Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 6, June 2016, pg.368 – 371

Web Browser Application-Test Automation Using Selenium Webdriver with Junit Framework

Rakshith C A[1], Varun Sharma N[2]

[1]Student, Department of CS, RIT, Hassan- 573201, INDIA
[2]Student, Department of CS, RIT, Hassan- 573201, INDIA
[1] rakshithca@gmail.com; [2] varun15sharma15@gmail.com

Abstract— Most of the software applications today are written and hosted as Web based applications, and can be run on any browsers. To test these applications with the agile methodology, there is a need for test automation, which means a software tool is required to run repeatable tests against the application to be tested. In this paper, one of the open source tools, Selenium Webdriver is used for automation testing of web based applications using Java.

Keywords— Selenium, Webdriver, Web Browser Application, Test Automation, Junit Framework

I. INTRODUCTION

Today most of the software applications are hosted as web based rather than the standalone applications. In order to reduce the risk due to any unknown factors during the design and development phases of a new software application, and to reduce uncertainty and eliminate errors, testing is necessary. Testing is the only way to eliminate defects, even though planning, design and quality analysis can reduce the number of defects.

In order to run repeatable tests against the application to be tested and for getting responsiveness for regression testing, test automation is required. Automation requires a tool to run these repeatable tests and these tools can be open source tools or vendor developed tools. Selenium is one such available as an open source tools for web application testing. It can be used extensively for testing any application hosted by any variety of application vendors. When integrated with the right framework like Junit or TestNG, it provides an accurate cost effective way of testing the performance of an application.

Selenium is available in the form of GUI based tool as Selenium IDE, Integrated Development Environment, which provides a simple record and play features for creating the suitable test suites and test cases of application under test. This component of selenium is used for prototype test case developments and is restricted to applications opened through Mozilla Firefox browser. For advanced test case development on applications that are dynamically changing or AJAX implementations, and to make the test case independent of browsers like Internet Explorer and Chrome, Selenium Webdriver is required.

II. AIM OF STUDY

To understand the various locator strategies to interact with web elements of an application and develop end to end test cases for automation testing using Selenium Webdriver and integrating it with Junit framework for generation of reports regarding the test case pass/fails of execution.

III.RELATED WORK

Web application automation testing with Selenium Webdriver using Java:

Selenium webdriver provides the flexibility of writing the scripts using many programming languages like Java, C#, Python, Ruby, etc. We have used Eclipse Indigo Java EE developer editor for developing the test cases using Java APIs of Selenium. jdk1.7.0_45 version of Java is used for Java build path and selenium 2.50.0 jar files are used as API for interacting with various web elements and for writing the complete code. These selenium jar files provides a lot of methods which can be invoked by importing the suitable classes or interfaces implemented.

The code is integrated with Junit framework, with @Test, @Before and @After annotations.

The function for setup under @Before is for opening the browser. Sample code is shown for opening the given web application in Firefox browser. The get method opens the URL of the respective web application in the chosen browser. Once the web application under test is opened, the respective web element for logging into the website is located using the suitable locator strategy. Any element can be uniquely identified using various locator strategies like ID, name, class, xpath or the CSS selector. Once the elements are located, respective actions need to be performed, like clicking on the link or button, sending some text input into the text fields, choosing an element under drop box etc. Timeouts are used to check for loading of the given web element, otherwise time out exception will be thrown.

