Closure

Closure

Closure merupakan merupakan sebuah *anonymous function* atau function tanpa nama yang dapat disimpan sebagai sebuah variable maupun dapat dijadikan sebagai nilai return dari sebuah function

Untuk pertama-tama, kita akan membahas bagaimana cara menggunakan closure yang disimpan sebagai variable.

http://4.bp.blogspot.com/-HO1GOGuQbSc/U5O83T5jB6I/AAAAAAAAA4M/TxIRYqK9pS8/w1200-h630-p-k-no-nu/Court-Closure.jpg

Closure(Declare closure in variable)

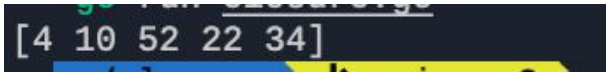
Cara untuk membuat *closure* yang dapat disimpan sebagai variable dapat dilihat seperti pada gambar pertama di sebelah kanan. Jika kita perhatikan, terdapat sebuah variable bernama *evenNumbers* yang mengandung sebuah *closure* berfungsi untuk mengumpulkan angka-angka genap pada sebuah *slice of int*.

Jika suatu variable mengandung sebuah *closure*, maka variable tersebut memiliki sifat seperti closure yang dikandungnya. Cara untuk memanggil *closure* tersebut, kita perlu memanggil *variable* yang mengandungnya.

Jika kita perhatikan pada gambar pertama, cara kita memanggil *closure* nya adalah dengan memanggil variable *evenNumbers*.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {
  var evenNumbers = func(numbers ...int) []int {
    var result []int

    for _, v := range numbers {
      if v % 2 == 0 {
        result = append(result, v)
      }
    }

    return result
  }

  var numbers = []int{4, 93, 77, 10, 52, 22, 34}

  fmt.Println(evenNumbers(numbers...))
}
```



Closure (IIFE)

IIFE atau singkatan dari immediately-invoked function expression merupakan sebuah *closure* yang dapat langsung tereksekusi ketika pertama kali dideklarasikan. Contohnya seperti pada gambar pertama di sebelah kanan.

Variable isPalindrome merupakan sebuah penampung dari *closure IIFE* yang digunakan untuk mengetahui apakah parameter yang diberikan merupakan sebuah kalimat palindrome atau bukan.

Jika kita ingin membuat suatu closure menjadi *IIFE*, maka kita perlu langsung memanggil *closure* tersebut secara langsung pada saat dideklarasikan. Perlu diingat bahwa kita tidak perlu lagi memanggil *closure IIFE* dengan tanda kurung () karena closure *IIFE* tereksekusi pada saat dideklarasikan.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua di sebelah kanan.

```go
package main

import "fmt"

func main() {

  var isPalindrome = func(str string) bool {
    var temp string = ""

    for i := len(str) - 1; i >= 0; i-- {
      temp += string(byte(str[i]))

    }

    return temp == str
  }("katak")

  fmt.Println(isPalindrome)

}
```

```
true
```

Closure (Closure as a return value)

Closure juga bisa dijadikan sebagai nilai kembalian dari suatu *function*. Contohnya seperti pada gambar sebelah kanan.

Function *findStudent* pada line 66 merupakan sebuah function yang menerima satu parameter dengan tipe data *slice of string* dan me-return sebuah yang *closure*. *Closure* yang direturn menerima satu parameter dengan tipe data string yang digunakan untuk mencari satu data *student* dari parameter yang diterima oleh function *findStudent*. Jika *closure* yang direturn menemukan data *student* nya, maka kalimat yang ada pada line 82 akan di return, jika tidak maka kalimat pada lone 80 akan di return.

Perhatikan juga pada line 73, kita menggunakan function *strings.ToLower* untuk mengecilkan huruf. Function *strings.ToLower* berasal dari package strings.

Kemudian pada line 60, variable bernama *find* akan menampung nilai return dari function *findStudent*. Karena function *findStudent* me-return sebuah *closure*, maka untuk memanggil *closure* yang di return oleh function *findStudent,* kita perlu memanggil variable *find* sekaligus memberikan parameternya.

