Clojure Unit Tests Utilities for Running Unit Tests In Org-mode Documents

February 16, 2017

Contents

One of the beauty of using Org-mode to author computer softwares is that we can write about the rational behind what we are doing (the purpose of the application), we can write the code and unit test it, all in the same document at the same time.

However the `clojure.test` API hasn't been developed with this style of software development in mind and this is why using it for that purpose can be sometimes unnatural.

This is why I am introducing unit testing utilities for Clojure unit testing within Org-mode documents. They should make the development more natural and intuitive.

`tests` macro

It is often the case that you write a function and then want to test it right away. And then you write another one and test it too. You will endup with multiple code blocks where you want to test the inner test(s) and get the results for those only.

With `clojure.test` we are limited in our options: we have `run-tests` and `run-all-tests` which provide a summary report of the executed tests. Since Clojure 1.6 we have access to the `test-vars` function which we can use to test one or multiple test case **with fixtures**. However, the usage of that function is a bit complex (in its syntax) and no reporting is provided except if the tests fails.

Its usage looks like this:

```
(test-vars [#'test-load-ontology-remote
    #'test-load-unexisting-remote-ontology
    #'test-load-ontology-local])
```

To make the usage of the `test-vars` function, we created a new `tests` macro to simplify its usage and to make it more intuitive. Also, it should report failures and successes.

```
(defmacro tests
  "Run one or multiple tests with fixtures. Returns successes or failures. Tests shoul
  [& args]
  `(binding [clojure.test/*test-out* (java.io.StringWriter.)]
     (clojure.test/test-vars [~@(mapv (fn [tname]
`(var ~tname))
      args)])
     (if (empty? (str clojure.test/*test-out*))
       (println "All tests succeeded.")
       (println (str clojure.test/*test-out*)))))
```

The usage of this macro is much simpler and intuitive. The only thing you have to do is to write the name of the test functions as arguments of the macro. That way you won't have to use the `#'` *var* reader macro anymore, you simply list the tests. Let's unit test this macro.

```
(use 'clojure.test)

(deftest test-fails-1
  (testing
    (is (= 1 2))))

(deftest test-fails-2
  (testing
    (is (= 2 3))))

(deftest test-succeed-1
  (testing
    (is (= 2 2))))

(deftest test-succeed-2
  (testing
    (is (= 3 3))))
```

With the following tests, the first two should fails and the error be reported in `*out*`:

```
(tests test-fails-1 test-fails-2 test-succeed-1 test-succeed-2)
```

All the following tests should be successful and reported as-is:

```
(tests test-succeed-1 test-succeed-2)
```

Unit Tests

Let's create the tests suites that will cover all the different cases that should be handled by the macro. We should test with:

1. No tests defined
2. 1 test that fails
3. 1 test that succeed
4. 1 test that fails, 1 that succeed
5. 2 tests that fails
6. 2 tests that succeed
7. 2 tests that succeed, two that fails

The first thing we have to do is to create the tests that succeed and fails in another folder that won't be run by normal testing procedures.

There is one thing to keep in mind here, is that we actually can't test the tests that fails since these tests will be evaluated by different softwares like Cider or Leiningen, which means that the tests suites will actually fails. However the code for these failing functions will remain. If ran using `clojure.test` then everything will be working as expected.

```
(deftest test--succeed-1
  (testing
      (is (= 2 2))))

(deftest test--succeed-2
  (testing
      (is (= 3 3))))
```

```
(deftest test-no-test-specified
  (testing
      (is (= "All tests succeeded.\n" (with-out-str (tests))))))

(deftest test-some-test-specified
  (testing
      (is (= "All tests succeeded.\n" (with-out-str (tests test--succeed-1))))))

(deftest test-one-test-succeed
  (testing
      (is (= "All tests succeeded.\n" (with-out-str (tests test--succeed-1))))))

(deftest test-two-tests-succeed
  (testing
      (is (= "All tests succeeded.\n" (with-out-str (tests test--succeed-1
   test--succeed-2))))))

(use 'org-mode-clj-tests-utils.core-test-resources)

(tests test-no-test-specified
       test-some-test-specified
       test-one-test-succeed
       test-two-tests-succeed)
```