Building tools and libraries around CouchDB in Erlang

2011 - Erlang Factory London - Benoît Chesneau

- Web craftsman
- Writing opensource for a living
- benoitc@apache.org
- couchbeam author, gunicorn author ...

Me

- library to write applications on top of CouchDB HTTP API
- started 2 years ago
- last stable version is 0.6.0
- compatible with 1.0.x, 1.1.x, trunk

What?

- Started to write couchdbproxy
- No library was doing what I wanted
- exhaustive CouchDB API support
- Handle streaming, changes &
- something simple.

Why?

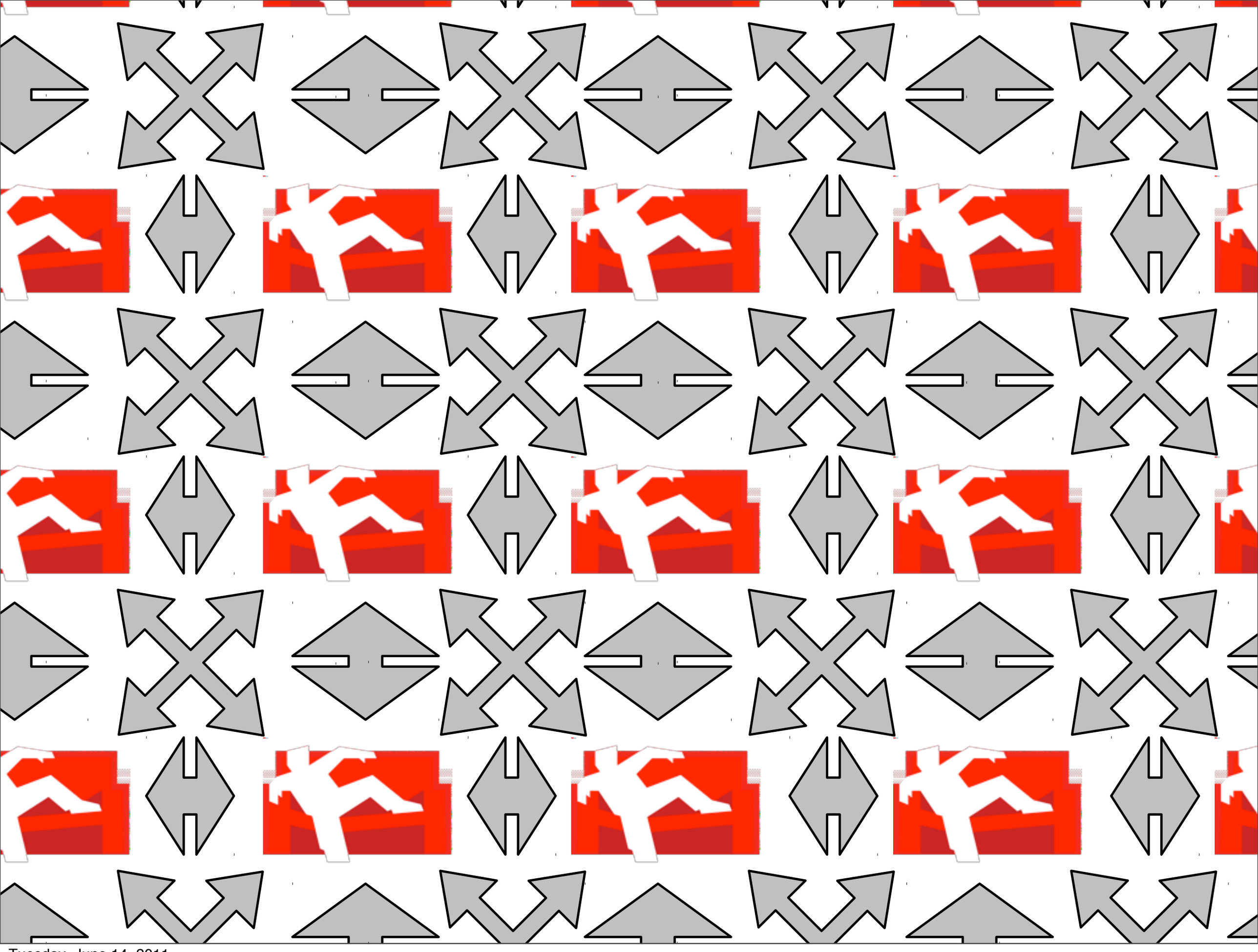
Tuesday, June 14, 2011

- Oriented document Database
- Append-only
- Erlang...With some C
- Multi-platform (desktop, phones, embedded hardware)

CouchDB

- M/R (incremental)
- Attachments
- Replication
- BigCouch, couchdb-lounge
- Couchapps

CouchDB



Tuesday, June 14, 2011

- M/R (incremental)
- Attachments
- Replication
- BigCouch, couchdb-lounge
- Couchapps

CouchDB

- HTTP interface
- Restish...
- Status codes & error handling
- JSON (utf8, bignum handling...)
- M/R in js by default.