```
50  package main
51
52  import (
53    "fmt"
54    "strings"
55  )
56
57  func main() {
58    var studentLists = []string{"Airell", "Nanda", "Mailo", "Schannel", "Marco"}
59
60    var find = findStudent(studentLists)
61
62    fmt.Println(find("airell"))
63
64  }
65
66  func findStudent(students []string) func(string) string {
67
68    return func(s string) string {
69      var student string
70      var position int
71
72      for i, v := range students {
73        if strings.ToLower(v) == strings.ToLower(s) {
74          student = v
75          position = i
76          break
77        }
78      }
79      if student == "" {
80        return fmt.Sprintf("%s does'nt exist!!!", s)
81      }
82      return fmt.Sprintf("We found %s at position %d", s, position+1)
83
84    }
85  }
```

Closure (Closure as a return value)

Jika syntax pada gambar di pertama di sebelah kanan dijalankan, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import (
  "fmt"
  "strings"
)

func main() {
  var studentLists = []string{"Airell", "Nanda", "Mailo", "Schannel", "Marco"}

  var find = findStudent(studentLists)

  fmt.Println(find("airell"))

}

func findStudent(students []string) func(string) string {

  return func(s string) string {
    var student string
    var position int

    for i, v := range students {
      if strings.ToLower(v) == strings.ToLower(s) {
        student = v
        position = i
        break
      }
    }
    if student == "" {
      return fmt.Sprintf("%s does'nt exist!!!", s)
    }
    return fmt.Sprintf("We found %s at position %d", s, position+1)

  }
}
```

Closure (Callback)

Callback adalah sebuah *closure* yang dijadikan sebagai parameter pada sebuah *function*. Contohnya seperti pada gambar pertama di sebelah kanan.

Function *findOddNumbers* merupakan sebuah function menerima 2 parameter. Parameter pertama memiliki tipe data *slice of int* dan parameter kedua merupakan sebuah *callback* yang menerima satu parameter dengan tipe data *int* dan me-return nilai *bool*.

Function *findOddNumbers* ini digunakan untuk mencari jumlah angka ganjil yang diberikan pada parameter pertamanya. Lalu pengecekan nya tersebut dilakukan oleh *callback* nya.

Jika *callback* me-return nilai *true,* maka variable *totalOddNumbers* pada line 104 akan ditambahkan. Dan function *findOddNumbers* akan me-return variable tersebut.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```
90
91  import "fmt"
92
93  func main() {
94    var numbers = []int{2, 5, 8, 10, 3, 99, 23}
95
96    var find = findOddNumbers(numbers, func(number int) bool {
97      return number%2 != 0
98    })
99
100   fmt.Println("Total odd numbers:", find)
101 }
102
103 func findOddNumbers(numbers []int, callback func(int) bool) int {
104   var totalOddNumbers int
105   for _, v := range numbers {
106     if callback(v) {
107       totalOddNumbers++
108     }
109   }
110   return totalOddNumbers
111 }
```

```
Total odd numbers: 4
```

HACKTIV8

Closure (Callback)

Jika kita perhatikan pada penulisan parameter *callback* pada halaman sebelumnya. Kita dapat menggunakan *type alias* untuk mempersingkat penulisan parameter *callback* nya. Contohnya seperti pada gambar di bawah yang telah di highlight.

```go
package main

import "fmt"

type isOddNum func(int) bool      You, seconds ago • Uncommitted changes

func main() {
  var numbers = []int{2, 5, 8, 10, 3, 99, 23}

  var find = findOddNumbers(numbers, func(number int) bool {
    return number%2 != 0
  })

  fmt.Println("Total odd numbers:", find)
}

func findOddNumbers(numbers []int, callback isOddNum) int {      You, sec
  var totalOddNumbers int
  for _, v := range numbers {
    if callback(v) {
      totalOddNumbers++
    }
  }
  return totalOddNumbers
}
```