



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To Setup and Run Selenium Tests in Jenkins Using Maven

Objective: Objective is to setup enables seamless integration of automated testing into the CI/CD pipeline, facilitating faster feedback loops and promoting a culture of continuous improvement in software development.

Theory:

Jenkins is the leading open-source continuous integration tool developed by Hudson lab. It is cross-platform and can be used on Windows, Linux, Mac OS and Solaris environments. Jenkins is written in Java. It has taken the place as one of the best open-source tools that allow continuous integration and build management.

Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass. Jenkins can schedule your tests to run at specific time. You can save the execution history and Test Reports. Jenkins supports Maven for building and Testing a project in continuous integration

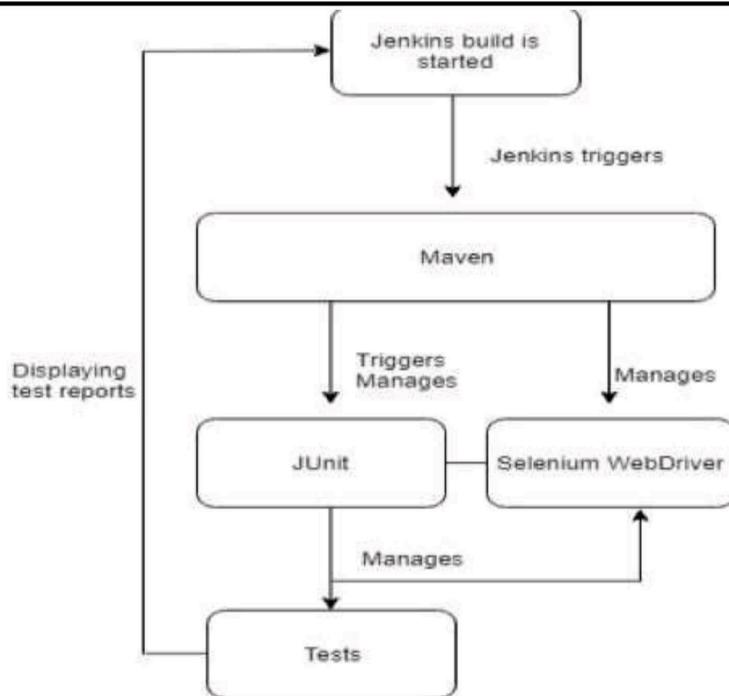
Maven is a powerful project / build management tool, based on the concept of a POM (Project Object Model) that includes project information and configuration information for Maven such as construction directory, source directory, dependency, test source directory, Goals, plugins, etc.

Integrating Maven with Selenium provides following benefits Apache Maven provides support for managing the full lifecycle of a test project. Maven is used to define project structure, dependencies, build, and test management. Using pom.xml(Maven) you can configure dependencies needed for building testing and running code. Maven automatically downloads the necessary files from the repository while building the project.



Vidyavardhini's College of Engineering and Technology

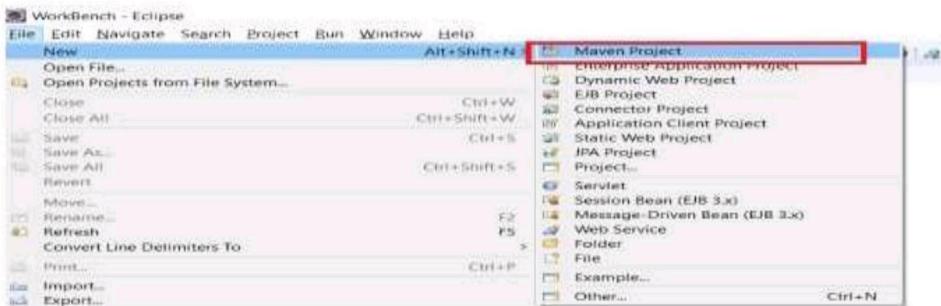
Department of Artificial Intelligence & Data Science



