Técnicas de programación segura

2ºDAM IoT

Introducción

En esta unidad vamos a ver:

- Validación de entradas con expresiones regulares
- Ficheros de log con la librería Log4j.

Expresiones regulares

Expresiones regulares

Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.

Las expresiones regulares proporcionan una manera muy flexible de buscar o reconocer cadenas de texto.

Expresiones regulares

La validación de datos mediante expresiones regulares permite:

- Mantener la consistencia de los datos: Por ejemplo, si a un usuario le indicamos que debe introducir un DNI con cierto formato nos aseguramos que lo ha introducido tal como esperábamos.
- Evitar desbordamientos de memoria (buffer overflow): Al comprobar el formato y la longitud del campo evitamos que se produzcan los desbordamientos de memoria.

Expresiones regulares

Los corchetes [] representan los posibles caracteres.

- [abc] representa a, b ó c.
- [123] representa 1, 2 o 3.

Dentro del corchete podemos poner un - para introducir un rango.

- [0-9] representa cualquier número entre 0 y 9.
- [a-z] representa cualquier letra en minúscula
- [A-Z] representa cualquier letra en mayúsculas.

Expresiones regulares

Ejercicio:

- Valida cualquier letra en mayúsculas o minúsculas.
- Valida un número del 1 al 5 y también el 9.
- Valida un carácter que puede ser un número del 1 al 5 o letras de la ‘a’ la ‘f’ o la ‘X’ mayúscula.

Expresiones regulares

Solución:

- Valida cualquier letra en mayúsculas o minúsculas. **[a-zA-Z]**
- Valida un número del 1 al 5 y también el 9. **[1-59]**
- Valida un caracter que puede ser un número del 1 al 5 o letras de la ‘a’ la ‘f’ o la ‘X’ mayúscula. **[1-5a-fX]**

Expresiones regulares

^ Símbolo de negación de un rango.

Términos que empiezan por ‘li’ y terminan por ‘e’ pero no tienen ‘v’ en el interior. live no sería válido y like sí. **li [^ v] e**

^ Comienzo de una cadena: Términos que empiecen por http. **^http**

| Operador alternativa, o una opción u otra.

Validar palabras blanco o negro. **blanco|negro**

Palabras que empiezan por A o B seguida de una vocal.

[A|B][aeiou]

Expresiones regulares

Clases de caracteres predefinidos. Son como atajos de teclado.

Expresión	Descripción
.	Cualquier carácter.
\d	[0-9]
\D	[^0-9]
\s	Espacio en blanco
\S	[^\s]
\w	[a-zA-Z_0-9]
\W	[^\w]

Expresiones regulares

Cuantificadores

Expresión	Descripción
n+	Al menos una ocurrencia de n.
n*	De 0 a más ocurrencias.
n?	Cero o una ocurrencias.

Palabras que empiezan por varias repeticiones de ‘ab’ seguidas de una c.
(ab)*c
Observa que utilizamos el operador de paréntesis para agrupar ab.
Télefono con +34 opcional.
(\+34)?[0-9]{9}
Observa que utilizo el carácter \ para el +.

Expresiones regulares

Cuantificadores

Expresión	Descripción
n{x}	x ocurrencias de n.
n{x,y}	De x a y ocurrencias de n.
n{x,}	Al menos x ocurrencias de n.

2 letras en minúsculas → **[a-z]{2}**

De 2 a 5 letras en minúsculas → **[a-z]{2,5}**

Más de 2 letras en minúsculas → **[a-z]{2,}**

Algunos ejemplos de expresiones regulares

Crea las siguientes expresiones regulares para validar lo siguiente:

- Teléfono con formato, 000-000000
- DNI formato xxxxxxxx-letra
- Cadenas de caracteres minúsculas y mayúsculas de longitud del 5 a 10.
- Números de longitud máxima de 4 caracteres que no empiezan por 0.
- Nombres de fichero con formato pdf.

Algunos ejemplos de expresiones regulares

Crea las siguientes expresiones regulares para validar lo siguiente:

- Teléfono con formato, 000-000000

  [0-9]{3}-[0-9]{6}

  \d{3}-\d{6}

- DNI

  [0-9]{8}-[a-zA-Z]

- Cadenas de caracteres minúsculas y mayúsculas de longitud del 5 a 10.
  [a-zA-Z]{5,10}

- Números de longitud máxima de 4 caracteres que no empiezan por 0.

  ^[1-9]\d{0,4}

  ^[1-9][0-9]{0,4}

- Nombres de fichero con formato pdf. **^.*\.pdf**

Expresiones regulares en Java

Expresiones regulares en Java

Vamos a utilizar las clases Pattern y Matcher de Java.

Importamos la librería: **`import java.util.regex.*;`**

Definimos el patrón (expresión regular):

```
Pattern pat = Pattern.compile(patron);
```

Creamos el objeto Matcher que comprueba si la cadena cumple con la expresión regular:

```
Matcher mat = pat.matcher(“cadena a comprobar”);
```

La clase Matcher tiene varios métodos para comprobar si un String cumple con la expresión regular.

- **matches()**: Busca la coincidencia exacta en el texto. Devuelve un booleano.

Ejemplo Aula Virtual → Expresiones Regulares