Experiment No.7
To Setup and Run Selenium Tests in Jenkins Using Maven.
Date of Performance:
Date of Submission:

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.

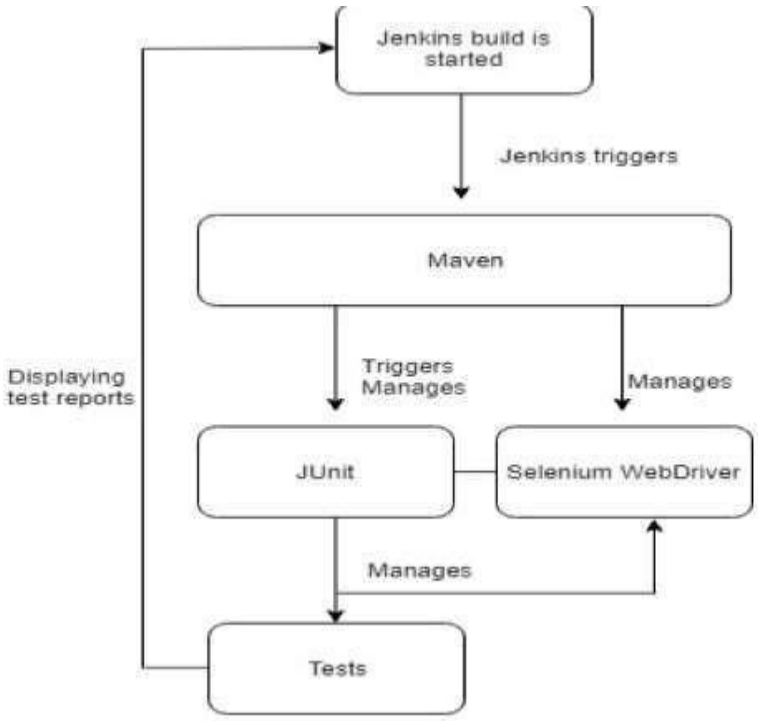
Theory:

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.



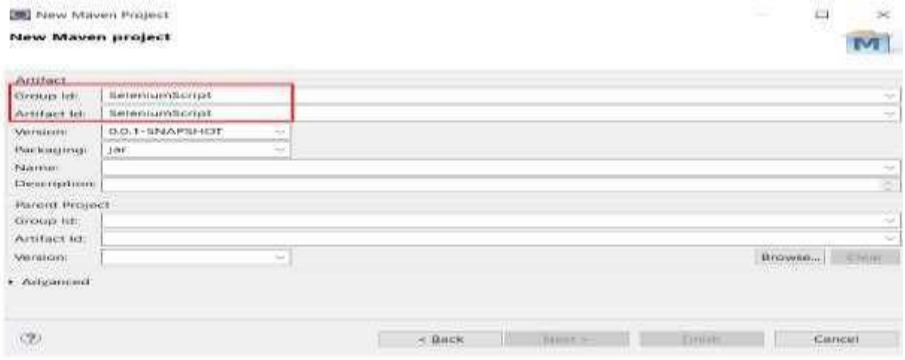
Steps:

---Create a Maven Selenium script---

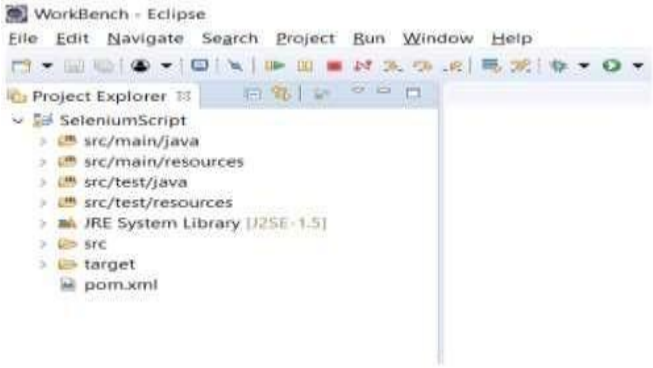
1. In Eclipse IDE, create a new project by selecting **File | New | Maven Project** from Eclipse menu.

