umleitung

an intro to mochiweb and CouchDB

mochiweb

- get mochiweb
- `svn checkout http://mochiweb.googlecode.com/svn/trunk/ mochiweb`
- create new application
- `mochiweb/scripts/new_mochiweb.erl umleitung`

run mochiweb

- make the project and run it
- `cd umleitung`
- `make`
- `./start-dev.sh`
- go to http://localhost:8000

directory structure

- we have our sources in /src
- we have our compiled binaries in /ebin
- the only file we need in the moment is:

  src/umleitung_web.erl

- open it in your editor

act on GET/HEAD

```
Method when Method =:= 'GET'; Method =:= 'HEAD' ->
  case lookup(Path) of
     {ok, Dest} ->
        Req:respond({200, [], Dest});
     _ ->
        Req:respond({501, [], "error"})
  end;
```

the lookup/1

```
%% Internal API

lookup(Path) ->
    io:format("Path is: ~s~n", [Path]),
    {ok, Path}.
...
```

testing

- recompile and see that the running project is reloaded
  - to recompile just run make
- request http://localhost:8000/test
- look at the console output as well

CouchDB

- get CouchDB from http://couchdb.apache.org
- best compile from source
- http://wiki.apache.org/couchdb/Installing_from_source
- run couchdb (sudo couchdb)
- start futon on http://localhost:5984/_utils

create DB

- click on “create database”
- name it “umleitung”
- create a document:

    type: redir

    path: test

    destination: http://ideegeo.com

create view

- select the view: Custom query...
- in the map function add:

```
function(doc) {
  if(doc.type == 'redir'){
    emit(doc.path, doc.destination);
  }
}
```

create view 2

- choose save as
- name the design document: _design/redir
- and the view: match

what we have

- we have a mochiweb server interpreting our path
- we have a couchdb that can save path:destination tuples
- ... lets connect them

erlang_couchdb

- based on mochiweb
- slim library, small foot print
- `git clone git://github.com/ngerakines/erlang_couchdb.git`
- `cd erlang_couchdb`
- `make dist-src`
- and link it to /deps (ext. dependencies)

the view request

- erlang_couchdb:invoke_view({"HOST", PORT}, "DB", "DESIGN DOC", "VIEW", [{"key", "\"" ++ REQUEST KEY ++ "\""}]),
- our request key is the “Path” variable that holds the path we want to match against
- note the enclosing ‘"’

major tuple hacking

- the erlang_couchdb lib is small, that means less comfort
- the data structure is deep nested json tuples
- the view request looks like this:

```
{json,{struct, Props}} =
erlang_couchdb:invoke_view( {"localhost", 5984},
"umleitung", "redir", "match", [{"key", "\"" ++ Path
++ "\""}]),
```

fiddle it together

```
lookup(Path) ->
    io:format("PATH: ~s~n", [Path]),
   {json,{struct, Props}} =
      erlang_couchdb:invoke_view({"localhost",
      5984}, "umleitung", "redir", "match",
      [{"key", "\"" ++ Path ++ "\""}]),
    try  proplists:get_value(<<"rows">>, Props) of
      [{struct, Rows} | _] ->
        {ok, proplists:get_value(<<"value">>, Rows)};
      _ -> {error, unknown}
    catch
        _:_ -> {error, unknown}
    end.
```

testing

- recompile the project again
- go to http://localhost:8000/test
- then go to http://localhost:8000/bla
- we have connected CouchDB to our mochiweb server
- ... relax ...

do the redirect

- the final step is doing the redirect now
- change the response line to:

```
Req:respond({302, [{"Location", Dest}], ""});
```

- recompile
- reload the http://localhost:8000/test

cosmetics

- getting rid of that ugly compile warning and serving a static file

```
case lookup(Path) of
    {ok, Dest} ->
        Req:respond({302, [{"Location", Dest}], ""});
     _ ->
        Req:serve_file("404.html", DocRoot)
end;
```

- add a 404.html to priv/www/

and now?

- sources are on github
- git clone git://github.com/norbu09/umleitung.git
- go out and play, extend, rewrite it
- add a web interface (based on nitrogen?)

hope you enjoyed it

lenz@ideegeo.com

http://github.com/norbu09/umleitung/tree