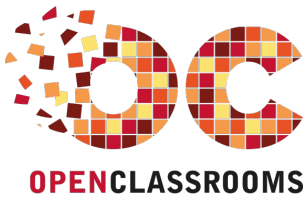
Créer des documents avec txt2tags

Par farvardin



www.openclassrooms.com

Dernière mise à jour le 3/08/2010

Sommaire

Créer des documents avec txt2tags

Par  farvardin

Mise à jour : 03/08/2010
Difficulté : Intermédiaire Durée d'étude : 2 heures

Txt2tags est une syntaxe wiki, un générateur de documents et un pré-processeur de textes très pratique, écrit par *Aurélio Marinho Jargas*.

Ce logiciel ressemble à ce qu'on peut avoir avec reStructuredText, la syntaxe MediaWiki (celle utilisé par wikipedia) ou à Markdown, mais en plus puissant grâce à son système de macros utilisant pré-processeur et post-processeur, permettant de personnaliser et d'étendre à l'infini la syntaxe de base de txt2tags. C'est donc une véritable alternative à ces projets, mais aussi un outil qui va bien au-delà des fonctionnalités de ce qu'on peut trouver ailleurs.

Txt2tags permet d'ailleurs d'exporter dans de nombreux autres langages à balises :
HTML, XHTML, SGML, LaTeX, Lout, Man page, Wikipedia / MediaWiki, Google Code Wiki, DokuWiki, MoinMoin, MagicPoint, PageMaker, Plein texte (texte brut)

Ainsi dans les utilisations possibles de txt2tags, on trouve la génération de pages internet statiques (et valides W3C), mais également la possibilité de générer des pages PHP, la création d'articles ou de livres en LaTeX destinés à être exportés en pdf ou imprimés, des documentations devant être converties dans divers formats... c'est aussi pour cela qu'un tutoriel complet nous semble nécessaire pour en appréhender les concepts et l'utilisation.

Multiplateforme, de par son code composé d'un unique script python, il a également été empaqueté pour de nombreuses distributions Linux, pour Mac OS X, et pour Windows, ce qui permet de l'installer rapidement au coeur de son ordinateur.

Nous apprendrons donc dans ce tutoriel comment installer txt2tags et en faire une utilisation basique, puis nous verrons quelques exemples concrets d'utilisation avancée des directives de remplacement, pour réellement exploiter toute la puissance de cet étonnant outil multi-fonctions.
Sommaire du tutoriel :

Présentation de txt2tags

On peut tout d'abord se demander ce qui peut nous pousser à encourager l'utilisation de txt2tags. Qu'est-ce que ça cache ?

Premièrement, c'est un outil tellement pratique qu'on ne comprend pas pourquoi il n'est pas plus connu. Ainsi on veut en faire profiter le maximum d'utilisateurs potentiels, car il gagne vraiment à être découvert. L'essayer, c'est l'adopter !

Ensuite, on se dit que si cet outil et sa syntaxe se généralisent et se démocratisent, cela voudra dire également que l'on trouvera par la suite de plus en plus de projets et de logiciels annexes compatibles avec txt2tags, ce qui profitera donc à tout le monde en retour, avec un meilleur support de la syntaxe dans les éditeurs de texte, dans les projets type wordpress ou autre CMS. Même le site du zéro ne propose pas le code txt2tags dans les exemples de code, aussi ce tutoriel restera un peu gris...

Certains peuvent également s'interroger : "mais pourquoi encore un nouveau langage de balise alors qu'il existe déjà Markdown ou MediaWiki ?". Et pourtant, txt2tags fait figure de vétéran car il existe déjà depuis 2001 : la première version étant sortie le 26 juillet 2001 !

Le concept de txt2tags

Bénéfices

Txt2tags, c'est 3 aspects (et avantages) à la fois :

- Une syntaxe simple et dépouillée, de type wiki, plus agréable à utiliser que celle de html, LaTeX ou même wikipedia. Quelques règles à apprendre permettant de traiter la plupart des cas souhaitées en publication (**gras**, *italiques*, souligné, listes à puce...), et leur expressivité même rend un code source écrit en txt2tags lisible et utilisable tel quel.

- La possibilité d'exporter dans les formats de création de documents et de publication les plus courants, ainsi que d'ajouter facilement de nouveaux formats d'exportation à la liste déjà grande des formats supportés.

- Son système de macros peut lever n'importe quelle limitation de la syntaxe initiale et l'étendre quasiment à l'infini, on n'est donc jamais bloqué dans son expression avec txt2tags.

S'il n'y avait que les deux premiers points, cela aurait été déjà très bien, mais txt2tags n'aurait pas été aussi génial sans le dernier aspect évoqué, en effet, en ce qui concerne ce système de macros avec le pré-processeur et post-processeur, il permettra de remplacer les parties de code que l'on souhaite, y compris en utilisant des expressions régulières complexes.

Le tout est servi dans un soucis de simplicité, propre à la philosophie KISS.

Critiques

Si j'avais quelques critiques à faire envers txt2tags, c'est que :

- lorsqu'on fait un seul retour à la ligne, il n'est pas pris en compte : il faut passer 1 ligne pour faire un saut de ligne, mais il n'est pas possible de base d'afficher un simple retour à la ligne sans sauter une ligne et sans rajouter de marque spéciale (type
 ou autre). Malgré tout il est possible avec une directive du préprocesseur (postproc(xhtml): "$" '
') de modifier le comportement par défaut, mais étant donné que txt2tags n'est pas prévu pour cela, on peut obtenir des résultats parfois non souhaités.

- l'insertion de texte de type verbatim (c'est à dire de parties qui devront figurer telles quelles, qui ne seront pas interprétées) n'est pas toujours très pratique, du fait des possibilités avancées de txt2tags : en effet, si on demande au postprocesseur de remplacer une portion de texte lors de l'exportation vers du html, même si on a placé cette portion dans les marques adéquates (entre ``` ou entre """), cela sera quand même modifié par le postprocesseur, en revanche le formatage de base ne sera pas interprété.

