Gestire i documenti HTML con Perl, HTML::TagReader

by Guido Socher (homepage)

About the author:

Guido adora Perl per la sua flessibilità e per la sua rapidità come linguaggio per script. Apprezza il motto di Perl: "C'è più di un modo per fare le cose" che rispecchia la libertà e le possibilità offerte dal mondo dell'opensource.

Abstract:

Se volete gestire un sito web con più di 10 pagine HTML, presto vi renderete conto che averete la necessità di ricorrere ad un programma che vi possa aiutare.
La maggior parte del software tradizionale processa riga per riga (o addirittura carattere per carattere). Sfortunanatamente le righe di un file non hanno alcun valore in SGML/XML/HTML. Questo tipo di file si basa sui tag (ovvero parole chiave) HTML::TagReader è un modulo che richede risorse moderare per processare un file costituito con dei tag.
Questo articolo parte dal presupposto che voi conosciare il linguaggio Perl abbastanza bene. Potete dare un'occhiata ai miei articoli sul Perl per prendervi confidenza.

Introduzione

Tradizionalmente i file si sono sempre basati sulla riga. Un classico esempio possono essere i file di configurazione di Unix, come per esempio, /etc/hosts, /etc/passwd,... Esistono anche sistemi operativi datati che hanno specifiche funzioni per estrarre il contenuto di un file riga per riga.
SGML/XML/HTML sono invece bastati sulle tag, e quindi la semplice riga perde di significato, tuttavia gli editor di testi ed il pensiero umano sono ancora in qualche modo legati al concetto di riga.

Questo è particolarmente vero di grossi file HTML che sono costituiti da svariate righe di codice. Esistono strumenti come "Tydy" per permettere di avere una formattazione del codice HTML e renderlo più leggibile. Tuttavia essi ricorrono sempre all`idea di riga e non di tag ed l'HTML è per l'appunto basato du tag! Il codice HTML potrebbe essere paragonato al codice C. Teoricamente potreste scrivere l'intero codice su di una sola riga, ma nessuno lo fa. Divvrebbe illegibile!
Tuttavia voi vi aspettate che un sistema di controllo della sintassi di un file HTML vi scriva "ERRORE: alla

riga nuermo ...." piuttosto che "ERRORE al tag 4123". Questo in quanto l'editor che andate ad utilizzare vi permette facilmente di raggiungere la riga in questione.

Quello che ci serve, quindi, è un semplice e leggero sistema di controllo per **controllare il file HTML tag per tag e, nello stesso tempo, tenere traccia della riga che stiamo analizzando** .

Una possibile soluzione

Il modo convezionale di leggere un file in Perl è quello di ricorrere al'operatore *while(<FILEHANDLE>)*. Questo permette di leggere i dati riga per riga e di assegnarne il contentuto alla variabile $_. Perchè il Perl si comporta così? Perchè ricorre ad una variabile interna detta INPUT_RECORD_SEPARATOR ($RS o $/) ove all'interno della stessa viene definito "\n" come carattere di fine riga. Se definite $/=">" allora Perl utilizzerà ">" come carattere di fine riga. Il seguente script in Perl formatterà il testo hmtl in modo che ogni riga termini sempre con ">":

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

un file html che apparirà nel seguente modo

```
<html><p>il vostro testo qui</p></html>
```

diverrà

```
<html>
<p>
il vostro testo qui</p>
</html>
```

Un aspetto molto importante di questo modo di procedere non è di certo la sua più lineare leggibilità. Per lo svlippatore software è più importante che ie dati sia correttamente trattari dalle funzioni, tag per tag. Questo rende molto semplice la possbilità di ricerche di tag, come per esemio "<a href="..." anche se nel codice originale le parole chiavi "a" ed "html" fossero su righe diverse.

