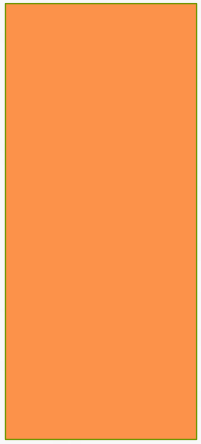
MANEJO DE CADENAS

PROGRAMANDO CON PHP



INTRODUCCIÓN

- El tratamiento de cadenas es muy importante en PHP.
  - El objetivo final del procesamiento de un fichero PHP está relacionado con la generación de documentos HTML (casi siempre)
- Existe un amplio conjunto de funciones para el manejo de cadenas.
  - Veremos las principales.

ACCESO AL CONTENIDO

ACCESO AL CONTENIDO DE UNA CADENA

- Se puede acceder a cada uno de los caracteres que componen una cadena de la misma forma que lo hacemos con los elementos de un array.
  - Haciendo referencia a su posición en la cadena.
  - Necesitamos saber la longitud de la cadena
- **strlen (cadena):** devuelve la longitud de la cadena.

EJERCICIO

- Escribir una página en PHP que a partir de una cadena de caracteres muestre cada uno de los caracteres en una celda distinta de una tabla. Se debe indicar siempre en qué posición está el carácter en cuestión.

Carácter	Posición
S	O
a	1
l	2
u	3
d	4
o	5
s	6

```php
<h2> Función <i>strlen<i> </h2>
<?php
    $cadena = "saludos";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr bgcolor='pink'><td>Carácter</td>";
    echo "<td>Posición</td></tr>";
    for($i=0; $i<strlen($cadena);$i++)
    {
        echo "<tr align='center'><td>".$cadena[$i];
        echo "</td><td>".$i."</td></tr>";
    }
    echo "</table>";
?>
```

Función *strlen*

Carácter	*Posición*
s	*0*
a	*1*
l	*2*
u	*3*
d	*4*
o	*5*
s	*6*

BÚSQUEDA EN CADENAS

BÚSQUEDA EN CADENAS

- **strstr (cade, buscar)** y **strchr (cade, buscar)**
  - Buscan la cadena 'buscar' dentro de 'cade'
  - Devuelve la subcadena que va desde que se encuentra 'buscar' hasta el final de 'cade'.
  - Si no la encuentra devuelve una cadena vacía.
  - Diferencia entre mayúsculas y minúsculas.

```php
<?php
    $cadena = "En un lugar de la mancha...";
    $buscar = 'lu';

    $encuentra = strstr($cadena, $buscar);
    if($encuentra != '')
    {
        echo "Cadena encontrada";
    }
    else
    {
        echo "Cadena no encontrada";
    }
?>
```

Cadena encontrada

BÚSQUEDA EN CADENAS

- **strrchr(cadena, carBusc)**
  - Busca la **última** aparición de un carácter dentro de una cadena.
  - Aunque en 'carBusc' haya una cadena, sólo se tendrá en cuenta el primer carácter de esta cadena.
  - Devuelve la subcadena que hay entre la última aparición del carácter hasta el final de la cadena.
  - Si no lo encuentra, devuelve una cadena vacía.
  - Diferencia entre mayúsculas y minúsculas

```php
<h2> Funcion <i>strchr</i> y <i>strrchr</i> </h2>
<?php
    $cadena = "saludos para todos...";
    $car1 = "do";
    $car2 = "pc";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena</td>";
    echo "<td>$cadena</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strchr(cadena,'$car1')</td>";
    echo "<td>".strchr($cadena, $car1)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strrchr(cadena,'$car1')</td>";
    echo "<td>".strrchr($cadena, $car1)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strchr(cadena,'$car2')</td>";
    echo "<td>".strchr($cadena, $car2)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strchr(cadena,'$car2')</td>";
    echo "<td>".strrchr($cadena, $car2)."</td></tr>";
        echo "</table>";
?>
```

Funcion *strchr* y *strrchr*

cadena	saludos para todos...
strchr(cadena,'do')	dos para todos...
strrchr(cadena,'do')	dos...
strchr(cadena,'pc')	
strchr(cadena,'pc')	para todos...

