

Automatización en Ruby 101

Denis Rodriguez
Santiago Vazquez



Soy Denis

QA
Altimetrik
Salesforce



Altimetrik y Salesforce

Altimetrik:

Empresa de software que tiene mas de 8 años en Uruguay

Lider en cloud computing

Especializada en Salesforce

Salesforce:

CRM que ofrece soluciones de Cloud computing



Solución diferente

Proyecto web

Extensión superior al año de trabajo

Cantidad de casos de prueba

Ciclos de ejecución demandantes

La conclusión es que la automatización es necesaria.



¿Qué es Ruby?

Lenguaje de programación dinámico

Código abierto

Simple y de sintaxis elegante

Fácil de leer y de escribir

Flexible y versatil

Portable



ATOM - Increíble editor de texto

Atom es un editor de texto open source altamente customizable

Permite la instalación de plugins para trabajar en entornos específicos

Tiene paquetes útiles para trabajar con Ruby

<https://atom.io/>



Rspec

```
drafts_page_spec.rb
describe Drafts do
  let(:current_user) do
    instance_double(User, :first_name => 'UnitTestUser')
  end
  let(:params) do
    {name: 'unit_test_page', head_layout: '', header_layout: '', nav_layout: '',
     related_to: '', layout: 'dynamic', use_head_layout: 'false', use_header_lay
     use_footer_layout: 'false', page_layout_id: PageLayout.first.id, use_templa
     proxy_cache_secs: '', version_comment_hdn: 'This is the comment for testing
  end
  let(:locale_params) do
    {en: {'page_title' => 'EN title', 'meta_description' => 'EN MD', 'head' => '
     de: {'page_title' => 'DE title', 'meta_description' => 'DE MD', 'head' => '
     ja: {'page_title' => 'JA title', 'meta_description' => 'JA MD', 'head' => '
     fr: {'page_title' => 'FR title', 'meta_description' => 'FR MD', 'head' => '
    }
  end
```



Cucumber intro

¿Qué es?

Es una herramienta de programación usada para correr casos de prueba automatizados escritos en BDD (Behavior-Driven development)

Está escrito en Ruby pero también existen versiones soportadas para JAVA

Es orientado al cliente

Está dividido en **Features**, subdividido en **Scenarios** los que contienen **Steps Definitions**



Cucumber caso de prueba

```
salesforce_login.feature
```

```
Feature: Login to Salesforce and create a simple object
```

```
Scenario: Login to Salesforce and create a simple object
```

```
Given I am in Salesforce Login page
```

```
Then I enter a valid username
```

```
And I enter a valid password
```

```
And I click the Login button
```

```
Then I check that I am logged in Salesforce Org
```

```
And I write Objects in the Quick Find text field on Salesforce Setup page
```

```
And I click Create > Objects link
```

```
And I click New Custom Object button inside Custom Objects page
```

```
Then I check that I am in New Custom Object page
```



Cucumber estructura

salesforce_login.feature

Feature: Login to Salesforce and create a simple object

Scenario: Login to Salesforce and create a simple object

Given I am in Salesforce Login page

Then I enter a valid username

And I enter a valid password

And I click the Login button

Then I check that I am logged in Salesforce Org

And I write Objects in the Quick Find text field on Salesforce Setup page

And I click Create > Objects link

And I click New Custom Object button inside Custom Objects page

Then I check that I am in New Custom Object page

salesforce_login_steps.rb

```
Given(/^I am in Salesforce Login page$/) do
```

```
  @browser.goto 'login.salesforce.com'
```

```
end
```

```
Then(/^I enter a valid username$/) do
```

```
  @browser.text_field(:css => '#username').set 'your login username here'
```

```
end
```

```
And(/^I enter a valid password$/) do
```

```
  @browser.text_field(:css => '#password').set 'your password'
```

```
end
```

```
And(/^I click the Login button$/) do
```

```
  @browser.button(:css => '#Login').click
```

```
end
```

```
Then(/^I check that I am logged in Salesforce Org$/) do
```

```
  @browser.wait_until(30) {@browser.image(:css => '#phHeaderLogoImage').exist?}
```

```
end
```

```
And(/^I write Objects in the Quick Find text field on Salesforce Setup page$/) do
```

```
  sleep(3)
```

```
  @browser.text_field(:css => '#setupSearch').set 'Objects'
```

```
end
```

Cucumber Palabras Clave de Test Scenario

Un **Scenario** está definido por una secuencia de **Steps**.

Las palabras clave en los steps son:

Given: Estado inicial antes de iniciar el test, precondition

When: Acción a realizar por el usuario en el test

Then: Resultado esperado de la acción realizada

También se usan para combinar las palabras clave uniones lógicas

And: Misma acción que en expresiones lógicas

But: Igual que el And pero usada de manera negativa



Cucumber 3 estados de ejecución

Pass

Done: Scenarios 1 of 1 (1m 45s 141ms)

```
/Users/denisrodriguez/.rvm/rubies/ruby-2.3.0/bin/ruby -EUTF-8 -e $stdout.sync=true;$stderr.sync=true;load($0=ARGV.shift) /Users/denisrodriguez/.rvm/gems/gems/cucumber-3.0.0/bin/cucumber --format progress --strict --tags '@@login, @@sanity_check, @@sanity_check_production, @@regression_testing, @@remediation_task, @@logout'
Testing started at 10:18 AM ...
Tags: @@login, @@sanity_check, @@sanity_check_production, @@regression_testing, @@remediation_task, @@logout
1 scenario (1 passed)
5 steps (5 passed)
1m40.290s

Process finished with exit code 0
```



Cucumber 3 estados de ejecución

Fail

Done: Scenarios 1 of 1 Failed: 1 (1m 18s 918ms)

Testing started at 10:21 AM ...

