**Xpath Siblings:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class XpathSiblings {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("http://www.qaclickacademy.com/interview.php");

driver.findElement(By.xpath("//li[text()=' Selenium ']")).click();

driver.findElement(By.xpath("//ul[@class='responsive-tabs\_\_list']/li[1]/following-sibling::li[2]")).click();

System.out.println(driver.findElement(By.xpath(".//\*[@id='tablist1-tab2']/parent::ul")).getAttribute("role"));

//

}

}

**Selenium Methods:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumMethods {

public static void main(String args[]) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

//Initialize browser

WebDriver driver =new ChromeDriver();

//open facebook

driver.get("https://www.facebook.com/");

//Maximize browser

driver.manage().window().maximize();

driver.findElement(By.id("email")).sendKeys("t.anusha2712@gmail.com");;

driver.findElement(By.className("inputtext")).sendKeys("namish");

driver.findElement(By.xpath(".//\*[@id='u\_0\_r']")).click();

//driver.findElement(By.name(".//\*[@id='u\_0\_r']")).click();

//driver.findElement(By.linkText("")).click();

driver.close();

}

}

RadioButtons:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class RadioButtons {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("http://www.echoecho.com/htmlforms10.htm");

driver.findElement(By.xpath("//input[@value='Butter']")).click();

//Handling radio buttons dynamically. We need to iterate through the values and specify index in the getindex() method

int count = driver.findElements(By.xpath("//input[@name='group1']")).size();

for(int i=0;i<count;i++)

{

//driver.findElements(By.xpath("//input[@name='group1']")).get(i).click();

String text = driver.findElements(By.xpath("//input[@name='group1']")).get(i).getAttribute("value");

if(text.equals("Cheese"))

{

driver.findElements(By.xpath("//input[@name='group1']")).get(i).click();

}

}

driver.close();

}

}

Page Object Framework:

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

// Import package pageObject.\*

import PageObjects.Home;

import PageObjects.Login;

public class POM {

private static WebDriver driver = null;

public static void main(String[] args) {

driver = new FirefoxDriver();

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

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

// Use page Object library now

Login.txtbx\_UserName(driver).sendKeys("testuser\_1");

Login.txtbx\_Password(driver).sendKeys("Test@123");

Login.btn\_LogIn(driver).click();

System.out.println(" Login Successfully, now it is the time to Log Off buddy.");

driver.quit();

}

}

Java Alerts:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.remote.server.handler.SendKeys;

public class JavaAlerts {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("www.tizag.com");

driver.findElement(By.xpath("html/body/table[3]/tbody/tr[1]/td[2]/table/tbody/tr/td/div[4]/form/input")).click();

//driver.switchTo().alert().sendKeys(""); In case if we want to write something in the edit box

driver.switchTo().alert().getText();//If we want to display the text in the alert box

driver.switchTo().alert().accept();//Any positive action

driver.switchTo().alert().dismiss();//Negative action

}

}

OOPS Concepts:

**public** **class** Inheritance {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

// Derived Class Object

ClassB obj\_b=**new** ClassB();

obj\_b.Display\_A("Bhavana");

obj\_b.Display\_B("Harish");

}

}

Form Methods:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class FormMethods {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("https://www.makemytrip.com/");

System.out.println("Before clicking on multi-city button");

System.out.println(driver.findElement(By.xpath(".//\*[@id='hp-widget\_\_return']")).isDisplayed());

driver.findElement(By.xpath(".//\*[@id='js-switch\_\_option']/div[3]/label")).click();

System.out.println("After clicking on multi-city button");

System.out.println(driver.findElement(By.xpath(".//\*[@id='hp-widget\_\_return']")).isDisplayed());

// If you want to validate the object which is present in web page or code base

int count = driver.findElements(By.xpath(".//\*[@id='js-hp-widgetSection']/div[1]")).size();

if(count==0)

{

System.out.println("Verified");

}

driver.close();

