Clojure tools & ecosystem

Clojure User Group Bonn,
Gerrit Hentschel

9 July 2013

Agenda

1. Editors/IDEs
2. Build tools
3. Documentation and help
4. The bigger Clojure picture
5. Web development libraries
6. Hack

Prerequisites

- Java 1.6+
  - Oracle HotSpot/ OpenJDK

Editors/IDEs

- Emacs + nREPL
- Eclipse + Counterclockwise
- Intellij + LaClojure
- vim-fireplace
- Catnip
- Lighttable

How can you code without a full-blown IDE?

- Far less typing
  - No need for boilerplate class/method signature/toString/equals/toHashcode generation
- (Almost) never need a debugger

- Simple auto-completion sufficient
- Short feedback loops through REPL
- Help with parentheses

REPL

- **R**ead: User enters input:
  (+ 5 x)
- **E**val: Input is parsed into Clojure datastructure and evaluated:
  (clojure.core/+ 5 user/x)
- **P**rint: Displays result of evaluation to user:
  => *8*
- **L**oop: Repeat...

REPL-driven development

- Define some functions
- Invoke functions in the REPL
- Test edge-cases
- Verify your assumptions
- Iterate quickly

Emacs + nREPL

- https://github.com/overtone/emacs-live
  - Easy setup
  - Comes with nREPL + paredit
  - Steep Emacs learning curve
  - High productivity!

- Install by executing:

bash <(curl -fksSL https://raw.github.com/overtone/emacs-live/master/installer/install-emacs-live.sh)

- On Mac: (setq mac-command-modifier 'meta)

Paredit

- keeps parentheses balanced: () [] {} ""
- navigate and manipulate S-expressions
  - forward, backward, up, down
  - wrap, splice
  - raise
  - split, join
  - slurp, barf
- http://www.emacswiki.org/emacs/ParedditCheatsheet

- Supported in Counterclockwise as "Strict structural edit mode"

Light Table

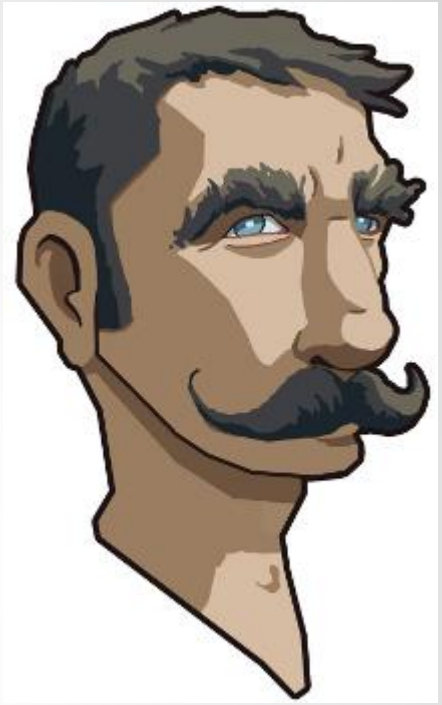
- new Clojure IDE written in ClojureScript
- Instant feedback
- Installers for Mac OS X, Linux, Windows

www.lighttable.com

Hands-on session: Install IDE

1. Install your preferred IDE

Build tools - Leiningen



Leiningen is for automating Clojure projects without setting your hair on fire.

- project creation
- dependency management
  - on top of Maven
- test execution
- REPL
- packaging code into .jar

Leiningen - Installation

1. Make sure you have a Java JDK version 6 or later.
2. Download leiningen: https://raw.github.com/technomancy/leiningen/stable/bin/lein
3. Place it on your $PATH. (~/bin is a good choice if it is on your path.)
4. Set it to be executable.
   (chmod 755 ~/bin/lein)

Build tools - Leiningen

- Create project:
  - lein new my-new-project
- Project definition in *project.clj*
  - equivalent to pom.xml / build.xml
  - Clojure datastructure

```
(defproject ugb "0.1.0-SNAPSHOT"
  :description "Awesome clojure project"
  :license {:name "Eclipse Public License"
            :url "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.5.1"]])
```

- Start REPL with all dependencies available:
  - lein repl

Leiningen - Dependencies

- Jars to add to the classpath
- Syntax:

  [org.clojure/clojure "1.5.1"]

  [<group id>/<artifact id> "<version>"]

- Find published artifacts & versions
  - www.clojars.org
  - http://search.maven.org

Leiningen - Usage

- lein test
  - Executes all tests
- lein repl
  - Starts a REPL from the command line
- lein run
  - runs the -main function of the :main namespace defined in the project.clj
- lein uberjar
  - compiles all dependencies into single executable .jar file

Namespace basics

- ns form defines namespace:

```
(ns example.core
 (:require [org.httpkit.server :as httpkit]
           [ring.util.response :refer [redirect]])
 (:import java.util.Date))
```

- in the REPL:

```
user> (require '[org.httpkit.server :as httpkit])
=> nil

user> (import 'java.util.Date)
=> java.util.Date
```

Hands-on session: Install Leiningen

1. Install leiningen
2. Create a new project
3. Update the project version to 0.1.1 and add a dependency for the latest version of *ring (1.2.0), http-kit (2.1.6) and compojure (1.1.5)*
4. Start a repl on the console

