IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 03, Sept 2012
ISSN (Online): 2231-5268
www.ijcsms.com

Implementation of Selenium with JUNIT and Test-Ng

Deepti Gaur[1], Dr. Rajender Singh Chhillar[2]

[1]M.tech Student, Department Of Computer Science and Application,
M.D University, Rohtak-124001, Haryana, India
deeptigaur50@gmail.com

[2]Professor, Department Of Computer Science and Application,
M.D University, Rohtak-124001, Haryana, India
Chhillar02@gmail.com

Abstract

Software testing is complex and time consuming. One way to reduce the effort associated with testing is to generate test data automatically. Testing is very important part of software development. Quality is not an absolute term; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. Software testing process can produce several artifacts. So, we proposed a model to improve quality and correctness and also we reduce the software testing time. In this paper we will implement selenium with different frameworks i.e. junit and testng.

Keyword:-*Selenium, SeleniumRC, Junit, TestNG.*

Introduction

"Software testing is technique of evaluating the attributes (i.e. correctness, completeness, security, consistency, unambiguousness, quality etc.) of software and determining that whether it meets its required functionality or not"**.** Purpose of testing is to find out Defects and causes and fixed them as early as possible. Testing simply refers to the validation and verification specially to build good quality software. There is some basic terms used in software testing that are as follows :-

Verification

Are we building the product right? It refers to the correctness of the function specifications.

Validation

Are we building the right product? It refers to the user expectation whether the product developed meets the user requirement or not.

Testing is an activity to find the bugs in software that may perform by tester or by applying strategies like white box or black box. So, the activities involved in the testing should be in planned way.

Black –box

This testing methodology looks at what are the available inputs for an application and what he expected outputs are that should result from each input.

White-box

This testing methodology looks under the covers and into the subsystem of an application. Whereas black-box testing concerns itself exclusively with the inputs and outputs of an application, white-box testing enables you to see what is happening inside the application.

Software testing is focused on finding defects in the final software before give it to the user. So it is the responsibility of the developer and the tester that he/she will examine all core functionality and the components associated with the software.

Selenium

Selenium IDE is the only flavor of Selenium which allows you to record user action on browser window. It can also record user actions in most of the popular languages like Java, C#, Perl, Ruby etc. This

eliminates the need of learning new vendor scripting language. For executing scripts created in these languages, you will need to use Selenium Remote Control. If you do not want to use Remote Control than you will need to create your test scripts in HTML format. Selenium can be accessed from tool--> Selenium IDE in your browser toolbar if the installation is completed successfully. As compared to most of the test automation tools it is very simple and lightweight. The small red button on the right hand side gives you an indication on whether Selenium is in recording mode or not. Also, Selenium IDE will not record any operation that you do on your computer apart from the events on Firefox browser window. So go ahead read your mail, open a word doc or do anything else, Selenium will record only your actions on browser. Other options present on the Selenium IDE toolbar are related to test execution. Run will execute the tests with the maximum possible speed, Walk will execute them with relatively slow speed and in step mode you will need to tell Selenium to take small steps. Final button present on the Selenium IDE toolbar is the Selenium Test Runner. Test Runner gives you nice browser interface to execute your tests and also gives summary of how many tests were executed, how many passed and failed. It also gives similar information on commands which were passed or failed. Test Runner is also available to tests developed in HTML Only. If you open the option window by going to Option, you will see there are some self explanatory options available. For example, encoding of test files, timeout etc. It allows us to:

- Record user actions when browsing in Firefox
- Replay recorded scripts
- Convert recorded scripts into programming languages such as Java, Ruby, and more
- Add verification and synchronization steps to the script during the recording process The IDE provides excellent support for writing automated test scripts in Selenium and gets better with every release.

Selenium RC:-

Selenium Remote Control is the server version of Selenium. You write your tests using a programming language and client library. Your tests issue commands which the client library sends to the server. The server then 'runs' your actions for you in the browser and reports the results back to your client. Using Selenium-RC allows you to write automated tests in any supported programming language. Tests written in this way allow you to use standard programming practices to make them easy to maintain, robust and easy to collaborate on as a team.

Selenium RC allows the test automation expert to use a programming language for maximum flexibility and extensibility in developing test logic. For example, if the application under test returns a result set and the automated test program needs to run tests on each element in the result set, the iteration / loop support of programming language's can be used to iterate through the result set, calling Selenium commands to run tests on each item.

