Guide – Ett FitNesse-test

Ladda ned RubySlim

Från: fitnesse.org -> plugins -> Ruby Slim for Ruby 1.9.3

Klicka på länken till github

Leta reda på Zip-knappen -> ladda ned

Lägg rubyslim valfri katalog – jag kommer referera till sökvägen till denna katalog som [SLIM_PATH] nedan.

OBS! undvik mellanslag i sökvägen!

Undvik gem install slim och gem install rubyslim

I FitNesse (localhost efter vi startat FitNesse med ”java –jar fitnesse-standalone”):

Starta fitnesse med java –jar fitnesse-standalone.jar (detta måste göras i katalogen som fitnesse-standalone.jar ligger i)

För virtual machine: java –jar fitnesse-standalone.jar –p 8080

Gå in på localhost i din browser (locahost:8080 för virtual machine)

Skapa Innehållssida med typ: static – Ex: Add: ”MinStartSida”

Gör en sida av typen static – kalla den vad du vill

Gör länk från startsidan genom att editera den och lägga in ”.MinStartSida” eller ”>MinStartSida”

Om adressen för din startsida är ”localhost/FrontPage.MinStartSida” så är det en undersida till FrontPage – om adressen är ”localhost/MinStartSida” så är det en huvudsida.

Skapa länk för Undersida: ”>MinStartSida”

Skapa länk för Huvudsida: ”.MinStartSida”

Skapa testsida när du står på MinStartSida med add (typ: test) – ge den ett namn, ex: ”MinTestSida”

Skapa en ny tabell på ”MinTestSida” genom edit och lägg till följande:

```
|My test|
|first name|last name|fullname?|
|David|Gullmarsvi|David Gullmarsvik|
```

Det du skriver som tabellrubrik (My test) måste senare återspeglas i Fixture-klassen (Som måste heta MyTest) och filen som Fixture-klassen ligger i (Som måste heta my_test.rb).

Kör test: Fail

FitNesse vet inte vilken test runner den skall ha ännnu

Lägg till init-info på din start-sida

```
!define TEST_SYSTEM {slim}
!define TEST_RUNNER {[SLIM_PATH]/bin/rubyslim}
!define COMMAND_PATTERN {ruby –I %p %m}
!define PATH_SEPARATOR { -I }
!path [SLIM_PATH]/lib
```

TEST_RUNNER- och !path-sökvägar måste ersättas med sökvägar från er dator. Använd windows-sökvägar om du är på windows (Dvs: något i stil med ”C:\rubyslim\bin\rubyslim”, etc – windows sökvägar börjar med C: (eller D:, etc. om filerna ligger på en annan hårddisk) och använder ”\” istället för ”/”)

Kör test: Fail

FitNesse vet inte var våra fixture-klasser + projekt-klasser är

Lägg till fixture-klasser

Fixture-filer skall ligga i katalogen [PROJECT_PATH]/lib/fixtures

[PROJECT_PATH] är en sökväg på din lokala dator, där din projekt-mapp ligger

Ersätt ”/” med ”\” för windows.

Vi döper filen till my_test.rb – detta är baserat på tabellrubriken vi tidigare lade till på vår testsida.

Filen innhåller följande:

```
module Fixtures
         class MyTest
         end
end
```

modulnamnet Fixtures måste vara samma som katalogen “fixtures”, fast med CamelCase

Klassnamnet måste vara samma som filen ”my_test.rb” och tabellrubriken på vår testsida ”My test”, fast med CamelCase

Vi har inga metoder ännu

Lägg till metoderna set_first_name, set_last_name och fullname i fixture-filen. Input-metoder måste börja med ”set_” och är baserade på de namn som vi har i de kolumnrubriker som inte avslutas med ”?” på vår testsida. Metodnamnen måste vara lowercase och med ”_” istället för mellanslag

Namnet på output-metoder är baserade på de namn i kolumnrubrikerna på testsidan som avslutas med ”?”

Lägg till logik i output-metod

Vi börjar med att returnera ”David Gullmarsvik” – detta är samma som finns i vår ”fullname?”-kolumn i tabellen vi skapade tidigare

```
module Fixtures
	class MyTest
		def set_first_name(arg)
		end

		def set_last_name(arg)
		end

		def fullname
			“David Gullmarsvik”
		end
	end
end
```

Citat-tecknen kommer bli felaktiga om ni kopierar ovanstående – ersätt dem.

Till en början vill vi bara testa att kopplingen mellan FitNesse och våra Fixture-klasser fungerar som tänkt. Dvs. vi vill inte ha någon logik som kan ge upphov till fel.

Vi returnerar därför ”David Gullmarsvik” i fullname-metoden, för att vi angav detta i vår tabell på test-sidan tidigare.

Kör test: Fail

FitNesse vet inte var dina fixtures är

Lägg till Class Path till Fixture + klasser på start-sidan

```
!path [PROJECT_PATH]/lib/fixtures
!path [PROJECT_PATH]/lib
```

Behåll den tidigare texten och lägg till ovanstående nederst. Ersätt som vanligt ”\” med ”/” på Windows-system.

Vi behöver även tala om för FitNesse i vilken modul våra fixtures ligger i. Detta gör vi på en SetUp-sida.

Välj Add från vår innehållssida (type: Test, name: SetUp)

Lägg till följande i SetUp-sidan:

```
|Import|
|Fixtures|
```

Fixtures skall vara en CamelCase variant av den module du angav I din fixture-fil

Kontrollera att det fungerar genom att kika på MinTestSida efter en text som innehåller: ”included Page: SetUp”

Kör Test: Nu borde dina tester vissa grönt.

Om du får ett Exception när du kör ett test kan du se detta direkt från test-sidan.

