Introducción a PHP



http://creativecommons.org/licenses/by-sa/2.0/

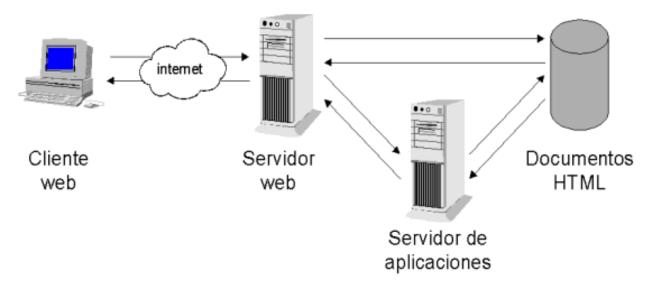
Mayo 2005

Jordi Llonch <jordi@laigu.net>

Basado en:
MMLab: Curso de PHP
Por Vicente Aguilar
y
Sesiones en PHP por Sergio Sáez Barona

1. INTRODUCCIÓN: Lenguajes script lado servidor

- PHP es un lenguaje de script del lado del servidor. Otros lenguajes similares son ASP, JSP o ColdFusion
- Los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta y ejecuta antes de servir las páginas al cliente
- El cliente no ve el código PHP sino los resultados que produce



Características de PHP

- Rápido
- Lenguaje fácil de aprender y potente
- Integración perfecta con diversos servidores
- Acceso a diversos tipos de Bases de Datos
- Diseño modular de fácil ampliación
- Licencia abierta

Historia y Desarrolladores

Fechas

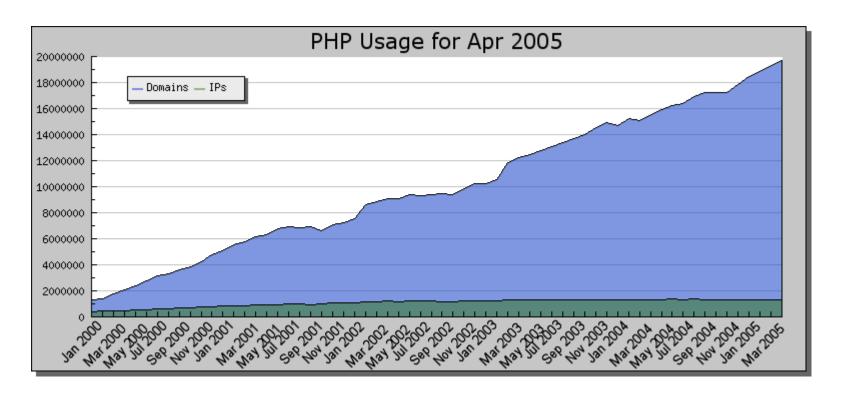
- •Inicio del desarrollo en otoño de 1994
- •PHP Versión 1 en primavera 1995
- PHP Versión 2 1995-1997
- •PHP Versión 3 1997-2000
- •PHP Versión 4 en el segundo trimestre de 2000
- PHP Versión 5 2004

•

- •Equipo de Desarrollo (+ de 200 personas con acceso al CVS)
 - Zeev Suraski y Andi Gutmans (Israel)
 - Shane Caraveo (Florida)
 - •Stig Bakken (Norway)
 - Andrei Zmievski (Lincoln, Nebraska)
 - Sascha Schumann (Dortmund, Germany)
 - Thies C. Arntzen (Hamburg, Germany)
 - •Jim Winstead (Los Angeles)
 - •Sam Ruby (Raleigh, NC)
 - Rasmus Lerdorf (San Francisco)

Estadísticas de uso

- Netcraft Statistics
 - PHP: 19,720,597 Domains, 1,310,181 IP Addresses



Plataformas soportadas

•Plataformas (actualidad):

- UNIX (todas las variantes)
- •Win32 (NT/W95/W98/W20
- •QNX
- Mac (WebTen)
- •OS/2
- BeOS

•

•Servidores:

- Apache (UNIX,Win32)
- Apache 2.0
- •CGI
- fhttpd
- •ISAPI (IIS, Zeus)
- NSAPI (Netscape iPlanet)
- •Java servlet
- AOLServer
- Roxen

•Plataformas (en preparación):

- •OS/390
- •AS/400

•

Servidores (en preparación):

- •WSAPI (O'Reilly WebSite)
- phttpd
- thttpd

•

Bases de datos soportadas

•SQL

- Otros
- Adabas D
- Empress
- •IBM DB2
- •Informix
- Ingres
- Interbase
- Frontbase
- •mSQL
- Direct MS-SQL
- MySQL
- •ODBC
- Oracle (OCI7,OCI8)
- PostgreSQL
- Raima Velocis
- Solid
- Sybase
- . . .

- dBase
- •filePro (sólo lectura)
- •dbm (ndbm, gdbm, Berkeley db)
- . . .

2. El lenguaje PHP

2.1. Extensión de los ficheros

- .php3 Indica código PHP 3.x.
- .php4 Indica código PHP 4.x.
- php Indica código PHP. Preferiremos esta extensión por ser más genérica.
- > .phtml Actualmente en desuso.

2.2. Delimitadores

```
<? echo 'Primer método de delimitar código PHP'; ?>
<?php echo 'Segundo método, el más usado'; ?>
<script language="php">
echo 'Algunos editores (como el FrontPage) Sólo entien
</script>
<% echo 'Método de compatibilidad con ASP'; %>
<?= expresión ?> equivale a <? echo expresión ?>
```

2.2. Delimitadores. Ejemplo.

```
<html>
<body>
<?php
if ( Hour(time)>20 || Hour(time)<4)</pre>
{
    echo "Buenas noches.";
else
    echo "Buenos días.";
</body>
</html>
```

2.3. Fin de línea

2.4. Comentarios

```
/* Comentarios estilo C.
  * Pueden extenderse durante varias líneas.
  */
// Comentarios estilo C++. Hasta fin de línea.
# Comentarios estilo Perl. Hasta fin de línea.
```

2.5.1. Variables. Declaración y Uso.

- NO hace falta declararlas
- Llevan delante el signo del dólar '\$'.

```
$var_1 = 123;
$var_2 = 'hola';
$var_3 = $var_1 * 2;
$var_4 = "$var_2 mundo";
```

2.5.2. Variables. Tipado. Tipos de datos

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

2.5.2. Variables. Tipado. Funciones de interés

- 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:

 - La función var_dump() muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays
 - La función print_r() también muestra cualquier tipo de variable. Interesante con los arrays

2.5.2. Variables. Tipado.

Variables débilmente tipadas (tipo mixed).

