```
<!-- Manual de PHP de WebEstilo.com -->
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
function Conectarse()
{
  if (!($link=mysql_connect("localhost","usuario","Password")))
  {
    echo "Error conectando a la base de datos.";
    exit();
  }
  if (!mysql_select_db("base_datos",$link))
  {
    echo "Error seleccionando la base de datos.";
    exit();
  }
  return $link;
}

$link=Conectarse();
echo "Conexión con la base de datos conseguida.<br>";

mysql_close($link); //cierra la conexion
?>
</body>
</html>
```

```
<?php
function Conectarse()
{
  if (!($link=mysql_connect("localhost","usuario","Password")))
  {
    echo "Error conectando a la base de datos.";
    exit();
  }
  if (!mysql_select_db("base_datos",$link))
  {
    echo "Error seleccionando la base de datos.";
    exit();
  }
  return $link;
}
?>
```

```
<!-- Manual de PHP de WebEstilo.com -->
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<?php
  include("conex.phtml");
  $link=Conectarse();
  $result=mysql_query("select * from prueba",$link);
?>
  <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
    <TR><TD> Nombre</TD><TD> Apellidos </TD></TR>
<?php

  while($row = mysql_fetch_array($result)) {
    printf("<tr><td> %s</td><td> %s </td></tr>", $row["Nombre"],$row["Apellidos"]);
  }
  mysql_free_result($result);
```

```
  mysql_close($link);
?>
</table>
</body>
</html>
```

```
<?php
  include("conex.phtml");
  $link=Conectarse();
  $result=mysql_query("select * from prueba",$link);
?>
```

Una de las herramientas mas importantes en cualquier lenguaje de programación son las funciones. Una función consiste en un conjunto de rutinas y acciones que a lo largo del script van a ser ejecutadas multitud de veces agrupados en una FUNCION y desde cualquier punto del script puede ser llamada y ejecutada. A su vez, esta función puede recibir parámetros externos de los cuales dependa el resultado de una función.

Las funciones deben ser colocadas siempre antes de realizar la llamada a la función (como es lógico). La sintaxis de una función es la siguiente:

function nombre(parámetros){
** instrucciones de la función**
}
para llamar a la función sería de la siguiente forma: **nombre(parámetros)**

Un ejemplo para entender el uso de funciones es el siguiente:
Crearemos una función que realice la suma de dos números y muestre el resultado

```
function sumar($sumando1,$sumando2){
  $ suma=$sumando1+$sumando2
echo $sumando1."+".$sumando2."=".$suma;
}

sumar(5,6)
```

Un hecho relevante que cabe destacar es que las variables que declaremos dentro de la función solo existirán o tendrán dicho valor dentro de la función.

Existen casos en los cuales no sabemos el número de parámetros que le pasaremos a la función y en estos casos debemos usar las funciones creadas al efecto como son:

func_num_args() Numero de parámetros que se le han pasado a la función
func_get_args() Devuelve un elemento de los que forman la lista de argumentos

INCLUSION DE CODIGO DESDE UN FICHERO

En ocasiones es necesario cargar el mismo archivo en diferentes scripts y de esa forma no repetir el mismo código en diversos scripts. Parece entonces razonable que debe existir algún método o forma de cargar algún archivo externo y de esa forma generar páginas web mas dinámicas.

A este efecto PHP dispone de dos instrucciones para poder cargar archivos:

require("archivo");
Cuando se incluye un archivo con require el interprete abandona el modo PHP y entra en modo HTML, una vez abierto el fichero lo incluye hasta su ultima línea y posteriormente abandona el modo HTML para volver a posicionarse en el modo PHP. Su principal diferencia con include() es que no permite la carga condicional.

include("archivo");
Cuando se incluye un archivo con require el interprete abandona el modo PHP y entra en modo HTML, una vez abierto el fichero lo incluye hasta su ultima línea y posteriormente abandona el modo HTML para volver a posicionarse en el modo

PHP. Permite la carga condicional, es decir, que podemos cargar un archivo o otro según si se cumple o no una condición.

CADENAS DE CARACTERES

En este capítulo se comentaran todos los pormenores relacionados con cadenas de caracteres, las funciones existentes, etc.

Cadenas de caracteres

Una cadena consiste en una secuencia de caracteres que se encuentran comprendidos entre unos delimitadores que pueden ser:
-Comillas simples **' '**
-Comillas dobles **" "**
-Documento incrustado **<<< >>>**

En caso de que se desee por ejemplo unas " " dentro de las comillas de la cadena de caracter es necesario realizar la acción que se denomina escapar un caracter que consiste en precederlo de una es decir ". Los caracteres especiales que pueden aparecer dentro de un documento con delimitación son:

Secuencia	**Significado**
n	Nueva línea
r	Retorno de carro
t	Tabulación horizontal
	Barra invertida
$	Signo de dólar
"	Comillas dobles
[0-7]{1,3}	Caracter ASCII que coincide con el numero octal
x[0-9A-Fa-f]{1,2}	Caracter ASCII que coincide con el numero hexadecimal

El caso de documento incrustado es diferente al de ambas comillas. Su sintaxis es la siguiente:

<<<Identificador
Cadena de caracteres
Identificador;

El resultado obtenido con documento incrustado es el texto mostrado igual que se ha introducido.

