Learning Report – Selenium – Web & Mobile Testing (Java)

L&T Technology Services

GLOBAL ENGINEERING ACADEMY

Genesis

Document History

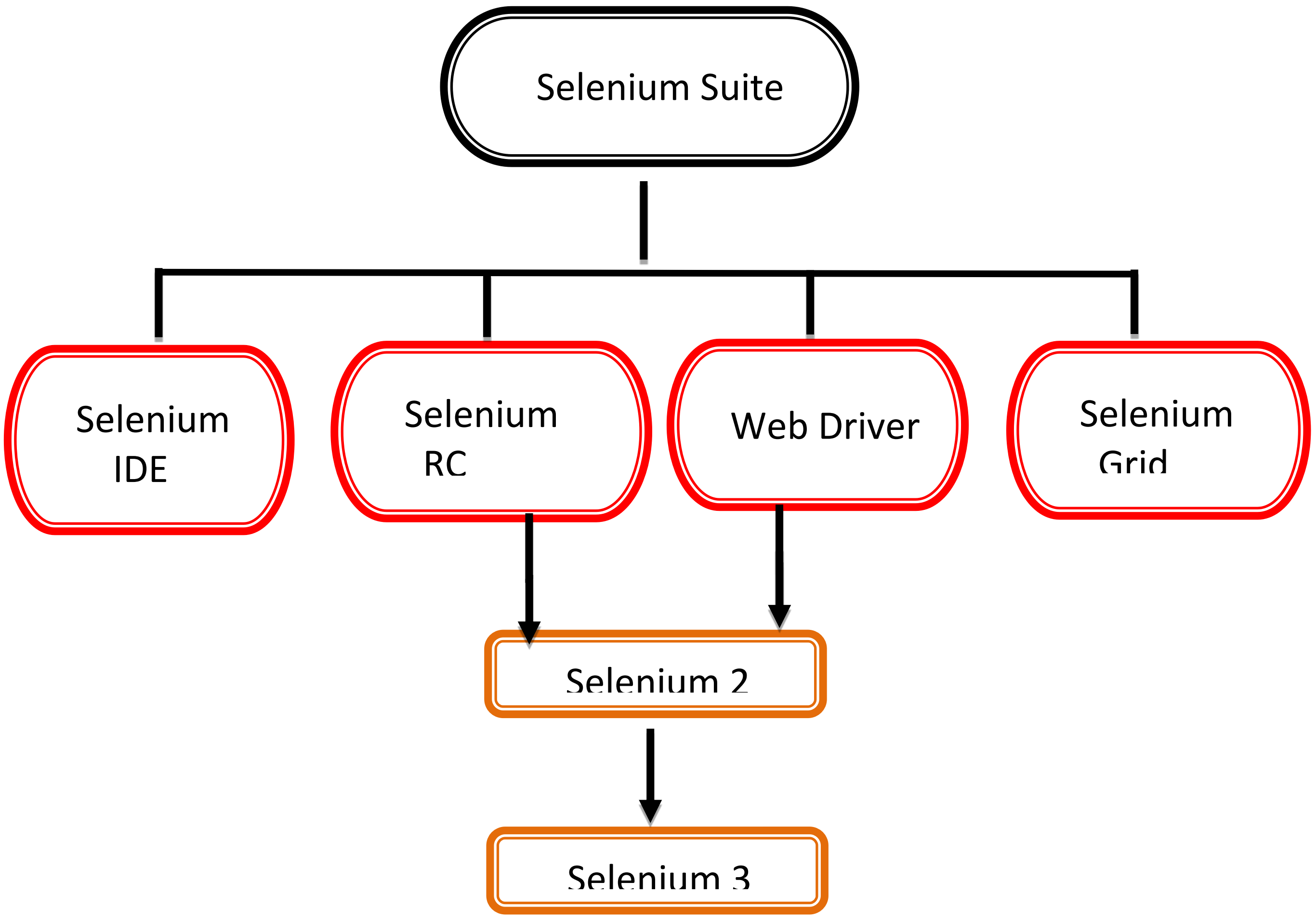
Ver.Rel. No.	Release Date	Prepared. By	Reviewed By	To be approved By	Remarks/Revision Details
1		Sneha Anand			
2					
3					

