**1. You are given a project to implement @Test and other related annotations.**

package demo2;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class VerifyTitle {

ChromeDriver driver;

@BeforeTest

public void launchApplication() {

//1)Open the Browser

driver = new ChromeDriver();//Class object = new Class()

//2)Naviagte to application

//object.method()

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

}

@Test

public void titleVerification() {

//3)Verify the visitor on the page sees the title -'Facebook- log in or sign up'

String expectedTitle = "Facebook – log in or sign up";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest

public void closeBrowser() {

driver.quit();

}

}

**2. You are given a project to work with groups attribute of @Test and perform parallel execution.**

**package** demo2;

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

**public** **class** GroupingDemo {

@Test(groups = "Luxury Cars")

**public** **void** Car1() {

System.***out***.println("Mercedes");

}

@Test(groups = "Luxury Cars")

**public** **void** Care() {

System.***out***.println("BMW");

}

@Test(groups ={"Car","Luxury Cars"})

**public** **void** Car3() {

System.***out***.println("Toyota");

}

@Test(groups = "Car")

**public** **void** Car4() {

System.***out***.println("Hyundai");

}

@Test(groups = "Car")

**public** **void** Car5() {

System.***out***.println("Honda");

}

}

**3. You are given a project to implement soft and hard assertions on your test cases.**

**package** demo2;

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

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

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

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

**import** org.testng.asserts.SoftAssert;

**public** **class** Assertions {

SoftAssert soft = **new** SoftAssert();

WebDriver driver;

@Test

**public** **void** Launch() {

//System.setProperty("webdriver.chrome.driver", "./Resources/chromedriver.exe");

driver = **new** ChromeDriver();

**try** {

Thread.*sleep*(3000);

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Test(dependsOnMethods = { "Launch" })

**public** **void** Facebook() {

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

soft.assertEquals("Facebook – log in or sign up", driver.getTitle());

**try** {

Thread.*sleep*(2000);

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Test(dependsOnMethods = { "Facebook" })

**public** **void** Login() {

driver.findElement(By.*id*("email")).sendKeys("batman554466@gmail.com");

driver.findElement(By.*id*("pass")).sendKeys("12345");

driver.findElement(By.*name*("login")).click();

soft.assertAll();

**try** {

Thread.*sleep*(3000);

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**4. Demonstrate how extent reports are generated**.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.simplilearn</groupId>

<artifactId>MavenBasics</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>MavenBasics</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.5</version>

</dependency>

<dependency>

<groupId>com.sun.mail</groupId>

<artifactId>javax.mail</artifactId>

<version>1.6.2</version>

</dependency>

<dependency>

<groupId>javax.activation</groupId>

<artifactId>activation</artifactId>

<version>1.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-1.2-api</artifactId>

<version>2.11.1</version>

</dependency>

</dependencies>

<build>

<pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->

<plugins>

<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

</plugin>

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.2</version>

</plugin>

<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->

<plugin>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</pluginManagement>

</build>

</project>

<?xml version="1.0" encoding="UTF-8"?>

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

<suite name="Suite">

<listeners>

<listener class-name="com.Listeners.TestListener"></listener>

</listeners>

<test thread-count="5" name="Test">

<packages>

<package name="demo"></package>

</packages>

</test> <!-- Test -->

</suite> <!-- Suite -->

package demo;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class VerifyTitle {

ChromeDriver driver;

@BeforeTest

public void launchApplication() {

//1)Open the Browser

driver = new ChromeDriver();//Class object = new Class()

//2)Naviagte to application

//object.method()

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

}

@Test

public void titleVerification() {

//3)Verify the visitor on the page sees the title -'Facebook- log in or sign up'

String expectedTitle = "Facebook – log in or sign up";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest

public void closeBrowser() {

driver.quit();

}

}

**package** demo;

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

**public** **class** GroupingDemo {

@Test(groups = "Luxury Cars")

**public** **void** Car1() {

System.***out***.println("Mercedes");

}

@Test(groups = "Luxury Cars")

**public** **void** Care() {

System.***out***.println("BMW");

}

@Test(groups ={"Car","Luxury Cars"})

**public** **void** Car3() {

System.***out***.println("Toyota");

}

@Test(groups = "Car")

**public** **void** Car4() {

System.***out***.println("Hyundai");

