Clojure 1.10 Preview

Rise of Devs

Topics

- Error messages
- Datafy
- Taps
- PREPL
- Showcase
- MISC

Error Messages

- #1 complaint for a long time
- Improved error messages <> shorter
- Example:
    - 1.10: *Syntax error reading source at (2:0). Invalid token: :::5*
    - 1.9: *RuntimeException Invalid token: :::5 clojure.lang.Util.runtimeException (Util.java:221)*

Datafy

```
(-> String d/datafy :members first)
[CASE_INSENSITIVE_ORDER
 [#clojure.reflect.Field{:name CASE_INSENSITIVE_ORDER,
                         :type java.util.Comparator,
                         :declaring-class java.lang.String,
                         :flags #{:public :static :final}}]]
```

- object to data transformation

- The data transformation process can be influenced by consumers using protocols or metadata.

Taps

```
(def context (StringBuilder.))

(defn ->context [x]
  (doto context
    (.append x)))

;; Then let us add above fn to the tapset
(add-tap ->context)
;; Then from any where of our running code, we can do:
(tap> "******* tap start ********\n ")
(tap> "runing.....................\n")
(tap> "******* tap end **********\n ")

;; It will be executed in a separate dedicated thread and will not
;; block or interfere with our running code. Then we print out the context:
(str context)
;; which results in:
;;******* tap start ********
;; runing.....................
;;******* tap end **********

;; Remember to remove the ->context fn once you are done with that session:
(remove-tap ->context)
;; If there is no fn added to the tap, any values you send to tap will be discarded.
(tap> "your magic") ;; your magic will be discarded.
```

PREPL

- “Programmable REPL”
- prepl
- io-prepl
- remote-prepl

What Is PREPL?

```
-------------------------
clojure.core.server/prepl
([in-reader out-fn & {:keys [stdin]}])
  a REPL with structured output (for programs)
  reads forms to eval from in-reader (a LineNumberingPushbackReader)
  Closing the input or passing the form :repl/quit will cause it to return

  Calls out-fn with data, one of:
  {:tag :ret
   :val val ;;eval result
   :ns ns-name-string
   :ms long ;;eval time in milliseconds
   :form string ;;iff successfully read
   :clojure.error/phase (:execution et al per clojure.main/ex-triage) ;;iff error occurred
  }
  {:tag :out
   :val string} ;chars from during-eval *out*
  {:tag :err
   :val string} ;chars from during-eval *err*
  {:tag :tap
   :val val} ;values from tap>

  You might get more than one :out or :err per eval, but exactly one :ret
  tap output can happen at any time (i.e. between evals)
  If during eval an attempt is made to read *in* it will read from in-reader unless :stdin is su

  Alpha, subject to change.
```

Showtime

MISC

- Java 8 is now minimum required version for Clojure
- protocols extension by metadata - *:extend-via-metadata*
- **spec - 0.143 -> 0.2.176**
  - Add support for undefining a spec
  - Spec problem printing improved

```
user=> (s/def ::a string?)
:user/a
user=> (s/def ::a nil)
:user/a
user=> (s/get-spec ::a)
nil
```

```
(let [a 2 b ])
;;   Syntax error macroexpanding clojure.core/let at (spec.clj:5:1).
;;   () - failed: Insufficient input at: [:bindings :init-expr] spec: :clojure.core.specs.alpha/bindin
```

```
S. error in 1.9
CompilerException clojure.lang.ExceptionInfo: Call to clojure.core/let did not conform to spec:
In: [0] val: () fails spec: :clojure.core.specs.alpha/bindings at: [:args :bindings :init-expr] predicate: any?,  Insufficient input
#:clojure.spec.alpha{:problems [{:path [:args :bindings :init-expr], :reason "Insufficient input", :pred clojure.core/any?, :val (),
```

Conj 2018

- Check the talks!

Resources

- **Examples in jumarko/clojure-repl-experiments**
- **Changes to Clojure in Version 1.10**
- **Datafy and tap> in Clojure 1.10**
- CLJ-2373 (error messages)
- Clojure 1.10.0-RC1 (error messages examples)
- Clojure 1.10.0-beta2 is now available (datafy)