iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** V **Month of publication:** May 2018

DOI: http://doi.org/10.22214/ijraset.2018.5121

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue V, May 2018- Available at www.ijraset.com

Automation Script Development using Capybara

Yesha N B[1], Dr. Jitendranath Mungara[2]
[1]*Student, B.E, Information Science and Engineering Department, NHCE, Bangalore, India*
[2]*Head of Department, Information Science and Engineering Department, NHCE, Bangalore, India*

Abstract: With the advent of technology and Software Development, testing has become a major activity and gained focus in the software development life cycle in order to produce and deliver quality software. Testing involves finding defects and errors in the early stages of the development process. Testing is very crucial and important activity that has to be carried out for detecting faults in the application. This paper focuses on Automation based testing for testing the Graphical User Interface of the application. One of the trending testing technologies called Capybara testing along with RSpec framework to test Ruby applications is discussed in detail. The Paper also briefs about the drawbacks of manual testing and the effectiveness of automation testing.

Keywords: GUI, Capybara, RSpec, Web Driver, Web browser, Automation testing.

I. INTRODUTION

With the enhancement in software development and production in recent years, it has led to major concern over reliability and security of the application[2]. Software testing has become an important and crucial stage in the software development life cycle. Life critical applications needs to be highly available and accurate for it to be successful. This accuracy is achieved by carrying out the best practices in testing the software. By employing effective testing techniques quality software can be delivered to the customer.

GUI (Graphical User Interface) testing is the process of testing the Graphical User Interface of the application to ensure that it fulfils the stated requirements in the business requirement document[1]. Graphical User Interface is the most essential and eminent parts of the software used in today's Information Technology[1].

Graphical User interface testing is carried out using various testing frameworks that generate the user interface events such as mouse clicks and keystrokes and observe the changes in the user interface and compare the results obtained with the expected behaviour and success or failure is reported[9]. GUI testing tests the various controls on the screen like icons, menus, buttons, toolbar, menu bar, windows and dialog boxes, etc[9].

GUI testing can be carried out in two ways, manual testing and the automation based testing.

Manual testing is carried out by humans whereas in automated testing the tests are carried out automatically by automation tools[11].

Manual testing is carried out by a tester, who plays the role of an end user to check if all the elements of the application are working properly. Manual testing finds any of the defects or bugs in the software application[3]. The application is manually tested by a tester to see if it is conforming with the requirements stated in the business requirements document. Manual testing is often prone to errors, less accurate as the test cases are manually carried out. It is very tedious process as same tests need to be executed repetitively and requires more time and resources.

Automation Testing involves testing the various elements of the software application using automation tools and testing frameworks. Automation testing is the best way to carry out the UI testing as it improves accuracy, increases tests coverage, saves time and resources[10].

Capybara is a web-based automation software that simulate how the user interacts with the application. The capybara test scripts simulate the various aspects of the browser just like how the real end user would experience. This paper focuses on how the test automation scripts are developed using capybara along with the RSpec framework.

II. GUI TESTING USING CAPYBARA – RSPEC

A. Capybara

Is a testing tool which is designed to be able to use the web application just like how the real end user would automatically and find the possible defects or bugs in the web application[7].

Capybara is a library/Gem which has it's own set of user-friendly DSL(Domain Specific Language) which is used for describing the various action that are executed by the web-driver. Capybara is used on top of the underlying web-driver[5].

The following web-drivers are supported by capybara:

1) *rack::test :* Capybara uses rack::test driver as it's default driver. It is written in pure Ruby. With rack::test driver the JavaScript elements of the application cannot be accessed and tested[5].And the HTTP resources cannot be accessed outside of the application[5]. For this the server need not be started because Capybara directly interacts with the rack test[6]. Therefore it can be used only for the Rack applications.
2) *Selenium Web-Driver :* Selenium is most commonly used in web-based automation frameworks. Capybara supports Selenium 2.0 and greater versions. JavaScript can be executed using Selenium. HTTP resources can be accessed with the Selenium web-driver. selenium-webdriver gem has to be installed in order to use Selenium[6]. In order to use this driver you will need to add the following line in the setup: Capybara.default_driver = :selenium
3) *Capybara Web-Kit :* actually opening the browser. The tests are executed in the background. Using headless time consumption needed to carry out the test can be reduced. It supports JavaScript and it is faster when compared to Selenium as it doesn't have to load the entire browser[5]. In order to use this driver you will need to add the following line in the setup: Capybara.default_driver = :webkit

