LinuxFocus article number 269
http://linuxfocus.org

Gérer le HTML avec Perl, HTML::TagReader

par Guido Socher (homepage)

L´auteur:

Guido aime Perl parce que c’est un langage de script très versatile et rapide. Il aime la devise de Perl "Il y a plus d’une façon de le faire" parce qu’elle reflète la liberté et les possibilités offertes par l’"opensource".

Traduit en Français par:
Georges Tarbouriech
<gt/at/linuxfocus.org>

Résumé:

Si vous gérez un site web de plus de 10 pages HTML, vous vous rendrez vite compte qu’il est impératif d’utiliser quelques programmes pour vous aider dans cette tâche.
La plupart des logiciels traditionnels lisent les fichiers ligne par ligne (ou caractère par caractère). Malheureusement les lignes n’ont aucun sens dans les fichiers SGML/XML/HTML. Les fichiers SGML/XML/HTML reposent sur des balises. HTML::TagReader est un module très léger permettant d’examiner un fichier par balise.

Cet article suppose que vous connaissez bien Perl. Penchez-vous sur mes tutoriels Perl (Janvier 2000) si vous souhaitez apprendre ce langage.

Introduction

Traditionnellement les fichiers ont été basés sur la ligne. Quelques exemples sont fournis par les fichiers de configuration d’Unix comme /etc/hosts, /etc/passwd .... Certains systèmes d’exploitation plus anciens possédaient même des fonctions permettant de retrouver et d’écrire des données ligne à ligne.
Les fichiers SGML/XML/HTML sont basés sur des balises, les lignes n’ayant ici aucune signification, même si les éditeurs de texte et les individus sont encore dépendants des lignes.

Les gros fichiers HTML en particulier, consistent en de nombreuses lignes de code HTML pour votre serveur. Il existe même des outils comme "Tidy" pour indenter le HTML et le rendre lisible. Bien qu’il repose sur des tags, nous utilisons quand même des lignes pour le HTML. Nous pourrions le comparer à du code C. Théoriquement, vous pouvez écrire le code complet sur une seule ligne. Personne ne fait une

chose pareille : ce serait illisible.
Par conséquent, vous attendez d'un correcteur de syntaxe HTML qu'il écrive "ERROR: line ..." plutôt que "ERROR after tag 4123" (erreur après la balise 4123). Ceci parce que votre éditeur de texte vous permet facilement de sauter à une ligne donnée du fichier.

Il nous faut donc un moyen léger et efficace **pour examiner un fichier HTML balise par balise tout en conservant la référence aux numéros de lignes**.

Une solution éventuelle

La manière habituelle de lire un fichier avec Perl consiste à utiliser l'opérateur *while(<FILEHANDLE>)*. Celui-ci lit les données ligne par ligne et passe chacune d'elles à la variable $_ . Pourquoi Perl procède-t-il ainsi ? Perl possède une variable interne nommée INPUT_RECORD_SEPARATOR ($RS ou $/) dans laquelle il est défini que "\n" correspond à la fin d'une ligne. Si vous définissez $/=">", Perl utilisera alors ">" comme "end of line" (fin de ligne). La commande de script Perl suivante reformate le texte html de manière à ce qu'il finisse toujours par ">":

perl -ne 'sub BEGIN{$/=">";} s/\s+/ /g; print "$_\n";' file.html

Un fichier html qui ressemble à

```
<html><p>some text here</p></html>
```

deviendra

```
<html>
<p>
some text here</p>
</html>
```

Toutefois, l'essentiel n'est pas la lisibilité. Pour le développeur, l'important c'est que les données soient passées aux fonctions de son code balise par balise. Ainsi, il sera facile de rechercher "<a href= ..." même si le fichier html d'origine contient "a" et "href" sur des lignes différentes.

Modifier le "$/" (INPUT_RECORD_SEPARATOR) ne provoque aucune surcharge et tout est très rapide. Il est également possible d'utiliser l'opérateur "match" et des expressions régulières servant d'itérateur; on peut alors examiner le fichier avec des expressions régulières. C'est un peu plus compliqué et lent mais souvent utilisé.

Où est le problème ?? Le titre de cet article contient HTML::TagReader, mais jusqu'à présent je n'ai parlé que d'une solution plus simple qui n'a aucun besoin de modules supplémentaires. Quelque chose ne convient pas dans cette solution :

- La plupart des fichiers HTML au monde sont impropres. Il existe des millions de pages qui contiennent des exemples de code C qui en code HTML ressemblent à
  if (limit > 3) ....
  au lieu de
  if (limit > 3) ....
  En HTML "<" débute une balise et ">" la termine. Aucun des deux ne doit apparaître dans le texte.

