Scala | Closures

Last Updated : 05 Nov, 2019

Scala Closures are functions which uses one or more free variables and the return value of this function is dependent of these variable. The free variables are defined outside of the Closure Function and is not included as a parameter of this function. So the difference between a closure function and a normal function is the free variable. A **free variable** is any kind of variable which is not defined within the function and not passed as the parameter of the function. A free variable is not bound to a function with a valid value. The function does not contain any values for the free variable.

Example:

If we define a function as shown below:

```
def example(a:double) = a*p / 100
```

Now on running the above code we'll get an error starting not found p. So now we give a value to **p** outside the function.

```
// defined the value of p as 10
val p = 10

// define this closure.
def example(a:double) = a*p / 100
```

Now the above function is ready to run as the free variable has a value. Now if we run the functions as:

```
Calling the function: example(10000)
Input: p = 10
Output: double = 1000.0
```

Now what if the value of the free variable changes, how does the value of the closure function changes?
So basically what closure function does is, that it takes the most recent state of the free variable and changes the value of the closure function accordingly.

```
Input: p = 10
Output: double = 1000.0

Input: p = 20
Output: double = 2000.0
```

A closure function can further be classified into *pure* and *impure* functions, depending on the type of the free variable. If we give the free variable a type **var** then the variable tends to change the value any time throughout the entire code and thus may result in changing the value of the closure function. Thus this closure is a impure function. On the other-hand if we declare the free variable of the type **val** then the value of the variable remains constant and thus making the closure function a pure one.

Example:

```
// Addition of two numbers with
// Scala closure

// Creating object
object GFG
{
    // Main method
    def main(args: Array[String])
    {
        println( "Final_Sum(1) value = " + sum(1))
        println( "Final_Sum(2) value = " + sum(2))
```

Related Articles

Save for later

```
    // define closure function
    val sum = (b:Int) => b + a
}
```

Output:

```
Final_Sum(1) value = 5
Final_Sum(2) value = 6
Final_Sum(3) value = 7
```

Here, In above program function **sum** is a closure function. var a = 4 is impure closure. the value of a is same and values of b is different.

Example:

```
// Scala closure program to print a string

// Creating object
object GFG
{
    // Main method
    def main(args: Array[String])
    {

        var employee = 50
        // define closure function
        val gfg = (name: String) => println(s"Company name is $name"+
                      s" and total no. of employees are $employee")

        gfg("geeksforgeeks")
    }
}
```

Output:

```
Company name is geeksforgeeks and total no. of employees are 50.
```

Here, In above example **gfg** is a closure. var employee is mutable variable which can be change.

Like 0

Previous

Anonymous Functions in Scala

Next

Recursion in Scala

▲

RECOMMENDED ARTICLES

Page : 1 2 3

01 Closures in Golang
13, Mar 20

02 Closures in Ruby
01, Jul 20

03 Scala Tutorial – Learn Scala with Step By Step Guide
25, Nov 19

04 Scala short <(x: Short): Boolean
26, Nov 19

05 Scala short <(x: Char): Boolean
26, Nov 19

06 Scala Extractors
02, Apr 19

07 Scala | Partially Applied functions
28, Mar 19

08 Scala String indexOf(String str) method with example
30, Sep 19

Article Contributed By :

ShikharMathur1
@ShikharMathur1

Vote for difficulty

Easy Normal Medium Hard Expert

Improved By : aarthipa, hoobas20

Article Tags : Picked, Scala, Scala-Method, Scala

▲

Improve Article

Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

▲