In the code provided below, we have used a combination of the locator strategies and suitable actions are performed. All these activities are categorized under @Test of Junit, to accomplish all the test validations with the help of assert and verifications. These assertions and verifications are done for validating the expected values with that of actuals obtained by interacting with the web elements. After the completion of the test execution, @After function code is executed to close the respective applications opened. The sample code used for this study is provided below.

```
import java.util.regex.Pattern;     // selenium jar files imported onto the code space
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class MakeMyTripTC {                 // creating a java class
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before                                   // Junit annotation to specify what needs to be done before test
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "http://www.makemytrip.com";  // defining the base URL of makemytrip.com
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test                                     // Junit annotation to specify what needs to be done during  test
  public void testMakeMyTripTC() throws Exception {
```



```
    driver.get(baseUrl + "/");   //open url
    //driver.findElement(By.xpath("//*[@id='ssologinlink']")).click(); //find login element and click

    driver.findElement(By.id("username")).clear();
    driver.findElement(By.id("username")).sendKeys("user id"); // type userid into the userid field

    //driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("password"); //type password into password field
    driver.findElement(By.id("login_btn")).click();            //click on login button
    assertEquals("MakeMyTrip, India's No 1 Travel Site | Book Flights, Hotels, Holiday Packages & Bus
Tickets", driver.getTitle());
    driver.findElement(By.cssSelector("#round_trip_button1 > span.radio_state")).click(); //validate if the right
page is opened
    driver.findElement(By.id("from_typeahead1")).click();   //fill in the location details
    driver.findElement(By.id("to_typeahead1")).click();
    driver.findElement(By.xpath("//a[@id='start_date_sec']/span[2]")).click(); //fill in the journey date details
    driver.findElement(By.linkText("31")).click();
    driver.findElement(By.xpath("//a[@id='return_date_sec']/span[4]/span[2]/span")).click();
    driver.findElement(By.linkText("29")).click();
    driver.findElement(By.id("flights_submit")).click();  //submit once all the fields are filled up
    assertEquals("Flight Split Listing View", driver.getTitle());
  }

  @After              // Junit annotation to specify what needs to be done after test
  public void tearDown() throws Exception {
    driver.quit();       //close the browser
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {    //assert if the element is present
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {       //assert if the alert is present
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {  //check if alert is closed
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
```



```
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

IV. RESULTS

The screenshot of the sample test reports are shown below. The various test cases are validated and the reports of test case failures and passes are generated. As shown in the report, when the locator identification or inputs provided are not valid, the results are indicated as failure. With the right inputs, the test cases are shown as pass. Test case failures are obtained because of element not found or timeout or invalid input given; otherwise the test case pass is generated.

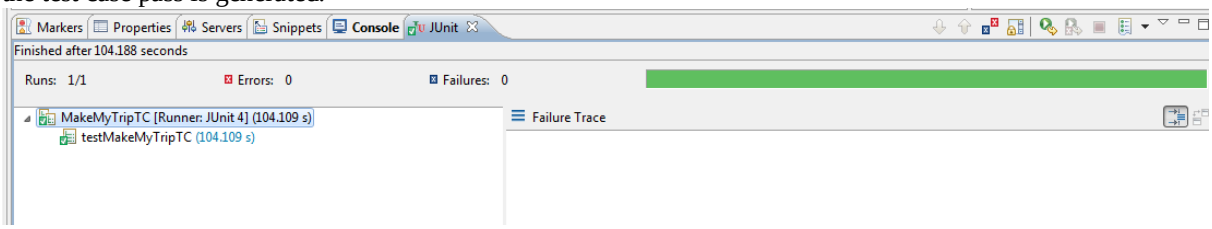

Fig.1 Snapshot of the result with complete test case pass report, with the right validations done for various assertions and verifications.

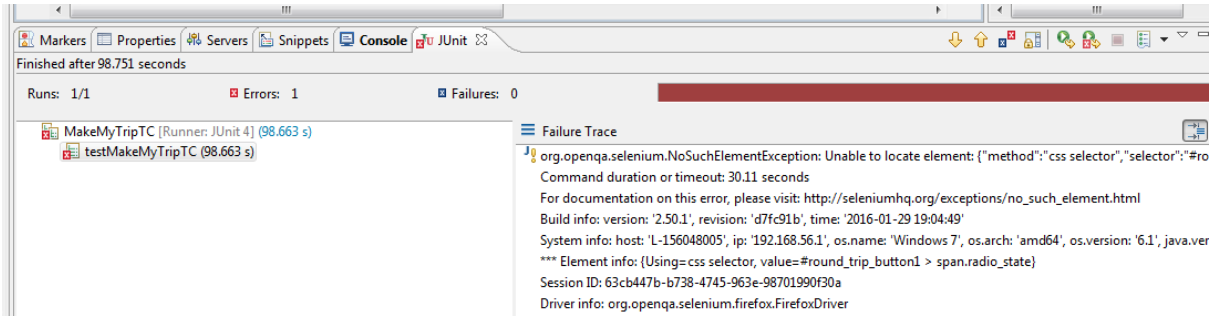

Fig.2 Snapshot of the report failure, because of element not recognized with valid locator identification.

V. CONCLUSIONS

In our work, we have attempted to test automate the chosen web application with the development of multiple test cases, to ascertain and validate the test results. By integrating it with Junit framework, we have been able to generate the test results in the specific format. Further work can be done to make the complete end to end application testing of any web application which can be browser independent.

REFERENCES

[1] http://docs.seleniumhq.org/download/
[2] http://docs.seleniumhq.org/docs/
[3] http://www.makemytrip.com/
[4] http://www.nickjenkins.net/prose/testingPrimer.pdf
[5] https://en.wikipedia.org/wiki/Regression_testing