La plupart des navigateurs afficheront les deux versions correctement et cacheront l'erreur.

- Modifier le "$/" affecte l'ensemble du programme. Si vous voulez examiner un autre fichier ligne à ligne pendant que vous lisez le fichier html, vous allez rencontrer un problème.

En d'autres termes, il n'est possible que dans certains cas d'utiliser le "$/" (INPUT_RECORD_SEPARATOR).

J'ai encore un exemple de programme utile correspondant à ce dont nous avons parlé jusqu'ici. Il définit "$/" en "<". Les navigateurs ne gèrent pas aussi bien un "<" erroné qu'un ">" et par conséquent il existe beaucoup moins de pages html contenant un "<" incorrect qu'un ">" mal placé. Le programme se nomme tr_tagcontentgrep (cliquez pour le visualiser) et vous pouvez voir dans le code comment conserver le numéro de ligne. tr_tagcontentgrep peut être utilisé pour rechercher ("grep") une chaîne (par ex."img") dans une balise, y compris si la balise s'étend sur plusieurs lignes. Comme ceci :

tr_tagcontentgrep -l img file.html
index.html:53: <IMG src="../images/transpix.gif" alt="">
index.html:257: <IMG SRC="../Logo.gif" width=128 height=53>

HTML::TagReader

HTML::TagReader résout les deux problèmes en modifiant le INPUT_RECORD_SEPARATOR et offre ainsi un moyen beaucoup plus élégant de séparer le texte des balises. Il n'est pas aussi lourd qu'un véritable HTML::Parser (analyseur de code HTML) et offre ce dont vous avez besoin pour examiner du code html : une méthode de lecture balise par balise.

Assez de bla-bla. Voici comment l'utiliser. Vous devez d'abord écrire
use HTML::TagReader;
dans votre code pour charger le module. Ensuite vous appelez
my $p=new HTML::TagReader "filename";
pour ouvrir le fichier "filename" et obtenir un objet référence renvoyé dans $p. Vous pouvez maintenant appeler $p->gettag(0) ou $p->getbytoken(0) pour atteindre la balise suivante. gettag ne renvoie que le contenu des balises (ce qui se trouve entre < et >) alors que getbytoken fournit également le texte contenu entre les balises en vous indiquant la nature (balise ou texte). Grâce à ces fonctions il est très facile d'examiner les fichiers html. Essentiel pour gérer un gros site. Une description complète de la syntaxe se trouve dans la page de manuel de HTML::TagReader.

Voici un véritable programme exemple. Il affiche les titres des différents documents examinés :

```
#!/usr/bin/perl -w
use strict;
use HTML::TagReader;
#
die "USAGE: htmltitle file.html [file2.html...]\n" unless($ARGV[0]);
my $printnow=0;
my ($tagOrText,$tagtype,$linenumber,$column);
#
for my $file (@ARGV){
```

```
  my $p=new HTML::TagReader "$file";
  # read the file with getbytoken:
  while(($tagOrText,$tagtype,$linenumber,$column) = $p->getbytoken(0)){
  if ($tagtype eq "title"){
    $printnow=1;
    print "${file}:${linenumber}:${column}: ";
    next;
  }
  next unless($printnow);
  if ($tagtype eq "/title" || $tagtype eq "/head" ){
    $printnow=0;
    print "\n";
    next;
  }
  $tagOrText=~s/\s+/ /; #kill newline, double space and tabs
  print $tagOrText;
  }
}
# vim: set sw=4 ts=4 si et:
```

Comment fonctionne-t-il ? Nous lisons le fichier html par $p->getbytoken(0) et lorsque nous trouvons <title> ou <Title> ou <TITLE> (ils sont renvoyés en tant que $tagtype eq "title") nous positionnons un drapeau ($printnow) pour démarrer l'affichage et nous arrêtons ce dernier lorsque nous rencontrons </title>.
Vous utilisez ce programme de cette manière :

htmltitle file.html somedir/index.html
file.html:4: the cool perl page
somedir/index.html:9: joe's homepage

Bien sûr, il est possible d'activer le tr_tagcontentgrep ci-dessus avec HTML::TagReader. C'est plus court et plus facile à écrire :

```
#!/usr/bin/perl -w
use HTML::TagReader;
die "USAGE: taggrep.pl searchexpr file.html\n" unless ($ARGV[1]);
my $expression = shift;
my @tag;
for my $file (@ARGV){
  my $p=new HTML::TagReader "$file";
  while(@tag = $p->gettag(0)){
    # $tag[0] is the tag (e.g <a href=...>)
    # $tag[1]=linenumber $tag[2]=column
    if ($tag[0]=~/$expression/io){
      print "$file:$tag[1]:$tag[2]: $tag[0]\n";
    }
  }
}
```

