Lab 2 – Testautomatisering

Deadline: 2013-03-04 19:59

Lab 2 utförs i par om två stycken elever.

Inlämning sker med en zippad fil ”Lab2.zip” mailad till david.g@jetas.se.

Zip-filen skall innehålla samtliga .rb-filer ni skapat för denna lab. Utöver ruby-filerna skall zip-filen även innehålla en textfil ”info.txt”. ”info.txt” skall innehålla följande:

- Fullständigt Namn + Födelsenr. för båda eleverna.
- Skriv vilka uppgifter ni utfört. Ex: ”Utfört Uppg: 1,3,4,5,6,8”.
- Skriv svar på de frågor där det anges i uppgiften att ni skall svara i ”info.txt”. Ex: ”Fråga 2: [Svar här]”.

Om ni upplever att någon av uppgiftsbeskrivningarna är otydliga eller har andra tankar/kommentarer – hör av er direkt till mig. Jag kommer göra en nyhetspostning på kurshemsidan ifall jag förtydligar någon uppgiftsbeskrivning (Ni kan prenumerera på kurshemsidan f.ö.).

När jag talar om att skriva ut olika resultat så syftar jag på ruby-funktionerna ”puts”, ”print”, etc.

Syfte

Syftet med denna lab är att ni skall få prova att skriva komponentstester för några enkla Ruby-klasser. Ni kommer även få öva lite på Ruby då klasserna som skall testas skall implementeras.

Beskrivning

Ni skall implementera och testa delar av en Webbshop. De entiteter som vi kommer skapa och testa är: User, Product, Cart, Cartline.

User är klassen som representerar en användare på webbshopen.

Product är klassen som representerar en produkt i webbshoppen.

Cart representerar kundkorgen på webbshoppen. En Cart innehåller 0 eller flera CartLines och en totalsumma.

CartLine är en enskild ”rad” i kundvagnen. Dvs. en enskild produkt, pris per produkt, antal utav produkten och delsumman för raden. Ex: ”Product: Cleaning Kit – $19.99 – #: 2 – Subtotals: $39.98”

Uppgifter på nivå godkänt (Krav, G - 20/30p)

OBS! Du kommer behöva ändra implementationen av dina klasser allteftersom för att få enhetstesterna att visa ”passed”.

Uppgift 1: 4p

Skapa en klass User.

User skall innehålla följande attribut:

- first_name
- last_name
- email
- user_name
- id

User skall innehålla följande metoder:

- initialize(id, username, password)
- to_string
- full_name
- change_password(password, password_again)
- get_password
- authenticate(password)

Spara implementationen för denna I user.rb

Uppgift 2: 3p

Skapa en klass Product.

Product skall innehålla följande attribut:

- id
- name
- price
- description

Product skall innehålla följande metoder:

- initialize(id, name, price, description)
- to_string

Lägg implementationen för denna klass i product.rb

Uppgift 3: 2p

Skriv ett test som kontrollerar att full_name för en user returnerar first_name och last_name separerade med ett mellanslag om både first_name och last_name är satta.

Lägg dina test för User i en klass TestUser i tc_user.rb

Uppgift 4: 1p

Skriv ett test som kontrollerar att full_name för en user returnerar strängen ”-” ifall first_name och last_name inte är satta.

Uppgift 5: 1p

Skriv ett test som kontrollerar att full_name endast returnerar first_name (inget mellanslag sist) om first_name är satt och last_name inte är satt.

Anmärkning: Som ni märker på uppgift 3-5 så kan det lätt bli många tester även för väldigt enkla metoder (Och vi har inte skrivit samtliga ”vettiga” test för metoden full_name ännu). Framöver så kommer testerna inte vara lika uttömmande. Men ni får

Uppgift 6: 2p
Skriv ett test som kontrollerar att en user inte kan initieras utan username och password

Tips:

En startegi för att göra detta test är att se till att en user inte kan skapas utan att username och password sätts (se till att initialize i User-klassen tar username och password som argument).

Därefter kan vi i ett test göra en assert_raise som kontrollerar att ett exception kastas om vi försöker skapa en User utan att ange username och password.

