Introducción a PHP

PHP es un lenguaje de scripting que permite la generación dinámica de contenidos en un servidor web. El significado de sus siglas es HyperText Preprocessor. Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje y su escasez de consumo de recursos.

El código **PHP** puede incluirse dentro del código html de la pagina. Para delimitar la seccion de codigo PHP podemos hacerlo de varias formas:

-Usando las etiquetas **<?php** y **<?**
-Usando las etiquetas **<?** y **?>**
-Mediante **<script languaje="php"> </script>**

El funcionamiento de las páginas en **PHP** alojadas en un servidor es el siguiente:

-El navegador del cliente solicita el documento **PHP**.

-Llega la solicitud del servidor y el servidor localiza el documento, lanza el intérprete de PHP y ejecuta todo su codigo.

-Una vez ejecutado el código se genera el resultado en HTML y lo devuelve al servidor para que lo transfiera al cliente.

-El servidor transfiere el resultado en HTML y es mostrado en el navegador del cliente.

Variables

Una **variable** podría definirse como una posición de memoria creada para introducir o asignar cualquier valor o dato. Durante la ejecución del script el valor de la variable puede "variar"(valga la redundancia) tanto de tipo como de valor. En PHP (al igual que en JavaScript) no hace falta declarar la variable ya que simplemente anteponiendo el caracter $ al nombre de la variable estamos indicando que es una variable.

Otro hecho que cabe destacar a la hora de programar en PHP y la declaración de variables es que PHP es un lenguaje "CASE SENSITIVE" es decir que diferencia entre mayúsculas y minúsculas y debido a esta razón no sería lo mismo $miVariable=valor; que $MiVaRiABle=valor; ya que PHP lo interpretaría como dos variables completamente diferentes.

Los tipos de datos posibles que puede almacenar una variable son los siguientes :

Integer	Números enteros positivos y negativos
Double	Números decimales o de coma flotante
String	Cadenas de texto
Boolean	Valores True o False
Array	Tipo especial de colección de valores
Object	Tipo especial de dato complejo

En capítulos posteriores daremos cabida a los dos últimos tipos de datos(array y object) que aqui no han sido explicados ni comentados apenas.

Convertir tipos

PHP es un lenguaje que realiza la conversión de tipos en función de los operandos y del operador. De esta forma si intentamos sumar la cadena '10' y el número 20 la acción que realizaría sería convertir la primera variable a numérica y de esa forma podría sumar 10+20=30.

Aparte de que PHP en algunos casos realiza la conversión existen dos funciones especiales del propio lenguaje las cuales nos permiten saber el tipo de variable que estamos usando y también pueden convertir el tipo de variable:

gettype() Recibe el tipo de variable que es
settype($variable,'tipo variable') Transforma el tipo de variable del modo actual a el modo que le introduzcamos.

Variables características de PHP

argv	Array de argumentos pasados en la ejecución del script.
$_SERVER['argc']	Número de parametros pasados al script en su ejecución.
$_SERVER['PHP_SELF']	Nombre del script que se está ejecutando
$_COOKIE	Array asociativo de pares (clave,valor) pasados a través de cookies
$_GET	Array asociativo de pares (clave,valor) pasados a través de GET
$_POST	Array asociativo de pares (clave,valor) pasados a través de POST
$_FILES	Array asociativo que contiene información de los ficheros recibidos mediante POST
$_ENV	Array asociativo de pares (clave,valor) del entorno
$_SERVER	Array asociativo de pares (clave,valor) del servidor
$_SESSION	Array asociativo de pares (clave,valor) de sesion

Introducción a PHP

Constantes

Son valores que se van a mantener constantes a lo largo de la ejecución del script y que posiblemente queramos usar a lo largo del script un gran número de veces. Para ello usamos la siguiente sintaxis:

define("nombre de la constante", valor);

Mi Primer Script PHP

Una vez que ya tenemos instalados **PHP** y **MySQL** , y el servidor **Apache** configurado para usarlos, podemos comenzar a escribir nuestro primer script en PHP.

Ejemplo script php

```
<html>
<body>
<?php
$myvar = "Hola. Este es mi primer script en PHP n";
//Esto es un comentario
es mi primer script en PHP n";
//Esto es un comentario
echo $myvar;
?>
</body>
</html>
```

Una vez escrito esto lo salvamos en un fichero con la extensión **php**, y lo colocamos en nuestro servidor, http://mi_servidor/php/test.php . Ahora si ponemos esta URL en nuestro navegador veremos una línea con el texto **"Hola. Este es mi primer script en PHP"** .

Lo primero que apreciamos en el script son sus delimitadores. En la primera línea del script vemos **<?php** que nos indica que comienza un script en PHP, y en la última colocamos **?>** para indicar el final del script. Hay que destacar que todas las líneas que se encuentre entre estos delimitadores deben acabar en **punto y coma** , excepto las sentencias de control (if, swicht, while, etc.).

Como en toda programación, es importante poner muchos comentarios, para lo cual si queremos comentar una sola línea tenemos que poner al principio de la línea **//** , si lo que queremos es comentar varias utilizaremos los delimitadores **/* - */** .

Para que el servidor envié texto utilizaremos la instrucción **echo** , aunque también podemos utilizar **printf** de uso similar al del **C** o **Perl** .

Finalmente, vemos que la palabra **myvar** comienza con el signo dólar (**$**) . Este símbolo le indica a **PHP** que es una variable. Nosotros le hemos asignado un texto a esta variable, pero también pueden contener números o tablas (arrays). Es importante recordar que todas las variables comienza con el **signo dólar** . También habréis observado que el texto que le asignamos a la variable termina con **n** , esto no se imprime sirve para indicarle al navegador una nueva línea.

Operadores en PHP

Al desarrollar cualquier programa empleamos normalmente operadores que nos sirven para realizar diversas operaciones que le otorgan un cierto grado de complejidad a nuestros programas, ya que, de otro modo el programa realizaría siempre lo mismo y por tanto no seria un programa útil.

Operadores aritméticos

+ Suma dos valores
- Resta dos valores (o pasa a negativo un valor)
***** Multiplica dos valores
/ Divide dos valores
% Resto de dividir dos valores
++ Incremento en una unidad
-- Decremento en una unidad

Operadores de asignación

= Asigna a la parte derecha el valor izquierdo
+= Realiza la suma de la derecha con la izquierda y la asigna a la derecha
-= Realiza la resta de la derecha con la izquierda y la asigna a la derecha
***=** Realiza la multiplicación de la derecha con la izquierda y la asigna a la derecha
/= Realiza la división de la derecha con la izquierda y la asigna a la derecha
%= Se obtiene el resto y se asigna
.= Concatena el valor de la izquierda con la derecha y lo asigna a la derecha

Operadores lógicos

! Operador NO o negacion.Si era true pasa a false y viceversa
and Operador Y, si ambos son verdaderos vale verdadero
or Operador O, vale verdadero si alguno de los dos es verdadero
xor Verdadero si alguno de los dos es true pero nunca ambos
&& True si ambos lo son
|| True si alguno lo es

Operadores condicionales

== Comprueba si dos números son iguales
!= Comprueba si dos números son distintos
> Mayor que, devuelve true en caso afirmativo
< Menor que, devuelve true en caso afirmativo
>= Mayor o igual
<= Menor o igual

Estructuras de Control

A partir de ahora vamos a dotar de mas "dinamismo" a nuestros scripts ya que a partir de diversas estructuras indicaremos que acción debe realizar en cada caso, además también nos dará la posibilidad de realizar una misma acción multitud de veces con tan solo una línea de código.

Estructura IF

IF es una estructura de control utilizada para tomar decisiones según se cumpla una condición (o varias) o no. Su estructura básica es la siguiente:

```
if(condición/es){
   acción a realizar;
}
else{
   acción a realizar en caso de que no se cumpla;
}
```

Veamos un ejemplo básico para entenderlo mejor:

```
if($edad>=18){
   Comprar cerveza;
}
else{
   echo "No puedes comprar cerveza porque no tienes 18 años";
}
```

e incluso podemos realizar condicionales mas completas como el siguiente caso:

```
if(($edad>=18)&&($dinero>0)){
   Puedes comprar cerveza porque tienes 18 y tu dinero es mayor que 0;
}
else{
   echo "O no tienes pelas o no tienes los 18" ;
}
```

Estructura SWITCH

Toma distintas decisiones en función de distintos estados de la variable.Su sintaxis es la siguiente:

```
switch(expresión){
   case valor1:
     sentencia a ejecutar cuando la expresión tiene como valor valor1
break
   case valor2:
     sentencia a ejecutar cuando la expresión tiene como valor valor2
break
   case valor3:
     sentencia a ejecutar cuando la expresión tiene como valor valor3
break
   default:
     sentencia que se ejecutar por defecto cuando no se cumpla ninguna de las condiciones anteriores
```

Bucle FOR

El bucle for se usa para repetir una misma operación un número determinado de veces. Su sintaxis es la siguiente:

```
for(inicialización;condición;actualización){
   sentencia a ejecutar mientras se cumpla la condición
}
```

El bucle for esta compuesto de 3 partes:

- **Inicialización**: Se ejecuta tan solo al iniciar por primera vez el bucle.En esta parte se suele colocar la variable que contara el numero de veces que se repite el bucle.

- **Condición**: Es la condición que se evaluara cada vez que se inicie el bucle.Esta condición es la que determina la duración del bucle.
- **Actualización**: Sirve para indicar los cambios que queremos ejecutar en las variables cada vez que se ejecuta el bucle.

Un ejemplo de su uso seria el siguiente:

```
for($i=1;i<=10;i++){
    echo "El número actual es".$i;
}
```

De esta forma escribiría todos los números contenidos entre 0 y 10.

Bucles WHILE y DO WHILE

Bucle WHILE
Este bucle se usa cuando queremos repetir la ejecución de unas sentencias un número indefinido de veces. Su sintaxis es la siguiente:

```
while(condición){
   sentencia a ejecutar
}
```

Para entender mejor el uso de while nos serviremos del siguiente ejemplo:

```
while($color != "rojo"){
   color= dame un color;
}
```

Este es un ejemplo de lo que se puede hacer con while. En este caso siempre y cuando el color no sea rojo nos dirá que introduzcamos un color.

Bucle DO...WHILE

Este bucle se usa cuando no sabemos el número de veces que va a ejecutarse un bucle pero lo que si tenemos claro es que por lo menos una vez si que se ejecutara la accion.Su sintaxis es la siguiente:

```
do{
    sentencia del bucle
}while(condicion)
```

BREAK y CONTINUE

BREAK

Se usa para detener el bucle y dejar de interpretar el código que sigue después de el break

CONTINUE

Sirve para volver al principio del bucle desde cualquier parte del bucle.

Funciones en PHP

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

Inclusión de código 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.

Matrices (array)

En la realización de un script en PHP en múltiples ocasiones existen variables que tienen información similar y se procesan de forma semejante. Para ello PHP (y otros lenguajes) poseen un elemento denominado array. Un array es un conjunto de variables agrupadas bajo un único nombre. Cada variable dentro de la matriz se denomina elemento. Dentro de la misma matriz pueden existir variables de diferentes tipos y no es necesario que sean todas del mismo tipo.

Hay que diferenciar entre los dos tipos de matrices existentes:
- **Indexada**: Aquella cuyo acceso a los elementos se realiza por la posición que ocupan dentro de la estructura (se inician siempre desde la posición 0). Ejemplo: $amigos[0]
- **Asociativa**: Es aquella en la que los elementos están formados por pares clave-valor y el acceso se realiza proporcionando una determinada clave. Ejemplo: $amigos['edad']

Para crear matrices en PHP existen dos formas:

- **De forma implícita**, que consistiría en indicarle el elemento(ya sea proporcionando su posición o su clave).Ejemplo: $nombres[0]='Javier';
En caso de no indicarle una posición el array tomara el valor siguiente al ultimo valor introducido.Ejemplo: $nombres[] ='Lucas' // tomaría como valor 1 ya que lo ultimo introducido era 0.

- **Mediante array()** en el cual le pasamos los elementos como parámetros. En caso de matriz indexada toman la posición que ocupan en la creación de la matriz, mientras que los de la matriz asociativa se les asigna su valor mediante "=>".Ejemplo: $amigo=array('Nombre'=>'Jose','Direccion'=>'Neopatria 21');

Cabe destacar que PHP no solo se limita a la existencia de matrices por sí solo sino que existen matrices de matrices, o lo que es lo mismo, matrices multidimensionales. Ejemplo: $amigos[2]['Pedro']

Recorrido de una matriz

Disponemos de diversas herramientas para poder acceder a los elementos de una matriz. En cada momento se mantiene una referencia del elemento de la matriz al que se tiene acceso, por tanto, para recorrer una matriz bastará con modificar dicha referencia. En caso de una matriz indexada el recorrido se realizara mediante un bucle y para ello debemos saber el número de elementos totales que posee la matriz. Para ello nos basamos de la función **count (variable)** donde variable representa la variable de la que se quiere obtener el número de elementos. Si **variable** es una matriz devuelve el número de elementos que tiene, devuelve 1 si solo tiene un elemento (aunque no sea matriz) y 0 si no tiene ningún valor.

Otra función que nos permite saber el número de elementos es **sizeof(matriz)** .
Para acceder a los elementos de una matriz asociativa debemos usar la función **ecah()** que recupera el par formado por clave y valor y además avanza una posición de puntero. Su sintaxis es each(matriz) y los valores que devuelve la matriz asociativa son los siguientes:

Clave	Significado
0	Nombre de la clave
1	Valor asociativo de la clave
key	Nombre de la clave
value	Valor asociado a la clave

La función que realiza el constructor **list(variable1,variable2...variableN)** es asignar los valores del elemento actual de una matriz a las variables indicadas como parámetro.

Navegación sobre matrices

Cuando se trata de matrices indexadas la navegación es sencilla ya que tan solo basta acceder al elemento que queremos mostrar, pero al tratarse de alguna matriz asociativa no se puede aplicar el mismo tratamiento. Para ello existen un conjunto de funciones prefabricadas que nos permiten realizar multitud de acciones:

Sintaxis	Acción
reset(matriz);	El puntero interno vuelve a la primera posición
end(matriz);	El puntero interno va a la ultima posición
next(matriz);	El puntero va al elemento siguiente
prev(matriz);	Accede al elemento anterior
current(matriz);	Devuelve el contenido del elemento actual

Inserción de elementos

Para la inserción de elementos dentro de un array existen una serie de funciones que nos permiten añadir elementos. Entre ellas destacamos:

array_push(matriz,variable1,variableN);
Añade elementos al final de la función y su longitud se incrementara tantos elementos como se hayan añadido

array_unshift(matriz,variable1,variableN);
Añade elementos al principio de la función desplazando a los otros tantas posiciones como elementos haya.

array_pad(matriz,nuevo_tamaño,valor_relleno);
Aumenta el tamaño de la matriz empleando un valor proporcionado como relleno.

Eliminación de elementos

array_shift(matriz);
Elimina el primer elemento de la matriz

array_pop(matriz);
Elimina el último elemento de la matriz

array_splice(entrada,pos_ini,[tamaño],[sustitutos]);
Se usa para reemplazar o borrar el contenido de una porción de matriz, para ello debemos especificar la posición desde la cual queremos iniciar el borrado o sustitución, el tamaño o número de elementos que se verán afectados y los sustitutos (en caso que deseemos sustituirlo por algún elemento).

array_keys(matriz,[valor buscado]);
Se emplea cuando deseamos eliminar un elemento cuya posición desconocemos.

array_values(matriz);
Devuelve una matriz indexada con todos los valores almacenados en la matriz pasada como parámetro.

Manipulación masiva de matrices

array_walk(matriz,nombre_de_la_funcion,lista_parametros);
Se emplea para realizar el mismo proceso definido en la función en todos los elementos incluidos.

Obtención de submatrices

array_slice(matriz,posicion,tamaño);
Permite extraer una secuencia de elementos de una matriz. Los parámetros a pasarle son la matriz en la cual queremos extraer dichos elementos, la posición desde la que se inicia la extracción y el tamaño de la extracción (posiciones que abarcamos a partir de la inicial).

Ordenación de matrices

Criterio	Función
Orden ascendente(matriz indexada)	sort(matriz)
Orden descendente(matriz indexada)	rsort(matriz)
Orden ascendente por valor(matriz asociativa)	asort(matriz)
Orden descendente por valor(matriz asociativa)	arsort(matriz)
Orden ascendente por clave(matriz asociativa)	ksort(matriz)
Orden descendente por clave(matriz asociativa)	krsort(matriz)

Otras funciones

En este apartado se comentaran una serie de funciones (no todas porque seria imposible) que nos pueden servir en cierto momento.

compact() Devuelve una matriz asociativa a partir de un numero indeterminado de parámetros

extract() Crea variables desde matriz asociativa
array_unique() Devuelve matriz sin datos repetidos ya que algunos se eliminan
array_reverse() Devuelve matriz con mismos elementos pero en orden inverso
shuffle() Modifica el orden de elementos de forma aleatoria
array_count_values() Devuelve una matriz asociativa que contiene frecuencias de repetición de los valores de la matriz

in_array() Permite comprobar si un valor esta en la matriz
array_merge() Combina elementos de dos matrices en 1.

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

Clases

Las **Clases** son máximo exponente de la Programación Orientada a Objetos (POO). PHP no es un lenguaje orientado a objeto, pero implementa las características que permiten definir las clases.

Pero, ¿qué son las Clases y para que sirven?. Empecemos por los segundo, sirven hacer el código más legible, y lo que es más importante, reutilizable. Escribir una Clase es sin duda más largo que escribir el código directamente, pero a la larga es más rentable por su portabilidad a otras aplicaciones y su mantenimiento.

Las Clases no son más que una serie de variables y funciones que describen y actúan sobre algo. Por ejemplo, vamos a crear la clase automóvil , la cual tendrá diversas variables, $color , $modelo , $marca , $potencia , $matricula y habrá una serie de funciones que actuarán sobre la clase automóvil como Precio() , Acelerar() , Frenar() , Girar() y Reparar() .

Como ejemplo vamos a crear la clase mysql , que nos servirá para realizar consultas a las bases de datos MySQL.

```
<?php

class DB_mysql {

/* variables de conexión */

var $BaseDatos;

var $Servidor;

var $Usuario;

var $Clave;

/* identificador de conexión y consulta */

var $Conexion_ID = 0;

var $Consulta_ID = 0;

/* número de error y texto error */

var $Errno = 0;

var $Error = "";

/* Método Constructor: Cada vez que creemos una variable

de esta clase, se ejecutará esta función */

function DB_mysql($bd = "", $host = "localhost", $user = "nobody", $pass = "") {

$this->BaseDatos = $bd;

$this->Servidor = $host;

$this->Usuario = $user;

$this->Clave = $pass;

}

/*Conexión a la base de datos*/

function conectar($bd, $host, $user, $pass){
```

```
if ($bd != "") $this->BaseDatos = $bd;

if ($host != "") $this->Servidor = $host;

if ($user != "") $this->Usuario = $user;

if ($pass != "") $this->Clave = $pass;

// Conectamos al servidor

$this->Conexion_ID = mysql_connect($this->Servidor, $this->Usuario, $this->Clave);

if (!$this->Conexion_ID) {

$this->Error = "Ha fallado la conexión.";

return 0;

}

//seleccionamos la base de datos

if (!@mysql_select_db($this->BaseDatos, $this->Conexion_ID)) {

$this->Error = "Imposible abrir ".$this->BaseDatos ;

return 0;

}

/* Si hemos tenido éxito conectando devuelve

el identificador de la conexión, sino devuelve 0 */

return $this->Conexion_ID;

}

/* Ejecuta un consulta */

function consulta($sql = ""){

if ($sql == "") {

$this->Error = "No ha especificado una consulta SQL";

return 0;

}

//ejecutamos la consulta

$this->Consulta_ID = @mysql_query($sql, $this->Conexion_ID);

if (!$this->Consulta_ID) {

$this->Errno = mysql_errno();

$this->Error = mysql_error();
```

```
}
/* Si hemos tenido éxito en la consulta devuelve
el identificador de la conexión, sino devuelve 0 */
return $this->Consulta_ID;
}
/* Devuelve el número de campos de una consulta */
function numcampos() {
return mysql_num_fields($this->Consulta_ID);
}
/* Devuelve el número de registros de una consulta */
function numregistros(){
return mysql_num_rows($this->Consulta_ID);
}
/* Devuelve el nombre de un campo de una consulta */
function nombrecampo($numcampo) {
return mysql_field_name($this->Consulta_ID, $numcampo);
}
/* Muestra los datos de una consulta */
function verconsulta() {
echo "<table border=1>n";
// mostramos los nombres de los campos
for ($i = 0; $i < $this->numcampos(); $i++){
echo "<td><b>".$this->nombrecampo($i)."</b></td>n";
}
echo "</tr>n";
// mostrarmos los registros
while ($row = mysql_fetch_row($this->Consulta_ID)) {
echo "<tr> n";
for ($i = 0; $i < $this->numcampos(); $i++){
echo "<td>".$row[$i]."</td>n";
```

```
}

echo "</tr>n";

}

}

} //fin de la Clse DB_mysql

?>
```

Como habreis observado, para crear una clase utilizamos la sentencia class , y además hemos creado una función con el mismo nombre que la clase, a esa función se le llama **constructor** y se ejecutará cada vez que definamos una variable de esa clase. No es obligatorina variable de esa clase. No es obligatorio crear un **constructor** en una definición de clase.

Otra cosa importante en las clases es el operador -> , con el que indicamos una variable o método (parte derecha del operador) de una clase (parte izquierda del operador). Para hacer referencia a la clase que estamos creando dentro de su definición, debemos utilizar this .