B. RSpec

=RSpec is a Testing framework written in Ruby for testing the Rails applications. Capybara test scripts are written within the RSpec framework. RSpec provides syntax which provides easy readability and describes the behavior of the code. The flow of the RSpec Testing framework works in the following manner: given a context, when some event occurs, what is the outcome expected[8]. The keywords describe(), context() and it() blocks form the skeleton of the test code.

1) *describe() :* describe() method is used to describe an example group. This is the outermost block which contains the test code. Describe block can be nested as well. All the capybara scripts are written within this block. It takes the argument that describes the behavior of the example group[8].

```
describe "Registration" do
        ……..
 end
```

2) *context() :*context() block describes the context of the class or method in the describe block. describe() and context() can be used interchangeably[8].

```
describe "Registration" do
    context "name field not filled" do
        ……..
    end
end
```

3) *it() :*it() can be thought of as the actual test case. it() block takes a String as an argument. The actual test code that is written inside the block is expected to perform the action[8].

```
describe "Registration" do
    context "name field not filled" do
        it "should not submit the form" do
            …….
        end
    end
end
```

III.MODULES INVOLVED IN TESTING THE GUI USING CAPYBARA

A. User Interface

The user interface of the application to be tested has to be developed first using programming languages. Then it is given for testing the various controls on the screen like menus, buttons, dialog boxes and windows, etc.

B. Viewing the source Code of Application

1) The source code of the application is viewed using the Inspect element of the browser.
2) Inspect element allows to view the Visual elements of the web page that is hidden from the user.

3) By viewing the HTML and CSS source code we can get the IDs and the class selectors of the tags that serve as major elements for writing the capybara tests.

C. *Writing the Test Script*

1) The Test scripts are written in Capybara using RSpec Framework.
2) The various test cases are written in Capybara within the underlying Rspec Framework.

D. *Use the Web Driver to Automate*

1) Web Driver allows Capybara to interact with the Web Browser. It is an intermediate between these two. Web-Driver understands the capybara tests and translates to the way the browser understands.
2) Two most effective web drivers used to carry out the tests are:
3) Selenium Web Driver
4) Poltergeist- for headless execution

E. *Browser to Run the Test:*

1) Using selenium web driver and Firefox browser installed.
2) Using PhantomJS – PhantomJS is headless browser which is used along with the Poltergeist web-driver for automation[4].

IV. PROCESS INVLOVED

A. The automation scripts are written in Capybara for different web pages in an application using Rspec Framework. The inspect element of the browser helps to find the controls on the screen like menus, dialog boxes, tables, buttons and links ,etc. using unique IDs and selectors for each of the tags
B. Once the Capybara tests are written according to the business requirement document, they are executed and the results are compared with the specification
C. The Capybara tests are interpreted by the web driver(Selenium or Poltergeist).
D. The web driver translates the scripts into the way the browser understands and tells the browser what to do.
E. When the test is run, the browser automatically opens and will complete all the assertion as per the test script and once execution is done the success or failure message is presented. When headless browser is used, the test execution is carried out in the background without opening the browser and results are presented.
F. In this way the whole UI of the application is validated automatically just like the end user interacts with user interface.

