Тестирование Go кода с GoUnit and Minimock

Max Chechel
hexdigest@gmail.com
@maxchechel

~ 10% кода это бизнес-логика
всё остальное это тесты

Это очень много кода

Тестирование Go кода подразумевает

- Определение зависимостей через интерфейсы
- Подмена зависимостей моками
- Написание тестов
- Запуск тестов

Определение зависимостей через интерфейсы

```
type Logger interface {
  Log(level, format string, args ...interface{}) error
}

type Service struct {
  Logger Logger
}
```

Подмена зависимостей моками

```
type Logger interface {
  Log(level, format string, args ...interface{}) error
}

type Service struct {
  Logger Logger
}


service := Service{
  Logger: NewLoggerMock(t).LogMock.
      Expect("debug", "processed request: %s", "Hello Minsk!").
      Return(nil),
}
```

Генерация моков

github.com/gojuno/minimock

- Интегрирован со стандартным пакетом “testing”
- Реализует Wait(time.Duration) и Finish() хелперы для для проверки, что все моки были использованы в тесте
- Счётчики вызова моков
- Генерация нескольких моков за один вызов minimock

GoUnit - утилита для генерации тестов

github.com/hexdigest/gounit

Утилита генерирующая table driven test по сигнатуре функции.

Есть плагины для

- Vim
- Emacs
- Atom
- Sublime

Возможность управления шаблонами для тестов

Почему не GoTests?

github.com/cweill/gotests

Много багов!

- Ищет совпадение по имени функции/метода. Если в файле 2 метода имеют одинаковое имя будут сгенерированы тесты для обоих
- Нет разделения потоков вывода (Stdout/Stderr)
- Нет exit-кодов
- Нет механизма сообщить редактору в какое место поставить курсор в сгенерированном тесте
- Не было поддержки шаблонов
- Плохой дефолтный шаблон теста

Плохой шаблон в GoTests

```
func TestService_Echo(t *testing.T) {
    type fields struct {
        Logger Logger // ← как проверить что передано в логгер в момент выполнения?
    }
    type args struct {
        w http.ResponseWriter  // ← как проверить содержимое после выполнения
метода?
        r *http.Request
    }
    tests := []struct {
        name    string
        fields  fields
        args    args
        wantErr bool
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
```

Хороший шаблон в GoUnit

```
$ go get github.com/hexdigest/gounit/cmd/gounit

$ curl https://raw.githubusercontent.com/hexdigest/gounit/master/templates/minimock > /tmp/minimock && \
  gounit template add -f /tmp/minimock && \
  gounit template use -n 2 && rm /tmp/minimock
```

Хороший шаблон в GoUnit

```
func TestService_Echo(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name       string
        init       func(t minimock.Tester) Service
        inspect    func(r Service, t *testing.T) //inspects r after execution
        args       func(t minimock.Tester) args
        wantErr    bool
        inspectErr func(err error, t *testing.T)//use for precise error evaluation
    }{
        /* TODO: Add test cases */
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            mc := minimock.NewController(t)
            defer mc.Wait(time.Second)
            tArgs := tt.args(mc)
            receiver := tt.init(mc)
```

Давайте напишем немного кода.