Y ahora veamos un ejemplo de la clase que hemos creado, y supongamos que el código anterior lo hemos guardado en un fichero llamado **clase_mysql.inc.php** .

```
<body>

<html>

<?php

require ("clase_mysql.inc.php");

$miconexion = new DB_mysql ;

$miconexion->conectar("mydb", "localhost", "nobody", "");

$miconexion->consulta("SELECT * FROM agenda");

$miconexion->verconsulta();

?>

</body>

</html>
```

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];

Conexión 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.

Formularios con PHP

Los Formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML. Vamos a dedicar en este capítulo algunas líneas al HTML, para entrar posteriormente a tratarlos con PHP.

Todo formulario comienza con la etiqueta **<FORM ACTION="lo_que_sea.php" METHOD="post/get">** . Con . Con ACTION indicamos el script que va procesar la información que recogemos en el formulario, mientras que METHOD nos indica si el usuario del formulario va ha enviar datos (post) o recogerlos (get). La etiqueta <FORM> indica el final del formulario.

A partir de la etiqueta **<FORM>** vienen los campos de entrada de datos que pueden ser:

Cuadro de texto:

<input type="text" name="nombre" size="20" value="jose">

Cuadro de texto con barras de desplazamiento:

<textarea rows="5" name="descripcion" cols="20">Es de color rojo</textarea>

Casilla de verificación:

<input type="checkbox" name="cambiar" value="ON">

Botón de opción:

<input type="radio" value="azul" checked name="color">

Menú desplegable:

<select size="1" name="dia">

<option selected value="lunes">lunes</option>

<option>martes</option>

<option value="miercoles">miércoles</option>

</select>

Boton de comando:

<input type="submit" value="enviar" name="enviar">

Campo oculto:

<input type="hidden" name="edad" value="55">

Este último tipo de campo resulta especialmente útil cuando que remos pasar datos ocultos en un formulario.

Como habrás observado todos lo tipos de campo tienen un modificador llamado name , que no es otro que el nombre de la variable con la cual recogeremos los datos en el script indicado por el modificador ACTION de la etiqueta FORM FORM , con value establecemos un valor por defecto.

A continuación veamos un ejemplo, para lo cual crearemos un formulario en HTML como el que sigue y lo llamaremos **formulario.htm** :

<HTML>

<BODY>

```
<FORM METHOD="post" ACTION="mis_datos.php">
<input type="hidden" name="edad" value="55">
<p>Tu nombre <input type="text" name="nombre" size="30" value="jose"></p>
<p>Tu sistema favorito
<select size="1" name="sistema">
<option selected value="Linux">Linux</option>
<option value="Unix">Unix</option>
<option value="Macintosh">Macintosh</option>
<option value=&qmargin-left: 75"><option value="Windows">Windows</option>
</select></p>
<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol" value="ON"></p>
<p>¿Cual es tu sexo?</p>
<blockquote>
<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>
<p>Mujer <input type="radio" name="sexo" value="mujer"></p>
</blockquote>
<p>Aficiones</p>
<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>
<p><input type="submit" value="Enviar datos" name="enviar">
<input type="res-left: 50"> <input type="reset" value="Restablecer" name="B2"></p>
</FORM>
</BODY>
<HTML>
```

Y ahora creemos el script PHP llamado desde le formulario **mis_datos.php** :

Todos los datos se encuentran en la variable $_POST, ya que el formulario está enviado por el método post.

```
<?PHP;
if (isset($_POST['enviar']) {
echo "Hola <b>" . $_POST['nombre'] . "</b> que tal estás<BR>n";
echo "Eres " . $_POST['sexo'] . "<BR>n";
echo "Tienes " . $_POST['edad'] . "<BR>n";
```

```
echo "Tu sistema favorito es " . $_POST['sistema'] . "<BR>n";

if (isset($_POST['futbol']) {

echo "Te gusta el futbol <BR>n";

} else odigo" style="margin-left: 50">} else {

echo "NO te gusta el futbol <BR>n";

}

if ($_POST['aficiones'] != "") {

echo "Tus aficiones son: <BR>n";

echo nl2br($_POST['aficiones']);

} else {

echo "NO tienes aficiones <BR>n";

}

}

echo "<a href='formulario.htm'>VOLVER AL FORMULARIO</a>"

?>
```

Una vez rellenados los datos del formulario, pulsamos el botón **Enviar datos** , con lo que le campo **enviar** toma lo que su etiqueta value indica, es decir **enviar="Enviar datos"** . En nuestro script lo primero que evaluamos es que se haya enviado el formulario, y para ello nada mejor que comprobar que la variable $enviar no está vacía. Le ponemos el signo dólar delante a **enviar** , ponemos el signo dólar delante a **enviar** , ya que en PHP todas las variables se les refiere con este signo.

Hay que tener en cuenta que si fusionáramos el código de ambos ficheros, nos ahorraríamos uno, pero no también se puede hacer en dos como lo estamos haciendo. Si la variable $enviar está vacia, enviamos el formulario.

```
<?PHP;

if ($enviar) {

echo "Hola <b>" . $nombre . "</b> que tal estás<BR>n";

echo "Eres " . $sexo . "<BR>n";

echo "Tienes " . $edad . "<BR>n";

echo "Tu sistema favorito es " . $sistema . "<BR>n";

if ($futbol) {

echo "Te gusta el futbol <BR>n";

} else {

echo "NO te gusta el futbol <BR>n";

}
```

```
if ($aficiones != "") {

< stuot;)>

echo "Tus aficiones son: <BR>n";

echo nl2br($aficiones);

} else {

echo "NO tienes aficiones <BR>n";

}

echo "<a href='$PHP_SELF'>VOLVER AL FORMULARIO</a>"

} else {

<HTML>

<BODY>

<FORM METHOD="post" ACTION="<?PHP echo $PHP_SELF ?>">

<input type="hidden" name="edad" value="55">

<p>Tu nombre <input type="text" name="nombre" size="30" nombre" size="30" value="jose"></p>

<p>Tu sistema favorito

<select size="1" name="sistema">

<option selected value="Linux">Linux</option>

<option value="Unix">Unix</option>

<option value="Macintosh">Macintosh</option>

<option value="Windows">Windows</option>

</select></p>

<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol" value="ON"></p>

<p>¿Cual es tu sexo?</p>

<blockquote>

<p>Hombre<input type="radio" value="hombre" checked name="sexo"></p>

<p>="codigo" style="margin-left: 100"><p>Mujer <input type="radio" name="sexo" value="mujer"></p>

</blockquote>

<p>Aficiones</p>

<p><textarea rows="5" name="aficiones" cols="28"></textarea></p>

<p><input type="submit" value="Enviar datos" name="enviar">
```

```
<input type="reset" value="Restablecer" name="B2"></p>

</FORM>

</BODY>

</HTML>

<?PHP

} //fin IF

?>
```

La variable de entorno $PHP_SELF , es una variable de entorno que nos devuelve el nombre del script que estamos ejecutando. Y por último, hacer notar el uso de la función nl2br() , nl2br() , con la cuál sustituimos los retornos de carro del texto, los cuáles no reconocen los navegadores, por la etiqueta
 .

Descarga de ficheros desde un formulario

Vamos a ver un caso especial, como descargar un archivo desde un formulario. Para ello utilizaremos una etiqueta INPUT de tipo FILE , soportada a partir de las versiones de los navegadores Nestcape Navigator 2.0 e Internet Explorer 4.0.

El formulario debe usar el método post , y el atributo post , y el atributo enctype debe tener el valor multipart/form-data . Además al formulario debemos añadirle un campo oculto de nombre MAX_FILE_SIZE , al cuál le daremos el valor en bytes del tamaño máximo del archivo a descargar.

```
<FORM ENCTYPE="multipart/form-data" ACTION="7-3.php" METHOD="post">

<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">

<INPUT NAME="archivo" TYPE="file">

<INPUT TYPE="submit" VALUE="Descargar Archivo">

</FORM>
```

Cuando el formulario es enviado, PHP detectará automáticamente que se está descargando un archivo y lo colocará en un directorio temporal en el servidor. Dicho directorio será que el que esté indicado en el archivo de configuración **php.ini** , o en su defecto en el directorio temporal del sistema.

Cuando PHP detecta que se está descargando un archivo crea varias variables con el prefijo del nombre del archivo pero con distintas terminaciones. La variable terminada en $_FILES['archivo']['name'] contiene el nombre original del archivo, $_FILES['archivo']['size'] contiene el tamaño en bytes de éste, y la variable $_FILES['archivo']['type'] nos indicará el tipo de archivo si éste es ofrecido por el navegador.

Si el proceso de descarga no ha sido correcto la variable archivo tomará el valor none y _size será 0 , y si el proceso ha sido correcto, pero la variable $_FILES['archivo']['size'] da 0 , quiere decir que el archivo a descarga supera el tamaño máximo indicado por MAX_FILE_SIZE .

Una vez descargado el archivo, lo primero que debemos hacer es moverlo a otro lugar, pues sino se hace nada con él, cuando acabe la ejecución de la página se borrará.

Veamos un ejemplo de todo lo dicho.

```
<HTML>

<BODY>

<?PHP
```

```
if (isset($_POST['enviar']) {
if ($_FILES['archivo']['name'] != "" && $_FILES['archivo']['size'] != 0){
echo "Nombre: $archivo_name <BR>n";
echo "Tamaño: $archivo_size <BR>n";
echo "Tipo: $archivo_type <BR>n";
if (! move_uploaded_file ($_FILES['archivo']['tmp_name'], "directorio/".$_FILES['archivo']['name'])) {
echo "<h2>No se ha podido copiar el archivo</h2>n";
}
} elseif ($_FILES['archivo']['name'] != "" && $FILES['archivo']['size'] == 0) {
echo "<h2>Tamaño de archivo superado</h2>n";
} else {
echo "<h2>No ha escogido un archivo para descargar</h2>n";
}
echo "<HR>n";
}
?>
<FORM ENCTYPE="multipart/form-data" ACTION="<?php echo $_SERVER['PHP_SELF']; ?>" METHOD="post">
<INPUT type="hidden" name="MAX_FILE_SIZE" value="100000">
<p><b>Archivo a descargar<b><br>
<INPUT type="file" name="archivo" size="35"></p>
<p><INPUT type="submit" name="enviar" value="Aceptar"></p>
</FORM>
</BODY>
</HTML>
```

Practica con la fecha

Imprime la fecha de ayer en formato dd/mm/YYYY.

Ver Solución.

```
<?php

echo date("d/m/Y",time()-86400);

?>
```

Práctica con Matrices (array)

Crea un array con 5 números y realiza la suma de todos ellos.

Ver Solución.

```
<?php

$nums = array(5,12,76,2,5);

$sum = 0;

for($i=0;$i<count($nums);$i++) {

        $sum += $nums[$i];

}

echo $sum;
```

Practica con Fechas

¿Cómo podemos hacer que se muestre la fecha de ayer?

Ver Solución.

```
<?php

echo date("d-m-Y H:i:s",time()-86400); /* time() da el timestamp actual, un día tiene 86400 segundos, tan sólo tenemos que restárselo al timestamp de hoy */

?>
```

Practica con las funciones

Realiza una función que transforme de centímetros a pulgadas y viceversa. La función debe operar de esta forma:

echo distancia(0.39,"cm"); // Si ponemos "cm" entonces convierte de pulgadas a centímetros

echo distancia(2.54,"inch"); // Si ponemos "inch" entonces convierte de centímetros a pulgadas.

Si no ponemos nada en el segundo parámetro, echo distancia(0.39); debe convertir a centímetros.

1 inch = 2.54cm

Ver Solución.

```
<?php
function distancia($dis, $tipo='cm') {
	if($tipo == "cm") {
		return $dist*2.54;
	} else if($tipo == "inch") {
		return $dis/2.54;
	} else {
		return "Error";
	}
}
?>
```

Practica con las estructuras de Control

Tenemos dos variables, $a y $b, le damos un valor a cada una, el valor más pequeño a $a y el mayor a $b, por ejemplo 8 y 16, queremos que aparezcan todos los números desde $a hasta $b excepto los números múltiples de 10. En el caso de 3 y 16 el resultado debe ser:

8
9
11
12
13
14
15
16

¿Cómo lo harías en PHP?

Ver Solución.

```
<?php

$a = 8;

$b = 16;

for($i=$a;$i<=$b;$i++) {

        if($i % 10 != 0) {

                echo $i."<br>";

        }

}

?>
```

Practica con los Operadores

Crea dos variables, $a y $b y dale un valor a cada una, por ejemplo 8 y 17. Guarda en las variables $suma, $resta, $multiplicacion, $division, $resto el resultado de la operación correspondiente al nombre de cada variable.

Ver Solución.

```
<?php

$suma = $a+$b;

$resta = $a-$b;

$multiplicacion = $a*$b;

$division = $a/$b;

$resto = $a%$b;

?>
```

Practica con las variables

Queremos almacenar en 3 variables la cadena "Hola Mundo", el número 55 y el valor booleano "true". Además queremos definir una constante que sea el número PI, 3.141593. ¿Cómo lo harías?

Ver Solución.

```
<?php

$saludo = "Hola Mundo"; // Las cadenas siempre entre comillas, ya sean dobles (") o simples (').

$numero = 55; // Los números se indican sin comillas.

$bool = true; // Los valores booleanos también se indican sin comillas y se suelen indicar en minúsculas.

DEFINE("PI",3.141593); /* Al definir la constante indicamos su nombre entre comillas pero cuando la utilicemos no la indicaremos entre comillas (por ejemplo echo PI; ) */

?>
```

FAQ

1) PHP es un lenguaje HTML-embeeded que te permite incluir un código de programación en las páginas html para producir un output dinámico en relación con las exigencias de los usuarios.

2) El código en PHP incluido en una página html se ejecuta en el servidor antes de que se envíe la página al usuario que lo pida. Para hacer esto es necesario que en el servidor se instale el paquete PHP que procede a la compilación del código.

3) Desde el punto de vista del funcionamiento, ninguna. Ambos son server-script engine que procesan las páginas html que contienen un código de programación específico.

4) No. Mientras que Asp usa una extensión de Visual Basic (VBScript), el lenguaje de programación de PHP es un conjunto de instrucciones y funciones que han sido inventadas por los reveladores. Sin embargo, algunas funciones/palabras clave tienen una sintaxis parecida a muchos de los lenguajes más utilizados de los últimos años (C, Perl, etc.).

5) Para empezar, es necesario descargar el paquete software del sitio **http://www.php.net** e instalarlo después en el servidor en el que se quiere operar, siguiendo las instrucciones adjuntas.

6) PHP no cuesta nada. Cualquiera lo puede utilizar para sus sitios, después de haber leído la General Public License (**http://www.gnu.org**).

7) Es suficiente introducir el código en los tag: "<?PHP" e "?>". El código será ejecutado por el servidor cuando se cargue la página.

8) Basta con crear un archivo que contenga la línea:

<? PHPinfo(); ?>

Cargando la página con el propio navegador se tendrían que ver una serie de informaciones sobre PHP. Si no es así, significa que PHP no está presente en el servidor.

9) Existen manuales disponibles gratuitamente en el url http://www.php.net/docs.PHP3. Por desgracia, por el momento no existen todavía traducciones de los manuales al italiano.

10) Siguiendo las instrucciones indicadas en los archivos Readme o Readme.txt de la distribución del programa. Las instrucciones son muy simples, aunque la instalación en ámbito Windows necesita que se copien manualmente algunos archivos.

11) Todas las opciones de PHP se pueden settar a través del archivo "PHP.ini" de la distribución del programa.

12) Por supuesto. PHP funciona tanto con Windows como con Linux/Unix, a no ser que no se configure correctamente el servidor de web instalado, de modo que compile el código PHP.

13) Probablemente tu servidor web no está configurado correctamente para ejecutar el código PHP. Con Apache basta con incluir la línea:

AddType application/x-httpd-PHP3 .PHP3

en el gile httpd.conf, teniendo cuidado de que las páginas que contienen el código tengan extensión ".PHP3".
Con los servidores web de Windows hay varias posibilidades, para las cuales aconsejamos que se ejecuten bien las notas de instalación entregadas con ella y, en todo caso, el manual **http.//www.php.net/manual**.

14) Para empezar, intenta averiguar cuál es la línea que ha generado el error. Después, consulta el manual para saber si la sintaxis que has utilizado es correcta.
En último término, si te parece que has descubierto un bug en el programa, puedes comparar tu impresión con los bugs señalados por los usuarios en el sitio **http://bugs.PHP.net**.

15) Sí. Normalmente se utiliza el driver ODBC a través de las funciones ODBC puestas a disposición por PHP. Lo importante es configurar el driver ODBC desde el panel de control, de modo que acceda a las bases de datos que se quieren usar.

16) Existe una serie de funciones puestas a disposición por las más importantes bases de datos que hay en el mercado. PHP permite la interfaz a Oracle, MS Access, MySql, PostgreSql, Informix, Sybase y algunos más.

17) Se distribuyen las versiones binarias además de las fuentes que se tienen que compilar. Además, en algunos sitios están a disposición también los rpm.

18) Cuidado: si se quiere usar PHP con MySql hay que configurar PHP incluyendo la opción:

--with-mysql

y hay que volver a configurar Apache con la opción:

--activate-module=src/modules/PHP3/libPHP3.a

para otros problemas de instalación, el manual es bastante preciso.

19) Muy probablemente se trata de un error. Para evitar molestos inconvenientes, es preferible usar la sintaxis:

mysql_query($texto_query_sql) or die("Error:".mysql_error());

De esta manera, en caso de error, la query se suspende y se visualiza el error específico.

20) En la mayor parte de los casos, se está intentando acceder al resultado de una query que, por algún motivo, no va bien. Controlad la query y eventuales errores de sintaxis en el SQL.

21) En el url **http://www.php.net/manual** están disponibles manuales continuamente actualizados y en los que se incluyen los comentarios de los usuarios. Si no, se puede hojear http://bugs.PHP.net.

22) Muy probablemente no ha sido configurado correctamente el servidor web para hacer que las páginas que contienen códigos en PHP se compilen correctamente.

23) Sí, PHP soporta sesiones de forma nativa. **http://www.php.net/manual/en/ref.session.PHP**

24) Por supuesto. En PHP exiete la orden mail() que permite esta operación.

25) PHP permite la manipulación de archivos en el disco a todos los niveles: lectura, escritura, ejecución, etc. Hay funciones específicas para cada exigencia.

26) Con la instrucción "incluye("/path/nomefile.txt");"

27) Con la instrucción: date("d/m/Y");

28) Si una form pasa la variable <input name="campo" type="text"> a otro archivo, será disponible en la variable $_GET ['campo'] o $_POST['campo'] según el tio de formulario.

29) Con las variables de sistema puestas a disposición por PHP, que se pueden ver con la instrucción: "PHPinfo();"

30) Claro que sí, mira **aquí**. http://www.manualdephp.com/codigos-php/indice-codigos.html

Usuarios activos con PHP

Calculamos de una forma sencilla el número de visitantes presentes en nuestro sitio.

En nuestro manual de PHP abordamos en su momento el **uso de sesiones** y dimos algún ejemplo práctico en el que este tipo de variables pueden ser utilizadas para dar a nuestro sitio un aspecto más dinámico.

Muchos de vosotros habéis podido ver en ciertos sitios un contador de usuarios que se encuentran en ese momento navegando por las mismas páginas que vosotros. Si os habéis fijado bien, habréis podido observar que, para la mayoría de los casos (sino la totalidad), el sitio en cuestión esta programado con ASP como lenguaje de servidor.

Efectivamente, ASP permite una gestión más accesible de las sesiones por medio del famoso archivo global.asa. Esto no quiere decir sin embargo, que PHP es incapaz de realizar el mismo tipo de tareas sino que, más bien, hemos de trabajar un poco más para conseguirlas. En efecto, todos los lenguajes tienen sus pros y sus contras y, en este caso particular, ASP resulta más versátil aunque hay que admitir que, con su versión 4, PHP ha mejorado mucho en todo lo que respecta al tratamiento de sesiones.

Aquí os mostramos una forma sencilla de contabilizar los usuarios activos de vuestro sitio usando para ese propósito una tabla. Dentro de dicha tabla, iremos almacenando los distintos números IP de los visitantes de nuestro sitio y la hora y fecha en la que el visitante ha ejecutado por ultima vez el script.

Asimismo, la tabla ha de ir borrando progresivamente las sesiones que no hayan sido renovadas en un tiempo que nosotros consideremos límite. Nosotros hemos fijado dicho límite en 24 minutos que es el tiempo máximo tomado por defecto por PHP para suprimir los datos de una sesión. Para modificar este tiempo de vida máxima de una sesión puede hacerse en el *php.ini* a partir del parámetro *session.gc_maxlifetime* donde expresaremos dicho plazo en segundos. Ojo, este tiempo máximo es restaurado a su valor inicial cada vez que el usuario realiza una petición al servidor, esto quiere decir que un visitante podrá navegar cuanto tiempo quiera por el sitio guardando la misma sesión siempre y cuando no se quede más de 24 minutos sin realizar ningún tipo de acción.

Para el correcto funcionamiento del script, es necesario antes de nada crear una tabla en nuestra base de datos. Esta sentencia SQL puede ayudaros en la tarea:

```
CREATE TABLE control_ip (
ip VARCHAR(15) NOT NULL,
fecha INT(14) UNSIGNED NOT NULL,
INDEX (ip)
);
```

Como veis, el campo ip, que almacena el número IP del visitante, está indexado. Esto nos permitirá una selección rápida. En contrapartida, como todo campo indexado, su tamaño en memoria será doblado lo cual no tiene mucha importancia ya que la tabla tendrá un número de registros bastante limitado. Lo importante en efecto es que el script se ejecute rápidamente sin consumir demasiados recursos del servidor, sobretodo teniendo en cuenta que se trata de un código que será sistemáticamente ejecutado en cada una de las páginas del sitio.