}

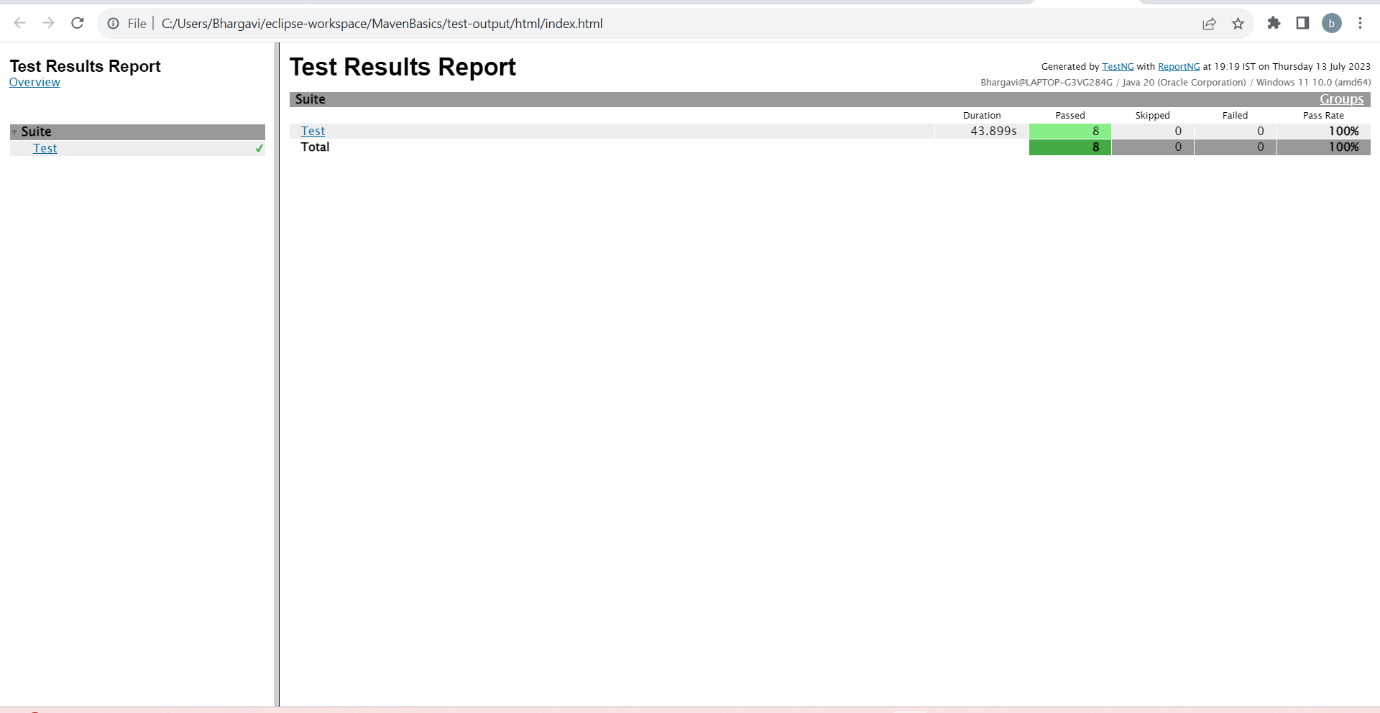
@Test(groups = "Car")

**public** **void** Car5() {

System.***out***.println("Honda");

}

}



**12. Demonstrate how data is read from an Excel sheet in Selenium**.

package utils;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.DataFormatter;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ReadExcel {

public static String[][] getData(String fileName, String sheetName) throws IOException {

File file = new File(fileName);

FileInputStream ips = new FileInputStream(file); // FileOutputStream for writing the data on excel sheet

Workbook Wb = new XSSFWorkbook(ips);

Sheet Sh = Wb.getSheet(sheetName);

int rowNum = Sh.getLastRowNum() + 1;

int colNum = Sh.getRow(0).getLastCellNum();

String[][] data = new String[rowNum][colNum];

for (int i = 0; i < rowNum; i++) {

Row row = Sh.getRow(i);

for (int j = 0; j < colNum; j++) {

Cell cell = row.getCell(j);

String value = new DataFormatter().formatCellValue(cell);

data[i][j] = value;

