Contents

Selenium 12

JAVA BASICS 12

OBJECT ORIENTED PROGRAMMING 12

CLASS 14

OBJECTS 14

CREATING A CLASS 15

RUNNING A CLASS 16

HelloSelenium.java class 16

ACCESS MODIFIERS 18

o Default 18

o Private 18

o Public 18

o Protected 18

VARIABLES 19

PRIMITIVE VARIABLE 19

COMPLEX VARIABLE 20

USING THE VARIABLES 20

Local Variables 20

Class Variables 20

Base class have the main function to call its subclass 21

Base class is the same as Super class or Parent class 21

Subclass is the same as Child class 21

Activity 1 Variables 21

Class Rectangle.java base class 21

Instantiation.java Rectangle.java subclass 22

Activity 2 Data Types 23

Activity2\_dataTypes.java 23

dataTypes.java base class 23

Activity2\_dataTypes.java subclass 24

JAVAOPERATORS 25

1. Mathematical 25

2. Equality or Relational 25

3. Assignment 25

4. Conditional 25

EQUALITY OPERATORS 26

CONDITIONAL OPERATORS 27

Activity 3 27

Operators.java base class 27

Operators.java subclass 28

JAVA CONTROL STATEMENT 29

1. IF STATEMENT 29

2. ELSE-IF STATEMENT 30

3. SWITCH STATEMENT 31

4. WHILE LOOP 32

Variable Trace Table 33

5. FOR LOOP 33

Variable Trace Table 34

6. DO WHILE LOOP 34

Variable Trace Table 35

ControlFlow.java base class 35

ControlFlow.java subclass 36

Calculator.java base class 37

Calculator.java subclass 39

BASIC JAVACONCEPTS 39

1. ENCAPSULATION 40

ENCAPSULATION: SAMPLE 40

Encapsulation.java base class 40

EncapsulationMain.java subclass 41

2. INHERITANCE 42

INHERITANCE: SAMPLE 43

Inheritance.java base class 43

Inheritance.java subclass 44

The “extends” keyword is used to inherit a class 44

INHERITANCE: DISCUSSION 45

3. POLYMORPHISM 46

• Method Overloading 46

• Method Overriding 46

METHOD OVERLOADING& OVERRIDING 47

1. Overloading 47

2. Overriding 47

METHOD OVERLOADING: SAMPLE 47

PolymorphismMethodOverloading.java base class 47

PolymorphismMethodOverloadingChild.java subcass 48

METHOD OVERRIDING: SAMPLE 49

MethodOverriding.java base class 49

MethodOverridingSubclass.java sub class 49

MethodOverridingTestMethod.java subclass 50

4. ABSTRACTION 50

Abstraction.java base class 51

AbstractionChild.java base class 51

SUMMARY 52

LOCATING WEB ELEMENTS AND BASIC SELENIUM WEBDRIVER COMMANDS 53

LEARNING OBJECTIVES 53

FINDING HTML ELEMENTS 53

LOCATORS 58

• ID 59

WebElementelement = driver.findElement(By.id("q")); 59

• Name 59

WebElementelement = driver.findElement(By.name("q")); 59

• Class Name 59

WebElementelement = driver.findElement(By.className("downloadBox")); 59

• Tag Name 60

WebElementelement = driver.findElement(By.tagName(“table")); 60

• CSS 60

WebElementelement =driver.findElement(By.cssSelector("#q")); 60

• Link Text/ Partial Link Text 60

WebElementelement = driver.findElement(By.linkText(“Download Selenium")); 60

•Xpath 60

WebElementelement = driver.findElement(By.xpath("//input[@id='q']")); 61

Or "//input[@name="userName"]" 61

WILD CARD \* CHARACTER 61

BASIC SELENESE COMMANDS 61

1. Typing a Value 61

WebElementelement = driver.findElement(By.name("q")); 62

element.sendKeys("Cheese!"); 62

2. Clicking an Element 62

WebElementelement = driver.findElement(By.xpath(“smpleElem")); 62

element.click(); 62

3. Selecting a value from Dropdown 62

Select select= new Select(driver.findElement(By.tagName("select"))); 62

select.selectByVisibleText("Edam"); 62

Select select= new Select(driver.findElement(By.tagName("select"))); 62

select.selectByIndex(1); 62

4. Switching Windows 62

driver.switchTo().window("windowName"); 62

Or if the window name is not known: 63

for (String handle : driver.getWindowHandles()) { 63

driver.switchTo().window(handle); 63

} 63

5. Switching to Frames 63

driver.switchTo().frame("frameName"); 63

6. Handling Popup Alert 63

Alert alert= driver.switchTo().alert(); 63

alert.accept(); 63

7. Navigation 63

driver.navigate().forward(); 63

driver.navigate().back(); 63

Closing a Browser 63

driver.quit(); 63

Wait command 63

1. Implicit Wait 64

WebDriver driver = new FirefoxDriver(); 64

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); 64

driver.get("http:somedomain/url\_that\_delays\_loading"); 64

WebElementmyDynamicElement= driver.findElement(By.id("myDynamicElement")); 64

2. Explicit Wait 64

WebDriver driver = new FirefoxDriver(); 64

driver.get("http:somedomain/url\_that\_delays\_loading"); 64

WebElementmyDynamicElement= (new WebDriverWait(driver, 10)) .until(ExpectedConditions.presenceOfElementLocated(By.id("myDynamicElement"))); 64

Selenium project steps: 64

FirstSeleniumScript.java 65

M4.java Selenium test script 67

WHAT IS TESTNG? 72

Annotations: 72

• @BEFOREMETHOD 72

• @AFTERMETHOD 72

• @TEST 72

• @BEFORETEST 72

• @AFTERTEST 72

INSTALLING TESTNG IN ECLIPSE 72

CREATING TESTNG TEST 72

FirstTestNGTest.java 73

M3\_Activity1.java 73

M5.java TestNG 76

WHAT IS A FRAMEWORK 82

1. Linear Automation Framework 82

2. Module Based Testing Framework 83

3. Library Architecture Testing Framework 83

4. Data Driven Testing Framework 84

5. Keyword Driven Testing Framework 85

6. Behavior Driven Development Framework 86

7. Hybrid Testing Framework 87

SUMMARY 89

•Advantages 90

Types of Automation Frameworks 90

SETTING UP MAVEN 92

ADDING DEPENDENCIES IN POM 93

PAGEOBJECTS 96

CREATING PAGEOBJECTS 96

CREATING PAGEOBJECTS 98

BasePage.java common package 98

GooglePage.java pageobject 100

BaseTest.java common package 102

CREATING THE TEST 103

GoogleTest.java test package 104

USING A FRAMEWORK 108

FRAMEWORK STRUCTURE 108

FRAMEWORK STRUCTURE 109

1. Core package 109

2. Data Objects package 110

3. Web Elements package 111

4. Page Objects package 112

5. Tests package 113

SETTING UP MAVEN 114

How to setup projects to Maven: 115

How to setup existing Maven projects 119

1. CREATING PAGE OBJECT 119

2. CREATING ACTION IN PAGE OBJECT 120

MainPage.java 120

3. MANAGING PAGE OBJECT ACTION 121

SignInPage.java 122

4. CREATING THE TEST DATA OBJECT 124

5. CREATING THE TEST DATA OBJECT 124

TestData.java 127

6. CREATING TEST SCRIPT 128

7. RUNNING THE TEST 130

SUMMARY 136

Testscript SignInTest.java in src/test/java 137

MainPage.java pageobject 138

SignInPage.java pageobject 139

TestData.java dataobject 140

Module 7 Activity 142

CreateNewAccountTest.java 142

HomePage.java pageobject 144

RegisterPage.java pageobject 145

SignInPage.java pageobject 148

TestData.java dataobject 150

SELENIUM WEBDRIVER TEST DEBUGGING 154

LEARNINGOBJECTIVES 154

DEBUGGING 154

COMMON DEBUGG INGISSUES 154

SUMMARY 158

SELENIUM WEBDRIVER BEST PRACTICES 158

LEARNINGOBJECTIVES 158

BENEFITS OF BESTPRACTICE 158

BEST PRACTICES FOR SELENIUM 159

SUMMARY 160

COURSE SUMMARY 160

SCHOOL RECAP 161

THE END OF THE ROAD 161

Final Exam Hybrid Test Framework 162

TricentisHome.java pageobject 162

AutomobileInsurancePage.java pageobject 163

TestDataExam.java dataobject 173

SuccesfulPurchaseOfItems.java tests script 178

Selenium

FEATURES AND BENEFITS

• Selenium can be integrated with different third party tools to make it more

customizable and more efficient depending on the user’s requirement ..(e g MS

Excel, Jenkins, Jira, Maven)

• It enables thorough validation of applications through a full complement of

Actions, Accessors and Assertions

• It ensures the smooth implementation of application testing and improve the

business processes before production deployment

• It rapidly isolates and diagnose issues/defects encountered with reports

JAVA BASICS

LEARNINGOBJECTIVES

At the end of this module, you will be able to:

–Define Java

–Create Java class

–Identify different data types

–Learn Object Oriented Programming

OBJECT ORIENTED PROGRAMMING

CLASS

OBJECTS

OBJECTS: EXAMPLES

CREATING A CLASS

Step 1: Open Eclipse application.

Step 2: In the Package Explorer tab find the “src” folder under the Java project. Right click on the “src” folder > New > Click Class.

Step 3: Enter “samplepackage” in the Package (other name could be used)

Step 4: Enter “HelloWorld” as class name (other name could be used).

Step 5: Click the checkbox for “public static void main(String[] args)”. Then click Finish button.

Step 6: The created class will be added in the Java project and will be automatically opened.

RUNNING A CLASS

HelloSelenium.java class

public class HelloSelenium

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

System.out.println("Hello Selenium World!");

}

}

Output:

ACCESS MODIFIERS

o Default

o Private

o Public

o Protected

VARIABLES

PRIMITIVE VARIABLE

You must declare all variables before they can be used. It is declared with 3 parts:

1. Data type –every variable in Java has to be declared with its data type; it starts with lower case

2. Variable name –always start with lower case alphabetical or underscore character ( \_ ) but not numeric

3. Initial value –optional

Sample:

COMPLEX VARIABLE

• Complex variables are instances of classes.

• Same with primitive variables, they’re declared in 3 parts

• Data type starts with upper case

• Initialization uses keyword and class constructor

• The name of the constructor method must match with the name of the class

• It has 3 parts: Data type, Variable name and Initial value (optional)

USING THE VARIABLES

Local Variables

• Variables declared inside a function are local to the function

• Access modifiers cannot be used for local variables.

Class Variables

• Also known as Fields, when declared outside a function it belongs to the whole class

Base class have the main function to call its subclass

Base class is the same as Super class or Parent class

Subclass is the same as Child class

Activity 1 Variables

Class Rectangle.java base class

public class Rectangle

{

int length;

double width;

public void computeArea()

{

double area = length \* width;

System.out.println("The are of the rectangle is: " + area);

}

public void computePerimeter()

{

double perimeter = (2 \* length) + (2 \* width);

System.out.println("The perimeter of the rectangle is: " + perimeter);

}

}

Instantiation.java Rectangle.java subclass

public class Instantiation {

public static void main(String[] args)

{

//Initialize complex variable rectangle using Rectangle class constructor method

Rectangle rectangle = new Rectangle();

//Instantiate properties of new rectangle

rectangle.length = 2;

rectangle.width = 3;

//Call class methods

rectangle.computeArea();

rectangle.computePerimeter();

}

}

Output:

Activity 2 Data Types

Activity2\_dataTypes.java

dataTypes.java base class

public class dataTypes {

int qty;

float price;

char code;

boolean isBought;

String message;

public void printQty()

{

System.out.println("qty = " + qty);

}

public void printPrice()

{

System.out.println("price = " + price);

}

public void printCode()

{

System.out.println("code = " + code);

}

public void printBought()

{

System.out.println("isBought = " + isBought);

}

public void printMessage()

{

System.out.println("message = " + message);

}

}

Activity2\_dataTypes.java subclass

public class Activity2\_dataTypes

{

public static void main(String[] args)

{

dataTypes dataType = new dataTypes();

dataType.qty = 23;

dataType.price = 16.50f;

dataType.code = 'C';

dataType.isBought = false;

dataType.message = "Hello!!";

dataType.printQty();

dataType.printPrice();

dataType.printCode();

dataType.printBought();

dataType.printMessage();

}

}

Output:

JAVAOPERATORS

Java provides different kinds of operators.