Pasemos a continuación a mostrar el script que utilizaremos:

```
<?
////////////////////////////////////////////
//USUARIOS ACTIVOS
//Calcula el numero de usuarios activos
////////////////////////////////////////////

function usuarios_activos()
{
  //permitimos el uso de la variable portadora del numero ip en nuestra funcion
  global $REMOTE_ADDR;

  //asignamos un nombre memotecnico a la variable
  $ip = $REMOTE_ADDR;
  //definimos el momento actual
  $ahora = time();

  //conectamos a la base de datos
  //Usad vuestros propios parametros!!
  $conn = mysql_connect($host,$user,$password);
  mysql_select_db($db,$conn);

  //actualizamos la tabla
  //borrando los registros de las ip inactivas (24 minutos)
  $limite = $ahora-24*60;
```

```
  $ssql = "delete from control_ip where fecha < ".$limite;
  mysql_query($ssql);

  //miramos si el ip del visitante existe en nuestra tabla
  $ssql = "select ip, fecha from control_ip where ip = '$ip'";
  $result = mysql_query($ssql);

  //si existe actualizamos el campo fecha
  if (mysql_num_rows($result) != 0) $ssql = "update control_ip set fecha = ".$ahora." where ip = '$ip'";
  //si no existe insertamos el registro correspondiente a la nueva sesion
  else $ssql = "insert into control_ip (ip, fecha) values ('$ip', $ahora)";

  //ejecutamos la sentencia sql
  mysql_query($ssql);

  //calculamos el numero de sesiones
  $ssql = "select ip from control_ip";
  $result = mysql_query($ssql);
  $usuarios = mysql_num_rows($result);

  //liberamos memoria
  mysql_free_result($result);

  //devolvemos el resultado
  return $usuarios;
}
?>
```

Podéis observar, como viene siendo norma, que el script es expresado en forma de función. Después de definir la IP y el momento en el que el script está siendo ejecutado, pasamos a interaccionar con la base de datos. Dentro de este proceso, podemos distinguir distintas fases:

Conexión con la base de datos
Barrido de la tabla para eliminar los registros obsoletos.
Inserción o actualización de un registro dependiendo de si el visitante es nuevo o no.
Cómputo del número de registros de la tabla

La función finaliza liberando el espacio de memoria utilizado en sus consultas y enviando el resultado del cálculo efectuado.

En líneas generales el script es de fácil comprensión. Puede que alguna de las funciones empleadas os sea desconocida. Si es así, acudid al la **pagina oficial de PHP** donde podréis encontrar detalles en cuanto a su empleo.

Para sacar el valor proporcionado por la función a nuestro script principal tendremos que realizar una llamada clásica del tipo:

$active_users = usuarios_activos();

He aquí en definitiva un script sencillo que puede dar a vuestro sitio una imagen un poco más dinámica. Además, podéis utilizarlo y mejorarlo para crear vuestro propio sistema de estadísticas internas. Eso os lo dejamos a vosotros...

Rubén Alvarez
http://www.desarrolloweb.com/articulos/615.php?manual=6

Mostrar Contenido de acuerdo al pais en php

El siguiente script para PHP envía el código del país y del lenguaje cuando el navegador lo solicita.

Muchas veces, y sobre todo los webmaster internacionales, nos hemos encontrado con la necesidad de mostrar un contenido de acuerdo al pais y los idiomas.

En este Taller de PHP vamos a ver un sencillo código que permitirá conocer el país del usuario y mostrar una página distinta para cada caso.

```
<?php

// cambia pagina.php por el archivo correcto de acuerdo al pais

if (isset($pais) && ($pais <> '')) {
$dgo_pais = substr($pais,0,5);
} else {
$dgo_pais = substr($_SERVER["HTTP_ACCEPT_LANGUAGE"],0,5);
}
switch ($dgo_pais) {

case "es-mx":
//si el pais es mexico
include("pagina.php");
break;

case "es-ar":
//si el pais es argentina
include("pagina.php");
break;

case "es-cl":
//si el pais es chile
include("pagina.php");
break;

case "es-ve":
//si el pais es venezuela
include("pagina.php");
break;

case "pt-br":
//si el pais es brasil
include("pagina.php");
break;

case "es-co":

//si el pais es colombia
include("pagina.php");
break;

case "es-ec":

//si el pais es ecuador
include("pagina.php");
break;

case "es-uy":

//si el pais es uruguay
include("pagina.php");
break;

default:
//si es algun otro pais
include("pagina.php");
break;
}
?>
```

Recuerda cambiar pagina.php por la pagina de cada pais, por ejemplo, pagina-mexico.php. Se puede usar este script como pagina principal, para que sea distinta en función del país.

Aaron Gomez Perez.
http://www.elwey.com/

Crear un log de errores

Como crear un archivo que almacena los errores que se han producido durante la ejecución de un programa, añadir un log de errores a nuestra página.

Un log de errores, nos permitirá controlar cuando se ha producido un error para corregirlo y evitar que se repita en el futuro.

Para crear un log, abriremos el archivo en modo 'a' (escritura al final) y escribiremos el error indicando la fecha, para simplificar el trabajo lo podemos incluir todo en una función:

```
<?php
function error($numero,$texto){
$ddf = fopen('error.log','a');
fwrite($ddf,"[".date("r")."] Error $numero: $texto\r\n");
fclose($ddf);
}
?>
```

Una vez declarada la función, tansolo tendremos que llamarla de la siguiente manera cuando se produzca un error para que se guarde en error.log:

```
<?php
// Si no existe la cookie sesion
if(!isset($_COOKIE['sesion'])){
// Guardamos un error
error('001','No existe la cookie de sesion');
}
?>
```

De esta manera, cada vez que un usuario entra a esta página sin la cookie sesion, se almacena una nueva línea en el fichero indicando:

[fecha] Error 001: No existe la cookie de sesion

Vamos a ver ahora como podemos mejorar esto de manera que ademas de poder grabar los errores que nosotros definamos en nuestro sitio, nos almacene los errores producidos durante la ejecución del script php.

Esto lo conseguiremos indicando al interprete Zend que llame a la función error() cada vez que el código PHP contenga un error con la función set_error_handler:

```
<?php
set_error_handler('error');
?>
```

Entonces, el código completo nos queda de la siguiente manera:

```
<?php
function error($numero,$texto){
$ddf = fopen('error.log','a');
fwrite($ddf,"[".date("r")."] Error $numero:$texto\r\n");
fclose($ddf);
}
set_error_handler('error');
?>
```

Y de esta manera damos por finalizado nuestro script para generar un log de errores personales y de PHP.

Eloi de San Martín
http://www.programacionweb.net/articulos/articulo/?num=264

Formateo de una cadena

Script PHP que formatea una cadena a introducir en una base de datos, para asegurarse de que no estropea una sentencia SQL.

¿Cuántas veces has estado temeroso de que una cadena que vas a introducir en un campo de una tabla no tenga el formato correcto, y la has revisado, y vuelto a revisar... y luego descubres que ha dado un error al introducirla porque había caracteres no válidos?

Pues ya puedes respirar con alivio porque con el script que te proporcionamos tus dolores de cabeza tienen los días contados:

```
function str_asegurar($cadena){
//elimino etiquetas HTML y PHP
$cadena = strip_tags($cadena);
//elimino el caracter comilla, que puede estropear una sentencia
$cadena = str_replace("'","´",$cadena);

return $cadena;
}
```

Después de haber ejecutado el script, sólo queda realizar la sentencia de inserción.

Miguel Angel Alvarez
http://www.desarrolloweb.com/articulos/1440.php?manual=6

Detectar país del visitante

Script para conocer el país del visitante de tu página web en php.

Una de las grandes necesidades de todo desarrollador web radica al momento de conocer el país del visitante bien sea para redireccionarlos a módulos o páginas con particularidades o características propias de cada determinación, entre otras funciones que se pueden desarrollar en el entorno.

Ejemplo: Si tenemos una página de productos y catálogos (Comercio Web) y deseamos mostrare al visitante el precio del referido producto o artículo expresado monetariamente en su moneda local.

Entre otras infinidades de particularidades.

En realidad el proceso de la detección del país a través del nick se realiza a partir de la dirección IP devuelta por superglobales como $_SERVER['REMOTE_ADDR'], tomando como soporte o base una serie criterios de posibilidades almacenados lógicamente en archivos de base de datos, para su posterior validación y deducción de sus cuatros componentes esenciales menores todos a 255.

En Php solo existen alguna funciones que permiten obtener información acerca de maquinas conectadas a una red especifica por medio de las Funciones de Red, pero no creamos que una función o un par de funciones nos harán todo el trabajo; por el contrario solo serán bases para todo el ensamblaje.

Ciertamente la empresa Maxmind, desarrolladora por excelencia de este tipo de herramientas con soporte para diferentes tecnologías “GeoIp”, logro solucionar este gran problema con un 95% de efectividad con un conjunto de funciones y un archivo de bases de datos.

Para poder utilizar esta herramienta deberemos descargar la base de datos:
GeoIP.dat.gz

Biblioteca de vínculos y funciones:
Geoip.inc

Fianlemente realizaremos un test con las siguientes lineas:

```
<?
require("geoip.inc"); /*requerimos la biblioteca o liberia */
$abir_bd = geoip_open("GeoIP.dat",GEOIP_STANDARD); /* apertura y lectura del archivo utilizando la constante GEOIP_STANDARD como forma de lectura que adquiere valor en geoip.inc */
/* imprimimos el nick del país */
echo geoip_country_name_by_addr($abir_bd, $_SERVER['REMOTE_ADDR']);
/* liberamos memoria cerrando el resorce devuelto por $abir_bd de la apertura*/
geoip_close($abir_bd);
?>
```

Desventajas y Sugerencias:

Seria un gravísimo error que incluyeramos estas lineas en todas nuestra páginas, recordemos que cada vez que esta se carge llamara a si mismo el archivo geoip.inc y GeoIP.dat, este ultimo supera los 700 Kb de tamaño y puede considerablemente extender el tiempo de carga de la página.

Sugerencias:
Podemos almacenar el valor del nick o código del país en una session ($_SESSION) preferiblemente, o en su defecto en COOKIES; para que de este modo solo se lea una vez ya que lógicamente el visitante no cambiara de IP a cada vez que refresque o navegue en una página.

Emmanuel García De Caro
http://www.blasten.com/mostrar.php/articulos/descarga/18995/?accion=recomendar

Uso del FTP con PHP

Se explican las funciones para realizar transferencia de archivos, utilizando el protocolo FTP, desde páginas PHP. Ejemplo de código para aprender a subir archivos al servidor.

Constantemente nos encontramos bajando archivos de Internet: Un programilla que alguien nos recomienda, la foto de la modelo de moda o los clásicos MP3's . Sin darnos cuenta de uno u otro modo hacemos uso del protocolo FTP (File Transfer Prococol) para bajar archivos desde un Servidor. Descargamos más de los que subimos... en otras palabras "Tomamos más de los que damos". Esté artículo pretende explicarle el cómo subir -o permitir que sus usuarios suban (aporten) - archivos al Servidor usando las funciones FTP incluidas en el PHP. Recuerden el viejo dicho que cita: "Mas vale dar que recibir".

¿QUE ES EL FTP?

Siglas de File Transfer Protocol o Protocolo de Transferencia de Archivos. Como su propio nombre lo indica, es un protocolo (perteneciente a TCP/IP) que se encarga de la transferencia de archivos entre computadoras conectadas en red. Al basarse en la arquitectura Cliente / Servidor, el FTP hace uso de dos componentes básicos:

Un cliente FTP. El cual se encarga de conectarse a un servidor FTP para descargar o subir archivos.
Un servidor FTP. Se encarga de procesar las peticiones de los clientes FTP, permitiéndoles descargar o subir archivos desde él.

Para conectarnos a un servidor FTP, y así poder realizar consultas en él, necesitaremos los siguientes datos:

Nombre del Servidor. Es la IP o Nombre del Servidor FTP al cual nos hemos de conectar, tal como: 65.134.10.5 o ftp.billysite.net
Puerto. Número del puerto del servidor. Por defecto es 21.
Cuenta de Usuario. Es el nombre de la cuenta de usario que se nos ha asignado. Hay que asegurarse que cuenta con los permisos necesarios para subir o bajar archivos. De no tener una cuenta de usuario se puede acceder como usuario anónimo utilizando el nombre de usuario *anonymous.*
Clave de acceso. Es nuestra contraseña de cuenta de usuario. De acceder como usuario anónimo colocaremos como clave nuestro correo-e a manera de cortesía.

Una vez conectados al servidor FTP podremos hacer uso de su comandos para realizar las tareas que mejor creamos conveniente. Este artículo no pretende esbozar en gran medida el tema de FTP, al no ser éste el objetivo fundamental del mismo. Para més información sobre este aspecto le recomiendo revisar el árticulo de Michael Calore: "El ABC de la transferencia de archivos por Internet", disponible en el web site de **WebMonkey**.

FUNCIONES FTP EN PHP.

PHP hace uso de funciones FTP para acceder aun servidor web, a manera de cliente. A continuación mostraremos las funciones básicas a usar en el script, así como una breve descripción de las misma. Si quiere mayor detalle de estas y otras funciones FTP le aconsejo consultar la documentación oficial del PHP, disponible en: **http://www.php.net/docs.php**.

Función	Sintaxis	Descripción
ftp_connect	int ftp_connect (string host [, int port]) **host**: Nombre o IP de Servidor FTP. **port**: Puerto, por defecto 21.	Establece una conexión FTP al host especificado.
ftp_login	int ftp_login(int ftp_stream, string username, string password) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **username**: Nombre de usuario. **password**: constraseña de usuario.	Comienza la sesion en una conexión FTP.
ftp_pasv	int ftp_pasv (int ftp_stream, int pasv) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **pasv**: Si es TRUE activa el modo pasivo, si es FALSE lo desactiva.	Activa o desactiva el modo pasivo. En modo pasivo, las conexiones de datos son iniciadas por el cliente, en lugar de ser iniciadas por el servidor.
ftp_pwd	int ftp_pwd (int ftp_stream) **ftp_stream**: Manejador FTP obtenido con ftp_connect.	Devuelve el nombre del directorio actual.
ftp_put	int ftp_put (int ftp_stream, string remote_file, string local_file, int mode) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **remote_file**: Nombre con el cual se guardará el archivo en el Servidor FTP.	Sube un fichero al Servidor FTP.

	local_file: Archivo local que se encuentra en la máquina cliente. **mode**: Modo de transferencia, puede ser FTP_ASCII o FTP_BINARY.	
ftp_nlist	int ftp_nlist (int ftp_stream, string directory) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **directory**: Ruta del directorio a listar.	Devuelve una lista de ficheros del directorio dado.
ftp_size	int ftp_size (int ftp_stream, string remote_file) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **remote_file**: Nombre del archivo en el Servidor FTP.	Devuelve el tamaño del fichero especificado. No todos los servidores soportan esta característica.
ftp_mdtm	int ftp_mdtm (int ftp_stream, string remote_file) **ftp_stream**: Manejador FTP obtenido con ftp_connect. **remote_file**: Nombre del archivo en el Servidor FTP.	Devuelve la fecha de última modificación del fichero especificado. No todos los servidores soportan esta característica
ftp_quit	int ftp_quit (int ftp_stream) **ftp_stream**: Manejador FTP obtenido con ftp_connect.	Cierra una conexión FTP

Nota: Debe asegurarse que se encuentren habilitadas las funciones ftp en la configuración de la version de PHP que posee y de tener los permisos necesarios en su cuenta FTP para subir y bajar archivos.

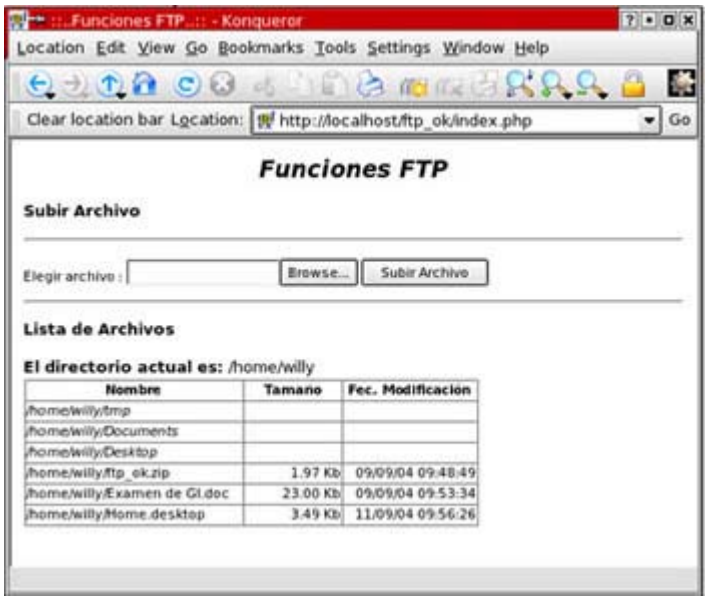
CODIGO FUENTE.

/inc/ftpfunc.php.
Script que contendrá las constantes y funciones a usar en **index.php**. En este script deberá modificar los valores de las constantes para ajustarlo a sus necesidades. La función **ConectarFTP** le permitirá conectarse al Servidor FTP; la función **SubirArchivo** tiene la tarea de subir un archivo al Servidor; y finalmente, la función **ObtenerRuta** le otorgará la ruta del directorio actual en el cual está trabajando el Servidor.

```
<?
# FUNCIONES FTP

# CONSTANTES
# Cambie estos datos por los de su Servidor FTP
define("SERVER","localhost"); //IP o Nombre del Servidor
define("PORT",21); //Puerto
define("USER","willy"); //Nombre de Usuario
define("PASSWORD","12345"); //Contraseña de acceso
define("PASV",true); //Activa modo pasivo

# FUNCIONES

function ConectarFTP(){
//Permite conectarse al Servidor FTP
$id_ftp=ftp_connect(SERVER,PORT); //Obtiene un manejador del Servidor FTP
ftp_login($id_ftp,USER,PASSWORD); //Se loguea al Servidor FTP
ftp_pasv($id_ftp,MODO); //Establece el modo de conexión
return $id_ftp; //Devuelve el manejador a la función
}

function SubirArchivo($archivo_local,$archivo_remoto){
//Sube archivo de la maquina Cliente al Servidor (Comando PUT)
$id_ftp=ConectarFTP(); //Obtiene un manejador y se conecta al Servidor FTP
ftp_put($id_ftp,$archivo_remoto,$archivo_local,FTP_BINARY);
//Sube un archivo al Servidor FTP en modo Binario
ftp_quit($id_ftp); //Cierra la conexion FTP
}

function ObtenerRuta(){
//Obriene ruta del directorio del Servidor FTP (Comando PWD)
$id_ftp=ConectarFTP(); //Obtiene un manejador y se conecta al Servidor FTP
$Directorio=ftp_pwd($id_ftp); //Devuelve ruta actual p.e. "/home/willy"
ftp_quit($id_ftp); //Cierra la conexion FTP
return $Directorio; //Devuelve la ruta a la función
```

```
}
?>
```

index.php.
Script que contiene un formulario (**form_ftp**) que nos permite buscar un archivo y subirlo al Servidor FTP, además nos muestra una lista de los directorios y archivos del mismo.




```
<?php echo "<?xml version="1.0" encoding="iso-8859-1"?".">"; ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>::..Funciones FTP..::</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>

<body>
<p align="center"><font size="5" face="Verdana, Tahoma, Arial"><strong><em>
Funciones FTP
</em></strong></font></p>
<p><font face="Verdana, Tahoma, Arial">

<?
include('inc/ftpfunc.php'); //Incluye el archivo de funciones
if(!empty($_POST["archivo"])){ //Comprueba si la variable "archivo" se ha definido
SubirArchivo($_POST["archivo"],basename($_POST["archivo"]));
//basename obtiene el nombre de archivo sin la ruta
unset($_POST["archivo"]); //Destruye la variable "archivo"
}
?>
<strong><font color="#000000" size="3">Subir Archivo</font></strong></font></p>
<hr />

<!--Formulario para elejir el archivo a subir -->
<form action="" method="post" name="form_ftp" id="form_ftp">
<p><font size="2" face="Verdana, Tahoma, Arial"> Elegir archivo :
<input name="archivo" type="file" id="archivo" />
<input name="Submit" type="submit" value="Subir Archivo" />
</font><font size="2" face="Verdana, Tahoma, Arial"> </font> </p>
</form>

<hr />
<p><font face="Verdana, Tahoma, Arial"><strong><font color="#000000" size="3">
Lista de Archivos
```

```
</font></strong></font></p>
<table width="69%" border="1" cellspacing="0" cellpadding="0">
<tr>
<td width="48%"><div align="center"><font size="2" face="Verdana, Tahoma,
Arial"><strong>Nombre</strong></font></div></td>
<td width="22%"><div align="center"><font size="2" face="Verdana, Tahoma,
Arial"><strong>Tama&ntilde;o</strong></font></div></td>
<td width="30%"><div align="center"><font size="2" face="Verdana, Tahoma, Arial"><strong>Fec.
Modificaci&oacute;n</strong></font></div></td>
</tr>
<?
$id_ftp=ConectarFTP(); //Obtiene un manejador y se conecta al Servidor FTP
$ruta=ObtenerRuta(); //Obtiene la ruta actual en el Servidor FTP
echo "<b>El directorio actual es: </b> ".$ruta;
$lista=ftp_nlist($id_ftp,$ruta); //Devuelve un array con los nombres de ficheros
$lista=array_reverse($lista); //Invierte orden del array (ordena array)
while ($item=array_pop($lista)) //Se leen todos los ficheros y directorios del directorio
{
$tamano=number_format(((ftp_size($id_ftp,$item))/1024),2)." Kb";
//Obtiene tamaño de archivo y lo pasa a KB
if($tamano=="-0.00 Kb") // Si es -0.00 Kb se refiere a un directorio
{
$item="<i>".$item."</i>";
$tamano=" ";
$fecha=" ";
}else{
$fecha=date("d/m/y h:i:s", ftp_mdtm($id_ftp,$item));
//Filemtime obtiene la fecha de modificacion del fichero; y date le da el formato de salida
}
?>

<tr>
<td><font size="2" face="Verdana, Tahoma, Arial"><? echo $item ?></font></td>
<td align="right"><font size="2" face="Verdana, Tahoma, Arial"><? echo $tamano ?></font></td>
<td align="right"><font size="2" face="Verdana, Tahoma, Arial"><? echo $fecha ?></font></td>
</tr>
<? } ?>
</table>
</body>
</html>
```

Bueno, espero que éste aporte a al Comunidad Web les haya sido de utilidad, cualquier duda o sugerencia no duden en hacermesla llegar. Saludos.

