Clojure@Echo 02

Технологии вокруг Clojure

clojure-toolbox.com

clojars.org

Leiningen
Project automation

Управление зависимостями
Сборка
Тестирование
Packaging
REPL
Интеграция

Leiningen

Устройство

Использует maven repos

Значительно проще!

Расширяемый

Leiningen

lein2

Профили

Загрузка on demand

Leiningen

Пример

```
(defproject leiningen.org “1.0.0”
  :description “Generate static HTML for lein”
  :dependencies [[enlive “1.0.1”]
                 [cheshire “4.0.0”]
                 [org.markdownj/markdownj “0.3.7”]]
  :main leiningen.web)
```

Clojure contrib

clojure.test
tools.logging
data.json
data.xml
core.match
core.logic
core.cache
algo.monads
…

UI

Seesaw

Unified API

Composable functions

Binding + pipeline processing

CSS селекторы по формам

Динамизм!

Avout

Распределенный MVCC
Ref-ы, atom-ы и транзакции
  между машинами
Distributed locks
Zookeeper или mongodb-backed

Storm

Обработка потоков данных
в реальном времени
Предоставляет инфраструктуру
Произвольные топологии
Быстрый
Гарантирует обработку
Живучий
Любые языки

SQL

Korma

```
(select users
  (aggregate (count :*) :cnt)
  (where (or (> :visits 20)
             (< :last_login dt))))
```

SQL

ClojureQL

```
(-> (table :users)
    (project [:id :name :email])
    (sort [:id#asc])
    (take 5)
    (drop 2))
```

Incanter

Charting & visualization functions
Mathematical functions
Statistical functions
Matrix & linear algebra functions
Data manipulation functions

Pallet

Deployment/provisioning automation
Облачные провайдеры ч/з jcloud
Сервера ч/з ssh/bash
Скриптуется ч/з clojure,
  порождающую bash

Typed Clojure

```
(ann use-map [(HMap {:a Number}) -> Number])
(defn use-map [a]
  (get a :a))
```

