BROUILLON

Tout sur les expressions régulières dans Calibre

Les expressions régulières sont souvent utilisées dans Calibre pour manipuler le contenu des e-books et les métadonnées. Ce tutoriel est une introduction à l'utilisation des expressions régulières dans Calibre.

cf. http://manual.calibre-ebook.com/regexp.html

Tout d'abord, un mot d'avertissement pour vous donner du courage.

Cela va être un peu technique car les expressions régulières sont un outil technique pour faire des trucs techniques.

Je vais devoir utiliser un jargon et des concepts qui peuvent sembler compliqués.

Je vais essayer d'expliquer ces concepts aussi clairement que possible, mais ils sont incontournables.

Cela dit, ne soyez pas découragés par le jargon, car j'ai essayé d'expliquer toute nouvelle notion.

Bien que les expressions régulières peuvent sembler obscures, je vous promets qu'elles ne sont pas si compliquées que ça.

Même ceux qui comprennent les expressions régulières ont du mal à lire les expressions les plus complexes, mais leur écriture est plus facile car on construit l'expression étape par étape. Donc, faire un pas et qu'il me suive dans le trou de lapin.

Où utiliser des expressions régulières dans Calibre ?

Il y a dans Calibre quelques endroits où sont utilisées les expressions régulières :

- dans les options de conversion, Recherche et Remplacer
- dans les paramètres d'importation la détection des métadonnées à partir des noms de fichiers
- et lors de l'édition par lot des métadonnées de livres.

L'éditeur de livre de calibre peut également utiliser des expressions régulières dans les fonctions de recherche et remplacement.

Que diable est une expression régulière ?

Une expression régulière est une façon de décrire des ensembles de chaînes.

Une seule expression régulière peut correspondre à des chaînes différentes. C'est ce qui rend si puissantes les expressions régulières qui peuvent décrire de manière concise un grand nombre de variantes.

J'utilise ici le mot chaîne dans le sens utilisé dans les langages de programmation : une chaîne d'un ou plusieurs caractères, caractères (caractères, chiffres, signes de ponctuation et espaces, sauts de ligne, tabulations, etc.

Notez que les caractères majuscules et minuscules sont généralement considérés comme différents.

Dans calibre, les expressions régulières sont insensibles à la casse dans la barre de recherche, mais pas dans les options de conversion.

Quelques explications ?

Tout d'abord, voici le concept le plus important pour les expressions régulières : une chaîne est elle-même une expression régulière qui se correspond. C'est à dire que, si je veux correspondre à la chaîne 'Bonjour tout le monde!' en utilisant une expression régulière, l'expression régulière à utiliser serait 'Bonjour tout le monde !'.

Notez cependant que cela ne correspond qu'à la chaîne exacte 'Bonjour tout le monde!', et non pas par exemple à 'Bonjour le monde!' ni 'Bonjour, monde!' ni toute autre variante.

Et ensuite ?

Nous avons dit que les expressions régulières peuvent correspondre à plusieurs chaînes.

La réalité est un peu compliquée. Supposons, par exemple, que l'e-book à convertir a dans le pied de page un compteur de pages gênant, comme 'Page 5 of 423'. Le numéro de page va de 1 à 423, ce qui obligerait à correspondre à 423 chaînes différentes ? En fait, non : les expressions régulières permettent de définir des jeux de caractères qui correspondent.

Pour définir un jeu, vous mettez tous les caractères du jeu entre crochets.

Par exemple, l'ensemble [abc] correspond au caractère 'a', 'b' ou 'c'.

Les jeux de caractères ne correspondent qu'à l'un des caractères du jeu.

Ils acceptent les plages de caractères, par exemple,

- pour faire correspondre tous les caractères minuscules, utilisez [az]
- pour les caractères minuscules et majuscules, [a-zA-Z]
- et ainsi de suite.

Evidemment, en utilisant l'expression 'Page [0-9] sur 423' vous pouvez faire correspondre les 9 premières pages.

En mettant 'Page [0-9][0-9] sur 423', vous repérez tous les numéros de page à deux chiffres, etc.