William Wong Garay
http://billysite.net

De Segundos a Horas, Minutos y Segundos

¿Como llevar de segundos a Horas (si las hay), Minutos (si los hay) y Segundos.

Por ejemplo si tengo 28240 Segundos, como hacer que imprima: 7:50:40 Horas

El proceso es sin duda sencillo, veamos el código y luego la explicación:

Código Fuente:

```
<?
function segundos_tiempo($segundos){
$minutos=$segundos/60;
$horas=floor($minutos/60);
$minutos2=$minutos%60;
$segundos_2=$segundos%60%60%60;
if($minutos2<10)$minutos2='0'.$minutos2;
if($segundos_2<10)$segundos_2='0'.$segundos_2;

if($segundos<60){ /* segundos */
$resultado= round($segundos).' Segundos';
}elseif($segundos>60 && $segundos<3600){/* minutos */
$resultado= $minutos2.':'.$segundos_2.' Minutos';
}else{/* horas */
$resultado= $horas.':'.$minutos2.':'.$segundos_2.' Horas';
}
return $resultado;
}
$segundos=date('h')*60*60+(date('i')*60)+date('s');

echo 'Segundos: '.$segundos.' Resultado: '.segundos_tiempo($segundos);
?>
```

Explicando:

Definimos una función con el nombre de segundos_tiempo(ARG1), donde el ARG1 es el valor que indicamos (segundos) cuando llamamos la función.

Realizaremos una serie de operaciones aritméticas, para luego condicionar el resultado.
$Minutos, tal como se ve; sera igual a la división entre $segundo y 60.
Ejemplo: 120 segundos / 60 es igual a 2 minutos.

$horas es igual a la división de $minutos entre 60 , y lo redondeamos hacia abajo con la función matemática floor()

$minutos2 será igual al residuo (resto) devuelto por la división entre $minutos y 60.

Y $segundos_2 el residuo de la división entre $segundos, 60, 60, 60 sucesivamente.

Que no son más que los datos acumulados en los procesos anteriores.

Luego iniciamos una condición para verificar si el número es menor a 10 y colocarle un 0 (cero) al comienzo: Por ejemplo 9 => 09 , el proceso lo repetimos con los segundos y minutos, con la horas seria innecesario.

Creamos otra condición para indicar si se ha definido Horas, minutos ó segundos y retornar el formato y estilo correcto.

Si $segundos es menor que 60 (entonces solo hay segundos)
if($segundos<60)

De lo contrario si $segundos es menor que 60 y es menor que 3600 (Se han definido minutos y segundos)
elseif($segundos>60 && $segundos<3600)

De lo contrario a todas estas (se ha definido horas, minutos y segundos)
else

Finalmente retornamos el resultado
return $resultado;

Y aplicamos la función, con los segundos actuales del servidor para verificar que nos devuelve la hora exacta.

Emmanuel García De Caro
http://www.blasten.com/contenidos/19077

Comprimir página PHP

Para aligerar el tiempo de carga de nuestras páginas generadas con PHP, podemos enviarlas al navegador comprimidas con GZip

Para aligerar el tiempo de carga de nuestras páginas generadas con PHP, podemos enviarlas al navegador comprimidas con GZip utilizando las funciones de control de salida, para ello, llamaremos a la funcion predefinida ob_gzhandler cómo tratante de la funcion ob_start, veamos un ejemplo:

```
<?
ob_start("ob_gzhandler");

// Contenido de la página, puede contener
// tanto HTML cómo PHP

ob_end_flush();
?>
```

Tener en cuenta que todo el contenido debe estar en el lugar indicado por lo que los primeros carácteres del documento deben ser <? y los dos últimos ?> y no se debe añadir nada excepto donde se indica, si no vamos con cuidado recibiremos un error parecido al siguiente:

Warning: Cannot add header information...

Otra forma más completa todavía de compresion, consiste en aplicar la misma funcion, pero eliminando a su vez los espacios y saltos de linea de la fuente del documento, lo que no tendra ningun efecto visual pero disminuira el tiempo de descarga, veamos cómo hacerlo:

```
<?
ob_start();

// Contenido de la página, puede contener
// tanto HTML cómo PHP

$cntACmp =ob_get_contents();
ob_end_clean();
$cntACmp=str_replace("\n",' ',$cntACmp);
$cntACmp=ereg_replace('[[:space:]]+',' ',$cntACmp);
ob_start("ob_gzhandler");
echo $cntACmp;
ob_end_flush();
?>
```

Este método funciona igual que el anterior solo que antes de comprimir elimina los saltos de linea y espacios innescesarios, pero debemos tener en cuenta las mismas precauciones o no funcionará.

Este método de compresion lo he desarrollado para ProgramacionWeb, como podreis comprovar al ver el código fuente de esta misma página, todo el código aparece sin saltos de linea ni tabulado por una simple cuestion de compresion.

Eloi de San Martin Lagranje
http://www.programacionweb.net/articulos/articulo/?num=162

Pasar las variables SESSION, POST Y GET a variables normales en PHP

Cómo hacer que unas variables globales como son GET, SESSION y POST se traten como variables normales en PHP.

Hace un par de semanas se me encomendó migrar un sitio (desarrollado en php) de servidor, el tema iba relativamente en paz hasta que di cuenta de un problema recurrente en este tipo de situaciones, el servidor antiguo tenia las variables globales en On y el nuevo las tenia en Off ... recurrente no ?

La verdad es que no quise complicarme la vida como otras veces, y me di cuenta de algo que me podría ahorrar un par de horas de trabajo, todos los php hacían un include a un archivo php, recurrente también no?
Asi es que me puse a pensar en un pedazo de código que me permitiera tomar estos 3 tipos de variables (SESSION, POST Y GET) y me las dejara como variables normales, si no lo tienen claro el ejercicio es mas o menos el siguiente :

Supongamos que la variable viene desde un formulario via POST, el código decía algo asi ...

```
<?

if ($variable = "algo")
{
   echo "esta variable hace alguna cosa";
}

?>
```

Es decir, le faltaba esta instrucción antes del if

```
<?php

$variable = $_POST["variable"]; // le faltaba esta instrucción antes del if

if ($variable = "algo")
{
   echo "esta variable hace alguna cosa";
}

?>
```

Tenia 2 opciones, o me ponía a buscar TODAS estas situaciones y las arreglaba una a una, o colocaba un pedazo de código en archivo al que todos los demás le hacían un include que me corrigiera este "error" propio de aquellos que estamos a trabajar con los "servidores en producción", o sea, con las variables globales en On.

Bueno, después de muchos cabecearme encontré la solución, y aquí la muestro a todos uds.

```
<?

if($_POST)
{
   $keys_post = array_keys($_POST);
   foreach ($keys_post as $key_post)
    {
     $$key_post = $_POST[$key_post];
     error_log("variable $key_post viene desde $ _POST");
    }
}

if($_GET)
{
   $keys_get = array_keys($_GET);
   foreach ($keys_get as $key_get)
    {
      $$key_get = $_GET[$key_get];
      error_log("variable $key_get viene desde $ _GET");
    }
}

if($_SESSION)
{
   $keys_sesion = array_keys($_SESSION);
   foreach ($keys_sesion as $key_sesion)
    {
      $$key_sesion = $_SESSION[$key_sesion];
```

```
      error_log("variable $key_sesion viene desde $ _SESSION");
    }
}

?>
```

Son básicamente 3 if que hacen lo mismo, toman las claves del arreglo en cuestión (SESSION , POST o GET) y generan una "variable variable", creo que hay un articulo en desarrolloweb que habla de este tema, y esto hace toda la magia, el error_log fue una implementación para ver que variables se iban ocupando (si no tienes acceso al servidor puedes sacarlo sin problemas).

Para alguien que guste de las funciones puede también guardarlo como función y llamarla cuando lo necesite. O también iterar el proceso con las 3 variables en cuestión para no tener que hacer 3 if, o pasarle cualquier arreglo para que haga el mismo proceso ... en fin, la idea esta, solo espero haber contribuido en algo al ahorro de tiempo de alguno de ustedes.

Juan Edgardo Jorquera Uribe
http://www.chile.com/

PHP Perfect Form Items v1.0 (Formularios sin errores)

¿Cuántas veces que hemos necesitado **integrar código PHP con elementos de formularios**, hemos conseguido una mezcla de HTML y etiquetas <?php ?> disminuyendo la legibilidad del código o haciendo que tengamos que escribir más lineas de la cuenta?

Con estas sencillas pero útiles funciones podremos sin problemas emplazar campos a los formularios que diseñemos sin tantos quebraderos de cabeza.

Las funciones que se explican en este artículo se encuentran en la siguiente dirección **http://www.distintiva.com/jose/_perf_form/perf_form.zip**. El código puede ser cambiado, mejorado y distribuido libremente. Ha sido programado totalmente por Jose Carlos García de Distintiva Solutions (**www.distintiva.com**).

Empecemos con el ejemplo más difícil y el causante de que tuviera que programar estas funciones.

<SELECT>

Crear elementos de selección <select></select>
Con php es muy habitual trabajar con arrays, ya sea con datos que nos proporcione una consulta SQL o simples valores.

Veamos un ejemplo en el que queremos mostrar un <select> para que el usuario seleccione su rango de edad:

```
<select name="edad">
<option value=0>Seleccione</option>
<option value=1>Entre 0-18</option>
<option value=2>Entre 19-30</option>
<option value=3>Entre 31-50</option>
<option value=4>Más de 50</option>
<select/>
```

Esto se complica cuando hay que hacerlo de forma dinámica ya que esta lista puede variar, para evitar tener que cambiar el HTML “a pelo” ya que necesitamos construir este fragmento de HTML con un bucle, etc.

Se complica aún más cuando una opción debe estar preseleccionada.
La función que nos salvará la vida es:

frm_select($name, $arr_txt, $arr_vals, $default='', $extra_tag='')

$**name** = Nombre del elemento del formulario
$**arr_txt** = Array con los textos a mostrar
$**arr_vals** = Array con los valores asociados a cada texto
$**default**= [opcional] si se indica el valor aparecerá preseleccionado dicha opción
$**extra_tag**= [opcional] por si necesitamos incluir información adicional al select, como por ejemplo $extra_tag=” class=cssazul ” o $extra_tag=” onChange=alert()”

Ejemplo:
$arr_txt=array(‘españa’, ‘portugal’, ‘francia’);
$arr_vals=array(‘ES’, ‘PT’, FR’);
<?=frm_select (‘paises’, $arr_txt, $arr_vals) ?>

De esta forma cualquier modificación sólo hay que hacerla en los arrays sin alterar la parte visual.

A veces es conveniente conservar el valor de un campo del formulario entre llamadas o posts del mismo por ejemplo cuando estamos validando entradas y hay que volver al formulario para que rellene algún campo requerido. Para hacer esto de forma sencilla tan solo tenemos que usar el parámetro $default de la siguiente forma:

<?=frm_select (‘paises’, $arr_txt, $arr_vals, $_POST[‘paises’]) ?>

(o $_GET dependiendo el método que usemos en nuestro formulario)

<SELECT> (tipo lista con varios elementos visibles)

Funciona exactamente igual que el anterior pero en este caso tenemos la típica lista de selección con scroll mostrando X elementos.

La función es:

frm_list($name,$size, $arr_txt, $arr_vals, $default='', $extra_tag='')

En este caso el nuevo parámetro es $size que indica cuántos elementos serán mostrados visiblemente en la lista.

<SELECT> (tipo lista con varios elementos visibles y con multiselección)

Igual que el ejemplo anterior pero permitimos que el usuario seleccione uno o varios elementos de la lista con el CTRL+Click o SHIFT+Click

La función es:

frm_list_multi($name, $size, $arr_txt, $arr_vals, $default='', $extra_tag='')

... Y también tenemos funciones correspondientes para los demás elementos de formulario aprovechando la capacidad de mantener el valor entre posts.

<CHECKBOX>

No hace falta decir para que sirven estos elementos. La función es:

frm_check($name, $ck_val, $var_in='', $extra_tag='')

$**name**= Nombre del campo
$**ck_val**= Valor que se enviará cuando esté seleccionado
$**var_in**=[opcional] Funciona como el $default y permitirá que se muestre checkeado

Ejemplo:
Se muestra un checkbox que se mantiene seleccionado entre posts del formulario
<?= frm_check('fumador', 'SI', $_POST['fumador'])?>

<RADIO>

En este caso para mostrar radiobuttons se hace exactamente como el caso anterior pero con la siguiente función:

frm_radio($name, $val, $var_in='', $extra_tag='')

<INPUT>

Permite mostrar cuadros de texto con las ventajas que voy ofreciendo en todas las funciones y para ello hay que usar:

frm_text($name, $val, $size, $max_length, $extra_tag='')

$**name** = Nombre del campo
$**val** = Valor o texto que se mostrará (funciona como el $default de las otras funciones)
$**size** = Tamaño del campo de texto
$**max_length** = Longitud máxima permitida

<PASSWORD>

El caso es exactamente como el anterior pero esta vez oculta el texto con asteriscos.

frm_password($name, $val, $size, $max_length, $extra_tag='')

Jose Carlos García
http://www.webestilo.com/php/articulo.phtml?art=48

Validar email en PHP

Comprobar la validez de una dirección de correo electrónico, es decir, validar la buen redacción de un email.

Vamos a ver una función muy útil en PHP que sirve para comprobar la validez de un correo. En realidad comprueba si una dirección de correo electrónico está bien escrita sintácticamente, dejando de lado las comprobaciones de si ese mail existe o no realmente, que no se pueden hacer tan fácilmente.

Vamos a escribir una función que se llama comprobar_email y recibe la cadena de texto con el email que queremos validar. Si dicho email es correcto desde el punto de vista sintáctico, es decir, si tiene un nombre de usuario, una arroba y una terminación con el nombre de un dominio o subdominio, etc, devolverá un 1, es decir, verdadero. En caso de que el email no esté correctamente escrito, la función devolvería 0, que equivale a falso.

La función en si da por hecho inicialmente que el email es erróneo y realiza una serie de comprobaciones que, si todas responden correctamente, dan por conclusión que el email sí estaba bien escrito. Si alguna de esas comprobaciones no era correcta, no se llegaría al final de las comprobaciones y quedaría el resultado como se ha supuesto en un principio, es decir, como incorrecto.

código de la función

```
function comprobar_email($email){
   $mail_correcto = 0;
   //compruebo unas cosas primeras
   if ((strlen($email) >= 6) && (substr_count($email,"@") == 1) && (substr($email,0,1) != "@") && (substr($email,strlen
($email)-1,1) != "@")){
      if ((!strstr($email,"'")) && (!strstr($email,"\"")) && (!strstr($email,"\\")) && (!strstr($email,"\$")) && (!strstr($email," "))) {
         //miro si tiene caracter .
         if (substr_count($email,".")>= 1){
            //obtengo la terminacion del dominio
            $term_dom = substr(strrchr ($email, '.'),1);
            //compruebo que la terminación del dominio sea correcta
            if (strlen($term_dom)>1 && strlen($term_dom)<5 && (!strstr($term_dom,"@")) ){
               //compruebo que lo de antes del dominio sea correcto
               $antes_dom = substr($email,0,strlen($email) - strlen($term_dom) - 1);
               $caracter_ult = substr($antes_dom,strlen($antes_dom)-1,1);
               if ($caracter_ult != "@" && $caracter_ult != "."){
                  $mail_correcto = 1;
               }
            }
         }
      }
   }
   if ($mail_correcto)
      return 1;
   else
      return 0;
}
```

Las comprobaciones

En el primer if compruebo que el email tiene por lo menos 6 caracteres (el mínimo), que tiene una arroba y sólo una y que no está colocada ni al principio ni al final.

En el segundo if comprueba que no tiene algunos caracteres no permitidos. Y los restantes hacen comprobaciones de las distintas partes de la dirección de correo, a saber: Que hay un punto en algún lado y que la terminación del dominio es correcta y que el principio de la dirección también es correcto.

Finalmente, se devuelve la variable local utilizada para guardar la validez o incorrección del correo.

Miguel Angel Alvarez
http://www.desarrolloweb.com/articulos/990.php?manual=6

Zonas horarias

Script PHP para sacar un menú donde poder elegir la zona horaria deseada.

Otras de las grandes necesidades originadas por la distancia entre diferentes países y en algunos casos ciudades es la hora.

Por ejemplo: ¿Si un usuario se encuentra en cualquier parte del mundo, como hacer para que cuando emita una opinión, esta aparezca con su hora local ó de su zona horaria?

Para solucionar este gran problema y tantos otros relacionados con este género utilizaremos las zonas horarias de las principales ciudades del mundo.

Utilizaremos dos aspectos cruciales o decisivos la hora con respecto al meridiano de la ciudad local utilizada como base para calcular la diferencia con respecto a la ciudad distante tomando como referencia la hora del Meridiano de Greenwich u hora 0 (CERO).

