Lenguaje PHP. Matrices, cadenas y fechas

Objectivos

En este módulo se profundizará en el manejo de las matrices. La necesidad de su aprendizaje se basa en que gran parte de las funciones de PHP manejan vaiables de este tipo. La manipulación de las cadenas de caracteres es necesaria cuando se debe mostrar información al usuario, por lo que se verán gran variedad de funciones que manejan este tipo de datos. Y por último se estudiarán las funciones existentes en PHP para el manejo de fechas.

Contenidos

Arrays o matrices

En muchas ocasiones, aparecen un conjunto de variables que contienen información semejante que ha de procesarse de forma similar. En estos casos existe una solución mejor que definir tantas variables como datos distintos tengamos, esta solución es utilizar arrays o matrices. Este tipo de variable funciona de manera parecida en otros lenguajes, pero con alguna particularidad.

Definición de arrays

```
<?php
/*-------------------------------------------------------------------
* Módulo: 3 Práctica: 1
* Autor:      Fecha:
* Descripción: Definición de arrays
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

$nota[0]="Insuficiente";
$nota[1]="Suficiente";
$nota[2]="Bien";
$nota[]="Notable";
$nota[]="Sobresaliente";

/* Si no indicamos subíndice, automaticamente se asignará la siguiente posición libre */

echo "Calificación: ". $nota[3];
echo "<br><br>";

/* Definición de un array utilizando la función array

$nota=array("Insuficiente","Suficiente","Bien","Notable",

"Sobresaliente");

echo "Calificación: ". $nota[4];
?>
```

*Listado 3.1.1. Definición de los elementos de un **array***

Observar que el primer número de índice de los arrays es siempre 0 y no 1.

Arrays asociativos

Una de las particularidades de PHP es la posibilidad de poder substituir los números de índice por alguna palabra que se asocie a cada elemento. Ver el siguiente ejemplo:

```
<?php
/*-------------------------------------------------------------------
```

```
* Módulo:  Práctica: 2
* Autor:      Fecha:
* Descripción: Arrays asociativos
* Pre condi.:
* Post cond.:
-----------------------------------------------------------------*/

$nota["I"]="Insuficiente";
$nota["S"]="Suficiente";
$nota["B"]="Bien";
$nota["N"]="Notable";
$nota["E"]="Sobresaliente";

echo "Valores asignados directamente <br>";
echo "Calificación: ". $nota["B"];
echo "<br><br>";

// Definición utilizando la función array

$nota=array("I"=>"Insuficiente","S"=>"Suficiente","B"=>"Bien",
"N"=>"Notable","E"=>"Sobresaliente");

echo "Definición con función array <br>";
echo "Calificación: ". $nota["E"];
?>
```

*Listado 3.1.2. Definición de los elementos de un **array** asociativo*

En lugar de asignar a cada elemento un número de índice, le asignamos una etiqueta o clave con la cual se relacionan. Al utilizar la función array debemos realizar la asignación del valor con el operador **=>** .

Matrices o arrays de dos dimensiones

Con el fin de ver las funciones que tiene PHP para la manipulación de matrices, utilizaremos como ejemplo un listado de alumnos, de los cuales almacenaremos una serie de datos personales. Utilizaremos un array asociativo con claves: nombre teléfono, correo, nota. Y una matriz indexada, de forma que cada alumno ocupe una posición de la matriz, para almacenar los 35 alumnos del curso.

```
/* Declaración implícita de una matriz bidimensional */

$M1[0][0] = 'José';
$M1[1][0] = 'María';
$M1[2][0] = 'Juan';
$M1[0][1] = '123456789';

/* Declaración implícita de una matriz bidimensional asociativa */

$M1[0]['Nombre'] = 'José';
$M1[1]['Nombre'] = 'María';
$M1[2]['Nombre'] = 'Juan';
$M1[0]['Telefono'] = '123456789';

/* Declaración de una matriz bidimensional asociativa utilizando la función array */

$curso=array(array('Nombre'   =>'José',
                   'Telefono' =>'123456789',
                   'Correo'   =>'a@a.es',
                   'Nota'     =>'0'),
             array('Nombre'   =>'María',
```

```
            'Telefono' =>'876543219',
            'Correo'   =>'b@b.es',
            'Nota'     =>'0'),
        array('Nombre'   =>'Juan',
            'Telefono' =>'432187659',
            'Correo'   =>'c@c.es',
            'Nota'     =>'0'));
```

Listado 3.1.3. Declaración de la matriz ejemplo

Para acceder a los elementos de una matriz multidimensional el mecanismo es el mismo que para acceder a matrices unidimensionales, basta con añadir tantos corchetes al final del nombre de la variable como dimensiones tenga ésta.

