MATERI

Hari 7 - Struct dan Method

Struct

Go tidak memiliki class yang ada di bahasa-bahasa strict OOP lain. Tapi Go memiliki tipe data struktur yang disebut dengan Struct.

Struct adalah kumpulan definisi variabel (atau property) dan atau fungsi (atau method), yang dibungkus sebagai tipe data baru dengan nama tertentu. Property dalam struct, tipe datanya bisa bervariasi. Mirip seperti `map`, hanya saja key-nya sudah didefinisikan di awal, dan tipe data tiap itemnya bisa berbeda.

1. Deklarasi Struct

Keyword `type` digunakan untuk deklarasi struct. Di bawah ini merupakan contoh cara penggunaannya.

```
type student struct {
    name string
    grade int
}
```

Struct `student` dideklarasikan memiliki 2 property, yaitu `name` dan `grade`.

2. Penerapan Struct

berbeda dengan map, struct tidak langsung bisa di gunakan begitu saja, struct bisa di gunakan dengan membuat data atau biasa disebut object berdasarkan dari struct yang sudah di buat. berikut contohnya:

```
func main() {
  var john student
  john.name = "john doe"
  john.grade = 2

  fmt.Println("name  :", john.name)
  fmt.Println("grade :", john.grade)
}
```

seperti yang terlihat pada contoh diatas bahwa cara membuat object dari sebuah struct sama seperti kita mendeklarasikan variabel, di mulai dari nama variabel dan diikuti dengan tipe data, pada contoh diatas tipe datanya adalah student yang didapatkan dari type student struct

3. Struct Literals

Sebelumnya kita telah membuat object dari struct, namun sebenarnya ada banyak cara yang bisa kita gunakan untuk membuat data dari struct, berikut ini contohnya:

```
// cara pertama
var john = student{}
john.name = "wick"
john.grade = 2

// cara kedua tetapi isinya harus berurutan
var doe = student{"doe", 2}

// cara ketiga dengan nama property tetapi tidak
harus berurutan
var jack = student{name: "jack", grade: 2}

fmt.Println("student 1 :", john.name)
fmt.Println("student 2 :", doe.name)
fmt.Println("student 3 :", jack.name)
```

4. Embedded Struct

Embedded struct adalah mekanisme untuk menempelkan sebuah struct sebagai properti struct lain. Agar lebih mudah dipahami, mari kita bahas kode berikut.

```
import "fmt"

type person struct {
  name string
  age  int
}

type student struct {
  grade int
  person
}

func main() {
  // contoh 1
  var john = student{}
  john.name = "john"
  john.age = 21
  john.grade = 2

  fmt.Println("name  :", john.name)
  fmt.Println("age   :", john.age)
  fmt.Println("age   :", john.person.age)
  fmt.Println("grade :", john.grade)

  // contoh 2
  var doeData = person{name: "doe", age: 21}
  var doe = student{person: doeData, grade: 2}

  fmt.Println("name  :", doe.name)
  fmt.Println("age   :", doe.age)
  fmt.Println("grade :", doe.grade)
}
```

dapat terlihat diatas bahwa object john di inisialisasi berdasarkan object student dan property yang sebelumnya terdapat di object person dapat di gunakan jika struct student embed struct dari person

5. Anonymous Struct

Anonymous struct adalah struct yang tidak dideklarasikan di awal sebagai tipe data baru, melainkan langsung ketika pembuatan object. Teknik ini cukup efisien untuk pembuatan object yang struct-nya hanya dipakai sekali.

```
package main

import "fmt"

type person struct {
  name string
  age  int
}

func main() {
  var john = struct {
    person
    grade int
  }{}
  john.person = person{"wick", 21}
  john.grade = 2

  fmt.Println("name  :", john.person.name)
  fmt.Println("age   :", john.person.age)
  fmt.Println("grade :", john.grade)
}
```

Pada kode di atas, variabel `john` langsung diisi objek anonymous struct yang memiliki property `grade`, dan property `person` yang merupakan embedded struct.

Salah satu aturan yang perlu diingat dalam pembuatan anonymous struct adalah, deklarasi harus diikuti dengan inisialisasi. Bisa dilihat pada `john` setelah deklarasi struktur struct, terdapat kurung kurawal untuk inisialisasi objek. Meskipun nilai tidak diisikan di awal, kurung kurawal tetap harus ditulis.

```
  grade int
}{}

// anonymous struct dengan pengisian property
var doe = struct {
  person
  grade int
}{
  person: person{"wick", 21},
  grade:  2,
}
```

6. Nested struct

Nested struct adalah anonymous struct yang di-embed ke sebuah struct. Deklarasinya langsung didalam struct peng-embed. Contoh:

```
type student struct {
  person struct {
    name string
    age  int
  }
  grade   int
}
```

Method

Method adalah fungsi yang menempel pada `type` (bisa `struct` atau tipe data lainnya). Method bisa diakses lewat variabel objek.

Keunggulan method dibanding fungsi biasa adalah memiliki akses ke property struct hingga level *private* (level akses nantinya akan dibahas lebih detail pada bab selanjutnya). Dan juga, dengan menggunakan method sebuah proses bisa di-enkapsulasi dengan baik.

Penerapan Method

Cara menerapkan method sedikit berbeda dibanding penggunaan fungsi. Ketika deklarasi, ditentukan juga siapa pemilik method tersebut. Contohnya bisa dilihat pada kode berikut:

```
package main

import "fmt"
import "strings"

type student struct {
  name  string
  grade int
}

func (s student) sayHello() {
  fmt.Println("halo", s.name)
}
```

Cara deklarasi method sama seperti fungsi, hanya saja perlu ditambahkan deklarasi variabel object di sela-sela keyword `func` dan nama fungsi. Struct yang digunakan akan menjadi pemilik method.

`func (s student) sayHello()` maksudnya adalah fungsi `sayHello` dideklarasikan sebagai method milik struct `student`. Pada contoh di atas struct `student` memiliki dua buah method, yaitu `sayHello()`

Contoh pemanfaatan method bisa dilihat pada kode berikut.

```
func main() {
    var john = student{"john wick", 21}
    john.sayHello()
}
```

Referensi Video:

Referensi Tulisan:

- https://dasarpemrogramangolang.novalagung.com/A-struct.html
- https://dasarpemrogramangolang.novalagung.com/A-method.html

Rating - Feedback

Berikan Rating pada posting ini:

0

Berikan kritik dan saran..

Submit