Veamos el código fuente y luego la explicación como de costumbre:

```
<?
if(isset($_POST[hora])){
   settype ($_POST[hora],"integer");
   settype ($_POST[min],"integer");
   settype ($_POST[seg],"integer");
   if($_POST[hora]<=0 OR $_POST[hora]>12){
      echo '<strong> Hora incorrecta </strong>';
   }elseif($_POST[min]<=0OR $_POST[min]>60){
      echo '<strong> Minutos incorrectos </strong>';
   }elseif($_POST[seg]<=0 OR $_POST[seg]>60){
      echo '<strong> Segundos Incorrectos </strong>';
   }else{
      $hor=$_POST[hora];
      if($_POST[tm]=='pm' && $_POST[hora]>=1 && $_POST[hora]<12){
        $_POST[hora]+=12;
   }
   $total_minutos_=floor($_POST[hora]*60)+$_POST[min];
   $dia="del mismo día";
   if($_POST[ciudad1]>=0){ // es positiva la zona h
      $gmt= (int) $total_minutos_-$_POST[ciudad1];
   }else{ // es negativo
   $_POST[ciudad1]= (int) $_POST[ciudad1]*-1;
   $gmt= $total_minutos_+$_POST[ciudad1];
   }
   if($gmt<0){
   $dia=' del día anterior';
   $gmt+=1440; //minutos de 1 dia
   }
   if($_POST[ciudad2]>0){
      $tiempo2=$_POST[ciudad2]+$total_minutos_;
   }else{
   $tiempo2=$_POST[ciudad2]+$gmt;
   if($tiempo2 > 1440)
   {
      $dia = "del dia siguiente";
      $tiempo2-= 1440;
   }
   if ($tiempo2 < 0)
   {
       $dia = "del dia anterior";
       $tiempo2 += 1440;
   }
   }
$hora_de_la_otra_ciudad=floor($tiempo2/60);

echo 'En la otra ciudad son: '.$hora_de_la_otra_ciudad.' : '.$_POST[min].' : '.$_POST[seg].' '.$dia;
  }
} ?>

<form action="<? echo $PHP_SELF?>" method="post">Si en mi pc son las
<input name="hora" type="text" id="hora" value="<? echo $hor?>" size="2" maxlength="2">
:
```

```
<input name="min" type="text" id="min" value="<? echo $_POST["min"] ?>" size="2" maxlength="2">
:
<input name="seg" type="text" id="seg" value="<? echo $_POST["seg"] ?>" size="2" maxlength="2">
<select name="tm" id="tm">
    <option value="am">am</option>
   <option value="pm">pm</option>
</select>
: y estoy en :

<SELECT name=ciudad1 size=1 id="ciudad1">
    <OPTION value=180>Addis Ababa</OPTION>
   <OPTION value=570>Adelaida</OPTION>
   <OPTION value=180>Aden</OPTION>
   <OPTION value=-360>Aklavik</OPTION>
   <OPTION value=60>Argel</OPTION>
   <OPTION value=120>Amman</OPTION>
   <OPTION value=60>Amsterdam</OPTION>
   <OPTION value=720>Anadyr</OPTION>
   <OPTION value=120>Ankara</OPTION>
   <OPTION value=180>Antananarivo</OPTION>
   <OPTION value=-240>Asunció</OPTION>
   <OPTION value=120>Atenas</OPTION>
   <OPTION value=-300>Atlanta</OPTION>
   <OPTION value=-360>Austin</OPTION>
   <OPTION value=0>Azores</OPTION>
   <OPTION value=180>Baghdad</OPTION>
   <OPTION value=420>Bangkok</OPTION>
   <OPTION value=60>Barcelona</OPTION>
   <OPTION value=180>Beirut</OPTION>
   <OPTION value=60>Belgrado</OPTION>
   <OPTION value=60>Berlín</OPTION>
   <OPTION value=-240>Bermuda</OPTION>
   <OPTION value=60>Berna</OPTION>
   <OPTION value=120>Biel</OPTION>
   <OPTION value=-300>Bogotá</OPTION>
   <OPTION value=-300>Boston</OPTION>
   <OPTION value=-180>Brasilia</OPTION>
   <OPTION value=600>Brisbane</OPTION>
   <OPTION value=60>Bruselas</OPTION>
   <OPTION value=120>Bucarest</OPTION>
   <OPTION value=60>Budapest</OPTION>
   <OPTION value=-180>Buenos Aires</OPTION>
   <OPTION value=330>Calcuta</OPTION>
   <OPTION value=-240 selected>Caracas</OPTION>
   <OPTION value=0>Casablanca</OPTION>
   <OPTION value=-300>Chicago</OPTION>
   <OPTION value=120>Ciudad del Cabo</OPTION>
   <OPTION value=60>Copenhague</OPTION>
   <OPTION value=-360>Dallas</OPTION>
   <OPTION value=180>Dar es Salaam</OPTION>
   <OPTION value=570>Darwin</OPTION>
   <OPTION value=-420>Denver</OPTION>
   <OPTION value=-300>Detroit</OPTION>
   <OPTION value=360>Dhaka</OPTION>
   <OPTION value=0>Dublín</OPTION>
   <OPTION value=-420>Edmonton</OPTION>
    <OPTION value=120>Estambul</OPTION>
   <OPTION value=60>Estocolmo</OPTION>
   <OPTION value=180>El Cairo</OPTION>
   <OPTION value=60>Francfort</OPTION>
   <OPTION value=60>Ginebra</OPTION>
   <OPTION value=-360>Guatemala</OPTION>
   <OPTION value=420>Hanoi</OPTION>
   <OPTION value=120>Harare</OPTION>
   <OPTION value=120>Helsinki</OPTION>
   <OPTION value=420>Ho Chi Minh City</OPTION>
   <OPTION value=480>Hong Kong</OPTION>
   <OPTION value=600>Honolulú</OPTION>
   <OPTION value=-360>Houston</OPTION>
   <OPTION value=-300>Indianápolis</OPTION>
   <OPTION value=300>Islamabad</OPTION>
   <OPTION value=420>Jakarta</OPTION>
```

```
<OPTION value=120>Jerusalén</OPTION>
<OPTION value=120>Johanesburgo</OPTION>
<OPTION value=270>Kabúl</OPTION>
<OPTION value=720>Kamchatka</OPTION>
<OPTION value=300>Kathmandú</OPTION>
<OPTION value=120>Khartoum</OPTION>
<OPTION value=120>Kigali</OPTION>
<OPTION value=-300>Kingston</OPTION>
<OPTION value=480>Kuala Lumpur</OPTION>
<OPTION value=180>Kuwait</OPTION>
<OPTION value=-240>La Habana</OPTION>
<OPTION value=-240>La Paz</OPTION>
<OPTION value=0>Las Palmas</OPTION>
<OPTION value=60>Lagos</OPTION>
<OPTION value=-300>Lima</OPTION>
<OPTION value=0>Lisboa</OPTION>
<OPTION value=0>Londres</OPTION>
<OPTION value=-480>Los Ángeles</OPTION>
<OPTION value=60>Madrid</OPTION>
<OPTION value=-360>Managua</OPTION>
<OPTION value=480>Manila</OPTION>
<OPTION value=240>Masqat</OPTION>
<OPTION value=600>Melburne</OPTION>
<OPTION value=-360>Méjico D.F.</OPTION>
<OPTION value=300>Miami</OPTION>
<OPTION value=60>Milán</OPTION>
<OPTION value=-360>Minneápolis</OPTION>
<OPTION value=-180>Montevideo</OPTION>
<OPTION value=-300>Montreal</OPTION>
<OPTION value=180>Moscú</OPTION>
<OPTION value=330>Mumbai</OPTION>
<OPTION value=60>Múnich</OPTION>
<OPTION value=180>Nairobi</OPTION>
<OPTION value=60>Nápoles</OPTION>
<OPTION value=-300>Nassau</OPTION>
<OPTION value=330>Nueva Delhi</OPTION>
<OPTION value=-360>Nueva Orleans</OPTION>
<OPTION value=-300>Nueva York</OPTION>
<OPTION value=60>Oslo</OPTION>
<OPTION value=-240>Ottawa</OPTION>
<OPTION value=60>París</OPTION>
<OPTION value=480>Pequín</OPTION>
<OPTION value=480>Perth</OPTION>
<OPTION value=-420>Phoenix</OPTION>
<OPTION value=420>Phnom Penh</OPTION>
<OPTION value=60>Praga</OPTION>
<OPTION value=540>Pyongyang</OPTION>
<OPTION value=0>Reikiavik</OPTION>
<OPTION value=-180>Río de Janeiro</OPTION>
<OPTION value=180>Riyadh</OPTION>
<OPTION value=60>Roma</OPTION>
<OPTION value=-240>St. John's</OPTION>
<OPTION value=-360>St. Paul</OPTION>
<OPTION value=-480>San Francisco</OPTION>
<OPTION value=-240>San Juan</OPTION>
<OPTION value=-360>San Salvador</OPTION>
<OPTION value=-300>Santo Domingo</OPTION>
<OPTION value=-240>Santiago</OPTION>
<OPTION value=-180>Sao Paulo</OPTION>
<OPTION value=-480>Seattle</OPTION>
<OPTION value=540>Seúl</OPTION>
<OPTION value=480>Shanghai</OPTION>
<OPTION value=480>Singapur</OPTION>
<OPTION value=60>Sofía</OPTION>
<OPTION value=720>Suva</OPTION>
<OPTION value=600>Sydney</OPTION>
<OPTION value=480>Taipei</OPTION>
<OPTION value=120>Tallinn</OPTION>
<OPTION value=300>Tashkent</OPTION>
<OPTION value=-210>Teherán</OPTION>
<OPTION value=540>Tokio</OPTION>
<OPTION value=-300>Toronto</OPTION>
```

```
   <OPTION value=60>Turín</OPTION>
   <OPTION value=-300>Vancúver</OPTION>
    <OPTION value=60>Varsovia</OPTION>
   <OPTION value=60>Venecia</OPTION>
   <OPTION value=60>Viena</OPTION>
   <OPTION value=600>Vladivostok</OPTION>
   <OPTION value=-300>Washington</OPTION>
   <OPTION value=720>Wellington</OPTION>
   <OPTION value=-360>Winnipeg</OPTION>
   <OPTION value=390>Yangon</OPTION>
   <OPTION value=60>Zagreb</OPTION>
   <OPTION value=60>Zurich</OPTION>
</SELECT>
<br>
Entonces en

<SELECT name=ciudad2 size=1 id="ciudad2">
   <OPTION value=180>Addis Ababa</OPTION>
   <OPTION value=570>Adelaida</OPTION>
   <OPTION value=180>Aden</OPTION>
   <OPTION value=-360>Aklavik</OPTION>
   <OPTION value=60>Argel</OPTION>
   <OPTION value=120>Amman</OPTION>
   <OPTION value=60>Amsterdam</OPTION>
   <OPTION value=720>Anadyr</OPTION>
   <OPTION value=120>Ankara</OPTION>
   <OPTION value=180>Antananarivo</OPTION>
   <OPTION value=-240>Asunció</OPTION>
   <OPTION value=120>Atenas</OPTION>
   <OPTION value=-300>Atlanta</OPTION>
   <OPTION value=-360>Austin</OPTION>
   <OPTION value=0>Azores</OPTION>
   <OPTION value=180>Baghdad</OPTION>
   <OPTION value=420>Bangkok</OPTION>
   <OPTION value=60>Barcelona</OPTION>
   <OPTION value=180>Beirut</OPTION>
   <OPTION value=60>Belgrado</OPTION>
   <OPTION value=60>Berlín</OPTION>
   <OPTION value=-240>Bermuda</OPTION>
   <OPTION value=60>Berna</OPTION>
   <OPTION value=120>Biel</OPTION>
   <OPTION value=-300>Bogotá</OPTION>
   <OPTION value=-300>Boston</OPTION>
   <OPTION value=-180>Brasilia</OPTION>
   <OPTION value=600>Brisbane</OPTION>
   <OPTION value=60>Bruselas</OPTION>
   <OPTION value=120>Bucarest</OPTION>
   <OPTION value=60>Budapest</OPTION>
   <OPTION value=-180>Buenos Aires</OPTION>
   <OPTION value=330>Calcuta</OPTION>
   <OPTION value=-240>Caracas</OPTION>
   <OPTION value=0>Casablanca</OPTION>
   <OPTION value=-300>Chicago</OPTION>
   <OPTION value=120>Ciudad del Cabo</OPTION>
   <OPTION value=60>Copenhague</OPTION>
   <OPTION value=-360>Dallas</OPTION>
   <OPTION value=180>Dar es Salaam</OPTION>
   <OPTION value=570>Darwin</OPTION>
   <OPTION value=-420>Denver</OPTION>
   <OPTION value=-300>Detroit</OPTION>
   <OPTION value=360>Dhaka</OPTION>
   <OPTION value=0>Dublín</OPTION>
   <OPTION value=-420>Edmonton</OPTION>
   <OPTION value=120>Estambul</OPTION>
   <OPTION value=60>Estocolmo</OPTION>
   <OPTION value=180>El Cairo</OPTION>
   <OPTION value=60>Francfort</OPTION>
   <OPTION value=60>Ginebra</OPTION>
   <OPTION value=-360>Guatemala</OPTION>
   <OPTION value=420>Hanoi</OPTION>
   <OPTION value=120>Harare</OPTION>
   <OPTION value=120>Helsinki</OPTION>
```

```
<OPTION value=420>Ho Chi Minh City</OPTION>
<OPTION value=480>Hong Kong</OPTION>
<OPTION value=600>Honolulú</OPTION>
<OPTION value=-360>Houston</OPTION>
<OPTION value=-300>Indianápolis</OPTION>
<OPTION value=300>Islamabad</OPTION>
<OPTION value=420>Jakarta</OPTION>
<OPTION value=120>Jerusalén</OPTION>
<OPTION value=120>Johanesburgo</OPTION>
<OPTION value=270>Kabúl</OPTION>
<OPTION value=720>Kamchatka</OPTION>
<OPTION value=300>Kathmandú</OPTION>
<OPTION value=120>Khartoum</OPTION>
<OPTION value=120>Kigali</OPTION>
<OPTION value=-300>Kingston</OPTION>
<OPTION value=480>Kuala Lumpur</OPTION>
<OPTION value=180>Kuwait</OPTION>
<OPTION value=-240>La Habana</OPTION>
<OPTION value=-240>La Paz</OPTION>
<OPTION value=0>Las Palmas</OPTION>
<OPTION value=60>Lagos</OPTION>
<OPTION value=-300>Lima</OPTION>
<OPTION value=0>Lisboa</OPTION>
<OPTION value=0>Londres</OPTION>
<OPTION value=-480>Los Ángeles</OPTION>
<OPTION value=60>Madrid</OPTION>
<OPTION value=-360>Managua</OPTION>
<OPTION value=480>Manila</OPTION>
<OPTION value=240>Masqat</OPTION>
<OPTION value=600>Melburne</OPTION>
<OPTION value=-360>Méjico D.F.</OPTION>
<OPTION value=300>Miami</OPTION>
<OPTION value=60>Milán</OPTION>
<OPTION value=-360>Minneápolis</OPTION>
<OPTION value=-180>Montevideo</OPTION>
<OPTION value=-300>Montreal</OPTION>
<OPTION value=180>Moscú</OPTION>
<OPTION value=330>Mumbai</OPTION>
<OPTION value=60>Múnich</OPTION>
<OPTION value=180>Nairobi</OPTION>
<OPTION value=60>Nápoles</OPTION>
<OPTION value=-300>Nassau</OPTION>
<OPTION value=330>Nueva Delhi</OPTION>
<OPTION value=-360>Nueva Orleans</OPTION>
<OPTION value=-300>Nueva York</OPTION>
<OPTION value=60>Oslo</OPTION>
<OPTION value=-240>Ottawa</OPTION>
<OPTION value=60>París</OPTION>
<OPTION value=480>Pequín</OPTION>
<OPTION value=480>Perth</OPTION>
<OPTION value=-420>Phoenix</OPTION>
<OPTION value=420>Phnom Penh</OPTION>
<OPTION value=60>Praga</OPTION>
<OPTION value=540>Pyongyang</OPTION>
 <OPTION value=0>Reikiavik</OPTION>
<OPTION value=-180>Río de Janeiro</OPTION>
<OPTION value=180>Riyadh</OPTION>
 <OPTION value=60>Roma</OPTION>
<OPTION value=-240>St. John's</OPTION>
<OPTION value=-360>St. Paul</OPTION>
<OPTION value=-480>San Francisco</OPTION>
<OPTION value=-240>San Juan</OPTION>
<OPTION value=-360>San Salvador</OPTION>
<OPTION value=-300>Santo Domingo</OPTION>
<OPTION value=-240>Santiago</OPTION>
<OPTION value=-180>Sao Paulo</OPTION>
<OPTION value=-480>Seattle</OPTION>
<OPTION value=540>Seúl</OPTION>
 <OPTION value=480>Shanghai</OPTION>
<OPTION value=480>Singapur</OPTION>
<OPTION value=60>Sofía</OPTION>
<OPTION value=720>Suva</OPTION>
```

```
   <OPTION value=600>Sydney</OPTION>
   <OPTION value=480>Taipei</OPTION>
    <OPTION value=120>Tallinn</OPTION>
   <OPTION value=300>Tashkent</OPTION>
   <OPTION value=-210>Teherán</OPTION>
   <OPTION value=540>Tokio</OPTION>
   <OPTION value=-300>Toronto</OPTION>
   <OPTION value=60>Turín</OPTION>
   <OPTION value=-300>Vancúver</OPTION>
   <OPTION value=60>Varsovia</OPTION>
   <OPTION value=60>Venecia</OPTION>
   <OPTION value=60>Viena</OPTION>
   <OPTION value=600>Vladivostok</OPTION>
   <OPTION value=-300 selected>Washington</OPTION>
   <OPTION value=720>Wellington</OPTION>
   <OPTION value=-360>Winnipeg</OPTION>
   <OPTION value=390>Yangon</OPTION>
   <OPTION value=60>Zagreb</OPTION>
   <OPTION value=60>Zurich</OPTION>
</SELECT>
son

<input type="submit" value="Ver hora"></form>
```

Explicación:

Primero que nada iniciaremos una condición para comprobar si $_POST[hora] se ha definido en algún momento a través de la función isset(); (solo tomara valor cuando se envíen datos a través de HTTP-POST)

Cambiaremos el tipo de dato de $_POST[hora], $_POST[min] y $_POST[segundo] de String o Cadena a Integer o Entero a través de la función settype():

Realizaremos una validación con una serie de condiciones para comprobar:

Sí El Valor es menor que 0 (cero) o mayor que 12 ó 60 según sea el caso, de ser contrarias todas estas situaciones iniciaremos el ensamblaje de una serie de instrucciones.

Condición: if($_POST[tm]=='pm' && $_POST[hora]>=1 && $_POST[hora]<12)

Expresamos si la hora es PM pero además esa hora debe ser mayor o igual que 1 y menor a 12, pues si evalúa TRUE nos hará llevar la hora a en base 24 :

Por ejemplo: si se inserta 1 HORA PM debemos llevarlo a 13 horas, para ello sumamos $_POST[hora] más 12 horas transcurridas como constante.

Luego asociamos a la variable $total_minutos_el valor devuelto por la multiplicación entre $_POST[hora] y 60 más los minutos “$_POST[min]” redondeado hacia abajo mediante la función matemática floor(), para llevar la expresión de Hora:Minutos a minutos totales.

Definimos el valor de la variable $dia, este valor se mantendrá siempre y cuando las circunstancias así lo toleran, pues puede ser cambiado o remplazado si alguna de las condiciones subsiguientes evalúa TRUE.

Sí $_POST[ciudad1] es mayor o igual a 0,

Recordemos que el valor de $_POST[ciudad1] será la hora con respecto al meridiano de la ciudad tomada como referencia, Ese entero puede ser negativo o positivo y esta condición evaluara TRUE solo si es positivo; para realizar una resta entre $total_minutos_ y el valor de $_POST[ciudad1] asociándolo a la variable $gmt.

De lo contrario a sí $_POST[ciudad1] es mayor o igual a 0,

Quiere decir que $_POST[ciudad1] es negativo y lo pasaremos a positivo, multiplicándolo por -1 .
$_POST[ciudad1]*-1;
Y realizamos una suma entre $_POST[ciudad1] y $total_minutos_, lo contrario del caso anterior.

¿ Para que y con que sentido?

Debemos determinar si el valor resultante es positivo o negativo para constatar si la diferencia es del día anterior o del día siguiente.

Si es menor a 0 (negativo)
Cambiamos el valor anteriormente asignado a $dia por día anterior y le agregamos a $gmt “1440” minutos totales de un día.

Sí $_POST[ciudad2] es mayor que 0 , esta hacia el ESTE del meridiano
Sumamos $tiempo2 y $total_minutos_, y lo asociamos a $tiempo2.

De lo contrario a Sí $_POST[ciudad2] es mayor que 0.

Realizamos un operación similar a la anterior pero esta vez le sumamos el valor de $gmt, definido en procesos anteriores.

Dentro de ese marco realizamos dos condiciones mas:

Si $tiempo2 es mayor que el total de minutos de un día (1440), entonces es el día siguiente.
Si $tiempo2 es menor que 0 entonces es del día anterior.

Finalmente sin importar la rutina empleada retornamos el resultado, transformando esos minutos resultantes en horas : para ello lo dividiremos entre 60 y redondeamos ese valor hacia abajo, asociamos el resultado a la variable: $hora_de_la_otra_ciudad

Y damos salida al navegador por medio de un echo.

Emmanuel García de Caro
http://www.blasten.com/contenidos/19100

Recomendar un sitio usando PHP

Script que conmina a un usuario a visitar el sitio web, via e-mail.

Este sistema permite al usuario enviar desde la web, un e-mail a otra persona, invitándolo a visitar el sitio.

Solo requiere un módulo PHP que se encargará de mostrar el formulario de recomendación, enviar el e-mail y devolver un acuse de envío.

recomendar.php

```
<!-- formulario de recomendación -->

<? if ($HTTP_GET_VARS["accion"] == "") {
?>

<form method="post" action="recomendar.php?accion=enviar" name="recomienda">
<b>Recomienda este sitio</b><br><br>
Tu Nombre: <input type="text" name="n_remitente" size="10"><br>
Tu E-mail: <input type="text" name="e_remitente" size="20"><br>
Nombre de tu amigo: <input type="text" name="n_destinatario" size="10"><br>
E-mail de tu amigo: <input type="text" name="e_destinatario" size="20"><br><br>
<input type="submit" value="Recomendar">
</form>

<!-- envío del formulario y acuse de envío o información de errores -->
<?
}

elseif ($HTTP_GET_VARS["accion"] == "enviar") {

// recojo las variables que vienen desde el formulario
$n_destinatario = $HTTP_POST_VARS["n_destinatario"];
$e_destinatario = $HTTP_POST_VARS["e_destinatario"];
$n_remitente = $HTTP_POST_VARS["n_remitente"];
$e_remitente = $HTTP_POST_VARS["e_remitente"];

// si los campos no están vacíos
  if ($n_destinatario != "" && $e_destinatario != "" && $n_remitente != "" && $e_remitente != "") {

//indica la url de tu sitio
    $url = "http://www.tusitio.com";
//indica el nombre de tu sitio
    $nombre_del_sitio = "Tu Sitio";
//indica el asunto del mensaje
    $asunto = $n_remitente . " te recomienda un sitio";
//redacta el mensaje
    $mensaje = "Hola " . $n_destinatario . ":<br>";
    $mensaje .= $n_remitente . " te recomienda que visites <b>" . $nombre_del_sitio . "</b>.<br>";
    $mensaje .= "Puedes verlo en <a href='" . $url . "'>" . $url . "</a><br><br>Saludos!";

//indica que el mail se envía en formato HTML

    $encabezado = "From:".$e_remitente."\nReply-To:".$e_remitente."\n";
    $encabezado .= "X-Mailer:PHP/".phpversion()."\n";
    $encabezado .= "Mime-Version: 1.0\n";
    $encabezado .= "Content-Type: text/html";

//envía el mensaje

    mail($e_destinatario,$asunto,$mensaje,$encabezado);

//Informa al usuario que se ha enviado el mensaje

    echo "<b>El mensaje ha sido enviado</b>.<br>Gracias por recomendarnos!";

    }

//si existen campos vacíos, envía un mensaje de error

    else {
```

```
    echo "Por favor, es necesario que completes todos los campos.<br>";
    echo "<a href='recomendar.php'>Pincha aquí</a> para corregir los campos.";

    }

}
?>
```

Eugenia Bahit
http://www.cmzk.com.ar/

Bucle para recibir todas las variables por POST en PHP

Una manera muy rápida de recibir todas las variables de un formulario, enviado por post. Mediante un recorrido genérico del array $_POST, en el lenguaje PHP.

Vamos a ver una manera muy rápida de recibir todas las variables de un formulario, enviado por post, en el lenguaje PHP. Os aseguro que es una pequeña porción de código que os ahorrará escribir un montón de líneas de código.

Quién no se ha visto alguna vez en la tediosa tarea de recibir un montón de datos de un formulario, asignando una por una todas las variables en PHP? Eso se hacía con líneas como ésta:

```
$nombre = $_POST["nombre"];
$edad = $_POST["edad"];
$ciudad = $_POST["ciudad"];
....
```

Si el formulario tuviera 10 elementos no sería muy pesado escribir las 10 líneas de código, pero si fueran 50 o 100 la cosa sería mucho menos agradable. El código que vamos a ver ahora nos solucionará la vida en esos casos.

```
foreach($_POST as $nombre_campo => $valor){
   $asignacion = "\$" . $nombre_campo . "='" . $valor . "';";
   eval($asignacion);
}
```

Se realiza un bucle foreach que va recorriendo cada uno de los elementos del post. En cada iteración, se van accediendo a todos los elementos del post y se guarda en $ nombre_campo el nombre del campo recibido por el formulario y en $valor, el valor que se había introducido en el formulario.