2.5.2. Variables. Tipado. Conversión automática.

PHP realiza conversiones automáticas de tipo:

```
$mivar = 123;
echo $mivar; // Se convierte a string
```

```
$mivar = '3';  // Se convierte a entero
$mivar = 2 + $mivar; // para realizar la suma
```

2.5.2. Variables. Tipado. Conversión explícita.

Operador cast:

```
$mivar = (string)123;
```

Cambiar el tipo de una variable:

```
$mivar = 12;
settype($mivar, "double");
```

2.5.3. Variables. Ámbito.

- En el cuerpo de un fichero, las variables son GLOBALES al fichero y ficheros incluidos.
- En una función, son LOCALES a esa función.
- Dentro de una clase, sólo pueden ser accedidas a través del operador "->" sobre el nombre del objeto.

2.5.4. Referencias.

Se definen con el carácter '&':

```
$alias = &$variable
```

Se puede eliminar una referencia con la función *unset()*:

```
$a = 1;
$b = &$a;
unset ($a); // Pero $b sigue valiendo 1
```

2.6. Tipos de datos.

```
    Enteros, en decimal, octal o hexadecimal.

   MiVar = 123;

    Punto flotante.

   MiVar = 1.3e4;
Arrays.
   MiVar[2] = 123;
Strings.
   $MiVar = "Cadena de texto\n";
Objetos:
   $MiVar = new MiClase();
```

2.6.3. Tipos de datos. Arrays.

```
$MiArray[0] = 1;
$MiArray[1] = "hola!!";
$MiArray[] = 3;
echo $MiArray[2]; // 3
```

2.6.3. Tipos de datos. Arrays (2).

Funcionan como vectores o tablas hash al mismo tiempo:

```
$MiArray["nombre"] = "Homer";
echo $MiArray[0];  // (sin valor)
echo $MiArray["nombre"]; // "Homer"
```

Y pueden tener más de una dimensión:

```
$MiOtroArray[1]["pepe"][4] = "3 dimensiones!";
```

2.6.3. Tipos de datos. Arrays (3).

También se pueden definir con el constructor array():

```
$0troArrayMas = array( 1, "hola", 5);

$Y0troArray = array(
    0 => 1,
    1 => "hola",
    2 => 5,
    3 => 8,
    "nombre" => "Homer"
);
```

2.6.4. Tipos de datos. Strings. Comillas dobles.

 Si se delimitan entre comillas dobles ("), se expandirá cualquier variable que haya dentro de la cadena. Además, se pueden incluir ciertas secuencias de escape, al igual que en C:

Secuencia	Significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
11	Barra invertida
\\$	Símbolo del dólar
\"	Dobles comillas
\0[0-7]	Carácter en octal
\0[xX][0-9a-fA-F]	Carácter en hexadecimal

2.6.4. Tipos de datos. Strings. Comillas dobles.

- 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>

2.6.4. Tipos de datos. Strings. Comillas dobles.

- 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>
```

2.6.4. Tipos de datos. Strings (2). Comillas simples.

 Si se delimitan entre comillas simples ('), las variables no se expanden y además las únicas secuencias de escape que se reconocen son "\\" y "\\" (barra invertida y comillas simples.)

2.6.4. Tipos de datos. Strings (3). "Here Doc"

Utilizando la sintaxis "here doc" de Perl. Las variab

```
$cadena = "Esto es un ejemplo de cadena como \"here
La variable \$a vale $a.
Ahora vamos a finalizar la cadena:
"
```

2.6.4. Tipos de datos. Strings (4).

Para concatenar cadenas se utiliza el operador '.':

```
$cad = 'A esta cadena ';
$cad = $cad . 'le vamos a añadir más texto.';
```

Se puede acceder a cada carácter como si fuera un array:

```
$cad2 = "Tercer carácter de \$cad : '$cad[2]'";
```

2.7. Constantes.

Las constantes se definen con la función define():

```
define("SALUDO", "Hola, mundo!");
echo "La constante SALUDO vale " . SALUDO;
```

ncian de las variables en que:

<mark>lólar d</mark>elante.

<mark>desde</mark> cualquier parte del código donde han sido definidas, sin restriccio s o borradas una vez definidas.

ores escalares, no vectores.

2.8. Mayúsculas y minúsculas.

rtamiento mixto en variables y funciones:

- En las <u>variables</u>, las mayúsculas y minúsculas IMPORTAN.
- <mark>En los</mark> nombres de <u>funciones y palabras reservadas</u>, las mayúsculas NC

2.9.1. Operadores aritméticos.

Operación	Nombre	Resultado
\$a + \$b	Suma	Suma de \$a y \$b.
\$a - \$b	Resta	Diferencia entre \$a y \$b.
\$a * \$b	Multiplicación	Producto de \$a y \$b.
\$a / \$b	División	Cociente de \$a y \$b.
\$a % \$b	Módulo	Resto de la operación \$a/\$b.

2.9.2. Auto-incremento y auto-decremento.

Operación	Nombre	Resultado
++\$a	Pre-incremento	Incrementa \$a en 1, y devuelve \$a (incrementado).
\$a++	Post-incremento	Devuelve \$a, y después lo incrementa en 1.
\$a	Pre-decremento	Decrementa \$a en 1, y después lo devuelve.
\$a	Post-decremento	Devuelve \$a, y después lo incrementa en 1.

2.9.3. Operadores de bits.

Operación	Nombre	Resultado
\$a & \$b	Y	Se ponen a 1 los bits que están a 1 en \$a y \$b.
\$a \$b	0	Se ponen a 1 los bits que están a 1 en \$a o \$b.
\$a ^ \$b	O Exclusivo	Se ponen a 1 los bits que están a 1 en \$a o \$b, pero no en ambos.
~ \$a	No	Se invierten los bits (se cambian 1 por 0 y viceversa.)
\$a << \$b	Desp. Izq.	Desplaza \$b posiciones a la izquierda todos los bits de \$a.
\$a >> \$b	Desp. Drch.	Desplaza \$b posiciones a la derecha todos los bits de \$a.

2.9.4. Operadores lógicos.

Operación	Nombre	Resultado
\$a and \$b	Υ	Cierto si \$a y \$b son ciertos.
\$a or \$b	0	Cierto si \$a o \$b es cierto.
\$a xor \$b	O Exclusivo.	Cierto si \$a o \$b es cierto, pero no ambos.
! \$a	No	Cierto si \$a es falso.
\$a && \$b	Υ	Cierto si \$a y \$b son ciertos.
\$a \$b	0	Cierto si \$a o \$b es cierto.

2.9.5. Operadores. Asignación, igualdad e identidad.

Operación	Nombre	Resultado
\$a = \$b	Asignación	Asigna el valor de una variable o expresión del segundo término a la variable del primer término.
\$a == \$b	141121424	Compara si el valor de los dos operandos es el mismo.
\$a === \$b		Compara si el valor es el mismo y, además, el tipo coincide.

2.9.5. Operadores. Asignación, igualdad e identidad. Ejemplo.

2.9.5. Operadores. Asignación, igualdad e identidad. Error.