API particularities

- Internally uses: ejson, ibrowse, erlang-oauth
- rebar
- OTP application

couchbeam

```
Host = "localhost",
Port = 5984,
Prefix = "",
Options = [],
```

```
S = couchbeam:server_connection(Host, Port,
                                Prefix, Options).
{ok, _Version} = couchbeam:server_info(S).
```

hello world

```
Options = [],
{ok, Db} = couchbeam:open_db(Server, "testdb", Options).
```

Open a DB

```
{ok, Doc2} = couchbeam:open_doc(Db, "test").
```

Open a doc

```
Doc = {[
   {<<"_id">>, <<"test">>},
   {<<"content">>, <<"some text">>}
]}.
{ok, Doc1} = couchbeam:save_doc(Db, Doc).
```

Save a doc

```
{[{<<"_id">>,<<"test">>},
  {<<"content">>,<<"some text">>},
  {<<"_rev">>,<<"1-7e08a50107b1771b20c3816d1eec55d7">>}]}
```

Save a doc

```
[{<<"_id">>,<<"test">>},
 {<<"content">>,<<"some text">>},
 {<<"_rev">>,<<"1-7e08a50107b1771b20c3816d1eec55d7">>}]}
```

Save a doc

```
Options = [{include_docs, "true"}],
{ok, AllDocs} = couchbeam:all_docs(Db, Options).
```

Get all docs

```
{ok,{[{<<"total_rows">>,4},
      {<<"offset">>,0},
      {<<"rows">>,
       [{[{<<"id">>,<<"7a0ce91d0d0c5e5b51e904d1ee3266a3">>},
          {<<"key">>,<<"7a0ce91d0d0c5e5b51e904d1ee3266a3">>},
          {<<"value">>,
{[{<<"rev">>,<<"15-15c0b3c4efa74f9a80d28ac040f18bdb">>}]}},
          {<<"doc">>,
           {[{<<"_id">>,<<"7a0ce91d0d0c5e5b51e904d1ee3266a3">>},
{<<"_rev">>,<<"15-15c0b3c4efa74f9a80d28ac040f18"...>>}]}}]},
       ]}]}}.
```

Example of result

```
Options = [],
DesignNam = "designname",
ViewName = "viewname",
{ok, View} = couchbeam:view(Db, {DesignNam, ViewName}, Options).
```

Get views

- fold
- foreach

Get views results

- Attachments handling
- inline or standalone
- streaming
- doc helpers (get/ add values)

Other stuffs

- Get all changes
- but also handle Longpolling & Continuous modes
- Register a change consumer
- gen_changes behaviour

_changes

```
ChangesFun = fun(ReqId, F) ->
    receive
        {ReqId, done} ->
            ok;
        {ReqId, {change, Change}} ->
            io:format("change ~p ~n", [Change]),
            F(ReqId, F);
        {ReqId, {error, E}}->
            io:format("error ? ~p ~n", [E])
    end
end,
Pid = self(),
Options = [{heartbeat, "true"}],
{ok, ReqId} = couchbeam:changes_wait(Db, Pid, Options),
ChangesFun(ReqId, ChangesFun).
```

subscribe

```
ChangesFun = fun(ReqId, F) ->
    receive
        {ReqId, done} ->
            ok;
        {ReqId, {change, Change}} ->
            io:format("change ~p ~n", [Change]),
            F(ReqId, F);
        {ReqId, {error, E}}->
            io:format("error ? ~p ~n", [E])
    end
end,
Pid = self(),
Options = [{heartbeat, "true"}],
{ok, ReqId} = couchbeam:changes_wait(Db, Pid, Options),
ChangesFun(ReqId, ChangesFun).
```

