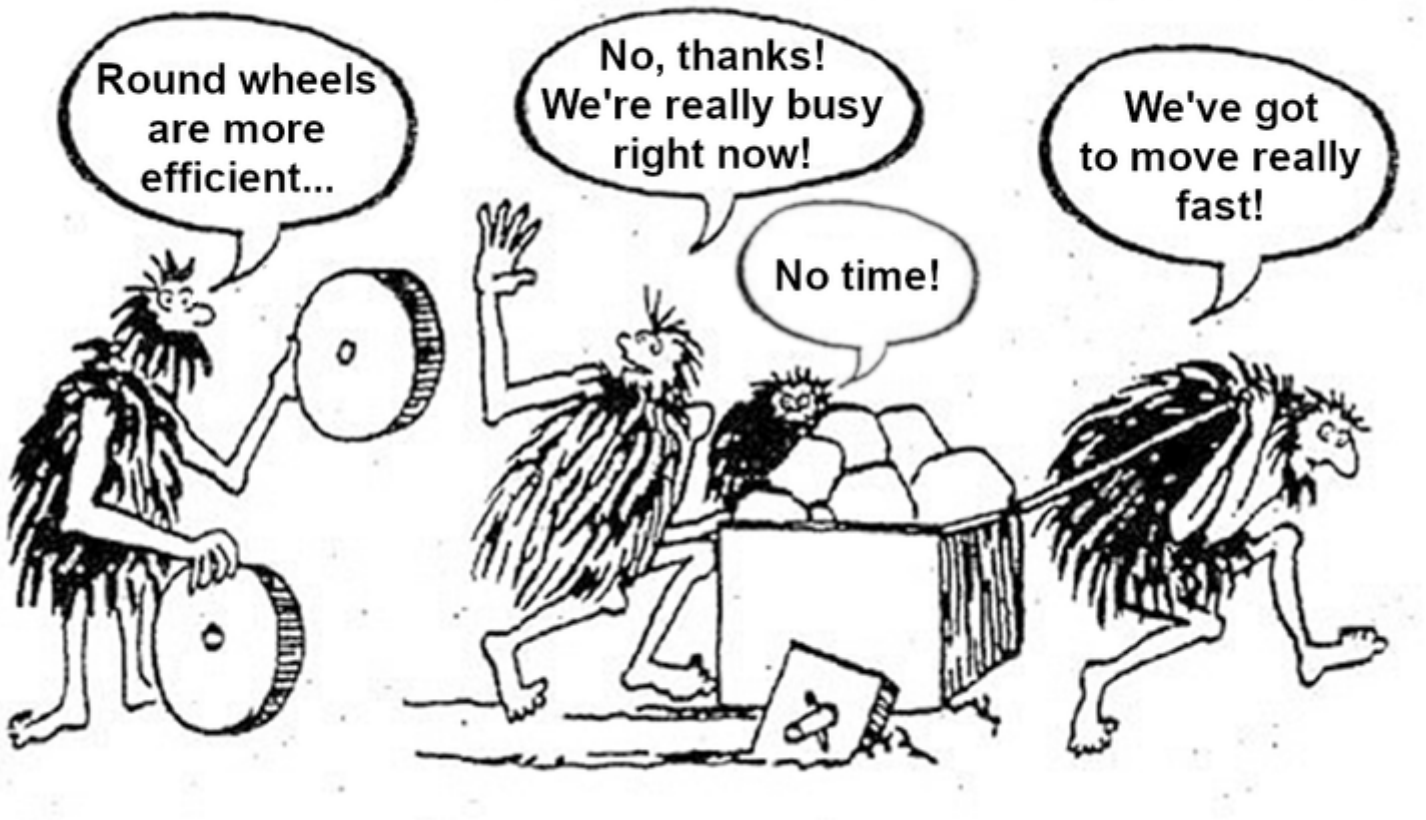
Code generation in Go.

Tips & Tricks.