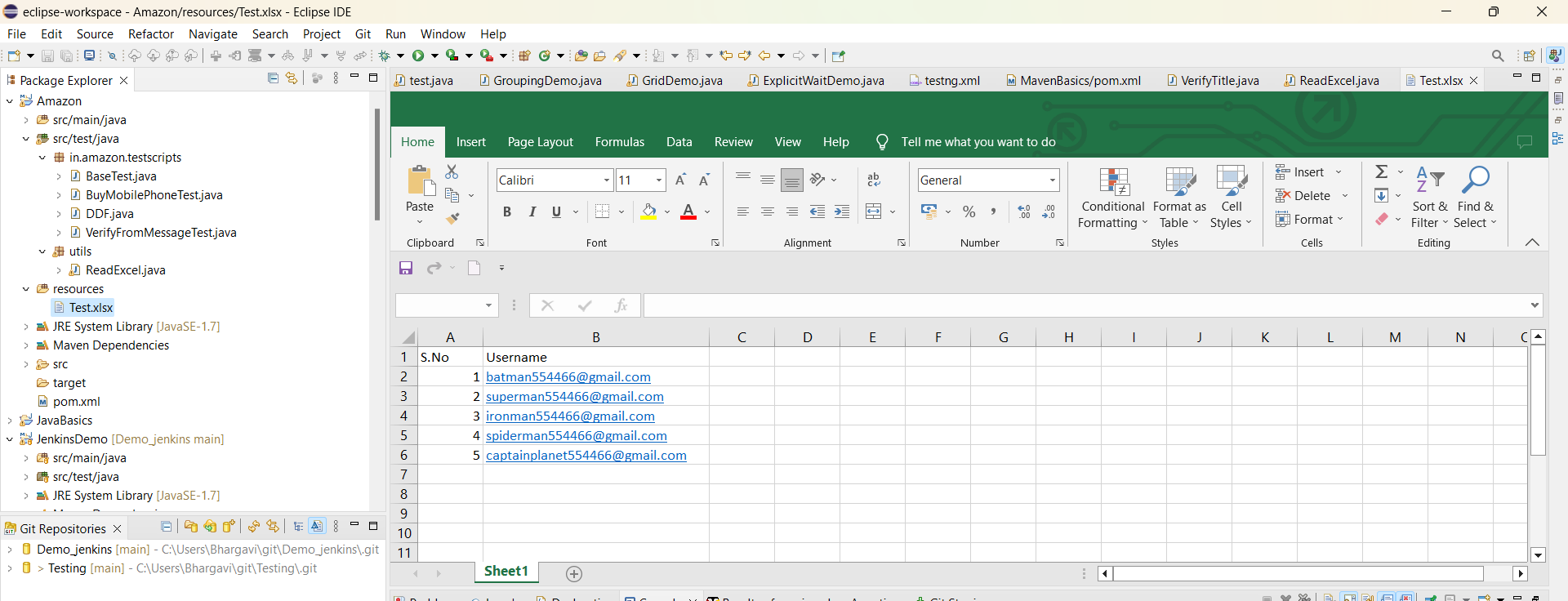
}

}

return data;

}

}



8.Configure Selenium Grid using JSON.

Setting up a Hub (Making a computer - ‘Hub’)

* Download Selenium Server Standalone jar file.
* Open CMD and point to the same folder location where the above jar file is present.
* Write the following command-



* Update dependency of Selenium to the following –

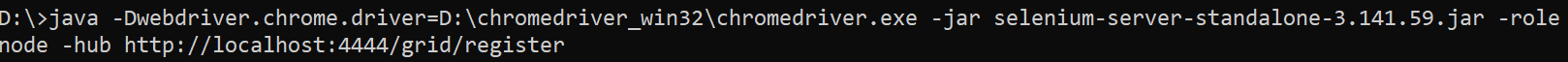
<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

* Download ‘chromedriver.exe’ from <https://chromedriver.chromium.org/downloads>. Its version should match to the version of Chrome you have in installed in your computer. Download and unzip this file wherever you want in your computer.
* Write the following line in CMD to connect node to the Hub –
* 

**9.Demonstrate Running Tests on Selenium Grid on Multiple Browsers**

package demo;

import java.net.MalformedURLException;

import java.net.URL;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class GridDemo {

WebDriver driver;

@BeforeTest

public void launchApplication() throws MalformedURLException {

//Specify the application of the node where the test case is to be executed using DesiredCapabilities class.

DesiredCapabilities cap = new DesiredCapabilities();

cap.setBrowserName("chrome");

cap.setPlatform(Platform.WINDOWS);

driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"),cap);

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

}

@Test

public void titleVerification() {

//3)Verify the visitor on the page sees the title -'Facebook- log in or sign up'

String expectedTitle = "Facebook – log in or sign up";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest

public void closeBrowser() {

driver.quit();

}

}

**10.Demonstrate page object design pattern in Selenium**.

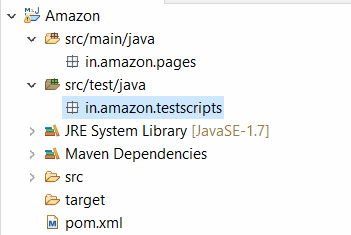
What is it – It is some set of rules (framework) which tells us how to write automation script when the test case logic is spread across multiple pages.

Definition- It states that for each new page in the application, there must be a corresponding class in the project.

Code Structure – Create methods in Page classes and call those methods in the test class. All the main logic like mouse hovering, clicking, writing text, selecting value from drop-down etc. will go in the page class in the form a method. This method will then be called in the test class.

Steps

1. Create a Maven project.
2. Delete the default packages and create 2 new packages – one in src/main/java folder and the other in src/test/java folder as shown below –



1. Update pom.xml with Selenium and TestNG (or any other dependencies) dependencies.

**11. Demonstrate how Apache POI is configured in Selenium.**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>in.amazon</groupId>

<artifactId>Amazon</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Amazon</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.10.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.8.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>5.2.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.3</version>