subscribe

```
ChangesFun = fun(ReqId, F) ->
    receive
        {ReqId, done} ->
            ok;
        {ReqId, {change, Change}} ->
            io:format("change ~p ~n", [Change]),
            F(ReqId, F);
        {ReqId, {error, E}}->
            io:format("error ? ~p ~n", [E])
    end
end,
Pid = self(),
Options = [{heartbeat, "true"}],
{ok, ReqId} = couchbeam:changes_wait(Db, Pid, Options),
ChangesFun(ReqId, ChangesFun).
```

subscribe

```
ChangesFun = fun(ReqId, F) ->
    receive
        {ReqId, done} ->
            ok;
        {ReqId, {change, Change}} ->
            io:format("change ~p ~n", [Change]),
            F(ReqId, F);
        {ReqId, {error, E}}->
            io:format("error ? ~p ~n", [E])
    end
end,
Pid = self(),
Options = [{heartbeat, "true"}],
{ok, ReqId} = couchbeam:changes_wait(Db, Pid, Options),
ChangesFun(ReqId, ChangesFun).
```

subscribe

```
-module(test_gen_changes).

-behaviour(gen_changes).
-export([start_link/2,
         init/1,
         handle_change/2,
         handle_call/3,
         handle_cast/2,
         handle_info/2,
         terminate/2]).

-export([get_changes/1]).

-record(state, {changes=[]}).

start_link(Db, Opts) ->
    gen_changes:start_link(?MODULE, Db, Opts, []).
```

gen_changes

```
-module(test_gen_changes).

-behaviour(gen_changes).
-export([start_link/2,
         init/1,
         handle_change/2,
         handle_call/3,
         handle_cast/2,
         handle_info/2,
         terminate/2]).

-export([get_changes/1]).

-record(state, {changes=[]}).

start_link(Db, Opts) ->
    gen_changes:start_link(?MODULE, Db, Opts, []).
```

gen_changes

```
init([]) ->
    {ok, #state{}}.

get_changes(Pid) ->
    gen_changes:call(Pid, get_changes).

handle_change(Change, State=#state{changes=Changes}) ->
    NewChanges = [Change|Changes],
    {noreply, State#state{changes=NewChanges}}.

handle_call(get_changes, _From, State=#state{changes=Changes}) ->
    {reply, Changes, State}.

handle_cast(_Msg, State) -> {noreply, State}.

handle_info(Info, State) ->
    io:format("Unknown message: ~p~n", [Info]),
    {noreply, State}.

terminate(Reason, _State) ->
    io:format("~p terminating with reason ~p~n", [?MODULE, Reason]),
    ok.
```

gen_changes

```
init([]) ->
    {ok, #state{}}.

get_changes(Pid) ->
    gen_changes:call(Pid, get_changes).

handle_change(Change, State=#state{changes=Changes}) ->
    NewChanges = [Change|Changes],
    {noreply, State#state{changes=NewChanges}}.

handle_call(get_changes, _From, State=#state{changes=Changes}) ->
    {reply, Changes, State}.

handle_cast(_Msg, State) -> {noreply, State}.

handle_info(Info, State) ->
    io:format("Unknown message: ~p~n", [Info]),
    {noreply, State}.

terminate(Reason, _State) ->
    io:format("~p terminating with reason ~p~n", [?MODULE, Reason]),
    ok.
```

gen_changes

- Erlang is different.
- Don’t overuse gen_servers
- Instead of throwing errors, return them
- Don’t try to cache
- boolean(), nifs, ... -> support different erlang ‘s versions

Lessons learnt

- Parallelize connections, manage workers
- Multipart handling
- _show, _list, _update ...
- RPC client?

Future

- couchapp in erlang
- from Python to Erlang
- writing shell app in Erlang
- pattern matching fit well

erica

Tuesday, June 14, 2011

- push, clone commands
- Template management
- hooks...

erica

```
$ erica create-app appid=myapp lang=javascript
==> tmp (create-app)
Writing myapp/_id
Writing myapp/language
Writing myapp/.couchapprc

$ ls -fla myapp/
total 24
drwxr-xr-x   6 benoitc  wheel  204 Jun  7 11:13 .
drwxrwxrwt  13 root     wheel  442 Jun  7 11:13 ..
-rw-r--r--   1 benoitc  wheel   18 Jun  7 11:13 .couchapprc
drwxr-xr-x   2 benoitc  wheel   68 Jun  7 11:13 _attachments
-rw-r--r--   1 benoitc  wheel   13 Jun  7 11:13 _id
-rw-r--r--   1 benoitc  wheel   10 Jun  7 11:13 language
```