Bonus: Display the doc string of *partial* at the REPL (Hint: clojure.repl namespace)

Connecting the IDE with a REPL

Emacs:

1. Open a .clj file
2. M-x nrepl-jack-in

Light Table:

1. Open a .clj file
2. Ctrl/Cmd + Enter

Adds your dependencies to the classpath

Help getting started

- clojure.org

- clojuredocs.org
- clojure-doc.org

- 4clojure.org
- clojurekoans.com
- github.com/jackdempsey/labrepl

Literature

- Programming Clojure
- The Joy of Clojure
- Clojure Programming
- Clojure in Action
- Practical Clojure

The bigger Clojure picture

Client: ClojureScript

Server: Clojure

Database: Datomic

Data exchange: EDN

ClojureScript

- Compiles Clojure to JavaScript
- Emits code compatible with Google Closure compiler
- Can share code between server and client
- REPL support

Clojure on the server

- Request and response are simple maps
  - Ring
- Middleware based on higher-order functions
  - compose and re-use functionality piece by piece
- DSLs for generation of HTML/CSS
  - Hiccup/Garden

Datomic

- Database as a value
- Time-based facts
  - allows to revisit the past
- Powerful querying with Clojure datastructures
  - `[:find ?c :where [?c :community/name "belltown"]]`
- ACID transactions
- ...

EDN - Extensible Data Notation

- Comparable to JSON
- Subset of serializable Clojure
- Rich set of built-in elements
  - Usual primitives
  - Keywords, symbols
  - Lists, sets, maps
  - Dates: #inst "2013-07-08T19:15:13.829-00:00"
  - #uuid "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"

Libraries

- Ring, Compojure, Hiccup
- Pedestal
- http-kit

- core.logic, core.match
- Overtone
- Quill
- Prismatic Graph
- ...

Web development libraries

- ring
- compojure
- hiccup
- enlive
- http-kit
- ...

Ring

- similar to Ruby Rack & Python WSGI
- abstracts HTTP into unified API
- models requests and responses as maps
- Handlers create responses
- Middleware are higher-order functions that add functionality to handlers

https://github.com/ring-clojure/ring/wiki

Ring

Request:

```
{:uri "/my-ip"
 :headers {...}
 :remote-addr "192.168.123.4"
 :request-method :get
 :query-params {"q" "clojure"}
 ...}
```

Handler & response:

```
(defn what-is-my-ip-handler [request]
  {:status 200
   :headers {"Content-Type" "text/plain"}
   :body (:remote-addr request)})
```

Ring

Middleware:

```
(defn wrap-content-type [handler content-type]
  (fn [request]
    (let [response (handler request)]
      (assoc-in response
                [:headers "Content-Type"]
                content-type))))
```

Compose middlewares:

```
(def app
  (-> handler
      (wrap-content-type "text/html")
      (wrap-keyword-params)
      (wrap-params)))
```

Compojure

- Routing library for Ring

```
(ns hello-world
 (:require [compojure.core :refer [defroutes GET POST]]
           [compojure.route :as route))

(defroutes app
  (GET "/" [] "<h1>Hello World</h1>")
  (POST "/post" req (store! req) "<h1> Done. </hi>"
  (route/not-found "<h1>Page not found</h1>"))
```

https://github.com/weavejester/compojure/wiki

Compojure

Bind url parts to parameters

```
(GET "/:foo/:bar" {{foo :foo
                    bar :bar} :params}
  (process foo bar))
```

Support for nesting of routes

Hiccup

- Turns Clojure datastructures into HTML string
- DSL implemented via Clojure macros

```
[:a.class1 {:href "http://github.com"} "GitHub"]
```

(html5 ...)

```
<a href="http://github.com" class="class1">GitHub</a>
```

http-kit

- Ring-compatible HTTP client/server for Clojure
- Supports websockets, HTTP long-polling/streaming

Client:

```
(require '[org.httpkit.client :as http])

(let [response (http/get "http://clojure.org/")]
  (println "response's status: " (:status @response)))
```

Server:

```
(require '[org.httpkit.server :as httpkit])

(httpkit/run-server #'app {:port 8080 :join? false})
```

Hack session

1. Use hiccup/enlive to generate an HTML page containing a text input form and an unordered list
2. Populate unordered list with messages from an atom
3. Add a GET route using compojure that returns the page
4. Add a POST route that stores the entered text in the atom and redirects back to the page
5. Create a server using http-kit that serves your app

Hack session - Light Table hints

- Choose "Command" -> "Tabset: Add a tabset"
- Choose "Connect" -> "Add connection" -> "Browser" to display your page in the editor
- Press Cmd/Ctrl + R after updating code to reload browser

Hack session - Hints

Useful namespaces:

- ring.util.response
- hiccup.page
- compojure.core
- compojure.handler
- org.httpkit.server

Hack session - Hints

- Hiccup boilerplate helper & bootstrap

```
(defn layout [ & body]
  (html5 [:head
          [:title "your title"]
          [:link {:rel "stylesheet" :href "//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css"}]]
      [:body
      (list body)]))
```

- Hiccup form:

```
[:form {:method "POST"
          :action "/new-message"
          :class "form-inline"}
        ...]
```

Hack session - Bonus

- Add websocket support & ClojureScript client
- Add database to store messages