Simpler webapps with Clojure

#{“October” “Amsterdam” “Clojure”}

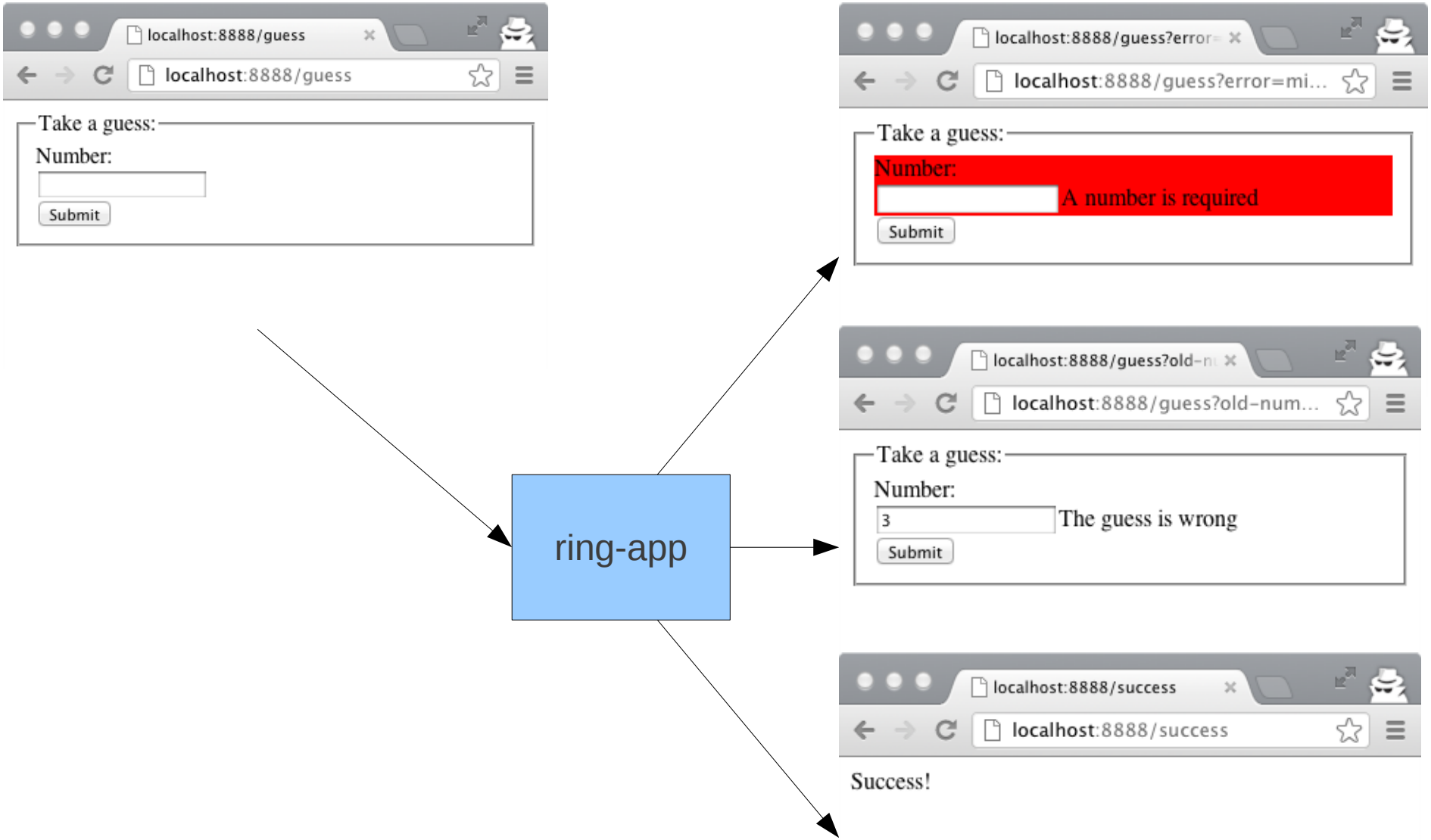
27th October 2012

Gijs Stuurman

About me

- Gijs Stuurman / thegeez.net / @thegeez / thegeez.github.com
  - Using Clojure since May 2010
  - Worked at a website for a year
    - using Ruby on Rails
  - Freelance Clojure programmer
  - Currently at a start-up using Clojure
    - Located in Amsterdam, we are hiring!