Contents

SELENIUM

Selenium is a portable framework for testing web applications. It is a popular open source web-based automation tool used to validate web applications across different browsers and platforms. We can use multiple programming languages like Java, C#, Python etc. to create Selenium Test Scripts. The tests can then be run against most modern web browsers. Selenium runs on Windows, Linux and macOS. Selenium was released under the Apache License 2.0.
Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization. The complete list of tools is as followed:

- Selenium Integration Development Environment (IDE)
- Selenium Remote Control (RC)
- Web Driver
- Selenium Grid



1.1 Selenium Integration Development Environment (IDE)

Selenium IDE is a complete integrated development environment (IDE) for Selenium tests. It is implemented as a Firefox Add-On and as a Chrome Extension. It allows for recording, editing and debugging of functional tests.

Scripts may be automatically recorded and edited manually providing auto completion support and the ability to move commands around quickly. Scripts are recorded in Selenese, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser.

Why choose Selenium IDE?

- To create tests with little or no prior knowledge in programming.
- To create simple test cases and test suites that you can export later to RC or WebDriver.
- To test a web application against Firefox and Chrome only.

1.2 Selenium Remote Control (RC)

Selenium Remote Control (RC) is a server, written in Java that accepts commands for the browser via HTTP. With RC it is 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.

Selenium Remote Control was a refactoring of Driven Selenium or Selenium B designed by Paul Hammant. Selenium RC served as the flagship testing framework of the entire project of selenium for a long-standing time.

Why choose Selenium RC?

- To design a test using a more expressive language than Selenese.
- To run our test against different browsers or different OS.
- To deploy our tests across multiple environments using Selenium Grid.
- To test our application against a new browser that supports JavaScript

1.3 Selenium WebDriver

Selenium WebDriver is the successor to Selenium RC. Selenium WebDriver accepts commands and sends them to a browser. This operation is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results. Selenium WebDriver does not need a special server to execute tests. The WebDriver directly starts a browser instance and controls it. However, Selenium Grid can be used with WebDriver to execute tests on remote systems. WebDriver uses native OS level functionality rather than browser-based JavaScript commands to dive the browser thus bypassing problems with subtle difference between native and JavaScript commands including security restrictions.

Why choose Selenium WebDriver?

- To use a certain programming language in designing our test case.
- To test applications that are rich in AJAX-based functionalities.
- To create customized test results.

1.4 Selenium Grid

Selenium Grid is a server that allows tests to use web browser instance running on remote machines. Here one server acts as the central hub. For browser instances access tests contact the hub. The hub has a list of servers that provide access to browser instance and lets tests use these instances. Selenium Grid allows running tests in parallel on multiple machines and to manage different browser version and browser configuration centrally.

Why choose Selenium Grid?

- To run our Selenium RC scripts in multiple browsers and OS simultaneously.
- To run a huge test suite, that needs to complete in the soonest time possible.

SELENIUM WEBDRIVER

Selenium WebDriver is an open source collection of APIs which is used for testing web applications. The tool is used for automating web application testing to verify that it works as expected or not. It is faster than selenium RC because of its simple architecture.

Language supported by WebDriver:

- Java
- .NET
- PHP
- Python
- Perl
- Ruby
- C#

Selenium WebDriver supports browsers like Firefox, Chrome, Safari and Internet Explorer. It also permits us to execute cross-browser testing.
Using Element Locators and WebDriver methods we can create test cases.

2.1 WebDriver Installation

The steps to get going with WebDriver is as follows:

- Install JDK
- Setup Environment Variable for JDK
- Install Eclipse JDE
- Download Selenium Java Client Driver and extract the zip file.
- Adding Selenium jar files to the project:
  - In eclipse right click on the created project and select
    Build path -> Libraries Tab -> Add External jars
- Add all the jar files from extracted file.

2.2 WebDriver: Element Locators

Element locators are used to identify web elements. These are standard for various browsers.