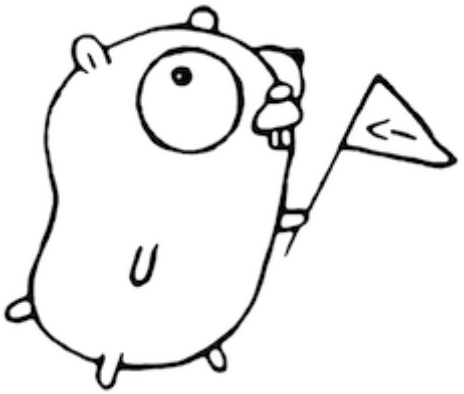
28 November 2019

Stanislav Afanasev
Gopher, MessageBird


Round wheels are more efficient...
No, thanks! We're really busy right now!
No time!
We've got to move really fast!

Agenda

- Mocks
- Wrappers



Mocks. The problem.

Mocks. The problem.

1. UserStorage interface.

```
type UserStorage interface {
    User(ctx context.Context, ID int) (*User, error)
    Create(ctx context.Context, user *User) error
    Update(ctx context.Context, user *User) error
    Delete(ctx context.Context, ID int) error
}
```

2. Service.

```
type Service struct {
    // ...
    userStorage UserStorage
    // ...
}

func NewService(u UserStorage) *Service {
    return &Service{
        userStorage: u,
    }
}
```

Mocks. The problem#0

```go
type UserStorageMock struct{}

func (m UserStorageMock) User(_ context.Context, _ int) (*User, error) {
    return &User{}, nil
}

func (m UserStorageMock) Create(_ context.Context, _ *User) error {
    return nil
}

func (m UserStorageMock) Update(_ context.Context, _ *User) error {
    return nil
}

func (m UserStorageMock) Delete(_ context.Context, _ int) error {
    return nil
}
```

~ 1 minute

Mocks. The problem#1

```
type UserStorageMock struct {
    t        *testing.T
    UserMock func(ctx context.Context, id int) (*User, error)
    //...
}

func (m UserStorageMock) User(ctx context.Context, id int) (u *User, err error) {
    if m.UserMock != nil {
        return m.UserMock(ctx, id)
    }

    m.t.Fatal("Unexpected call of UserStorageMock.User")
    return
}
```

Mocks. The problem#2

```
    userStorageMock := UserStorageMock{
        UserMock: func(ctx context.Context, id int) (*User, error) {
            // ...
            return nil, errors.New("something is wrong")
        },
    }

    service := NewService(userStorageMock)
    // ...
```

~ 10 minutes

Mocks. The problem#3

```
type UserStorageMock struct {
    t               *testing.T
    UserMock        func(ctx context.Context, id int) (*User, error)
    UserMockCounter uint64
    //...
}

func (m UserStorageMock) User(ctx context.Context, id int) (u *User, err error) {
    if m.UserMock != nil {
        atomic.AddUint64(&m.UserMockCounter, 1)
        return m.UserMock(ctx, id)
    }

    m.t.Fatal("Unexpected call of UserStorageMock.User")
    return
}
```

~ 4 minutes

Mocks. This is a real pain 😭

```
type Interface interface {
    Discovery() discovery.DiscoveryInterface
    AdmissionregistrationV1() admissionregistrationv1.AdmissionregistrationV1Interface
    AdmissionregistrationV1beta1() admissionregistrationv1beta1.AdmissionregistrationV1beta1Interface
    AppsV1() appsv1.AppsV1Interface
    AppsV1beta1() appsv1beta1.AppsV1beta1Interface
    AppsV1beta2() appsv1beta2.AppsV1beta2Interface
    AuditregistrationV1alpha1() auditregistrationv1alpha1.AuditregistrationV1alpha1Interface
    AuthenticationV1() authenticationv1.AuthenticationV1Interface
    AuthenticationV1beta1() authenticationv1beta1.AuthenticationV1beta1Interface
    AuthorizationV1() authorizationv1.AuthorizationV1Interface
    AuthorizationV1beta1() authorizationv1beta1.AuthorizationV1beta1Interface
    AutoscalingV1() autoscalingv1.AutoscalingV1Interface
    AutoscalingV2beta1() autoscalingv2beta1.AutoscalingV2beta1Interface
    AutoscalingV2beta2() autoscalingv2beta2.AutoscalingV2beta2Interface
    BatchV1() batchv1.BatchV1Interface
    BatchV1beta1() batchv1beta1.BatchV1beta1Interface
    // +24 methods below
}
```

