Expresiones regulares

Lenguajes Interpretado en el Servidor

Objetivos

- Tener claridad del concepto de expresiones regulares y su utilidad.
- Adquirir la habilidad de construir expresiones regulares de forma adecuada.
- Emplear las expresiones regulares para realizar operaciones de validación de datos de entrada en formularios.
- Utilizar las expresiones regulares para programas en los que se requiera encontrar coincidencias, realizar sustituciones y reemplazos.

Contenido

- Fundamentos de expresiones regulares.
- Patrones de caracteres.
- Agrupamiento de patrones.
- Metasímbolos.
- Secuencia.
- Cuantificadores.
- Modificadores.

Contenido

- Paréntesis.
- Alternativas.
- Fijación de patrones.
- Escape de caracteres.
- Precedencia.
- Expresiones regulares en cadenas de PHP.
- Formas de trabajar con expresiones regulares en PHP.
- Funciones de expresión regular de PHP.

Fundamentos de expresiones regulares

Expresiones regulares en PHP

- En PHP se puede trabajar con expresiones regulares de dos formas:
  - Usando el estándar POSIX (**P**ortable**O**perating **S**ystem **I**nterface uni**X**).
  - Usando expresiones regulares PCRE (**P**erl **C**ompatible **R**egular **E**xpressions).
- En vista que la primera opción ha sido declarada obsoleta a partir de la versión 5.3.0 de PHP, nos vamos a centrar únicamente en las expresiones regulares compatible con Perl (PCRE).

```
//Address: State code (US)
'/\\b(?:A[KLRZ]|C[AOT]|D[CE]|FL|GA|HI|I[ADLN]|K

//Address: ZIP code (US)
'\b[0-9]{5}(?:-[0-9]{4})?\b'
```

Fundamentos de expresiones regulares

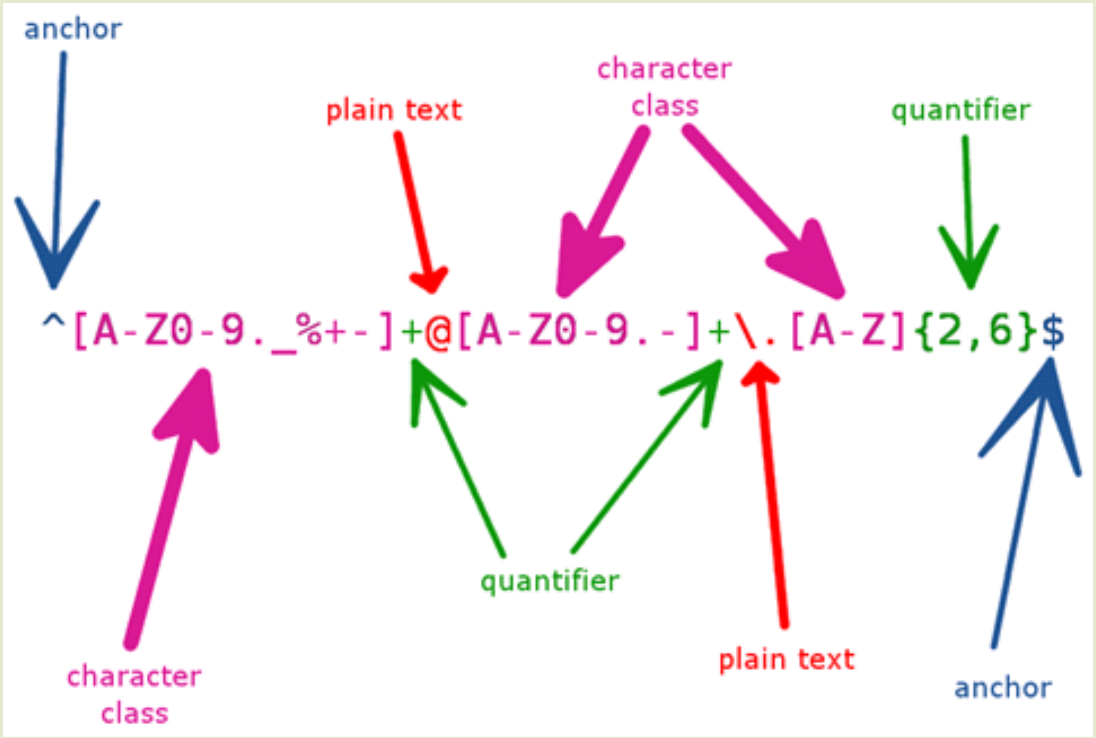
- Podemos definir una expresión regular como una secuencia o patrón de caracteres que describe a un conjunto de cadenas sin enumerar explícitamente sus elementos.
- Las expresiones regulares son definidas con el propósito de confrontarlas contra una cadena y determinar si se ajusta al patrón o modelo que describe a la expresión regular.
- Lo anterior permitirá realizar búsquedas complejas, sustituciones múltiples en textos largos.

Fundamentos de expresiones regulares

- Las expresiones regulares se delimitan con barras inclinadas ('/'). Así, podemos definir una expresión regular simple de la siguiente forma:
  - `$patron = '/http:\/\//';`
- Para confrontarlo contra la siguiente cadena: http://www.udb.edu.sv.

