Les Pièges du Go 5

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.

Les assertions sur les types peuvent être parfois obscures:

```
package main

import "fmt"

func main() {
    var data interface{} = "string"
    if cast, ok := data.(int); ok {
        fmt.Printf("%v is an int\n", cast)
    } else {
        fmt.Printf("%v is not an int\n", cast)
    }
}
```

Sur le Playground

Si nous exécutons ce code, nous obtenons:

```
$ go run broken.go
0 is not an int
```

Pourquoi cela ? Comment corriger ce code ?

Explication

L'expression `cast, ok := data.(int)` réalise une assertion de type. Ce qui implique que :

- Nous essayons de convertir `data` en `int`
- Si cela fonctionne, on affected à `cast` la valeur en `int` et `ok` vaut `true`
- Si cela ne fonctionne pas, nous affectons à `cast` la valeur nulle des `int` et `ok` vaut `false`

Par conséquent, comme `data` n'est pas un entier, `ok` vaut `false` et on affecte *0* à `cast`, donc on ne lui affecte pas la valeur chaîne de `data`

Nous pouvons corriger cela en réalisant un deuxième transtypage, comme suit :

```
package main

import "fmt"

func main() {
    var data interface{} = "string"
    if cast, ok := data.(int); ok {
        fmt.Printf("%v is an int\n", cast)
    } else {
                str := data.(string)
        fmt.Printf("%v is not an int\n", str)
    }
}
```

Sur le Playground

Enjoy!