- Les directives du pre/post-processeur ne fonctionnent que sur une seule et même ligne, on ne peut pas faire un filtre qui prend en compte en entrée les retour à la ligne. Néanmoins on peut bien entendu rajouter des retours à la ligne après le passage dans un filtre pre/post-processeur (soit avec \n, soit avec
 etc.).

Malgré tout, sachez bien que l'on peut faire presque tout avec txt2tags, donc ce qui peut sembler une limitation pourra toujours être contourné à un moment ou un autre par votre esprit astucieux ! 

- La dernière critique, c'est que comme c'est implémenté en python, cela n'est pas aussi accessible que si c'était en php, pour une utilisation en ligne chez les hébergeurs traditionnels.

Mais voyons comment l'installer sur notre ordinateur maintenant.

Installer txt2tags

Windows

Pour utiliser txt2tags sous Microsoft Windows, à moins que vous ne l'ayez déjà, il vous faudra installer Python depuis le site http://www.python.org/
Il s'agit d'un langage de programmation et de bibliothèques permettant de faire tourner le script de txt2tags.

La source de txt2tags étant dans une archive tar.gz pas très pratique à manipuler depuis windows, nous vous conseillons de simplement récupérer le code txt2tags à cette adresse http://txt2tags.googlecode.com/svn/trunk/txt2tags et de le sauvegarder sous le nom txt2tags.py

En double cliquant sur ce fichier, vous accéderez à son interface graphique, mais vous pourrez également l'utiliser dans un fichier batch (.bat) ou en ligne de commande (cmd.exe).

Linux/BSD

Sous Linux ou BSD, vous avez déjà python d'installé, et txt2tags est prévu dans la plupart des dépôts des distributions les plus courantes, aussi sur la majorité des systèmes vous n'aurez qu'à faire :

- Fedora : *yum install txt2tags*
- Ubuntu, Debian : *sudo apt-get install txt2tags*
- Archlinux : *pacman -S txt2tags*
- FreeBSD : *pkg_add -r txt2tags*

Mac OS X

Il existe un paquet disponible ici : http://txt2tags.sourceforge.net/download.html *(le créateur de txt2tags utilise d'ailleurs Mac OS X régulièrement.)*

Ce paquet permet d'installer le script txt2tags ainsi que le manuel et les exemples, mais il n'utilise pas de lanceur adapté pour Mac OS X, il faudra donc passer par le terminal pour l'utiliser. Il existe malgré tout une interface graphique, que nous verrons bientôt.

Autres systèmes d'exploitation

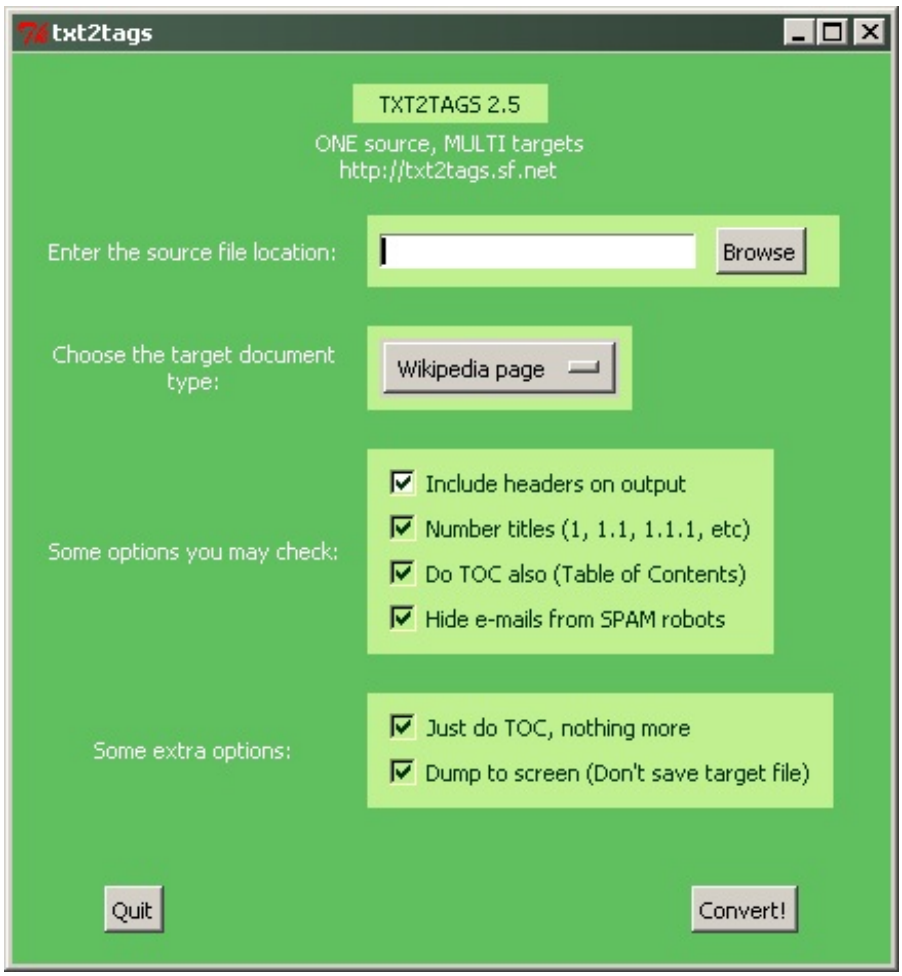
De façon générale, txt2tags étant un simple script python composé d'un seul fichier, vous pouvez récupérer l'archive (qui contient en plus le manuel, des outils, colorations syntaxique et des exemples) ici : http://txt2tags.sourceforge.net/download.html.

La dernière version en cours de développement de ce script est directement téléchargeable ici : http://txt2tags.googlecode.com/svn/trunk/txt2tags.