On peut répéter un caractère ou un jeu de caractère avec des caractères spéciaux dits jokers ou quantificateurs, '+', '?' et '*', qui répètent le seul élément qui les précède. (par élement, nous disons un caractère unique, un jeu de caractères, une séquence d'échappement ou un groupe (nous verrons cela plus loin)

?
: correspond à 0 ou 1 fois l'élément précédent

*
: correspond à 0 ou plus élément précédent

+
: correspond à 1 ou plus de l'élément précédent.

Quelques exemples :

a?
: correspond à “” (une chaîne vide) ou à “a”

a*
: correspond à “”, “a”, “aa” ou un nombre quelconque de a dans une ligne

a+
: correspondrait à “a”, “aa” ou un nombre quelconque de a dans une ligne mais pas à la chaîne vide.

Même chose pour les ensembles: L'expression [0-9]+ correspond à n'importe quel nombre entier !

Dans le cas de nos numéros numéros de page, l'expression “Page [0-9]+ sur 423” correspondrait à chaque numéro de page du livre!

Une note sur ces quantificateurs :

Ils essaient généralement de correspondre à autant de texte que possible, soyez donc prudent dans leur utilisation. Ceci est appelé “greedy behaviour”.

Cela devient problématique lorsque, par exemple, vous essayez de faire correspondre une balise.

Soit par exemple la chaîne “ Title here ” et supposons que vous voulez faire correspondre la balise d'ouverture (la partie entre la première paire de crochets, nous détaillerons plus loin).

L'expression “<p.*>” correspond à la totalité de la chaîne ! (Le caractère “.” est un autre caractère spécial. Il correspond à tout sauf les sauts de ligne, de sorte que l'expression “.*” correspond à une ligne unique. Si vous essayez “<p.*?>” qui empêche le quantificateur '*' de s'étendre. Cette expression correspond à la première balise d'ouverture, comme prévu.

Un autre moyen est l'expression “<p[^>]*>” qui correspond à la même balise d'ouverture.

Et pour correspondre à un point ou un point d'interrogation ?

En mettant un antislash devant un caractère spécial, il est interprété comme le caractère littéral, sans aucune signification particulière.

Cette paire d'une barre oblique inverse suivie d'un seul caractère est appelé une séquence d'échappement, et l'acte de mettre une barre oblique inverse devant un caractère spécial est appelé échapper ce caractère.

Une séquence d'échappement est interprétée comme un seul élément.

D'autres séquences font plus que simplement l'échappement des caractères spéciaux, par exemple “\t” désigne une tabulation.

Nous verrons quelques séquences d'échappement plus loin.

Pour les caractères spéciaux : considérez que tout caractère spécial doit être échappé si vous voulez le caractère littéral.

Alors, quels sont les ensembles les plus utiles ?

Certains ensembles jeux sont utiles :

[0-9]
: correspond à un numéro unique

[az]
: correspond à une seule lettre minuscule

[AZ]
: correspond à une seule lettre majuscule

[a-zA-Z]
: correspond à une seule lettre

[a-zA-Z0 -9]

correspond à une seule lettre ou un chiffre

Vous pouvez également utiliser une séquence d'échappement comme raccourci:

\d
: équivaut à [0-9]

\w
: équivaut à [a-zA-Z0-9_]

\s
: équivaut à n'importe quel espace

Note

“espace” veut dire ici espace, tabulation, saut de ligne, saut de page et retour chariot.

Dernière note sur les sets :

Vous pouvez également définir un ensemble qui signifie tout autre élément que ceux de l'ensemble.

Pour cela, mettez en tête du set le caractère “^”

Ainsi, [^a] correspond à tout caractère sauf “a”.

On appelle cela complémenter l'ensemble.

Les raccourcis de séquences d'échappement vus précédemment peuvent également être complémentés, en utilisant la majuscule correspondante.

Ainsi, : “\D” désigne tout caractère non-numérique, équivalent à [^0-9].