```
$var1 = 1;
$var2 = 2;
if( $var1 = $var2 )
{
    echo 'iguales';
}
else
{
    echo 'distintas';
}
```

2.9.6. Comparaciones.

Operación	Nombre	Resultado
\$a != \$b	No igual	Cierto si el valor de \$a no es igual al de \$b.
\$a !== \$b	No idéntico	Cierto si \$a no es igual a \$b, o si no tienen el mismo tipo.
\$a < \$b	Menor que	Cierto si \$a es estrictamente menor que \$b.
\$a > \$b	Mayor que	Cierto si \$a es estrictamente mayor que \$b.
\$a <= \$b	Menor o igual que	Cierto si \$a es menor o igual que \$b.
\$a >= \$b	Mayor o igual que	Cierto si \$a es mayor o igual que \$b.

2.9.7. Operadores de cadenas.

```
$a = 1;
$b = 2;
$c = 'El resultado de ' . $a . ' + ' . $b . ' es ' . ($a + $b);
```

2.9.8. Operadores de control de error

```
@. Antepuesto a una expresión, evita cualquier mensaje de
```

```
@$handle = fopen ("/home/rasmus/file.txt", "r");
```

2.9.9. Atajos en la asignación.

2.9.10. Precedencia.

```
or

xor

and

print

= += -= *= /= .= % = &= |

= ^= ~= <<= >>=

?:

||

&&
|
```

```
&
==!===!==
< <= >>=
< <>>>
+ - .
* / %
! ~ ++ -- (int) (double)
(string) (array) (object) @
[
new
```

2.10.1. Estructuras de control. if ... elseif ... else

```
if (expresión)
{
    comandos
}
```

```
if (expresión)
{
    comandos_cierto
}
else
{
    comandos_falso
}
```

2.10.1. Estructuras de control. *if ... elseif ... else* (2)

```
if (expresion1)
    comandos1
elseif (expresion2)
    comandos2
elseif (expresion3)
    comandos3
else
    comandosElse
```

2.10.2. while y do ... while

```
while (expresión)
{
    comandos
}
```

```
do
{
    comandos
}
while (expresión);
```

2.10.4. for

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

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

2.10.4. for (2)

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

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

2.10.5. foreach

```
foreach (array as variable)
{
    comandos
}
```

```
$a = array (1, 2, 3, 17);
foreach ($a as $v)
{
    print "Valor actual de \$a: \$v.\n";
}

// Valor actual de \$a: 1
// Valor actual de \$a: 2
// Valor actual de \$a: 3
// Valor actual de \$a: 17
```

2.10.5. foreach (2)

```
foreach (array as indice => variable)
{
    comandos
}
```

```
$a = array ("a"=>1,"b"=>2,"c"=>3,"d"=>17);
foreach ($a as $k => $v)
{
    print "Valor actual de \$a[$k]: $v.\n";
}

// Valor actual de $a[a]: 1
// Valor actual de $a[b]: 2
// Valor actual de $a[c]: 3
// Valor actual de $a[d]: 17
```

2.10.6. switch

```
switch (variable)
    case valor1:
     comandos1
    case valor2:
        comandos2
    case valorN:
        comandosN
    default:
        comandosDefault
```

2.10.6. switch (2)

```
switch ($i)
{
    case 1:
        echo "Código del 1";
    case 2:
        echo "Código del 2";
    case 3:
        echo "Código del 3";
        break;
    case 4:
        echo "Código del 4";
```

2.11. Cierto o falso. Valores numéricos.

```
$x = 1; // $x
if( $x ) // se evalúa a cierto

$x = 0; // $x definida como el entero 0
if( $x ) // se evalúa a falso
```

2.11. Cierto o falso. Strings.

2.11. Cierto o falso. Arrays.

2.11. Cierto o falso. Objetos.

```
Class Yod {} // clase vacía
$x = new Yod();
if( $x ) // se evalúa a falso

Class Yod { // clase no vacía
   var $x = 1;
}
$x = new Yod();
if( $x ) // se evalúa a cierto
```

2.11. Cierto o falso. Constantes.

•

•TRUE es el valor entero decimal 1.

•

• FALSE es la cadena vacía.

2.12. Funciones.

```
function nombre ($arg_1, $arg_2, ..., $arg_n)
{
   comandos
   return $salida;
}
```

2.12. Funciones. (2) Ejemplo.

```
function factorial ($valor) {
    if ($valor < 0) {
        return -1; // Error
    if ($valor == 0 ) {
        return 1;
    if ($valor == 1 || $valor == 2) {
        return $valor;
    ret = 1;
    for ($i = 2; $i <= $valor; $i++) {
        $ret = $ret * $i;
    return $ret;
}
$factorial5 = factorial(5);
```

2.12. Funciones. (3) Valores por defecto.

```
function enlace($url = "www.php.net")
{
    echo '<a href="' . $url . '">Pulsa aquí</a>';
}
```

2.12.1. Funciones. Argumentos por referencia.

```
function MiFuncion(&$var)
{
     $var++;
}

$a = 5;
MiFuncion($a);
// Aquí $a == 6
```

2.12.2. Funciones. Devolución por referencia.

