XML Parser:

●        A Parser is a program that takes a physical representation of some data and converts it into an in-memory form for the program as a whole to use. Parsers are used everywhere in software.

●        An XML Parser is a parser that is designed to read XML and create a way for programs to use XML.

There are mainly three types of XML parsers:

    1.7.1.1 SAX

    1.7.1.2 DOM

    1.7.1.3 Pull parser

**Step 1.7.1.1:** SAX

SAX stands for ‘Simple API for XML’. It does not create any internal structure. Clients do not know what methods to call. They just override the methods of the API and place his own code inside the method. It is an event-based parser, it works as an event handler in Java.

●  Advantages

●      Since it reads each unit of XML, it creates an event so that the calling program can use it.

●      SAX uses what it likes to, by ignoring the bits which it doesn’t care about.

●      It is memory efficient.

●      It’s very fast and works for huge documents.

●      Disadvantages

●      The main disadvantage of SAX is that the Calling program must keep track of everything it might ever need.

●      Since its Event-based, its API is less Intuitive.

DOM

DOM stands for ‘Document Object Model’. A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object. DOM Parser has a tree-based structure.

●  Advantages

●      It supports both Read and Write operations.

●      It is preferred when there is random access to widely separated parts of the documents required.

●       It builds the entire XML document representation in memory and then hands the calling program the whole chunk of memory.

●      Disadvantages

●      It consumes more memory since the whole XML document will be loaded into the memory.

Pull Parser

Pull parser waits for the application to come calling. That is, they ask for the next available event, and the application basically loops until it runs out of XML.

●      Advantages

●      It is designed to be used with large data sources.

●      Pull parser chooses to skip the events (whole section of the document) which it is not interested in.

This demo on reports in excel will not work as the jar file is deprecated.

But an older demo shows us how the report will be in excel sheet

Adding AT Excel report jars

●      Extent Reports jar file is already present in your practice lab in /home/ubuntu/libs directory.

●      Add the Extent Reports jar file to your project: Right-click on project->Build path->Configure build path->Add external Jars.

●      Click on Apply and then click OK.

**Step 1.5.3** Executing the test suites to see the generated report in Excel sheet

●      Write the test script in the Test\_01 class.

**package** Testcases;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

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

**import** base.Baseclass;

**public** **class** TEST\_01 **extends** Baseclass {

**@Test**

**public** void t\_001()

     {

       driver.findElement(By.xpath("(//\*[contains(text(),'Categories')])[1]")).click();

       driver.findElement(By.xpath("//\*[contains(text(),'Central')]")).click();

**System**.out.println("Test\_01 executed successfully");

     }

}

●      Write the test script in the Test\_02 class.

**package** Testcases;

**import** org.openqa.selenium.By;

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

**import** base.Baseclass;

**public** **class** TEST\_02 **extends** Baseclass {

**@Test**

**public** void t\_002() {

     driver.findElement(By.xpath("(//\*[contains(text(),'Popular')])[1]")).click();

**System**.out.println("Test\_02 executed successfully");

     }

}

●      Write the test script for the extended Base class, where all annotations are declared here.

**package** base;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.BeforeMethod;

**import** org.automationtesting.excelreport.Xl;

**public** **class** Baseclass {

**public** WebDriver driver;

**@BeforeMethod**

**public** void baseclass1()

     {

**System**.setProperty("webdriver.gecko.driver",  "/home/ubuntu/Downloads/gechodriver");

       driver = **new** geckodriver();

       driver.get("https://mvnrepository.com/");

     }

**@AfterMethod**

**public** void quitDriver() {

            driver.close();

     }

**@AfterSuite**

**public** void generateReport() **throws** **Exception** {

            Xl.generateReport("Report\_Excel.xlsx");

     }

}

●      Execute the test suite with multiple test cases and the testng.xml file will look like :

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="ExportReport">

     <test name="TEST1">

            <classes>

                    <**class** name="Testcases.TEST\_01"></class>

            </classes>

</test>

 <test name="TEST2">

            <classes>

                    <**class** name="Testcases.TEST\_02"></class>

            </classes>

     </test>

</suite>

Finally, the executed script can generate the report in Excel and the graph will look like :

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

RedBus locators for HomePage:

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

package phase1.lesson4.project;

import java.io.File;

import java.io.IOException;

import java.time.Duration;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.interactions.Actions;

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

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

public class RedbusHomePageDemo {

public static void main(String[] args) throws InterruptedException, IOException {

// TODO Auto-generated method stub

// Code for removing push notification:

//Create a class to handle external elements.  (Alerts)

ChromeOptions options = new ChromeOptions();

options.addArguments("--disable-notifications");

WebDriver driver = new ChromeDriver();

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

driver.manage()017

// Updated code -> SAMYAK JAIN

// Clear the source text box

driver.findElement(By.xpath("//div[@class='main-wrapper search-box-wrapper']/descendant::input[1]")).clear();

// enter input string in text box

driver.findElement(By.xpath("//div[@class='main-wrapper search-box-wrapper']/descendant::input[1]")).sendKeys("Paris");

Thread.sleep(5000);

// capture element form the list

driver.findElement(By.xpath("//div[@class='autoFill autosuggest-ul rdc-src']/descendant::li[3]")).click();

// destination

driver.findElement(By.xpath("//div[@class='main-wrapper search-box-wrapper']/descendant::input[2]")).clear();

driver.findElement(By.xpath("//div[@class='main-wrapper search-box-wrapper']/descendant::input[2]")).sendKeys("London");

Thread.sleep(5000);

driver.findElement(By.xpath("//div[@class='autoFill autosuggest-ul rdc-dest']/descendant::li[3]")).click();

//calander

Thread.sleep(3000);

driver.findElement(By.id("date-box")).click();

Thread.sleep(5000);

//Select date

driver.findElement(By.xpath("//div[@class='DatePicker\_\_CalendarContainer-sc-1x9sb82-0 bkhtIz']/descendant::span[41]")).click();

  Thread.sleep(5000);

  //click on search button

  driver.findElement(By.id("search\_butn")).click();

// Upload file using autoIT

driver.get("https://www.remove.bg/");

Thread.sleep(3000);

WebElement e1 = driver.findElement(By.xpath("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]"));

WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(10));

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[@class='mx-auto w-full px-8 max-w-5xl relative']/descendant::button[1]")));

e1.click();

Runtime.getRuntime().exec("C:\\Users\\sonal\\Desktop\\AutiitScripts\\Scriptupload3.exe");

// handle webelement using mouse hover

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

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

WebElement e2 = driver.findElement(By.xpath("//\*[@class='nav-line-2 ']"));

Actions a = new Actions(driver);

a.moveToElement(e2).perform();

Thread.sleep(5000);

driver.findElement(By.xpath("(//span[@class='nav-text'])[7]")).click();

// DB demo

// Screenshot

File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

File destFile = new File("./Screenshots/project.png");

FileUtils.copyFile(srcFile, destFile);

}

}