Todo lo anterior se deduce de la primera línea. En las siguientes se compone en cada iteración, cada una de las asignaciones que deberíamos haber escrito manualmente. Es decir, en la variable asignación guardaremos una línea de código PHP que realiza la declaración de la variable de formulario dentro de PHP y su inicialización con el valor que se hubiera escrito.

En la siguiente línea, donde está la función eval(), se ejecuta la sentencia generada en el anterior paso. La función eval () de PHP ejecuta el contenido de una cadena de caracteres como si fuera una sentencia PHP. (Podemos ver la documentación de la función eval() en la página de PHP **http://es.php.net/manual/es/function.eval.php**)

Esperamos que os haya interesado este minúsculo, pero útil, código PHP.

Miguel Angel Alvarez
http://www.desarrolloweb.com/articulos/1326.php?manual=6

Contador simple para páginas PHP

Creamos un contador, programado en PHP, que lleva la cuenta de las impresiones que se han realizado en una página web, utilizando un archivo de texto como apoyo.

Hice una modificación al Script publicado en el artículo Escritura de archivos con PHP, en el que se enseña a escribir archivos de texto mediante PHP, tocando los temas de lectura y escritura. El objetivo es llevar un conteo de las veces que se ha visitado una página.

Puse el siguiente script PHP al final de la página, se entenderá bien si se lee el articulo señalado antes.

```
<?
$archivo = "contador.txt";
$contador = 0;

$fp = fopen($archivo,"r");
$contador = fgets($fp, 26);
fclose($fp);

++$contador;

$fp = fopen($archivo,"w+");
fwrite($fp, $contador, 26);
fclose($fp);

echo "Esta página ha sido visitada $contador veces";

?>
```

Además, creé un archivo llamado "contador.txt" que lo guardé en el mismo directorio que la página. Dicho archivo fue inicializado con un cero (0) como único texto.

> Nota: si tenéis problemas a la hora de escribir en un archivo, casi con toda probabilidad, estará protegido contra escritura. O bien el archivo o bien el directorio.
>
> Si tenéis vuestro propio servidor tendréis que modificar los permisos de tal archivo o directiorio por vosotros mismos. Sin embargo, si estáis publicando en un alojamiento contratado en un proveedor tendréis que enteraros de qué mecanismo hay que poner en marcha en ese proveedor para conseguir los permisos. En muchos casos existirá un panel de control donde modificar esas cosas, en otros casos tendréis que escribir a soporte técnico para que lo hagan a mano éllos o os digan cómo hacerlo, si es que es posible.

Con esto ya está hecho un contador muy simple, pero muy funcional.

Creación de gráficas en PHP con JpGraph

Presentación de la librería JpGraph, que sirve para generar imágenes con todo tipo de gráficas de datos en PHP.

Una tarea, a la que más tarde o más temprano vamos a tener que enfrentarnos a lo largo de nuestra andadura como profesionales del web, es la creación de gráficas a partir de datos. Cuando hablo de gráficas me refiero a todo tipo de imágenes que sirvan para representar datos, como gráficas de barras, de líneas de progreso, de tarta, etc.

Obviamente, la creación de gráficas no es un tema trivial, sino que requerirá de una gran dosis de dedicación y esfuerzo. Las gráficas, que generalmente se muestran en imágenes, pueden ser de muchos tipos distintos y sólo el hecho de tratar de dibujar en una imagen líneas, barras o incluso tartas en tres dimensiones, puede ser sobradamente complicado.

Sin embargo, existen sistemas como JpGraph, que nos pueden facilitar la tarea de una manera muy interesante, pues ofrecen una serie de mecanismos para la generación de las imágenes con las gráficas, de modo que nosotros sólo tenemos que centrarnos en cargar los datos a representar y escoger el tipo de gráfica que deseamos visualizar.

Qué es JpGraph

Es una librería que incluye una serie de clases -código orientado a objetos- que sirven para crear imágenes con todo tipo de gráficas, dinámicamente desde páginas PHP.

El sistema está muy depurado y soporta multitud de funcionalidades, por lo que seguramente encontraremos solución a casi cualquier necesidad en el ámbito de creación de gráficas. Además, la mayoría de las configuraciones de las gráficas vienen con opciones por defecto, así que resulta bastante sencillo obtener resultados rápidamente.

Algunas de las características del sistema son:

Reducido peso en bytes de las imágenes resultado. Habitualmente unas pocas KB.
Soporte a las librerías GD1 o GD2.
Uso de la Interpolación matemática para obtener curvas a partir unos pocos valores.
Diversos tipos de gráficas 2D o 3D, como de puntos, líneas, tartas, barras, cajas...
Escalas flexibles tanto en el eje X como el Y, que se ajustan al juego de datos que se tenga que representar.
Soporte para generar gráficas con varios juegos de valores a la vez.
Configurable con distintos tipos de colores, leyendas, tipografías, imágenes de fondo, etc.

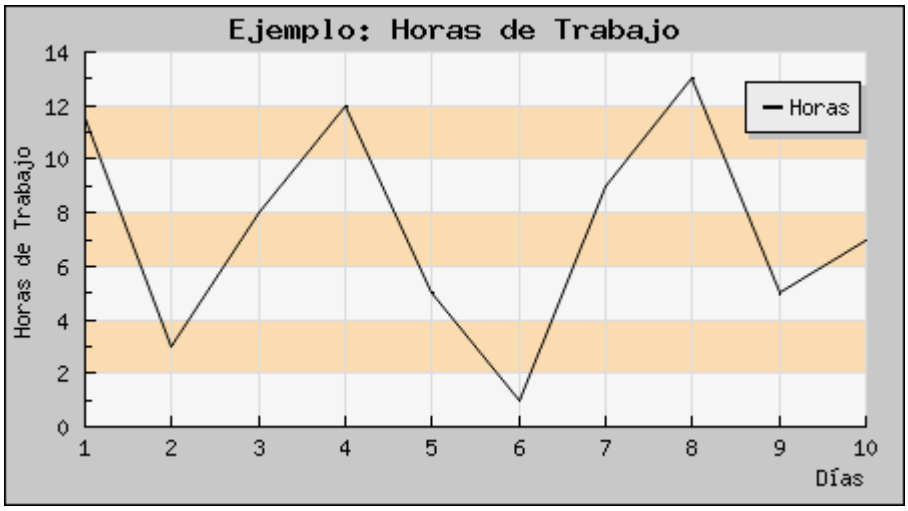
Cómo usar JpGraph

Este juego de librerías dispone de una extensa documentación y tutoriales para aprender a manejarlo. En la documentación se encuentran además numerosos ejemplos de su uso, desde los que podemos partir para solucionar nuestras necesidades.

El modo de trabajo para usar esta librería es muy simple, se trata de crear una imagen con la etiqueta <*img*> de HTML, en cuyo atributo src colocaremos la ruta hacia el script PHP que se encargará de generar la gráfica.

En el archivo PHP que generará la gráfica tendremos que incluir las librerías apropiadas para el tipo de gráfica que deseemos realizar, también habrá que instanciar el objeto JpGraph correspondiente, cargar los datos a visualizar y llamar a los métodos adecuados para mostrar la imagen. Un mecanismo bastante sencillo que veremos en un par de ejemplos a continuación.

Ejemplo 1: una gráfica de línea.



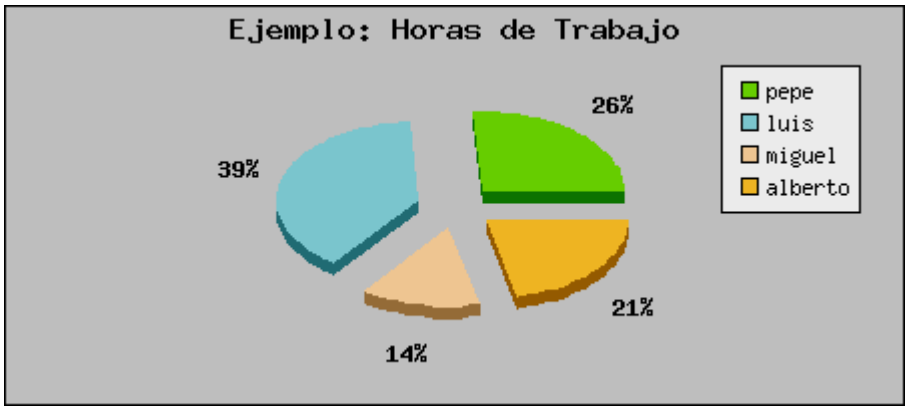
En este ejemplo vamos a crear una gráfica lineal en la que mostraremos las horas de trabajo de una persona a lo largo de 10 días.

La generación de la gráfica de este ejemplo la hacemos en un archivo que hemos llamado grafico_linea.php, por lo tanto, la llamada a este archivo dentro de una imagen será la siguiente:

<img src="grafico_linea.php" alt="" border="0"> El código PHP del archivo grafico_linea.php es el siguiente:

```
<?php
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_line.php");

// Some data
$ydata = array(11.5,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(450,250,"auto");
$graph->SetScale("textlin");
$graph->img->SetAntiAliasing();
$graph->xgrid->Show();

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->SetColor("black");
$lineplot->SetWeight(2);
$lineplot->SetLegend("Horas");

// Setup margin and titles
$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Ejemplo: Horas de Trabajo");
$graph->xaxis->title->Set("Días");
$graph->yaxis->title->Set("Horas de Trabajo");
$graph->ygrid->SetFill(true,'#EFEFEF@0.5','#F9BB64@0.5');
//$graph->SetShadow();

// Add the plot to the graph
$graph->Add($lineplot);

// Display the graph
$graph->Stroke();
?>
```

Ejemplo 2: una gráfica de tarta en 3D



Por otra parte, vamos a realizar un ejemplo de una gráfica de tarta, en la que aparecen las horas realizadas por cada uno de los empleados y el porcentaje respecto a las totales. En este caso, la tarta va a presentarse en un dibujo en 3 dimensiones.

El archivo donde se genera la gráfica se llama grafico_tarta.php. Lo llamaríamos dentro de una imagen con este código HTML.

```
<img src="grafico_tarta.php" alt="" border="0">
```

El código PHP del archivo grafico_tarta.php será el siguiente:

```
<?php
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_pie.php");
include ("jpgraph/jpgraph_pie3d.php");

$data = array(40,60,21,33);

$graph = new PieGraph(450,200,"auto");
$graph->img->SetAntiAliasing();
$graph->SetMarginColor('gray');
//$graph->SetShadow();

// Setup margin and titles
$graph->title->Set("Ejemplo: Horas de Trabajo");

$p1 = new PiePlot3D($data);
$p1->SetSize(0.35);
$p1->SetCenter(0.5);

// Setup slice labels and move them into the plot
$p1->value->SetFont(FF_FONT1,FS_BOLD);
$p1->value->SetColor("black");
$p1->SetLabelPos(0.2);

$nombres=array("pepe","luis","miguel","alberto");
$p1->SetLegends($nombres);

// Explode all slices
$p1->ExplodeAll();

$graph->Add($p1);
$graph->Stroke();
?>
```

Conclusión

JpGraph es una herramienta muy potente para la generación de gráficos en nuestra página web y gracias a su uso nos damos cuenta de sus muchas bondades:

Es una librería gratuita (para uso no comercial), fácil de instalar y de fácil manejo.
Incluye una completa documentación con multitud de ejemplos de los distintos gráficos que se pueden generar.
Además de generar muchos tipos de gráficos, permite 'customizar' casi todo lo que se ve, resultando esto muy útil para integrar perfectamente el gráfico en el diseño de nuestra web.

La forma de integrar el gráfico es muy sencilla: únicamente es necesario incluir una imagen (etiqueta <IMG>) cuyo src sea el script PHP que generará nuestro gráfico (ver ejemplos).

Pocos defectos hemos encontrado en la librería, tan solo se podría mejorar lo siguiente:

Los mensajes de error son algo escasos, y la mayotía de las veces cuando algo falla no obtenemos ninguna explicación. Se echa de menos algo de definición en los gráficos, sobre todo en los gráficos de sectores.

Autor: Pablo González
http://www.xski.net/

Frases Aleatorias con PHP

Gracias a un vector y un pequeño generador de números aleatorios, podrás incorporar a tu sitio un script para mostrar frases en forma aleatoria.

El funcionamiento del Script es realmente sencillo, se trata simplemente de un vector (o array) y la utilización de la función rand de PHP:

- Completar las frases dentro del vector:

```
$vector = array(
1 => "Nada nuevo hay bajo el sol, pero cuántas cosas viejas hay que no conocemos.",
2 => "El verdadero amigo es aquel que está a tu lado cuando preferiría estar en otra parte.",
3 => "La sabiduría es la hija de la experiencia.",
4 => "Nunca hay viento favorable para el que no sabe hacia dónde va.",
);
```

- Obtener un número aleatorio entre 1 y la cantidad de frases incluidas.

```
$numero = rand(1,4);
```

- Imprimir el contenido del vector

```
echo "$vector[$numero]";
```

Script completo

```
<?
// Completamos el vector con frases
$vector = array(
1 => "Nada nuevo hay bajo el sol, pero cuántas cosas viejas hay que no conocemos.",
2 => "El verdadero amigo es aquel que está a tu lado cuando preferiría estar en otra parte.",
3 => "La sabiduría es la hija de la experiencia.",
4 => "Nunca hay viento favorable para el que no sabe hacia dónde va.",
);

// Obtenemos un número aleatorio
$numero = rand(1,4);

// Imprimimos la frase
echo "$vector[$numero]";
?>
```

Artículo por cortesía de Fabian Muller
WebExperto.com

Expresiones Regulares en PHP

Serie de carácteres que forman un patrón para poder compararlo con otro grupo de carácteres.

Las expresiones regulares son una serie de carácteres que forman un patrón, normalmente representativo de otro grupo de carácteres mayor, de tal forma que podemos comparar el patrón con otro conjunto de carácteres para ver las coincidencias.

Las expresiones regulares estan disponibles en casi cualquier lenguaje de programación, pero aunque su sintaxis es relativamente uniforme, cada lenguaje usa su propio dialecto.

Si es la primera vez que te acercas al concepto de expresiones regulares (regex para abreviar) te animará saber que seguro que ya las has usado, aún sin saberlo, al menos en su vertiente mas básica. Por ejemplo, cuando en una ventana DOS ejecutamos dir *.* para obtener un listado de todos los archivos de un directorio, estamos utilizando el concepto de expresiones regulares, donde el patrón * coincide con cualquier cadena de caracteres.

Unos ejemplos simplificados:

<? am // este es nuestro patrón. Si lo comparamos con:
am // coincide
panorama // coincide
ambicion // coincide
campamento // coincide
mano // no coincide
?>

Se trata sencillamente de ir cotejando un patrón (pattern) -que en este ejemplo es la secuencia de letras 'am'- con una cadena (subject) y ver si dentro de ella existe la misma secuencia. Si existe, decimos que hemos encontrado una coincidencia (match, en inglés).

Otro ejemplo:

patrón: el
el ala aleve del leve abanico
Hasta ahora los ejemplos han sido sencillos, ya que los patrones usados eran literales, es decir que solo encontramos coincidencias cuando hay una ocurrencia exacta.

Si sabemos de antemano la cadena exacta a buscar, no es necesario quebrarse con un patrón complicado, podemos usar como patrón la exacta cadena que buscamos, y esa y no otra será la que de coincidencia. Asi, si en una lista de nombres buscamos los datos del usuario pepe podemos usar pepe como patrón. Pero si ademas de pepe nos interesa encontrar ocurrencias de pepa y pepito los literales no son suficientes.

El poder de las expresiones regulares radica precisamente en la flexibilidad de los patrones, que pueden ser confrontados con cualquier palabra o cadena de texto que tenga una estructura conocida.

De hecho normalmente no es necesario usar funciones de expresiones regulares si vamos a usar patrones literales. Existen otras funciones (las funciones de cadena) que trabajan mas eficaz y rapidamente con literales.

Caracteres y meta caracteres

Nuestro patrón puede estar formado por un conjunto de carácteres (un grupo de letras, numeros o signos) o por meta caracteres que representan otros carácteres, o permiten una búsqueda contextual.

Los meta-caracteres reciben este nombre porque no se representan a ellos mismos, sino que son interpretados de una manera especial.

He aqui la lista de meta caracteres mas usados:

. * ? + [ ] ( ) { } ^ $ |

Iremos viendo su utilización, agrupandolos segun su finalidad.

Meta caracteres de posicionamiento, o anclas

Los signos ^ y $ sirven para indicar donde debe estar situado nuestro patrón dentro de la cadena para considerar que existe una coincidencia.

Cuando usamos el signo ^ queremos decir que el patrón debe aparecer al principio de la cadena de carácteres comparada. Cuando usamos el signo $ estamos indicando que el patrón debe aparecer al final del conjunto de carácteres. O mas exactamente, antes de un caracter de nueva linea Asi:

```
<?

^am // nuestro patrón
am // coincide
cama // no coincide
ambidiestro // coincide
Pam // no coincide
caramba // no coincide

am$
am // coincide
salam // coincide
ambar // no coincide
Pam // coincide

^am$
am // coincide
salam // no coincide
ambar // no coincide

?>
```

o como en el ejemplo anterior:

patrón: ^el
el ala aleve del leve abanico
Las expresiones regulares que usan anclas solo devolveran una ocurrencia, ya que por ejemplo, solo puede existir una secuencia el al comienzo de la cadena.

patrón: el$
el ala aleve del leve abanico
Y aqui no encontramos ninguna, ya que en la cadena a comparar (la linea en este caso) el patrón "el" no está situado al final.
Para mostrar una coincidencia en este ejemplo, tendriamos que buscar "co":

patrón: co$
con el ala aleve del leve abanico
Hemos comenzado por unos metacaracteres especiales, ya que ^ $ no representan otros carácteres, sino posiciones en una cadena. Por eso, se conocen tambien como anchors o anclas.

Escapando caracteres

Puede suceder que necesitemos incluir en nuestro patrón algun metacaracter como signo literal, es decir, por si mismo y no por lo que representa. Para indicar esta finalidad usaremos un carácter de escape, la barra invertida .
Así, un patrón definido como 12$ no coincide con una cadena terminada en 12, y sí con 12$:

patrón: 100$
el ala aleve del leve abanico cuesta 100$
patrón: 100$
el ala aleve del leve abanico cuesta 100$
Fijate en los ejemplos anteriores. En el primero, no hay coincidencia, porque se interpreta "busca una secuencia consistente en el número 100 al final de la cadena", y la cadena no termina en 100, sino en 100$.
Para especificar que buscamos la cadena 100$, debemos escapar el signo $

Como regla general, la barra invertida convierte en normales caracteres especiales, y hace especiales caracteres normales.

El punto . como metacaracter

Si un metacaracter es un caracter que puede representar a otros, entonces el punto es el metacaracter por excelencia. Un punto en el patrón representa cualquier caracter excepto nueva línea.

Y como acabamos de ver, si lo que queremos buscar en la cadena es precisamente un punto, deberemos escaparlo: .
patrón: '.l'
el ala aleve del leve abanico
Observa en el ejemplo anterior como el patrón es cualquier caracter (incluido el de espacio en blanco) seguido de una l.

Metacaracteres cuantificadores

Los metacaracteres que hemos visto ahora nos informan si nuestro patron coincide con la cadena a comparar. Pero ¿y si queremos comparar con nuestra cadena un patrón que puede estar una o mas veces, o puede no estar ? Para esto usamos un tipo especial de meta carácteres: los multiplicadores.

Estos metacaracteres que se aplican al caracter o grupo de caracteres que les preceden indican en que número deben encontrarse presentes en la cadena para que haya una ocurrencia.

