# COMILLAS

Existen dos tipos de comillas: dobles y sencillas. Cuando es preciso anidar comillas deben utilizarse tipos distintos para las exteriores y las interiores.

Para que una etiqueta echo interprete unas comillas como texto y no como el texto y no como final de cadena es necesario anteponerle un signo de barra invertida. En ningún caso esta permitido ni con echo ni con print esta permitido sustituir las comillas exteriores por \”.

# CONCATENACION DE CADENAS

A las variables tipo **cadena** se les puede asignar los valores de dos formas:

Escribiendo el *contenido entre comillas*:

**$var="***Texto del contenido***";**

y por medio de la *sintaxis de documento incrustado* que es lasiguiente:

**$var= <<< *EOD***

*... contenido de la cadena...*

*... puede ir ....*

*.. en varias líneas...*

***EOD*;**

Donde **EOD** es una *palabra cualquiera* que debe repetirse *exactamente igual* al final de la instrucción.

*El nombre de la variable, el signo igual que la precede, los tres símbolos < y el EOD deben escribirse en la misma línea.*

El *cierre* de la instrucción debe hacerse –siempre– escribiendo el EOD en una nueva línea –independiente– que ahora sí tiene que llevar el *punto y coma* que indica a PHP el final de una instrucción.

# CONSTANTES PREDIFINADAS

PHP dispone de algunas constantes predefinidas que no requieren la instrucción:

define("Nombre","Valor")

Algunas de ellas son estas:

\_\_FILE\_\_

Recoge el nombre del fichero que se está ejecutando y la ruta completa de su ubicación en el servidor.

\_\_LINE\_\_

Recoge el número de línea (incluidas líneas en blanco) del fichero PHP cuyos scripts está interpretando.

Puede resultar muy útil para depurar programas escritos en PHP.

PHP\_OS

Recoge información sobre el Sistema Operativo que utiliza el servidor en el que se está interpretando el fichero.

PHP\_VERSION

Recoge la versión de PHP que está siendo utilizada por el servidor.

# CONSTANTES Y VARIABLES

es un valor –un número o una cadena– que no va a ser modificado a lo largo del proceso de ejecución de los scripts que contiene un documento. En PHP las constantes se definen mediante la siguiente instrucción:

define("Nombre","Valor")

No es necesario escribir entre comillas los valores de las constantes cuando se trata de constantes numéricas.

## FUNCION ECHO

Si se realizan operaciones aritméticas con constantes tipo cadena, y su valor comienza por una letra, PHP les asigna valor

cero. El punto entre caracteres numéricos es considerado como separador de parte decimal. Mediante una sola instrucción echo se pueden presentar (en la ventana del navegador del cliente) de forma simultánea varias cadenas de variables. Basta con ponerlas una a continuación de otra utilizando una coma como separador entre cada una de ellas.

Si en vez de utilizar la coma usáramos un punto (el concatenador de cadenas) conseguiríamos el mismo resultado.

## FUNCION PRINT

Las instrucciones print también permiten concatenar cadenas en una misma instrucción.

En este caso solo es posible usar el punto como elemento de unión. Si

pusiéramos comas –como hacíamos con echo– PHP nos daría un error.

# VARIABLES Y SUS AMBITOS

La sintaxis es esta:

$variable=valor;

## AMBITO DE LAS VARIABLES

Si una variable es definida dentro

de una función sólo podrá ser utilizada dentro esa función. Si definimos dos variables con el mismo nombre, una dentro de una función y otra fuera, PHP las considerará distintas. La función utilizará –cuando sea ejecutada– sus propios valores sin que sus resultados modifiquen la variable externa. Las funciones pueden utilizar valores de variables externas a ellas pero ello requiere incluir dentro de la propia función la siguiente instrucción:

global nombre de la variable;

Por ejemplo: global $a1;

# VARIABLES PREDEFINIDAS

Las variables superglobales se introdujeron en PHP a partir de la versión 4.1.0 y no existían con anterioridad.

## Variables de sesión

Las identificaremos por los nombres $\_SESSION o por $HTTP\_SESSION\_VARS. Este tipo de variables las utilizaremos cuando hagamos mención al uso de sesiones.

## Variables del método POST

Las identificaremos por los nombres $HTTP\_POST\_VARS o por $\_POST.

Este tipo de variables –que utilizaremos con frecuencia– recogen la información que se envía desde el cliente para ser utilizada por el servidor.

## Variables del método GET

Las identificaremos por los nombres $HTTP\_GET\_VARS o por $\_GET Son muy similares a las anteriores. La existencia de los dos tipos se justifica porque también existen dos tipos de métodos (maneras) de enviar datos desde el cliente hasta el servidor.

## Variables de transferencia de ficheros

Las identificaremos por el nombre $HTTP\_FILES\_VARS o por $\_FILES.

Para el caso de transferencia de ficheros desde el cliente al servidor –«subir ficheros»– es necesario un procedimiento distinto de los anteriores. Será en este caso cuando se utilicen variables de este tipo.

## El tipo GLOBALS

A diferencia de las anteriores, las variables de este tipo disponen de una sintaxis única –$GLOBALS– sin que quepa ninguna otra opción. Su finalidad es recoger en una tabla los nombres de todas las variables establecidas como globales –en cada momento– así como sus valores. Si observas la tabla que tienes aquí a la derecha, quizá te sorprenda leer nombre o página. ¿De donde han salido esos valores?. Bueno... en esta página utilizamos scripts PHP y esos son los nombres de unas variables que hemos incluido en esos script.

## Variables estáticas

Para poder conservar el último valor de una variable definida dentro de una función basta con definirla como estática. La instrucción que permite establecer una variable como estática es la siguiente:

static nombre = valor;

P. ej: si la variable fuera $a y el valor inicial asignado fuera 3 escribiríamos:

static $a=3;

La variable conservará el último de los valores que pudo habérsele asignado durante la ejecución de la función que la contiene. No retomará el valor inicial hasta que

se actualice la página. En sucesivas llamadas a la función este valor se conserva en la variable

# TIPOS DE VARIABLES

Las variables en PHP pueden ser de

tres tipos:

– Enteras (tipo Integer)

– De coma flotante (tipo Double)

– Cadenas (tipo String)

PHP dispone de la función:

gettype(variable)

que devuelve una cadena de caracteres indicando el tipo de la variable que contiene.

La cadena devuelta por esta función puede ser: Integer, double o string.

## Forzado de tipos

Al asignar un valor a una variable, se puede forzar su tipo de la siguiente forma. Si deseamos que la variable pase a ser tipo de double basta con anteponer a su valor –entre paréntesis– tal como se indica una de las expresiones: (double), (real) o (float).

$a=((double)45); o

$a=((float)45); o

$a=((real)45);

$b=((integer)4.5); o

$b=((int)45);

La forma más aconsejable de forzado de tipos en variables que ya estuvieran definidas previamente, es el uso de la siguiente instrucción: settype (var,tipo)

Un ejemplo podría ser este:

settype($a,'integer')

La ejecución de la instrucción settype devuelve (da como resultado) un valor que puede ser: true o false (1 ó 0) según la conversión se haya realizado con

éxito o no haya podido realizarse. Al operar con una cadena lo hace

como si se tratara de un entero. Si hay caracteres numéricos al comienzo, los extrae (hasta que aparezca un punto o un carácter no numérico) y los opera como un número entero.

Si una cadena no comienza por un carácter numérico PHP la operará tomando su valor numérico como CERO.

# OPERACIONES ARITMETICAS

## Sintaxis de print y echo

Si queremos encadenar en una sola instrucción –echo ó print– el resultado de una operación junto con otras variables (o cadenas) es imprescindible poner entre paréntesis las instrucciones de la operación. Esta norma, solo tiene dos excepciones: en caso de que el print solo contenga la propia operación o cuando utilicemos echo y el separador sea una coma.

ejemplo16.php

## Operaciones aritméticas

Operación Sintaxis A B Resultado

Suma $a+$b 12 -7.3 4.7

Diferencia $a-$b 12 -7.3 19.3

Producto $a\*$b 12 -7.3 -87.6

Cociente $a/$b 12 -7.3 -1.64383561644

Cociente entero (int)($a/$b) 12 -7.3 -1

Resto de la división $a%$b 12 5 2

Potencias a b pow($a,$b) 12 5 248832

Raíz cuadrada Sqrt($a) 12 3.46410161514

Raíz cuadrada Sqrt($a) -7.3 NAN

Raíz enésima pow($a,(1/$b) 12 3; 2.28942848511

Redondeos

tipo Sintaxis A Resultado

Parte entera (int)$a 12 12

Parte entera (int)$a -7.3 -7

Parte entera (int)$a -13.8546 -13

<?

