



Error, Panic & Recover

Error

Error merupakan sebuah tipe data pada bahasa Go yang digunakan untuk me-return sebuah error jika ada error yang terjadi. Umumnya, nilai *error* akan di-return pada pada posisi terakhir dari nilai-nilai return sebuah *function*.

Contohnya seperti pada function *strconv.Atoi* yang digunakan untuk mengkonversi tipe data *string* menjadi tipe data *int*.

Function *strconv.Atoi* akan me-return *error* jika nilai dengan tipe data *string* tersebut tidak bisa di konversi menjadi sebuah tipe data *int*, seperti contohnya nilai string yang mengandung nilai diluar angka "123GH".

Perhatikan pada gambar disebelah kanan.

Terdapat 2 variable bernama *number* dan *err*. Variable *number* memiliki tipe data *int* dan variable *err* memiliki tipe data *error*. Lalu kedua variable tersebut di reassign untuk menampung hasil dari function *strconv.Atoi*.

```
1 package main
2
3 import (
4     "fmt"
5     "strconv"
6 )
7
8 func main() {
9     var number int
10    var err error
11
12    number, err = strconv.Atoi("123GH")
13
14    if err == nil {
15        fmt.Println(number)
16    } else {
17        fmt.Println(err.Error())
18    }
19
20    number, err = strconv.Atoi("123")
21
22    if err == nil {
23        fmt.Println(number)
24    } else {
25        fmt.Println(err.Error())
26    }
27
28 }
```



Error

Karena argumen yang kita berikan pada function `strconv.Atoi` mengandung huruf, maka function `strconv.Atoi` akan me-return nilai dengan tipe data `error` yang kemudian akan ditampung oleh variable `err`.

Ketika tidak ada error yang di return maka variable `err` akan menjadi `nil` karena zero value dari tipe data `error` adalah `nil`.

Maka dari itu kita bisa menggunakan kondisional untuk memeriksa jika variable `err` sama dengan `nil` maka variable `number` dapat ditampilkan, dan jika tidak maka variable `err` akan ditampilkan.

Lalu kita juga menggunakan method `Error` untuk menampilkan pesan jika terjadi sebuah error. Method `Error` berasal dari tipe data `error`.

```
1 package main
2
3 import (
4     "fmt"
5     "strconv"
6 )
7
8 func main() {
9     var number int
10    var err error
11
12    number, err = strconv.Atoi("123GH")
13
14    if err == nil {
15        fmt.Println(number)
16    } else {
17        fmt.Println(err.Error())
18    }
19
20    number, err = strconv.Atoi("123")
21
22    if err == nil {
23        fmt.Println(number)
24    } else {
25        fmt.Println(err.Error())
26    }
27
28 }
```



Error

Kemudian pada line 20, karena argumen yang kita berikan pada function `strconv.Atoi` tersebut adalah angka semua, maka tidak akan ada error yang di return dan line 23 dapat ditampilkan.

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

```
1 package main
2
3 import (
4     "fmt"
5     "strconv"
6 )
7
8 func main() {
9     var number int
10    var err error
11
12    number, err = strconv.Atoi("123GH")
13
14    if err == nil {
15        fmt.Println(number)
16    } else {
17        fmt.Println(err.Error())
18    }
19
20    number, err = strconv.Atoi("123")
21
22    if err == nil {
23        fmt.Println(number)
24    } else {
25        fmt.Println(err.Error())
26    }
27 }
28
```

```
go run error.go
strconv.Atoi: parsing "123GH": invalid syntax
123
```



Error (Custom error)

Kita juga dapat membuat pesan error kita sendiri dengan menggunakan method *New* yang merupakan sebuah method miliki tipe data *error*. Contohnya seperti pada gambar pertama di sebelah kanan.

Function *fmt.Scanln* digunakan untuk menangkap input yang kita tuliskan pada terminal. Lalu kita memberikan tanda memberikan nilai *pointer* dari variable *password* pada argumennya agar inputan kita dapat di simpan pada variable *password*.

```
32 package main
33
34 import (
35     "errors"
36     "fmt"
37 )
38
39 func main() {
40     var password string
41
42     fmt.Scanln(&password)
43
44     if valid, err := validPassword(password); err != nil {
45         fmt.Println(err.Error())
46     } else {
47         fmt.Println(valid)
48     }
49 }
50
51
52 func validPassword(password string) (string, error) {
53     pl := len(password)
54
55     if pl < 5 {
56         return "", errors.New("password has to have more than 4 characters")
57     }
58
59     return "Valid password", nil
60 }
```



Error (Custom error)

Kemudian kita membuat function *validPassword* yang digunakan untuk memeriksa apakah password yang kita input pada terminal memiliki panjang karakter lebih dari 4.

Jika memang lebih dari 4 karakter maka function *validPassword* akan mereturn sebuah pesan berupa "Valid password" dan nilai *error* yang direturn berupa *nil*.

