ERL#1 - Erlang - Apprendre, Comprendre, Débuter.

Mathieu Kerjouan <contact@steepath.eu>

But de l’atelier

Cette présentation fait suite au premier Meetup Erlang rennais. Elle a pour but d’appronfondir ce que nous avons déjà pu voir et permettre de commencer à développer avec Erlang. Nous verrons, dans les grandes lignes:

- Présentation rapide d’Erlang
- Présentation de la programmation fonctionelle
- Présentation de la programmation concurrentielle
- Première prise en main
- Dédiabolisons l'échec!
- Just Do It! NOW!

Avant de commencer - Pourquoi Erlang?

- Erlang est éprouvé, testé et utilisé depuis plus de 30ans.
- Erlang est une réponse à des problématiques complexes (concurrence/clustering/gestion des erreurs/haute dispo).
- Erlang est fournit avec OTP, un regroupement de logiciel/librairie et répondant aux besoins des développeurs.
- Erlang utilise une syntaxe atypique (issue de Prolog) mais est très bien adapté aux tâches complexes.
- Erlang est libre et open-source.

Présentation rapide d’Erlang

- La société Ericsson recherche une solution pour la stabilité et productivité offrant des fonctions concurrentielles et un haut niveau d’abstraction. Le programme de recherche commence en 1981 dans le CSLab.

> suggest new architectures, concepts and structures for future processing systems developments.

- Création du langage en 1987.
- Création d'un PABX complet codé en Erlang en 1988.

> Hello Mike, I've got Robert on the other end here, would you give me a moment? Thanks
>
> *Erlang The Movie*
> — Joe Armstrong

- Création de la machine virtuelle JAM implémenté en C, en 1989, basé sur la machine virtuelle de prolog.

Présentation rapide d'Erlang (suite)

- Ericsson crée la société Erlang System AB pour vendre Erlang à des clients externes.
- Création d'OTP en 1996
- Ericsson Radio banni Erlang en 1998
- Erlang/OTP est mis en open source la même année
- La société Bluetail AB est créé pour promouvoir un internet plus fiable.
- Grosses réussites dans de nombreux domaines depuis…
  - Software (Chef, CouchDB, RabbitMQ, ejabberd)
  - Entreprises (Amazon, Ericsson, Facebook, Github, Rackspace, Whasapp, Yahoo…)

> And then I came across Erlang — "What is this Erlang thing" and it was the first time I'd heard of it and so I began to research. It turned out to be the best engineering decisions we ever made.
>
> — Jan Koum

Présentation de la programmation fonctionnelle

- Offre une abstraction logiciel de théorie mathématiques (théorie des groupes, lambda calculus).
- Base de l'informatique, la machine de turing est basé sur ces concepts.

- Qu’est-ce que la programmation fonctionnelle?
  - une assurance pour le controle d'état
  - une meilleure structuration logique du code
  - une hiérarchie des types
  - l’utilisation de la récursion en remplacement des boucles
  - une fonction aura un retour identique, si l’entrée est identique
  - référence transparente: une variable ne peux changer de valeur
  - pour résumer: tout est fonction

> La récurrence est humaine, la récursivité est d’essence divine.
>
> — inconnue

Présentation de la programmation concurrente

- Abstraction d’un modèle extrèmement complexe
- Coordinations sur des ressources partagées
- Simplification de la communication entre les utilisateurs de ressources (mémoire partagées et/ou passage de message)
- Qu’est-ce que la programmation concurrentielle?
  - comment répartir une charge de travail
  - comment communiquer entre des processus

> Everybody who learns concurrency thinks they understand it, ends up finding mysterious races they thought weren’t possible, and discovers that they didn’t actually understand it yet after all.
>
> — Herb Sutter

Erlang dans tout ça?

- Erlang est un langage fonctionnel
  - il est fortement typé
  - il utilise la récursion et non les boucles

    - les listes sont ses structures de données de prédilection
    - utilisation des pattern-matching
- Erlang est un langage concurrentiel
    - il fait abstraction des problèmes de parallélisme
    - il est tolérent aux erreurs
    - il utilise un système de micro-processus avec stack dédié
    - il utilise un système de mailbox pour chaque processus
    - chaque noeud est raccordé entre eux (mesh)
    - standardisation des échanges entre noeuds

> The world **is** parallel – we are parallel.
>
> — Joe Armstrong

Première prise en main - installation

Debian/Ubuntu

```
apt-get install erlang
```

RHEL/CentOS

```
yum install erlang
```

OpenBSD

```
pkg_add erlang
```

FreeBSD

```
pkg install erlang
```

Première prise en main - installation (suite)

