Go Booby Traps 2

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.

Let's say we want to upper case first letter of names in `User`, as follows (code on the 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)
}
```

If we run this code, we see that it doesn't work as expected:

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

Why is this code not working? How could we fix it?

Explanation

This code doesn't work as expected because in each loop, we copy struct `User` and thus when we modify it, original value is not updated.

We can fix it as follows (code on the 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)
}
```

We modify the original value and thus it works as expected:

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

Conclusion

You should be aware that when we loop with `range`, we copy values and thus if we modify them, the original one is not.

Enjoy!