Il est enfin possible d'utiliser txt2tags directement en ligne, depuis son site internet :
http://txt2tags.sourceforge.net/online.php

Utilisation de base

Avec l'interface graphique



Cette interface graphique rudimentaire et optionnelle fonctionne sur toutes les plateformes, du moment que le module tkinter de python est installé. Nous ne nous étendrons pas dessus, il suffit juste de choisir le fichier à convertir, les options et le format d'exportation souhaités. Dans le cas où l'entête contient déjà des directives sur les options et le format d'exportation, ils seront déjà prédéfinis dans l'interface.

En ligne de commande

L'utilisation de la ligne de commande est bien plus rapide pour traiter rapidement des conversions de documents.

Sous Windows

Un petit fichier nommé convertir.bat contenant le code :

Code : Autre

```
python txt2tags.py mon_document.t2t
```

permettra de générer directement votre document sous windows.

Vous pouvez également créer un convertisseur automatique, qui vous permettra de glisser-déposer un document txt2tags sur le fichier .bat, et de le convertir au format souhaité (défini dans son entête, comme on verra juste ensuite).

Le code du fichier batch, à adapter selon votre cas, sera :

Code : Autre

```
Set CURRENTDIR=%CD%
python.exe "C:\Documents and Settings\Utilisateur\Mes documents\Chemin_vers_txt2t
pause
```

Sous Unix (Linux, BSD, Mac OS X)

Nous vous recommandons d'utiliser un makefile, bien qu'il soit tout à fait possible d'invoquer simplement une conversion avec la commande :

Code : Autre

```
txt2tags mon_document.t2t
```

Un makefile simple pourra être de cette forme :

Code : Autre

```
DOCUMENT = mon_document

TXT2TAGS = txt2tags

html:
          $(TXT2TAGS) -t xhtml --toc --
outfile $(DOCUMENT).html $(DOCUMENT).t2t

pdf:
          $(TXT2TAGS) -t tex --
outfile $(DOCUMENT).tex $(DOCUMENT).t2t
          -pdflatex -interaction batchmode $(DOCUMENT).tex
```

make html générera donc le document html, tandis que *make pdf*, exportera en fichier .tex puis en pdf via la commande pdflatex.

Syntaxe

La structure du document

À de rares exceptions près, les documents txt2tags sont structurés sous cette forme :

- Entêtes (3 lignes contenant le titre, l'auteur, et éventuellement un sous-titre)
- Configuration (options)
- Corps du text

Ce qui donne :

Code : Autre

```
Le titre du document
L'auteur
Dernière mise à jour (ligne optionnelle) : %%mtime(%c)

%!target  : html
%!style   : mon_style.css
%!encoding: utf-8
%!options : --toc --outfile mon_document.html


Ceci est le début de mon document, le corps de texte.
Bonne continuation avec txt2tags...
```

Les balises

Dans txt2tags, la plupart des balises sont créées avec des symboles encadrant le texte, ce qui est très pratique. De plus, toutes les balises pour la mise en forme de base (gras, souligné, barré, italique) vont toujours par paire de 2 symboles.

Mettre en forme :

D'autres wiki concurrents utilisent pour toutes les mises en forme un symbole unique (par exemple l'apostrophe), mais répétés de façon différente à chaque fois, par exemple :

'''gras (pas valable pour txt2tags)''' et ''italique (pas valable non plus pour txt2tags)''

'''''Gras et italique (heureusement pas utilisé dans txt2tags)'''''

Des choses amusantes se passent si on utilise en plus de vrais apostrophes dans son texte, cela devient rapidement l'''anarchie''... Et dire que cette syntaxe est utilisée sur un des sites les plus consultés au monde...

La règle est donc que dans txt2tags, tout ce qui est ****gras****, __souligné__, --barré-- ou //*italique*// se trouve toujours entre 2 paires de symboles.

Hiérarchiser > Les titres :

On a également d'autres wiki qui utilisent pour la mise en forme des entêtes un soulignement du texte, par exemple :

Exemple de titre niveau 1 que l'on ne trouvera pas dans txt2tags
==========

Et le titre niveau 2 qui va avec (pas valable non plus pour txt2tags)
---

ou encore en précédant uniquement le titre avec des symboles d'un seul côté :

++ Titre niveau 2 (pas valable pour txt2tags)
+++ Titre niveau 3 (pas valable pour txt2tags)

Cela rend malheureusement les documents utilisant cette syntaxe wiki plus difficiles à analyser syntaxiquement (pour modifier du texte lors de l'exportation par exemple).

Évoquons rapidement certains qui font cela à l'envers :

====== Titre 1 (pas avec txt2tags) ======
===== Titre 2 (pas avec txt2tags) =====

Cela peut sembler plus pratique pour mieux voir un titre (il a plus d'importance, donc il prend plus de signes), mais ce n'est pas très logique ni facile à retenir (titre 1 c'est 6 ou 5 signes déjà ?)

Dans txt2tags, le nombre de = pour qualifier un titre est en fonction du niveau de ce titre (titre niveau 2 implique 2 signes *égal* de chaque côté du titre). Il est également possible de rajouter un nom à la suite de ce titre, entre crochets et collé au dernier signe =, et cela sera utilisé pour la création d'ancres pour les liens nommés, par exemple un titre simple == Titre niveau 2 == pourra être indiqué == Titre niveau 2 : la compilation du code ==[compil]

Hiérarchiser > Les listes :

Les listes dans txt2tags dérogent à la règle des symboles qui entourent le formatage, mais les éléments de celles-ci on rarement besoin d'être reformatés avant d'être exportés, et au pire des cas on peut toujours faire précéder et suivre la liste entière avec des marquages spéciaux si nécessaire. On débute chaque élément par un tiret (-) ou un plus (+), selon que l'on souhaite une liste non ordonnée ou ordonnée. On peut rajouter des tirets avec un décalage d'un espace pour créer des sous-éléments.