</dependency>

</dependencies>

<build>

<pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->

<plugins>

<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

</plugin>

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.2</version>

</plugin>

<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->

<plugin>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</pluginManagement>

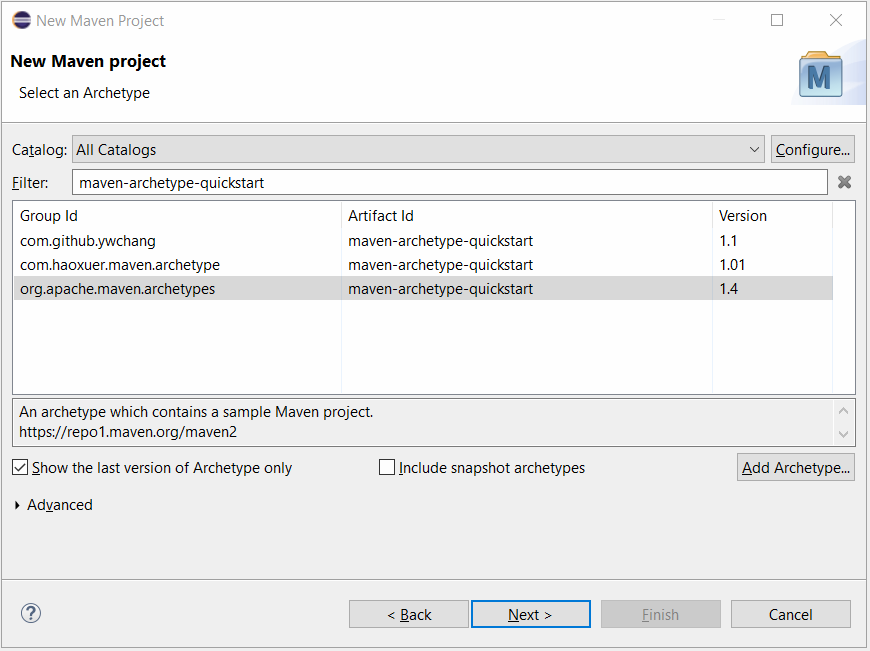
</build>

</project>

13.Integrate Selenium with Maven

**Creating a Maven project in Eclipse**

1. Go to File > New > Project > Maven > Maven Project > Next > Next > Write ‘maven-archetype-quickstart’ in the Filter box > Choose the last option and then Click ‘Next’ as shown below –



1. Write Group id > Write Artifact ID (Name of the project) > Finish.

3 famous types of Reports used with Selenium –

1. Default TestNG reports
2. ReportNG reports
3. Extent Reports (Used in real time by most of the IT companies)

Steps to generate Extent reports –

* Update pom.xml with following dependencies-

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.5</version>

</dependency>

<dependency>

<groupId>com.sun.mail</groupId>

<artifactId>javax.mail</artifactId>

<version>1.6.2</version>

</dependency>

<dependency>

<groupId>javax.activation</groupId>

<artifactId>activation</artifactId>

<version>1.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-1.2-api</artifactId>

<version>2.11.1</version>

</dependency>

**15. Demonstrate using listeners in Selenium.**

<?xml version="1.0" encoding="UTF-8"?>

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

<suite name="Suite">

<listeners>

<listener class-name="com.Listeners.TestListener"></listener>

</listeners>

<test thread-count="5" name="Test">

<packages>

<package name="demo"></package>

</packages>

</test> <!-- Test -->

</suite> <!-- Suite -->

package com.Listeners;

import java.io.File;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.reporter.ExtentHtmlReporter;

import com.aventstack.extentreports.reporter.configuration.ChartLocation;

import com.aventstack.extentreports.reporter.configuration.Theme;

public class ExtentManager {

private static ExtentReports extent;

private static String reportFileName = "Test-Automation-Report"+".html";

private static String fileSeperator = System.getProperty("file.separator");

private static String reportFilepath = System.getProperty("user.dir") +fileSeperator+ "TestReport";

private static String reportFileLocation = reportFilepath +fileSeperator+ reportFileName;

public static ExtentReports getInstance() {

if (extent == null)

createInstance();

return extent;

}

//Create an extent report instance

public static ExtentReports createInstance() {

String fileName = getReportPath(reportFilepath);

ExtentHtmlReporter htmlReporter = new ExtentHtmlReporter(fileName);

htmlReporter.config().setTestViewChartLocation(ChartLocation.TOP);

htmlReporter.config().setChartVisibilityOnOpen(true);

htmlReporter.config().setTheme(Theme.STANDARD);

htmlReporter.config().setDocumentTitle(reportFileName);

htmlReporter.config().setEncoding("utf-8");

htmlReporter.config().setReportName(reportFileName);

htmlReporter.config().setTimeStampFormat("EEEE, MMMM dd, yyyy, hh:mm a '('zzz')'");

extent = new ExtentReports();

extent.attachReporter(htmlReporter);

//Set environment details

extent.setSystemInfo("OS", "Windows");

extent.setSystemInfo("AUT", "QA");

return extent;

}

//Create the report path

private static String getReportPath (String path) {

File testDirectory = new File(path);

if (!testDirectory.exists()) {

if (testDirectory.mkdir()) {

System.out.println("Directory: " + path + " is created!" );

return reportFileLocation;

} else {

System.out.println("Failed to create directory: " + path);

return System.getProperty("user.dir");

}

} else {

System.out.println("Directory already exists: " + path);

}

return reportFileLocation;