Example Application



Bare metal Http Request

```
curl -v --data "number=3" localhost:8888/guess
* About to connect() to localhost port 8888 (#0)
*   Trying ::1... connected
* Connected to localhost (::1) port 8888 (#0)
> POST /guess HTTP/1.1
> User-Agent: curl/7.21.0 (i686-pc-linux-gnu) libcurl/7.21.0
OpenSSL/0.9.8o zlib/1.2.3.4 libidn/1.18
> Host: localhost:8888
> Accept: */*
> Content-Length: 8
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 302 Found
< Date: Wed, 24 Oct 2012 19:46:52 GMT
< Location: /guess?old-number=3
< Content-Length: 0
< Server: Jetty(7.6.1.v20120215)
<
* Connection #0 to host localhost left intact
* Closing connection #0
```

Bare metal http request in Clojure

```clojure
(ns example.curl-test
  (:use example.core
        clojure.test))

(deftest post-submit-test-fail
  (testing "A http request as Clojure code"
    (let [request {:scheme :http
                   :request-method :post
                   :content-type "application/x-www-form-urlencoded"
                   :uri "/guess"
                   :server-name "localhost"
                   :params {"number" "3"}
                   :headers {"user-agent" "Mozilla/5.0 Firefox/11.0"}
                   :content-length 8
                   :server-port 8888}
          response {:status 302
                    :headers {"Location" "/guess?old-number=3"}
                    :body ""}]
      (is (= (ring-app request)
             response)))))
```

Data representation

- Scalars

Type	Example	Java equivalent
String	“foo”	String
regex	#”fo*”	Pattern
Integer	42	Integer/Long
boolean	true	Boolean
nil	nil	null
symbol	foo, +	
keyword	:foo	

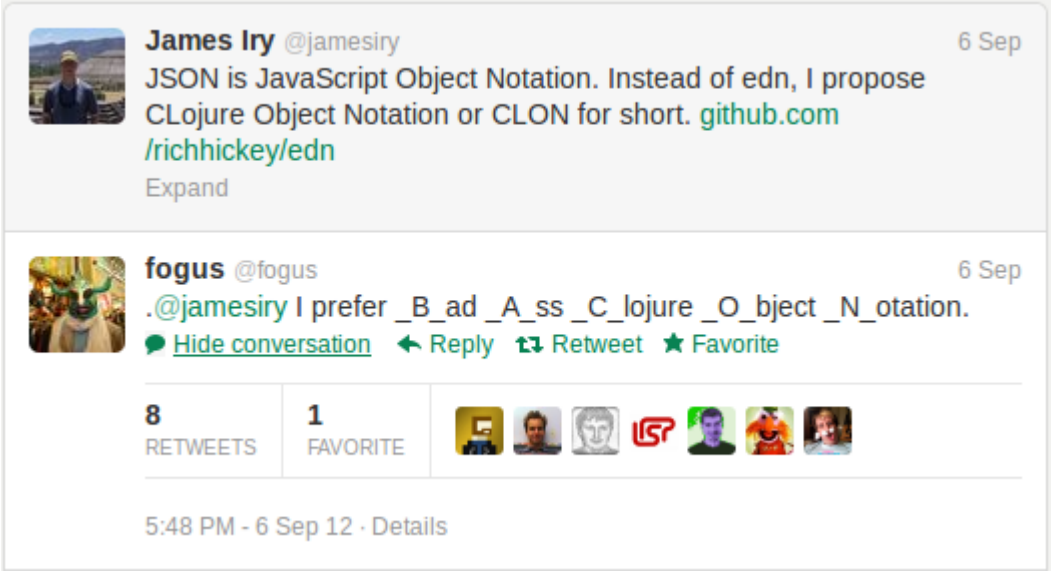
- Collections

Type	Properties	Example
list	singly-linked, insert at front	(1 2 3)
vector	indexed, insert at rear	[1 2 3]
map	key/value	{:a 100 :b 90}
set	key	#{:a :b}

Slide: https://github.com/stuarthalloway/clojure-presentations/blob/master/Clojure-4up.pdf

