# Selenium Webdriver Tool To Perform Auotmation Testing In Web Applications

**M.Archana Reddy**
M.Tech,
Department of Computer Science Engineering,
Srinivasa Institute of Technology and Sciences,
Kadapa.

**K. Rajasekhar Reddy**
M.Tech, Head of the Department,
Department of Computer Science Engineering,
Srinivasa Institute of Technology and Sciences,
Kadapa.

## Abstract:

Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language. Selenium Remote Control (RC) is a server, written in Java, that accepts commands for the browser via HTTP. RC makes it possible to write automated tests for a web application in any programming language, which allows for better integration of Selenium in existing unit test frameworks. In this paper we studied and implemented one extension of the Selenium RC tool to execute tests in web applications that involve checking data in databases. This work aspires to contribute to the system quality by reducing the effort during the testing process, since the verification of UI and database elements will be performed at the same time during execution of the test scripts. The basic idea of the project is to reduce the effort to perform testing in a software project, that is, we will be able to execute a highest number of test cases in less time.

## Keywords:

Testing framework, Selenium, Selenium Remote Control, Selenium IDE, Grid, WebDriver, Server.

## Introduction:

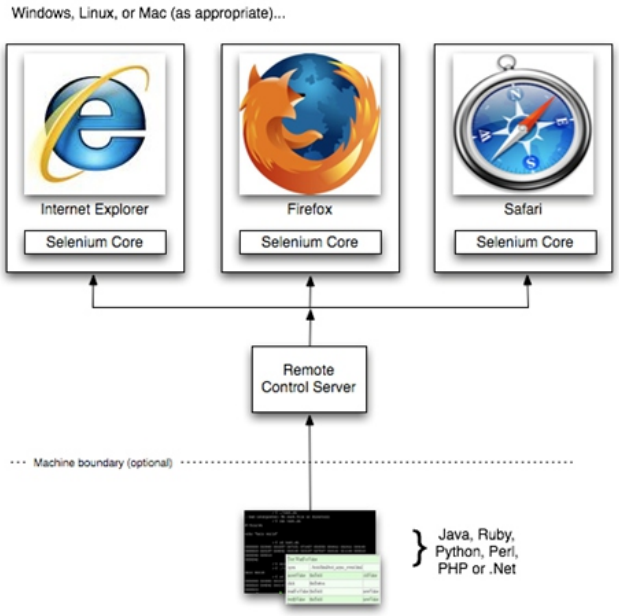
No matter how well you develop the project, it cannot be 100 % defect free .Thus every project must be tested thoroughly before it is released to the world. The Selenium framework is considered the most popular open source functional test tool for web-based applications Here we use the selenium RC automation tool with the extension to test the databases in web application.Selenium RC is a test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser.

## Selenium RC comes in two parts:

1.A server which automatically launches and kills browsers, and acts as a HTTP proxy for web requests from them.

2.Client libraries for your favorite computer language.

## Architecture:




The diagram shows the client libraries communicate with the Server passing each Selenium command for execution. Then the server passes the Selenium command to the browser using Selenium-Core JavaScript commands. The browser, using its JavaScript interpreter, executes the Selenium command. This runs the Selenese action or verification you specified in your test script. Selenium Remote Control is great for testing complex AJAX-based web user interfaces under a Continuous Integration system. It is also an ideal solution for users of Selenium IDE who want to write tests in a more expressive programming language than the Selenese HTML table format.

## Selenium Server:

Selenium Server receives Selenium commands from your test program, interprets them, and reports back to your program the results of running those tests. The RC server bundles Selenium Core and automatically injects it into the browser. This occurs when your test program opens the browser (using a client library API function). Selenium-Core is a JavaScript program, actually a set of JavaScript functions which interprets and executes Selenese commands using the browser's built-in JavaScript interpreter. The Server receives the Selenese commands from your test program using simple HTTP GET/POST requests. This means you can use any programming language that can send HTTP requests to automate Selenium tests on the browser.

## Client Libraries:

The client libraries provide the programming support that allows you to run Selenium commands from a program of your own design. There is a different client library for each supported language. A Selenium client library provides a programming interface (API), i.e., a set of functions, which run Selenium commands from your own program. Within each interface, there is a programming function that supports each Selenese command.

The client library takes a Selenese command and passes it to the Selenium Server for processing a specific action or test against the application under test (AUT). The client library also receives the result of that command and passes it back to your program. Your program can receive the result and store it into a program variable and report it as a success or failure, or possibly take corrective action if it was an unexpected error.

