clojure.tools.trace

A tool to help trace the activity of your clojure code.

before you start

You can find clojure.tools.trace documentation at…
http://clojure.github.io/tools.trace/index.html

…and the source at…
https://github.com/clojure/tools.trace

You either (i) need to point at the relevant jar from your project or (ii) download the source into your project, then load it.

When that is done, make the tracer available as follows…

```
user=> (require '[clojure.tools.trace :refer :all])
nil
```

The trace documentation describes various useful fns/macros. We will use two of them.

trace-vars & trace-ns

```
(defn sigma [n]
  (if (zero? n) 0
    (+ n (sigma (dec n)))))
#'user/sigma

user=> (trace-vars sigma)
#'user/sigma

user=> (sigma 5)
TRACE t267: (user/sigma 5)
TRACE t268: | (user/sigma 4)
TRACE t269: | | (user/sigma 3)
TRACE t270: | | | (user/sigma 2)
TRACE t271: | | | | (user/sigma 1)
TRACE t272: | | | | | (user/sigma 0)
TRACE t272: | | | | | => 0
TRACE t271: | | | | => 1
TRACE t270: | | | => 3
TRACE t269: | | => 6
TRACE t268: | => 10
TRACE t267: => 15
15
```

```
user=> (untrace-vars sigma)
#'user/sigma

user=> (sigma 5)
15
```

next we modify the definition of sigma to use add…

```
(defn add [x y] (+ x y))
#'user/add

(defn sigma [n]
  (if (zero? n) 0
    (add n (sigma (dec n)))))
#'user/sigma

user=> (trace-vars add)
#'user/add
user=> (sigma 4)
TRACE t367: (user/add 1 0)
TRACE t367: => 1
TRACE t368: (user/add 2 1)
TRACE t368: => 3
TRACE t369: (user/add 3 3)
TRACE t369: => 6
TRACE t370: (user/add 4 6)
TRACE t370: => 10
10

user=> (trace-vars sigma)
#'user/sigma
user=> (sigma 4)
TRACE t375: (user/sigma 4)
TRACE t376: | (user/sigma 3)
TRACE t377: | | (user/sigma 2)
TRACE t378: | | | (user/sigma 1)
TRACE t379: | | | | (user/sigma 0)
TRACE t379: | | | | => 0
TRACE t380: | | | | (user/add 1 0)
TRACE t380: | | | | => 1
TRACE t378: | | | => 1
TRACE t381: | | | (user/add 2 1)
TRACE t381: | | | => 3
TRACE t377: | | => 3
TRACE t382: | | (user/add 3 3)
TRACE t382: | | => 6
TRACE t376: | => 6
TRACE t383: | (user/add 4 6)
TRACE t383: | => 10
TRACE t375: => 10
10
user=> (untrace-vars sigma add)
#'user/add
user=> (sigma 4)
10
```

```
user=> (trace-ns 'user)   ;; trace name-space
nil

user=> (sigma 4)
TRACE t333: (user/sigma 4)
TRACE t334: | (user/sigma 3)
TRACE t335: | | (user/sigma 2)
TRACE t336: | | | (user/sigma 1)
TRACE t337: | | | | (user/sigma 0)
TRACE t337: | | | | => 0
TRACE t338: | | | | (user/add 1 0)
TRACE t338: | | | | => 1
TRACE t336: | | | => 1
TRACE t339: | | | (user/add 2 1)
TRACE t339: | | | => 3
TRACE t335: | | => 3
TRACE t340: | | (user/add 3 3)
TRACE t340: | | => 6
TRACE t334: | => 6
TRACE t341: | (user/add 4 6)
TRACE t341: | => 10
TRACE t333: => 10
10
```

A note about tail recursion & recur…

```
;; explicit tail recursion…
(defn tsig
  ([n]   (tsig n 0))
  ([n r] (if (zero? n) r
           (tsig (dec n) (+ n r)))))
#'user/tsig

user=> (trace-vars tsig)
#'user/tsig
user=> (tsig 4)
TRACE t395: (user/tsig 4)
TRACE t396: | (user/tsig 4 0)
TRACE t397: | | (user/tsig 3 4)
TRACE t398: | | | (user/tsig 2 7)
TRACE t399: | | | | (user/tsig 1 9)
TRACE t400: | | | | | (user/tsig 0 10)
TRACE t400: | | | | | => 10
TRACE t399: | | | | => 10
TRACE t398: | | | => 10
TRACE t397: | | => 10
TRACE t396: | => 10
TRACE t395: => 10
10
```

```
;; recursing with recur…
(defn tsig
  ([n]   (tsig n 0))
  ([n r] (if (zero? n) r
           (recur (dec n) (+ n r)))))

user=> (trace-vars tsig)
#'user/tsig
user=> (tsig 4)
TRACE t408: (user/tsig 4)
TRACE t409: | (user/tsig 4 0)
TRACE t409: | => 10
TRACE t408: => 10
10
```