Le script est court et ne se préoccupe pas trop de la gestion des erreurs mais il est parfaitement fonctionnel. Pour extraire (grep) les balises contenant la chaîne "gif", tapez :

taggrep.pl gif file.html
file.html:135:15: <img src="images/2doc.gif" width=34 height=22>
file.html:140:1: <img src="images/tst.gif" height="164" width="173">

Un autre exemple ? Voici un programme qui supprime toutes les balises <font...> et du code html. Ces balises de polices sont parfois utilisées en grande quantité par quelques éditeurs html graphiques mal écrits et posent de nombreux problèmes lorsque les pages sont visualisées avec des navigateurs différents sur des résolutions différentes. Ce simple script élimine ces balises. Vous pouvez le modifier de façon à ce qu'il ne supprime que les balises définissant la police ou la taille tout en laissant la couleur inchangée.

```
#!/usr/bin/perl -w
use strict;
use HTML::TagReader;
# strip all font tags from html code but leave the rest of the
# code un-changed.
die "USAGE: delfont file.html > newfile.html\n" unless ($ARGV[0]);
my $file = $ARGV[0];
my ($tagOrText,$tagtype,$linenumber,$column);
#
my $p=new HTML::TagReader "$file";
# read the file with getbytoken:
while(($tagOrText,$tagtype,$linenumber,$column) = $p->getbytoken(0)){
  if ($tagtype eq "font" || $tagtype eq "/font"){
    print STDERR "${file}:${linenumber}:${column}: deleting $tagtype\n";
    next;
  }
  print $tagOrText;
}
# vim: set sw=4 ts=4 si et:
```

Comme vous le voyez, il est très facile d'écrire des programmes utiles en quelques lignes.
Le paquetage de code source de HTML::TagReader (voir références) contient quelques applications de HTML::TagReader:

- tr_blck -- vérifie les liens relatifs brisés dans des pages HTML
- tr_llnk -- liste les liens des fichiers HTML
- tr_xlnk -- étend les liens vers des répertoires en liens vers des fichiers index
- tr_mvlnk -- modifie les balises dans des fichiers HTML avec des commandes perl.
- tr_staticssi -- étend les directives SSI #include virtual et #exec cmd et produit une page html statique.
- tr_imgaddsize -- ajoute width=... et height=... à <img src=...>

tr_xlnk et tr_staticssi sont très pratiques lorsque vous souhaitez créer un CDRom à partir d'un site web. Le serveur web vous fournira par exemple http://www.linuxfocus.org/index.html même si vous n'avez tapé que http://www.linuxfocus.org/ (sans index.html). Toutefois, si vous vous contentez de graver tous les fichiers et répertoires sur un CD et que vous consultiez ce CD directement avec votre navigateur (file:/mnt/cdrom) vous obtiendrez une liste de répertoires au lieu d'un fichier index.html. La société qui a gravé le premier CD de LinuxFocus a fait cette erreur et le résultat était catastrophique. Maintenant que les données sont obtenues par tr_xlnk les CD fonctionnent très bien.

Je suis certain que trouverez HTML::TagReader très pratique. Programmez bien !

Références

- La page de manuel de HTML::TagReader

- Tutoriel Perl : Perl III (Janvier 2000)
- Le programme tr_tagcontentgrep (celui qui n'utilise pas HTML::TagReader): tr_tagcontentgrep (txt) ou tr_tagcontentgrep (html)
- Le code source de HTML:TagReader:
  http://cpan.org/authors/id/G/GU/GUS/
  ou
  http://main.linuxfocus.org/~guido/
- Tidy est indispensable si vous créez des sites web : tidy, un utilitaire de vérification de syntaxe html
  Comment utiliser tidy? Facile :
  `tidy -e file.html`
  affichera les erreurs html
  `tidy -im -raw file.html`
  éditera le fichier et l'indentera correctement. Il corrigera aussi les fautes (si tidy est capable de deviner le sens de ce qui est écrit).

Site Web maintenu par l´équipe d´édition LinuxFocus	Translation information:
© Guido Socher	en --> -- : Guido Socher (homepage)
"some rights reserved" see linuxfocus.org/license/	en --> fr: Georges Tarbouriech <gt/at/linuxfocus.org>
http://www.LinuxFocus.org	

2005-01-14, generated by lfparser_pdf version 2.51