Testautomatisering

Enhetstester

Idag

- FM:
  - Enhetstester
- EM:
  - Handledning Lab2

Denna veckan

- Fre:
  - ”Bra” Unit Test
  - TDD
  - Videoföreläsning

Unit Test

- Unit Test
- Enhetstester
- Komponenttester
- (Test med ramverk för Unit Test)

Unit Test

- Vad?
  - Test av en aspekt av en enskild komponent i systemet. Vi vill testa en aspekt av t.ex. en metod eller en klass
    - (Dvs. det räcker normalt inte med ett Unit Test för att testa en metod).

- Med andra ord så är vi så långt ifrån End-To-End-test vi kan komma.

Unit Test

Unit Test

- Vad?
  - Bygger på xUnit
    - xUnit är ett testramverk som finns för flera olika språk.
    - Det finns andra testramverk, men xUnit brukar vara det största
    - JUnit: Java
    - NUnit: C#
    - JSJUnit: Javascript
    - …Dvs ni kommer kunna använda ett ramverk med liknande syntax i många andra språk

Unit Test

- Vad?
  - ...Men vi kan också skapa integrationstest / end-to-end test mha av ett test-ramverk avsett för Unit Test. Vi kommer beröra detta på fredag.

Unit Test

- Att komma igång:
  - require “test/unit”
    - Ingår i en normal rubyinstallation
  - require “[class].rb”
    - Klassen vi skall testa
  - (Vi skulle kunna skriva implementationen av vår klass och test för klassen I samma fil, men detta görs normalt inte)

Unit Test

- Test Class
  - Innehåll?
  - Ett antal testfall (test methods)
  - Gemensam Setup
  - Gemensam Tear Down
  - Ev. hjälpmetoder
    - Om dessa är mer allmänna så lägga de i en separat fil

Unit Test

- Test Class
  - Namngivningsstandard: “Test[Class]”
    - Ex: TestAccountService

Unit Test

- Test Class
  - Ärver från TestCase
    - class TestSimpleNumber < Test::Unit::TestCase
  - Arv innebär att vi får metoder från basklassen (TestCase) på köpet i subklassen (TestSimpleNumber)
  - Arv markeras med ”<” i klassdefinitionen

Unit Test

- Testmetod
  - Namn
    - Måste börja med ”test_”

Unit Test

- Testmetod
  - vad kommer den innhålla?
  - Ett specifikt test
    - Ex: Kontrollera att ett konto ej kan skapas utan att vara associerat till en person
    - Med skapas menas här Account.new och inte något som sker i ex. ett GUI

Unit Test

- Testmetod
  - vad kommer den innhålla?
  - Med ett specifikt test menas att vi utför något (ex: Account.new) och sedan testar antaganden som vi vill skall gälla (Kontot skall vara associerat med en användare).

Unit Test

- Testmetod
  - Vanligaste sättet att arrangera ett test på är minnesregeln ”Arrange - Act – Assert”

Unit Test

- Testmetod
  - Arrange: Setup för en pre condition
    - T.ex: test_user = User.new

Unit Test

- Testmetod
  - Act: Utför det som skall testas
    - T.ex: test_account = Account.new(test_user)

Unit Test

- Testmetod
  - Assert: Testa att antaganden gäller

```
def test_new_account_should_have_a_user
    // Arrange
    test_user = User.new

    // Act
    test_account = Account.new(test_user)

    // Assert
    // Ett nytt konto skall vara associerat med en användare
 end
```

Unit Test

- Assertions, Assertions, Assertions
  Exempel på ett gäng assertions
  Ge alltid felmeddelande för dina test

Unit Test

- Testmetod – vidare
  - Resulterar alltid i pass eller fail (Dvs. de kommer inte resultera i “ErrorCode 38912” eller något annat godtyckligt

Unit Test

- Testmetod – vidare
  - Testa endast en sak

Unit Test

- Testmetod – vidare
  - Resulterar alltid i pass eller fail
    - (Dvs. de kommer inte resultera i “ErrorCode 38912” eller något annat godtyckligt

Unit Test

- Testmetod – vidare
  - håll dem små

- Testmetod – vidare
  - namngivning - tydliga, beskrivande namn
    - Det blir väldigt ofta väldigt långa namn här - detta är normalt

Unit Test

- Testmetod – vidare
  - Förekommer och är en bra idé: kommentar som beskriver affärsnyttan med testet
    - saknas dock ofta

Unit Test

Unit Test

- Testmetod – vidare
  - vi kan inte testa privata metoder normalt
  - => men vi vill förmodligen göra det i vissa fall?
  - En möjlig lösning: implementera .publize_methods
  - källa: http://blog.jayfields.com/2007/11/ruby-testing-private-methods.html

- Testmetod – vidare
  - vidare diskussioner om vad som utgör ett bra Unit Test på fredag

Unit Test

Unit Test

- Setup
  - Vad/Varför?
    - En metod ”setup” som körs innan varje test
    - För att slippa återupprepa kod

```
def setup
  @helper_object = HelperObject.new
end
```

Unit Test

- Tear Down
  - Vad/Varför?
    - En metod ”teardown” som körs efter varje test
    - För att slippa återupprepa kod och se till att vi inte har öppna resurser
    - Används för det mesta inte

```
def teardown
  @browser.close
end
```

- Tear Down
  - Vad/Varför?
  - När?
    - t.ex. - vi har kört watir-webdriver i våra tester - kör browser.close här

Unit Test

Unit Test

- Helper klasser/metoder
  - Exempel?
    - Fake DB Data

Unit Test

- Hur kör vi våra test?
  - test runner
  - Normalt: ruby -w tc_simpleNumber2.rb

Unit Test

- Hur kör vi våra test?
  - test runner
  - Enstaka test: ruby -w tc_simpleNumber2.rb --name test_typecheck
  - Flera test: ruby -w tc_simpleNumber2.rb --name /test_type.*/

Unit Test

- Hur kör vi våra test?
  - Tänk på:
    - Håll testen små!
    - Håll testen snabba!
    - Vi vill kunna köra dessa test efter varje ändring vi gör

Unit Test

- Namngivning för testfiler
- Två typer av filer:
  - “Test Case”-filer (ett antal test för ett koncept)
  - ”tc_[…].rb”
  - “Test Suite”-filer (en grupp av “Test Case”-filer som vi vill köra tillsammans)
  - ”ts_[…].rb”

Unit Test

- Var lägger vi våra test?
  - projekt-rot: /
  - tester: /test/
  - test som rör klassen “user”: /test/test_user.rb
  - ...men detta kan skilja sig från projekt till projekt

Unit Test

- Resurser:
  - http://en.wikibooks.org/wiki/Ruby_Programming/Unit_testing
  - Ruby Best Practices - Kap 1 (Överkurs - TDD)
    - (Kan vara svårpenetrerat - Dock nyttigt)
  - (Överkurs: The Art of Unit Testing)

Nästa gång

- Unit Test – Fortsättning
- Lab 2 Handledning

- ...

Fin