MATERI

Hari 9 - Defer, Panic, Recover & Error

Defer

Defer function adalah function yang bisa kita jadwalkan untuk dieksekusi setelah sebuah function selesai di eksekusi. Defer function akan selalu dieksekusi walaupun terjadi error di function yang dieksekusi. berikut ini contoh penggunaannya:

```
package main

import (
  "fmt"
)

func logging(){
  fmt.Println("selesai memanggil function")
}

func runApplication(){
  defer logging()
  fmt.Println("Run Application")
}

func main(){
  runApplication()
}
```

dapat terlihat diatas terdapat function runApplication yang didalamnya terdapat kode defer logging(). defer logging ini akan di eksekusi meskipun terjadi error di program dan defer logging di eksekusi terakhir meskipun di letakkan diatas. secara default defer function lebih baik di baris pertama dari function.

Panic

Panic function adalah function yang bisa kita gunakan untuk menghentikan program. Panic function biasanya dipanggil ketika terjadi error pada saat program kita berjalan. Saat panic function dipanggil, program akan terhenti, namun defer function tetap akan dieksekusi. berikut ini contoh penggunaannya:

```
package main

import (
  "fmt"
)

func endApp(){
  fmt.Println("End App")
}

func runApp(error bool){
  defer endApp()
  if error{
    panic("ERROR")
  }
}

func main(){
  runApp(true)
}
```

pada kode diatas terlihat bahwa didalam function runApp terdapat kode panic("ERROR") ini berarti program akan di berhentikan dengan panic dan messagenya adalah "ERROR".

```
goroutine 1 [running]:
main.runApp(0x0)
        C:/Workspaces/Materi-Golang/Example/panic.go:14 +0x65
main.main()
        C:/Workspaces/Materi-Golang/Example/panic.go:19 +0x1e
exit status 2

C:\Workspaces\Materi-Golang\Example>
```

Recover

Recover adalah function yang bisa kita gunakan untuk menangkap data panic. Dengan recover proses panic akan terhenti, sehingga program akan tetap berjalan. berikut contoh penggunaannya:

```
package main

import (
  "fmt"
)

func endApp(){
  fmt.Println("End App")
  message := recover()
  fmt.Println("Terjadi Error", message)
}

func runApp(error bool){
  defer endApp()
  if error{
    panic("ERROR")
  }
}

func main(){
  runApp(true)
}
```

pada kode tersebut jika dibandingkan dengan contoh di panic, terdapat perbedaan, yaitu pada function endApp terdapat variabel message yang berisi recover(), recover() tersebut berfungsi untuk menangkap pesan error yang di keluarkan oleh panic.

```
C:\Workspaces\Materi-Golang\Example>go run recover.go
End App
Terjadi Error ERROR

C:\Workspaces\Materi-Golang\Example>
```

Error

Error merupakan sebuah tipe. Error memiliki 1 buah property berupa method Error(), method ini mengembalikan detail pesan error dalam string. Error termasuk tipe yang isinya bisa nil. cara menuliskan tipe data error cukup dengan menulis error (huruf kecil semua).

Di Go, banyak sekali fungsi yang mengembalikan nilai balik lebih dari satu. Biasanya, salah satu kembalian adalah bertipe error. Contohnya seperti pada fungsi strconv.Atoi(). Fungsi tersebut digunakan untuk konversi data string menjadi numerik. Fungsi ini mengembalikan 2 nilai balik. Nilai balik pertama adalah hasil konversi, dan nilai balik kedua adalah error. Ketika konversi berjalan mulus, nilai balik kedua akan bernilai nil. Sedangkan ketika konversi gagal, penyebabnya bisa langsung diketahui dari error yang dikembalikan.

Dibawah ini merupakan contoh program sederhana untuk deteksi inputan dari user, apakah numerik atau bukan. Dari sini kita akan belajar mengenai pemanfaatan error.

```
  "fmt"
  "strconv"
)

func main() {
  var input string
  fmt.Print("Type some number: ")
  fmt.Scanln(&input)

  number, err := strconv.Atoi(input)

  if err == nil {
    fmt.Println(number, "is number")
  } else {
    fmt.Println(input, "is not number")
    fmt.Println(err.Error())
  }
}
```

Jalankan program, maka muncul tulisan "Type some number: ". Ketik angka bebas, jika sudah maka enter. Statement fmt.Scanln(&input) dipergunakan untuk men-capture inputan yang diketik oleh user sebelum dia menekan enter, lalu menyimpannya sebagai string ke variabel input.

Selanjutnya variabel tersebut dikonversi ke tipe numerik menggunakan strconv.Atoi(). Fungsi tersebut mengembalikan 2 data, ditampung oleh number dan err. Data pertama (number) berisi hasil konversi. Dan data kedua err, berisi informasi errornya (jika memang terjadi error ketika proses konversi).

Setelah itu dilakukan pengecekkan, ketika tidak ada error, number ditampilkan. Dan jika ada error, input ditampilkan beserta pesan errornya.

Pesan error bisa didapat dari method Error() milik tipe error.

Membuat Custom Error

Selain memanfaatkan error hasil kembalian suatu fungsi internal yang tersedia, kita juga bisa membuat objek error sendiri dengan menggunakan fungsi `errors.New()` (harus import package `errors` terlebih dahulu).

berikut contoh penggunaanya:

```
package main

import (
  "fmt"
  "errors"
)

func pembagian(nilai uint, pembagi uint)(float64,
error){
  if pembagi == 0 {
    return 0.0, errors.New("Maaf pembagi tidak
boleh NOL")
  }else{
    return float64(nilai/pembagi), nil
  }
}

func main(){
  hasil, err := pembagian(8,4)
  if err == nil{
    fmt.Println("Hasil", hasil)
  }else{
    fmt.Println("Error", err.Error())
  }
}
```

pada kode tersebut terdapat function pembagian dimana terdapat pengecekan pembagi sama dengan nol atau tidak jika pembaginya nol makan akan dikembalikan nilai errornya.

Referensi Video:

- Defer, Panic & Recover (Programmer Zaman Now)
- error interface (Programmer Zaman Now)

- https://dasarpemrogramangolang.novalagung.com/A-defer-exit.html
- https://dasarpemrogramangolang.novalagung.com/A-error-panic-recover.html

Rating - Feedback

Berikan Rating pada posting ini:

0

Berikan kritik dan saran..

Submit