



Departamento de Lenguajes y
Sistemas Informáticos



Universitat d'Alacant
Universidad de Alicante

PHP

Programación en Internet
Curso 2006-2007

Programación en Internet – Curso 2006-2007

Índice (I)

- Introducción
 - Ventajas
 - Multiplataforma
- Características del lenguaje
 - Sintaxis:
 - Variables
 - Arrays
 - Cadenas de texto
 - Constantes
 - Operadores
 - Sentencias de control
 - Funciones
- Funciones y librerías
 - Include/Require
 - Manejo de ficheros
 - Acceso al sistema de ficheros
 - Funciones matemáticas y de fechas

Programación en Internet – Curso 2006-2007

Índice (II)

- Orientación a objetos
- Entorno de desarrollo en Internet
 - Variables predefinidas
 - Objetos y funciones del entorno
 - Tratamiento de formularios
 - Tratamiento de sesiones
 - Subir ficheros
 - Cabeceras HTTP

Programación en Internet – Curso 2006-2007

Índice (y III)

- Acceso a bases de datos
 - Conceptos generales
 - Acceso a MySQL
 - Acceso a ODBC
- Depuración
- Otras funcionalidades
- Bibliografía

Introducción

- PHP: *Hipertext Preprocessor*
- Historia:
 - Inicio del desarrollo: otoño de 1994
 - PHP Versión 1 en primavera 1995
 - PHP Versión 2 1995-1997
 - PHP Versión 3 1997-2000
 - PHP Versión 4 en el segundo trimestre de 2000
- Tecnología de servidor interpretada, el código se intercala entre el HTML
- Basado en C, C++, Java, Awk, Perl y Bash (*shell script* de Unix).
- Tiene características de OO: permite crear clases y objetos

Ventajas

- Multiplataforma
- Alta velocidad de respuesta (¿el más rápido?)
- Software libre bajo licencia GPL:
 - Es gratuito
 - El código fuente está disponible
 - Y existe el permiso para modificarlo
- Existe una gran cantidad de módulos y añadidos para complementar y aumentar sus prestaciones
- La curva de aprendizaje es baja, ya que está basado en lenguajes conocidos y muy comunes

Multiplataforma

- Funciona sobre diversos sistemas operativos:
 - UNIX (todas las variantes)
 - Win32 (NT/W95/W98/W2000/XP)
 - Mac (WebTen), OS/2, BeOS
- Funciona con distintos servidores web:
 - Apache (UNIX, Win32)
 - ISAPI (IIS, PWS)
 - NSAPI (Netscape iPlanet)
 - Java servlet
 - AOLServer
- Permite el acceso a más de 20 SGBD:
 - Nativo: Oracle, BD2, Informix, MySQL, PostgreSQL, Sybase, dBase
 - ODBC: MS-Access, SQL-Server, etc.

Ficheros de PHP

- Los ficheros que contienen código PHP tienen que tener una de las siguiente extensiones:
 - .php3, para código de la versión 3
 - .php4, para código de la versión 4
 - .php, genérico y el más utilizado
 - .phtml, cada vez menos utilizado
- PHP4 es compatible con PHP3 (a excepción de unas pocas características)

Características del lenguaje (I)

- Delimitadores de código de servidor:

```
<? ... ?>
```

```
<?php ... ?>
```

```
<script language="php">...</script>
```

```
<% ... %>, no siempre disponible, según  
configuración del intérprete
```

- Comentarios:

```
/* Comentario tipo C  
   multilínea */
```

```
// Comentario tipo C++, una sola línea
```

```
# Comentario tipo Bash/Perl, una línea
```

Características del lenguaje (y II)

- Final de instrucción: punto y coma (;)
- Para imprimir cadenas de caracteres:

```
echo "cadena de texto";
```

```
<?="cadena de texto"?>
```

- Mayúsculas y minúsculas:

– Con los nombres de variable, Sí que importan:

```
$MiNumero es diferente de $minumero
```

– Con los nombres de funciones y palabras reservadas, NO importa:

```
PRINT() es igual que print()
```

Variables (I)

- PHP es un lenguaje débilmente tipado
- No hace falta declarar las variables, se declaran automáticamente al aparecer por primera vez en el código fuente
- TODAS LAS VARIABLES LLEVAN EL SÍMBOLO \$ DELANTE DE SU NOMBRE:

```
$var1 = 123;  
$var2 = 'hola mundo';
```
- El tipo de las variables es 'mixed', similar tipo 'variant' en VBScript
- Sin embargo, hay unos tipos básicos:
 - int, integer → Enteros. 0NNN en base 8, 0xNN en base 16.
 - float, double, real → Coma flotante
 - array, string, object

Variables (II)

- Ejemplos de tipos:
 - Enteros, en decimal, octal o hexadecimal:

```
$Var = 123;
```
 - Coma flotante:

```
$Var = 1.3e4;
```
 - Arrays.

```
$Var[2] = 123;
```
 - Cadenas:

```
$Var = "Cadena de texto\n";
```
 - Objetos:

```
$Var = new oClase();
```

Variables (III)

- Una variable puede tener diferentes tipos a lo largo del tiempo
- Para evitar errores y ambigüedades, PHP realiza las conversiones de tipo necesarias (*casts*) a la hora de operar con variables y contenidos de tipos diferentes:

```
$num = 123;  
echo $num; //num se transforma en String
```

- Para realizar una conversión explícita:

```
$var = (string)123;
```
- También se puede cambiar el tipo con `settype()`:

```
$var = 12;  
settype($var, double);
```

Variables (IV)

- **Ámbito** → Según el lugar donde esté declarada:
 - Global a un fichero
 - Local a una función
 - Local a una clase/objetos (variables de clase o atributos). Accesibles mediante operador ‘->’
- Para acceder a una variable global desde una función → ‘global’

```
$mivar = 3;  
function mifuncion() {  
    global $mivar;  
    echo $mivar;  
}
```

Variables (y V)