La función **chr(valor)** nos devuelve en una variable del tipo cadena el caracter de la tabla de códigos ASCII asociado a un valor que recibe como parámetro, el valor que se pase debe estar entre 0 y 255.
La función **ord(cadena)** nos devuelve un número entero que se corresponde con el código ASCII del primer caracter que recibe como parámetro.

Visualización de cadenas

- Echo: Es el modo de visualización mas empleado. Su sintaxis es la siguiente: echo "texto";
- Print: Es la mas sencilla de todas y se encarga de mostrar una cadena de caracteres sobre su salida estándar. No soporta ningún formato de salida y su sintaxis es: print(cadena);
- **Printf(formato,[valores]);** Su funcionamiento es el mismo que en el caso anterior. La única diferencia es que este soporta formatos de salida como su alineación (por defecto a la izquierda), valor numérico (numero mínimo de caracteres que deben mostrarse), numero de decimales y tipo de datos cuyas posibilidades son:

Símbolo	**Significado**
%	Representa el símbolo del porcentaje
b	El argumento se trata como nº entero y se representa en codificación binaria.
c	El argumento se trata como nº entero y se muestra el caracter cuyo código ASCII se corresponde con el valor.
d	El argumento se trata como nº entero y se representa en codificación decimal sin parte fraccionaria
f	El argumento se trata como un nº de tipo double y se representa como un decimal sin coma flotante
o	El argumento se trata como un nº entero y se representa en codificación octal

s	El argumento se trata y representa como una cadena de caracteres
x	El argumento se considera un nº entero y se representa en codificación hexadecimal en minúsculas
X	El argumento se considera un nº entero y se representa en codificación hexadecimal en mayúsculas

- **Sprintf(formato,[valores]);** su funcionamiento es idéntico a printf. Lo único que la cadena resultante de aplicarle un determinado formato se guarda en una variable.

Alteración del contenido

En ocasiones es necesario emplear dichas funciones para alterar el formato de salida de las cadenas. Las funciones empleadas para modificar dicho formato son:

- chop(cadena); Devuelve la cadena de caracteres con los caracteres de blanco y nueva línea eliminados
- ltrim(cadena); Elimina los blancos que aparecen a la derecha de una cadena de caracteres
- rtrim(cadena) ; Elimina los blancos que aparecen por la derecha en una cadena de caracteres
- trim(cadena); Elimina los blancos que aparecen a izquierda y derecha de la cadena de caracteres
- str_pad(cadena,longitud,relleno,lugar); Comprueba si la longitud es menor que el valor indicado, si es así añade los caracteres necesarios.El lugar de añadir puede ser:
str_pad_left añade por la derecha(opción por defecto), str_pad_right añade por la izquierda y str_pad_both añade por ambos extremos.
- str_repeat(caracter,numero_veces); Repite un caracter el numero de veces indicado
- strtolower(cadena); Pasa toda la cadena a letras minúsculas
- strtoupper(cadena); Pasa toda la cadena a letras mayúsculas
- ucfirst(cadena); Pasa a mayúscula el primer caracter de una cadena
- ucwords(cadena); Pone en mayúsculas el primer caracter de cada palabra de la cadena
- str_replace(subcadena1,subcadena2,cadena); Sustituye una palabra por otra dentro de una cadena
- strtr(cadena,originales,traducidos); Traduce ciertos caracteres .Ejemplo: $persona=strt($persona,"áéíóú","a,e,i,o,u"); de esta forma cambiaría todas las vocales con acento por vocales sin acento.
- substr_replace(cadena,nueva,comienzo,longitud); Sustituye una porción del contenido de una cadena

Acceso al contenido

- strlen(cadena); Indica el nº de caracteres de una cadena
- count_chars(cadena,modo); Numero de repeticiones de un caracter en una cadena. Los modos posibles son:

0->Matriz indexada con frecuencia de aparición de todos los caracteres del código ASCII
1->Matriz con caracteres ASCII con frecuencia mayor que 0
2->Matriz con caracteres que no aparecen en la cadena
3->Cadena con caracteres usados en el código ASCII
4->Cadena con caracteres no usados en el código ASCII

