Cadenas alfanuméricas (II)

Ver índice

Funciones de cadenas

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

AddSlashes(*cadena*)

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

stripslashes(*cadena*)

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

chunk_split(*cad, n, sep*)

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

parse_str(*cadena*)

Devuelve las variables –con su valor– indicadas dentro de la cadena (observa la sintaxis del ejemplo).
Dentro de la cadena cada variable se denomina con un nombre que va seguido de un signo igual. Los espacios se señalan con el signo **+** y los separadores de variables son signos **&**

explode(*sep, cad,n*)

Devuelve un **array** cuyos elementos contienen cada una de las porciones de la cadena (**cad**) comprendidas entre dos de los caracteres señalados como (*sep*) hasta el máximo de porciones señaladas (**n**).
Los caracteres separadores no son incluídos en las cadenas resultantes.
Si no se indica la cantidad de porciones, será fraccionada toda la cadena.
Si se indica número, el último *trozo* contendrá **toda** la cadena restante.

implode(*sep, array*)

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

join(*sep, array*)

Formatos de salida

Estos son algunos ejemplos en los que continuamos con las aplicaciones de las funciones PHP al manejo de cadenas:

Marcas, divisiones y uniones de cadenas

Variable cadena	Sintaxis	Resultado
$a="Esta ' y \ y también NUL"	AddSlashes($a)	**$a="Esta \' y \\ y también NUL"**
$a="Esta \' y \\ y también NUL"	stripslashes($a)	**Esta ' y y también el nulo**
$a="Esta es una cadena larga que presuntamente será troceada"	chunk_split($a,5,"–")	**Esta –es un–a cad–ena l–arga –que p–resun–tamen–te se–rá tr–ocead–a–**
$a="Esta es una cadena larga que presuntamente será troceada"	chunk_split($a,5)	**Esta es un a cad ena l arga que p resun tamen te se rá tr ocead a**
$a="Esta es una cadena larga que presuntamente será troceada"	chunk_split($a,76,"-")	**Esta es una cadena larga que presuntamente será troceada-**
$todo="**v1=Esto+sera+una+variable&v2=esto+otra&p[]=incluso+un+array**"		
Divide la cadena **$todo** en sus componentes	parse_str($todo); echo $v1; echo $v2; echo $p[0];	**Esto sera una variable esto otra incluso un array**
$a="Esta cadena sera devuelta en trozos"		
Recogerá en un array cada uno de los trozos delimitados por los separadores	$trozo1=explode(" ",$a); echo $trozo1[0]; echo $trozo1[1]; echo $trozo1[2]; echo $trozo1[3]; echo $trozo1[4]; echo $trozo1[5]; >	**Esta cadena sera devuelta en trozos**
Recogerá en un array cada uno de los trozos delimitados por los separadores	$trozo2=explode("a",$a); echo $trozo2[0]; echo $trozo2[1]; echo $trozo2[2]; echo $trozo2[3]; echo $trozo2[4]; echo $trozo2[5];	**Est c den ser devuelt en trozos**
Recogerá en un array cada uno de los trozos delimitados por los separadores hasta un máximo de 3	$trozo3=explode(" ",$a,3); echo $trozo3[0]; echo $trozo3[1]; echo $trozo3[2]; echo $trozo3[3]; echo $trozo3[4]; echo $trozo3[5];	**Esta cadena sera devuelta en trozos**
Recogerá en un array cada uno de los trozos delimitados por los separadores hasta un máximo de 3	$trozo4=explode("a",$a,3); echo $trozo4[0]; echo $trozo4[1]; echo $trozo4[2]; echo $trozo4[3]; echo $trozo4[4]; echo $trozo4[5];	**Est c dena sera devuelta en trozos**
implode(" ",$trozo1)	**Esta cadena sera devuelta en trozos**	
implode("*",$trozo2)	**Est* c*den* ser* devuelt* en trozos**	
implode("-",$trozo3)	**Esta-cadena-sera devuelta en trozos**	
implode(":",$trozo4)	**Est: c:dena sera devuelta en trozos**	
join(" ",$trozo1)	**Esta cadena sera devuelta en trozos**	
join("*",$trozo2)	**Est* c*den* ser* devuelt* en trozos**	
join("-",$trozo3)	**Esta-cadena-sera devuelta en trozos**	
join(":",$trozo4)	**Est: c:dena sera devuelta en trozos**	
$cadena="Esta cadena será dividida con la función strtok"		

$trocin = strtok ($cadena," ");
 	
while ($trocin) {
echo "$trocin
";
$trocin = strtok (" ");
}
		**Esta
		cadena
		será
		dividida
		con
		la**

Es *idéntica* a **implode**.

strtok(*cad , sep***)**

Esta función divide la cadena **cad** en *trozos* delimitados por el separador que se indica en **sep**.
Cuando se invoca la primera vez –extrae el primer trozo– debe llevar las sintaxis **strtok(***cadena*,*sep***)**.
Al invocarla sucesivamente, se escribe *solo* **strtok (" ")** e irá recogiendo de forma *secuencial* los trozos sucesivos.