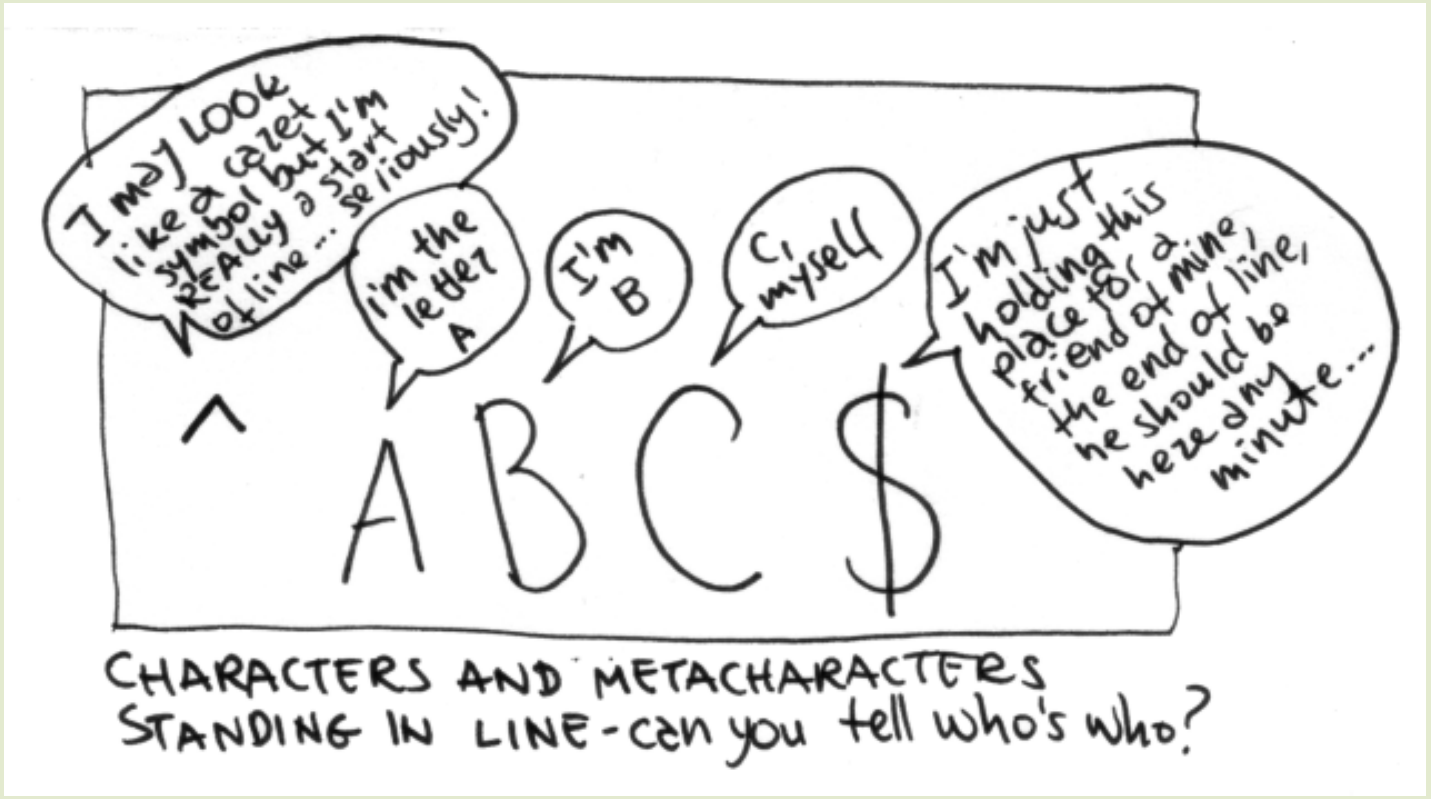
Fundamentos de expresiones regulares

- Una expresión regular puede ser considerada como una cadena formada por dos tipos de caracteres:
  - Caracteres normales.
  - Metacaracteres.