- substr_count(cadena,subcadena); Frecuencia de aparición de una cadena
-strchr(cadena,caracter); Devuelve la subcadena que comienza en la primera aparición del caracter indicado
- strstr(cadena,subcadena); Localiza subcadena dentro de la cadena original
- stristr(cadena,subcadena); Igual que la función anterior pero sin distinción entre mayúsculas y minúsculas
- strpos(cadena,subcadena); Primera ocurrencia de una cadena en otra
- strrpos(cadena,subcadena); Ultima ocurrencia de una cadena en otra
- ord(cadena); Devuelve el valor ASCII de un caracter
- substr(cadena,comienzo,longitud); Porción de texto que empieza en una posición y tiene una longitud
- strcmp(cadena1,cadena2); Compara dos cadenas siendo sensible a mayúsculas y minúsculas
- strcasecmp(cadena1,cadena2); Compara dos cadenas sin ser sensible a mayúsculas y minúsculas
- strncmp(cadena1,cadena2,tamaño); Compara los N primeros caracteres de una cadena
- strnatcmp(cadena1,cadena2); Sensible a mayúsculas y minúsculas. Compara dos cadenas.
- strnatcasecmp(cadena1,cadena2); No sensible a mayúsculas y minúsculas. Compara dos cadenas.
- chunk_split(cadena,longitud,separador); Coge una cadena de caracteres e introduce separadores a una distancia determinada. No modifica el original sino que es una función nueva.
- explode(separador,cadena,limite); Permite obtener una matriz de cadenas de caracteres extraídas del original.
- implode(separador,elementos); Junta en una cadena los elementos de una matriz usando como concatenación el separador pasado como parámetro.
- parse_str(cadena); Permite extraer y crear variables que forman parte de una cadena que se corresponde con un "query string" recibido de una URL.

***Apoyo a HTML**

- addcslashes(cadena,lista); Devuelve una cadena que tiene escapados todos los caracteres como parámetro.
- addslashes(cadena); Devuelve una cadena que tiene escapados todos los caracteres lógicos
- stripcslashes(); y stripslashes(); Reciben cadenas que pueden contener caracteres de de escapes y los desescapan
- quotemeta(cadena); Esacapa los caracteres especiales

- htmlspecialchars(cadena); Lleva a cabo conversiones como &->&"->&quot
- htmlentities(); Convierte todos los caracteres a entidades html. á pasa a ser á
- get_html_translation_table(htmlentities o html_specialchars); Obtiene la relación de traducción de cada caracter especial.
- array_flip(); Intercambia las claves por los valores en array asociativo.
- get_meta_tags(nombre_fichero,include_path); Devuelve todos los meta tags que contiene un HTML.
- strip_tags(cadena,mostrar_tags); Omite etiquetas PHP y HTML , lo de mostrar_tags son las cadenas HTML y PHP no deben ser omitidas en la lectura.
- n12br(cadena); Permite sustituir saltos de línea por
- parse_url(cadena_url); Devuelve una matriz asociativa con los siguientes campos:

Campo	Significado
scheme	Http
host	Ip o DNS
port	puerto
user	nombre usuario
password	contraseña
path	path completo al recurso
query	query string con datos al recurso
urldecode	decodifica la información
urlencode	Codifica la información

FECHAS

En este capítulo se estudiarán las funciones existentes en PHP para el empleo de fechas. Este tipo de funciones existen en la mayoría de lenguajes de programación y van orientadas a su obtención y representación en diferentes formatos.

El tiempo en cualquier lenguaje de programación se suele tomar con respecto al inicio de la "era UNIX" que es el 1 de enero de 1970 a las 00:00:00. La función más sencilla que se basa en esta marca de tiempo es la función time() cuyo valor devuelto es el numero entero que representa la marca de tiempo correspondiente al instante en que se ejecutó la función con respecto a la era unix.

En algunas aplicaciones es necesario poseer una marca de tiempo mas detallada y por ello usamos microtime() que devuelve una cadena de caracteres con los segundos y microsegundos.

En caso de que quisiéremos tener como valor de referencia la hora del ordenador desde el cual se ejecuta emplearíamos la función **gettimeofday()** en la cual pasaríamos como parámetro interno , **sec** (para saber los segundos), **usec** (microsegundos), **minuteswest** (nº segundos al oeste de greenwich) y **dstime** (tipo de corrección en horarios de verano e invierno).

Estas funciones citadas anteriormente son poco utilizadas ya que la existencia de otras funciones más completas, como por ejemplo la función **getdate()** que obtiene una matriz asociativa con la información de la fecha y hora del sistema. Los elementos de dicha matriz son:

Clave	Contenido
seconds	Numero de segundos de la hora actual
minutes	Numero de minutos de la hora actual
hours	Numero de horas de la hora actual
mday	Día correspondiente del mes
wday	Día de la semana en valor numérico(empezando por 0)
mon	Mes del año en valor numerico.Del 1 al 12.
year	Valor numérico del año
yday	Día del año en valor numérico
weekday	Cadena de caracteres que contiene el día de la semana(en ingles)
month	Cadena de caracteres que contiene el mes del año(en ingles)
0	Marca de tiempo obtenida por la función getdate()

Si no le pasamos ningún parámetro a la función entonces se considera la hora actual del sistema y si se recibe como parámetro un numero entero entonces lo convierte a la fecha correspondiente.

Otra función para obtener la hora es la función **localtime(marca_tiempo,tipo_matriz);** cuyos valores pasamos a comentar a continuación:

Índice	Clave	Contenido
0	**tm_sec**	Numero de segundos de la fecha indicada
1	**tm_min**	Numero de minutos de la fecha indicada
2	**tm_hour**	Numero de horas de la fecha indicada
3	**tm_mday**	Día correspondiente del mes
4	**tm_wday**	Día de la semana en valor numérico(empezando por 0)
5	**tm_mon**	Mes del año en valor numerico.Del 0 al 11.
6	**tm_year**	Valor numérico del año.(se ve afectado por el efecto 2000)
7	**tm_yday**	Día del año en valor numérico
8	**tm_isdst**	Indica si esta activado el efecto del cambio de hora.

Formatos de fechas

Las funciones vistas anteriormente nos permitían convertir el valor entero de la fecha en un valor mas fácilmente entendible, aunque para poder acceder a dicha información hay que pasar por el paso previo de obtener una matriz. Para evitar ese paso intermedio, PHP pone a tu disposición la función**date(formato,marca_tiempo);**

Esta función nos devuelve una cadena de caracteres que se corresponde con una fecha a la que se ha aplicado un determinado formato. Para definir el formato de la fecha se dispone de las siguientes opciones:

Opción	Descripción
a	Hace que en la hora aparezca la cadena am o pm
A	Hace que en la hora aparezca la cadena AM o PM
d	Día del mes con dos dígitos desde 01 a 31
D	Día de la semana como cadena de tres letras(en ingles).Ejemplo: "Mon"
F	Nombre del mes completo como una cadena de caracteres.Ejemplo: "March"
h	Hace que la hora aparezca en formato 01 a 12
H	Hace que la hora aparezca en formato 00 a 23
g	Hace que la hora aparezca en formato 1 a 12
G	Hace que la hora aparezca en formato 0 a 23
i	Hace que los minutos aparezcan en formato 00 a 59
j	Hace que el día aparezca en formato 1 a 31
l(L min)	Día de la semana completo.Ejemplo: Monday
L	Escribe 0 si no es año bisiesto y 1 si lo es
m	Hace que el mes aparezca en formato 01 a 12
M	Hace que el mes aparezca en formato 1 a 12
s	Hace que los segundos aparezcan en formato 00 a 59
S	Cadena de caracteres con el sufijo ordinal.Ejemplo: "th","nd".
t	Número de días del mes especificado de 28 a 31
U	Número de segundos desde el comienzo de la "era UNIX"
w	Número del día de la semana de 0 a 6
Y	Año con cuatro cifras
y	Año con dos cifras
z	Día del año de 0 a 365
Z	Obtiene la diferencia horaria en segundos con respecto al GMT

La función **strftime()** representa otra posibilidad para aplicar formatos a una fecha. Esta función utiliza las convenciones locales de la máquina desde la que se ejecuta el script para devolver una cadena con el formato definido en el idioma seleccionado. Su formato queda definido por los siguientes valores:

Opción	Descripción
%a	Nombre del día de la semana abreviado en el idioma actual
%A	Nombre del día de la semana completo en el idioma actual

%b	Nombre del mes abreviado en el idioma actual
%B	Nombre del mes completo en el idioma actual
%c	Representación de fecha y hora en el idioma actual
%d	Día del mes en formato 01 a 31
%H	Hora como numero de 01 a 12
%I	Hora como numero de 01 a 12
%j	Día del año como numero de 001 a 366
%m	Mes como numero de 01 a 12
%M	Minuto en numero
%p	am o pm según la hora dada
%S	Segundos en numero
%U	Numero de la semana del año como el primer domingo como primer día de la semana
%W	Numero de la semana del año como el primer lunes como primer día de la semana
%w	Día de la semana en numero de 0 a 6
%x	Representación por defecto de la fecha sin hora
%X	Representación por defecto de la hora sin fecha
%y	Año en numero de 00 a 99
%Y	Año en numero de cuatro cifras
%Z	Nombre o abreviatura de la zona horaria
%%	Caracter %

Estableciendo horas y fechas

Una vez conocida la forma de obtener la fecha actual, es necesario disponer de una forma de poder fijar una determinada hora para establecer por ejemplo la fecha de caducidad de una cookie, es decir, la forma de obtener una marca de tiempo correspondiente a una determinada hora.

Para ello PHP dispone de dos funciones que son **mktime()** y **gmmktime()** cuyo funcionamiento explicaremos a continuación:

La función **mktime(hora,minuto,segundo,mes,dia,año,[ajuste->0 horario de verano y 1 invierno]);** nos devuelve un valor entero que representa la marca de tiempo UNIX de una determinada fecha. Cada uno de los valores mencionados puede omitirse siempre y cuando a partir del valor omitido no se representen mas valores a su derecha.

La función **gmmktime()** funciona de la misma forma lo que considera que los parámetros representan una hora GMT.