Cambiando il valore di "$/" (INPUT_RECORD_SEPARATOR) non si avrà nessun sovraccarico del sistema ed il tutto sarà molto veloce. Sarebbe anche possibile ricorrere all'operatore di uguaglianza ed utilizzare le espressioni (ovvero più stringhe) come iteratori e quindi processare il file per mezzo di espressioni predefinite. Questo è ovviamente un poco più complesso e meno veloce nell'esecuzione, ma nonostante tutto, molto utilizzato.

Ma allora qual'è il problema? Il titolo di questo articolo recita: "HTML::TagReader". Noi fino ad ora abbiamo parlato di soluzioni molto semplici che non richiedono alcun modulo esterno. Ci deve essere, quindi, qualcosa che non va nella soluzione che ho proposto fino ad ora:

- Purtroppo la maggior parte dei file HTML presenti sono imperfetti. Vi sono milioni di pagine che, per esempio, contengono del codice C che, se scritto in HTML, appare così:
  ```
  if ( limit > 3) ....
  ```
  invece di
  ```
  if ( limit &gt; 3) ....
  ```
  In HTML "<" dovrebbe essere l`inizio di un tag e ">" ne dovrebbe essere la sua chiusura. Nessuno di essi dovrebbe mai apparire nel testo stesso. Questo però non è sempre così e molti browser mostrano entrambi gli stili in modo corretto, nascondendo quindi l`eventuale errore.

- Cambiare il valore di "$/" influisce sull'intero programma. Se volete processare un altro file, riga per riga, mentre sta trattando il file html, potreste incappare in molti problemi.

In altre parole, vi sono pochissimi casi in sui si possa ricorrere all'utilizzo della viariabile "$/" (INPUT_RECORD_SEPARATOR).

Ho alcuni programmi d'esempio per voi che sfruttano il sistema fino a qui descritto. Ho comunque perferito assegnare alla variabile "$/" il valore "<" in quanto i browser web non sono in grado di gestire il carattere "<" se posizionato erroneamente al contrario di quanto avviene con il carattere ">", e comunque sono molte meno le pagine web con il carattere "<" posizionato erroneamente che non quelle con il carattere ">". Questo semplice programma l'ho chiamato tr_tagcontentgrep (un click per vederlo) e all'interno del medesimo potete anche vedere il codice che ho utilizzato per tenere traccia del numero di riga. tr_tagcontentgrep può anche essere utilizzato per cercare una stringa di un tag all'interno del file medesimo (per esempio "img"). La ricerca di parte del tag può anche espandersi su più righe. Vediamo un esempio:

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 risolve i due problemi che si hanno quando si ricorre alla modifica della variabile INPUT_RECORD_SEPARATOR ed offre un sistema molto più apprezzabile per separare il testo dai tag. Oltretutto non è avido di risorse come HTML:Parser ed offre anche la possibilità di interfacciarsi meglio con noi: ci fornisce un metodo per leggere da tag a tag.

Abbiamo divagato fin troppo. Ecco come usarlo. Prima di tutto dovrete scrivere
use HTML::TagReader;
all'interno del vosto codice per poter caricare in memoria questo modulo. Per utilizzarlo dovrete semplicemente scrivere
my $p=new HTML::TagReader "filename";
per analiazzare il file "filename" ed otterrete un oggetto nella variabile $p. Ora potete chiamare la variabile $p–>gettag(0) o $p–>getbytoken(0) per ottenere il prossimo tag. gettag ci restituisce un solo tag (quella strana scritta tra < e >), mentre getbytoken ci restituisce il testo contenuto e ci dice anche se si tratti di testo o di un ulteriore tag. Con queste funzioni risulta molto facile processare file html. Ecco che diviene uno strumento essenziale per mantere grossi siti web. La sintassi completa e una spiegazione più esaustiva può essere trovata sulla pagina del manuale di HTML::TagReader.

Eccoci ora ad un vero esempio di funzionamento del programma. Ci faremmo stampare i titoli dei documenti html che gli sottoporremo:

```
#!/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:
```

Come funziona? Andiamo al leggere il file html per mezzo di $p−>getbytoken(0) e quando troviamo la stringa <title> o <Title> o <TITLE> (ci sono restituiti come $tagtype equivalente a "title") andremmo ad assegnare un flag ($printnow) per iniziare la stampa del titolo e non appena troveremo la stringa </title> interromperemo la stampa.
Utilizzare il programma come qui di seguito mostrato:

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

Certo tutto questo lo possiamo integrare anche con il programma tr_tagcontentgrep. Sicuramente un poco più breve e più semplice da scrivere:

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

Lo script ora è molto breve e non ha particolari strumenti per gestire gli errori, ma è completamente funzionale. Per cercare i tag che contengano al loro interno la stringra "gif" dovremmo digitare:

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 ulteriore esempio? Qui abbiamo un programma che rimuove il tag inerente la modifica dei caratteri, ovvero i tag: <font...> e . Questo tipo di tag viene spesso utilizzato in maniera esagerata da alcuni

scadenti edito grafici per html e spesso questo massiccio uso crea problemi guardando le pagine con diversi browser e con diverse risoluzioni video. Questa versione semplificata rimuove tutti i tag "font". Nessuno vi vieta di modificarla per far si che rimuova solo determinate caratteristiche dei caratteri (il tipo, il colore, la dimensione) lasciando il resto del tag intatto.

```
#!/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:
```

Come potete notare è molto semplice scrivere programmi molto utili con ppchissime righe di codide.
Il sorgente del pacchetto HTML::TagReader contiene alcuni esempi ed applicazioni del pacchetto stesso (trovate i riferimenti a ciò alla fine dell'articolo):

- tr_blck –– controlla la presenza di link non validi nella pagine HTML
- tr_llnk –– ci mostra la lista dei link nel file HTML
- tr_xlnk –– espande i link che puntano ad una cartella nel file indice della medesima
- tr_mvlnk –– modifica i tag nei file HTML per mezzo dei comandi di perl.
- tr_staticssi –– espande le direttive SSI #include virtual e #exec cmd trasformando la pagina in html statico.
- tr_imgaddsize –– aggiunge il flag altezza (height=) e larghezza (width=) dell'immagine al tag <img src=...>

tr_xlnk and tr_staticssi sono molto utili quando vogliate creare dei cdrom partendo da un sito web. Per esempio, quando digitate http://www.linuxfocus.org/ il server web vi farà vedere il seguente indirizzo http://www.linuxfocus.org/index.html (caricherà automaticamente la pagina index.html). Se tuttavia copiare tutti i file del sito web sul cdrom ed accedete al cdrom con il vostro browser web, vedete solo una lista di file e cartelle invece del file index.html. La prima società che fece la prima edizione su cdrom di LinuxFocus fece questo errore e fù assai scomodo consulatare il cdrom. Ora utilizzano tr_xlnk per ottenere i dati da mettere sul cdrom, e, difatti, ora i cdrom funzionano perfettamente.

Sono sicuro che troverete HTML::TagReader uno strumento molto ultile I am sure you will find HTML::TagReader useful. Happy programming!

Bibliografie e riferimenti

- La pagina del manuale di HTML::TagReader
- Perl tutorial: Perl III (Gennaio 2000)
- Il programma tr_tagcontentgrep(ovvero quello che non ricorre alla funzione HTML::TagReader): tr_tagcontentgrep (formato testo) o tr_tagcontentgrep (formato html)

- Il codice sorgente di HTML:TagReader:
  http://cpan.org/authors/id/G/GU/GUS/
  o
  http://main.linuxfocus.org/~guido/
- Tidy può essere uno strumento essenziale per il web designer: tidy, uno stumento per la verifica della sintassi dell'html
  Come utilizzare tidy? Semplicissimo:
  `tidy -e file.html`
  Ci mostrerà tutti gli errori html
  `tidy -im -raw file.html`
  ci permetterà di modificare il file e di dargli un migliore aspetto nella sua formattazione. Permetterà anche di correggere degli errori (in quanto spesso tidy è in gradi di indovinare quale fosse quello che si voleva fare).

2005−01−10, generated by lfparser_pdf version 2.51