Lier au monde extérieur :

Enfin, pour les liens, ceux-ci suivent tout le temps la partie à mettre en lien, et là aussi c'est plus pratique et logique lors de la relecture, car au lieu de [http://fr.wikipedia.org/wiki/Utilisabilit%C3%A9 buter sur un lien comme sur certains wiki], d'aller au bout du lien, et de continuer la lecture pour voir ce à quoi il se rapporte, on a directement le texte, [dans une forme immédiatement accessible http://fr.wikipedia.org/wiki/Utilisabilit%C3%A9], même si on rajoute des choses après ce lien.

Autres possibilités (images, tableaux) :

Il est possible de rajouter des images (nom du fichier entre crochets, comme un lien, par exemple [zozor.png]), d'entrer du texte tel quel (verbatim ou raw text), entre `` ou "".

Pour les tableaux, cela se fait à l'aide du symbole "pipe" |, comme ceci :

Code : Autre

```
|| Titre de colonne  | Titre de l'autre colonne |
| contenu de tableau |   encore du texte        |
| lorem ipsum        |  etc...                  |
| etc...             |  etc...                  |
```

Il existe encore d'autres subtilités, mais vous devriez plutôt vous référer au manuel complet en ligne pour cela, car cela dépasse le cadre de ce tutoriel qui se focalise plutôt sur la création de documents.

Mémo

Voici donc un résumé des points les plus important de la syntaxe :

- **gras** : **gras**
- //italique// : *italique*
- __soulignage__ : soulignage
- --barré-- : ~~barré~~
- = titre = (à différents niveaux, on a ainsi == titre 2 ==, === titre 3 === etc)
- - liste
- + liste numérotée
- ``code``
- [image.jpg]
- [lien vers un site http://www.site.com] : lien vers un site
- | table |

Premier exemple de remplacements

On peut d'ailleurs même tester txt2tags directement depuis son site internet.

Copiez donc ce code dans le formulaire en ligne http://txt2tags.sourceforge.net/online.php :

Code : Autre

```
%!preproc: 'MYBESTFRIEND' 'Nathalie'

Ce matin j'ai fait la rencontre de MYBESTFRIEND en me baladant dans le parc.
MYBESTFRIEND y faisait du footing.
```

Le résultat sera donc :

Code : Console

```
Ce matin j'ai fait la rencontre de Nathalie en me baladant dans le parc. Nathalie y
```

Ici je mets exprès de côté le formatage en gras, italique, souligné, qui est facile à comprendre et que vous pouvez tester avec la page par défaut du formulaire en ligne.

Ainsi on a vu la manière la plus basique de créer un remplacement de texte. Si dans un document on a de nombreuses références à un mot (voire même un lien internet) appelé à changer ou à être modifié par la suite, on peut créer cette sorte de variable pour la remplacer au dernier moment.

Il n'est pour le moment pas possible d'entrer des lettres accentuées dans ce formulaire de test, c'est un bug dans la conception de la page, et non pas dans txt2tags.

L'indication preproc signifie que le texte sera remplacé avant tout autre remplacement habituel de la syntaxe txt2tags (avant d'être converti en html par exemple), tandis que postproc permet de remplacer le texte après qu'il ait été converti par txt2tags. En utilisant ces 2 possibilités, cela permet de jongler avec tous les cas de figure possible, comme on va le voir ensuite.

Le schéma de remplacements de txt2tags est donc :

tous les remplacements preproc (dans l'ordre d'apparition, et ligne par ligne)
⇓
conversion html (ou tex, ou mediawiki etc)
⇓
tous les remplacements postproc (dans l'ordre d'apparition, et ligne par ligne)

Maintenant, on va par exemple vouloir remplacer un mot, mais également l'entourer avec un effet de texte (titre visible lorsqu'on passe la souris au-dessus), et cela uniquement lors d'une exportation en html (on indique donc html entre parenthèses après le preproc). Pour cela, tapez :

Code : Autre

```
%!preproc(html): 'MYBESTFRIEND' <span title="Oui, MYBESTFRIEND c'est ma meilleure

%!preproc: 'MYBESTFRIEND' 'Natacha'

Ce matin j'ai fait la rencontre de MYBESTFRIEND en me baladant dans le parc.
MYBESTFRIEND y faisait du footing.
```

Petit problème, car on obtient maintenant :

Code : Console

```
Ce matin j'ai fait la rencontre de <span title="Oui, Natacha c'est ma meilleure ami
en me baladant dans le parc.
<span title="Oui, Natacha c'est ma meilleure amie !">Natacha</span> y faisait du fo
```

En effet, dans ce que l'on a généré, txt2tags a compris que l'on voulait vraiment obtenir l'affichage des caractères < et >, et il a donc remplacé < par < et > par >

On pourrait donc résoudre cela simplement en indiquant une nouvelle balise qui signifierait "je veux que ce qui se trouve à l'intérieur de cette balise soit remplacé par un titre dans un span". Mais cela fait appel à des expressions régulières que je ne veux pas encore expliquer pour le moment.

On peut donc plus simplement remplacer après coup tous les symboles d'affichage html < et > par le véritable symbole < et >, en utilisant la directive postproc.

Malgré tout c'est un peu moins rigoureux comme méthode, c'est à dire qu'en 'bidouillant' cela, cela peut se retourner contre nous à un moment ou un autre, si par exemple on veut justement afficher un < ou > à un autre endroit du texte.


Mais dans la majorité des cas, cette astuce va quand même fonctionner correctement.

Code : Autre

```
%!preproc(html): 'MYBESTFRIEND' <span title="Oui, MYBESTFRIEND c'est ma meilleure

%!preproc: 'MYBESTFRIEND' 'Natacha'

%!postproc(html): '&lt;' '<'
%!postproc(html): '&gt;' '>'

Ce matin j'ai fait la rencontre de MYBESTFRIEND en me baladant dans le parc.
MYBESTFRIEND y faisait du footing.
```

Ainsi avec le nouveau postproc (qui ne touche ici que l'exportation html), lorsque tout a été remplacé dans la conversion html, txt2tags passe encore une fois pour faire les derniers remplacements.

