Clojure Basic Training - Module 7

Testing

Overview

- Unit tests, Acceptance tests
- TDD, BDD, RDD
- Outside-in, bottom-up
- Clojure (FP) enables any of the above
- Mocking is much easier with pure functions

Built-in basics

- no need to require a namespace
- **(test)** finds fn at key :test in var metadata
- **(assert)** Evaluates expr and throws an exception
- **:pre**(conditions) **:post**(conditions)

clojure.core/test

```
(defn add+
 {:test #(do
     (assert (= (add+ 2 3) 5))
     (assert (= (add+ 4 4) 8)))}
  [x y] (+ x y))
```

- Tests are “embedded” in the var metadata
- Useful for small quick assertions
- Triggers with (test #’add+)
- Tests are visible with (:test (meta #’add+))

:pre & :post conditions

```
(defn generate [ctx]
  {:pre  [(map? ctx)]
   :post [(string? %)]}
  (str ctx))
$> (generate [1 2 3])
Assert failed: (map? ctx) user/generate
$> (generate {:a "1"})
"{:a \"1\"}"
```

- Added to an additional map argument to defn
- A vector of predicates compiled down into “asserts”
- Useful for cheap type checking
- (set! *assert* false) will globally enable/disable

Built-in xUnit-like testing

- clojure.test namespace is included in Clojure
- more expressive xunit-like checkers
- supports for fixtures (setup/teardown equiv.)
- leiningen/automation enabled

clojure.test basics

```
1 (ns clj-training.core-test
2   (:require [clojure.test :refer :all]
3             [clj-training.core :refer :all]))
4
5 (deftest a-test
6   (testing "Should not fail"
7     (is (= 1 1))))
8
9 (deftest test-with-grouped-asserts
10   (testing "many way to sum up to 5"
11     (are [x y] (= 5 (+ x y))
12          2 3
13          1 4
14          3 2)))
```

Convention appending "-test"

Target ns

Multiple "is" grouped as "are"

fixtures - stubbing

```clojure
(defn check-int? [a]
  (or (instance? Long a)
      (string? a)))

(defn some-fixture
  [f]
  (try
    (with-redefs [clojure.core/integer? check-int?]
      (f))
    (finally
      "release db-conn.")))

(use-fixtures :once some-fixture)

(deftest mocking-and-fixtures
  (testing "I really want strings as ints."
    (is (integer? 4))
    (is (integer? "4"))))
```

with-redefs enable local re-binding of functions

Things like connecions can be released here

This fixture is applied “once” for all tests

References

- “Clojure Programming”, O’Reilly
- https://clojure.github.io/clojure/clojure.test-api.html
- For even more advanced features, look into Midje https://github.com/marick/Midje