ErlyWeb

A web development framework for Erlang

Yariv Sadan
4/31/2009

Records

Defining

-record(artist, {name, album}).
-record(album, {name, song}).
-record(song, {name, length}).

Creating

Song = #song{name = “One”, length=2.13}.

Getting

Name = Song#song.name.

Setting

Song1 = Song#song{name = “Zoo Station”}

More Records

Getting

```
Song = ((Artist#artist.album)
          #album.song)#song.name
```

Setting

```
Artist2 =
Artist#artist{album=
  (Artist#artist.album)#song{name=
    ((Artist#artist.album)
      #album.song)#song{name= ”Zoo Station"}}}.
```

Recless

Creating

```
Artist =
  #artist{name = ”U2",
          album = #album{name = ”Achtung Baby",
                         song = #song{name = ”One" }}}
```

Getting

```
Artist2 = Artist.album.song.name = ”Zoo Station"
```

Setting

```
Song = Artist.album.song.name
```

Using Recless

Add this declaration

-compile({parse_transform, recless}).

This doesn’t work

get_name(Person) -> Person.name.

Instead, you must write this

get_name(Person = #person{}) ->
Person.name.

ErlyWeb Benefits

- Erlang/OTP
- Functional programming
- Concurrent programming
- MVC
- Component-oriented
- Database abstraction (ErlyDB)
- Protection from SQL injection attacks
- Hot code swapping
- Best platform for Comet applications
- Lots of fun!

ErlyDB: Database Abstraction

- DDL:

- CREATE TABLE painting (
- id integer auto_increment primary key,
- title varchar(255))

- painting.erl:

- -module(painting).
- -compile(export_all).

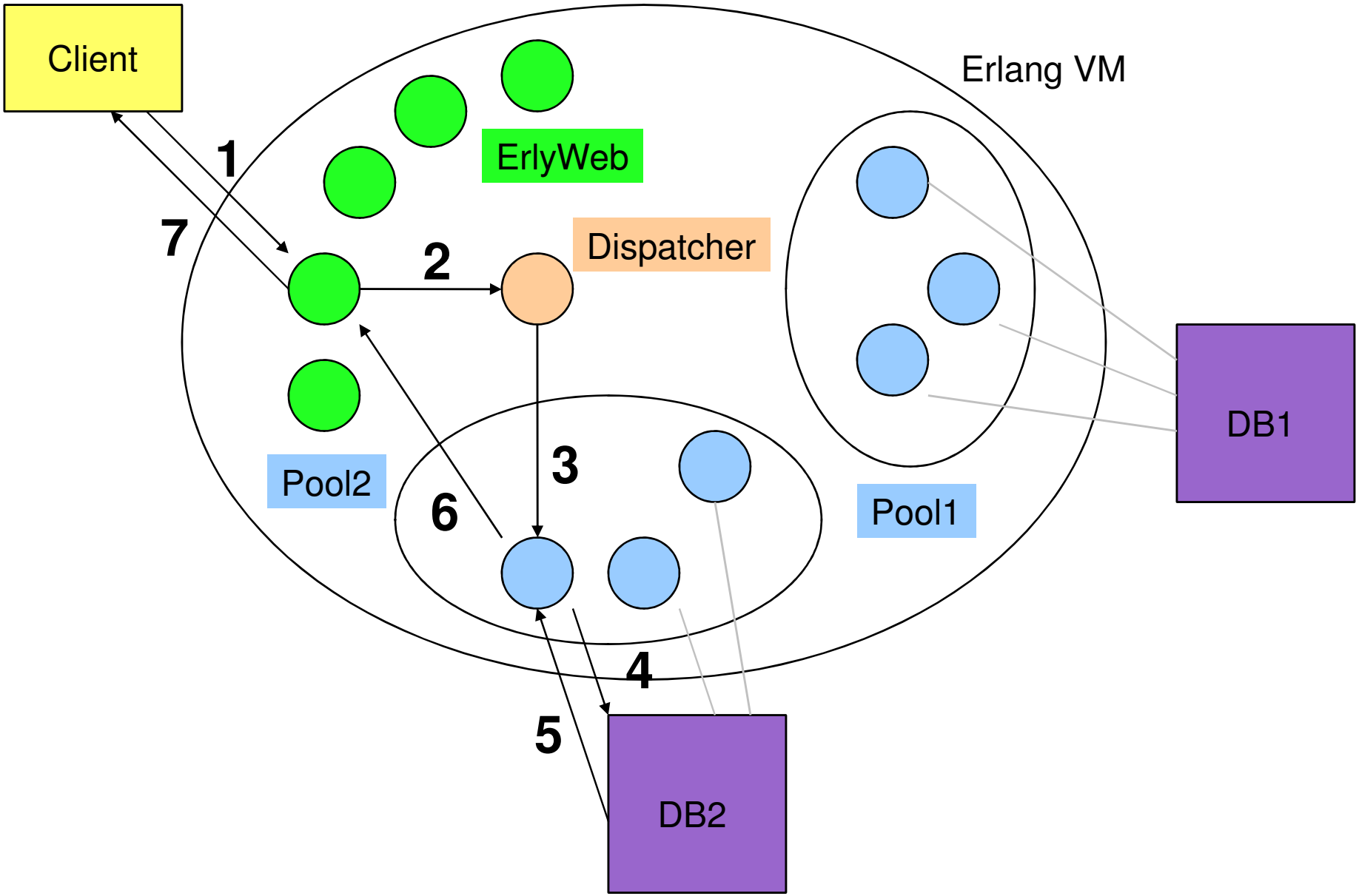
DB Access Code Example

```
Title = “landscape”,
P = painting:new(Title),  %% create a new record
painting:transaction(
 fun() ->
  P1 = painting:save(P),   %%  INSERT
  P2 = painting:title(P1, “beach”), %% change the title
  painting:save(P2),  %%  UPDATE
  %% SELECT
  Paintings =  painting:find({'or', [{id, '=', 1}, {title,like,”monster%”}]}),
  painting:delete(P)   %% DELETE
end)
```