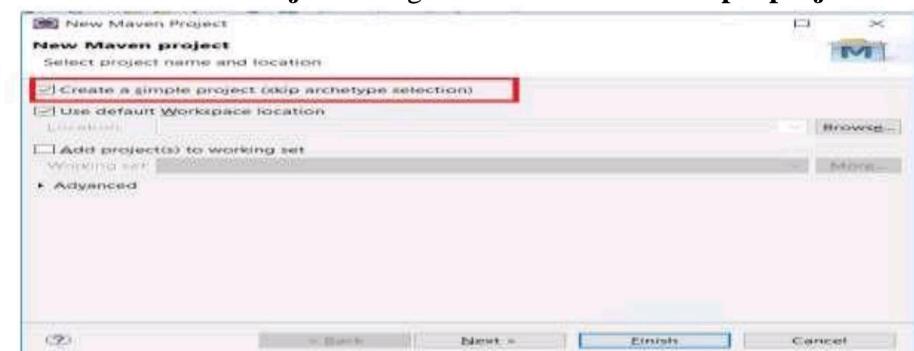
Steps:

---Create a Maven Selenium script---

1. In Eclipse IDE, create a new project by selecting **File | New | Maven Project** from the Eclipse menu.



2. On the **New Maven Project** dialog select the **Create a simple project** and click **Next**

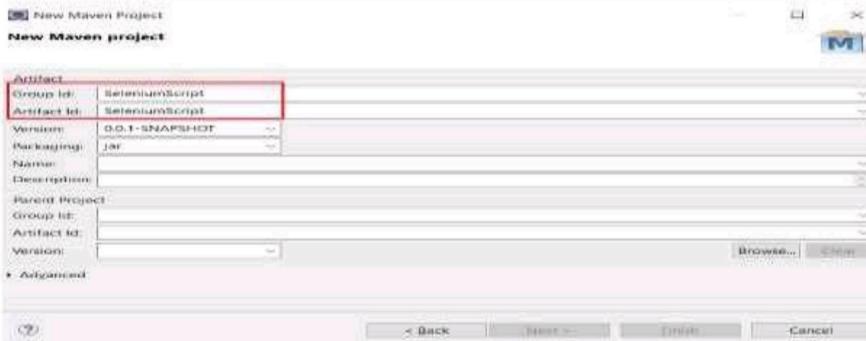


3. Enter **SeleniumScript** in **Group Id:** and **Artifact Id:** and click **finish**

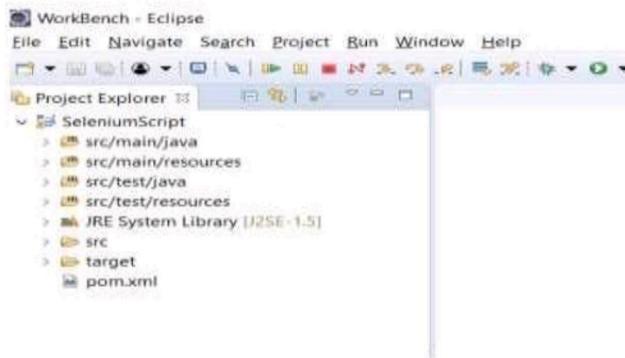


Vidyavardhini's College of Engineering and Technology

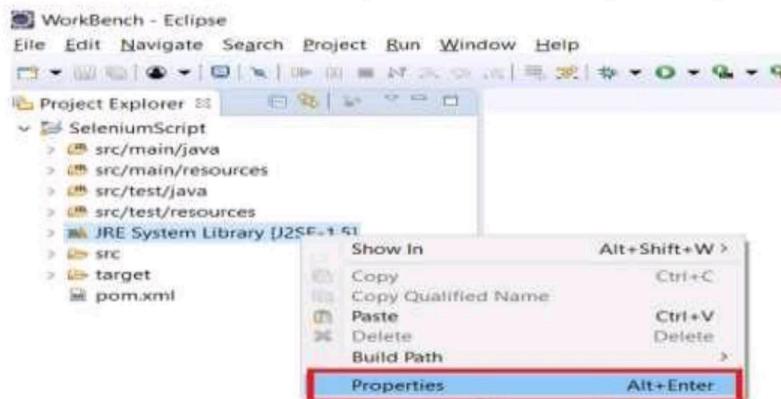
Department of Artificial Intelligence & Data Science



