TVVS – Acceptance Tests Class

Carolina Azevedo | Nelson Costa
(up201506509 | up201403128)

October 2nd 2019

Exercise 1:

Imagine you're a software tester and you're asked to write some acceptance tests for Booking.com website, considering these two user requirements:

1. User Login;
2. Book a room;

As for now, you'll need to clone this git repo: https://github.com/mrnelsoncosta/TVVS

Exercise 2 – Simple Calculator Testing:

For this exercise, we'll need to use an Acceptance Testing tool. We're using Fitnesse.org, as presented in the class. To be able to use it fast, please follow the instructions:

1. On the command line, make sure you're on the folder containing the provided tool file and execute it, using the command *java -jar fitnesse-standalone.jar -p <PortNumber>*
    a. For the *PortNumber* aspect, you can type any, for example 2222.
2. On your browser, type http://localhost:<portNumber> and you'll be redirected to our Fitnesse workspace.
3. You'll need to comipile the *Calculator.java* inside the src/TVVS folder. Do it with the following command *javac -cp ../../fitness-standalone.jar Calculator.java*.
4. On your Test File, don't forget to specify the path of your .class files.

As shown in the presentation, add more tests into the decision table, with support for power functions.

Exercise 3 – Social Network:

Decision Table	Supplies the inputs and outputs for decisions. This is similar to the Fit Column Fixture
Baseline Decision Table	Almost identical to a >Decision Table but more readable for very big tables with many columns as only the changed values from one line to the baseline must be specified. The fixture implementation is identical to >Decision Table.
Dynamic Decision Table	Has the same syntax as a >Decision Table, but differs in the fixture implementation. It passes the column headers as parameters to the fixture.
Hybrid Decision Table	Combines the advantages in the fixture implementation of a Decision and a Dynamic Decision Table. Also supported by a Baseline Decision Table
Query Table	Supplies the expected results of a query. This is similar to the Fit Row Fixture
Subset Query Table	Supplies a subset of the expected results of a query.
Ordered query Table	Supplies the expected results of a query. The rows are expected to be in order. This is similar to the Fit Row Fixture
Script Table	A series of actions and checks. Similar to Do Fixture.
Table Table	Whatever you want it to be!
Import	Add a path to the fixture search path.
Comment	A table that does nothing.
Scenario Table	A table that can be called from other tables.
Library Table	A table that installs fixtures available for all test pages
Define Table Type	A helper table that defines the default table type for named fixtures.
Define Alias	A helper table that defines alias names for fixtures.

Slim has a lot of tables, as you can see above.

Another quite useful is the Script Table, so, to get to know a little bit of them, we're using a Social Network example:

```
|script | TVVS.SocialNetworkFixture |
|$SOCIALNETWORK = | getSocialNetwork() |
```

This tells the tool we want to build a script table, based on the class *SocialNetworkFixture* and create a new Social Network, to be saved into the variable *$*SOCIALNETWORK. Script tables are useful to do checks to some actions.

So, we want you to write the rows of script tables that answer to the following User Requirments:

- As an Admin, I want to check how many profiles my Social Network has, so that I can keep track of how popular it is.
- As a User, I want to create an account into the social network, so that I can start enjoying all its features.
- As a User, I want to see my information, so that I can check it.
- As a User, I want to create a post on to the social network, so that I can be an active member.