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

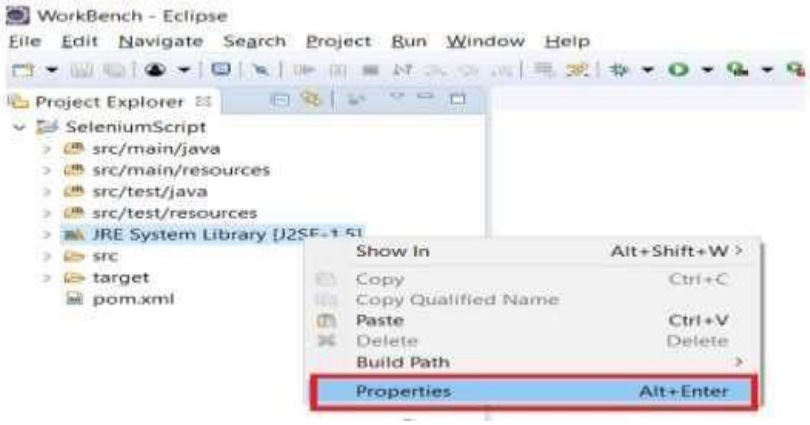
3. Enter **SeleniumScript** in **Group Id**: and **Artifact Id**: and click finish



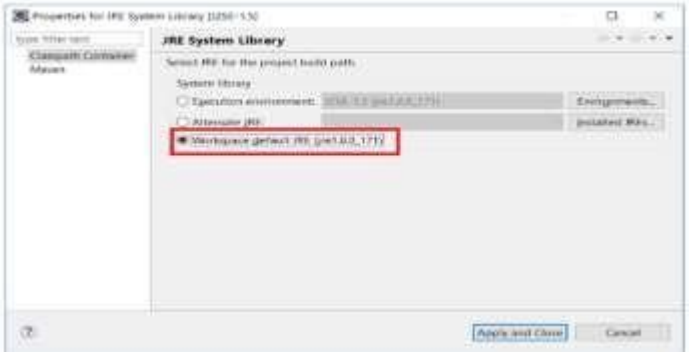
4. Eclipse will create webdriverTest.



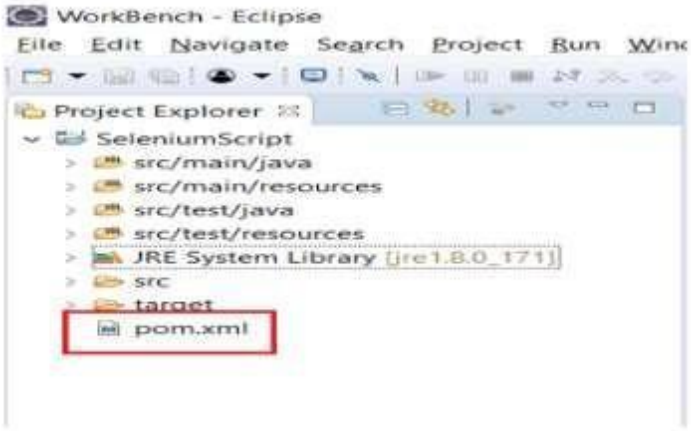
5. Right click on JRE System Library and select the Properties option from the menu.