}

}

**package** com.Listeners;

**import** java.util.HashMap;

**import** java.util.Map;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**public** **class** ExtentTestManager {

**static** Map<Integer, ExtentTest> *extentTestMap* = **new** HashMap<Integer, ExtentTest>();

**static** ExtentReports *extent* = ExtentManager.*getInstance*();

**public** **static** **synchronized** ExtentTest getTest() {

**return** (ExtentTest) *extentTestMap*.get((**int**) (**long**) (Thread.*currentThread*().~~getId~~()));

}

**public** **static** **synchronized** **void** endTest() {

*extent*.flush();

}

**public** **static** **synchronized** ExtentTest startTest(String testName) {

ExtentTest test = *extent*.createTest(testName);

*extentTestMap*.put((**int**) (**long**) (Thread.*currentThread*().~~getId~~()), test);

**return** test;

}

}

**package** com.Listeners;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

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

**import** org.openqa.selenium.io.FileHandler;

**import** org.testng.ITestContext;

**import** org.testng.ITestListener;

**import** org.testng.ITestResult;

**import** com.aventstack.extentreports.MediaEntityBuilder;

**import** com.aventstack.extentreports.Status;

**import** com.aventstack.extentreports.model.Log;

**import** org.apache.log4j.Logger;

**public** **class** TestListener **implements** ITestListener {

**private** **static** Logger *log* = Logger.*getLogger*(Log.**class**.getName());

**public** **void** onStart(ITestContext context) {

System.***out***.println("\*\*\* Test Suite " + context.getName() + " started \*\*\*");

}

**public** **void** onFinish(ITestContext context) {

System.***out***.println(("\*\*\* Test Suite " + context.getName() + " ending \*\*\*"));

ExtentTestManager.*endTest*();

ExtentManager.*getInstance*().flush();

}

**public** **void** onTestStart(ITestResult result) {

System.***out***.println(("\*\*\* Running test method " + result.getMethod().getMethodName() + "..."));

ExtentTestManager.*startTest*(result.getMethod().getMethodName());

}

**public** **void** onTestSuccess(ITestResult result) {

System.***out***.println("\*\*\* Executed " + result.getMethod().getMethodName() + " test successfully...");

ExtentTestManager.*getTest*().log(Status.***PASS***, "Test passed");

}

**public** **void** onTestFailure(ITestResult result) {

*log*.info("\*\*\* Test execution " + result.getMethod().getMethodName() + " failed...");

*log*.info((result.getMethod().getMethodName() + " failed!"));

//ITestContext context = result.getTestContext();

WebDriver driver = (WebDriver) result.getTestContext().getAttribute("driver");

String targetLocation = **null**;

String testClassName = result.getInstanceName().trim();

String timeStamp = **new** SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(**new** java.util.Date()); // get timestamp

String testMethodName = result.getName().toString().trim();

String screenShotName = testMethodName + timeStamp + ".png";

String fileSeperator = System.*getProperty*("file.separator");

String reportsPath = System.*getProperty*("user.dir") + fileSeperator + "TestReport" + fileSeperator

+ "screenshots";

*log*.info("Screen shots reports path - " + reportsPath);

**try** {

File file = **new** File(reportsPath + fileSeperator + testClassName); // Set

// screenshots

// folder

**if** (!file.exists()) {

**if** (file.mkdirs()) {

*log*.info("Directory: " + file.getAbsolutePath() + " is created!");

} **else** {

*log*.info("Failed to create directory: " + file.getAbsolutePath());

}

}

File screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.***FILE***);

targetLocation = reportsPath + fileSeperator + testClassName + fileSeperator + screenShotName;// define

// location

File targetFile = **new** File(targetLocation);

*log*.info("Screen shot file location - " + screenshotFile.getAbsolutePath());

*log*.info("Target File location - " + targetFile.getAbsolutePath());

FileHandler.*copy*(screenshotFile, targetFile);

} **catch** (FileNotFoundException e) {

*log*.info("File not found exception occurred while taking screenshot " + e.getMessage());

} **catch** (Exception e) {

*log*.info("An exception occurred while taking screenshot " + e.getCause());

}

// attach screenshots to report

**try** {

ExtentTestManager.*getTest*().fail("Screenshot",

MediaEntityBuilder.*createScreenCaptureFromPath*(targetLocation).build());

} **catch** (IOException e) {

*log*.info("An exception occured while taking screenshot " + e.getCause());

}

ExtentTestManager.*getTest*().log(Status.***FAIL***, "Test Failed");

}

**public** **void** onTestSkipped(ITestResult result) {

System.***out***.println("\*\*\* Test " + result.getMethod().getMethodName() + " skipped...");

ExtentTestManager.*getTest*().log(Status.***SKIP***, "Test Skipped");

}

**public** **void** onTestFailedButWithinSuccessPercentage(ITestResult result) {

System.***out***.println("\*\*\* Test failed but within percentage % " + result.getMethod().getMethodName());

}

}

**16.Demonstrate how artifactory can be installed.**

package in.amazon.JenkinsDemo;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class VerifyTitle {

WebDriver driver;