BÚSQUEDA EN CADENAS

- **stristr (cadena, cadenaBuscar)**
  - Igual que strstr() pero no diferencia entre mayúsculas y minúsculas.
- **strpos (cade1, cade2 [,despl])**
  - Busca la primera 'cade2' dentro de 'cade1' desde la posición que se indique.
  - Si no se indica posición será desde el principio.
  - Devuelve la **posición** de la primera ocurrencia de 'cade2'
  - Diferencia entre mayúsculas y minúsculas.

BÚSQUEDA EN CADENAS

- **strrpos (cadena, caracter [,despl])**
  - Busca **posición** de la última aparición del carácter en la cadena.
  - Si no lo encuentra, devuelve false.
  - Diferencia entre mayúsculas y minúsculas.

```php
<h2> Funcion <i>strpos</i> y <i>strrpos</i> </h2>
<?php
    $cadena = "saludos para todos...";
    $car = "do";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena</td>";
    echo "<td>$cadena</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strpos(cadena,'$car')</td>";
    echo "<td>".strpos($cadena, $car)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strrpos(cadena,'$car')</td>";
    echo "<td>".strrpos($cadena, $car)."</td></tr>";
        echo "</table>";
?>
```

Funcion *strpos* y *strrpos*

cadena	saludos para todos...
strpos(cadena,'do')	4
strrpos(cadena,'do')	15

BÚSQUEDA EN CADENAS

- **strpbrk (cadena, lista_car):**
  - Busca cualquier ocurrencia de la lista de caracteres pasada como parámetro dentro de la cadena.
  - Devuelve una cadena que va desde el carácter encontrado o false si no lo encuentra.
  - Diferencia entre mayúsculas y minúsculas.

```php
<h2> Función strpbrk</i> </h2>
<?php
    $cadena = "Cadena para buscar caracteres";
    $buscar1 = "khfj";
    $buscar2 = "hxzp";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena</td>";
    echo "<td>$cadena</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strpbrk(cadena,$buscar1)</td>";
    echo "<td>".strpbrk($cadena, $buscar1)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strpbrk(cadena,$buscar2)</td>";
    echo "<td>".strpbrk($cadena, $buscar2)."</td></tr>";
        echo "</table>";
?>
```

Función strpbrk

cadena	Cadena para buscar caracteres
strpbrk(cadena,khfj)	
strpbrk(cadena,hxzp)	para buscar caracteres

EXPRESIONES REGULARES

EXPRESIONES REGULARES

- Las expresiones regulares son una potente herramienta que nos permite contrastar un texto con un **patrón de búsqueda**.

- Esta tarea resulta fundamental en algunos programas, y en otros puede facilitarnos increíblemente el trabajo.

EXPRESIONES REGULARES

- Una expresión regular, consiste en **comparar un patrón frente a un texto**, para comprobar si el texto contiene lo especificado en el patrón.
- Ejemplo:
  - Patrón: **in**
  - Coindicen:
    - **in**tensidad
    - c**in**ta
    - **in**terior
  - Patrón: **[mp]adre**
  - Coindicen:
    - Mi **madre** se llama Luisa
    - Tu **padre** es jardinero

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **El punto**
    - El punto representa **cualquier carácter**. Escribiendo un punto en un patrón querrás decir que ahí hay un carácter, cualquiera. Desde la A a la Z (en minúscula y mayúscula), del 0 al 9, o algún otro símbolo.
    - **Ejemplos:**
      ca.a coincide con *ca**n**a*, *ca**m**a*, *ca**s**a*, *ca**j**a*, etc…
      No coincide con *ca**st**a* ni *caa*

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Principio y fin de cadena**
    - Si queremos indicar al patrón qué es el principio de la cadena o qué es el final, debemos hacerlo con **^ para inicio y $ para final**.
  - **Ejemplos:**
    - "^*olivas*" coincide con "***olivas*** *verdes*", pero no con "*quiero olivas*"
    - "olivas$" sin embargo, va a coincidir con "quiero **olivas**" pero no con "olivas verdes"