⏰ *~ I'm crying hours*

github.com/kubernetes/client-go/blob/master/kubernetes/clientset.go#L68

(https://github.com/kubernetes/client-go/blob/master/kubernetes/clientset.go#L68)

Mocks. Generate it with Minimock

- Expect/Inspect; When/Then helpers
- Counters
- Finish
- It puts *go generate* instruction to mocks!

github.com/gojuno/minimock (https://github.com/gojuno/minimock)

Minimock. Example

io.ReadCloser

```
// ReadCloser is the interface that groups the basic Read and Close methods.
type ReadCloser interface {
    Reader
    Closer
}
```

CustomReader

```
type CustomReader struct {
    io.ReadCloser
}

func (r CustomReader) Read(p []byte) (int, error) {
    return r.ReadCloser.Read(p)
}
```

Usage

```
minimock -i io.ReadCloser
```

Minimock. Expect/Inspect/Return

Checking arguments

```
readCloserMock := NewReadCloserMock(t).
    ReadMock.Expect([]byte{0, 0, 0}).Return(3, nil).
    CloseMock.Return(nil)

reader := CustomReader{readCloserMock}

n, err := reader.Read(make([]byte, 3))
// ReadCloserMock.Read got unexpected parameters, want ...

reader.Close()
// Unexpected call to ReadCloserMock.Close ...
```

Minimock. When/Then

Usage of When/Then helpers

```
readCloserMock := NewReadCloserMock(t).
    ReadMock.When([]byte{}).Then(0, io.EOF).
    ReadMock.When([]byte{0}).Then(1, nil).
    ReadMock.When([]byte{0, 0, 0}).Then(3, nil)

reader := CustomReader{readCloserMock}

n, err := reader.Read([]byte{})
// err == io.EOF
```

- One-line syntax is allowed (that's 🔥 for table tests)

Minimock. Finish

Checks whether all mocks are being used

```
mc := minimock.NewController(t)
defer mc.Finish()

stringerMock := NewStringerMock(mc)
stringerMock.StringMock.Return("minimock")
// Expected call to StringerMock.String

readCloserMock := NewReadCloserMock(mc)
readCloserMock.ReadMock.Return(5, nil)
```

Minimock is nice

- Expect/Inspect; When/Then helpers
- Counters
- Finish
- It puts *go generate* instruction to mocks!

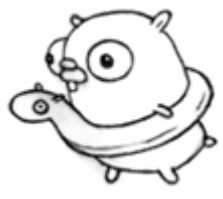
github.com/gojuno/minimock (https://github.com/gojuno/minimock)

Alternatives

github.com/golang/mock (https://github.com/golang/mock)

github.com/matryer/moq (https://github.com/matryer/moq)

github.com/vektra/mockery (https://github.com/vektra/mockery)



Wrappers

Wrappers. The problem.

UserStorage interface.

```
// Sender represents any payload sender.
type Sender interface {
    Send(ctx context.Context, to string, payload []byte) error
}
```

Example
- Several implementations: Push and Email.
- Instrumenting with tracing.

```
smsSender := sms.New()
emailSender := email.New()

sender := sender.New(
    sender.NewSenderWithTracing(smsSender, "SMS"),
    sender.NewSenderWithTracing(emailSender, "Email"),
)
```

Wrappers. SenderWithTracing.

