











Sessions

Description	Keystroke	Function	Note
<div> <div> Emacs Session Management </div> <div> <ul style="list-style-type: none"> Emacs built-in desktop.el External packages: <ul style="list-style-type: none"> desktop-registry desktop+ desktop-recover </div> </div>	<ul style="list-style-type: none"> Emacs built-in desktop.el provides basic session management, saving information about frames, windows and buffers their major modes, buffer positions, etc are saved inside a file named “.emacs.desktop” located in the current directory. If the desktop feature is enabled at init time, when you restart Emacs from that directory the session is reloaded automatically. The file also provides a set of user option variables that control various aspects of the feature. Although useful, the built-in session management does not allow saving and restoring sessions by names. Also if you want to start Emacs from a directory that contains an already saved session you must either start Emacs with the ‘<code>–no-desktop</code>’ command line option to prevent it from automatically loading the session. That can be annoying if you use several Emacs instances or want to work in a directory without restoring that given session. The desktop-registry external package simplifies the manual management of desktops saved per directories by allowing changing desktops, listing them and clearing the current session. It does, however still relies on session to be stored inside directories and it ties the session to the directories. It is still not possible to have different sessions that encompasses several directories in conflicting ways or use multiple different sessions for a given directory. The desktop+ external package solves this problem. It provides central management of all recorded sessions by names, storing the information inside the ~/.emacs.d directory. The desktop-recover external package is older and not available from any Elpa-compliant archive. It is mainly used to recover sessions after an Emacs cash and therefore mainly useful when developing Emacs core code. PEL does not support it. PEL provides simple control to activate the built-in desktop session management and incorporate either external desktop management package. It provides command bindings for saving and restoring sessions and deals with installing and activating the packages, all through a set of user options. <div>  PEL activates session management with the pel-use-desktop user option variable that can be set to the following variables: <ul style="list-style-type: none"> nil: not used t: use built-in desktop session management (but do not activate automatic save & restore of sessions). with-desktop-automatic: use built-in desktop session management and activate automatic save & restore of sessions per directories). with-desktop-registry : use desktop.el with the desktop-registry external package. Do not activate auto save & load of sessions. with-desktop-registry-automatic: use desktop.el with the desktop-registry external package and activate auto save & load of sessions. with-desktop+ : use desktop.el with the desktop+ external package, to allow centralized named sessions. No session auto-load. </div> <div>  For new users, desktop+ is the easiest to use. It also saves more information: it can save more types of buffers than the others. </div> <div>  Emacs desktop user options: Group: desktop <ul style="list-style-type: none"> desktop-save-mode : activator: non-nil to enable desktop-save-mode (which saves the session when Emacs exists) desktop-save: whether desktop saves session and if so whether and when it asks user. ... </div>		
<div> <div>Open this PDF file.</div> <div>See also: 🔗 Help/Info</div> </div>	<code><f11> S <f1></code>	(pel-help-pdf & optional OPEN-WEB-PAGE)	Open the 🔗 Sessions local PDF. If the prefix argument (like C-u or M--) is used, then it opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
