

DOOM EMACS HANDBOOK



# For beginners



```
5   has_many :comments
6 end

require 'rails_helper'
RSpec.describe Message, type: :model do
  subject { Message.new(params) }

  context 'when params are valid' do
    let(:params) { { title: 'My Emacs Message', body: 'fa' } }

    it { expect(subject).to be_valid }
  end

  context 'when params are invalid' do
    let(:params) { { title: '', body: '' } }

    it { expect(subject).to be_invalid }
  end
end
```

● 129 doom-emacs-demo/app/models/message.rb 7:0 All Ruby 2.7.3 ● 387 doom-emacs-demo/spec/models/message\_spec.rb 2:8 All

-- mode: rspec-compilation; default-directory: "~/doom-emacs-demo/" --  
RSpec Compilation started at Wed Sep 22 13:44:27

bundle exec rspec --options /home/otavio/doom-emacs-demo/.rspec /home/otavio/doom-emacs-demo/spec/models/message\_spec.rb

..

Finished in 0.01338 seconds (files took 0.47858 seconds to load)  
2 examples, 0 failures

RSpec Compilation finished at Wed Sep 22 13:44:27

**BEST COMMANDS FOR DAILY USAGE - ROR DEVELOPMENT**

Config: <https://github.com/otavioschwanck/doom-emacs-on-rails>

# Learn vim before start

**First, You have to learn the basics of  
Vim:**

```
$ sudo apt install vim
```

```
$ vimtutor
```

---

UNDERSTANDING COMMANDS:

C = CONTROL. EXAMPLE: C-O = CTRL+O

M = ALT. EXAMPLE: M-O = ALT + O

S = SHIFT.

FOR SOME EXERCISES, PRESS M-X AND  
SEARCH FOR MR-MIYAGI

# FILE NAVIGATION

## Generic Buffer (file) change

<b>SPC ,</b>	Change Buffer
<b>C-,</b>	Previous Buffer
<b>C-;</b>	Next Buffer
<b>SPC SPC</b>	Find Files in project
<b>SPC .</b>	Find File (cur dir)

## Rails Navigation

<b>SPC r m</b>	Find Model
<b>SPC r a</b>	Find Locales
<b>SPC r z</b>	Find Serializer
<b>SPC r v</b>	Find View
<b>SPC r V</b>	Find View (current resource)
<b>SPC r c</b>	Find Controller
<b>SPC r s</b>	Find Service
<b>SPC a</b>	Go to test*
<b>SPC A</b>	Go to test and vsplit*

\*upper case is always for  
current resource

## Go-to

<b>gd</b>	Go to definition
<b>gD</b>	Go to usages (reference)
<b>gf</b>	Go to file
<b>ga</b>	Go to route Ex: (foo_path)



# SEARCH / JUMP ON FILE

## On Current Line

<b>f</b>	go to character (1)
<b>S</b>	go to character (2)
<b>0</b>	go to first character of line
<b>-</b>	go to end of line
<b>w</b> and <b>W</b>	jump 1 word (start) (W jumps higher)
<b>e</b> and <b>E</b>	jump 1 word (end) (E jumps higher)
<b>b</b> and <b>B</b>	jump -1 word (start) (B jumps higher)
<b>t</b>	go to before character(1)

\*you can keep pressing f or t or s to repeat the search

\*f, s and t can be used backwards using uppercase

## On Current File

<b>/</b>	Search Forward
<b>?</b>	Search Backward
<b>n</b> and <b>N</b>	Go forward and backward (search)
<b>SPC s s</b>	Advanced search on current file
<b>SPC s i</b>	Search symbols / defs on current file
<b>#</b>	Search word at cursor at current file
<b>M-s</b>	Search and jump using two characters
<b>gg</b>	First line of file
<b>G</b>	Last line of file
<b>20G</b>	Go to line 20
<b>]g</b> and <b>[g</b>	Next / Previous git change
<b>]]</b> and <b>[[</b>	Next / Previous method definition
<b>])</b> and <b>[)</b>	Next / Previous parenthesis
<b>]} and [}</b>	Next / Previous brackets
<b>C-j</b> and <b>C-k</b>	Next / Previous section