Pour information, voici la méthode avec des expressions régulières (regex), que nous verrons en détail au chapitre suivant :

Code : Autre

```
%!preproc: 'MYBESTFRIEND' 'Natacha'
%!postproc: '@@titre@@([^ ].*?
)@@/titre@@' <i><span title="Oui, c'est ma meilleure amie !">\1</span></i>

Ce matin j'ai fait la rencontre de @@titre@@MYBESTFRIEND@@/titre@@ en me baladant
@@titre@@MYBESTFRIEND@@/titre@@ y faisait du footing.
```

Ici on a donc créé les nouvelles balises :

Code : Autre

```
@@titre@@ et @@/titre@@
```

qui servent à définir notre nouvelle syntaxe. J'utilise souvent ces 2 arobases (@) lorsque je veux créer de nouvelles balises, car cela a l'avantage de ressortir du reste du texte tout en restant facile et rapide à écrire. On aurait pu l'indiquer entre < > mais on se retrouvait dans le cas de figure évoqué plus haut, avec le remplacement des < > par '<' et '>'

Mais cessons donc nos balades et courses dans le parc, et revenons à notre sujet.

Utilisation avancée

L'utilisation de base de txt2tags permet de rédiger un grand nombre de documents, mais on peut également vouloir le maîtriser pour créer des choses encore plus complexes, que cela soit pour utiliser dans un site internet ou un document LaTeX final. Nous verrons ainsi dans cette section des astuces supplémentaires pour étendre les possibilités de txt2tags.

Les expressions régulières

Ce qu'il est bon de savoir, c'est que txt2tags peut et doit abuser des expressions régulières (regex), son auteur Aurélio ayant d'ailleurs écrit des outils et des livres sur le sujet.

Je vous renvoie à ce tutoriel sur le sujet pour en apprendre plus sur les expressions régulières.

Le premier type de regex que l'on peut utiliser, c'est ce que l'on a vu lors de notre footing plus haut dans ce tutoriel, pour simplifier :

Code : Autre

```
%!postproc: 'ABC([^ ].*?)JKL' 'ZZZ\1YYY'
```

Ce qui signifie que n'importe quel texte (chaîne de caractères, nombres...), symbolisé par le bloc ([^].*?), se trouvant entre les chaînes de caractères ABC et JKL (sans retour à la ligne), sera maintenu : on replacera cette partie de texte filtrée par ([^].*?), dans la seconde partie symbolisée par le \1, mais celui-ci sera inséré à la place entre les chaînes ZZZ et YYY.

Ainsi dans cet exemple **ABC***DEFG***JKL** sera remplacé par **ZZZ***DEFG***YYY**.

On pourra même avoir plusieurs remplacements successifs, par exemple :

Code : Autre

```
%!postproc: 'ABC([^ ].*?)JKL([^ ].*?)PLOUM' 'ZZZ\1YYY\2BOOOM'
```

ABC*DEFG***JKL***MNO***PLOUM** deviendra **ZZZ***DEFG***YYY***MNO***BOOOM**

On peut bien entendu inverser le placement des \1 et \2 selon l'ordre que l'on veut donner aux remplacements. Le site de txt2tags donne quelques exemples intéressants de remplacements à étudier, ici et ici.

Sur le même modèle, la regex (\d+) ne remplacera que des nombres, ^ sera pour signifier le début d'une ligne, et $ la fin d'une ligne.
Il est parfois nécessaire de penser à rajouter des séquences d'échappement avec un antislash \ devant certains caractères que l'on veut afficher, comme [, (etc.

Nous allons maintenant présenter quelques cas concrets de substitutions de textes, utilisables pour faciliter l'utilisation de technologies courantes (phpBB, zcode, xhtml).

phpBB

En exemple simple de remplacement, on peut donner celui de phpBB : imaginons que l'on ait rédigé un certain nombre de messages ou d'articles sur un forum phpBB, et que l'on veuille récupérer ces messages pour les inclure dans un autre document, par exemple un article wikipedia ou une publication avec LaTeX en pdf. On pourrait bidouiller avec le rendu en html, mais plus simplement on peut éditer les messages phpBB et récupérer le code source avec sa syntaxe assez simple.

Ainsi si on écrit en phpBB dans notre source en txt2tags, le code spécifique à phpBB sera détecté en premier, et converti en syntaxe txt2tags avant de convertir ensuite dans le langage ciblé. Il suffit donc de rajouter simplement ces directives de remplacement dans sa source txt2tags ou dans un fichier de configuration inclus :

Code : Autre

```
%!preproc: '\[b\]([^ ].*?)\[/b\]' '**\1**'
%!preproc: '\[u\]([^ ].*?)\[/u\]' '__\1__'
%!preproc: '\[i\]([^ ].*?)\[/i\]' '//\1//'

%!preproc: '\[code\]$' '```\n'
%!preproc:  '^\[/code\]' '```\n'

%!preproc: '\[code\]([^ ].*?)\[/code\]' '``\1``'

%!preproc: '\[quote\]$' '```\n'
%!preproc:  '^\[/quote\]' '```\n'

%!preproc: '\[url=([^ ].*?)\]([^ ].*?)\[/url\]' '[\2 \1]'
%!preproc: '\[img\]([^ ].*?)\[/img\]' '[\1]'
```

xhtml strict

Le xhtml généré par txt2tags n'est pas encore en xhtml strict. Si on souhaite cela, il suffit de rajouter ce code :

Code : Autre

```
%!postproc: '<i>' '<em>'
%!postproc: '</i>' '</em>'
%!postproc: '<b>' '<strong>'
%!postproc: '</b>' '</strong>'
```

sur le même modèle on peut même faire :

Code : Autre

```
%!postproc: '<u>' '<span class="underline">'
```

zcode

Même le site du zéro peut bénéficier de cet outil ! Par exemple pour rédiger ce document destiné au tutoriel, je le fais dans un fichier en texte simple, en utilisant la syntaxe txt2tags que j'affectionne. Je pourrai ainsi exporter mon tutoriel en zcode à la fin de la rédaction, ainsi qu'en xhtml pour mon site, et en pdf.