```
function &buscar_cliente($nombre)
{
    // ... buscamos ...
    return $registro;
}
$cliente = buscar_cliente("Juan");
echo $cliente->dni;
```

2.13. include y require

- Inclusión de ficheros externos:
 - include()
 - require()
- Ambos incluyen y evalúan el fichero especificado
- Diferencia:
 - Comportamiento
 - include() se incluye el fichero si se llega a esta instruc
 - require() el parser incluye directament el fichero antes
 - En caso de error
 - include() produce un warning
 - require() un error fatal
- Se usará require() si al producirse un error debe interrumpirse la carga de la j

2.13. include y require (2)

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

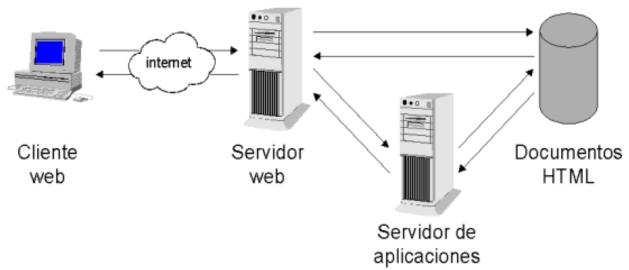
Ejercicios

Parte 1

3. Programando en PHP

3.1.1. Formularios

Desde PHP se puede acceder fácilmente a los datos



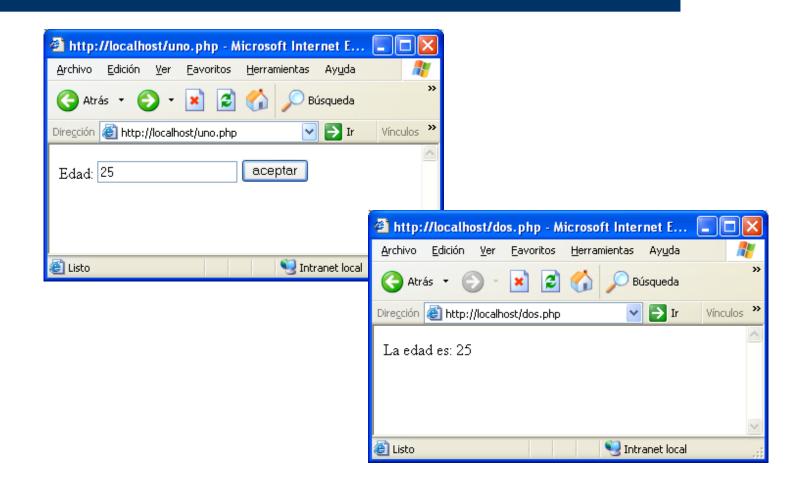
Veámoslo con un ejemplo simple

3.1.1. Formularios. Acceso a datos.

Fichero uno.php <HTML> < < BODY > <FORM ACTION="dos.php" METHOD="POST">
 Edad: <INPUT TYPE="text" NAME="edad"> <INPUT TYPE="submit" VALUE="aceptar"> - </FORM> </BODY> - </HTML> Fichero dos.php

```
<ht>HTML>
  <BODY>
- <?PHP</p>
      print ("La edad es: $edad");
 ?>
</BODY>
- </HTML>
```

3.1.1. Formularios. Acceso a datos. (2)



3.1.1. Formularios. Acceso a datos. (3)

- A partir de PHP 4.2.0, el valor por defecto de la directiva de PHP re
- Esto tiene una gran importancia sobre los formularios, ya que no es
- Se puede poner register_globals = on en el fichero de configuració
 - \$edad = \$_REQUEST['edad'];

3.1.1. Formularios. Acceso a datos. (4)

Fichero uno.php <HTML> <BODY> <FORM ACTION="dos.php" METHOD="POST">
 Edad: <INPUT TYPE="text" NAME="edad"> <INPUT TYPE="submit" VALUE="aceptar"> - </FORM> </BODY> - </HTML>

Fichero dos.php

```
<HTML>
  <BODY>
 <?PHP
      $edad = $_REQUEST['edad'];
print ("La edad es: $edad");
  ?>
</BODY>
 </HTML>
```

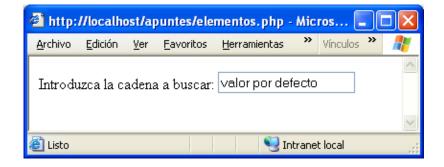
3.1.2. Formularios. Tipos de elementos.

- Acceso a los diferentes tipos de elementos de entrada de formulari
 - Elementos de tipo INPUT
 - TEXT
 - RADIO
 - CHECKBOX
 - BUTTON
 - FILE
 - HIDDEN
 - PASSWORD
 - SUBMIT
 - Elemento SELECT
 - Simple / múltiple
 - Elemento TEXTAREA

3.1.2. Forms. Tipos de elementos. TEXT

TEXT

```
Introduzca la cadena a buscar:
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto" SIZE="20">
<?PHP
    print ($cadena);
    //print ($_REQUEST ['cadena']);
?>
```



3.1.2. Forms. Tipos de elementos. RADIO

RADIO

```
<INPUT TYPE="radio" NAME="titulacion" VALUE="II" CHECKED>I.Informática
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIG">I.T.I. Gestión
<INPUT TYPE="radio" NAME="titulacion" VALUE="ITIS">I.T.I. Sistemas

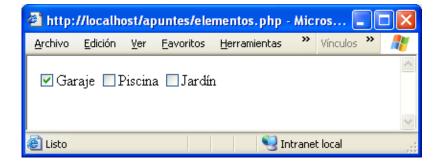
<?PHP
    print ($titulacion);
    //print ($_REQUEST ['titulacion']);
?>
```



3.1.2. Forms. Tipos de elementos. CHECKBOX

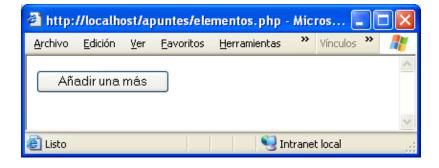
CHECKBOX

```
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="garaje" CHECKED>Garaje
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="piscina">Piscina
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="jardin">Jardin
<?PHP
    $n = count ($extras);
    for ($i=0; $i<$n; $i++)
        print ("$extras[$i]<BR>\n");
    //foreach ($_REQUEST['extras'] as $extra)
        //print ("$extra<BR>\n");
```



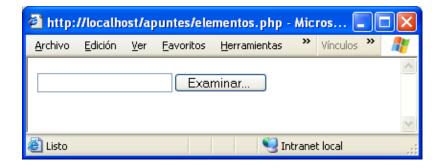
3.1.2. Forms. Tipos de elementos. BUTTON

BUTTON



3.1.2. Forms. Tipos de elementos. FILE

FILE



3.1.2. Forms. Tipos de elementos. HIDDEN

HIDDEN

