# ERT — Emacs Lisp Regression Testing

| Description | Keystroke | Function | Note |
|---|---|---|---|
| **Using ERT** | If you write Emacs Lisp code, writing unit test code normally helps increase code quality and maintainability. The ERT library provides a simple, yet powerful, environment to write test you can run from the command line and interactively, inside Emacs with the extra ability to debug failing code.<br><br>ERT is part of Emacs standard distribution since Emacs 24.<br><br>ERT provides a command to run tests that have been written using the ERT macros. Tests are normally written inside separate .el files, with names using the same prefix as the file being tested. To run the test load the test file and then type **`M-x ert RET t RET`** . That executes the ert command described below and provides the selector **t** which means running every test. You can also use other regular expression that identify the names of the test functions to run (for example use "^foo-" to run all tests that have a name that begins with foo-. | | |
| **Run test interactively** | | **(ert** SELECTOR &optional OUTPUT-BUFFER-NAME MESSAGE-FN) | Run the tests specified by SELECTOR and display the results in a buffer.<br><br>• SELECTOR works as described in 'ert-select-tests'. (Use **t** to run all tests, or name the test to execute.<br>• OUTPUT-BUFFER-NAME and MESSAGE-FN should normally be nil; they are used for automated self-tests and specify which buffer to use and how to display message.<br>• By default, the results are stored inside the *ert* buffer, opened in ERT-Results mode. |
| ***ert* buffer commands** | The following single key commands are available in the *ert* buffer window and used to look into test failures. | | |
| **Re-run test** | **`r`** | (**ert-results-rerun-test-at-point**) | Re-run the same test |
| **Jump to test source** | **`.`** | (**ert-results-find-test-at-point-other-window**) | Jump to the source code of the test (in another window) |
| **List *should* forms** | **`l`** | (**ert-results-pop-to-should-forms-for-test-at-point**) | Shows the list of all *should* forms executed during the test before it failed |
| **View backtrace** | **`b`** | **(ert-results-pop-to-backtrace-for-test-at-point)** | View the backtrace for the failed test at point |
| **Re-run test with debugging** | **`d`** | (**ert-results-rerun-test-at-point-debugging-errors**) | Re-run the same test with debugging enabled |
| **Show messages** | **`m`** | (**ert-results-pop-to-messages-for-test-at-point**) | Show what messages were printed before the test failed |
| **Toggle condition printing** | **`L`** | (**ert-results-toggle-printer-limits-for-test-at-point**) | Toggle how much of the condition to print for the test at point. |
| **Delete obsolete tests** | **`D`** | (**ert-delete-test** TEST-NAME) | Delete obsolete tests (test whose code might have changed) |
| **Re-run all tests** | **`R`** | (**ert-results-rerun-all-tests**) | Re-run all tests, using the same selector |
| **Move to the next test result** | **`n`** | (**ert-results-next-test**) | Move to the next test results. |
| **Move to previous test result** | **`p`** | (**ert-results-previous-test**) | Move to the previous test results. |
| **Jump between summary and result** | **`j`** | (**ert-results-jump-between-summary-and-result**) | Jump between test result and summary. By positioning point on the test result character (., F or f) and then typing **j** point will move to the test summary (and will create one if the test passed). From the summary you can easily press RET to move point to the source code of the test (in another window). |
| **Show help for test** | **`h`** | (**ert-results-describe-test-at-point**) | Get help for the test corresponding to the test result character (or test summary) at point. |
| **Describe available commands** | **`?`** | (**describe-mode** &optional BUFFER) | Describe mode (and these commands) |
| **Quit - close window** | **`q`** | (**quit-window** &optional KILL WINDOW) | Quit window |

## Emacs Lisp Testing — References

| Topic & Link | Description |
|---|---|
| **ERT : Emacs Lisp Regression Testing** | ERT Manual, part of Emacs. |
| **Elisp Unit Testing with ERT** | Quick overview of ERT in an August 2012 blog written by Chris Wellons. Since then the cl.el library was replaced by the cl-lib.el and the flet function was deprecated to cl-flet, but aside from these small items the description is still valid. |
| **Emacs Lisp Mock @ EmacsWiki** | The original location of that mock library that can be used with ert. |
| **El mock @ GitHub** | The new location for el-mock.el |