}

}

Dropdown:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

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

public class Dropdown {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "C:\\Users\\Anusha\\Downloads\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("www.spicejet.com");

Select s = new Select(driver.findElement(By.id("ctl00\_mainContent\_ddl\_Adult")));

s.selectByIndex(0);

s.selectByValue("2");

s.selectByVisibleText("3 Adults");

//driver.findElement(By.id("ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

/\*driver.findElement(By.xpath(".//a[@value='GOI']")).click();

// Need to indicate the instance

driver.findElement(By.cssSelector("(a[value='DEL'])[2]")).click();\*/

//validate whether checkbox is selected by default

System.out.println(driver.findElement(By.id("ctl00\_mainContent\_chk\_IndArm")).isSelected());

driver.findElement(By.id("ctl00\_mainContent\_chk\_IndArm")).click();

}

}

Page Objects:

package PageObjects;

import org.openqa.selenium.\*;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public class Login{

private static WebElement element = null;

public static WebElement txtbx\_UserName(WebDriver driver){

element = driver.findElement(By.id("log"));

return element;

}

public static WebElement txtbx\_Password(WebDriver driver){

element = driver.findElement(By.id("pwd"));

return element;

}

public static WebElement btn\_LogIn(WebDriver driver){

element = driver.findElement(By.id("login"));

return element;

}

TestNG:

package testNG;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.BeforeSuite;

public class Annot2 {

@BeforeSuite

public void Install()

{

System.out.println("I am the first");

}

@AfterSuite

public void Close()

{

System.out.println("I am the Last");

}

}

package testNG;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Annotations {

@BeforeMethod

public void EmailIdGen()

{

System.out.println("This block executes before each and every method");

}

@AfterMethod

public void Report()

{

System.out.println("This annotation executes after each and every method");

}

@BeforeTest

public void Cookie()

{

System.out.println("This executes before all test cases");

}

@AfterTest

public void CookieClose()

{

System.out.println("This executes after all test cases execution is done");

}

@Test

public void OpenBrowser()

{

System.out.println("@Test sample");

}

@Test

public void LogIn()

{

System.out.println("Email Login");

}

}

**package** testNG;

**import** org.testng.annotations.Test;

**public** **class** DependencyMethods {

@Test

**public** **void** OpenBrowser()

{

System.***out***.println("@Test sample");

}

@Test(dependsOnMethods={"OpenBrowser"},alwaysRun=**true**)//It executes even OpenBrowser fails

**public** **void** LogIn()

{

System.***out***.println("Email Login");

}

@Test(enabled=**false**)//If we dont want to run this test case

**public** **void** Payment()

{

System.***out***.println("New test case for disabling the test case to run");

}

@Test(timeOut=5000)

**public** **void** Card()

{

}

}

When we spy on the drop down list and if it shows select tag then it is a static drop down list.

**Web services:**

It is open standard based web application which interacts with other application in order to exchange data. Web services can convert an application into web application. Web service is any piece of code available itself in the internet and uses XML messaging system. If a client invokes a web service by sending an XML message, then he waits for a corresponding XML response. Since all the communication is XML based web services are bounded to any operating system or any language. Any two different languages can be communicated.

**Components of web services:**

The components that are used by web services are:

SOAP (Simple Object Access Protocol)

UDDI (Universal Description, Discovery and Integration)

WSDL (Web Services Description Language)

**Working of web services:**

Web services enables communication between application by using open standards such as XML, HTML, WSDL, SOAP. XML is used to tag the data, SOAP is to transfer the message, WSDL is used to describe the availability of service.

**WSDL:**

testing webservices:

how would you know what are the operations present in web service?

how would you decide what are the parameters you need to send and what are the datatypes that would be given to invoke an operation in an web service?

how many operations are present in web service, how many ways you can test it?

we need to have some document to explain this web service and this document is nothing but WSDL file. WSDL stands for web service description language.it is a standard format for describing a web service.

WSDL is often used in combination with SOAP and XML to provide web services over the internet.