```
<?PHP
   print("<INPUT TYPE='hidden' NAME='username' VALUE='$usuario'>\n");
?>

<?PHP
   print ($username);
   //print ($_REQUEST ['username']);
?>
```



3.1.2. Forms. Tipos de elementos. PASSWORD

PASSWORD

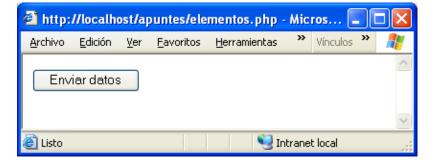
```
Contraseña: <INPUT TYPE="password" NAME="clave">
    <?PHP
        print ($clave);
        //print ($_REQUEST ['clave']);
?>
```



3.1.2. Forms. Tipos de elementos. SUBMIT

SUBMIT

```
<!NPUT TYPE="submit" NAME="enviar" VALUE="Enviar datos">
<!PHP
   if ($enviar)
      print ("Se ha pulsado el botón de enviar");
   //if ($_REQUEST ['enviar'])
      //print ("Se ha pulsado el botón de enviar");
?>
```



3.1.2. Forms. Tipos de elementos. SELECT simple

SELECT simple

```
<SELECT NAME="titulacion">
     <OPTION VALUE="II" SELECTED>Ingeniería Informática
     <OPTION VALUE="ITIG">Ingeniería Técnica en Informática de Gestión
     <OPTION VALUE="ITIS">Ingeniería Técnica en Informática de Sistemas
</SELECT>

<?PHP
    print ($titulacion);
    //print ($_REQUEST ['titulacion']);
?>
http://localhost/apuntes/elementos.php - Micros...
```

Edición Ver Favoritos

Ingeniería Informática

>> Vínculos >>

🐫 Intranet local

Herramientas

3.1.2. Forms. Tipos de elementos. SELECT múltiple

SELECT múltiple

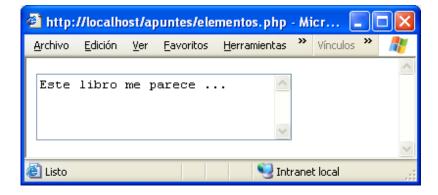
```
<SELECT MULTIPLE SIZE="3" NAME="idiomas[]">
   <OPTION VALUE="ingles" SELECTED>Inglés
   <OPTION VALUE="frances">Francés
                                             🙆 http://localhost/apuntes/elemento... 🖃
   <OPTION VALUE="aleman">Alemán
   <OPTION VALUE="holandes">Holandés
                                              Archivo Edición Ver Favoritos >> Vínculos >>
</SFLECT>
                                               Inalés
                                               Francés
                                               Alemán.
<?PHP
   n = count (sidiomas);
                                             El Listo
                                                             🐫 Intranet local
   for ($i=0; $i<$n; $i++)
       print ("$idiomas[$i]<BR>\n");
   //foreach ($_REQUEST['idiomas'] as $idioma)
       //print ("$idioma<BR>\n");
?>
```

3.1.2. Forms. Tipos de elementos. TEXTAREA

TEXTAREA

```
<TEXTAREA COLS="30" ROWS="4" NAME="comentario">
Este libro me parece ...
</TEXTAREA>

<?PHP
    print ($comentario);
    //print ($_REQUEST ['comentario']);
?>
```



3.1.3. Formularios. Trabajando formularios

- La forma habitual de trabajar con formularios en PHP es utilizar un
- Ventajas:
 - Disminuye el número de ficheros
 - Permite validar los datos del formulario en el propio formular
- Procedimiento:
 - si se ha enviado el formulario:
 - Procesar formulario
 - si no:
 - Mostrar formulario
 - fsi

3.1.3. Formularios. Trabajando formularios (2)

Para saber si se ha enviado el formulario se acude a la variable con

```
Input type="submit" name="enviar" value="procesar">
Input type="submit" name="submit" nam
```

3.1.4. Formularios. Subida de ficheros

- Para subir un fichero al servidor se utiliza el elemento de entrada F
- Hay que tener en cuenta una serie de consideraciones importantes
 - El elemento FORM debe tener el atributo ENCTYPE="multip
 - El fichero tiene un límite en cuanto a su tamaño. Este límite :
 - En el fichero de configuración php.ini
 - En el propio formulario

3.1.4. Formularios. Subida de ficheros (2)

php.ini

formulario

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE='1024000'>
<INPUT TYPE="FILE" NAME="fichero">
```

3.1.4. Formularios. Subida de ficheros (3)

- Consideraciones (cont)
 - Debe darse al fichero un nombre que evite coincidencias cor
 - El fichero subido se almacena en un directorio temporal y he
- Procedimiento:

- si se ha subido correctamente el fichero:
- Asignar un nombre al fichero
- Mover el fichero a su ubicación definitiva
- si no:
- Mostrar un mensaje de error
- fsi

_

3.1.4. Formularios. Subida de ficheros (4)

HTML

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- La variable \$_FILES contiene toda la información del fichero subido:
 - \$_FILES['imagen']['name']
 - Nombre original del fichero en la máquina cliente
 - \$_FILES['imagen']['type']
 - Tipo mime del fichero. Por ejemplo, "image/gif"
 - \$_FILES['imagen']['size']
 - Tamaño en bytes del fichero subido
 - \$_FILES['imagen']['tmp_name']
 - Nombre del fichero temporal en el que se almacena
 - \$_FILES['imagen']['error']
 - Código de error asociado al fichero subido

3.1.4. Formularios. Subida de ficheros (5)

PHP

3.1.5. Formularios. Validación de formularios

- Toda la información proveniente de un formulario debe considerarse por la
- Lo más eficiente es mostrar los errores sobre el propio formulario para fac

```
si se ha enviado el formulario:
si hay errores:
Mostrar formulario con errores
si no:
fsi
si no:
Mostrar formulario
fsi
fsi
fsi
```