Uppgift 7: 1p
Skriv ett test som kontrollerar att en user inte kan initieras med ett password som är kortare än 6 tecken.

Anmärkning: Här hade vi kunnat fortsätta testa en rad andra regler för password. T.ex. vilka tecken de innehåller, etc.

Uppgift 8: 1p
Skriv ett test som kontrollerar att authenticate-metoden returnerar true om rätt lösenord för en användare anges

Uppgift 9: 1p
Skriv ett test som kontrollerar att authenticate-metoden returnerar false om fel lösenord anges för en användare

Uppgift 10: 1p
Skriv ett test som kontrollerar att change_password returnerar false om password och password_again-parametrarna ej är identiska.

Uppgift 11: 3p
Skriv ett test som kontrollerar att change_password ändrar password om password och password_again är identiska (du kan göra en assert med hjälp av authenticate-metoden – se till att testerna för authenticate ligger innan testerna för change_password)

Uppgift 12: 2p
Skriv ett test som kontrollerar att get_password ej är publikt tillgänglig (det skall inte gå att anropa ex. user.get_password)

Uppgift 13: 3p
Skapa ett test som kontrollerar att det inte går att initiera ett Product-objekt utan att ange ”id”, ”name”, ”price”

Lägg dina test för Product i en klass TestProduct i filen tc_product.rb

Uppgift 14: 2p
Skapa ett test som kontrollerar att description sätts till ”N/A” om man inte anger description när man initierar ett product-objekt.

Uppgift 15: 3p
Skapa en fil ”ts_all_tests.rb” som kör alla test för både User och Product om man skriver ”ruby –w ts_all_tests.rb” på command-line prompten.

Uppgift 16: 2p
Ange i info.txt 2 st. ytterligare test vi hade kunnat implementera.

Uppgifter på nivå väl godkänd (Krav, VG - 15/20p på VG-delen + 21/32p på G-delen)

Uppgift 17: 2p
Skapa en klass CartLine.

CartLine skall innehålla följande attribut:

- product
- number_of_items

CartLine skall innehålla följande metoder:

- to_string
- add_item(number)
- remove_item(number)
- calculate_line_totals

Lägg klassen CartLine I en fil cart_line.rb

Uppgift 18: 3p
Skapa en klass Cart

Cart skall innehålla följande attribut

- cart_lines

Cart skall innehålla följande metoder:

- to_string
- add_to_cart(product, number_to_add)
- remove_from_cart(product, number_to_remove)
- add_shipping(amount)
- calculate_totals # totals = summa(line_totals) + shipping
- show_cart # visar alla cart lines + shipping + totals

Uppgift 19: 5p

Skriv ett lämpligt antal tester för att kontrollera att vi kan lägga till och ta bort items i ett CartLine-objekt. Kontrollera även att vi inte kan få ett negativt tal i number_of_items. Kontrollera att calculate_line_totals räknar ut summan av produkterna i en cart line på ett korrekt sätt.

Lägg testerna för CartLine i filen tc_cart_line.rb – lägg till tc_cart_line.rb i ts_all_tests.rb

Uppgift 20: 5p

Skriv ett lämpligt antal tester för att kontrollera att vi kan lägga till och ta bort produkter i en cart. Kontrollera att calculate totals gör rätt beräkningar. Kontrollera även att show_cart skriver ut innehållet i en cart, samt visar den totala kostnaden, inklusive shipping för en cart.

Lägg testerna för Cart i filen tc_cart.rb – lägg till tc_cart.rb i ts_all_tests.rb

Uppgift 21: 5p

Skapa en ny fil, tc_smoke_test.rb

Skapa en klass TestSmoke i tc_smoke_test.rb

Skriv ett test som använder watir-webdriver (eller watir) för att öppna addressen www.google.com i ett unit test

Använd setup för att initiera browsern.

Använd teardown för att köra browser.close

Anmärkning: Detta är inte ett Unit Test, men det visar hur vi kan använda ett Unit Test ramverk för att skapa andra typer av test.

Denna lösning är även intressant om vi vill köra GUI-tester på en build server eller i en continuous integration tjänst.