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Cuantificadores**
    - Indicar que cierto elemento del patrón va a repetirse un **número indeterminado de veces**,
    - **Usando + =>** el elemento anterior aparece una o más veces.
    - **Usando * =>** el elemento anterior aparece cero o más veces.
    - **Usando ? =>** el elemento anterior aparece una o ninguna vez.
  - **Ejemplos:**

    "gafas+" coincide con "gafassss" pero no con "gafa"

    "gafas*" coincide con "gafassss" y también con "gafa"

    "gafas?" coincide con "gafa" y " gafas " pero no con "gafassss"

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - Para definir la **cantidad de veces que se repetirá el elemento**: { }
    - "*abc{4}*" coincide con "*abcccc*", pero no con "*abc*" ni "*abcc*",
    - "*abc{1,3}*" coincide con "*abc*", "*abcc*", "*abccc*", pero no con "*abcccc*"
  - Si un parámetro queda vacío, significa "un **número indeterminado**". Por ejemplo: "x*{5,}*" significa que la x ha de repetirse 5 veces, o más.

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Rangos**
    - Los corchetes [] permiten especificar el **rango de caracteres**.
    - Basta que exista *cualquiera de ellos* para que se de la condición.
  - **Ejemplos:**
    - "*c[ao]sa*" coincide con "*casa*" y con "*cosa*"
    - "*[a-f]*" coincide con todos los caracteres alfabéticos de la "*a*" a la "*f*"
    - "*[0-9][2-6][ANR]*" coincide con "*12A*", "*35N*", "*84R*",
  - Dentro de los corchetes,
    - el símbolo ^ ya no significa inicio, si no que es un negador.

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Alternancia**
    - Para alternar entre varias opciones, usaremos el símbolo |
    - Con este mecanismo haremos un disyuntor, que nos permitirá dar varias opciones.
    - Si una de ellas coincide, el patrón será cierto.
    - **Ejemplos**:
      - "*aleman(ia|es)*" coincide con "*aleman**ia***" y con "*aleman**es***"
      - "*(norte|sur|este|oeste)*" coincide con cualquiera de los puntos cardinales.

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Agrupadores**
    - Los paréntesis nos sirven para agrupar un subconjunto.
    - Es útil para definir la alternancia, pero agrupar un subpatrón nos permite trabajar con él como si fuera un único elemento.
  - **Ejemplos:**

"*(abc)+*" coincide con "*abc*", "*abcabc*", "*abcabcabc*", etc

"*ca(sca)?da*" coincide con "*cascada*" y con "*cada*"

EXPRESIONES REGULARES

- **Sintaxis y metacaracteres**
  - **Escapar caracteres**
    - Si queremos que en el patrón hubiese un punto, o un símbolo asterisco, sin que se interprete como metacarácter, tendremos que "escaparlo".
    - Esto se hace poniendo una barra invertida justo antes: \. o *
    - Esto puede hacerse con cualquier carácter que quieras introducir de **forma literal**, y no interpretada.

RESUMEN METACARACTERES

Carácter	Descripción	Ejemplo	Correcto	Incorrecto
. (punto)	Cualquier carácter obligatorio	pa.o	pato, palo, paco	pablo, pao
^	Inicio de cadena	^mio	miope, miocardio	simio, rumio
$	Final de cadena	mio$	simio, bohemio	miope, miopia
+	1 o más repeticiones	Mis+	Mis, Miss, Missss	Mi
*	0 o más repeticiones	Mis*	Mi, Miss, Misss	Mio
?	0 o 1 vez	Mis?	Mi, Mis	Miss, Misss
{}	Cantidad de veces	Peca{3} Peca{1,3} Peca{2, }	Pecaaa Peca, Pecaa, Pecaaa Pecaa, Pecaaa, Pecaaaa…	
[]	Rangos de valores	Met[ao] Met[a-z]	Meta, Meto Meta, Meti, Metf	
()	Agrupador	Repi(to)*	Repi, Repito, Repitoto	
|	Alternador	Rep(etir | ito)	Repetir, Repito	

