



Regular Expressions

by Dylan TAILDEMAN

C'est quoi une RegEx ?

Chaine de caractères décrivant un ensemble de correspondances possibles.

`/-?\d+(\.\d*)?/`

`/#[0-9A-Fa-f]{6}/`

`/(?:2[0-3]|[01]\d|\d):(?:[0-5]\d)/`

`/[A-Za-z0-9\.,;:!?()"'"%\-]+/`

`/^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])\d{4}$/`

`/<([^\>\|]+)>(.*)<\|1>/`

Sh (Terminal)

```
echo «Becode est en Belgique!» > file.txt  
grep --color -E 'Be[a-z]+' file.txt
```

Becode est en Belgique!

Javascript

```
let regex = /Be[a-z]+/g;  
let text = "Becode est en Belgique!";  
let result = text.match(regex);  
  
console.log(result);  
/* ["Becode", "Belgique"] */
```

Python

```
import re  
  
regex = 'Be[a-z]+'  
text = "Becode est en Belgique!"  
result = re.findall(regex, text)  
  
print(result)  
# ['Becode', 'Belgique']
```

PHP

```
<?php  
$regex = '/Be[a-z]+/';  
$text = "Becode est en Belgique";  
preg_match_all($regex, $text, $matches);  
  
print_r($matches);  
// Array ([0] => Becode [1] => Belgique)  
?>
```

Deux types de RegEx ?

BRE (Basic Regular Expression)

VS

ERE (Extended Regular Expression)

BRE: sed, grep, vi

ERE: egrep, grep -E, awk

La différence ?

Certains symboles littéraux et spéciaux seront inversés. Il faudra les échapper en fonction de nos besoins.

Ainsi, dans une BRE:

+ => + littéral
\+ => quantificateur

Alors que dans une ERE:

+ => quantificateur
\+ => + littéral

Composition d'une Regex

Des quantificateurs

Des ancres

`/^(w{3}\.)?[a-z]+(\.[a-z]{2,})+$/`

Des délimiteurs

Des caractères
(ou ensembles de caractères)

Les délimiteurs

- définissent le début et la fin de la RegEx,
- peuvent changer d'un langage de programmation à un autre,
- les plus courants sont l'apostrophe ('), le guillemet (") et le slash (/).

```
/[A-Za-z0-9\.,;:!?()""%\-]+/
```

```
'Be[a-z]+'
```

Les ancres

- indiquent à la RegEx que le motif recherché se trouvent en début (^) ou en fin (\$) de ligne,
- attention à ne pas confondre le symbole d'exclusion [^a-f] avec l'ancre de début de ligne!

```
/^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/d{4}$/
```

```
/^(w{3}\.)?[a-z]+(\.[a-z]{2,})+$/
```

Les caractères (ou ensembles de caractères)

mot	ver[ts]	m[^a]ts	[a-z]	[a-zA-Z0-9]	.	voiture moto
mot motif marmotte	ver de terre vers l'infini feu vert	mets mats mots	a p 9	A a 0	Tous les caractères	voiture moto bus

\d	\w	\t	\smot\s
Tous les chiffres	Equivalent à [a-zA-Z0-9_]	Tabulation	mot motif marmotte

Les quantificateurs

?	*	+	{3}	{3,}	{,8}	{3,6}
0 ou 1	0 ou plus	1 ou plus	Exactement 3	Minimum 3	Maximum 8	Entre 3 et 6

Le groupement

`/USA|B/`

=> **USA**, **B**
x> **USB**

`/US(A|B)/`

=> **USA**, **USB**
x> **B**

La référence

Il est possible de faire référence à un groupe capturé précédemment à l'aide de \1, \2, etc...

`/(A|E)Y\1/`

=> **AYA**, **EYE**
x> **AYE**, **EYA**

La capture

Les parenthèses simples servent à la fois à grouper et à capturer.

Ainsi, selon l'exemple précédent:

La RegEx `/US(A|B)/` appliquée sur le texte «**USA**» renverra:

Une correspondance: '**USA**'

Un groupe capturé: '**A**'

Si on souhaite grouper SANS capturer, on placera alors « **?:** » au début de notre groupe:

`/US(?:A|B)/`

(Cette RegEx ne renverra alors qu'une correspondance.)

Exercices !

/#[0-9A-Fa-f]{6}/

#F841E2
#V84G14
#FFFFFF

/[a-zA-Z0-9_\. -]+@([a-zA-Z0-9_-]+(?:\.[a-zA-Z0-9_-]{2,})+)/

taildeman_dylan@outlook.com
Laurent de la Clergerie
james.bond007@agent.gov.uk
charleroi@becode.org

/<([^\>\|]+)>(.*?)<\|1>/

<html>coucou</html>
<h1></h1>
<h2>Titre raté</h3>



Merci à tous !



/V(((?![*+?])(?:[^\r\n[/\]\.\.|\[(:[^\r\n]\)\]\.\.)*\])+V((?:g(?:im?|mi?)?|i(?:gm?|mg?)?|m(?:gi?|ig?)?)?)/