EDN - extensible data notation

https://github.com/edn-format



Building data programmatically

- Building

```
(def v [1 2 3])

(conj v 4)
;; => [1 2 3 4]

(def m {:a 2})

(assoc m :b v)
;; => {:a 2
;;     :b [1 2 3 4]}
```

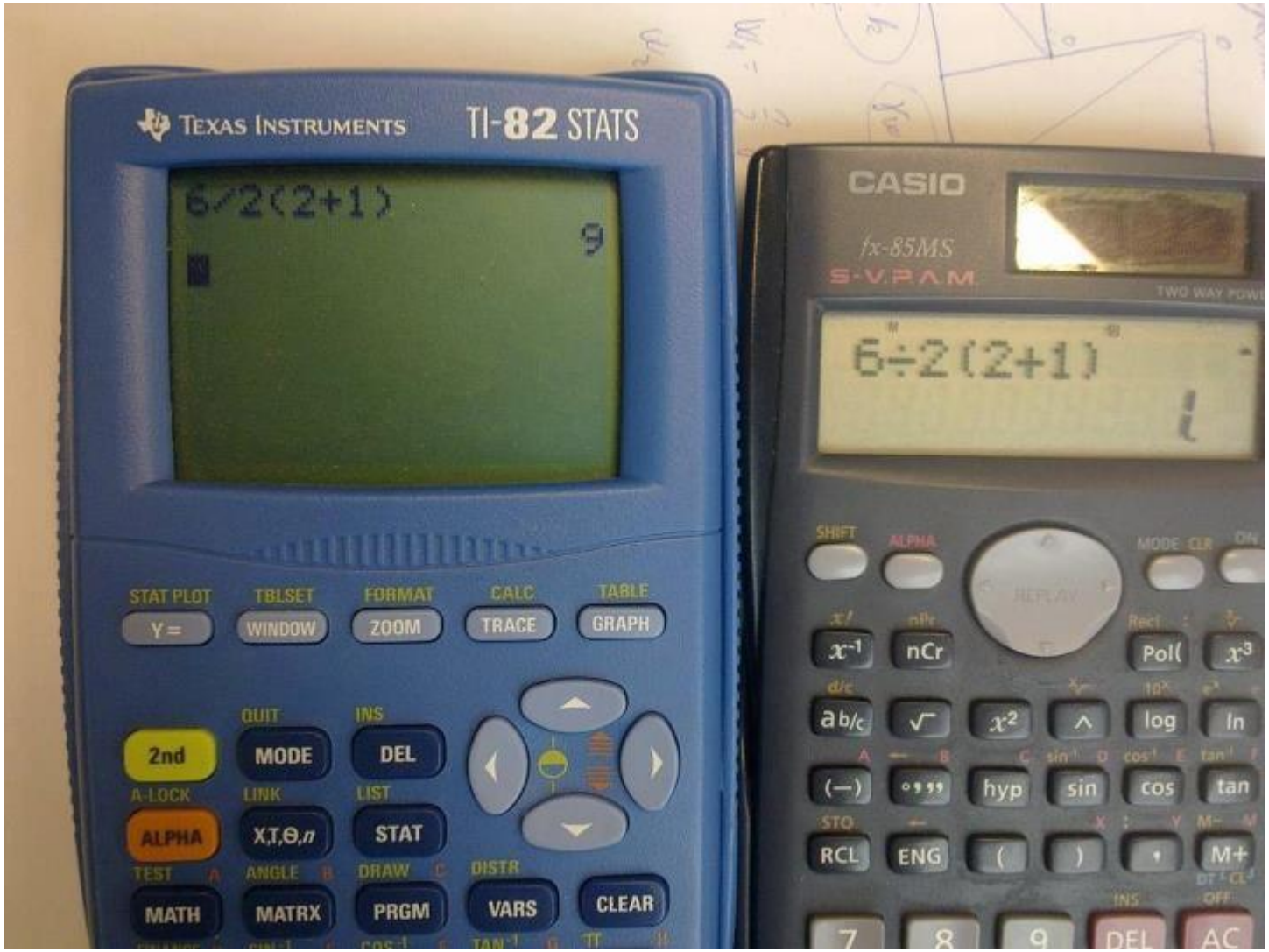
- Retrieving content

```
(get v 2)
;; => 3

(v 2)
;; => 3

(get m :a)
;; => 2

(get-in (assoc m :b v) [:b 2])
;; => 3
```

