**Integrating Selenium with Jenkins**

This section will guide you to:

* Integrate Selenium with Jenkins
* Configure Maven build

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* JRE: OpenJDK Runtime Environment 11.0.2

This lab has ten subsections, namely:

* + 1. Creating a Maven project
    2. Editing the pom.xml and adding Selenium and JUnit dependencies
    3. Creating a Java class named NewTest
    4. Adding TestNG libraries to the Class Path
    5. Converting the project into TestNG and changing the run configuration
    6. Running the project as Maven test
    7. Installing Jenkins
    8. Adding Maven plugins to Jenkins
    9. Adding the location of pom.xml in Jenkins CI Job
    10. Pushing the code to your GitHub repositories

**Step 2.4.1:** Creating a Maven project

* Open Eclipse
* Go the **File** menu. Choose **New->Other->Maven->Maven Project**
* On the **New Maven Project** dialog, select **Create a simple project** and click **Next**
* Enter **SeleJenk** in **Group Id** and **Artifact Id** and click on **Finish**

**Step 2.4.2:** Editing the pom.xml and adding Selenium and JUnit dependencies

* In the Project Explorer, expand the project **SeleJenk**
* Select **pom.xml** from **Project Explorer**
* Enter the following code:

<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>SeleJenk</groupId>

<artifactId>SeleJenk</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>2.45.0</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.6.0</version>

<configuration>

<goalPrefix>plugin</goalPrefix>

<outputDirectory>target/dir</outputDirectory>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Step 2.4.3:** Adding TestNG libraries to the Class Path

* In the Project Explorer, right click on **Test Assertions**
* Select **Properties**. Select **Java Build Pat**h from the list. Go to **Libraries**
* Click on **Add Library.** Select **TestNG** (Refer FSD: Lab Guide - Phase 5). Click on **Next**. Click on **Finish**
* Click on **Apply and Close**

**Step 2.4.4:** Creating a TestNG class named NewTest

* In the Project Explorer, expand **SeleJenk**
* Right click on **SeleJenk**. Click on **New->Other->TestNG->TestNG Class**
* Enter **Package name** as **com.example** and **NewTest** in the **Name** textbox and click on **Finish**
* Enter the following code:

package com.example;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class NewTest {

private WebDriver driver;

SoftAssert soft=new SoftAssert();

@Test

public void testEasy() {

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

driver=new ChromeDriver();

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

String title = driver.getTitle();

soft.assertEquals("FB Login",title);

}

@BeforeTest

public void beforeTest() {

driver = new FirefoxDriver();

}

@AfterTest

public void afterTest() {

driver.quit();

}

}

**Step 2.4.5:** Converting the project into TestNG and changing the run configuration

* In the Project Explorer, expand **SeleJenk**
* Right click on **SeleJenk** and choose **TestNG->convert to TestNG**

**Step 2.4.6:** Running the project as Maven test

* Right click on **SeleJenk**
* Click on **Run AS->Maven Test**

**Step 2.4.7:** Installing Jenkins

* Jenkins is already installed in your Practice lab.(Refer FSD: Lab Guide - Phase 5)
* Use the following commands to navigate to the above-mentioned directory.

*cd /usr/share*

*ls*

**Step 2.4.8:** Adding Maven plugins to Jenkins

* In the Jenkins dashboard, click on **Manage Jenkins**
* Click on **Manage Plugins**
* Select the **Available** tab, then find the **Maven Integration** plugin
* Click **Install** without restart

**Step 2.4.9:** Adding the location of pom.xml in Jenkins CI Job

* Click on **New Item** to create **CI Job**
* Select the **Maven project radio** button and enter **Item Name** as **SeleJenk**
* Click on **Build Environment**
* In **Root POM,** specify the location of pom.xml from your Eclipse workspace
* In **Goals and Options**, type **clean test.** Click on **Save**
* Click on the **SeleJenk** project page and click on the **Build Now** link

**Step 2.4.10:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**