@BeforeTest

public void launchApplication() {

//1)Open the Browser

driver = new ChromeDriver();//Class object = new Class()

//2)Naviagte to application

//object.method()

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

}

@Test

public void titleVerification() {

//3)Verify the visitor on the page sees the title -'Facebook- log in or sign up'

String expectedTitle = "Facebook – log in or sign up";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest

public void closeBrowser() {

driver.quit();

}

}

**17. Build and configure CI/CD pipeline with Maven Project.**

**18. Demonstrate how to build and configure CI/CD pipeline with Selenium WebDriver.**

**19. Demonstrate Selenium integration with Jenkins.**

1. **Setting up the cloud machine**

* Login into AWS Management console.
* Search for EC2 service. (Services > Compute > EC2)
* Click on Launch Instance Drop down > Select Launch Instance
* Provide a name for the new instance.
* Download key in .ppk (windows) or .pem(mac) format
* Click ‘Launch Instance’.
* Click ‘View All Instances’

**2) Connecting to the cloud machine**

For Mac users –

* Click on ‘Connect’ tab > SSH Client tab and then follow the instructions mentioned on the page. SSH Client means -> Teminal

For Window users –

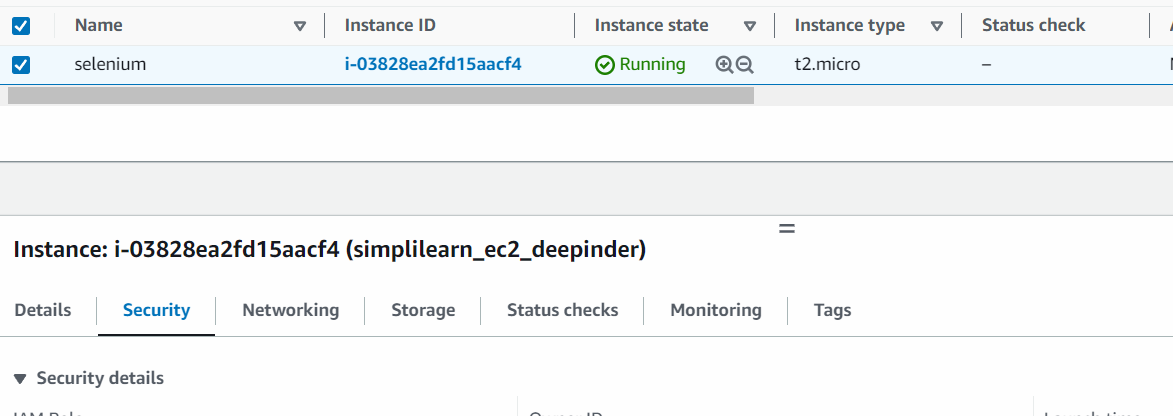
* Launch putty in your machine.
* Enter public ip address of cloud machine in putty.
* Expand SSH > Expand Auth > Click Credentials > Browse the .ppk file downloaded while creating the cloud machine.
* Click Open button.
* Accept the Security Alert, a black screen will appear.
* Type ‘ec2-user’ on the black screen and hit enter.

Run following commands using Putty on Cloud Machine

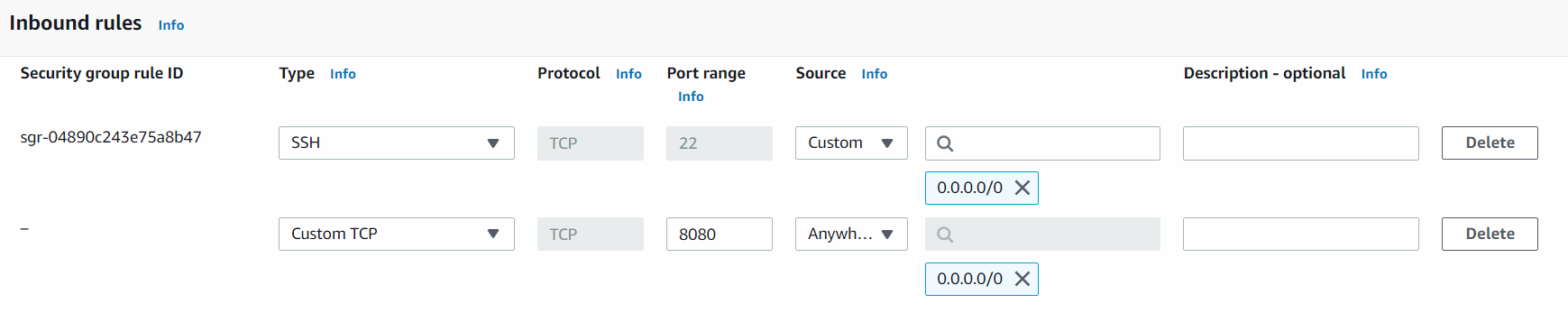
1. sudo yum update –y
2. sudo wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo
3. sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
4. sudo yum install jenkins java –y
5. sudo chkconfig --add jenkins
6. sudo systemctl start jenkins
7. sudo systemctl status jenkins