NetBSD

```
cd /usr/pkgsrc/lang/erlang && make && make install
```

Mac

```
brew install erlang
port install erlang
```

Windows

```
https://www.erlang.org/downloads
```

Compilation depuis les sources

```
git clone https://github.com/erlang/otp.git
cd otp
./otp-build
make
make release_tests
make install # en root
```

Première prise en main - erl

- `erl` est la commande permettant d'avoir accès au shell interactif d'erlang.
- ce qu'il peut faire:
  - permet d'avoir un accès direct au code
  - permet de tester des solutions avant intégration
  - utilisable comme bac à sable
- ce qu'il ne peut pas faire:
  - création de modules à la volée

- ne peux pas tout débugger et tester

Lancement du shell interactif erlang

```
erl
```

Première prise en main - codons un peu! (1)

Assignement de variables

```
MyInt    = 1.                % entier
MyFloat  = 1.234.            % nombre flottant
MyAtom   = test.             % atom
MyList   = [1, 2, 3].        % liste
MyTuple  = {1, 2, 3}.        % tuple
MyString = "test".           % équivalent à [$t, $e, $s, $t]
MyMap    = #{ key => value }. % map
MyBin    = <<"thisisabin">>.  % iolist/binary
MyPropList = [{key, "value"}] % liste de tuple aka proplist
```

Une variable ne peut-être attribué qu'une fois

```
MyInt = 2. % génère une exception:
           % exception error: no match of right hand side value
```

structure de donnée spécifique, records

```
% les records permettent de créer des structures de données
% pratiquement identiques aux maps. Pour créer un record dans le shell
% il est nécessaire d'utiliser rd(), rf() et rl().
rd(myrecord, {key = value}).
rl(myrecord).
```

Première prise en main - codons un peu! (2)

Utilisation de fonction intégrée (Built-in Function aka BIF)

```
date().          % équivalent à erlang:date().
time().          % équivalent à erlang:time().
now().           % équivalent à erlang:now().
erlang:'+'(1, 2) % équivalent à 1+2.
```

Utilisation de fonction externe (modules)

```
% fonction liée aux listes
lists:last([1,2,3]).
lists:reverse("tuorp").
lists:seq(1,3).

% fonction liée aux maps
maps:new(). % retourne #{}
maps:find(test, #{test => toto}). % retourne {ok, toto}
maps:put(key, "value", #{}). % retourne #{key => "value}"

% fonction liée au système
os:type().
os:cmd(ls). % équivalent à os:cmd("ls").
```

Création de fonctions anonymes (lambda)

```
MyFunc = fun(X,Y) -> X + Y end.

MyPattern = fun (toto) -> "je ne te parle pas!";
                (N) when is_atom(N) -> "bonjour " ++ atom_to_list(N);
                (N) when is_list(N) -> "bonjour " ++ N; (_) -> error
                end.

MyLoop = fun F([]) -> ok; F([H|T]) -> io:format("My head is ~p~n",
             [H]), F(T) end.
```

Première prise en main - codons un peu! (3)

Pattern Matching - routage des arguments

```
PatternMatching = fun(1) -> 2;
                      (3) -> 4;
```

```
                    (test) -> ok;
                    (X) -> X end.
```

Guards - protection des arguments

```
Guards = fun(X) when is_list(X) -> list;
           (X) when is_atom(X) -> atom;
           (X) when is_integer(X) -> integer;
           (X) -> other end.
```

Case - switching de fonction en fonction de l'entrée

```
Case = fun(X) ->
       case X of
         test -> io:format("test~n");
         _Else -> io:format("not found ~p~n", [_Else])
       end end.
```

If - guard sans protection d'arguments

```
If = fun(X, Y) ->
     if X >= Y -> X;
        true -> Y end end.
```

Catch/Catch - gestion des exceptions

```
TryCatch = doh.
```

List Comprehensions - Sucre syntaxique de manipulation de liste

```
MyList = [ X || X <- lists:seq(1,3) ].
MyList2 = [ X*2 || X <- lists:seq(1,3) ].
MyList3 = [ <<X>> || X <- lists:seq(1,100), X rem 4 =:= 0 ].
```

Receive/Send - pattern matching sur la mailbox

```
% domaine spécifique, nous voyons ça après! :)
```

Première prise en main - codons un peu! (4)

le shell plus en profondeur

```
help().  % imprime l'aide du shell (shell_default:help()).
b().  % retourne la liste des variables enregistrées

f().  % supprime toutes les variables enregistrées

f(MyVariable). % supprime la variable MyVariable

h().  % affiche l'historique

e(N).  % re-exécute lacommande "N". N est à trouver dans l'historique.

c(File).  % compile un fichier

lc([File]).  % compile la liste de module

m(Module).  % affiche les informations concernant le module Module

memory().  % affiche l'utilisation mémoire

pwd().  % retourne le répertoire courant

pid(X,Y,Z).  % retourne un type PID <0.88.0>
```

Le shell est un processus, comme tout ce qui se trouve dans Erlang.