- Se pueden definir 'alias' de una variable: dos o más variables que apuntan a un mismo dato (como si fueran punteros)

- El operador '&' para obtener las referencias:

```
$malnom = &$variable;
```

- Se emplea `unset()` para eliminarlas:

```
unset ($malnom);
```

- Se pueda acceder al contenido de una variable (v1) a través de otra variable (v2) que almacena el nombre de la variable (v1) mediante '\$\$':

```
$a = 123;  
$b = 'a';  
echo $$b; // 123
```

Arrays (I)

- Se declaran y acceden los elementos con los corchetes: `[]`
- La primera componente es la cero
- Los elementos pueden tener distinto tipo en un mismo vector
- Se puede acceder a un elemento mediante un índice asociativo (tablas hash)
- Arrays multidimensionales
- Constructor: `array()`

Arrays (II)

- Tipos de los elementos:

```
$vector1[0] = 1;  
$vector1[1] = 'hola';  
$vector1["nom"] = "juan";
```

- Constructor:

```
$vector2 = array (1, "jorge", 3);  
$vector3 = array(  
    0 => 1,  
    1 => "jorge",  
    "nom" => "jaime",  
    3 => 5 );  
// índice => valor
```

Arrays (III)

- Otra forma:

```
$a[] = 'a';  
$a[] = 'b';  
$a[] = 'c';  
// equivale a  
$a = array('a', 'b', 'c');
```

Programación en Internet – Curso 2006-2007

Arrays (IV)

- Un índice puede ser una cadena o un entero
- Cuando el índice es una cadena, no existe el correspondiente índice entero
- Cuando el índice se omite, automáticamente se genera un índice empezando desde 0
- Si un índice es un entero, el siguiente índice generado será el mayor índice entero + 1
- Cuando se definen dos índices idénticos, el último sobrescribe al primero

Programación en Internet – Curso 2006-2007

Arrays (V)

```
$firstquarter = array(1 => 'January', 'February',  
    'March');
```

```
print_r($firstquarter);
```

Genera:

```
Array ( [1] => January [2] => February [3] =>  
    March )
```

Nota: Array que empieza en 1 en vez de 0

Programación en Internet – Curso 2006-2007

Arrays (VI)

```
$fruits = array (
  "fruits" => array("a" => "orange", "b" => "banana", "c" =>
    "apple"),
  "numbers" => array(1, 2, 3, 4, 5, 6),
  "holes" => array("first", 5 => "second", "third")
);

print_r($fruits);
```

Genera:

```
Array (
  [fruits] => Array ( [a] => orange [b] => banana [c] =>
    apple )
  [numbers] => Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4
    [4] => 5 [5] => 6 )
  [holes] => Array ( [0] => first [5] => second [6] => third
    ) )
```

Genera: array multidimensional

Programación en Internet – Curso 2006-2007

Arrays (VII)

```
$array = array(1, 1, 1, 1, 1, 8 => 1, 4 => 1,
  19, 3 => 13);

print_r($array);
```

Genera:

```
Array ( [0] => 1 [1] => 1 [2] => 1 [3] => 13 [4]
  => 1 [8] => 1 [9] => 19 )
```

Genera:

- El valor 13 sobrescribe al anterior de la componente 3
- El valor 19 se aloja en la componente 9, que es la de valor máximo (8) más 1

Arrays (VIII)

- `count($array)`: devuelve el número de elementos del vector '\$array'
- `in_array($elem, $array[, $strict])`: busca el elemento '\$elem' dentro del array '\$array' y devuelve cierto o falso según esté o no esté (si '\$strict' vale true, tiene en cuenta los tipos de los valores)
- Recorrido de un array:
 - `reset($array)`: posiciona el puntero que recorre un array en la primera posición
 - `current($array)`: devuelve el elemento actual
 - `next($array)` y `prev($array)`: avanza el puntero hacia delante o hacia atrás
 - `end($array)`: posiciona el puntero que recorre un array en la última posición

Arrays (IX)

- `list($var1, ...)`: asigna valor a una lista de variables en una sola operación
- `each($array)`: devuelve el par clave/valor actual para el array y avanza el cursor
 - Devuelve un array de 4 elementos (0, 1, key, value)
- Otra forma de recorrer un array:

```
$unarray = array('uno', 'dos', 'tres');
```

```
reset($unarray);
```

```
while (list($clave, $valor) = each($unarray))  
    echo "$clave => $valor\n";
```

Arrays (X)

- `sort($array, $flags)`: ordena los valores de un array según el criterio establecido en `$flags` (`SORT_NUMERIC`, `SORT_REGULAR`, `SORT_STRING`, ...)
- `explode($limite, $cadena)`: trocea 'cadena' según el carácter '\$limite' y pone cada trozo dentro de un array
- `implode($union, $array)`: une los elementos de '\$array' en una cadena poniendo entre cada elemento el carácter '\$union'.

Arrays (y XI)

- Otras funciones:
 - `array_diff()`: calcula las diferencias entre dos arrays
 - `array_fill()`: rellena un array con valores
 - `array_reverse()`: devuelve un array con los elementos en orden inverso
 - `array_search()`: buscar un valor y devuelve su posición
 - `array_sum()`: calcula la suma de los valores
 - `array_walk()`: aplica una función a cada valor de un array

Cadenas de texto (I)

- Delimitador “””: inserta el valor de las variables
- Delimitador “”: no inserta el valor de las variables.
Escape: \\$, \\, \', Más rápido que el método anterior
- Tipo ‘*here doc*’ de Perl: se comporta como “””.

```
$cadena = <<<DELIMITADOR
text
text
text
...
text
DELIMITADOR
```
- Concatenación con el operador punto ‘.’
- Se puede acceder a un carácter de una cadena con [] (empieza en 0)

Cadenas de texto (II)

- Caracteres escapados:

Secuencia	Significado
\n	Nueva línea (LF o 0x0A en ASCII)
\r	Retorno de carro (CR o 0x0D en ASCII)
\t	Tabulación horizontal (HT o 0x09 en ASCII)
\\	Barra invertida
\\$	Símbolo del dólar
\"	Comillas dobles
\'	Comillas simples
\[0-7]{1,3}	Un carácter en notación octal
\x[0-9A-Fa-f]{1,2}	Un carácter en notación hexadecimal

Cadenas de texto (III)

- Con comillas dobles:
`$cad1 = "hola.\n\${var1}=${var1}";`
- Sintaxis *'here doc'* de Perl:
`$cad2 = <<<FICAD`
Esto es un ejemplo de cadena.
La variable `\$a` val `$a`.
Ahora termina la cadena:
`FICAD`
- Concatenación:
`$cad1 = $cad1 . $cad2;`
- Acceso a caracteres:
`$character5 = $cad1[4];`

Cadenas de texto (IV)

- Funciones:
 - `substr($cadena, $inicio, $lon)`: devuelve una subcadena de longitud '`$lon`' a partir de la posición '`$inicio`' de la cadena '`$cadena`'
 - `strpos($cadena, $subcadena)`: devuelve la posición donde '`$subcadena`' aparece en '`$cadena`'
 - `trim($cadena)`: elimina espacios en blanco al inicio y al final de '`$cadena`'
 - `strrev($cadena)`: devuelve la cadena invertida

Cadenas de texto (y V)

- Funciones:
 - `chr($int)`: devuelve el carácter ASCII del número `$int`
 - `ord($car)`: devuelve el código ASCII del carácter `$car`
 - `strlen($cad)`: devuelve el número de caracteres de `$cad`
 - `printf()`, `sprintf()`: imprime cadenas con formato (similar al funcionamiento en C)

Constantes

- Se declaran con la función `define()`
 - `define('nomConst', valor, noMayúsculas)`
 - Si `noMayúsculas` es 1, el nombre de la constante no será 'case sensitive'.
- No llevan el símbolo '\$' al principio
- Son globales y se pueden acceder desde cualquier parte del código
- Sólo pueden ser valores escalares
- No se puede modificar
- Ejemplo:

```
define('slt', 'Hola mundo!');  
echo "La constante slt vale: " . slt;
```


Operadores

- Aritméticos: +, -, *, /, %
- Autoincremento y autodecremento:
\$a++, ++\$a, \$a--, --\$a
- De bits: &(AND), |(OR), ^(XOR), ~(NOT), >>, <<
- Lógicos: and, or, xor, !, &&, ||
- Comparaciones: ==, ===, !=, !==, <, >, <=, >=.
- Cadenas: '.', (concatenación)

Evaluación cierto/falso

- Para los valores numéricos, 0 es falso y cualquier otro valor verdadero
- Para cadenas de texto, una cadena vacía ("") es falso y una cadena no vacía es verdadero
 - Excepción: una cadena con valor "0"
- Para los arrays, será falso si no contiene ningún elemento y verdadero en caso contrario
- Para objetos, un objeto se evalúa a falso si es un objeto vacío (su clase no contiene ni métodos ni atributos) y verdadero en caso contrario
- Dos constantes: TRUE y FALSE
 - TRUE es un entero de valor 1
 - FALSE es una cadena de caracteres vacía

Asignación, igualdad e identidad (1)

- Asignación: =
 - \$a = \$b, asigna a \$a el valor de \$b.
- Igualdad: ==, !=
 - \$a == \$b, compara si los valores de los dos operandos son iguales
 - \$a != \$b, compara si los valores de los dos operandos son distintos
- Identidad: ===, !==
 - \$a === \$b, compara si los valores son iguales y además si el tipo de los operandos coincide
 - \$a !== \$b, compara si los valores son distintos y además si el tipo de los operandos no coincide

Asignación, igualdad e identidad (2)

```
if("0" == 0)
  echo "SI";
else
  echo "NO";

if("0" === 0)
  echo "SI";
else
  echo "NO";
```

Programación en Internet – Curso 2006-2007

Asignación, igualdad e identidad (y 3)

```
if("0" == 0)
  echo "SI"; → SI
else
  echo "NO";

if("0" === 0)
  echo "SI";
else
  echo "NO"; → NO
```

Programación en Internet – Curso 2006-2007

Otros operadores de asignación

- +=, -=
- *=, /=, %=
- &=, ^=
- .=
- >>=, <<=

Programación en Internet – Curso 2006-2007

Asociatividad	Operando
Izquierda	,
Izquierda	or
Izquierda	xor
Izquierda	and
Derecha	print
Izquierda	= += -= *= /= .= %= &= = ^= -= <<= >>=
Izquierda	?:
Izquierda	
Izquierda	&&
Izquierda	
Izquierda	^
Izquierda	&
No asociativo	== != === !==
No asociativo	< <= > >=
Izquierda	<< >>
Izquierda	+ - .
Izquierda	* / %
Derecha	! ~ ++ -- (int) (double) (string) (array) (object) @
Derecha	[
No asociativo	new

Programación en Internet – Curso 2006-2007

Operador ?:

- Funciona igual que en C y C++:
`-(expr1) ? (expr2) : (expr3);`
- Ejemplo:
`$cad = $a > $b ? "a es mayor que b" :
"a no es mayor que b";`

Operador @

- Operador de control de errores: cuando se añade como prefijo a una expresión, cualquier error que pueda generar la expresión será ignorado
- Ejemplo:

```
$fichero = @file('unfichero.txt')
```

- Si la función produce un error, `$fichero` contendrá `null`, pero se evita que el intérprete de PHP inserte el mensaje de error correspondiente