Por ello se llaman cuantificadores o multiplicadores. Los mas usados son * ? +

```
<?
* // coincide si el caracter (o grupo de caracteres) que le
// precede esta presente 0 o mas veces
// ab* coincide con "a", "ab", "abbb", etc.
//ejemplo:
cant*a // coincide con canta, cana, canttta

? // coincide si el carácter (o grupo de caracteres) que precede
// esta presente 0 o 1 vez
// ab? coincide con "a", "ab", no coincide con "abb"
// ejemplo:
cant?a // coincide con canta y cana
d?el // coincide con del y el
(ala)?cena // coincide con cena y alacena

+ // coincide si el carácter (o grupo) que le precede está
// presente al menos 1 o mas veces.
// ab+ coincide con "ab", "abbb", etc. No coincide con "a"
//ejemplo:
cant+a // coincide con canta, cantttta, NO coincide con
// cana ?>
```

patrón: 'a*le'
el ala aleve del leve abanico

patrón: ' ?le'
el ala aleve del leve abanico

patrón: ' +le'
el ala aleve del leve abanico
Ademas de estos cuantificadores sencillos tambien podemos especificar el numero de veces máximo y mínimo que debe darse para que haya una ocurrencia:

patrón: (.a){2}
el ala aleve del leve abanico

```
<?
{x,y} // coincide si la letra (o grupo) que le precede esta presente
// un minimo "x" veces y como máximo "y" veces
// "ab{2}" coincide con "abb": exactamente dos ocurrencias de "b"
// "ab{2,}" coincide con "abb", "abbbb" ... Como mínimo dos
// ocurrencias de b, máximo indefinido
// "ab{3,5}" coincide con "abbb", "abbbb", o "abbbbb": Como minimo
// dos ocurrencias, como máximo 5

a{2,3} // coincide con casaa, casaaa

a{2, } // coincide con cualquier palabra que tenga al
```

```
// menos dos "a" o mas: casaa o casaaaaaa, no con casa

a{0,3} // coincide con cualquier palabra que tenga 3 o
// menos letras "a".
// NOTA: puedes dejar sin especificar el valor máximo. NO
// puedes dejar el valor inicial vacío

a{5} // exactamente 5 letras "a"
?>
```

Por tanto, los cuantificadores * + ? pueden también ser expresados así:

* equivale a {0,} (0 o mas veces)
+ equivale a {1,} (1 o mas veces)
? equivale a {0,1} (0 o 1 vez)

Metacaracteres de rango

Los corchetes [] incluidos en un patrón permiten especificar el rango de caracteres válidos a comparar. Basta que exista cualquiera de ellos para que se de la condición:

```
<?

[abc] // El patrón coincide con la cadena si en esta hay
// cualquiera de estos tres carácteres: a, b, c

[a-c] // coincide si existe una letra en el rango ("a", "b" o "c")
c[ao]sa // coincide con casa y con cosa

[^abc] // El patrón coincide con la cadena si en esta NO hay
// ninguno de estos tres carácteres: a, b, c
// Nota que el signo ^ aqui tiene un valor excluyente

c[^ao]sa // Coincide con cesa, cusa, cisa (etc); no coincide
// con casa ni cosa

[0-9] // Coincide con una cadena que contenga cualquier
// número entre el 0 y el 9

[^0-9] // Coincide con una cadena que NO contenga ningun
// número

[A-Z] // Coincide con cualquier carácter alfabetico,
// en mayúsculas. No incluye numeros.

[a-z] // Como el anterior, en minúsculas

[a-Z] // Cualquier carácter alfabético, mayusculas o minusculas

?>
```

Una cuestión a recordar es que las reglas de sintaxis de las expresiones regulares no se aplican igual dentro de los corchetes. Por ejemplo, el metacarácter ^ no sirve aqui de ancla, sino de caracter negador. Tampoco es necesario escapar todos los metacaracteres con la barra invertida. Solo será necesario escapar los siguientes metacaracteres:] ^ -

El resto de metacaracteres pueden incluirse ya que son considerados -dentro de los corchetes- caracteres normales.

patrón: [aeiou]
el ala aleve del leve abanico

patrón: [^aeiou]
el ala aleve del leve abanico

patrón: [a-d]
el ala aleve del leve abanico

Como estos patrones se usan una y otra vez, hay atajos:

```
<?
// atajo equivale a significado

d [0-9] // numeros de 0 a 9
D [^0-9] // el contrario de d

w [0-9A-Za-z] // cualquier numero o letra
W [^0-9A-Za-z] // contrario de w, un carácter que no

// sea letra ni numero

s [ \t\n\r] // espacio en blanco: incluye espacio,

// tabulador, nueva linea o retorno



S [^ \t\n\r] // contrario de \s, cualquier carácter
// que no sea espacio en blanco

// solo regex POSIX
[[:alpha:]] // cualquier carácter alfabético aA - zZ.
[[:digit:]] // Cualquier número (entero) 0 - 9
[[:alnum:]] // Cualquier carácter alfanumérico aA zZ 0 9
[[:space:]] // espacio
?>
```

Metacaracteres de alternancia y agrupadores

```
<?
(xyz) // coincide con la secuencia exacta xyz
x|y // coincide si esta presente x ó y

(Don|Doña) // coincide si precede "Don" o "Doña"
?>
```

patrón: (al)
el ala aleve del leve abanico

patrón: a(l|b)
el ala aleve del leve abanico
Los paréntesis sirven no solamente para agrupar secuencias de caracteres, sino también para capturar subpatrones que luego pueden ser devueltos al script (backreferences). Hablaremos mas de ello al tratar de las funciones POSIX y PCRE, en las paginas siguientes.

Un ejemplo típico sería una expresion regular cuyo patrón capturase direcciones url validas y con ellas generase links al vuelo:

```
<? $text = "una de las mejores páginas es http://www.blasten.com";
$text = ereg_replace("http://(.*.(com|net|org))",
"1", $text);
print $text;
?>
```

El anterior ejemplo produciría un enlace usable, donde la url se tomaría de la retro-referencia y la parte visible de la retro-referencia 1 una de las mejores páginas es www.ignside.net

Fijate que en el ejemplo anterior usamos dos grupos de parentesis (anidados), por lo que se producirian dos capturas: La retro-referencia coincide con la coincidencia buscada. Para capturarla no es preciso usar parentesis.

La retro-referencia 1 coincide en este caso con "www.blasten.com" y es capturada por el parentesis (.*.(com|net|org))

La retro-referencia 2 coincide con "net" y se corresponde con el parentesis anidado (com|net|org)

Ten en cuenta que esta caracteristica de capturar ocurrencias y tenerlas disponibles para retroreferencias consume recursos del sistema. Si quieres usar parentesis en tus expresiones regulares, pero sabes de antemano que no vas a reusar las ocurrencias, y puedes prescindir de la captura, coloca despues del primer parentesis ?:

```
<?
text = ereg_replace("http://(.*.(?:com|net|org))",
"<a href="">1</a>", $text);
?>
```

Al escribir (?:com|net|org) el subpatron entre parentesis sigue agrupado, pero la coincidencia ya no es capturada.

Como nota final sobre el tema, PHP puede capturar hasta 99 subpatrones a efectos de retro-referencia, o hasta 200 (en total) si buscamos subpatrones sin capturarlos.

Un ejemplo práctico

Hemos dicho que las expresiones regulares son uno de los instrumentos mas útiles en cualquier lenguaje de programación. ¿Para que podemos usarlas?. Uno de sus usos mas típicos es el de validar entradas de datos que los visitantes de una página puedan mandarnos a través de formularios html.
El ejemplo mas corriente es el de una dirección email. Imaginemos que queremos filtrar las direcciones introducidas por los visitantes, para evitar introducir en la base de datos la típica dirección basura ghghghghghghg. Todos sabemos la estructura de una dirección email, formada por la cadena nombreusuario, el signo @ y la cadena nombredominio. Tambien sabemos que nombredominio esta formado por dos subcadenas, 'nombredomino', un '.' y un sufijo 'com', 'net', 'es' o similar.
Por tanto la solución a nuestro problema es idear una expresión regular que identifique una dirección email valida típica, y confrontarla con la cadena (dirección email) pasada por el visitante
Por ejemplo:

```
<?
^[^@ ]+@[^@ ]+.[^@ .]+$
?>
```

Vamos a diseccionar nuestra expresión regular:

```
<?
^ // queremos decir que el primer carácter que buscamos
// debe estar al principio de la cadena a comparar.

[^@ ] // ese primer signo no debe ser ni el signo @
// ni un espacio

+ // y se repite una o mas veces
@ // luego buscamos el signo @

[^@ ]+ // Seguido de otro signo que no es un @ ni un
// espacio y se repite una o mas veces

. // Seguido de un .

[^@ .] // Seguido de un carácter que no sea ni @,
// ni espacio ni punto

+$ // Que se repite una o mas veces y el último esta
// al final de la cadena
?>
```

Y para comprobarlo en la práctica, usamos una de las funciones de php relacionadas con las expresiones regulares:ereg().
Acudiendo al manual php, podemos averiguar que esta función tiene la siguiente sintaxis:
ereg (string pattern, string string)
Busca en string las coincidencias con la expresión regular pattern. La búsqueda diferencia entre mayúsculas y

minúsculas.
Devuelve un valor verdadero si se encontró alguna coincidencia, o falso in no se encontraron coincidencias u ocurrió algún error. Podríamos usar esta funcion para un validador email con algo asi como:

```
<?

// establecemos una secuencia condicional: si la variable $op no existe y
// es igual a "ds", se muestra un formulario

if ($op != "ds") { ?>
<form>
<input type=hidden name="op" value="ds">
<strong>Tu email:</strong><br />
<input type=text name="email" value="" size="25" />
<input type=submit name="submit" value="Enviar" />
</form>
<?
}

// Si $op existe y es igual a "ds"m, se ejecuta la función ereg buscando
// nuestra cadena dentro del patrón $email que es la direccion enviada por
// el usuario desde el formulario anterior

else if ($op == "ds")
{
if (ereg("^[^@ ]+@[^@ ]+.[^@ .]+$", $email ) )
{
print "<BR>Esta dirección es correcta: $email"; }
else {echo "$email no es una dirección valida";}
}

?>
```

No hace falta advertir que se trata de un ejemplo muy elemental, que dará por válida cualquier dirección email que tenga una mínima apariencia de normalidad (por ejemplo, daria por valida 'midireccionnn@noteimporta.commm')

Para tener a mano ...
Una breve referencia de los meta carácteres y su significado, tomada de un comentario del manual de php.net.

```
<?

^ // Comienzo de la zadena
$ // Final de la cadena

n* // Cero o mas "n" (donde n es el carácter precedente)
n+ // Uno o mas "n"
n? // Un posible "n"

n{2} // Exactamente dos "n"
n{2,} // Al menos dos o mas "n"
n{2,4} // De dos a cuatro "n"

() // Parentesis para agrupar expresiones
(n|a) // o "n" o "a"

. // Cualquier carácter

[1-6] // un número entre 1 y 6
[c-h] // una letra en minuscula entre c y h
[D-M] // una letra en mayúscula entre D y M
[^a-z] // no hay letras en minuscula de a hasta z
[_a-zA-Z] // un guion bajo o cualquier letra del alfabeto

^.{2}[a-z]{1,2}_?[0-9]*([1-6]|[a-f])[^1-9]{2}a+$
```

```
/* Una cadena que comienza por dos carácteres cualquiera
Seguidos por una o dos letras (en minuscula)
Seguidos por un guion _ bajo opcional
Seguidos por cero o mas números
Seguidos por un numero del 1 al 6 o una letra de la -a- a la -f-
Seguidos por dos carácteres que no son números del 1 al 9
Seguidos de uno o mas carácteres al final de la cadena
Tomado de una anotacion al manual de php.net, de mholdgate -
wakefield dot co dot uk */

?>
```

Funciones PHP para expresiones regulares

PHP tiene dos conjuntos distintos de funciones relacionadas con expresiones regulares, llamadas POSIX y PCRE.

Las funciones "PCRE" son "PERL Compatible", es decir, similares a las funciones nativas Perl, aunque con ligeras diferencias. Son bastante mas poderosas que las funciones POSIX, y correlativamente mas complejas.

POSIX
Incluye seis funciones diferentes. Como nota común, pasas primero el patrón (la expresión a buscar) y como segundo argumento la cadena a comparar

ereg confronta la cadena con el patrón de búsqueda y devuelve TRUE o FALSE segun la encuentre.
eregi como la anterior, SIN distinguir entre mayusculas-minusculas

```
<?
ereg ("^am", "america"); // TRUE

$es_com = ereg("(.)(com$)",$url);
// buscamos dos subpatrones en $url. El primero es
// un punto (literal) y por eso va escapado con la barra.
// el segundo subpatron también literal busca la secuencia "com"
// al final de una palabra.

?>
```

ereg_replace: Busca cualquier ocurrencia del patrón en la cadena y la reemplaza por otra.
eregi_replace: como la anterior pero sin distinguir mayusculas minusculas:

patrón, reemplazo, cadena a confrontar

```
<?
$cadena = ereg_replace ("^am", "hispano-am", "america"); // $cadena = hispano-america
?>
```

split() divide una cadena en piezas (que pasan a un array) usando expresiones regulares:
spliti: como el anterior, sin diferenciar Mayusculas-minusculas

Es básicamente igual que explode, pero utilizando expresiones regulares para dividir la cadena, en lugar de expresiones literales

```
<?
$date = "24-09-2003";
list($month, $day, $year) = split ('[/.-]', $date);

?>
```

Almacenando los resultados con ereg

Podemos pasar un patrón con subpatrones agrupados en parentesis. Si en este caso usamos ereg, podemos añadir un tercer parámetro: el nombre de un array que almacenará las ocurrencias; $array[1] contendrá la subcadena que empieza en el primer paréntesis izquierdo; $array[2] la que comienza en el segundo, etc. $array[0] contendrá una copia de la cadena.

```
<?
$date = "24-09-2003"; // pasamos una fecha formato dd-mm-yyyy

if (ereg ("([0-9]{1,2})-([0-9]{1,2})-([0-9]{4})", $date, $mi_array)) {
echo "$mi_array[3].$mi_array[2].$mi_array[1]"; // coincide. Lo mostramos en orden inverso porque somos asi : )
} else {
echo "Invalid date format: $date"; // no coincide
}
?>
```

Almacenando los resultados con ereg_replace: backreferences

De forma muy similar a ereg, las funciones de búsqueda y sustitución ereg_replace y eregi_replace pueden almacenar y reutilizar subocurrencias (subpatrones encontrados en la cadena).

La principal diferencia es la forma de llamar las subocurrencias almacenadas, ya que necesitamos utilizar la barra invertida: \0, \1, \2, y asi hasta un máximo de 9.

La primera referencia \0 hace referencia a la coincidencia del patrón entero; el resto, a las sub-ocurrencias de los sub-patrones, de izquierda a derecha.

Por ejemplo vamos a usar esta capacidad para convertir una cadena que muestra una url en un enlace:

```
<?
$url = "la página blasten.com (http://www.blasten.com)";
$url = ereg_replace ("(http|ftp)://(www.)?(.+).(com|net|org)","<a href="\0 ">\3</a>",$url);
echo $url; ?>
```

Funciones PCRE

Las funciones PCRE son "perl-compatibles". Perl es uno de los lenguajes de programación con mejor motor de expresiones regulares, y además es muy conocido. Esta librería es utilizada (con distintas variantes) no solo por Perl, sino por el propio PHP y tambien en otros entornos, como el server Apache, Phyton o KDE.

Las funciones de expresiones regulares PCRE de PHP son mas flexibles, potentes y rápidas que las POSIX.

Prácticamente no existe ninguna diferencia sintáctica entre un patrón PCRE o POSIX; muchas veces son intercambiables. Naturalmente existe alguna diferencia. La mas evidente, que nuestro patrón deberá estar marcado en su principio y final unos delimitadores, normalmente dos barras:

/patron/

Delimitadores

Se puede usar como delimitador cualquier carácter especial (no alfanumerico) salvo la barra invertida .

La costumbre mas extendida es, como hemos visto, usar la barra /, sin embargo, si posteriormente vamos a necesitar incluir el mismo caracter delimitador en el patrón, tendremos que escaparlo, por lo que tiene sentido usar en esos casos unos delimitadores distintos: (), {}, [], o < >

Modificadores

Se colocan despues del patrón:

```
<?
m // multilínea. Si nuestra cadena contiene varias lineas físicas (n)
// respeta esos saltos de línea, lo que significa, por ejemplo,
// que las anclas ^ $ no se aplican al principio y final de la
// cadena, sino al principio y final de cada linea

s // El metacaracter . representa cualquier carácter menos el de
// nueva linea. Con el modificador "s" tambien representa la nueva linea.

i // Se confronta el patrón con la cadena ignorando Mayusculas minusculas
```

x // ignora espacios (salvo que esten escapados o incluidos
// específicamente dentro del rango de búsqueda. Ignora cualquier caracter
// despues de almohadilla (#) hasta nueva línea. Sirve para incluir
// comentarios y hacer mas legible el patrón.

e // solo en preg_replace. Evalua las ocurrencias como código php antes
// de realizar la sustitución.

A // El patrón es forzado a ser "anclado", esto es, solo existirá una
// ocurrencia si es al inicio de la cadena.

E // el carácter $ en el patrón casará con el fin de la cadena.
// Sin este modificador, $ casa tambien con el carácter inmediatamente
// antes del de una nueva línea.

U // Este modificador invierte la "codicia" de los cuantificadores.
*// Si aplicamos U, * se convierte en perezoso (lazy) y *? vuelve a su*
// comportamiento normal.
// Un cuantificador es "codicioso" (greedy) cuando intenta capturar todas
// las ocurrencias posibles, y perezoso (lazy) cuando captura la
// ocurrencia mas corta. ?>

Delimitadores de palabra

En las funciones PCRE puedes usar b que indica el comienzo y fin de una palabra (o mejor, de una secuencia alfanumerica):
/bpatronb/

B, por el contrario, se refiere a un patrón que no está al comienzo o fin de una palabra.

Codicioso o no

Las expresiones regulares que usan cuantificadores tienden a ser todo lo codiciosas que les sea permitido, siempre que respeten el patrón a seguir. Con el modificador U se invierte este comportamiento.

En modo codicioso el patrón casará todas las ocurrencias que pueda, mientras que en modo lazy o ungreedy, casará solo la mas corta posible (e).
Advierte que las dos soluciones son correctas.

patrón: /http://.*.(com|net|org)/esto es un link: http://www.abc.com y este otro mas: http://www.blah.com

patrón: /http://.*.(com|net|org)/Uesto es un link: http://www.abc.com y este otro mas: http://www.blah.com

Las funciones

Tenemos cinco funciones PCRE:

preg_match()
preg_match_all()
preg_replace()
preg_split()
preg_grep()

preg_match busca el patrón dentro de la cadena, devolviendo TRUE si es hallado:
patron, cadena [,array de resultados]

Si indicamos el tercer parámetro, tendremos los resultados en un array; $array[0] contendrá la ocurrencia del patrón; $array[1] tendrá la cadena que case con el primer subpatrón y así sucesivamente.

preg_match_all encuentra todas las ocurrencias del patrón en la cadena.
patron, cadena, array de patrones, orden

Si proporcionamos el cuarto parámetro, colocara las ocurrencias en un array siguiendo el orden indicado.

preg_replace busca y reemplaza el patrón en la cadena. Tanto el patrón como la sustitución pueden pasarse en array, y pueden contener expresiones regulares.
Puede tambien especificarse un límite máximo de ocurrencias.

preg_split opera como split aunque tambien puedes pasarle expresiones regulares

preg_grep busca el patrón dentro de un array, y devuelve otro array con las ocurrencias.
patron, array

Autor: Blasten
http://www.blasten.com/contenidos/19073?pag=7

Buscador Simple

En este artículo veremos como crear un buscador con PHP y MySQL que servirá para cualquier tabla MySQL de nuestra base de datos y que podrá ser mostrado facilmente en nuestro sitio web.

El script consta de tres partes. La configuración, el formulario y el proceso del formulario. La primera parte, la más sencilla de todas, es donde tendrás que poner los datos de tu base de datos y la tabla donde quieres que el buscador realice las búsquedas. No tiene pérdida. Al final de la configuración realizamos la conexión a la base de datos ya que la usaremos cada vez que se acceda al buscador.

En la segunda parte se trata la creación del formulario. Como no conocemos los campos de la tabla tenemos que generar ese camop del formulario dinámicamente. Para ello utilizamos la sentencia "SHOW FIELDS FROM table" que nos devuelve información de todos los campos que hay en la tabla. De esta forma conseguimos que nos liste todos los campos de la tabla que hayamos escogido al configurar, en el formulario.

En la tercera parte, que solo se ejecuta si se ha enviado el formulario, se realiza la búsqueda con los datos obtenidos. Para ello utilizamos una sentencia de SQL de este tipo: "SELECT * from tabla WHERE campo LIKE '%valor%'" la cual nos devuelve todas las filas donde haya encontrado algo que contenga el valor en el campo seleccionado. Finalmente mostramos los resultados obtenidos utilizando un bucle para recorrer todos los campos de la tabla.

Configurar y subir, así de facil :)

Archivo: buscador_generico.php

```
<?
// Buscador para tablas MySQL escrito en PHP. Por Alex para www.webtaller.com
// Creado el 13-10-2003


//////////////////////////
// Configuración
//////////////////////////

//modifica estas variables según tu servidor de MySQL

$bd_servidor = "localhost";

$bd_usuario = "pepito";

$bd_contrasenya = "grillo";

$bd_bdname = "mybd";

$bd_tabla = "unatabla"; // Tabla donde se harán las búsquedas

// Conexión y selección de la base de datos

$link = mysql_connect($bd_servidor,$bd_usuario,$bd_contrasenya);

mysql_select_db($bd_bdname,$link);

//////////////////////////
// Formulario
//////////////////////////

?>

<center>
<p><h2>Introduce las palabras para la búsqueda</h2></p>
<p><form name="buscador" method="post" action="buscador_generico.php"><br>
Buscar en:
<select name="campo">
<?php
```

```
//Con este query obtendremos los campos por los cuales el usuario puede buscar

$result = mysql_query("SHOW FIELDS FROM `$bd_tabla`",$link);

while($row = mysql_fetch_row($result)) {

// en $row[0] tenemos el nombre del campo
// de esta manera no necesitamos conocer el nombre de los campos
// por lo que cualquier tabla nos valdrá

?>
<option value="<?php echo $row[0]; ?>"><?php echo $row[0]; ?></option>
<?php

}

?>
</select>
Palabra(s): <input type="text" name="palabra"><br>
<input type="submit" value="Enviar" name="enviar">
</form></p>
</center>

<?

//////////////////////////////
// Proceso del Formulario
//////////////////////////////

if(isset($_POST['enviar'])) {

// Solo se ejecuta si se ha enviado el formulario

$query = "SELECT * from $bd_tabla WHERE `{$_POST['campo']}` LIKE '%{$_POST['palabra']}%'";

$result = mysql_query($query,$link);



$found = false; // Si el query ha devuelto algo pondrá a true esta variable

while ($row = mysql_fetch_array($result)) {

$found = true;

echo "<p>";

foreach($row as $nombre_campo => $valor_campo) {

// Tenemos que mostrar todos los campos de las filas donde se haya
// encontrado la búsqueda.


if(is_int($nombre_campo)) {

continue; //Cuando hacemos mysql_fetch_array, php genera un array
// con todos los valores guardados dos veces, uno con
// índice numérico y otro con índice el nombre del campo.
// Solo nos interesa el del nombre del campo.

}

echo "<b>".$nombre_campo."</b> : ".$valor_campo."<br>";
}

echo "</p>";
```

```
}

if(!$found) {

echo "No se encontró la palabra introducida";

}

}
?>
```

Alex.

