

Strumenti per programmare in Go

I comandi Go

L'installazione di Go include vari strumenti per compilare, eseguire e formattare il codice, oltre ad altri strumenti di utilità.

- ▶ `go fmt`: formatta il codice di un singolo file o di un intero package
- ▶ `go run <programma.go>`: compila ed esegue un programma Go
- ▶ `go build`: compila un package creando un file eseguibile
- ▶ `go doc`: restituisce la documentazione relativa ad un package Go o ad una funzione
- ▶ `go help <comando>`: restituisce la documentazione relativa ad un comando Go

Go fmt

```
$ ls  
main.go
```

```
$ go fmt main.go
```

Codice prima della formattazione:

```
package main  
import "fmt"  
func main() {  
    fmt.Println("Hello world!")  
}
```

Codice dopo la formattazione

```
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello world!")  
}
```

Go doc

```
$ go doc fmt.Println  
package fmt // import "fmt"
```

```
func Println(a ...interface{}) (n int, err error)
```

Printf formats using the default formats for its operands and writes to standard output. Spaces are always added between operands and a newline is appended. It returns the number of bytes written and any write error encountered.

Go run

Comandi shell:

```
$ ls  
main.go
```

```
$ go run main.go  
Hello world!
```

Sorgente:

```
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello world!")  
}
```

Go build

Comandi shell:

```
$ ls  
main.go
```

```
$ go build -o Hello
```

```
$ ls  
Hello main.go
```

```
$ ./Hello  
Hello world!
```

Sorgente:

```
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello world!")  
}
```

Go help

```
$ go help build
```

```
usage: go build [-o output] [-i] [build flags] [packages]
```

Build compiles the packages named by the import paths, along with their dependencies, but it does not install the results.

If the arguments to build are a list of .go files from a single directory, build treats them as a list of source files specifying a single package.

When compiling packages, build ignores files that end in '_test.go'.

...

Esercizi

Esercizio 1

Usare gli strumenti `go run` e `go build` per eseguire il codice `helloworld.go`.

```
package main
```

```
import "fmt"
```

```
func main() {  
    fmt.Println("Hello World!")  
}
```

Esercizio 2

Utilizzate lo strumento `go fmt` per formattare in modo automatico il codice formattazione.go.

```
package main
import "fmt"
func main() {
    var x int
        x = 10
                var y int
y = 10
    fmt.Println(x+y    )
}
```

Esercizio 3

Lo strumento `go fmt` non è in grado di formattare il codice quando sono presenti degli errori. È quindi importante formattare il proprio codice durante la scrittura.

Correggere gli errori del codice nella prossima slide, formattarlo ed eseguirlo.

Esercizio 3 (codice)

```
package main

import fmt

func main {

    var x int
    x = 10
    var y int
    y = 15

    sum = x + y

    fmt.Println("La somma è ", sum

}

}
```