Sentencias de control de flujo

- Como en C, C++, Java y JavaScript:
 - if...elseif...else
 - switch
 - while, do...while
 - break y continue
 - for
- Similar a JavaScript:
 - foreach

if...elseif...else

- Simple:

```
if (expresión) {
  Instrucciones
}
```
- Con un else:

```
if (expresión) {
  Instrucciones si
  true
}
else {
  Instrucciones si
  false
}
```
- Con 'elseif'

```
if (expresión) {
  Instrucciones si true
}
elseif (expresión 2) {
  Instrucciones bloque 2
}
...
else {
  Instrucciones finals
}
```

switch...case...default

```
switch (variable) {
  case valor1:
    instrucciones1
    break;
  case valor2:
    instrucciones2
    break;
  ...
  case valorN:
    instruccionesN
    break;
  default:
    instruccionesDefault
}
```

Programación en Internet – Curso 2006-2007

while y do...while

```
while (expresión) {  
    instrucciones  
}  
-----  
do {  
    instrucciones  
} while (expresión);
```

Programación en Internet – Curso 2006-2007

break y continue

- `break;`
 - Nos permite salir inmediatamente de un bucle
- `break n;`
 - Nos permite salir de 'n' bucles anidados
- `continue;`
 - Termina la iteración actual y pasa a la siguiente

for (...)

- Tiene una estructura similar a la de C:

```
for (expresión1; expresión2;
    expresión3) {
    instrucciones
}
```

For: ejemplos

- Uno:

```
$factorial5 = 1;
for ($i = 2; $i <= 5; $i++ ) {
    $factorial5 *= $i;
}
```

- Dos:

```
for ($factorial5 = 1, $i = 2; $i <= 5; $i++ ) {
    $factorial5 *= $i;
}
```

- Tres:

```
for ($factorial5=1, $i=2; $i<=5; $factorial5*=$i,
    $i++);
```


foreach

- Nuevo en PHP4:

```
foreach ($array as $variable) {  
    instrucciones  
}
```

- Recorre todos los elementos de un vector o colección
- En cada iteración, \$variable toma el valor de un elemento de \$array

Funciones (I)

```
function nomFuncion ($arg_1, $arg_2,  
    ..., $arg_n) {  
    instrucciones  
    return $valorSsalida;  
}
```

- Los parámetros se pasan por valor (copia local a la función)
- return es opcional y permite devolver un valor. Además, finaliza la ejecución de la función
- No permite sobrecarga de funciones

Funciones (II)

- Se pueden proporcionar valores por defecto a los parámetros, de derecha a izquierda:

```
function nomFuncion ($arg1, $arg2="valor") {...}
```

- Para pasar parámetros por referencia se tiene que añadir '&' al nombre del parámetro:

```
function nomFuncion ($arg1, &$arg2) {...}
```

Ejercicio PHP

```
<html><head><title>Curso de PHP - Actividad 1</title></head>
<body>
<?php
function actividad1($numero) {
    lista[] = 2;
    for (int i = 3; i <= numero; i++) {
        $es = TRUE
        foreach ($lista as $j) {
            if $i % $j = 0 then
                $es = FALSE;
                break;
        }
        if ($es)
            $lista[] = $i;
    }
    return $lista;
}
$resultado = actividad1(200);
foreach ($resultado as $r) {
    echo '$r<br>';
}
</body></html>
```

Ejercicio PHP: Solución

```
<html><head><title>Curso de PHP - Actividad 1</title></head>
<body>
<?php
function actividad1($numero) {
    $lista[] = 2;
    for ($i = 3; $i <= $numero; $i++) { // int
        $ses = TRUE;
        foreach ($lista as $j) {
            if ($i % $j == 0) { //then
                $ses = FALSE;
                break;
            }
        }
        if ($ses)
            $lista[] = $i;
    }
    return $lista;
}
$resultado = actividad1(200);
foreach ($resultado as $r) {
    echo "$r<br>";
}
?>
</body></html>
```

include y require

- `require(fichero)` e `include(fichero)`:
 - Insertan el contenido del fichero cada vez que se ejecutan
 - `require()`: si el fichero no existe, se produce un mensaje de error y finaliza la ejecución
 - `include()`: si el fichero no existe, se produce un mensaje de advertencia y continúa la ejecución
- `require_once()`, `include_once()`: para asegurarse que se incluye una sola vez (evita problemas de redefinición de funciones, reasignación de variables, etc.)

Manejo de ficheros (I)

- Muy similar a las funciones de C (`stdio.h`)

- Abrir:

```
fopen(nombre, modo)
```

nombre: local o remoto ("http://" o "ftp://")

modo: 'r' (sólo lectura), 'r+' (lectura/escritura), 'w' (sólo escritura), 'w+' (lectura/escritura), 'a' (sólo escritura, añadir al final), 'a+' (lectura/escritura, añadir al final)

Devuelve un identificador que se emplea en el resto de funciones (o FALSE en caso de error)

- Cerrar:

```
fclose(identificador)
```

Manejo de ficheros (II)

- Leer:

```
fgets(identificador)
```

```
fscanf(identificador, formato, var1, ...)
```

- Leer todo el contenido y almacenar línea a línea en un array:

```
file(nombre)
```

- Escribir:

```
fwrite(identificador, cadena)
```

- Final de fichero:

```
feof(identificador) (TRUE → fin, FALSE → no fin)
```

Manejo de ficheros (III)

- Ejemplo: leer el contenido de un fichero línea a línea:

```
<?
  $a = fopen('datos.txt', 'r');

  while(!feof($a))
    echo fgets($a) . '<br>';

  fclose($a);
?>
```

Manejo de ficheros (IV)

- Ejemplo: leer el contenido de un fichero de una vez

```
<?
  $a = file('datos.txt');

  foreach($a as $linea)
    echo $linea . '<br>';

?>
```

Manejo de ficheros (y V)

- Ejemplo: mostrar una página web de otro sitio web (las imágenes fallan si las URL son relativas)

```
<?
$a =
file('http://www.ua.es/index.html');

foreach($a as $linea)
    echo $linea;
?>
```

Sistema de ficheros (I)

- Muchas funciones para gestionar el sistema de ficheros del servidor
- Específicas de sistemas Unix → Algunas no tienen sentido en otros sistemas operativos:
 - Permisos
 - Modo y grupo
 - Crear enlaces (*soft* y *hard*)
 - ...