Länka till klasser vi vill testa

Hittills har inte Fixture-klassen pratat med några klasser som hör till implementationen av vårt projekt.

Detta är för att vi inte vill ha något som stör när vi sätter upp kopplingen mellan FitNesse och våra testfixtures. Nu skall vi koppla vår fixture till den faktiska klassen vi vill testa.

Skapa en klass User inuti Fixtures-modulen i my_test.rb – vi skall gradvis flytta denna definition till rätt ställe

```
class User
        attr_accessor :first_name, :last_name

        def fullname
                “#{@first_name} #{@last_name}”
        end
end
```

Tänk på att ändra citat-tecken.

Flytta ut klassen utanför Fixture-modulen I Fixture fil och lägg till en module runt klassen. Ex: ”module UserProject”. Tänk på att en module måste avslutas med end (nu behöver vi lägga till aktuell modul innan User då vi använder klassen i fixture-filen till UserProject::User)

Flytta ut klassdefinitionen + omgivande module till egen fil – user.rb. Nu behöver du även ange ”require ’./user.rb’” i my_test.rb.

Flytta ned user-filen till /lib-katalogen (det är här den skall ligga)

Flytta ned: Fail!

- Require funkar inte
- (Tänk på att du kan använda ex: irb hela vägen för att testa)

Det finns flera olika sätt att lösa detta på – t.ex: absolut path:

require ’/absolute/path/to/class/lib/user’

Den bättre lösningen är att lägga till katalogen som user.rb ligger i till våra sökvägar. Detta gör vi med följande ändring i fixture-filen:

```
$LOAD_PATH.unshift(‘../’, __FILE__)
require ‘user’
```

tänk på att ändra citattecknen ovan.

I slutändan ser våra två filer ut såhär:

[PROJECT_PATH]/lib/user.rb:

```
module UserProject
	class User
		attr_accessor :first_name, :last_name

		def fullname
			“#{@first_name} @last_name”
		end
	end
end
```

[PROJECT_PATH]/lib/fixtures/my_test.rb:

```
$LOAD_PATH.unshift(‘../’, __FILE__)
require ‘user’

module Fixtures
	class MyTest
		def initialize
			@user = UserProject::User.new
		end

		def set_first_name(arg)
			@first_name = arg
```

```
                end

                def set_last_name(arg)
                        @last_name = arg
                end

                def fullname
                        @user.fullname
                end
        end
end
```

Kör Test: Nu borde testen på vår test-sida fungera igen.

Flera test på samma sida

Skapa tabell 2 på test-sidan

> Detta funkar fint – vi kan lägga till en kopia på den tidigare tabellen (i normala fall hade vi haft andra tester i den nya tabellen – t.ex. test som vi förväntar oss skall ge errors eller liknande).

Kör test: Testen bör fungera.

Test Suite

Skapa Suite sida från innehålls-sidan (Add, typ: Suite). Ex: ”MainSuite”

Skapa 2 Undersidor med typ: Suite under den första Suite-sidan. Ex: ”SubSuite1”, SubSuite2”

Skapa 1 test-sida för varje Suite och lägg till tabeller (du kan kopiera den tidigare tabellen vi definierade). Ex: ”TestPage1”

Kör individuellt test på någon av test-sidorna. Detta bör fungera.

Navigera till någon av SubSuite-sidorna. Här bör du kunna trycka på ”Suite”-knappen för att köra alla tester i testsidor som ligger under vår SubSuite-sida.

Navigera till vår MainSuite-sida. Här bör du kunna trycka på ”Suite”-knappen för att köra alla tester i testsidor som ligger under vår MainSuite-sida (dvs samtliga tester under bägge SubSuites).

Du hittar Fitnesse-filerna i FitNesseRoot katalogen i katalogen du startar FitNesse ifrån.

Om din startsida är en ToppSida så ligger den som en katalog med samma namn som din sida direkt i FitNesseRoot – om den ligger under ex. FrontPage så kommer du hitta den i en katalog med samma namn som din sida under katalogen ”FrontPage”

Demo

I den nya Virtual Machine jag lagt upp här: http://www.loudelou.se/xubuntu2.zip så hittar ni ett fungerande FitNesse-demo.

Se guide från Kursvecka 1 under dokument/filer för hur du importerar denna Virtual Machine i Virutal Box. (du skall dock inte ladda ned xubuntu.zip som länkas i den filen – det är ovanstående fil ”xubuntu2.zip” som gäller).

Allt som är intressant för detta demo hittar ni under ”ta”-katalogen som hittas under hem-katalogen för användaren ”testaut”.

Följande kataloger finns i ”ta”-katalogen:

”demo”:

”/lib/user.rb”: klassen vi vill testa

”/lib/fixtures/my_test.rb”: Fixture-klass för FitNesse/RubySlim

“fitnesse”:

“fitnesse-standalone.jar”: För att sätta igång FitNesse. Startas med ”java –jar fitnesse-standalone.jar –p 8080”

”/FitNesseRoot/DemoPage”: Katalog som kommer innehålla de sidor jag skapat under ”DemoPage”-sidan i FitNesse

”rubyslim”:

rubyslim-filerna. Dessa behöver ni dock inte röra eller kika på.

”lab4”:

Här skulle ni kunna börja implementera fixture-filer och implementations-filer enligt den struktur som finns i demo-mappen (se ovan)

För att se på Demo-sidan så går ni in i katalogen ”fitnesse” och kör ”java –jar fitnesse-standalone.jar –p 8080”. Efter det kan ni öppna en browser och gå till sidan ”localhost:8080”. Längst ned på sidan bör ni ha en länk till ”.DemoPage”. Denna sida har samma struktur och innehåll som finns beskrivet i guiden ovan. Testerna på dessa sidor skall vara körbara – samtliga tester skall köras och ge grönt.