Wyrażenia regularne

Wyrażenia regularne - wprowadzenie

Problem:

- Testujemy system rezerwacji biletów na koncert. Po wygenerowaniu biletu zwraca on tekst zawierający datę koncertu, numer biletu i losową wartość weryfikacyjną (służącą do przeciwdziałania fałszerstwom)
-

Koncert: 2 Cellos, data: 21.09.2019, miejsce: Ergo Arena, klucz weryfikacyjny: XD-213A

- Jak zweryfikować poprawność wygenerowanego tekstu?

Wyrażenia regularne

Często mamy do czynienia z danymi wyjściowymi, które mogą przyjmować różną wartość. Często tak jest w przypadku np. sprawdzania logu programu, do którego jest dodana data. Albo danych generowanych całkowicie lub częściowo losowo.

W takim wypadku możemy skorzystać z pomocy wyrażeń regularnych

Wyrażenia regularne

Wzorce które *opisują* **łańcuchy symboli**

Mogą *określać* zbiór **pasujących łańcuchów** lub *wyszczególniać* pasujący **fragment**

\d{2}-\d{3}

80-288
ul. Szczęśliwa 6, 81-123 Gdańsk

\w+\.?\w+@\w+(\.\w+)*(\.\w+)+

napisz proszę ;) : misiaczek87@lubimysie.pl
mój adres email to: jan.kowalski@firma.com
james@company.co.uk

Z czego się składają wyrażenia regularne - podstawowe symbole (1)

symbol	**znaczenie**	**przykład**
.	dowolny znak	.la → Ala Ola Ela Tola
^	początek tekstu (wiersza)	^Ala → Ala ma kota A Ala ma kota
$	koniec tekstu (wiersza)	kota$ → Ala ma kota Ala ma kota a kot...

Z czego się składają wyrażenia regularne - podstawowe symbole (2)

symbol	**znaczenie**	**przykład**
\d	cyfra	\d → 1 03 9 -1 \d\d-\d\d\d → 83-123
\w	znak alfabetu, cyfra lub znak _	\w → p@ssw0rD!_01
\s	biały znak - spacja lub tabulator, znak końca wiersza	
[ABC]	jeden ze znaków w nawiasie	[ABC] → Abecadło ABECADŁO
[a-zA-Z], [1-9]	jeden ze znaków z zakresu	[c-e] → abcdefghaijk [0-9][0-9]-[0-9][0-9][0-9] → 52-150

Z czego się składają wyrażenia regularne - podstawowe symbole (3)

symbol	**znaczenie**	**przykład**
?	zero lub jeden znak poprzedzający	mask?ak?ra → masakra maskara T?.la → Ala Ola Ela Tola
*	zero lub wiele znaków poprzedzających	Witam!* → Witam Witam! Witam!!!! A*.?sprzedam rower → AAAAA sprzedam rower sprzedam rower
+	jeden lub wiele znaków poprzedzających	bu+m! → bum! buuum!

Z czego się składają wyrażenia regularne - podstawowe symbole (4)

symbol	znaczenie	przykład
{2,}	znak poprzedzający ma wystąpić 2 lub więcej razy	r{2,}\w+ → rower rrrrrower rrower
{4}	znak poprzedzający ma wystąpić dokładnie 4 razy	bu{3}m! → bum! buuum! buuuum!
{2,3}	znak poprzedzający ma wystąpić 2 lub 3 razy	(\d{2,3}-){3}\d{2,3} → 12-123-23-442
|	Alternatywa - jedna albo druga grupa znaków	[A-D]+|[X-Z]+ → ABCDEF..TXYZ

Z czego się składają wyrażenia regularne - znaki specjalne (5)

W wyrażeniach regularnych mamy grupę znaków specjalnych - znaków, które mają określone znaczenie:

. ? \ | ^ $ + * ( ) { } [ ]

Aby ich użyć w ich ‘normalnym’ - widocznym kontekście, należy zastosować znak \, który znosi specjalne znaczenie następnego znaku:

\. \? \\ \| \^ \$ \+ * \(\) \{ \} \[\]

Np.
[\w\s:]+[\.\?\\\|\^\$\+*\(\)\{\}\[\]]+ → Znaki specjalne w wyrażeniach regularnych to:\?+()[]{}.*
\d+\+\d+=\d+ → 2+2=5 12+123=135

Wyrażenia zachłanne i leniwe

W poniższym przykładzie chcemy wyszczególnić kraje, to których wg testowanego systemu możemy kupić tani bilet lotniczy.

Działanie zachłanne (greedy)

\[.*\]

Do wyboru jest [Gruzja], [Norwegia] lub [Wyspy Owcze]

Zwraca maksymalne pasujące dopasowanie

Działanie leniwe (lazy, ungreedy)

\[.*?\]

Do wyboru jest [Gruzja], [Norwegia] lub [Wyspy Owcze]

Zwraca minimalne pasujące dopasowanie

Grupy nazwane

W wielu wypadkach chcemy podzielić znaleziony wzorzec na części i wykorzystać te części. Możemy do tego celu wykorzystać **grupy nazwane**. Tworzy się je za pomocą:

(?<nazwa_grupy>wyrażenie_regularne)

Np dla daty:

(?<dzien>\d{1,2})-(?<miesiac>\d{1,2})-(?<rok>\d{4})

Dzięki temu możemy wyciągnąć wartość poszczególnych grup posługując się ich nazwą

Dłuższy wzorzec czyli lepszy wzorzec….prawdopodobnie

Bardziej skonkretyzowany wzorzec pozwoli szybciej odrzucić tekst niespełniający wymagań i skróci czas wykonania programu.

Porównajmy ze sobą dwa wyrażenia:

.* (.*)\[(.*)\]:.* **oraz** **[12]\d{3}-[01]\d-[0-3]\d (.*)\[(.*)\]:.***

Oba służa do dopasowania takiego tekstu:

2014-08-26 app[web.1]: 50.0.134.125 - - [26/Aug/2014 00:27:41] "GET / HTTP/1.1" 200 14 0.0005

Ale co jeżeli spróbujemy dopasować tekst, który nie spełnia wymagań?

50.0.134.125 - - [26/Aug/2014 00:27:41] \"GET / HTTP/1.1\" 200 14 0.0005

Nieskończone pętle

Da się skonstruować takie wzorce dla wyrażeń regularnych, które spowodują wykonywanie się nieskończonej pętli.

Przykładowy wzorzec:

(.*)+

Z tego powodu często w wyrażeniach regularnych używa się timeoutów, które kończą szukanie dopasowania po wyznaczonym czasie działania.