Bare metal http request in Clojure

```clojure
(ns example.curl-test
  (:use example.core
        clojure.test))

(deftest post-submit-test-fail
  (testing "A http request as Clojure code"
    (let [request {:scheme :http
                   :request-method :post
                   :content-type "application/x-www-form-urlencoded"
                   :uri "/guess"
                   :server-name "localhost"
                   :params {"number" "3"}
                   :headers {"user-agent" "Mozilla/5.0 Firefox/11.0"}
                   :content-length 8
                   :server-port 8888}
          response {:status 302
                    :headers {"Location" "/guess?old-number=3"}
                    :body ""}]
      (is (= (ring-app request)
             response)))))
```

Parenthesis



Parenthesis

```
(deftest a-test-case
  (testing "Arithmetic should work"
    (is (= (/ 6
              (* 2
                 (+ 2 1)))
           1))
    (is (= (* (/ 6 2)
              (+ 2 1))
           9))))
```

Functional programming

```
;; f(a,b) = a + b
(def plus (fn [a b]
            (+ a b)))

(defn plus [a b]
  (+ a b))

(plus 3 4)
;; => 7
```

Bare metal http request in Clojure

```clojure
(ns example.curl-test
  (:use example.core
        clojure.test))

(deftest post-submit-test-fail
  (testing "A http request as Clojure code"
    (let [request {:scheme :http
                   :request-method :post
                   :content-type "application/x-www-form-urlencoded"
                   :uri "/guess"
                   :server-name "localhost"
                   :params {"number" "3"}
                   :headers {"user-agent" "Mozilla/5.0 Firefox/11.0"}
                   :content-length 8
                   :server-port 8888}
          response {:status 302
                    :headers {"Location" "/guess?old-number=3"}
                    :body ""}]
      (is (= (ring-app request)
             response)))))
```

Ring

"Ring"

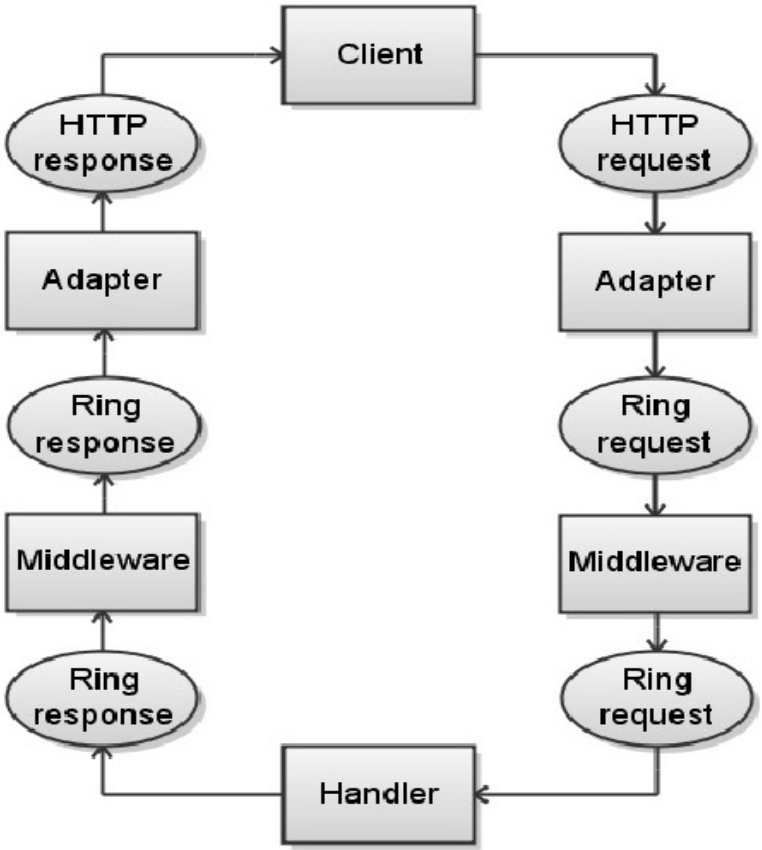
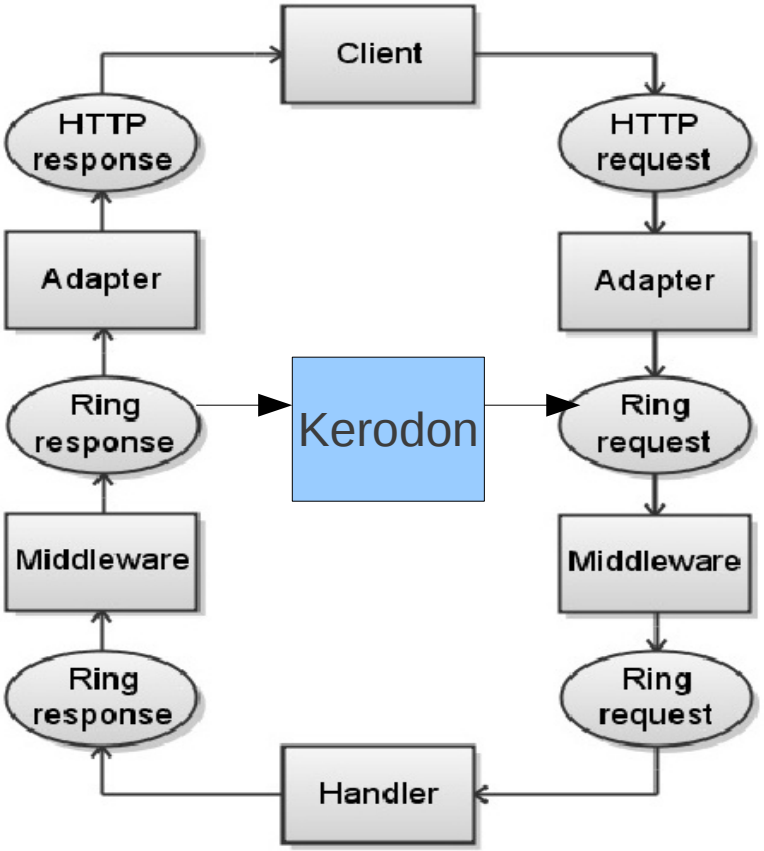


