Notepad++, les expressions régulières

Extrait de nliautaud.fr

Une expression régulière (abrégé expreg), en anglais regular expression (abrégé regexp), est une chaine de caractère permettant de décrire un ensemble variable par l'utilisation d'une syntaxe précise. Cette chaine de caractères est appelée motif, en anglais pattern.

Voir Expression rationnelle.

La maîtrise de cette syntaxe vous permettra une manipulation de textes sans limite dans Notepad++ mais aussi dans la majeure partie des langages de programmation.

Distinguer les motifs et leurs résultats

Il faut bien saisir qu'un motif de recherche vise à chercher des résultats lui correspondant dans un texte, qui peuvent être remplacés selon un motif de remplacement :

- Le motif de recherche décrit quoi chercher dans ce texte, par exemple “mots commençant par un N”.
- Les résultats de la recherche pourront alors être par exemple “Nature, Nounours, Nord”.
- Optionnellement, le motif de remplacement décrit par quoi remplacer chaque résultat.

Syntaxe du motif de recherche

Notepad++ (ou plus précisément sa composante Scintilla) ne prend en compte que la syntaxe de base des expressions régulières, mais permet déjà un très grand nombre d'opérations.

Les caractères

- . : un point désigne n'importe quel caractère.
- tout autre caractère, s'il ne fait pas parti de la syntaxe des expressions régulières, se désigne lui-même.

```
Les expressions régulières dans Notepad++ ne sont sensibles à la casse
que si l'option de recherche Respecter la casse est activée.
```

Exemples

- a.c désigne littéralement “la lettre a, puis n'importe quel caractère, puis la lettre c”. Cela pourrait être abc ou a:c mais pas 123.
- .-5% désigne littéralement “n'importe quel caractère, un tiret, le chiffre 5 puis le symbole pourcent”. Cela pourrait être a-5% mais pas a-6% ou a5%.

Début et fin de ligne

- ^ : l'accent circonflexe désigne le début d'une ligne.
- $: le dollar désigne la fin d'une ligne.

Exemples

- ^a désigne “la lettre a en début de ligne”.
- ^. désigne “n'importe quel caractère en début de ligne”.
- a$ désigne “la lettre a en fin de ligne”.
- ^a$ désigne “la lettre a seule sur une ligne”.

Les répétitions

- * : une étoile indique que le symbole précédent est présent n'importe quel nombre de fois (même 0 fois, qu'il n'est donc pas présent).
- + : un plus indique que le symbole précédent est présent au moins une fois.

Exemples

- ab*c désigne littéralement “la lettre a, n'importe quel nombre de fois la lettre b, puis la lettre c”. Cela peut désigner ac, abc, abbbbbc…
- ab+c désigne littéralement “la lettre a, au moins une fois la lettre b, puis la lettre c”. Cela peut désigner abc, abbc, abbbbbc…

Les ensembles

- […] : un ensemble de caractères entre crochets désigne n'importe quel caractère présent dans cet ensemble.
- [^…] : un ensemble entre crochets commençant par un accent circonflexe désigne n'importe quel caractère qui n'est pas présent dans cet ensemble.
- […-…] : un tiret séparant deux caractères dans un ensemble désigne une plage de caractères.

Exemples

- [abc7] désigne littéralement “les lettres minuscules a, b et c ou le chiffre 7”.
- [^abc7] désigne littéralement “n'importe quel caractère à part les lettres minuscules a, b, c et le chiffre 7”.
- [a-z] désigne littéralement “n'importe quelle lettre minuscule”.
- [a-z0-9] désigne littéralement “n'importe quelle lettre minuscule ou n'importe quel chiffre”.

Les groupes

- (…) : les parenthèses définissent un groupe qui peut être répété ou récupéré dans un motif de

remplacement.

Exemples

- a(bc)+ désigne littéralement “la lettre a puis au moins une fois le groupe de lettres ab”. Cela pourrait être abc, abcbc, abcbcbcbcbc…

Échappement

- Pour désigner un caractère qui est utilisé dans la syntaxe des expressions régulières, comme un crochet [, il faut l'échapper au moyen de l'antislash : \[

Exemples

- a\[bc\] désigne littéralement “la lettre a, puis les lettres b et c entre crochets”. C'est à dire la chaine de caractères a[bc].
- a[\bc] désigne littéralement “la lettre a, puis b ou c ou un antislash”. Cela pourrait être a\, ab ou ac.

Les mots

- \< : désigne le début d'un mot.
- \> : désigne la fin d'un mot.
- \w : désigne un caractère d'un mot. Correspond à [a-zA-Z0-9_].

Exemples

- \w+ désigne littéralement “un mot d'au moins une lettre”.

Syntaxe du motif de remplacement

- Tout caractère se désigne lui-même.
- \x : avec x un chiffre entre 1 et 9 (\1, \2, …) désigne le résultat d'un groupe du motif de recherche.

Exemples

- Dans le motif de recherche a(bc), le motif de remplacement z\1 désigne la lettre z suivie du résultat du premier groupe, donc zbc.
- Dans le motif de recherche (a(bc))(de), \1 désigne abc, \2 désigne bc et \3 désigne de.

Utilisation dans Notepad++

La fenêtre de recherche et remplacement

Lors d'une recherche ou d'un remplacement, il est proposé de changer de Mode de recherche : choisissez Expression Régulière.

