SCALA CLOSURES

A **closure** is a function, whose return value depends on the value of one or more variables declared outside this function. Consider the following piece of code with anonymous function:

```
val multiplier = (i:Int) => i * 10
```

Here the only variable used in the function body, i * 0, is i, which is defined as a parameter to the function. Now let us take another piece of code:

```
val multiplier = (i:Int) => i * factor
```

There are two free variables in multiplier: **i** and **factor**. One of them, i, is a formal parameter to the function. Hence, it is bound to a new value each time multiplier is called. However, **factor** is not a formal parameter, then what is this? Let us add one more line of code:

```
var factor = 3
val multiplier = (i:Int) => i * factor
```

Now, **factor** has a reference to a variable outside the function but in the enclosing scope. Let us try the following example:

```
object Test {
   def main(args: Array[String]) {
      println( "muliplier(1) value = " +  multiplier(1) )
      println( "muliplier(2) value = " +  multiplier(2) )
   }
   var factor = 3
   val multiplier = (i:Int) => i * factor
}
```

When the above code is compiled and executed, it produces the following result:

```
C:/>scalac Test.scala
C:/>scala Test
muliplier(1) value = 3
muliplier(2) value = 6

C:/>
```

Above function references **factor** and reads its current value each time. If a function has no external references, then it is trivially closed over itself. No external context is required.