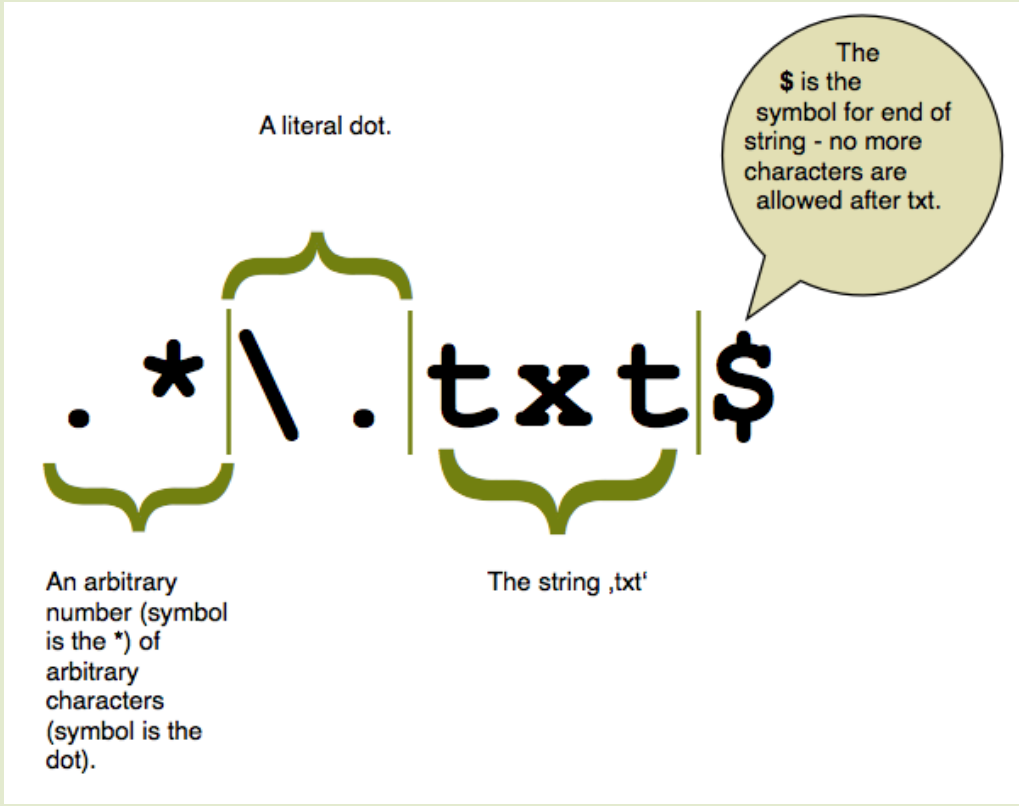
Caracteres normales

- Se les denomina caracteres normales a los caracteres que se representan a sí mismos dentro de la definición de un patrón de expresión regular.



Metacaracteres

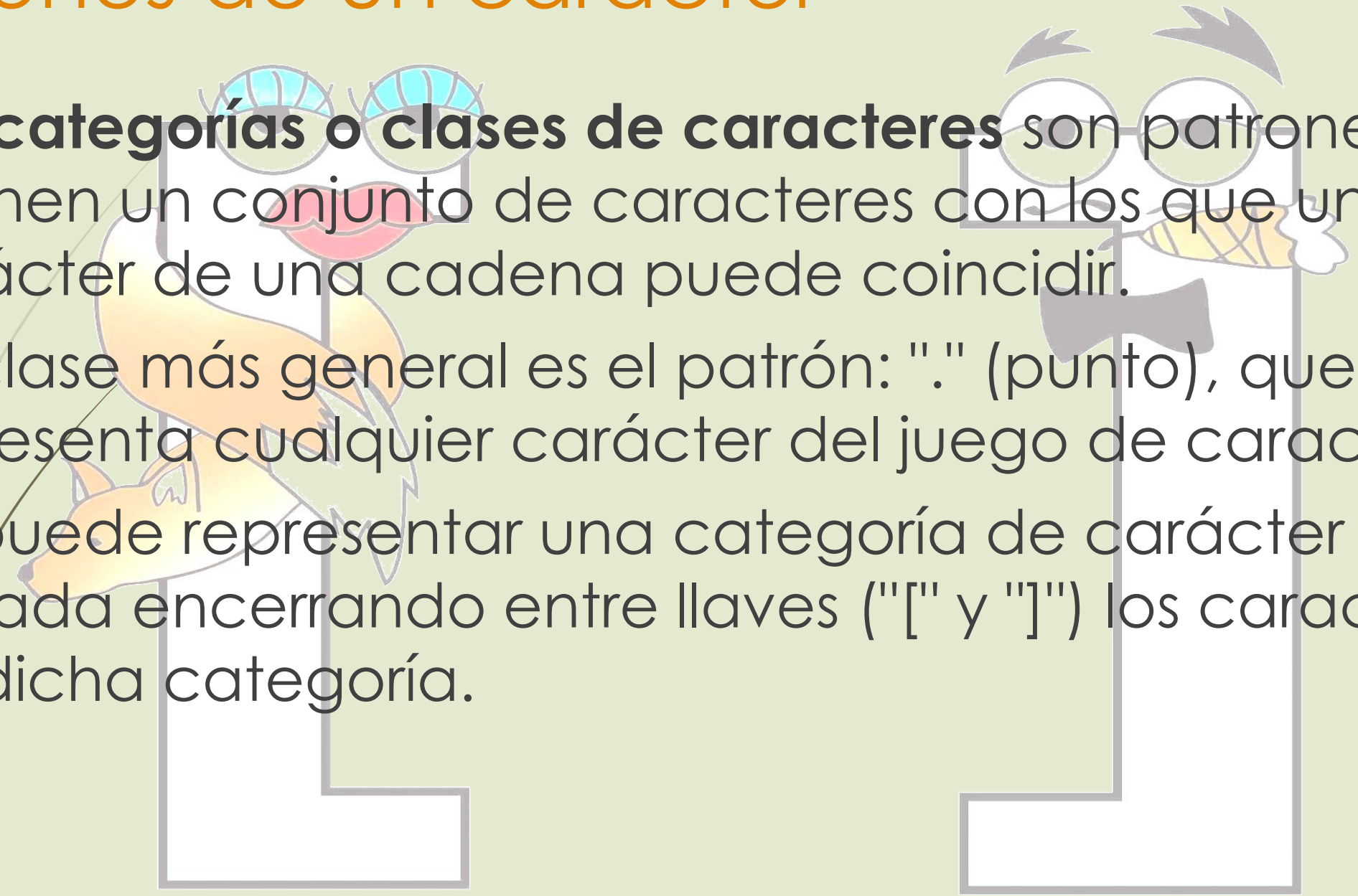
- Se les llama así a los caracteres que tienen un significado especial dentro del patrón que define a la expresión regular.
- Algunos metacaracteres se utilizan para denotar patrones que representan un solo carácter, mientras que otros se utilizan para definir operadores de agrupamiento de patrones que se corresponden con múltiples caracteres.