Como generar un Thumbnail en PHP usando GD

Antes que nada avisar que se requieren las librerías GD para poder realizar los thumbnails. Y en función de la versión de GD, podremos usar una u otra función

Primero pegaré el código de una función con la que generaremos directamente un thumbnail de alta calidad.

```
function thumbjpeg($imagen,$altura) {

// Lugar donde se guardarán los thumbnails respecto a la carpeta donde está la imagen "grande".
$dir_thumb = "thumbs/";

// Prefijo que se añadirá al nombre del thumbnail. Ejemplo: si la imagen grande fuera "imagen1.jpg",
// el thumbnail se llamaría "tn_imagen1.jpg"
$prefijo_thumb = "tn_";
$camino_nombre=explode("/",$imagen);

// Aquí tendremos el nombre de la imagen.
$nombre=end($camino_nombre);

// Aquí la ruta especificada para buscar la imagen.
$camino=substr($imagen,0,strlen($imagen)-strlen($nombre));

// Intentamos crear el directorio de thumbnails, si no existiera previamente.
if (!file_exists($camino.$dir_thumb))
mkdir ($camino.$dir_thumb, 0777) or die("No se ha podido crear el directorio $dir_thumb");

// Aquí comprovamos que la imagen que queremos crear no exista previamente
if (!file_exists($camino.$dir_thumb.$prefijo_thumb.$nombre)) {
echo $camino.$dir_thumb.$prefijo_thumb.$nombre." NO existía<br>n";
$img = imagecreatefromjpeg($camino.$nombre) or die("No se encuentra la imagen $camino$nombre<br>n");

// miramos el tamaño de la imagen original...
$datos = getimagesize($camino.$nombre) or die("Problemas con $camino$nombre<br>n");

// intentamos escalar la imagen original a la medida que nos interesa
$ratio = ($datos[1] / $altura);
$anchura = round($datos[0] / $ratio);

// esta será la nueva imagen reescalada
$thumb = imagecreatetruecolor($anchura,$altura);

// con esta función la reescalamos
imagecopyresampled ($thumb, $img, 0, 0, 0, 0, $anchura, $altura, $datos[0], $datos[1]);

// voilà la salvamos con el nombre y en el lugar que nos interesa.
imagejpeg($thumb,$camino.$dir_thumb.$prefijo_thumb.$nombre);
}
}
```

Para llamar a la función sencillamente hacer:

```
thumbjpeg($imagen, 125);
```

En este caso, '$imagen', es la imagen que queremos reducir, y '125', es la altura en píxeles que queremos que tenga la imagen reducida, de modo que el ancho quede proporcionado respecto a la imagen original.

Con la instrucción “or die()” se mostrará en pantalla el mensaje entrecomillado solo en el caso de que fallara la primera sentencia de la línea.

NOTAS:

Si en lugar de GD2 disponemos de GD1, no podremos utilizar la función **imagecopyresampled()** y deberíamos conformarnos con la función **imagecopyresized()** , que utiliza los mismos parámetros, lo malo es que ésta última da como resultado una imagen de calidad más pobre. A cambio, es mucho más rápida que la anterior.

Así si tenemos GD2, podemos usar la que más nos interese, si queremos CALIDAD y no nos importa el número de ciclos consumidos en el servidor,

imagecopyresampled();

en caso contrario:

imagecopyresized();

Si lo que queremos es crear imágenes "al vuelo", sin guardarlas en ningún archivo (cosa poco recomendable si usamos la función de más calidad), el segundo parámetro de la función imagejpeg() no se debe de poner, quedando la llamada en el scritp que hay más arriba:

imagejpeg($thumb);

Bueno, démonos cuenta que este script solo sirve para reducir imágenes de tipo JPEG, no de otro, para otros formatos, tan solo habría que cambiar un par de funciones por las correspondientes en los formatos que nos interesen.

Ejemplo:

imagecreatefromjpeg() por **imagecreatefrompng()** o **imagecreatefromwbmp()** o etc.

y

imagejpeg() por **imagepng()** o **imagewbmp()** o etc.

Autor: Basilio Vera.
http://www.aclantis.com/section-viewarticle-145.html

Encuesta con PHP

Vamos a ver cómo podemos crear una sencilla encuesta con MySQL que almacene las votaciones de los usuarios en la base de datos junto con su direccion ip para controlar que cada usuario vote una sola vez.

El primer paso que haremos será el acondicionamiento de la base de datos, en la que crearemos una tabla usando la siguiente sentencia:

```
CREATE TABLE `encuesta` (
`ip` VARCHAR( 16 ) NOT NULL ,
`voto` INT( 1 ) NOT NULL ,
UNIQUE (
`ip`
)
);
```

Excepto si queremos que el mismo usuario pueda votar repetidas veces en cuyo caso haremos:

```
CREATE TABLE `encuesta` (
`ip` VARCHAR( 16 ) NOT NULL ,
`voto` INT( 1 ) NOT NULL
);
```

El funcionamiento de la encuesta es muy sencillo, el programa leerá las posibles opciones de voto de un array llamado **$opciones**, que podras modificar a tu voluntad, entonces iterará tantas veces como elementos tenga el array para mostrar los resultados actuales de cada opción.

Para poder mostrar los porcentajes, lo primero que hacemos es una consulta general que nos devuelve el número total de votos recibidos, entonces, en cada opcion haremos la siguiente operación:

```
$porcentaje = round($votos/$total*100,2);
```

Que nos devolverá el porcentaje de votos redondeado a 2 decimales usando la función **round**.

Artículo por cortesía de Eloi de San Martín
www.programacionweb.net

Editar y Borrar datos con MySQL y PHP

Editar

La edición de datos en mysql, combina opciones de Insertar datos a MySQL y de Consultas MySQL , tendremos que hacer una consulta cómo la siguiente:

```
UPDATE tabla SET campo = 'valor' WHERE condicion
```

Como veis, volvemos a utilizar la clausula **WHERE** para escojer las entradas que hay que editar, podemos actualizar varios campos de la siguiente manera:

```
UPDATE tabla SET campo = 'valor', campo2 = 'valor2' WHERE condicion
```

El metodo no tiene mas secretos que esto, veamos un ejemplo real para ver cómo funciona exactamente desde PHP:

```
<?php
$sql = "UPDATE agenda SET telefono = 555405181 WHERE nombre = 'eloi'" ;
mysql_query ( $sql , $db );
?>
```

Recordar que $db contiene un identificador de la conexión.

Borrar
Borrar unas determinadas de una tabla de MySQL es incluso más sencillo que editarlas, pues solo es nescesario indicar que entradas queremos borrar con una clausula **WHERE** y en que tabla lo haremos, y esto junto con la palabra **DELETE FROM** nos darán el resultado que esperamos:

```
DELETE FROM tabla WHERE condicion
```

No creo que sea nescesario poner el ejemplo de este caso, si ahún asi salen dudas consultar en el **foro** .

Autor: Eloi de San Martín.
www.programacionweb.net

Mostrar la fecha en Español con PHP

Si imprimimos con normalidad la fecha en nuestra página, mediante PHP, seguramente esta aparecerá en inglés. Si ponemos algo así:

```
<?php echo strftime("%A %d de %B del %Y"); ?>
```

Aparecerá por ejemplo "Thursday 17 de February del 2005", cosa que no quedará muy acorde con nuestra página, si esta está en idioma Español.

La mejor solución es configurar las locales y PHP hará el resto por nosotros. Las locales son traducciones de cosas básicas, como la fecha, que suelen venir en el sistema operativo. Veamos como configurar las locales para el idioma Español:

```
set_locale(LC_ALL,"es_ES@euro","es_ES","esp");
```

Ponemos varias opciones por si la primera no está disponible, saltará a la siguiente y así sucesivamente. Si ahora probamos el mismo código:

```
<?php

set_locale(LC_ALL,"es_ES@euro","es_ES","esp");
echo strftime("%A %d de %B del %Y");

?>
```

Aparecerá "Jueves 17 de Febrero del 2005".

Si no disponemos de locales, podemos hacer la traducción nosotros mismos. Por ejemplo, para el día de la semana haríamos algo así:

```
<?php

$dias = array("Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sábado");
echo "Hoy es ".$dias[date('w')];

?>
```

date('w') devuelve el día de la semana, entre 0 y 6. 0 es el domingo y 6 el sábado. Para el mes podemos hacer algo parecido, sabiendo que date('n') devuelve el mes, entre 1 y 12. Hay que tener en cuenta que en este caso no empieza desde 0.

Por Alex.

Proteger Descargas en PHP

Muchas veces, desde otra página web, enlazan directamente a un archivo de nuestro servidor, esto sobrecarga nuesto ancho de banda sin reportarnos ningún beneficio, para evitarlo podemos tomar algunas medidas a la hora de enviar el archivo.

Comprovar el referente
Podemos restringir a que solo se pueda descargar el archivo cuando se proviene de una determinada página, esto sería efectivo si no hubiera usuarios que navegan detrás de proxys que eliminan el referente, en este caso ninguno de estos usuarios podría descargar el archivo ni que proviniera de la página correcta.

Usar una cookie
En mi opinión este método es mas eficaz, guardaremos una cookie en la página donde situamos el enlace hacia nuestra descarga, de esta manera luego podremos comprobar que préviamente se ha pasado por esa página, es decir, que la descarga está autorizada:

```
<?php
// Esto tiene que estar al principio del
// todo del documento antes de enviar nada
// al navegador (ni siquiera un espacio)
// de lo contrario tendremos un error
setcookie ( 'descarga' , '1' );
?>
```

Luego situaremos el archivo a descargar en una carpeta secreta a la que daremos un nombre aleatório para que no sea fácil de encontrar, por ejemplo 23hi938fdfgh39, y crearemos el siguiente archivo:

descargar.php

```
<?php
// Indicamos el nombre del directório
define ( 'dir' , '23hi938fdfgh39' );
// Comprovamos que exista la cookie
if( $_COOKIE [ 'descarga' ] == '1' ){
// Si existe la cookie intentamos
   // leer el archivo
$archivo = $_GET [ 'archivo' ];
   if( file_exists ( dir . '/' . $archivo )){
// Si existe el archivo lo enviamos
header ( 'Content-Type: application/octet-stream' );
header ( 'Content-Disposition: attachment; filename=' . $archivo );
     echo file_get_contents ( dir . '/' . $archivo );
   } else {
// Sino existe el archivo enviamos
     // un error 404
header ( 'HTTP/1.0 404 Not Found' );
     echo '<h1>ERROR</h1><br />No se h' ,
'a encontrado el archivo sol' ,
'icitado' ;
   }
} else {
// Sino hay cookie enviamos un error
   // 401
header ( 'HTTP/1.0 401 Unauthorized' );
   echo '<h1>ERROR</h1><br />No puedes' ,
'acceder a este archivo desde ' ,
'un servidor externo' ;
}
?>
```

Entonces desde la página donde hemos creado la cookie podemos llamar a descargar el archivo haciendo un enlace hácia descargar.php?archivo=NOMBRE.ZIP, por ejemplo:

```
<a href="descargar.php?archivo=chat.zip">Descargar</a>
```

Y con este sencillo método habremos protegido nuestros archivos de la descarga externa mediante cookies.

Artículo por cortesía de Eloi de San Martín
www.programacionweb.net

Pasar la resolución de JavaScript a PHP

En esta ocasión voy a mostrarles como pasar la resolución de JavaScript a PHP.

Todos sabemos que el lenguaje PHP, al ser del lado del servidor, no puede tomar nuestra resolución. Pues, hay una forma sencilla de pasar las medidas de JavaScript a PHP con un solo click!

Para setear una cookie con JavaScript, el comando es el siguiente:

document.cookie = 'NOMBRE=VALOR; expires=FECHA; path=CAMINO; domain=DOMINIO; SECURE';

Teniendo esto en cuenta, lo que haremos será un script, mezcla de PHP y JavaScript. Consistirá en dos archivos: mostrar.php y getres.php .

El código es el siguiente:

```
<?
/*****************************************
JavaScript to PHP Screen Resolution vBETA
by Lenn García
Date: 08-29-2004
*****************************************/
?>
<script language="javascript">
function SetCookie() {
var width = screen.width;
var height = screen.height;
var res = width + 'x' + height;
document.cookie = 'PHPRes='+res;
location = '<?=$GLOBALS['siteurl'];?>';
}

function CheckResolution(width, height) {
if(width != screen.width && height != screen.height) {
SetCookie();
}
}
</script>
<?
if(isset($_COOKIE['PHPRes']) || !empty($_COOKIE['PHPRes'])) {
$res = explode("x",$_COOKIE['PHPRes']);
$width = $res[0];
$height = $res[1];
?>
<script language="javascript">
CheckResolution(<?=$width;?>,<?=$height;?>);
</script>
<?
} else {
?>
<script language="javascript">
SetCookie();
</script>
<?
}
?>
```

Lo que hacemos con este código es lo siguiente:

1. En la función JavaScript SetCookie definimos dos variables con el ancho y el alto de la resolución, y lo colocamos en un string de valor anchoxalto .
2. Seteamos la cookie de nombre PHPRes con el valor mencionado anteriormente, y refrescamos la página.
3. Luego, en la función JavaScript CheckResolution comparamos los valores actuales de la resolución y los invocados como parámetros en la función. Si no son iguales, se crea la cookie con los datos actualizados.
4. Pasando a PHP, se verifica la existencia de la cookie PHPRes , y si la misma existe, se separan los valores numéricos de la x .
5. Los valores de ancho y alto se llaman en la función JavaScript CheckResolution , que verificará si todo está actualizado correctamente.

6. Si la cookie no existe, se la creará.

El archivo desde el que se llama al script debe cumplir ciertos requisitos:

Antes de invocar al archivo getres.php debe definir una variable $GLOBALS con la dirección de la página original ($_SERVER['REQUEST_URI']).

La página original (mostrar.php) sería así:

```
<?
$siteurl = $_SERVER['REQUEST_URI'];
$GLOBALS['siteurl'] = $siteurl;
require('getres.php');
echo $width.'x'.$height;
?>
```

Lo que hace el código es tomar la dirección desde el dominio principal (pero sin él), y luego convertirla en variable super-global. Pide el archivo getres.php , y luego se muestran las variables.

Recuerda que debes tener habilitadas las cookies, en tu navegador, para poder utilizar este método.

Bueno, eso es todo por hoy! :D

Espero que esto te sea de ayuda, y cualquier cosa estoy en los foros.

Autor: Lenn García

Original de webexperto

Encriptar contraseñas en MD5 con PHP

Si en nuestra página web tenemos un sistema de usuarios y queremos proteger las contraseñas para prevenir posibles vulnerabilidades en nuestro servidor, es una medida eficaz encriptar las contraseñas, de manera que si alguien puede acceder a ellas no pueda ver la contraseña si no su encriptación.

Para mejorar este sistema, lo que haremos es usar un algoritmo de encriptación de un solo sentido, es decir que no se puede desencriptar de ninguna manera, como por ejemplo md5.

Para guardar la contraseña encriptada en md5, usaremos la funcion **md5()** de PHP:

```
<?
$contrasena = md5 ( $contrasena );
?>
```

Ya podemos guardar la contraseña en nuestra base de datos o fichero, pero... ¿como haremos para comprovar la contraseña en el inicio de sesión?

Muy fácil, como en la base de datos tenemos la contraseña en md5, encriptaremos la contraseña que escribe el visitante de la misma manera que hemos encriptado la contraseña del usuario en el momento de su regístro, ahora ya podremos comparar la contraseña enviada con la almacenada en el servidor, si la encriptación coincide es que la contraseña es correcta.

Pero si no usamos una transmisión segura (pe: SSL) sigue habiendo un problema, cuando el usuario envía los datos al servidor, la contraseña es enviada sin encriptar, y en ese momento puede ser capturada por un tercero. Para evitar esto, podemos encriptar la clave en el ordenador del cliente usando JavaScript gracias a **Javascript MD5** , en lugar de encriptarla en el servidor usando PHP.

Artículo por cortesía de Eloi de San Martín
www.programacionweb.net

Enviar un e-mail HTML con PHP

Enviar un e-mail con PHP es muy sencillo, tan solo tenemos que utilizar la función mail. Pero cuando escribimos código HTML en el cuerpo del mensaje, este lo recibimos como texto y no como una página web, como querríamos. Esto tiene facil solución, solo necesitamos añadir la cabecera "Content-type: text/html" en el e-mail y el código que enviemos se interpretará como HTML. Veamos como:

```
<?php

$codigohtml = '

        <html>
        <head>
        <title>E-Mail HTML</title>
        </head>
        <body>
        <a href="http://www.webtaller.com">Ir a WebTaller</a>
        </body>

';

$email = 'pepito@grillo.com';
$asunto = 'E-Mail HTML';
$cabeceras = "Content-type: text/html\r\n";

mail($email,$asunto,$codigohtml,$cabeceras);

?>
```

De esta forma, los e-mails que enviemos se verán como una página Web. En las cabeceras podemos añadir otras cosas, como por ejemplo si queremos especificar quien envía el e-mail haremos:

```
        $cabeceras = "From: direccion@email.dom\r\nContent-type: text/html\r\n";
```

De esta forma, el remitente del e-mail sería **direccion@email.dom**

Por Alex.

Subir Archivos

Para subir archivos a un servidor, lo único que debemos hacer es poner en un formulario un campo de archivo como los siguientes:

```
<form action="" method="post" enctype="multipart/form-data">
  <input type="file" name="file" />
  <input type="submit" name="submit" value="Subir imagen" />

</form>
```

Al enviar el formulario, el navegador envía automaticamente el archivo del campo a la carpeta temporal del servidor, pero el problema es una vez en esta carpeta cómo moverlo a la carpeta que se nos antoje.

Y aquí es donde entra en acción PHP y la función **move_uploaded_file** que moverá el archivo subido de la carpeta temporal a la carpeta que nosotros le digamos e incluso con el nombre que le pongamos.

```
<?
$destino = 'uploaded' ;
move_uploaded_file ( $_FILES [ 'file' ][ 'tmp_name' ], $destino . '/' . $_FILES [ 'file' ][ 'name' ]);

?>
```

Donde $_FILES['file']['tmp_name'] identificara el archivo temporal subido al servidor, $destino, la carpeta en la que lo queremos mover y $_FILES['file']['name'] el nombre original del archivo.

Además también podemos conocer otros parámetros del fichero subido como por ejemplo el tamaño, vamos a ver un ejemplo:

```
<?
$destino = 'uploaded' ;
// Leemos el tamaño del fichero
$tamano = $_FILES [ 'file' ][ 'size' ];
// Comprovamos el tamaño
if( $tamano < 500 ){
move_uploaded_file ( $_FILES [ 'file' ][ 'tmp_name' ], $destino . '/' . $_FILES [ 'file' ][ 'name' ]);

}
else echo "El tamaño es superior al permitido" ;
?>
```

Tambien podemos saber el tipo de archivo subido con la siguiente variable: $_FILES['file']['type'];

Nota: Para versiones anteriores a la 4.0.1 de PHP, en lugar del vector $_FILES, debemos usar $HTTP_POST_FILES.

Artículo por cortesía de Eloi de San Martín
www.programacionweb.net

Enviar un Formulario por Correo Electrónico

Este artículo habla de como enviar un formulario a una dirección de e-mail. Esto es de lo más sencillo que se puede hacer con PHP, aunque bastante recurrente y a los principiantes les puede servir para ir entendiendo como funcionan los lenguajes de servidor.

El formulario es un formulario básico y se le pueden añadir los campos que se desee, ya que la aplicación recogerá todas las variables pasadas por el método "post" y las enviará por el mail, por lo que el método (method) del formulario debe ser "post" y el "action" debe ir a la página PHP que contenga la función.

la función recibe los campos "para", "asunto" y "texto", y los envía con el mail, pasando los campos "para" y "asunto" tal cual y añadiendo al campo "texto" los datos del formulario. También recibe un campo "de" que recibe el mail de quién se quiera que aparezca como emisor del mail.

```
<!--Copyright © McAnam.com (Generador de formularios V. 1.1)-->
<html>
<head>
<title>Rellene el formulario</title>
</head>
<body>
<form name='formulario' id='formulario' method='post' action='pagina_mail.php' target='_self'>
<p>Nombre <input type='text' name='Nombre' id='Nombre'></p>
<p>Apellidos <input type='text' name='Apellidos' id='Apellidos'></p>
<p>E-mail <input type='text' name='E-mail' id='E-mail'></p>
<p><input type='radio' value='Hombre' name='Sexo' id='Sexo'>Hombre</p>
<p><input type='radio' value='Mujer' name='Sexo' id='Sexo'>Mujer</p>
<p align='center'>
<input type='submit' value='Enviar formulario'>
<input type='reset' value='resetear formulario'>
</p>
</form>
</body>
</html>
```

```
<?php
    //Copyright © McAnam.com

    function form_mail($sPara, $sAsunto, $sTexto, $sDe){

        if ($sDe)$sDe = "From:".$sDe;

        foreach ($_POST as $nombre => $valor)
            $sTexto = $sTexto."n".$nombre." = ".$valor;

        return(mail($sPara, $sAsunto, $sTexto, $sDe));
    }

    //Ejemplo de como usar:
    if (form_mail("usuario@suweb.com",
                "Activación de formulario",
                "Los datos introducidos en el formulario son:nn",
                "tu@tuweb.com"
                )
        )
        echo "Su formulario ha sido enviado con exito";

?>
```

Original de McAnam.com