Les Pièges du Go 2

Michel Casabianca
casa@sweetohm.net

Le langage de programmation Go a la réputation d'être simple à apprendre. Cependant, il recèle quelques pièges qui peuvent être difficiles à détecter pour un novice. Cette série d'articles propose d'en désamorcer quelques uns.

Supposons que vous vouliez mettre en majuscule la première lettre du nom de `User`, comme ci-dessous (code sur le Playground) :

```
package main

import (
        "fmt"
        "strings"
)

type User struct {
        Name string
}

func UpperCase(users []User) {
        for _, user := range users {
                user.Name = strings.Title(user.Name)
        }
}

func main() {
        users := []User{{"foo"}, {"bar"}}
        UpperCase(users)
        fmt.Printf("users: %#v\n", users)
}
```

Si on exécute ce code, nous constatons qu'il ne fonctionne pas comme attendu :

```
$ go run broken.go
users: []main.User{main.User{Name:"foo"}, main.User{Name:"bar"}}
```

Pourquoi ce code ne fonctionne-t-il pas ? Comment peut-on le corriger ?

Explication

Ce code ne fonctionne pas comme attendu car à chaque boucle, nous recopions la structure du `User` et donc lorsque nous la modifions, l'original de l'est pas.

Nous pouvons corriger ce code de la manière suivante (code sur le Playground) :

```
package main

import (
        "fmt"
        "strings"
)

type User struct {
        Name string
}

func UpperCase(users []User) {
        for i := range users {
                users[i].Name = strings.Title(users[i].Name)
        }
}

func main() {
        users := []User{{"foo"}, {"bar"}}
        UpperCase(users)
        fmt.Printf("users: %#v\n", users)
}
```

Nous modifions alors l'original et donc le code fonctionne comme attendu :

```
$ go run fixed.go
users: []main.User{main.User{Name:"Foo"}, main.User{Name:"Bar"}}
```

Conclusion

Il faut faire attention au fait que lorsqu'on boucle avec `range`, nous effectuons une copie et donc si nous la modifions, l'original ne l'est pas.

Enjoy!