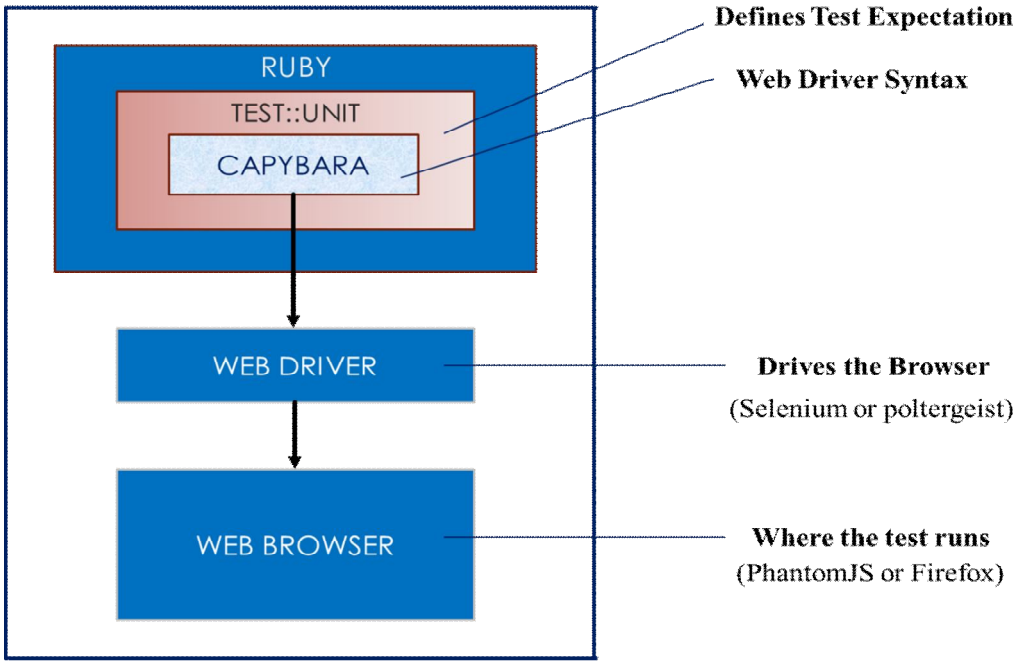


Fig.1. SYSTEM DESIGN

V. TESTING OF A SIMPLE LOGIN SCREEN USING CAPYBARA

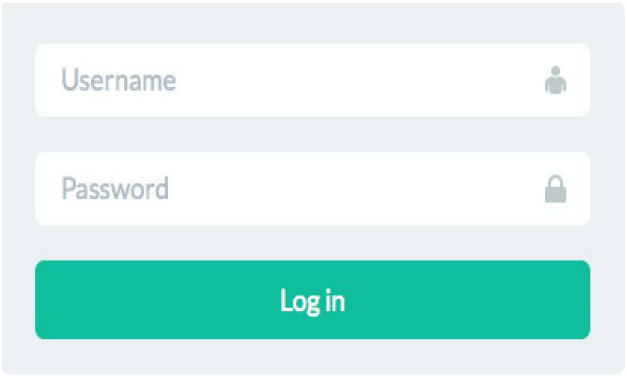


Fig.2. UI of a Login Screen

```
describe "Sign in" do
  before do
    fill_in "username", with: 'xyz'
    fill_in "password", with: 'foo123'
    click_link('login')
  end

  it "should be able to log in with valid credentials" do
    page.should have_content('Welcome...')
  end
end
```

