07

Criando o Hello World

Transcrição

Para construirmos o nosso primeiro programa em Go, primeiramente devemos entender algumas instruções iniciais dele. Se queremos que o nosso programa seja um executável, ou seja, se queremos que depois o programa seja executado no nosso computador, devemos declarar o programa assim:

```
package main
```

Essa linha informa que o nosso programa será o principal pacote da nossa aplicação, e que o nosso código deve começar por ele. Além disso, todo programa principal tem a função principal. As funções em Go são declaradas com `func`, seguida do seu nome e os argumentos entre parênteses, com o bloco de código ficando entre chaves:

```
package main

func main() {

}
```

Uma das características da função principal é não retornar nada e nem receber argumento nenhum. Quando executamos o programa, essa função é iniciada, e quando ela é finalizada, o programa termina.

Mas como imprimimos a mensagem de *Hello World*?

Exibindo uma mensagem

Se queremos imprimir uma mensagem, devemos utilizar a função `Println`, mas para utilizá-la, devemos importá-la, do pacote `fmt`. Esse é um pacote de formatação, possuindo funções de impressão e escaneamento do texto:

```
package main

import "fmt"

func main() {

}
```

Com o pacote importado, podemos utilizar a função:

```
package main

import "fmt"

func main() {
    fmt.Println()
}
```

O fato da função `Println` estar com a primeira letra maiúscula pode causar uma estranheza para quem vem de uma outra linguagem de programação. Quando estamos trabalhando com Go, a função que vem de um pacote externo, ou seja, uma função que não está declarada no nosso arquivo, é utilizada com a primeira letra maiúscula. Nós chamamos o pacote externo (no nosso caso, o `fmt`), e a função com a primeira letra maiúscula (no nosso caso, `Println`). Isso é uma convenção da linguagem, que faz com que saibamos que a função veio de um pacote externo

Agora, falta somente passarmos a mensagem que queremos exibir como argumento para a função `Println`:

```
package main

import "fmt"

func main() {
    fmt.Println("Olá Mundo")
}
```

Resta agora executar esse programa.

Executando um programa em Go

Como Go é uma linguagem compilada, para executar um programa seu, devemos compilá-lo para um executável, e para isso, nós devemos utilizar o terminal/linha de comando.

Nele, nós entramos dentro da pasta **go/src/hello** e executamos o comando `go build` seguido do nome do programa que queremos executar:

```
go build hello.go
```

Caso não dê nenhum erro, ao verificar o conteúdo da pasta, temos o seguinte:

```
alura@alura-01:~/go/src/hello$ ls
hello  hello.go
```

Ou seja, o executável foi criado. Agora basta executá-lo:

```
alura@alura-01:~/go/src/hello$ ./hello
Olá Mundo
```

Toda vez que alterarmos o nosso código, devemos compilá-lo para depois executá-lo, mas o Go é uma linguagem que facilita muita coisa para nós, e isso inclui o processo de compilação. Em vez de executar o `go build`, para depois rodar o executável gerado, podemos executar o comando `go run` seguido do nome do programa que queremos executar:

```
alura@alura-01:~/go/src/hello$ go run hello.go
Olá Mundo, alunos!!!
```

O comando `go run` compila e executa o programa de uma vez só! Isso evita com que tenhamos sempre que executar dois comandos para executar um simples programa.

Terminal embutido do Visual Studio Code

Por último, estamos trabalhando com dois programas, o Visual Studio Code e o terminal, mas o Visual Studio Code já possui um terminal embutido! No menu superior, acessando *View -> Integrated Terminal*.

Nele, nós podemos trabalhar do mesmo jeito como estávamos trabalhando antes, com a facilidade de ter tudo em uma mesma tela, de programar Go sem sair do Visual Studio Code.