Sistema de ficheros (II)

- Copiar un fichero:
`copy(origen, destino)`
- Renombrar (o mover) un fichero:
`rename(origen, destino)`
- Borrar un fichero:
`unlink(fichero)`

Sistema de ficheros (III)

- Última modificación de un fichero:
`filemtime(fichero)`
- Tamaño de un fichero:
`filesize(fichero)`

Sistema de ficheros (IV)

- Cambiar de directorio:
`chdir(directorio)`
- Crear un directorio:
`mkdir(directorio)`
- Borrar un directorio:
`rmdir(directorio)`
- Apertura de un directorio (devuelve un identificador para su gestión):
`opendir(directorio)`
- Lectura de un directorio (devuelve el nombre del siguiente fichero en el directorio):
`readdir(identificador_directorio)`

Sistema de ficheros (y V)

- Mostrar los ficheros de un directorio con su tamaño:

```
<?
$dir = opendir('.');

while(($fichero = readdir($dir)) != FALSE)
    echo "$fichero: " . filesize($fichero) .
    '<br>';

closedir($dir);
?>
```


Funciones matemáticas (I)

- Constantes:
 - M_PI: 3.1415...
 - M_E: 2.7182...
 - M_LOG2E: 1.4426...
 - M_LOG10E: 0.4342...
 - M_LN2: 0.6931...
 - ...

Funciones matemáticas (II)

- Funciones:
 - abs: valor absoluto
 - cos, asin, atan, cos, sin, ...
 - ceil: mínimo entero mayor o igual
 - floor: máxima entero menor o igual
 - log, log10
 - max, min
 - rand, srand: números aleatorios enteros (desde 4.2.0 no hace falta inicializar con srand)
 - round: redondea
 - ...

Funciones matemáticas (y III)

- Ejemplo:

```
<?
echo M_PI . '<br>';
echo M_E . '<br>';

echo getrandmax() . '<br>';
echo rand() . '<br>';
echo rand(1, 10) . '<br>';
?>
```

Funciones de fechas (I)

- Para obtener la fecha y la hora en el servidor web, no en el cliente (navegador)
- Fecha y hora actual: `getdate()`
- Devuelve un array asociativo:
 - "seconds": segundos
 - "minutes": minutos
 - "hours": horas
 - "mday": día del mes
 - "wday": día de la semana (numérico: 0 Domingo hasta 6 Sábado)
 - "mon": mes (numérico)
 - "year": año
 - "yday": día del año (desde 0)
 - "weekday": día de la semana (texto)
 - "month": mes (texto)

Funciones de fechas (II)

- Valida una fecha:
`checkdate(mes, dia, año)`
- El instante actual medido en segundos desde la Época Unix (1 de Enero de 1970, 00:00:00 GMT):
`time()`

Funciones de fechas (y III)

- Ejemplo:

```
<?
$ahora = getdate();

echo 'día de la semana: ' . $ahora['weekday'] .
'<br>';
echo 'día del mes: ' . $ahora['mday'] . '<br>';
echo 'día del año: ' . $ahora['yday'] . '<br>';
echo 'mes (numérico): ' . $ahora['mon'] .
'<br>';
echo 'mes: ' . $ahora['month'] . '<br>';
echo 'año: ' . $ahora['year'] . '<br>';
?>
```

Orientación a objetos (I)

- No es un lenguaje OO puro porque presenta ciertas carencias (ocultación, herencia múltiple, polimorfismo), pero las características más empleadas sí que están disponibles (encapsulación, herencia simple, constructor, etc.)

Orientación a objetos (II)

- Declaración de una clase:

```
class NombreClase
{
    var $atributo1, $atributo2, ...;
    function NombreClase($arg1, $arg2, ...) {...}
    function Metodo1(...) {...}
    function Metodo2(...) {...}
    ...
}
```

- Los atributos se tienen que declarar explícitamente (único caso en PHP)
- Un único constructor por clase

Orientación a objetos (III)

- Creación de un objeto

```
$objeto = new NombreClase(...);
```

- Acceso a los miembros de una clase: ->

- Desde fuera: `$objeto->atributo(metodo)`

- Desde dentro: `$this->atributo(metodo)`

Orientación a objetos (IV)

```
<?
class Persona
{
    var $nombre, $apellidos;

    function Persona($n, $a)
    {
        $this->nombre = $n;
        $this->apellidos = $a;
    }
}

$p1 = new Persona('Juan', 'Gómez Gómez');
$p2 = new Persona('José', 'Pérez Pérez');

echo "$p1->apellidos, $p1->nombre<br>";
echo "$p2->apellidos, $p2->nombre<br>";
?>
```

Orientación a objetos (V)

- Se puede realizar herencia simple mediante `extends`
- Importante: el constructor de la clase base **NO** se invoca automáticamente
- Sintaxis:

```
class ClaseDerivada extends ClaseBase
{
    ...
}
```

Orientación a objetos (y VI)

```
<?
...
class Cliente extends Persona
{
    var $codigo;

    function Cliente($n, $a, $c)
    {
        $this->Persona($n, $a);
        $this->codigo = $c;
    }
}

$c1 = new Cliente('Luis', 'Jiménez Jiménez', 123);
echo "$c1->apellidos, $c1->nombre: $c1->codigo<br>";
?>
```

Variables predefinidas (I)

- Existen una serie de variables globales predefinidas
- Son de tipo `superglobal`: se pueden emplear en cualquier contexto sin tener que declararlas con `global`
- Son arrays asociativos (clave, valor)
- Su contenido depende del modo de ejecución y del sistema operativo

Variables predefinidas (II)

- Variables predefinidas:
 - `$GLOBALS`: acceso a variables globales
 - `$_COOKIE`: *cookies*
 - `$_ENV`: variables de entorno
 - `$_FILES`: envío de ficheros mediante HTTP
 - `$_GET`: datos mediante HTTP GET
 - `$_POST`: datos mediante HTTP POST
 - `$_REQUEST`: `$_GET`, `$_POST` y `$_COOKIE`
 - `$_SERVER`: variables del servidor web
 - `$_SESSION`: variables de sesión

Variables predefinidas (III)

- `$_ENV`: variables de entorno del sistema operativo
- Ejemplo:
 - HOME
 - PATH
 - PWD
 - USER
 - ...