**SOAP PROTOCOL:**

SOAP is an xml based protocol for exchanging information between webservices and its full form is simple object access protocol. it is a communication protocol. it provides data transfer for web services. soap enables client applications to easily connect to remote services and invoke remote methods.

SOAP message is an ordinary XML document consisting **of following elements:**

consider example of calculator performing add operation

soap message consists of soap request and soap response. In soap request we will be giving what operation to be done and what the parameters to be passed. when it reaches the calculator web service it interprets the soap request message which we have sent and it invokes the add operation successfully. That output is again wrapped up into a soap response message and sent it to the client side.

now, we will see what kind of structure or protocol to be followed while developing soap message?

**Envelope:(mandatory)**

SOAP Envelope indicates the start and end of the message so that receiver understands when the entire message has been received. the soap envelope solves the problem of when we are done with receiving the message and start the processing of it.

1. every soap message has a root envelope element

2. envelope element is mandatory part of soap message

<Envelope>

</Envelope>

**Header(optional):**

contains any optional attributes of the message used in processing the message,either at an intermediatery point or at the ultimate end point.

headers are intended to add new features and functionalites.

**Body(Mandatory):**

contains the XML data comprising the message being sent.

the SOAP body is a mandatory element which contails the application defined XML data being exchanged in the SOAP message. The body must be contained within the envelope and must follow any headers that might be defined for the message.

**Fault(optional):**

an optional fault element that provides information about errors that occurred while processing the message.

when an error occurs during processing, the response to a SOAP message is a SOAP fault element in the body of the message and the fault is returned to the sender of the soap message.

**instructions to configure web service in local machine:**

1. download the files from resources tab of this udemy lecture and unzip axis server file

2. set the axis server path in environment variables

3. download mysql installer file and set username and paswword to root and root, port to 3306

4. open workbench and execute the query(get the query from query.txt document)

**query**:create database Employeeportal;

use Employeeportal;

CREATE TABLE IF NOT EXISTS Employeeinfo(

name VARCHAR(100) NOT NULL,

id VARCHAR(200), dept varchar(20),

age INT NOT NULL,

PRIMARY KEY(name));

5. open axisserver.bat file

6. hit localhost:(portnum)

**BASIC ASSERTIONS:**

**ADD EMPLOYEE TESTCASE:**

verify response returning True- CONTAINS ASSERTION

verify response not having null or false value- NOT CONTAINS ASSERTION

verify response is valid soap response- SOAP RESPONSE ASSERTION

verify response have valid HTTP code- valid HTTP status code

verify that response time is with in boundaries

verify security threat

**GET EMPLOYEE DETAILS TESTCASE:**

verify ID of the employee is 545

verify department is computer science

check SoapUI response pattern

verify age attribute display

verify only unique name returns

**PROPERTIES IN SOAPUI:**

1. testcase level properties access: goto testcase-goto custom properties-add any value for example add age value and while using age in soap request we don't have to give value hardcoded. we have to right click--get data--testcase level--property(age). then it will displayed as ${#TestCase#age} in soap request.

2. testsuite level properties access:goto testcase-goto custom properties-add any value for example add age value and while using age in soap request we don't have to give value hardcoded. we have to right click--get data--testcase level--property(age). then it will displayed as ${#TestSuite#name} in soap request.

3. project level properties access-goto project level properties-goto custom properties-add any value for example add age value and while using age in soap request we don't have to give value hardcoded. we have to right click--get data--project level--property(id). then it will displayed as ${#Project#id} in soap request.