Code : Autre

```
%!postproc(html):         '<B>' <gras>
%!postproc(html):         '</B>' </gras>

%!postproc(html):         '<I>' <italique>
%!postproc(html):         '</I>' </italique>

%!postproc(html):         '<U>' <souligne>
%!postproc(html):         '</U>' </souligne>

%!postproc(html):         '<S>' <barre>
%!postproc(html):         '</S>' </barre>

%!postproc(html):         '<UL>' <liste>
%!postproc(html):         '</UL>' </liste>

%!postproc(html):         '<LI>([^ ].*?)$' <puce>\1</puce>
%!postproc(html):         '<LI>' <puce>
%!postproc(html):         '</LI>' </puce>

%!postproc(html):         '<IMG ALIGN="middle" SRC="([^ ].*?
)" BORDER="0" ALT="">' <image legende="\1">\1</image>

%!postproc(html):         '<P>' ''
%!postproc(html):         '</P>' ''

%!postproc(html):         '<A NAME="([^ ].*?)"></A>' ''
%!postproc(html):         '<A HREF="([^ ].*?)">([^ ].*?
)</A>' '<lien url="\1">\2</lien>'

%!postproc(html):         '<PRE>' '<code>'
%!postproc(html):         '</PRE>' '</code>'

%!postproc(html):         '<H3>([^ ].*?)</H3>' '<titre1>\1</titre1>'
%!postproc(html):         '<H2>([^ ].*?)</H2>' '<titre1>\1</titre1>'
%!postproc(html):         '<H1>([^ ].*?)</H1>' '<titre0>\1</titre0>'

%%% Colors for xhtml
%!postproc(xhtml): '@@COLOR@@([^ ].*?)@@'   '<span style="color:\1">'
%!postproc(xhtml): '@@/COLOR@@'   '</span>'


%%% Colors for LaTeX
```

```
%!postproc(tex): '@@COLOR@@([^ ].*?)@@'  '\\color{\1}{'
%!postproc(tex): '@@/COLOR@@'  '}'


%!preproc(xhtml):         '<information>' '//@@COLOR@@#6D7D2E@@**(i)** '
%!preproc(xhtml):         '</information>' @@/COLOR@@//

%!preproc(tex):         '<information>' '//@@COLOR@@OliveGreen@@**(i)** '
%!preproc(tex):         '</information>' @@/COLOR@@//

%!preproc(xhtml):         '<attention>' '//@@COLOR@@#835639@@**<!>** '
%!preproc(xhtml):         '</attention>' @@/COLOR@@//

%!preproc(tex):         '<attention>' '//@@COLOR@@BrickRed@@**<!>** '
%!preproc(tex):         '</attention>' @@/COLOR@@//

%!preproc(xhtml):       '<taille valeur="([^ ].*?)">' '**'
%!preproc(xhtml):       '</taille>' '**'
```

j'ai utilisé ici l'export html simple pour signifier l'export en zcode.

Images et LaTeX

Les fonctions avancées d'inclusion d'images avec LaTeX ne sont pas prises en compte dans txt2tags. Mais si on veut par exemple entourer une image avec du texte, il est possible d'ajouter dans le fichier cette macro :

Code : Autre

```
%!postproc(tex): 'wrap=([^ ].*?
)=wrap' '\\begin{wrapfigure}{l}{0\\textwidth} \\vspace{-
00mm} \1 \\vspace{-00mm} \\end{wrapfigure}'
```

ce qui aura pour effet de remplacer toutes les occurences du simplicime *wrap=[images.png]=wrap* par *\begin{wrapfigure}{l}{0\textwidth} \vspace{-00mm} \includegraphics[height=4cm]{image.png} \vspace{-00mm} \end{wrapfigure}*

On va ensuite faire la même chose pour le (x)html :

Code : Autre

```
%!postproc(xhtml): 'wrap=<img ([^ ].*?
)/>=wrap' '<img class="wrap" \1/>'
```

en utilisant un appel à une classe de css qui sera par exemple :

Code : Autre

```
.wrap {
          float: left;
          margin-top: 0;
          margin-right: 20px;
          margin-bottom: 10px;
          margin-left: 0;
          border: 1px solid #000;
          padding: 2px;
  }
```

On vient ainsi d'inventer la convention d'écriture *wrap= =wrap*, mais on aurait pu créer n'importe quelle autre syntaxe similaire, par exemple #= =# si on préfère.

Txt2cyoa

Il s'agit d'un système de création de "livres dont vous êtes le héros" en ligne, via un navigateur, avec des liens à faire pointer vers des paragraphes, selon les choix du joueur.

On voulait que cela ait une syntaxe simple à utiliser pour l'auteur. Simple comme du txt2tags par exemple !