Variables predefinidas (IV)

- `$_SERVER`: variables definidas por el servidor web
- Podemos encontrar las variables de CGI
- Ejemplo:
 - HTTP_ACCEPT
 - PATH
 - REMOTE_ADDR
 - REMOTE_PORT
 - SERVER_NAME
 - SERVER_PORT
 - ...

Variables predefinidas (y V)

- Ejemplo:

```
<?
echo "$_SERVER['REMOTE_ADDR']";
?>
```

- Como obtener la clave y el valor de cada elemento:

```
<?
reset($_SERVER);
while((list($clave, $valor) = each($_SERVER)) !=
FALSE)
    echo "$clave => $valor<br>";
?>
```

Manejo de formularios (I)

- Acceso a través de unos arrays globales:
 - \$_GET: parámetros enviados mediante GET o en la URL
 - \$_POST: parámetros enviados mediante POST
 - \$_REQUEST: la unión de \$_GET y \$_POST
- Existe la opción de activar `register_globals` en `php.ini` (crea una variable global para cada parámetro recibido), pero se desaconseja por razones de seguridad

Manejo de formularios (II)

- Formulario:

```
<form action="respuesta.php?valor=10"
  method="post">
<input type="text" name="nombre">
</form>
```

- PHP (¿todos correctos?):

```
<?
echo 'valor: ' . $_GET['valor'] . '<br>';
echo 'valor: ' . $_POST['valor'] . '<br>';
echo 'valor: ' . $_REQUEST['valor'] . '<br>';
echo 'nombre: ' . $_GET['nombre'] . '<br>';
echo 'nombre: ' . $_POST['nombre'] . '<br>';
echo 'nombre: ' . $_REQUEST['nombre'] . '<br>';
?>
```

Manejo de formularios (III)

- Formulario:

```
<form action="respuesta.php?valor=10"
  method="post">
<input type="text" name="nombre">
</form>
```

- PHP (¿todos correctos?):

```
<?
echo 'valor: ' . $_GET['valor'] . '<br>';
echo 'valor: ' . $_POST['valor'] . '<br>'; → Vacío
echo 'valor: ' . $_REQUEST['valor'] . '<br>';
echo 'nombre: ' . $_GET['nombre'] . '<br>'; → Vacío
echo 'nombre: ' . $_POST['nombre'] . '<br>';
echo 'nombre: ' . $_REQUEST['nombre'] . '<br>';
?>
```

Manejo de formularios (IV)

- Valores vectoriales desde un formulario:
 - Listas de selección múltiple, botones de comprobación con el mismo nombre
 - Si no se indica nada, sólo se tiene acceso a un valor
 - En el código HTML hay que añadir “[]” al nombre del control
 - Devuelve un array, con `count ()` podemos conocer su tamaño

Manejo de formularios (y V)

- **Formulario:**

```
<select name="lista[]" multiple>
<option>Alicante</option>
<option>Valencia</option>
<option>Castellón</option>
</select>
```

- **PHP:**

```
<?
    $lista = $_POST['lista'];
    for($i = 0; $i < count($lista); $i++)
        echo "$lista[$i]<br>";
?>
```

Programación en Internet – Curso 2006-2007

Variable predefinida \$_FILE

- Se emplea para manejar los ficheros subidos desde el cliente al servidor
- Se tiene que modificar el formulario:
 - enctype="multipart/form-data"
- \$_FILES['fichero']['name']: nombre original del fichero en el cliente
- \$_FILES['fichero']['tmp_name']: nombre del fichero temporal que se genera para guardar el fichero subido
- \$_FILES['fichero']['size']: tamaño en bytes del fichero
- \$_FILES['fichero']['type']: tipo MIME (image/gif), siempre que el navegador lo proporcione
- \$_FILES['fichero']['error']: código de error asociado con la subida del fichero
- php.ini: parámetro upload_tmp_dir y upload_max_filesize

Programación en Internet – Curso 2006-2007

Ejemplo subir ficheros (I)

```
<html><body>
Insercción de la fotografía del usuario:
<form action="foto_ins_r.php" method="post"
enctype="multipart/form-data">
<?
echo '<table>';
echo
  '<tr><td>Título</td><td>Text</td><td>Fichero</td></tr>';
echo '<tr valign="top">';
echo "<td><input type='text' name='titulo'></td>";
echo "<td><textarea name='text' cols='40'
  rows='5'></textarea></td>";
echo "<td><input type='file' name='fichero'></td>";
echo '</tr>';
echo '</table>';
?>
<input type="submit" value="Enviar">
</form></body></html>
```

Ejemplo subir ficheros (II)

```
<html><body>
<?
echo " name:      " . $_FILES['fichero']['name'] . "\n";
echo " tmp_name:  " . $_FILES['fichero']['tmp_name'] . "\n";
echo " size:      " . $_FILES['fichero']['size'] . "\n";
echo " type:      " . $_FILES['fichero']['type'] . "\n";

Define ("PICDIR", "d:\\pipo\\");
$uploadfile = PICDIR . $_FILES['fichero']['name'];
echo $_FILES['fichero']['name'];

if(move_uploaded_file($_FILES['fichero']['tmp_name'],
    $uploadfile))
    echo ' ok<br>\n';
else
    echo ' error<br>\n';
?>
</body></html>
```

Variable predefinida \$_FILE

- Precauciones:
 - Permisos de escritura en el directorio temporal
 - Permisos de escritura en el directorio de destino
 - Atención con los ficheros que puedan subir los usuarios
 - Troyanos, scripts, ejecutables, etc.
 - Atención con los usuarios de otras plataformas:

Macintosh HD:Users:Marieta:Desktop:¡Documentos míos!:¿Documentos de ayer?.doc

Cabeceras HTTP

- Función para enviar cabeceras HTTP:

```
header("cabecera: valor");
```

- Ejemplos:

```
header("location: http://www.ua.es");
```

```
header("HTTP/1.0 404 Not Found");
```

```
header("Pragma: no-cache");
```

- Otras: cache-control, expires, last-modified, etc.