Donc, pour en revenir à l'exemple <p[^>]*> de la section précédente, vous voyez que le jeu de caractères qu'il utilise fait correspondre à tout caractère sauf une équerre de fermeture.

Et pour faire correspondre plusieurs chaînes différentes ?

Considérons cet exemple. Le livre vous convertissez a “Titre” écrit sur les pages impaires et “auteur” sur les pages paires.

Vous pouvez grouper des expressions entières entre parenthèses normales, et le caractère “|” permet de correspondre à l'expression de droite ou de gauche.

D'abord, regroupons les expressions pour les pages paires et impaires, obtenant ainsi (Titre)(Auteur) comme les deux expressions qu'il nous faut.

Maintenant, simplifions avec la barre verticale “|” : si vous utilisez l'expression (Titre|Auteur), vous obtenez soit une correspondance avec “Titre” (sur les pages impaires) ou avec “Auteur” (sur les

pages paires).

Vous pouvez utiliser la barre verticale sans parenthèses de regroupement.

L'expression “Titre|Auteur” correspond soit à la chaîne “Titre” ou à la chaîne “auteur”, comme l'exemple ci-dessus en utilisant le regroupement. La barre verticale permet de choisir entre l'expression précède la barre et celle qui la suit.

Donc, si vous voulez faire correspondre les chaînes “Calibre” et “calibre” et que vous voulez sélectionner seulement entre les “c” majuscules et minuscules, vous devriez utiliser l'expression (c|C)alibre, où le groupement assure que seul le “c” est choisi. Si vous utilisiez “c|Calibre”, vous obtiendrez une correspondance avec la chaîne 'c' ou la chaîne 'Calibre', ce qui n'est pas ce que nous voulons.

You missed...

Si vous avez un groupe que vous avez déjà fait correspondre, vous pouvez utiliser des références à ce groupe plus loin dans l'expression : Les groupes sont numérotés à partir de 1, et vous y faire référence en échappant le numéro du groupe que vous souhaitez référencer, donc le cinquième groupe serait référencé \5. Donc, si vous avez recherché ([^]+) \1 dans la chaîne “Test Test”, vous souhaitez faire correspondre l'ensemble de la chaîne!

Et pour rendre une expression régulière insensible à la casse ?

Vous pouvez utiliser des drapeaux avec la construction spéciale (?drapeau) en remplaçant “drapeau” par les drapeaux spécifiques que vous voulez.

Pour ignorer la casse, le drapeau est i, donc vous incluez (?i) dans votre expression. Ainsi, test(?i) correspondrait à “Test”, “tEst”, “TEst” et toutes les variations.

Le drapeau s fait correspondre le point à tout caractère y compris la nouvelle ligne

Pour utiliser plusieurs drapeaux dans une expression, il suffit de les mettre dans la même instruction : (?is) ignore la casse et fait que le point correspond à tout.

L'ordre des drapeaux n'importe pas, (?si) serait équivalent à ce qui précède.

De bons endroits pour placer des drapeaux dans votre expression sont soit le début soit la fin. Ainsi,ils ne se mélangent pas avec autre chose.

Conversions

Let’s begin with the conversion settings, which is really neat. In the Search and Replace part, you can input a regexp (short for regular expression) that describes the string that will be replaced during the conversion. The neat part is the wizard. Click on the wizard staff and you get a preview of what calibre “sees” during the conversion process. Scroll down to the string you want to remove, select and copy it, paste it into the regexp field on top of the window. If there are variable parts, like page numbers or so, use sets and quantifiers to cover those, and while you’re at it, remember to escape special

characters, if there are some. Hit the button labeled Test and calibre highlights the parts it would replace were you to use the regexp. Once you're satisfied, hit OK and convert. Be careful if your conversion source has tags like this example:

Maybe, but the cops feel like you do, Anita. What's one more dead vampire? New laws don't change that.   Generated by ABC Amber LIT Conv erter, http://www.processtext.com/abclit.html It had only been two years since Addison v. Clark. The court case gave us a revised version of what life was