Image by Mark McGranaghan

```clojure
(ns example.core
  (:use compojure.core)
  (:require [compojure.route :as route]
            [hiccup.core :as hiccup]
            [hiccup.form :as form]
            [ring.adapter.jetty :as jetty]
            [ring.middleware.reload :as reload]
            [ring.middleware.session :as session]
            [ring.middleware.params :as params]
            [ring.util.response :as response]))

(defn guess-page [req]
  (hiccup/html
      (form/form-to
       [:post "/guess"]
       [:fieldset
        [:legend "Take a guess:"]
        (let [old-number (get-in req [:params "old-number"])
              missing-number-error (= (get-in req [:params "error"]) "missing-number")]
          [:div (when missing-number-error
                  {:style "background-color: red"})
           (form/label :number "Number:")
           [:div.controls
            (form/text-field :number old-number)
            (when old-number
              [:span.message "The guess is wrong"])
            (when missing-number-error
              [:span.error-message "A number is required"])]]])
        (form/submit-button "Submit")])))

(defn handle-guess-page [req]
  (let [number-str (get-in req [:params "number"])]
    (if-let [number (try (Integer/parseInt number-str)
                         (catch Exception e
                           nil))]
      (if (= number 42)
        (response/redirect "/success")
        (response/redirect (str "/guess?old-number=" number)))
      (response/redirect (str "/guess?error=missing-number")))))

(defn app-routes [req]
  (routing req
           (GET "/index" [] "Hello, world!")
           (GET "/guess" [] guess-page)
           (POST "/guess" [] handle-guess-page)
           (GET "/success" [] "Success!")
           (route/not-found "Page not found")))

(def ring-app
     (-> app-routes
         params/wrap-params
         (reload/wrap-reload ['example.core])))

(defn -main
  [& args]
  (jetty/run-jetty #'ring-app {:port 8888 :join? false}))
```

Kerodon

"Ring"