3.1.5. Formularios. Validación de formularios (2)

Este procedimiento se puede resumir para que sólo haya que mostrar una vo

```
si se ha enviado el formulario:
validar datos
fsi
si se ha enviado el formulario y no hay errores:
Procesar formulario
si no:
Mostrar formulario con valores por defecto o ya
```

- fsi

3.2.1. Sesiones

- Las sesiones son un mecanismo que nos permite almacenar inforr
- Cuando un cliente visita nuestra página se le asigna un identificac
 - El identificador se almacenará en una cookie o en la prop
- Cuando se inicia una sesión PHP comprueba automáticamente si

3.2.2. Sesiones Iniciar una sesión en PHP

- Se puede iniciar una sesión invocando la función session_start().
 - Comprueba si ya existía alguna sesión. En caso afiri
- Si el identificador de sesión se almacena como una cookie, la sesi
 - El lugar dónde se almacena el identificador de sesió

3.2.3. Sesiones Sesiones en PHP

- Existen dos formas de acceder al contenido de una sesión en PHP
- •1. Las funciones session_register(),
- •session_is_registered() y session_unregister().
 - Si la opción register_globals está activada, al iniciar
 - •2. A través de la variable \$_SESSION.
 - Esta opción funciona en cualquier caso y se prefiere
 - •A partir de la versión 4.3 se pueden mezclar.

3.2.4. Sesiones Usando \$_SESSION

- Consultar si una variable está establecida.
- isset(\$_SESSION["variable"])
- Modificar el valor de una variable.
- \$_SESSION["variable"] = \$valor;
- Acceder al valor de una variable establecida.
- echo \$_SESSION["variable"];
- Eliminar una variable establecida en una sesión.
- unset(\$_SESSION["variable"]); // 4.3 o posterior
- session_unregister("variable"); // 4.3 o anterior

3.2.4. Sesiones Usando \$_SESSION (2)

```
<?php
session_start();
// Use $HTTP_SESSION_VARS with PHP 4.0.6 or less
if (!isset($_SESSION['count'])) {
                                       <?php
  $_SESSION['count'] = 0;
                                       session_start();
} else {
  $_SESSION['count']++;
                                       unset($_SESSION[`count']);
                                       ?>
Hello visitor, you have seen this
page <?php echo $_SESSION['count']; ?> times.
```

3.2.6. Sesiones Destruyendo una sesión

- La función **session_destroy()** destruye todos los datos asociados
 - Si está activada la opción register_globals, para des

```
<?php
// Initialize the session.
session_start();
// Unset all of the session variables.
$_SESSION= array();
// Finally, destroy the session.
session_destroy();
?>
```

3.2.7. Sesiones. Ejemplo

```
// Ejemplo, un contador
session_start();
print($_SESSION["contador"]);
$_SESSION["contador"]++;
```

<A HREF="siguiente.php?<?=SID?>">Continuar

3.3. Cookies.

echo \$_COOKIE["PruebaCookie"];

3.4.1. Tratamiento de errores. Operador @

```
$nombre = '/etc/shadow';

$fichero = @fopen ($nombre, 'r');

if( !$fichero )
{
    die("No se pudo abrir el fichero ($nombre)");
}
```

3.4.2. Tratamiento de errores. Error Handling

```
<?php
// usaremos nuestra propia funcion para el tratamiento de errores
error_reporting(0);
// función definida por el usuario de tratamiento de errores
function userErrorHandler ($errno, $errmsg, $filename, $linenum, $vars) {
   // timestamp para la entrada del error
   dt = date("Y-m-d H:i:s (T)");
   // define un array asociativo de string de errores
   // en realidad solo se tendrían que considerar
   // las entradas de 2,8,256,512 y 1024
    $errortype = array (
                   => "Error",
                1
               2 => "Warning",
               4 => "Parsing Error",
               8 => "Notice",
               16 => "Core Error",
               32 => "Core Warning",
               64 => "Compile Error",
               128 => "Compile Warning",
               256 => "User Error",
               512 => "User Warning",
               1024=> "User Notice"
               );
```

3.4.2. Tratamiento de errores. Error Handling (2)

```
// se prepara el error
    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errornum>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";
    $err .= "</errorentry>\n\n";
   // para testear
   // echo $err;
   // guardar en el log de errores
   error_log($err, 3, "/usr/local/php4/error.log");
   // si se trata de un error crítico se envía un correo al usuario
   if ($errno == E_USER_ERROR)
        mail("phpdev@example.com", "Critical User Error", $err);
}
```

3.4.2. Tratamiento de errores. Error Handling (3)

```
function distance ($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger error("Incorrect parameters, arrays expected", E USER ERROR);
        return NULL;
    if (count($vect1) != count($vect2)) {
        trigger error("Vectors need to be of the same size", E USER ERROR);
        return NULL;
    for ($i=0; $i < count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                            E USER WARNING);
            $c1 = 0.0;
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                            E USER WARNING);
            $c2 = 0.0;
        d += c2*c2 - c1*c1;
    return sqrt($d);
}
```

3.4.2. Tratamiento de errores. Error Handling (4)

```
$old_error_handler = set_error_handler("userErrorHandler");

// constante no definida, genera un aviso
$t = I_AM_NOT_DEFINED;

// define algun "vector"
$a = array(2,3,"foo");
$b = array(5.5, 4.3, -1.6);
$c = array (1,-3);

// genera un error de usuario
$t1 = distance($c,$b)."\n";

// genera otro error de usuario
$t2 = distance($b,"i am not an array")."\n";

// genera un aviso
$t3 = distance($a,$b)."\n";
```

Ejercicios

Parte 2

3.5.1. Cadenas. Comparación.

```
int strcmp (string str1, string str2)
int strcasecmp (string str1, string str2)

// Ejemplo:
if (strcmp($a, $b) == 0)
{
   echo 'iguales';
}
```

3.5.2. Cadenas. Subcadenas.

```
string substr (string cadena, int inicio
        [, int tamaño])

$str = substr('abcdef', 2, 3); // cde
$str = substr('abcdef', -2); // ef
$str = substr('abcdef', -2, 1); // e
$str = substr('abcdef', 1, -2); // bcd
```

3.5.2. Cadenas. Subcadenas. (2)

```
int strpos (string cadena, string referencia
            [, int inicio])
int strrpos (string cadena, char referencia)
string strstr (string cadena, string referencia)
$i = strpos('cadena de prueba', 'de');
// $i = 2
$i = strpos('cadena de prueba', 'de', 5);
// \$i = 7
$s = strrpos('cadena de prueba', 'de');
// $s = 7
$s = strstr('cadena de prueba', 'de');
// $s = dena de prueba
```

3.5.3. Cadenas. Imprimir y formatear.

```
int printf (string formato [, mixed args...])
string sprintf (string formato [, mixed args...])
```

Secuencias de

formato

- 1. Relleno
- 2. Alineación
- 3. Número de caracteres
- 4. Precisión
- 5. Tipo
- % El carácter de tanto por ciento.
- **b** Entero en binario.
- c Entero como carácter ASCII.
- **d** Entero en decimal.
- **f** Double en punto flotante.
- **o** Entero en octal.
- **s** Cadena.
- **x** Entero en hexadecimal (minúsculas).
- **X** Entero en hexadecimal (mayúsculas).

3.5.3. Cadenas. Imprimir y formatear. (2)

```
printf("%02d/%02d/%04d", $dia, $mes, $año);

$pago1 = 68.75;
$pago2 = 54.35;
$pago = $pago1 + $pago2;

// echo $pago mostraría "123.1"
// Mostrar al menos un dígito entero y exactamente //
printf ("%01.2f", $pago);
```

3.5.4. Escapar caracteres. SQL.

```
$busca = "D'Alton";

// Habrá que escapar el apóstrofe
$busca = addslashes($busca);
$sql = "SELECT *
          FROM usuarios
          WHERE apellido = '$busca'";
```

3.5.4. Escapar caracteres. Shell.

```
string system (string comando [, int valor_salida])
echo system("finger $usuario");
```

¿Qué pasa si \$usuario="pepe; apachectl stop"?

string escapeshellcmd (string comando)

3.5.4. Escapar caracteres. HTML.

```
$valor = "a>b";
echo '<input type=hidden name=var value="' .
    htmlspecialchars($valor) . '">';

// <input type=hidden name=var value="a&gt;b">
```

```
string nl2br (string cadena)
```

3.5.5. Extraer campos.

3.5.5. Extraer campos. Expresiones regulares.

```
array split (string delimitador, string cadena [, int límite])
```

```
$fecha = "12/4 2000";
$campos = split ('[ /.-]', $fecha);
```

3.5.6. Recorrer un array.

```
reset(), end(), next(), each(), current(), key()
$arr = array(1, 'cosa', 1.57, 'gato'=>'raton', 'perro'=>'gato');
current($arr); // 1
next($arr); // cosa
current($arr); // cosa
prev($arr); // 1
end($arr); // gato
current($arr); // gato
key($arr); // perro
reset($arr); // 1
each($arr);  // array(0,1)
each($arr);  // array(1,'foo')
each($arr); // array(2,1.57)
```

3.5.7. Ordenar un array.

• *sort()*: Ordena el array por contenido en orden ascendente.

•

rsort(): Ordena por contenido en orden descendente.

• *ksort()*: Ordena por el índice en orden ascendente.

• *rksort()*: Ordena por el índice en orden descendente.

3.5.8. Otras funciones.

Eliminar espacios en blanco:

```
string trim (string cadena)
string ltrim (string cadena)
string rtrim (string cadena)
```

Mayúsculas y minúsculas:

```
string strtoupper (string cadena) string strtolower (string cadena) string ucfirst (string cadena)
```

3.6.1. Ficheros. Abrir y cerrar.

```
int fopen (string nombre, string modo
                 [, int include_path])
• Modos:
         • 'r' Sólo lectura. Puntero al inicio.
         •'r+' Lectura/escritura. Puntero al
        inicio.
         • 'w' Sólo escritura. Se trunca el
        fichero.
         •'w+' Lectura/escritura. Se trunca el
        fichero.
                Sólo escritura. Puntero al final.
(Int. identificador)
Lectura/escritura. Puntero al final.
```

3.6.2. Ficheros. Leer y escribir.

```
string fgets (int identificador, int tamaño)
mixed fscanf (int identificador, string formato
              [, string var1...])
int feof (int identificador)
array file (string fichero
            [, int include_path])
int fwrite (int identificador, string cadena
            [, int tamaño])
```

3.6.3. Ficheros. Copiar / renombrar / borrar.

```
int copy (string origen, string destino)
int rename (string origen, string destino)
int unlink (string fichero)
```

3.6.4. Directorios.

```
int chdir (string directorio)
int mkdir (string nombre, int modo)
int rmdir (string nombre)
```

```
int opendir (string nombre)
string readdir (int identificador)
void closedir (int identificador)
```

3.6.4. Directorio. Listado de contenidos.

```
$direcotrio = opendir('.');
while (($fichero = readdir($directorio)) !== FALSE)
{
    echo "$fichero\n";
}
closedir($directorio);
```

3.7.1. POO. Definición de una clase.

```
class NombreClase
{
    var $variables;

    function metodos ($parametros)
    {
        codigo
    }
}
```

2.7.1. POO. Definición de una clase. Ejemplo.

```
class Coche {
   var $velocidad; // Velocidad actual
    // Constructor por defecto. El coche está parado.
    function coche() {
        this->velocidad = 0;
    // Constructor que indica la velocidad inicial.
   function coche($vel) {
        $this->velocidad = $vel;
   // Método acelerar. El coche va más rápido.
    function acelerar() {
        $this->velocidad++;
    // Método frenar. El coche va más lento hasta frenar.
    function frenar() {
        if ($this->velocidad > 0) {
            $this->velocidad--;
```

2.7.2. **POO.** Herencia.

```
class ClaseDerivada extends ClaseBase
{
    // definición de métodos y variables
    // exclusivos de ClaseDerivada,
    // y redefinición (especialización)
    // de métodos de ClaseBase
}
```

2.7.2. POO. Herencia. Ejemplo.

```
class CocheFantastico extends coche() {
    // Frenado instantáneo
    function frena() {
        $this->velocidad = 0;
    // ¡El coche habla!
    function habla() {
        echo "Hola, Michael.";
    // ¡Salta!
    function salta() {
        echo "Boing!!";
    // Turbo propulsión
    function turbo() {
        $this->velocidad = 200;
```