En nuestro ejemplo: para acceder a nombre del alumno que ocupa la 4 posición del listado, deberíamos escribir:

```
$alu4 = $curso[3]["Nombre"];
```

Si quisieramos recuperar todos los datos de un alumno, por ejemplo el tercero, obtendríamos una array. Deberíamos escribir:

```
$alu3 = $curso[2];
```

Las matrices como variables que son, pueden ser expandidas dentro de una cadena de caracteres delimitada con comillas dobles ("). Sin embargo debe ser diferente cuando se trabaje con matrices multidimensionales. hay que encerrar dicha referencia entre llaves ({}). Con matrices unidimensionales no hay que hacer nada especial siendo correctos los siguientes ejemplos:

```
/* mostraría el nombre del alumno, siendo $alumno un array unidimensional asociativo */

echo "Alumno número 1 $alumno['nombre'] <br>\n";

/* en nuestro ejemplo de matriz bidimensional */

echo "Alumno número 1 {$curso[0]['Nombre']} <br>\n";
```

Recorrido de una matriz

Para recorrer una matriz indexada necesitamos saber de antemano el número de elementos que la componen. Esta información nos la proporciana la función *count()*. Con la siguiente sintaxis:

int count(cualquier_tipo mi_variable)

Donde mi_variable representa la variable de la que se quiere obtener el número de elementos.

- Si se trata de una matriz, tanto indexada como asociativa, la función devolverá el número de elementos.
- Si se trata de una variable que contiene un elemento, sea o no matriz, devolverá 1.
- Si la variable no tiene asignado ningún valor o tiene asignado el valor null, devolverá un 0.

Existe otra función de PHP que devuelve el numero de elementos, esta función es *sizeof()* y su sintaxis es la siguiente:

int sizeof(array matriz)

Para recorrer una matriz indexada tan solo debemos construir un bucle, que empieza en la posición 0 y que terminará cuando ya no queden más elementos que tratar. En cada interacción del bucle se accede a una posición distinta.

Para recorrer una matriz asociativa PHP dispone de varias funciones:

Función	**resultado**
each()	Recupera el par formado por la clave y el valor del elemento actual y además avanza una posición el puntero de la matriz. Sintaxis: array each(array matriz);

	Devuelve false cuando no quedan elementos por tratar
List()	Asigna los valores del elemento actual de una matriz a las variables que se hayan pasado com parametro. sintaxis: void list($var1, $var2, ...);
reset()	Hace que el puntero interno apunte a la primera posición de la matriz.
end()	Hace que el puntero interno apunte a la última posición de la matriz.
next() **prev()**	Permite ir al siguiente / anterior elemento. En caso de estar al principio o al final devuelve el valor falso.
current()	Devuelve el contenido del elemento actual. Esta función no desplaza el puntero interno de posición y devuelve el valor false cuando se encuentre después del último elemento o si la matriz no tiene elementos.

Cuando se trata de matrices indexadas la navegación es sencilla, basta con acceder directamente a la posición que contiene el elemento buscado. Por lo contrario, cuando se trata de una matriz asociativa no puede aplicarse el mismo tratamiento. Hemos visto 4 funciones para poder desplazarnos en una matriz asociativa.

En PHP existen una gran cantidad de funciones que permiten manipular matrices que permiten, por ejemplo, ordenarlas, realizar una busqueda, insertar elementos, tando detrás como delante de la matriz, etc. pero no es el objetivo de este curso entrar en profundidad todas y cada una de ellas. Todas estas funciones pueden ser consultadas en la siguiente direccción Funciones con arrays.

