Clojure in the Wild Web

7 Reflections

Ignacio Thayer, ReadyForZero.com

ReadyForZero

Summer 2010: The Beginning

- 2 person team
- Validation is key
- No communication overhead
  - "No conventions required"

ReadyForZero

1: Frameworks: easy things are easy

- ... And some complicated things are too
- Up and running
- Relatively simple at the start
- Their advantages run out
- Strong in areas that Clojure is currently weak

Early 2011: The Launch

- Django
- jQuery Soup
- Common bug patterns emerged

2: "The Most Common Bug"

Non-existent key

- Javascript "undefined"
- Python's "has no attribute"
- Clojure's missing key in map

2: "The Most Common Bug"

Contracts

```
(defconstrainedfn create-client
    "Creates Billpay client ..."
    [user params]
    [(-> params :first-name string?)
     (-> params :last-name string?)
     (-> params :dob date-string?)
     (-> params :address1 string?)
     (-> params :address2 ((optional string?)))
    =>
    valid-response?]
    ;; do-stuff)
```

ReadyForZero

2: "The Most Common Bug"

Checked threading

```
# (def x {:y {:z 1}}])
# (-> x :y :z)
1
# (-> x :y :q)
nil
# (-!> x :y :q)     ;; "(-!> (-!> x :y) :q) is nil!"
```

ReadyForZero

3 months after launch

Began using Clojure

- Initially for analysis
- REPL sold it

3: The Clojure REPL is a delight

In general

- Build code up
- Real data

What's different?

Composable syntax

```
# (map :date_joined users)
[2012-01-02 2012-01-03 ...]
# (filter after-xmas? (map :date_joined users))
[2012-12-26 ...]
# (count (filter after-xmas? (map :date_joined users))
1291
```

Concatenative Programming

Threading (->)

```
# (-> user (assoc :logins (get-logins user))
           (assoc :balance (get-balance user)))

  {:id 1 :email "..." :logins [1 3 5] :balance 123.0}
```

3: The Clojure REPL is a delight

Concatenative Programming

Thrush (->>)

```
# (->> users count)
1000
# (->> users (take 5) (map println))
...
# (->> users (map :date_joined) (take 5) (map println))
....
# (->> users (map :date_joined) (filter after-xmas?))
```

ReadyForZero

Series A

Thinking ahead

- Next Milestones
- Hiring

ReadyForZero

Webapp in Clojure

- Noir
- Korma
- Postgres
- Mongo (analytics only)
- Backbone.js

4: Code as Communication

- Succinct nor verbose is comprehensible
- Use the expressiveness of the language to promote comprehension
- Keep namespaces clean

DSLs: Web endpoints

```
(defendpoint mobile1 [:post "/login"]
  (out-example examples/login)
  (in {:email string?
       :password string?})
  (out {:success not-nil?})
  (return {:as creds}
      ....))
```

DSL: User Notifications

```
(defnotification :successful-payment
  (email {:subject "Your payment has arrived"
                :template "rfz_plus_successful_payment"})
  (web {:text "Your payment of {{amt}} to {{dest}}..."})
  (mobile-push {:text "Payment to {{dest}} delivered."
                 :location "rfz://tabs/payments"})
  (limit 1 :by :id :ever))
```

Same level as the language

Why should there be privileged syntax?

```
# (defn+ mult [x] (* x 3))
# (if+ (even? x) (/ 2 x) (-> x (* 3) (+ 1))
# (let+ [x 3] (println x))
```

5: Humility and convention

- Code should look like the code around it
- Be humble, agree to them as a team, and enforce them (☃ x)

ReadyForZero

6: Maps

- Think in pipelines with ->
- Prefer maps to tuples or vectors
- Keep them flat
- Use built-in functions

7: Skip the trickiest code

Concurrency

- pmap
- pvalues

```
(let [[l b] (pvalues (get-logins) (get-balance))]
   (-> user (assoc :logins l)
            (assoc :balance b)
```

ReadyForZero

Clojure is great for non-trivial apps

- Tricky made simple
- Convention and culture
- Expressiveness

nacho@readyforzero.com

ReadyForZero