ThoughtWorks®

Introdução + Coding Dojo

TDD EM GO

Prática, ferramentas e bibliotecas para
desenvolvimento guiado por testes em #golang

Luciano Ramalho | @standupdev | @ramalhoorg

Repositório com examplos e slides: https://tgo.li/tddgo

MENSAGEM DA LAUREN DO @WWGSAMPA PARA VOCÊS

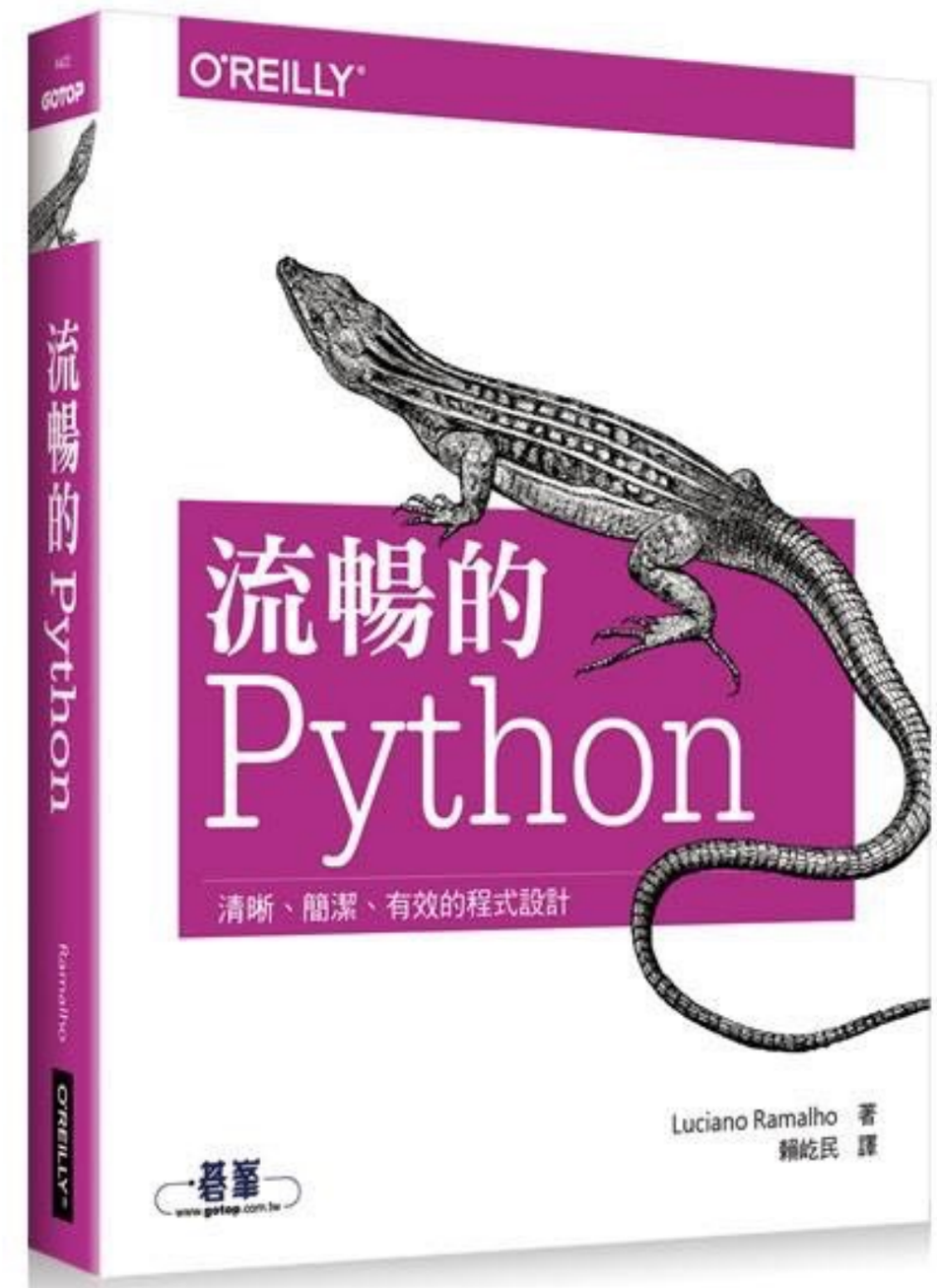
ThoughtWorks®

LUCIANO RAMALHO

Technical Principal

@ramalhoorg
luciano.ramalho@thoughtworks.com

FLUENT PYTHON, MEU 1º LIVRO



Fluent Python (O'Reilly, 2015)

Python Fluente (Novatec, 2015)

Python к вершинам мастерства (DMK, 2015)

