Clojure: A *Common Sense* language

Suresh Krishna Devendran And Zach Keane.

March 27, 2019

A language that fully utilizes the power of concurrency and multi-threading is Clojure. Clojure is a really fun language to learn and write in. It works well with the established JVM platform. Clojure was primarily designed with concurrency in mind. The workflow using REPL is phenomenal, hence Clojure is a very competitive and compelling language to program in. Clojure is a functional programming language that uses the Lisp dialect. Hence it is used in many platforms and it is very portable. The primary focus of this paper is to explore the data types and control structures of Clojure.

Like all other languages, Clojure provides some of the basic control structures. This includes if-statements, formatted similarly to other functional languages such as Scheme, even down to the "cond" statement for if-elif-else type branches. Switch statements are called "case" statements and performing "case 5" would execute the code that corresponds to "5". There are many ways to do loops, and a simple example would be to use the "dotimes" function, giving us the ability to choose our variable with the number of iterations very easily. Exceptions are identical to Java's exceptions, except in a Lisp format. We "catch" the exceptions as some Exception object, where we can decide to do what we wish. Finally blocks also exist for exceptions, as they will run no matter if the exception is caught or not, just as in the Java language.

There are many different data types in Clojure, and it often corresponds to the data types of Java. In fact, Clojure provides support for the primitive data types in Java, such as boolean, char, int, etc. The first category of data types in Clojure are Decimal Integers, which can be represented as Short, Long, or Int. In order to declare an Integer, just type a whole number, such as 52. Just like in other languages, you can declare Octal Numbers as 047 or Hexadecimal Numbers as 0xabc. Radix numbers are included, as you can declare the base of a number, followed by an "r", followed by the number itself. For example, 8r32 would be equal to 26, as 032 is equivalent to 26. Finally, there are also floating-point numbers, which are 32-bits as they are in Java. In Clojure, however, you can represent floats as scientific numbers, as you can declare a value as 6.22e-22.

To represent a null value as you would in Java, Clojure provides the shorter term, "nil." Any type of data, as in other languages, can be given this nil value to represent a lack of a real value.

As it exists in Scheme, the Atom data type also exists in Clojure. The intended use in this language is to contain an immutable data structure, such as an int, and you can change the value of this immutable data only through functions.

In summary, Clojure is a very powerful language that takes advantage of the Lisp dialect from other functional languages. It combines well with the established JVM module. We have an upper hand on basic functional languages with the tools that Clojure offers.