Administrando HTML com o Perl, HTML::TagReader

by Guido Socher (homepage)

About the author:

O Guido gosta de Perl porque é uma linguagem de scripting bastante flexível e rápida. Ele gosta do lema do Perl "Há mais do que um modo para fazer as coisas" o que reflecte a liberdades e as possibilidades que se têm quando se lida com o opensource.

Abstract:

Se quiser administrar um website com mais do que 10 páginas HTML, então cedo se aperceberá que necessita de alguns programas para lhe dar suporte. A maior parte do software tradicional lê linha a linha (ou caracter a caracter). Infelizmente as linhas não têm significado nos ficheiros SGML/XML/HTML, estes ficheiros são baseados em Tags. O HTML::TagReader é um módulo leve para procesasr um ficheiro atrvés de Tags.

Este artigo assume que já conhece bem o Perl. Dê uma vista de olhos nos meus toturiais de Perl (Janeiro 2000) se quiser aprender Perl.

Introdução

Os ficheiros tradicionais têm sido à base de linhas. Exemplo disto são os ficheiros de configuração do Unix, como /etc/hosts, /etc/passwd... Existem, inclusivé sistemas operativos que têm funções para ler e escrever linha a linha.
Os ficheiros SGML/XML/HTML são baseados em Tags, as linhas aqui não têm significado, contudo os editores de texto e os humanos são, ainda de algum modo, baseados em linhas.

Normalmente, os ficheiros grandes HTML, consistem em diversas linhas de código HTML. Existem utilitários como o "Tidy" para identar html e o tornar legível. Usamos linhas apesar do HTML ser baseado em Tags e não linhas. Pode compará–lo ao código C. Teoricamente podia escrever todo o código numa só linha. Ninguém faz isto. Seria Ilegível.
Assim espera que o analisador de sintaxe HTML escreva: "ERROR: linha..." em vez de "ERROR após tag 4123". Isto porque o seu editor de texto lhe permite saltar facilmente para a dada linha.

O que aqui é preciso é um bom e leve modo de **processar um ficheiro HTML Tag a Tag mantendo o número de linhas**

Uma possível solução

O modo normal de ler um ficheiro em Perl é usar o operador *while(<FILEHANDLE>)*. Isto lê linha a linha, passando cada linha para a variável $_. Porque é que o Perl faz isto? O Perl tem uma variável interna chamada INPUT_RECORD_SEPARATOR ($RS ou $/) onde está definido que "\n" é o fim de uma linha. Se definir $/=">" então o Perl usa ">" como "fim de linha". O seguinte comando Perl reformata o texto html para terminar sempre em ">":

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

Um ficheiro html que se parece com

```
<html><p>algum texto aqui</p></html>
```

tornar−se−á

```
<html>
<p>
algum texto aqui</p>
</html>
```

Contudo o conteúdo principal ainda não é legível. Para o programador de software é importante que os dados sejão passados às funções no seu código Tag a Tag. Com isto será fácil procurar "<a href= ..." mesmo que o código original tenha "a" e "href" em linhas separadas.

Alterar a variável "$/" (INPUT_RECORD_SEPARATOR) faz com que o overhead do processamento seja mínimo e muito rápido. É ainda possível usar o operador de igualdade (match) e as expressões regulares como iteradores e processar o ficheiro com expressões regulares. Isto é um pouco mais complicado e mais lento mas também é usado.

Onde é que está o porblema?? O título deste artigo dizia: HTML::TagReader mas até agora estive todo o tempo a falar de uma solução mais simples e que não requer módulos extra. Deve haver algo de errado com esta solução:

- Praticamente todos os ficheiros HTML no mundo têm erros. Existem milhões de páginas que contêm, por exemplo, exemplos de código C que aparecem no código HTML do seguinte modo:
  ```
  if ( limit > 3) ....
  ```
  em vez de
  ```
  if ( limit &gt; 3) ....
  ```
  No HTML "<" devia começar uma tag e >" devia terminá−la. Nenhuma deles deve aparecer sozinha algures no texto. A maioria dos browser apresenta ambas correctamente escondendo o erro.
- Alterar o "$/" afecta todo o programa. Se quiser processar outro ficheiro linha a linha enquanto lê o ficheiro html, então tem um problema.

Por outras palavras, só em casos especiais é que é possível usar o "$/" (INPUT_RECORD_SEPARATOR).