流暢的 Python (Gotop, 2016)

also in **Simplified Chinese, Polish, Korean...**

87

4.6 out of 5 stars

5 star	85%
4 star	4%
3 star	5%
2 star	3%
1 star	3%

Mais de 40,000 exemplares vendidos até maio de 2018

IDEIA DESTE TUTORIAL

Aproveitar ao máximo o tempo presencial

Prática de TDD, juntas

Aplicação simples e completa, com testes

Discussão sobre estilos e variantes de TDD

Visão panorâmica sobre ferramentas e bibliotecas

Muitas referências para estudar mais

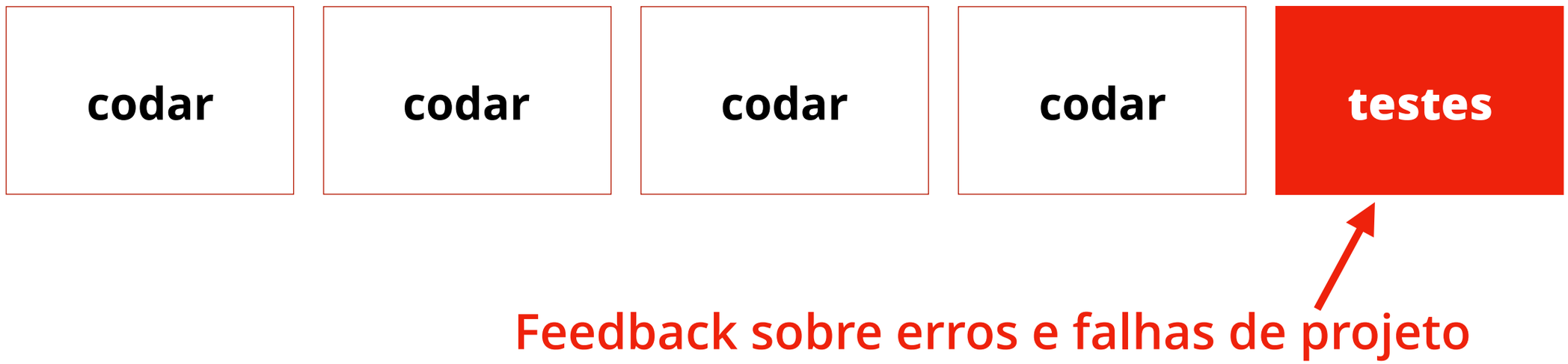
Repositório com examplos e slides: https://tgo.li/tddgo