Selenium RC provides an API and library for each of its supported languages. This ability to use Selenium RC with a high level programming language to develop test cases also allows the automated testing to be integrated with the project's automated build environment.

Automated Integration Testing with Selenium applications:

Automated integration tests can be useful particularly for the following types .

* Existing applications that haven't run any unit tests (e.g., legacy applications)
* CRUD applications that have a very simple middle tier and therefore don't have/require unit tests
* Applications that have business logic tightly coupled to the environment in which they run (e.g., business logic embedded in DAOs or servlets)

Options for implementing automated integration tests:

* Use a different framework for each tier
* Use tools, such as Watir or Watij

To create and run integration tests with Selenium, you must complete the following steps:

1. Use the Selenium IDE to record and play tests.
2. Export tests created with the Selenium IDE as JUnit tests.
3. Add the JUnit tests to your Java project in your IDE and run the tests.

Implementation and Result

JUNIT with Selenium

Import selenium-java-client-driver-0.9.2.jar and seleniumserver-coreless-1.0-0081010.060147.jar into your IDE (Eclipse, NetBeans, IntelliJ, etc.) project.

The destroy method will stop the Selenium server. Start the application server with the application that you are trying to test and then run the JUnit4 tests. The JUnit4 init and destroy methods will be called once only for each run and they will start the Selenium server. Lastly, the JUnit4 test cases will be

run. V. Implementation The implementation of the system is described in the following steps

- Take a URL
- Record the Web Application
- Test the Navigations and hyperlinks
- Obtain the Results

Test Cases

A good test case is one that has high probability of finding an undiscovered error. A successful test is one that uncovers an undiscovered error. Test Cases in Selenium are nothing but recording the Web Application and testing that again using the Selenium tool. The IDE allows many options for running your test case. You can run a test case all at once, stop and start it, run it one line at a time, run a single command you are currently developing, and you can do a batch run of an entire test suite. Execution of test case is very flexible in the IDE.

To Run a Test Case

- Click the Run button to run the currently displayed test case.
- Run a Test Suite
- Click the Run All button to run all the test cases in the currently loaded test suite.
- Stop and Start

  The Pause button can be used to stop the test case while it is running. The icon of this button then changes to indicate the Resume button. To continue click Resume.
- Stop in the Middle

  You can set a breakpoint in the test case to cause it to stop on a particular command. This is useful for debugging your test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.
- Start from the Middle

  You can tell the IDE to begin running from a specific command in the middle of the test case. This also is used for debugging. To set a start point, select a command, rightclick, and from the context menu select Set/Clear Start Point.
- Run Any Single Command

  Double-click any single command to run it by itself. This is useful when writing a single command. It lets you immediately test a command you are constructing, when you are not sure if it is correct. You can double-click it to see if it runs correctly

TestNG – Test Automation with Selenium

TestNG framework can be used for automation testing with Selenium (web application automation testing tool). .

```
package com.selftechy.testng;

import com.thoughtworks.selenium.*;
import org.testng.annotations.*;

public class TestNGDemo extends SeleneseTestBase{
        public Selenium selenium;

        @BeforeMethod
        public void setUp()throws Exception{
                selenium = new DefaultSelenium("localhost",4444,"*chrome","http://selftechy.com");
                selenium.start();
                selenium.windowMaximize();
        }

        @Test
        public void testNGDemo() throws Exception {
                selenium.open("/");
                selenium.click("link=TestNG (Next Generation Testing Framework) – Understanding Annotations");
                Thread.sleep(10000);

        verifyTrue(selenium.isTextPresent("Annotation:"));

        selenium.click("link=Selenium");
                Thread.sleep(10000);

        selenium.click("css=a[title=\"Introduction to Selenium\"]");
                Thread.sleep(10000);
        }

        @Test
        public void testTestAbout() throws Exception {
                selenium.open("/");
                selenium.click("link=About");

        selenium.waitForPageToLoad("30000");

        verifyTrue(selenium.isTextPresent("Selenium, QTP, Java"));
                selenium.click("link=Home");
```