Basic Types of Operators:

1. Mathematical

2. Equality or Relational

3. Assignment

4. Conditional

EQUALITY OPERATORS

CONDITIONAL OPERATORS

Activity 3

Operators.java base class

public class Operators

{

double Grade1, Grade2, Grade3;

public void printGrades()

{

System.out.println("Grade 1 = " + Grade1);

System.out.println("Grade 2 = " + Grade2);

System.out.println("Grade 3 = " + Grade3);

}

public void printAverage()

{

double average = (Grade1 + Grade2 + Grade3) / 3.0;

System.out.println("The average is: " + average);

}

}

Operators.java subclass

public class Activity3\_Operators

{

public static void main(String[] args)

{

Operators operator = new Operators();

operator.Grade1 = 88;

operator.Grade2 = 90;

operator.Grade3 = 96;

operator.printGrades();

operator.printAverage();

}

}

Output:

JAVA CONTROL STATEMENT

1. IF STATEMENT

It tells the program to execute a certain part of the code only if the particular test evaluates to true.

2. ELSE-IF STATEMENT

3. SWITCH STATEMENT

4. WHILE LOOP

Variable Trace Table

5. FOR LOOP

Variable Trace Table

6. DO WHILE LOOP

Variable Trace Table

ControlFlow.java base class

public class ControlFlow

{

String name;

int printTimes;

public void nameLine()

{

System.out.println("Name to print: " + name);

}

public void printLines()

{

System.out.println("Number of times to print: " + printTimes);

}

public void namePrint()

{

if (printTimes < 4)

{

System.out.println(name);

}

else

{

printTimes = printTimes - 1;

while (printTimes != 0)

{

System.out.println(name);

printTimes--;

}

}

}

}

ControlFlow.java subclass

public class Activity4\_ControlFlow

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

ControlFlow controlFlow = new ControlFlow();

controlFlow.name = "John";

controlFlow.printTimes = 5;

controlFlow.nameLine();

controlFlow.printLines();

controlFlow.namePrint();

}

}

Output:

Calculator.java base class

public class Calculator

{

int appleOrder;

public void compute()

{

float totalAmountDue;

float orderPrice = 50.00f;

float totalOrderedPrice = appleOrder \* orderPrice;

float tax = totalOrderedPrice \* 0.12f;

float deliveryCharge = 250.00f;

if (totalOrderedPrice > 1000 )

{

deliveryCharge = 0;

totalAmountDue = totalOrderedPrice + deliveryCharge;

}

else

{

totalAmountDue = totalOrderedPrice + deliveryCharge + tax;

}

System.out.println("How many apples do you want to order? " + appleOrder);

System.out.println("Order Price: " + totalOrderedPrice);

System.out.println("Tax: " + tax);

System.out.println("Delivery Charge: " + deliveryCharge);

System.out.println("TOTAL: " + totalAmountDue);

}

}

Calculator.java subclass

public class Activity5\_Calculator

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

Calculator calculator = new Calculator();

calculator.appleOrder = 30;

calculator.compute();

}

}

Output:

BASIC JAVACONCEPTS

1. ENCAPSULATION

• It is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as single unit.

• Access to individual functions may be restricted

• The true nature of encapsulated data may be hidden

• Functions can be break into small maintainable units which will be helpful in the long term

ENCAPSULATION: SAMPLE

Encapsulation.java base class

public class Encapsulation {

// private variables

private String name;

private String color;

private String breed;

// public accessible methods

public String getDogName()

{

return name;

}

public String getDogColor()

{

return color;

}

public String getDogBreed()

{

return breed;

}

public void setName(String DogName)

{

name = DogName;

}

public void setColor(String DogColor)

{

color = DogColor;

}

public void setBreed(String DogBreed)

{

breed = DogBreed;

}

}

EncapsulationMain.java subclass

public class EncapsulationMain {

public static void main(String[] args) {

// TODO Auto-generated method stub

Encapsulation newEncapsulation = new Encapsulation();

newEncapsulation.setName("Max");

newEncapsulation.setColor("White");

newEncapsulation.setBreed("Shitzu");

System.out.println("Dog Name: " + newEncapsulation.getDogName());

System.out.println("Color: " + newEncapsulation.getDogColor());

System.out.println("Breed: " + newEncapsulation.getDogBreed());

}

}

2. INHERITANCE

Inheritance

is the relationship between multiple classes It can be defined as the process where one class (child class) acquires the properties (methods and fields) of another (parent class).

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class)

INHERITANCE: SAMPLE

Inheritance.java base class

public class Inheritance

{

private String name;

private String color;

public String breed = "Shitzu";

public int eyes = 2;

public String getDogName()

{

return name;

}

public String getDogColor()

{

return color;

}

public String getDogBreed()

{

return breed;

}

public void setName(String DogName)

{

name = DogName;

}

public void setColor(String DogColor)

{

color = DogColor;

}

public void setBreed(String DogBreed)

{

breed = DogBreed;

}

protected void feet()

{

int feet = 4;

System.out.println("Dog have " + feet + "feet.");

}

}

Inheritance.java subclass

The “extends” keyword is used to inherit a class

public class InheritanceChildClass extends Inheritance

{

public static void main(String[] args)

{

Inheritance newInheritance = new Inheritance();

newInheritance.setName("Max");

newInheritance.setColor("White");

System.out.println("Dog name: " + newInheritance.getDogName());

System.out.println("Color: " + newInheritance.getDogColor());

System.out.println("Breed: " + newInheritance.getDogBreed());

}

public void DogBody()

{

feet();

System.out.println("Dog have " + eyes + "eyes.");

}

}

INHERITANCE: DISCUSSION

• The instance variable of parent class (breed) can be accessed using object reference of child class inside static methods (newDog.breed).

• The non-static method of parent class (getDogName, getDogColor) can be accessed using object reference of child class inside static methods (newDog.getDogName, newDog.getDogColor).

• The static class variables of parent class (eyes) can be directly accessed inside the child class’ non static methods.

• The non-static methods of parent class (Feet()) can be directly accessed inside the child class’ non static methods.

• The private variable of parent class (name, color) cannot be accessed directly in child class.

3. POLYMORPHISM

It refers to as the ability of an object to take on many forms.

It allows you define one interface and have multiple implementations.

Following concepts demonstrate different types of polymorphism in java:

• Method Overloading

• Method Overriding

METHOD OVERLOADING& OVERRIDING

1. Overloading

Methods can be overloaded when it is created with the same name but the arguments list or parameters are different.

2. Overriding

Methods can be overridden in a child class when the method is created with the same name, argument list, and parameters as that of it’s parent class.

Methods declared as private, static, and final cannot be overridden.

METHOD OVERLOADING: SAMPLE

PolymorphismMethodOverloading.java base class

public class PolymorphismMethodOverloading

{

public void demo (int a)

{

System.out.println("a is: " + a);

}

public void demo (int a, int b)

{

System.out.println ("a and b are: " + a + " and " + b);

}

public double demo(double a)

{

System.out.println("double is: " + a);

return a \* a;

}

}

PolymorphismMethodOverloadingChild.java subcass

public class PolymorphismMethodOverloadingChild

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

PolymorphismMethodOverloading newPolymorphismMethodOverloading = new PolymorphismMethodOverloading();

double result;

newPolymorphismMethodOverloading.demo(5);

newPolymorphismMethodOverloading.demo(5,10);

result = newPolymorphismMethodOverloading.demo(5.5);

System.out.println("Result is: " + result);

}

}

METHOD OVERRIDING: SAMPLE

MethodOverriding.java base class

public class MethodOverriding {

public void MethodToOverride()

{

System.out.println("This is a method BaseClass.");

}

}

MethodOverridingSubclass.java sub class

public class MethodOverridingSubclass extends MethodOverriding

{

public void MethodToOverride()

{

System.out.println("This is a method in SubClass.");

}

}

MethodOverridingTestMethod.java subclass

public class MethodOverridingTestMethod

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

MethodOverriding parent = new MethodOverriding();

MethodOverridingSubclass child = new MethodOverridingSubclass();

parent.MethodToOverride();

child.MethodToOverride();

}

}

4. ABSTRACTION

A class which is declared as “ is an abstract class This also applies to

Methods

AbstractClass

• It may or may not include abstract methods

• If a class includes abstract methods, then the class itself must be declared abstract

• Abstract classes cannot be instantiated, but they can be subclassed

• To use an abstract class you have to inherit it from another class, provide all implementations to the abstract methods in it

Abstract Method

• Is a method which is declared abstract

• An abstract method is a method that is declared without an implementation

• If a class includes abstract methods, then the class itself must be declared abstract

Abstraction.java base class

public class Abstraction

{

public void method()

{

}

}

AbstractionChild.java base class

public class AbstractionChild extends Abstraction

{

public void method()

{

System.out.println("I'm overriding abstract method");

}

public static void main(String[] args)

{

// TODO Auto-generated method stub

AbstractionChild newAbstractionChild = new AbstractionChild();

newAbstractionChild.method();

}

}

SUMMARY

Java is a powerful Object Oriented Programming language developed by Sun Microsystems.

Java is an Object Oriented Language which is a approach to design a program using classes and objects.

Concepts supported by OOP:

• Object

• Class

• Encapsulation

• Inheritance

• Polymorphism

• Abstraction

LOCATING WEB ELEMENTS AND BASIC SELENIUM WEBDRIVER COMMANDS

LEARNING OBJECTIVES

At the end of this module, you will be able to:

–Learn how to identify and find web elements

–Identify the of use different Selenesecommands

FINDING HTML ELEMENTS

We need to identify the html elements we need to use for our test to run, we can do this by accessing the HTML Document Object Model (DOM) of the web application.

Step 1:To see the html codes of the web application, open a web browser then press F12in the keyboard.

This will display the console of the web browser which the attributes of the web elements can be seen. This can be done with other browsers.

Step 2:Click the Inspect button in the console then click on the part of the web page you want to inspect the attributes.

It will display the html tag of the web element including its attributes like id, name, class, etc. which can be used as a locator to run tests.

You can accomplish the same task by right clicking on the web element you want to inspect then click “Inspect element”

Inspecting of web elements can be done across different browsers.

LOCATORS

Locators need to be identified first before some Selenese commands to perform.

There are many ways to locate an element:

• ID

This is the most efficient way of locating an element.

Code to use (Java):

WebElementelement = driver.findElement(By.id("q"));

• Name

Find the input element with matching name attribute.

Sample web page source code:

Code to use (Java):

WebElementelement = driver.findElement(By.name("q"));

• Class Name

This refers to the attribute on the DOM element

Code to use (Java):

WebElementelement = driver.findElement(By.className("downloadBox"));

• Tag Name

The DOM Tag Name of the element.

Code to use (Java):

WebElementelement = driver.findElement(By.tagName(“table"));

• CSS

Like the name implies it is a locator strategy by css. Beware that not all browsers were created equal, some css that might work in one version may not work in another.

Sample code to use (Java):

WebElementelement =driver.findElement(By.cssSelector("#q"));

• Link Text/ Partial Link Text

Find the link element with matching visible text.

Sample web page source code:

Code to use (Java):

WebElementelement = driver.findElement(By.linkText(“Download Selenium"));

•Xpath

WebDriver uses a browser’s native XPath capabilities wherever possible. Xpath is a syntax for defining parts of an XML document.

Sample web page source code:

Code to use (Java):

WebElementelement = driver.findElement(By.xpath("//input[@id='q']"));

Or "//input[@name=\"userName\"]"

WILD CARD \* CHARACTER

\* - is the one of most used wild card character with xpath in selenium webdriver, we can use it instead of tag name and attribute//\*-matches all the elements present in the html (including html)

//div/\* - matches all the immediate element(s) inside the div tag //input[@\*] - matches all the element(s) with input tag and have at least one attribute, attribute value may or may not present//\*[@\*]-matches all the element(s) which have at least one attribute.

BASIC SELENESE COMMANDS

After the locators have been identified we can now start passing actions to those web elements, but first we need to open a browser and navigate to a page.

We can open a Firefox browser by using:

System.setProperty("webdriver.gecko.driver",<Path of geckodriver.exe>);

WebDriver driver = new FirefoxDriver();

and navigate to a page by using any of these two methods:

driver.get("http:www.google.com");

driver.navigate().to("http:www.google.com");

1. Typing a Value

Example:

WebElementelement = driver.findElement(By.name("q"));

element.sendKeys("Cheese!");

2. Clicking an Element

Example:

WebElementelement = driver.findElement(By.xpath(“smpleElem"));

element.click();

3. Selecting a value from Dropdown

There are 2 ways to select from a dropdown, one is by visible text and other by index.