Les concepts d'Erlang en action

micro-process

```
MyProcess = fun F() -> receive
  {name, Name} -> io:format("Bonjour ~p!~n", [Name]), F();
  {throw, MyThrow} -> throw(MyThrow), F();
  {error,MyError} -> error(MyError);
  _Else -> io:format("~p~n", [_Else]), F()
```

```
  end
end.
```

spawning

```
MyPid = spawn(MyProcess).
```

communicating

```
MyPid ! toto.  MyPid ! {name, toto}.
```

monitoring

```
MyPidMonitor = spawn_monitor(MyProcess).
flush().
```

linking

```
MyPidLink = spawn_link(MyProcess).
```

Et si on sortait du shell?

création d'un exécutable erlang

```
touch main.erl
```

le code

```
%%%-------------------------------------------------------------------
%%% @author Nom de l'auteur
%%% @copyright Le copyright
%%% @version 0.0.1
%%% @title Le titre de la documentation ou de l'application
%%% @doc
%%%     Le contenu de la documentation
%%% @end
%%% @todo une simple todo-list
```

```
%%%----------------------------------------------------------------
% nom du module
-module(main).

% export des fonctions désirées
(publique) % -export([bonjour/1, start/0]).

% option de compilation
-compile([export_all]).

% librairie de test eunit pour les assertions
-include_lib("eunit/include/eunit.hrl").

%%----------------------------------------------------------------
%% @doc
%%      Contenu de la documentation pour la fonction bonjour.
%% @end
%%----------------------------------------------------------------
% définition des specifications, permet de savoir quelle type de
% donnée une fonction accepte et qu'elle sera le type de sortie.
% utiliser par dialyzer
-spec bonjour(list()) -> list().

% définition de la fonction, le code source se trouve ici!
bonjour(Name)
  when is_list(Name) ->
    io_lib:format("Bonjour ~s!", [Name]).

% définition des tests, permet d'avoir des exemples concernant la
% fonction, utilisé par eunit
bonjour_test() ->
  [?assert(bonjour("test") =:= "Bonjour test!")].
```

la compilation via erlc ou directement dans le shell erlang

```
erlc main.erl # compilation via erlc
```

```
c(main). % compilation de main.erl
edoc:file(main, [{dir, "./"}]). % création de la doc (main.html)
```

l'utilisation

```
main:test(). % exécution des tests
main:bonjour("toto"). % exécute la fonction bonjour
```

Dédiabolisons l'échec!

- L'erreur est humaine… L'entêtement est diabolique.
- Il est IMPOSSIBLE aujourd'hui de tout controler.
- Le risque d'erreur est partout.
- N'ayons donc plus peur des erreurs!

> Let it crash!

- La philosophie d'Erlang s'appuie sur le fait que tous les processus peuvent planter. Ils ont donc alors créé le système de supervision qui permet de relander un processus en cas d'erreur.
- OTP (Open Telecom Platform) regroupe de nombreuses fonctionnalités permettant de se simplifier la vie, d'éviter les erreurs et d'offrir une qualité de service importante au travers d'un framework complet intégré à Erlang.

OTP (Open Telecom Platform)

- extension des fonctionnalités d'Erlang
  - standardisation des comportements (behaviour)
  - heartbeat (heart)
- librairie cohérente avec Erlang
  - parser (leex, yecc),
  - cryptographie (crypto, ssl),
  - database (ets, mnesia, odbc)
  - GUI (wx)
- outils de développement

- compiler (HiPE)
- builder (make)
- monitoring (observer, etop)
- test (eunit, common_test)
- analyse (dialyzer, cover, typer)
- debugger (debugger)
- benchmarking (eprof, cprof, fprog)
- documentation (edoc)

Nous verrons ça lors du prochain Meetup! :D

Atelier - Partageons nos ressources!