4. Eclipse will create webdriverTest.



5. Right click on JRE System Library and select the Properties option from the menu.

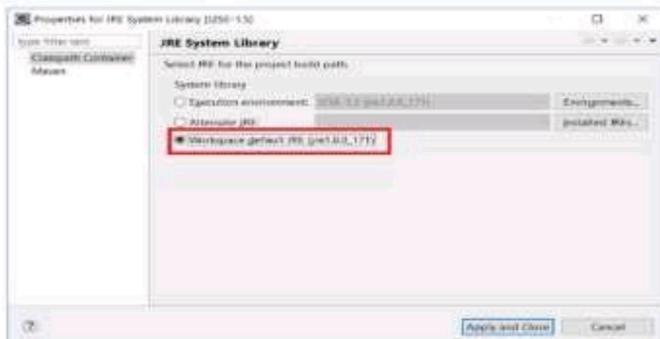


6. On the Properties for JRE System Library dialog box , make sure Workspace default JRE is selected and click ok.

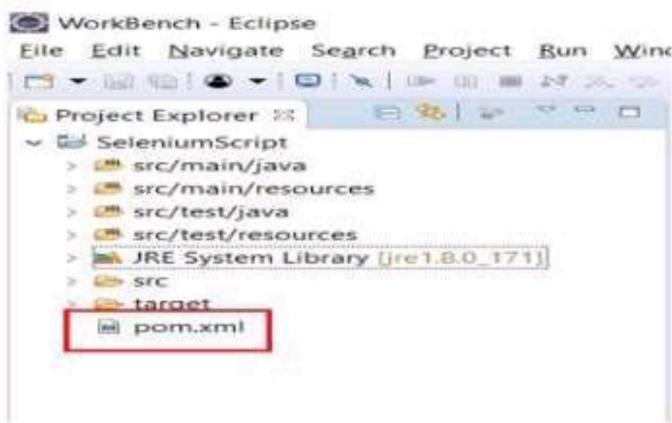


Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science



7. Select pom.xml from project explorer.



8. Add selenium, Maven, TestNG, Junit dependencies to pom.xml in the code.

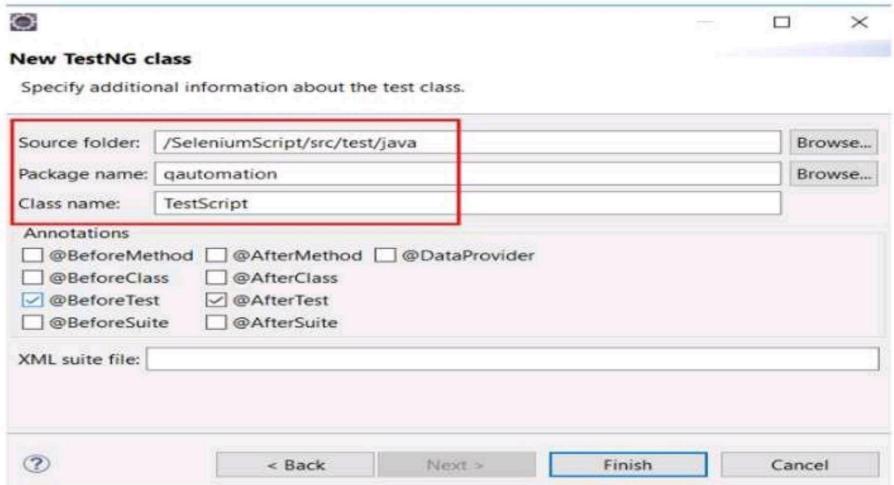
```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>2.45.0</version>
    </dependency>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>6.8</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

9. Create a new file TestNG class File|New|Others|TestNG|TestNG Class. Enter Package name as “Qautomation” and “TestScript” in the Name:textbox and click on the Finish button.

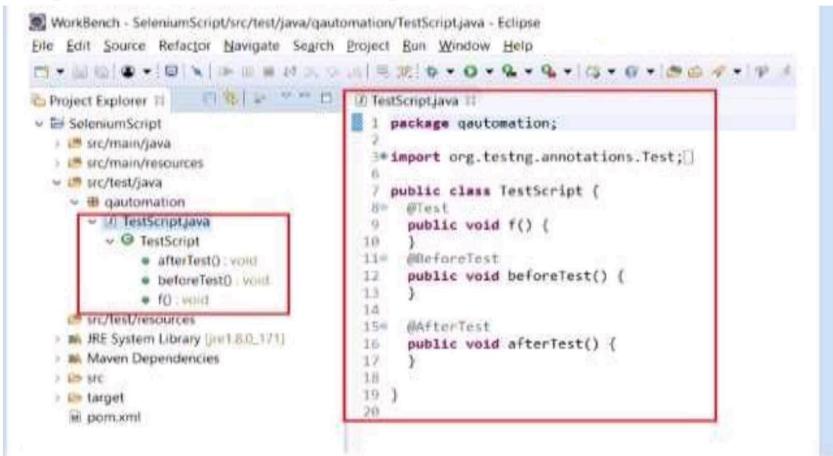


Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science



10. Eclipse will create the TestScript class



11. Add following code to the TestScript class and respective browser drivers for chrome,firefox and IE.

```
package qautomation;
import org.testng.annotations.Test;
import org.testng.annotations.BeforeTest;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.ie.InternetExplorerDriver;
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
import org.openqa.selenium.DesiredCapabilities;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
public class TestScript {
    public static WebDriver driver=null;
    public String browser = System.getProperty("browser");
    public String url = System.getProperty("URL");

    @BeforeTest
    public void beforeTest() {

        if(browser.equalsIgnoreCase("Chrome"))
        {
            System.setProperty("webdriver.chrome.driver",
            System.getProperty("user.dir")+"\chromedriver.exe");
            Map<String, Object> prefs = new HashMap<String, Object>();
            ChromeOptions options = new ChromeOptions();
            options.setExperimentalOption("prefs", prefs);
            options.addArguments("--disable-arguments");
            options.addArguments("--test-type");
            options.addArguments("test");
            options.addArguments("disable-infobars");
            driver = new ChromeDriver(options);
        }
        else if(browser.equalsIgnoreCase("FireFox"))
        {
            System.setProperty(FirefoxDriver.SystemProperty.DRIVER_USE_MARIONETTE
            ,"true");
            System.setProperty(FirefoxDriver.SystemProperty.BROWSER_LOGFILE,Syste
            m.getProperty("user.dir")+"\FireFoxLogs.txt");
            System.setProperty("webdriver.gecko.driver",
            System.getProperty("user.dir")+"\geckodriver_v23.exe");
            FirefoxProfile profile = new FirefoxProfile();
            profile.setAcceptUntrustedCertificates(false);
            FirefoxOptions options = new FirefoxOptions().setProfile(profile);
            driver = new FirefoxDriver(options);
            driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
            driver.manage().window().maximize();
        }
        else if (browser.equalsIgnoreCase("IE"))
        {
            System.setProperty("webdriver.ie.driver",
            System.getProperty("user.dir")+"\IEDriverServer351.exe");
            DesiredCapabilities caps = DesiredCapabilities.internetExplorer();
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

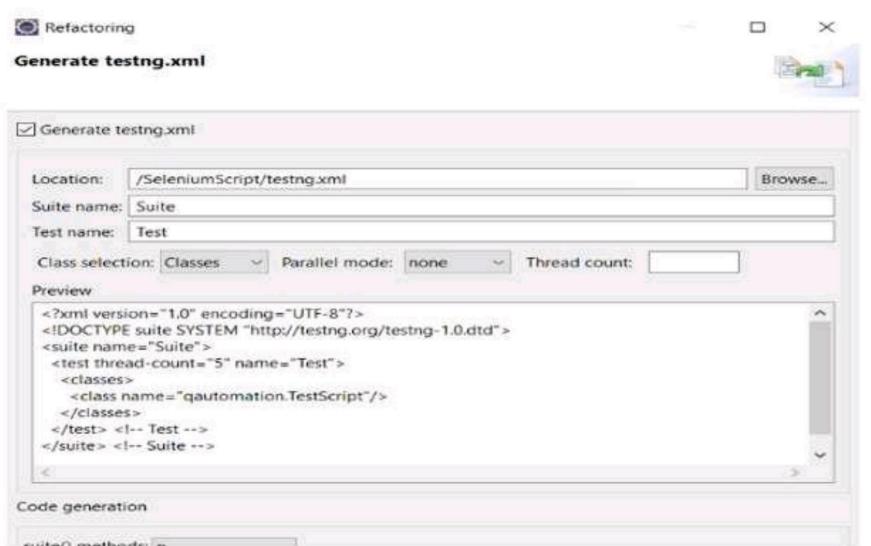
```
caps.setCapability(InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNO
RING_SECURITY_DOMAINS,true);
caps.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING,true);
caps.setCapability(InternetExplorerDriver.UNEXPECTED_ALERT_BEHAVIOR,"
accept");
caps.setCapability(InternetExplorerDriver.REQUIRE_WINDOW_FOCUS,true);
caps.setCapability(InternetExplorerDriver.INITIAL_BROWSER_URL,"http://www.
google.com/");
driver = new InternetExplorerDriver(caps);
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
driver.manage().window().maximize();
}
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
driver.manage().window().maximize();
}
@Test
public void TestApplication() {
driver.get(url);
String title = driver.getTitle();
System.out.println("Title="+title);
Assert.assertTrue(title.contains("QAutomation"));
}
@AfterTest
public void afterTest() {
    driver.quit();
}
}
```

12. Right click on the WebdriverTest and select TestNG| Convert to TestNG. Eclipse will create testing.xml which says that you need to run only one test with the name TestApplication.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science



13. Adding dependencies and plugins

Additionally we need to add

1. Maven-compiler-plugin
2. Maven-surefire-plugin
3. Testng.xml

-----Integrating your test to Jenkins-----

1. Launch and login into jenkins URL – <http://localhost:8080/>



2. Click on new item and enter an appropriate name for the new job , select Maven Project and click on save.



Vidya Vardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins dashboard with a search bar at the top. Below it, there is a section titled "Enter an item name" with a text input field. A list of project types is displayed:

- Freestyle project:** This is the central flavor of Jenkins. Jenkins will build your project, containing any SCM with any build system, and this can be used for something other than software builds.
- Maven project:** Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline:** Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines, formerly known as workflows, or for executing complex activities that do not easily fit in the static job type.
- Multi-configuration project:** Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform specific builds, etc.
- Bitbucket Team Project:** Create a Bitbucket Cloud Team (or Bitbucket Server) Project for all repositories matching some defined manner.
- Folder:** Groups a collection that shares a common name in it. Useful for putting things together. Unlike a view, which is just a filter, a folder creates a separate view, so you can have multiple things of the same name as long as they are in different folders.

3. A new empty job has been created at this point.

The screenshot shows the "General" configuration page for a Jenkins job. The "Maven Info Plugin Configuration" section is expanded, showing the following settings:

- Assign visible name:
- Assign description:
- This project is parameterized
- Static project
- This project is generalized
- Jenkins build
- Disable this project
- Execute concurrent builds if necessary

Buttons at the bottom: **Save** and **Apply**.

4. Jenkins Parameterized Build in Jenkins just check the checkbox **This project is parameterized** and add the parameter by **Add Parameter** as per your project requirement.

The screenshot shows the "Maven Info Plugin Configuration" section of the Jenkins job configuration. Two parameters have been added:

- Choice Parameter:**
 - Name: **Browser**
 - Options: Chrome, Firefox, IE
 - Description: Select a browser for testing
- String Parameter:**
 - Name: **URL**
 - Default Value: **https://appiumacademy.com**
 - Description: Enter application URL

Buttons at the bottom: **Save** and **Apply**.

5. If code is located on Git Under **Source Management**, select the appropriate repository for the location of project and pass the URL and credentials.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Source Code Management

None
 CVS
 CVS Projectset
 Git

Repositories Repository URL: git@bitbucket.localgroup: SeleniumScript.git

Credentials

- In the “pre-steps” build section another set of parameters can be passed to the Jenkins build. Specify the Maven targets that need to be executed in order to run test.

if your source code is located on Git the do below setting under **Build** section:

Build

Root POM: pom.xml

Goals and options: test -Dsurefire.suiteXmlFiles="\$TestSuite" -Dbrowser="\$BROWSER" -DURL="\$APP_URL"

If you have selenium code on your local just pass the pom.xml path in **Root POM**.

Build

Root POM: C:\AutomationTesting\Scripts\WorkBench\SeleniumScript\pom.xml

Goals and options: test -Dsurefire.suiteXmlFiles="\$TestSuite" -Dbrowser="\$BROWSER" -DURL="\$APP_URL"

- Run the test in Jenkins by clicking on Building with Parameters.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

8. Run the test in Jenkins by clicking on *Build with Parameters*.

The screenshot shows the Jenkins dashboard for the 'SeleniumJenkins' project. On the left sidebar, there is a 'Build with Parameters' button, which is highlighted with a red box. The main area displays the project name 'Maven project SeleniumJenkins' and various navigation links like 'Workspace', 'Recent Changes', and 'Permalinks'. Below the sidebar, there is a 'Build History' section with a search bar and RSS feed links.

