EmacsVtlMode

VTL Syntax Coloring in Emacs

Contributed by Brian Leonard <brian@brainslug.org>

"Here's a small minor mode that does font-locking for vtl code. Nothing fancy, just coloring. As a minor mode, it insinuates itself into whatever major mode you're using, so you don't lose the major mode's goodness. One caveat: it doesn't color multi-line comments as you type. It will color them when you fontify the entire buffer (M-x font-lock-fontify-buffer).

Load this file, load the file you want to edit, enable this mode with M-x vtl-mode. Or see the documentation for a more convenient method. Enjoy."

vtl.el:

```
;;; vtl.el --- fontify velocity template language code

;; Author: Brian Leonard <brian@brainslug.org>
;; Maintainer: brian@brainslug.org
;; Keywords: extensions
;; Created: 2002-01-19

;; $Id: vtl.el 191743 2005-06-21 23:22:20Z dlr $

;;; Commentary:

;; $variable     - font-lock-variable-name-face
;; #foreach      - font-lock-keyword-face
;; #mymacro      - font-lock-function-name-face
;; ## comment    - font-lock-comment-face
;; #* comment *# - font-lock-comment-face
;;
;; $customer.Address
;;    ^^^      ^^^
;;     |        +--- font-lock-variable-name-face
;;     +--- font-lock-type-face
;;
;; $customer.getAddress()
;;    ^^^        ^^^
;;     |          +--- font-lock-function-name-face
;;     +--- font-lock-type-face
;;

;; One useful way to enable this minor mode is to put the following in your
;; .emacs:
;;
;;      (autoload 'turn-on-vtl-mode "vtl" nil t)
;;      (add-hook 'html-mode-hook 'turn-on-vtl-mode t t)
;;      (add-hook 'xml-mode-hook 'turn-on-vtl-mode t t)
;;      (add-hook 'text-mode-hook 'turn-on-vtl-mode t t)

;;; Code:

(require 'font-lock)
(require 'cl)


(defgroup vtl nil
  "Fontifies VTL code.  see http://jakarta.apache.org/velocity/"
  :group 'vtl
  :group 'font-lock
  :group 'extensions)

;;;###autoload
(defcustom vtl-mode nil
  "*If non-nil, fontify vtl code

This variable is buffer-local."
  :type 'boolean)
(make-variable-buffer-local 'vtl-mode)
```

```
;;;###autoload
(defcustom vtl-minor-mode-string " VTL"
  "*String to display in mode line when VTL Mode is enabled."
  :type 'string
  :group 'vtl)

;;;###autoload
(defun turn-on-vtl-mode ()
  "Unequivocally turn on vtl-mode (see variable documentation)."
  (interactive)
  (font-lock-mode 1)
  (vtl-mode 1))

;; Put minor mode string on the global minor-mode-alist.
;;;###autoload
(cond ((fboundp 'add-minor-mode)
       (add-minor-mode 'vtl-mode 'vtl-minor-mode-string))
      ((assq 'vtl-mode (default-value 'minor-mode-alist)))
      (t
       (setq-default minor-mode-alist
                     (append (default-value 'minor-mode-alist)
                             '((vtl-mode vtl-minor-mode-string))))))



;;;###autoload
(defun vtl-mode (&optional prefix)
  "Toggle VTL Mode.

If called interactively with no prefix argument, toggle current condition
of the mode.
If called with a positive or negative prefix argument, enable or disable
the mode, respectively."
  (interactive "P")

  (setq vtl-mode
        (if prefix
            (>= (prefix-numeric-value prefix) 0)
          (not vtl-mode)))

  (cond (vtl-mode
         ;; first, grab default
         (font-lock-mode 0)
         (font-lock-set-defaults)

         ;; add vtl regexps
         (setq font-lock-keywords
               (let ((new-keywords
                      (cond ((null font-lock-keywords)
                             vtl-keywords)
                            (t
                             (list* (car font-lock-keywords)
                                    (append (cdr font-lock-keywords)
                                            vtl-keywords))))))
                 new-keywords))

         ;; and restart font-lock
         (font-lock-mode 1)
         (font-lock-fontify-buffer))

        (t
         ;; reset to major mode's defaults
         (font-lock-mode 0)
         (font-lock-set-defaults)
         (font-lock-mode 1)
         (font-lock-fontify-buffer)))

  (and (interactive-p)
       (if vtl-mode
           (message "vtl-mode is enabled")
         (message "vtl-mode is disabled")))
```

```
  vtl-mode)



(defvar vtl-keywords
  (let
      ((directive (concat "\\(#\\(set\\|if\\|elseif\\|else\\|foreach\\|end\\|"
                          "include\\|parse\\|stop\\|macro\\)\\)"))
       (variable "\\(\\$[!]?[{]?[a-zA-Z][a-zA-Z0-9---_]*[}]?\\)")
       (property
        (concat "\\(\\$[{]?[a-zA-Z][a-zA-Z0-9---_]*\\.\\)"
                "\\([a-zA-Z][a-zA-Z0-9---_]*\\)\\([}]?\\)"))
       (method
        (concat "\\(\\$[{]?[a-zA-Z][a-zA-Z0-9---_]*\\.\\)"
                "\\([a-zA-Z][a-zA-Z0-9---_]*\\)[ ]*\\(([^)]*)\\)[}]?"))
       (vmmacro "\\(#[a-zA-Z][a-zA-Z0-9---_]*\\)[ ]*\\(([^)]*)\\)")
       (line-comment "##.*$")
       (long-comment "\\(#\\*\\([^\\*]\\|\\*[^#]\\)*\\*#\\)"))

    (list
     (list variable '(1 font-lock-variable-name-face t))
     (list property '(1 font-lock-type-face t))
     (list property '(2 font-lock-variable-name-face t))
     (list property '(3 font-lock-type-face t))
     (list method '(1 font-lock-type-face t))
     (list method '(2 font-lock-function-name-face t))
     (list vmmacro '(1 font-lock-function-name-face t))
     (list directive '(1 font-lock-keyword-face t))
     (list line-comment '(0 font-lock-comment-face t))
     (list long-comment '(0 font-lock-comment-face t)))))



(provide 'vtl)
```