Bug GNU Emacs Config

Eduardo Valencio Santos

August 4, 2023

Contents

1 PROGRAMS LOAD FIRST

1.1 Frame Config Layout

```
(setq inhibit-startup-message t)
;(load-theme 'deeper-blue t)
(global-display-line-numbers-mode 1)
(menu-bar-mode -1)
(tool-bar-mode -1)
(scroll-bar-mode -1)
(electric-pair-mode 1)
;;(global-visual-line-mode t) ;; Wrap
(setq scroll-preserve-screen-position 1)
```

1.2 Util Cofig

Rolagem mais suave

```
(setq mouse-wheel-scroll-amount '(2 ((shift) . 1))
mouse-wheel-progressive-speed nil
mouse-wheel-follow-mouse 't
scroll-step 1
scroll-margin 5)
```

Mandar os arquivos de Backup para outro lugar

```
(setq backup-directory-alist '(("." . "~/.saves")))
```

1.3 Package Manager

```
(require 'package)
(add-to-list 'package-archives
     '("melpa" . "https://melpa.org/packages/") t)
(package-initialize)
;;(package-refresh-contents)

;; Use Package
(unless (package-installed-p 'use-package)
    (package-install 'use-package))
```

1.4 Evil Mode Config

```
(use-package evil
    :ensure t
    :init
    (setq evil-want-integration t) ;; This is optional since it's already set to t by
    (setq evil-want-keybinding nil)
    :config
(evil-mode 1))

(use-package evil-collection
  :after evil
  :ensure t
  :config
    (evil-collection-init)
    (evil-collection-init))
```

1.5 General Keybinding

```
(use-package general
    :ensure t
    :config
(general-evil-setup)
(general-create-definer dt/leader-keys
    :states '(normal insert visual emacs)
    :keymaps 'override
    :prefix "SPC"
    :global-prefix "M-SPC")
(dt/leader-keys
    "." '(find-file :wk "Find File")
    "fc" '((lambda () (interactive) (find-file "~/.config/emacs/config.org")) :wk "Con
    "TAB TAB" '(comment-line :wk "Comment Lines"))
(dt/leader-keys
    "b" '(:ignore t :wk "Buffer")
    "bb" '(switch-to-buffer :wk "Switch to buffer")
    "bd" '(kill-this-buffer :wk "Kill this buffer")
    "bn" '(next-buffer :wk "Change to next buffer")
    "bp" '(previous-buffer :wk "Change to previous buffer"))
(dt/leader-keys
    "e" '(:ignore t :wk "Evaluete")
    "eb" '(eval-buffer :wk "Evaluete buffer")
    "ed" '(eval-defun :wk "Evaluete defun containing or after point")
    "es" '(eval-last-sexp :wk "Evaluete elisp expression before point")
    "ee" '(eval-expression :wk "Evaluete and elisp expression")
    "er" '(eval-region :wk "Evaluete elisp in region"))
)
```

1.6 Which Key

```
(use-package which-key
    :ensure t
    :init
(which-key-mode 1)
    :config
(setq which-key-sparator " > "))
```

1.7 ORG MODE

1.7.1 Table Content

```
(use-package toc-org
:commands toc-org-enable
:init (add-hook 'org-mode-hook 'toc-org-enable))
```

1.7.2 Org Bullets

```
(add-hook 'org-mode-hook 'org-indent-mode)
(use-package org-bullets)
(add-hook 'org-mode-hook (lambda () (org-bullets-mode 1)))
```

1.8 Tabs

```
(use-package centaur-tabs
    :ensure t
:demand
:config
(centaur-tabs-mode t)
(setq centaur-tabs-set-icons t)
:bind
("C-<prior>" . centaur-tabs-backward)
("C-<next>" . centaur-tabs-forward))
```

1.9 AutoComplete

```
(use-package auto-complete
:ensure t
:init
(ac-config-default)
(global-auto-complete-mode t)
:config
(define-key ac-mode-map (kbd "M-TAB") 'auto-complete)
(ac-set-trigger-key "TAB")
)
```

