**GURUKULA WEB APPLICATION AUTOMATION**

Contents

[Introduction 2](#_Toc451096304)

[Reason for Selenium/TestNG with Java 2](#_Toc451096305)

[Framework used for Automation 2](#_Toc451096306)

[Software/Tools used 2](#_Toc451096307)

[Setting up the Environment 2](#_Toc451096308)

[Preparing the inputs for the execution 3](#_Toc451096309)

[Implementation of Gurukula Automation Project 3](#_Toc451096310)

[Automation Test cases 3](#_Toc451096311)

[Utils 4](#_Toc451096312)

[WebObjects 4](#_Toc451096313)

[WebPage Classes 4](#_Toc451096314)

[WebPageUrl 4](#_Toc451096315)

[Execution of Automation Suite 4](#_Toc451096316)

[Output Result File 5](#_Toc451096317)

# Introduction

Gurukula Web Application is mainly used to create and maintain Staff records along with their bank branch details. As part of it, a new user should be able to register a new account, login to the application, create/edit/delete branches, create/edit/delete staff associated to a branch.

In order to automate the above mentioned functionality, here i have used Selenium/TestNG with Java1.8 to complete it.

# Reason for Selenium/TestNG with Java

Selenium Web driver/TestNG and Java are two open source technologies which doesn't require any licensed products to implement them. Moreover, they are OS independent which makes flexible enough to run on any Windows/Linux machine.

# Framework used for Automation

In order to automate the current Gurukula application, i have chosen to go with "Page object Model" framework. Basically Page Object Model is writing functionalities and reusable components of a page inside a single class related to it. For instance, Gurukula application has a login page which has specifically functional objects as:

1. Login with credentials
2. Automatic login
3. Register a new user
4. Accounts menu, which in turn has:
   1. Register,
   2. Authenticate

While all the above Web Objects are identified and saved under a repository class "WebObjects", they are used in the class "LoginPage" where their actual functionality is defined.

# Software/Tools used

1. OS - Windows 2012
2. Java - JDK1.8 (64-bit)
3. IDE – Eclipse
4. Browser - Mozilla Firefox 46.0.1
5. Object Identifier - Firebug 2.0.16
6. Selenium version - selenium-java-2.53.0, selenium-java-2.53.0-srcs, selenium-server-standalone-2.53.0
7. TestNG version - TestNG6.9.9

# Setting up the Environment

1. Copy/Download the source code from GitHub to local machine.
2. Extract the Project folder on to your target location
3. In case, you are using any other version of Selenium version or TestNG version, go to..\GurukulaPageFactory\lib.
   1. Replace all the existing jar files with your jar files of Selenium/TestNG
4. Open a DOS command prompt and go to the..\GurukulaPageFactory.
5. Set the ProjectPath variable. For Ex: if you have copied the GurukulaPageFactory under c:\ folder, then copy the path of GurukulaPageFactory to Command Prompt and type "set ProjectPath=c:\GurukulaPageFactory"
6. To cross check the path, type echo %ProjectPath%"
7. Then set the classpath as "set classpath=%ProjectPath%\bin;%ProjectPath%\lib\\*"
8. Now we need to add the selenium jar files under eclipse.
9. Open Eclipse and import the Project in it.
10. Right click on the Project-->Build Path-->Configure Build Path-->Libraries.
11. Click on "Add External JARs". Go to "%ProjectPath%\lib\\*" and select all three selenium jar and testng6.9.9 jar files.
12. Click on OK to apply them.

# Preparing the inputs for the execution

1. Go to ..\GurukulaPageFactory and find the TestNG.xml file in Notepad++ or any editable tool.
2. You will find the below list of Parameters:

<parameter name="loginname" value="admin"/>

<parameter name="loginpassword" value="admin"/>

<parameter name="NewBranchName" value="\*\*"/>

<parameter name="NewBranchCode" value="\*\*" />

<parameter name="DelBranchName" value="\*\*"/>

<parameter name="DelBranchCode" value="\*\*" />

<parameter name="NewStaff" value="\*\*" />

NOTE: When you run the script for the first time, i suggest to keep the values of "loginname" and "loginpassword" as same so as to login properly into the application.

1. For rest of the parameters, user can define their own inputs.

# Implementation of Gurukula Automation Project

Gurukula Automation Project is broken down in to multiple Packages depending on their significance:

1. com.gurukula.TestCases - Contains Automation Testcases classes
2. com.gurukula.Utils - Contains common utility functions which can be used globally as part of automation
3. com.gurukula.WebObjects - Web Objects identities are present
4. com.gurukula.WebPages - Web Page classes are present
5. com.gurukula.WebPageUrl - Web Page Urls are maintained are used in Automation to check if the page navigated is correct or not.

# Automation Test cases

1. Each Automation test case consists of following:
   1. Required Objects/variables declaration
   2. @BeforeTest function - All steps that need to be performed before the actual test
   3. @Test function - Where actual test functions are called in the Automation Test cases
   4. @AfterTest - Steps that need to be performed after test (ex: closing/quiting the driver, which in turn will close the browser launched for the test)

# Utils

It is intended to have frequently used functions across the automation. Here i have created couple of functions:

1. WaitUntilLoad - This waits until a particular WebElement(or a WebObject is displayed) identified by the driver on the Web Page.
2. SleepTime - Wait till a particular time period(here input is taken in seconds)

(This class is mainly intended for creating many more common functions, like selecting object under a drop down menu or performing a search item)

# WebObjects

All Web Objects identification are stored in this Package/Class. This acts as a repository for Web Page class to get the required WebElement identity.

# WebPage Classes

Each WebPage Class, by default contains below list:

1. List of Web Objects used in the class
2. Constructor with parameter as driver which will be passed from Automation Test case to its Page class (in current automation, I have used only firefox driver. But there is scope for Chromedriver and IEdriver as well)
3. Functions for the potential Web Objects which drives the functionality of Web Application.

# WebPageUrl

This Package/Class consists of some static Url strings that are used as part of automation. It is used in Assert functions whenever a link click or menu item object is selected which navigates to another page.

# Execution of Automation Suite

1. Execution of Automated Test cases are listed under the TestNG.xml file (which is mentioned under the section: Preparing the inputs for the execution). Here are the list of test cases currently present under the TestNG.xml file:

<classes>

<class name="com.gurukula.TestCases.TC\_WelcomePage" />

<class name="com.gurukula.TestCases.TC\_BranchOperations" />

<class name="com.gurukula.TestCases.TC\_StaffOperations" />

<class name="com.gurukula.TestCases.TC\_DeleteBranch" />

<class name="com.gurukula.TestCases.TC\_ValidateAutomaticLogin" />

</classes>

1. Whenever a new test case is written, it can be appended under the <Classes> (similar to the one mentioned above). All test cases will be executed sequentially in the same order mentioned under <Classes>
2. In order to execute the Test suite, run the below command from Command prompt:
   1. Traverse to %ProjectPath% where TestNG.xml is present.
   2. Type the command "java org.testng.TestNG %ProjectPath%\TestNG.xml"

# Output Result File

You can find the output result file under %ProjectPath%\test-output\emailable-report.html file