# ENTERING ON INSERT MODE

## On Current Line

<b>i</b>	Enter on insert mode
<b>a</b>	Enter on insert mode after point
<b>I</b>	Enter on insert mode at beg of line
<b>A</b>	Enter on insert mode at end of line
<b>V then A</b>	Add text at end of line at all selected
<b>V then I</b>	Add text at beg of line at all selected



# SEARCH GLOBALLY

## On Project

<b>SPC s p</b>	Search text on Project
<b>SPC s d</b>	Search text on current dir
<b>SPC *</b>	Search item at cursor on project
<b>M-x find-dired</b>	Grep search files on project

## Other

<b>SPC f P</b>	Private doom emacs files
<b>C-o</b>	Return to last jump
<b>C-i</b>	Go forward on jump
<b>SPC f r</b>	Find Recent Files

# FILE EDITING

## Current File

<b>SPC f D</b>	Delete current file
<b>SPC f Y</b>	Copy current file path
<b>SPC f R</b>	Rename / Move current File
<b>\ or ç or SPC f s</b>	Save file



# COPY DELETE AND PASTE

## Intro

To use one of the operators, call the **operator** + the **motion**. **Example: Delete a word = dw**

## The operators

<b>d</b>	Delete (Cut)
<b>c</b>	Delete (Cut) and enter on insert mode
<b>y</b>	Copy
<b>p and P</b>	Paste before   Past e after cursor
<b>C-p</b>	*after pasting* Navigate on copy history
<b>M-y</b>	Search on copy (yank) history (and paste)

## Selecting

To start selecting something, press **v**. To line select, press **S-V**. To block select, press **C-v**.

**After you done selecting what you want, you can:**

<b>d or y or c</b>	Execute the operator action
<b>S</b>	Add delimitator to selection. Example: S) = (selection)      S( = ( selection ) S] = [selection]      S[ = [ selection ] S" = "selection" St<emacs> = <emacs>selection</emacs>

**FOR HOW TO USE SNIPPETS,  
CHECK THE GIF ON README**

# COOL STUFF WITH OPERATORS

Before read:

lets call the operators (**d, y or c**): **OP**

## Hidden goodies

<b>OP OP</b>	Do the operator on current line. Ex: dd
<b>4 OP OP</b>	Do the operator on 4 lines. Ex: 4dd
<b>ds</b>	Delete the delimiter. Ex: ds" Delete ""
<b>OP io</b>	Operator on thing at point
<b>OP ij</b>	Operator on current indentation
<b>OP i)</b>	Operator inside ()
<b>OP i"</b>	Operator inside ""
<b>OP i]</b>	Operator inside []
<b>OP it</b>	Operator inside tag
<b>OP ii</b>	Operator inside indentation
<b>UPPERCASE OP</b>	Operator from point to end of line
<b>DT OP</b>	Operator from point to before char
<b>DF OP</b>	Operator from point to char
<b>OP ia</b>	Operator in argument
<b>M-c</b>	Toggle case (snake_case, camel, etc)

\*every command with **i** can be used with **a** (but the action includes the delimiters)

Text: my\_var = (i love doom emacs)  
di) => my\_var = ()  
dai => my\_var =



# RUNNING STUFF

## Basic Stuff

<b>SPC v</b>	Toggle Quick Terminal
<b>SPC o T</b>	Open a new terminal
<b>SPC e</b>	Open tree (Prefer <b>SPC .</b> over tree)
<b>SPC m k k</b>	Run a rake task
<b>SPC m b i</b>	Run bundle install
<b>SPC =</b>	Run rubocop on current file
<b>SPC -</b>	Indent current file
<b>SPC m P</b>	Run rubocop on project
<b>SPC r r</b>	Rails Console
<b>SPC r R</b>	Rails Server
<b>C-c t</b>	Google Translator

## Tests

<b>SPC t v</b>	Run tests of current file
<b>SPC t a</b>	Run all tests
<b>SPC t r</b>	Rerun last test
<b>SPC t l</b>	Run only failures of last test
<b>SPC t s</b>	Run tests on cursor

