Testautomatisering

BDD, Cucumber

Idag

- FM:
  - Cucumber
  - BDD
  - Test Doubles
- EM:
  - Lab 3
  - Handledning

Idag

- FM:
  - Cucumber
  - BDD
  - Test Doubles
- EM:
  - Lab 3
  - Handledning

Idag

- Cucumber
  - Verktyg för automatiserade acceptanstest (eller story tests)
  - Installation:
    - gem install cucumber
    - Xubuntu: sudo gem install cucumber

Idag

- Cucumber
  - Med hjälp av cucumber beskriver vi Features med hjälp av scenarion.
  - Jfr RSpec

- Cucumber
  - Exempel
  - Feature: Division
    - In order to avoid silly mistakes
    - Cashiers must be able to calculate a fraction
    - Scenario: Regular numbers
      - * I have entered 3 into the calculator
      - * I have entered 2 into the calculator
      - * I press divide
      - * the result should be 1.5 on the screen

Idag

Idag

- Cucumber
  - En feature beskrivs i cucumber på ett sådant sätt att vem som helst kan förstå det.
  - Det är därför även ett utmärkt verktyg för kommunikation och sammarbete mellan olika intressenter i ett projekt.

Cucumber

- Cucumbers delar
  - Feature
  - Cucumber-kommandot
  - Step definitions

Cucumber

- Cucumbers delar
  - Feature
    - Språk: Gherkin
    - Titel (Fritext)
    - Feature Narrative (Fritext)
    - X antal scenarion
      - Innehåller X steps

Cucumber

- Cucumbers delar
  - Step Definition
    - Språk: Ruby
    - Definitionen av ett enstaka steg i ett scenario

Cucumber

- Cucumbers delar
  - Cucumber-kommandot
    - Tolkar stegen vi beskrivit i ett scenario
    - Försöker knyta varje steg till en step definition som vi skrev i ruby.

- Feature
  - Feature Title
    - För att snabbt kommunicera vad en feature innebär
    - Exempel
      - Stock clerk adds inventory item
      - Anonymous visitor adds blog comment
      - …etc.

Cucumber

- Feature
  - Feature Narrative
    - Under titeln
    - Fritext
    - Men följer ofta ett standardformat.

Cucumber

- Feature
  - Feature Narrative
    - As a <role>
    - I want <feature>
    - So that <business value>
    - OBS! business value

Cucumber

- Feature
  - Feature Narrative
  - Feature: Division
    - As a cashier
    - I want to be able to calculate a fraction
    - So that I can give the customer the correct change

Cucumber

- Scenario
  - Gherkin
  - Enkelt språk, med ganska få keywords.

Cucumber

- Gherkin
  - Feature
  - Background
  - Scenario
  - Scenario outline
  - Scenarios
  - Given
  - When
  - Then
  - And/But
  - |
  - ”””
  - #

Cucumber

- Gherkin
  - Given - precondition
  - When – En händelse
  - Then - postcondition
  - And/But – specificera ovanstående ytterligare

Cucumber

- Feature: Traveler books room
  - In order to reduce staff
  - As a hotel owner
  - I want travelers to boook rooms on the web

Cucumber

- Feature: Traveler books room
  - In order to reduce staff
  - As a hotel owner
  - I want travelers to boook rooms on the web

  - Scenario: Successful booking

Cucumber

- Feature: Traveler books room
  - In order to reduce staff
  - As a hotel owner
  - I want travelers to boook rooms on the web

  - Scenario: Successful booking
    - Given: a hotel with ”5” rooms and ”0” bookings

Cucumber

Cucumber

- Cucumber-filerna sparas som *.feature
- Och körs med cucumber-kommandot

- given /^a hotel with ”([^”]*)” rooms and ”([^”]*)” bookings$/ do |arg1, arg2|
  - Pending # hotel = Hotel.new(arg1,arg2)
- end

Cucumber

- Feature: Traveler books room
  - In order to reduce staff
  - As a hotel owner
  - I want travelers to boook rooms on the web

  - Scenario: Successful booking
    - Given: a hotel with ”5” rooms and ”0” bookings
    - When: I book a room
    - Then: I should get a confirmation

Cucumber

- when /^I book a room$/ do
  - Pending # express the regexp with the code
    # you wish you had
  - # @message = hotel.book()
- end

Cucumber

- Feature: Traveler books room
  - In order to reduce staff
  - As a hotel owner
  - I want travelers to boook rooms on the web

  - Scenario: Successful booking
    - Given: a hotel with ”5” rooms and ”0” bookings
    - When: I book a room
    - Then: I should get a confirmation

Cucumber

- then /^I should see ”([])”$/ do | greeting |
  - Pending #@message.should == greeting
- end

Cucumber

Cucumber

- Vi utvecklar i cucumber tills vi stöter på ett logiskt fel
- Då växlar vi över till rspec och utvecklar beteendet därifrån.