Indiquez dans le champ *Recherche* votre motif de recherche, et en cas de remplacement indiquez dans *Remplacer* votre motif de remplacement.

Les plugins utiles

La fenêtre Find/Replace de TextFX est faite pour la manipulation d'expressions régulières, et contiens plusieurs fonctionnalités supplémentaires rendant le travail plus aisé.

Le plugin RegEx Helper est un atout important, car il permet de visualiser directement tous les résultats d'un motif.

Exemples

Insérer du texte en début de ligne

Ouvrir la fenêtre de remplacement avec Ctrl+H, et sélectionner le mode Expressions Régulières.

Rechercher le premier caractère en début de ligne (dans un groupe pour pouvoir le réécrire) avec :

```
^(.)
```

Remplacer par le texte voulu suivi du résultat du groupe, par exemple :

1. \1

Ainsi le texte :

```
abricot
banane
cerise
```

Devient :

1. abricot
2. banane
3. cerise

Insérer du texte en fin de ligne

Ouvrir la fenêtre de remplacement avec Ctrl+H, et sélectionner le mode Expressions Régulières.

Rechercher le dernier caractère en fin de ligne (dans un groupe pour pouvoir le réécrire) avec :

```
(.)$
```

Remplacer par le résultat du groupe suivi du texte voulu, par exemple :

```
\1 :
```

Ainsi le texte :

```
abricot
banane
cerise
```

Devient :

```
abricot :
banane :
cerise :
```

Supprimer des décimales

Ouvrir la fenêtre de remplacement avec Ctrl+H, et sélectionner le mode Expressions Régulières.

Rechercher « un chiffre ou plus suivi d'un point ou d'une virgule puis d'au moins un chiffre » :

```
([0-9]+)[.,][0-9]+
```

L'utilisation d'une parenthèse capturante permet de ne conserver au remplacement que la valeur entière, au moyen de :

```
\1
```

Par exemple :

```
8 1.0 56.82 589,273
```

Devient :

```
8 1 56 589
```

Arrondir des valeurs

La recherche/remplacement ne permet pas d'opérations proprement conditionnelles : Notepad++ n'est pas d'un interpréteur ou un compilateur et ne peut traiter des opérations mathématiques.

```
Il est donc, de manière générale, impossible d'arrondir
automatiquement et facilement des valeurs dans Notepad++.
```

La méthode suivante permet d'arrondir toute valeur positive à l'unité la plus proche. Elle nécessite toutefois un grand nombre d'opérations, augmentant avec la taille des valeurs :

- valeurs comprises entre 0 et 9.9999… : 11 remplacements
- valeurs comprises entre 0 et 99.999… : 21 remplacements
- valeurs comprises entre 0 et 999.99… : 31 remplacements

Il est donc très vite plus intéressant de faire appel à un langage de programmation.

On prendra pour exemple la série suivante :

- 0.0 58.256 982.498
- 4.5 98.663 506.942
- 19.6 59.701 189.842

D'abord, arrondir les valeurs dont la première décimale est inférieure à 5 correspond à en supprimer les décimales. On recherchera donc :

```
([0-9]+)[.,][0-4][0-9]*
```

Pour n'en garder que la valeur entière :

```
\1
```

```
0    58      982           (première décimale inférieure à 5)
4.5  98.663  506.942
19.6  59.701  189.842
```

Ensuite, les valeurs dont la première décimale est comprise entre 5 et 9 doivent être augmentées de un : l'arrondi de 12.71 est 13.

Les valeurs dont le chiffre des unités est inférieur à 9 n'influeront pas le chiffre des dizaines. On peut donc déterminer neuf remplacements à effectuer afin de remplacer le chiffre des unités en conservant les chiffres précédents :

```
([0-9]*)0[.,][5-9][0-9]* remplacé par \11
```

```
([0-9]*)1[.,][5-9][0-9]* remplacé par \12
```

```
([0-9]*)2[.,][5-9][0-9]* remplacé par \13
```

```
([0-9]*)3[.,][5-9][0-9]* remplacé par \14
```

```
([0-9]*)4[.,][5-9][0-9]* remplacé par \15
```

```
([0-9]*)5[.,][5-9][0-9]* remplacé par \16
```

```
([0-9]*)6[.,][5-9][0-9]* remplacé par \17
```

```
([0-9]*)7[.,][5-9][0-9]* remplacé par \18
```

```
([0-9]*)8[.,][5-9][0-9]* remplacé par \19
```

```
0    58      982
5    99      507           (chiffre des unités inférieur à 9)
19.6  59.701  189.842
```

Il ne reste à ce stade plus que les valeurs avec une unité de 9, dont le remplacement doit influer sur les chiffres des dizaines.

Les remplacer demande alors de reprendre le raisonnement précédent : neuf remplacements pour les dizaines comprises entre 0 et 8 en mettant le chiffre des unités à 0.

```
([0-9]*)09[.,][5-9][0-9]* remplacé par \110
```

```
([0-9]*)19[.,][5-9][0-9]* remplacé par \120
```

```
...
```

```
0    58      982
5    99      507
20    60      190          (chiffre des dizaines inférieur à 9)
```

Et l'on recommence les neuf opérations pour les dizaines égales à 9, puis pour les centaines, etc.