# AWESOME TIPS

## Multiple Cursors

<b>M-d</b>	Start multiple cursors (keep pressing)
<b>M-S-d</b>	Find item of multiple cursor on top
<b>C</b>	Edit in all cursors

## Search and Replace in entire project

<b>SPC s p</b>	Search On Entire Project
<b>C-c C-e</b>	Edit search result

After this, you can edit all search results like a single file. To confirm, press **C-c C-C**. To cancel, **C-c C-k**

**\*You can use multiple cursors or**  
**:%s/old\_text/new\_text/gr to change**

## Rename multiple files on project

<b>M-x find dired</b>	Search with grep the files. To match all <b>foo rb</b> files by example, run with <b>with -name "foo*.rb"</b> .
<b>C-c C-e</b>	Edit search results

After this, you can edit all search results like a single file. To confirm, press **C-c C-C**. To cancel, **C-c C-k**

# WINDOW MANAGEMENT

## Window Actions

<b>C-w v</b>	Divide verically
<b>C-w s</b>	Divide horizontally
<b>C-w C-o</b>	Maximize Window
<b>C-w C-u</b>	Undo Windows Alteration
<b>C-w C-r</b>	Redo Windows Alteration
<b>C-w =</b>	Balance Windows
<b>C-w T</b>	Detach Window
<b>C-w q</b>	Close Window

## Windows Moviment

<b>M-o</b>	Go to next window
<b>M-h</b>	Go to window on left
<b>M-j</b>	Go to window on bottom
<b>M-k</b>	Go to window on top
<b>M-l</b>	Go to window on right



# RUBY REFACTORING

## Refactoring like a pro

<b>SPC m i</b>	Toggle if unless (single / multi line)
<b>SPC m m</b>	Move selected text to a new method
<b>SPC m n</b>	Create a method from text in cursor
<b>SPC m [</b>	Toggle do end => { } and vice-versa
<b>SMC m v</b>	Move selected text to a new variable
<b>SPC m V</b>	Move selected text to a constant
<b>C-c s</b>	Add http code at point (humanized)

## Using Snippets + Auto Complete

<b>TAB</b>	Next item on autocomplete
<b>S-TAB</b>	Previous item on autocomplete
<b>RET</b>	Select item on autocomplete
<b>M-RET</b>	Newline while autocomplete open
<b>C-o</b>	Start snippet / go forward on snippet Example: def<TAB> = def my_method(args) end
<b>C-S-o</b>	Go back on snippet selection
<b>C-d</b>	Delete selected text (while on snippet)
<b>C-RET</b>	Complete text from all buffers

## Using Emmet (create methods and tags)

### **M-e** Toggle emmet

```
On ruby: init@name;call@values<C-e> =  
def initialize(name)  
  @name = name  
end
```

```
def call(values)  
  | cursor here  
end
```

```
On web-mode: .name>li.item*3<C-e>=  
<div class="name">  
  <li class="item"></li>  
  <li class="item"></li>  
  <li class="item"></li>  
</name>
```

# RUBY PLUGINS

## RAILS ROUTES

**C-c o** Add route at point (using cache)  
**C-c C-o** Add route at point (refreshing cache)  
**ga** Go to route

## RAILS i18n

**C-c i** Add i18n at point (using cache)  
**C-c C-i** Add i18n at point (refreshing cache)\_

## Ruby JSON to Hash

**SPC m J** Convert JSON at point into hash  
**SPC m j** Send key of the converted hash to a new let



Ruby

# GIT STUFF

## On a file

<b>SPC g r</b>	Revert modification at point (can be used to see the diff)
<b>SPC g t</b>	Time Machine mode (Use C-n and C-p to navigate)
<b>]g</b>	Go to next git hunk
<b>[g</b>	Go to previous git hunk
<b>SPC g R</b>	Revert all modifications on file

## On Magit Status

<b>SPC gg</b>	Open Magit Status
<b>f</b>	Fetch
<b>F</b>	Pull
<b>P</b>	Push
<b>cc</b>	Commit
<b>ca</b>	Ammend
<b>Z</b>	Stash
<b>?</b>	See all available options
<b>m</b>	Merge
<b>r</b>	Rebase