Example:

Select select= new Select(driver.findElement(By.tagName("select")));

select.selectByVisibleText("Edam");

Select select= new Select(driver.findElement(By.tagName("select")));

select.selectByIndex(1);

4. Switching Windows

Example:

If we know the name of the window we can directly use:

driver.switchTo().window("windowName");

Or if the window name is not known:

for (String handle : driver.getWindowHandles()) {

driver.switchTo().window(handle);

}

5. Switching to Frames

Example:

driver.switchTo().frame("frameName");

6. Handling Popup Alert

Example:

Alert alert= driver.switchTo().alert();

alert.accept();

7. Navigation

The “navigate” interface also exposes the ability to move backwards and forwards in your browser’s history:

Example:

driver.navigate().forward();

driver.navigate().back();

Closing a Browser

Example:

driver.quit();

Wait command

Waiting is having the automated task execution pause a certain amount of time before continuing with the next step. It is important to include this command to lessen the script failure due to the web application’s unpredictable loading time. There are 2 kinds of wait:

• Implicit

• Explicit

1. Implicit Wait

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object instance.

Sample code:

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http:somedomain/url\_that\_delays\_loading");

WebElementmyDynamicElement= driver.findElement(By.id("myDynamicElement"));

2. Explicit Wait

An explicit waits is code you define to wait for a certain condition to occur before proceeding further in the code. There are many conditions that can be applied in each of the explicit wait used.

Sample code:

WebDriver driver = new FirefoxDriver();

driver.get("http:somedomain/url\_that\_delays\_loading");

WebElementmyDynamicElement= (new WebDriverWait(driver, 10)) .until(ExpectedConditions.presenceOfElementLocated(By.id("myDynamicElement")));

Selenium project steps:

1. Create new java project

2. Click Next

3. Click Libraries tab

4. "Add External JARs..."

5. Add selenium server standalone in Libraries by adding

6. Click Finish

7. Paste chromedriver.exe to project folder in workspace

FirstSeleniumScript.java

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedCondition;

import org.openqa.selenium.support.ui.WebDriverWait;

public class FirstSeleniumScript

{

public static void main(String[] args)

{

// Create a new instance of the Chrome driver

// Notice that the remainder of the code relies on the interface,

// not the implementation.

// 1. Open chrome browser

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "/chromedriver.exe");

// 2. Instantiate new WebDriver object of the web browser

WebDriver driver = new ChromeDriver();

// 3. Navigate to URL with driver.get() or driver.nagigate().to() methods

driver.get("http://www.google.com");

// Alternatively the same thing can be done like this

// driver.navigate().to("http://www.google.com");

// 4. Verify title of the page

System.out.println("Page title is: " + driver.getTitle());

// 5. Instantiate new WebElement to find element with browser findElement

WebElement element = driver.findElement(By.name("q"));

// 6. Enter values to the new object element

element.sendKeys("Cheese!");

// 7. Now submit the form. WebDriver will find the form for us from the element

element.submit();

// Google's search is rendered dynamically with JavaScript.

// 8. Wait for the page to load, timeout after 10 seconds

(new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>()

{

public Boolean apply(WebDriver d)

{

return d.getTitle().toLowerCase().startsWith("cheese!");

}

});

// 9. Should see: "cheese! - Google Search" in page title

System.out.println("Page title is: " + driver.getTitle());

// 10. Close the browser

driver.quit();

}

}

Browser output:

Console Output:

Starting ChromeDriver 2.38.552522 (437e6fbedfa8762dec75e2c5b3ddb86763dc9dcb) on port 26236

Only local connections are allowed.

Nov 05, 2019 9:20:21 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

Page title is: Google

Page title is: Cheese! - Google Search

M4.java Selenium test script

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

public class M4

{

public static void main(String[] args)

{

// Open chrome browser

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "/chromedriver.exe");

// Instantiate new WebDriver object of the web browser

WebDriver driver = new ChromeDriver();

// Navigate to URL with driver.get() or driver.nagigate().to() methods

driver.get("http://newtours.demoaut.com/");

// Verify title of the page

System.out.println("Page title is: " + driver.getTitle());

// Instantiate new WebElement to find REGISTER link with browser findElement

WebElement element = driver.findElement(By.linkText("REGISTER"));

// Click the REGISTER link

element.click();

// Find registration elements and fill values with sendKeys, select dropdown and click Submit button

driver.findElement(By.xpath("//input[@name=\"firstName\"]")).sendKeys("Claive Alvin");

driver.findElement(By.xpath("//input[@name=\"lastName\"]")).sendKeys("Acedilla");

driver.findElement(By.xpath("//input[@name=\"phone\"]")).sendKeys("0123456789");

driver.findElement(By.xpath("//input[@name=\"userName\"]")).sendKeys("cabpacedilla@gmail.com");

driver.findElement(By.xpath("//input[@name=\"address1\"]")).sendKeys("Lawaan, Talisay");

driver.findElement(By.xpath("//input[@name=\"city\"]")).sendKeys("Cebu");

driver.findElement(By.xpath("//input[@name=\"state\"]")).sendKeys("Visayas");

driver.findElement(By.xpath("//input[@name=\"postalCode\"]")).sendKeys("6045");

Select select = new Select(driver.findElement(By.name("country")));

select.selectByVisibleText("PHILIPPINES");

driver.findElement(By.xpath("//input[@name=\"email\"]")).sendKeys("cabpa");

driver.findElement(By.xpath("//input[@name=\"password\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"confirmPassword\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"register\"]")).click();

// Get Registration page title

System.out.println("Page title is: " + driver.getTitle());

// Click sign-in link in Registration confirmation message

WebElement signIn = driver.findElement(By.linkText("sign-in"));

signIn.click();

// Fill up credentials and click login

driver.findElement(By.xpath("//input[@name=\"userName\"]")).sendKeys("cabpa");

driver.findElement(By.xpath("//input[@name=\"password\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"login\"]")).click();

// Get successful login page title

System.out.println("Page title is: " + driver.getTitle());

}

}

Console output:

Web browser output

WHAT IS TESTNG?

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

•Benefits of using TestNG

-Uses annotations to control the flow of testing

Annotations:

• @BEFOREMETHOD

• @AFTERMETHOD

• @TEST

• @BEFORETEST

• @AFTERTEST

-Generates report in a readable format

INSTALLING TESTNG IN ECLIPSE

Step 1: On the menu bar of Eclipse, click “Help”.

Step 2: Select the “Eclipse Marketplace…” option.

Step 3: From the Search Tab type “TestNG” on the search box and press enter.

Step 4: Click Install button.

Step 5: Click Confirm button on the next window.

Step 5: Click Confirm button on the next window.

Step 6: Select option: “I accept the terms of the license agreement” then click Finish

button.

Step 6: Wait for the software to install.

Step 7: Click Yes button to restart Eclipse.

CREATING TESTNG TEST

Step 1: In the “Package Explorer” section, right click on the “src” folder.

Step 2: Hover on “New” option.

Step 3: Select “Class” option.

Step 6: Add methods to the class.

Step 7: Add “@Test” annotation for each method.

Step 8: Hover on the “@Test” annotation.

Step 9: Select “Import ‘Test’ (org.testing.annotations) option.

Step 10: Save the file.

Step 11: Convert script to TestNG

Step 12: Run script as TestNG in testing.xml

FirstTestNGTest.java

import org.testng.annotations.Test;

public class FirstTestNGTest

{

@Test

public void test1()

{

System.out.println("Test number 1");

}

@Test

public void test2()

{

System.out.println("Test number 2");

}

}

M3\_Activity1.java

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class M3\_Activity1

{

// 1. Create a new instance of the Chrome driver

// Notice that the remainder of the code relies on the interface,

// not the implementation.

public WebDriver driver = new ChromeDriver();

@BeforeTest

public void setUpThenNavURL()

{

System.setProperty("webdriver.chrome.driver", "C:\\Workspace\\SeleniumProject\\chromedriver.exe");

// 2. Navigate to test URL

driver.get("http://newtours.demoaut.com/");

}

@Test

public void registerThenLogin()

{

// 3. Find the element by its locators and populate all fields

WebElement registerButton = driver.findElement(By.xpath("/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[2]/td/table/tbody/tr/td[2]/a"));

registerButton.click();

WebElement firstName = driver.findElement(By.name("firstName"));

firstName.sendKeys("Claive");

WebElement lastName = driver.findElement(By.name("lastName"));

lastName.sendKeys("Acedilla");

WebElement phone = driver.findElement(By.name("phone"));

phone.sendKeys("0123456789");

WebElement userName = driver.findElement(By.name("userName"));

userName.sendKeys("boyet");

WebElement address1 = driver.findElement(By.name("address1"));

address1.sendKeys("Lawaan, Talisay");

WebElement address2 = driver.findElement(By.name("address2"));

address2.sendKeys("Pioneer Building, Ayala Business Park");

WebElement city = driver.findElement(By.name("city"));

city.sendKeys("Cebu");

WebElement state = driver.findElement(By.name("state"));

state.sendKeys("Visayas");

WebElement postalCode = driver.findElement(By.name("postalCode"));

postalCode.sendKeys("6000");

Select selectCountry = new Select(driver.findElement(By.xpath("/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[5]/td/form/table/tbody/tr[12]/td[2]/select")));

selectCountry.selectByVisibleText("PHILIPPINES");

WebElement email = driver.findElement(By.name("email"));

email.sendKeys("cabpacedilla@gmail.com");

WebElement password = driver.findElement(By.name("password"));

password.sendKeys("vippassword");

WebElement confirmPassword = driver.findElement(By.name("confirmPassword"));

confirmPassword.sendKeys("vippassword");

// 4. Click Submit button to register the account

WebElement submitBtn = driver.findElement(By.xpath("/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[5]/td/form/table/tbody/tr[18]/td/input"));

submitBtn.click();

// 5. Create Account Success will appear, click the sign-in link

WebElement signIn = driver.findElement(By.xpath("/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[3]/td/p[2]/font/a[1]"));

signIn.click();

// 6. Login with registered credentials in Sign-in page

WebElement userNameLogin = driver.findElement(By.name("userName"));

userNameLogin.sendKeys("boyet");

WebElement passwordLogin = driver.findElement(By.name("password"));

passwordLogin.sendKeys("vippassword");

// 7. Click Submit button to login

WebElement submitBtnLogin = driver.findElement(By.xpath("/html/body/div/table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[5]/td/form/table/tbody/tr[4]/td/input"));

submitBtnLogin.click();

}

@AfterTest

public void closeBrowser()

{

// 8. Close the browser

driver.quit();

}

}

M5.java TestNG

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import org.testng.annotations.AfterTest;

public class M5

{

// Declare global WebDriver variable

WebDriver driver;

@BeforeTest

public void openBrowser()

{

// Open chrome browser

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "/chromedriver.exe");

// Instantiate new WebDriver object of the web browser

driver = new ChromeDriver();

}

@Test(priority=1)

public void loadURL()

{

// Navigate to URL with driver.get() or driver.nagigate().to() methods

driver.get("http://newtours.demoaut.com/");

}

@Test(priority=2)

public void assertMainPageTitle()

{

// Verify title of the page

System.out.println("Page title is: " + driver.getTitle());

}

@Test(priority=3)

public void clickRegister()

{

// Instantiate new WebElement to find REGISTER link with browser findElement

WebElement element = driver.findElement(By.linkText("REGISTER"));

// Click the REGISTER link

element.click();

}

@Test(priority=4)

public void fillRegister()

{

// Find registration elements and fill values with sendKeys, select dropdown and click Submit button

driver.findElement(By.xpath("//input[@name=\"firstName\"]")).sendKeys("Claive Alvin");

driver.findElement(By.xpath("//input[@name=\"lastName\"]")).sendKeys("Acedilla");

driver.findElement(By.xpath("//input[@name=\"phone\"]")).sendKeys("0123456789");

driver.findElement(By.xpath("//input[@name=\"userName\"]")).sendKeys("cabpacedilla@gmail.com");

driver.findElement(By.xpath("//input[@name=\"address1\"]")).sendKeys("Lawaan, Talisay");

driver.findElement(By.xpath("//input[@name=\"city\"]")).sendKeys("Cebu");

driver.findElement(By.xpath("//input[@name=\"state\"]")).sendKeys("Visayas");

driver.findElement(By.xpath("//input[@name=\"postalCode\"]")).sendKeys("6045");

Select select = new Select(driver.findElement(By.name("country")));

select.selectByVisibleText("PHILIPPINES");

driver.findElement(By.xpath("//input[@name=\"email\"]")).sendKeys("cabpa");

driver.findElement(By.xpath("//input[@name=\"password\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"confirmPassword\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"register\"]")).click();

}

@Test(priority=5)

public void assertRegisterPageTitle()

{

// Get Registration page title

System.out.println("Page title is: " + driver.getTitle());

}

@Test(priority=6)

public void signIn()

{

// Click sign-in link in Registration confirmation message

WebElement signIn = driver.findElement(By.linkText("sign-in"));

signIn.click();

}

@Test(priority=7)

public void Login()

{

// Fill up credentials and click login

driver.findElement(By.xpath("//input[@name=\"userName\"]")).sendKeys("cabpa");

driver.findElement(By.xpath("//input[@name=\"password\"]")).sendKeys("password");

driver.findElement(By.xpath("//input[@name=\"login\"]")).click();

}

@Test(priority=8)

public void assertLoginSuccessPageTitle()

{

// Get successful login page title

System.out.println("Page title is: " + driver.getTitle());

}

@AfterTest

public void closeBrowser()

{

driver.quit();

}

}

Output:

[RemoteTestNG] detected TestNG version 6.14.3

Starting ChromeDriver 2.38.552522 (437e6fbedfa8762dec75e2c5b3ddb86763dc9dcb) on port 39301

Only local connections are allowed.

