Listado Expresiones Regulares

Fecha entrega: 01-03-2015

Autores: Pedro J. Ramos
Estela Muñoz

LISTADO DE EJERCICIOS. EXPRESIONES REGULARES (RegEx).

1. Describe las clases implicadas en java.util.regex para manejar las expresiones regulares.

Pattern sirve para buscar una expresión regular en el contenido del Matcher. Y PatternSyntaxException indica un error ocurrido en la sintaxis del Pattern.

2. Considera el literal "hola". Indica el índice de inicio y el índice de fin. Explica su significado.

"hola" está entre el índice 0 y 3. La cadena empieza en la posición 0 y ocupa 4 caracteres.

3. Explica qué son los metacaracteres. Enuméralos.

Son caracteres que se utilizan para buscar expresiones regulares con determinadas características (que se repiten o no, si van dentro de un rango, etc).
Son los siguientes:

.
^
$
?
*
+
[…]
[^...]
[a-z1-9]
{}
{,}
A|B
AB
\d
\D
\w
\W
\b

4. Indica cómo se fuerza a que un metacarácter sea tratado como un carácter ordinal.

Con \ delante se tratan como un carácter ordinal.

5. Indica los caracteres que representan:

**a. ** \
b. \n Espacio en blanco.
c. \t Espacio en blanco.

6. Explica el significado de los siguientes metacaracteres:

a. . Un punto indica cualquier carácter.
b. ^ El símbolo ^ indica el principio del String. En este caso el String debe contener la expresión al principio.
c. $ El símbolo $ indica el final del String. En este caso el String debe contener la expresión al final.
**d. ** Con \ delante de un metacarácter se tratan como un carácter ordinal.
e. […] Los corchetes representan una definición de conjunto.
f. [^…] El símbolo ^ dentro de los corchetes indica negación.
g. [a-z] Indica un rango de caracteres.
h. […&&…] Indica que aparecen ambos.
i. ? Indica que el elemento que va delante puede aparecer de 0 a 1 veces.
j. + Indica que el elemento que va delante puede aparecer 1 o más veces.
k. * Indica que el elemento que va delante puede aparecer 0 o más veces.

7. Analiza el carácter "-" en las siguientes expresiones. ¿En ambas funciona como metacarácter? Utiliza coincidencias positivas y negativas.

a. A-Z Aquí se usa como un carácter más, como la A o la Z.
b. [A-Z] Aquí sirve para indicar que hay un rango de la A a la Z.

8. Describe las siguientes clases de caracteres predefinidas. Busca su expresión regular equivalente mediante corchetes (clases)

a. \d Dígito. Equivale a [0-9].
b. \s Espacio en blanco. Equivale a [\t\n\x0b\r\f].
c. \w Una letra mayúscula o minúscula, un dígito o el carácter "_". Equivale a [a-zA-Z0-9_].

9. Busca dos expresiones que representen lo opuesto de:

a. \d \D y [^0-9].
b. \s \S y [^\s].
c. \w \W y [^\w].

10. Busca la coincidencia:

a. (hola){2} holahola

11. Explica la utilidad de los siguientes métodos de la clase java.util.regex.Pattern, indicando argumentos y valores devueltos:

a. compile() Recoge la expresión regular a buscar.

b. matcher() Recoge la cadena con la que se compara la expresión regular.

c. matches() Comprueba si la *cadena* contiene exactamente el patrón.

d. split() Para separar cadenas. Este método divide el String en cadenas según la expresión regular que recibe.

12. Explica la utilidad de los siguientes métodos de la clase java. util.regex.Matcher:

a. matches() Comprueba si la *cadena* contiene exactamente el patrón.

b. lookingAt() Compara con la secuencia buscada desde el principio con el patrón

c. find() Busca la siguiente subsecuencia que coincida con el patrón.

d. group() Busca en la cadena tratándola como un conjunto.

e. start() Vuelve al principio de la cadena comparada.

f. end() Va al final de la cadena comparada.

13. ¿Cuándo conviene utilizar un matches() u otro?

Dependiendo de si deseas buscar desde el principio de la cadena (lookingAt()), o si quieres posicionarte al principio o al final de la cadena (start() y end()), o si quieres usar la cadena como un conjunto (group()), o si quieres buscar de subsecuencia en subsecuencia (find()).

14. Objetos que hay que crear antes de buscar coincidencias (matches) de la expresión regular. ¿Puede hacerse de otras formas? Investiga

Uno de tipo Pattern y otro de tipo Matcher.

15. Utilidad de los caracteres ^y $

Para indicar el principio y final de una expresión regular. También puede usarse ^ como negación.

16. Indica mediante coincidencias positivas y negativas las diferencias entre estas expresiones regulares:

a. t.*s t....s ó ts

b. t.*?s t.s ó ts

17. Indica las expresiones regulares correspondientes a las siguientes especificaciones:

a. Palabras que comiencen por la letra t: ^t.$

b. Palabras que finalicen por la letra s: ^.s$

18.Copia el siguiente código y comprueba su funcionamiento.
Entrega TestValidatorOnline y compruébalo con las expresiones regulares:

a. hola "hola" / "ola"

b. \sHola " Hola" / "Hola"

c. hola{3} "holaaa" / "hola"

d. **(hola){3}** "holaholahola" / "hola"

e. [hola]{3} "hhh" / "hohoho"

f. .bat "abat" / "bat"

g. \.bat ".bat" / "bat"

h. \\.bat "\abat" / "\.bat"

i. \\hola "\hola" / "hola"

j. \whola "_hola" / "hola"

k. \Whola "€hola" / "3hola"

l. ([A-Z][a-zA-Z]*)\s\1 "Aa Aa" / "aa "

m. \d\d "89" / "ab"

n. \b "casa" / "vete a casa"

o. \\b "\casa" / "casa"

Envía el resultado de la ejecución con dos entradas para cada expresión: una que coincida y otra que no.