La función **setlocale(categoria,pais);** nos permite establecer el idioma en los que aparecerán la fecha,hora,etc. Las categorías posibles son:

Opción	**Descripción**
LC_TYPE	Conversión de cadenas a configuración regional
LC_NUMERIC	Separadores numéricos
LC_TIME	Para aplicar formatos de fecha y hora con strftime()
LC_ALL	Todos los anteriores

Validación de fechas

Existen numerosas ocasiones en las que es necesario la creación de un sistema para comprobar si la fecha introducida por el usuario es valida o no. Para ello PHP nos brinda dos funciones capaces de realizar dicha comprobación:

- **checkdate(mes,dia,año);** Comprueba que la fecha introducida sea correcta .
- **strtotime(cadena_fecha);** Comprueba que la cadena de fecha sea correcta. Para ello la fecha debe estar en formato ingles, es decir, mm/dd/aa

ENTRADA Y SALIDA

Las operaciones de entrada/salida en PHP tienen una gran importancia en cualquier lenguaje de programación ya que no tiene sentido que un lenguaje de programación no pueda escribir, leer, actualizar datos de una base de datos, etc. En este capítulo nos centraremos básicamente en las operaciones de entrada y salida con archivos y posteriormente explicaremos las operaciones con bases de datos.

Supongamos que deseamos hacer una tienda de compra online. Imaginemos el gran esfuerzo que supondría tener que modificar todas las paginas HTML de aquellos productos en los cuales en la temporada de oferta su precio se viera afectado. La solución más primitiva para el almacenamiento de datos es un fichero de texto, el contenido del fichero de texto puede ser cualquiera.

¿Cómo abrimos un fichero?

Para abrir un fichero PHP pone a disposición una función. Su sintaxis es la siguiente:
fopen(fichero,modo); la ruta del fichero se indica en **fichero**, y **modo** determina los diferentes modos de lectura de un archivo:

Atributo	Efecto
r	Solo lectura
r+	Lectura y escritura
w	Sólo escritura. Borra el contenido anterior
w+	Lectura y escritura. Borra el contenido anterior.
a	Solo escritura. Conserva el contenido anterior.
a+	Lectura y escritura. Conserva el contenido anterior

La función fopen devuelve un manejador de fichero que es el que utilizaremos en las funciones relacionadas con la lectura y escritura de ficheros.

¿Cómo se recorre un fichero?

Para leer un fichero lógicamente la operación es muy sencilla. Consiste en ir leyendo todos los caracteres hasta llegar al EOF (end of file... final del archivo) el cual determina el final del texto.

Al igual que en las matrices los ficheros poseen un cursor interno que indica su posición actual con respecto a todo el texto.

Para comprobar si se ha llegado al final del archivo, PHP pone a nuestra disposición la función
feof(manejador_fichero); que se encarga de comprobar si la posición actual del fichero es la marca del final.

La función mas genérica de lectura es **fread(manejador_fichero,nº_bytes);** que se encarga de leer un numero determinado de bytes del fichero y devolvérnoslo en una cadena de caracteres.

En cambio, si lo que pretendemos es leer el fichero caracter a caracter debemos usar la función
fgetc(manejador_fichero); .

Otras de las funciones mas características son: **fgets(fichero,nº bytes);** que nos devuelve una cadena de caracteres con la información leída. **fgetss(manejador_fichero,nº bytes,[mostrar_tags]);** nos lee un fichero HTML omitiendo las etiquetas características del código, en mostrar tags debemos introducir las etiquetas que deseamos que se muestren.

Si al realizar la lectura lo que se pretenden leer los valores que siguen un determinado formato de entrada nos podemos ahorrar un gran trabajo usando la función **fscanf(manejador_fichero,formato,variables);** Esta función obtiene los datos de entrada de un fichero siguiendo un formato determinado. Posteriormente dicha información debe ser tratada por el programador.

La ultima función que vamos a estudiar para abrir ficheros es la función **file(nombre_fichero);** con esta función no es necesario usar las funciones fopen() y fclose() ya que se ejecutan de forma implícita.

¿Cómo se cierra un fichero?

Cerrar es la ultima operación que se debe realizar al manipular un fichero y permite cerrar la referencia que se hace al fichero en el script ejecutado. Para poder cerrar un fichero usamos la función **fclose(manejador_fichero);**

Escritura en ficheros

Para escribir en un fichero básicamente debemos realizar tres operaciones: abrir el fichero en modo escritura, realizar la operación de escritura realizada, cerrar el fichero. Las funciones que empleamos para escribir en un fichero son **fputs(manejador_fichero,cadena);** y **fwrite(manejador_fichero,cadena);** que nos permiten añadir una cadena de caracteres al texto anterior contenido en el texto.

Acceso directo en ficheros

Existen un conjunto de funciones que nos permiten situarnos en una determinada posición del fichero para leer a partir de ahí.

La función **fseek(manejador_fichero,posicion);** Nos permite empezar a leer un fichero a partir de una posición según se determine: **seek_set** (se toma con el principio del fichero), **seek_cur** (posición actual), **seek_end** (posición final).

Existen dos funciones básicas que nos posicionan al principio del fichero -> **rewind(manejador_fichero);** o que nos devuelven el valor actual de la posición-> **ftell(manejador_fichero);**

Funciones variadas para el manejo de ficheros

fpassthru(manejador_fichero); Nos muestra el contenido referenciado por manejador de fichero. Devuelve el numero de bytes mostrados si no se produce ningún fallo.
set_file_buffer(fichero,tamaño); Determina el tamaño del buffer de ese archivo que usara PHP.
readfile(fichero); Lee un fichero y lo muestra por el método de salida estándar.

