Les Pièges du Go 6

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.

Nous voudrions inverser un slice. Cela semble une tâche toute trouvée pour les Generics, donc nous écrivons le code suivant:

```
package main

import "fmt"

func reverse[S []any](s S) {
        for i, j := 0, len(s)-1; i < j; i, j = i+1, j-1 {
                s[i], s[j] = s[j], s[i]
        }
}

func main() {
        slice := []int{1, 2, 3}
        reverse(slice)
        fmt.Printf("%#v\n", slice)
}
```

Sur le Playground

Mais il ne compile pas, avec le message d'erreur suivant : `[]int does not implement []any` ! Pourquoi ce code ne compile-t-il pas ? Comment le corriger ?

Explication

Tout d'abord, il faut indiquer que les Generics ne sont pas la cause du problème qui est plus fondamental et se résume à la question suivante : pourquoi ne peut-on envoyer le type `[]int` à une fonction qui attend `[]any` (ou `[]interface{}`) ?

Nous allons faire une démonstration par l'absurde et supposer que l'on puisse passer `[]int` à la fonction `F([]interface{})` dans l'exemple suivant, qui devrait compiler :

```
package main

func F(v []interface{}) {
        v[0] = "Oops!"
}
```

```
func main() {
        v := []int{1, 2, 3}
        F(v)
        println(v[0])
}
```

Sur le Playground

Cet exemple ne compile pas avec le message d'erreur `cannot use v (variable of type []int) as type []interface{} in argument to F` et nous comprenons maintenant pourquoi : cela permettrait d'assigner, dans notre exemple, une chaîne dans un slice d'entiers.

Pour revenir à notre problème de départ, nous devons trouver un moyen de décrire le type du paramètre de notre fonction `reverse()` sans utiliser `[]any`.

Nous pouvons corriger le code comme suit :

```
package main

import "fmt"

func reverse[S []E, E any](s S) {
        for i, j := 0, len(s)-1; i < j; i, j = i+1, j-1 {
                s[i], s[j] = s[j], s[i]
        }
}

func main() {
        slice := []int{1, 2, 3}
        reverse(slice)
        fmt.Printf("%#v\n", slice)
}
```

Sur le Playground

Nous indiquons alors au compilateur que notre argument est un slice du type `E` et que le type `E` est quelconque. Le compilateur est content, CQFD !

Pour une explication plus détaillée, voir cet article (en anglais).

Enjoy!