Erlang
in 5 minutes or less

Devon Smith

OCLC
Programs & Research

February 27, 2008

Origins

- Comes from Ericsson
- Designed for some of their specific needs - telecom switches
  - very high uptime
  - lots and lots of concurrency
- From back in the 80's, so well established and stable
  - lots of libraries
  - increasing interest from outside telecom industry

Hardware

- Increasing CPUs (processors, cores)
- Not well supported in languages, or not supported at all
- ... so hardware isn't being used to potential

Some apps

- erlyweb - Web Framework
- yaws - Web Server
- ejabberd - Jabber/XMPP server
- CouchDB - ask Dan Scott
- RabbitMQ - Enterprise Messaging System

Bene's

- Message passing concurrency
  - Starting a process is extremely cheap and fast
  - No shared memory
  - So no semaphores, deadlocks, or other threaded evilness
- Hot code updates
  - Rarely have to bring server down to load new code
  - Good for production, also for development
- Functional language
  - Thankfully not object-oriented
  - Functional is about relationships, not instructions

Code

```
get_field ( Tag, Record ) ->
    first(
        fun (X) -> X#data.tag == Tag end,
        Record#marc.datafields,
        #data{}).
```

```
title(Record) ->
    Field = get_field("245", Record),
    join(
        lists:map(fun (X) -> element(2, X) end, Field#data.subfields),
        " ").
```

Code

```
next(File) ->
    RegName = list_to_atom(File),
    case whereis(RegName) of
        undefined ->
            register(RegName, spawn_link(marc_read, iterate, [File, self()]));
        _ -> true
    end,
    RegName ! { next, self() },
    receive
        { ok, Record } ->
            Record;
        eof ->
            unregister(RegName),
            eof;
        {'EXIT', _From, Reason} ->
            Reason;
```

Links

- http://www.erlang.org/
- http://www.pragprog.com/titles/jaerlang/index.html - PragProg Book