Fig.3. Snapshot of code Example

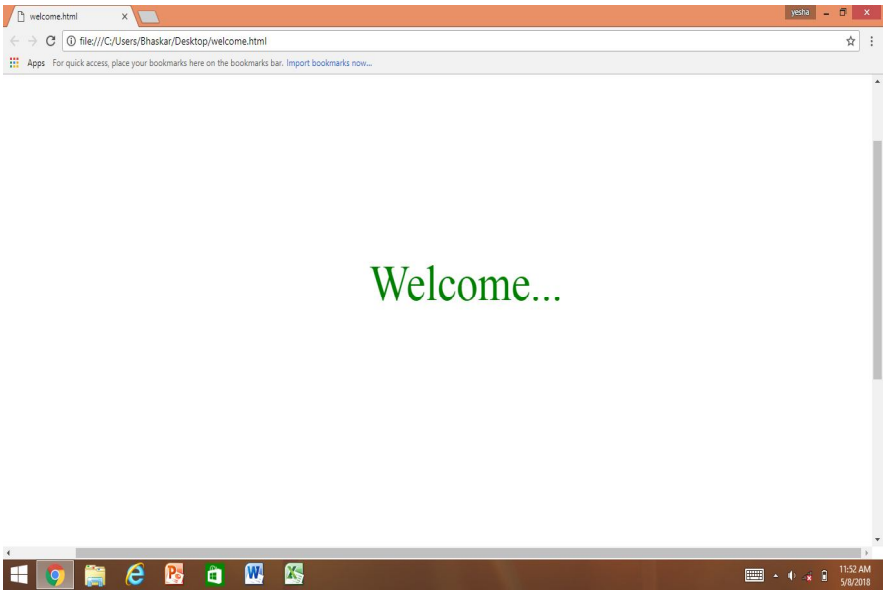


Fig.4. Snapshot of redirection to next page when test case passes

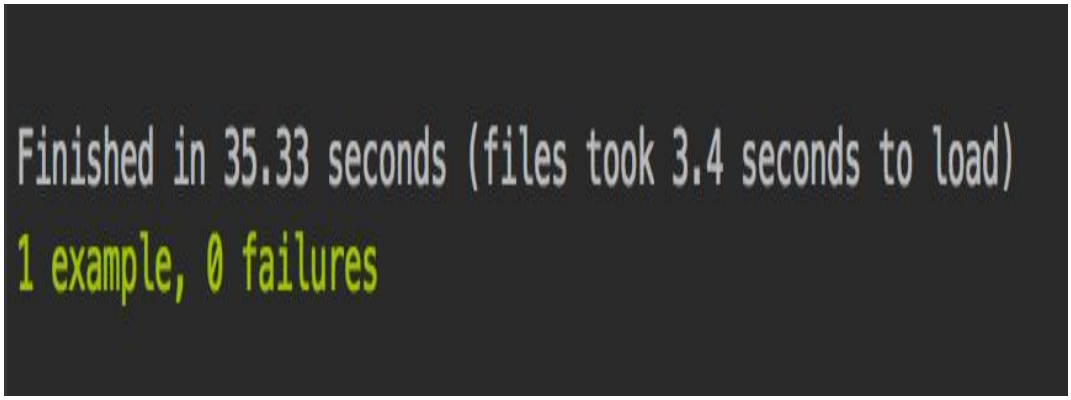


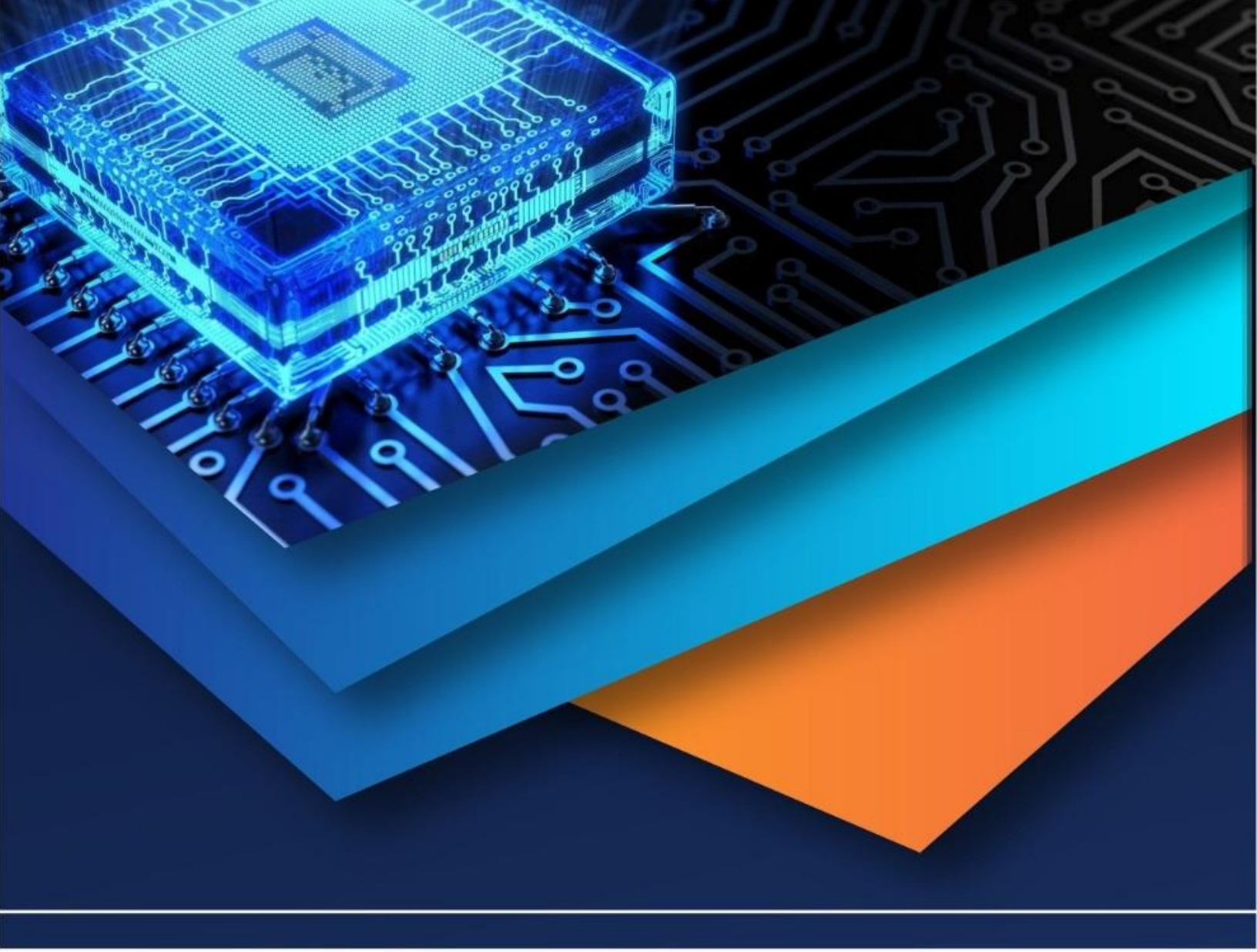
Fig.4. Result of the code Example

VI. CONCLUSION

Automated testing is the best way to test the user interface of the application[3]. Using automation the effectiveness, efficiency and test coverage can be increased[3]. There is not much overhead involved in automation testing. Once the test scripts are written, they can be repetitively executed any number of times unlike manual testing which requires more time and resources. This paper covers one of the most efficient GUI testing methodology called Capybara where capybara along with the RSpec framework can be used for complete testing of all the controls on the screen. A brief explanation about Capybara testing, various Web-Drivers supported by capybara and detailed explanation about RSpec framework is discussed. Capybara testing tool tests the web application just like how the real end user would experience. This paper also discusses the modules and the process involved in testing the user interface along with the system design. A code example for one test case using capybara is written for a simple Log in screen and result is shown.

REFERENCES

