http://www.erlang-factory.com

# Erlang Factory Lite Kraków 2009

25 November 2009

# What is Erlang Factory Lite?

Erlang Factory

- 2 day conference
- More than 40 speakers
- London and San Francisco Bay Area

Erlang Factory Lite

- 1 day conference
- Several talks
- First edition in Krakow, Poland

Erlang User Group Meetings

- One or two talks
- Organized all around the world

# Erlang Factory Lite Kraków 2009

## 16:00 Erlang Live!

- Present some of the most popular open source and commercial projects that adapted Erlang

## 16:30 Erlang Extreme

- How Erlang concurrency model is used to build scalable and fault-tolerant designs

## 17:00 Erlang & Student

- How Erlang can help to develop academic projects and how to integrate it with other technologies

## 17:30 Erlang's Journey to the Clouds

- How Erlang is breaking out of the clusters of last century and entering today's cloud computing environments

## 19:30 Erlounge

- Trelkovsky Cafe, ul. Czarnowiejska 55

http://www.erlang-factory.com

# Erlang Live!

## Erlang Factory Lite Kraków 2009

25 November 2009

# Agenda

- One-pager about Erlang history
- One-pager about Erlang properties
- Open source and commercial projects
- Erlang community

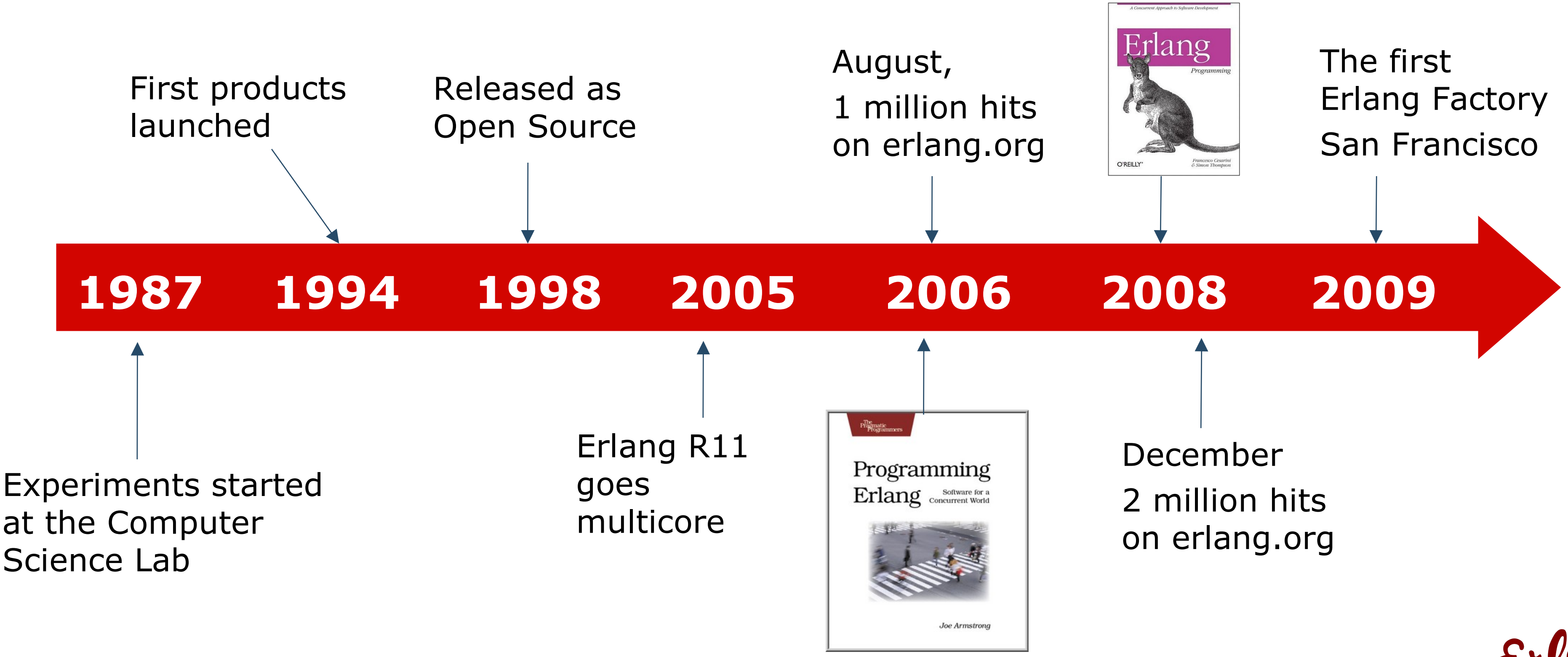
# Erlang History

…if we had to start again we would probably use Erlang…

*Mike Shaver*
*Mozilla Foundation*

"Erlang is going to be a very important language. It could be the next Java."