🔗 Customize PEL session management control	<code><f11> S <f2></code>	(pel-customize-pel & optional OTHER-WINDOW)	Customize PEL Session support. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u) , display in other window.
🔗 Customize Emacs session management control	<code><f11> S <f3></code>	(pel-customize-library & optional OTHER-WINDOW)	Customize Emacs Session support: desktop.
Show name of desktop	<code><f11> S ?</code>	(pel-desktop-show)	Display name of currently used desktop if any.
Emacs built-in desktop.el	Emacs built-in desktop provides session management of buffers and windows (including the window layout: PEL ensures that window layout is controlled in terminal mode: normally Emacs only enables it in graphics mode. <div>  PEL activates: <ul style="list-style-type: none"> the standard desktop commands when the pel-use-desktop user option is set to: t, with-desktop-automatic, with-desktop-registry or with-desktop-registry-automatic. automatic saving and restoring of sessions on Emacs startup when pel-use-desktop is set to with-desktop-automatic, or with-desktop-registry-automatic. </div>		
Toggle Desktop Save Mode	<code><f11> S M-s</code>	(desktop-save-mode & optional ARG)	Toggle desktop saving (Desktop Save mode), where desktop is automatically saved when Emacs terminates and also saved periodically. <ul style="list-style-type: none"> With a prefix argument ARG, enable Desktop Save mode if ARG is positive, and disable it otherwise. <p>When Desktop Save mode is enabled, the state of Emacs is saved from one session to another. In particular, Emacs will save the desktop when it exits (this may prompt you; see the option ‘desktop-save’). The next time Emacs starts, if this mode is active it will restore the desktop.</p> <ul style="list-style-type: none"> To manually save the desktop at any time, use the command <code><f11> S S</code>. To load it, use <code><f11> S L</code>. <p>Once a desktop file exists, Emacs will auto-save it according to the option ‘desktop-auto-save-timeout’.</p> <ul style="list-style-type: none"> To see all the options you can set, browse the ‘desktop’ customization group, with: M-x customize-group desktop <div>  Use this to turn on/off the current automatic saving of the session. It does not control whether Emacs will automatically load a session on startup. </div>
Load/read a session defined in specified directory	<code><f11> S L</code>	(desktop-read & optional DIRNAME)	Read and process the desktop file in directory DIRNAME. <ul style="list-style-type: none"> Look for a desktop file in DIRNAME, or if DIRNAME is omitted, look in directories listed in ‘desktop-path’. If a desktop file is found, it is processed and ‘desktop-after-read-hook’ is run. If no desktop file is found, clear the desktop and run ‘desktop-no-desktop-file-hook’.
Save current session in a directory	<code><f11> S S</code>	(desktop-save DIRNAME & optional RELEASE ONLY-IF-CHANGED VERSION)	Save the desktop in a desktop file. <ul style="list-style-type: none"> Prompts for DIRNAME: specifies where to save the desktop file.
Clear current session - close all its buffers	<code><f11> S c</code>	(desktop-clear)	Empty the Desktop. <ul style="list-style-type: none"> This kills all buffers except for internal ones and those with names matched by a regular expression in the list ‘desktop-clear-preserve-buffers’. Furthermore, it clears the variables listed in ‘desktop-globals-to-clear’. When called interactively and ‘desktop-restore-frames’ is non-nil, it also deletes all frames except the selected one (and its minibuffer frame, if different). Prompt for saving modified buffers visiting files.
Changed to a session saved in a different directory	<code><f11> S d</code>	(desktop-change-dir DIRNAME)	Change to desktop saved in DIRNAME. <ul style="list-style-type: none"> Kill the desktop as specified by variables ‘desktop-save-mode’ and ‘desktop-save’, then clear the desktop and load the desktop file in directory DIRNAME.
Revert to last loaded session	<code><f11> S r</code>	(desktop-revert)	Revert to the last loaded desktop.