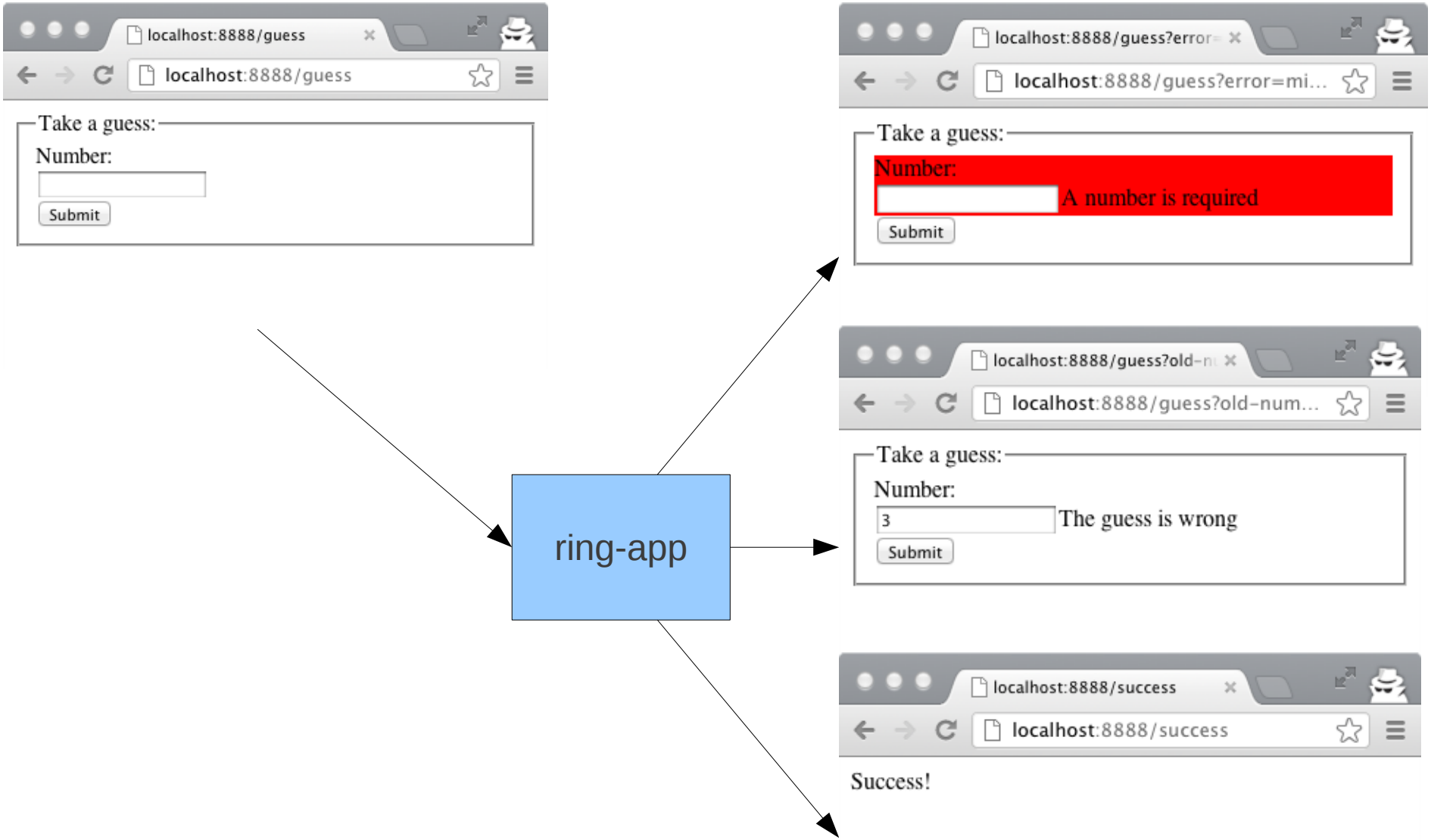
Kerodon / Peridot

- https://github.com/xeqi/kerodon
- Interaction testing, specify what the user would do and should see
- Similar to Capybara from the Ruby on Rails ecosystem
- The output/input data for Ring is the input/output data for Kerodon
- Mocking a browser is 500 lines

Testing the ring app

```
(deftest get-page-route
  (testing "Page shows a guess box"
    (-> (session ring-app)
        (visit "/guess")
        (within [:legend]
                (has (text? "Take a guess:"))))))

;; validation fail
(deftest validation-fail
  (testing "Submitting without content in input should show error message"
    (-> (session ring-app)
        (visit "/guess")
        (press "Submit")

        (follow-redirect)
        (within [:span.error-message]
                (has (text? "A number is required"))))))
```

