03

Funções com múltiplos retornos

Transcrição

Para entendermos melhor a questão das funções que retornam mais de um valor, nada melhor do que aprendermos isso fazendo uma função desse tipo. Vamos criar a função `devolveNome`, que retorna um nome, logo uma string:

```
// restante do código omitido

func devolveNome() string {
    nome := "Douglas"
    return nome
}
```

E na função `main`, vamos guardar o retorno dessa função em uma variável e imprimi-la. Além disso, vamos comentar as chamadas das funções `exibeIntroducao` e `exibeMenu`, somente para não ficarmos com muita informação na tela:

```
// restante do código omitido

func main() {

    //exibeIntroducao()
    //exibeMenu()

    nome := devolveNome()
    fmt.Println(nome)

    comando := leComando()

    switch comando {
    case 1:
        iniciarMonitoramento()
    case 2:
        fmt.Println("Exibindo Logs...")
    case 0:
        fmt.Println("Saindo do programa")
        os.Exit(0)
    default:
        fmt.Println("Não conheço este comando")
        os.Exit(-1)
    }
}
```

Agora, além de devolver o nome, vamos fazer com quê a função também retorne a idade, para isso, basta separar os retornos por vírgula:

```
// restante do código omitido

func devolveNomeEIdade() string {
    nome := "Douglas"
```

```
    idade := 24
    return nome, idade
}
```

Além disso, precisamos adicionar o tipo `int` como retorno da função. Nesse caso, quando temos mais de um tipo, precisamos colocá-lo entre parênteses:

```
// restante do código omitido

func devolveNomeEIdade() (string, int) {
    nome := "Douglas"
    idade := 24
    return nome, idade
}
```

Logo, o primeiro valor que a função retorna é uma string, e o segundo é um número inteiro. E como agora a função retorna dois valores, nós precisamos criar duas variáveis:

```
// restante do código omitido

func main() {

    //exibeIntroducao()
    //exibeMenu()

    nome, idade := devolveNomeEIdade()
    fmt.Println(nome, "tem", idade, "anos")

    comando := leComando()

    switch comando {
    case 1:
        iniciarMonitoramento()
    case 2:
        fmt.Println("Exibindo Logs...")
    case 0:
        fmt.Println("Saindo do programa")
        os.Exit(0)
    default:
        fmt.Println("Não conheço este comando")
        os.Exit(-1)
    }
}
```

Com isso, conseguimos criar uma função que retorna dois valores, uma string e um inteiro.

Como ignorar algum retorno da função?

Isso é bem interessante no Go, e o mesmo ocorre com a função `Get`, ela retorna a resposta da requisição e um possível erro que possa ter ocorrido. Mas e se só estamos interessados em somente um dos retornos, como devemos fazer?

Quando não queremos saber de um dos retornos, queremos ignorá-lo, nós utilizamos o operador de identificador em branco (_):

```
// restante do código omitido

func main() {

    //exibeIntroducao()
    //exibeMenu()

    _, idade := devolveNomeEIdade()
    fmt.Println(idade)

    comando := leComando()

    switch comando {
    case 1:
        iniciarMonitoramento()
    case 2:
        fmt.Println("Exibindo Logs...")
    case 0:
        fmt.Println("Saindo do programa")
        os.Exit(0)
    default:
        fmt.Println("Não conheço este comando")
        os.Exit(-1)
    }
}
```

Esse operador diz para o Go ignorar o retorno, no nosso caso, o primeiro deles, pois só estamos interessados no segundo retorno. Então, se não queremos algum retorno de uma determinada função, no momento em que formos declarar as variáveis, basta utilizar esse operador no seu lugar.

Do mesmo jeito, como não queremos tratar possíveis erros de requisição nesse momento, podemos ignorar esse retorno:

```
// restante do código omitido

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
    resp, _ := http.Get(site)
    fmt.Println(resp)
}
```

Feito isso, podemos voltar com o código que havíamos comentado, e remover a função que criamos, visto que ela foi feita somente para entendermos as funções de múltiplos retornos:

```
package main

import (
    "fmt"
    "net/http"
    "os"
```

```
)

func main() {

    exibeIntroducao()
    exibeMenu()
    comando := leComando()

    switch comando {
    case 1:
        iniciarMonitoramento()
    case 2:
        fmt.Println("Exibindo Logs...")
    case 0:
        fmt.Println("Saindo do programa")
        os.Exit(0)
    default:
        fmt.Println("Não conheço este comando")
        os.Exit(-1)
    }
}

func exibeIntroducao() {
    nome := "Douglas"
    versao := 1.2
    fmt.Println("Olá, sr.", nome)
    fmt.Println("Este programa está na versão", versao)
}

func exibeMenu() {
    fmt.Println("1- Iniciar Monitoramento")
    fmt.Println("2- Exibir Logs")
    fmt.Println("0- Sair do Programa")
}

func leComando() int {
    var comandoLido int
    fmt.Scan(&comandoLido)
    fmt.Println("O comando escolhido foi", comandoLido)

    return comandoLido
}

func iniciarMonitoramento() {
    fmt.Println("Monitorando...")
    site := "https://www.alura.com.br"
    resp, _ := http.Get(site)
    fmt.Println(resp)
}
```

Com isso, podemos seguir com a construção da nossa aplicação, a partir do próximo vídeo.