1.10 Neo Tree

```
(use-package neotree
:ensure t
:config (setq neo-theme (if (display-graphic-p) 'icons 'arrow)))
```

Configurando evil-collection para neotree

```
(defun evil-collection-neotree-setup ()
  "Set up ‘evil’ bindings for ‘neotree’."

  (evil-set-initial-state 'neotree-mode 'normal) ;; Neotree start in normal by default

  (evil-collection-define-key 'normal 'neotree-mode-map

    (kbd "RET") (neotree-make-executor
      :file-fn 'neo-open-file
      :dir-fn 'neo-open-dir)
    (kbd "<tab>") (neotree-make-executor
   :dir-fn 'neo-open-dir)
    "z" (neotree-make-executor
 :dir-fn 'neo-open-dir)
    "ZZ" 'quit-window
    "gd" (neotree-make-executor
  :dir-fn 'neo-open-dired)
    "gD" (neotree-make-executor
  :dir-fn 'neo-open-dired)
    "go" (neotree-make-executor
  :file-fn 'neo-open-file
  :dir-fn 'neo-open-dir)
    "gO" 'neotree-quick-look
    "gr" 'neotree-refresh
    "q" 'neotree-hide
    "H" 'neotree-hidden-file-toggle
    "gh" 'neotree-hidden-file-toggle
    (kbd "C-k") 'neotree-select-up-node
    "gk" 'neotree-select-up-node
    "[[" 'neotree-select-up-node
    (kbd "C-j") 'neotree-select-down-node
    "gj" 'neotree-select-down-node
    "]]" 'neotree-select-down-node
    "gv" 'neotree-open-file-in-system-application
    "c" 'neotree-create-node
    "y" 'neotree-copy-node
    "r" 'neotree-rename-node
    "R" 'neotree-change-root
    "d" 'neotree-delete-node
    "J" 'neotree-dir
    "+" 'neotree-stretch-toggle
    "=" 'neotree-stretch-toggle
    "ge" 'neotree-enter
    "j" 'neotree-next-line
    "k" 'neotree-previous-line

    ;; Unchanged keybings.
    "a" (neotree-make-executor
 :file-fn 'neo-open-file-ace-window)
    "|" (neotree-make-executor
 :file-fn 'neo-open-file-vertical-split)
    "-" (neotree-make-executor
 :file-fn 'neo-open-file-horizontal-split)
    "S" 'neotree-select-previous-sibling-node
    "s" 'neotree-select-next-sibling-node
    (kbd "C-c C-c") 'neotree-change-root
    (kbd "C-x 1") 'neotree-empty-fn
    (kbd "C-x 2") 'neotree-empty-fn
    (kbd "C-x 3") 'neotree-empty-fn
    (kbd "C-x C-f") 'find-file-other-window
    (kbd "C-c C-f") 'find-file-other-window))
```

1.11 All the icons

```
(use-package all-the-icons
:ensure t
:config)
```

1.12 Thema

```
(use-package doom-themes
  :ensure t
  :config
  ;; Global settings (defaults)
  (setq doom-themes-enable-bold t    ; if nil, bold is universally disabled
doom-themes-enable-italic t) ; if nil, italics is universally disabled
(load-theme 'doom-ayu-mirage t)

;; Enable flashing mode-line on errors
(doom-themes-visual-bell-config)
;; Enable custom neotree theme (all-the-icons must be installed!)
(doom-themes-neotree-config)
;; or for treemacs users
(setq doom-themes-treemacs-theme "doom-atom") ; use "doom-colors" for less minimal i
(doom-themes-treemacs-config)
;; Corrects (and improves) org-mode's native fontification.
(doom-themes-org-config))
```

1.13 Set JAVA ENV

1.13.1 LSP

1. Company Auto complete O LSP roda automaticamento o `company-capf`

   ```
   (use-package company :ensure t)
   ```

2. Yasnippet Template para abreviação e associação de texto

   ```
   (use-package yasnippet
   :ensure t
   :config (yas-global-mode))
   (use-package yasnippet-snippets :ensure t)
   ```

3. FlyCheck

   ```
   (use-package flycheck :ensure t :init (global-flycheck-mode))
   ```

4. Debug Protocol

   ```
   (use-package dap-mode
     :ensure t
     :after (lsp-mode)
     :functions dap-hydra/nil
     :config
     (require 'dap-java)
     :bind (:map lsp-mode-map
    ("<f5>" . dap-debug)
   ```

```
 ("M-<f5>" . dap-hydra))
  :hook ((dap-mode . dap-ui-mode)
    (dap-session-created . (lambda (&_rest) (dap-hydra)))
    (dap-terminated . (lambda (&_rest) (dap-hydra/nil)))))

(use-package dap-java :ensure nil)
```

5. Treemacs Elementos graficos parao LISP UI

```
(use-package lsp-treemacs
  :after (lsp-mode treemacs)
  :ensure t
  :commands lsp-treemacs-errors-list
  :bind (:map lsp-mode-map
 ("M-9" . lsp-treemacs-errors-list)))

(use-package treemacs
  :ensure t
  :commands (treemacs)
  :after (lsp-mode))
```