More ErlyDB Features

- Relations (many-to-one, one-to-man, many-to-many)
  - artist:add_painting(Artist, Painting).
  - artist:paintings(Artist, {title,'=',”beach”}).
- Drivers for MySQL, Postgres and Mnesia
- Supports multiple DB's
- DB connection pooling
  - Uses Erlang concurrency
    - Dispatcher process + one process per connection
  - Transactions “Just Work”

DB Connection Pooling (MySQL)



Uses for Concurrency in Web Apps

- Connection pooling
- Parallelizing DB queries, component renderings, web service calls, etc.
- Performing background tasks
  - Updating counters, processing data/assets, communicating with backend services, etc.
- Storing shared (session) data in memory for fast access
- Comet

Components

- Component = Controller + View
- Components can be embedded in other components
  - Controllers decide what to embed, views decide where
- Phased rendering
  - First, render requested component
  - Pass the result (if any) to the enclosing component

Controller Example

```
-module(artist_controller).
-export([show/2]).

show(A, Id) ->
  %% look up the artist and related paintings
  Artist = artist:find_id(Id),
  Paintings = artist:paintings_with(Artist, [{order_by, {created_on, desc}}, {limit, 10}]),
  %% pass the artist name and a list of of rendered 'painting' subcomponents
  %% to the view function
  [{data, artist:name(Artist)},
   [{ewc, painting, [A, Painting]} || Painting <- Paintings]].
```

Views

- Views are Erlang modules (benefits: speed, reusability)
- Each controller has a view
- View function names map to controller function names
- View functions return iolists (nested lists of strings and/or binaries)
    - [“what”, [$a, <<”great”>>, [<<”painting”>>]]]
- Can be implemented in Erlang or ErlTL

ErlTL Example

- <%@ index({ok, {Username, Painting}}) %>
- Hi <% Username %>!
- Here's today's top painting: <% Painting %>

- <%@ index({error, Msgs}) %>
- Oops, the following errors occured:
- <% [err(Msg) || Msg <- Msgs] %>

- <%@ err(Msg) %> <% Msg %>

Phased Rendering Example

```
hook(A) ->
  {phased,
    {ewc, A},   %% first, render the requested component
    fun({ewc, Controller, _View, _Func, _Params}, Data, _PhasedVars) ->
      case Controller of
        ajax_controller ->
          %% if the client requested the 'ajax' component, return rendered result unchanged
          {data, Data};
        _ ->
          %% otherwise, embed the result in html_container before returning
          {ewc, html_container, [A, {data, Data}]}
      end
end}
```

ErlyWeb is Comet-Ready

- Message passing primitives
- Lightweight processes (location transparent)
- Preemptive scheduling
- Per-process heaps
- Immutable data
- Port-based interface to native code
- Mnesia (distributed store for shared data)
- Hot code swapping.

Thank you