Testing *Go* code

Tomasz Grodzki github.com/tg

AlphaSOC alphasoc.com

Create test file

Pattern: **_test.go*

```
package/unit.go
package/unit_test.go
```

Write test

```
package party

import "testing"

func TestBeer(t *testing.T) {
    if !FridgeContains("beer") {
        t.Fatal("expected some cold beer")
    }
}
```

Or... (from outside)

```
package party_test

import (
    "testing"
    "rihanna.com/party"
)

func TestBeer(t *testing.T) {
    if !party.FridgeContains("beer") {
        t.Fatal("expected some cold beer")
    }
}
```

Run

```
~$ go test rihanna.com/party

party$ go test .

parties$ go test ./...
```

Disappoint

```
$ go test

--- FAIL: TestBeer (0.00s)
    party_test.go:11: expected some cold beer
FAIL
FAIL rihanna.com/party 0.010s
```

Celebrate

```
$ go test rihanna.com/party
ok   rihanna.com/party 0.008s

$ go test rihanna.com/party/...
ok   rihanna.com/party 0.020s
ok   rihanna.com/party/dancefloor  0.019s
```

testing.T

```
Error        Fatal        Skip
Errorf       Fatalf       SkipNow
Fail         Log          Skipf
FailNow      Logf         Skipped
Failed       Parallel
```

Examples

```
func ExampleHandsUp() {
  fmt.Println(HandsUp("o oo o"))
  // Output:
  // \o/ \o/\o/ \o/
}
```

Naming convention

```
func Example()        // package
func ExampleF()       // function
func ExampleT()       // type
func ExampleT_M()     // method

func Example*_xyz()// more...
```

Documentation

```
$ go doc -ex=true .
[...]
func HandsUp(s string) string

    Example:
    fmt.Println(HandsUp("o oo o"))
    // Output:
    // \o/ \o/\o/ \o/
```

Benchmarks

```
func BenchmarkHeadSpin(b *testing.B) {
	for i := 0; i < b.N; i++ {
		HeadSpin()
	}
}
```

Run

```
$ go test -bench .
PASS
BenchmarkHeadSpin    10000         148475 ns/op
BenchmarkDrinkBeer       1   3354617382 ns/op
ok   rihanna.com/party   4.876s
```

testing.B

```
testing.T  +    ReportAllocs
                ResetTimer
                RunParallel
                SetBytes
                SetParallelism
                StartTimer
                StopTimer
```

More control

```go
// since go 1.4
func TestMain(m *testing.M) {
   os.Exit(m.Run())
}
```

Packages

- `net/http/httptest`
- `testing/iotest`
- `testing/quick`

net/http/httptest

- ResponseRecorder
  - implements http.*ResponseWriter*
  - captures *Code, HeaderMap, Body*
- Server
  - listens on loopback interface
  - exposes *URL* for http.Get

httptest.ResponseRecorder

```go
handler := func(w http.ResponseWriter, r *http.Request) {
    http.Error(w, "Uh huh", http.StatusBadRequest)
}
r, err := http.NewRequest("GET", "http://test.com", nil)

w := httptest.NewRecorder()
handler(w, r) // handle request, store result in w
if w.Code != http.StatusOK {
    t.Fatal(w.Code, w.Body.String())
}
```

httptest.Server

```go
hlr := func(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintln(w, "Ella ella eh eh eh")
}
ts := httptest.NewServer(http.HandlerFunc(hlr))
defer ts.Close()

res, err := http.Get(ts.URL) // e.g. http://127.0.0.1:6301
greeting, err := ioutil.ReadAll(res.Body)
```

testing/iotest

`DataErrReader(r io.Reader) io.Reader`

- New reader behaves like *r*, but an error (typically *io.EOF*) will be reported along with the last data chunk.

`TimeoutReader(r io.Reader) io.Reader`

- New reader will return *iotest.ErrTimeout* on the second read (with no data). Subsequent reads succeed.

testing/iotest

`HalfReader(r io.Reader) io.Reader`

- New reader reads up to half requested bytes.

`OneByteReader(r io.Reader) io.Reader`

- New reader reads up to one byte each time.

`TruncateWriter(w io.Writer, n int64) io.Writer`

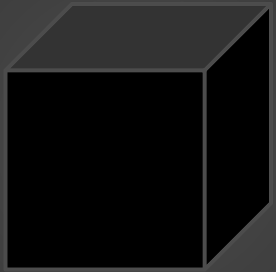
- New writer writes to *w*, but stops silently after *n* bytes.

testing/iotest

```
NewReadLogger(p string, r io.Reader) io.Reader
NewWriteLogger(p string, w io.Writer) io.Writer
```

- Return new reader/writer, which log all reads/writes (using *log.Print*) to standard error, printing the prefix *p* and hexadecimal data read/written.