6. LSP UI Dependencia para vários pacotes para que possa existir elementos ui

```
(use-package lsp-ui
:ensure t
:after (lsp-mode)
:bind (:map lsp-ui-mode-map
 ([remap xref-find-definitions] . lsp-ui-peek-find-definitions)
 ([remap xref-find-references] . lsp-ui-peek-find-references))
:init (setq lsp-ui-doc-delay 1.5
      lsp-ui-doc-position 'bottom
  lsp-ui-doc-max-width 100
))
```

7. Helm LSP Fornece funcionalidades para trabalhar com código

```
(use-package helm-lsp
:ensure t
:after (lsp-mode)
:commands (helm-lsp-workspace-symbol)
:init (define-key lsp-mode-map [remap xref-find-apropos] #'helm-lsp-workspace-sym
```

8. LSP Instalando o pacote principal do lsp e integrando com o Which key

   ```
   (use-package lsp-mode
   :ensure t
   :hook (
      (lsp-mode . lsp-enable-which-key-integration)
      (java-mode . #'lsp-deferred)
   )
   :init (setq
       lsp-keymap-prefix "C-c l"              ; this is for which-key integration doc
       lsp-enable-file-watchers nil
       read-process-output-max (* 1024 1024)  ; 1 mb
       lsp-completion-provider :capf
       lsp-idle-delay 0.500
   )
   :config
       (setq lsp-intelephense-multi-root nil) ; don't scan unnecessary projects
       (with-eval-after-load 'lsp-intelephense
       (setf (lsp--client-multi-root (gethash 'iph lsp-clients)) nil))
   (define-key lsp-mode-map (kbd lsp-keymap-prefix) nil)
   (define-key lsp-mode-map (kbd "C-c l") lsp-command-map)
   )
   ```

   Com isso é possivel iniciar um server lsp digitando o C-c l s s ou buscando com C-c l

9. LSP Java

   ```
   (use-package lsp-java
   :ensure t
   :config (add-hook 'java-mode-hook 'lsp))

   (setenv "JAVA_HOME"  "/opt/jdk/jdk-20.0.2")
   ;(setq lsp-java-java-path "/opt/jdk/jdk-20.0.2")

   (use-package exec-path-from-shell :ensure t)
   (exec-path-from-shell-initialize)
   ```

1.13.2 Projectile

Navegação para projetos

```
(use-package projectile
:ensure t
:init (projectile-mode +1)
:config
(define-key projectile-mode-map (kbd "C-c p") 'projectile-command-map)
)
```

1.13.3 Helm

```
(use-package helm
:ensure t
:init
(helm-mode 1)
(progn (setq helm-buffers-fuzzy-matching t))
:bind
(("C-c h" . helm-command-prefix))
(("M-x" . helm-M-x))
(("C-x C-f" . helm-find-files))
(("C-x b" . helm-buffers-list))
(("C-c b" . helm-bookmarks))
(("C-c f" . helm-recentf))   ;; Add new key to recentf
(("C-c g" . helm-grep-do-git-grep)))  ;; Search using grep in a git project
```

Describes para o Helm

```
(use-package helm-descbinds
:ensure t
:bind ("C-h b" . helm-descbinds))
```

Helm Swoop auxiliar na busca

```
(use-package use-package-chords
:ensure t
:init
:config (key-chord-mode 1)
(setq key-chord-two-keys-delay 0.4)
(setq key-chord-one-key-delay 0.5) ; default 0.2
)
```

```
(use-package helm-swoop
:ensure t
:chords
("js" . helm-swoop)
("jp" . helm-swoop-back-to-last-point)
:init
(bind-key "M-m" 'helm-swoop-from-isearch isearch-mode-map)

;; If you prefer fuzzy matching
(setq helm-swoop-use-fuzzy-match t)

;; Save buffer when helm-multi-swoop-edit complete
(setq helm-multi-swoop-edit-save t)

;; If this value is t, split window inside the current window
(setq helm-swoop-split-with-multiple-windows nil)

;; Split direction. 'split-window-vertically or 'split-window-horizontally
(setq helm-swoop-split-direction 'split-window-vertically)

;; If nil, you can slightly boost invoke speed in exchange for text color
(setq helm-swoop-speed-or-color nil)

;; ;; Go to the opposite side of line from the end or beginning of line
(setq helm-swoop-move-to-line-cycle t))
```

1.13.4 Run Code

Rodar codigo de forma rápida apertando Ctrl+c r

```
(use-package quickrun
:ensure t
:bind ("C-c r" . quickrun))
```

1.13.5 Org Babel for JAVA

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((java . t)))
```