Un ejemplo

En nuestro ejemplo construiremos una tabla, en la que cada fila será un alumno con todos sus datos, y las cabeceras serán las claves de la matriz asociativa.

```
<!doctype html public "-//W3C//DTD HTML 4.0 //EN">
<html>
<head>
      <title>Title here!</title>
</head>
<body>
<?php
/* Declaración de una matriz bidimensional asociativa utilizando la función array */

$curso=array(array('Nombre'   =>'José',
                   'Telefono' =>'123456789',
                   'Correo'   =>'a@a.es',
                   'Nota'     =>'0'),
             array('Nombre'   =>'María',
                   'Telefono' =>'876543219',
                   'Correo'   =>'b@b.es',
                   'Nota'     =>'1'),
             array('Nombre'   =>'Juan',
                   'Telefono' =>'432187659',
                   'Correo'   =>'c@c.es',
                   'Nota'     =>'2'));

//  escribe cabeceras

echo "<TABLE BORDER=3 ALIGN=CENTER>\n";
echo "<TR>\n";
while ($elemento = each($curso[0]))
        echo "<TH>$elemento[0]</TH>\n";
```

```
echo "</TR>\n";

reset($curso[0]);

//recorrer la matriz

for ($i = 0; $i < count($curso); $i++)
{
  echo "<TR>\n";
  // recorrer cada elemeto de la matriz asociativa
  // utilizamos el contructor list
  while (list($clave,$valor) = each($curso[$i]))
    echo "<TD>$valor</TD>\n";
  echo "</TR>\n";
}
echo "</TABLE>\n";
?>
</body>
</html>
```

Comprobar que si no escribimos la sentencia reset($curso[0]), el primer elemento de la matriz no será visualizado. Cuando recorramos una matriz asociativa debemos posicionar el puntero interno en el primer elemento.

En el tema anterior vimos una sentencia que nos facilita el recorrido de las matrices, esta sentencia es foreach(), que en cada iteracción del bucle la variable irá cogiendo un valor del array y avanzando al siguiente elemento en cada pasada, hasta llegar al final del array de forma automática.

Veamos el mismo ejemplo, pero utilizando la sentencia foreach.

```
<!doctype html public "-//W3C//DTD HTML 4.0 //EN">
<html>
<head>
      <title>Title here!</title>
</head>
<body>
<?php
/* Declaración de una matriz bidimensional asociativa utilizando la función array */

$curso=array(array('Nombre'   =>'José',
                   'Telefono' =>'123456789',
                   'Correo'   =>'a@a.es',
                   'Nota'     =>'0'),
             array('Nombre'   =>'María',
                   'Telefono' =>'876543219',
                   'Correo'   =>'b@b.es',
                   'Nota'     =>'1'),
             array('Nombre'   =>'Juan',
                   'Telefono' =>'432187659',
                   'Correo'   =>'c@c.es',
                   'Nota'     =>'2'));

//  escribe cabeceras
echo "<TABLE BORDER=3 ALIGN=CENTER>\n";
echo "<TR>\n";
foreach ($curso[0] as $k=>$v)
        echo "<TH>$k</TH>\n";
echo "</TR>\n";
//recorrer la matriz

foreach ($curso as $e)
{
  echo "<TR>\n";
  foreach ($e as $v)
    echo "<TD>$v</TD>\n";
```

```
  echo "</TR>\n";
}
echo "</TABLE>\n";
?>
</body>
</html>
```

Cadenas de caracteres

En este capítulo se verán una serie de funciones de PHP, que se agruparán de forma temática abordando los siguientes aspectos: formas de mostrar en salida estándard el contenido de las cadenas de caracteres y funciones para manipular su contenido.

En PHP las cadenas se pueden especificar mediante tres tipos de delimitadores:

- Comillas simples: '
- Comillas dobles: "
- Documento incrustado: <<<

La primera de ellas es la más básica, y permite asignar a una variable de tipo cadena de caracteres todo el texto que vaya encerrado entre las dos comillas simples. Si el texto contiene comillas simples ('), ésta deberá ir precedida por el carácter barra invertida (\). Esto se denomina escapar un carácter.

Si dentro de de una cadena delimitada entre comillas simples se hace referencia al contenido de una variable, ésta no será tenida en cuenta. Por ejemplo:

```
<?
// Ejemplo de comillas simples en el contenido de un cadena

$cadena1 = 'Este es un ejemplo de \'comillas simples \' en texto';
$cafes = 2;

// Ejemplo de utilización de variables en cadenas

$cadena2 =' El contenido de la variable $cadena1, no es tenida en cuenta \n, este salto tampoco será interpretado';

$cadena3 ='El contenido de la variable $cafe, no es tenida en cuenta \n, este salto tampoco será interpretado';

echo $cadena1;
echo $cadena2;
echo $cadena3;

?>
```

La delimitación mediante comillas dobles realiza la misma función que las simples , pero es más avanzada, en este tipo de cadenas los caracteres especiales definidos por el lenguaje son interpretados. Deberemos recurrir a la técnica de escapar los caracteres especiales que pudiendo ser interpretados no nos interesa, por ejemplo dobles comillas y barra invertida.

