `defined()` – Si existe una constante dada
- Constantes Mágicas (Ej. __LINE__)
- Constantes Predefinidas (Ej. PHP_VERSION)

Expresiones y operadores

- 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:
  ==, !=, <, >, <=, >=
- Operador Ternario

Expresion ? Verdadero : Falso ;

- 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
- Operadores de cadena:
  concatenación: . (punto)
  asignación con concatenación: .=

Expresiones y operadores

- Precedencia de operadores (de mayor a menor):

```
++, --
*, /, %
+,-
<, <=, >, >=
==, !=
&&
||
and
or
```

Estructuras de control

- Estructuras selectivas:
  - *if-else*
  - *switch*
- Estructuras repetitivas:
  - *while*
  - *do …. while*
  - *for*
  - *foreach*
- Control de Ejecución
  - *die*
  - *exit*

Estructuras de control

- 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 C
- Las sentencias compuestas se encierran entre llaves
- elseif puede ir todo junto

Estructuras de control

- Ejemplo de estructura selectiva if-else:

```
<?PHP
   if ($sexo == 'M')
    $saludo = "Bienvenida, ";
   else
    $saludo = "Bienvenido, ";
   $saludo = $saludo . $nombre;
   print ($saludo);
?>
```

Estructuras de control

- 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 C, sólo que la expresión del case pue ser integer, float o string

Estructuras de control

- 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);
```

Estructuras de control

- Estructura repetitiva **while**

```
while (condición)
   sentencia
```

- Mismo comportamiento que en C

Estructuras de control

- Ejemplo de estructura repetitiva do … while:

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

Estructuras de control

- Ejemplo de estructura repetitiva while:

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

Estructuras de control

- Estructura repetitiva **for**

```
for (inicialización; condición; incremento)
   sentencia
```

- Mismo comportamiento que en C

- Comandos `break y continue`

Estructuras de control

- Ejemplo de estructura repetitiva for:

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

Estructuras de control

- Ejemplo de estructura repetitiva for múltiple:

```
<?php
for ($x = 1 , $y = 1, $z = 1; $y < 10, $z < 10; $x++, $y
   = $y -
2, $z = $z + 3) {
echo ("$x, $y, $z<br>");
}
?>
```

Estructuras de control

- **Ejemplo de estructura repetitiva foreach:**

- La estructura *foreach* simplemente da un modo fácil de iterar sobre arrays. *foreach* funciona solo sobre arrays (y objetos) y resultará en un error al intentar usarlo en una variable con un diferente tipo de datos o una variable no inicializada

```
<?php
   $arr = array(1, 2, 3, 4);
   foreach ($arr as &$value) {
       $value = $value * 2;
   }
   // $arr ahora es array(2, 4, 6, 8)
   ?>
```

Estructuras de control

Finalización de la ejecución de un programa

- `die()` y `exit()`

Funciones

- Ejemplo:

```
function suma ($x, $y)
{
   $s = $x + $y;
   return $s;
}

$a=1;
$b=2;
$c=suma ($a, $b);
print $c;
```

Funciones

- 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
```

- Los valores de variable dentro de una funcion son locales y dinamicas
- Si quiero variables estaticas:

```
<?php
function contador() {
static $contador = 0;
$contador = $contador + 1;
return $contador;
}
```

Funciones

- Argumentos por defecto

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

- Salida:

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

Funciones de manipulación de argumentos:

`func_num_args() , func_get_arg() y func_get_args()`

Funciones

- Los argumentos con valores por defecto deben ser siempre los últimos:

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

- Salida:

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

Funciones

Funciones de manipulación de argumentos:

`func_num_args() , func_get_arg() y func_get_args()`

```
<?php
function capitales() {
$numero_argumentos = func_num_args();
$Pais = $numero_argumentos > 0 ? func_get_arg(0) : "España";
$Capital = $numero_argumentos > 1 ? func_get_arg(1) : "Madrid";
$habitantes = $numero_argumentos > 2 ? func_get_arg (2) :
"muchos " ;
return ("Número de argumentos es: $numero_arguraentos. La capital
de $Pais es $Capital y tiene $habitantes habitantes.<br>");
}
echo capitales();
echo capitales("Portugal", "Lisboa");
echo capitales("Francia", "Paris", "muchísimos");
?>
```