OPERACIONES CON FICHEROS

En el desarrollo y administración de sitios webs resulta bastante habitual tener que acceder a ficheros del servidor para manipularlos. Por esta razón en este capítulo vamos a describir las funciones creadas en PHP para realizar dichas operaciones.

Cambio, creación y borrado de directorios

chdir(ruta_al_directorio); Nos permite cambiar el directorio activo a la ruta establecida como parámetro.
mkdir(ruta_al_directorio,permisos); Esta función crea un nuevo directorio en la ruta que hemos indicado, el segundo parámetro debe ser un numero octal y es por el que vienen determinados los permisos.
rmdir(ruta_directorio); Borra el directorio pasado como parámetro.

Procesamiento de los elementos de un directorio

Supongamos que queremos realizar una operación determinada como una búsqueda, visualización, etc sobre todos los ficheros de un directorio. PHP nos proporciona una solución a este problema: el manejador de directorios (representa una conexión lógica con un directorio determinado que permite leer la lista con los nombres de los elementos contenidos en el directorio actual).

La función empleada para abrir un directorio es **opendir(ruta);** cuya función como ya se ha comentado es abrir el directorio de la ruta especificada. Una vez se ha ejecutado opendir() podemos realizar tres operaciones:

La función **readdir(manejador);** nos devuelve una cadena con el nombre del siguiente elemento del directorio, ya sea un subdirectorio o un fichero.
La función **rewinddir(manejador);** procesa un directorio y sitúa el puntero interno en el primer directorio.
La función **closedir(manejador);** finaliza el tratamiento de entradas de directorio.

La clase dir

PHP nos proporciona una pseudoclase predefinida para el manejo de ficheros. Esta clase no aporta ninguna funcionalidad que no hayamos visto hasta este punto pero recopila todas las funciones a partir de una sola. Para poder trabajar con un directorio primero hay que crear una instancia de clase dir por medio de su constructor.
$directorio=dir(ruta_directorio);

Este objeto cuenta con 3 métodos y 2 propiedades(las propiedades sólo de consulta por lo que no pueden ser modificadas. Los métodos empleados son **read(),rewind() y close()**

Copiado,borrado y renombrado de ficheros

copy(fichero_origen,fichero_destino); Realiza una copia de un fichero.
unlink(nombre_fichero); Elimina el fichero.
rename(nombre_antiguo,nombre_nuevo); Renombra el fichero pasado como parámetro.

Atributos de ficheros y directorios

Los ficheros y directorios poseen una serie de características propias denominadas atributos. PHP pone a nuestra disposición un conjunto de funciones que nos permitirán obtener información sobre los archivos o carpetas.

La función **file_exists(elemento);** Comprueba que el elemento pasado como parámetro exista.

filesize(nombre_fichero); nos informa sobre el tamaño del fichero en bytes.

La función fileatime(fichero); nos informa sobre el ultimo acceso al fichero.

La función filemtime(fichero); nos informa sobre la ultima modificación del fichero.
La función filectime(fichero); nos informa sobre el último cambio al fichero.
La función filetype(fichero); nos devuelve el tipo de elemento que estamos tratando. Los resultados posibles que puede devolver son:

Resultado	Significado
block	Dispositivo de bloques
char	Caracteres
dir	Directorio
fifo	FIFO
file	Fichero
link	Enlace
unknown	Desconocido

Chmod(elemento_directorio,permisos); recibe como parámetro el elemento y los permisos que deseamos otorgarle a dichos elementos

EL LENGUAJE SQL Y PHP

En este capítulo nos dedicaremos a explicar el lenguaje SQL ya que posteriormente lo usaremos mucho en las conexiones de PHP con MySQL.

Creación y modificación de Tablas en SQL

MySQL esta organizado a partir de tablas y dichas tablas contienen campos. Cada campo es capaz de contener un tipo de dato. Los tipos de datos que es posible crear en el lenguaje SQL son:

Tipo	Descripción
Tinyint[Unsigned]	Entero de 0 a 255 o de -128 a 128
Smallint[Unsigned]	Entero de 0 a 65535 o de -32768 a 32768
Int o Integer	Entero normal.Rango de -2147483648 a 214783648
Float[(M,D)]	Número de coma flotante de simple precisión si no se pasa ningun argumento M es el nº de digitos y D el nº de decimales
Double [(M,D)]	Número de coma flotante de doble precision. Siempre dispone de signo M y D
Decimal [(M [,D])]	Número almacenado como cadena de caracteres M es el número total de dígitos y D el nº de decimales
Date	Tipo fecha.Admite formatos "AAAA-MM-DD" o "AA-MM-DD" o "AAMMDD"
Time	Tipo hora.Admite formato "HH:MM:SS" o "HHMMSS" o "HHMM" o "HH"
Char(longitud)	Cadena de caracteres de la longitud indicada.Se reserva el espacio en caracteres aunque no se usen
Varchar(longitud)	Cadena de caracteres de la longitud indicada que se almacena con su ocupacion.Máxima longitud: 255 caracteres

Blob	Tipo destinado a almacenar bits sin intrepretar.Se usa para almacenar texto mas largo de 255 caracteres.Diferencia mayúsculas de minúsculas.
Text	Tipo destinado a almacenar bits sin intrepretar.Se usa para almacenar texto mas largo de 255 caracteres.No diferencia mayúsculas de minúsculas.