On s'était donc donné comme contrainte que les choix soient du type :

Code : Autre

```
- Que l'aventure commence : 2
- Non, finalement, je préfère rester chez moi : 14
```

tandis que les paragraphes vers lesquels ces choix pointeront seront simplement créés avec la syntaxe habituelle de txt2tags :

Code : Autre

```
== 2 ==

Description du paragraphe 2

  /.../

== 14 ==

Description du paragraphe 14
```

Sur les principes expliqués plus haut, on a donc créé un fichier txt2tags à inclure dans l'histoire, utilisant notamment ces directives de préprocesseur :

- Faire en sorte que le dernier nombre d'une ligne soit transformé en lien nommé (named link), vers le même nom que lui-même :

  Code : Autre

  ```
  %!preproc: '(\d+)$'  '**[\1 #\1]**'
  ```

- Faire en sorte qu'un titre puisse avoir une ancre avec son propre nom :

  Code : Autre

  ```
  %!preproc: '^== (\d+) ==' '==\1==[\1]'
  ```

- On a également prévu l'inclusion de blocs de textes au format GBL, qui est un système similaire, mais qui utilise plutôt ce genre de syntaxe :

  Code : Autre

  ```
  >Que l'aventure commence !=2
  ```

  Notre code de remplacement est donc :

  Code : Autre

  ```
  %!preproc: '^>([^ ].*?)=(\d+)' '- \1 rendez-vous au \2'
  ```

et ensuite notre texte sera traité comme si on avait tapé dans la source, au lieu du code GBL, notre propre syntaxe : *- Que l'aventure commence ! rendez-vous au 2*

On peut également créer avec graphviz un organigramme de la structure de l'aventure, toujours avec txt2tags !

Vous pouvez obtenir les sources de cet outil, et éventuellement l'étudier, sur le site de txt2cyoa.

Le Textallion

Le Textallion est une archive d'outils et de scripts pour permettre de faciliter la création de documents divers (fichier pdf, code html, livre électronique epub...) à partir d'une même source, en txt2tags bien entendu. Cela étend donc la syntaxe basique de txt2tags, en utilisant le pré ou post-processeur de txt2tags, par exemple pour créer des lettrines en LaTeX et html, juste en rajoutant le symbole -**- en début d'un paragraphe :

Code : Autre

```
%!postproc(tex): "-\*\*-(.)(.*?
) " "\n\n\\lettrine[lines=2, lhang=0.33, loversize=0.25]{\1}{\2} "
%!postproc(xhtml): '-\*\*-(.)' '<span style="font-
size: 200%; vertical-align: baseline">\1</span>'
```

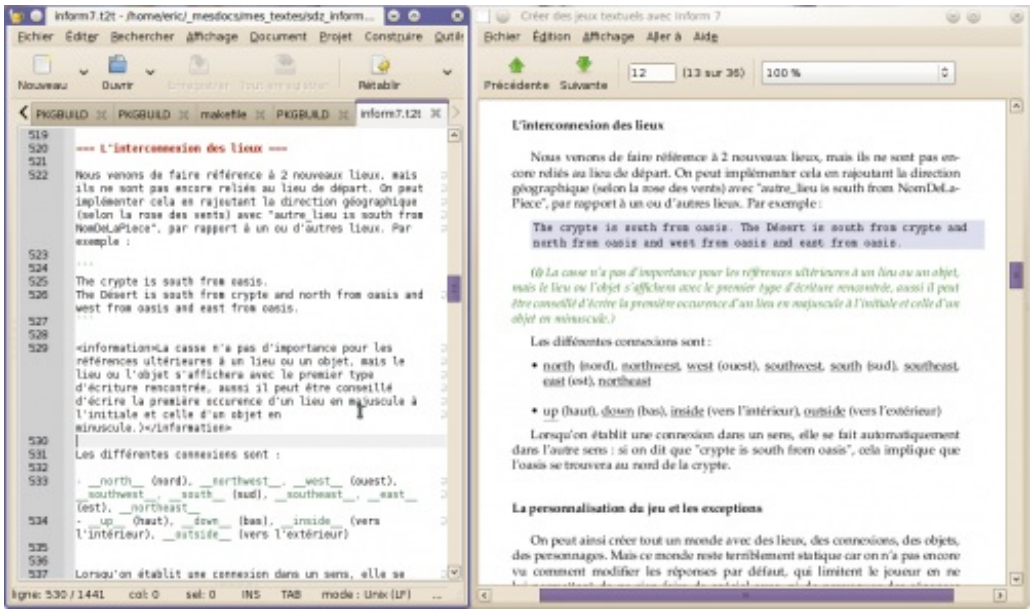
Par la suite, je me suis rendu compte que -**- n'était pas la syntaxe idéale pour cela, ne serait-ce que parce les 2 étoiles affichent ce qui suit comme si c'était en gras, avec certains éditeurs de textes. J'ai donc préféré définir tous ces symboles additionnels entre accolades, et j'ai choisi pour la lettrine le symbole {*~~~}. Mais au lieu de modifier les 2 lignes postproc plus haut, on peut simplement rajouter avant celles-ci la directive :

Code : Autre

```
%!preproc: '\{\*~~~\}' '-**-'
```

Comme cela la nouvelle syntaxe sera prise en compte, mais elle sera reconvertie en interne selon l'ancienne convention avant d'être transformée en lettrine pour les modes tex ou xhtml. De plus les textes plus anciens utilisant l'ancienne syntaxe -**- seront toujours valables sans avoir à les retoucher.

Le Textallion est disponible ici.



Une session de travail avec à gauche de code source édité dans geany, à droite le rendu exporté en pdf.

Autres astuces

- Inclure un include en php avec la syntaxe @@include"mon_fichier.php" :
  Code : Autre

```
%!postproc:  '@@include"([^ ].*?)"'  '<?\n   include("\1")\n ?
>'
```

- Faire une note de bas de page en LaTeX, en entourant la note avec des °° °°. Le même document exporté en html donnera le contenu de la note entre de simples parenthèses.
  Code : Autre

```
%!postproc(tex): '°°(.*?)°°' '\\footnote{\1}'
%!postproc(xhtml): '°°(.*?)°°' ' (\1)'
```

- Indiquer des notes de révisions et les faire disparaître ensuite.
  J'avais besoin de signifier dans un texte des notes pour des parties amenées à être remaniées ou disparaître. J'ai décidé de les mettre entre double parenthèses, et que cela serait transformé dans la version LaTeX en notes de bas de page, barrées. Voici la directive de remplacement à faire pour cela :
  Code : Autre