Locators in WebDriver are:

- id
- name
- className
- tagName
- linkText
- partialLinkText
- cssSelector
- xpath

Element locator syntax:

```
<object>.findElement(By.<locator>(<"value">))
```

Various Element locators are as followed:

- id:

example:

```
driver.findElement(By.id("userid"));
```

- name:

example:

```
driver.findElement(By.name("username"));
```

- className:

example:

```
driver.findElement(By.className("className"));
```

- tagName:

example:

```
driver.findElement(By.tagName("input"));
```

- linkText:

example:

```
5.   driver.findElement(By.linkText("Text box"));
```

- partialLinkText:

example:

```
6.   driver.findElement(By.partialLinkText("text"));
```

- cssSelector:

example:

```
7.   driver.findElement(By.cssSelector("value"));
```

- xpath:

example:

```
8.   driver.findElement(By.xpath("//input[@id='loginId']"));
```

2.3 WebDriver: Web Elements

Following are the web elements available for use:

Edit box	Button	Link	Radio button
Checkbox	Drop Down box	List box	Combo box
Image, Image link	Frames	Web Table	HTML Table

WebDriver Methods

Handling Edit Boxes

- *sendKeys()*
- *isDisplayed()*
- *isEnabled()*
- *getAttribute()*
- *clear()*

Handling Text Area

- *getText () method is used to capture text from a specific location/area from the current browser page.*

Handling Links

- *click()*
- *isDisplayed()*
- *isEnabled()*
- *getAttribute()*
- *getText()*

Handling Radio Buttons

- *click()*
- *isDisplayed()*
- *isEnabled()*
- *isSelected()*
- *getAttribute()*

Handling Checkboxes

- *click()*
- *isDisplayed()*
- *isEnabled()*
- *isSelected()*
- *getAttribute()*

Handling Alerts or pop up window

- *accept()*
- *dismiss()*
- *getText()*

Wait methods

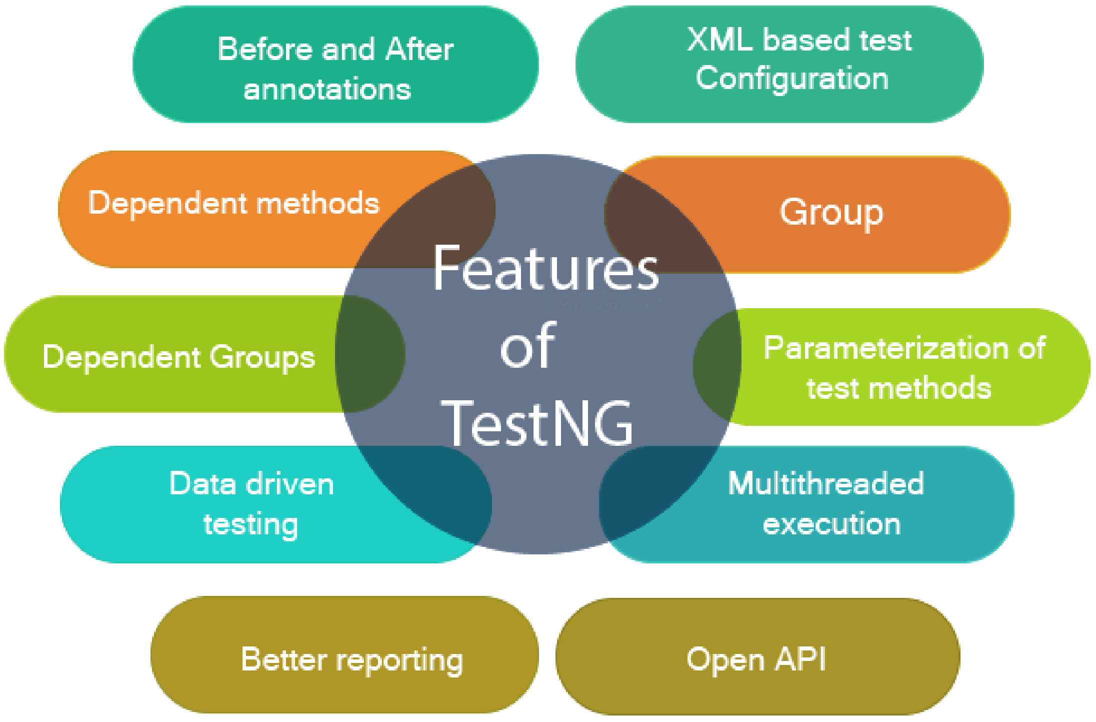
- Thread.sleep(): It waits for specific number of ms.
- implicitWait()
- Explicit Wait:

Handling Dropdown box
Handling Frames

TestNG

TestNG is an automation testing framework in which NG stands for ‘Next Generation’. The framework is inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easy to use.