Description	Keystroke	Function	Note
Using desktop-registry	<p>When using the desktop-registry, the desktop-registry commands listed below are available. The Emacs built-in desktop.el commands described above are also available.</p> <p> The desktop-registry external package provides the following commands.</p> <p> PEL activates the commands when the pel-use-desktop user option is set to with-dektop-registry or with-desktop-registry-automatic. It activates automatic saving and loading of session on Emacs startup when it is set to with-desktop-registry-automatic.</p>		
List registry of session/desktops	<f11> S R l	(desktop-registry-list-desktops)	<p>Display a list of registered desktops inside a *Desktop Registry* buffer.</p> <p>The following command keys are available in this buffer:</p> <ul style="list-style-type: none"> A desktop-registry-add-current-desktop R desktop-registry-rename-desktop a desktop-registry-add-directory d desktop-registry-remove-desktop o desktop-registry-change-desktop
Add a director-identified session to the sessions/desktops registry	<f11> S R a	(desktop-registry-add-directory DIR &optional NAME)	<p>Add DIR to the desktop registry, possibly using NAME.</p> <ul style="list-style-type: none"> Prompts for DIRectory. With universal argument (C-u), also prompt for a name. <ul style="list-style-type: none"> This is useful when the directory name is not the project name or when it would result in duplicate entries in 'desktop-registry-registry'.
Add current session to the session/desktop registry	<f11> S R A	(desktop-registry-add-current-desktop &optional NAME)	<p>Add the currently opened desktop file to 'desktop-registry-registry'.</p> <ul style="list-style-type: none"> If NAME is specified use that as the name for the registry entry. With universal argument (C-u), also prompt for a name. <ul style="list-style-type: none"> This is useful when the directory name is not the project name or when it would result in duplicate entries in 'desktop-registry-registry'.
Remove the session/desktop from the registry	<f11> S R d	(desktop-registry-remove-desktop DESKTOP)	<p>Remove desktop.</p> <ul style="list-style-type: none"> If cursor is in the *Desktop Registry* buffer, on a desktop name in the desktop list, the current desktop is used, otherwise the command prompts for the desktop name (with completion support).
Change current session/desktop to the name specified in registry	<f11> S R o	(desktop-registry-change-desktop NAME)	<p>Change to the desktop named NAME.</p> <ul style="list-style-type: none"> If cursor is in the *Desktop Registry* buffer, on a desktop name in the desktop list this desktop name is used, otherwise the command prompts for the desktop name (with completion support). This function just calls 'desktop-change-dir' with the directory attached to NAME.
Rename session/desktop in registry	<f11> S R R	(desktop-registry-rename-desktop OLD NEW)	<p>Rename desktop OLD to NEW.</p> <ul style="list-style-type: none"> If cursor is in the *Desktop Registry* buffer, on a desktop name in the desktop list, the listed desktop is used as OLD, otherwise the command prompts for the OLD desktop name (with completion support). It always prompts for the NEW desktop name.
Using desktop+	<p> When using the desktop+ only four commands are available to manage sessions (desktops) by name. Overall it's a simpler interface. The 4 commands are organized in 2 groups:</p> <ol style="list-style-type: none"> Save/load manually named session. On loading you can use naming completion. If name is left bank, uses the current directory as name. Save/load using automatically generated session name: uses the current directory as the session name. <p> PEL activates the commands when the pel-use-desktop user option is set to with-desktop+</p> <p> The information about each named session is stored inside the <code>~/.emacs.d/desktops</code> directory tree.</p> <ul style="list-style-type: none"> The session names correspond to the directories under <code>~/.emacs.d/desktops</code>. You can rename a session by renaming the corresponding directories, delete a session by deleting the corresponding directory. When a session is named automatically the directory has a name with the complete original path using <code>'.'</code> as directory separators. 		
Save session with a specific name	<f11> S s	(desktop+-create NAME)	<p>Create a new session, identified by a name.</p> <ul style="list-style-type: none"> The session is created in a subdirectory of 'desktop+-base-dir'. It can afterwards be reloaded using 'desktop+-load'. As a special case, if NAME is left blank, the session is automatically named after the current working directory.
Load session identified by name	<f11> S l	(desktop+-load NAME)	<p>Load a session previously created using 'desktop+-create'.</p> <ul style="list-style-type: none"> NAME is the name which was given at session creation. When called interactively, it is asked in the minibuffer with auto-completion. As a special case, if NAME is left blank, the session is automatically named after the current working directory.
Save current session named after current directory	<f11> S S	(desktop+-create-auto)	<p>Create a new session, identified by the current working directory.</p> <ul style="list-style-type: none"> The session is created in a subdirectory of 'desktop+-base-dir'. It can afterwards be reloaded using 'desktop+-load'.
Load session previously created for the current directory	<f11> S L	(desktop+-load-auto)	<p>Load a session previously created using 'desktop+-create-auto'.</p> <ul style="list-style-type: none"> The session is identified by the current working directory.

Sessions— References

Topic & link	Description
GNU Emacs Manual - Saving Emacs Sessions	Describes Emacs built-in desktop session management feature.
desktop+	The external package desktop+, hosted on Github and on the MELPA package archive, that extends Emacs session management desktop.el by providing the ability to control sessions by name instead of having to rely on the directory location where save sessions were stored. It also allows saving/restoring several types of buffers, which are not supported by the other packages.
desktop-registry	Another package that extends the built-in desktop.el with a registry of desktops (sessions).