Manejo de Cadenas

Programando con PHP

Introducción

- El tratamiento de cadenas es muy importante en PHP.
- El control de usuarios y contraseñas está muy ligado al tratamiento de cadenas.
- Existe un amplio conjunto de funciones para el manejo de cadenas.
  - Veremos las principales.

Delimitadores

Delimitadores de cadenas

- Una cadena está formada por cero o más caracteres encerrados entre comillas.
  - Pueden ser Dobles o Simples.
- Es obligatorio utilizar siempre el mismo tipo de comilla para una cadena.
- Se pueden entremezclar ambos, teniendo cuidado.

Delimitadores de cadenas

- Con comillas dobles: podemos incluir variables o caracteres especiales que **serán evaluados**.
- Esto no ocurre con las comillas simples.

Visualización de cadenas

Funciones de cadenas

- echo(cadena)
  - Muestra información por la salida estándar.
  - Sólo admite una única cadena como argumento.

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.
- 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	0
a	1
l	2
u	3
d	4
o	5
s	6

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

Funcion *strlen*

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

Búsqueda en cadenas

Búsqueda en cadenas

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

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 es una secuencia de caracteres que sirven de **patrón para comparar frente a un texto.**
- Se busca siempre comprobar si el texto contiene lo especificado en el patrón.
- Ejemplo:
  - Patrón: **in**
  - Coinciden:
    - **in**tensidad
    - c**in**ta
    - **in**terior
  - Patrón: [**mp**]**adre**
  - Coinciden:
    - 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*"

Expresiones regulares

- **Sintaxis y metacaracteres**
  - **Cuantificadores**
    - Indicar que cierto elemento del patrón va a repetirse un **número indeterminado de veces**.
    - Los cuantificadores son: + y * .
    - Usando + queremos decir que el elemento anterior aparece una o más veces.
    - Usando * queremos decir que el elemento anterior aparece cero o más veces.
  - **Ejemplos:**
    "*gafas+*" coincide con "*gafa**ssss***" pero no con "*gafa*"

Expresiones regulares

- **Sintaxis y metacaracteres**
  - **Puede que esté (una vez) o puede que no**: ?
    - "*coches?*" coincide con "*coche*" y con "*coches*"
  - 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 aparezca *cualquiera de ellos*.
  - **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.

Búsqueda con expresiones regulares

Búsqueda con expresiones regulares

- preg_match(patrón, cadena)
  - Comprueba si una cadena cumple con un patrón dado.
  - El patrón debe ser una expresión regular.
  - El patrón será una cadena que empezará y terminará con barra (/)

Ejemplos

- Comprobar si una cadena contiene alguna “r”

```php
<?php
    $cad="riesgo";
        if (preg_match("/r/", $cad)) echo "SI";
        else echo "NO";
?>
```

- Comprobar si una cadena contiene algún número

```php
<?php
    $cad="moto";
        if (preg_match("/[1-9]/", $cad)) echo "SI";
        else echo "NO";
?>
```

Ejemplos

- Comprobar si una cadena tiene dos números

```php
<?php
    $cad="riesgo33";
        if (preg_match("/[0-9]{2}/", $cad)) echo "SI";
        else echo "NO";
?>
```

- Comprobar si una cadena empieza por dos números

```php
<?php
    $cad="riesgo33";
        if (preg_match("/^[0-9]{2}/", $cad)) echo "SI";
        else echo "NO";
?>
```

Ejemplos

- Comprobar si una cadena termina por S

```php
<?php
    $cad="riesgo33S";
        if (preg_match("/S$/", $cad)) echo "SI";
        else echo "NO";
?>
```

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

```php
<?php
    $cad="riesgo3T3S";
        if (preg_match("/[0-9][a-z][0-9]/", $cad)) echo "SI";
        else echo "NO";
?>
```

Ejemplos

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

```php
<?php
    $cad="riesgo3T3S";
        if (preg_match("/[0-9]([a-z]|[A-Z])[0-9]/", $cad)) echo "SI";
        else echo "NO";
?>
```

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

```php
<?php
    $cad="e3seA3S";
        if (preg_match("/[0-9][a-z]{2,}A/", $cad)) echo "SI";
        else echo "NO";
?>
```

Comparación de cadenas

Comparación de cadenas

- strcmp (cad1, cad2):
  - Compara ambas cadenas y devuelve
    - Valor < cero, si cad1 es va antes que cad2.
    - Valor > cero, si cad1 es va después 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
<?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.

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 dentro de cad1 el trozo que va desde 'inicio' hasta el final (o el número de caracteres) 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.
- 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.

```php
<?php
    $cadena = "abcdefghijkl";
    $cadB = "aei";
    $cadS = "AEI";
    echo "<table border='1' cellpadding='2' cellspacing='2'>";
    echo "<tr align='center'><td bgcolor = 'pink'>cadena</td>";
    echo "<td>$cadena</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>Patrón</td>";
    echo "<td>$cadB</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>Sustitución</td>";
    echo "<td>$cadS</td></tr>";
    echo "<tr align='center'><td bgcolor = 'pink'>";
    echo "strtr(cadena,patrón, sustitucion)";
    echo "</td><td>".strtr($cadena, $cadB, $cadS)."</td></tr>";
        echo "</table>";
?>
```

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

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

Relleno de cadenas

- str_pad (cadena, long, car, modo)
  - Rellena una cadena con un carácter de relleno (por defecto espacio en blanco) hasta que la cadena tenga la longitud deseada.
  - 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.

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.

Otras funciones con cadenas

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