IDE

Swank + SLIME

Компиляция

eval

Отладка

Документация

Есть lein-swank

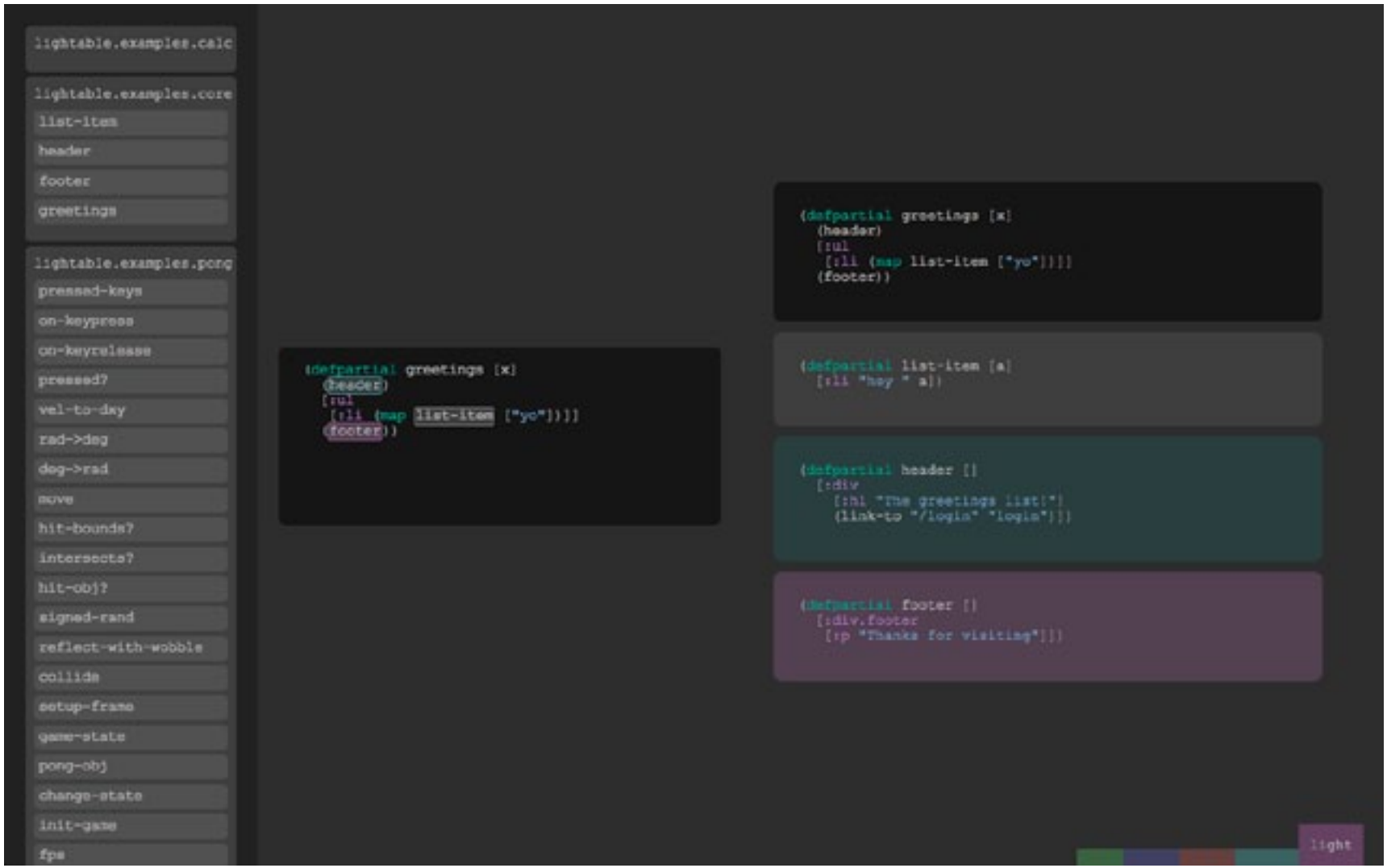
IDE

Другое

Eclipse — Counterclockwise

Idea — La Clojure

IDE
LightTable



IDE

LightTable

Web-стек

Ring
to rule them all

github.com/ring-clojure/ring/blob/master/SPEC

request map → response map

Ring

Request & response maps

```
{:keys [uri                    →  {:keys [status
        query-string                      headers
        request-method                    body]}
        headers
        body
        …]}
```

Ring

Выводы

Нет реализации —
  нет зависимостей, багов, етц
Маленькая —
  легко реализовать
Низкоуровневая
Вход-выход —
  просто тестировать
Веб-сокеты не засунешь

Ring

Адаптеры

ring-servlet
ring-jetty
ring-mongrel2-adapter
ring-netty-adapter
ring-devel для lein

Middleware

```
[(req→resp) & opts] → (req→resp)
```

ring.middleware:

+ Sessions
+ Cookies
+ Uploads
+ Form parsing

…

Диспатчинг

Moustache

```
(def my-app (app
  [“hi”] {:get “hello world only for GET!”}
  [“hi” name] {:get [“hello “ name]}))
```

Compojure

```
(defroutes app
  (GET “/” [] “<h1>Hello World</h1>”)
  (route/not-found “<h1>Page not found</h1>”))
```

Генерация HTML

Hiccup

```
(html [:span {:class “foo”} “bar”])
  => <span class=”foo”>bar</span>

(html [:div#foo.bar.baz “bang”])
  => <div id=”foo” class=”bar baz”>bang</div>
```

Генерация HTML
Enlive

Парсинг и трансформация HTML
Код отдельно от верстки
Переиспользование snippets
Модификации можно комбинировать
Макросы!

Генерация HTML

Enlive пример

```
(at a-node
  [:a :selector] a-transformation
  [:another :selector] another-transformation
  ...)

(html/deftemplate index “tutorial/template1.html”
  [ctxt]
  [:p#message] (html/content
    (get ctxt :message “Nothing to see here”))
```

Noir

Ring + Compojure + Hiccup

```
(defpage “/validate” []
         (vali/rule (= 3 3)
                    [:math “3 != 3”])
         (vali/rule (= 1 2)
                    [:math “1 != 2”])
         (layout
           [:p “Let’s check your math: “]
           [:p (str (vali/get-errors :math))])
```

WebSockets
Aleph

Сетевая библиотека
  общения по каналам
Почти как Ring, только
  request и response разделены
HTTP, WebSockets, TCP, UDP, Redis
Ожидаются:
  socket.io, AMQP, SPDY, BSON

WebSockets
Aleph пример

```
(def broadcast-channel (channel))

(defn chat-handler [ch handshake]
  (receive ch
    (fn [name]
      (siphon (map* #(str name ": " %) ch) broad-
cast-channel)
      (siphon broadcast-channel ch))))

(start-http-server
  chat-handler
  {:port 8080 :websocket true})
```

ClojureScript

Задача

Разработка больших приложений
в браузере

Проблемы

Рост кодебазы
Коллективная разработка
Взаимодействие со
  сторонними библиотеками
Оптимизация, отладка

Не-решения

CoffeeScript —
ничего не меняет в семантике

Решение

Семантические проблемы:

- состояние,
- иммутабельность,
- ФП,
- макросы,
- строгая типизация,
- протоколы

Решение-2

Одно стандартное решение
для стандартных проблем:
- неймспейсы,
- управление зависимостями,
- структуры данных,
- типовые операции

Решение-3

Компиляция в JS

Может **увеличить** производительность

Снимает browser quirks

Как это работает

Компилируется на «большой» Clojure

Генерирует Javascript

  под Google Closure Compiler

Оптимизируется

  Google Closure Compiler

Зависимости ч/з

  Goolge Closure Library

Персистентные структуры данных

Штуки

Browser-connected REPL

Reader (общий формат
передачи данных клиент-сервер)

Интероп

Библиотеки подключаются
и используются

Enfocus — enlive в браузере

Недостатки

Дебаггинг (ждем source maps)

Никита Прокопов

tonsky.livejournal.com

Echo, Ульяновск

25 июля 2012

aboutecho.com

echorussia.ru

Обсуждение лекций:

tonsky.livejournal.com/tag/clojure