

## Re: easily embedding html into Lisp

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-09/0957.html>

---

*From:* Rob Warnock ([rpw3\\_at\\_rpw3.org](mailto:rpw3_at_rpw3.org))

*Date:* 09/15/04

Date: Wed, 15 Sep 2004 06:19:31 -0500

Andreas Thiele <[nospam6329@nospam.com](mailto:nospam6329@nospam.com)> wrote:

+-----  
| I'd like to introduce a simple approach of embedding html into Lisp.  
+-----

You should look at <<http://www.cliki.net/Lisp%20Markup%20Languages>>, which lists several readily-available packages which already do this (or similar). I tend to use HTOUT, but CL-WHO is also nice. Also see <<http://www.cliki.net/HTML-from-sexpr>>.

+-----  
| Let's assume we already have three functions 'table 'tr and 'td.  
| Our html code to make a table then might look like  
| (table :cellpadding 0 :cellspacing 5  
| (tr (td "hello") (td "world"))) )  
| producing  
| <table cellpadding="0"  
| cellspacing="5"><tr><td>hello</td><td>world</td></tr></table>  
+-----

In HTOUT, that would be written this way.

```
(with-html-output (s stream)
  ((:table :cellpadding 0 :cellspacing 5)
   (:tr (:td "hello") (:td "world"))))
```

and generates:

```
<TABLE CELLPADDING='0'
CELLSPACING='5'><TR><TD>hello</TD><TD>world</TD></TR></TABLE>
```

+-----  
| I assume html tags can be described liked  
| <tag-name attribute1=value1 ... > content </tag-name>  
+-----

HTOUT uses keywords for tags, so you don't need to define functions for all of them. Or, if the first element of a form is itself a list

the CAR of which is a keyword, then the CAAR is taken as the tag and the CDAR is a property list of "attribute value..." (as shown in the example above).

```
+-----  
| Now let's go one step ahead. For each html tag there has to be such a  
| function definition.  
+-----
```

Packages such as HTOUT & CL-WHO don't require this, using symbols in the keyword package as tag markers instead. Any list whose CAR is not a keyword or whose CAR is not a list starting with a keyword is taken to be pure Lisp code, which is simply passed through to the compiler. Also, inside a WITH-HTML-OUTPUT, macrolets are automatically defined for several common operations, including going back into HTML mode from Lisp code [note the use of HTM in the following example].

[There is an alternate syntax available which puts the attributes in a required list after the tag, but I won't go into that here.]

For example, instead of just saying "hello" and "world", suppose you wanted to write a function that took an entire list of lists, and gave you an HTML table row per element, with a data item per sub-element. Furthermore, suppose you want the first element to be the column-head labels (that is, use <TH> instead of <TD> on the first row). Here's one version [the LFD calls produce newlines in the HTML, only needed for human readability]:

```
(defun list-html-table (lists &key (stream *standard-output*))  
  (let ((column-names (car lists))  
        (rows (cdr lists)))  
    (with-html-output (s stream)  
      (:h3 (fmt "Results: ~d rows" (length rows)))  
      (lfd)  
      ((:table :border 1 :cellspacing 0 :cellpadding 1) ; make compact  
       (:tr (lfd)  
            (loop for name in column-names do  
              (htm ((:th :nowrap) name) (lfd))))  
       (loop for row in rows do  
         (htm (:tr (lfd)  
                (loop for v in row do  
                  (let ((vv (if (or (null v) (string-equal v ""))  
                                "&nbsp;"  
                                (escape-string v))))  
                    (htm ((:td :nowrap) vv) (lfd))))))))  
      (lfd))))
```

So this call:

```
(list-html-table  
  '(("Name" "Room" "Phone")
```

comp.lang.lisp: Re: easily embedding html into Lisp

```
("Joe" "37A" "2345")  
("Bill" "37B" "5432")  
("Sally" "22C" "1234"))
```

outputs this:

```
<H3>Results: 3 rows</H3>  
<TABLE BORDER='1' CELLSPACING='0' CELLPADDING='1'><TR>  
<TH NOWRAP>Name</TH>  
<TH NOWRAP>Room</TH>  
<TH NOWRAP>Phone</TH>  
</TR><TR>  
<TD NOWRAP>Joe</TD>  
<TD NOWRAP>37A</TD>  
<TD NOWRAP>2345</TD>  
</TR><TR>  
<TD NOWRAP>Bill</TD>  
<TD NOWRAP>37B</TD>  
<TD NOWRAP>5432</TD>  
</TR><TR>  
<TD NOWRAP>Sally</TD>  
<TD NOWRAP>22C</TD>  
<TD NOWRAP>1234</TD>  
</TR></TABLE>
```

+-----  
| When talking about html I must mention Kevin Rosenberg's LML package  
| at <http://lml.b9.com>  
+-----

That one is also listed on the above-mentioned CLiki pages, as well as  
his LML2 (for generating XHTML documents).

+-----  
| I am interested in your criticism to my approach.  
+-----

It certainly works. I used something very similar to it [in Scheme]...  
before I found HTOUT. ;-}

-Rob

-----  
Rob Warnock <[rpw3@rpw3.org](mailto:rpw3@rpw3.org)>  
627 26th Avenue <[URL:http://rpw3.org/](http://rpw3.org/)>  
San Mateo, CA 94403 (650)572-2607