Oct 13, 2019 7:22:07 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

Page title is: Welcome: Mercury Tours

Page title is: Register: Mercury Tours

Page title is: Find a Flight: Mercury Tours:

PASSED: loadURL

PASSED: assertMainPageTitle

PASSED: clickRegister

PASSED: fillRegister

PASSED: assertRegisterPageTitle

PASSED: signIn

PASSED: Login

PASSED: assertLoginSuccessPageTitle

===============================================

Default test

Tests run: 8, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 8, Failures: 0, Skips: 0

===============================================

Browser output:

WHAT IS A FRAMEWORK

A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

•is a scaffolding that is laid to provide an execution environment for the automation test scripts.

•It provides the user various benefits that helps them to develop, execute and report the automation test scripts efficiently.

•Test automation framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

Some of the known Test Automation Frameworks for Selenium:

•Linear Automation Framework

•Module Based Testing Framework

•Library Architecture Testing Framework

•Data Driven Testing Framework

•Keyword Driven Testing Framework

•Behavior Driven Development Framework

•Hybrid Testing Framework

1. Linear Automation Framework

•Based on record and play and follow the procedural code.

•Steps are written in a sequential order.

2. Module Based Testing Framework

•Based on one of the popularly known OOPs concept –Abstraction

•Divides the entire “Application Under Test” into a number of logical and isolated modules

•After breaking down the application into individual modules, a test script is created for each part and then combined to build larger tests

3. Library Architecture Testing Framework

•Based on the modular framework, but has some additional benefits.

•Instead of dividing the application under test into test scripts, we segregate the application into functions or rather common functions can be used by the other parts of the application as well

•Requires creation of common library constituting of common functions for the application under test.

•These libraries can be called from the test scripts whenever required.

4. Data Driven Testing Framework

•Segregates the test script logic and test data.

•Test data is stored in an external database.

–The data is conventionally stored in key-valuepairs. Thus, the key can be used to access and populate the data within the test scripts.

The external databases can be:

•XML

•excel file

•.txt

•.csv

•ODBC repositories

Sample database structure for D.A.T.A

5. Keyword Driven Testing Framework

The Keyword driven testing framework

•An extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts.

•It also keeps the certain set of code belonging to the test script into an external data file.

•The keywords and the test data are stored in a tabular like structure like in the data driven structure.

6. Behavior Driven Development Framework

The Behavior Driven Development (BDD) framework

•BDD framework is an agile testing framework derived from BDD methodology in which an application is specified and designed by describing how its behavior should appear to an outside observer.

•Most of BDD frameworks use Gherkin language such as Given, When, And and Then for writing tests

•The focus of BDD is on identifying required behavior in the user story and writing acceptance tests based on them

7. Hybrid Testing Framework

Hybrid Testing Framework

•Combination of any of the previously mentioned frameworks set up to leverage the advantages of some and mitigate the weaknesses of others.

•As the concept implicates, it leverages the benefits of all the test frameworks.

SUMMARY

•Test Automation Framework provides the user various benefits that helps them to develop, execute and report the automation test scripts efficiently.

•Advantages

•Ease of scripting

•Modularity

•Easy to understand

•Defines every process

•Re-usability

•Cost

•Easy to maintain

Types of Automation Frameworks

•Linear Automation Framework

•Module Based Testing Framework

•Library Architecture Testing Framework

•Data Driven Testing Framework

•Keyword Driven Testing Framework

•Behavior Driven Development Framework

•Hybrid Testing Framework

SELENIUM HYBRID TESTING FRAMEWORK

FRAMEWORKOVERVIEW

For this course we’ll be using a simple framework for Selenium which can be used as a baseline:

•The easiest way to setup a Selenium project is to use Maven. Maven will download the java bindings and all its dependencies, and will create the project for you using a project configuration file.

•We’ll be implementing the hybrid testing framework, as discussed in the previous chapter. This framework utilizes module based and data driven testing framework.

•Page object model will be applied as well. This will greatly help in the maintenance and reusability of the codes.

•TestNG is also integrated with the framework to include validations and better test management.

•After the execution, a test report will be generated including the screenshots.

MAVEN OVERVIEW

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project’s build, reporting and documentation from a central piece of information5

1. Developers declare required jar/library (for example: Spring, Servlet, Hibernate, etc.) on pom.xml.

2. Developers using some libraries on their source codes.

3. The source codes go to compiling and building phase.

4. At the compiling and building phase, the IDE will call the Maven to build the Java project.

5. Maven reads required library on pom.xml.

6. Maven downloads required library from offline repository, or …

7. Maven downloads required library from online repository. And then Maven will 'install' the library into targeted Java project.

8. Compiling and building process will produce a Java apps (either .jar or .war).

SETTING UP MAVEN

Apache Maven is a software project management and comprehension tool that can manage a project's build, reporting and documentation from a central piece of information. It will download the java bindings and all its dependencies, and will create the project for you using a project configuration file.

1. Right click on the project file > Configure > click “Convert to Maven Project”

2. Click Finish button and wait for the loading to finish.

3. A new file, “pom.xml”, will be created under the project files.

MAVEN POM

In the pom.xml file we can manage the configurations settings for the project file. We’ll be adding the jar files needed for the project but first removed the jar file added in the project library.

ADDING DEPENDENCIES IN POM

Step 1: We add the “dependencies” tag first in the pom.xml and then in between those tags we add all the dependencies needed for the project

Step 2: In the Maven dependencies repositories website (http://mvnrepository.com/) you can search for the jar files needed and copy the code given for maven.

Step 3: Paste the copied code between the “dependencies” tag, then save the pom.xml file.

After saving you can notice that the project is currently building, it means that Maven started downloading the added dependency.13

All the files that will be downloaded by Maven will be stored in the “repository” folder, you can find it in C:\Users\<eid>\.m2

PAGEOBJECTS

It is a popular test automation design pattern for improving test and lessening the duplication of codes. A page object is an object oriented class which the test codes are separated from the page specific codes.

Rules in creating Page Objects:

• Page objects themselves should never make verifications or assertions. It should included in the test itself.

• The only validation/ assertion that the page object should contain are the critical elements of the page like if the URL of the page is loaded correctly.

• Page objects should contain the representation of the page and the actions that can be executed within the page, but it doesn’t really need to include all the components of the page.

CREATING PAGEOBJECTS

We’ll apply the page objects model to “FirstSeleniumScript.java” script. Each of the action in the test will be divided into individual component.

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedCondition;

import org.openqa.selenium.support.ui.WebDriverWait;

public class FirstSeleniumScript

{

public static void main(String[] args)

{

// 1. Create a new instance of the Chrome driver

// Notice that the remainder of the code relies on the interface,

// not the implementation.

System.setProperty("webdriver.chrome.driver", "C:\\Workspace\\SeleniumProject\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

// 2. And now use this to visit Google

driver.get("http://www.google.com");

// Alternatively the same thing can be done like this

// driver.navigate().to("http://www.google.com");

// 3. Check the title of the page

System.out.println("Page title is: " + driver.getTitle());

// 4. Find the text input element by its name

WebElement element = driver.findElement(By.name("q"));

// 5. Enter something to search for

element.sendKeys("Cheese!");

// 6. Now submit the form. WebDriver will find the form for us from the element

element.submit();

// Google's search is rendered dynamically with Javascript.

// 7. Wait for the page to load, timeout after 10 seconds

(new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>()

{

public Boolean apply(WebDriver d)

{

return d.getTitle().toLowerCase().startsWith("cheese!");

}

});

// Should see: "cheese! - Google Search"

// 8. Get page title again

System.out.println("Page title is: " + driver.getTitle());

// 9. Close the browser

driver.quit();

}

}

CREATING PAGEOBJECTS

Step 1: Create a new package named “common” then create an abstract class named “BasePage” under it.

We will add the basic commands in the BasePage that will be inherited by other classes.

BasePage.java common package

package common;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public abstract class BasePage {

protected WebDriver driver;

public BasePage(WebDriver driver)

{

this.driver = driver;

}

public void click (String locator)

{

WebElement element = driver.findElement(By.name(locator));

element.click();

System.out.println();

}

public void enterText(String locator, String value)

{

WebElement element = driver.findElement(By.name(locator));

element.sendKeys(value);

System.out.println("Text has been entered");

}

}

Step 2: Create a new package named “pageobjects” then create a new class under it and named it “GooglePage”

Step 3: In the “GooglePage” class inherit the “BasePage” abstract class. Call out the method from BasePageclass and create specific commands that can be used for google home page.

GooglePage.java pageobject

package pageobjects;

import org.openqa.selenium.WebDriver;

import common.BasePage;

public class GooglePage extends BasePage

{

public GooglePage(WebDriver driver)

{

super(driver);

}

public void enterSearchBar(String value)

{

enterText("q", value);

System.out.println("Entered valued in the google search bar...");

}

public void clickSearchButton()

{

click("btnG");

System.out.println("Clicked search button...");

}

}

Step 4: Create a new class named “BaseTest” under the “common” package.

Step 5: In the “BaseTest” class create individual methods for global commands that can be used across different pages in the web application.

This commands will be directly called out in the test class/ script.19

BaseTest.java common package

package common;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class BaseTest {

public static WebDriver driver;

public void openBrowser()

{

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "/chromedriver.exe");

driver = new ChromeDriver();

System.out.println("Test is running in Chrome");

}

public void openURL(String url)

{

driver.get(url);

System.out.println("Opened the url.");

}

public void endTest()

{

driver.quit();

System.out.println("This test is finished.");

}

}

CREATING THE TEST

Step 1: Create a new package named “test” then create a class named “GoogleTest” under it. Inherit the “BaseTest” class.

Step 2: Create a class then include “@Test” annotation. Call out the needed steps from “GooglePage” and “BaseTest” then arrange then in a sequential order.20

// GoogleTest.java test package

package test;

import org.testng.annotations.Test;

import common.BaseTest;

import pageobjects.GooglePage;

public class GoogleTest extends BaseTest{

@Test

public void Googletest() throws Exception

{

openBrowser();

openURL("http://www.google.com");

GooglePage actions = new GooglePage(driver);

actions.enterSearchBar("Cheese!");

actions.clickSearchButton();

endTest();

}

}

Step 1: Right click on the test > Run As > then click TestNGTest

Or you can use the shortcut key Alt + Shift + X, N21

Step 2: The test will run and after the execution it will display the logs in the “Console” tab.

Step 3:After the execution, the test will generate the default TestNGreports. Open the “emailable-report.html” file.

Step 4:The report will show if the test passed and the duration of the execution.

USING A FRAMEWORK

Going forward we will use a hybrid testing framework that is already built. The framework will provides pre build commands and functionalities for

•Access to data source

•Generating console and test report logs

•Taking screenshots after test failure or test successful completion

•Generating a report (Extent Report)

•Basic browser commands and browser run options

•Wait functionality for script and AUT synchronization

FRAMEWORK STRUCTURE

The diagram below show the structure of a sample hybrid framework.27

FRAMEWORK STRUCTURE

1. Core package

It contains utility classforhandling web browsers on different platform, accessing data source, capturing screenshots, generating customized readable logs as well as the report.28

2. Data Objects package

Contains class to create data objects which will store the value from the data source (xsl).

3. Web Elements package

Contains class for web page elements (e.g. button, textbox, ..) and all available actions that we can perform on the element (e.g. click, enter, ..)

4. Page Objects package

Contains class which are representation of each unique web pages. This is where element locators (e.g. id, name, xpath, etc.) is assigned to a web element to create the page objects.

Reusable methods for a specific page object actions is also created here which is used in developing the test scripts.

5. Tests package

Use Page Objects and Data Objects to create a test script and form a test scenario.

SETTING UP MAVEN

Apache Maven is a software project management and comprehension tool that can

manage a project's build, reporting and documentation from a central piece of

information. It will download the java bindings and all its dependencies, and will

create the project for you using a project configuration file.

How to setup projects to Maven:

Step 1: Follow the step by step guide in configuring your machine with Maven

Step 2: Open Eclipse application.

Step 3: Right click on the project file > Configure > click “Convert to Maven Project”

Step 4: Click Finish button and wait for the loading to finish.

Step 5: A new file, “pom.xml”, will be created under the project files.

In the pom.xml file we can manage the configurations settings for the project file. We’ll be adding the jar files needed for the project but first removed the jar file added in the project library.