# definamos dos variables numéricas asignandoles valores

$a=23; $b=34;

/\* hagamos una suma y escribamos directamente los resultados

utilizando las instrucciones print y echo

con todas sus posibles opciones de sintaxis \*/

print("La suma de $a + $b es: " . $a . "+" . $b . "=" . ($a+$b)."<br>");

print "La suma de $a + $b es: " . $a . "+" . $b . "=" . ($a+$b) ."<BR>";

print ("La suma de $a + $b es: " . $a . "+" . $b . "=" . ($a+$b) ."<BR>");

echo "La suma de $a + $b es: " . $a . "+" . $b . "=" . ($a+$b) ."<BR>";

echo "La suma de $a + $b es: " , $a , "+" , $b . "=" , ($a+$b) ."<BR>";

echo "La suma de $a + $b es: " , $a , "+" , $b , "=" , $a+$b ,"<BR>";

# guardemos ahora el resultado de esa operación en una nueva variable

$c=$a+$b;

/\*ahora presentemos el resultado utilizando esa nueva variable

adviertiendo el la salida \*/

print ("Resultados recogidos en una nueva variable<br>");

print "La suma de $a + $b es: " . $a . "+" . $b . "=" . $c ."<BR>";

print ("La suma de $a + $b es: " . $a . "+" . $b . "=" . $c ."<BR>");

echo "La suma de $a + $b es: " . $a . "+" . $b . "=" . $c ."<BR>";

echo "La suma de $a + $b es: " , $a , "+" , $b . "=" , $c ."<BR>";

echo "La suma de $a + $b es: " , $a , "+" , $b , "=" , $c ,"<BR>";

/\* modifiquemos ahora los valores de $a y $b comprobando que el cambio

no modifica lo contenido en la variable $c \*/

$a=513; $b=648;

print ("<br> C sigue valiendo: " . $c ."<br>");

# experimentemos con los paréntesis en un supuesto de operaciones combinada

# tratemos de sumar la variable $a con la variable $b

# y multiplicar el resultado por $c.

# Si escribimos print($a+$b\*$c) nos hará la multiplicación antes que la suma

print "<br>No he puesto paréntesis y el resultado es: ".($a+$b\*$c);

# Si escribimos print(($a+$b)\*$c) nos hará la suma y luego multiplicará

print "<br>He puesto paréntesis y el resultado es: ".(($a+$b)\*$c);

?>

PHP permite utilizar un tipo de variables –las numéricas– cuyos valores puedan ser operados de la misma forma que se hace con los números en la vida cotidiana. Los resultados de las operaciones pueden utilizarse de forma directa o ser recogidos en una nueva variable. Aquí a la derecha tienes un ejemplo de ambas opciones. Si asignamos a una nueva variable el resultado de una operación el valor contenido en ella no se modifica, aunque cambien los de las variables que intervinieron su creación.

### Sintaxis de print y echo

Si queremos encadenar en una sola instrucción –echo ó print– el resultado de una operación junto con otras variables (o cadenas) es imprescindible poner entre paréntesis las instrucciones de la operación. Esta norma, solo tiene dos excepciones: en caso de que el print solo contenga la propia operación o cuando utilicemos echo y el separador sea una coma.

## Redondeo de resultados

PHP tiene tres opciones de redondeo:

Redondeo por defecto

floor($z)

Redondeo por exceso

ceil($z)

Redondeo tradicional

round($z)

# NUMEROS ALEATORIOS

Bastaría con añadir los valores de los extremos de ese intervalo como parámetros de la función.

La sintaxis sería esta:

rand(extremo inferior , extremo superior)

y para la función mejorada

mt\_rand(extremo inferior , extremo superior)

# OPERADORES DE INCREMENTO

## Operadores de pre incremento

++$A y --$A

Este operador incrementa el valor de la variable en una unidad (+1 o -1) antes de ejecutar el contenido de la instrucción.

$A+=n y $A-=n

Este operador incrementa el valor de la variable en n unidades (+n o -n) y luego ejecuta el contenido de la instrucción.

## Operadores de post-incremento

$A++ y $A--

Cuando los operadores ++ o – están situados a la derecha de la variable los incrementos no se producen hasta que se ejecute la instrucción siguiente.

# OPERADORES DE COMPARACION

Hay un orden de prioridad que es el siguiente: NOT, &&, ||, AND, XOR y, por último, OR.

# OPERACIONES CON CADENAS

La concatenación de cadenas Para concatenar (unir), en una sola, varias porciones de texto hemos venido utilizando –en las instrucciones print y echo– un punto (.).

El operador .

Este punto es un elemento muy importante que, además de la forma que hemos visto en las páginas anteriores, tiene los siguientes usos:

Unir dos cadenas y recogerlas en una variable Con la sintaxis:

$a="cad1" . "cad2";

o mediante

$a= $b . $c

Añadir contenidos a una variable tipo string Si utilizamos una sintaxis como esta:

$a .="cad1"

o de este otro tipo

$a .=$b

# CADENAS ALFANUMERICAS

printf(cadena de formato,variable1,variable2,..)

## Cadena de formato

Dentro de la cadena de formatos deben repetirse tantos formatos como variables se pretenda manejar

"%[rell1][alin1][anc1][prec1][tipo1][sepa1]%[rell1][alin1][anc1][prec1][tipo1][sepa1]"

Hemos de mencionar aquí los separadores ya que no fueron mencionados en la página anterior

Se puede introducir una cadena de separación al final de una cadena de formato que puede hacer, entre otras, función de separación entre dos cadenas. Por ejemplo,

printf("%'\*15.2f Euros",1475.875) nos devolvería:

\*\*\*\*\*\*\*\*1475.88 Euros

La función printf() permite presentar varios valores o variables con distintos formatos utilizando

la sintaxis que se indica más arriba.

Este ejemplo :

printf("%'\*15.2f Euros=%'\*18.0f Pesetas",1475.875,1475.875\*166.386)

devuelve como resultado:

\*\*\*\*\*\*\*\*1475.88 Euros=\*\*\*\*\*\*\*\*\*\*\*\*245565 Pesetas

Existe otra función PHP con características muy similares a la anterior. Se trata de sprintf().

La sintaxis es idéntica sprintf (cadena de formato, variable1,variable2, ...) y su única diferencia con printf es que, mientras que printf() imprime las variables utilizando el formato indicado, sprintf() puede guardar en una nueva variable la cadena resultante de la aplicación del formato.

## Funciones de cadenas

Algunas de las funciones que permiten manejar los formatos de las cadenas de caracteres son estas:

chr(n)

Devuelve el carácter cuyo código ASCII es n.

ord(cadena)

Devuelve el código ASCII del primero de los caracteres de la cadena.

strlen(cadena)

Devuelve la longitud (número de caracteres) de la cadena. Los espacios son considerados como un caracter más.

strtolower(cadena)

Cambia todos los caracteres de la

cadena a minúsculas.

strtoupper(cadena)

Convierte en mayúsculas todos los

caracteres de la cadena.

ucwords(cadena)

Convierte a mayúsculas la primera letra de cada palabra.

ucfirst(cadena)

Convierte a mayúsculas la primera letra de la cadena y pone en minúsculas todas las demás.

ltrim(cadena)

Elimina todos los espacios que pudiera haber al principio de la cadena.

rtrim(cadena)

Elimina todos los espacios que existieran al final de la cadena.

trim(cadena)

Elimina los espacios tanto al principio como al final de la cadena

chop(cadena)

Elimina los espacios al final de la cadena. Es idéntica a rtrim.

**Advertencia**

Tanto trim, como ltrim y rtrim eliminan, además de los espacios, las secuencias: \n, \r, \t, \v y \0; llamadas también caracteres protegidos.

substr(cadena,n)

Si el valor de n es positivo extrae todos los caracteres de la cadena a partir del que ocupa la posición enésima a contar desde la izquierda. Si el valor de n es negativo serán extraídos los n últimos caracteres contenidos en la cadena.

substr(cadena,n,m)

Si n y m son positivos extrae m caracteres a partir del que ocupa la posición enésima, de izquierda a derecha. Si n es negativo y m es positivo extrae m (contados de izquierda a derecha) a partir del que ocupa la posición enésima contada de derecha a izquierda. Si n es positivo y m es negativo extrae la cadena comprendida entre el enésimo carácter (contados de izquierda a derecha) hasta el emésimo, contando en este caso de derecha a izquierda Si n es negativo y m también es negativo extrae la porción de cadena comprendida entre el emésimo y el enésimo caracteres contando, en ambos casos, de derecha a izquierda. Si el valor absoluto de n es menor que el de m devuelve una cadena vacía.

strrev(cadena)

Devuelve la cadena invertida

str\_repeat(cadena, n)

Devuelve la cadena repetida tantas veces como indica n.

str\_pad(cad, n, rell, tipo)

Añade a la cadena cad los caracteres especificados en rell (uno o varios, escritos entre comillas) hasta que alcance la longitud que indica n (un número) El parámetro tipo puede tomar uno de estos tres valores (sin comillas):

STR\_PAD\_BOTH (rellena por ambos lados)

STR\_PAD\_RIGHT (rellena por la derecha)

STR\_PAD\_LEFT (rellena por la izquierda).

