**Assignment 8 – Rspec**

<http://rspec.info/>

Rspec is a Behavior-Driven Development testing framework. It is similar to Cucumber in that it describes the behavior of the application under test. Cucumber takes BDD a step further than Rspec and allows you to express your tests in plain text—Rspec tests are written in Ruby code.

Rspec also greatly simplifies adding *assertions* (validation points) to your scripts by providing should and should\_not methods to your classes. Those assertions are the part of Rspec that we use within our Manheim automation framework, but it does much more than that!

Let’s take a look at how an Rspec scenario works:

require 'watir-webdriver'

require 'rspec'

describe "Travelocity Login" do

before(:each) do

@browser = Watir::Browser.new :firefox

@browser.goto "http://www.travelocity.com"

end

it "should display welcome message when user is logged in" do

# Home Page

@browser.span(id: "signUpOrChange").when\_present.click

# Membership Page

@browser.text\_field(id: "es\_alias").when\_present.set "askwwld@gmail.com"

@browser.text\_field(id: "es\_passwd").set "wwld"

@browser.span(text: "Log In").click

# My Trips Page

@browser.link(text: "Home").when\_present.click

# Home Page - Header

welcome\_message = @browser.span(id: "memberGreeting").when\_present.text

welcome\_message.should == "Hello Lance!"

end

after(:each) do

@browser.close

end

end

Let’s break this down:

First we require ‘rspec’ to make the gem available for us to use.

Next, the describe keyword defines the beginning of an Rspec feature file.

The before(:each) hook gets executed before each it scenario. Inside the before hook, we are launching the browser and navigating to the Travelocity site.

The it keyword defines the beginning of an Rspec scenario. You can have any number of it scenarios inside your Rspec feature file. The code inside this scenario should look familiar, but I’ve added a should validation for the welcome message.

The after(:each) hook gets executed after each it scenario. In this case we are closing the browser. Using the after hook to close the browser helps to prevent browsers from being orphaned if the test scenario encounters a runtime error.

Create a new file called login\_spec.rb in RubyMine and add the above code (Don’t copy/paste as you might get non-ascii characters). You can run it from the command line by opening your terminal project, navigating to your Travelocity directory and type the following command:

* rspec login\_spec.rb

You should see output that looks like this:

.

Finished in 30.77 seconds

1 example, 0 failures

Change the expected result for the welcome message validation by removing the exclamation point and execute the test again:

F

Failures:

1) Travelocity Login displays welcome message when user is logged in

Failure/Error: welcome\_message.should == "Hello Lance"

expected: "Hello Lance"

got: "Hello Lance!" (using ==)

# ./rspec\_login.rb:24:in `block (2 levels) in <top (required)>'

Finished in 33.05 seconds

1 example, 1 failure

Failed examples:

rspec ./rspec\_login.rb:11 # Travelocity Login displays welcome message when user is logged in

Try This:

Create a few new it scenarios for your login spec:

* Attempt to login with a bad username. Validate error message is displayed.
* Attempt to login with a bad password. Validate error message is displayed.
* Successful Login. Validate ‘LANCE HOWARD’s account’ text on the ‘My Trips’ page

**Bonus**

Create a Rspec file for flight results.

Modif one of your previous assignments to the Rspec format.

Add an Rspec should validation.