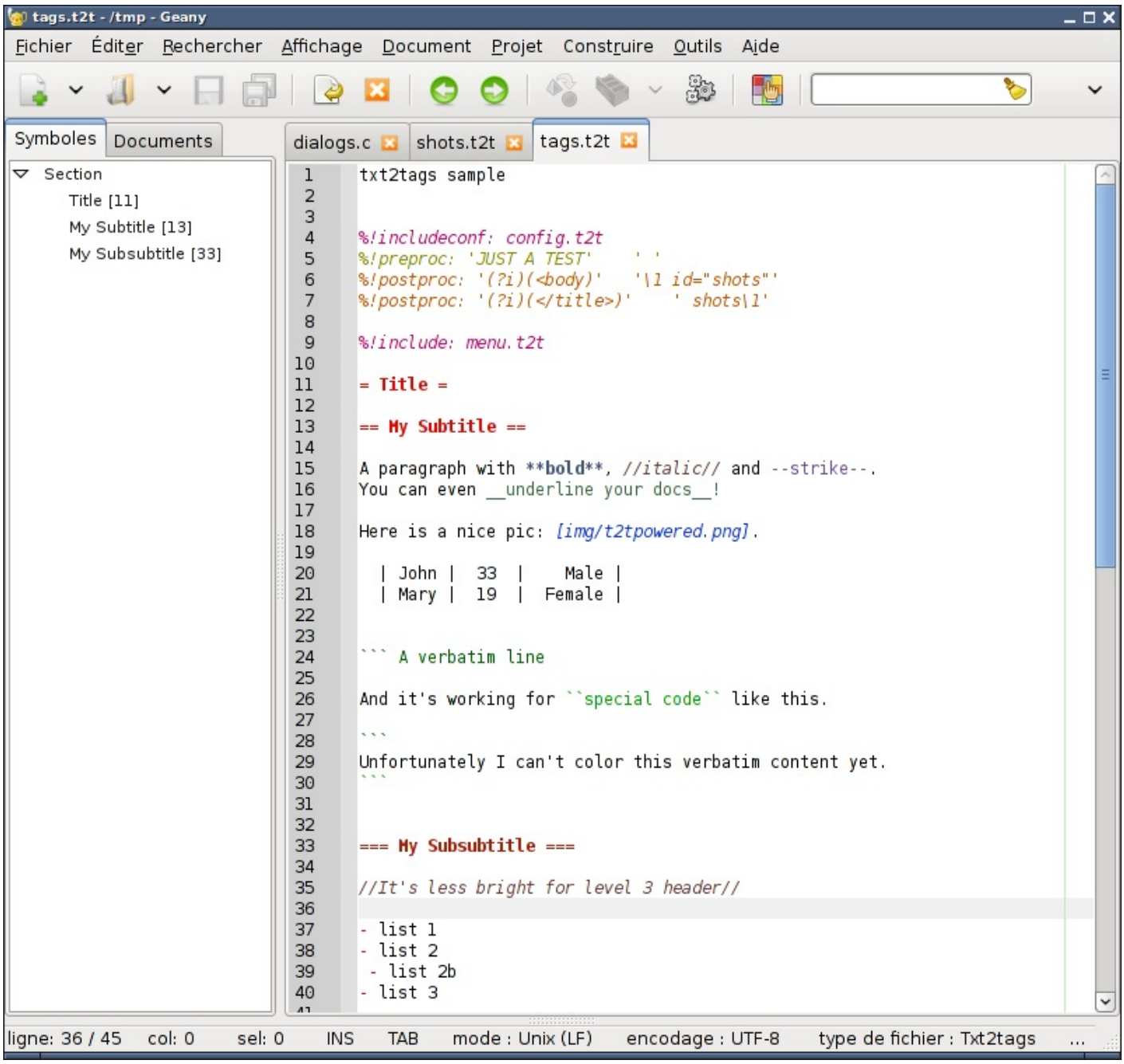
```
%!preproc(tex): '\(\((.*?)\)\)' '*°°--\1--°°'
```

  Par la suite, il suffit d'indiquer dans l'entête du fichier cette nouvelle directive et de commenter la précédente pour faire disparaître totalement ces notes du document pdf final :
  Code : Autre

```
%!preproc(tex): '\(\((.*?)\)\)' ''
```

Outils compatibles

Il existe de nombreux outils pour aider le rédacteur utilisant txt2tags, on pourra citer des colorations syntaxiques pour vim, emacs, kate, gedit, scintilla (scite, geany)..., un module Apache pour la conversion à la volée en HTML, une interface KDE, des scripts shell ou Ruby, un module d'aide à la rédaction pour Openoffice.org, un script vim permettant de convertir du HTML en txt2tags etc.



Et si on peut exporter depuis txt2tags vers d'autres langages de balisage léger, on peut également faire l'inverse, et importer directement la syntaxe txt2tags dans des wiki existants, comme par exemple pmwiki :
http://www.pmwiki.org/wiki/Cookbook/Txt2tags

Liens additionnels

- Site internet officiel de txt2tags
- Un autre tutoriel sur txt2tags (issu du site officiel)
- Manuel "Écrire des livres avec txt2tags"
  par l'auteur de txt2tags (Intéressant mais le contenu est un peu vieillissant, et il n'évoque pas la possibilité d'édition avancée avec des notes de bas de page par exemple)
- Astuces supplémentaires pour txt2tags

En conclusion, nous espérons que ce tutoriel vous aura intéressé, sinon convaincu, de la valeur de txt2tags comme outil polyvalent pour le webdesigner, pour l'écrivain, pour le rédacteur.
Ce n'est donc pas uniquement "encore un autre système de wiki", ou "bien trop limité par rapport à LaTeX" comme on l'entend parfois dire, mais un système qui se suffit à lui-même pour de nombreuses activités.

Ainsi avec finalement peu d'investissement et de sacrifices, vous pourrez passer à txt2tags pour vous faciliter la vie et produire

avec aisance des documents variés et complets.