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



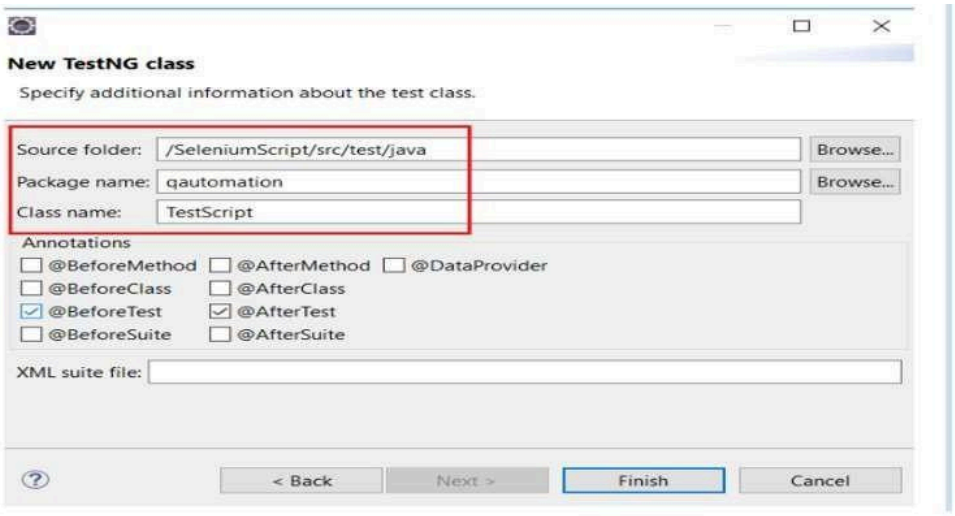
7. Select pom.xml from project explorer.



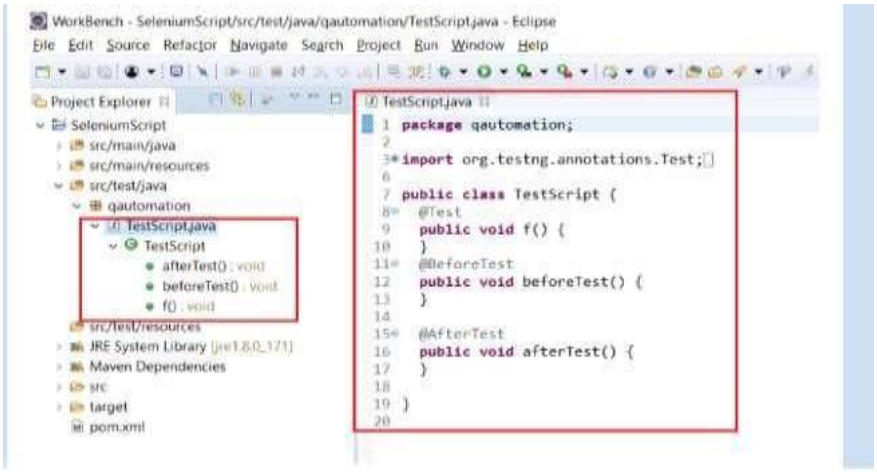
8. Add selenium, Maven, TestNG, Junit dependencies to pom.xml in the code.

```
<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.



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;
```