Encriptación de cadenas

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

bin2hex(*cadena***)**

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

crypt(*cadena***)**

Devuelve la cadena *encriptada* utilizando una *semilla aleatoria* de dos caracteres.

Por su caracter aleatorio, si se ejecuta dos veces seguidas –tal como puedes observar en el ejemplo– dará dos resultados diferentes.

crypt(*cadena***,"xx")**

Devuelve la cadena *encriptada* utilizando como *semilla* los dos caracteres (entre comillas) que se escriben como segundo parámetro de la función.
Tanto en este supuesto como en el anterior, los dos primeros caracteres de la cadena encrip- tada coinciden con los que han sido utilizados como *semilla*.

md5(*cadena***,"xx")**

Aplica el algoritmo *md5* –lo comentamos a la derecha– y devuelve la *huella digital* generada por él.

crc32(*cadena***)**

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

Se utiliza muchísimo en los programas de *compresión* y *descompresión* de ficheros. Se aplica en el momento de *comprimir* y se incluye el valor obtenido dentro del fichero comprimido. Después de la *descompresión* se vuelve a aplicar el mismo algoritmo y se comparan ambos valores. La coincidencia será

	funcion strtok
$trocin = strtok ($cadena," "); echo $trocin," "; $trocin1 = strtok (" "); echo $trocin1," "; $trocin2 = strtok (" "); echo $trocin2," ";	**Esta cadena Esta**

$trocin = strtok ($cadena,"a");
while ($trocin) {
echo "$trocin
";
$trocin = strtok ("a");
}
	Est c den será dividid con l función strtok

Encriptaciones y codificaciones

Variable cadena	Sintaxis	Resultado
$a="Esta es la cadena"	bin2hex($a)	**4573746120657320636164656e610a**
$a="Encriptame"	crypt($a)	**1jQ5.w22.$QPBJQ/mNSxxdZ2ccJpZo41**
$a="Encriptame"	crypt($a)	**$1$3v1.O.5.$MxTxlSqC2VwZiDyZjdz0o0**
$a="Encriptame"	crypt($a,"zq")	**zqQ4qOeELzPFg**
$a="Encriptame"	crypt($a,"zq")	**zqQ4qOeELzPFg**
$a="Encriptame"	crypt($a,"@$")	**@$eKFJms35tL.**
$a="Encriptame"	md5($a)	**67c3ca0aefda2595138168a85e7b33a0**
$a="Encriptame"	md5($a)	**67c3ca0aefda2595138168a85e7b33a0**
$a="Encriptame"	crc32($a)	**-1128189886**

Búsquedas y recuentos de caracteres

Variable cadena	Sintaxis	Resultado
$a="Contando caracteres"	count_chars($a,0)	**Array**
$a="Contando caracteres"	$b=count_chars($a,0); echo $b[97];	**3**
$a="Contando caracteres"	$b=count_chars($a,0); echo $b[ord("o")]	**2**
$a="Pepe Perez el perverso pecador en penitencia"	substr_count($a,"Pe");	**2**
$a="Pepe Perez el perverso pecador en penitencia"	substr_count($a,"pe");	**4**

La función **count_char($a,0)** devuelve un **array** cuyos *índices* son los *códigos ASCII de los caracteres* y cuyos *valores* son el *número de veces que se repite* cada uno de ellos.

La función **substr_count($a,"*cadena*")** determina el *número de veces* que aparece la *cadena* dentro de *$a*. Diferencia entre mayúsculas y minúsculas.

El algoritmo *md5*

Este algoritmo presenta como peculiaridades que –tenga la dimensión que tenga la cadena a la que se aplique– genera siempre una *huella digital* que no es otra cosa que una cadena formada por **32 caracteres** y que **no dispone** de ningún mecanismo *inverso*.

Seguramente habrás vivido esta experiencia. En muchos espacios de Internet –grupos de noticias, cuentas de correo web, etcétera– que requieren un login y una contraseña cuando utilizas la opción de *recuperar contraseñas* no te envían tu contraseña anterior, sino que te generan y envían una nueva.

Esto ocurre porque, por razones evidentes de seguridad, las contraseñas se almacenan usando estas *huellas digitales* y resulta imposible recuperar los valores originales.

La única solución en estos casos es crear una nueva contraseña (suelen hacerlo con las funciones de números aleatorios), enviarla de forma automática por correo electrónico y sustituir el valor anterior del registro de usuarios por el resultado de la codificación **md5** de la nueva contraseña.

la garantía de que el fichero obtenido
es idéntico al original.