Como complemento a las descritas en la página anterior, añadimos aquí algunas otras funciones PHP que también permiten manejar cadenas de caracteres.

AddSlashes(cadena)

Inserta el carácter \ delante los siguientes: ", ', \ y NUL (el bit nulo).

stripslashes(cadena)

Quita las marcas añadidas a una cadena con la función AddSlashes().

chunk\_split(cad, n, sep )

Devuelve la cadena (cad) después de haberle insertado, cada n caracteres, la cadena indicada en el parámetro sep. Si no se indica sep PHP pondrá un espacio. Si no se establece el parámetro n insertará el separador cada 76 caracteres. Esta función coloca siempre un separador al final de la cadena.

parse\_str(cadena)

Devuelve las variables –con su valor– indicadas dentro de la cadena

Dentro de la cadena cada variable se denomina con un nombre que va seguido de un signo igual. Los espacios se señalan con el signo + y los separadores de variables son signos &

explode(sep, cad,n)

Devuelve un array cuyos elementos contienen cada una de las porciones de la cadena (cad) comprendidas entre dos de los caracteres señalados como (sep) hasta el máximo de porciones señaladas (n).

Los caracteres separadores no son incluídos en las cadenas resultantes. Si no se indica la cantidad de porciones, será fraccionada toda la cadena. Si se indica número, el último trozo contendrá toda la cadena restante.

implode(sep, array)

Devuelve una cadena formada por todos los elementos del array separados mediante los caracteres indicados en sep.

join(sep, array)

Es idéntica a implode.

strtok(cad , sep)

Esta función divide la cadena cad en trozos delimitados por el separador

que se indica en sep. Cuando se invoca la primera vez

–extrae el primer trozo– debe llevar las sintaxis strtok(cadena,sep).

Al invocarla sucesivamente, se escribe solo strtok (" ") e irá

recogiendo de forma secuencial los trozos sucesivos.

## Encriptación de cadenas

PHP dispone de funciones que permiten codificar o encriptar cadenas de caracteres.

bin2hex(cadena)

Devuelve una cadena ASCII que contiene la representación hexadecimal de la cadena. La conversión se realiza byte a byte, con los 4 bits superiores primero.

crypt(cadena)

Devuelve la cadena encriptada utilizando una semilla aleatoria de dos caracteres. Por su caracter aleatorio, si se ejecuta dos veces seguidas –tal como puedes observar en el ejemplo– dará dos resultados diferentes.

crypt(cadena,"xx")

Devuelve la cadena encriptada utilizando como semilla los dos caracteres (entre comillas) que se escriben como segundo parámetro de la función.

Tanto en este supuesto como en el anterior, los dos primeros caracteres de la cadena encrip- tada coinciden con los que han sido utilizados como semilla.

md5(cadena,"xx")

Aplica el algoritmo md5 y devuelve la huella digital generadapor él. crc32(cadena)

Aplica el algoritmo crc32 de comprobación de integridad y devuelve el valor del mismo.

# ARRAY ESCALAR Y ASOCIATIVO

## Arrays escalares

Los elementos de un array escalar puede escribirse con una de estas

sintaxis:

$a[]=valor

ó

$a[xx]=valor

### Arrays asociativos

Los elementos de un array asociativo pueden escribirse usando la siguiente sintaxis:

$a['indice']=valor

En este caso estamos obligados a escribir el nombre del índice que habrá de ser una cadena y debe ponerse entre comillas. Tanto en este supuesto como en el anterior, es posible –y bastante frecuente– utilizar como índice el contenido de una variable. El modo

de hacerlo sería:

$a[$ind]=valor

En este caso, sea cual fuere el valor de la variable $ind, el nombre de la variable nunca se pone entre comillas.

$n=array\_count\_values(ar)

Con array\_count\_values la variable $n será un array que tendrá como índices cada uno de los valores distintos que contenga el array ar, y como valores el resultado de contar el número de veces que se repite cada uno de los valores contenidos en el array inicial.

## Búsqueda de elementos en un array

clav=array\_keys(arr)

Devuelve un array escalar (clav) que contiene como valores los índices del array inicial (arr). clav=array\_keys(arr, valor) Devuelve un array escalar (clav) que contiene como valores los índices de los elementos del array inicial cuyo valor coincide con el indicado mediante el parámetro valor

.

valores=array\_values(arr)

Esta función recoge en una nueva matriz (valores) todos los valores contenidos en otro array. Es una forma de conversión de un array asociativo en otro escalar.

### Ordenación por valores sin mantener índices

sort(array)

Ordena los valores del array en sentido creciente y lo reindexa asignando índice CERO al menor de los valores.

rsort(array)

Ordena la matriz en sentido decreciente de sus valores y la reindexa asignando índice CERO al mayor de estos.

### Ordenación por índices

ksort(array)

Ordena la matriz según sus índices y en sentido creciente de estos.

krsort(array)

Ordena la matriz por índices en sentido decreciente de los mismos. Localización de valores en una matriz

in\_array(valor,array)

La función in\_array busca en la matriz (array) el valor (numérico o cadena) contenido en el parámetro valor. Si lo encuentra devuelve 1, y, si no existiera devolvería NUL.

### Posicionamientos en una matriz

Mediante estas funciones se puede modificar la posición del puntero interno de una matriz y determinar los índices de los elementos a los que apunta en cada momento.

key(array)

Devuelve el índice del elemento de la matriz al que apunta en ese momento el puntero interno de la matriz.

¡Cuidado!

Pese a que hemos pedido a PHP que nos muestre, mediante la instrucción echo $a[3], esa lectura no mueve el puntero interno, que sigue señalando a la primera posición (devuelve CERO).

reset(array)

Desplaza el puntero interno a la posición del primer índice del array.

end(array)

Desplaza el puntero interno a la posición del último índice del array.

pos(array)

Mantiene el puntero interno en la posición del actual.

next(array)

Avanza el puntero interno en una posición respecto a la actual.

prev(array)

Retrocede el puntero interno en una posición respecto a la actual.

### Modificaciones en arrays

var= range(inf,sup)

Crea una nueva matriz (var) escalar en la que los valores de los elementos serán los números enteros (ordenados) pertenecientes al intervalo comprendido entre los valores inf y sup, incluidos estos. Los valores inf y sup deben ser números enteros. shuffle(array) Intercambia de modo aleatorio los valores de un array y los reindexa.

Igual que ocurría en caso de los números aleatorios, la función shuffle deberá ir precedida de una semilla del tipo srand. En el ejemplo hemos usado como semilla la función: srand(time()).

var= array\_flip(array)

Devuelve un array (var) que contiene como valores los índices de la matriz array y como índices los valores de aquella.

Como quiera que los valores pueden estar repetidos y no es posible que lo estén los índices, esta función, en caso de valores repetidos, toma cada uno de esos valores una sola vez, lo utiliza como índice del nuevo array y asigna como valor del nuevo elemento el mayor de los índices –del array original– de los elementos que contuvieran ese valor. Insertando elementos en un arrays

array\_unshift(arr, v1,v2,..)

Inserta al principio de la matriz arr los valores v1, v2, etcétera que pueden ser tantos como se deseen y deben estar separados por comas.

array\_push(array, v1,v2,..)

Inserta al final de la matriz array los valores v1, v2, etcétera, que igual que en el caso anterior, pueden ser tantos como se deseen y deben estar separados por comas.

Tanto array\_unshift como array\_push asignan a los nuevos elementos índices numéricos.

array\_pad(array, n, var)

### Modificación de arrays

array\_unshift($c,97,"Pepe",128);

foreach($c as $clave=>$valor){

echo "Clave: ",$clave," Valor: ",$valor,"<br>";

}

echo "<h3>Inserta elementos al final de una matriz</h3>";

array\_push($a,3.4,"Luis",69);

Inserta nuevos elementos en array y les asigna el valor contenido en var. Insertará tantos nuevos elementos como sea necesario para que el array alcance una longitud de n elementos.

Si el valor de n es positivo inserta los elementos al final del array, si fuera negativo los insertaría al comienzo del mismo. A los nuevos elementos del array se les asignan índices numéricos.

array\_merge($a, $b)

Crea un nuevo array escalar en el que se incluyen todos los elementos contenidos en los arrays $a y $b.

### Quitar elementos de un array

array\_shift($a)

La función array\_shift extrae el primer elemento del array $a.

array\_pop($a)

La función array\_pop extrae el último elemento del array $a.

array\_slice($a,n)

La función array\_slice extrae n elementos del array $a. Si el valor de n es positivo extraerá todos los elementos a partir del que ocupa la posición n contando desde primero hasta el último según el orden de creación de los elementos. Si el valor de n es negativo extraerá todos los elementos a partir del enésimo, esta vez, contando desde el último hasta el primero.

array\_slice($a,n, m)

La función array\_slice con dos parámetros permite extraer una parte de los valores de una matriz siguiendo estos criterios:

Si n y m son positivos, extraerá m elementos a partir del que ocupa la posición enésima de primero a último.