- synchronisation des versions d'erlang

```
# création d'un utilisateur séparé
adduser erltest
su - erltest

# récupération des sources
git clone --single-branch -bOTP-18.3.3  https://github.com/erlang/otp.git
cd otp
./otp-build setup
```

- création de noeuds

```
./bin/erl -sname $myname -setcookie "ErlangIsJustAwesome"
```

- interconnexion des noeuds et test de connectivité

```
erlang:node().
net_adm:ping("remote_host").
```

- résolution de problème complexe:

```
% nous pouvons donc maitenant envoyer des messages sur les autres
% noeuds
{ 'remotenode', Pid } ! MyMessage.
```

Project Euler

Pour pouvoir concrétiser un peu ce que nous avons vu, et se salir les mains, nous allons faire quelques exercices en provenance du projet euler.

- exemples:
  - 10001st prime
  - Large sum

La prochaine fois

- Présentation en profondeur des modules livrés:
  - ETS
  - Mnesia
  - Inets
- Développement avec OTP.
- Début d'un projet commun. Plusieurs propositions:
  - Jeu de role CLI en réseau (hack/hunt like)
  - Développement d'un site web simple
  - Librairie l'utilisation blockchain
  - Création d'un système de logs décentralisé
  - Création d'un réseau de botnet

Liens

Présentation rapide d'Erlang

- https://en.wikipedia.org/wiki/Erlang_%28programming_language%29
- https://www.cs.umd.edu/class/spring2014/cmsc433-0201/Lectures/erlang-intro.pdf
- https://www.erlang-factory.com/upload/presentations/416/MikeWilliams.pdf
- http://www.josetteorama.com/erlang-the-written-history/
- http://www.erlang-factory.com/upload/presentations/247/erlang_vm_1.pdf
- http://wambook.sourceforge.net/
- https://www.infoq.com/interviews/williams-erlang
- http://genius.com/Jim-goetz-jim-goetz-and-jan-koum-at-startup-school-sv-2014-annotated

Programmation fonctionnelle

- https://en.wikipedia.org/wiki/Functional_programming
- https://www.ibm.com/developerworks/java/library/j-ft20/index.html
- http://www.defmacro.org/ramblings/fp.html
- https://www.youtube.com/watch?v=AOl2y5uW0mA&list=PLtRG9GLtNcHBv4cuh2w1cz5VsgY6adoc3
- https://www.coursera.org/specializations/scala

Programmation concurrentielle

- https://wiki.haskell.org/Parallelism_vs._Concurrency
- https://en.wikipedia.org/wiki/Concurrent_computing
- http://www.enseignement.polytechnique.fr/informatique/INF431/X09-2010-2011/AmphiTHC/SynchronizationPrimitives.pdf
- http://www.faculty.idc.ac.il/gadi/book.htm
- https://www.coursera.org/learn/parprog1

Prise en main

- installation
    - http://erlang.org/doc/installation_guide/users_guide.html
    - https://www.erlang.org/downloads

    - https://github.com/erlang/otp/
    - https://github.com/erlang/otp/blob/maint/HOWTO/INSTALL.md
    - https://www.freshports.org/lang/erlang
    - http://openports.se/lang/erlang
    - http://pkgsrc.se/lang/erlang
    - https://packages.debian.org/search?keywords=erlang
    - http://packages.ubuntu.com/precise/erlang
- variables
    - http://erlang.org/doc/reference_manual/data_types.html
- fonctions
    - http://erlang.org/doc/reference_manual/functions.html
    - http://erlang.org/doc/reference_manual/typespec.html
    - http://erlang.org/doc/apps/
- shell
    - http://erlang.org/doc/man/shell.html
    - http://erlang.org/doc/man/shell_default.html
    - http://erlang.org/doc/man/erl.html

Concepts d'Erlang

- behaviour
    - http://erlang.org/doc/design_principles/des_princ.html
- supervisor
    - http://erlang.org/doc/design_principles/sup_princ.html#spec
    - http://erlang.org/doc/man/supervisor.html
- events
    - http://erlang.org/doc/design_principles/events.html
- finite state machine
    - http://erlang.org/doc/design_principles/fsm.html

- server
    - http://erlang.org/doc/design_principles/gen_server_concepts.html

Autre

- http://www.thinkingparallel.com/2007/03/20/ten-questions-with-joe-armstrong-about-parallel-programming-and-erlang/

Bibliographie

- Learn you some Erlang for great good!
- Concurrent Programming in Erlang
- Erlang Programming
- Erlang and OTP in Action
- Introducing Erlang
- Designing for Scalability with Erlang/OTP
- Making reliable distributed systems in presence of software error
- Stuff Goes Bad, Erlang in Anger

Last updated 2020-08-17 05:46:46 UTC