

Re: RFC: utils.pm

Source: <http://coding.derkeiler.com/Archive/Perl/comp.lang.perl.misc/2004-02/1271.html>

From: Ben Morrow (usenet_at_morrow.me.uk)

Date: 02/11/04

Date: Wed, 11 Feb 2004 22:06:45 +0000 (UTC)

Tore Aursand <tore@aurand.no> wrote:

> *On Wed, 11 Feb 2004 15:23:37 +0000, Ben Morrow wrote:*

> >> *split_csv(\$value) – Easy CSV splitting. (*)*

>

> > *As you say, there are modules to do this. Or*

> >

> > *@values = split /,/ , \$values;*

>

> *Not quite as powerful as this one;*

>

> *my @array = ();*

> *push(@array, \$+) while \$string =~ m{*

> *"([^\"]*(?:\\.[^\"]*)*)",? # groups the phrase inside the quotes*

 ^^

 Unnecessary

> *| ([^,]+),?*

> *|,*

> *}gx;*

> *push(@array, undef) if (substr(\$string, -1, 1) eq ',');*

use Text::Balanced qw/extract_multiple extract_delimited/;

extract_multiple \$string,

 [sub { extract_delimited \$_[0], "" }, qr/[^\,]*/],

 undef, 1;

Or, there are modules to do it.

> >> *random_string(\$length) – Generates a random string \$length*

> >> *characters long.*

>

> > *This is a little harder... also *much* less useful.*

>

> *Maybe, but I find myself constantly using this one when I want to create*

> *cookies. Nice to have. :)*

Well, yeah; but hardly of general utility. Perl != CGI.

```
> >> as_string( $value, [$default] ) – Always returns a defined value,  
> >> optionally $default if $value  
> >> isn't defined.  
>  
> > Eh what? You don't *need* to cast in Perl.  
> >  
> > $value || $default  
>  
> This only returns $default if $value isn't true. So if we have this  
> function:
```

Yeah, yeah; I know that. It's good enough most of the time, though. Roll on 5.10 when (I think) they're going to put // in (which tests for definedness rather than truth).

```
> > or defined($value) ? $value : $default  
>  
> Better, but I'm tired to writing this god damn line each time I want to  
> make sure that a value is defined. :)
```

But it's a lot clearer... if the default is not supplied, your function is a noop; otherwise, it is a defined-or-default function. Not what I'd call `as_string`.

```
> When it comes to "casting". Maybe it's not the correct word to use, but  
> when dealing with ie. CGI parameters, I find my functions useful. Here's  
> an example;  
>  
> my $firstname = as_string( $cgi->param('firstname') );
```

What, precisely, is the difference between this and

```
my $firstname = $cgi->param('firstname');
```

?

```
> >> as_int( $value ) – Always returns $value as an integer.  
>  
> > err.. int($value)  
>  
> What if $value isn't a number? Ah. A warning occurs, of course. I don't  
> want warnings.
```

So turn it off. Not exactly hard. And, again, an explicit statement to a reader of your code that you are prepared to tolerate non-numeric inputs.

```
> >> as_decimal( $value, [$decimals] ) – Always returns $value as a  
> >> decimal number with $decimals
```

```
> >> numbers after the decimal point.
>
> > sprintf
>
> Exactly what 'as_decimal' uses, except that I prefer writing
>
> my $value = as_decimal( 1.23456789, 2 );
>
> instead of
>
> my $value = sprintf( '%.2f', 1.23456789 );
```

Why? This slightly peculiar taste of yours hardly justifies inclusion in a core module.

```
> >> as_boolean( $value ) – Always returns $value as a boolean value (ie.
> >> TRUE/1 or FALSE/0).
>
> > !!$value
>
> Yeah, but 'as_boolean' takes care of "other" boolean values to;
>
> sub as_boolean {
> my $value = as_string( shift );
> return ( $value =~ m,^1|y|yes|on|true$,i ) ? 1 : 0;
> }
```

Whoa... not what it said on the tin **at* *all**. I'd definitely much prefer that to be explicit in the code; and you **really** don't need that *?: m//* already returns a boolean value...

```
> >> as_date( $value ) – Always returns $value as a date (YYYY-MM-DD).
> >> as_time( $value ) – Always returns $value as a time (HH:MM:SS).
> >> as_datetime( $value ) – Always returns $value as a datetime (which
> >> means combining as_date() and as_time()).
>
> > POSIX::strftime
>
> 'as_date', 'as_time' and 'as_datetime' are – of course – a lot easier to
> use than 'strftime'. :)
```

'Of course'? I think not. Apart from anything else, it is clear from a *strftime* call what format the result will be in, whereas your functions are not.

```
> >> VALIDATION
> >> Each of the CASTING functions also have a is_* function, which returns
> >> TRUE/1 or FALSE/0 depending on whether the input argument conforms to
> >> the datatype.
>
> > Ummm... everything is (can be) a string.
```



```
> sub is_decimal {
> my $value = as_string( shift );
> return ( $value =~ m,^[+-]?(\d+)?\.\d+,$ ) ? 1 : 0;
          ^^^^^^
          \d*
> }
```

Right. Once. As I said.

```
> >> random_number( $min, $max ) – Returns a random number in the range
> >> $min to $max.
>
> > (rand * ($max - $min)) + $min
>
> Right, but what if only one of $min or $max is known?
```

Err... what if? What do you *want* to happen in that case? (And how do you implement it?... insofar as I am aware, the random-number generator will not accept 'infinity' as an argument.)

```
> >> format_number( $value, $separator ) – Formats a number with a given
> >> separator; 1234 becomes 1,234.
>
> > Less trivial... is probably better done by something locale-aware.
>
> Who says that a 'Utils' module can't be locale-aware? :)
```

OK; but personally I'd rather that sprintf be made aware of the ' modifier from SUSv2 (which does just this).

```
> >> unique( $arrayref ) – Returns only the unique elements in $array.
> >> intersection( $arrayref1, $arrayref2 ) – Computes the intersection
> >> of two array references.
> >> union( $arrayref1, $arrayref2 ) – Computes the union of two array
> >> references.
>
> > These should all be in a module called Set::Util... feel free to write
> > it.
>
> It isn't in a module already? Doesn't it fit into List::Util?
```

Well, maybe, but they aren't in there. Personally, I'd have said there's a useful distinction to be made between 'operations on ordered sets of data' (List::Util) and 'operations on unordered sets of data' (Set::Util).

Anyway, if you thought it was in a module already *why aren't you using it*?

Ben

comp.lang.perl.misc: Re: RFC: utils.pm

--

```
If you put all the prophets, | You'd have so much more reason
Mystics and saints          | Than ever was born
In one room together,       | Out of all of the conflicts of time.
ben@morrow.me.uk |-----+-----| The Levellers, 'Believers'
```