Investigación expresiones regulares

Una expresión regular define un patrón de búsqueda para cadenas de caracteres.

La podemos utilizar para comprobar si una cadena contiene o coincide con el patrón. El contenido de la cadena de caracteres puede coincidir con el patrón 0, 1 o más veces.

Algunos ejemplos de uso de expresiones regulares pueden ser:

* para comprobar que la fecha leída cumple el patrón dd/mm/aaaa
* para comprobar que un NIF está formado por 8 cifras, un guión y una letra
* para comprobar que una dirección de correo electrónico es una dirección válida.
* para comprobar que una contraseña cumple unas determinadas condiciones.
* Para comprobar que una URL es válida.
* Para comprobar cuántas veces se repite dentro de la cadena una secuencia de caracteres determinada.

-           Etc. Etc.

El patrón se busca en el String de izquierda a derecha. Cuando se determina que un carácter cumple con el patrón este carácter ya no vuelve a intervenir en la comprobación.

Ejemplo:

La expresión regular  "010" la encontraremos dentro del String "010101010" solo dos veces: "010101010"

Símbolos comunes en expresiones regulares

Símbolos comunes en expresiones regulares

| **Expresión** | **Descripción** |
| --- | --- |
| . | Un punto indica cualquier carácter |
| ^expresión | El símbolo ^ indica el principio del String. En este caso el String debe contener la expresión al principio. |
| expresión$ | El símbolo $ indica el final del String. En este caso el String debe contener la expresión al final. |
| [abc] | Los corchetes representan una definición de conjunto. En este ejemplo el String debe contener las letras a ó b ó c. |
| [abc][12] | El String debe contener las letras a ó b ó c seguidas de 1 ó 2 |
| [^abc] | El símbolo ^ dentro de los corchetes indica negación. En este caso el String debe contener cualquier carácter excepto a ó b ó c. |
| [a-z1-9] | Rango. Indica las letras minúsculas desde la a hasta la z (ambas incluidas) y los dígitos desde el 1 hasta el 9 (ambos incluidos) |
| A|B | El carácter | es un OR.  A ó B |
| AB | Concatenación. A seguida de B |

Meta caracteres

| **Expresión** | **Descripción** |
| --- | --- |
| \d | Dígito. Equivale a [0-9] |
| \D | No dígito. Equivale a [^0-9] |
| \s | Espacio en blanco. Equivale a [ \t\n\x0b\r\f] |
| \S | No espacio en blanco. Equivale a [^\s] |
| \w | Una letra mayúscula o minúscula, un dígito o el carácter \_  Equivale a  [a-zA-Z0-9\_] |
| \W | Equivale a [^\w] |
| \b | Límite de unapalabra. |

En Java debemos usar una doble barra invertida \\

Por ejemplo para utilizar  \w tendremos que escribir \\w. Si queremos indicar que la barra invertida en un carácter de la expresión regular tendremos que escribir  \\\\.

Cuantificadores

| **Expresión** | **Descripción** |
| --- | --- |
| {X} | Indica que lo que va justo antes de las llaves se repite X veces |
| {X,Y} | Indica que lo que va justo antes de las llaves se repite mínimo X veces y máximo Y veces. También podemos poner {X,} indicando que se repite un mínimo de X veces sin límite máximo. |
| \* | Indica 0 ó más veces. Equivale a {0,} |
| + | Indica 1 ó más veces. Equivale a {1,} |
| ? | Indica 0 ó 1 veces. Equivale a {0,1} |

Para usar expresiones regulares en Java se usa el package java.util.regex

Contiene las clases Pattern y Matcher y la excepción PatternSyntaxException.

Clase Pattern: Un objeto de esta clase representa la expresión regular. Contiene el método compile(Stringregex) que recibe como parámetro la expresión regular y devuelve un objeto de la clase Pattern.

La clase Matcher: Esta clase compara el String y la expresión regular. Contienen el método matches(CharSequence input) que recibe como parámetro el String a validar y devuelve true si coincide con el patrón. El método find() indica si el String contienen el patrón.