
unit
test

Objectifs de l’atelier

Création d’un jeu de tests unitaires

Couverture du code et génération de documents

Création de benchmarks

```go
type myInt int

func (mi *myInt)Add(i int) {
      mi = mi / i
}
```

```go
func TestAdd(t *testing.T) {
     data := []struct{
        title  string
        value  myInt
        param  int
        should myInt
    }{
        {"A",1,1,2},
        {"B",2,1,3},
        {"C",9,1,10},
    }
    for _, v := range data {
      mi := v.value
      mi.Add(v.param)
      if mi != v.should{
        t.Error("for", v.title, "got", mi,
"should got", v.should)
      }
    }
}
```

Problème

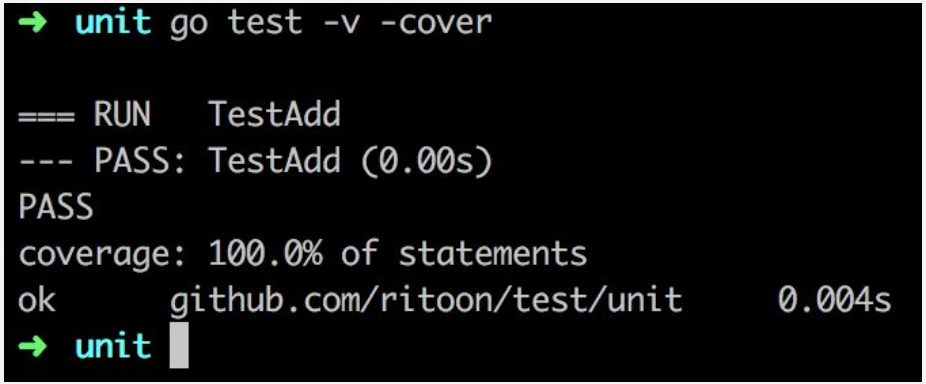
```
type myInt int

func (mi *myInt)Add(i int) {
    mi = mi / i
}
```

```
mi := myInt{6}
mi.Add(2) // ok sooo ?
```

commandes

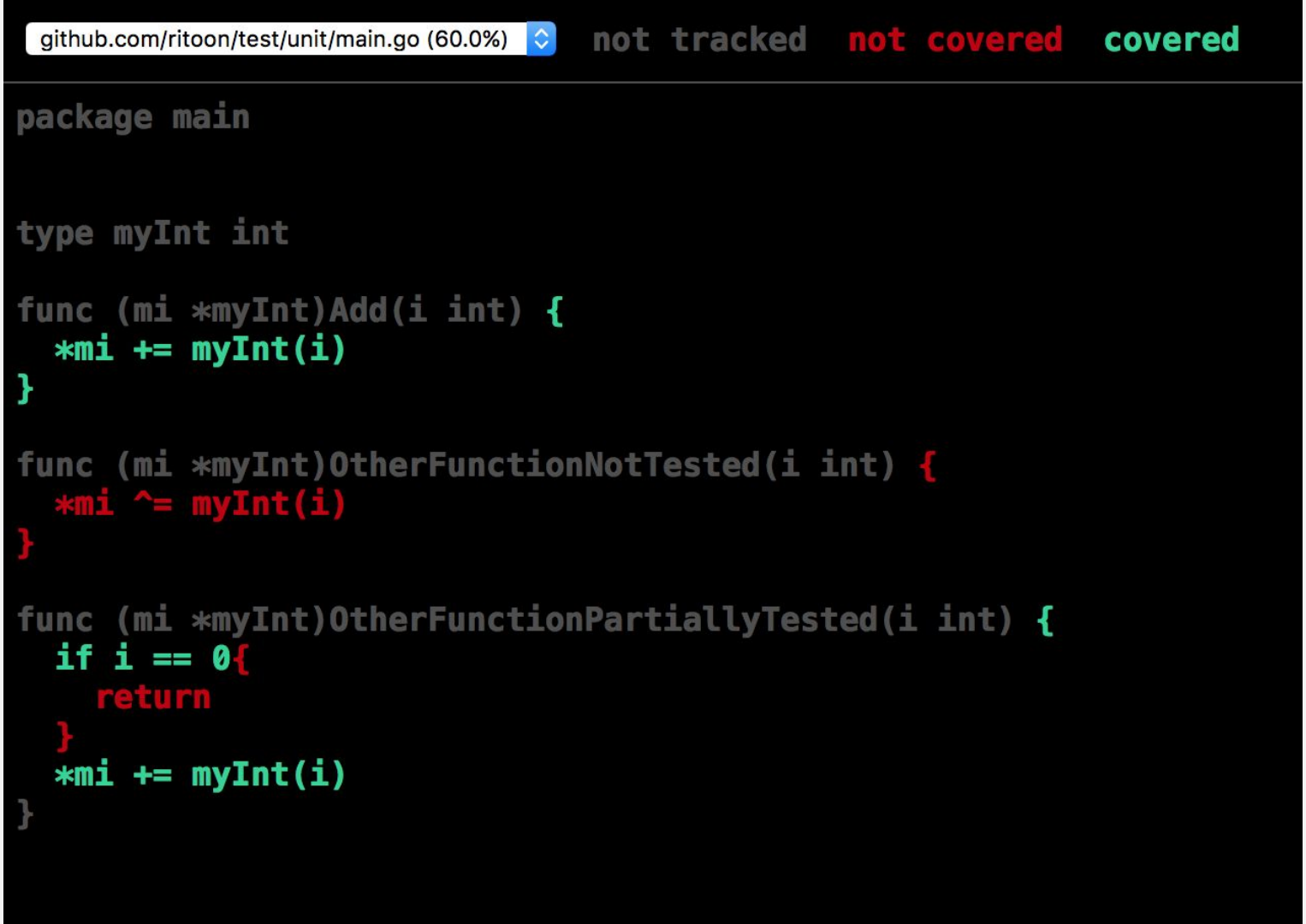
```
# test all code
go test -v -cover
# test only one function
go test -run TestAdd
```