FEATURES OF TestNG:



TestNG has following features:

- Supports annotations
- TestNG uses more Java and OO features
- Supports testing integrated classes
- Separates compile-time test code from run-time configuration/data info.
- Flexible runtime configuration
- Introduces ‘test groups’
- Supports dependent test methods, parallel testing, load testing and partial failure
- Flexible plug-in API
- Support for multi-threaded testing

Annotations

Annotations in TestNG are lines of code that can control how the method below them will be executed.

They are always preceded by the **@** symbol.

Annotation	Description
@BeforeSuite	The annotated method runs only once before all tests in the suite have run.
@AfterSuite	The annotated method runs only once after all tests in the suite have run.
@BeforeClass	The annotated method run only once before the first test method in the current class is invoked.
@AfterClass	The annotated method run only once after all the test methods in the current class have run.
@BeforeTest	The annotated method will run before any test method belonging to the classes inside the <test> tag is run.
@AfterTest	The annotated methods will run after all the tests methods belonging to the classes inside the <test> tag is run.
@BeforeGroups	The method runs shortly before the first test method is invoked.
@AfterGroups	The method runs shortly after the last test method is invoked.
@BeforeMethod	The annotated method will be run before each test method.
@AfterMethod	The annotated method will be run after each test method.

Benefits of using Annotations:

- TestNG identifies the methods it is interested in through annotation hence method names are not restricted to any pattern or format.
- We can pass additional parameters to annotations.

Attributes

- **Priority:** It is used to execute the test cases in user defined order.

```
@Test(priority=3)
```

- **Description:** It is used to specify the description of the test case.

```
@Test(description=”login Test Case”)
```

- **Enabled:** It is used to execute the specific test case.

```
@Test(enable=false)
```

- **dependsOnMethod:** It is used to execute the test cases in user defined order.

```
@Test(dependsOnMethods={“method name”})
```

- **alwaysRun:** It will execute the test cases if the method mentioned in depends on method failed.

```
@Test(dependsOnMethods={“method name”}, alwaysRun=true)
```

Projects

Automating ICICI loan eligibility form page

Objective of project: To automate ICICI loan eligibility page by using Selenium webdriver wherein the script below automates the process of filling the input parameters.

```
//ICICI BANK PAGE AUTOMATION
    public class task5 {
                public static void main(String[] args) throws Exception {

                            System.out.println("welcome to selenium");

    System.setProperty("webdriver.chrome.driver","D:\\selenium\\extracts\\chromedriver\\chromedriver.exe");

                            WebDriver driver=new ChromeDriver();

                            Actions act = new Actions(driver);

                            //Select slt = new Select();
                            //Page open
                            driver.get("https://loan.icicibank.com/asset-portal-all/check-
eligibility?loanType=al&WT.mc_id=Desk_OAPN_cms_cl_index_btn_applynow");

                            //Type of loan
                            driver.findElement(By.id("carRelbtn")).click();
                            act.moveToElement(driver.findElement(By.linkText("New Car Loan"))).click().perform();

                            //Mobile number
                            driver.findElement(By.id("MOBILENUM")).sendKeys("987546215");

                            //Name
                            driver.findElement(By.id("CUSTFIRSTNAME")).sendKeys("SNEHA");
                            driver.findElement(By.id("CUSTLASTNAME")).sendKeys("ANAND");
                            Thread.sleep(500);

                            //Location
                            WebElement city = driver.findElement(By.id("CUSTCOMMUCITY"));
                            city.sendKeys("MYS");
                            Thread.sleep(500);
                            city.sendKeys(Keys.ARROW_DOWN);
                            Thread.sleep(500);
                            city.sendKeys(Keys.ENTER);
                            Thread.sleep(500);


                            //Residence type
                            driver.findElement(By.id("residancebtn")).click();
                            act.moveToElement(driver.findElement(By.linkText("Owned"))).click().perform();
```

```

		//Current residence move
		driver.findElement(By.id("yrbtn")).click();
		act.moveToElement(driver.findElement(By.linkText("2020"))).click().perform();
		driver.findElement(By.id("mnthbtn")).click();
		act.moveToElement(driver.findElement(By.linkText("Dec"))).click().perform();

		//Birthday selection
		driver.findElement(By.id("DATEOFBIRTH")).sendKeys("05121998");

		//Specified dealer product
	act.moveToElement(driver.findElement(By.xpath("//*[@id=\"DLR\"]/div/div[2]/label/span"))).click().perform();

		//Car cost calculation
		WebElement carmod = driver.findElement(By.id("CAR_MODEL_NAME"));
		carmod.sendKeys("HYU");
		Thread.sleep(500);
		carmod.sendKeys(Keys.ARROW_DOWN);
		Thread.sleep(500);
		carmod.sendKeys(Keys.ENTER);
		Thread.sleep(500);

		WebElement citymod = driver.findElement(By.id("CITY_NAME"));
		citymod.sendKeys("CHENN");
		Thread.sleep(500);
		citymod.sendKeys(Keys.ARROW_DOWN);
		Thread.sleep(500);
		citymod.sendKeys(Keys.ENTER);
		Thread.sleep(500);

		//Work Information
		driver.findElement(By.id("selSalbtn")).click();
		act.moveToElement(driver.findElement(By.linkText("Salaried"))).click().perform();



		WebElement company = driver.findElement(By.id("EMPLOYERNAME"));
		company.sendKeys("LARSEN AND");
		Thread.sleep(500);
		company.sendKeys(Keys.ARROW_DOWN);
		Thread.sleep(500);
		company.sendKeys(Keys.ENTER);



		driver.findElement(By.xpath("//*[@id='SelfEB']/*[@id='JDCE']/div/*[@class='col-xs-6']/div/*[@id='yrbtn']")).click();
		act.moveToElement(driver.findElement(By.linkText("2021"))).click().perform();

