

# Re: traversing a variable with regex instead of a file

**Source:** <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2003-10/0427.html>

---

**From:** James Edward Gray II (*james\_at\_grayproductions.net*)

**Date:** 10/10/03

Date: Fri, 10 Oct 2003 12:15:52 -0500  
To: angie ahl <angie@vertebrate.co.uk>

On Friday, October 10, 2003, at 10:51 AM, angie ahl wrote:

> on 2003-10-10 James Edward Gray II said:  
>  
>> *Keep your replies on the list, so you can get help from all the people  
>> smarter than me. ;)*  
>  
> *If there are people smarter than you out there I must be an amoeba ;)*

There are some very brilliant people hiding here. Stick around and you'll spot on eventually...

>> *Okay, why put this inside an if block. If it doesn't find a match it  
>> will fail and do nothing, which is what you want, right? I don't  
>> think  
>> you need the if.*  
>  
> *Good point.*

Thanks, I thought so too, but you forgot to take my advice. You still don't need the if block you have in the code. :D

>> `$content =~ s/^([^n]*)($kw)/substr($1, 0, 2) ne 'qz' ? "$1\n$2\n" :`  
>> `"$1$2"/mge;`  
>>  
>  
> *Ok. I had to stop to pick myself up off the floor then. WOW.*

I love that silly little /e modifier myself. I use it a lot. It wouldn't surprise me if one of the Regex Gurus could do it even better though. I'm still reading Mastering Regular Expressions myself. ;)

> *# get line breaks to make <br>'s at the end*  
> `$content =~ s/\n/-qbr-/g;`  
>

perl.beginners: Re: traversing a variable with regex instead of a file

```
> # find markup and add markers so it doesn't get processed by regex,  
> # no keyword links to be made inside other tags  
> $content =~ s/(\[img/page/link/mp3\]=.*?\])/^nqz$1\n/g;  
>  
> # find HTML so it doesn't get processed by regex,  
> # no keyword links to be made inside valid HTML  
> $content =~ s/(<.*?>)/^nqz$1\n/g;
```

There are some excellent HTML parser modules on the CPAN that could possibly help you do this kind of thing, just FYI.

```
> for my $href ( @Keywords ) {  
>  
> # get each keyword and llok for it in content.  
> for $kw ( keys %$href ) {  
> if ( $content =~ ^b($kw)\b/g ) {
```

The if above is not needed, as near as I can tell, though you should probably add the \b...\b around the \$kw in my expression. There's certainly no reason to use a global search above, since you're just checking to see if it's present.

```
> # do the very clever reg with help from and thanks to  
> # james@grayproductions.net  
> $content =~ s/^(\[^\n]*)($kw)/substr($1, 0, 2) ne 'qz' ?  
> "$1\nqz[link=\"$href->{$kw}\" title=\"$2\"]\n" : "$1$2"/mge;  
> }  
> }  
> }  
>  
> # clean up those line breaks and markers;  
> $content =~ s/\n(qz)?//g;  
>  
> # put in <br>'s  
> $content =~ s/-qbr-/<br>\n/g;  
>  
> print $content;  
> _____  
>  
> As you can see I've adapted your regex a little to put in the full  
> markup around  
> the keyword.
```

Making changes already. You're a natural.

```
> The regex itself made perfect sense, it was the  
>  
> "" ? "" : "" bit that I've never seen before. That's really useful.  
>  
> I assume it means  
>
```

perl.beginners: Re: traversing a variable with regex instead of a file

> *"if statement" ? "do if true" : "do if false"*

You assume right. The Ternary Operator (or Conditional Operator) is a simple if/else shorthand. It's not too popular, probably for readability reasons, but I use it a lot in things like Regexes and map(). If you want to know more:

perldoc perlop

Then type:

/Conditional Operator

> *Please do correct me if I'm wrong. What do you call that? I think I'm going it be using that quite a bit ;)*  
>  
> *do I even need the if false bit in this case?*

Yes, and this is important. The way I wrote the match, it will find ALL \$kw occurrences, even the ones on qz lines. So after we learn we're on a qz line, we need to put things back with our replace, so it looks like we never messed with it.

>> *I used the /e modifier for the replacement, which allows me to use Perl code in there. It's pretty simple. If the line didn't start with a qz, we do a normal replace.*  
>  
> *That's going in my BBEdit gold dust code snippets glossary.*

A BBEdit user, and by extension a Mac user too, eh? That settles it, we're going to be great friends you and I. ;)

Unfortunately, the /e trick doesn't work in BBEdit's Search/Replace dialogs, though it has excellent Grep support.

>> *Your right about it being inefficient, of course. It was easier to read than my Regex though, eh? <laughs>*  
>  
> *Are you implying that regex isn't easy to read ;)*

Me, I love a little line noise in the morning, but that might be why I'm a Super Geek.

>> *The first choice may be slow, but on modern computers they may both work in the blink of an eye. Save worrying about speed for when you need to and try and keep your life as a programmer as easy as possible until then.*  
>  
> *Sadly then is now. That's why I joined up to this list today ;)*

Re: traversing a variable with regex instead of a file

perl.beginners: Re: traversing a variable with regex instead of a file

>  
> *This code will be run on every single page of a website, in one go. So  
> it needs  
> to be as efficient as physically possible. The site will only be a few  
> hundred  
> pages, and not all pages will always be processed. It's a system that  
> makes it's  
> own links and maintains them, so everytime a page's keywords change  
> this has to  
> be done to all pages that contain that keyword.*

Fair enough. Just remember that measuring the different between two pieces of code that both run in under a second is usually pretty pointless. Start optimizing AFTER you start waiting on executions, not before.

> *I know this is not a task for the beginner, but this is actually  
> version 3 of  
> the code. my old programming language started to show it's dislike for  
> regex.*

You'll do fine and you know where to get help if you need it.

>>> *If you have any suggestions I would be most grateful to hear them.  
>>  
>> Those are my best shots. Hope they help.  
>  
> They did, thank you so much.*

Happy to help. Welcome to the list.

James