En este tipo de cadenas, si podemos hacer referencia al contenido de las variables, siendo estas sustituidas por su contenido, en la cadena.

```
<?
// Ejemplo de comillas dobles en el contenido de un cadena

$cadena1 = "Este es un ejemplo de \"comillas dobles \" en texto";
```

```
// Ejemplo de utilización de variables en cadenas

$cadena2 ="El contenido de la variable $cadena1, SI es tenida en cuenta \n,  y este salto también será interpretado";

echo $cadena1;
echo $cadena2;

?>
```

Escribe el código de los dos listados anteriores y comprueba la diferencia de resultados.

Cuando se delimita una cadena utilizando la técnica del documento incrustado , es válido todo lo explicado para los delimitadores de comillas dobles. La diferencia radica en que con este tipo de delimitadores la cadena de caracteres puede ocupar tantas línas como sea necesario, además la cadena guardará la apariencia con la que fue escrita. Estas cadenas deben ir delimitadas por un identificador que el programador define. Por ejemplo:

```
<?

// ejemplo de delimitadores <<<

$cadena1= <<< DELI1
      Me llamo Jose. Soy profesor de
      informática. \n Esto es un salto interpretado.
DELI1;

$cadena2=<<< DELI2
      Este es el contenido de la variabe
      $cadena1
DELI2;

echo $cadena1, "\n";
echo $cadena2;

?>
```

Escribe el código del ejemplo anterior y comprueba el resultado.

Visualización de cadenas

PHP dispone de funciones que muestran el contenido de variablas con y sin formato específico. estas funciones son:

Función	resultado
echo()	Es la más utilizada, muestra cadenas de caracteres en la salida estándar. No acepta formato de salida. Sintaxis: echo string arg1 [, string arg2]; Sintaxis: echo (string arg1 [, string arg2]);
print()	Es la más sencilla, y muestra el contenido de una cadena de caracteres en la salida estándar. No acepta formato de salida. Sintaxis: printf(string cadena);
printf()	Realiza la misma acción que la función anterior, con la diferencia que ésta si acepta argumentos de formato de salida. Sintaxis: printf(string formato [, cualquier valor , ...]);

Aunque lo más lógico sería recurrir a etiquetas HTML para dar formato a la salida de la función echo(), veremos un ejemplo donde se puede apreciar como utilizar cada una de las funciones anteriores.

```
<?
// Escribir un literal con echo()
echo "Esto es la cabecera del listado";
echo ("Esta sentencia es equivalente a la anterior");

// Escribe el contenido de cualquier variable
$var1 = 18;
$var2 = 35.20;
$var3 = "José";

echo $var1;
echo $var2;
echo $var3;

// combinación de cadenas y variables
echo "La suma de ", $var1, " y ", $var2, " es ", $var1 + $var2;

// combinación de cadenas y variables utilizando concatenación
echo "La suma de " . $var1 . " y " . $var2 . " es " . ($var1 + $var2);

// con print y printf
print("Esta es otra forma de visualizar cadenas \n");
print("Mi nombre es $var3 \n");

// con formato
printf("La suma:  %d + %.5f = %.5f \n", $var1, $var2, $var1 + $var2);

?>
```

Escribe el código del ejemplo anterior y comprueba el resultado.

Funciones de alteración del contenido

En este apartado veremos algnunas de las funciones más utilizadas para manipular el contenido de las variables string.

Función	resultado
chop()	Devuelve una cadena de caracteres a la que se han eliminado los caracteres en blanco y el de nueva línea que aparece al final de la cadena. Sintaxis: string chop(string cadena);
ltrim() **rtrim()** **trim()**	Eliminan los caracteres en blanco que aparecen a la izquierda, a la derecha y a la izquierda y la derecha, respectivamente. Sintaxis: string ltrim(string cadena);
str_pad()	Ajusta el tamaño de una cadena de caracteres a una longitud determinada, permitiendo especificar el carácter de relleno.Realiza la misma acción que la función anterior, con la diferencia que ésta si acepta argumentos de formato de salida. Sintaxis: string str_pad(string cadena, int longitud [, string relleno [, int lugar]]);
strtolower	1. Convierte todos los caracteres alfabéticos a minúsculas. 2. Convierte todos los caracteres alfabéticos a mayúsculas. 3. Convierte todos los caracteres alfabéticos a minúsculas,