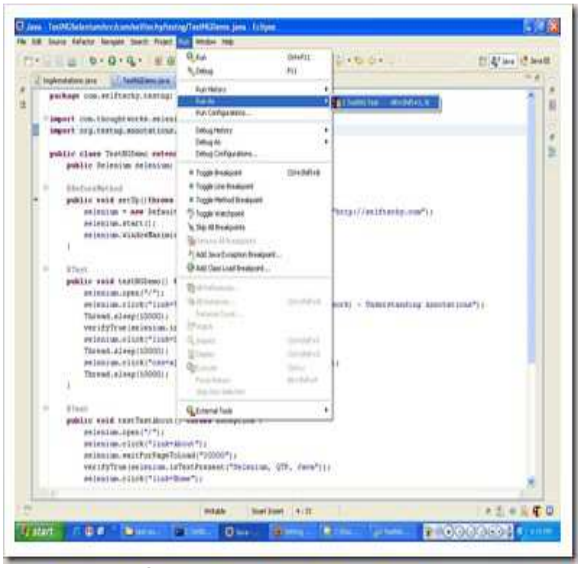
```
		selenium.waitForPageToLoad("30000");
			selenium.click("link=TestNG
– Next Generation Testing Framework");

		selenium.waitForPageToLoad("30000");

		verifyTrue(selenium.isTextPresent("Need
for a Testing Framework:"));
		}

		@AfterMethod
		public void tearDown(){
			selenium.stop();

		}

}
```

The Java class "TestNGDemo" is implemented in Eclipse IDE using TestNG framework.

In the above code, there are two test methods, which are marked with @Test annotation. There are two other methods, "setUp" and "tearDown" which are marked with @BeforeMethod and @AfterMethod annotations. Hence Before executing each test, setUp method will be executed. After the execution of each test, tearDown gets executed. Hence, Selenium gets instantiated and browser gets opened and closed twice during the execution.

How to execute the test?

Click Run –> Run As –> TestNG Test



Output of the execution should be as below

```
[TestNG] Running:
  C:\Documents and Settings\deepti\Local
Settings\Temp\testng-eclipse-1339593504\testng-
customsuite.xml

PASSED: testNGDemo
PASSED: testTestAbout

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================

===============================================
Default suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================

[TestNG] Time taken by
org.testng.reporters.EmailableReporter@1a679b7: 15
ms
[TestNG] Time taken by
org.testng.reporters.SuiteHTMLReporter@1e51060:
16 ms
[TestNG] Time taken by
org.testng.reporters.JUnitReportReporter@337d0f: 0
ms
[TestNG] Time taken by
org.testng.reporters.XMLReporter@e102dc: 0 ms
[TestNG] Time taken by [TestListenerAdapter]
Passed:0 Failed:0 Skipped:0]: 0 ms
```

Test execution also creates set of XML & HTML reports. TestNG creates "test-output" folder in the root folder, inside which we can see the reports. Below are the screenshots of some of the reports created by TestNG.

Report 1 created by TestNG

Report 2 created by TestNG

Conclusion

JUnit 4 and TestNG are both very popular unit test framework in Java. Both frameworks look very similar in functionality.

Feature comparison between JUnit 4 and TestNG.

Annotation Support

The annotation supports are implemented in both JUnit 4 and TestNG look similar.

Feature	JUnit 4	TestNG
test annotation	@Test	@Test
run before all tests in this suite have run	–	@BeforeSuite
run after all tests in this suite have run	–	@AfterSuite
run before the test	–	@BeforeTest
run after the test	–	@AfterTest
run before the first test method that belongs to any of these groups is invoked	–	@BeforeGroups
run after the last test method that belongs to any of these groups is invoked	–	@AfterGroups
run before the first test method in the current class is invoked	@BeforeClass	@BeforeClass
run after all the test methods in the current class have been run	@AfterClass	@AfterClass
run before each test method	@Before	@BeforeMethod
run after each test method	@After	@AfterMethod
ignore test	@ignore	@Test(enbale=false)
expected exception	@Test(expected = ArithmeticExcepti on.class)	@Test(expectedExceptions = ArithmeticException.class)
Timeout	@Test(timeout = 1000)	@Test (timeout = 1000)

References

[1] Glenford J. Myers "The Art of Software Testing, Second Edition" published in 2004. Antonia Bertoline" SOFTWARE TESTING" published in IEEE-Trial (Version 0.95) – May 2001.

[2] R. Boddu, G. Lan, G. Mukhopadhyay, B. Cukie, "RETNA" from requirements to testing in a natural way," 12th IEEE

[3] International Requirements Engineering Conference, pp.262-271, 2004.

[4] Beck, K. (2003). Test Driven Development – by Example Boston, Addison Wesley.

[5] E. van Veendaal (2008), Test Improvement Manifesto, in: Testing Experience, Issue 04/08, December 2008.

