EXPRESIONES REGULARES

PHP

Expresiones regulares

- Una **expresión regular** es un **patrón** que se compara con una **cadena objetivo** de izquierda a derecha, carácter a carácter.
- Las expresiones regulares permiten definir patrones de coincidencia y aplicarlas a cadenas de texto, para saber si la cadena (o parte de ella) cumple ese patrón.
- Las **expresiones regulares** van encerradas en **delimitadores**. Los delimitadores más utilizados son las **barras oblícuas** (**/**)

preg_match ((exp_regular , cadena [,$matriz_coincidencias [, PREG_OFFSET_CAPTURE [, $comienzo]]])

- Función que realiza una comparación con una expresión regular.
- Esta función **para la búsqueda con la primera coincidencia** encontrada, la cual se puede guardar en el argumento opcional coincidencias, que tiene forma de matriz.
- Devuelve **1 si el patrón y la cadena coincide**, y 0 si no coinciden, o FALSE si ocurrió un error.
- Los patrones deben empezar y acabar con el delimitador / (barra).

preg_match ((exp_regular , cadena [,$matriz_coincidencias [, PREG_OFFSET_CAPTURE [, $comienzo]]])

- Por defecto, las búsquedas son **sensibles a mayúsculas y minúsculas**, para cambiar esto se puede añadir una **i** de **insensitive** al final. Esto es lo que se llama un **modificador de patrón**.
- Para marcar **el inicio o el fin del string** se debe usar **^ y $**

```
</head>
<body>
<?php
$cadena1 = "1234567";
$cadena2 = "abcdefg";
$patron = "/^[[:digit:]]+$/";

if (preg_match($patron, $cadena1)) {
    print "<p>La cadena $cadena1 son sólo números.</p>\n";
} else {
    print "<p>La cadena $cadena1 no son sólo números.</p>\n";
}

if (preg_match($patron, $cadena2)) {
    print "<p>La cadena $cadena2 son sólo números.</p>\n";
} else {
    print "<p>La cadena $cadena2 no son sólo números.</p>\n";
}
?>
</table>
</form>
```

La cadena 1234567 son sólo números.

La cadena abcdefg no son sólo números.

DESARROLLO WEB EN ENTORNO SERVIDOR CON PHP

preg_match (exp_regular , cadena [,$matriz_coincidencias [, PREG_OFFSET_CAPTURE [, $comienzo]]])

- Las coincidencias encontradas se pueden guardar en el argumento $matriz_coincidencias y, si se añade el modificador PREG_OFFSET_CAPTURE, se guardan también en el argumento $matriz_coincidencias la posición de cada coincidencia encontrada.
- El argumento $comienzo debe ser un número entero que indica en qué carácter de la cadena se inicia la búsqueda, por defecto será en la primera posición.

```php
<body>
<?php
$cadena = "aaAA";
$patron1 = "/^[a-z]+$/";
$patron2 = "/^[a-z]+$/i";

if (preg_match($patron1, $cadena)) {
    print "<p>La cadena $cadena son sólo letras minúsculas.</p>\n";
} else {
    print "<p>La cadena $cadena no son sólo letras minúsculas.</p>\n";
}

if (preg_match($patron2, $cadena)) {
    print "<p>La cadena $cadena son sólo letras minúsculas o mayúsculas.</p>\n";
} else {
    print "<p>La cadena $cadena no son sólo letras minúsculas o mayúsculas.</p>\n";
}
?>
```

La cadena aaAA no son sólo letras minúsculas.

La cadena aaAA son sólo letras minúsculas o mayúsculas.

DESARROLLO WEB EN ENTORNO SERVIDOR CON PHP

preg_match_all ((exp_regular , cadena [,$matriz_coincidencias [, PREG_OFFSET_CAPTURE [, $comienzo]]])

- Realiza una comparación global de una expresión regular.
- Después haber encontrado la primera coincidencia, las búsquedas subsiguientes continuarán desde el final de la dicha coincidencia.
- Devuelve **el número de coincidencias** del patrón completo (el cual puede ser cero), o FALSE si se produjo un error.

```php
<?php
    $nro=preg_match_all("/de/",
    "<b>ejemplo: </b><div align=left>esto es una cadena de  prueba de de </div>",
    $salida, PREG_PATTERN_ORDER);
    foreach ($salida[0] as $ind=>$valor)
    {echo "encontrado en la pos $ind el caracter $valor<br>";}
    Print "he encontrado $nro caracteres como el patron<br>";
print "<pre>".print_r($salida)."</pre>";
?>
```

encontrado en la pos 0 el caracter de
encontrado en la pos 1 el caracter de
encontrado en la pos 2 el caracter de
encontrado en la pos 3 el caracter de
he encontrado 4 caracteres como el patron
Array ([0] => Array ([0] => de [1] => de [2] => de [3] => de))

1

Sintaxis de las expresiones regulares o Metacaracteres .

Simbolo	Descripcion
^	Inicio de la expresion regular
$	Final de la expresion regular
.	Cualquier caracter
[xyz] o [abc]	Define un grupo de caracteres permitidos
[a-z]	Se define que se accepta el grupo de letras minusculas que van de la a hasta la z
[A-Z]	Se define que se accepta el grupo de letras mayusculas que van de la a hasta la z
[0-9]	Se define que se accepta el grupo de numeros que van de 0 a 9; si fuese [6-8] acceptaria 6,7 y 8

Sintaxis de las expresiones regulares o Metacaracteres .