() **strtoupper()** **ucfirst()** **ucwords()**	excepto el primero de la cadena. 4. Convierte todos los caracteres alfabéticos a minúsculas, excepto el primer carácter de cada palabra. Sintaxis: string strtoupper(string cadena);
str_replace()	Recibe una cadena de caracteres y devuelve otra en la que se han sustituido todas las apariciones de una subcadena2 por otra subcadena1. Sintaxis: string str_replace(string subcadena1, string subcadena2, string cadena);
strstr()	Permite establecer una correspondencia entre los diferentes caracteres a traducir y sus sustitutos. Sintaxis: string strstr(string cadena, string originales, string traducidos); ejemplo: $nom = strstr($nom, "áéíóúÁÉÍÓÚ", "aeiouAEIOU");

Funciones de acceso al contenido

Función	**resultado**
strlen()	Devuelve un número entero que indica cuántos caracteres tiene la cadena que recibe como parámetro. Sintaxis: int strlen(string cadena);
strchr()	Devuelve la subcadena que comienza en la primera aparición del carácter indicado. Sintaxis: string strchr(string cadena, char caracter);
strstr()	Permite localizar una subcadena dentro de una cadena original. Es sensible a las mayúsculas y minúsculas. Existe stristr() que tiene la misma funcionalidad pero no es sensible. Sintaxis: string strstr(string cadena, string subcadena);
strpos() **strrpos()**	Indica la posición de la primera (última) ocurrencia de una cadena en otra. Sintaxis: int strpos(string cadena, string subcadena [, int pos_inicial]);
substr()	Devuelve la porción de cadena original que empieza en una determinada posición y que tiene una determinada longitud. Sintaxis: string substr(string subcadena1, int comienzo, int longitud);
strcmp() **strcasecmp()** **strncmp()**	1. Permite comparar dos cadenas, siendo sensible a mayúsculas y minúsculas. 2. Permite comparar dos cadenas, NO siendo sensible a mayúsculas y minúsculas. 3. Permite comparar los n primeros caracteres de dos cadenas. Sintaxis: int strcmp(string cadena1, string cadena2);

Manipulación de fechas

En este capítulo se verán algunas de las funciones más utilizadas para el empleo de fechas.

El día 1 de enero de 1970 a la 00:00:00 GMT comenzo la "era UNIX". Cuando se solicita la hora a un ordenador, realmente no se obtienen la hora, los minutos y los segundos de un día de un mes y de un año determinado, se obtienen los segundos transcurridos desde el inicio de la era de UNIX, a este valor se le denomina "timestamp".

La función time() devuelve un número entero que representa la marca de tiempo (timestamp) correspondiente al instante en que se ejecuta dicha función.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 3 Práctica: 1
* Autor:  Fecha:
* Descripción:
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/
?>
<html>
<?php
// definimos la constante para el salto de línea HTML
define("B","<br>");

echo B;
echo B;
echo B;
echo "PHP<br> \n";
echo "------------------<br> \n";
echo "Hola <b>Mundo</b> \n";
echo "Ahora son las: ",time()," \n";