[6] X. Qu. M.B. Cohen and G. Rothermel. Configuration ware regression testing : an empirical study of

sampling and prioritization. In ISSTA '08. 2008.

[7] R. Lutz, I.C. Milkulski, "Requirements discovery during the testing of safety - critical software," Software Engineering 25th IEEE International Conference, pp. 578-583, 2003.

[8] Wagner, S. & Seifert, T. 2005, Software Quality Economics for Defect Detection Techniques Using Failure Prediction. In Processings of the 3rd Workshop on Software Quality (3-WoSQ). ACM Press.

[9] S. Berner, R. Weber, R.K. Keller, "Requirements & testing : Observations and lesson learned from automated testing," Proceedings of 27th international conference on software Engineering, pp. 571-579, 2005.

[10] Boehm, B., Huang, L., Jain, A. & Madachy, R. 2004. The ROI of Software Dependability : The iDAVE Model, IEEE Software, 21(3).

[11] K.R. Walcott, M.L. Soffa, Gregory M. Kafhammer and Robert S. Roos. Time-aware test suite prioritization. In Proceedings of the 2006 International Symposium on Software testing and Analysis, page 1-12, July 2006.

[12] Peter Sestoft "Systematic software testing" published in IT University of Copenhagen, Denmark Version 2, 2008-02-25.

[13] D. Jeffrey and N. Gupta. Test Care Prioritization Using Relevant Slices. In proceedings of Computer Software and Application COMPSA'06, Chicago, USA, Pages 411-420, 2006. K. Yukse, S. Dupont, D. Harnoir and C. Froidure, "FTTx automated test solution : Requirements and experimental implementation," IEE Magazine Electronics Letter, Vol. 41, No. 9, pp. 546-547, 2005.

[14] Williams, L., E.M. Maximilien, et. al. (2003). Test Driver Development as a Defect-Reduction Practice. IEEE International Symposium on Software Reliability Engineering, Denver, CO, IEEE Computer Society.

[15] M.B. Cohen, J. Snyder, and G. Rothermel. Testing across configurations : implications for combinatorial testing. SIGSOFT Softw. Eng. Notes, 31(6)1-9, 2006.

[16] Herzlich, P. 2005. The Need for Software Testing. Ovum Research : London UK.

[17] Gregory M. Kapfhammer "Software Testing" ACM 2008.

[18] John E., Bentley "Software Testing Fundamentals – Concepts, Roles and Terminology: 2005.

[19] Goutam Kumar Saha "Understanding Software Testing Concepts" ACM 2008.

[20] Mark Uttinga, Alexander Pretschnerb and Bruno Legeradc" A Taxonomy of Model-Based Testing" White paper in 200.

[21] Peter Miller "Testing? What testing?" 2000.

[22] SANTOSH KUAMR SWAIN, SUBHENDU KUMAR PANI, DURGA PRASAD MOHAPATRA "MODEL BASED OBJECT-ORIENTED SOFTWARE TESTING" Journal of Theoretical and Applied Information Technology in 2006.

[23] Cem Kaner, J.D. Ph.D." The Ongoing Revolution in Software Testing" Software Test & Performance Conference, December 8, 2004.

[24] Inspection vs. Testing 2003.

[25] Gerry Gaffney "Conducting a Walkthrough" 2002.

[26] "USER ACCEPTANCE TESTING (UAT) PROCESS" 2008.

[27] Andreas Leitner, Illinca Ciupa, Bertrand Meyer "Reconciling Manual and Automated Testing : the Auto Test Experience" 40th Hawaii International Conference on System Sciences – 2007.

[28] http://en.wikipedia.org/wiki/TestNG

[29] http://www.ibm.com/developerworks/java/library/j-testng/

[30] http://testng.org/doc/index.html

a. http://beust.com/weblog/

[31] http://en.wikipedia.org/wiki/JUnit

[32] http://www.ibm.com/developerworks/java/library/j-junit4.html

[33] http://junit.sourceforge.net/doc/faq/faq.htm

[34] http://www.devx.com/Java/Article/31983/0/page/

[35] http://ourcraft.wordpress.com/2008/08/27/writing-a-parameterized-junit-test/

[36] http://docs.codehaus.org/display/XPR/Migration+to+JUnit4+or+TestNG

[37] http://www.ibm.com/developerworks/java/library/j-cq08296/index.html

[38] http://www.cavdar.net/2008/07/21/junit-4-in-60-seconds/