BÚSQUEDA CON EXPRESIONES REGULARES

BÚSQUEDA CON EXPRESIONES REGULARES

- **preg_match(patrón, cadena)**
  - Chequea el patrón en una cadena alfanumérica.
  - Devuelve true si coincide
  - False en caso contrario.
  - Necesita que el patrón empiece y termine por un carácter en concreto.
  - Puede ser cualquiera, pero se aconseja “ ` “

EJEMPLOS

- Comprobar si una cadena contiene alguna “r”

```php
<?php
    $cad = "riesgo";
    if (preg_match("`r`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

- Comprobar si una cadena contiene algún número

```php
<?php
    $cad = "moto";
    if (preg_match("`[1-9]`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

EJEMPLOS

- Comprobar si una cadena tiene dos números

```php
<?php
    $cad = "riesgo45";
    if (preg_match("`[0-9]{2}`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

- Comprobar si una cadena empieza por dos números

```php
<?php
    $cad = "riesgo45";
    if (preg_match("`^[0-9]{2}`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

EJEMPLOS

- Comprobar si una cadena termina por S

```php
<?php
    $cad = "riesgo45S";
    if (preg_match("`S$`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

- Comprobar que una cadena tenga un número después una letra minúscula y después un número

```php
<?php
    $cad = "riesgo45S";
    if (preg_match("`[0-9][a-z][0-9]`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

EJEMPLOS

- Comprobar que una cadena tenga un número, después una sola letra mayúscula o minúscula y después otro número

```php
<?php
    $cad = "riesgo45S7";
    if (preg_match("`[0-9]([a-z]|[A-Z])[0-9]`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

- Comprobar que una cadena tenga un número después dos o más letras minúsculas y después la A

```php
<?php
    $cad = "a3sdeA34";
    if (preg_match("`[0-9][a-z]{2,}A`", $cad))
        echo "Sí cumple el patrón";
    else
        echo "No cumple el patrón";
?>
```

COMPARACIÓN DE CADENAS

COMPARACIÓN DE CADENAS

- **strcmp (cad1, cad2):**
  - < cero, si cad1 es menor que cad2.
  - > cero, si cad1 es mayor que cad2.
  - Cero, si ambas son iguales.
  - Distingue entre mayúsculas y minúsculas.
- **strcasecmp (cad1, cad2):**
  - Igual que la anterior, pero no diferencia entre mayúsculas y minúsculas.

```php
<h2> Funcion <i>strcmp</i> y <i>strcasecmp</i> </h2>
<?php
    $cad1 = "Saludos";
    $cad2 = "saludos";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena 1</td>";
    echo "<td>$cad1</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena 2</td>";
    echo "<td>$cad2</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strcmp(cad1, cad2)";
    echo "</td><td>".strcmp($cad1, $cad2)."</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>strcasecmp(cad1, cad2)";
    echo "</td><td>".strcasecmp($cad1, $cad2)."</td></tr>";
        echo "</table>";
?>
```

Funcion *strcmp* y *strcasecmp*

cadena 1	Saludos
cadena 2	saludos
strcmp(cad1, cad2)	-1
strcasecmp(cad1, cad2)	0

COMPARACIÓN DE CADENAS

- **strncmp (cad1, cad2, num):**
  - Igual que strcmp() pero sólo compara los 'num' primeros caracteres de cada cadena.

```php
$cadena1 = "pais";
$cadena2 = "paisaje";

$num1 = strncmp($cadena1, $cadena2, 4);
$num2 = strncmp($cadena1, $cadena2, 5);
```

caracteres comparados	Resultado
4	0
5	-1

OPERAR CON SUBCADENAS

OPERAR CON SUBCADENAS

- **substr (cad, inicio [,tam])**
  - Devuelve la subcadena que va desde 'inicio' hasta el fin, o si se indica, el número de caracteres indicados.
- **substr_replace(cad1, cad2, inicio [,tam])**
  - Devuelve una cadena que es el resultado de sustituir parte cad1 por cad2.
  - La cadena original no sufre modificación.

OPERAR CON SUBCADENAS

- **str_replace(cadbus, cadree, cadena)**
  - Devuelve una cadena en la que todas las apariciones de 'cadbus' se cambian por 'cadree' en cadena.
  - La cadena original no sufre cambios.