Cuando n es negativo y m es positivo se extraerán m elementos contados a partir del enésimo, esta vez recorriendo el array de último a primero.

En el caso en que n tenga valor positivo y m sea negativo extraerá los comprendidos entre el enésimo contado de primero a último y el emési- mo contado desde el último hasta el primero.

Si n es negativo y m es también negativo extraerá los caracteres 2 de 3 comprendidos entre el enésimo contado de último a primero y el emésimo contado en el mismo sentido.

En este caso se requiere que el valor absoluto de n sea mayor que el de m. En caso de no cumplirse esta condición devolverá un array vacío. Invertir el orden de un array.

array\_reverse(array)

Devuelve un nuevo array cuyos elementos están en orden inverso al del array original.

De esta forma el elemento que ocupaba la última posición pasa a ocupar la primera y así sucesivamente.

# BUCLE FOREACH

El bucle foreach es específico de los array y aplicable a ellos tanto si son escalares como si son de tipo asociativo.

Lectura de índices y valores Con una sintaxis como la que sigue se pueden leer no sólo los valores de un array sino también sus índices.

foreach( array as v1 => v2 ) {

...instrucciones...

}

# FUNCIONES DE USUARIO

La función pueden estar escrita dentro de cualquier script y en cualquier parte del documento, sin que tenga importancia alguna el lugar en el que se incluya la llamada a la misma. También es posible –y bastante habitual– incluir funciones de uso frecuente en documentos externos de modo que pueden ser compartidas. En este caso, además de invocarla es necesario indicar a PHP el lugar donde debe buscarla. Hablaremos de ello cuando estudiemos lo relativo a include. Las funciones de usuario requieren la siguiente sintaxis:

function nombre(){

.....

... instrucciones ...

.....

}

Para invocar una función la sintaxises la siguiente:

nombre()

Si dentro de una función se utiliza un nombre de variable idéntico al de otra externa a ella (sin definirla global) la nueva variable se inicia con valor nulo y los eventuales valores que pudiera ir conteniendo se pierden en el momento en que se acaba su ejecución. Una manera de lograr que los valores una variable externa puedan ser modificados por una función, es lo que se llama en argot informático

«pasar variables por referencia».

Hay que anteponer al nombre de la variable el símbolo & y PHP interpretará que la estamos pasando por referencia.

El & puede anteponerse tanto en la definición de la función como en la llamada a la función

# FUNCIONES FECHA

date (cadena de formato)

Devuelve valores de fecha y hora actuales utilizando los parámetros que se señalan en la tabla para establecer el formato de salida de los datos. Dentro de la misma cadena de formato puede contener tantos parámetros como se deseen.



<? echo "Son las ", date("h : i : s")," y hoy es ", date("j-n-Y")?>

gmdate(cadena formato)

Se comporta de forma idéntica a date() con la única diferencia de que devuelve la hora y fecha GMT.

Si te fijas en el ejemplo habrá una o dos horas de diferencia según accedas a esta página en verano o invierno.

checkdate(mes,día,año)

Comprueba si los valores de los parámetros mes están dentro del rango permitido (de 1 a 12), si el parámetro día es un valor válido para ese mes (considera años bisiestos) y si el valor del año pertenece al rango 0 a 32767.

Devuelve VERDADERO si los valores corresponden a una fecha correcta y FALSO en el caso de que no ocurra así.

gettimeofday()

Esta función devuelve un array asociativo con los siguientes

índices:

sec

El valor asociado a este índice del array recoge la hora actual (Unix Each) expresada en segundos usec

El valor asociado a usec recoge la fracción en microsegundos de hora actual (Unix Each)

minuteswest

Devuelve los minutos al Oeste de Greenwich



getdate()

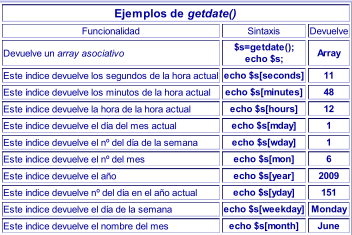
Devuelve un array asociativo con

parámetros de la fecha actual.

Los índices de este array y sus

valores son los que puedes ver en el

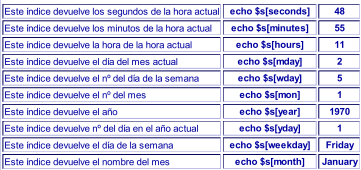
ejemplo.



getdate(número)

Interpreta el número como una fecha Unix Each (segundos transcurridos desde el día 1 de Enero de 1970) y devuelve un array asociativo con los valores relativos a esa fecha. Los índices de este array y sus valores son idénticos a los de getdate() y puedes verlos en el ejemplo.





microtime()

Esta función devuelve la fracción de microsegundos de la hora actual expresada en tiempo Unix.

time()

Esta función devuelve la hora actual en segundos expresada en tiempo

Unix.

mktime (hora, min, seg, mes, día, año , horario)

# CLASES Y OBJETOS

La sintaxis es la siguiente:

class nombre {

....

... definición de variables....

....

.. constructores (opcional)... ....

.. definición de funciones...

....

}

Siempre que desde una función –contenida en una clase– se trate de invocar una variable definida en la misma clase ha de hacerse con la siguiente sintaxis:

$this->variable

Prestemos mucha atención. El $ va siempre delante de la palabra this y solo se escribe una vez y en esa posición.

El nombre de la variable (que va siempre después de -> no lleva

pegado el $.

Una vez definido un objeto ya se le pueden aplicar las funciones definidas en la clase. La sintaxis es la siguiente:

$nombre->funcion()

Esa función -el constructor- se ejecuta de forma automática en el momento en que se define un nuevo

objeto (new)

Según como esté definido, el constructor puede ejecutarse de distintas formas:

• Con los valores predefinidos en las variables de la clase.

• Mediante la asignación de valores preestablecidos en los propios parámetros de la función constructor.

class nuev extends base {

....

... definición de variables....

....

.. constructores (opcional)...

....

.. definición de funciones...

....

}

PHP permite llamar a una función definida en una clase sin necesidad de crear un objeto. La sintaxis es la siguiente:

clase :: funcion()

donde clase es el nombre de una clase y funcion() es una función definida dentro de esa clase.

# EJEMPLO CLASES Y OBJETOS

<?php

abstract class Figuras {

abstract function area();

abstract function perimetro();

}

class Cuadrado extends Figuras {

var $lado;

function \_\_construct ($lad){

$this->lado = $lad;

}

function area() {

$ladoFun = $this->lado;

echo 'El area del cuadrado es :', pow($ladoFun,2),'<br>';

}

function perimetro() {

$ladoFun = $this->lado;

echo ' el perimetro del cuadrado es ', pow($ladoFun,4),'<br>';

}

}

class Triangulo extends Figuras {

var $base;

var $altura;

function \_\_construct($bas,$altur){

$this->base = $bas;

$this->altura = $altur;

}

function area() {

$baseFun = $this->base;

$altuFun = $this->altura;

echo 'El area del triangulo es :',($baseFun \* $altuFun)/2,'<br>';

}

function perimetro() {

$baseFun = $this->base;

echo ' el perimetro del triangulo equilatero es ',$baseFun\*3,'<br>';

}

}

class Circulo extends Figuras{

var $diametro;

function \_\_construct ($diametr){

$this->diametro = $diametr;

}

function area() {

$diaFun = $this->diametro;

echo 'El area del cuadrado es :',pi()\* pow($diaFun/2,2),'<br>';

}

function perimetro() {

$diaFun = $this->diametro;

echo ' el perimetro del circulo es ',2\*pi()\* pow($diaFun/2,2),'<br>';

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN DEL PROGRAMA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

$grupoF = array(new Cuadrado(2),new Triangulo(3,3),new Circulo(5));

for ($cont = 0; $cont<count($grupoF);$cont++){

$grupoF[$cont]->area();

$grupoF[$cont]->perimetro();

echo '<br>';

}

?>

# EXPRESIONES REGULARES



Se trabaja con <http://www.gskinner.com/RegExt/>

## FUNCIONES DE EXPRESIONES REGULARES

$texto = "00000002222333";

$patron = '/[1-9]\*$/';

preg\_match($patron,$texto,$matches);

print\_r($matches);

Con preg\_match obtenemos un array con las conincidencias($matches) que se puede imprimir con la función print\_r

En el patrón se pueden hacer grupos de patrones utilizando paréntesis y el match resultante será un array con varios elementos.

