

Re: Parsing Baseball Stats

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-08/msg00430.html>

- *From:* "Paul McGuire" <ptmcg@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 25 Jul 2006 23:01:35 GMT
-

"Anthra Norell" <anthra.norell@xxxxxxxxxxxxxx> wrote in message
<news:mailman.8551.1153861590.27775.python-list@xxxxxxxxxxxxxx>

Paul,

I think self-respect is a very essential attitude and I am pleased to know that you value it as much as I do.

The off topic thus dispatched, let me first express my appreciation or your interest. Next I agree to anything you say about

pyparsing, because I know little about it. Perhaps you will agree that parsing is overkill if you don't need to. Correct me if I'm

wrong, but I have a hunch that HTM tags don't nest. So you can actually strip them with two or three lines of plain python code,

looking at the letters one by one.

As to my proposal I can only invite you to run my program first and then pass judgment, unlike the clerical Authorities in the

story who steadfastly refused to take a peek through Galileo's telescope, arguing that, since planets didn't have rings, taking a

peek was an exercise of utter futility.

And do tell me what a TOS is. If I am missing a point it is rather this one. Does it have to do with copyright? How would it

be then if you did your pyparsing demo on a site with a more accommodating TOS?

Re: Parsing Baseball Stats

Regards

Frederic

Frederic –

HTML parsing is one of those slippery slopes – or perhaps "tar babies" might be a better metaphor – that starts out as a simple problem, but then one exception after the next drags the solution out for daaaays. Probably once or twice a week, there is a posting here from someone trying to extract data from a website, usually something like trying to pull the href's out of some random `<a>` (is for "anchor") tags, usually followed by a well-meaning regexp suggestion along the lines of "Use '``', followed by "oh wait, I forgot to mention that sometimes there are other attributes in the tag besides href, or sometimes the referenced url is not in double quotes, but single quotes, or no quotes, or href is sometimes HREF, or Href, or there are spaces around the '=' sign, or spaces before the closing '>', or there are two spaces between the 'a' and 'href', followed by "no problem, just use *this* regexp: `(@#\$(#&\$)q(\|\|< *w&)(Q&!!!!`". You are correct – in general, HTML tags themselves do not nest, and this is certainly a blessing. But HTML tags expressions *do* nest – lists within lists, tables within tables, bold within italic, italic within bold, etc. And HTML, being less well-behaved than XML, also allows tags to overlap, as in `<foo><bar></foo></bar>`. Not to mention, some HTML contains comments, so href's inside comments *probably* want to be ignored.

So utilities like BeautifulSoup, targeted specifically at cracking less-than-pristine HTML, come to the fore, to bring some order to this parsing chaos. pyparsing is more generic than BeautifulSoup, and I believe less HTML-capable, especially with overlapping tags. But it does include some out-of-the-box helpers, such as `makeHTMLTags("tagname")`, which generates opening and closing `<tagname>` and `</tagname>` tag expressions, including handling for odd casing, unpredictable whitespace, and unanticipated tag attributes. And for applications such as this, pyparsing (I believe) provides some programming interfaces that are easier to deal with than the BeautifulSoup accessors I've seen posted here.

I'm less and less open to the "parsing is overkill, I'll just slap together a regexp" line of reasoning. With a little practice, pyparsing apps can come together fairly quickly, and keep their readability for a long time. I find regexp's, which I *don't* use every day, and whose typoglyphics quickly bleed out of my frontal lobe, to lose their readability over time, especially when the expression to be cracked contains any of the regex special characters, namely `(, [,], ., /, ^, $, +, *, ?, \, ...` they become a riddle of backslashes. The difficulty is that regexp's commingle their semantic punctuation with the text they are trying to process, and it demands an extra mental focus to decipher them (doubly so when trying to debug them!).

Re: Parsing Baseball Stats

I try to save my (limited number of remaining) mentally focused moments for things like overall system design, and so I wrote pyparsing to help separate the syntax and mental processing used for text content vs. pattern symbology. Pyparsing uses explicit Python classes, such as Optional, OneOrMore, Group, Literal, and Word, with *some* operator shortcuts, such as + for And, | for match-first-alternative Or, ^ for match-longest-alternative Or, and ~ for Not. Yes, the expressions are not as terse as regexps, and they are not as fast at runtime. But they are quickly graspable at a glance. Here is a BNF for part of the Verilog language, and the corresponding code in the Verilog language parser:

```
<UDP> ::= primitive <name_of_UDP> ( <name_of_variable>
<,<name_of_variable>>* ) ;
<UDP_declaration>+
<UDP_initial_statement>?
<table_definition>
endprimitive
```

```
udp = Group( "primitive" + identifier +
"(" + Group( delimitedList( identifier ) ) + ")" + semi +
OneOrMore( udpDeclaration ) +
Optional( udpInitialStmt ) +
udpTableDefn +
"endprimitive" )
```

and the udp expression is inherently tolerant of random embedded whitespace (and comments too, as provided for later in the code).

Is it overkill to use this same tool to allow me to write:

```
anchorStart,anchorEnd = makeHTMLTags("a")
print [ t.href for t,s,e in anchorStart.scanString(htmlSourceText) ]
```

to list out the href's in an arbitrary body of HTML, independent of whitespace, presence of other attributes, casing of tag and attribute names, and other typical pitfalls of HTML parsing? (Note: this 2-liner does not handle HTML comments, this would require at least one more line, something like "aStart.ignore(htmlComment)", but I haven't tested this for this mini-manifesto.)

Frederic, I'm sorry, but I probably wont get around to giving SE a try. I'm not that interested in becoming proficient in writing (or even just being able to read) expressions such as these:

```
Tag_Stripper = SE.SE ("~<.*?>~=" " ~<[<>]*~=" " ~[<>]*>~=" ')
CSV_Maker = SE.SE (' "~\s+~=(9)" ')
CSV_Maker = SE.SE (" ~\s+~=",')
```

If these make sense to you, more power to you. If I were to puzzle them out this evening, the gift would be lost within a week. I can't afford to lose my grasp of code I've written in so short a time.

As for "TOS", it is "Terms of Service," and it's kind of like copyright.

Re: Parsing Baseball Stats

It's one of those unilateral contracts that basically says, "I offer this information, but only if you follow my terms – such as, don't rip off my data just to set up a competing service, or to sell the data in some repackaged form – and if you don't like my terms, you're welcome to not use my service" (liberally translated from the legalese). Friendly consumer sites like Google, UPS, FedEx, Amazon all list some form of TOS stating "you're welcome to use our website as long as it is within the bounds of normal consumer use – and no bots, agents, etc. allowed, 'cause they'll bring our server to its knees and the innocent bystander users end up losing out because of your selfishness/greed/lack of consideration for your fellow human person." As much as the content on the Internet is freely available, it isn't always free. (Pyparsing comes with some simple HTML scrapers that use public websites, such as NIST's listing of public NTP servers.)

So what started out as a little joke (microscopic, even) has eventually touched a nerve, so thanks and apologies to those who have read this whole mess. Frederic, SE looks like a killer – may it become the next regexp!

-- Paul