*Ralph Johnson*
*Co-author, "Design Patterns" (the "Gang-of-Four book")*



1987 – Experiments started at the Computer Science Lab

1994 – First products launched

1998 – Released as Open Source

2005 – Erlang R11 goes multicore

2006 – August, 1 million hits on erlang.org

2006 – Programming Erlang

2008 – Erlang Programming

2008 – December 2 million hits on erlang.org

2009 – The first Erlang Factory San Francisco

# Erlang properties

| | |
|---|---|
| Declarative | Functional programming language, high abstraction level, pattern matching and concise readable programs |
| Concurrency | Either transparent or explicit concurrency, light-weight processes and highly scalable |
| Soft real-time | Response times in the order of milliseconds per-process garbage collection |
| Robustness | Simple and consistent error recovery, supervision hierarchies and "Program for the correct case" |
| Distribution | Explicit or transparent distribution<br>Network-aware runtime system |
| Hot code loading | Easily change code in a running system. Enables non-stop operation Simplifies testing |
| External interfaces | "Ports" to the outside world behave as Erlang processes |
| Portability | Erlang runs on any UNIX, Windows, VxWorks, ...Supports heterogeneous networks |
| SMP Support | Symmetric multiprocessing support. Takes full advantage of multiple CPU architectures. |

# Apache CouchDB

## Document-oriented database

- schema-free
- replication with bi-directional conflict detection

## Queried and indexed in a MapReduce fashion

- using JavaScript
- RESTful JSON API

## NoSQL

## Who uses CouchDB

- Ubuntu Karmic Koala content repository for desktop applications
- I Play WoW – a facebook application
- Mozilla Raindrop – led by the team responsible for Thunderbird

# Facebook Chat Feature

## Chat backend in Erlang

- 1+ billion user messages / day
- 10+ million active channels at peak
- 100+ channel machines

## Architecture

- one message queue per user (Erlang process)
- HTTP long poll
- User ID space partitioned statically

Facebook chat team got **Erlang User of the Year** award in 2009!