Variables de aplicación (I)

- Variables accesibles desde todos los usuarios y todas las páginas de un mismo sitio web en un mismo servidor
- PHP no soporta variables de aplicación
- El programador tiene que implementar su propia solución y dar respuesta a diversos problemas:
 - Caducidad de una variable
 - Portabilidad de los sitios web
 - Disminución o aumento del rendimiento
 - Condiciones de carrera (accesos concurrentes)

Variables de aplicación (y II)

- Soluciones más empleadas:
 - Memoria compartida
 - Con funciones propias de las librerías de PHP
 - Invocando programas externos a PHP que se ejecutan en el servidor
 - Con ficheros alojados en el servidor
 - Un fichero para cada variable
 - Todas las variables en un único fichero
 - Con una base de datos

Gestión de sesiones (I)

- Las sesiones permiten almacenar información particular a cada visitante
- Se pueden gestionar las sesiones de dos formas:
 - Mediante cookies → Método por defecto
 - Mediante paso de un identificador (*session ID*) a través de la URL: PHP automáticamente añade el identificador en todos los enlaces relativos de las páginas → Hay que configurar PHP

Gestión de sesiones (II)

- `session_start()` inicia la gestión de sesiones
- Se tiene que incluir AL PRINCIPIO (antes de escribir en la página nada porque envía encabezados HTTP) de todas las páginas donde se quiera disponer de la sesión
- `session_register('variable')` registra una variable para que se almacene en la sesión
- `session_id()` devuelve el identificador de sesión actual

Gestión de sesiones (III)

- `session_unregister('variable')` finaliza el registro de una variable en la sesión
- `session_is_registered('variable')` devuelve TRUE si la variable ha sido registrada en la sesión actual
- `session_destroy()` finaliza la sesión actual (se borran todos los datos registrados)

Gestión de sesiones (IV)

- Página 1:

```
<?
  session_start();
  session_register('contador');
  $contador++;
?>
<html> ... </html>
```
- Página 2:

```
<? session_start(); ?>
<html>
<body>
<?
  echo "contador: $contador";
?>
</body>
</html>
```

Gestión de sesiones (V)

- Otra forma: variable predefinida global
\$_SESSION
- No hace falta usar:
 - session_register()
 - session_unregister()
 - session_is_registered()
- Podemos usar las funciones de arrays
(count, foreach, etc.) para consultar las
variables de sesión

Gestión de sesiones (y VI)

- Página 1:

```
<?
  session_start();
  $_SESSION['contador']++;
?>
<html> ... </html>
```

- Página 2:

```
<? session_start(); ?>
<html>
<body>
<?
  echo "contador: " . $_SESSION['contador'];
?>
</body>
</html>
```

Acceso a bases de datos (I)

- Funciones específicas para cada SGBD
- Formato funciones:
nombreSGBD_nombreFuncion()
- Soporta más de 20 SGBD (pero para algunos necesita las librerías de cliente del fabricante):
 - Informix, InterBase
 - Microsoft SQL Server, mSQL, MySQL,
 - Oracle
 - PostgreSQL
 - Sybase
 - ...

Acceso a bases de datos (y II)

- Problema: si se emplean las funciones específicas, el código no será portable
- Solución: añadir una capa intermedia (como ODBC o soluciones de terceros), pero se pierde rendimiento y posibilidad de emplear características específicas del SGBD
- MySQL es la base de datos que mejor se integra con PHP

Acceso a BD con MySQL (I)

- Abrir una conexión:
`mysql_connect(servidorBD, usuario, contraseña)`
- Devuelve un identificador que se emplea en todas las funciones (o FALSE en caso de error)
- Cerrar una conexión:
`mysql_close(identificador)`

Acceso a BD con MySQL (II)

- Verifica que la conexión funciona
`mysql_ping(identificador)`
- Selecciona una base de datos:
`mysql_select_db(nombreBD,
identificador)`
- Ambas funciones devuelven TRUE en caso de éxito y FALSE en caso contrario

Acceso a BD con MySQL (III)

- Ejecutar una sentencia SQL:
`mysql_query(sentencia, identificador)`
- Devuelve un resultado (SELECT, SHOW, EXPLAIN o DESCRIBE, ...) o TRUE (INSERT, UPDATE, DELETE, ...) si todo es correcto o FALSE en caso contrario

Acceso a BD con MySQL (IV)

- Para obtener el número de tuplas (filas) afectadas durante la última operación:

- Si fue INSERT, UPDATE, ..., la última operación:

```
mysql_affected_rows(identificador)
```

- Si fue SELECT la última operación:

```
mysql_num_rows(resultado)
```

Acceso a BD con MySQL (V)

- Para recorrer un resultado:

```
mysql_fetch_array(resultado)
```

- Devuelve un array que representa una fila (registro) o FALSE en caso de error (por ejemplo, llegar al final del resultado)
- Al array se puede acceder de forma numérica (posición de la columna) o asociativa (nombre de la columna)

Acceso a BD con MySQL (y VI)

- Ejemplo:

```
<?
if(!($siden = mysql_connect("127.0.0.1", "usuario", "clave")))
    die("Error: No se pudo conectar");

if(!mysql_select_db("bd", $siden))
    die("Error: No existe la base de datos");

$sentencia = "SELECT * FROM Alumnos";
$alumnos = mysql_query($sentencia, $siden);
if(!$alumnos)
    die("Error: no se pudo realizar la consulta");

while($fila = mysql_fetch_array($alumnos))
{
    echo $fila['Apellidos'] . ', ' . $fila['Nombre'] . '<br>';
}
mysql_close($siden);
?>
```