Arrays (Tablas)

- Sintaxis:

```
array ([clave =>] valor, ...)
```

- La clave es una cadena o un entero no negativo. El valor puede ser de cualquier tipo válido en PHP, incluyendo otro array
- Ejemplos:

```
$color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);
$medidas = array (10, 25, 15);
```

- Acceso:

```
$color['rojo'] // No olvidar las comillas
$medidas[0]
```

- El primer elemento es el 0

Arrays (Tablas)

- Creación
- Asignación directa:

Asignando subindice

```
$mi_array[1] = 23; // Asignación directa
```

Subindice implicito

```
<?php
$mi_array[] = 23; // Empieza en el índice 0
$mi_array[] = 54; // índice 1
?>
```

Arrays (Tablas)

- Creación
- Función `array()`

```
<?php
$mi_array = array(23,45,76,23,65);
?>
```

- Función `array()` utilizando índices no implícitos

```
<?php
$mi_array = array(0 => 23, 1 => 45, 2 => 76);
?>
<?php
$mi_array = array("cero" => 23, "uno" => 45, 2 => 76);
?>
```

Arrays (Tablas)

- Array Multidimensional

```
<?php
$colores = array( "fuertes" => array ( "rojo" => "FF0000",
                  "verde" => "00FF00",
                  "azul" => "0000FF"),
       "suaves" => array ( "rosa" => "FE9ABC",
                  "amarillo" => "FDF189",
                  "malva" => "9A2F68"));
echo $colores["fuertes"] ["rojo"];
?>
```

Bibliotecas de funciones

- Existen muchas bibliotecas de funciones en PHP
- Algunos ejemplos:
  - Funciones de manipulación de cadenas
  - Funciones de fecha y hora
  - Funciones de arrays
  - Funciones de ficheros
  - Funciones matemáticas
  - Funciones de bases de datos
  - Funciones de red
- Algunas bibliotecas requieren la instalación de componentes adicionales
- Todas las funciones de biblioteca están comentadas en la documentación de PHP

Bibliotecas de funciones

- Funciones de manipulación de cadenas
  - explode()
    - Divide una cadena en subcadenas
    - array **explode** (string separator, string string [, int limit])
  - rtrim(), ltrim(), trim()
    - Eliminan caracteres a la derecha, a la izquierda o por ambos lados de una cadena
    - string **rtrim** (string str [, string charlist])
  - strstr()
    - Busca la primera ocurrencia de una subcadena
  - strtolower() / strtoupper()
    - Convierte una cadena a minúscula / mayúscula
  - strcmp() / strcasecmp()
    - Compara dos cadenas con/sin distinción de mayúsculas
  - strlen()
    - Calcula la longitud de una cadena

Bibliotecas de funciones

- Funciones de fecha y hora
  - date()
    - Formatea una fecha según un formato dado
    - Ejemplo:

```
$fecha  = date ("j/n/Y H:i");
print ("$fecha");
Resultado:
 26/9/2005 17:36
```

  - strtotime()
    - Convierte una fecha en un *timestamp* de UNIX
    - Ejemplo:

```
$fecha = date ("j/n/Y", strtotime("5 april 2001"));
print ("$fecha");
Resultado:
 5/4/2001
```

Bibliotecas de funciones

- Funciones de arrays
  - array_count_values()
    - Calcula la frecuencia de cada uno de los elementos de un array
  - array_search()
    - Busca un elemento en un array
  - count()
    - Cuenta los elementos de un array
  - sort(), rsort()
    - Ordena y reindexa un array (r=decreciente)
  - ksort(), krsort()
    - Ordena por claves un array (r=decreciente)