ThoughtWorks®

INTRODUÇÃO

O que é TDD

TESTES TRADICIONAIS

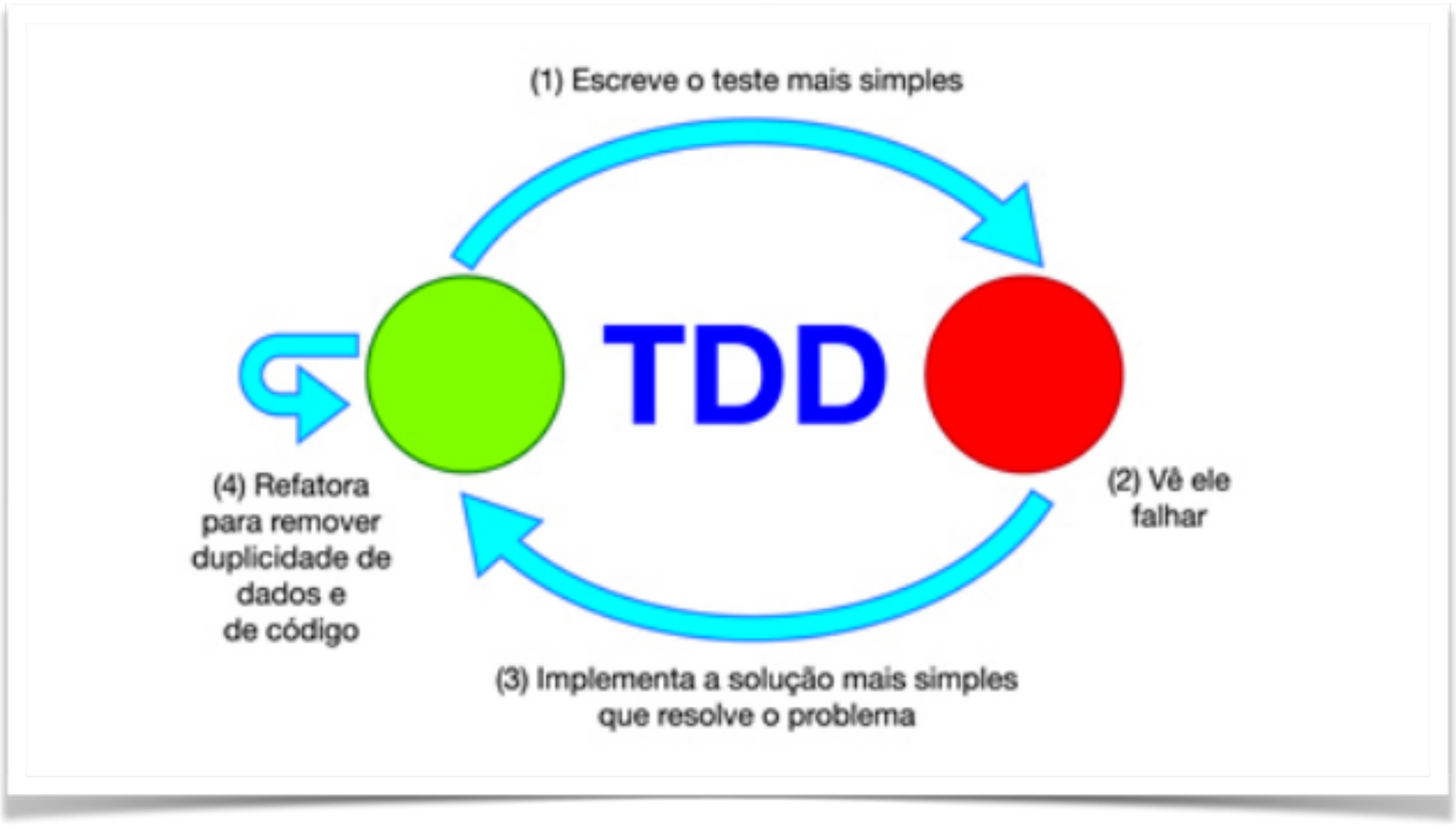


Fonte: **Test-Driven Development** – Hugo Corbucci & Mauricio Aniche

TEST-DRIVEN DEVELOPMENT

Fonte: **Test-Driven Development** – Hugo Corbucci & Mauricio Aniche

TDD CYCLE



Fonte: **Test-Driven Development** – Hugo Corbucci & Mauricio Aniche

BOA PRÁTICA: BABY STEPS

Implementar em pequenos incrementos

Tempo entre estados vermelho/verde medido em minutos, não horas

No início: pratique com os menores incrementos que consegue imaginar

Como engatar a 1ª reduzida: esteja disposta e pronta para engatar marcha reduzida quando a subida for íngreme.

BOA PRÁTICA: CANTAR TACADA

Antes de rodar o teste, “cante” o resultado esperado.

Ao parear, co-pilota deve cantar o resultado.

— Teste vai gerar erro porque `parseLine` não foi definida

— Teste vai falhar porque função devolve **1**, mas o resultado esperado é **42**.

— Teste vai passar.

Fonte: **Test-Driven Development** – Hugo Corbucci & Mauricio Aniche

BOA PRÁTICA: MELHORAR MENSAGEM DE FALHA

Figure 5.2 *Improving the diagnostics as part of the TDD cycle*

Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

ThoughtWorks®

CODING DOJO

Mão na massa!

CODING DOJO: REGRAS PARA A PRÁTICA RANDORI

Rotação de duplas: pilota + co-pilota.

Após 7 minutos, nova co-pilota voluntária.

Quando os testes estão verdes, todas podem opinar sobre o próximo teste.

Quando um teste está falhando, platéia só deve se manifestar a pedido da dupla.

NOSSA META

```
$ runes cat eyes
U+1F638 😸      GRINNING CAT FACE WITH SMILING EYES
U+1F63B 😻      SMILING CAT FACE WITH HEART-SHAPED EYES
U+1F63D 😽      KISSING CAT FACE WITH CLOSED EYES
$
```

TESTE-EXEMPLO SIMPLES

Uma função Example()
em um arquivo
*_test.go é um teste.

```
func Example() {
        main()
        // Output:
        // Please provide one or more words to search.
}
```

Saída do programa é
comparada ao conteúdo
do comentário
// Output:

PACOTE TESTING: API DO TIPO T

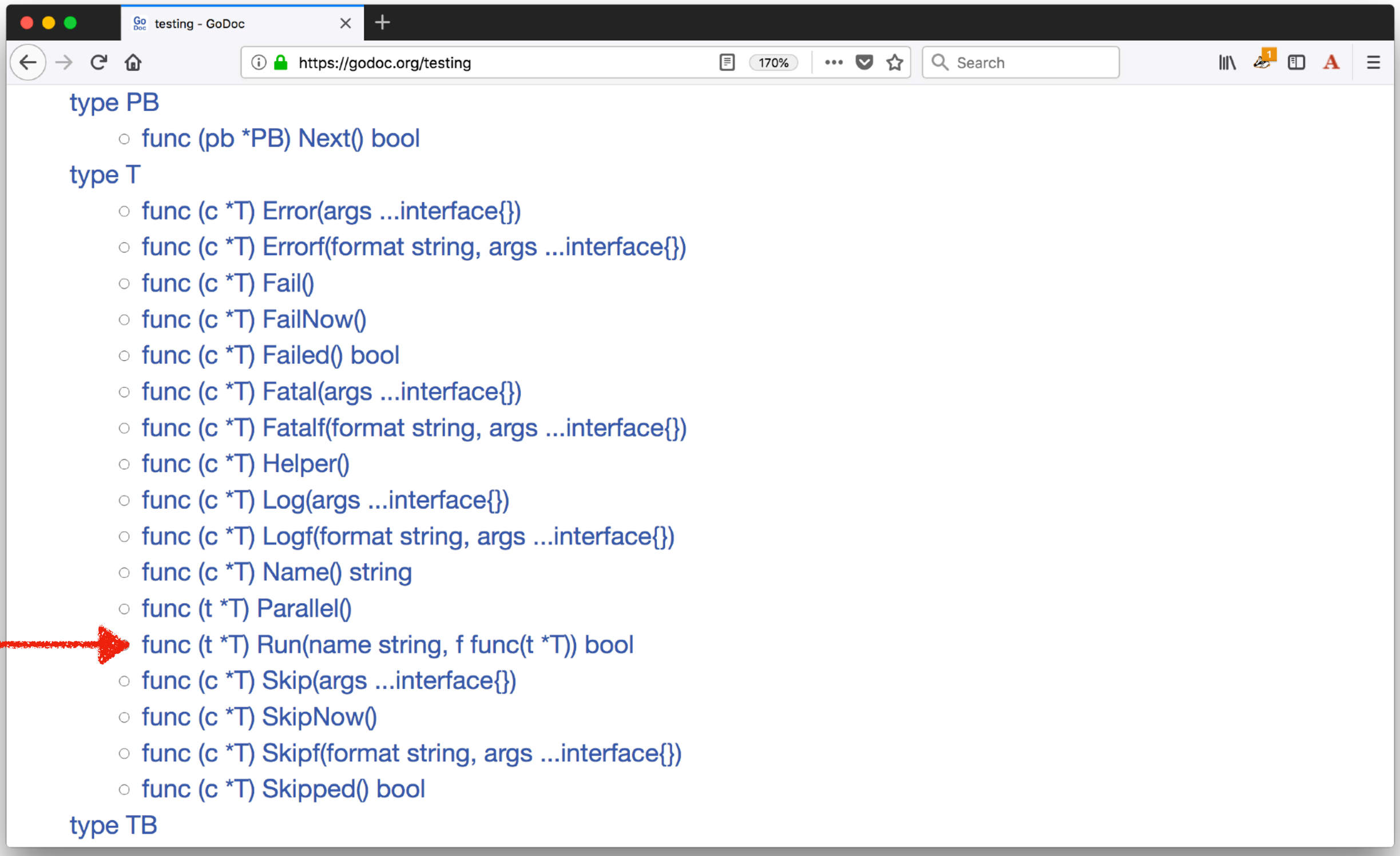


TABLE TEST COM SUB-TESTS

```go
func TestMake(t *testing.T) {
        testCases := []struct {
                elems   []string
                wantLen int
        }{
                {[]string{}, 0},
                {[]string{"a"}, 1},
                {[]string{"a", "b"}, 2},
                {[]string{"a", "b", "a"}, 2},
        }
        for _, tc := range testCases {
                t.Run(fmt.Sprintf("%v gets %d", tc.elems, tc.wantLen), func(t *testing.T) {
                        s := Make(tc.elems...)
                        assert.Equal(t, tc.wantLen, s.Len())
                })
        }
}
```

t.Run() é a forma atual para executar testes de tabela com sub-testes

PACOTE TESTIFY: API DO SUB-PACOTE ASSERT

assert - GoDoc

https://godoc.org/github.com/stretchrcom/testify/asser

150%

Search

Index

Variables
func CallerInfo() []string
func Condition(t TestingT, comp Comparison, msgAndArgs ...interface{}) bool
func Conditionf(t TestingT, comp Comparison, msg string, args ...interface{}) bool
func Contains(t TestingT, s, contains interface{}, msgAndArgs ...interface{}) bool
func Containsf(t TestingT, s interface{}, contains interface{}, msg string, args ...interface{}) bool
func DirExists(t TestingT, path string, msgAndArgs ...interface{}) bool
func DirExistsf(t TestingT, path string, msg string, args ...interface{}) bool
func ElementsMatch(t TestingT, listA, listB interface{}, msgAndArgs ...interface{}) (ok bool)
func ElementsMatchf(t TestingT, listA interface{}, listB interface{}, msg string, args ...interface{}) bool
func Empty(t TestingT, object interface{}, msgAndArgs ...interface{}) bool
func Emptyf(t TestingT, object interface{}, msg string, args ...interface{}) bool
func Equal(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool
func EqualError(t TestingT, theError error, errString string, msgAndArgs ...interface{}) bool
func EqualErrorf(t TestingT, theError error, errString string, msg string, args ...interface{}) bool
func EqualValues(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool
func EqualValuesf(t TestingT, expected interface{}, actual interface{}, msg string, args ...interface{}) bool
func Equalf(t TestingT, expected interface{}, actual interface{}, msg string, args ...interface{}) bool
func Error(t TestingT, err error, msgAndArgs ...interface{}) bool
func Errorf(t TestingT, err error, msg string, args ...interface{}) bool
func Exactly(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool

93 funções!

TESTE-EXEMPLO: ARGUMENTOS DE LINHA DE COMANDO

Use defer com uma função anônima para restaurar os argumentos originais.

```
func Example_2WordQuery() { // ❶
        oldArgs := os.Args // ❷
        defer func() { os.Args = oldArgs }()
        os.Args = []string{"", "cat", "smiling"}
        main() // ❸
        // Output:
        // U+1F638    😸    GRINNING CAT FACE WITH SMILING EYES
        // U+1F63A    😺    SMILING CAT FACE WITH OPEN MOUTH
        // U+1F63B    😻    SMILING CAT FACE WITH HEART-SHAPED EYES
}
```

ThoughtWorks®

FERRAMENTAS

Bibliotecas e utilitários para TDD em Go

Pacotes de awesome-go mais estrelados*

5251 ★	stretchr/testify
3685 ★	smartystreets/goconvey
2166 ★	onsi/ginkgo
1452 ★	golang/mock
902 ★	DATA-DOG/go-sqlmock
884 ★	gavv/httpexpect
709 ★	onsi/gomega
575 ★	google/go-cmp
512 ★	franela/goblin
502 ★	h2non/baloo
496 ★	h2non/gock
404 ★	DATA-DOG/godog
387 ★	go-check/check