Acceso a BD con ODBC (I)

- Abrir una conexión:

```
odbc_connect(nomDSN, usuario, contraseña)
```

Devuelve un identificador que se emplea en todas las funciones (o FALSE en caso de error)

- Cerrar una conexión:

```
odbc_close (identificador)
```

```
odbc_close_all()
```

Acceso a BD con ODBC (II)

- Ejecutar una sentencia SQL:

```
odbc_exec(identificador, sentencia)
```

Devuelve un resultado (SELECT, SHOW, EXPLAIN o DESCRIBE, ...) o TRUE (INSERT, UPDATE, DELETE, ...) si todo es correcto o FALSE en caso contrario

Acceso a BD con ODBC (III)

- Para obtener el número de filas (tuplas, registros) afectadas durante la última transacción:

```
odbc_num_rows(resultados)
```

- Para obtener el número de columnas de un resultado:

```
odbc_num_fields(resultados)
```

- Para recorrer un resultado:

```
Array = odbc_fetch_array(resultados[, num_fila])
```

Devuelve un array asociativo con el contenido de la siguiente fila o FALSE si no hay más filas en el resultado

Acceso a BD con ODBC (IV)

- Para recorrer un resultado (otra función):

```
odbc_fetch_row(resultados[, num_fila])
```

Devuelve TRUE si se puede avanzar a la siguiente fila o FALSE si no hay más filas en el resultado

- Para acceder a los datos de la fila actual del array resultado:

```
odbc_result(resultados, num)
```

```
odbc_result(resultados, 'columna')
```

- Al array se puede acceder de forma numérica (posición de la columna) o asociativa (nombre de la columna)

Acceso a BD con ODBC (V)

- Ejemplo:

```
<?
if(!($siden = odbc_connect("dsnPI", "usuario", "clave")))
    die("Error: No se ha podido conectar");

$sentencia = "SELECT * FROM Alumnos";
$alumnos = odbc_exec($siden, $sentencia);
if(!$alumnos)
    die("Error: no se ha podido realizar la consulta");

while($fila = odbc_fetch_array($alumnos))
{
    echo $fila['Apellidos'] . ', ' . $fila['Nombre'] . '<br>';
}
odbc_close($siden);
?>
```


Acceso a BD con ODBC (y VI)

- Otras funciones:
 - `odbc_commit`, `odbc_rollback`, para el control de transacciones
 - `odbc_data_source`, para obtener información sobre el origen de datos
 - `odbc_error`, `odbc_errormsg`, para obtener información del último error ocurrido
 - `odbc_result_all`, imprime el resultado (todas las filas) como una tabla HTML

Depuración (I)

- Algunas funciones que ayudan al proceso de depuración
- `gettype()`: devuelve una cadena que representa el tipo de una variable → `boolean`, `integer`, `double`, `string`, `array`, etc.
- Mejor usar (más rápido y compatible con versiones futuras): `is_array()`, `is_bool()`, `is_float()`, `is_integer()`, etc.

Depuración (II)

- `var_dump()`: muestra información sobre una variable

- Ejemplo:

```
$b = 3.1;
```

```
$c = true;
```

```
var_dump($b, $c);
```

Genera:

```
float(3.1)
```

```
bool(true)
```

Depuración (III)

- `print_r(var[, devolver])`: muestra información sobre una variable (similar a `var_dump`)
- Si `devolver` vale `TRUE`, en vez de mostrarse el resultado se devuelve

Depuración (y IV)

- `$php_errormsg`: variable predefinida que contiene el último mensaje de error generado por PHP
 - Sólo está disponible en el ámbito donde se produce el error
 - La opción `track_errors` de `php.ini` tiene que estar activa

Otras funcionalidades (I)

- PHP tiene una amplia librería de funciones que le proporcionan una gran potencia
- Algunas necesitan la instalación (compilación) de productos de terceros y no están disponibles para todas las plataformas

Otras funcionalidades (II)

- Manejo de ficheros comprimidos:
 - Bzip2 (.bz2)
 - Zip (.zip)
 - Zlib (.gz)
- Calendarios: permite convertir fechas entre distintos calendarios (Gregoriano, Judío, Juliano, etc.)
- CURL (Client URL Library): convierte la página en un cliente de Internet. Permite realizar peticiones HTTP, FTP, con soporte de cookies, etc.
- Gestión de correo (IMAP, POP3 y SMTP) y noticias (NNTP)

Otras funcionalidades (III)

- Envío de correo:
 - `bool mail (string to, string subject, string message [, string additional_headers [, string additional_parameters]])`
- Ejemplo:

```
$para = "marieta@alu.ua.es"
$titulo = "Un mensaje de prueba"
$mensaje = 'Este es un mensaje de prueba.
Tiene que llegar sin problemas.
PHP hace de todo.';
$encabezados = "From: webmaster@alu.ua.es\r\n" .
"Reply-To: webmaster@alu.ua.es\r\n";
$correcto = mail($para, $titulo, $mensaje,
$encabezados);
```
- Nota: hay que configurar el correo electrónico en el servidor

Otras funcionalidades (y IV)

- Creación y manejo de imágenes
- Creación de recursos multimedia en Flash
- Manejo de documentos XML (con DOM y XSLT)
- Creación de documentos PDF
- ...

Bibliografía

- Professional PHP4, Argerich, L. Et Al., WROX Pres 2002.
- Introducción a PHP 4. Curso interno del Laboratorio Multimedia de la Universitat de Alicante. Vicente Aguilar. Septiembre 2000
- PHP: Hypertext Preprocessor. Página principal del PHP. <http://www.php.net>
- PHP Annotated Manual. Manual de PHP. <http://www.php.net/manual/>
- Zend / Where PHP meets eBusiness. Empresa de los creadores del motor de PHP (Zend). <http://www.zend.com>