**Change Cloud machine’s security settings so that it can be accessed from any outer IP**

1. Click on ‘Security’ tab.

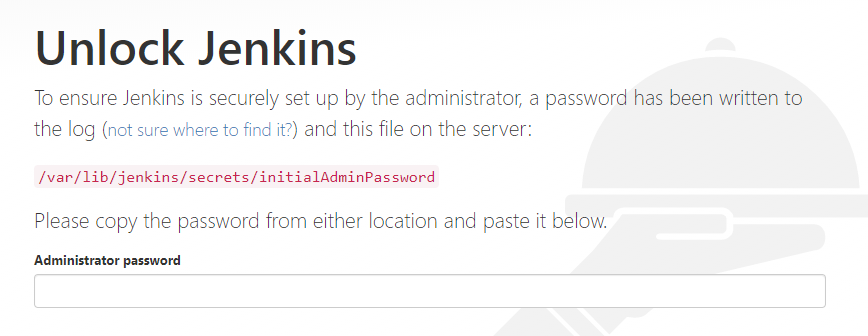


1. Click on alphanumeric string under ‘Security groups’.
2. On next page, click on ‘Edit Inbound rules’.
3. Click on ‘Add’ button on next page and select following settings –



1. Click ‘Save Rules’ button.
2. Go to ‘Details’ tab again > Copy Public IPv4 DNS.
3. Open any browser, paste the DNS from last step and type ‘:8080’ at the end and hit enter.

After executing above 7 steps, we have to configure Jenkins, for that we have to read a temp password stored at a location which is shown here –



To read the password from this location, write the commands –

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

* On the next screen, click ‘Install Suggested Plugins’.

Configure Jenkins

1. Go to Manage Jenkins > Manage Plugins > Available tab> Search for GitHub Integration and GitHub Authentication.

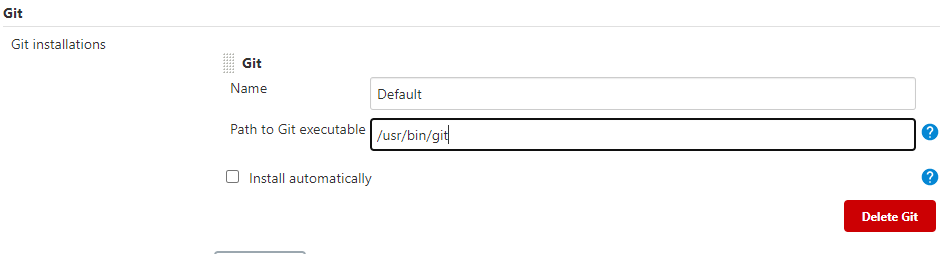
Now install git in AWS instance so that we can pull the code into it. Run the following commands on Putty –

a) sudo yum install git –y

Get the path of installation of git in AWS machine by writing in Putty

a) which git

1. Go to Manage Jenksin > Tools> Update the path as shown –

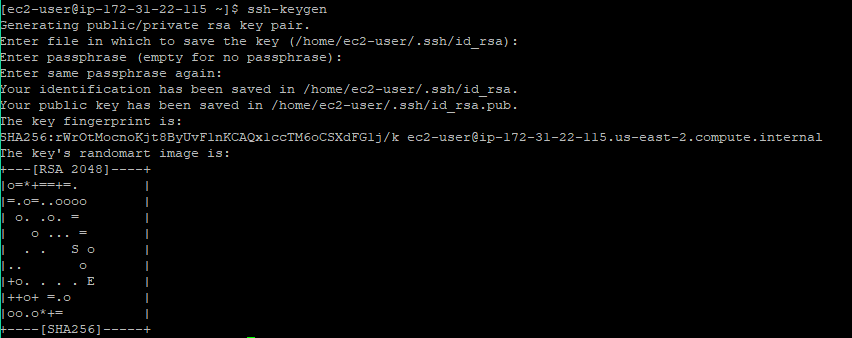


Add cloud computer as a Trusted Source in GitHub

1. Write in putty –

ssh-keygen

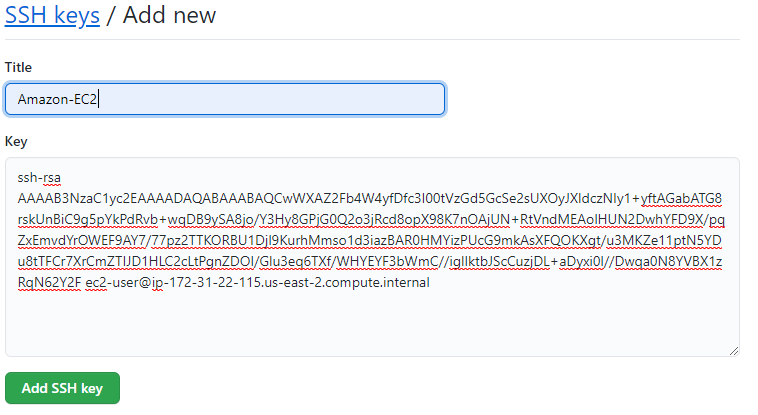
You will see following screen –



1. The key generated in the previous step is to be read from the location specified. For that write –

sudo cat /home/ec2-user/.ssh/id\_rsa.pub

1. A unique string will be printed on screen, copy that.
2. Go to GitHub and login with your credentials
3. Go to Settings > SSH and GPG Keys > New SSH Key > Paste the key copied in step 3 above> Add SSH Key

