

Test-A-Monial

Testers guide to Test Automation

ASIDE

Getting started with Ruby, Cucumber and Capybara

May 23, 2014 ~ Simon Franklin

Since I started in test automation I have always been asked by fellow testers 'How can I set up my machine so I can start writing automated tests?'. When I first started out I also asked this very question.

There are lots of great resources out there which help to answer the above question but sometimes it can be like looking for a needle in a haystack. I have found many articles that are very technical and sometimes tailored towards users who have an existing development background, this post is for those guys who don't.

Here is my guide to setting up a simple cucumber project on a mac, there will no doubt be other ways and possibility better ways, but this is how I have done it and I have found it to be very effective.

Prerequisites for for this project: a mac with git, home brew, a text editor and the latest version of xcode including command line tools. There are many online tutorials taking you through how to install these tools. If you require the above take a look at Moncef Belyamani's blog post <http://www.moncefbelyamani.com/how-to-install-xcode-homebrew-git-rvm-ruby-on-mac/> (<http://www.moncefbelyamani.com/how-to-install-xcode-homebrew-git-rvm-ruby-on-mac/>)

1. Installing Ruby

Ruby comes pre installed on a mac. Check the version by opening the terminal and entering the following:

```
ruby -v
```

For the purposes of this project we will be using a more recent version of Ruby. Ruby Version Manager, often abbreviated as RVM is a great way of managing multiple versions of Ruby. For more info visit <https://rvm.io/>

To install RVM open the terminal and type:

```
\curl -sSL https://get.rvm.io | bash -s stable --ruby
```

Once the installation is complete we must now load RVM into the shell. To do this use a text editor and open the `.bash_profile` file in the home directory of your machine. I like to use vim but if you're not familiar with this editor use something like TextEdit. Depending on your text editor on the command line run

```
open -a TextEdit ~/.bash_profile
```

or

```
vim ~/.bash_profile
```

Once the `.bash_profile` is open copy and paste the below line, save then close

```
[[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm" # Load
```

Exit and reopen the terminal, run the following on the command line:

```
type rvm | head -n 1
```

If 'rvm is a shell function' appears RVM has successfully installed.

A stable version of ruby should have been installed along with RVM, to check run the following on the command line:

```
rvm list
```

if the latest version of ruby (at the time of writing this blog : v 2.1.1) is shown skip to section 2, if not following the next set of instructions.

A version of ruby must now be installed, run the following on the command line:

```
rvm install 2.1.1
```

set it as the default ruby version

```
rvm --default use 2.1.1
```

2. Install Chrome and ChromeDriver

For the purpose of this project we will be using Chrome to run our tests in. In order for the tests to talk to chrome we need a handler that enables selenium libraries, so we must use chromedriver.

If you haven't previously installed the chrome browser, get it from <http://www.google.co.uk/chrome> (<http://www.google.co.uk/chrome>)

Once Chrome is installed you must now download and install ChromeDriver, within the terminal type

```
brew install chromedriver
```

3. Installing Bundler

Our project will rely on a number of gems for it to function. Bundler is a great tool for managing all the gem dependencies, for more info on bundle visit <http://bundler.io/> (<http://bundler.io/>)

To install bundler, open the terminal and run

```
gem install bundler
```

4. Initial Project Setup

Now it's time to set up our project and install all the gems we need to get started. First make a new directory where you would like to keep your project.

```
mkdir cucumber_project
```

```
cd cucumber_project
```

Once the directory has been created, cd into the directory and create a new Gemfile by running:

```
bundle init
```

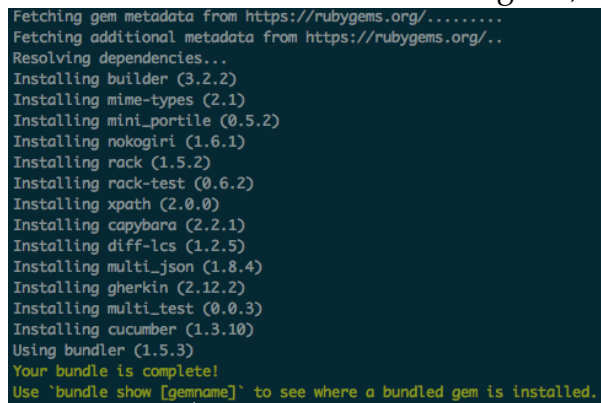
The Gemfile will list all the gems we need in order to get our simple cucumber project up and running. Open the gem file with your favorite text editor add the following:

```
1 source 'https://rubygems.org (https://rubygems.org)'  
2 gem 'capybara'  
3 gem 'cucumber'  
4 gem 'selenium-webdriver'
```

The listed gems must now be installed along with their dependencies, close the Gemfile and within the cucumber_project on the command line line run

```
bundle install
```

Bundler will now install the listed gems, once completed you should see something like this



```
Fetching gem metadata from https://rubygems.org/.....  
Fetching additional metadata from https://rubygems.org/..  
Resolving dependencies...  
Installing builder (3.2.2)  
Installing mime-types (2.1)  
Installing mini_portile (0.5.2)  
Installing nokogiri (1.6.1)  
Installing rack (1.5.2)  
Installing rack-test (0.6.2)  
Installing xpath (2.0.0)  
Installing capybara (2.2.1)  
Installing diff-lcs (1.2.5)  
Installing multi_json (1.8.4)  
Installing gherkin (2.12.2)  
Installing multi_test (0.0.3)  
Installing cucumber (1.3.10)  
Using bundler (1.5.3)  
Your bundle is complete!  
Use 'bundle show [gemname]' to see where a bundled gem is installed.
```

(<https://sifranklin.files.wordpress.com/2014/02/screenshot-2014-02-17-at-9-52-53-pm.png>)