Namun jika terjadi sebaliknya, maka function *validPassword* akan me-return akan mereturn sebuah empty *string* dan sebuah pesan error yang dibuat menggunakan method *errors.New*.

```
32 package main
33
34 import (
35 |     "errors"
36 |     "fmt"
37 )
38
39 func main() {
40 |     var password string
41
42 |     fmt.Scanln(&password)
43
44 |     if valid, err := validPassword(password); err != nil {
45 |         fmt.Println(err.Error())
46 |     } else {
47 |         fmt.Println(valid)
48 |     }
49 }
50
51
52 func validPassword(password string) (string, error) {
53 |     pl := len(password)
54
55 |     if pl < 5 {
56 |         return "", errors.New("password has to have more than 4 characters")
57 |     }
58
59 |     return "Valid password", nil
60 }
```



Error, Panic & Recover - Sesi 5

Error (Custom error)

Pada line 44 hingga 48, terdapat sebuah kondisional if else untuk memeriksa hasil dari function validPassword. Dan jika kita jalankan pada terminal kita diikuti dengan mengetikkan input, maka hasilnya akan seperti pada gambar kedua.

Bisa dilihat, pesan yang ditunjukkan pada gambar kedua merupakan pesan error yang dibuat menggunakan method *errors.New*. Pesan error ditampilkan karena kita menginput *asd* yang dimana panjangnya tidak lebih dari 4 karakter.

Namun misalkan kita menginput lebih dari 4 karakter, maka hasilnya akan seperti pada gambar ketiga.

```
32 package main
33
34 import (
35     "errors"
36     "fmt"
37 )
38
39 func main() {
40     var password string
41
42     fmt.Scanln(&password)
43
44     if valid, err := validPassword(password); err != nil {
45         fmt.Println(err.Error())
46     } else {
47         fmt.Println(valid)
48     }
49 }
50
51
52 func validPassword(password string) (string, error) {
53     pl := len(password)
54
55     if pl < 5 {
56         return "", errors.New("password has to have more than 4 characters")
57     }
58
59     return "Valid password", nil
60 }
```

```
asd
password has to have more than 4 characters
```

```
asdfgh
Valid password
```



Panic

Panic digunakan untuk menampilkan stack trace error sekaligus menghentikan flow goroutine (karena main() juga merupakan goroutine, maka behaviour yang sama juga berlaku). Setelah ada panic, proses akan terhenti, apapun setelah tidak di-eksekusi kecuali proses yang sudah di-defer sebelumnya (akan muncul sebelum panic error).

Panic menampilkan pesan error di console, sama seperti fmt.Println(). Informasi error yang ditampilkan adalah stack trace error, jadi sangat mendetail dan heboh.

```
70 func main() {
71     var password string
72
73     fmt.Scanln(&password)
74
75     if valid, err := validPassword(password); err != nil {
76         panic(err.Error())
77     } else {
78         fmt.Println(valid)
79     }
80
81 }
82
83 func validPassword(password string) (string, error) {
84     pl := len(password)
85
86     if pl < 5 {
87         return "", errors.New("password has to have more than 4 characters")
88     }
89
90     return "Valid password", nil
91 }
```



Error, Panic & Recover - Sesi 5

Panic

Perhatikan gambar pertama di sebelah kanan. Kita mengganti function *fmt.Println* pada line 76 menjadi function *Panic*.

Lalu mari kita jalankan terminal kita dan mengetikan sebuah input kurang dari 5 karakter agar menghasilkan error. Hasilnya akan seperti pada gambar kedua.

```
70 func main() {
71 |     var password string
72 |
73 |     fmt.Scanln(&password)
74 |
75 |     if valid, err := validPassword(password); err != nil {
76 |         panic(err.Error())
77 |     } else {
78 |         fmt.Println(valid)
79 |     }
80 |
81 | }
82 |
83 | func validPassword(password string) (string, error) {
84 |     pl := len(password)
85 |
86 |     if pl < 5 {
87 |         return "", errors.New("password has to have more than 4 characters")
88 |     }
89 |
90 |     return "Valid password", nil
91 | }
```

```
qq
panic: password has to have more than 4 characters

goroutine 1 [running]:
main.main()
  /Users/sofyan/go/src/GLNG-Materi-Baru-PTP-2021/Sesi5/panic/error.go:76 +0x199
exit status 2
```



Recover

Function *recover* digunakan untuk menangkap error yang terjadi pada sebuah *Goroutine*. Sekarang kita akan menambahkan sebuah function bernama *catchErr* pada codingan kita sebelumnya yang akan menangkap error panic nya menggunakan function *recover*.

Dan perlu diingat bahwa kita perlu memanggil function *catchErr* dengan keyword *defer*.

```
70 func main() {
71     defer catchErr()
72
73     var password string
74
75     fmt.Scanln(&password)
76
77     if valid, err := validPassword(password); err != nil {
78         panic(err.Error())
79     } else {
80         fmt.Println(valid)
81     }
82
83 }
84
85 func catchErr() {
86     if r := recover(); r != nil {
87         fmt.Println("Error occured:", r)
88     } else {
89         fmt.Println("Application running perfectly")
90     }
91 }
92
93 func validPassword(password string) (string, error) {
94     pl := len(password)
95
96     if pl < 5 {
```



Recover

Jika kita jalankan, maka hasilnya akan seperti pada gambar dibawah.

Bisa kita lihat, pesan dari panic error tidak ditampilkan karena telah di tangkap menggunakan function *recover* yang terdapat didalam function *catchErr*.

```
123  
Error occured: password has to have more than 4 characters
```

