EXPRESIONES REGULARES

ACCESO A DATOS

Documento realizado por
Oriol Raventós

ÍNDICE

EL ORIGEN

Las expresiones regulares surgieron en la década de 1950 gracias al matemático Stephen Kleene, quien propuso un sistema algebraico para representar patrones de cadenas de caracteres.

Estos patrones se volvieron fundamentales en la teoría de autómatas y la lógica matemática.

Con el tiempo, las expresiones regulares se popularizaron gracias a su integración en herramientas informáticas, especialmente en el sistema operativo UNIX desarrollado por Ken Thompson.

Desde entonces, han sido parte esencial de la programación y el procesamiento de texto.

¿QUÉ SON?

Las expresiones regulares, también denominadas regex o regexp, son patrones de búsqueda que permiten encontrar y trabajar con texto en cadenas de caracteres de manera efectiva.

Estos patrones tienen la capacidad de realizar operaciones avanzadas de búsqueda, validación, reemplazo y manipulación de datos dentro de programas informáticos.

Su utilidad es extensa. Permiten especificar patrones complejos, como identificar ciertos tipos de caracteres, manejar patrones repetitivos o incluso definir rangos de caracteres. Su aplicación es diversa, desde editores de texto hasta el procesamiento avanzado de texto en programas informáticos, análisis de datos y validación de formatos.

Por ejemplo, una expresión regular simple como:
‘**\d+’**, identifica una secuencia de uno o más dígitos dentro de un texto.
En contraste, una expresión más elaborada como:
‘**^([a-zA-Z0-9_.+-]+)@([a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)$’**, verifica si una cadena de texto tiene el formato de una dirección de correo electrónico válida.

A pesar de las variaciones que pueden surgir dependiendo del lenguaje de programación o del contexto en el que se utilicen, la mayoría de los patrones mantienen una estructura y funcionamiento coherentes. Estas expresiones son una herramienta fundamental para trabajar con texto en programación.

ESTRUCTURA

La estructura varía dependiendo del patrón que desees buscar o manipular en un texto. Existen componentes comunes que se utilizan para definir estos patrones:

- **Caracteres literales**, es decir, caracteres que marcan que se debe hallar un carácter como él para hacer match.

- Las **clases de caracteres**, que representan tipologías de caracteres ,como por ejemplo;

    - '**[abc]**' para designar un carácter que puede ser cualquiera de los indicados (‘**a**’, '**b**’, o '**c**’).

    - ‘**[o-9]**’ cualquier carácter de la secuencia indicada (del '0' al '9'). ‘**[a-zA-Z]**' conjuntos de secuencias (letras minúsculas y mayúsculas).

    - ‘**[^abc]**’, que es la negación de las clases indicadas (en este caso, cualquier carácter que no sea ‘**a**’, '**b**’, o ‘**c**’).

- Hay otro metacaracter especial de uso muy común que es **‘ . ’** (punto), que representa cualquier carácter, exceptuando saltos de línea (pero exactamente un solo carácter). Por ejemplo, el patrón '**[aA].[^0-9]**', buscará todas las secuencias de tres caracteres que empiecen por la letra a mayúscula o minúscula, contengan cualquier carácter en medio, y acaben por un carácter no numérico.

- Los **Cuantificadores**, son **metacaracteres** que indican como se pueden repetir los caracteres o grupos anteriores. Por ejemplo:

  - Un **?** indica que el carácter anterior es opcional (se puede repetir entre 0 ó 1 vez).

  - Un ***** indica que el carácter anterior se puede repetir entre 0 y más veces.

  - Un **+** indica que puede ser entre 1 y más veces.

  - Para acabar **{a,b}**, indica que el carácter se puede repetir un mínimo de **‘a’** veces y un máximo de **b** veces (‘**a’** y ‘**b’** son números). Por ejemplo, **{2,}** indica un mínimo de 2 y sin máximo, **{,3}** significa un máximo de 3 y sin mínimo, o **{4}** indica que se debe repetir exactamente 4 veces).

    - Por ejemplo, ‘**las?**’ hará match con '**la**' y con ‘**las**’, ya que se ha indicado que la '**s**’ es opcional. O ‘**[0-9]+**’ buscará secuencias de dígitos (con al menos un dígito como mínimo).

- Hay dos **Anclajes** básicos, ‘**^**’ indica inicio de línea, y '**$**' indica final de línea. Sirven, para asegurar que la coincidencia es de la línea entera. Estos anclajes no consumen caracteres (zero-Length anchor). Por ejemplo ‘**^a**’, contra el texto ‘**a b a c**’ solo haría match con la primera ‘**a**’. Pero el patrón ‘**^**’, es un anclaje al principio de cada línea, pero no devuelve ningún carácter.

