Concepts of Programming Languages

Elixir programming language

Yahia Fawzy Rashwan
20221466198
Ahmed Ashraf Saeed
20221454558
Amr Ahmed Abdullatif
20201615334

Feb 24th, 2023

- **Elixir** is a functional, concurrent, general-purpose programming language designed for building scalable, fault-tolerance, and maintainable applications. It was created by José Valim and was released in 2011. **Elixir** is built on top of the Erlang Virtual Machine (BEAM) and is heavily inspired by Erlang's syntax and features. It combines the productivity and expressiveness of modern programming languages with the robustness and fault tolerance of the Erlang ecosystem.
- **Elixir** is aimed at large-scale sites and apps. It uses features of Ruby, Erlang, and Clojure to develop a high-concurrency and low-latency language. It was designed to handle large data volumes. **Elixir** is also used in telecommunications, e-commerce, and finance.
- **Elixir** has garnered favor among developers due to its productivity, performance, and resilience since its inception. It has found widespread application in diverse fields such as web development, distributed systems, real-time communication, Internet of Things (IoT), and gaming. The language consistently advances through frequent releases, updates, and collaborative contributions from the community, positioning itself as a promising choice for constructing scalable and easily maintainable applications.

1 Why Elixir

i. ***Concurrency***: **Elixir** was built with concurrency in mind since it was made for telecommunications. In **Elixir** concurrency is not handled in a usual way. Instead, it is handled through light-weight processes that are scheduled on the BEAM (Bogdan / Björn's Erlang Abstract Machine) VM and managed by the VM's built-in scheduler that acts in a similar manner to an operating system scheduler. Likewise, these processes communicate with each other through messages as if they were actual operating system processes. It also has new concurrency primitives that enable fine-grained control for the developer.

ii. ***Scalability***: **Elixir** has native support for scalable applications. It enables both vertical (Adding more power to your machine) and horizontal (adding more nodes to your cluster) scaling by relying on the powerful features of the BEAM VM. The BEAM VM makes use of its built-in schedulers that implement work-stealing to effeciently distribute the work load over the cluster. This eases the implementation of distrubuted systems using **Elixir**. This support for scalability is even more enhanced by partitioning the program into separate, isolated processes that are allowed to fail without the whole system failing, that talk to each other using messages as discussed above. This makes it possible to distribute these processes on the nodes of the cluster.

iii. ***Reliabilty and fault-tolerance***: **Elixir** implements an actor model. It splits the program into isolated processes called "actors", and these processes enjoy similar structure to a regular OS process – each contains its isolated memory (heap & stack) – and communication between these processes takes place through message passing, which is also similar to what happens in an OS. It also implements supervision trees where processes are structured into a tree of supervision. Each node in that tree represents a process that supervises the processes underneath it. In the unfortunate case when a child process crashes or stops functioning properly, its supervisor process will attempt to relaunch it.

iv. ***Hot code swapping***: **Elixir** allows hot-code swapping, which is the altering of code during runtime. This feature is one of the few features that truly sets **Elixir** apart from other programming Languages. This ensures high-availability even when different code is needed. You no longer need to take your service down to update the features you're offering or fix bugs. With **Elixir** you can just swap that part of your code out for a newer improved one.

v. ***Meta-programming***: Meta-programming in **Elixir** is mainly about its implementation and use of programming macros that expand during compile-time. These macros act in a simliar manner to the concept of reflection in Java. It allows the developer to make a program that is able to change its code base dynamically depending on the environment around it and how the macros are defined.

2 Project Idea: HTTP Web Server

The main goal of the project is to make a simple HTTP server that is basically a dumbed-down version of Nginx. It should server simple static HTML web pages. It also should have support for concurrency to handle multiple requests. And it should be configurable through a config file that allows the user to customize their experience. To add to that, it should log its errors in a log file in an appropriate place for the server admin to view to clean up when necessary.

3 Why Elixir for Web Servers?

We find that **Elixir** is very suitable for pogramming web servers and servers in general. It focus on **reliabilty and fault-tolerance** reserves a place for it on the web server space. Any web server needs to be up almost *all the time*. And **Elixir**'s supervision trees gives a unique and efficient approach to solving that problem.

Other than that, **Elixir**'s well-thought-out **concurrency** model makes the handling of multiple requests a walk in the park. And everyone knows that web servers, of all programs, get a ton of requets all the time.

And as we have mentioned earlier, **Elixir** also comes with built-in support for scalable apps that can easily distribute their wordload over multiple machines inside a cluster. This native support for **scalability** makes **Elixir** a perfect choice for web servers that tend to need lots of processing power and frequently get run over multple powerful machines.

Furthermore, **Elixir** has support for **hot-code swapping** which is a highly useful feature when it comes to web servers. Many times during the lifetime of a web server, the need may arise to add or remove features to/from it. Having the ability to swap code out/in during runtime without taking the server down is a really desirable feature that will avoid significant losses if that server was running vital services.

All of these features make **Elixir** the right choice for making web servers.