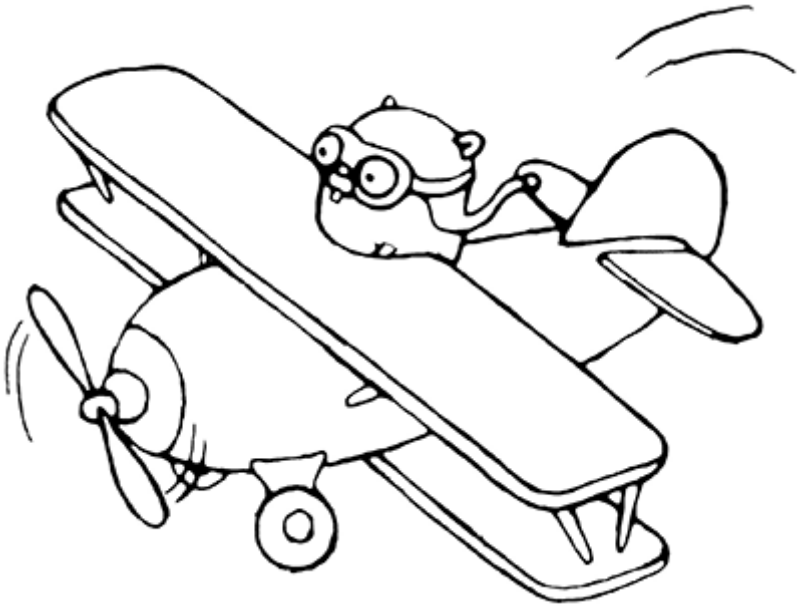
What I'd want to get

```
// SenderWithTracing implements Sender interface instrumented with opencencus
type SenderWithTracing struct {
    name string
    Sender
}

// NewSenderWithTracing returns SenderWithTracing
func NewSenderWithTracing(s Sender, name string) SenderWithTracing {
    return SenderWithTracing{
        name:   name,
        Sender: s,
    }
}

// Send implements Sender
func (s SenderWithTracing) Send(ctx context.Context, to string, payload []byte) (err error) {
    ctx, span := trace.StartSpan(ctx, s.name+".Sender.Send",
        trace.WithSpanKind(trace.SpanKindClient))
    defer span.End()

    return s.Sender.Send(ctx, to, payload)
}
```

Gowrap.

- A lot of prepared templates (e.g., fallback, logrus, prometheus, etc.)
- Custom templates
- It puts *go generate* instruction to generated decorators

github.com/hexdigest/gowrap (https://github.com/hexdigest/gowrap)



Gowrap. Templating

```
{{ $decorator := (or .Vars.DecoratorName (printf "%sWithTracing" .Interface.Name)) }}
// {{$decorator}} implements {{.Interface.Type}} interface instrumented with opencencus
type {{$decorator}} struct {
  _name string
  {{.Interface.Type}}
}

// New{{$decorator}} returns {{$decorator}}
func New{{$decorator}} (name string, base {{.Interface.Type}}) {{$decorator}} {
  return {{$decorator}} { _name: name, {{.Interface.Name}}: base}
}

{{range $method := .Interface.Methods}}
  // {{$method.Name}} implements {{$.Interface.Type}}
  func (_d {{$decorator}}) {{$method.Declaration}} {
    {{if $method.AcceptsContext}}
        ctx, span := trace.StartSpan(ctx, _d._name + ".{{$.Interface.Type}}.{{$method.Name}}",
    trace.WithSpanKind(trace.SpanKindClient))
        defer span.End()
    {{end}}
    {{$method.Pass (printf "_d.%s." $.Interface.Name) }}
  }
{{end}}
```

Gowrap. Templating is easy.

Gowrap. Templating

Type definition

```
{{ $decorator := (or .Vars.DecoratorName (printf "%sWithTracing" .Interface.Name)) }}
// {{$decorator}} implements {{.Interface.Type}} interface instrumented with opencencus
type {{$decorator}} struct {
  _name string
  {{.Interface.Type}}
}
```

Constructor

```
// New{{$decorator}} returns {{$decorator}}
func New{{$decorator}} (base {{.Interface.Type}}, name string) {{$decorator}} {
  return {{$decorator}} { _name: name, {.Interface.Name}}: base }
}
```