testing/quick



testing/quick

- Black box testing
- Generates random test cases
- quick.*Check* and quick.*CheckEqual*
- Inspired by QuickCheck for Haskell (paper by John Hughes)

quick.Check

```
func TestIntToStrToInt(t *testing.T) {
    f := func(x int) bool {
        return x == ToInt(ToStr(x))
    }
    if err := quick.Check(f, nil); err != nil {
        t.Error(err)
    }
}
```

quick.Check

```
type OddInt int

func (x OddInt) Generate(r *rand.Rand, size int) reflect.Value {
    return reflect.ValueOf(OddInt(r.Int() | 1))
}

func TestOddMod2(t *testing.T) {
    f := func(x OddInt) bool {
        return Mod(int(x), 2) == 1
    }
    if err := quick.Check(f, nil); err != nil {
        t.Error(err)
    }
}
```

quick.CheckEqual

```
func IntToStr(x int) string {
    return "42"
}

func TestIntToStr(t *testing.T) {
    err := quick.CheckEqual(IntToStr, strconv.Itoa, nil);
    if err != nil {
        t.Error(err)
    }
}

// failed on input 4106209714314777601.
// Output 1: "42". Output 2: "4106209714314777601"
```

quick.Config

```go
type Config struct {
   // Set max number of iterations
   MaxCount int
   // Scale max number of iterations
   MaxCountScale float64
   // Source of random numbers
   Rand *rand.Rand
   // Generator of values
   Values func([]reflect.Value, *rand.Rand)
}
```

Other tools

- go vet
  - checks for common mistakes
- *-race* flag
  - enables data race detector
  - works with go build, install, run, test
- interfaces
  - e.x. use io.*Reader* instead of os.*File*

Test profiles

- Write profile files for external analysis:
    - go test -coverprofile *cover.out*
    - go test -cpuprofile *cpu.out*
    - go test -memprofile *mem.out*

pprof

```
$ go test -c && ./party.test -test.cpuprofile=cpu.pro -test.bench=.

$ go tool pprof -top party.test cpu.pro

390ms of 1340ms total (29.10%)
   flat  flat%   sum%        cum   cum%
   200ms 14.93% 14.93%     1310ms 97.76%  rihanna.com/party.HeadSpin
   190ms 14.18% 29.10%      190ms 14.18%  math/rand.(*rngSource).Int63
   100ms  7.46% 36.57%      150ms 11.19%  runtime.mallocgc
    90ms  6.72% 43.28%       90ms  6.72%  runtime.memeqbody
    90ms  6.72% 50.00%      250ms 18.66%  runtime.rawstring
[...]

// Broken on OSX: https://github.com/golang/go/issues/6047
```

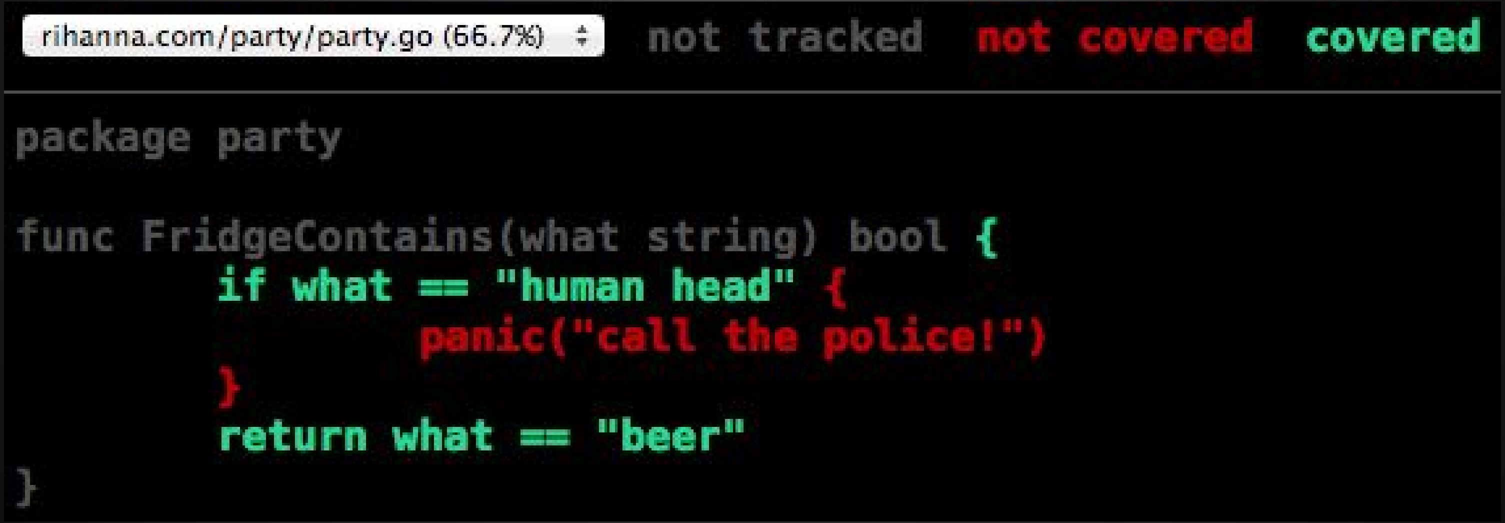
cover

```
$ go test -cover

PASS
coverage: 66.7% of statements
ok  rihanna.com/party 0.009s
```

cover

```
$ go test -coverprofile=c.pro
$ go tool cover -html=c.pro
```



```go
package party

func FridgeContains(what string) bool {
	if what == "human head" {
		panic("call the police!")
	}
	return what == "beer"
}
```

References

golang.org/pkg/testing
blog.golang.org

```
go help test
go help testflag
```