Step 1: We add the “dependencies” tag first in the pom.xml and then in between those tags we add all the dependencies needed for the

project.

Step 2:

In the Maven dependencies repositories website ( http://mvnrepository.com/ com/) you can search for the jar files needed and copy the code given for maven.

Step 3: Paste the copied code between the “dependencies” tag, then save the pom.xml

file.

After saving you can notice that the project is currently building, it means that Maven started downloading the added

dependency.

All the files that will be downloaded by Maven will be stored in the “repository” folder, you can find it in

C: Users \\< eid>\\.m2

How to setup existing Maven projects

Step 1: Right click on the Package Explorer tab then click “Import”

Step 2: Click “Maven” > click “Existing Maven Projects” then click Next button

Step 3: Click the “Browse…” button

Step 4: Find and select the “SeleniumProject” folder (path: C:\Workspace) then click OK button.

Step 5: The selected folder should appear in the Projects. Click Finish button.

Step 6: The project will now be displayed in the Project Explorer tab. Maven will start downloading the files needed for the project file. Wait for the build of the codes to finish which is displayed on the bottom-right corner of Eclipse.

Step 7: Right click on the project > Maven > click “Update Project…”

Step 8: Click the checkbox “Force Update of Snapshots/Releases” then click OK button.

Step 9: Maven will start downloading the files needed for the project file. Wait for the build of the codes to finish which is displayed on the bottom-right corner of Eclipse.

Step 10: Make sure that there are no errors displayed in the “Problems” tab.

1. CREATING PAGE OBJECT

Step 1: Create a new class under the “com.seleniumbootcampframework.pageobjects” package and name it as <page name + Page>

Step 2: Inside the page object class, create a web element object constructor from the web element class (like Button, TextBox, Link) and initialize it with web element description as well as the web locator type and value.

Name the web element variable starting the underscore character <“\_” + Web Element Class + Web Element Name Description like ‘\_linkContactNo’>

private static Link \_linkContactNo = new Link("Contact Us", By.xpath("//a[@title='Contact Us']"));

2. CREATING ACTION IN PAGE OBJECT

Step 3: Create a method and name it using this format <action + Web Element name>.

If the command needed an input add parameters in the method

Note: You can group multiple page object actions into one method to create single command as given in the example for signInmethod.

MainPage.java

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Link;

public class MainPage {

public static class Header {

private static Link \_linkContactNo = new Link("Contact Us", By.xpath("//a[@title='Contact Us']"));

private static Link \_linkSignIn = new Link("Sign in", By.xpath("//a[@class='login']"));

public static void clickContactUsLink() {

\_linkContactNo.click();

}

public static void clickSignInLink() {

\_linkSignIn.click();

}

public static void verifyLoginUser(String userName) {

Link \_linkAccount = new Link("Account: " + userName,

By.xpath("//a[@title='View my customer account']/span[text()='" + userName + "']"));

\_linkAccount.verifyDisplayed();

}

}

}

3. MANAGING PAGE OBJECT ACTION

Step 4: Create nested sub classes (as appropriate) to group page object and page object methods based on the relative positioning on the application under test. This will add readability and maintainability with our code.

SignInPage.java

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

public class SignInPage {

public static class AlreadyRegistered {

private static Button \_buttonSignIn = new Button("Sign In", By.xpath("//button[@id='SubmitLogin']"));

private static Link \_linkForgotYourPassword = new Link("Forgot your password",

By.xpath("//a[@title='Recover your forgotten password']"));

private static TextBox \_textBoxEmailAddress = new TextBox("Email Address", By.xpath("//input[@id='email']"));

private static TextBox \_textBoxPassword = new TextBox("Password", By.xpath("//input[@id='passwd']"));

public static void clickForgotYourPasswordLink() {

\_linkForgotYourPassword.click();

}

public static void signIn(String emailAddress, String password) {

\_textBoxEmailAddress.setText(emailAddress);

\_textBoxPassword.setText(password);

\_buttonSignIn.click();

}

}

public static class CreateAnAccount {

private static Button \_buttonCreateAnAccount = new Button("Create an account",

By.xpath("//button[@id='SubmitCreate']"));

private static TextBox \_textBoxEmailAddress = new TextBox("Email Address",

By.xpath("//input[@id='email\_create']"));

public static void createAnAccount(String emailAddress) {

\_textBoxEmailAddress.setText(emailAddress);

\_buttonCreateAnAccount.click();

}

}

public static class ErrorMessage {

public static void verifyFailedAuthentication() {

Element \_elementFailedAuthenticationMessag = new Element("Authentication failed message",

By.xpath("//li[text()='Authentication failed.']"));

\_elementFailedAuthenticationMessag.verifyDisplayed();

}

}

}

4. CREATING THE TEST DATA OBJECT

Test data source is located at {Project root folder}\src\main\resources\Data.

We will use the existing excel file Datatable.xls to store our data and create data objects for it.

Step 1: Open the excel file. Starting from column B, add the column name for our data and start populating the value from those columns47

Step 2: Create a new class under the “com.seleniumbootcampframework. dataobjects” package and name it simply as TestData

5. CREATING THE TEST DATA OBJECT

Step 3: Inside the TestDataclass, create a method that will return cell value by using getCellValuemethod from our utility class DataTable.

Pass the column name followed by the row name in getCellMethod to get the cell value in our data source

Step 4: Use nested subclasses to group and categorize sets of data for your test.

TestData.java

package com.seleniumbootcampframework.dataobjects;

import java.io.IOException;

import com.seleniumbootcampframework.core.DataTable;

import jxl.read.biff.BiffException;

public class TestData {

public static class AccountInformation {

public static class RegisteredUser {

public static String FullName() throws BiffException, IOException {

return DataTable.getCellValue("Fullname", 3);

}

}

}

public static class Credentials {

public static class RegisteredUser {

public static String Password() throws BiffException, IOException {

return DataTable.getCellValue("Password", 3);

}

public static String Username() throws BiffException, IOException {

return DataTable.getCellValue("Username", 3);

}

}

public static class UnregisteredUser {

public static String Password() throws BiffException, IOException {

return DataTable.getCellValue("Password", 6);

}

public static String Username() throws BiffException, IOException {

return DataTable.getCellValue("Username", 6);

}

}

}

public static class Urls {

public static String myStoreUrl() throws BiffException, IOException {

return DataTable.getCellValue("URL", 3);

}

}

}

6. CREATING TEST SCRIPT

Step 1: Create new class under the “com.seleniumbootcampframework.test” package and name it based on the scenario or general functionality that the script will perform.

e.g. For the sign in scenario, we will name it SignInTest

Step 2: Create a method and add “@Test” annotation. Use test case name as the name of the method.

Step 3: Inside the method, add the methods we created in the pageobject and dataobject classes to create the script.

package com.seleniumbootcampframework.tests;

import org.testng.annotations.Test;

import com.seleniumbootcampframework.core.Browser;

import com.seleniumbootcampframework.core.Log;

import com.seleniumbootcampframework.dataobjects.TestData;

import com.seleniumbootcampframework.pageobjects.MainPage;

import com.seleniumbootcampframework.pageobjects.SignInPage;

public class SignInTest extends Browser {

@Test

public void signInWithUnregisteredUser() throws Exception

{

// Use prebuilt methods from utility class log to add test descriptions

Log.setStoryName("Sign In Functionality");

Log.setTestScriptName("Unregistered user sign in");

Log.setTestScriptDescription("Unsuccessful sign in using unregisterd account");

//Navigate to Sign in page

MainPage.Header.clickSignInLink();

// Sign in using unregistered user

SignInPage.AlreadyRegistered.signIn(TestData.Credentials.UnregisteredUser.Username(), TestData.Credentials.UnregisteredUser.Password());

// Verify failed authentication message

SignInPage.ErrorMessage.verifyFailedAuthentication();

}

}

7. RUNNING THE TEST

Step 1: Create a folder named “TestSuite”.

We need to setup the configurations for the test before it could be executed.

Step 2: Right click on the test script > TestNG > then click “Convert to TestNG”.53

Step 3: Enter “test” in the test name then click Finish button.

Step 4: Move the generated xml file to the “TestSuite” folder

Step 5: Open the xml file. Remove thread-count property and save.

Step 6: Open the pom.xml and provide the path of our testng.xml at suiteXmlFiletag.

Step 7: Update browser.instance property value to desired browser option (like chrome, firefox, ie, …).

Step 8: Update environment property value to indicate the excel sheet name of our data. Save pom.xml file.

Step 9: Right click on the pom.xml and select Run As>>Maven test

Step 10: The execution of the test will commence and the logs will be displayed in the Console tab.

Step 11: After the execution, the test will generate an Extent report with a screenshot at {Project root folder}\target\extentReport

Step 12: Open html report to check the execution status

SUMMARY

•Implementing a framework helps in creating reusable and maintainable set of codes which be beneficial in long term.

•Components of a hybrid framework

–Page Objects –implements OOP which greatly helps in code reusability and maintainability

–Maven –is a build tool used to setup a Selenium project easily which download the java bindings and dependencies.

–TestNG–provides better test management

–MS Excel –separates the test data from the test script itself for better test data management

Testscript SignInTest.java in src/test/java

package com.seleniumbootcampframework.tests;

import org.testng.annotations.Test;

import com.seleniumbootcampframework.core.Browser;

import com.seleniumbootcampframework.core.Log;

import com.seleniumbootcampframework.dataobjects.TestData;

import com.seleniumbootcampframework.pageobjects.MainPage;

import com.seleniumbootcampframework.pageobjects.SignInPage;

public class SignInTest extends Browser {

@Test

public void signInWithUnregisteredUser() throws Exception

{

// Use prebuilt methods from utility class log to add test descriptions

Log.setStoryName("Sign In Functionality");

Log.setTestScriptName("Unregistered user sign in");

Log.setTestScriptDescription("Unsuccessful sign in using unregisterd account");

//Navigate to Sign in page

MainPage.Header.clickSignInLink();

// Sign in using unregistered user

SignInPage.AlreadyRegistered.signIn(TestData.Credentials.UnregisteredUser.Username(), TestData.Credentials.UnregisteredUser.Password());

// Verify failed authentication message

SignInPage.ErrorMessage.verifyFailedAuthentication();

}

}

MainPage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Link;

public class MainPage {

public static class Header {

private static Link \_linkContactNo = new Link("Contact Us", By.xpath("//a[@title='Contact Us']"));

private static Link \_linkSignIn = new Link("Sign in", By.xpath("//a[@class='login']"));

public static void clickContactUsLink() {

\_linkContactNo.click();

}

public static void clickSignInLink() {

\_linkSignIn.click();

}

public static void verifyLoginUser(String userName) {

Link \_linkAccount = new Link("Account: " + userName,

By.xpath("//a[@title='View my customer account']/span[text()='" + userName + "']"));

\_linkAccount.verifyDisplayed();

}

}

}

SignInPage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.Element;

import com.seleniumbootcampframework.webelements.Link;

import com.seleniumbootcampframework.webelements.TextBox;

public class SignInPage {

public static class AlreadyRegistered {

private static Button \_buttonSignIn = new Button("Sign In", By.xpath("//button[@id='SubmitLogin']"));

private static Link \_linkForgotYourPassword = new Link("Forgot your password",

By.xpath("//a[@title='Recover your forgotten password']"));

private static TextBox \_textBoxEmailAddress = new TextBox("Email Address", By.xpath("//input[@id='email']"));

private static TextBox \_textBoxPassword = new TextBox("Password", By.xpath("//input[@id='passwd']"));

public static void clickForgotYourPasswordLink() {

\_linkForgotYourPassword.click();

}

public static void signIn(String emailAddress, String password) {

\_textBoxEmailAddress.setText(emailAddress);

\_textBoxPassword.setText(password);

\_buttonSignIn.click();

}

}

public static class CreateAnAccount {

private static Button \_buttonCreateAnAccount = new Button("Create an account",

By.xpath("//button[@id='SubmitCreate']"));

private static TextBox \_textBoxEmailAddress = new TextBox("Email Address",

By.xpath("//input[@id='email\_create']"));

public static void createAnAccount(String emailAddress) {

\_textBoxEmailAddress.setText(emailAddress);

\_buttonCreateAnAccount.click();

}

}

public static class ErrorMessage {

public static void verifyFailedAuthentication() {

Element \_elementFailedAuthenticationMessag = new Element("Authentication failed message",

By.xpath("//li[text()='Authentication failed.']"));

\_elementFailedAuthenticationMessag.verifyDisplayed();

}

}

}

TestData.java dataobject

package com.seleniumbootcampframework.dataobjects;

import java.io.IOException;

import com.seleniumbootcampframework.core.DataTable;

import jxl.read.biff.BiffException;

public class TestData {

public static class AccountInformation {

public static class RegisteredUser {

public static String FullName() throws BiffException, IOException {

return DataTable.getCellValue("Fullname", 3);