- Las **agrupaciones** sirven para agrupar conjuntos de caracteres, para que, por ejemplo, les aplique un mismo metacaracter. Por ejemplo, un ‘**(ma)+b**’ haría coincidencia con ‘**mab**’ y con ‘**mamab**’, pero no con ‘**maab**’. Se indican agrupando el conjunto con paréntesis ‘**()**’. Ademas, tambien permiten:

    - Las **agrupaciones** pueden ser referenciadas posteriormente, con ‘**\i**’ para indicar el numero de agrupación (backreference).Por ejemplo, ‘**([abc])y\1**’, haría match con ‘**a y a**’, pero no con ‘**a y b**’ ya que ‘**\1**’ indica el valor capturado en el primer grupo (‘a’ en el ejemplo).

    - También se pueden indicar referencias a los grupos capturados, en procesadores de texto que permitan reemplazar texto (Find&Replace). Para indicar en el texto de sustitución, el valor de un grupo capturado (\1-\9 del primer al noveno grupo, y \0 para todo el texto coincidente).

- Se pueden indicar **Alternativas**, como si fuera una OR lógica, con '**|**'. Por ejemplo, '**niño|a**', trata la '**a**' y la '**o**' adyacentes al pipe como alternativas, y haría match con '**niña**' y con '**niño**'. Se permite tratar agrupaciones como alternativas, por ejemplo '**(hombre)|(mujer)**' haría match con '**hombre**' y con '**mujer**'.

- En caso que se quiera indicar un carácter que si se quiere que se use con el valor formal de este, y no como un metacaracter, este debe ser escapado con una contrabarra '****'. Por ejemplo, '**\.\?**' hace match con el texto '**.?**', pero no significa un carácter cualquiera opcional (que es lo que significaría la Regex '**.?**')

¿COMO SE UTILIZAN?

son herramientas muy útiles en la programación y el manejo de texto. Se usan para **buscar, validar y manipular** cadenas de caracteres en diferentes lenguajes y herramientas informáticas. La forma de utilizarlas puede cambiar un poco según dónde las apliques, pero generalmente sigues estos pasos:

1. Elige un lenguaje o herramienta: Selecciona el lenguaje de programación o la herramienta en la que trabajarás con expresiones regulares, como Python, JavaScript o Perl, cada uno con su propia forma de usarlas.

2. Aprende la sintaxis: Familiarízate con la sintaxis de expresiones regulares en tu lenguaje o herramienta. Cada uno tiene su manera específica de escribir y emplear estos patrones.

3. Define el patrón: Crea la expresión regular que describa el patrón que necesitas buscar. Por ejemplo, para números de teléfono, podrías usar algo como \d{3}-\d{3}-\d{4}.

4. Usa funciones o métodos: Utiliza las funciones o métodos proporcionados por tu lenguaje o herramienta para aplicar la expresión regular al texto que estás analizando. Dependiendo del propósito de tu búsqueda, hay diversas funciones disponibles, como search, match, findall, replace, entre otras.

5. Aplica la expresión regular: Ejecuta la función o método con el patrón de expresión regular y el texto que deseas analizar. Esto llevará a cabo la búsqueda o manipulación de acuerdo con el patrón que definiste.

Por ejemplo, en Python con el módulo re, podrías emplear la función re.findall() para buscar todas las coincidencias de un patrón en un texto.
Esto es solo un ejemplo básico. Las expresiones regulares tienen un amplio alcance, permitiendo una variedad de operaciones con texto en distintos contextos. Su uso se ajusta a las necesidades específicas del proyecto o tarea en curso.

¿PARA QUE SE UTILIZAN?

Las expresiones regulares son como un **conjunto de reglas** que se usan en programación y otras herramientas informáticas para **buscar, validar o manipular texto** de forma precisa. Se usan en situaciones donde necesitas encontrar patrones específicos en un texto, como direcciones de correo electrónico, números de teléfono o cualquier formato definido por una estructura repetitiva.

Imagina que son como una especie de buscador avanzado que te permite hacer tareas específicas, como comprobar si un texto tiene el formato correcto de un número de teléfono o extraer información precisa de un documento. También sirven para cambiar o manipular texto, como reemplazar palabras específicas o limpiar datos.

En entornos más complejos, como en la construcción de programas, análisis de datos o procesamiento de texto, estas expresiones son herramientas fundamentales que facilitan la búsqueda y manipulación de texto basadas en patrones predefinidos.

FUENTES

www.esic.edu/rethink/tecnologia/que-es-una-expresion-regular-que-tipos-existen-c

es.wikipedia.org/wiki/Expresión_regular#El_signo_de_suma_"+"

medium.com/@jmz12/expresiones-regulares-215af64acab1

platzi.com/blog/que-expresion-regular/

4geeks.com/es/lesson/regex-tutorial-regular-expression-ejemplo