# ejabberd

## Jabber/XMPP instant messaging server

- cross-platform
- fault-tolerant
- can be distributed on a cluster

## Implements many XEPs

- supports MySQL, PostgresSQL, ODBC, LDAP
- SASL authentication, STARTTLS, SSL

## Who uses ejabberd

- Nasza Klasa NKtalk
- jabber.org, jabster.pl
- Facebook XMPP gateway

# GitHub

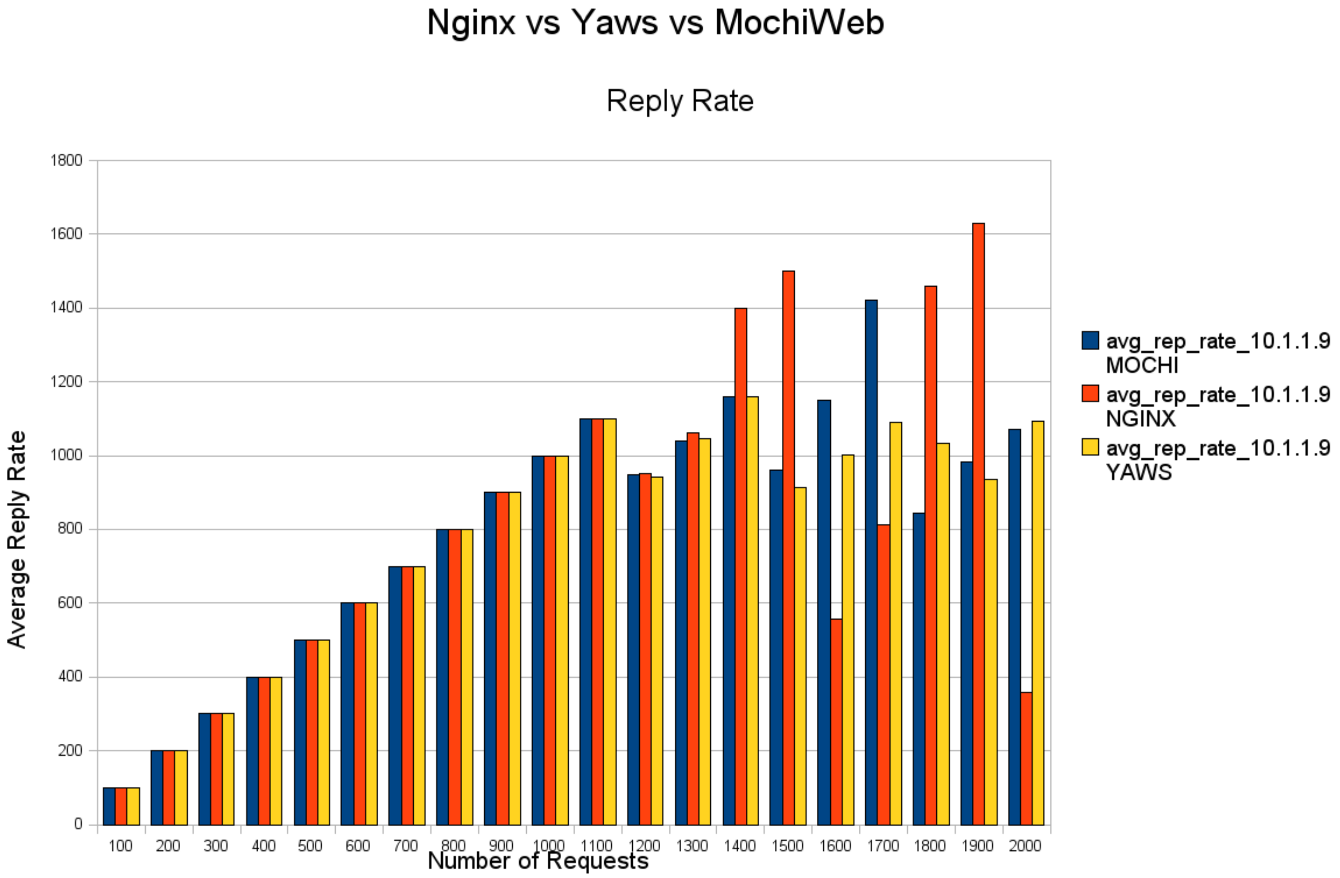
Erlang tier dispatches request to Ruby machines

- BERT (JSON-like RPC calls)
- scales on a cluster

# Yaws HTTP server

## Scalable HTTP server

- Serves static and dynamic content
- Standalone or embedded mode
- High performance



http://www.joeandmotorboat.com/2009/01/03/nginx-vs-yaws-vs-mochiweb-web-server-performance-deathmatch-part-2/

# Web frameworks

## Nitrogen

- event-driven
- support for Ajax events and Comet-like data pushes

## Erlang Web

- Classic MVC
- Django templating language
- Two-tier architecture for load balancing
- Caching layer