}

}

}

public static class Credentials {

public static class RegisteredUser {

public static String Password() throws BiffException, IOException {

return DataTable.getCellValue("Password", 3);

}

public static String Username() throws BiffException, IOException {

return DataTable.getCellValue("Username", 3);

}

}

public static class UnregisteredUser {

public static String Password() throws BiffException, IOException {

return DataTable.getCellValue("Password", 6);

}

public static String Username() throws BiffException, IOException {

return DataTable.getCellValue("Username", 6);

}

}

}

public static class Urls {

public static String myStoreUrl() throws BiffException, IOException {

return DataTable.getCellValue("URL", 3);

}

}

}

Module 7 Activity

CreateNewAccountTest.java

package com.seleniumbootcampframework.tests;

import org.testng.annotations.Test;

import com.seleniumbootcampframework.core.Browser;

import com.seleniumbootcampframework.core.Log;

import com.seleniumbootcampframework.dataobjects.TestData;

import com.seleniumbootcampframework.pageobjects.HomePage;

import com.seleniumbootcampframework.pageobjects.RegisterPage;

import com.seleniumbootcampframework.pageobjects.SignInPage;

public class CreateNewAccountTest extends Browser {

@Test

public void createNewAccount() throws Exception

{

Log.setStoryName("Account Registration");

Log.setTestScriptName("Create new Account");

Log.setTestScriptDescription("Register and create a user account and Login in the application using the user account created");

//Navigate to URL

getDriver().get(TestData.Urls.demoToursUrl());

// Click Register link

HomePage.RegisterSection.clickRegisterLink();

// Fill-up Registration fields

RegisterPage.fillRegistrationFields.setFirstName(TestData.AccountInformation.UnregisteredUser.firstName());

RegisterPage.fillRegistrationFields.setLastName(TestData.AccountInformation.UnregisteredUser.lastName());

RegisterPage.fillRegistrationFields.setPhone(TestData.AccountInformation.UnregisteredUser.phone());

RegisterPage.fillRegistrationFields.setUsername(TestData.AccountInformation.UnregisteredUser.email());

RegisterPage.fillRegistrationFields.setAddress1(TestData.AccountInformation.UnregisteredUser.address1());

RegisterPage.fillRegistrationFields.setCity(TestData.AccountInformation.UnregisteredUser.city());

RegisterPage.fillRegistrationFields.setState(TestData.AccountInformation.UnregisteredUser.state());

RegisterPage.fillRegistrationFields.setPostalCode(TestData.AccountInformation.UnregisteredUser.postalCode());

RegisterPage.fillRegistrationFields.setCountry(TestData.AccountInformation.UnregisteredUser.country());

RegisterPage.fillRegistrationFields.setEmail(TestData.AccountInformation.UnregisteredUser.username());

RegisterPage.fillRegistrationFields.setPassword(TestData.AccountInformation.UnregisteredUser.password());

RegisterPage.fillRegistrationFields.setConfirmPassword(TestData.AccountInformation.UnregisteredUser.confirmPassword());

RegisterPage.fillRegistrationFields.clickSubmit();

// Click sign-in link in Registration confirmation message

SignInPage.signInLink.clickSignIn();

// Fill-up credentials and click login

SignInPage.Login.setUsername(TestData.AccountInformation.UnregisteredUser.username());

SignInPage.Login.setPassword(TestData.AccountInformation.UnregisteredUser.password());

SignInPage.Login.clickSubmit();

}

}

HomePage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.Link;

import com.seleniumbootcampframework.webelements.TextBox;

public class HomePage {

public static class RegisterSection

{

private static Link \_linkRegister = new Link("REGISTER", By.linkText("REGISTER"));

public static void clickRegisterLink()

{

\_linkRegister.click();

}

}

public static class SignInSection {

private static TextBox \_textBoxUsername = new TextBox("User Name", By.xpath("//input[@name='userName']"));

private static TextBox \_textBoxPassword = new TextBox("Password", By.xpath("//input[@name='password']"));

private static Button \_buttonSignIn = new Button("Sign In", By.xpath("//input[@name='login']"));

public static void signIn(String emailAddress, String password) {

\_textBoxUsername.setText(emailAddress);

\_textBoxPassword.setText(password);

\_buttonSignIn.click();

}

}

//Create Page Object and page objects method here..

}

RegisterPage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.ListBox;

import com.seleniumbootcampframework.webelements.ListBox;

import com.seleniumbootcampframework.webelements.TextBox;

public class RegisterPage

{

public static class fillRegistrationFields

{

private static TextBox \_txtBoxFirstName = new TextBox("First Name", By.xpath("//input[@name='firstName']"));

private static TextBox \_txtBoxLastName = new TextBox("Last Name", By.xpath("//input[@name='lastName']"));

private static TextBox \_txtBoxPhone = new TextBox("Phone", By.xpath("//input[@name='phone']"));

private static TextBox \_txtBoxUsername = new TextBox("Email", By.xpath("//input[@name='userName']"));

private static TextBox \_txtBoxAddress1 = new TextBox("Address", By.xpath("//input[@name='address1']"));

private static TextBox \_txtBoxCity = new TextBox("City", By.xpath("//input[@name='city']"));

private static TextBox \_txtBoxState = new TextBox("State/Province", By.xpath("//input[@name='state']"));

private static TextBox \_txtBoxPostalCode = new TextBox("Postal Code", By.xpath("//input[@name='postalCode']"));

private static ListBox \_lstBoxCountry = new ListBox("Country", By.xpath("//select[@name='country']"));

private static TextBox \_txtBoxEmail = new TextBox("User Name", By.xpath("//input[@name='email']"));

private static TextBox \_txtBoxPassword = new TextBox("Password", By.xpath("//input[@name='password']"));

private static TextBox \_txtBoxConfirmPassword = new TextBox("Confirm Password", By.xpath("//input[@name='confirmPassword']"));

private static Button \_btnSubmit = new Button("SUBMIT", By.xpath("//input[@name='register']"));

public static void setFirstName(String firstname)

{

\_txtBoxFirstName.setText(firstname);

}

public static void setLastName(String lastname)

{

\_txtBoxLastName.setText(lastname);

}

public static void setPhone(String phone)

{

\_txtBoxPhone.setText(phone);

}

public static void setUsername(String username)

{

\_txtBoxUsername.setText(username);

}

public static void setAddress1(String address1)

{

\_txtBoxAddress1.setText(address1);

}

public static void setCity(String city)

{

\_txtBoxCity.setText(city);

}

public static void setState(String state)

{

\_txtBoxState.setText(state);

}

public static void setPostalCode(String postalcode)

{

\_txtBoxPostalCode.setText(postalcode);

}

public static void setCountry(String country)

{

\_lstBoxCountry.selectByVisibleText(country);

}

public static void setEmail(String phone)

{

\_txtBoxEmail.setText(phone);

}

public static void setPassword(String password)

{

\_txtBoxPassword.setText(password);

}

public static void setConfirmPassword(String confirmpassword)

{

\_txtBoxConfirmPassword.setText(confirmpassword);

}

public static void clickSubmit()

{

\_btnSubmit.click();

}

}

}

SignInPage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.Link;

import com.seleniumbootcampframework.webelements.TextBox;

public class SignInPage

{

public static class signInLink

{

private static Link \_linkSignIn = new Link("sign-in", By.linkText("sign-in"));

public static void clickSignIn()

{

\_linkSignIn.click();

}

}

public static class Login

{

private static TextBox \_txtBoxUsername = new TextBox("User Name", By.xpath("//input[@name='userName']"));

private static TextBox \_txtBoxPassword = new TextBox("Password", By.xpath("//input[@name='password']"));

private static Button \_btnSubmit = new Button("SUBMIT", By.xpath("//input[@name='login']"));

public static void setUsername(String username)

{

\_txtBoxUsername.setText(username);

}

public static void setPassword(String password)

{

\_txtBoxPassword.setText(password);

}

public static void clickSubmit()

{

\_btnSubmit.click();

}

}

}

TestData.java dataobject

package com.seleniumbootcampframework.dataobjects;

import java.io.IOException;

import com.seleniumbootcampframework.core.DataTable;

import jxl.read.biff.BiffException;

public class TestData {

public static class Urls

{

public static String demoToursUrl() throws BiffException, IOException

{

return DataTable.getCellValue("URL", 1);

}

}

public static class AccountInformation

{

public static class RegisteredUser

{

public static String FullName() throws BiffException, IOException

{

return DataTable.getCellValue("Fullname", 1);

}

}

public static class UnregisteredUser

{

public static String firstName() throws BiffException, IOException

{

return DataTable.getCellValue("Firstname", 1);

}

public static String lastName() throws BiffException, IOException

{

return DataTable.getCellValue("Lastname", 1);

}

public static String phone() throws BiffException, IOException

{

return DataTable.getCellValue("Phone", 1);

}

public static String email() throws BiffException, IOException

{

return DataTable.getCellValue("Email", 1);

}

public static String address1() throws BiffException, IOException

{

return DataTable.getCellValue("Address1", 1);

}

public static String city() throws BiffException, IOException

{

return DataTable.getCellValue("City", 1);

}

public static String state() throws BiffException, IOException

{

return DataTable.getCellValue("State", 1);

}

public static String postalCode() throws BiffException, IOException

{

return DataTable.getCellValue("PostalCode", 1);

}

public static String country() throws BiffException, IOException

{

return DataTable.getCellValue("Country", 1);

}

public static String username() throws BiffException, IOException

{

return DataTable.getCellValue("Username", 1);

}

public static String password() throws BiffException, IOException

{

return DataTable.getCellValue("Password", 1);

}

public static String confirmPassword() throws BiffException, IOException

{

return DataTable.getCellValue("ConfirmPassword", 1);

}

}

}

public static class Credentials

{

public static class RegisteredUser

{

public static String Password() throws BiffException, IOException

{

return DataTable.getCellValue("Password", 1);

}

public static String Username() throws BiffException, IOException

{

return DataTable.getCellValue("Username", 1);

}

}

}

}

SELENIUM WEBDRIVER TEST DEBUGGING

LEARNINGOBJECTIVES

At the end of this module,

you will be able to:

–Identify the common issues encountered when Selenium test fails.

–Debug failed test and address the issues.

DEBUGGING

In software testing, debugging is the process of finding and resolving bugs or defects that prevent correct operation of computer software or a system.

In test automation, debugging is fixing an error in the test code and making the logic of the test is correct and functioning as expected.

COMMON DEBUGG INGISSUES

When a test failed we must investigate the cause of the issue why the script’s result is not as expected. There are many reason for a script’s failure, some of the most common issues encountered are:

Error: The test did not land to the expected page.

Cause: No internet connection or the incorrect link is being accessed.

Solution: Make sure that there is stable internet connection and the correct link/ page is being accessed.

Error: The locator/ web element is not found in the page.

Cause: Wrong locator is being used or the locator type do not match with the locator name. The locator is not displayed in the current page during execution.

Solution: Make sure that the locator name and the locator type is correct. The locator that is being used must be present in the current screen during the execution.

Error: Encountered this error message when running the test.

Cause: Selenium cannot find the test given in the testing.xml because the class path or the test name given is incorrect.

Solution: Make sure the class path and the test name is correct.

Error: Encountered this error message when running the test.

Cause: The URL used in the test is missing a protocol.

Solution: Instead of www.google.comit should be defined as http://www.google.com.

Error: Encountered this error message when running the test.

Cause: This is an assertion error, the expected value (text or locator) did not match the actual

Cause: This is an assertion error, the expected value (text or locator) did not match the actual value in the current screen.

Solution: Actual value of the expected value (text or locator) might already changed in the page. Value (text or locator) should be in the page should be the same in the script. The test should be in the right page when executing the validation.

Error: Encountered this error message when running the test.

Cause: As defined in the console error, you can identify that the Excel file is not found during execution of the test.

Solution: Modify the file path given in the test script and make sure the correct path is given

Error: Encountered this error message when running the test.

Cause: The Excel sheet name given is incorrect or cannot be found in the Excel file.

Solution: Modify the Excel sheet name given in the test script and make sure the correct path is given

Error: Encountered this message where there’s no test that has been executed.

Cause: There’s no defined test in the script.

Solution: Check the script and make sure there’s “@Test” annotation in the method name.

Error: Encountered this error message when running the test.

Cause: Fields/Column name declared is missing or is incorrect in the excel file.

Solution: Check the defined column name in the test script if it matches in the Excel sheet. Make sure the file has been saved before executing the test.

SUMMARY

•Debugging is fixing an error to make sure that the codes in the application and the test are working as expected.

•Debugging is a necessary process in almost any new software or hardware development process, whether a commercial product or an enterprise or personal application program.