2.7.3. POO. Creación y uso de objetos.

```
// Creación (instanciación)
$MiCoche = new Coche;
$MiCocheSeMueve = new Coche(10);

// Uso
$MiCoche->acelerar();
$MiCoche->acelerar();
$MiCoche->acelerar();
echo $MiCoche->velocidad;
$MiCoche->frenar();
```

2.8.1. Plantillas

- Use "plantillas" para simplificar el mantenimiento del o
- Separa datos de las páginas con elementos HTML
- Una "plantilla" es un simple archivo de texto que contie
- Un 'contenedor" de variable es una variable definida en

2.8.2. Plantillas Ejemplo

```
<!-- begin: example.thtml -->
<html>
<head>
</head>
  <body>
   <br/><b>Employee Name</b>: {EMP_FNAME} {EMP_LNAME}
   >
   <b>Job</b>: {JOB}
   >
    <br/><b>Email address</b>: {EMAIL_ADDRESS}
    >
    <br/><b>Job Description</b>: {DESCRIPTION}
  </body>
</html>
<!-- end: example.thtml -->
```

2.8.2. Plantillas Ejemplo (2)

```
PHP Script para la plantilla

<?
// example.php - genera salida usando plantillas
// incluye archivo con la clase
  include("template.inc");
// instancia al nuevo objecto
  $t = new Template(".");

// asigna nombres a los archivos de plantilla
// "example" ahora referencia la plantilla "./example.thtml"
  $t->set_file("example", "example.thtml");
```

2.8.2. Plantillas Ejemplo (3)

```
// asigna valores a los contenedores de
// variable de la plantilla
// esto se podría también hacer con un array
// asociativo con los pares key-value
  $t->set_var("EMP_FNAME", "Jane");
 $t->set_var("EMP_LNAME", "Doe");
 $t->set_var("JOB", "Engineer");
 $t->set_var("EMAIL_ADDRESS", "jdoe@anonymous.com");
  $t->set_var("DESCRIPTION", "All around work horse");
// parse de la plantilla "example", se almacena
// en el handler "someoutput"
  $t->parse(someoutput, "example");
// muestra el contenido del handler "someoutput"
 $t->p(someoutput);
?>
```

2.8.3. Plantillas Recursos

- Smarty Template Engine: http://smarty.php.net
- Algunas de las características de Smarty:
 - Es extremamente rápido.
 - Solo compila una vez y él está atento para recompilar los archivos de plantil
 - Se puede crear funciones habituales y modificadores de variables customiza
 - Los construtores if/elseif/else/endif son pasados por el interpretador de PHP
 - Permite un anidamiento ilimitado de sections, ifs, etc.
 - Soporte de caching incrustado.
 - Arquitectura de Plugin .

2.8.4. Plantillas Ejemplo con Smarty

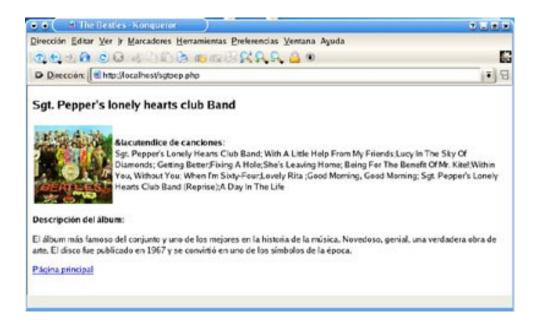
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"</pre>
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-2" />
<meta name="description" content="Discografía del conjunto The Beatles" />
<title>{$albumName}</title>
</head>
<body>
<h3>{$albumName}</h3>
<img src="{$albumCover}" alt="" />
   <b>Índice de canciones:</b><br />{$albumSongs}
 <h4>Descripción del álbum:</h4>
{$albumSynopsis}
<a href="">Página principal</a>
</body>
</html>
```

2.8.4. Plantillas Ejemplo con Smarty (2)

```
<?
require ("Smarty.class.php");
$smarty = new Smarty;
$smarty->assign("albumName", "Sgt. Pepper's lonely hearts club Band");
$smarty->assign("albumCover", "grafika/sgtpep.jpg");
$smarty->assign("albumSongs", "Sgt. Pepper's Lonely Hearts Club Band;
   With A Little Help From My Friends; Lucy In The Sky Of Diamonds;
   Getting Better; Fixing A Hole; She's Leaving Home;
   Being For The Benefit Of Mr. Kite!; Within You, Without You;
   When I'm Sixty-Four; Lovely Rita; Good Morning, Good Morning;
   Sgt. Pepper's Lonely Hearts Club Band (Reprise); A Day In The Life");
$smarty->assign("albumSynopsis", "El álbum más famoso del conjunto y uno
   de los mejores álbumes en la historia de la música. Novedoso, genial -
   - una verdadera obra de arte. El disco fue introducido al mercado en 1967 y se o
   de los símbolos de esa época.");
$smarty->display("album.tpl");
?>
```

2.8.4. Plantillas Ejemplo con Smarty (3)

Resultado



2.8.5. Plantillas Smarty – section

```
<?php
// ----- index.php -----
require("Smarty.class.php");
$smarty = new Smarty;
$smarty->assign("FirstName",array("Ned","Bart","Montgomery"));
$smarty->assign("LastName",array("Flanders","Simpson","Burns"));
$smarty->display("index.tpl");
?>
```

2.8.5. Plantillas Smarty – section (2)

2.8.6. Plantillas Smarty – Otros

• Todo lo que se encuentra entre {literal} y {/literal} no será interp

```
{literal}
<style>
p {color: #000000;
font-size: 12px}
</style>
{/literal}
```

• Incluir otras plantillas en la plantilla actual:

```
{include file="header.tpl" title="Main Menu" table_bgcolor="#c0c0c0"}
{* el cuerpo del template va aqui *}
{include file="footer.tpl" logo="http://my.domain.com/logo.gif"}
```

2.8.7. Plantillas Mecanismo de Smarty

1) "Compilación" de la plantilla. Conversión de la plantilla a un documento normal de PHP Esto significa que, por ejemplo, la notación {\$var} será cambiada a:

```
<?php
echo $this->_tpl_vars['var'];
?>
```

- 2) El documento es salvado automáticamente dentro del directorio /templates_c.
- 3) En la siguiente llamada al archivo file.php, el servidor enviará al navegador los archivos ya convertidos, lo que p

