Welcome To Golang By Example

Function/Method Overloading in Golang (Alternatives/Workaround)

Posted on December 1, 2019

Function/Method Overloading means that that the same function/method name can be used with a different number and types of parameters

See this post for difference between function and method in Go – https://golangbyexample.com/difference-between-method-function-go

Eg.

```
func X()
func X(name string)
func X(name, address string)
func X(name string, age int)
```

Go doesn’t support method/function overloading. See this faq for the reason https://golang.org/doc/faq#overloading

According to the above faq things are simpler without it.

We can workaround Method/Function overloading in GO using

- **Variadic Function** – A Variadic Function is a function that accepts a variable number of arguments
- **Empty Interface** – It is an interface without any methods.

There are two cases for Method/Function Overloading

1.**Different number of parameters but of the same type:**

Above case can easily be handled using variadic functions. Notice in below code the parameters are of one type i.e. **int.**

```
package main

import "fmt"

func main() {
    fmt.Println(add(1, 2))
    fmt.Println(add(1, 2, 3))
    fmt.Println(add(1, 2, 3, 4))
}

func add(numbers ...int) int {
    sum := 0
    for _, num := range numbers {
        sum += num
    }
    return sum
```

Output:

```
3
6
10
```

2.**Different number of parameters and of different types**

This case can be handled using both variadic function and empty interface

```
package main

import "fmt"

func main() {
    handle(1, "abc")
    handle("abc", "xyz", 3)
    handle(1, 2, 3, 4)
}

func handle(params ...interface{}) {
    fmt.Println("Handle func called with parameters:")
    for _, param := range params {
        fmt.Printf("%v\n", param)
    }
}
```

Output:

```
Handle func called with parameters:
1
abc
Handle func called with parameters:
abc
xyz
3
Handle func called with parameters:
1
2
3
4
```

We can also use a switch case to get the exact parameters and use them accordingly. See the below example.

```
package main

import "fmt"

type person struct {
    name   string
    gender string
    age    int
}

func main() {
    err := addPerson("Tina", "Female", 20)
    if err != nil {
        fmt.Println("PersonAdd Error: " + err.Error())
    }

    err = addPerson("John", "Male")
    if err != nil {
        fmt.Println("PersonAdd Error: " + err.Error())
    }

    err = addPerson("Wick", 2, 3)
    if err != nil {
        fmt.Println("PersonAdd Error: " + err.Error())
    }
}

func addPerson(args ...interface{}) error {
    if len(args) > 3 {
        return fmt.Errorf("Wront number of arguments passed")
    }
    p := &person{}
    //0 is name
    //1 is gender
    //2 is age
    for i, arg := range args {
        switch i {
        case 0: // name
            name, ok := arg.(string)
            if !ok {
                return fmt.Errorf("Name is not passed as string")
            }
            p.name = name
        case 1:
            gender, ok := arg.(string)
            if !ok {
                return fmt.Errorf("Gender is not passed as string"
            }
            p.gender = gender
        case 2:
            age, ok := arg.(int)
            if !ok {
                return fmt.Errorf("Age is not passed as int")
            }
            p.age = age
        default:
            return fmt.Errorf("Wrong parametes passed")
        }
    }
    fmt.Printf("Person struct is %+v\n", p)
    return nil
}
```

Note: Wherever the arg is not passed it is substituted as default.

Output:

```
Person struct is &{name:Tina gender:Female age:20}
Person struct is &{name:John gender:Male age:0}
PersonAdd Error: Gender is not passed as string
```

function go golang method overloading

?

Privacy Badger has replaced this Disqus widget

Allow once   Always allow on this site

Follow @golangbyexample

Search ...

Popular Articles

Golang Comprehensive Tutorial Series

All Design Patterns in Go (Golang)

Slice in golang

Variables in Go (Golang) – Complete Guide

OOP: Inheritance in GOLANG complete guide

Using Context Package in GO (Golang) – Complete Guide

All data types in Golang with examples

Understanding time and date in Go (Golang) – Complete Guide