## WebMachine

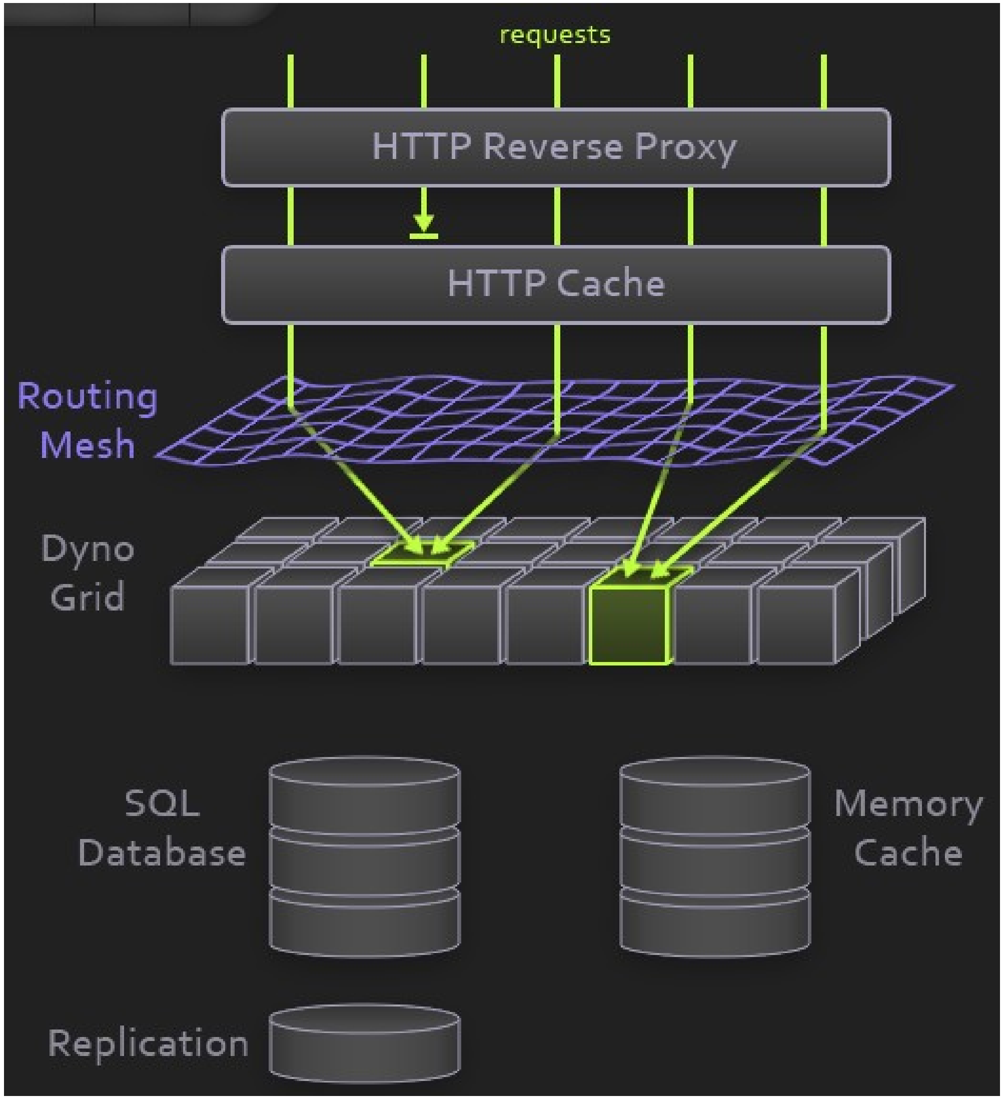
- a REST toolkit
- Built on top of mochiweb web server
- Direct mapping of HTTP to REST

