

ÉRICA PALOMINO

TEMA 5

# MANEJO DE CADENAS

**[erica.palomino@escuelaartegranada.com](mailto:erica.palomino@escuelaartegranada.com)**

ESCUELAARTEGRANADA

# 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).

Por otro lado, dependiendo de los inputs necesitaremos manejar estas cadenas.



# **ACCESO A CADENAS**

# ACCESO AL CONTENIDO

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.

Con **strlen (cadena)**, se devuelve la longitud de la cadena.

## ACCESO A CADENAS - EJERCICIOS

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.



# **BÚSQUEDA EN CADENAS**

# BÚSQUEDA EN CADENAS

Con **strstr (cadena, buscar)** y **strchr (cadena, buscar)**, podemos buscar la subcadena 'buscar' dentro de 'cadena'.

En caso de encontrarla, devuelve desde la coincidencia hasta el final de 'cadena'. En caso contrario devuelve una cadena vacía.

# BÚSQUEDA EN CADENAS

Con **strrchr (cadena, char)**, podemos buscar **última** aparición de un carácter dentro de una cadena.

Aunque en char haya una cadena, sólo se tendrá en cuenta el primer carácter de esta cadena.

En caso de encontrarla, devuelve desde la coincidencia hasta el final de 'cadena'. En caso contrario devuelve una cadena vacía.



# BÚSQUEDA EN CADENAS

**stristr (cadena, cadenaBuscar)** cumple la misma función que strstr() pero no es Case Sensitive.

**strpos (cadena, cadena2 [,despl])** busca la **posición** de la primera aparición de la cadena2 dentro de cadena1.

**strrpos (cadena, caracter [,despl])** cumple la misma función que strrpos(), pero con caracteres. Devuelve la posición de la última aparición.



# **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.

Ejemplos:

- Patron: **pre**

- Coinciden:

- im**pre**sionante
- **pre**sa
- neo**pre**no

- Patron: **[mp]esa**

- Coinciden:

- **me**sa
- **pe**sa

# EXPRESIONES REGULARES

## 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: "^ropa" coincide con "**ropa** vieja" pero no con "comprar **ropa**". "ropa\$" coincide con "comprar **ropa**" pero no con "**ropa** vieja".

# EXPRESIONES REGULARES

## Cuantificadores

**{}** funciona como el resto de cuantificadores pero indicando un número determinado de veces que se repite.

- **"abc{4}"** indica que la cadena ha de ser "abcccc" para que coincida.
- **"abc{1,3}"** contempla entre 1 y 3 repeticiones de **c**.

# EXPRESIONES REGULARES

## Cuantificadores

**{}** funciona como el resto de cuantificadores pero indicando un número determinado de veces que se repite.

- **"abc{4}"** indica que la cadena ha de ser "abcccc" para que coincida.
- **"abc{1,3}"** contempla entre 1 y 3 repeticiones de **c**.

# EXPRESIONES REGULARES

## Rangos

Los corchetes `[]` permiten especificar el rango de caracteres. Basta que exista cualquiera de ellos para que se de la condición. En este caso `^` se usa como negador.

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"`,

# EXPRESIONES REGULARES

## 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.

Ejemplo: "aleman(ia|es)" coincide con "aleman**ia**" y con "aleman**es**"



# EXPRESIONES REGULARES

## 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.

Ejemplo: "(abc)+" coincide con "abc", "abcab", "abcabcab", etc


# EXPRESIONES REGULARES

## Usar caracteres especiales

Si queremos que en el patrón hubiese un punto, o un símbolo asterisco, sin que se interprete como metacarácter, tendremos que “escaparlo”. Esto se consigue utilizando **\** **seguido del carácter.**

# EXPRESIONES REGULARES - TABLA RESÚMEN

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 EXPRESIONES REGULARES

**preg\_match(patrón, cadena):** Chequea el patrón en una cadena alfanumérica. Devuelve true si coincide y false en caso contrario.

Necesita que el patrón empiece y termine por un carácter en concreto, éste puede ser cualquiera, pero se aconseja  
" ^ "

# EJEMPLOS

- Comprobar si una cadena contiene alguna 'r'

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

- Comprobar si una cadena tiene algún número

```
<?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 contiene 2 números

```
<?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 comienza por 2 números

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

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

```
<?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
$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 de dos o más letras minúsculas y después la A

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

# EJERCICIOS

- Haz un script en PHP que compruebe si las cadenas almacenadas en un array empiecen por mayúscula, contengan 2 o más letras minúsculas y acaben por 3 números.
- Haz un programa en PHP que compruebe si una cadena de caracteres contiene al menos una letra mayúscula, una minúscula, un número y un carácter especial. Dichos caracteres pueden estar ordenados de cualquier forma.



# **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.

**substr\_replace(cad1, cad2, inicio [,tam])** Devuelve una cadena que es el resultado de sustituir parte de 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.

**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.

**substr\_count(cadena, buscar)** Devuelve el número de veces que aparece 'buscar' en 'cadena'.

# EJERCICIOS

- Implementa un script en PHP en el que, dado un array de cadenas de caracteres, se reemplace una subcadena por otra indicada y se muestre, en una tabla; la cadena original, la cadena nueva y el número de veces que se ha hecho la sustitución, de cada una de las posiciones del array.



# **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.





# **RELLENO DE CADENAS**

# RELLENO DE CADENAS

**str\_pad (cadena, long\_total, car, opc)** 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.



# **CONVERSIÓN MAYÚSCULAS- 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.



# **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.

# DIVISIÓN DE CADENAS

**explode(separador, cadena, límite).** Divide una cadena en porciones de menor tamaño utilizando el carácter separador.

***límite*** indica el número de trozos en el que se dividirá la cadena.

- **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

# DIVISIÓN DE CADENAS

**implode (nexo, array).** Devuelve una cadena resultado de unir todos los elementos de un array. Utiliza 'nexo' como unión (este carácter o cadena se incluirá entre las cadenas del array) .





# **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.

**htmlspecialchars(cadena)** Convierte los caracteres con un significado especial en su traducción en HTML. Ej: &, ", <, >



# **OTRAS FUNCIONES**

## OTRAS FUNCIONES

**chr(entero).** Recibe un código ASCII y devuelve el carácter asociado a dicho código.

**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.