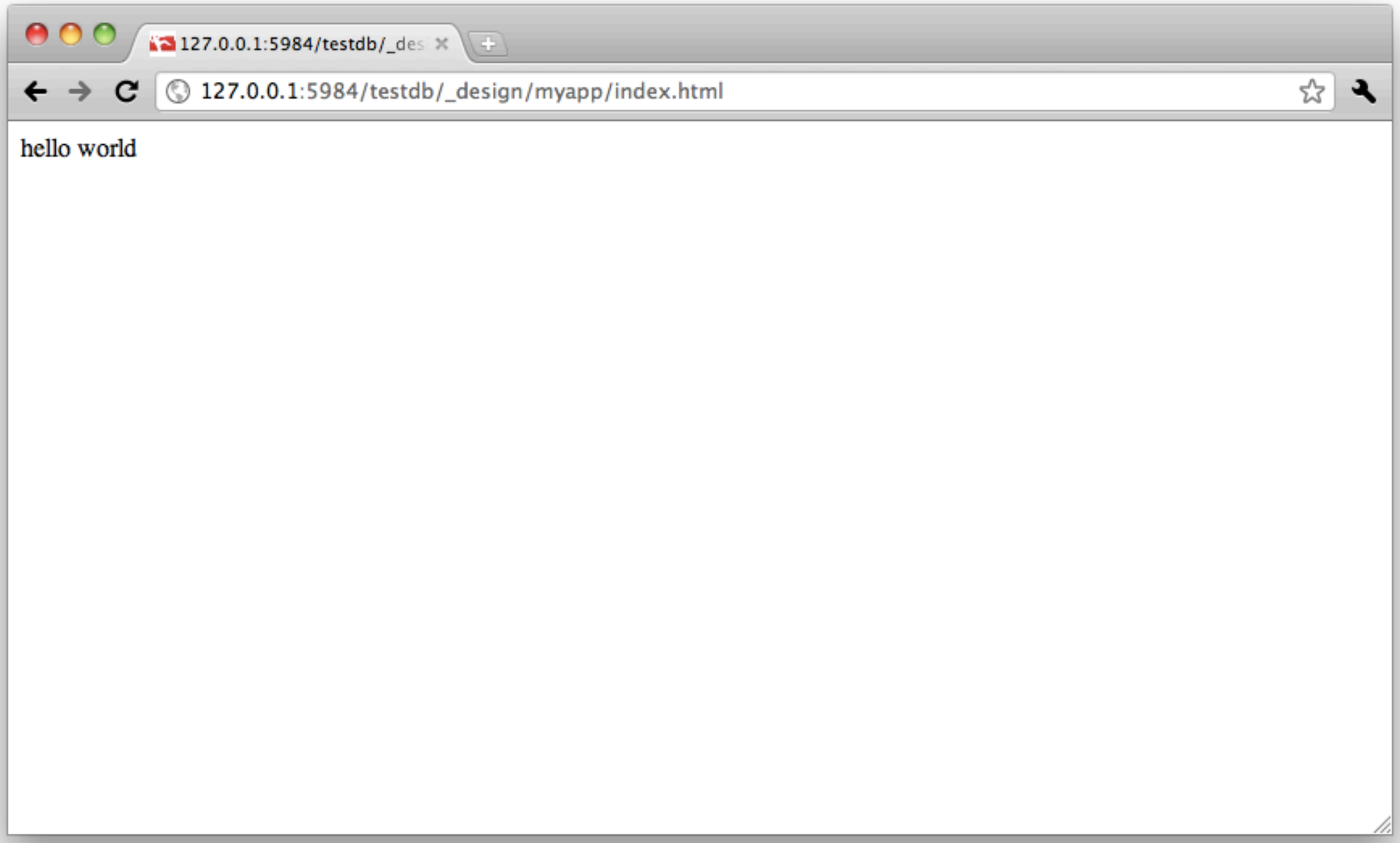
erica

```
$ cd myapp
$ echo "hello world" > _attachments/index.html
$ erica push http://127.0.0.1:5984/testdb
```

erica

```
cd myapp
echo "hello world" > _attachments/index.html
erica push http://127.0.0.1:5984/testdb
```

erica


127.0.0.1:5984/testdb/_des
127.0.0.1:5984/testdb/_design/myapp/index.html
hello world

- bootstrap to create an exe
- commands are functions / modules
- resources integrated in the exe
- Miss a way to create a full exe not depending on Erlang

erica

- http://github.com/benoitc/couchbeam
- http://github.com/benoitc/erica
- @benoitc

Thanks!

?