SELENIUM WEBDRIVER BEST PRACTICES

LEARNINGOBJECTIVES

At the end of this module,

you will be able to:

–Learn the practices that should be kept when using the framework.

BEST PRACTICE

It is a method or technique that has consistently shown results superior to those achieved with other means, and that is used as a benchmark In addition, a " practice can evolve to become better as improvements are discovered

BENEFITS OF BESTPRACTICE

There are some advantages when implementing when automating scripts with selenium:

•Standard naming will boost readability and uniformity

•Easier version control

•Scripts will be easier to maintain

•Increase the productivity

BEST PRACTICES FOR SELENIUM

•Handle scripts and data separately

–Automated test scripts should be clearly separated from the input data store (e.g. XML, Ms-Excel files, Flat files or Databases), so that no modifications are required to the test scripts whenever data has to be changed for multiple input values.

•Create libraries

–A library should contain all reusable components and external connections such as databases, generic functions, application functions etc. Software testers should be exposed only to the implemented libraries and tests should be performed by invoking these libraries.

•Follow coding standards

–Scripting standards should always be maintained across the test automation framework, which will discourage individual coding practices and help in maintaining code uniformity, which makes it easier for software testers and developers to interpret.

•Extensibility and Maintenance

–An ideal test automation framework should steadily support all new enhancements to the software application and allow modification of existing features e.g. A reusable library can be created, which would help in enhancing application features with minimal effort.

•Script/Framework version control

–Versions of framework / scripts should be maintained either in a local repository or versioning tool, which would help in easy monitoring of changes to the software code.

•Locator to use

–Locator type that should be considered to be used when creating test is by ID, if not found use locators by Name. Xpathshould be considered last when used as locator.

•Test Script dependencies

–Test scripts should be dependent in the base codes such as page objects and base test. It should not be dependent to another test script it will lessen the flexibility and maintainability of the scripts.

•Naming Convention

–Naming convention for the methods, test scripts, test data and other files should follow the same standard to create uniformity and better readability of the scripts.

•Add comments in the scripts

–For better maintainability and version controlling of the codes, comments should be added which contains author, date and description. This helps the developer to back track the changes made in the codes.

SUMMARY

•Best Practices is a method to create a set of standards that can be followed to increase the efficiency and effectivity of the framework.

•Best Practices for Selenium

–Handle scripts and data separately

–Create libraries

–Follow coding standards

–Version control tool is needed

–Continuous Improvement

COURSE SUMMARY

•After completing this course, you should now be able to:

–Comprehend Selenium and the tools under it.

–Comprehend Selenium WebDriver

–Comprehend basic Java OOP and create classes and methods

–Execute basic Selenesecommands

–Install and setup Selenium framework

–Execute and debug test scripts

–Apply best practices when using the framework

SCHOOL RECAP

Through presentations, discussions, and activities, this course has covered the following topics:

•Course Introduction

•Introduction to Selenium

•Advantages using Selenium

•Basic Java OOP

•Using Selenium WebDriver

•Automation Frameworks

•How to setup Data Driven Framework

•Execute test scripts

•Selenium Best Practices

THE END OF THE ROAD

Do you have any final questions/ comments?

Next items that should be learned:

•Jenkins

•Tortoise SVN, GitHub or other versioning control application

•Creation of batch run file

Thanks for participating!

Final Exam Hybrid Test Framework

TricentisHome.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

// Inherit element classes

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.Element;

import com.seleniumbootcampframework.webelements.Link;

import com.seleniumbootcampframework.webelements.TextBox;

public class TricentisHome {

public static class navSection

{

private static Link \_linkAutomobile = new Link("Automobile", By.id("nav\_automobile"));

private static Link \_linkTruck = new Link("Truck", By.id("nav\_truck"));

private static Link \_linkMotorcycle= new Link("Motorcycle", By.id("nav\_motorcycle"));

private static Link \_linkCamper= new Link("Camper", By.id("nav\_camper"));

// Verify vehicle nav links

public static void verifyAutomobileNavLink()

{

\_linkAutomobile.verifyDisplayed();

}

public static void verifyTruckNavLink()

{

\_linkTruck.verifyDisplayed();

}

public static void verifyMotorcycleNavLink()

{

\_linkMotorcycle.verifyDisplayed();

}

public static void verifyCamperNavLink()

{

\_linkCamper.verifyDisplayed();

}

// Click Automobile nav link

public static void clickAutomobileNavLink()

{

\_linkAutomobile.click();

}

}

}

AutomobileInsurancePage.java pageobject

package com.seleniumbootcampframework.pageobjects;

import org.openqa.selenium.By;

// Inherit element classes

import com.seleniumbootcampframework.webelements.Button;

import com.seleniumbootcampframework.webelements.CheckBox;

import com.seleniumbootcampframework.webelements.Element;

import com.seleniumbootcampframework.webelements.Link;

import com.seleniumbootcampframework.webelements.ListBox;

import com.seleniumbootcampframework.webelements.RadioButton;

import com.seleniumbootcampframework.webelements.TextBox;

public class AutomobileInsurancePage {

public static class navSection

{

// Locate elements

private static Element \_lblAutoInsurance = new Element("Automobile Insurance", By.id("selectedinsurance"));

private static Link \_linkEnterVehicleData = new Link("Enter Vehicle Data", By.id("entervehicledata"));

private static Link \_linkEnterInsurantData = new Link("Enter Insurant Data", By.id("enterinsurantdata"));

private static Link \_linkEnterProductData = new Link("Enter Product Data", By.id("enterproductdata"));

private static Link \_linkSelectPriceOption = new Link("Select Price Option", By.id("selectpriceoption"));

private static Link \_linkSendQuote = new Link("Send Quote", By.id("sendquote"));

// Verify Automobile Insurance nav links with element method

public static void vrfyAutoInsuranceLbl()

{

\_lblAutoInsurance.verifyDisplayed();

}

public static void vrfyVehicleDataNavLink()

{

\_linkEnterVehicleData.verifyDisplayed();

}

public static void vrfyInsurantDataNavLink()

{

\_linkEnterInsurantData.verifyDisplayed();

}

public static void vrfyProductDataNavLink()

{

\_linkEnterProductData.verifyDisplayed();

}

public static void vrfyPriceOptionNavLink()

{

\_linkSelectPriceOption.verifyDisplayed();

}

public static void vrfySendQuoteNavLink()

{

\_linkSendQuote.verifyDisplayed();

}

}

public static class vehicleDataSection

{

// Locate elements

private static ListBox \_lstBoxMake = new ListBox("Make", By.xpath("//select[@id='make']"));

private static TextBox \_txtBoxEnginePerformance = new TextBox("Engine Performance", By.xpath("//input[@id='engineperformance']"));

private static TextBox \_txtBoxDateOfManufacture = new TextBox("Date of Manufacture", By.xpath("//input[@id='dateofmanufacture']"));

private static ListBox \_lstSeatNums = new ListBox("Number of Seats", By.xpath("//select[@id='numberofseats']"));

private static ListBox \_lstFuelType = new ListBox("Fuel Type", By.xpath("//select[@id='fuel']"));

private static TextBox \_txtBoxListPrice = new TextBox("List Price", By.xpath("//input[@id='listprice']"));

private static TextBox \_txtBoxLicensePlateNumber = new TextBox("Licencse Plate Number", By.xpath("//input[@id='licenseplatenumber']"));

private static TextBox \_txtBoxAnnualMileage = new TextBox("Annual Mileage", By.xpath("//input[@id='annualmileage']"));

private static Button \_btnNext = new Button("Next", By.xpath("//button[@id='nextenterinsurantdata']"));

// Process data with the elements using a method of the elements

public static void selMake(String make)

{

\_lstBoxMake.selectByVisibleText(make);

}

public static void setEnginePerformance(String engineperf)

{

\_txtBoxEnginePerformance.setText(engineperf);

}

public static void setDateOfManufacture(String manufacturedate)

{

\_txtBoxDateOfManufacture.setText(manufacturedate);

}

public static void selNumOfSeats(String seatnums)

{

\_lstSeatNums.selectByVisibleText(seatnums);

}

public static void selFuelType(String fueltype)

{

\_lstFuelType.selectByVisibleText(fueltype);

}

public static void setListPrice(String listprice)

{

\_txtBoxListPrice.setText(listprice);

}

public static void setLicensePlateNumber(String licenseplate)

{

\_txtBoxLicensePlateNumber.setText(licenseplate);

}

public static void setAnnualMileage(String annualmileage)

{

\_txtBoxAnnualMileage.setText(annualmileage);

}

public static void clickNext()

{

\_btnNext.click();

}

}

public static class insurantDataSection

{

// Locate elements

private static TextBox \_txtBoxFirstName = new TextBox("First Name", By.xpath("//input[@id='firstname']"));

private static TextBox \_txtBoxLastName = new TextBox("Last Name", By.xpath("//input[@id='lastname']"));

private static TextBox \_txtBoxDateOfBirth = new TextBox("Date of Birth", By.xpath("//input[@id='birthdate']"));

private static RadioButton \_rdoGender = new RadioButton("Gender", By.xpath("(//label[@class=\"ideal-radiocheck-label\"])[1]"));

private static TextBox \_txtBoxStreetAddress = new TextBox("Street Address", By.xpath("//input[@id='streetaddress']"));

private static ListBox \_lstCountry = new ListBox("Country", By.xpath("//select[@id='country']"));

private static TextBox \_txtBoxZipCode = new TextBox("Zip Code", By.xpath("//input[@id='zipcode']"));

private static TextBox \_txtBoxCity = new TextBox("City", By.xpath("//input[@id='city']"));

private static ListBox \_lstOccupation = new ListBox("Occupation", By.xpath("//select[@id='occupation']"));

private static CheckBox \_chkHobbySpeeding = new CheckBox("Hobbies", By.xpath("(//span[@class=\"ideal-check\"])[1]"));

private static CheckBox \_chkHobbySkydiving = new CheckBox("Hobbies", By.xpath("(//span[@class=\"ideal-check\"])[3]"));

private static Button \_btnNextProduct = new Button("Next", By.xpath("//button[@id='nextenterproductdata']"));

// Process data with the elements using a method of the elements

public static void setFirstName(String firstname)

{

\_txtBoxFirstName.setText(firstname);

}

public static void setLastName(String lastname)

{

\_txtBoxLastName.setText(lastname);

}

public static void setDateOfBirth(String dob)

{

\_txtBoxDateOfBirth.setText(dob);

}

public static void selGender()

{

\_rdoGender.click();

}

public static void setStreetAddress(String streetaddress)

{

\_txtBoxStreetAddress.setText(streetaddress);

}

public static void selCountry(String country)

{

\_lstCountry.selectByVisibleText(country);

}

public static void setZipCode(String zipcode)

{

\_txtBoxZipCode.setText(zipcode);

}

public static void setCity(String city)

{

\_txtBoxCity.setText(city);

}

public static void selOccupation(String occupation)

{

\_lstOccupation.selectByVisibleText(occupation);

}

public static void selHobbySpeeding()

{

\_chkHobbySpeeding.click();

}

public static void selHobbySkydiving()

{

\_chkHobbySkydiving.click();

}

public static void clickNextProductBtn()

{

\_btnNextProduct.click();

}

}

public static class productDataSection

{

// Locate elements

private static TextBox \_txtBoxStartDate = new TextBox("Start Date", By.xpath("//input[@id='startdate']"));

private static ListBox \_lstInsuranceSum = new ListBox("Insurance Sum", By.xpath("//select[@id='insurancesum']"));

private static ListBox \_lstMeritRating = new ListBox("Merit Rating", By.xpath("//select[@id='meritrating']"));

private static ListBox \_lstDamageInsurance = new ListBox("Damage Insurance", By.xpath("//select[@id='damageinsurance']"));

private static CheckBox \_chkOptionalProducts = new CheckBox("Optional Products", By.xpath("(//span[@class=\"ideal-check\"])[5]"));

private static ListBox \_lstCourtesyCar = new ListBox("Courtesy Car", By.xpath("//select[@id='courtesycar']"));

private static Button \_btnNextPrice = new Button("Next", By.xpath("//button[@id='nextselectpriceoption']"));

// Process data with the elements using a method of the elements

public static void setStartDate(String startdate)

{

\_txtBoxStartDate.setText(startdate);

}

public static void selInsuranceSum(String insurancesum)

{

\_lstInsuranceSum.selectByVisibleText(insurancesum);

}

public static void selMeritRating(String rating)

{

\_lstMeritRating.selectByVisibleText(rating);

}

public static void selDamageInsurance(String damageinsurance)

{

\_lstDamageInsurance.selectByVisibleText(damageinsurance);

}

public static void selOptionalProducts()

{

\_chkOptionalProducts.click();

}

public static void selCourtesyCar(String courtesycar)