*Dados coletados em 2018-07-05

ThoughtWorks®

TÉCNICAS

Coding tips

TESTE-EXEMPLO USANDO A INTERFACE STRINGER

Implementar um método String() facilita a depuração e testes.

```
func ExampleMake() {
        w := []string{"beta", "alpha", "gamma", "beta"}
        s := Make(w...)
        fmt.Println(s)
        // Output: Set{alpha beta gamma}
}
```

TESTE-EXEMPLO COM SAÍDA NÃO ORDENADA

```
func ExampleSet_Channel() {
        set := MakeFromText("beta alpha delta gamma")
        // iteration order over underlying map is undefined
        for elem := range set.Channel() {
                fmt.Println(elem)
        }
        // Unordered output:
        // alpha
        // beta
        // delta
        // gamma
}
```

Unordered faz o teste aceitar as linhas da saída em qualquer ordem

TESTE COM VARIÁVEL DE AMBIENTE FAJUTA

```
func restore(nameVar, value string, existed bool) {
        if existed {
                os.Setenv(nameVar, value)
        } else {
                os.Unsetenv(nameVar)
        }
}
```

Use defer para restaurar o estado original da variável

```
func TestGetUCDPath_isSet(t *testing.T) {
        pathBefore, existed := os.LookupEnv("UCD_PATH")
        defer restore("UCD_PATH", pathBefore, existed)
        ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
        os.Setenv("UCD_PATH", ucdPath)
        got := getUCDPath()
        if got != ucdPath {
                t.Errorf("getUCDPath() [set]\nwant: %q; got: %q", ucdPath, got)
        }
}
```

FORJANDO LEITURA DE ARQUIVO (1/2)

```
const dataStr = `003C;LESS-THAN SIGN;Sm;0;ON;;;;;Y;;;;;
003D;EQUALS SIGN;Sm;0;ON;;;;;N;;;;;
003E;GREATER-THAN SIGN;Sm;0;ON;;;;;Y;;;;;
003F;QUESTION MARK;Po;0;ON;;;;;N;;;;;
0040;COMMERCIAL AT;Po;0;ON;;;;;N;;;;;
0041;LATIN CAPITAL LETTER A;Lu;0;L;;;;;N;;;;0061;
0042;LATIN CAPITAL LETTER B;Lu;0;L;;;;;N;;;;0062;
`


func TestFilter(t *testing.T) {
        query := "sign"
        data := strings.NewReader(dataStr)
        got := Filter(data, query)
        want := []string{
                "U+003C\t<\tLESS-THAN SIGN",
                "U+003D\t=\tEQUALS SIGN",
                "U+003E\t>\tGREATER-THAN SIGN",
        }
        assert.Equal(t, want, got)
}
```

Use strings.NewReader para construir um buffer com os dados (implementa a interface Reader)

FORJANDO LEITURA DE ARQUIVO (2/2)

Em vez e tipos específicos, faça suas funções aceitarem interfaces comuns, como io.Reader

```
func Filter(data io.Reader, query string) []string {
        queryTerms := strset.MakeFromText(strings.ToUpper(query))
        scanner := bufio.NewScanner(data)
        result := []string{}
        for scanner.Scan() {
                name, code := parseLine(scanner.Text())
                if match(queryTerms, name) {
                        line := fmt.Sprintf("U+%04X\t%c\t%s", code, code, name)
                        result = append(result, line)
                }
        }
        return result
}
```

TESTE COM UM DUBLÊ DE SERVIDOR HTTP

Pacote httptest da biblioteca padrão oferece dublês para testar clientes e servidores HTTP

```
func TestFetchUCD(t *testing.T) {
        srv := httptest.NewServer(http.HandlerFunc(
                func(w http.ResponseWriter, r *http.Request) {
                        w.Write([]byte(lines3Dto43))
                }))
        defer srv.Close()

        ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
        done := make(chan bool)                 // ❶
        go fetchUCD(srv.URL, ucdPath, done) // ❷
        _ = <-done                              // ❸
        ucd, err := os.Open(ucdPath)
        if os.IsNotExist(err) {
                t.Errorf("fetchUCD did not save:%v\n%v", ucdPath, err)
        }
        ucd.Close()
        os.Remove(ucdPath)
}
```

TESTE LENTO QUE PODE SER PULADO

Acesse testing.Short(). Se true, invoque t.Skip() para reportar que o teste foi pulado.

```
func TestOpenUCD_remote(t *testing.T) {
        if testing.Short() { // ❶
                t.Skip("skipped test [-test.short option]") // ❷
        }
        ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
        ucd, err := openUCD(ucdPath)
        if err != nil {
                t.Errorf("openUCD(%q):\n%v", ucdPath, err)
        }
        ucd.Close()
        os.Remove(ucdPath)
}
```