Tags: @login, @sanity_check, @sanity_check_production, @regression_testing, @remediation_task, @logout

RSpec::Expectations::ExpectationNotMetError:

expected true
got false

./features/step_definitions/holy_grail/holy_grail_steps.rb:13:in `/^I check that I am logged in\$/'

./features/dfc_automation/main_page/login/english_login.feature:9:in `Then I check that I am logged in'

1 scenario (1 failed)

5 steps (1 failed, 4 passed)

1m12.010s



Cucumber 3 estados de ejecución

Error

Done: Scenarios 1 of 1 Failed: 1 (35s 523ms)

Tags: @@login, @@sanity_check, @@sanity_check_production, @@regression_testing, @@remediation_task, @@logout

```
Watir::Exception::UnknownObjectException: unable to locate element, using {:css=>"#password-error", :tag_name=>"input or textarea", :type=>"(any text (eval):1:in `process_watir_call`
```

```
./features/support/pages/salesforce/sf_login_page.rb:104:in `login_window'
```

```
./features/support/pages/salesforce/sf_login_page.rb:25:in `login_operation'
```

```
./features/step_definitions/salesforce/sf_login_steps.rb:2:in `/^I login in the Salesforce login page( with the .pass org)?$/'
```

```
./features/dfc_automation/main_page/login/english_login.feature:8:in `And I login in the Salesforce login page'
```

Skipped step1 scenario (1 failed)

5 steps (1 failed, 1 skipped, 3 passed)

0m31.338s



Cucumber con SauceLabs

The screenshot displays the SauceLabs web interface. At the top, the SauceLabs logo is on the left, and navigation links for 'Join the Beta', 'Upgrade', 'Resources', 'Docs', 'Platforms', and 'devforrester' are on the right. Below the navigation is a tabbed interface with 'All tests' selected. The main content area is divided into four columns: 'Commands', 'Screencast', 'Selenium Log', and 'Metadata'. The 'Commands' column lists several test steps with their durations:

| Command | Duration |
|--|--------------------|
| POST execute args: [{"element-6066-11e4-a52e-4f735466cec": "0.28088211009520125-9"; ELEMENT... script: "arguments[0].scrollIntoView();"}] => null | 3m 22.01s (+0.00s) |
| POST execute args: [] script: "window.scrollBy(0, -100)" => null | 3m 22.80s (+0.02s) |
| GET window_handle => "CDwindow-20B7E454-08E9-46D0-BDF1-791BBA3A55..." | 3m 23.04s (+0s) |
| GET window_handles => ["CDwindow-20B7E454-08E9-46D0-BDF1-791BBA3A55..."] | 3m 23.21s (+0.01s) |
| POST frame id: null => null | 3m 23.38s (+0.01s) |
| POST element using: "css selector" value: "challenge-check-btn" => [{"ELEMENT": "0.28088211009520125-9"}] | 3m 23.55s (+0.01s) |
| GET element/0.28088211009520125-9/name | 3m 23.72s (+0.01s) |

The 'Screencast' column shows a video player with the title 'SCREENCAST >'. The video content shows a browser window with the URL 'https://developer.salesforce.com/trailhead/starting_force_com/starting_intro'. The page title is 'Getting Started with the Platform'. The main content area of the browser shows a challenge titled 'Create a simple garage manager app using the Force.com App Quick Start Wizard'. The challenge instructions are:

- Using the App Quick Start Wizard, create an app named 'Garage' whose main type of data will be 'Vehicle'.
- If you haven't already, sign up for a Developer Edition [here](#) to use when completing challenges in Trailhead (note: Don't use an old Developer Edition org, as this may cause problems when verifying your challenges).
- The App Quick Start Wizard must create a resulting application in the app drop down named 'Garage'.
- The 'Garage' application must have a tab named 'Vehicles' (plural label).

There is a 'Check Challenge' button and a 'Challenge' badge with '+500 points'. A sidebar on the right lists 'Topics' such as 'Learning Objectives', 'A Quick Introduction to Salesforce', 'Devising the Terminology', 'Using the Platform', 'Clicks and Code', 'Signing Up for a Developer Edition Organization', 'What's an App?', 'Building a Fundraising App', and 'Resources'. At the bottom of the sidebar, there is a 'Have questions about Trailhead or having problems using IT?' section with a 'Trailhead Forum' link and a 'Submit Feedback' button.



Principales ventajas de automatización

Los test pueden ser re usados

Correr tests de manera rápida y eficaz

Todos los integrantes del equipo pueden ver los resultados

Aunque tome tiempo al principio. Una vez que los casos de prueba son automatizados, el proceso se vuelve mucho más rápido

Es más interesante escribir/codear un Test Case en Cucumber que llenar manualmente la misma Form 100 veces en un test manual

Quien dijo que los QA no pueden escribir código?



Conclusión

Todos los proyectos pueden automatizar

Mas eficiencia en proyectos largos

Ruby es una herramienta estupenda para empezar



¡Muchas gracias!