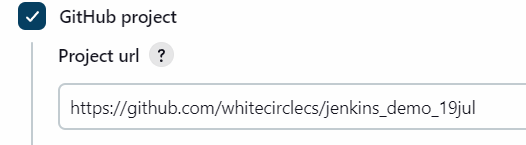


1. Go to Manage Jenkins> Plugins> ‘Available’> Search for ‘Maven Integration Plugin’.
2. Go to Manage Jenkins > Tools > Under ‘JDK’ option, for ‘JDK installations’, provide any value to ‘Name’ field, check ‘install automatically’. Do the same for Maven too.
3. Install Google Chrome in Cloud machine –

sudo curl https://intoli.com/install-google-chrome.sh | bash

sudo mv /usr/bin/google-chrome-stable /usr/bin/google-chrome

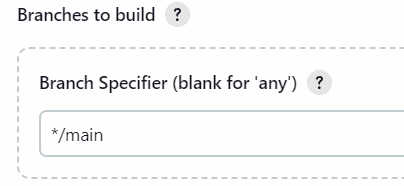
1. Go to Jenkins Dashboard > New Item > Write name of job – ‘jenkins\_job’ > Select ‘Maven Project’ > Click Ok.
2. On the next page that opens, write GitHub project URL –



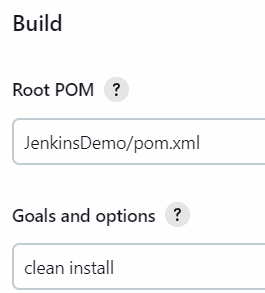
1. Scroll down to ‘Source Code Management’ > Select ‘Git’ and then write repo URL –



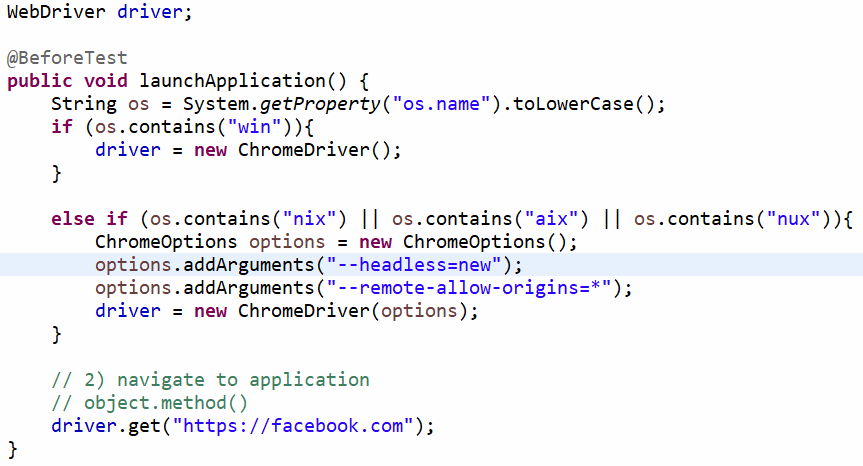
1. Now you have save your GitHub Credentials. For this Click ‘Add’> Jenkins> On the next window that opens, write GitHub username and Password Access Token. Click ‘Add’ button at the end after it.
2. The pop-up will close after last step > From the ‘Credentials’ drop-down select recently saved username and password combo.
3. Under ‘Branches to Build’, change ‘master to main’



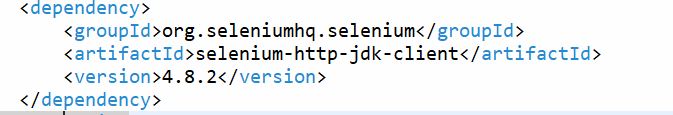
1. In the configure page, go to ‘Build’ option and in the ‘Root POM’ option provide the path of pom.xml file as shown below –



1. Click ‘Save’.
2. Make following changes in your Eclipse code –



Add the following dependency too in pom.xml –



1. Rename your Test class so that it ends with the word ‘Test’. For e.g. VerifyTitleTest and push these changes to GitHub.

**20.Demonstrate the TDD with TestNG**

package tdd;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class VerifyTitle {

WebDriver driver;

@BeforeTest

public void launchBrowser() {

//1)Open the browser

driver = new ChromeDriver();

//2)Navigate to application

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

}

@Test

public void verifyTitle() {

//3)Verify the title is-'Facebook - log in or sign up'

String expectedTitle = "Facebook - log in or sign up";

//String actualTitle= driver.getTitle();

String actualTitle= "Facebook - log in or sign up";

Assert.assertEquals(actualTitle, expectedTitle);

}

}