Example Application

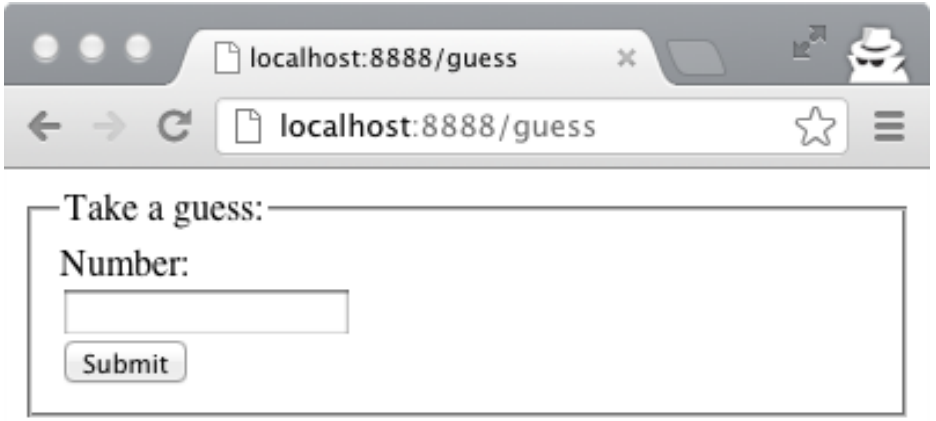


The -> threading macro

```
(deftest index-route
  (-> (session ring-app)
      (visit "/index")
      (has (text? "Hello, world!"))))

(deftest index-route-nested
  (has (visit (session ring-app)
              "/index")
       (text? "Hello, world!")))

(deftest index-route-let
  (let [session-begin (session ring-app)
        session-visit (visit session-begin "/index")]
    (has session-visit (text? "Hello, world!"))))
```

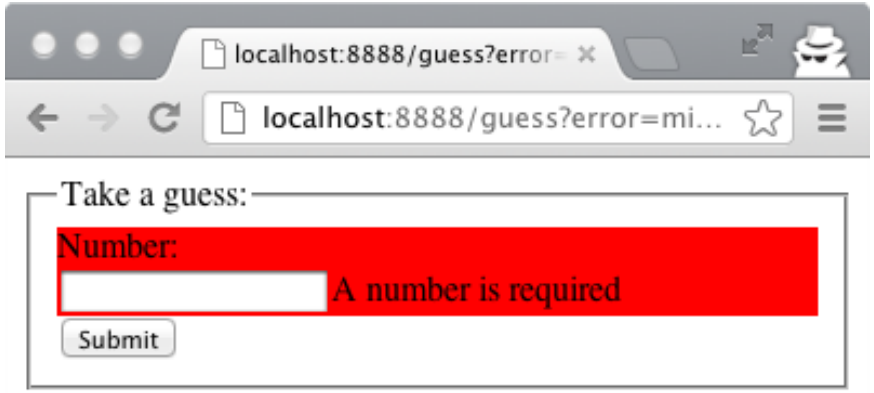
Basic test

```
(deftest get-page-route
  (testing "Page shows a guess box"
    (-> (session ring-app)
        (visit "/guess")
        (within [:legend]
                (has (text? "Take a guess:"))))))
```



Validation test

```
(deftest validation-fail
  (testing "Submitting without content in input should show error message"
    (-> (session ring-app)
        (visit "/guess")
        (press "Submit")

        (follow-redirect)
        (within [:span.error-message]
                (has (text? "A number is required"))))))
```

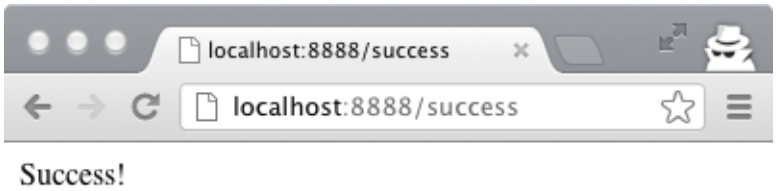


Wrong guess test

```
(deftest failure-page
  (testing "Submitting a wrong answer show a message"
    (-> (session ring-app)
        (visit "/guess")
        (fill-in "Number:" "3")
        (press "Submit")

        (follow-redirect)
        (within [:span.message]
                (has (text? "The guess is wrong"))))))
```

