LinuxFocus article number 269
http://linuxfocus.org

door Guido Socher (homepage)

Over de auteur:

Guido houdt van Perl omdat het een zeer flexibele en snelle scripttaal is. Hij hangt het motto "There's more than one way to it" (Er is meer dan een manier om het te doen) aan, wat de vrijheid en mogelijkheden reflecteert, die je krijgt met Open Source.

Vertaald naar het Nederlands door:
Guus Snijders
<ghs(at)linuxfocus.org>

HTML beheren met Perl, HTML::TagReader

Kort:

Als je een website wilt beheren met meer dan 10 HTML pagina's, zul je er snel achterkomen dat je programma's nodig hebt die jou ondersteunen.
De meeste traditionele software leest bestanden regel voor regel (of karakter voor karakter). Helaas hebben regels geen betekenis in SGML/XML/HTML bestanden. SGML/XML/HTML bestanden zijn gebaseerd op Tags. HTML::TagReader is een lichtgewicht module om een bestand tag voor tag te bewerken.

Dit artikel gaat ervan uit dat je Perl redelijk goed kent. Kijk naar mijn Perl tutorials (januari 2000) als je Perl wilt leren.

Introductie

Traditionele bestanden zijn regel-gebaseerd. Voorbeelden hiervan zijn Unix configuratie bestanden, zoals /etc/hosts, /etc/passwd... Er zijn zelfs oudere besturingssytemen waarbij je functies hebt om data regel voor regel te lezen en/of te schrijven.
SGML/XML/HTML bestanden zijn gebaseerd op Tags, regels hebben hier geen betekenis, echter tekst editors en mensen zijn op een of andere manier nog steeds regel gebaseerd.

Vooral grote HTML bestanden bestaan meestal uit meerdere regels HTML code. Er zijn zelfs tools zoals "Tidy" om html te laten inspringen en leesbaar te maken. We gebruiken regels, ondanks dat HTML is gebaseerd op Tags en niet op regels. Je kunt het vergelijken met C-code. Theoretisch kun je de volledige

code op een enkele regel schrijven. Niemand doet dat. Het zou onleesbaar worden.
Daarom verwacht je van een HTML syntax checker om een melding te geven als "ERROR: line ..." in plaats van "ERROR na tag 4123". Dit is omdat je tekst editor je toestaat om eenvoudig naar een bepaalde regel in het bestand te springen.

Wat hier nodig is is een goede en lichte manier om **een HTML bestand Tag voor Tag te bewerken met behoud van de regel nummers**.

Een mogelijke oplossing

De gebruikelijke manier om een bestand in Perl te lezen is door gebruik te maken van de *while(<FILEHANDLE>)* operator. Dit zal de data regel voor regel lezen en iedere regel in de $_ variabele plaatsen. Waarom doet Perl dit? Perl heeft een interne variabele genaamd INPUT_RECORD_SEPARATOR (R$ of $/) waarbij is gedefiniëerd dat "\n" het einde van een regel is. Als je $/=">" set, zal perl ">" gebruiken als "regeleinde". Het volgende commando regel Perl script zal html tekst herformatteren met ">" als regeleinde:

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

Een html bestand dat er uit ziet als

```
<html><p>hier wat tekst</p></html>
```

zal er uit komen te zien als

```
<html>
<p>
hier wat tekst</p>
</html>
```

Het belangrijkste hier is echter niet de leesbaarheid. Voor de software ontwikkelaar is het belangrijk dat de data Tag voor Tag aan de functies wordt gepresenteerd in zijn/haar code. Hiermee wordt het eenvoudig om te zoeken naar een "<a href= ..." zelfs asl de orginele html een "a" en "href" op verschillende regels had.

Het veranderen van de "$/" (INPUT_RECORD_SEPARATOR) geeft geen proces overhead en is erg snel. Het is ook mogelijk om de match operator en reguliere expressies als een iterator te gebruiken en het bestand met reguliere expressies te bewerken. Dit is iets gecompliceerder en trager, maar wordt ook veel gebruikt.

Wat is het probleem?? De titel van dit artikel is HTML::TagReader maar nu heb ik het alleen maar gehad over een veel simpeler oplossing die geen extra modules nodig heeft. Er moet iets mis zijn met deze oplossing:

- Bijna alle HTML bestanden in de wereld bevatten fouten. Er zijn miljoenen pagina’s die bijvoorbeeld C code voorbeelden bevatten, welke er op HTML niveau uitzien als
  `if ( limit > 3) ....`
  in plaats van
  `if ( limit > 3) ....`

In HTML zou "<" een tag starten en ">" zou het moeten beëindigen. Geen ervan zou op zichzelf moeten voorkomen in de tekst. De meeste browsers zullen beide correct weergeven en de fout verbergen.

- Het aanpassen van de "$/" beïnvloedt het hele programma. Als je een ander bestand regel voor regel wilt bewerken terwijl je een html bestand leest heb je een probleem.

In andere woorden, het is alleen maar in bijzondere gevallen mogelijk om de "$/" (INPUT_RECORD_SEPARATOR) te gebruiken.