Para crear una tabla usaremos la siguiente sintaxis:

```
CREATE TABLE Nombre_tabla
(Campo1  Tipo_dato  Not Null,
  Campo2 Tipo_dato,
  PRIMARY KEY (Campo3));
```

Esto nos crearía una tabla con 3 campos de los cuales Campo3 es un valor único, es decir, que no puede ser sobreescrito.

Para eliminar una tabla usaremos:

```
DROP TABLE Nombre_tabla;
```

Para modificar la estructura de la tabla usaremos la siguiente sintaxis:

```
ALTER TABLE Nombre_tabla
[ADD Nombre_atributo Definición]     //Añadiría un nuevo campo
[CHANGE AntiguoNombreAtributo NuevoNombreAtributo Definición]     //Cambiaría un campo
[DROP NombreAtributo];     //Borraría un campo
```

Los índices son una estructura de acceso que permiten organizar los datos contenidos en una tabla. Para crear un índice usaríamos la siguiente sintaxis:

```
CREATE [UNIQUE] INDEX NombreIndice
   ON Tabla (Campos);
```

Manipulación de datos

-Inserción de datos

Para insertar datos en la tabla se realiza mediante el comando insert y su sintaxis es la siguiente:

```
INSERT INTO NombreTabla [Campo1,Campo2...CampoN] VALUES (Valor1,Valor2...ValorN);
```

- **Consultas de datos**

Para esta acción usamos el comando **SELECT** y la sintaxis es la siguiente:

```
SELECT ([*]/[Atributos]) FROM Tabla/s [WHERE ListaCondiciones] [GROUP BY Campo] [HAVING ListaCondiciones]
[ORDER BY Campo]
```

Existen un conjunto de funciones dentro de las consultas de datos que nos permiten obtener información o realizar operaciones con respecto a las filas. Las funciones son:

función	**Descripción**
COUNT(*/DISTINCT Campo)	Cuenta el numero de filas
SUM(Campo)	Suma los valores del atributo indicado
AVG(Campo)	Obtiene la media aritmética del atributo
MAX(Campo)	Obtiene el valor máximo del atributo
MIN(Campo)	Obtiene el valor mínimo del atributo

- **Eliminación de datos**

Para eliminar datos usamos la sentencia **DELETE** cuya sintaxis es la siguiente:

```
DELETE FROM NombreTabla [WHERE Condición];
```

CONEXION CON MYSQL

Una vez que ya hemos explicado un poco por encima todas las operaciones posibles y lógicas que podemos hacer con una base de datos en el lenguaje SQL, llega el momento de combinarlo con la potencia de PHP y para ello usaremos el programa MySQL.

MySQL es uno de los gestores de bases de datos mas utilizados en entornos en los cuales se emplea PHP ya que PHP dispone de numerosas funciones que se compaginan perfectamente con MySQL. La forma genérica de obtener información de tablas en Mysql es la siguiente:

- Conexión con el gestor.
- Preparación de la consulta SQL.
- Ejecución de la consulta.
- Procesamiento del resultado obtenido en el cursor.
- Liberación de recursos (esta es opcional, aunque es recomendable).
- Cierre de la conexión con el gestor.

Para realizar estas y otras muchas mas cosas disponemos de las siguientes funciones:

Función	**Descripción**
mysql_connect("host","usuario","password")	Establece la conexión con el servidor. Recibe el host y el usuario y contraseña con el que debe conectar.
mysql_select_db("base de datos",conexión)	Selecciona la base de datos sobre la cual se va a trabajar
mysql_query(consulta,conexión)	Ejecuta la consulta SQL indicada como primer parámetro.
mysql_num_fields(cursor)	Devuelve el numero de atributos que figuran en el cursor que se le pasa como parámetro y en el que se almacena el resultado de la consulta
mysql_fetch_row(cursor)	Avanza a la siguiente posición de la fila en cursor. Devuelve un array que contiene en sus celdas cada uno de los valores de los atributos de la fila.
mysql_free_result(cursor)	Libera los recursos asociados al cursor.
mysql_close(conexion)	Cierra la conexion establecida con mysql_connect.