<?php

$texto = 'C:\folder\file.txt';

$patron = '#^[a-zA-Z]:\\\.\*\\\(.\*\\.[a-z0-9]+)$#';

preg\_match($patron, $texto, $coincidencias);

print($coincidencias[1]);

?>

# CONEXIONES A BASES DE DATOS

## TIPOS DE DATOS MYSQL

* + TinyInt: Es un número entero con signo o sin signo. Con signo su rango va desde **-128 a 127** y sin sigo de **0 a 255**.
  + Bit: Número entero que puede ser **0 o 1**.
  + SmallInt: Es un número entero con signo o sin signo. Con signo su rango va desde **-32768 a 32767** y sin signo de **0 a 65535**.
  + MediumInt: Es un número entero con signo o sin signo. Con signo su rango va desde **-8.388.608 a 8.388.607** y sin signo de **0 a 16777215**.
  + Integer o Int: Es un número entero con signo o sin signo. Con signo su rango va desde **-2147483648 a 2147483647**  y sin signo de **0 a 429.4967.295**.
  + BigInt: Es un número entero con signo o sin signo. Con signo su rango va desde **-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807** y sin signo de **0 a 18.446.744.073.709.551.615**.
  + Float: Es un número pequeño de coma flotante. Sus valores van desde   **0 a 23**  con precisión simple y de 24 a 53 con doble precisión.
  + Decimal o Numeric: Es un número de coma flotante que se almacena como una cadena y no redondea su valor.

tipos de fecha y hora existentes en MySQL.

* + DateTime: Es una combinación de fecha y hora con un rango de valores desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de **año-mes-dia horas:minutos:segundos**.
  + TimeStramp: Es una combinación de fecha y hora con un rango de valores desde el 1 de enero de 1970 al año 2037. El formato depende del tamaño del campo:

|  |  |
| --- | --- |
| **Tamaño** | **Formato** |
| 14 | aaaammddhhmmss |
| 12 | aammddhhmmss |
| 8 | aaaammdd |
| 6 | aammdd |
| 4 | aamm |
| 2 | aa |

* + Time: Almacena un hora con un rango que va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es **horas-minutos-segundos**.
  + Year: Almacena un año con un rango que va desde 1901 a 2155. Dependiendo del tamaño del campo se puede almacenar el año con dos o cuatro dígitos.

tipos de cadenas de caracteres existentes en MySQL.

* + Char: Almacena una cadena de longitud fija de entre **0 y 255** caracteres.
  + VarChar: Almacena una cadena de longitud variable de entre **0 y 255** caracteres.
  + TinyText y tinyBlob: Columna con una longitud máxima de **255** caracteres.
  + Blob y Text: Texto con un máximo de **65535** caracteres.
  + MediumBlob y MediumText: Texto con un máximo de **16.777.215** caracteres.
  + LongBlob y LongText: Texto con un máximo de **4.294.967.295** caracteres.
  + Enum: Campo que puede tener un único valor de una lista que se especifica. Acepta hasta **65535** valores distintos.
  + Set: Campo que puede tener ninguno, uno o varios valores de una lista específica. Acepta hasta **64** valores distintos.

## La instrucción OR DIE

Si añadimos a la instrucción $c=mysql\_conect('h','u','p') (sin paréntesis, ni comas, ni punto y coma, sólo separado por un espacio):

or die ('mensaje')

y ponemos el punto y coma de fin de instrucción después de cerrar este último paréntesis, en el caso de que se produzca un error se interrumpirá la ejecución del script y apare- cerá en la ventana del navegador el texto incluido en mensaje.

## CONEXIONES

$mysqli = new mysqli("localhost", "usuario", "contraseña", "basedatos");  
if ($mysqli->connect\_errno) {  
    echo "Fallo al contenctar a MySQL: (" . $mysqli->connect\_errno . ") " . $mysqli ->connect\_error;  
}

## EJECUCION DE CONSULTAS

$mysqli->query("CREATE TABLE test(id INT)"

### CONSULTAS SIMPLES

$resultado = $mysqli->query("SELECT id FROM test ORDER BY id ASC");

La consulta se guarda en la variable a la cual se asigna la instrucción. Una vez ejectuada la consulta es necesario liberarla con la instrucción:

$resultado->free();

Si queremos recorrer el resultado fila por fila es necesario hacer un while con la instrucción:

$resultado->fetch\_row() como condición. Esta instrucción se asigna una variable que será un array con cada fila.

Este array se recorre con un foreach dentro del while.

#### EJEMPLOS CONSULTA SIMPLE

$resultado = $conexion->query("SELECT Nombre, Apellido1, Apellido2 FROM demo4

WHERE (Sexo='M')");

//Creamos la cabecera de la tabla

echo "<table align=center border=2>";

//Hacemos un bucle para recoger el resultado de cada fila

//de la tabla en un array

while ($fila = $resultado->fetch\_row()){

//Salto de linea en la tabla

echo "<tr>";

//Almacenamos las filas obtenidas en distintas celdas

foreach($fila as $clave){

echo "<td>",$clave,"</td>";

}

}

### CONSULTAS MULTIPLES

Para efectuar multiconsultas se crea un string con todas las consultas, cada consulta se separa con punto y coma.

Despues se ejecuta la instrucción $conexion->multi\_querry($consulta). El objeto conexión llama a la instrucción multiquerry con el string multiconsulta como parámetro.

Se crea un do-while en el cual se abre un if con condición $resultado = $conexion->use\_result(). Esta instrucción ejecuta la primera consulta y la almacena en una variable que contiene el resultado. Se pone como condiccion dentro del if por que si no hay consulta no entra.

Dentro del if se crea un while con la instrucción fetch\_row() asignada a una variable resultado que será el array resultante de la fila de la consulta. Que dentro se recorre con un foreach.

Después del foreach y cerramos el if y el while se libera la consulta.

Entramos en la condición del do-while y en la condición se ejecuta $conexion ->more\_result() con un and y se ejecuta &conexión->next\_result()

Primero preguntamos si hay otra consulta y si la hay se desplaza el puntero.

### EJEMPLO MULTICONSULTA

<?php

$c = new mysqli('localhost','php','php');

$c->select\_db('practicas');

$consulta = "select \* from alumnos where REPETIDOR = 'si';";

$consulta.= "select \* from alumnos where DNI = 96088638;";

$consulta.= "select DNI from alumnos where NOMBRE = 'Iluminado'";

#primero se efectua la multiconsulta

if ($c->multi\_query($consulta)) {

do {

#dentro del do-while recogemos la consulta en una variable

if ($resultado = $c->use\_result()) {

#Ejecutamos el resultado linea por linea

while ($linea = $resultado->fetch\_row()) {

foreach ($linea as $valor) {

echo $valor;

}

echo '<br>';

}

#liberamos la variable

$resultado->free();

}

echo"----------------------------------",'<br>';

#en el while se comprueba si hay mas consultas y si las hay se cambia de consulta

} while ($c->more\_results()&&$c->next\_result());

}

/\* cerrar conexi�n \*/

$c->close();

?>

## INSERCION DE DATOS EN LA TABLA

$c->query("insert into alumnos (DNI,NOMBRE,APELLIDOS,fecha,REPETIDOR) values ('$dni','$nombre','$apellidos','$date','$repetidor')");

$filas = $c->affected\_rows();

## MANEJO DE ERRORES

if($conexion->query($crear)) {

echo "<h2> Tabla $tabla creada con EXITO </h2><br>";

}else{

echo "<h2> La tabla $tabla NO HA PODIDO CREARSE ";

//Mostramos el error

$numerror = $conexion->errno;

if ($numerror==1050){

echo "porque YA EXISTE</h2>";

Con la instrucción $conexion ->errno sabemos el numero de error y lo almacenamos en una variable.

$conexion->query($consulta);

/\*Comprobamos los posibles errores, si errno es 0 significa que no hay error

\* Si es 1062 significa clave duplicada

\* En otros posibles errores los visualizamos con mysqli\_errno (Que es el

\* número del error y mysqli\_error (Que es la explicación del error)

\*/

if ($conexion->error==0){

echo "<h2>Registros AÑADIDOS</b></H2>";

}else{

if ($conexion->errno==1062){

echo "<h2>No ha podido añadirse el registro<br>Ya existe un campo con este DNI</h2>";

}else{

$numerror = $conexion->errno;

$descrerror = $conexion->error;

echo "Se ha producido un error nº $numerror que corresponde a: $descrerror <br>";

}

}

La instrucción error que se puede almacenar en una variable sirve para describir el error mysql

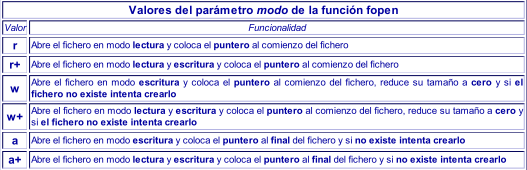
# FICHEROS EXTERNOS

## Abrir o crear ficheros

Para crear o modificar ficheros se utiliza la instrucción:

$f1=fopen(fichero,modo)

$f1 es una variable que recoge el identificador del recurso, un valor importante (será utilizado para referirnos a este fichero en instrucciones posteriores), fichero es el nombre (con extensión) del fichero a abrir o crear y deberá escribirse entre comillas, y modo, que es una cadena que debemos poner entre comillas, el indicador del modo de apertura elegido.



