Go Booby Traps 5

Michel Casabianca
casa@sweetohm.net

Go programming language is easy to learn, but there are some tricky traps. This article series is trying to show these booby traps so that you avoid them.

Type assertions might be a bit tricky sometimes:

```
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)
        }
}
```

On the Playground

When we run this, we get:

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

Why is that? How could we fix it?

Explanation

Expression `cast, ok := data.(int)` performs a type assertion. Which means:

- We try to cast `data` into an `int`
- If it works, `cast` is set with the `int` value and `ok` is `true`
- If it doesn't work, `cast` is set with `int` zero value and `ok` is `false`

Thus, as `data` is not an integer, `ok` is `false` and `cast` is set with *0* and it is not set with `string` value of data.

We can fix performing another cast, as follows:

```
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)
    }
}
```

On the Playground

Enjoy!