```

```
                    driver.findElement(By.xpath("//*[@id='SelfEB']/*[@id='JDCE']/div/div[@id='month-div_we']/div/button[@id='mnthbtn']")).click();
                    act.moveToElement(driver.findElement(By.linkText("Feb"))).click().perform();

                    driver.findElement(By.xpath("//*[@id='TWE']/div/div/div/button[@id='twbtn']")).click();
                    act.moveToElement(driver.findElement(By.linkText("2"))).click().perform();


                    driver.findElement(By.id("MONTHLY_NET_HM_SAL")).sendKeys("48000");

                    driver.findElement(By.id("GROSS_FIXED_M_SAL")).sendKeys("50000");

                    //Others
                    driver.findElement(By.id("TOTAL_CURR_EMI_PAID")).sendKeys("18000");

          }
  }
```

Automating the login process with webdriver and managing the workflow through TestNG

Objective of project: Write a script to login into “Orangehrm Demo” webpage in Mozilla Firefox and then log out of the webpage. The workflow is handled by TestNG.

```java
public class task2 {
		WebDriver driver;

		@Test(priority = 1,description = "launching page", alwaysRun = true)
		public void launch() throws Exception {
			System.out.println("Welcome to Selenium launch");
			System.setProperty("webdriver.gecko.driver", "D:\\selenium\\extracts\\firefoxdriver\\geckodriver.exe");
			driver = new FirefoxDriver();
			driver.get("https://opensource-demo.orangehrmlive.com/");
			Thread.sleep(200);
		}

		@Test(priority = 2,description = "performing login", enabled = true, dependsOnMethods = "launch")
		public void login() {
			driver.findElement(By.xpath("//*[@id=\"txtUsername\"]")).sendKeys("Admin");
			driver.findElement(By.xpath("//*[@id=\"txtPassword\"]")).sendKeys("admin123");
			driver.findElement(By.xpath("//*[@id=\"btnLogin\"]")).click();
		}

		@Test(priority = 3,description = "logging out", enabled = true, dependsOnMethods = "login")
		public void logout() throws Exception {
//			driver.findElement(By.xpath("//*[@id=\"welcome\"]")).click();
//			Thread.sleep(100);
			driver.findElement(By.linkText("Welcome")).click();
			Thread.sleep(200);
			driver.findElement(By.linkText("Logout")).click();
		}
}
```