# Heroku

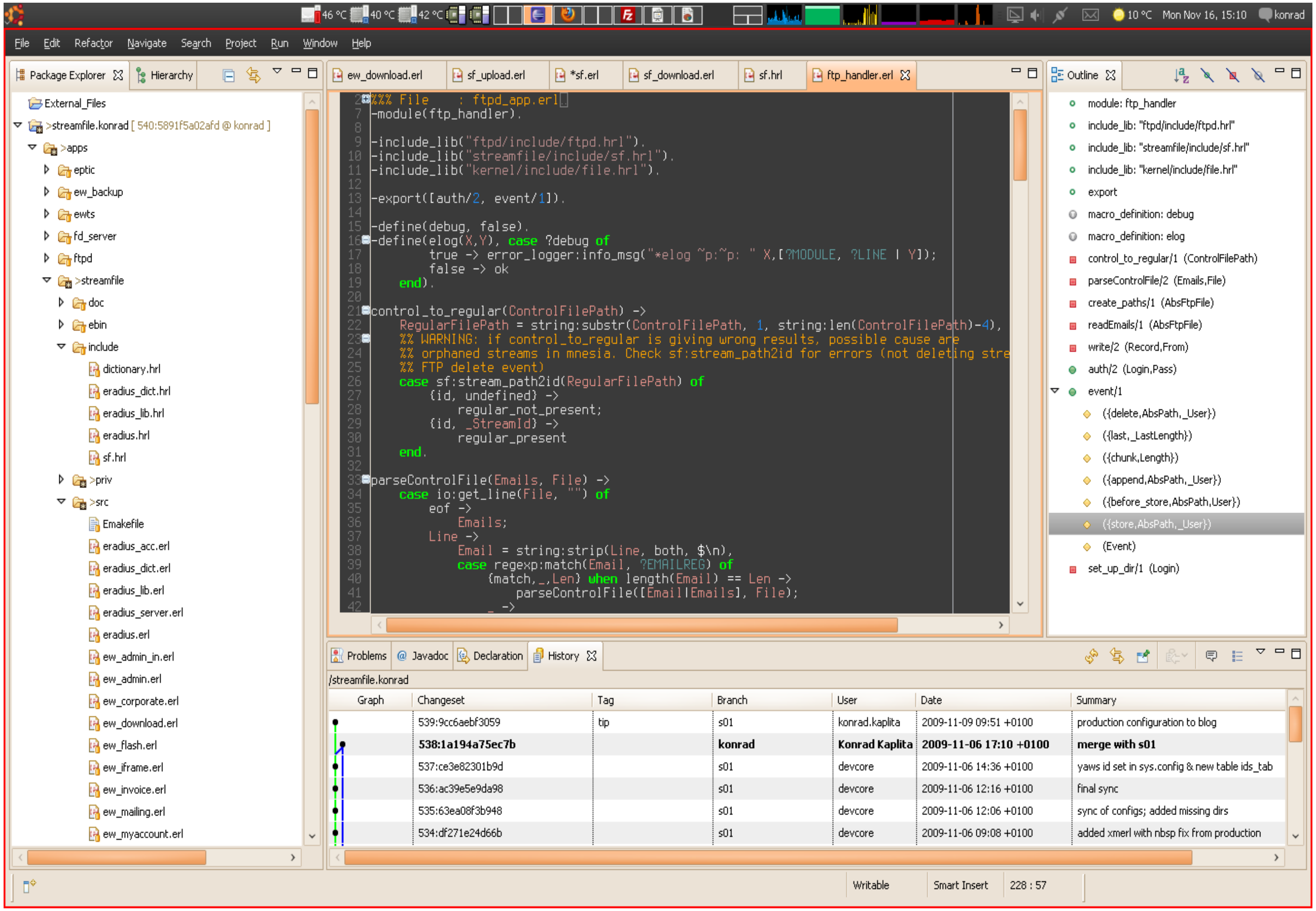
## Ruby Cloud platform

- Hosting of Ruby on Rails applications
- Over 40.000 deployed apps
- Automatic scaling in case of increased demand for resources
- Where in this picture Erlang fits?

# Heroku – Architecture overview



# Developer IDE - Eclipse

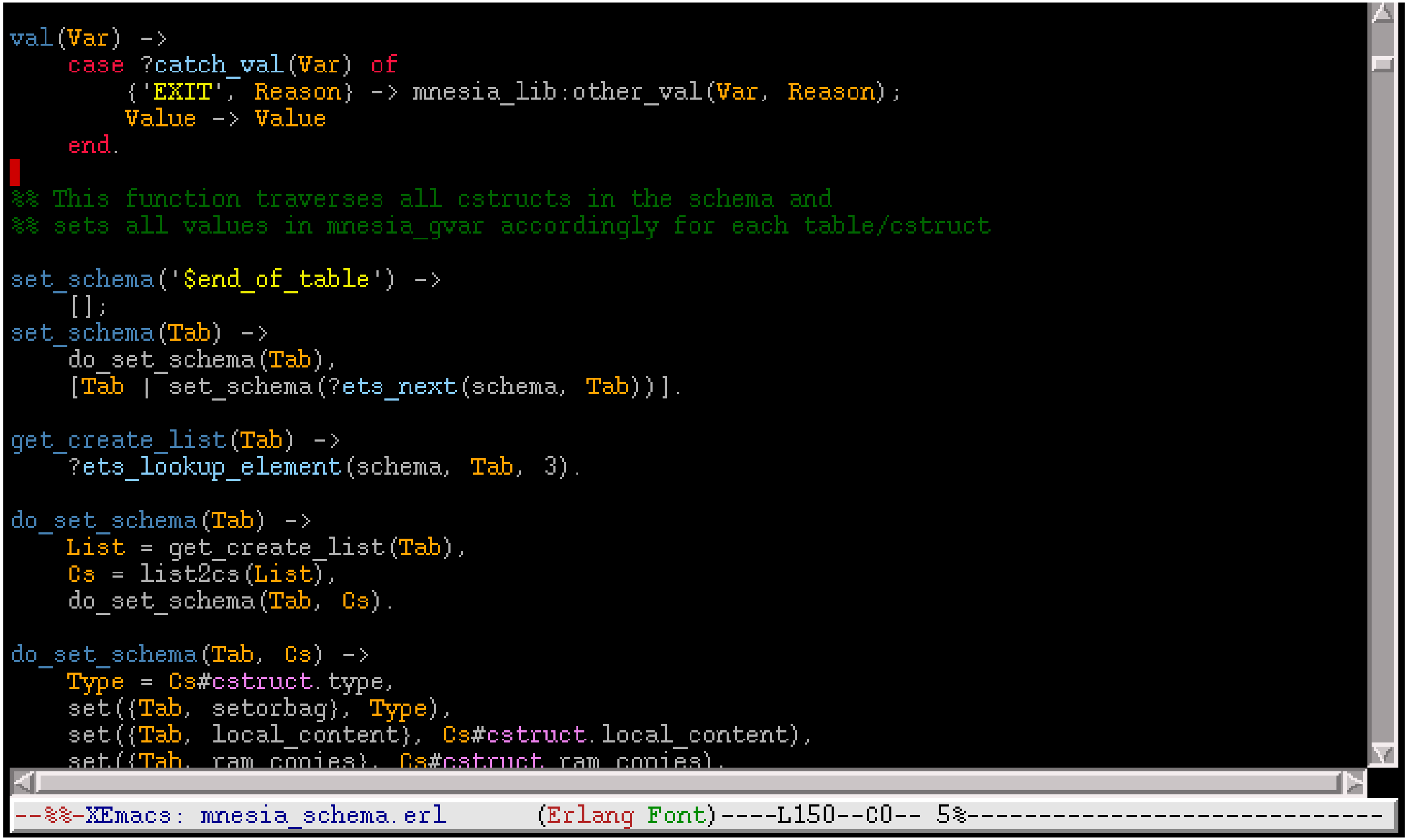


```
%%% File    : ftpd_app.erl
-module(ftp_handler).

-include_lib("ftpd/include/ftpd.hrl").
-include_lib("streamfile/include/sf.hrl").
-include_lib("kernel/include/file.hrl").

-export([auth/2, event/1]).

-define(debug, false).
-define(elog(X,Y), case ?debug of
        true -> error_logger:info_msg("*elog ~p:~p: " X,[?MODULE, ?LINE | Y]);
        false -> ok
    end).

control_to_regular(ControlFilePath) ->
    RegularFilePath = string:substr(ControlFilePath, 1, string:len(ControlFilePath)-4),
    %% WARNING: if control_to_regular is giving wrong results, possible cause are
    %% orphaned streams in mnesia. Check sf:stream_path2id for errors (not deleting stre
    %% FTP delete event)
    case sf:stream_path2id(RegularFilePath) of
        {id, undefined} ->
            regular_not_present;
        {id, _StreamId} ->
            regular_present
    end.

parseControlFile(Emails, File) ->
    case io:get_line(File, "") of
        eof ->
            Emails;
        Line ->
            Email = string:strip(Line, both, $\n),
            case regexp:match(Email, ?EMAILREG) of
                {match,_,Len} when length(Email) == Len ->
                    parseControlFile([Email|Emails], File);
                _ ->
```

