

software pilots

**TRIFORK.**

# Bruger du det rigtige test-framework?

En hands on-præsentation

Ole Friis Østergaard, Trifork A/S  
<ofo@trifork.com>

# Inden vi går i gang...

---

- Hvorfor bekymre sig?
- Java, .NET, Ruby.
- Grundlæggende frameworks, mock-frameworks, "acceptance test"-frameworks.
- Dette er ikke et TDD-kursus!
- Kode, ikke snak.
- Ah, og slides'ene... de er ikke kønne og er på noteform. Det er ikke dem vi bruger tid på under denne "Geek Night"!

# JUnit, NUnit, Test::Unit

---

- Eksempel: Et ordre-system. Lad os starte med selve ordren.
- Jeg har **snydt** lidt hjemmefra!

# Lidt om stil

- Hvorfor ikke navngive testen efter hvilket aspekt af klassen den tester?
- Så bliver testen en specifikation.

```
public class OrderTest {  
    private Order order;  
  
    @Before  
    public void setUp() { ... }  
  
    @Test  
    public void hasItems() { ... }  
  
    @Test  
    public void keepsTrackOfItems() { ... }  
  
    @Test  
    public void keepsTrackOfSumOfItems() { ... }  
  
    @Test  
    public void knowsCustomerEmail() { ... }  
}
```

# Det rigtige værktøj...

- RSpec-unit-test.
- easyb.

```
context 'with items' do
  before do
    @order.add_item('Carpets', 100.0)
    @order.add_item('Glue', 52.5)
  end

  it 'keeps track of items' do
    @order.items.size.should == 2
    @order.items[0].name.should == 'Carpets'
    @order.items[0].amount.should == 100
  end
end
```

# En assertion pr. test?

---

- Riot.

```
should('keep track of items') { topic.items.size == 2 }  
should('know names of items') { topic.items[0].name == 'Carpets' }  
should('know amount of items') { topic.items[0].amount == 100 }  
should('keep track of sum of items') { topic.total == 152.5 }
```

# Nå, men til mock-frameworks

---

- Eksempel: Behandling af en ordre. Her skal nok ske flere ting:
  - Kunden skal have en bekræftelses-e-mail.
  - Beløbet skal trækkes fra kundens Dankort.
  - Ordren skal bogføres i bogføringssystemet.

# Der er et hav af frameworks...

---

...og en del stilarter de benytter.

- JMock: Verbose, "fluent interface". Ikke behov for Record & Replay. Ingen Arrange, Act, Assert.
- EasyMock: Knap så verbose. Record & Replay. Ingen Arrange, Act, Assert.
- Mockito: Ingen krav om Record & Replay. Arrange, Act, Assert.
- NMock: "Fluent interface", metodenavne som strenge. Record & Replay.
- RhinoMocks: Kan bruges på stort set enhver måde.
- Moq: "Fluent interface". Ikke behov for Record & Replay. Arrange, Act, Assert.
- FlexMock: "Fluent Interface". Ingen Record & Replay. Ingen Arrange, Act, Assert. Metodenavne som strenge? :-)

# Fluent interface

---

- Et eksempel i NMock. Og FlexMock/RSpec.
- I NMock har vi metodenavne som strenge.
- Eksempel i Moq, hvor man bruger C#-lambda-udtryk til at opnå samme effekt, men uden strenge.

# Record & Replay

---

- Eksempel i JMock og EasyMock.

```
public void testXxx() {  
    (...record...)  
    mocks.replay();  
    (...replay...)  
    mocks.verify();  
}
```

# Arrange, Act, Assert

- Det er lidt sært at vores almindelige asserts står sidst i testene, mens vores tilsvarende assertions i mock-kode står først i testene.
- Eksempel i Mockito.

```
public void testXxx() {  
    stub.when(someMethod).return(45);  
  
    obj.someMethod();  
    assert(mock).gotCalledWith(someOtherMethod);  
    assert(obj.getValue()).equals(45);  
}
```

# "Acceptance test"-frameworks

- Nogle gange vil man gerne gøre tests forståelige for andre end programmørerne. Så kan man involvere sin PO og sin kunde.
- Framework for Integrated Tests.
- Mest udbredte framework: Fitnessse.

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	3.142857142857143~3.14
9	3	3.0<5
11	2	4<5.5<6
100	4	[25.0] expected [33]

# Andre specifikations-drevne frameworks

---

- Det skorter ikke på denne slags frameworks.
- Cucumber, JBehave.

```
Scenarie: En ordre med et antal linjer skal kende totalen
  Givet en tom ordre
  Når jeg tilføjer en linje med 45 kr tæpper
  Og jeg tilføjer en linje med 30 kr loftsplader
  Så skal totalen være 75
```

# Hamcrest-matchere

---

- Eksempel i JUnit.
- Og med JMock, hvor man kan matche fx properties i objekter der kastes rundt.

```
List<OrderItem> items = order.getItems();  
assertThat(items.size(), is(2));  
assertThat(items, hasItem(itemWithName("Carpets")));
```

# Vidste du...?

---

- Hamcrest-matchere i JMock, Mockito og NMock. Ikke i EasyMock - den har sit eget interface.
- Man kan også få mocks til at kaste exceptions.
- Sekvenser i JMock, EasyMock, NMock osv.
- Tilstandsmaskine i JMock.
- Mockito og andre frameworks har mulighed for callback. Nyttigt en sjælden gang imellem (en metode hvori en anden metode kaldes, og et af argumenterne herfra skal returneres).
- Visse frameworks har "spies"/"partial mocks", hvor man kan lave en "wrapper" om et rigtigt objekt og mocke dele af det.
- Mockito har en "ArgumentCaptor", så man kan fange de argumenter mock-metoden kaldes med, og senere "asserte" på dem.