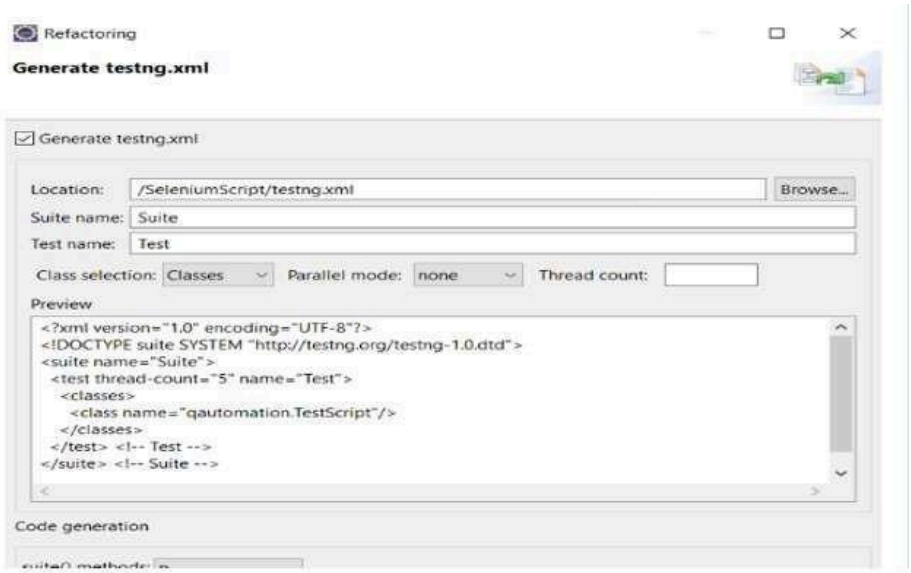
```
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
public class TestScript {
        public static WebDriver driver=null;
        public String browser = System.getProperty("browser");
        public String url = System.getProperty("URL");

 @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();
```

```
                                        caps.setCapability(InternetExplorerDriver.INTR
                                        ODUCE_FLAKINESS_BY_IGNO
                                        RING_SECURITY_DOMAINS,true);
                                        caps.setCapability(InternetExplorerDriver.IGNO
                                        RE_ZOOM_SETTING,true);
                                        caps.setCapability(InternetExplorerDriver.UNEX
                                        PECTED_ALERT_BEHAVIOR," accept");
                                        caps.setCapability(InternetExplorerDriver.REQU
                                        IRE_WINDOW_FOCUS,true);
                                        caps.setCapability(InternetExplorerDriver.INITIA
                                        L_BROWSER_URL,"http://www. google.com/");
                                        driver = new InternetExplorerDriver(caps);
                                        driver.manage().timeouts().implicitlyWait(20,
                                        TimeUnit.SECONDS);
    }                                   driver.manage().window().maximize();
    @Test                               }
                                        driver.manage().timeouts().implicitlyWait(20,
                                        TimeUnit.SECONDS);
                                        driver.manage().window().maximize();
    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.

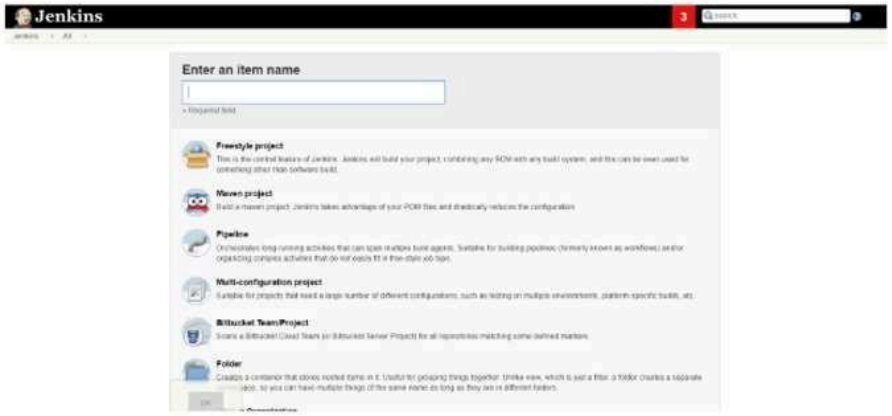


13. Adding dependencies and plugins Additionally we need to add
    1. Maven-compiler-plugin
    2. Maven-surefire-plugin
    3. Testng.xml

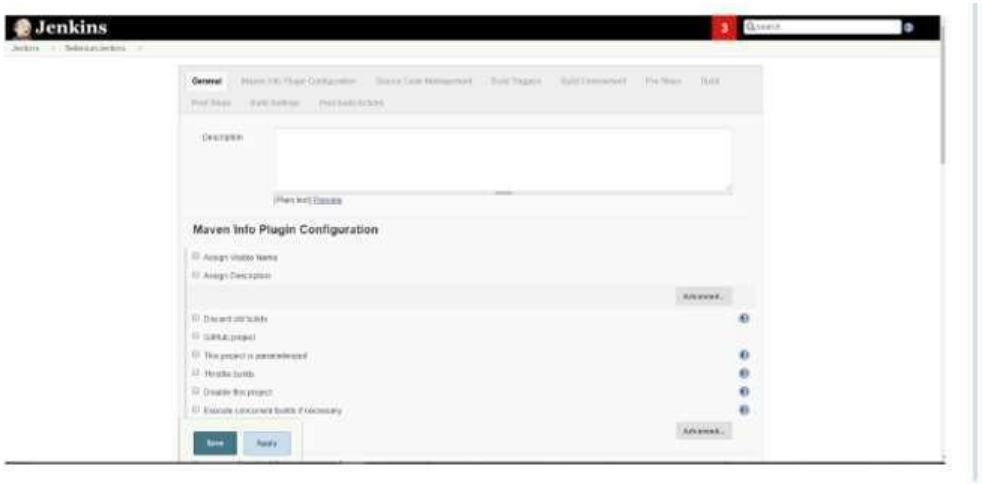
---------Integrating your test to Jenkins----------

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

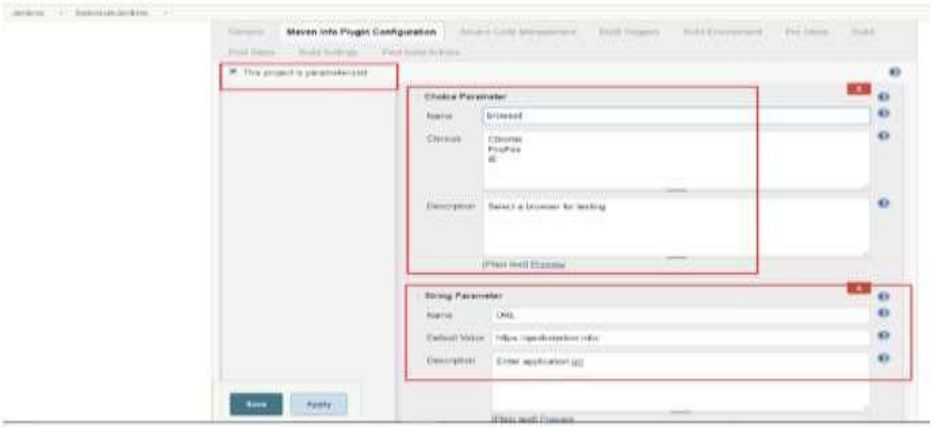
2. Click on new item and enter an appropriate name for the new job , select Maven Project and click on save.



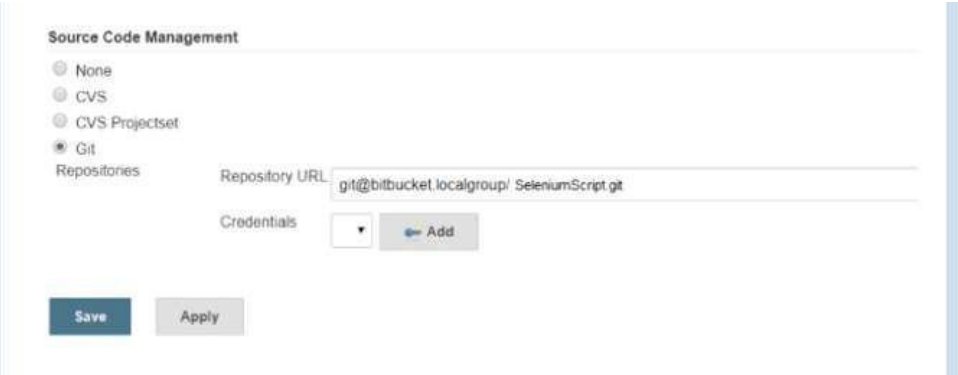
3. A new empty job has been created at this point.



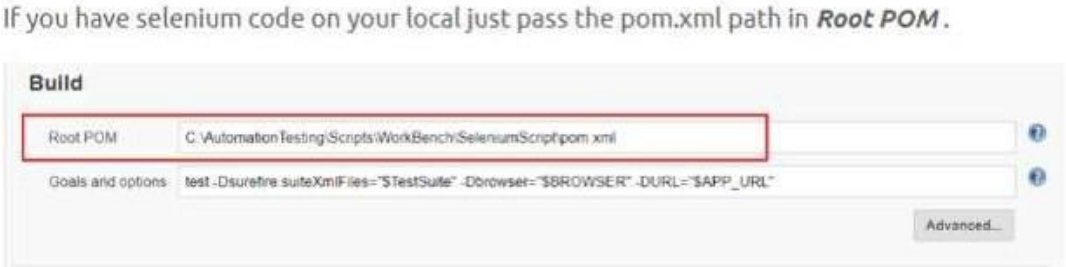
4. Jenkins Parameterized Build in Jenkins just check the checkbox **This project is paramerized** and add the parameter by **Add Parameter** as per your project requirement.



