Elixir

# Présentation

Elixir a été créé par José Valim, un acteur majeur l’écosystème Ruby et core contributeur au framework Ruby on Rails. En 2012, Valim a pour souhait de créer un langage aussi accessible que Ruby avec de meilleures performances et surtout une résilience à toute épreuve. C’est pour cette raison que Valim a décidé de créer Elixir sur la machine virtuelle Erlang, étant réputée pour les qualités souhaitées précédemment citées. Elixir simplifie la syntaxe du langage Erlang tout en gardant ses avantages.

# Justification du langage

Chris McCord, créateur du framework Phoenix (framework web en Elixir le plus utilisé) a déclaré en parlant d’Elixir : « What kind of business could I build if what before took a hundred servers could today take two servers? I think that’s enabling a lot of innovation. » Cette phrase bien non sans intérêt, est intéressante car plusieurs entreprises ayant migré vers Elixir donnent le même retour : le nombre de machines nécessaires au fonctionnement de leur application a pu largement diminuer. Alors comment cela est-ce possible ? Et bien notamment grâce aux principaux avantages du langage :

Elixir est conçu pour la concurrence et la performance, en partie grâce au fait qu’il se repose sur la machine virtuelle d’Erlang (langage inventé pour gérer les infrastructures téléphoniques grandissantes dans les années 80). Son système de processus légers que nous verrons par la suite permet de réaliser un nombre incalculable d’opérations simultanées en limitant au plus la latence sans bloquer un thread principal par exemple.

Mais dans ce cas-là, pourquoi ne pas tout simplement utiliser Erlang pourrait-on se dire ? Elixir propose une syntaxe moderne, facile à lire et à écrire, inspirée par le Ruby et le Python. Ce qui en fait à la fois un langage accessible tout en restant performant grâce à la VM Erlang.

C’est donc majoritairement ces deux aspects qui font que, selon moi, Elixir est un langage intéressant et mérite que l’on se penche dessus.

# Concepts

## Les bases

Elixir est un langage dynamiquement typé tout en restant fortement typé (à l’inverse de JavaScript par exemple). Il n’est donc pas possible de réaliser une opération du genre :

iex> 1 + "2"

(ArithmeticError) bad argument in arithmetic expression: 1 + "2"

L’annotation des types, appelée « typespecs » est possible mais facultative. Il est également possible de créer ses propres types.

Elixir est un langage fonctionnel, tous ses types de données sont immuables. Si nous essayons de réaffecter une variable existante, nous réassigner le label de la variable à une nouvelle valeur, pas la variable en lui-même qui sera supprimée par le Garbage Collector si elle n’est plus utilisée. TODO : pipe operator ici ?

Bien entendu, comme le langage est fonctionnel les boucles traditionnelles ne sont pas disponibles. Il faut utiliser des fonctions d’ordres supérieures (comme map ou reduce) ou la récursivité.

## Processes

TODO

## Fail fast / Let it crash

TODO

# Exemples mettant en évidence des points saillants du langage

# Définition du projet

L’idée est de mettre en pratique certains des avantages principaux du langage :

1. La concurrence
2. Son écosystème autour du web, notamment grâce au framework Phoenix, spécialisé dans le temps réel et les sockets.
3. Sa faible latence, ce qui est très bénéfique dans le cas d’un serveur web

Pour se faire, je souhaite réaliser une application web interactive entre les utilisateurs. Que ce soit un canvas partagé entre les utilisateurs comme pour une application de prise de notes / de dessin collaborative. Une application de dessin pourrait être bien adaptée car chaque point de chaque ligne de chaque utilisateur doit être envoyé au serveur centralisé pour synchroniser en temps réels tous les utilisateurs. Cette fonctionnalité se ferait via des connexions WebSockets, un aspect bien mis en avant par le framework Phoenix.

Le but principal est de générer un trafic important afin de pouvoir vraiment tester les avantages énoncés du langage. Pour ce faire, je pense utiliser un logiciel comme [artillery](https://www.artillery.io/) afin d’avoir une vraie montée en charge et pas uniquement quelques utilisateurs.

# En savoir plus

Apprendre les bases d’Elixir grâce au très bon livre d’introduction : <https://elixir-lang.org/getting-started/introduction.html>