Desktop-Save Mode

bmag.github.io/2015/12/26/desktop.html

Dec 26, 2015

Intro

Desktop-Save mode ("Desktop" from now on) is a global mode that automatically saves your Emacs session, so you can load it later and pick up exactly where you left Emacs last time. In practice, Desktop saves your session information and buffers to a file. Next time you start Emacs, you can load your session and have (almost) all your buffers available. The main benefit of Desktop is that it saves the hussle of reopening all the files you need and setting your windows the way you like them.

Usage

Enabling Desktop is very simple:

```
(desktop-save-mode)
;; optional: automatically load previous session on startup
(desktop-read)
```

Saving is done automatically every once in a while and when exiting Emacs, or manually with the `desktop-save` command. Loading can be done manually, or automatically at startup, by using commands `desktop-read` or `desktop-change-dir`.

Saving Your First Session

When exiting Emacs after the first time that you enable `desktop-save-mode`, Desktop won't find an existing desktop file and will ask you wether it should create one and where should it create it. If you don't want to wait until exiting Emacs, you can call `desktop-save` manually. You only need to do this once.

Loading Saved Sessions

To load your desktop file, use the command `desktop-read`. It will look for a desktop file in `desktop-path` and load the first one that it finds. By default, Desktop searches in `~` and `~/.emacs.d` directories. If your desktop file is saved in another directory, you should set `desktop-path` accordingly.

```
(desktop-save-mode)
;; optional - if your desktop is saved in "~.emacs.d/.cache/"
(setq desktop-path '("~/.emacs.d/.cache/"))
(desktop-read)
```

Working with Multiple Sessions

With Desktop, you can also have several different sessions saved on disk, and switch between them. The key here is to use the command `desktop-change-dir`. The command will close your current session (without exiting Emacs) and load another session from file.

Default Supported Buffer Types

By default, Desktop only supports these types of buffers:

- file buffers (via `desktop-restore-file-buffer`
- dired buffers (via `dired-restore-desktop-buffer`)
- vc-dir buffers (via `vc-dir-restore-desktop-buffer`)
- todo-mode buffers (via `todo-restore-desktop-buffer`)
- DocView buffers - PDFs, ODTs, DOCXs, etc. (via `doc-view-restore-desktop-buffer`)
- MH folders (via `mh-restore-desktop-buffer`)

Supporting More Buffer Types

Desktop can handle more than the default buffer types, by using the variables `desktop-save-bufer` and `desktop-buffer-mode-handlers`. To support a buffer type, you only need to write two functions and hook them into the correct places. One function saves necessary buffer state, and is hooked via `desktop-save-buffer`. The other function restores the buffer according to its saved state, and is hooked via `desktop-buffer-mode-handlers`.

desktop-save-buffer

Setting this buffer-local variable tells Desktop that this file should be saved. Normally you would set it to a function. That function should receive one argument, `DESKTOP-DIRNAME`, and return the “misc” data that is necessary for restoring the buffer. Example:

```
(defun sy-save-shell-buffer (desktop-dirname)
  ;; we only need to save the current working directory
  default-directory)
;; save all shell-mode buffers
(add-hook 'shell-mode-hook
          (lambda ()
            (setq-local desktop-save-buffer #'sy-save-shell-buffer)))
```

desktop-buffer-mode-handlers

The counterpart for `desktop-save-bufer`. This is an alist mapping major modes and restoration functions. When Desktop restores a buffer, it looks for the correct handler (restoration function) in `desktop-buffer-mode-handlers`. If no handler is found, the default is to use `desktop-restore-file-buffer`. A possible value of `desktop-restore-file-buffer`:

```
((doc-view-mode . doc-view-restore-desktop-buffer)
 (dired-mode . dired-restore-desktop-buffer)
 (vc-dir-mode . vc-dir-restore-desktop-buffer)
 (Info-mode . Info-restore-desktop-buffer))
```

Each restoration function takes three arguments: `BUFFER-FILE-NAME`, `BUFFER-NAME` and `MISC`, and should create an appropriate buffer accodring to their values. `BUFFER-FILE-NAME` is the file name of the original buffer, and is of course only useful for buffers that visited a file or a directory. `BUFFER-NAME` is the original name of buffer, which you should use for the new buffer, of course. `MISC` is the data that was returned by the corresponding save function, as described in the previous section. Example:

```
(defun sy-create-shell-buffer (_file-name buffer-name misc)
  "MISC is the value returned by `sy-save-shell-buffer'.
_FILE-NAME is nil."
  (let ((default-directory misc))
    ;; create a shell buffer named BUFFER-NAME in directory MISC
    (shell buffer-name)))
;; restore all shell-mode buffers
(add-to-list 'desktop-buffer-mode-handlers '(shell-mode . sy-create-shell-buffer))
```

Full Example of Desktop Configuration

```
(require 'desktop)

(defun sy-save-shell-buffer (desktop-dirname)
  ;; we only need to save the current working directory
  default-directory)

(defun sy-create-shell-buffer (_file-name buffer-name misc)
  "MISC is the value returned by `sy-save-shell-buffer'.
_FILE-NAME is nil."
  (let ((default-directory misc))
    ;; create a shell buffer named BUFFER-NAME in directory MISC
    (shell buffer-name)))

;; save all shell-mode buffers
(add-hook 'shell-mode-hook (lambda () (setq-local desktop-save-buffer #'sy-save-
shell-buffer)))
;; restore all shell-mode buffers
(add-to-list 'desktop-buffer-mode-handlers '(shell-mode . sy-create-shell-buffer))

(setq desktop-path '("~/.emacs.d/.cache/"))
(desktop-save-mode)
(desktop-read)
```

Outro

Desktop-Save mode is an easy way to save Emacs sessions, and quickly get back to what you where working on last time. I covered how to enable Desktop-Save mode, load and save desktops, what buffer types are saved and how to save more buffer types.

Due to time constraints, I didn’t cover some topics: Desktop and lazy-loading, configuring the restoration of frame and window configurations, how Desktop restores minor modes, and how Desktop can be synergyzed with other features. I hope to cover these topics in the future, and I hope that you found this article informative.