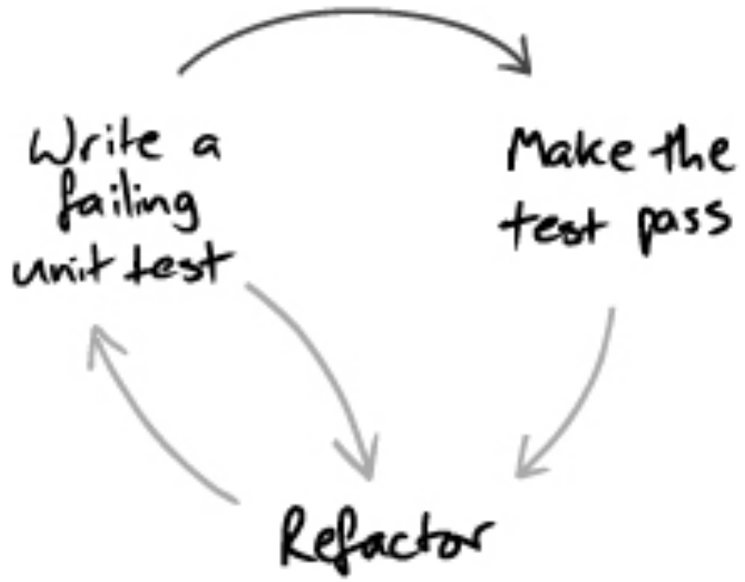
ThoughtWorks®

VARIAÇÕES

Além do estilo clássico de TDD

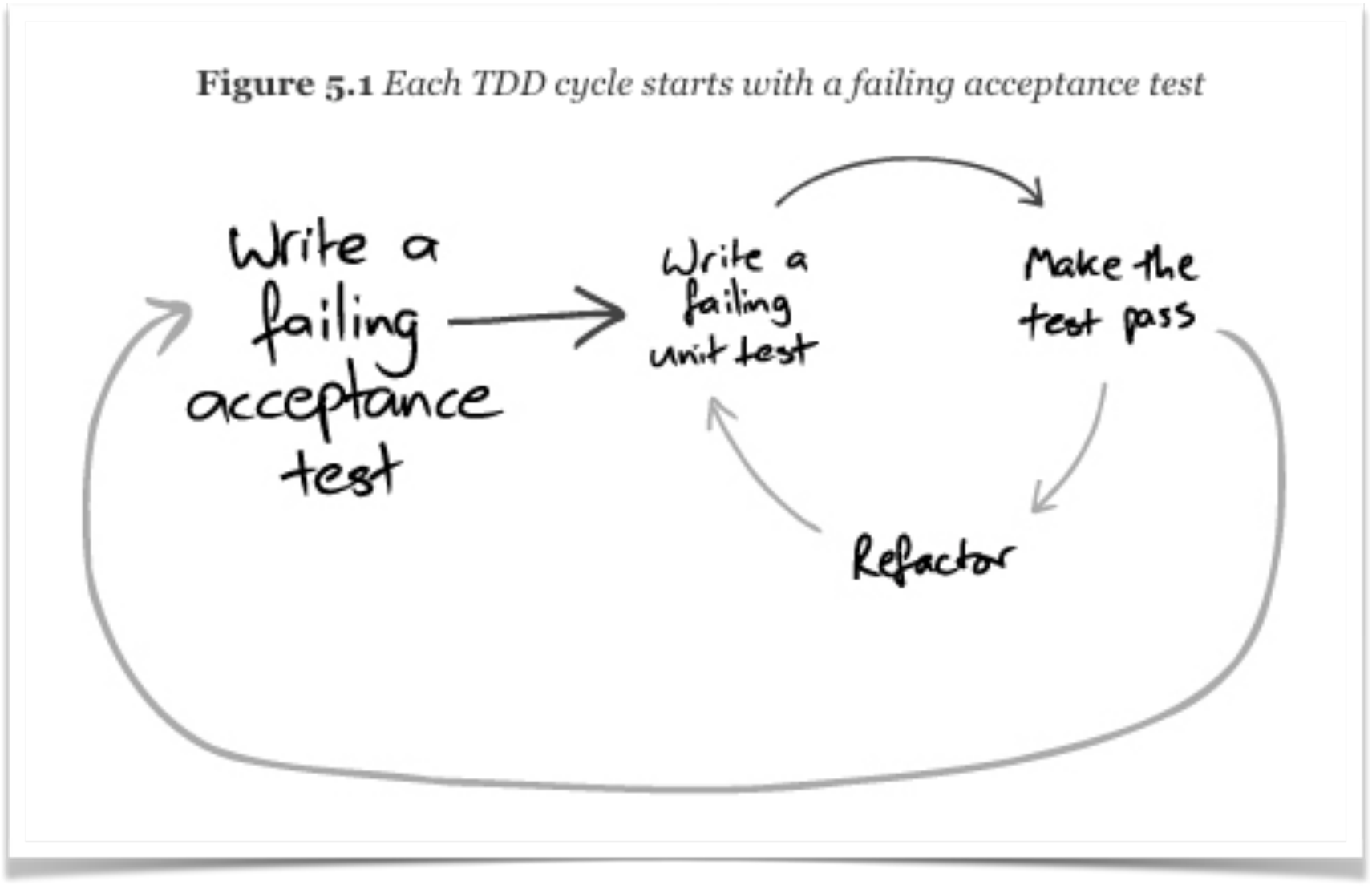
CICLO DE TDD: REFATORAR AO ESCREVER UM TESTE

Figure 5.3 *Difficulties writing tests may suggest a need to fix production code*



Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

CICLOS DE TDD CYCLES: ESTILO MOCKISTA



Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

ESTILOS DE TEDD

Estilo Chicago, ou “classic”

Basicamente de dentro para fora: de testes unitários até testes de aceitação

Estilo London, ou “mockista”

Basicamente de fora para dentro: de testes de aceitação até testes unitários

ANDREW GERRAND ON FAKES

"Go eschews mocks and fakes in favour of writing code that takes broad interfaces."