Gowrap. Templating

Methods

```
{{range $method := .Interface.Methods}}
  // {{$method.Name}} implements {{$.Interface.Type}}
  func (_d {{$decorator}}) {{$method.Declaration}} {
    {{if $method.AcceptsContext}}
        ctx, span := trace.StartSpan(ctx, _d._name + ".{{$.Interface.Type}}.{{$method.Name}}",
    trace.WithSpanKind(trace.SpanKindClient))
        defer span.End()
    {{end}}
    {{$method.Pass (printf "_d.%s." $.Interface.Name) }}
  }
{{end}}
```

Usage

```
gowrap gen -p . -i Sender -t tracing_template -o sender_tracing.go
```

Gowrap. Templating

Generated code

```
// DO NOT EDIT!
// This code is generated with http://github.com/hexdigest/gowrap tool
// using tracing_template template

//go:generate gowrap gen -p talk/code/gowrap -i Sender -t tracing_template -o sender_tracing.go
```

blog.golang.org/generate (https://blog.golang.org/generate)

Gowrap. Templating

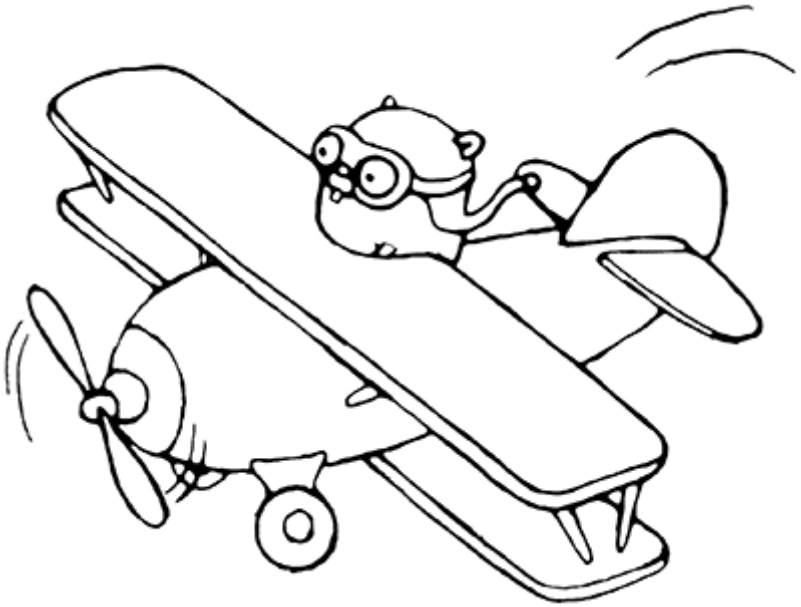
Generated code

```
// SenderWithTracing implements Sender interface instrumented with opencencus
type SenderWithTracing struct {
    _name string
    Sender
}

// NewSenderWithTracing returns SenderWithTracing
func NewSenderWithTracing(name string, base Sender) SenderWithTracing {
    return SenderWithTracing{_name: name, Sender: base}
}

// Send implements Sender
func (_d SenderWithTracing) Send(ctx context.Context, to string, payload []byte) (err error) {

    ctx, span := trace.StartSpan(ctx, _d._name+".Sender.Send",
        trace.WithSpanKind(trace.SpanKindClient))
    defer span.End()

    return _d.Sender.Send(ctx, to, payload)
}
```

Gowrap!

