Functional Programming and the Web

Dave King

June 13, 2011
About Me

- **Undergraduate**: University of Illinois at Champaign-Urbana
- **PhD**: Penn State University
  - “Retrofitting Programs for Complete Security Mediation”
  - Static analysis, type-based compiler
- **Racker**: since Fall 2009
- I’ve programmed a lot (years) of C++, Java, and ML
  - Lots of other dabbling
What I’ve Worked On At Rackspace

- Webmail Search
  - Email Parsing, Store Search Indices
- Log Search
  - Log Shipping and Parsing (Hadoop)
  - Log Hosting (Solr)
- Anti-Abuse
  - Spam Prevention
  - Blacklisting Architecture
- Cloud Control Panel:
  - Expose Rackspace Cloud functionality to our users
  - Rackspace Cloud Load Balancers
Language Topology

- Languages in the industry
Old Vanguards

- perl
- php

- Text processors, low abstraction level
Assembly Language 2.0

Unsafe languages – no runtime
  ▶ Programmers manage memory
  ▶ Thin layer on top of the machine

Can’t really trust the compiler
The New Hotness

- Interpreted full-stack solutions, high abstraction level
  - Make life easy for programmers
- No static type systems
Enterprise Languages

- Compiled full-stack solutions, high abstraction level
  - Leverage virtual machine for speed
  - Compile once, run anywhere
- Awesome static type systems
Functional Programming Languages

Common Lisp
Clojure
Ocaml
Haskell

Scheme
Racket
Standard ML
JavaScript
Scala
F#
Why Learn a Functional Language?

- New programming paradigms
  - More machine-agnostic
  - Emphasize and reuse known patterns of computation
- Powerful research applies directly to languages
How Do Functional Languages Compare?

- **Type systems:**
  - very powerful static guarantees
  - more type inference; write less types
  - **Contrast With:** Java
    - lots of type annotations

- **Expressive Syntax:**
  - well-founded macros
  - **Contrast With:** C
    - syntactic macros
  - **Contrast With:** Python
    - nice syntax – but no macros
Functional Programming

- Lots of definitions (many of them contradictory)
- Define a ‘function’ in the mathematical sense: a mapping from inputs to outputs
- A mathematical function $f$ takes arguments $x_1, \ldots, x_n$, doesn’t modify arguments, always returns the same result for the same input
- For this talk: *Functional programming is a style of programming that emphasizes building programs as composing mathematical functions*
Thesis Statements

- Learning a function language will make you a better programmer.
Common Themes

- Emphasis on:
  - recursion
  - single assignment variables
  - small units of computation
  - chaining functions together
Clojure

- Lisp implementation for the JVM
  - Lisp: one of the original high level languages
  - Common in artificial intelligence
- Main reasons to recommend:
  - Runs anywhere
  - JVM runtime
  - Lots of well-tested and mature libraries available
  - Active community
Functional Languages 101

- Read-Eval-Print-Loop interaction (REPL)
  - Build large programs out of small parts

- First-class functions

```clojure
user=> ((fn [x] (* x 3)) 5)
15
user=> (#(* 3 %1) 5)
15
```

- Pass functions to arguments

```clojure
(defn get-matching-routes [routes req]
  (filter (fn [r] ((:request r) req)) routes))
```
Maps

- **map:**
  - For each element in a sequence, perform an operation on it.

  ```clojure
  user=> (map #(* % 2) [1 2 3 4])
  (2 4 6 8)
  ```
Reduces

- reduce:
  - *From a list and a step function, build a new value.*

```clj
user=> (reduce * '(1 2 3 4 5))
120
user=> (range 1 6)
(1 2 3 4 5)
user=> (defn fact [n] (reduce * (range 1 (+ n 1))))
#'user/fact
user=> (fact 5)
120
```
Filter

- **filter:**
  - *From a list, remove elements that do not match a predicate.*

  ```clojure
  user=> (filter (fn [n] (= 0 (mod n 2))) '(1 2 3 4 5 6))
  (2 4 6)
  ```
Evangelism

- Functional languages divorce programming from the machine
- Solve big problems with small programs: recursion as a first-class citizen
- Each bit of your code is a unit of work
  - Easier to separate concerns
  - Less state means it’s easier to refactor
- **Possible Negative**: Adding extra dependencies to your functions is awkward
  - But did you need them?
  - Forces clean abstract datatypes
Java Interoperability

▷ Clojure can call any function on the classpath, just like Java.

```clojure
user=> (Integer/valueOf "42")
42
```

▷ Clojure and Java libraries often play well together

```clojure
(let [stream (java.io.ByteArrayInputStream (.getBytes (.trim xml)))]
  (xml/parse stream))
```