Patrones de un carácter

- Los patrones de un carácter se pueden dividir en dos tipos:
  - Los caracteres que se representan a sí mismos.
  - Las categorías o clases de caracteres.
- La expresión regular más simple posible es la que está formada por un solo carácter que se representa a sí mismo. Por ejemplo: /o/.
- La anterior expresión regular, describe todas las cadenas que contengan el carácter "o" minúscula.

Patrones de un carácter



- Las **categorías o clases de caracteres** son patrones que definen un conjunto de caracteres con los que un carácter de una cadena puede coincidir.
- La clase más general es el patrón: "." (punto), que representa cualquier carácter del juego de caracteres.
- Se puede representar una categoría de carácter más limitada encerrando entre llaves ("[" y "]") los caracteres de dicha categoría.

Patrones de un carácter

- Para definir una categoría limitada como letras minúsculas del alfabeto inglés, las vocales, las consonantes, los dígitos, se pueden definir las siguientes expresiones regulares:
  - "[abcdefghijklmnopqrstuvwxyz]"
  - "[aeiou]"
  - "[bcdfghjklmnpqrstvwxyz]"
  - "[0123456789]“
- También se puede utilizar una notación con guión, en algunos de los casos anteriores, para resumir:
  - "[a-z]"
  - "[0-9]"

Patrones de un carácter

- Cuando el metacarácter ^ aparece, justo a continuación del carácter de apertura de una clase de carácter ("["), entonces se interpreta como una negación de la lista de caracteres.
- Por ejemplo, la expresión regular: "[^0-9a-zñ]" representa a cualquier cadena que contenga, al menos, un carácter que sea distinto de un dígito o letra minúscula del alfabeto español.

Agrupamiento de patrones

- Los patrones que representan un carácter se pueden agrupar para formar expresiones regulares más complejas, siendo esta característica la razón de la gran potencia de este recurso.
- Son parte del agrupamiento de patrones:
  - Las secuencias.
  - Los metasímbolos.
  - Los cuantificadores.
  - Los paréntesis.
  - Las alternativas.
  - La fijación de patrones.
  - Los caracteres escapados.
  - La precedencia.
  - Las operaciones.

Metasímbolos

- Los metasímbolos son una serie de clases predefinidas dentro de las expresiones regulares estilo Perl.

Clase	Significado
\d	Cualquier dígito decimal. Equivalente a [0-9].
\w	Cualquier dígito decimal, letra en mayúscula o minúscula, incluidas la ñ y el guión bajo (_), letras acentuadas o letras con cualquier tipo de tilde. Equivalente a [0-9A-Za-zÑñÁÉÍÓÚáéíóúÜü_ÀÈÌÒÙàèìòù]
\s	Carácter de espacio en blanco ( ), tabulación (\t), retorno de carro (\r), nueva página (\f) o nueva línea (\n).
\D	Cualquier carácter que no sea un dígito decimal. Equivalente a [^0-9].
\W	Cualquier carácter que no sea dígito decimal, letra mayúscula o minúscula, incluyendo la letra ñ, el guión bajo, letras acentuadas con cualquier tipo de tilde.
\S	Cualquier carácter que no sea el espacio en blanco ( ), la tabulación (\t), el retorno de carro (\r), el carácter de nueva página (\f) y el carácter de nueva línea (\n).

Metasímbolos

- Por ejemplo, la expresión regular:

```
"/\d[A-Z]{3,}/"
```

Coincidiría con: "3DEF", "#33FFEE", "ABC3XYZ".

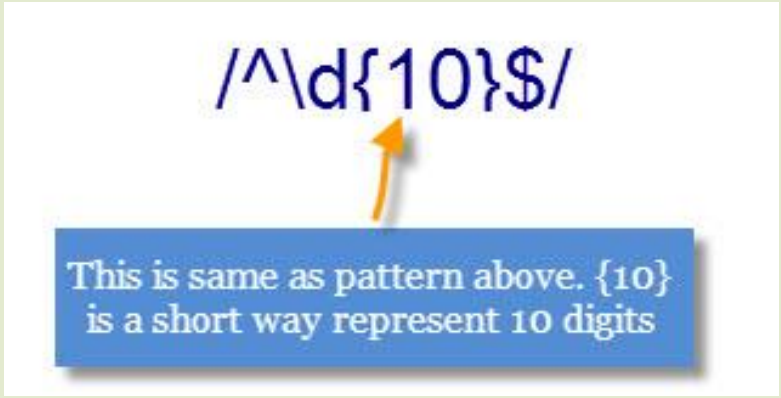
No coincidiría con: "ABCDEFGH", "#36FA96", "123456".

- Por ejemplo, la expresión regular:

```
"/\d{4}-\d{4}/"
```

Coincidiría con "2258-3202", "Mi número telefónico es: 7568-2692", "4321-1208-1175-0698".