8. Select the browser you want to run from dropdown.

The screenshot shows the 'Build with Parameters' configuration page for the 'SeleniumJenkins' project. It includes fields for 'APP_URL' (set to 'http://automation.info/') and 'TestSuite' (set to 'testng.xml'). A dropdown menu for 'BROWSER_NAME' is open, showing options: 'Chrome' (which is selected and highlighted with a red box), 'FireFox', and 'IE'. A 'Build' button is visible at the bottom.

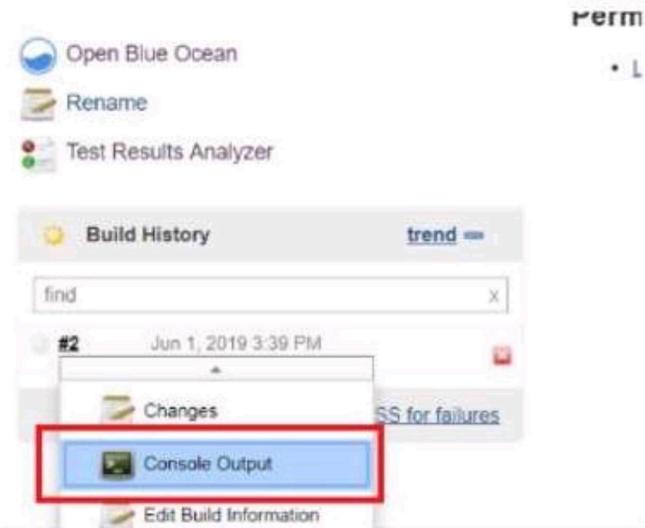
9. Select the TestSuit file.

The screenshot shows the Jenkins dashboard for the 'SeleniumJenkins' project. The 'TestSuite' dropdown menu is expanded, showing options: 'testng.xml', 'TestNG Suite', 'HTML Unit Test', 'JavaScript Test', and 'Selenium Test'. The 'testng.xml' option is highlighted with a red box.



Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

10. Click the build button and go to console output .



11. See the logs from **Console Output** window.

12. View the html report just click on the link



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Jenkins

Jenkins > SeleniumJenkins >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Delete Maven project
- Configure
- Modules
- HTML Report**
- Open Blue Ocean
- Rename
- Test Results Analyzer

Maven project SeleniumJenkins

HTML Report

Workspace

Recent Changes

Latest Test Result (no failures)

Latest Test Result (no failures)

Build History

trend =

Last build (#2) - 3 min.43 sec.ago

Test results

All suites: qautomation.TestScript

Suite: C:\AutomationTesting\Scripts\WorkBench\SeleniumScript\testng.xml

Test applications: C:\AutomationTesting\Scripts\WorkBench\SeleniumScript\testng.xml

Test results: 100% (100%)

Test methods: 100% (100%)

Test classes: 100% (100%)

Groups for Suite: 100% (100%)

Times for Suite: Total running time: 10 seconds

Reporter output for Suite: qaautomation.TestScript

Ignored methods: 0% (0%)

Tests For Suite: Test class

Groups for Suite:

Times for Suite:

Reporter output for Suite:

Ignored methods:

13. Click Test Analyzer to analyse the result.

Jenkins

Jenkins > SeleniumJenkins > Test Results Analyzer >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Delete Maven project

Options Download Test (CSV) Search TestClass/Package

Chart	Package/Class/Testmethod	Passed	Transitions	2
qautomation	100% (100%)	0	PASSED	
TestScript	100% (100%)	0	PASSED	
TestApplication	100% (100%)	0	PASSED	

Build details for all

Build Status

Passed: 0.0%

Skipped: 0.0%

Passed: 100.0%

Build outcome: Build no: 2
Passed: 1
Failed: 0
Skipped: 0

No of tests: 2

Legend: Skipped (Yellow), Failed (Red), Passed (Green)



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Conclusion:

Q1. Which browsers are supported by selenium webdriver?

Q2. What are some features of selenium 4?