Générer un rapport

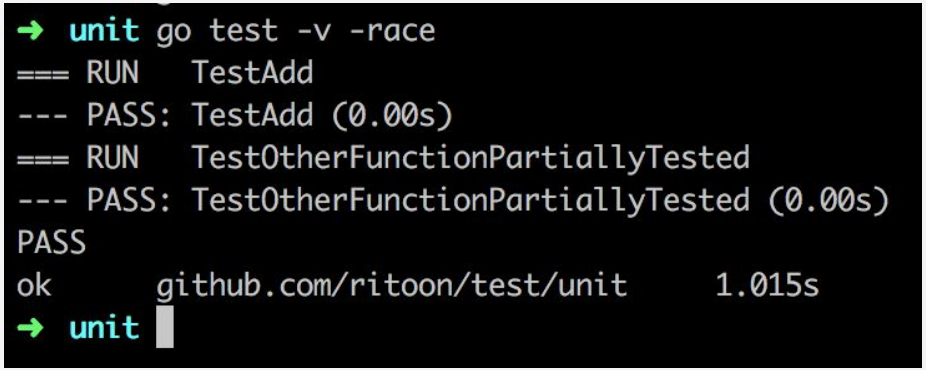
```
# générer un fichier html
go test -covermode=set -coverprofile=<package_name>.txt
go tool cover -html=<package_name>.txt -o <packge_name>.html

# créer un alias dans votre bash
alias test="go test -covermode=set -coverprofile=test.txt && go tool cover -html=test.txt
-o test.html && open test.html && rm test. txt"
```


github.com/ritoon/test/unit/main.go (60.0%)
not tracked
not covered
covered
package main
type myInt int
func (mi *myInt)Add(i int) {
*mi += myInt(i)
}
func (mi *myInt)OtherFunctionNotTested(i int) {
*mi ^= myInt(i)
}
func (mi *myInt)OtherFunctionPartiallyTested(i int) {
if i == 0{
return
}
*mi += myInt(i)
}

Concurrence

```
go test -v -race
```



```go
type myInt int

func (mi *myInt)Add(i int) {
  *mi += myInt(i)
  go Read(mi)
  go Write(mi)
}

func Read(m *myInt){
  _ = m
}

func Write(m *myInt){
  *m = 1
}
```

```
➜ unit go test -race
==================
WARNING: DATA RACE
Write at 0x00c420076468 by goroutine 8:
  github.com/ritoon/test/unit.Write()
      /Users/kong2/gocode/src/github.com/ritoon/test/unit/main.go:17 +0x3b

Previous read at 0x00c420076468 by goroutine 6:
  github.com/ritoon/test/unit.TestAdd()
      /Users/kong2/gocode/src/github.com/ritoon/test/unit/main_test.go:21 +0x1b5
  testing.tRunner()
      /usr/local/go/src/testing/testing.go:610 +0xc9

Goroutine 8 (running) created at:
  github.com/ritoon/test/unit.(*myInt).Add()
      /Users/kong2/gocode/src/github.com/ritoon/test/unit/main.go:9 +0xac
  github.com/ritoon/test/unit.TestAdd()
      /Users/kong2/gocode/src/github.com/ritoon/test/unit/main_test.go:20 +0x1a4
  testing.tRunner()
      /usr/local/go/src/testing/testing.go:610 +0xc9

Goroutine 6 (running) created at:
  testing.(*T).Run()
      /usr/local/go/src/testing/testing.go:646 +0x52f
  testing.RunTests.func1()
      /usr/local/go/src/testing/testing.go:793 +0xb9
  testing.tRunner()
      /usr/local/go/src/testing/testing.go:610 +0xc9
  testing.RunTests()
      /usr/local/go/src/testing/testing.go:799 +0x4ba
  testing.(*M).Run()
      /usr/local/go/src/testing/testing.go:743 +0x12f
  main.main()
      github.com/ritoon/test/unit/_test/_testmain.go:58 +0x1b8
==================
PASS
Found 1 data race(s)
exit status 66
FAIL    github.com/ritoon/test/unit     1.014s
➜ unit
```

Bench

```
go test -v -bench=.
go test -run=BenchmarkAdd -bench=.
```

```go
func BenchmarkAdd(b *testing.B) {
    var mi myInt
    for i := 0; i < b.N; i++ {
        mi.Add(1)
    }
}

func BenchmarkAddParallel(b *testing.B) {
    b.RunParallel(func(pb *testing.PB) {
        var mi myInt
        for pb.Next() {
            mi.Add(2)
        }
    })
}
```

```
➜ unit go test -bench=.
BenchmarkAdd-8                  2000000000               1.54 ns/op
BenchmarkAddParallel-8          2000000000               0.54 ns/op
PASS
ok      github.com/ritoon/test/unit     4.383s
➜ unit
```

Sub test add this !!!

https://blog.golang.org/subtests

Add optim

https://blog.codeship.com/real-life-go-benchmarking/

TP

Pour les fonctions avec myInt Créer les jeux de test pour avoir une couverture de code de 100%

Vous vous rappelez l'alias myInt c'est un int32

- **Divide**
- **Multiply**
- **Add**
- **Sub**

https://play.golang.org/p/XpygU6ceuY

utiliser la commande : go test -v -cover

Bonus : générez un fichier html pour afficher la couverture du code