?>
</html>
```

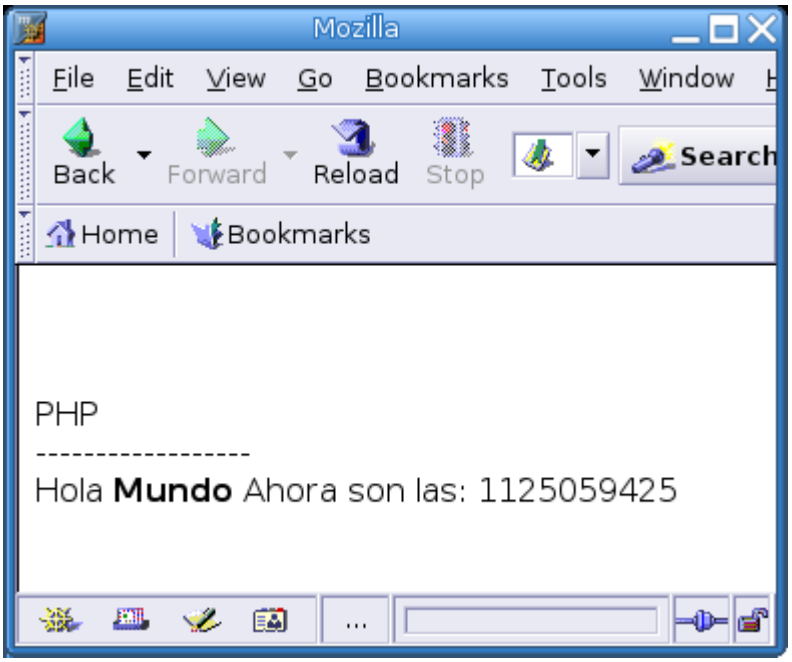

Figura 3.3.1. Ejemplo de la función time().

Este número entero es poco útil, a la hora de mostrar el tiempo, la forma normal de trabajar sigue siendo con días, meses, años, horas, minutos y segundos. Y para ello PHP dispone de una función que se encarga de obtener la fecha del sistema presentada de una forma determinada.

Esta función es getdate() que devuelve una matriz asociativa con la información de la fecha y hora del sistema.

Los elementos de la matriz son:

Clave	Descripción	Ejemplo de valores devueltos
"seconds"	Representación numérica de segundos	0 a 59
"minutes"	Representación numérica de minutos	0 a 59
"hours"	Representación numérica de horas	0 a 23
"mday"	Representación numérica del día del mes	1 a 31
"wday"	Representación numérica del día de la semana	0 (para el Domingo) a 6 (para el Sábado)
"mon"	Representación numérica de un mes	1 a 12
"year"	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
"yday"	Representación numérica del día del año	0 a 365
"weekday"	Una representación textual completa del día de la semana	Sunday a Saturday
"month"	Una representación textual completa de un mes, como January o March	January a December
0	Segundos desde el Epoch Unix, similar a los valores devueltos por time() y usados por date().	Depende del sistema, típicamente -2147483648 a 2147483647.

```php
<?php
/*-------------------------------------------------------------
* Módulo: 3 Práctica: 2
* Autor:  Fecha:
* Descripción:
* Pre condi.:
```

```
* Post cond.:
-----------------------------------------------------------------*/
?>
<html>
<?php
// definimos la constante para el salto de línea HTML
define("B","<br>");

echo B;
echo B;
echo B;
echo "PHP<br> \n";
echo "------------------<br> \n\n";
$hora=getdate();
echo "Ahora son las: ",$hora[hours]," horas \n";

?>
</html>
```

En el ejemplo anterior se muestra la hora actual del servidor.

Formato de fechas

PHP dispone de la función date() que devuelve una cadena de caracteres que se corresonden con la fecha a la que se ha aplicado un determinado formato.

Sintaxis: string date(strinf formto [,int hora]);

Para la definición del formato disponemos de las siguientes opciones, no se han mostrado todas las opciones de que dispone, solo las más utilizadas:

Opción	Descripción
d	Día del mes con dos dígitos.
D	Día de la semana con tres letras (ejemplo Mon).
F	Nombre del mes completo.
h	Hora con dos dígitos, con fromato 01 a 12.
H	Hora con dos dígitos, con fromato 00 a 23.
i	Minutos con dos dígitos.
l	Ele minúscula, aparece el nombre completo del día de la semana (ingles).
m	Mes con dos dígitos.
M	Mes con tres letras (ejemplo Mar).
s	Segundos con dos dígitos
Y	Año con cuatro dígitos
y	Año con dos dígitos

Veamos un ejemplo donde se muestra la utilización de estas opciones:

```
<?php
/*-----------------------------------------------------------------
* Módulo: 3 Práctica: 3
* Autor:  Fecha:
* Descripción:
* Pre condi.:
* Post cond.:
```

```
----------------------------------------------------------------*/
?>
<html>
<?php
// definimos la constante para el salto de línea HTML
define("B","<br>");