No coincidiría con: "2121_5776", "Mi número de teléfono es: 22583206", "Mi NIT es: 1217-280289-305-9".

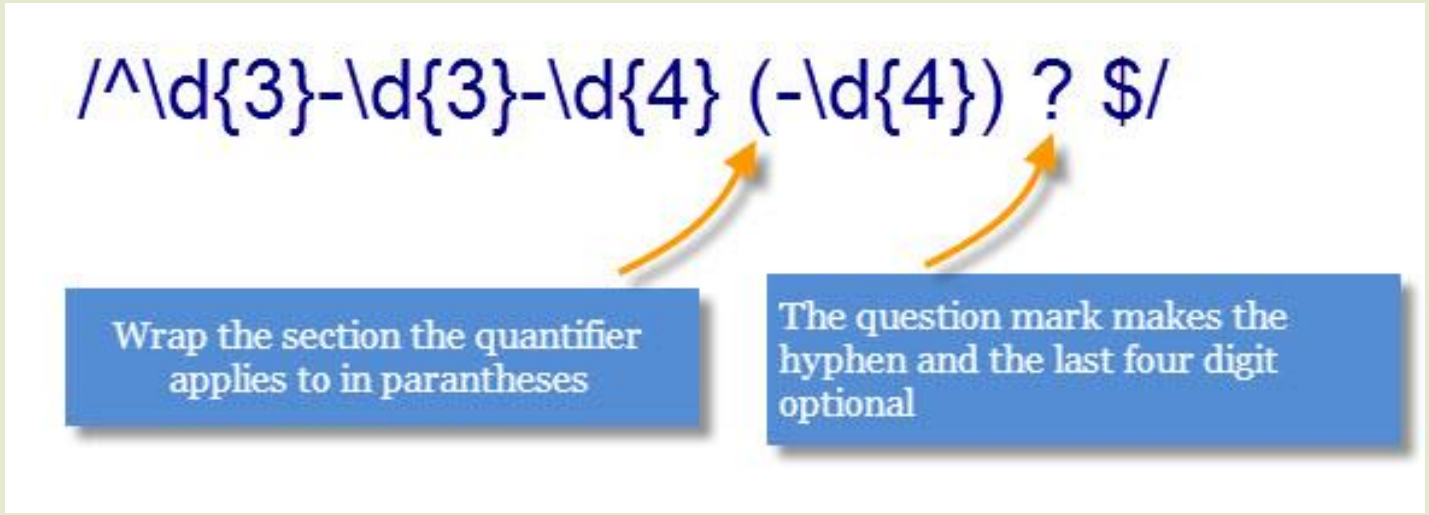


Secuencia

- Una secuencia de patrones está formada por dos o más patrones, uno a continuación del otro.
- Por ejemplo, la expresión regular: "a.[0-9]".
- La expresión regular anterior, está formada por la secuencia de patrones individuales: a, . y [0-9].
- Se producirá concordancia en aquellas cadenas que tengan caracteres que individualmente se correspondan con cada uno de los patrones en el mismo orden especificado en la secuencia.
- Por ejemplo, las cadenas: "a15", "4a.7" o "ar23456" encajan con la expresión regular. No así: "a9", "mi padre" o "3.14159".

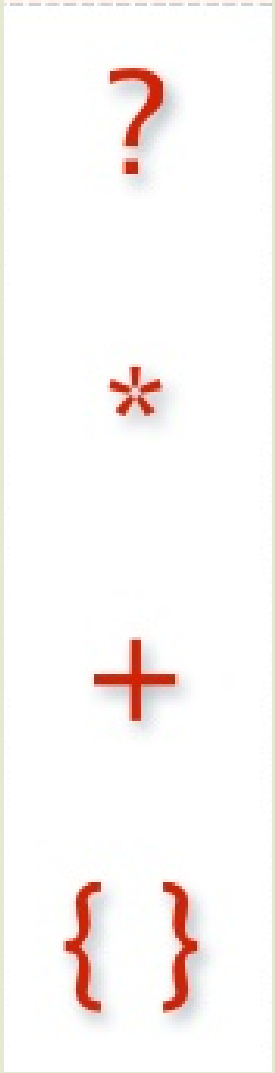
Cuantificadores

- Un cuantificador es un metacarácter que se aplica al patrón ubicado a su izquierda con el propósito de indicar el número de veces que debe aparecer dicho patrón para que se produzca la coincidencia.
- La utilidad de los cuantificadores es muy amplia, ya que permiten definir expresiones regulares para cualquier tamaño de cadenas.



Cuantificadores

- Existen cuatro tipos de cuantificadores fundamentales:
  - **Asterisco (*)**. El metacarácter "*" provoca que el patrón al que afecta se pueda repetir ninguna o más veces.
  - **Signo de adición (+)**. El metacarácter "+" provoca que el patrón al que afecta se pueda repetir una o más veces.
  - **Signo de interrogación (?)**. El metacarácter "?" provoca que el patrón al que acompaña pueda aparecer ninguna o una vez.



Cuantificadores

- **Cuantificador general ({m,n})**. Este formato de cuantificador permite definir un número inferior (m) y otro superior (n) de veces que se debe repetir un patrón para que se produzca la coincidencia.
- **Cuantificador general con límite inferior ({m,})**. Al indicar únicamente el límite inferior en un cuantificador general el patrón deberá repetirse como mínimo el número de veces que indique dicho límite para encontrar coincidencia.
- **Cuantificador general exacto ({m})**. Cuando se desea que en la expresión regular un patrón concreto concuerde un número fijo de veces, se indicará un valor entero específico entre llaves.