/streamfile.konrad

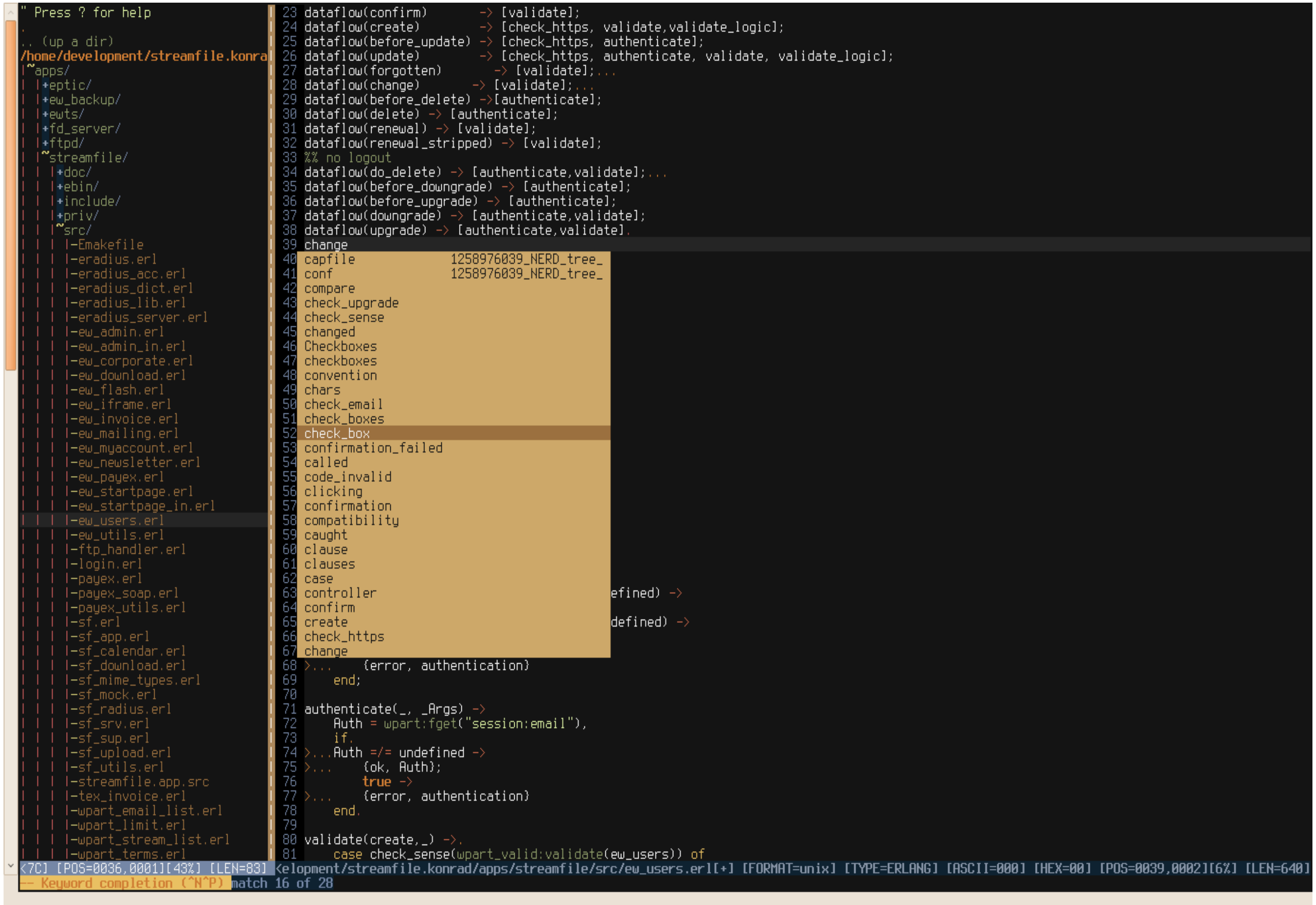
| Graph | Changeset | Tag | Branch | User | Date | Summary |
|---|---|---|---|---|---|---|
| ● | 539:9cc6aebf3059 | tip | s01 | konrad.kaplita | 2009-11-09 09:51 +0100 | production configuration to blog |
| ● | **538:1a194a75ec7b** | | **konrad** | **Konrad Kaplita** | **2009-11-06 17:10 +0100** | **merge with s01** |
| ● | 537:ce3e82301b9d | | s01 | devcore | 2009-11-06 14:36 +0100 | yaws id set in sys.config & new table ids_tab |
| ● | 536:ac39e5e9da98 | | s01 | devcore | 2009-11-06 12:16 +0100 | final sync |
| ● | 535:63ea08f3b948 | | s01 | devcore | 2009-11-06 12:06 +0100 | sync of configs; added missing dirs |
| ● | 534:df271e24d66b | | s01 | devcore | 2009-11-06 09:08 +0100 | added xmerl with nbsp fix from production |

# Developer IDE - emacs



```
val(Var) ->
    case ?catch_val(Var) of
        {'EXIT', Reason} -> mnesia_lib:other_val(Var, Reason);
        Value -> Value
    end.

%% This function traverses all cstructs in the schema and
%% sets all values in mnesia_gvar accordingly for each table/cstruct

set_schema('$end_of_table') ->
    [];
set_schema(Tab) ->
    do_set_schema(Tab),
    [Tab | set_schema(?ets_next(schema, Tab))].

get_create_list(Tab) ->
    ?ets_lookup_element(schema, Tab, 3).

do_set_schema(Tab) ->
    List = get_create_list(Tab),
    Cs = list2cs(List),
    do_set_schema(Tab, Cs).

do_set_schema(Tab, Cs) ->
    Type = Cs#cstruct.type,
    set({Tab, setorbag}, Type),
    set({Tab, local_content}, Cs#cstruct.local_content),
    set({Tab, ram_copies}, Cs#cstruct.ram_copies),
--%%-XEmacs: mnesia_schema.erl       (Erlang Font)----L150--C0-- 5%------------------------------
```