echo B;
echo B;
echo B;
echo "PHP<br> \n";
echo "------------------<br> \n\n";
echo "Hoy es ", date("D"), ", ", date("d"), " de ",
date("F"), " de ", date("Y"),"\n";
echo B;
echo B;
// directamente desde date()
echo date("\ho\y e\s D, d \de F \de Y");
?>
</html>
```


Escribe el código del ejemplo anterior y comprueba el resultado.

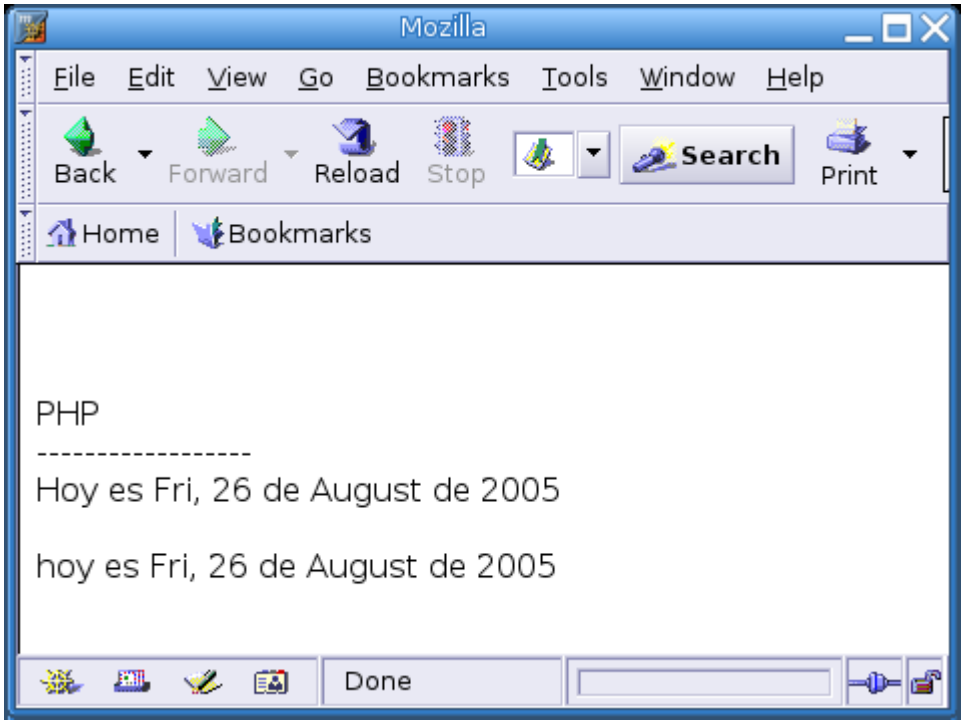

Figura 3.3.2. Ejemplo de la función date().

Manipulación de fechas

La función strftime() representa otra posibilidad de aplicar formato a una fecha. A diferencia de la anterior esta función utiliza las conversiones locales de la máquina desde la que se ejecuta el script para devolver una cadena con el formato definido. Los nombre del mes y del día de la semana, siguen los valores establecidos por la función setlocate(). El formato quedará definido por los siguientes valores:

Opción	Descripción
%a	Nombre del día de la semana abreviado (según idioma).
%A	Nombre del día de la semana completo (según idioma).

%b	Nombre del mes abreviado.
%B	Nombre del mes completo.
%c	Representación de fecha y hora (según idioma).
%d	Día del mes en formato 01 a 31.
%H	Hora como un núnero de 00 a 23.
%I	Hora como un núnero de 01 a 12.
%m	Mes como un número de 01 a 12.
%M	Minuto en número.
%S	Segundos en número.
%x	Representación por defecto de la fecha sin la hora.
%X	Representación por defecto de la hora sin la fecha.
%y	Año en número de 00 a 99.
%Y	Año con cuatro dígitos.

Ahora veremos el ejemplo anterior, para que aparezca en castellano.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 3 Práctica: 4
* Autor:  Fecha:
* Descripción:
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/
?>
<html>
<?php
// definimos la constante para el salto de línea HTML
define("B","<br>");

// definición de idioma
setlocale(LC_TIME,"es_ES");


echo B;
echo B;
echo B;
echo "PHP<br> \n";
echo "------------------<br> \n\n";
echo "Hoy es ", strftime("%A"), ", ", strftime("%d"), " de ",
strftime("%B"), " de ", strftime("%Y"),"\n";
echo B;
echo B;
// directamente desde date()
echo strftime("hoy es %A %d %B %Y ");
?>
</html>
```

Por último veremos la función checkdate(), que permite comprobar si una determinada combinación de día, mes y año representa una fecha válida. La función debe recibir tres parámetros numéricos correspondientes al mes, al día y al año y devuelve true si los parámetros resultan una fecha válida y false en cualquier otro caso.

La sintaxis es la siguiente: int checkdate(int mes, int dia, int anio);