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.

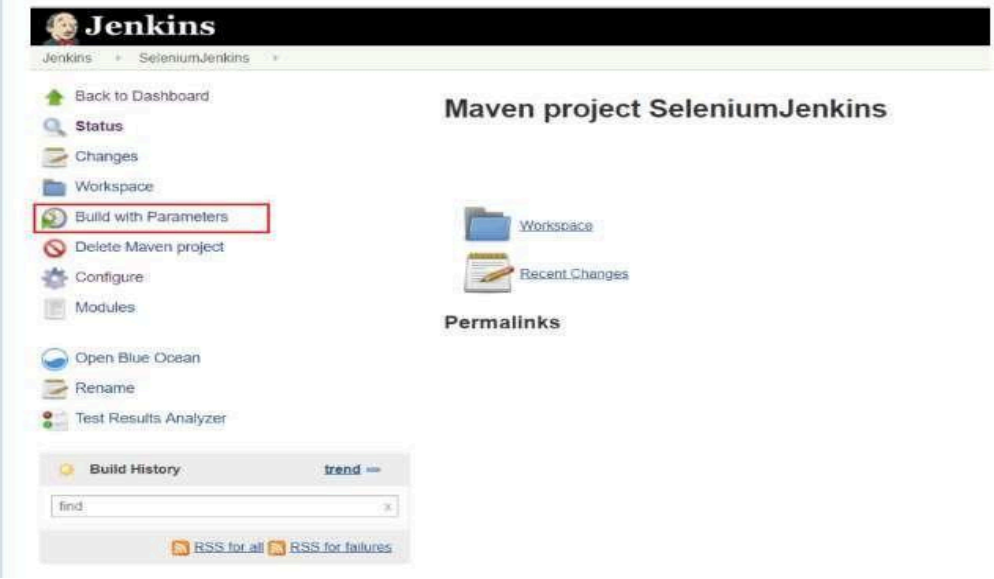


6. In the "pre-steps" build section another set of parameters can be passed to the Jenkins build. Specify the Maven targets that need to be executed in order to run test.

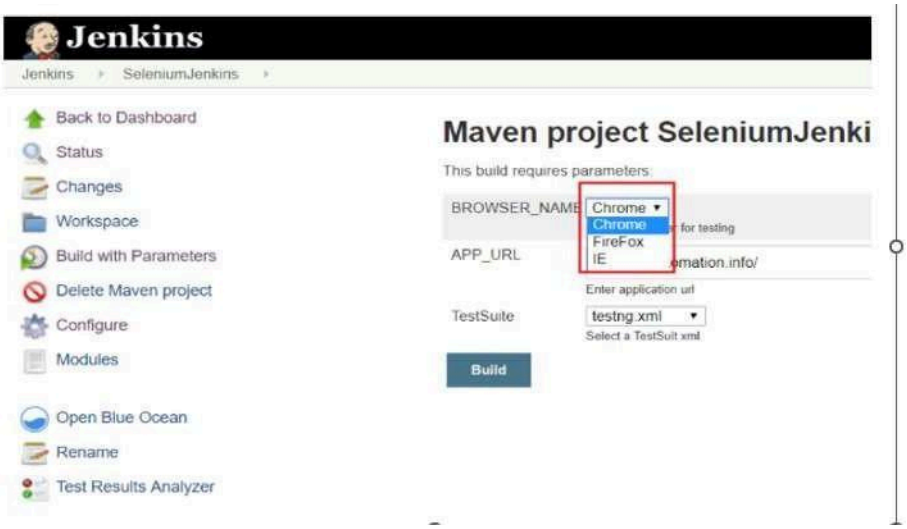


7. Run the test in Jenkins by clicking on Building with Parameters.

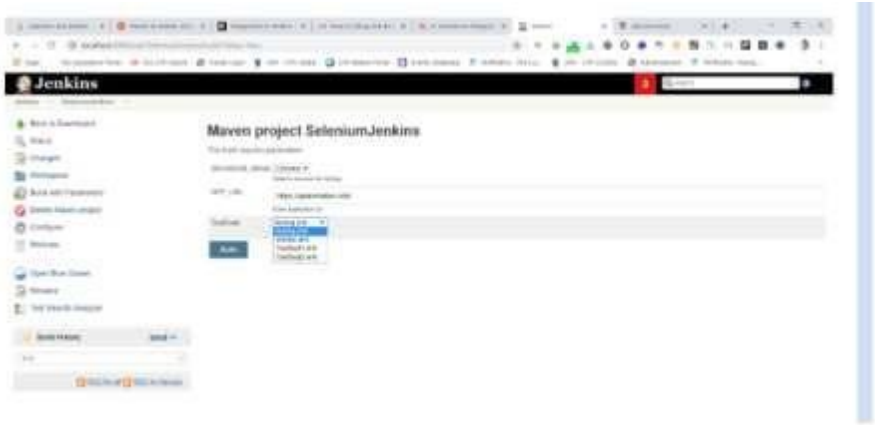
8. Run the test in Jenkins by clicking on ***Build with Parameters*** .



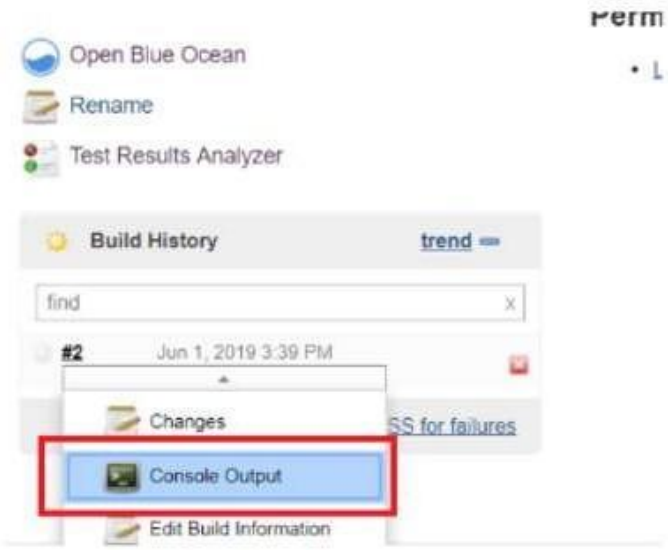
8. Select the browser you want to run from dropdown.



9. Select the TestSuit file.



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



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

Note : Blue color of ball of console output is that build is successful

12. View the html report just click on the link.


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
Build History
trend
Maven project SeleniumJenkins
HTML Report
Workspace
Recent Changes
Latest Test Result (no failures)
Latest Test Result (no failures)
Permalinks
Last build (#2), 3 min 43 sec ago

2
PASSED
PASSED
PASSED

Build Status


Test results
All suites
Suite
Info
Results
qautomation.TestScript
C:\AutomationTesting\Scripts\WorkBench\SeleniumScript\testng.xml
Tests for Suite
Groups for Suite
Times for Suite
Reporter output for Suite
0 ignored methods

Conclusion:

Q1. Which browsers are supported by selenium webdriver?

Ans: Selenium Webdriver supports all modern web browsers. The following browsers are supported by Selenium WebDriver:

1. Internet Explorer
2. Mozilla Firefox
3. Google Chrome
4. Safari
5. Edge
6. Opera

Q2. What are some features of selenium 4?

Ans: -

1. Improved Selenium Grid with better support for Docker, Kubernetes, and cloud-based execution environments.
2. - Introduction of Relative Locators for locating elements based on their proximity to other elements.
3. - Native support for Chrome DevTools Protocol (CDP) integration for advanced browser automation and debugging capabilities.
4. - Enhanced compliance with the W3C WebDriver protocol for better compatibility across different WebDriver implementations and browsers.
5. - Introduction of new APIs and commands for improved window management, JavaScript execution, and handling of iframes and shadow DOM elements.
6. - Continued support for multiple browsers, including Chrome, Firefox, Safari, Edge, and others, for effective cross-browser testing.