Correct guess

```
(deftest success-page
  (testing "Submitting the right answer gets you to /success"
    (-> (session ring-app)
        (visit "/guess")
        (fill-in "Number:" "42")
        (press "Submit")

        (follow-redirect)
        (has (text? "Success!")))))
```



Inspecting the session

```
(deftest failure-page-inspect
  (testing "Submitting a wrong answer show a message"
    (-> (session ring-app)
        (visit "/guess")
        (fill-in "Number:" "3")
        (press "Submit")

        (follow-redirect)
        ((fn [session]
           (println "session: " session)
           session))
        (within [:span.message]
                (has (text? "The guess is wrong"))))))
```

session:

```
{:response {:status 200, :headers {"Content-Type" "..."},
            :body "<form action=\"/guess\" method=\"POST\">...</form>"},
 :enlive ({:tag :html, :attrs nil, :content ("...")}),
 :request {:server-port 80,
           :server-name "localhost",
           :uri "/guess",
           :query-string "old-number=3",
           :scheme :http,
           :request-method :get,
           :headers {"referrer" "http://localhost/guess"}},
 :app #<reload$wrap_reload$fn__1412@16df0d6>}
```

Testing the browser with Selenium

"Ring"

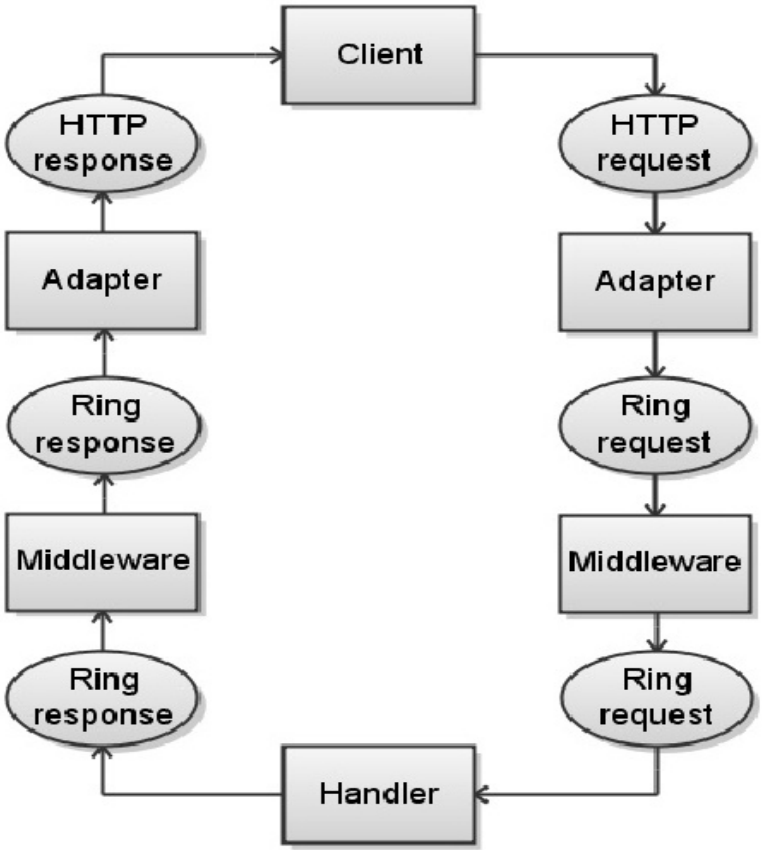


Image by Mark McGranaghan

Testing with Selenium

```
(deftest validation-fail
  (let [driver (w/new-driver :firefox)]
    (w/to driver "http://localhost:8888/guess")
    (w/input-text (w/find-element driver {:id "number"}) "3")
    (w/click (w/find-element driver {:type "submit"}))
    (is (= (w/text (w/find-element driver {:class "message"}))
           "The guess is wrong"))
    (w/quit driver)
    ))
```

Summary

- Start with Clojure, start with data and functions

Projects

- Ring: https://github.com/ring-clojure/ring
- Compojure: https://github.com/weavejester/compojure
- Kerodon: https://github.com/xeqi/kerodon
- Enlive: https://github.com/cgrand/enlive
- Clj-webdriver: https://github.com/semperos/clj-webdriver
- Example webapp: https://github.com/thegeez/amscljoct-example