Help with an interactive Emacs Lisp function for replacing text

Asked 15 years, 11 months ago Modified 15 years, 11 months ago Viewed 1k times

4

I've been using Emacs for a couple months now, and I want to get started in elisp programming. Specifically, I'd like to write my own `interactive` function. However, I'm more than a bit lost. `(interactive ...)` has tons of options and I'm not sure which one I want. Then, I don't really know the names of the functions I need. If someone could kindly help me turn my pseudocode into real code, I would be mighty appreciative! (And as always, any links to informative places would be good. Right now I've just been reading this.)

Here is pseudocode for what I'd like to do:

```
(defun my-func (buffer) ; I think I need the buffer as an arg?
  "does some replacements"
  (interactive ???) ; ?
  (let (replacements (list
   '("a-regexp-string" . "a-replacement-string-with-backreferences")
   ...)) ; more of the above
    (while replacements
      (let (current (car replacements)) ; get a regexp-replacement pair
        (some-regexp-replace-func buffer (car current) (cdr current)) ; do the
replacement
        (setq replacements (cdr replacements))))))
```

emacs elisp

Share Improve this question Follow

asked Jan 17, 2009 at 22:49

J Cooper

17.1k ● 13 ● 68 ● 112

1 Answer

Sorted by: Highest score (default)

5

First, from the looks of your function you would probably be doing it in the current buffer, so no, you don't need to have a 'buffer' argument. If that's a bad assumption, I can change the code. Next, in a 'let' if you are assigning to variables you need another set of parens around each pair of var/value. Finally, when looping through a list I prefer to use functional-programming-like functions (mapcar, mapc, etc.). I'll try to inline some comments here:

```
(defun my-func ()
  "Do some replacements"
  (interactive)
  (let ((replacements (list '("foo" . "bar")
                            '("baz" . "quux"))))
    (save-excursion ; So point isn't moved after this function
      (mapc (lambda (x) ; Go through the list, with this 'inline' function
                        ; being called with each element as the variable 'x'
              (goto-char (point-min)) ; Start at the beginning of the buffer
              (while (re-search-forward (car x) nil t) ; Search for the car of
the replacement
                (replace-match (cdr x)))) ; And replace it with the cdr
            replacements)))) ; The list we're mapc'ing through
```

As for what to read, I'd suggest the Elisp manual that comes with Emacs.

Share

Improve this answer

Follow

edited Jan 18, 2009 at 4:03

answered Jan 18, 2009 at 1:03

scottfrazer

17.3k ● 4 ● 53 ● 51

Thank you very much, that helped me understand what was going on and get on the right track. I just had to use `re-search-forward` instead of the non-regexp version, and not pass `t` to `replace-match` so that referencing of matched groups works. Thanks again! – J Cooper Jan 18, 2009 at 3:48

Yeah, I was just coming back to fix those :) – scottfrazer Jan 18, 2009 at 4:05