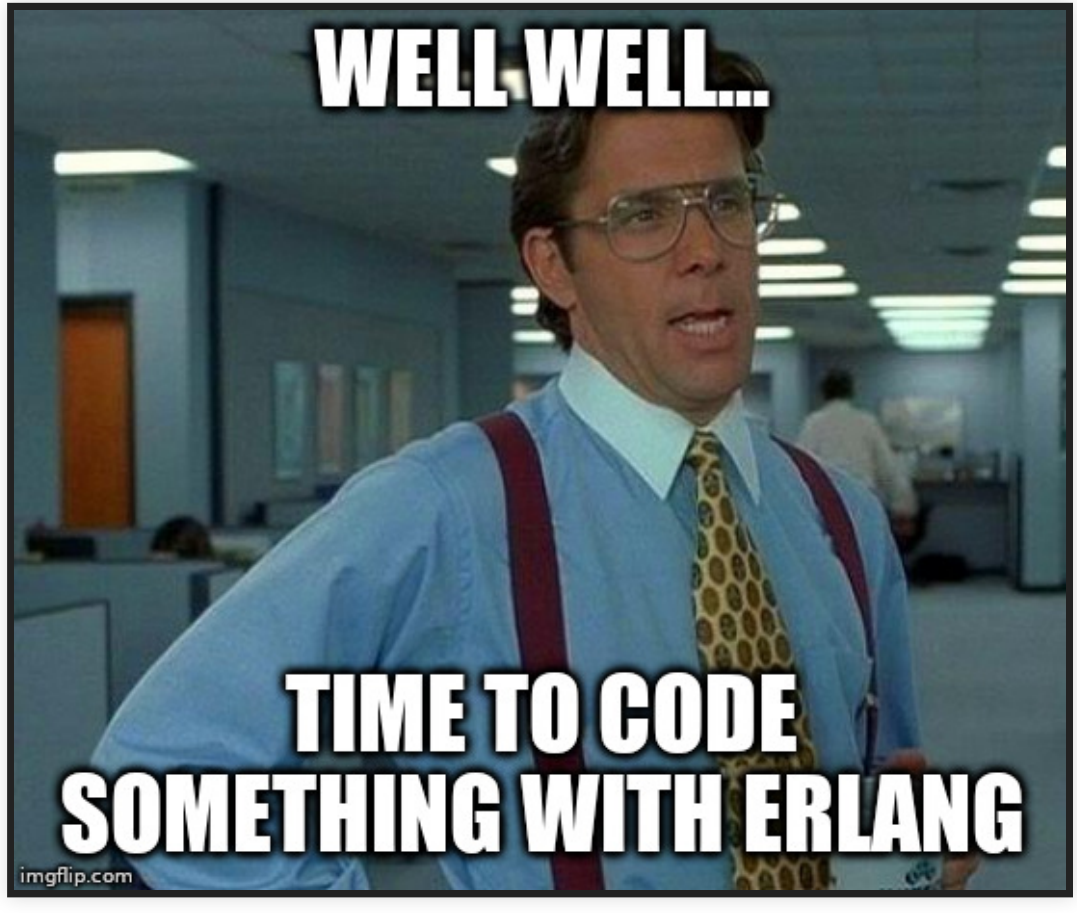
REAL WORLD ERLANG PROGRAMMING

Mathieu Kerjouan | @niamtokik


WELL WELL...
TIME TO CODE
SOMETHING WITH ERLANG
imgflip.com

WHAT WE NEED?

- A shell (CLI FTW)
- Text editor or IDE (e.g. emacs)
- Erlang release (R22 is the last one)
- Rebar3 or erlang.mk (to manage the project)

WORKSPACE INITIALIZATION

```
# create storage directory
mkdir ${HOME}/src
mkdir ${HOME}/bin
mkdir ${HOME}/work

# add ${HOME}/bin in your path
export PATH=${PATH}:${HOME}/bin
```

ERLANG/OTP LOCAL BOOTSTRAP

```
cd ${HOME}/src
git clone https://github.com/erlang/otp.git
cd otp
./otp_build release -a

# links the different executable in ${HOME}/bin
for exe in erl erlc escript ct_run dialyzer type;
do ln -s $(pwd)/bin/${exe} ${HOME}/bin; done
```

REBAR3 LOCAL BOOTSTRAP

```
cd ${HOME}/src
git clone https://github.com/erlang/rebar3
cd rebar3
./bootstrap

# link rebar3 in ${HOME}/bin
ln -s $(pwd)/rebar3 ${HOME}/bin
```

WHOIS SERVER PROTOCOL: ERLANG IMPLEMENTATION

- open specification RFC3912
- small requirement (~4pages)
- network oriented (TCP connection)
- client available (whois command)
- future ports requirement (TCP port <1024)

WHAT IS WHOIS?