Contudo ainda tenho um bom exemplo para si e que usa o que até então discutimos. Contudo define o "$/" para "<" porque os browsers não conseguem lidar tão bem com um "<" mal colocado como com um ">" por isso existem menos páginas web com "<" mal colocados do que com ">" O programa chama−se

tr_tagcontentgrep (clique para ver) pode também ver no código como manter o número da linha. O tr_tagcontentgrep pode ser usado para pesquisar uma string (por exemplo "img") numa Tag mesmo que esta ocupe várias linhas. Algo como:

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 resolve dois dos problemas da modificação do INPUT_RECORD_SEPARATOR e oferece um modo mais simpático de separar o texto das tags. Não é tão pesado quanto um HTML::Parser completo mas oferece o que quer quando processa o código html: Um método para ler Tag por Tag.

Basta de palavras. Eis o modo como se usa, primeiro deve escrever
use HTML::TagReader;
no seu código para carregar o módulo, depois chama
my $p=new HTML::TagReader "filename";
para abrir o ficheiro "filename" e obter a referência para um objecto retornardo em $p. Agora pode chamar $p–>gettag(0) ou $p–>getbytoken(0) para obter a próxima Tag. O gettag só retorna Tags (algo entre < e >) enquanto que o getbytoken retorna também o texto entre as tags e diz–lhe o que é (se uma Tag ou Texto). Com estas funções é muito fácil de processar ficheiros html. O Essencial para manter um grande website. A descripção total pode ser encontrada na página man do HTML::TagReader.

Eis agora um programa de exemplo real. Imprime os títulos dos documentos de um dado número de documentos:

```
#!/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:
```

Como é que trabalha? Lemos o ficheiro html com $p−>getbytoken(0) quando encontramos <title> ou <Title> ou <TITLE> (elas são retornadas como $tagtype iguais a "title") e depois define−se uma flag ($printnow) para iniciar a impressão e quando encontramos </title> paramos a impressão.
Usa o programa deste modo:

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

Claro que é possível implementar o tr_tagcontentgrep a partir do HTML::TagReader. Um pouco mais curto e simples de escrever:

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

A script é pequena e não tem muita validação de erros mas por outro lado está totalmente funcional. Para pesquisar por tags que contenham a string "gif" pode digitar:

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">

Mais um exemplo? Eis aqui um programa que obtem todas as tags <font...> e do seu código html. Estas tags de fontes são usadas de um modo massivo por editores gráficos de html pobres causando bastantes problemas quando se vêem páginas em diferentes browsers e com resoluções de ecrã diferentes. Esta versão simples remove todas as Tags de fontes. Pode alterá−la para somente remover aquelas que definem a fontface ou tamanho e que mantêm a cor inalterada.

```
#!/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:
```

Como pode ver é muito fácil escrever programas úteis somente com algumas linhas de código.
O pacote com o código fonte do HTML::TagReader (ver referências) contém já algumas das aplicações do HTML::TagReader:

- tr_blck −− verifica os links relativos não navegáveis nas páginas HTML
- tr_llnk −− lista os links nos ficheiros HTML
- tr_xlnk −− expande os links dos directórios num ficheiro index com links.
- tr_mvlnk −− modifica as tags dos ficheiros HTML com os comandos perl.
- tr_staticssi −− expande as directivas SSI #include virtual e #exec cmd e produz uma página estática html.
- tr_imgaddsize −− adiciona width=... e height=... a <img src=...>

O tr_xlnk e o tr_staticssi são bastante úteis quando pretende fazer um CDrom do seu Website. Por exemplo, o servidor dá−lhe http://www.linuxfocus.org/index.html mesmo que tenha digitado http://www.linuxfocus.org/ (sem o index.html). Se passar todos os ficheiros e directórios para o CD e aceder ao CD com um browser directamente (ficheiro:/mnt/cdrom) então verá uma listagem do directório em vez do index.html. A companhia que fez o primeiro CD da LinuxFocus fez este erro e foi terrível usar o CD. Agora que obtêm os dados através do tr_xlnk os CDs estão a trabalhar.

Tenho a certeza que encontrará o HTML::TagReader útil. Divirta−se a programar!

Referências

- A página man do HTML::TagReader
- Perl tutorial: Perl III (Janeiro 2000)
- O programa tr_tagcontentgrep (o que não usa o HTML::TagReader): tr_tagcontentgrep (txt) or tr_tagcontentgrep (html)
- O código fonte do HTML:TagReader:
  http://cpan.org/authors/id/G/GU/GUS/
  ou
  http://main.linuxfocus.org/~guido/
- O Tidy é essencial se trabalha em web design: tidy, um utilitário para verificar a sintaxe html
  Como usar o tidy? Fácil:
  `tidy −e file.html`
  imprime os erros html
  `tidy −im −raw file.html`
  editará o ficheiro identando−o de um modo simpático. Também corrige as falhas (desde que o tidy consiga advinhar o que se pretendia).

Translation information:
en −−> −− : Guido Socher (homepage)
en −−> pt: Bruno Sousa <brunosousa(at)linuxfocus.org>

2005−01−10, generated by lfparser_pdf version 2.51