# Developer IDE - VIM



```
" Press ? for help
.
.. (up a dir)
/home/development/streamfile.konra
|~apps/
| |+eptic/
| |+ew_backup/
| |+ewts/
| |+fd_server/
| |+ftpd/
| |~streamfile/
| | |+doc/
| | |+ebin/
| | |+include/
| | |+priv/
| | |~src/
| | | |-Emakefile
| | | |-eradius.erl
| | | |-eradius_acc.erl
| | | |-eradius_dict.erl
| | | |-eradius_lib.erl
| | | |-eradius_server.erl
| | | |-ew_admin.erl
| | | |-ew_admin_in.erl
| | | |-ew_corporate.erl
| | | |-ew_download.erl
| | | |-ew_flash.erl
| | | |-ew_iframe.erl
| | | |-ew_invoice.erl
| | | |-ew_mailing.erl
| | | |-ew_myaccount.erl
| | | |-ew_newsletter.erl
| | | |-ew_payex.erl
| | | |-ew_startpage.erl
| | | |-ew_startpage_in.erl
| | | |-ew_users.erl
| | | |-ew_utils.erl
| | | |-ftp_handler.erl
| | | |-login.erl
| | | |-payex.erl
| | | |-payex_soap.erl
| | | |-payex_utils.erl
| | | |-sf.erl
| | | |-sf_app.erl
| | | |-sf_calendar.erl
| | | |-sf_download.erl
| | | |-sf_mime_types.erl
| | | |-sf_mock.erl
| | | |-sf_radius.erl
| | | |-sf_srv.erl
| | | |-sf_sup.erl
| | | |-sf_upload.erl
| | | |-sf_utils.erl
| | | |-streamfile.app.src
| | | |-tex_invoice.erl
| | | |-wpart_email_list.erl
| | | |-wpart_limit.erl
| | | |-wpart_stream_list.erl
| | | |-wpart_terms.erl
```

```
dataflow(confirm)        -> [validate];
dataflow(create)         -> [check_https, validate,validate_logic];
dataflow(before_update) -> [check_https, authenticate];
dataflow(update)         -> [check_https, authenticate, validate, validate_logic];
dataflow(forgotten)        -> [validate];...
dataflow(change)        -> [validate];...
dataflow(before_delete) ->[authenticate];
dataflow(delete) -> [authenticate];
dataflow(renewal) -> [validate];
dataflow(renewal_stripped) -> [validate];
%% no logout
dataflow(do_delete) -> [authenticate,validate];...
dataflow(before_downgrade) -> [authenticate];
dataflow(before_upgrade) -> [authenticate];
dataflow(downgrade) -> [authenticate,validate];
dataflow(upgrade) -> [authenticate,validate].
change
capfile                 1258976039_NERD_tree_
conf                    1258976039_NERD_tree_
compare
check_upgrade
check_sense
changed
Checkboxes
checkboxes
convention
chars
check_email
check_boxes
check_box
confirmation_failed
called
code_invalid
clicking
confirmation
compatibility
caught
clause
clauses
case
controller                                  efined) ->
confirm
create                                      defined) ->
check_https
change
>...    {error, authentication}
    end;

authenticate(_, _Args) ->
    Auth = wpart:fget("session:email"),
    if.
>...Auth =/= undefined ->
>...    {ok, Auth};
        true ->
>...    {error, authentication}
    end.

validate(create,_) ->.
    case check_sense(wpart_valid:validate(ew_users)) of
```

```
<7C] [POS=0036,0001][43%] [LEN=83] <elopment/streamfile.konrad/apps/streamfile/src/ew_users.erl[+] [FORMAT=unix] [TYPE=ERLANG] [ASCII=000] [HEX=00] [POS=0039,0002][6%] [LEN=640]
-- Keyword completion (^N^P) match 16 of 28
```

# Developer tools

## Refactoring

- Wrangler (built into emacs and Eclipse)
- RefactorErl

## Testing

- QuickCheck
- eUnit

# QuickCheck

## Every testing method is expensive, even automated testing

- 35% of code written at Ericsson is testing code

## Property based testing – what is this?

- Generalisastion of use cases
- Generation of use cases for free
- Test specification consists of properites and generators – formal specification
- Controlled randomness

## Where can I use this tool?

- Replace/Extend regular unit tests
- Testing of communication protocols
- Testing against specification

# QuickCheck - example

```
prop_delete(I,L) ->
  not lists:member(I, lists:delete(I,L)).
```

```
(qc@host)1 > eqc:quickcheck(eqc:numtests(1000,examples:prop_delete())).
.................................................................
....................Failed! After 346 tests.
{2,[-7,-13,-15,2,2]}
Shrinking.(1 times)
{2,[2,2]}
```

# Erlang Community

## trapexit.org

- Community Site
- FAQ, tutorials, web-based forums
- Projects aggregator
- RSS feeds aggregator

## Krakow Erlang User Group

- http://tech.groups.yahoo.com/group/erlang-krakow/