Selenese as Programming Code: Here is the test script exported (via Selenium-IDE) to each of the supported programming languages.(eg: Java) In order to understand how Selenium runs Selenese commands by reading one of these examples.

```
/** Add JUnit framework to your classpath if not already there
   *  for this example to work
  */
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class NewTest extends SeleneseTestCase {
  public void setUp() throws Exception {
    setUp("http://www.google.com/", "*firefox");
  }
   public void testNew() throws Exception {
     selenium.open("/");
     selenium.type("q", "selenium rc");
     selenium.click("btnG");
     selenium.waitForPageToLoad("30000");
    assertTrue(selenium.isTextPresent("Results * for selenium rc"));
  }
}
```

## The Selenium-IDE generated code will look like this. This example has comments added manually for additional clarity:

```
package com.example.tests;
// We specify the package of our tests

import com.thoughtworks.selenium.*;
// This is the driver's import. You'll use this for instantiating a
// browser and making it do what you need.

import java.util.regex.Pattern;
// Selenium-IDE add the Pattern module because it's sometimes used for
// regex validations. You can remove the module if it's not used in your
// script.

public class NewTest extends SeleneseTestCase {
// We create our Selenium test case

   public void setUp() throws Exception {
    setUp("http://www.google.com/", "*firefox");
      // We instantiate and start the browser
   }

   public void testNew() throws Exception {
     selenium.open("/");
     selenium.type("q", "selenium rc");
     selenium.click("btnG");
     selenium.waitForPageToLoad("30000");
       assertTrue(selenium.isTextPresent("Results * for selenium rc"));
     // These are the real test steps
   }
}
```

## Description of Modules:

Utility-UI: Utility is the user defined java package where the java classes contain the methods which invoke the page objects.

The logic to perform the required validation is done in these methods.Functionalities utilized for the execution is withheld in a Helper Class, contained in Utility-UI.

**Page Object:** Page Objects is the user defined package which contains the methods to access the UI elements (objects) of the webpage using the locators.

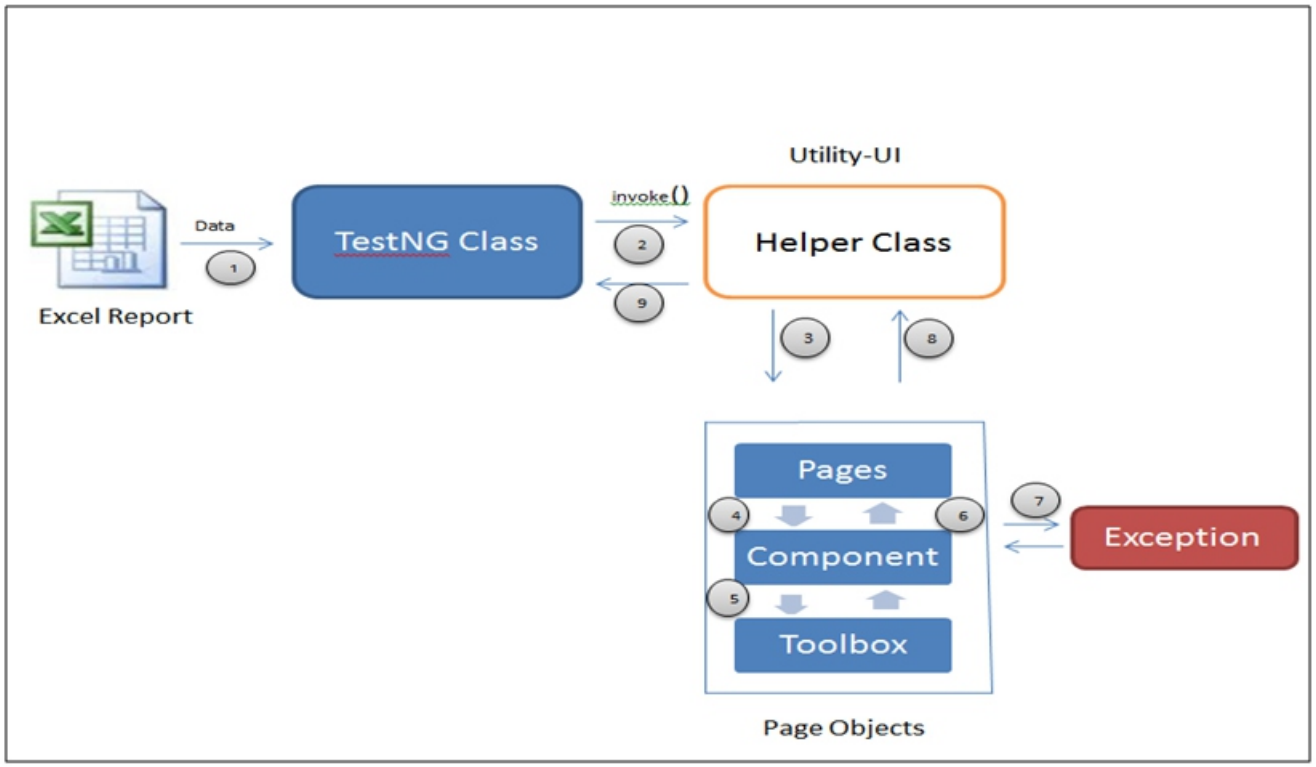
**A.Pages:** Webpage under test.