Cuantificadores

Veamos algunos ejemplos:

```
//Patrón *
"12*3" representa todas las cadenas que contienen el carácter 1, seguido de
       cualquier número de veces (o incluso ninguna) del carácter 2 y del carácter
       3.
Coincide con: "Tú edad es 13 años", "123 ahora", "2013", "1223", "01222223".
No coincide con: "Tú edad es 12", "23 goles", "2012", "0246".
```

Cuantificadores

Veamos algunos ejemplos:

```
//Patrón +
"12+3" representa todas las cadenas que contienen el carácter 1, seguido de una o
       más ocurrencias del carácter 2 y del carácter 3.
Coincide con: "Tú edad es 5123 días", "123 ahora", "2123", "1223", "01222223".
No coincide con: "Tú edad es 5023 días", "13 goles", "2013", "0246".
```

Cuantificadores

Veamos algunos ejemplos:

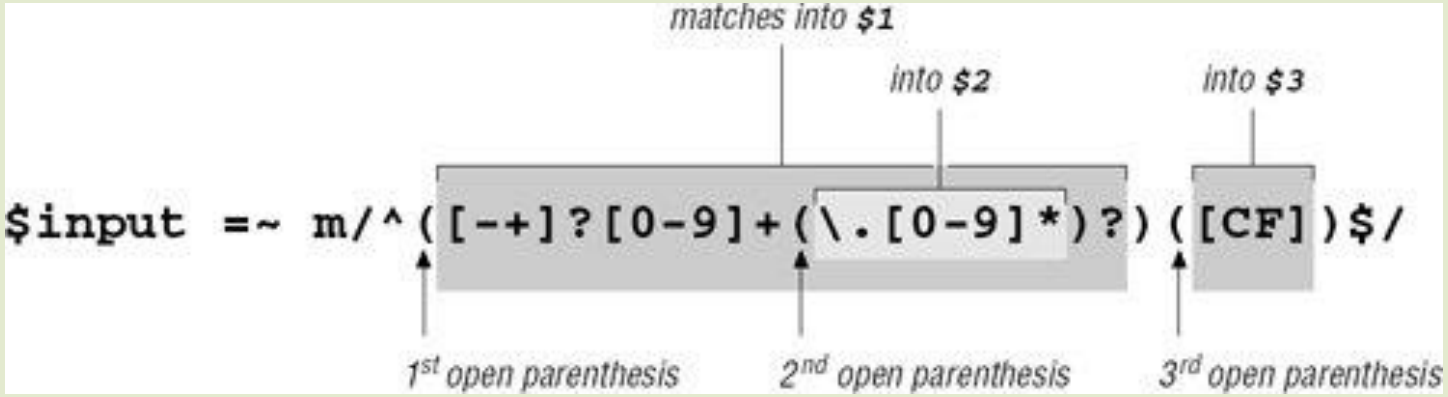
```
//Patrón ?

"12?3" representa todas las cadenas que contienen el carácter 1, seguido de ninguna
       o una ocurrencia del carácter 2 seguido del carácter 3.

Coincide con: "Tú edad es 13 años", "123 ahora", "2013", "Hoy es 13 de febrero",
       "C1230".

No coincide con: "Tú edad es 5023 días", "3223 puntos", "1223", "12?3".
```

Modificadores

Modificador	Significado
i	Provoca que no se haga distinción entre letras mayúsculas y minúsculas como sucede por defecto.
m	Provoca que se trate la cadena como que si estuviera formada por múltiples líneas.
s	Provoca que el carácter de salto de línea también se empareje con el matacarácter.
u	Hace que la cadenas de patrones sean tratadas como UTF-8. Sólo está disponible a partir de las versiones de PHP 4.1.0 en sistemas UNIX y a partir de PHP 4.2.3 en sistemas Windows.
x	Provoca que los caracteres de la clase "espacios en blanco" sean ignorados dentro del patrón, excepto cuando estén escapados o vayan dentro de una definición de clases de caracteres.

Paréntesis

- Los paréntesis realizan una doble función dentro de las expresiones regulares.
- Primera función: sirven para agrupar partes de un patrón de las que se desea que formen una entidad propia (subpatrón).
- Segunda función: sirven para memorizar las partes de una cadena que se empareja con el subpatrón rodeado por dichos paréntesis dentro de la expresión regular (subpatrón de captura).



Paréntesis

- Por ejemplo, cuando se desee que un cuantificador afecte a un patrón de más de un carácter, tendremos que encerrar la secuencia de caracteres o metacarácteres entre paréntesis.
- Por ejemplo, la expresión regular:

```
"(la){2}" representa cadenas como: "lala", "RalalaRa".
```

- Por ejemplo, la expresión regular:

```
$myPets = "favoritePet=Lucky, Rover=dog, Lucky=cat";
preg_match('/favoritePet\=(\w+).*\1\=(\w+)/', $myPets, $matches);
//Imprimirá "My favorite pet is a cat called Lucky
echo "My favorite pet is a " . $matches[2] . " called " . $matches[1] .
   ".";
```

Alternativas

- Las expresiones regulares también permiten crear alternativas utilizando el metacarácter de barra vertical: "|".
- Cuando aparece este metacarácter dentro de una expresión regular se crean alternativas de subpatrones de coincidencia, de modo que si cualquiera de los subpatrones combinados coincide con la cadena de prueba, todo el patrón coincide.