**It's**  
**Magit!**

# FILE MANAGER (DIRED)

Dired is an awesome file manager integrated on Emacs. To open in some folder, just press **SPC** . And select a folder instead a file. You can also show the current file on dired using **SPC o -**

## Operations in DIREDD

<b>d</b>	Mark a file to deletion
<b>x</b>	Delete the marked files
<b>m</b>	Mark a file to do some action
<b>u</b>	Unmark a file
<b>U</b>	Unmark all files
<b>C</b>	Copy the file (or files if has marks)
<b>R</b>	Move/Rename file (or files if has marks)
<b>-</b>	Go to parent directory
<b>=</b>	Diff this file with another file
<b>g?</b>	See all commands above
<b>SPC o -</b>	Visit current file on dired.

## Super cool tip (please read)

While Renaming / Copying you can press **M-n** to put the current file name on the text box (works in every text box).

While on text box:

<b>M-b</b>	Go previous word
<b>M-f</b>	Go next word
<b>M-d</b>	Delete next word
<b>C-w</b>	Delete previous word

# WORKSPACES

Everytime you open a project with SPC p p you create a workspace. To change between workspaces press M-1 to M-9. The prefix for workspace management is SPC TAB.

## Workspace Commands

<b>SPC TAB n</b>	New Workspace
<b>SPC TAB N</b>	New Named Workspace
<b>SPC TAB r</b>	Rename Workspace
<b>SPC TAB d</b>	Delete Workspace
<b>SPC TAB .</b>	Change Workspace
<b>SPC TAB s</b>	Save Workspace
<b>SPC TAB l</b>	Load Workspace
<b>M-1 to M-9</b>	Change Workspace



# CREATING YOUR OWN SNIPPETS

To create your own snippets, just type `M-x yas-new-snippet` on the file type you want.

`name` = Name that will show on autocomplete

`key` = key pressed to toggle snippet

After the comments, just put the commands that you want to create snippet.

**Syntax:**

```
def ${1:my_method_name}(${2:my_method_args})  
  ${0:}  
end
```

**What will happen:**

After pressing the snippet, you 1: Can change the method name, 2: change the args, and after that, it will put the cursor inside the method.

To persist your snippet, press `C-c C-c`. You can edit your snippet by searching for it with

`SPC f P`

**IMPORTANT:** The file name need to be the same as the key you put.

# HARPOON

Harpoon is a plugin to create bookmarks per project / branch. It is awesome to navigate between only the files that you are really working.

## Commands

<b>SPC 1 to SPC 9</b>	Navigate between files
<b>C-s</b>	Save file to harpoon
<b>C-SPC</b>	Select one file from harpoon
<b>SPC j f</b>	Edit harpoon file
<b>SPC j c</b>	Clean harpoon file



# EPIC TERMINAL MANAGEMENT

This config has a lot of helpers to work with terminal. You can configure single time commands and multiple commands term layouts on **SPC f m** (`~/.doom.d/user-settings.el`)

## Single Time Commands

To configure your single times commands, go to your configs (**SPC f m**) and search for **+add-command-to-term-list**. Example:

**(+add-command-to-term-list** `"Docker Compose"` . `"docker-compose up; read; exit"`)

Command name

Command to run

<b>SPC o t</b>	Execute an single time command
<b>SPC l</b>	List all terminals (and go to)
<b>SPC l</b>	(VISUAL) Send text to terminal
<b>SPC v</b>	Open a quick terminal (empty)

## Terminal templates

To configure terminal templates, go to your configs (**SPC f m**) and search for **+add-layout-to-term-list**. Example:  
**(+add-layout-to-term-list** `"Rails"` . `("rails console" "rails server" nil)`)

Layout name

Term 1

Term 2

Empty term

<b>SPC T</b>	Create terminals from layout
<b>M-1 to M-9</b>	Switch between workspaces

## Terminal quick tips

<b>C-c</b>	(On normal mode) send C-c C-c and and go to insert-mode
<b>M-n and M-p</b>	Navigate between command history