"That's generally how we get around dependency injection frameworks and large mocking frameworks: just by writing code that uses small interfaces. Then we have small fakes like the ResponseRecorder — small fakes that allow us to inspect how they were used. There are frameworks that generate those kinds of fakes — one of them is called Go Mock [...]. They're fine, but I find that on balance the hand-written fakes tend to be easier to reason about, and clearer to see what is going on. That's my personal experience. But I am not an "enterprise" Go programmer so maybe people need that, I don't know. That's my advice."

— Andrew Gerrand in *Testing Techniques* (I/O 2014) https://tgo.li/2upCkek

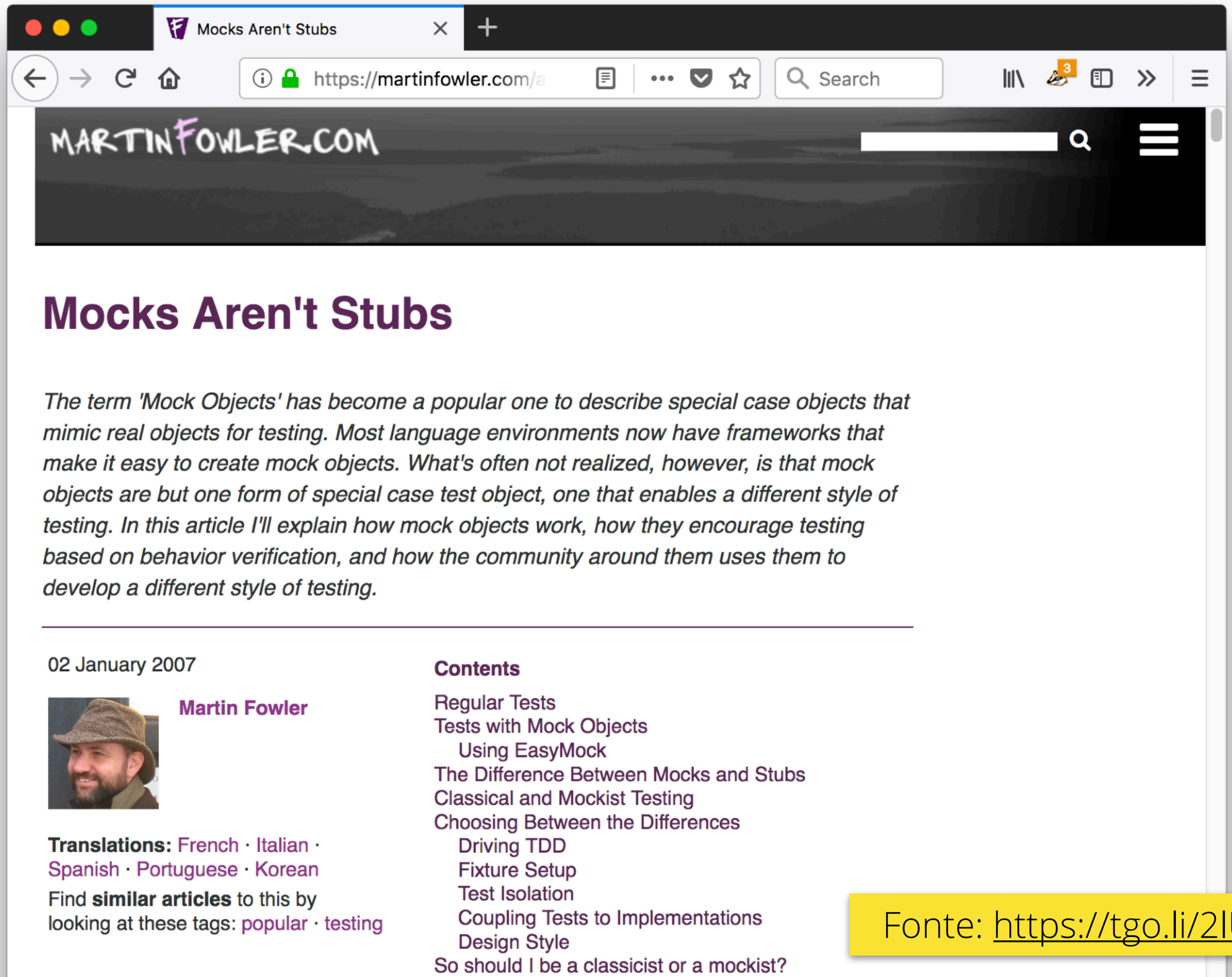
ANDREW GERRAND ON FAKES

“Go evita mocks e fakes complexos em favor de escrever código que usa interfaces abrangentes.”

“Geralmente, é como evitamos frameworks de injeção de dependência e mocks: apenas escrevendo código que usa pequenas interfaces. Então temos pequenos fakes, como o ResponseRecorder - pequenos fakes que permitem inspecionar como eles foram usados. Existem frameworks que geram esses tipos de mocks - um deles é Go Mock [...]. Eles são ótimos, mas acho que, no final, os dublês escritos à mão tendem a ser mais fáceis de entender, e mais transparentes para ver o que está acontecendo. Essa é minha experiência pessoal. Mas eu não sou um programador "enterprise" de Go, então talvez as pessoas precisem disso, não sei. Esse é o meu conselho.”