4.properties teststep-right click on teststep--properties--define some name and value example consider department. in soap request message it will appear as ${#Properties#department}

5. properties upload from external file- first open notepad give some key=value save the file with .properties extension. in soapui tool left side down under properties pan you can see the roll on option LOADS PROPERTY VALUES FROM AN EXTERNAL FILE. click on it and dump the file. you can also select adding missing values or deleting properties not in file. you can upload properties from external file in any level for example in testsuite level or in testcase level

6. property transfer teststep-

7. accessing custom properties-

**GROOVY SCRIPTING:**

it is a dynamic language for the java virtual machine

what is groovy scripting- it is a language which does not follow specific standard

goto soapUI--create new soapUI project. here no need of WSDL file because we are not testing we are writing scripts. In that project create new test suite--under that create new test case. groovy scripting is also called as

loosely coupled language

similar to JVM

printing logs

defining variable

operators

loosely coupled data types

if else simulator

for loop

for enhanced loop

while loop arrays

list

Q:how can we communicate with web services

A:by using soap messages

Performance Testing: HP Load Runner

Components of Load Runner

1. Load Generator

2. VuGen

3. Controller

Before Scenario

creates scenario

sets up run time configurations

During Scenario

Runs many users simultaneously

controls each vuser(Vuserint,action,pause,stop,vuserend)

Displays execution status of each user

Displays messages

Monitor server resources

After Scenario

collects and organizes performance data

launches the analysis tools.

4. Agent Process

5. Analysis

Load Testing tabs:

1. create/edit scripts

2. run load tests

3. analze load tests.

C, C++, Java, TSL are the scripting languages used for Load Runner. It also supports different protocols like Web UI, Silverlight, SAP etc.

VUScripts: The purpose of VUGen is to create VUScripts that are used to simulate a real-like virtual-user.

Actions: An action is set of user transactions performed in the System Under Load to achieve a defined task. An action can be compared to a function in other programming languages. Each Vuser script has 3 default functions

Vuser\_init(used to login into the application)

Action (used to record the business process)

Vuser\_end (used to logout of the application)

VUGen not only records scripts but also replays them (for only 1 VUser) to ensure script is recorded correctly. Once you ensure the script is recorded correctly you incorporate it in a LoadRunner scenario

Script Development process in VuGen:

1. Record the Script

2. Replay and verify

3. Enhance the script

4. Replay and verify

5. Configure Run time settings - browser, protocol selection

6. Use for load scenarios

Session id need to be correlated in order to run the script successfully. This is the exact which fails the scrit.

web\_reg\_save\_param("Parameter name","LB","RB") is an in built function used for parsing.

Automatic correlation steps:

1. Replay

2. select values to correlate

3. Veriy

click on show and resolve dynamic values, then select the user session value and click on correlate.

Manual correlation is suggested than automatic.

Parameterization:

1. It provides script to test at different values.

2. Reduces the script size.

Steps required for parameterization:

1. Replacing constant values in the vuser scripts with parameters.

2. Setting properties and data source for parameters so different values can be selected at run time.

Verification check points:

1. Image check

2. Text check

3. Output message.

web\_reg\_find

web\_image\_check

TestNG: Framework

It is used for unit testing like JUnit. It has more features than JUnit.

1. Batch of Test cases can be executed.

2. Can generate reports.

3. Data driven testing using TestNG-Data can be sent using TestNG which is in built.

4. TestNG can be integrated with diff tools like Jenkins, Maven.

Installation:

1. Goto Window

2. Click Preferences

3. Add link address in work with

4. click add

TestNgG:

@Test: If we want to execute a block of code then place it under @Test annotation.

Execution starts alphabetically in testNG.

@BeforeMethod: It gets executed before each and every test execution starts.

@BeforeTest: It can be executed only oncei.e before test case execution starts.It won't be treated as a separate test case.

It is treated as a pre request.

@AfterMethod: This method executes after each and every method.

@AfterTest:This method executes after all test cases execution.

@AfterSuite:

@BeforeSuite: It will check all the files and then if it finds any annotation with @BeforeSuite then it executes f

rst and then continues accordingly.

Testng.xml

We can invoke TestNG in several ways like with a testing .xml file, with ant and from the command line.

Assertions:

1. Soft assertions - it continues to execute even if there is a fail in between.

2. Hard assertions - execution stops if there is a failure