You will also notice a Gemfile.lock file has appeared alongside the GemFile, this means that the project now has a record of the exact gem versions that you are currently using. To read more about gems see <http://guides.rubygems.org/> (<http://guides.rubygems.org/>)

5.Cucumber Project

Its now time to set up the cucumber project. The first thing we need to do is create a features directory along with an actual feature file. Within the cucumber_project directory

```
mkdir features
```

```
cd features
```

```
touch google_search.feature
```

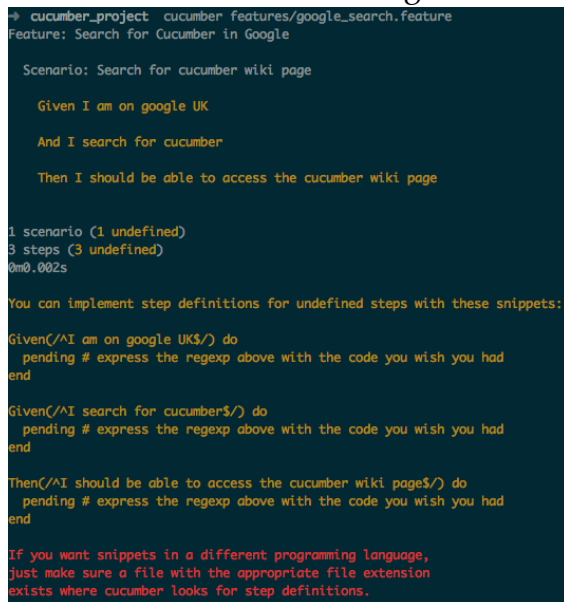
Open the google_search.feature file using your text editor and add the following:

```
1 Feature: Search for Cucumber in Google
2
3 Scenario: Search for cucumber wiki page
4   Given I am on google UK
5   And I search for cucumber
6   Then I should be able to access the cucumber wiki page
```

After saving the first feature file close and attempt to run the feature file from the projects home directory

```
cucumber features/google_search.feature
```

You should see the following



```
➔ cucumber_project cucumber features/google_search.feature
Feature: Search for Cucumber in Google

  Scenario: Search for cucumber wiki page

    Given I am on google UK

    And I search for cucumber

    Then I should be able to access the cucumber wiki page

1 scenario (1 undefined)
3 steps (3 undefined)
0m0.002s

You can implement step definitions for undefined steps with these snippets:

Given(/^I am on google UK$/) do
  pending # express the regexp above with the code you wish you had
end

Given(/^I search for cucumber$/) do
  pending # express the regexp above with the code you wish you had
end

Then(/^I should be able to access the cucumber wiki page$/) do
  pending # express the regexp above with the code you wish you had
end

If you want snippets in a different programming language,
just make sure a file with the appropriate file extension
exists where cucumber looks for step definitions.
```

(https://sifranklin.files.wordpress.com/2014/02/first_run.png)

Before we create the steps definitions we must create a project environmental file within a support directory

```
mkdir features/support
```

```
touch features/support/env.rb
```

Open the env.rb file and enter the following

```

1  require 'capybara'
2  require 'cucumber'
3  require 'selenium-webdriver'
4
5  Capybara.default_driver = :selenium
6  Capybara.app_host = "http://www.google.co.uk (http://www.google.co.uk)"
7  Capybara.register_driver :selenium do |app|
8    Capybara::Selenium::Driver.new app, browser: :chrome
9  end
10
11 World(Capybara::DSL)

```

We are now ready to create the step definitions, create a folder along with a step definition file. This file will hold the steps we need in order for the feature file to execute. Within the projects home directory

```
mkdir features/step_definitions
```

```
touch features/step_definitions/google_search_steps.rb
```

Now copy over the steps which cucumber has kindly provided for us, copy the steps from the terminal and paste into our newly created step definitions file and remove the lines of code that include 'pending #' so the steps file should look like this

```

1  Given(/^I am on google UK$/) do
2    end
3
4  Given(/^I search for cucumber$/) do
5    end
6
7  Then(/^I should be able to access the cucumber wiki page$/) do
8    end

```

Now we need to define the steps, copy and paste the following into your google_search_steps.rb file.

```

1  Given(/^I am on google UK$/) do
2    #visit method goes to google.co.uk the specified Capybara.app_host
3    visit '/'
4  end
5
6  Given(/^I search for cucumber$/) do
7    #fill the cucumber search box
8    fill_in 'gbqfq', :with => 'cucumber'
9    #click on the google search button
10   click_on 'gbqfb'
11 end
12
13 Then(/^I should be able to access the cucumber wiki page$/) do
14   #use xpath to find the link we require and click it
15   find(:xpath, "//a[contains(., 'Cucumber - Wikipedia, the free encycl
16   #test the h1 title is equal to Cucumber
17   find('h1').text == 'Cucumber'
18 end

```

Save and close the file, now back to your terminal, on the command line run

```
cucumber features/google_search.feature
```

There you have it!

We now have a fully functional testing framework!

If you would like a more in depth explanation on the code within the steps I would recommend taking a look at the documentation available on both the cucumber and capybara git hub pages.

<https://github.com/cucumber/cucumber> (<https://github.com/cucumber/cucumber>)

<https://github.com/jnicklas/capybara> (<https://github.com/jnicklas/capybara>)

Happy Testing 😊

Advertisements

Bookmark the permalink.

2 thoughts on “Getting started with Ruby, Cucumber and Capybara”

1. Pingback: [Setting up Calabash Android and Android Emulator | Life and Times of an Automation Tester](#)

2. Ravi Kiran

July 27, 2015 at 12:03 pm

Good article:)

it would be great “"” , and “>” should be shown the way they are used in script .. else beginners will get stuck 😞

Reply

Blog at WordPress.com.