Pruebas unitarias con Go

Workshop de Go

6 de abril 2016

Iván Jaimes
Backend developer, Iver

Workshop de Go

1.1. Que es una prueba unitaria

```
Es una función que prueba una parte específica de
código de un paquete o programa.
```

Objetivo

```
Determinar si un código en cuestión está trabajando como esperamos para un
escenario en particular.
```

- Positivos: Ejecución normal.
- Negativos: Produce un error y un error esperado.

Pruebas unitarias

2.1. Historia

- TDD - Test Driven Development

```
Kent Beck 2002
```

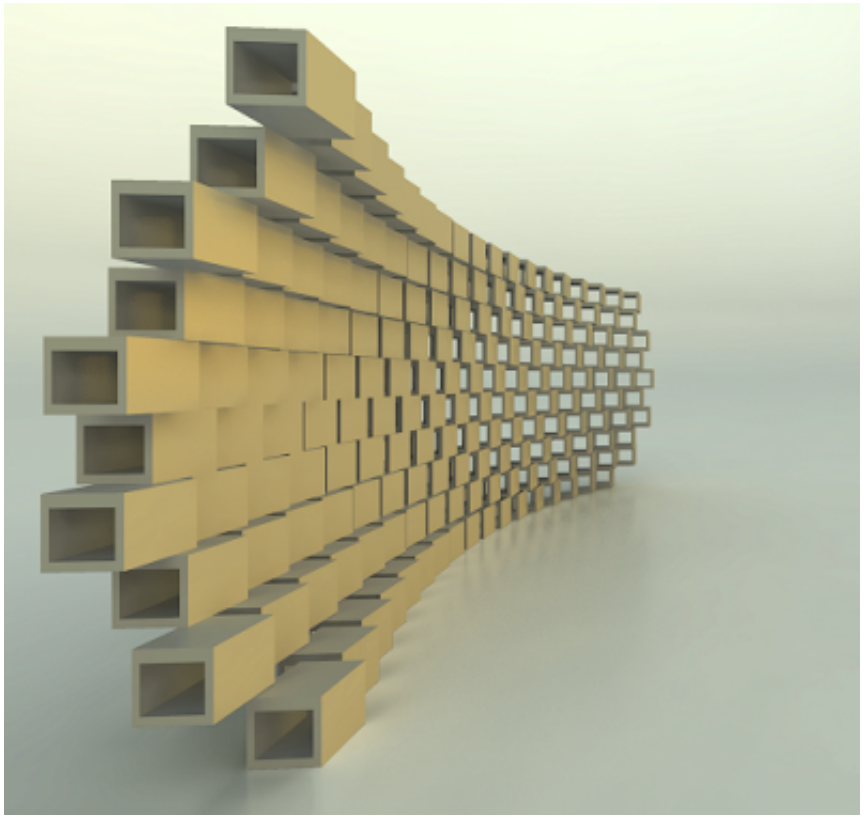
```
- (the first xUnit framework in Smalltalk I)
- Test-first development
```

- BDD - Behaviour Driven Development

```
Dan North 2006
- http://dannorth.net/
```

Ortogonalidad

Que un conjunto pequeño de elementos primitivos, se pueden
combinar en un número relativamente pequeño de maneras para construir el software.



Comandos

4.1. go test

Ejecución de pruebas

```
$ go test
```

Detalle de la ejecución

```
$ go test -v
```

Ejecución de pruebas independientes

```
$ go test -run=ExpReg
```

Comandos

5.1. go test -bench

```
$ go test -bench

func BenchmarkBigLen(b *testing.B) {
```

```
  big := NewBig()
     b.ResetTimer()
     for i := 0; i < b.N; i++ {
       big.Len()
     }
}
```

Pruebas con "ginkgo"

Instalación

```
$ go get -v -u github.com/onsi/gomega
$ go get -v -u github.com/onsi/ginkgo/ginkgo
```

Crear suite de pruebas

```
$ ginkgo bootstrap
```

Generar prueba unitaria

```
$ ginkgo generate
```

Instrucciones básicas de ginkgo

- It
- Describe && Context
- BeforeEach && AfterEach
- JustBeforeEach
- By

Algunas instrucciones de gomega

- Expect
- To
- Equal
- Should
- ShouldNot

Especificaciones Pendientes (Pending Specs)

Permite marcar instrucciones o bloques como pendientes empleando la letra **P** o **X**

```
PDescribe("some behavior", func() { ... })
PContext("some scenario", func() { ... })
PIt("some assertion")
PMeasure("some measurement")

XDescribe("some behavior", func() { ... })
XContext("some scenario", func() { ... })
XIt("some assertion")
XMeasure("some measurement")
```

Enfoque de especificaciones (Focused Specs)

Ejecución de un subconjunto de especificaciones

1) Anteponiendo la letra **F**

```
FDescribe("some behavior", func() { ... })
FContext("some scenario", func() { ... })
FIt("some assertion", func() { ... })
```

2) Empleando las banderas **--focus=REGEXP** y/o **--skip=REGEXP**

```
ginkgo -v --focus="Recibo"
ginkgo -v --focus="InApp" --skip="Connect"
ginkgo -v --skip="InApp"
```

Otro Framework Interesante

- http://agouti.org/

Tips para escribir buenas pruebas unitarias

- Hacer cada prueba ortogonal
- Evitar enunciados (asserts) innecesarios.
- Probar una unidad de código a la vez.
- Hacer “mocks” de los servicios y estados externos.
- Evitar pre-condiciones innecesarias.
- Evitar realizar pruebas de archivos de configuración
- Asignar nombres de manera clara y consistente

Thank you

6 de abril 2016
Tags: Go (#ZgotmplZ)

Iván Jaimes
Backend developer, Iver