```php
$texto = "Me gusta este pais";
$cadena1 = "pais";
$cadena2 = "paisaje";

$nueva_cadena = str_replace($cadena1, $cadena2, $texto);

echo "Original: $texto<br>";
echo "Modificada: $nueva_cadena";
```

Original: Me gusta este pais
Modificada: Me gusta este paisaje

OPERAR CON SUBCADENAS

- **strtr (cadena, cadbus, cadree)**
  - Igual que str_replace, pero se cambian cada uno de los caracteres de la cadena buscada, por su correspondiente en la cadena de sustitución.

```
<h2> Funcion <i>strtr</i> </h2>
<?php
    $cadena = "abcdefghijkl";
    $cadB = "aei";
    $cadS = "AEI";

    $resultado = strtr($cadena, $cadB,
?>
```

Funcion *strtr*

cadena	abcdefghijkl
Patrón	aei
Sustitución	AEI
strtr(cadena,patrón, sustitucion)	AbcdEfghIjkl

OPERAR CON SUBCADENAS

- **substr_count(cadena, buscar)**
  - Devuelve el número de veces que aparece ‘buscar’ en ‘cadena’

MODIFICACIÓN DEL CONTENIDO

LIMPIEZA DE CADENAS

LIMPIEZA DE CADENAS

- **rtrim(cadena)**
  - Devuelve la cadena sin los espacios en blanco y sin los caracteres de fin de línea que haya al final de la cadena.
- **ltrim(cadena)**
  - Devuelve la cadena pero elimina los espacios en blanco al principio de la cadena.
- **trim (cadena)**
  - Devuelve la cadena pero elimina los espacios en blanco del principio y del final de dicha cadena.

MODIFICACIÓN DEL CONTENIDO

RELLENO DE CADENAS

RELLENO DE CADENAS

- **str_pad (cadena, long_total, car)**
  - Rellena una cadena con un carácter de relleno (por defecto espacio en blanco
  - Opcionalmente se puede indicar el modo de relleno:
    - STR_PAD_RIGHT: relleno por la derecha (defecto)
    - STR_PAD_LEFT: relleno por la izquierda.
    - STR_PAD_BOTH: relleno por ambos lados, intenta colocar los mismos caracteres a derecha e izquierda.

```php
<?php
    $texto = "Cadena";

    $nueva_cadenaD = str_pad($texto, 20, 'r', STR_PAD_RIGHT);
    $nueva_cadenaI = str_pad($texto, 20, 'r', STR_PAD_LEFT);
    $nueva_cadenaB = str_pad($texto, 20, 'r', STR_PAD_BOTH);

    echo "Cadena original: $texto<br>";
    echo "Cadena rellena por la derecha: $nueva_cadenaD<br>";
    echo "Cadena rellena por la izquierda: $nueva_cadenaI<br>";
    echo "Cadena rellena por la ambos lados: $nueva_cadenaB<br>";
?>
```

Cadena original: Cadena
Cadena rellena por la derecha: Cadenarrrrrrrrrrrr
Cadena rellena por la izquierda: rrrrrrrrrrrrCadena
Cadena rellena por la ambos lados: rrrrrrCadenarrrrrr

MODIFICACIÓN DEL CONTENIDO

CONVERSIÓN ENTRE MAYÚSCULAS Y MINÚSCULAS

CONVERSIÓN MAYÚSCULAS MINÚSCULAS

- **strtolower(cadena)**
  - Convierte una cadena de caracteres a minúsculas.
- **strtoupper(cadena)**
  - Convierte una cadena de caracteres a mayúsculas.
- **ucfirst(cadena)**
  - Convierte a mayúsculas el primer carácter de una cadena.
- **ucwords(cadena)**
  - Convierte a mayúsculas el primer carácter de cada palabra de la cadena.

MODIFICACIÓN DEL CONTENIDO

DIVISIÓN DE CADENAS

DIVISIÓN DE CADENAS

- **strtok (cadena, divisor)**
  - Divide una cadena en subcadenas.
  - Utiliza "divisor" como carácter de división.
  - La primera vez que se llama a la función, devuelve el primer trozo obtenido.
  - Las siguientes veces que se llama a la función **NO** hay que pasar de nuevo la cadena a dividir.