WHOIS is a TCP-based transaction-oriented query/response protocol that is widely used to provide information services to Internet users.

HOW IT WORKS?

```
$ whois ddg.gg
Domain:
ddg.gg

Domain Status:
Active
Transfer Prohibited by Registrar

Registrant:
Duck Duck Go, Inc.
```

HOW IT WORKS (ALTERNATIVE)?

```
$ echo crsnic.net | nc whois.crsnic.net 43
Domain Name: CRSNIC.NET
Registry Domain ID: 4611887_DOMAIN_NET-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: http://www.cscglobal.com/global/web/csc/digital
```

CREATE OUR APPLICATION

```
cd ${HOME}/work
rebar3 new app name=whoisd
cd whoisd

ls LICENSE README.md
ls rebar.conf rebar.lock
ls src/whoisd.app.src
ls src/whoisd_app.erl
ls src/whoisd_sup.erl
```

`LICENSE` and `README.md`

- de facto standard
- project license
- README containing usage

`rebar.config`

- Erlang Term Format
- information about the project
- dependencies
- license
- release...

`rebar.lock`

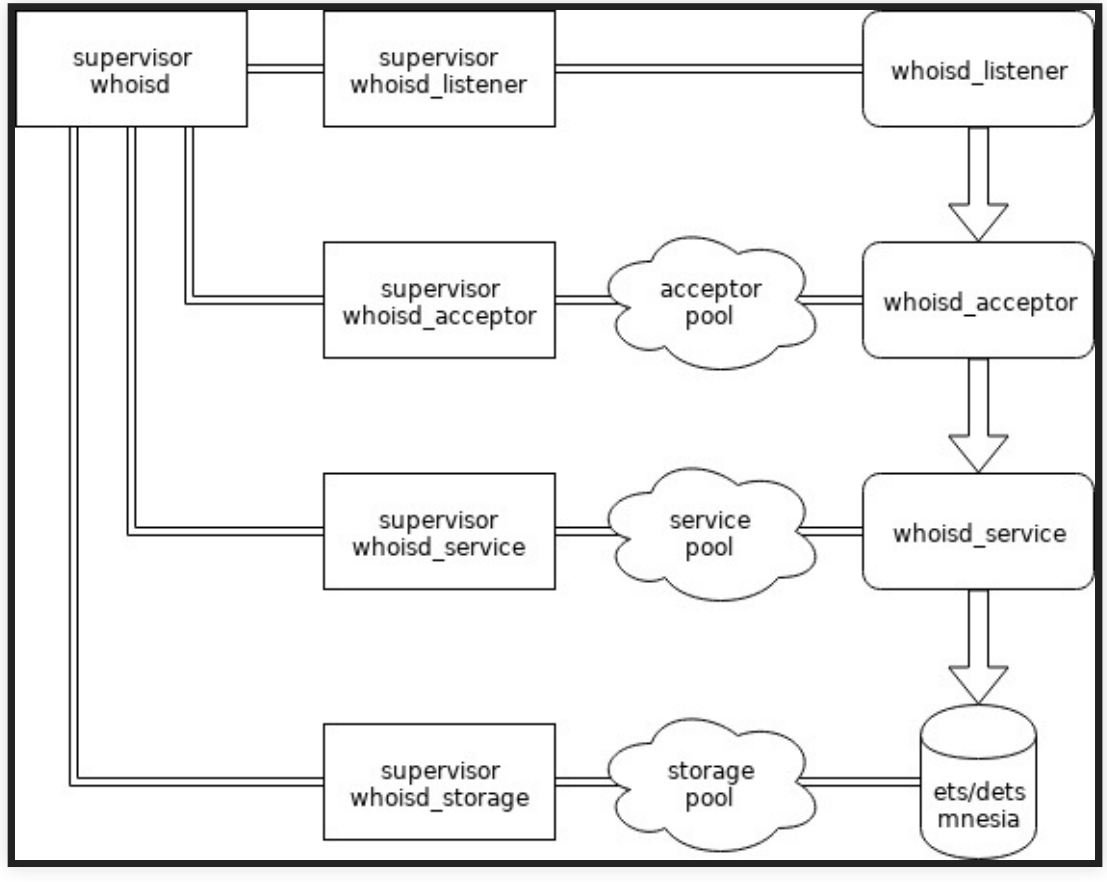
- Erlang Term Format
- information about the dependencies

`src/whoisd.app.src`

- OTP application resource file
- description
- service dependencies
- version
- start phases...