**B.Component:** Component contains the methods to access the UI elements (objects) of the respective component of the particular webpage using the locators Individual unique parts which integrate the webpage.

**C.Toolbox:** Handles the exceptions occurred while locating the page components.

**Common-UI:** Consists of Exception Classes.

## Flow of the diagram:




1.Data from the excel report is retrieved by the TestNG class.

2.TestNG class invokes Helper class, which consists of functionalities used for execution of requirements.

3.Helper Class invokes Pages present in 'Page Object' Component for retrieving the locators.

4.Pages invoke component class for retrieving the locators of individual elements for the respective page.

5. Exception occurred during the previous step, is handled by Toolbox.

6.Component of the respective page is returned to the Page.

7.If Exception, it is thrown by Exception Class else directly moves to Step 8.

8.Page returns the Page Object to the Helper Class.

9.Helper class can returns the respective Return Type.

## Conclusions:

The testing automation framework developed by Selenium Webdriver can share the test steps and test data among different testing, such as UI testing and the continuous integration tool used helps in keeping track of daily execution. The implementation of Browsermob Proxy helps in capturing the performance data for web apps (via the HAR format) to analyze the http request and response.It is convenient to switching in various types of testing for web applications. It supports multiple browsers and a variety of operating system. It can be widely used in web application test automation.

## Future work :

- Selenium Webdriver to be integrated with sikuli.
- Selenium Webdriver to perform testing on webservices.
- Enhancing the framework to make it more reusable.
- Extending Selenium Webdriver for mobile automation.

## References:

[1] Andreza M. F. V. de Castro, Gisele A. Macedo, Eliane F. Collins and Arilo C. Dias-Neto, Extension of Selenium RC Tool to Perform Automated Testing with Databases in Web Applications.AST 2013, San Francisco, CA, USA.

[2] Dianxiang Xu, Weifeng Xu, Bharath K bavikati, W.Eric Wong, "Mining Executable Specifications of Web Applications from Selenium IDE Tests" proc. IEEE 6th International Conference on Software Security and Reliability (SERE 12), IEEE Press, 2012, pp. 263-272, doi:10.1109/SERE.2012.39.

[3] Xinchun Wang and peijie Xu, "Build an Auto Testing framework Based on Selenium and FitNesse" proc. IEEE 2009 International Conference on Information Technology and Computer Science (ITCS 09), IEEE Press, 2009, pp. 436-439, doi:10.1109/ITCS.2009.228.

[4] Andreas bruns, Andreas Kornstadt, and Dennis Wichmann, "Web Application Tests with Selenium", Publised in IEEE Computer Society, 2009, pp. 88-91.

[5] Rosnisa Abdull Razak and Fairul Rizal Fahrurazi, "Agile Testing with Selenium" proc. 5th Malaysian Conference in Software Engineering(MySEC 11),IEEE Press , 2011, pp. 217-219.

[6] Xingen Wang, Bo Zhou and Wei Li,"Model based load testing of Web Applications", Proc. IEEE International Symposium on Parallel and Distributed Processing with Applications(ISPA 10), IEEE Press, 2010,pp. 483-490, doi:10.1109/ISPA.2010.24.

[7] Andreza M.F.V. de Castro, Gisele A.Macedo, Eliane F. Collins and Arilo C. Dias-neto, "Extension of Selenium RC Tool to Perform Automated Testing with Database in Web Applications",IEEE Press (AST 13), 2013, pp. 125-131.

[8] Ahmad Shahzad, Sajjad Raza,Muhammad N. Azam, khurram Bilal, Inam-ul-haqand Shafay Shamail ,"Automated Optimum test Case Genenration Using Web Navigation Graphs", Proc. IEEE International Conference on emerging Technology,IEEE2009,pp. 427-432.

[9] Wen-li Dong, hang YU,"Web Service Testing method based on Fault-coverage ",proc. 10th IEEE International Enterprise Distributed Object Computing Conference Workshops(EDOCW'06).

[10] Monika Sharma and Rigzin Angmo, "Web based Automation Testing and Tools",international journal of Computer Science And Information Technology (IJCSIT), Vol. 5(1),2014, ISSN:0975-9646, pp. 908-912.

[11]http://www.satisfice.com/blog/archives/118.

[12]http://whatis.techtarget.com/definition/framework.

[13]http://www.studyselenium.com/2014/01/difference-between-selenium-rc-and.html.

[14]http://seleniummaster.com/sitecontent/index.php/introduction-to-selenium-automation/selenium-automation-job-skills-menu/24-selenium-automation-job-interview-category/112-seleniumwebdriverandseleniumideinterviewquestions.

[15] http://docs.seleniumhq.org/.