```php
<?php
    $cad="probando esto de dividir en trozos una cadena";
    $patron=" ";

    $trozos = strtok($cad, $patron);
    while($trozos!=false)
    {
        echo $trozos;
        echo "<br>";
        $trozos=strtok($patron);
    }

?>
```

cadena	probando esto de dividir en trozos una cadena
strtok(cad,$patron)	
subcadena	probando
subcadena	esto
subcadena	de
subcadena	dividir
subcadena	en
subcadena	trozos
subcadena	una
subcadena	cadena

DIVISIÓN DE CADENAS

- **explode(separador, cadena, límite)**
  - Divide una cadena en porciones de menor tamaño.
  - Carácter separador
  - Cadena original
  - Número de trozos que quiere
    - Valor positivo: número de trozos que se crearán. En el último siempre irá el resto de la cadena
    - Valor negativo: NO se incluirán los X últimos trozos
    - Valor = 0: no se modificará la cadena

cadena	Esta es mi cadena de prueba
explode(' ', $cadena, 8))	
Trozo 0	Esta
Trozo 1	es
Trozo 2	mi
Trozo 3	cadena
Trozo 4	de
Trozo 5	prueba

cadena	Esta es mi cadena de prueba
explode(' ', $cadena, -2))	
Trozo 0	Esta
Trozo 1	es
Trozo 2	mi
Trozo 3	cadena

cadena	Esta es mi cadena de prueba
explode(' ', $cadena, 3))	
Trozo 0	Esta
Trozo 1	es
Trozo 2	mi cadena de prueba

cadena	Esta es mi cadena de prueba
explode(' ', $cadena, -4))	
Trozo 0	Esta
Trozo 1	es

cadena	Esta es mi cadena de prueba
explode(' ', $cadena, 0))	
Trozo 0	Esta es mi cadena de prueba

DIVISIÓN DE CADENAS

- **implode (nexo, array)**
  - Devuelve una cadena resultado de unir todos los elementos de un array. Utiliza 'nexo' como unión.

FUNCIONES RELACIONADAS CON HTML

HTML

- Gran parte de las cadenas con las que trabaja PHP tienen como función convertirse en parte de un documento HTML que será enviado al usuario.
- PHP incluye algunas funciones que evitan los problemas de transformación de texto en código PHP.

HTML

- **htmlspecialchars(cadena)**
  - Convierte los caracteres con un significado especial en su traducción en HTML

Carácter	Traducción HTML
&	&amp;
“	&quot;
<	&lt;
>	&gt;

OTRAS FUNCIONES CON CADENAS

OTRAS FUNCIONES

- **chr(entero)**
  - Recibe un codigo ASCII y devuelve el carácter asociado a dicho código.
- **count_chars(cadena [,modo])**
  - Cuenta el numero de apariciones de cada carácter en la cadena.
  - El modo en que lo devuelve depende del segundo parámetro:

0	**Matriz asociativa**
1	Sólo los caracteres que aparecen más de 0 veces.
2	Sólo los caracteres que no aparecen.
3	Cadena con caracteres que hay
4	Cadena con caracteres que no hay

OTRAS FUNCIONES

- **strrev(cadena)**
  - Devuelve la cadena invertida.

- **str_repeat(cadena, veces)**
  - Devuelve una cadena con la cadena que se le pasa repetida tantas veces como se pase en el 2º parámetro.

MANEJO DE CADENAS

PROGRAMANDO CON PHP