(shamelessly ripped out of this thread). You'd have to remove some of the tags as well. In this example, I'd recommend beginning with the tag , now you have to end with the corresponding closing tag (opening tags are <tag>, closing tags are </tag>), which is simply the next in this case. (Refer to a good HTML manual or ask in the forum if you are unclear on this point.) The opening tag can be described using <b.*?>, the closing tag using , thus we could remove everything between those tags using <b.*?>.*?. But using this expression would be a bad idea, because it removes everything enclosed by - tags (which, by the way, render the enclosed text in bold print), and it's a fair bet that we'll remove portions of the book in this way. Instead, include the beginning of the enclosed string as well, making the regular expression <b.*?>\s*Generated\s+by\s+ABC\s+Amber\s+LIT.*? The \s with quantifiers are included here instead of explicitly using the spaces as seen in the string to catch any variations of the string that might occur. Remember to check what calibre will remove to make sure you don't remove any portions you want to keep if you test a new expression. If you only check one occurrence, you might miss a mismatch somewhere else in the text. Also note that should you accidentally remove more or fewer tags than you actually wanted to, calibre tries to repair the damaged code after doing the removal. Adding books

Another thing you can use regular expressions for is to extract metadata from filenames. You can find this feature in the "Adding books" part of the settings. There's a special feature here: You can use field names for metadata fields, for example (?P<title>) would indicate that calibre uses this part of the string as book title. The allowed field names are listed in the windows, together with another nice test field. An example: Say you want to import a whole bunch of files named like Classical Texts: The Divine Comedy by Dante Alighieri.mobi. (Obviously, this is already in your library, since we all love classical italian poetry) or Science Fiction epics: The Foundation Trilogy by Isaac Asimov.epub. This is obviously a naming scheme that calibre won't extract any meaningful data out of - its standard expression for extracting metadata is (?P<title>.+) - (?P<author>[^_]+). A regular expression that works here would be [a-zA-Z]+: (?P<title>.+) by (?P<author>.+). Please note that, inside the group for the metadata field, you need to use expressions to describe what the field actually matches. And also note that, when using the test field calibre provides, you need to add the file extension to your testing filename, otherwise you won't get any matches at all, despite using a working expression. Bulk editing metadata

The last part is regular expression search and replace in metadata fields. You can access this by selecting multiple books in the library and using bulk metadata edit. Be very careful when using this last feature, as it can do Very Bad Things to your library! Doublecheck that your expressions do what you want them to using the test fields, and only mark the books you really want to change! In the regular expression search mode, you can search in one field, replace the text with something and even write the result into another field. A practical example: Say your library contained the books of Frank Herbert's Dune series, named after the fashion Dune 1 - Dune, Dune 2 - Dune Messiah and so on. Now you want to get Dune into the series field. You can do that by searching for (.*?) \d+ - .* in the title field and replacing it with \1 in the series field. See what I did there? That's a reference to the first group you're replacing the series field with. Now that you have the series all set, you only need to

do another search for .*? - in the title field and replace it with “” (an empty string), again in the title field, and your metadata is all neat and tidy. Isn’t that great? By the way, instead of replacing the entire field, you can also append or prepend to the field, so, if you wanted the book title to be prepended with series info, you could do that as well. As you by now have undoubtedly noticed, there’s a checkbox labeled Case sensitive, so you won’t have to use flags to select behaviour here.

Well, that just about concludes the very short introduction to regular expressions. Hopefully I’ll have shown you enough to at least get you started and to enable you to continue learning by yourself- a good starting point would be the Python documentation for regexps.

One last word of warning, though: Regexps are powerful, but also really easy to get wrong. calibre provides really great testing possibilities to see if your expressions behave as you expect them to. Use them. Try not to shoot yourself in the foot. (God, I love that expression…) But should you, despite the warning, injure your foot (or any other body parts), try to learn from it.

Pré-requis

Première étape

Autres étapes

Conclusion

Problèmes connus

Voir aussi

Basé sur Tout à propos de l’utilisation des expressions régulières dans calibre par calibre.