(gif|jpg)	Matches either “gif” or “jpg”

Alternativas

- Por ejemplo:

"blanco|negro": representará todas las cadenas que contengan la cadena "blanco" o bien la cadena "negro".

Coincide con:

"blanco", "blanco y negro", "tiro al blanco", "oro negro" y "negro panorama"

No coincide con:

"blanca nieves", "la negra Tomasa", "blanquita", "negrura" y "el negrito del batey".

Alternativas

- También se pueden crear múltiples alternativas, utilizando más de una vez el metacarácter barra vertical: "|".
- Como se indica a continuación:

"0|1|2|3|4|5": coincide con todas las cadenas que incluyan el 0, el 1, el 2, el 3, el 4 o el 5.

La expresión anterior equivale a "[0-5]".

- También puede incluir subpatrones agrupados dentro de las alternaticas:

"abc|(xyz){3}": coincide con cualquier cadena que contenga "abc" o "xyzxyzxyz".

Fijación de patrones

- Para determinar si una cadena es representada por una expresión regular, se tiene que recorrer de izquierda a derecha, buscando algún punto en donde se empareje completamente con dicho patrón.
- A veces es necesario que esa coincidencia se produzca en un punto fijo, casi siempre al inicio o al final o en ambas partes.
- Los metacarácteres que permiten fijar o anclar estas posiciones en el patrón son dos: ^ y $.

Fijación de patrones

- Para determinar si una cadena es representada por una expresión regular, se tiene que recorrer de izquierda a derecha, buscando algún punto en donde se empareje completamente con dicho patrón.
- A veces es necesario que esa coincidencia se produzca en un punto fijo, casi siempre al inicio o al final o en ambas partes.
- Los metacarácteres que permiten fijar o anclar estas posiciones en el patrón son dos: ^ y $.

Fijación de patrones

- El metacarácter circunflejo, "^" se utiliza para indicar que la cadena coincidirá únicamente si concuerda con el patrón al inicio de la misma.
- De forma similar, el metacarácter dólar, "$" se utiliza para establecer que la coincidencia de la cadena con el patrón debe producirse únicamente al final de la cadena.

Fijación de patrones

- Por ejemplo:

"^http(s)?:\/\/" coincidirá con las cadenas que inicien con la secuencia de caracteres "http://" como las URLs, pero no coincidirá con cadenas como: "http://www.udb.edu.sv" o "https://www.facebook.com".

Pero no con "mi web es http://www.yomismo.com" porque la secuencia http:// no está al inicio de la cadena.

"\.com(\.[a-z][a-z])?$" coincidirá con las cadenas que terminen con ".com", ".com.sv", ".com.br", ".com.us", ".com.uk", etc. Pero no con las que terminen con: ".com.gob".

Escape de caracteres

- Las expresiones regulares utilizan un total de 19 caracteres que son considerados caracteres especiales en la construcción de patrones. Estos caracteres son: `.`, `\`, `+`, `*`, `?`, `[`, `]`, `^`, `$`, `(`, `)`, `{`, `}`, `=`, `!`, `<`, `>`, `|` y `:`.
- Seguramente se preguntará cómo se puede incluir uno de esos caracteres para formar un patrón en una expresión regular.
- La respuesta es que hay que escaparlo, anteponiendo precisamente uno de esos símbolos, la barra inclinada "\" antes del carácter especial a incluir como parte del patrón.

Escape de caracteres

- Por ejemplo:

"^[0-9]+\.{1}[0-9]+$" coincidirá con cualquier número con parte decimal, pero no con números enteros.

Coincidirá con:

7.5, 27.25, 562.24, 121.112233, 3.14159.

No coincidirá con:

.25, a.15, 12e+3, 63, 2e.76, n11.24.

Precedencia

- La precedencia de patrones permite resolver ambigüedades en la utilización de los metacarácteres y símbolos utilizados en una expresión regular.
- Por ejemplo, si tenemos la expresión regular: `o|p{2}`, podría dar lugar a dos interpretaciones.
- La primera, en donde la expresión regular representaría a las cadenas que contengan el carácter "o" o contengan el carácter "p" repetido dos veces.
- La segunda, en la que el patrón representa a todas las cadenas que contienen una secuencia de dos caracteres formada indistintamente por los caracteres "o" o "p".

Precedencia

- En este caso, la interpretación correcta es la primera.
- La ambigüedad se resuelve utilizando paréntesis, de manera que si queremos aplicar la segunda interpretación, debería utilizarse una expresión regular como esta:

"(o|p){2}".

- La precedencia de operadores en orden decreciente para expresiones regulares es la siguiente:
  - Paréntesis: "(" y ")".
  - Cuantificadores: "+", "*", "?", "{", "}".
  - Secuencia y fijación: "abc", "^", "$", \b, \B.
  - Alternativa: "|".