`src/*.erl` and `src/*.hrl`

- contain Erlang code
- contain Erlang headers


supervisor
whoisd
supervisor
whoisd_listener
whoisd_listener
supervisor
whoisd_acceptor
acceptor
pool
whoisd_acceptor
supervisor
whoisd_service
service
pool
whoisd_service
supervisor
whoisd_storage
storage
pool
ets/dets
mnesia

WHOISD APPLICATION

- located in `src/whoisd_app.erl`
- use `application` behaviour
- ensure the service is correctly started

TODO-A01

Add a way to create process group when the application start by using pg2 module. The file to modify is `src/whoisd_app.erl`.

TODO-A02

Add a way to start the main `application supervisor` process named `whoisd_sup`. This file is stored in `src/whoisd_sup.erl`.

TODO-A03

Add a way to clean the application and remove different group from pg2.

WHOISD APPLICATION SUPERVISOR

- located in `src/whoisd_sup.erl`
- use `supervisor` behaviour
- first supervisor managing supervisor children.

TODO-SA1

Find a way to start these children:

- `whoisd_listener_sup`
- `whoisd_acceptor_sup`
- `whoisd_storage_sup`
- `whoisd_service_sup`
- `whoisd_manager_sup`

all as children `supervisor`. You can use the different helpers defined in this module.

LISTENER SUPERVISOR

- Located in `src/listener_sup.erl`
- manage `whoisd_listener` process

TODO-LS1

Find a way to configure this supervisor with a one_for_one strategy. you can read `supervisor` documentation.

TODO-LS2

Find a way to start the main worker child named `whoisd_listener`. you can use the internal function in this module.

LISTENER

- located in `src/whoisd_listener.erl`
- manage the listen filedescriptor created by `gen_tcp:listen/2`

TODO-L01

Find a way to automatically add this process in whoisd_listener pg2 group.

TODO-L02

find a way to configure default listen port and options. whoisd_listener need a port to listen and probably different options. These options can be retrieved from different part of the code. The first one is directly from the `Args` parameter, another one is by using application environment variables.

TODO-L03

Find a way to close the socket managed by whoisd_listener. You can read `gen_tcp` module man page.

TODO-L04

Find a way to create an API function to retrieve socket.

ACCEPTOR

- located in `src/whoisd_acceptor.erl`
- manage the accept filedescript created by `gen_tcp:accept/2`
- manage the user connection by receive tcp packets

TODO-A01

like whoisd_listener, find a way to add this process in whoisd_listener pg2 group.

TODO-A02

find a way to get the listener socket and use it to enable the acceptor process. There lot of methods possible and not good answer.

TODO-A03

when a message is coming in the mailbox, you will need to do something. find a way to print the message on the screen and reply something to the client.

TODO-A04

WHOIS protocol must close the socket after sending the answer to the client. Find a way to alert the client that you dont have new data. hints look `gen_tcp:shutdown/2`

MANAGER

- located in `src/whoisd_manager.erl`
- general API to retrieve information
- general API to start/stop services
- good way to synchronise actions between nodes

TODO-M01

Find a way to retrieve information about the different acceptors

TODO-M02

Find a way to retrieve information about the different established connections

TODO-M03

Find a way to spawn dynamically new acceptors

SERVICE

- located in `src/whoisd_service.erl`
- send the data to the client

TODO-SE1

this is your turn! create a full whoisd_service process, this one will react to different message sent by the clients. You will use `gen_server` behaviour.

STORAGE

- located in `src/whoisd_storage.erl`
- route the request to the content
- use different method to connect to the data

TODO-ST1

find a way to store local information about different domain name. you can use different solution there, Erlang can give you `ETS`/`DETS` but also `Mnesia`. You can also use some external libraries and store information in classical database like MySQL or PostgreSQL.

CLIENT

- use escript
- create a tcp client connection
- support standard CLI flags

TODO-C01

This is the main function of the whois client. You can pass your argument there. Arguments supported by whois command:

whois [-AadgIilmPQRr] [-c country-code | -h host] [-p port] name ...

Find a way to implement some of them by using only Erlang internal functions.

WHOISD

- located in `src/whoisd.erl`
- entry point for service daemon from CLI

TODO-E01

find a way to start the application as deaemon

TODO

- create unit testing (sorry about that)
- create common testing
- deploy it
- add heartbeat...

This code is available on github.

Questions?

RESOURCES

- Official Erlang documentation
- Rebar3