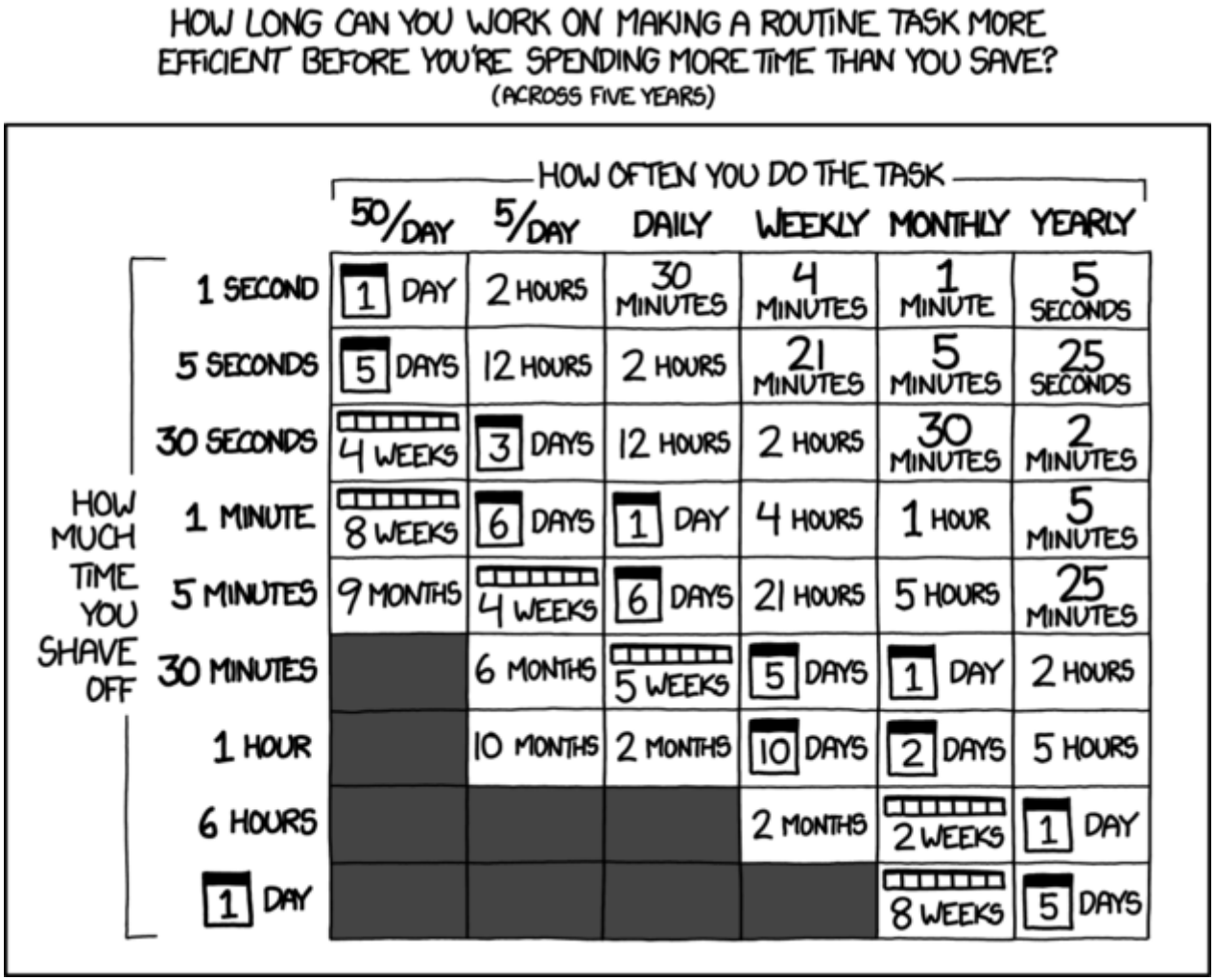
- A lot of prepared templates (e.g., fallback, logrus, prometheus, etc.)
- Custom templates
- It puts *go generate* instruction to generated decorators

github.com/hexdigest/gowrap (https://github.com/hexdigest/gowrap)



Do you need it?

Do you need it?



xkcd.com/1205/ (https://xkcd.com/1205/)

Conclusion

- Don't waste your time doing it manually.
- Generate as much as you can.

What else? - Links

- Genval

It generates boilerplate for **Validate** method

github.com/gojuno/genval (https://github.com/gojuno/genval)

- Prep

It generates SQL statements and prepares it.

github.com/hexdigest/prep (https://github.com/hexdigest/prep)

- Table tests

github.com/cweill/gotests (https://github.com/cweill/gotests)

github.com/hexdigest/gounit (https://github.com/hexdigest/gounit)

Thank you

Stanislav Afanasev
Gopher, MessageBird
@superstas88 (http://twitter.com/superstas88)