Ik heb een handig voorbeeld programma dat gebruik maakt van waar we het tot nog toe over gehad hebben. Het zet "$/" naar "<". De webbrowser kunnen niet goed met een misplaatste "<" als een ">" en daardoor zijn er minder web pagina's met misplaatste "<" als met een misplaatste ">". Het programma is genaamd tr_tagcontentgrep (klik om te bekijken) en je kunt ook in de code zien hoe de regelnummers behouden blijven. tr_tagcontentgrep kan ook gebruikt worden om een string te "grep"en (bijvoorbeeld "img") in een Tag zelfs als de Tag meerdere regels beslaat. Iets als:

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 lost het probleem op van het veranderen van de INPUT_RECORD_SEPARATOR en biedt bovendien een nettere manier om tekst van tags te onderscheiden. Het is niet zo zwaar als een volledige HTML::Parser en biedt wat je wilt als je html code wilt bewerken: Een methode om Tag voor Tag te lezen.

Genoeg gepraat. Hier is hoe je het gebruikt. Eerst schrijf je
uset HTML::TagReader;
in je code om de module te laden. Daarna roep je
my $p=new HTML::TagReader "filenaam"; aan
om het bestand "filename" te openen en een object referentie geretourneerd te krijgen in $p. Nu kun je $p->gettag(0) of $p->getbytoken(0) gebruiken om de volgende Tag te krijgen. gettag geeft alleen Tags terug (het spul tussen de < en >) terwijl getbytoken je ook de tekst tags geeft en je verteld wat het is (Tag of tekst). Met deze functies is het erg eenvoudig om html bestanden te bewerken. Essentiëel om een grotere website te onderhouden. Een volledige syntax beschrijving kan gevonden worden in de man pagina van HTML::TagReader.

Hier is nu een echt voorbeeld programma. Het print de document titels van een aantal documenten:

```
#!/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:
```

Hoe het werkt? We lezen het html bestand met $p-gt;getbytoken(0) als we <title> of <Title> of <TITLE> tegenkomen (ze worden geretourneerd als $tagtype eq "title") stellen we een parameter in ($printnow) om te beginnen met printen en als we </title> tegenkomen, stoppen we met printen.
Je kunt het programma zo gebruiken:

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

Natuurlijk is het mogelijk om de tr_tagcontentgrep van boven te implementeren met HTML::TagReader. Iets korter en eenvoudiger om te schrijven:

```
#!/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";
    }
  }
}
```

Het script is kort en doet niet veel aan fout afhandeling maar is verder volledig functioneel. Om tags te greppen die de string "gif" bevatten, type je:

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">

Nog een voorbeeld? Hier is een programma dat alle <font...> en zal strippen van de html code. Deze font tags worden soms gebruikt in massieve hoeveelheden door slecht ontworpen html editors en kunnen veel problemen veroorzaken bij het weergeven van de pagina's op verschillende browsers en verschillende scherm groottes. Deze eenvoudige versie stript alle font Tags. Je kunt het aanpassen zodat het alleen diegene verwijderd die de fontface of grootte aanpassen en de kleur ongemoeid laten.

```
#!/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:
```

Zoals je kunt zien is het erg eenvoudig om bruikbare programma's te schrijven met een slechts een paar regels.
Het broncode pakket van HTML::TagReader (zie de referenties) bevat al enkele applicaties van HTML::TagReader:

- tr_blck -- controleer op gebroken links in HTML pagina's
- tr_llnk -- geef de links in HTML bestanden weer
- tr_xlnk -- breid links naar directories uit naar links op index bestanden
- tr_mvlnk -- pas Tags in HTML bestanden aan met Perl commando's
- tr_staticssi -- breid SSI directieven uit, #include virtuele en #exec cmd en produceer een statische HTML pagina.
- tr_imgaddsize -- voeg width=... en height=... toe aan <img src=...>

tx_xlnk en tr_staticssi zijn erg bruikbaar als je een CDrom wilt maken van een website. De web server geeft je bijvoorbeeld http://www.linuxfocus.org/index.html terwijl je alleen maar http://www.linuxfocus.org/ (zonder de index.html) hebt getypted. Als je echter gewoon alle bestanden en directories op CD brandt, en de CD met je browser benadert (file:/mnt/cdrom) zul je een directory listing te zien krijgen in plaats van index.html. Het bedrijf dat de eerste LinuxFocusCD creëerde, maakte deze fout en het was verschikkelijk om gebruik te maken van de CD. Nu ze alle data door tr_xlnk halen werken de CDs.

Ik ben er zeker van dat je HTML::TagReader nuttig zult vinden. Veel programmeer plezier!

Referenties

- The man page van HTML::TagReader
- Perl tutorial: Perl III (January 2000)
- Het tr_tagcontentgrep programma (diegene die geen gebruik maakt van HTML::TagReader): tr_tagcontentgrep (txt) of tr_tagcontentgrep (html)
- De broncode van HTML:TagReader:
  http://cpan.org/authors/id/G/GU/GUS/
  of
  http://main.linuxfocus.org/~guido/
- Tidy is essentiëel als je aan web design doet: tidy, een utility om de syntax van html te controleren
  Hoe tidy te gebruiken? Simpel:
  `tidy -e file.html`
  zal de html fouten printen
  `tidy -im -raw file.html`
  zal het bestand bewerken en het netjes laten inspringen. Het zal ook fouten corrigeren (voor zover tidy kan gokken wat de bedoeling was).

Site onderhouden door het LinuxFocus editors team	Vertaling info:
© Guido Socher	en --> -- : Guido Socher (homepage)
"some rights reserved" see linuxfocus.org/license/	en --> nl: Guus Snijders <ghs(at)linuxfocus.org>
http://www.LinuxFocus.org	

2005-01-14, generated by lfparser_pdf version 2.51