- Rspec
  - Create new hotel
  - Make a booking
  - Get confirmation

Cucumber

- require “./hotel”

- describe Hotel, "#book" do
  - it “returns a welcome message" do
    - hotel = Hotel.new(5,0)
    - welcome = hotel.book()
    - welcome.should == “Welcome!”
  - end
- end

Cucumber

Idag

- Varför Cucumber + Rspec?
  - Cucumber tvingar oss att tänka på problemdomänen
  - Rspec är ett utmärkt verktyg för att översätta detta till den tekniska domänen och beskriva ensklida metoders och klassers beteende
  - Hade vi använd Test::Unit istället för rspec här så hade övergången inte alls varit lika smidig

Idag

- Varför Cucumber + Rspec?
  - Det är också en fråga om att avgöra när något är klart.
  - Är ett projekt klart när alla dess enhetstester är skrivna och är passed?

Idag

- Varför Cucumber + Rspec?
  - Ja kanske – men det beror på hur våra enhetstest ser ut. Dvs. om vi har rätt enhetstester så är vi klara om de visar status passed.
  - Cucumber hjälper oss se till att vi har rätt enhetstester.

- BDD Cycle
  - Outside-In

Idag

Idag

- Mocks, Fakes, Stubs, Test Doubles…
  - Det är i de flesta verkliga scenarion omöjligt att inte ha objekt som är beroende av andra objekt.
  - För automatiserade test så innebär detta ett problem. Våra test blir mer sköra och ett brutet test kan innebära en bug i det beroende objektet

Idag

- Mocks, Fakes, Stubs, Test Doubles…
  - Exempel
  - För vår webbshop så vill vi skicka ut ett mail då en beställning behandlats klart.
  - Klassen som är ansvarig för att behandla en beställning är ”OrderProcessor”
  - OrderProcessor vet inte hur man skickar mail. Detta är delegerat till ett Mailer-objekt.

Idag

- Mocks, Fakes, Stubs, Test Doubles…
  - Exempel
  - När vi skriver test för OrderProcessor så måste vi sätta ett Mailer-objekt för OrderProcessor
  - Det är dock ointressant ur ett Unit Test-perspektiv ifall ett faktiskt mail skickas eller inte (Det är dock intressant om vi skulle skriva integrationstester)
  - Det är dessutom kostsamt (tidsmässigt) att anropa ett Mailer-objekt

Idag

- Mocks, Fakes, Stubs, Test Doubles…
  - Exempel
  - För att lösa detta så skapar vi en MailerMock-objekt. Det beter sig från utsidan precis som ett vanligt Mailer-objekt, men har enklast möjliga implementation.
  - Denna typ av objekt kallas för Mock, Fake, Stub eller Test Double.

Idag

- Förklaringar till koncept boken
  - Algoritm: Stegvis, detaljerad, beskrivning av lösningen på ett problem
  - Code Smell: Potentiellt dålig källkod (dåligt strukturerad, svår att förstå eller liknande)
  - Single Responsibility Principle (SRP): Varje metod eller klass skall endast ha ett ansvarsområde.
  - …hittar ni fler saker – maila mig.

- Källor
  - The RSpec book
    - Främst Cucumber-kapitlen (Kap 17-18 i min utgåva)
    - Tidigare utgåvor hade ingen cucumber-del. Ni kommer dock hitta tillräckligt med info i Part I av boken.

RSpec

- Lab 3

  Utveckla ett mastermind spel enligt boken

  Jag vill att ni skall se helheten

  BDD är svårt om man inte är van vid TDD/UT

  Fundera över vad du skriver in och varför de växlar mellan cucumber och rspec

RSpec

Nästa gång

- Mer om BDD
- Lab 3

- ...

Fin