Expresiones regulares en cadenas de PHP

- Una expresión regular en PHP se expresa como una cadena de caracteres. Esto le da una dificultad añadida, ya que tanto las cadenas de caracteres como las expresiones regulares requerirán escapar algunos caracteres.
- Cuando se requiera buscar un carácter literal, que sea un metacaracter, dentro de una cadena. Para hacer referencia a éste dentro del patrón deberá indicarlo usando secuencias de escape, como se hace en las cadenas.

Expresiones regulares en cadenas de PHP

- Por ejemplo:

Cadena destino:

```
"25.2 + 7.5"
```

Patrón:

```
/^\d+(\. \d)*\s*([+-/*%]{1} \s*\d+(\. \d)*)$/
```

//Uniendo todo en código PHP

```
$operacion = "25.2 + 7.5";
$patron = "/^\d+(\.\d)*\s*([\+\-\/\*%]{1} \s*\d+(\.\d)*)$/";
echo preg_match($patron, $operacion); //Devolverá true o 1
```

Formas de trabajar con expresiones regulares en PHP

- En PHP se puede trabajar con expresiones regulares tipo POSIX, que han sido descontinuadas por las versiones más recientes de PHP, desde la versión 5.3.
- Las expresiones regulares estilo Perl, que son las que examinaremos en detalle en esta clase. Este nombre se debe a que fue en el lenguaje Perl donde se dieron a conocer antes que en muchos otros lenguajes.

Funciones de expresión regular en PHP

- En PHP se pueden utilizar las siguientes funciones para hacer comparaciones, búsquedas o sustituciones en expresiones regulares:
- **preg_match()**. Esta es la función de coincidencia de patrón principal en PHP.
- La sintaxis es la siguiente:

```
int preg_match(string $pattern , string $subject [, array
&$matches [, int $flags = 0 [, int $offset = 0 ]]]);
```

Funciones de expresión regular en PHP

- Estos argumentos son:

1. La expresión regular a buscar pasada como cadena.
2. La cadena en la cual se buscará la coincidencia.
3. Una matriz opcional en la que se guardará cualquier texto coincidente. El texto que coincida se almacena en el primer elemento de esta matriz.
4. Un entero opcional que especifica cualquier configuración para aceptar una coincidencia. Están soportada la bandera PREG_OFFSET_CAPTURE. Si se proporciona este argumento con esta bandera, por cada coincidencia producida, el índice de la cadena añadida también será devuelto.
5. Un desplazamiento entero, también opcional, que hace que la búsqueda no comience desde el principio si se proporciona un valor mayor que 0.

Funciones de expresión regular en PHP

- La función **preg_match()** devolverá cero (0) si el patrón no coincide o no es localizado en la cadena, o devolverá uno (1) si lo encuentra.
- Normalmente, si se encuentra una coincidencia del patrón en la cadena destino la búsqueda parará y se devolverá true, o sea, 1.

Funciones de expresión regular en PHP

- Ejemplo:

```
echo preg_match("/world/", "Everybody wants to rule the world");
//Mostrará el valor "1", o sea true
echo preg_match("/^world/", "Everybody wants to rule the world");
//Se haría coincidir sólo al inicio de la cadena, por tanto
//mostrará el valor "0", o sea, falso
echo preg_match("/world/", "Everybody wants to rule the world",
$coincidencias); //Mostrará "1"
foreach($coincidencias as $item):
  echo $ítem . "<br />"; //Mostrará "world"
endforeach;
```

```
1
0
1
world
```

Funciones de expresión regular en PHP

- **preg_match_all()**. Permite encontrar múltiples coincidencias de patrón en una misma cadena, algo que no hace preg_match() que solo encuentra la primera coincidencia.
- **preg_match_all()** utiliza los mismos argumentos que la función **preg_match()**.
- Al igual que preg_match(), preg_match_all() devolverá un valor entero con la diferencia que en el caso de la última ese número representa el número de coincidencias encontradas.

Funciones de expresión regular en PHP

- **preg_grep()**: Devuelve una matriz con aquellos elementos que concuerden con un patrón encontrado en otra matriz en lugar de una cadena. También se puede utilizar de forma inversa y obtener una matriz con todos los elementos que no concuerden con el patrón que se busque.
- **preg_replace()**: Busca concordancias del patrón con la cadena pasada como parámetro devolviendo el resultado de realizar el reemplazo correspondiente sobre dichas concordancias.

Funciones de expresión regular en PHP

- **preg_split()**. Divide una cadena utilizando algún delimitador que marca un punto de separación entre las palabras de la cadena. La función devuelve una matriz de subcadenas, cada una de las cuales es el resultado de dividir la cadena de entrada utilizando el separador o delimitador entre cada porción de la subcadena que coincida con el patrón indicado en la entrada de la función. Este delimitador, puede ser, en su forma más simple, un carácter normal o, en su forma más compleja, otra expresión regular.

Funciones de expresión regular en PHP

Ejemplo:

```
$texto = "Real Madrid, Bayern Munich, FC Barcelona, Arsenal, Juventus y Chelsea";
$equipos = preg_split("/,\s*|\s+y\s+/", $texto);
echo "<pre>";
print_r($equipos);
echo "</pre>";
```

Resultado:

```
Array
(
    [0] => Real Madrid
    [1] => Bayern Munich
    [2] => FC Barcelona
    [3] => Arsenal
    [4] => Juventus
    [5] => Chelsea
)
```

FIN