## Cerrar ficheros

Una vez finalizado el uso de un fichero es necesario cerrarlo. Para ello PHP dispone de la siguiente intrucción:

fclose($f1)

Esta función - que devuelve un valor booleano- permite cerrar el fichero especificado en $f1 que, como recordarás, es el valor del identificador de recurso que le fue asignado automaticamente por PHP en el momento de la apertura.

## Punteros internos

Se trata de las siguientes:

feof($f1)

Es un operador booleano que devuelve CIERTO (1) si el puntero señala el final del fichero y FALSO si no lo hace.

rewind($f1)

Coloca el puntero interno al comienzo del fichero indicado por el identificador del recurso $f1.

fseek($f1, posición)

Sitúa el apuntador del fichero señalado por el identificador del recurso $f1 en la posición (expresada en bytes) señalada por posición.

ftell($f1)

Devuelve (expresada en bytes) la posición actual del puntero interno del fichero.

## Lectura de ficheros

La lectura de los contenidos de un fichero puede hacerse de dos maneras: sin apertura previa o con apertura previa del mismo.

### Lectura de ficheros sin apertura previa

Las funciones que permiten la lectura de ficheros sin haber sido abiertos previamente son las siguientes:

readfile(fichero)

Escribe directamente en el punto de inserción del script el contenido completo del fichero.

¡Atención!

La función readfile escribe el contenido del fichero sin necesidad de ir precedido por echo ni print.

Si se pone echo o se recoge en una variable, además de su contenido añadirá un número que indica el tamaño del fichero expresado en bytes.

$var=file(fichero)

Crea $var –un array escalar– Una línea termina allí donde se haya insertado un salto de línea en el fichero original.

### Lectura de ficheros con apertura previa

Para la utilización de estas funciones los ficheros han de ser abiertos en un modo que permita la lectura.

La función que permite la lectura completa del fichero es:

fpassthru($f1)

Esta función presenta algunas peculiaridades importantes:

• Cierra el fichero de forma automática después de la lectura.

Por esa razón, si se escribe la función fclose a continuación de fpassthru, se produce un error.

• Si el resultado se recoge en una variable, o si va precedido de echo, además de escribir el contenido del mismo, añadirá el número de bytes que indican su tamaño.

fgets($f1,long)

Extrae del fichero señalado por el $f1 una cadena –que comienza en la posición actual del puntero– y

cuya longitud está limitada por el menor de estos tres valores:

• El valor (en bytes) indicado en long.

• La distancia (también en bytes) desde la posición actual del puntero hasta el final del fichero.

• La distancia que hay entre la posición actual del puntero y el primer salto de línea.

fgetc($f1)

Extrae el caracter siguiente al señalado por la posición actual del puntero.

Los valores del array devuelto por la función stat

Para la utilización de estas funciones los ficheros han de ser abiertos en un modo que permita la lectura.

La función que permite la lectura completa del fichero es:

fpassthru($f1)

Esta función presenta algunas peculiaridades importantes:

• Cierra el fichero de forma automática después de la lectura.

Por esa razón, si se escribe la función fclose a continuación de fpassthru, se produce un error.

• Si el resultado se recoge en una variable, o si va precedido de echo, además de escribir el contenido del mismo, añadirá el número de bytes que indican su tamaño.

fgets($f1,long)

Extrae del fichero señalado por el $f1 una cadena –que comienza en la posición actual del puntero– y cuya longitud está limitada por el menor de estos tres valores:

• El valor (en bytes) indicado en long.

• La distancia (también en bytes) desde la posición actual del puntero hasta el final del fichero.

• La distancia que hay entre la posición actual del puntero y el primer salto de línea.

fgetc($f1)

Extrae el caracter siguiente al señalado por la posición actual del puntero.

Escribir en un fichero Una vez abierto un fichero -en modo que permita escritura- la función PHP que nos permite escribir en el es la siguiente:

fwrite($f1,"texto",long)

donde: $f1 sigue siendo el identificador de recurso, texto la cadena de texto a insertar en el fichero y long el número máximo de caracteres que han de insertarse. Si la cadena de texto tiene menor o igual longitud que el parámetro long la escribirá en su totalidad, en caso 3 de 5contrario sólo escribirá el número de caracteres indicados.

También es posible utilizar:

fputs($f1,"texto",long)

que en realidad es un alias de la función anterior.

¡Atención!

Estas funciones realizan la inserción de la cadena a partir de la posición a la que apunte el puntero en el momento de ser invocadas. Para poder añadir contenidos a un fichero el puntero deberá apuntar el final del fichero preexistente y estar abierto en un modo que permita añadir contenidos.

### Borrado de ficheros

Para borrar ficheros se utiliza la siguiente instrucción:

unlink(fichero)

fichero ha de ser una cadena que contenga el nombre y la extensión del fichero y, en su caso, también el path.

### Duplicado de ficheros

La función:

copy(fich1, fich2)

Copia el fichero fich1 (debe indicarse nombre y extensión) en otro fichero cuyo nombre y extensión se establecen en la cadena fich2.

Esta función devuelve un valor booleano indicando si la copia se ha realizado con éxito TRUE (1) o FALSE (nul) si por alguna razón no ha podido realizarse la copia.

### Renombrar ficheros

La función:

rename(fich1, fich2)

Cambia el nombre del fichero fich1 (hay que poner nombre y extensión) por el indicado en la cadena fich2. También devuelve TRUE o FALSE. Si tratamos de cambiar el nombre a un fichero inexistente nos dará error.

### Funciones informativas

PHP dispone de funciones que nos facilitan información sobre ficheros.

Algunas de ellas son las siguientes:

file\_exists(fichero)

Esta función devuelve TRUE si el fichero existe, en caso contrario devuelve FALSE.

filesize(fichero)

Devuelve el tamaño del fichero expresándolo en bytes. En caso de que el fichero no existiera nos dará un error.

filetype(fichero)

Devuelve una cadena en la que se indica el tipo del fichero. En caso de que el fichero no existiera nos dará un error.

filemtime(fichero)

Devuelve –en tiempo Unix– la fecha de la última modificación del fichero.

stat(fichero)

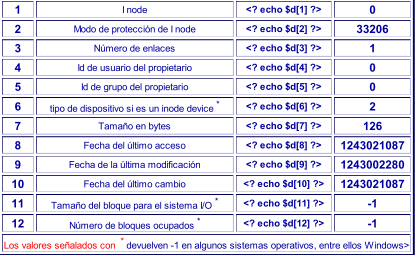
Devuelve un array que contiene información sobre el fichero.

Recogeremos en un array, que llamaremos $d, el resultado de la función stat aplicada sobre el fichero domingo.txt. Para ello vamos a utilizar la siguiente sintaxis:

$d=stat("domingo.txt")

El contenido y significado de los valores asociados a los índices de array $d son los que tenemos en la tabla de la siguiente.





# TRANSFERENCIA DE FICHEROS

La transferencia de un fichero requiere dos documentos:

Un formulario que la inicie y un script que la recoja.

El formulario

Se diferencia del que hemos visto en páginas anteriores en tres aspectos. Dos de ellos se refieren a cambios dentro de la etiqueta <form> y el tercero es un nuevo tipo de input del que aún no hemos hablado. En la etiqueta <form> hemos de incluir –obligatoriamente– el method='POST' y ENCTYPE = "multipart/form-data" ya que no soporta ni otro método ni otra forma de encriptación

El cuerpo del formulario ha de contener un nuevo tipo de input que

utiliza la siguiente sintaxis: <input type='file' name='nm'>

La tranferencia Una vez enviado el formulario, el fichero transferido se guarda en un directorio temporal del servidor

# UTILIZACION DE FORMULARIOS

localización. Editaremos ese fichero, buscaremos la línea donde dice register\_globals, comprobemos

que está en OFF y modifiquemos esa directiva dejándola como register\_globals=ON y guardaremos

los cambios..

¡Cuidado!

Antes de hacer una modificación en php.ini ó en httpd.conf desactiva el servidor Apache. Cuando hayas acabado con los cambios vuelve a ponerlo en marcha (de lo contrario te aparecerá el mensaje: no se puede encontrar el servidor) y ya arrancará atendiendo a la nueva configuración. ¡No te olvides nunca de hacerlo así! Te evitarás un montón de sobresaltos.

Recuerda también, si estás trabajando bajo Linux, que la modificación este tipo de ficheros de configuración requiere los privilegios de superusuario.

### PHP dinámico

Lo que hemos visto hasta el momento, solo nos ha servido para escribir una serie de scripts PHP y ver los resultados de su ejecución, pero aún no hemos visto de qué forma se puede lograr que interactúen el cliente y el servidor.

