Macro in Clojure – Assignment 2

Aurélien SABALY

Geoffrey HARRAZI

After a reflection in our group we tried to find some way to handle the problem.

Firstly, we think about the wrapping. If we can wrap the macro in a function on Clojure. So, we find different way to do this.

**(**map #**(**find **%)** **(/ 10 2))**

Or we can also make another macro to do a more generic wrapping.

**(**defmacro make**-**func **[**m**]**

`**(**fn **[&** args#**]**

**(**eval `**(~**'~m ~@args#))))

user**=>** **(**map **(**make**-**func safe**)** **(/ 10 2))**

5

But these two ways look like cheating because we don’t really do the macro work as a function, we just encapsulate the macro into a function.

The main difference between function and macro is what the compilator do for executing it. In fact, for the function the parameter are interpreter by the compilator before the execution of the function. So, if we send (/ 10 2) to a function, they will receive the already interpreted form (5). So we have to find a tip to handle this. We think about two way to send the parameter without interpretation to the function. Send like this ‘(/ 10 2) or like this (/) (10) (2).

After this we can write the same code with the ‘if’, ‘try’ and the ‘catch’ as the macro code. But we have to find a way to merge the attribute together and the compilator will launch the operation. After we can return the result. We are novice in Clojure so we don’t find the solution for aggregate the attribute together and launch the computation. But we expect this exists and we try to find relevant article on this subject.

For the SQL macro, we can also use the wrapping technique. To do for us the SQL macro was a huge challenge on this language. We don’t success yet to create it. But we think about a probable solution for coding this macro as a function.

The Query was send in the function as a string.

Then we can isolate the key word like “SELECT”, “FROM”, “=” ect.. After we will be able to fulfil a map like this for example:

(hash-map :select (name sexe age work), :from (persons), :where (), :orderBy (DESC))

And after this we can use different algorithm with conditions, loop. In this algorithm, we will read the attribute. First the “from” attribute then the “select”, then “where” and finally “order by”. So, with this we can searching on the “database” with our search attribute and return a list which correspond with the query.