Dave King
Functional Programming and the Web
Persistent Maps

- Maps are first-class citizens in Clojure

  ```clojure
  user=> (def test-map {:a 1 :b 5 :c "banana"})
  #'user/test-map
  user=> test-map
  {:a 1, :b 5, :c "banana"}
  user=> (:c test-map)
  "banana"
  
  ▶ Really handy
What’s Good About Clojure

- Lean on years of Java libraries
- Extend language syntax
- Lots of lightweight libraries being written
What’s Missing in Clojure

- Checked Exceptions (yay)
- Static type system (boo)
- Tooling (boo)
  - Getting better (if you like Emacs)
At Rackspace

- Rackspace Cloud Load Balancers
- Rackspace Cloud Control Panel

- Rackspace will be open sourcing the Load Balancer API as part of OpenStack (Atlas).
Development Pains

- Constantly changing backend API
- Main bulk of the 'hard' work in the frontend (JavaScript/JSP)
- How can we still develop when the backend is unavailable?
- *Restmock:* serve static content to develop frontend logic without hitting the backend.
  - Any tool used by the team has to be a drop-in solution
  - Developers on Windows, Linux, Macintosh
  - Want a flexible 'core' that is changed by configuration
Clojure Library: Ring

- Ring (hosted on Github) abstracts the HTTP request layer
- Requests
  - Treat HTTP requests as persistent maps
- Responses
  - Convert persistent maps into HTTP responses
Interaction With Ring

- Read config file consisting of a map from routes to handlers
  - A route is a criteria for matching an HTTP request
  - A handler is a function from requests to responses
- When server receives request:
  - check if the request matches a route
    - if so, apply handler to request
  - if no route matches, return a 404 error
Take 1: Config File

```xml
<routes>
  <route>
    <path>/foo</path>
    <type>text</type>
    <config>
      <text>foo</text>
    </config>
  </route>
  <route>
    <path>/person/([0-9]+)</path>
    <type>xml</type>
    <config>
      <file>person.xml</file>
    </config>
  </route>
</routes>
```
Build Handlers for Config File

- **config-zip**: takes config file name and returns a searchable structure
- **get-handler-for-route**: return a handler function for route

```clojure
(defn config-zip [config-xml]
  (let [xml-str (slurp (ClassLoader/getSystemResource config-xml))
        stream (java.io.ByteArrayInputStream.
            (.getBytes (.trim xml-str)))
        (zip/xml-zip (xml/parse stream))]

(defn get-handler-for-route [route-zip]
  (let [type (zf/xml1-> route-zip :type zf/text)]
    (match type
      "text" (text-handler (zf/xml1-> route-zip :config :text zf/text))
      "xml" (xml-handler (zf/xml1-> route-zip :config :file zf/text))))
```

Dave King
Functional Programming and the Web
On Request, Consume Config File

- matching-uri-handler:
  - Takes an in-memory config file and a request
  - Returns the matching response handler

```clojure
(defn matching-uri-handler [routes req]
  (let [req-uri (:uri req)
        matching-specs
        (filter
         (fn [spec]
          (re-matches
           (re-pattern (:uri-re spec)) req
           -uri))
        routes)
        handlers (map :handler matching-specs)]
    (if (empty? handlers)
      {:status 404}
      (do
        (log :info (str "[HANDLER] Matched route "))))
```

Dave King
Functional Programming and the Web
Take 2: DSLs

- Clojure supports well-founded macros: replacing code with other code.
- Instead of reading XML, read a DSL.

```clojure
(route "Hello, world!"
  (request (uri "/hello")
   (response (text "Hello, world!"))))
(route "Can retrieve all the kittens"
  (request (uri "/kittens")
    (method :get)
    (response (text "Some adorable kittens!"))
  )
(route "Can’t make a new kitten"
  (request (uri "/kittens")
    (method :post)
    (response (status 422))
  )
)

- DSL implemented as macros in restmock core.
Macros: Request Criteria

- A request criteria (on URI or HTTP verb) is a function that takes a request and returns true or false.

```
(defmacro uri
  "Specifies a criteria of matching a URI"
  [path]
  `(fn [req#]
      (if (nil? (:uri req#))
        false
      (not (nil? (re-matches (re-pattern ~path)
                               (:uri req#)))))

   ` prevents evaluation of the form (code is just data)
   evaluates path
   req# generates a new variable name each time to avoid overlap.
```

Dave King
Functional Programming and the Web
Macros: Match all Criteria

- map and reduce in action: transform a list of criteria and a request into a decision: true or false.

```clojure
(defmacro request
  "Specifies a list of criteria to match a request on"
  [& criteria]
  `(fn [req#
        (reduce #(and %1 %2)
              (map #(% req#
                   (list ~@criteria))))))
```
Macros: Response

- Handler that returns static text

```clojure
(defmacro text
  "Specifies a text response handler"
  [text]
  `(text-handler ~text))

(defn text-handler [text]
  (fn [req] (response text)))
```
Macros: Routes

- Routes macro defines all of the routes that the server listens to

```clojure
(defmacro routes
  "A routes is a collection of route handlers"
  [& routes]
  `(defn route-handler [req#
    (matching-uri-handler (list ~@routes) req#)))
```
Why is this good?

- No longer tie server to static semantics
- For example:
  - Define state in config
  - Define database connection
  - Wire POST up to add values to what’s retrieved by GET
- Restmock provides a basic DSL of routes to responses
  - (then gets out of the way)
Clojure Projects to Look At

- Ring: web application library
- Compojure: lightweight MVC framework
- Enlive: selector-based templating (HTML generation)
- FleetDB: lightweight agile database
- Moustache: minimal request-to-route
- Most hosted at github.com
Functional Programming Caveats

- Not for every project
  - Domain-driven design focused on *nouns*, natural fit for OO
- Not for every business
  - Can you staff your Clojure/OCaml/Haskell project?
- Easy to glue together a lot of functionality!
  - Keep your functions short and sweet
Resources

- Several Clojure books available
  - Programming Clojure
  - The Joy of Clojure
- Structure and Interpretation of Computer Programs (MIT intro book) free online (http://sicpinclojure.com)
- Learn You a Haskell For Great Good!: http://learnyouahaskell.com
- Hacker News (for general programming language links) news.ycombinator.com
Questions?

http://www.davehking.com

Slides up at http://www.davehking.com/talks, made with wiki2beamer