Veamos cómo hacerlo.

### Envío a través del navegador

La forma más simple de que un cliente pueda enviar valores a un servidor es incluir esos valores en la propia petición, insertándolos directamente en la barra de direcciones del navegador.

#### Forma de envío

Deberá insertarse –en la barra de direcciones del navegador– lo siguiente:

pagina.php?n1=v1&n2=v2

donde pagina.php será la dirección de la página que contiene el script que ha de procesar los valores transferidos. ? es un carácter obligatorio que indica que detrás de él van a ser insertados nombres de variables y sus valores. n1, n2, etcétera representan los nombres de las variables. = es el separador de los nombres de las variables y sus valores respectivos. v1, v2,... simbolizan el valor asignado a cada una de las variables. & es el símbolo que separa los distintos bloques variable = valor.

**Los nombres de las variables nunca llevan el signo $.**

Los valores de las variables –sean números o cadenas– nunca se escriben entre comillas.

Algunos caracteres especiales (& por ejemplo) no pueden escribirse directamente dado que se prestan a confusión (no sabría si es un valor de una variable o si se trata de el símbolo de unión.

### Utilizando formularios

En esos casos es necesario sustituir en carácter por su codificación URL que representa cada carácter anteponiendo el signo % al valor de su código ASCII expresando en formato hexadecimal.

Escribamos ahora en la barra de direcciones del navegador la siguiente dirección: <http://localhost> /cursoPHP/ejemplo14.php?a=21&b=456, (o pulsemos directamente en este enlace) y podremos comprobar que 21 x 456 aún es igual 9576.

¡Cuidado!

Aquí, más que nunca, conviene reiterar, una vez más, los errores de sintaxis más frecuentes:

Los nombres de variables son distintos si se cambian mayúsculas y minúsculas. Pon mucho cuidado en escribirlos correctamente.

Los nombres de las variables predefinidas, tales como **$HTTP\_GET\_VARS, $\_GET, etcétera van en mayúsculas.**

No olvides poner punto y coma al final de cada línea de instrucciones.

Presta atención a la apertura/cierre de comillas y mucha más atención aún si se trata de comillas anidadas. En este caso procura usar (") para las exteriores y (') para las interiores.

¡Cuidado!

En modo local puedes establecer las configuraciones de php.ini a tu antojo y, además, estás utilizando una versión –5.2.9-2– de PHP que permite superglobales. Esta versión –junto con la posibilidad de modificar php.ini– te permite utilizar cualquiera de las opciones de transferencia de variables. Pero, si pretendes publicar tus páginas utilizando un hosting ajeno debes cerciorarte de cual es su versión de PHP –no todos tienen instaladas versiones superiores a 4.1.0– y conocer la configuración de sus php.ini.

Ten en cuenta que allí no vas a poder modificar las configuraciones y de no tener en cuenta estos aspectos, puedes verte obligad@ a modificar tu código fuente para adecuarlo a la configuración de tu hosting.

### Un formulario

<?

/\* Modifiquemos la instrucción anterior y utilicemos las variables predefinidas

$HTTP\_GET\_VARS['a'], $HTTP\_GET\_VARS['b']

en vez de $a y $b que utilizabamos en el ejemplo anterior \*/

# Pongamos un comentario de advertencia. Recuerda que <br>

# sirve para insertar un salto de línea en la salida

print ("Este resultado es el que utiliza $HTTP\_GET\_VAR<br>");

print ($HTTP\_GET\_VARS['a']." x ".$HTTP\_GET\_VARS['b']." = ".

$HTTP\_GET\_VARS['a']\*$HTTP\_GET\_VARS['b']);

/\* Ahora trataremos de comprobar que también podemos

utilizar la superglobal $\_GET como $\_GET['a'] y $\_GET['b']

con iguales resultados que las anteriores \*/

# Un comentario para identificar el origen del resultado

print("<br>El resultado siguiente ha sido generado usando $\_GET <br>");

print ($\_GET['a']." x ".$\_GET['b']." = ".$\_GET['a']\*$\_GET['b']);

?>

<html>

En esos casos es necesario sustituir en carácter por su codificación URL que representa cada carácter anteponiendo el signo % al valor de su código ASCII expresando en formato hexadecimal.

En este enlace tienes una tabla de conversión. Se pueden incluir tantos nombre = valor como se desee. La única restricción es la longitud máxima permitida por el método GET (el utilizado en este caso) que se sitúa en torno a los 2.000 caracteres.

### Recepción de datos

Cuando es recibida por el servidor la petición de un documento con extensión .php en la que tras el signo ? se incluyen una o varias parejas nombre = valor, los nombres de las variables y sus valores respectivos se incluyen, de forma automática, en variables predefinidas del tipo:

$HTTP\_GET\_VARS['n1']

$HTTP\_GET\_VARS['n2']

en las que n1, n2, ... coinciden exactamente con nombres asignados a cada una de las variables en esa transferencia. Cada una de esas variables contendrá como valor aquel que hubiera recibido a través de la petición.

Si la versión de PHP es superior a la 4.1.0, esos mismos valores se incluirán en variables superglobales del tipo $\_GET de modo que –en el supuesto de que la versión lo soporte– los valores de la petición también (esta opción no excluye la anterior) estarían disponibles en:

$\_GET['n1']

$\_GET['n2']

Según el modo en que esté configurado el php.ini podría caber una tercera posibilidad de registro de esos valores.

Si la directiva register\_globals –en el fichero php.ini– está configurada con la opción ON, los valores transferidos desde el navegador –además de ser recogidos en las variables anteriores– son asignados a otras variables del tipo: $n1, $n2,

... cuyos nombres son el resultado de anteponer el símbolo $ a los nombres de las variables contenidas en la petición.

La elección a la hora de escribir los scripts de uno u otro tipo de variable debe hacerse teniendo en cuenta que:

– Esta última –sin duda la más cómoda– tiene el problema de que sólo es válida cuando register\_globals=on y, además, es la más insegura de todas.

– La superglobal $\_GET tiene una sintaxis más corta que su alternativa

y, además, añade como ventaja su condición de superglobal, que permite utilizarla en cualquier ámbito sin necesidad de declararla

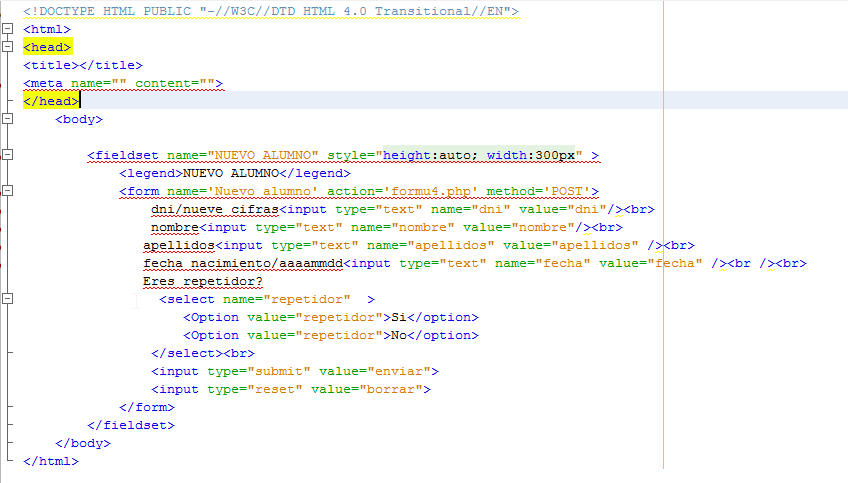
expresamente como global. Es la opción del futuro. Su único

inconveniente es que puede no estar disponible en hostings que aún mantienen versiones antiguas de PHP.

### Envío a través deformularios

La interacción cliente–servidor que acabamos de ver, resulta incómoda en su uso y no demasiado estética. Hay una segunda opción –la de uso más frecuente– que es la utilización de formularios.

Los formularios no son elementos propios de PHP –actúan del lado del cliente– y su estudio es más propio del ámbito de HTML que de PHP.



## Interpretación de los datos recibidos a través de formularios

### Método GET

No se diferencia en nada del descrito para el supuesto anterior. Utiliza las mismas variables predefinidas, las utiliza con idéntica sintaxis y s comporta de igual forma en lo relativo a las opciones de register\_globals. Los nombres de las variables son en este caso, los incluidos como name en cada una de las etiquetas del formulario.

### Método POST

En el caso de que el método de envío sea POST hay una diferencia a tener en cuenta en cuanto a las variables que recogen la información.

### Diferencias ente los métodos GET y POST

Las diferencias entre uno y otro método son las siguientes:

Método GET

Las particularidades de este método

son las siguientes:

– Al ser enviado el formulario se carga en el navegador la dirección especificada como action, se le añade un ? y a continuación se incluyen los datos del formulario.