Una de las ventajas que proporciona la altísima integración que PHP y MYSQL tienen es la existencia de funciones que permiten al programador acceder a las diferentes estructuras que conforman la base de datos. Algunas de las funciones son:

Función	**Descripción**
mysql_list_dbs(conexion)	Devuelve en un cursor los nombres de las bases de datos disponibles en el servidor al que se haya conectado con mysql_connect
mysql_list_tables(base_datos,conexion)	Devuelve en un cursor los nombres de las tablas disponibles en la base de datos.
mysql_tablename(cursor,numero_fila)	Devuelve el nombre de la tabla o base de datos en la que esta el cursor indicado
mysql_field_name(cursor,numero_col)	Devuelve el nombre del campo cuyo índice se pasa como segundo parámetro
mysql_field_type(cursor,numero_Col)	Devuelve el tipo del campo cuyo índice se pasa como segundo parámetro
mysql_field_len(cursor,numero_col)	Devuelve la longitud del campo cuyo índice se pasa como segundo parámetro
mysql_field_flags(cursor,numero_col)	Devuelve una serie de indicativos correspondientes a características del atributo cuyo índice se pasa como segundo parámetro
mysql_affected_rows(conexion)	Devuelve el numero de filas afectadas por una actualización o borrado
mysql_change_user(usuario,password)	Cambia de usuario
mysql_create_db(basedatos)	Crea una base de datos con el nombre pasado por parámetro
mysql_drop_db(basedatos)	Elimina la base de datos pasada por parámetro
mysql_insert_id(cursor)	Devuelve el valor generado para un AUTOINCREMENT

SESIONES

Generalmente una web se compone de una serie de páginas entre las que existe alguna relación. Un ejemplo claro es una página en la cual es necesario estar registrado para poder acceder a ellas ya que en función de la categoría del usuario nos permitirá acceder a unas secciones o otras. En estas aplicaciones será necesario ir comprobando los permisos de usuario y para ello usamos un elemento en PHP denominado "**sesiones**".

Una sesión se inicia cuando un usuario entra en la aplicación web y finaliza cuando el usuario abandona la aplicación (mas adelante comprenderemos lo de "abandonar aplicación").

Durante todo ese tiempo podemos manipular una serie de variables que se inician al iniciar la sesión y mantener un tipo de información común entre todas las páginas (en el caso de el usuario registrado seria los privilegios que posee).

Para mantener esta información constante es necesario que los datos se guarden en un fichero ya sea en el cliente (cookies) o en el servidor (en caso de que tenga desactivado las cookies).

Para el problema que consiste en diferenciar los diferentes usuarios existe una solución muy básica que consiste en un identificador de sesión diferente en cada caso.

Este identificador de sesión debe ser enviado de una pagina a otra para mantener la sesión activa(a menos que en la configuración del servidor tengamos activada la opción session_trans_id) y también es necesario pasar el identificador de sesión en los formularios como un campo **HIDDEN**.

Ejemplos:

- Hipervínculo
<a href="pagina.php?<? =SID ?>">Entrar

-Formulario
<input type="hidden" name="session_name()" value="SID">

Funciones de gestión de sesiones

función	**Significado**
session_start();	Si es la primera solicitud genera un identificador de sesión aleatorio cuyo nombre será sess_IDsesión; si es otra solicitud continua la sesión iniciada anteriormente.
session_destroy();	Elimina todos los datos asociados con una sesión, borra el archivo en el servidor pero no borra la cookie.
session_register(nombre);	Recibe como parámetro una serie de nombres de variable globales y los registra como variables de sesión en el fichero del servidor
session_unregister(nombre);	Eliminamos la variable global introducida y se elimina el contenido de esta variable en el fichero del servidor.Sin pasar el parámetro nombre eliminaremos todas las variables de la sesión.
session_is_registered(nombre);	Devuelve true en caso de que en la sesión se encuentre registrada una variable con dicho nombre.
session_unset();	Dejamos sin ningún valor asignado a todas las variables de la sesión
session_id([nombre]);	Si no le proporcionamos ningún parámetro nos da el identificador de sesión; si le proporcionamos el parámetro nombre cambia el valor del identificador por el parámetro nombre.
session_name([nombre]);	Si se invoca sin parámetro devuelve el nombre de la variable interna que tiene el id de sesiones; si se pasa parámetro cambia el nombre de la sesión.
session_get_cookie_params();	Permite definir nuevos valores para los parámetros de configuración de las cookies.Para que el cambio sea permanente hay que invocar el cambio en todos los documentos.
session_cache_limiter([cache_limiter]);	Si se le proporciona valor modifica el valor por defecto en cambio sino se muestra el caché que tiene por defecto.
session_encode();	Devuelve una cadena con la información de una sesión, después de usar esta función la información de la sesión queda actualizada
session_decode(cadena);	Descodifica la cadena que recibe como parámetro y que contiene la info de sesión, después de usar esta función se actualiza la info de sesión.

session_save_path([path]);	Devuelve el camino al directorio donde se guardan los ficheros asociados a la sesión.El efecto solo dura en el script actual.
session_module_name([modulo]);	Devuelve el nombre del modulo que se usa para realizar la gestión de sesiones. Cuando se invoca un parámetro se usa como nuevo gestor de sesiones.

session_set_save_handler(open,close,read,write,destroy,gc);

Permite definir su propio manejador para almacenar la información asociada con una sesión.De esta forma los datos pueden ser metidos en una BD en vez de en un fichero. Tenemos que pasarle como parámetro toda la información necesaria para crear y destruir sesiones.