[^a-c]	Se define que se acceptan todos los caracteres salvo el rango indicado (en este caso, a, b y c)
*	Indica que el caracter precedente se accepta de 0 a n veces
+	Indica que el caracter precedente se accepta de 1 a n veces. Por ejemplo en el grupo [a-z0.9._-]+, todos estos caracteres se accepta por lo menos una vez.
?	Indica que el caracter precedente se accepta de 0 a 1 vez.
{n}	Indica que el caracter precedente se accepta solo n veces.
{n, }	Indica que el caracter precedente se accepta al menos n veces.
{n,m}	Indica que el caracter precedente se accepta entre n y m veces.
\	Carácter de escape. Si se quiere emplear un símbolo en el sentido proprio, por ejemplo, que el navegador interprete el signo + como una cruz y no como una indicación sobre cómo debe reaccionar el navegador, se puede utilizar el carácter de escape para anular su funcionalidad \+

Sintaxis de las expresiones regulares o Secuencias especiales:

Son conjuntos predefinidos de caracteres que permiten reducir el tamaño de las expresiones regulares

\d	Indica digitos, es similar a [0-9].
\D	Indica salvo los digitos, es similar a [^0-9].
\s	Indica caracteres de espaciado (espacio, tabulador o salto de pagina)
\S	Indica que es un caracter cualquiera (salvo espacio)
\w	Indica cualquier caracter alfanumerico, incluye el guion bajo (_)
\W	Indica cualquier caracter salvo un alfanumerico o el guion bajo (_)

Sintaxis de las expresiones regulares

[[:alnum:]]	cualquier letra o dígito
[[:alpha:]]	cualquier letra
[[:digit:]]	cualquier dígito
[[:lower:]]	cualquier letra minúscula
[[:punct:]]	cualquier marca de puntuación
[[:space:]]	cualquier espacio en blanco
[[:upper:]]	cualquier letra mayúscula

Ejemplos de expresiones regulares

Patrón	Cadena	¿Cumple?	Comentario
abc	awbwc	No	Los caracteres tienen que estar seguidos.
	34abc	Sí	No importa que hayan caracteres antes...
	cbabcba	Sí	... o después.
a2b	g1da2b3	Sí	Las expresiones regulares detectan letras, números,
áb	3áb4	Sí	... incluso acentos, ...
^ab	cab	No	Los caracteres tienen que estar al principio
	abc	Sí	No importa que hayan caracteres después
ab$	abc	No	Los caracteres tienen que estar al final
	cab	Sí	No importa que hayan caracteres antes
^ab$	ab	Sí	Tiene que empezar y acabar por ab ...
	abab	No	... y no puede haber nada antes o después

ab?c	abcde	Sí	El carácter b puede estar entre a y c...
	acde	Sí	... o no estar entre a y c ...
	adcde	No	... pero no puede haber otro carácter
a.c	abc	Sí	El . representa cualquier carácter ...
	a c	Sí	... incluso el espacio el blanco, ...
	ac	No	pero no la ausencia del carácter
	abdc	No	o varios caracteres.
ab+c	abcde	Sí	El carácter b puede estar una vez...
	abbbcde	Sí	... o varias ...
	acde	No	... pero tiene que estar al menos una vez.
ab*c	abcde	Sí	El carácter b puede estar una vez...
	abbbcde	Sí	... o varias ...
	acde	Sí	... o ninguna.
ab{3}c	abbbc	Sí	Las llaves indican el número exacto de repeticiones del carácter,
	abbbbc	No	... no puede haber más ...
	abbc	No	... ni menos.
ab{2,4}c	abc	No	Se pueden definir rangos con límite inferior y superior
	abbc	Sí	
	abbbc	Sí	
	abbbbc	Sí	
	abbbbbc	No	
ab{2,}c	abc	No	Se pueden definir rangos sin límite superior

a(bc){2}d	abcbcd	Sí	Los paréntesis definen agrupaciones de caracteres. En este caso bc tiene que aparecer repetido
a(bc)?d	abcd	Sí	Aquí bc puede estar ...
	ad	Sí	... o no estar, ...
a(bc)?d	abcd	Sí	Aquí bc puede estar ...
	ad	Sí	... o no estar, ...
	abd	No	... pero no puede aparecer sólo la b, o sólo la c u otro carácter
^a(b|d)c$	abc	Sí	Entre la a al principio y la c al final puede estar el carácter b...
	adc	Sí	... o el carácter d, ...
	abdc	No	... pero no los dos, ...
	ac	No	... ni ninguno de ellos.
^(ab)|(dc)$	abc	Sí	Está la pareja ab al principio ...
	adc	Sí	... o dc ...
	abdc	Sí	... o las dos, ...
	ac	No	... pero no ninguna
^(ab)$\|^(dc)$	abc	No	Está la pareja ab, pero sobra la c ...
	adc	No	... o está la pareja dc, pero sobra la a.
	dc	Sí	Está una de las dos