Todos los datos de la petición van a ser visibles desde la barra de direcciones del navegador.

– Únicamente son aceptados los caracteres ASCII.

– Tiene una limitación en el tamaño máximo de la cadena que contiene los datos a transferir. En IE esa limitación es de 2.083 caracteres.

**Método POST**

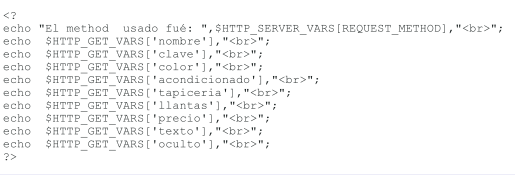
No tiene las limitaciones indicadas para el caso de GET en lo relativo a visibilidad ni en cuanto a aceptación de caracteres no ASCII.

Este método de transferencia de datos es el más habitual cuando se utilizan formularios.

La variable $\_REQUEST

PHP también dispone –a partir de su versión 4.1.0– de la variable $\_REQUEST (de tipo superglobal) que aúna las funcionalidades de $\_GET y $\_POST y que recoge en variables del tipo

$\_REQUEST['nombre'] tanto los valores transferidos mediante el método GET como mediante POST





# LA OPCION INCLUDE

include("nom. del fichero")

El parámetro nom. del fichero es una cadena que contiene el path y el nombre del fichero cuyos contenidos pretendemos incluir.

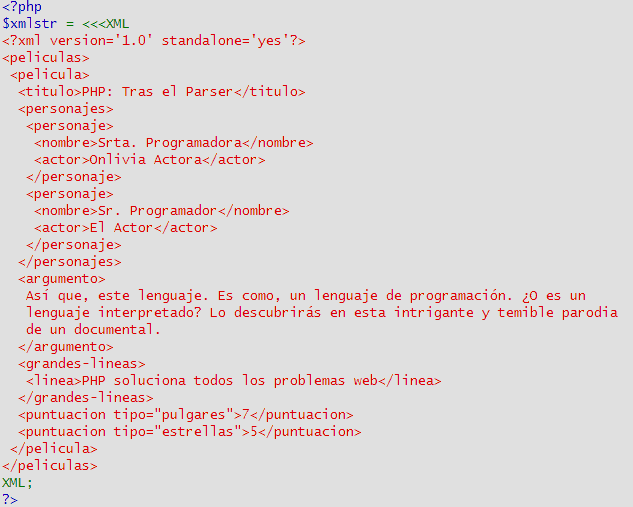
Este tipo de ficheros pueden contener: texto, etiquetas HTML y funciones.

La función require()

A diferencia de include, la etiqueta require lee y escribe –en el documento a partir del que es invocada– el archivo referenciado completo y no acepta condicionales que sí son aceptados por include.

# SIMPLE XML

Para aprender el funcionamiento se usa el siguiente ejemplo:



## ABRIR LOS CONTENIDOS XML

Para incluir un xml dentro de un archivo php se usa la siguiente sintaxis:

<?php  
$xmlstr = <<<XML

XML;  
?>

Para incluir un fichero que sea un xml en vez de php con un string xml la sintaxis en el programa php seria:

$ficheroXML = simplexml\_load\_file('alumnos.xml');

Si queremos obtener datos de un archivo php con el string xml y además queremos seleccionar una etiqueta del xml en concreto:

<?php  
include 'ejemplo.php';  
  
$peliculas = new SimpleXMLElement($xmlstr);  
  
echo $peliculas->pelicula[0]->argumento;  
?>

Se incluye el archivo en cuestión y se crea un objeto extrayendo el string del fichero incluido.

Si queremos acceder a contenidos no únicos dentro del string xml se recorrerá todo el contenido seleccionado con un foreach.

<?php  
include 'example.php';  
  
$peliculas = new SimpleXMLElement($xmlstr);  
  
/\* Para cada <personaje>, mostramos cada <nombre>. \*/  
foreach ($peliculas->pelicula->personajes->personaje as $personaje) {  
   echo $personaje->nombre, ' interpretado por ', $personaje->actor, PHP\_EOL;  
}  
  
?>  
Hasta aquí, únicamente se ha cubierto el trabajo de leer nombres de elementos y sus valores. SimpleXML puede también acceder a los atributos de los elementos. Para acceder a ellos, se realiza de la misma forma que si fuesen elementos de un [array](http://www.php.net/manual/es/language.types.array.php).

<?php  
include 'example.php';  
  
$peliculas = new SimpleXMLElement($xmlstr);  
  
/\* Acceso a los nodos <puntuacion> de la primera película.  
 \* Mostramos la escala (tipo) también. \*/  
foreach ($peliculas->pelicula[0]->puntuacion as $puntuacion) {  
    switch((string) $puntuacion['tipo']) { // Toma los atributos como índices del elemento  
    case 'pulgares':  
        echo $puntuacion, ' votos positivos';  
        break;  
    case 'estrellas':  
        echo $puntuacion, ' estrellas';  
        break;  
    }  
}  
?>

SimpleXML incorpora soprote XPath. Para encontrar todos los elementos

<?php  
include 'example.php';  
$peliculas = new SimpleXMLElement($xmlstr);  
  
foreach ($peliculas->xpath('//personaje') as $personaje) {  
    echo $personaje->nombre . ' interpretado por ' . $personaje->actor, PHP\_EOL;  
}  
?>

Un ejemplo de como tratar de buscar datos con xpath:

while($cont<count($numeroDeCampos)){

$dni = $ficheroXML->xpath("/alumnos/alumno['$cont']/dni");

$dni = $ficheroXML->xpath("/alumnos/alumno['$cont']/dni");

$nombre = $ficheroXML->xpath("/alumnos/alumno['$cont']/nombre");

$apellidos = $ficheroXML->xpath("/alumnos/alumno['$cont']/apellidos");

$fecha = $ficheroXML->xpath("/alumnos/alumno['$cont']/fecha");

$repetidor = $ficheroXML->xpath("/alumnos/alumno['$cont']/repetidor");

}

[Apuntes de xpath](UD5%20(en%20clase).doc)

Los datos en SimpleXML no tienen que ser constantes. El objeto permite que se manipulen todos sus elementos.

<?php  
include 'example.php';  
$peliculas = new SimpleXMLElement($xmlstr);  
  
$peliculas->pelicula[0]->personajes->personaje[0]->nombre = 'Señorita Programadora';  
  
echo $peliculas->asXML();  
?>

## **Añadiendo elementos y atributos**

<?php  
include 'example.php';  
$peliculas = new SimpleXMLElement($xmlstr);

$character = $peliculas->pelicula[0]->personajes->addChild('personaje');  
$character->addChild('nombre', 'Sr. Parser');  
$character->addChild('actor', 'John Doe');  
  
$rating = $peliculas->pelicula[0]->addChild('puntuacion', 'Todos los públicos');   
$rating->addAttribute('tipo', 'clasificación');  
  
echo $peliculas->asXML();  
?>

Adchild(), se usara invocado por el objeto mediante el cual se inserta un nodo hijo con nombre y datos como parámetro.

Si además queremos incluir atributos en ese nodo hijo se usara addAttribute(), con el nombre y el dato a incluir.

Si nos fijamos se crean objetos con rutas especiificas en las que se rellenaran los datos dentro de l xml.

La instrucción asXml() es la que escribirá todos los datos en el fichero indicado, que en el caso del ejemplo es el mismo.

Si queremos crear un archivo xml nuevo primero usamos la función fopen() y luego todo lo creado, ya sea proveniente de otro xml o de creación en el script se incluirá con :

$datos->asXML('peliculas.xml');

Siendo películas.xml la ruta del nuevo archivo creado anteriormente.

[Ejemplo completo de xml.](APOYO/6.8.php)

# REFERENCIAS PARA UN MAYOR APOYO PEDAGÓJICO

## [CUADERNO 2 CORREGIDO](APOYO/ProgramacionPHP-corr.pdf)

## [CUADERNO 3 CORREGIDO](APOYO/TecAccesoDatos.doc)

## [EXPLICACION XAMPP](APOYO/explicacion%20xampp.docx)

## [CONFIGURACION DE XAMPP COMO LOCAL](APOYO/CONFIGURACION%20PARA%20XAMPP.docx)

## [MANUAL COMPLEMENTARIO PHP (SESIONES…,)](APOYO/Manual%20PHP.pdf)

## [EXPRESIONES REGULARES PDF](APOYO/regular-expressions.pdf)

## [PDFS DE CLASE RENOMBRADOS Y ORDENADOS](APOYO/PDFS%20CURSO%20PHP%20RENOMBRADOS)

## [APLICACIÓN REFERENCIA PHP PARA ANDROID](https://play.google.com/store/apps/details?id=com.whawhawhat.referencephp&feature=search_result#?t=W251bGwsMSwyLDEsImNvbS53aGF3aGF3aGF0LnJlZmVyZW5jZXBocCJd)