{

\_lstCourtesyCar.selectByVisibleText(courtesycar);

}

public static void clickNextPriceBtn()

{

\_btnNextPrice.click();

}

}

public static class priceSection

{

// Locate elements

private static RadioButton \_rdoUltimate = new RadioButton("Ultimate", By.xpath("(//label[@class=\"choosePrice ideal-radiocheck-label\"])[4]"));

private static Button \_btnNextQuote = new Button("Next", By.xpath("//button[@id='nextsendquote']"));

// Process data with the elements using a method of the elements

public static void selUltimate()

{

\_rdoUltimate.click();

}

public static void clickNextQuoteBtn()

{

\_btnNextQuote.click();

}

}

public static class sendQuoteSection

{

// Locate elements

private static TextBox \_txtBoxEmail = new TextBox("E-mail", By.xpath("//input[@id='email']"));

private static TextBox \_txtBoxUsername = new TextBox("Username", By.xpath("//input[@id='username']"));

private static TextBox \_txtBoxPassword = new TextBox("Password", By.xpath("//input[@id='password']"));

private static TextBox \_txtBoxCofirmPassword = new TextBox("Confirm Password", By.xpath("//input[@id='confirmpassword']"));

private static Button \_btnSend = new Button("Next", By.xpath("//button[@id='sendemail']"));

// Process data with the elements using a method of the elements

public static void setEmail(String email)

{

\_txtBoxEmail.setText(email);

}

public static void setUsername(String username)

{

\_txtBoxUsername.setText(username);

}

public static void setPassword(String password)

{

\_txtBoxPassword.setText(password);

}

public static void setConfirmPassword(String confirmpassword)

{

\_txtBoxCofirmPassword.setText(confirmpassword);

}

public static void clickSendBtn()

{

\_btnSend.click();

}

}

public static class emailSuccess

{

// Locate elements

private static Element \_alrtEmailSuccess = new Element("Sending e-mail success", By.tagName("h2"));

private static Button \_btnYes = new Button("Next", By.xpath("//button[@class='confirm']"));

// Process data with the elements using a method of the elements

public static void vrfyEmailSuccessLbl()

{

\_alrtEmailSuccess.verifyDisplayed();

}

public static void clickYes()

{

\_btnYes.click();

}

}

}

TestDataExam.java dataobject

package com.seleniumbootcampframework.dataobjects;

import java.io.IOException;

import com.seleniumbootcampframework.core.DataTable;

import jxl.read.biff.BiffException;

public class TestDataExam {

public static class Urls

{

public static String tricentisUrl() throws BiffException, IOException

{

return DataTable.getCellValue("URL", 1);

}

}

public static class PurchaseInfo

{

public static class VehicleDataInfo

{

public static String Make() throws BiffException, IOException

{

return DataTable.getCellValue("Make", 1);

}

public static String EnginePerformance() throws BiffException, IOException

{

return DataTable.getCellValue("EnginePerformance", 1);

}

public static String ManufactureDate() throws BiffException, IOException

{

return DataTable.getCellValue("ManufactureDate", 1);

}

public static String SeatNumbers() throws BiffException, IOException

{

return DataTable.getCellValue("SeatNumbers", 1);

}

public static String FuelType() throws BiffException, IOException

{

return DataTable.getCellValue("FuelType", 1);

}

public static String ListPrice() throws BiffException, IOException

{

return DataTable.getCellValue("ListPrice", 1);

}

public static String LicensePlateNumber() throws BiffException, IOException

{

return DataTable.getCellValue("LicensePlateNumber", 1);

}

public static String AnnualMileage() throws BiffException, IOException

{

return DataTable.getCellValue("AnnualMileage", 1);

}

}

public static class InsurantDataInfo

{

public static String FirstName() throws BiffException, IOException

{

return DataTable.getCellValue("FirstName", 1);

}

public static String LastName() throws BiffException, IOException

{

return DataTable.getCellValue("LastName", 1);

}

public static String DateOfBirth() throws BiffException, IOException

{

return DataTable.getCellValue("DateOfBirth", 1);

}

public static String StreetAddress() throws BiffException, IOException

{

return DataTable.getCellValue("StreetAddress", 1);

}

public static String Country() throws BiffException, IOException

{

return DataTable.getCellValue("Country", 1);

}

public static String ZipCode() throws BiffException, IOException

{

return DataTable.getCellValue("ZipCode", 1);

}

public static String City() throws BiffException, IOException

{

return DataTable.getCellValue("City", 1);

}

public static String Occupation() throws BiffException, IOException

{

return DataTable.getCellValue("Occupation", 1);

}

}

public static class ProductDataInfo

{

public static String StartDate() throws BiffException, IOException

{

return DataTable.getCellValue("StartDate", 1);

}

public static String InsuranceSum() throws BiffException, IOException

{

return DataTable.getCellValue("InsuranceSum", 1);

}

public static String MeritRating() throws BiffException, IOException

{

return DataTable.getCellValue("MeritRating", 1);

}

public static String DamageInsurance() throws BiffException, IOException

{

return DataTable.getCellValue("DamageInsurance", 1);

}

public static String CourtesyCar() throws BiffException, IOException

{

return DataTable.getCellValue("CourtesyCar", 1);

}

}

public static class SendQuoteInfo

{

public static String Email() throws BiffException, IOException

{

return DataTable.getCellValue("Email", 1);

}

public static String Username() throws BiffException, IOException

{

return DataTable.getCellValue("Username", 1);

}

public static String Password() throws BiffException, IOException

{

return DataTable.getCellValue("Password", 1);

}

public static String ConfirmPassword() throws BiffException, IOException

{

return DataTable.getCellValue("ConfirmPassword", 1);

}

public static String EmailSuccess() throws BiffException, IOException

{

return DataTable.getCellValue("EmailSuccess", 1);

}

}

}

}

SuccesfulPurchaseOfItems.java tests script

package com.seleniumbootcampframework.tests;

import org.testng.annotations.Test;

import com.seleniumbootcampframework.core.Browser;

import com.seleniumbootcampframework.core.Log;

import com.seleniumbootcampframework.dataobjects.TestDataExam;

import com.seleniumbootcampframework.pageobjects.TricentisHome;

import com.seleniumbootcampframework.pageobjects.AutomobileInsurancePage;

// Inherit Browser.java class

public class SuccesfulPurchaseOfItems extends Browser

{

@Test

public void SuccessfulPurchaseOfItems() throws Exception

{

Log.setStoryName("Purchase Vehicle");

// 1. Navigate to URL using Browser class driver

getDriver().get(TestDataExam.Urls.tricentisUrl());

// Run Test scripts by calling base page element methods and TestDataExam base test values

// 2. Verify vehicle navigation links

TricentisHome.navSection.verifyAutomobileNavLink();

TricentisHome.navSection.verifyTruckNavLink();

TricentisHome.navSection.verifyMotorcycleNavLink();

TricentisHome.navSection.verifyCamperNavLink();

// 3. Click the Automobile hyperlink

TricentisHome.navSection.clickAutomobileNavLink();

// 4. Verify Automobile Insurance nav links

AutomobileInsurancePage.navSection.vrfyAutoInsuranceLbl();

AutomobileInsurancePage.navSection.vrfyVehicleDataNavLink();

AutomobileInsurancePage.navSection.vrfyInsurantDataNavLink();

AutomobileInsurancePage.navSection.vrfyProductDataNavLink();

AutomobileInsurancePage.navSection.vrfyPriceOptionNavLink();

AutomobileInsurancePage.navSection.vrfySendQuoteNavLink();

// 5. Populate Vehicle Data fields

AutomobileInsurancePage.vehicleDataSection.selMake(TestDataExam.PurchaseInfo.VehicleDataInfo.Make());

AutomobileInsurancePage.vehicleDataSection.setEnginePerformance(TestDataExam.PurchaseInfo.VehicleDataInfo.EnginePerformance());

AutomobileInsurancePage.vehicleDataSection.setDateOfManufacture(TestDataExam.PurchaseInfo.VehicleDataInfo.ManufactureDate());

AutomobileInsurancePage.vehicleDataSection.selNumOfSeats(TestDataExam.PurchaseInfo.VehicleDataInfo.SeatNumbers());

AutomobileInsurancePage.vehicleDataSection.selFuelType(TestDataExam.PurchaseInfo.VehicleDataInfo.FuelType());

AutomobileInsurancePage.vehicleDataSection.setListPrice(TestDataExam.PurchaseInfo.VehicleDataInfo.ListPrice());

AutomobileInsurancePage.vehicleDataSection.setLicensePlateNumber(TestDataExam.PurchaseInfo.VehicleDataInfo.LicensePlateNumber());

AutomobileInsurancePage.vehicleDataSection.setAnnualMileage(TestDataExam.PurchaseInfo.VehicleDataInfo.AnnualMileage());

// 6. Click Next button

AutomobileInsurancePage.vehicleDataSection.clickNext();

// 7. Populate Insurant Data fields

AutomobileInsurancePage.insurantDataSection.setFirstName(TestDataExam.PurchaseInfo.InsurantDataInfo.FirstName());

AutomobileInsurancePage.insurantDataSection.setLastName(TestDataExam.PurchaseInfo.InsurantDataInfo.LastName());

AutomobileInsurancePage.insurantDataSection.setDateOfBirth(TestDataExam.PurchaseInfo.InsurantDataInfo.DateOfBirth());

AutomobileInsurancePage.insurantDataSection.selGender();

AutomobileInsurancePage.insurantDataSection.setStreetAddress(TestDataExam.PurchaseInfo.InsurantDataInfo.StreetAddress());

AutomobileInsurancePage.insurantDataSection.selCountry(TestDataExam.PurchaseInfo.InsurantDataInfo.Country());

AutomobileInsurancePage.insurantDataSection.setZipCode(TestDataExam.PurchaseInfo.InsurantDataInfo.ZipCode());

AutomobileInsurancePage.insurantDataSection.setCity(TestDataExam.PurchaseInfo.InsurantDataInfo.City());

AutomobileInsurancePage.insurantDataSection.selOccupation(TestDataExam.PurchaseInfo.InsurantDataInfo.Occupation());

AutomobileInsurancePage.insurantDataSection.selHobbySpeeding();

AutomobileInsurancePage.insurantDataSection.selHobbySkydiving();

// 8. Click Next button

AutomobileInsurancePage.insurantDataSection.clickNextProductBtn();

// 9. Populate Product Data Fields

AutomobileInsurancePage.productDataSection.setStartDate(TestDataExam.PurchaseInfo.ProductDataInfo.StartDate());

AutomobileInsurancePage.productDataSection.selInsuranceSum(TestDataExam.PurchaseInfo.ProductDataInfo.InsuranceSum());

AutomobileInsurancePage.productDataSection.selMeritRating(TestDataExam.PurchaseInfo.ProductDataInfo.MeritRating());

AutomobileInsurancePage.productDataSection.selDamageInsurance(TestDataExam.PurchaseInfo.ProductDataInfo.DamageInsurance());

AutomobileInsurancePage.productDataSection.selOptionalProducts();

AutomobileInsurancePage.productDataSection.selCourtesyCar(TestDataExam.PurchaseInfo.ProductDataInfo.CourtesyCar());

// 10. Click Next button

AutomobileInsurancePage.productDataSection.clickNextPriceBtn();

// 11. Select Ultimate Price radio

AutomobileInsurancePage.priceSection.selUltimate();

// 12. Click Next button

AutomobileInsurancePage.priceSection.clickNextQuoteBtn();

// 13. Populate Send Quote data fields

AutomobileInsurancePage.sendQuoteSection.setEmail(TestDataExam.PurchaseInfo.SendQuoteInfo.Email());

AutomobileInsurancePage.sendQuoteSection.setUsername(TestDataExam.PurchaseInfo.SendQuoteInfo.Username());

AutomobileInsurancePage.sendQuoteSection.setPassword(TestDataExam.PurchaseInfo.SendQuoteInfo.Password());

AutomobileInsurancePage.sendQuoteSection.setConfirmPassword(TestDataExam.PurchaseInfo.SendQuoteInfo.ConfirmPassword());

// 14. Click Send button

AutomobileInsurancePage.sendQuoteSection.clickSendBtn();

// 15. Verify send email success

AutomobileInsurancePage.emailSuccess.vrfyEmailSuccessLbl();

// 16. Click Yes button

AutomobileInsurancePage.emailSuccess.clickYes();

}

}

• pom.xml

• TestSuite/testng.xml

• Core objects

o Browser.java

o DataTable.java

o Log.java

o OSChecker.java

o Screenshot.java

• WebElements objects

o Alerts.java

o Button.java

o CheckBox.java

o Element.java

o Frame.java

o Link.java

o ListBox.java

o ListElement.java

o NesteElement.java

o RadioButton.java

o Tab.java

o TextBox.java

o TextBox.java