— Andrew Gerrand in *Testing Techniques* (I/O 2014) https://tgo.li/2upCkek

MARTIN FOWLER SOBRE ESTILOS DE TDD



Fonte: https://tgo.li/2IUqTXv

ThoughtWorks®

REFERÊNCIAS

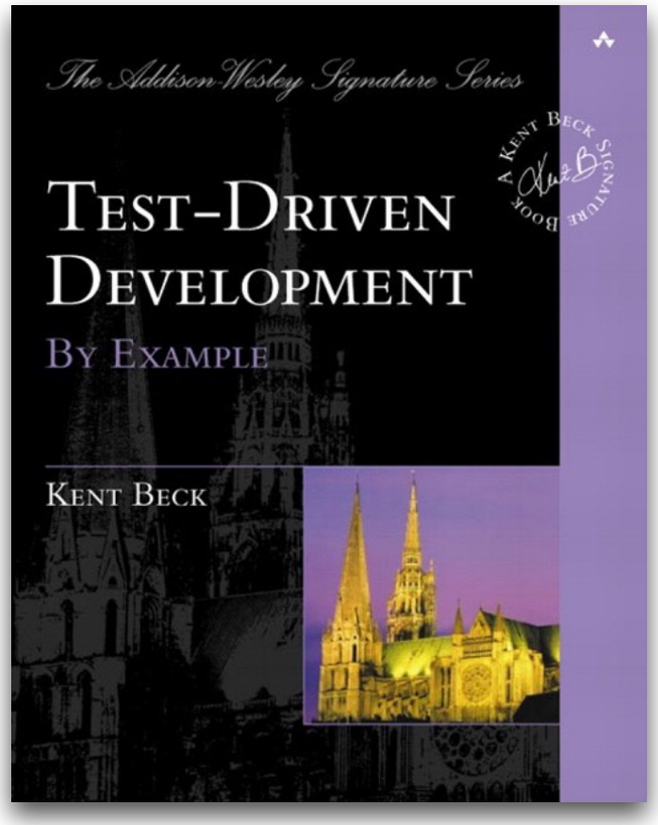
Onde aprender mais

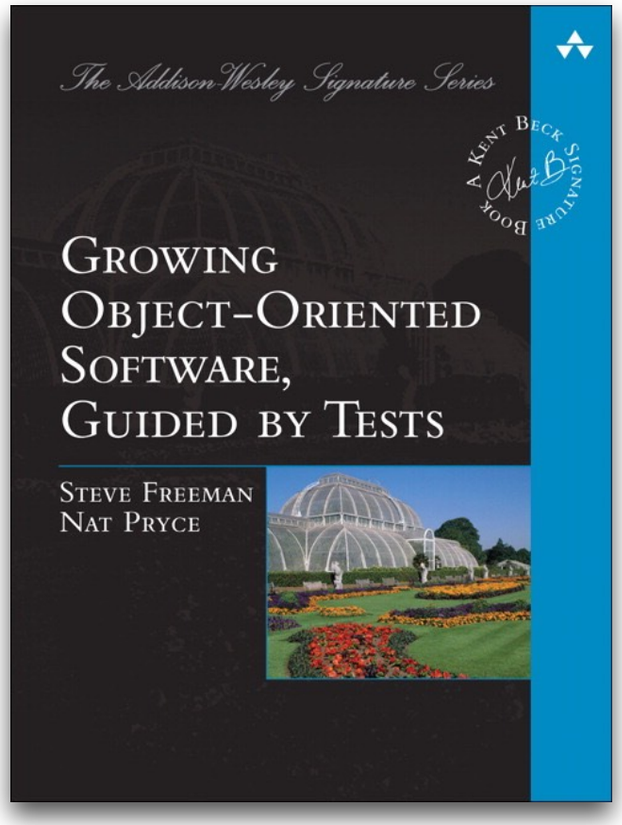
REFERÊNCIAS: LIVROS

Hugo Corbucci and Mauricio Aniche (book in Portuguese):
Test-Driven Development: Teste e design no mundo real com Ruby
https://tgo.li/2zGSl4N

Kent Beck: **Test Driven Development: By Example** https://tgo.li/2NvBfcX

Steve Freeman, Nat Pryce:
Growing Object-Oriented Software, Guided by Tests https://tgo.li/2tV8QoK





REFERÊNCIAS: POSTS, VIDEOS

Andrew Gerrand [video]
Go Testing Techniques (Google I/O 2014) https://tgo.li/2upCkek

Francesc Campoy [video]
Unit Testing HTTP Servers (justforfunc #16) https://tgo.li/2NSEGdZ

Martin Angers [post]
Lesser-known Features of Go-Test https://tgo.li/2m7ta1E

Martin Fowler [post]
Mocks Aren't Stubs https://tgo.li/2lUqTXv

Martin Fowler, Kent Beck, David Heinemeier Hansson [post + videos]
Is TDD Dead? https://tgo.li/2IWOAYn

Michael Feathers , Steve Freeman [video]
Test Driven Development: Ten Years Later https://tgo.li/2KD2Gnm

MUITO GRATO!

Por gentileza, mandem feedback:

Luciano Ramalho
@standupdev | @ramalhoorg
luciano.ramalho@thoughtworks.com

ThoughtWorks®