[1] Isabella, Emi Retna 'Study Paper on Test Case Generation For GUI Based Testing', International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
[2] Ritu Patidar, Anubha Sharma, Rupali Dave, 'Survey on Manual and Automation Testing strategies and Tools for a Software Application' International Journal of Advanced Research in Computer Science and Software Engineering, Volume 7, Issue 4, April 2017.S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.
[3] Vishawjyoti, Sachin Sharma, 'Study and Analysis Of Automation Testing Techniques', Journal of Global Research in Computer Science, Volume 3, No. 12, December 2012.
[4] GitHub, Inc.[Online], Available: https://github.com/RefugeRestrooms/refugerestrooms/wiki/What-is-Capybara%3F-What-is-PhantomJS%3F-What-is-Poltergeist%3F
[5] [Online], Available: https://www.sitepoint.com/basics-capybara-improving-tests/
[6] GitHub, Inc, [Online], Available: https://github.com/teamcapybara/capybara
[7] (9 December 2017), Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/Capybara_(software)
[8] (5 April 2018), Wikipedia, [Online], Available: https://en.wikipedia.org/wiki/RSpec
[9] (1998), J.M. Clarke. Automated test generation from a Behavioral Model. In Proceedings of Pacific Northwest Software Quality Conference. IEEE
[10] Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press
[11] Manual Specific Testing and Quality Evaluation for Embedded Software, Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY