**Jenkins + Maven + Selenium (on windows)**

Note: Why I am implementing this on windows? 🡪 In Realtime, most of the people work on the windows machines only. For example, in my home, if I want to book any train ticket in IRCTC website, I may open IE, chrome or Firefox browser in windows10 as my machine installed with Winndows10 operating system. In real scenario, we supposed to configure a **windows slave** to **Jenkins master (master might be either linux or windows)**. So that, selenium will run the automated tests on windows machines & windows compatible browsers. So that, application will work on windows compatible browsers. If the website is error free on windows compatible browsers, then the website will be a quality a product as MOST of the users are working on windows compatible browsers. If you want to test your website on android browsers & safari (for MAC OS) browsers you just need to install the respective browser on windows and test it.

**Required Software**:

1. Install Java – JDK1.8+
2. Maven
3. Notepad++ (just for editing files)
4. All the required browsers should be installed. (IE, Firefox, chrome, safari, android browser etc)
5. Download Selenium server jar file.
6. Tomcat7
7. Jenkins.war

**Practice**:

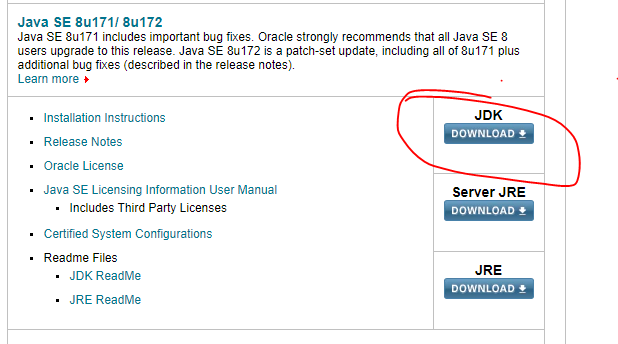
1. Sample selenium code to test the google title in chrome browser. (local maven build & Jenkins build)
2. Sample selenium code to test the google title in ALL the browsers. (local maven build & Jenkins build)
3. Sample web application to test the sample html page title in ALL the browsers. (local maven build & Jenkins build)

Known Issues:

1. Expected Errors & issues.
2. **Install JDK**:

Frist Download & install JDK: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

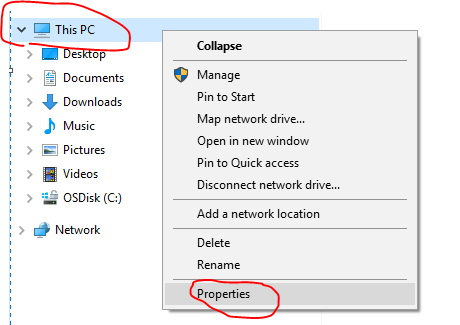
And setup the path in system environment variables.

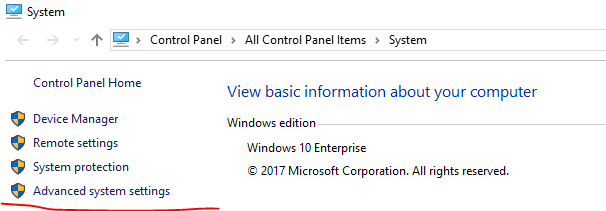


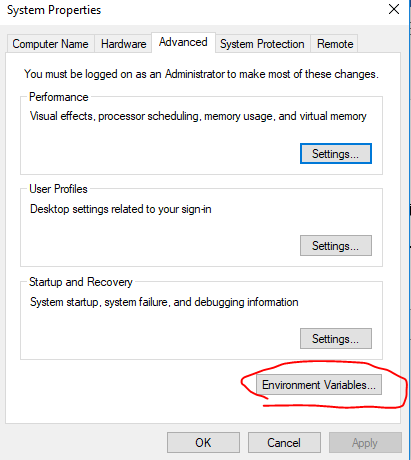
Once the JDK installed, set the PATH under system variables & create a new system variable for JAVA\_HOME.

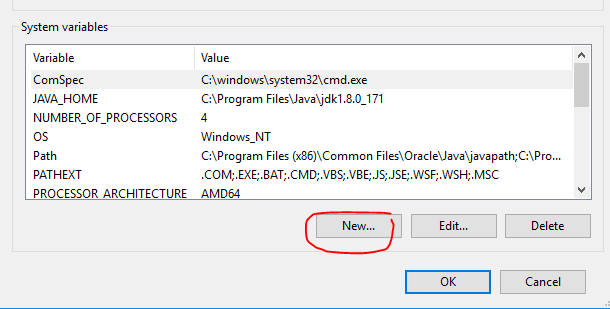
Assume, if the installed location of java is: C:\Program Files\Java\jdk1.8.0\_171

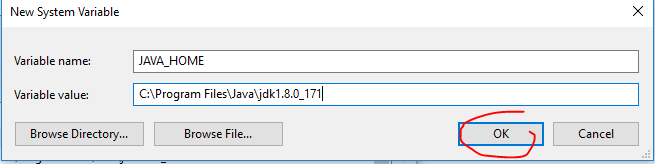
Right click on **This PC** 🡪 Properties 🡪 Advanced System settings 🡪 Environment Variables... 🡪 Click on “New” under the System variables 🡪 Add the Variable Name as “JAVA\_HOME” & value as “C:\Program Files\Java\jdk1.8.0\_171”. Follow the below steps for same process.



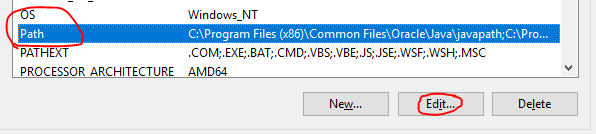


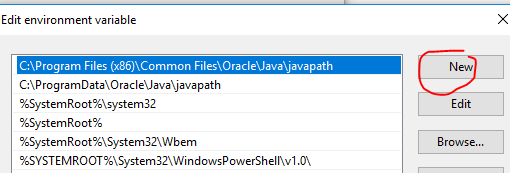






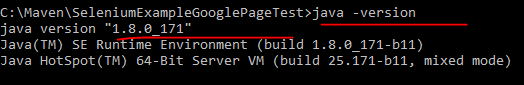
Now add the java class bin path: select path 🡪 edit 🡪 click on “New” button 🡪 add the path “C:\Program Files\Java\jdk1.8.0\_171\bin” 🡪 click on “Ok” for all the three windows. Close all the windows which are opened to setup the system properties.



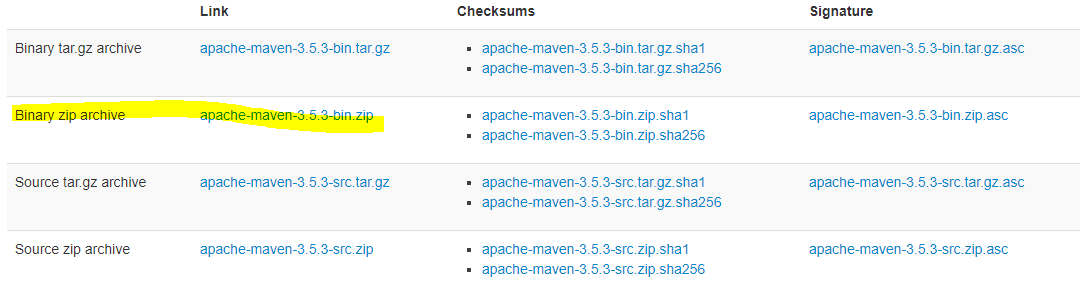




Check whether the java path setup done properly or not. Open a **new** command prompt and type the command “java -version”.



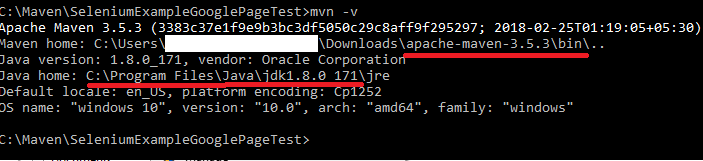
1. **Maven**: <https://maven.apache.org/download.cgi>



Unzip the maven file once you have downloaded the maven zip file. And setup the path in system environment variables.

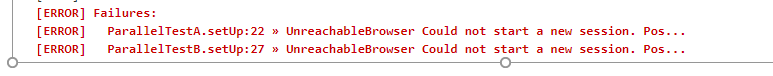
Right click on **This PC** 🡪 Properties 🡪 Advanced System settings 🡪 Environment Variables... 🡪 path 🡪 edit 🡪 click on “New” button 🡪 add the path “C:\Downloads\apache-maven-3.5.3\bin” 🡪 click on “Ok” for all the three windows. Close all the windows opened for system properties.

Check whether the maven setup done properly or not. Open a new command prompt and type the command “mvn -v”.



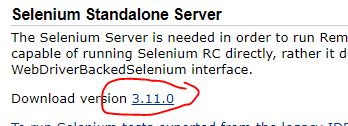
1. Notepad++: <https://notepad-plus-plus.org/download/> download the latest one and install it.
2. Download and install all the required browsers you want to test your app on different browsers. (I have installed all three main Firefox, IE, chrome browsers to test my web app).

If you don’t install the required browsers, the build will be failed as showing error message below.



1. **Selenium Grid setup**: Download the selenium jar from <https://www.seleniumhq.org/download/>
   1. Go to Selenium website: <https://www.seleniumhq.org/download/>

Clock on the latest version of selenium jar.



* 1. Downloading status:



* 1. Folder structure (its optional, you can setup the folder structure in your own way, here I am setting up the file/folder structure as below)

After the selenium jar downloaded, keep the same jar in below two folders ‘hub’, ‘nodes’.

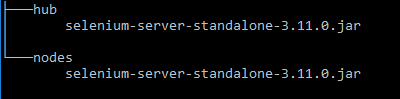
To avoid any confusions, better to create a folder structure as below.



**Parent folder**: selenium

**Sub-folders of ‘selenium’**: hub, nodes

keep the same jar ‘selenium-server-standalone-3.11.0.jar’ in below two folders ‘hub’, ‘nodes’.

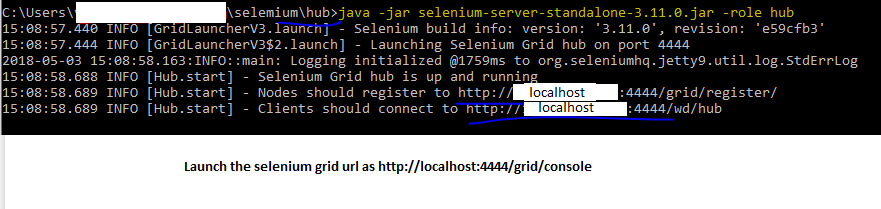


* 1. Configure Selenium Hub:

Go to the folder ‘hub’ and open a command prompt, and then run the below command. Please notice the colored lines.

|  |
| --- |
| java -jar selenium-server-standalone-3.11.0.jar -role hub |

You will see the command prompt as below.



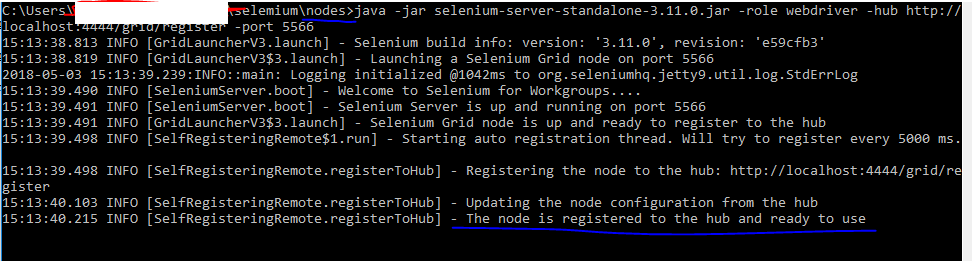
And now, launch <http://localhost:4444/grid/console> in any browser.



* 1. Configure Selenium Nodes:

Open a new command prompt at ‘nodes’ folder and run the below command. We have to pass the hub details to nodes as mentioned in the command.

|  |
| --- |
| java -jar selenium-server-standalone-3.11.0.jar -role webdriver -hub http://localhost:4444/grid/register -port 5566 |



* 1. Selenium Grid view (Hub + Nodes):

Refresh the bowser where you opened the UIRL: <http://localhost:4444/grid/console>



If you observe the above image, nodes are created for each browser, there are 1 instance/node for IE, and 5 instances created for chrome & Firefox by default. You can customize the hub & node configurations as mentioned below.

If you want to configure required number of instances & required browse types like only IE & chrome, you can configure them with the help of json configuration file.

This is just a sample demo, this will not work for automation testing. Please follow the steps to finish the Selenium Grid setup.

Stop the above two server’s hub & nodes by pressing **Ctrl+C** in the command prompt. **Also, we must configure the browser server’s to run the selenium script in different browsers.**

* 1. Customizing HUB Configuration:

The below json script is to customize the hub configuration, so that you can give different port number etc. But here I am keeping the same as 4444 port number. Copy the below content & save the file in json file format. Ex: “DefaultHub.json”.

|  |
| --- |
| {  "port": 4444,  "newSessionWaitTimeout": -1,  "servlets" : [],  "withoutServlets": [],  "custom": {},  "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",  "registry": "org.openqa.grid.internal.DefaultGridRegistry",  "throwOnCapabilityNotPresent": true,  "cleanUpCycle": 5000,  "role": "hub",  "debug": false,  "browserTimeout": 0,  "timeout": 1800  } |

You can refer the below file:



* 1. Customizing Node Configuration:

The below snippet is to customizing node configuration. Copy the below content & save this in json file format. Ex: “DefaultNodes.json”.

If you want to configure the nodes/instances only for **IE & Chrome** browsers, then configure those two browsers details only in the json file. (If you want the other browses, you can configure them, see below **next steps** how we can configure for Firefox).

|  |
| --- |
| {  "capabilities":  [  {  "browserName": "chrome",  "maxInstances": 5,  "seleniumProtocol": "WebDriver"  },  {  "browserName": "internet explorer",  "platform": "WINDOWS",  "maxInstances": 5,  "seleniumProtocol": "WebDriver"  }  ],  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",  "maxSession": 5,  "port": 5555,  "register": true,  "registerCycle": 5000,  "hub": "http://localhost:4444",  "nodeStatusCheckTimeout": 5000,  "nodePolling": 5000,  "role": "node",  "unregisterIfStillDownAfter": 60000,  "downPollingLimit": 2,  "debug": false,  "servlets" : [],  "withoutServlets": [],  "custom": {}  } |

You can refer the below file:



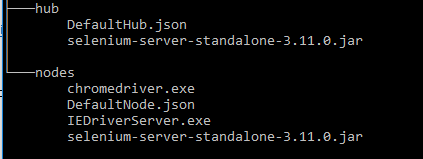
* 1. Selenium GRID setup with browsers server’s configuration:

Download the below zip file and unzip them and keep both the exe files under ‘nodes’ folder.

<http://selenium-release.storage.googleapis.com/3.4/IEDriverServer_x64_3.4.0.zip>

<https://chromedriver.storage.googleapis.com/2.38/chromedriver_win32.zip>

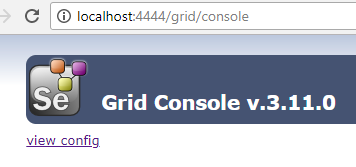
Download the above zip file and unzip them and keep both the exe files under ‘nodes’ folder. Find the file structure below, it should be similar as below.



Now, run the below two commands to setup the Selenium grid.

**Hub**: java -jar selenium-server-standalone-3.11.0.jar -role hub -hubConfig DefaultHub.json

**Node**: java -Dwebdriver.chrome.driver="chromedriver.exe" -Dwebdriver.ie.driver="IEDriverServer.exe" -jar selenium-server-standalone-3.11.0.jar -role node -nodeConfig DefaultNode.json

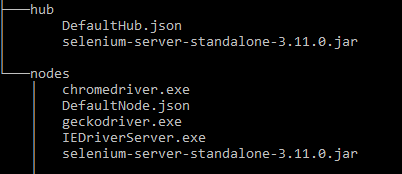




* 1. Configure nodes for Firefox:

If you want Firefox also, add the below snippet to **DefaultNode.json** file. Also, you must download the Firefox server file.

**FINAL** file structurershould be similar as below (if you configure any other browser like safari or opera it will be changed, you need to download opera server & need to update the DefaultNode.json file with opera browser details).



<https://github.com/mozilla/geckodriver/releases/download/v0.20.1/geckodriver-v0.20.1-win64.zip>

|  |
| --- |
| {  "browserName": "firefox",  "marionette": true,  "maxInstances": 5,  "seleniumProtocol": "WebDriver"  } |



Stop the node and run the command again and then refresh the browser or launch the URL again.

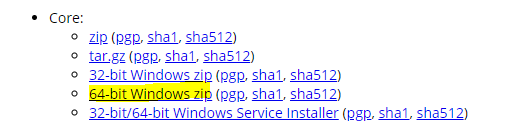
java -Dwebdriver.gecko.driver="geckodriver.exe" -Dwebdriver.chrome.driver="chromedriver.exe" -Dwebdriver.ie.driver="IEDriverServer.exe" -jar selenium-server-standalone-3.11.0.jar -role node -nodeConfig DefaultNode.json



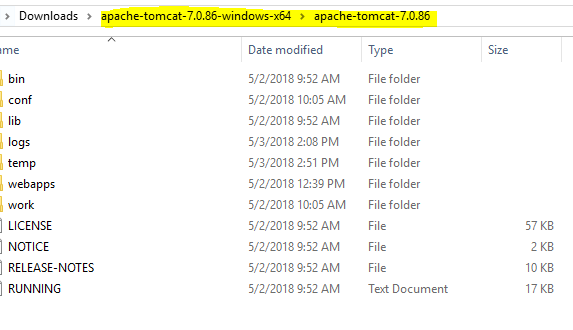
Note: If you want, you can reduce the number of instances count by updating "maxInstances": 3.



1. **Download & configure Tomcat7**: <https://tomcat.apache.org/download-70.cgi>
   1. Download the tomcat7 zip file as per your system type:

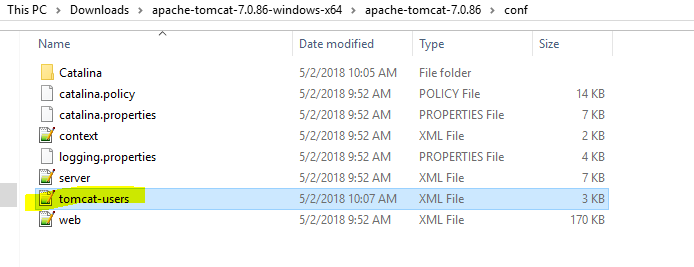


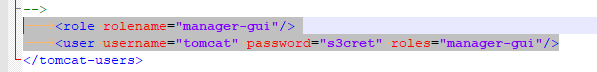
* 1. Unzip the zip file:



* 1. Update the tomcat-users with below user details:

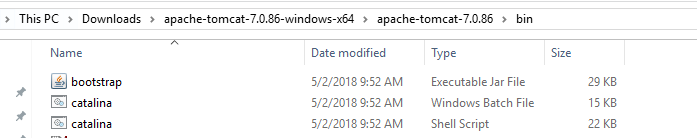
|  |
| --- |
| <role rolename="manager-gui"/>  <user username="tomcat" password="s3cret" roles="manager-gui"/> |





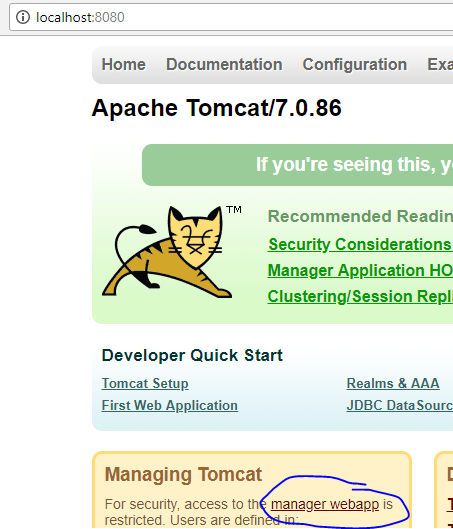
* 1. Start Tomcat:

Open command prompt at bin folder and start the tomcat server by run the bat file “startup.bat”

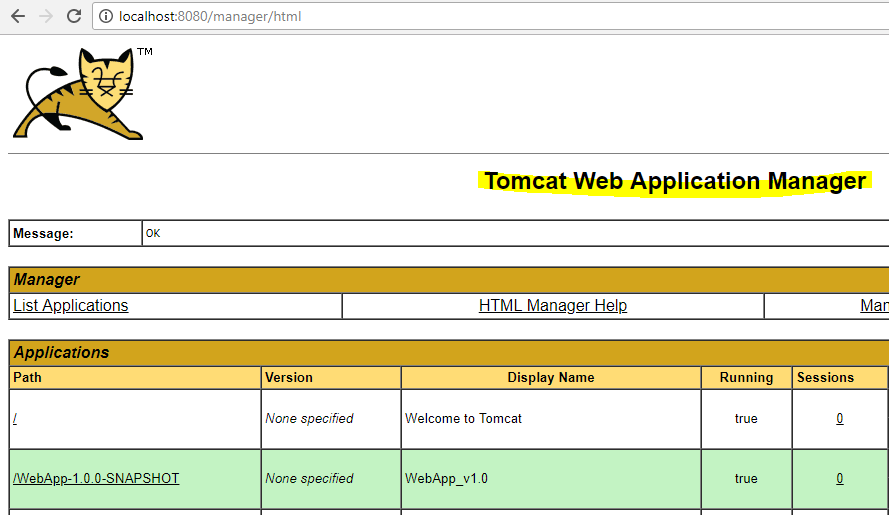




Launch the URL: <http://localhost:8080> & click on ‘manager webapp’, it will prompt a window to enter the username & pwd username="tomcat" password="s3cret" as we configured in tomcat-users.xml file in the step 4.3.



Once the authentication successful, it will navigate to “Tomcat Web Application Manager”, there you will find all the deployed WAR files (web applications), now we are going to deploy our Jenkins.war file. First, we have to download “Jenkins.war” file.



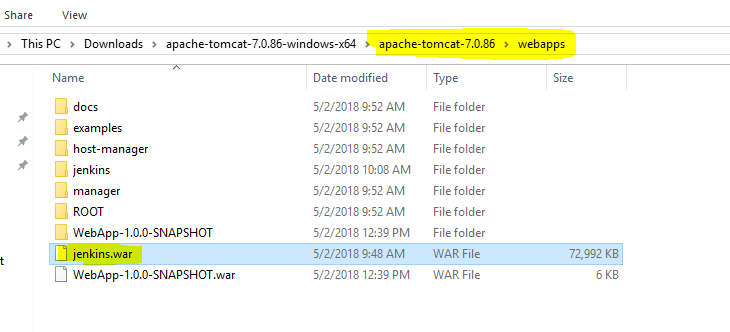
1. **Jenkins setup**:
   1. Download Jenkins.war: <https://updates.jenkins-ci.org/download/war/>

Download the latest war file.

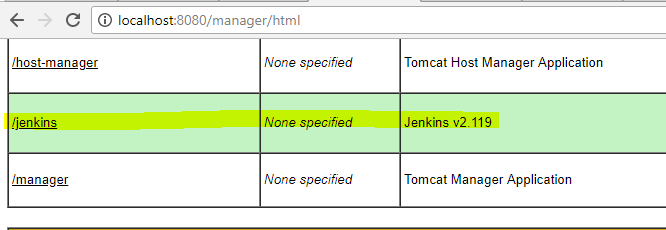


* 1. Deploy to Tomcat:

Copy the Jenkins.war file & paste it in <tomcatLocation>/webapps.



* 1. Restart the tomcat server & refresh the manager webapp page:



Click on Jenkins🡪 it will ask you to copy and paste the initialAdminPassword, the file which is in C://users/yourusername/.jenkins/secrets/initialAdminPasswod. Open the file “initialAdminPassword” with noidepad or notepad++ and copy and then paste the pwd in window.

Choose the first option as to install the default plugins 🡪 Signup with the username & pwd 🡪 continue 🡪 start Jenkins.

1. **Sample selenium code to test the google title in chrome browser. (local maven build & Jenkins build)**:
   1. Local Maven build:

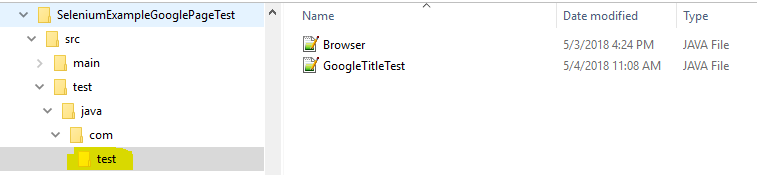
Now, let’s test the web page [www.google.com](http://www.google.com) title.

Scenario is: Print the [www.google.com](http://www.google.com) page title & if the page title mis-match with expected title ‘Google’, then the test case should be failed.

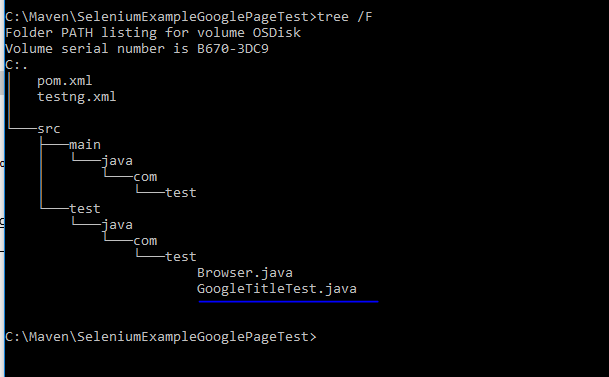
* + 1. Selenium Script & file structure:
       1. See the red highlighted line, we are passing the google web site for testing [www.google.com](http://www.google.com).
       2. See the yellow highlighted line, getting the title of the google web page.
       3. See the pink highlighted line, printing the title of the google web page.
       4. See the green highlighted line, comparing the actual google page title & expected google page title. If this is mismatch build will be failed due to test cases.

|  |
| --- |
| package com.test;  import java.net.MalformedURLException;  import org.openqa.selenium.By;  import org.openqa.selenium.remote.RemoteWebDriver;  import org.testng.Assert;  import org.testng.annotations.AfterClass;  import org.testng.annotations.BeforeClass;  import org.testng.annotations.Parameters;  import org.testng.annotations.Test;  public class GoogleTitleTest {  public static RemoteWebDriver driver;  public static String appURL = "http://www.google.com";  @BeforeClass  @Parameters({ "browser" })  public void setUp(String browser) throws MalformedURLException {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  driver = Browser.getDriver(browser);  driver.manage().window().maximize();  }  @Test  public void testGooglePageTitleInIE() {  driver.navigate().to("http://www.google.com");  String strPageTitle = driver.getTitle();  System.out.println("Title of the Google page: "+strPageTitle);  Assert.assertTrue(strPageTitle.equalsIgnoreCase("Google"), "Page title doesn't match");  }  @AfterClass  public void tearDown() {  if(driver!=null) {  System.out.println("Closing the browser");  driver.quit();  }  }  } |

Copy & save the above code in the location “**src/test/java/**com/test/GoogleTitleTest.java”





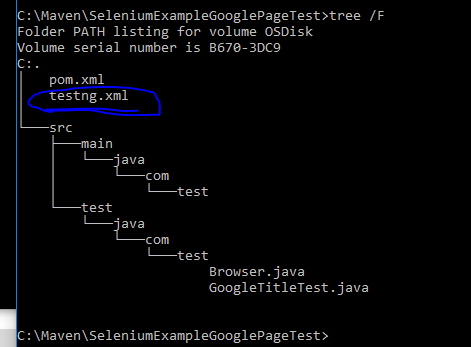


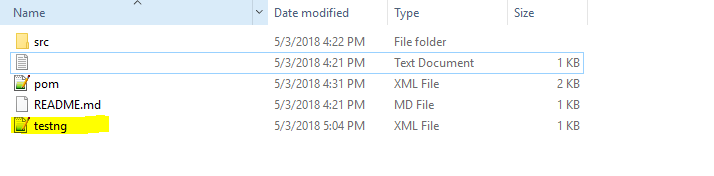
* + 1. Configure testng.xml file:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >  <suite name="Main Test Suite" parallel="tests" verbose="1">  <test name="Grid chrome Test">  <parameter name="browser" value=" chrome"/>  <classes>  **<class name="com.test.GoogleTitleTest"/>**  </classes>  </test>  </suite> |



Save the above snippet in the file “testing.xml” in root folder level. i.e., under the project name along with **src, pom.xml**.

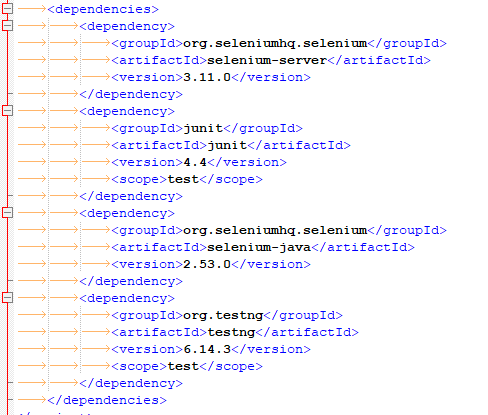




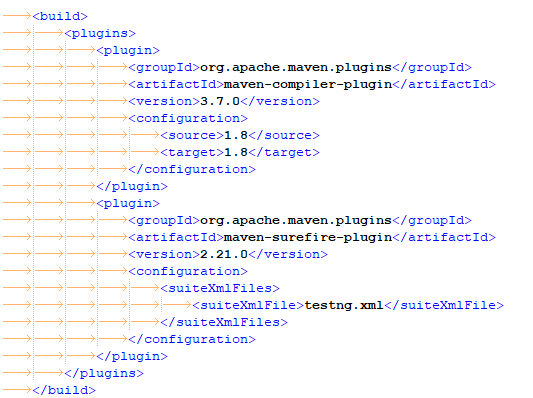
**Browser.java** file will read this testing.xml file & read the line **<parameter name="browser" value=" chrome"/>**, based on this parameter & value it will load the respective browser. Here we are testing it on chrome. Copy & save the below code in the location “**src/test/java/**com/test/Browser.java”.

|  |
| --- |
| package com.test;  import java.net.MalformedURLException;  import java.net.URL;  import org.openqa.selenium.remote.DesiredCapabilities;  import org.openqa.selenium.remote.RemoteWebDriver;  public class Browser {  public static RemoteWebDriver getDriver(String browser) throws MalformedURLException {  return new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), getBrowserCapabilities(browser));  }  private static DesiredCapabilities getBrowserCapabilities(String browserType) {  switch (browserType) {  case "firefox":  System.out.println("Opening firefox driver");  return DesiredCapabilities.firefox();  case "chrome":  System.out.println("Opening chrome driver");  return DesiredCapabilities.chrome();  case "IE":  System.out.println("Opening IE driver");  return DesiredCapabilities.internetExplorer();  default:  System.out.println("browser : " + browserType + " is invalid, Launching Firefox as browser of choice..");  return DesiredCapabilities.firefox();  }  }  } |

* + 1. Configure pom.xml with required dependencies & plugins:
       1. Configure required dependencies**:**



* + - 1. Configure plugins:

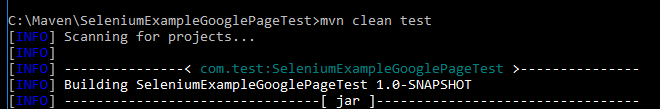




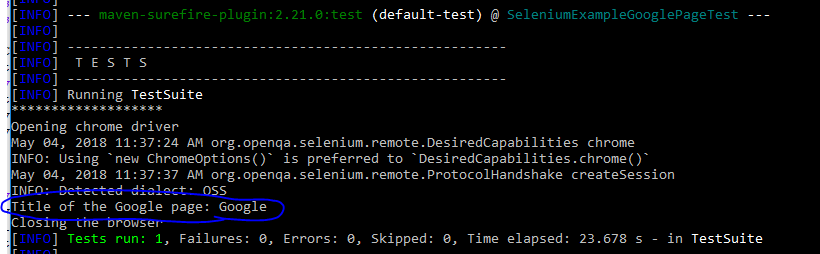
The complete code is available at <https://github.com/DIGITALAPPLICATION/SeleniumGooglePageTitleTest.git>

* 1. Run the maven build: mvn clean test
     1. **Successful build**: First let’s see the successfully running automated selenium test case.

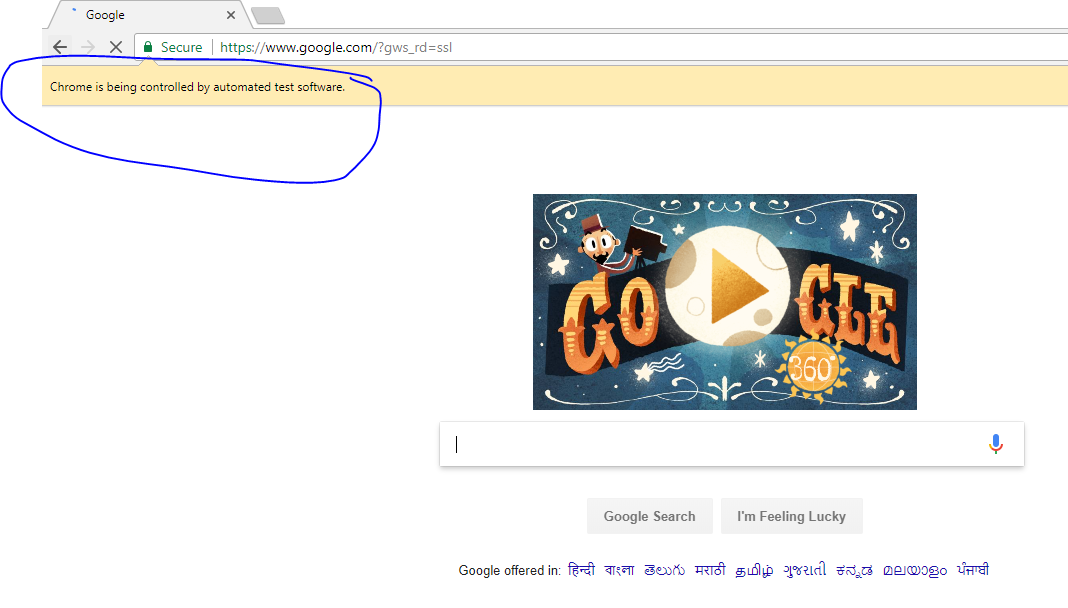
Build started. Automatically, selenium will open the browsers and launch the google website.



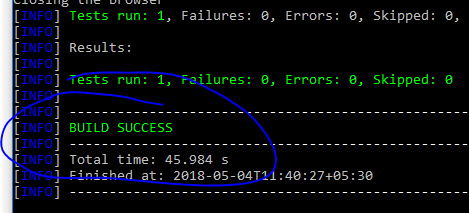
See the build console, it prints the google page title.



Google page loaded:



Build success.



* + 1. Fail build: Now, let’s see the fail build. Update the expected title name as “GoogleSearchPage” in the jave file “GoogleTitleTest.java” as shown below.

@Test

public void testGooglePageTitleInIE() {

driver.navigate().to("http://www.google.com");

String strPageTitle = driver.getTitle();

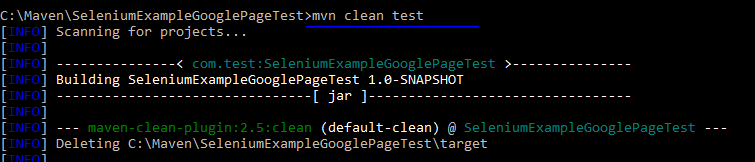
System.out.println("Title of the Google page: "+strPageTitle);

Assert.assertTrue(strPageTitle.equalsIgnoreCase("**GoogleSearchPage**"), "Page title doesn't match");

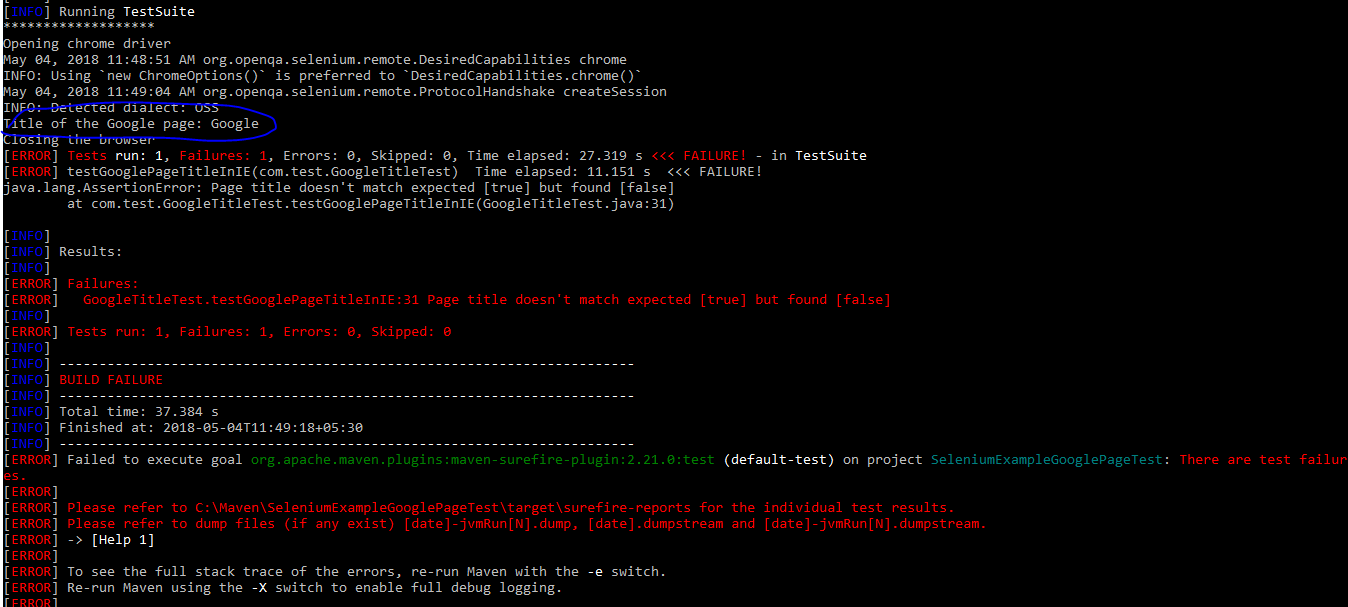
}

Run the build now, mvn clean test

Build is running.



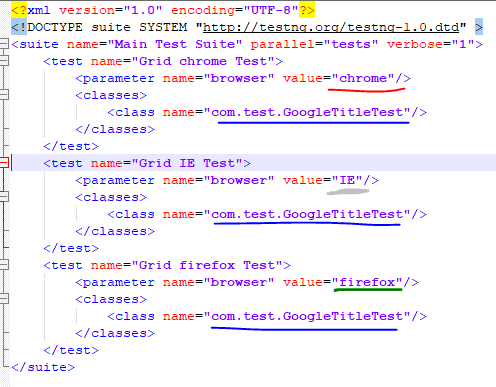
“Google” is the actual value of the google page title. But we are comparing the expected value “**GoogleSearchPage**”, then build is failing due to the test failures.



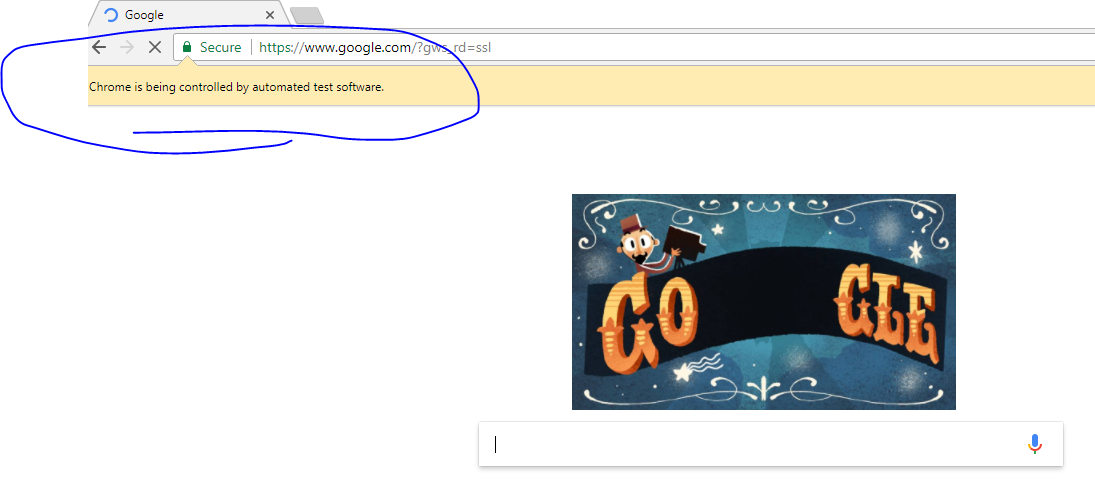
This is the way we have to test the website in the browsers with selenium automated script.

1. Sample selenium code to test the google title in ALL the browsers. (local maven build & Jenkins build): Test the successful code on all three browsers IE, chrome, Firefox.

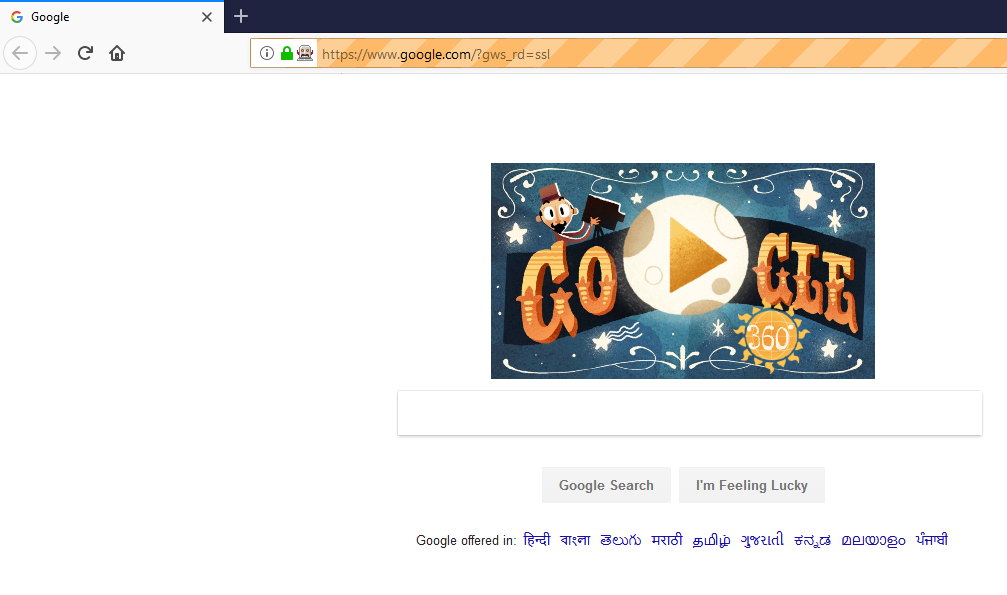
Update the testing.xml file as below. The same test case, GoogleTitleTest.java will run parallelly on different browsers. i.e., we are testing the same website on different browsers to check the compatibility.



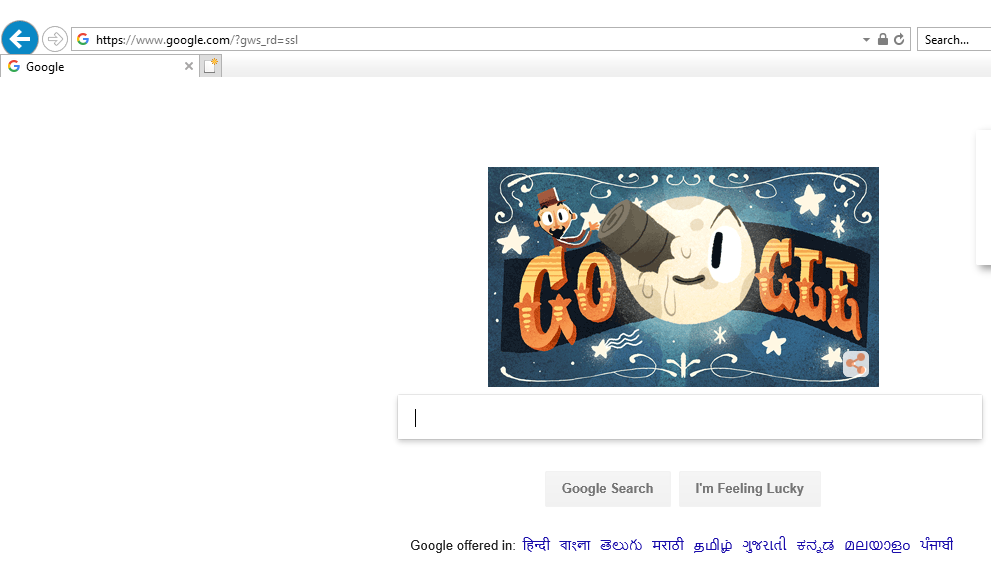
Tested on chrome browser:



Tested on Firefox browser:

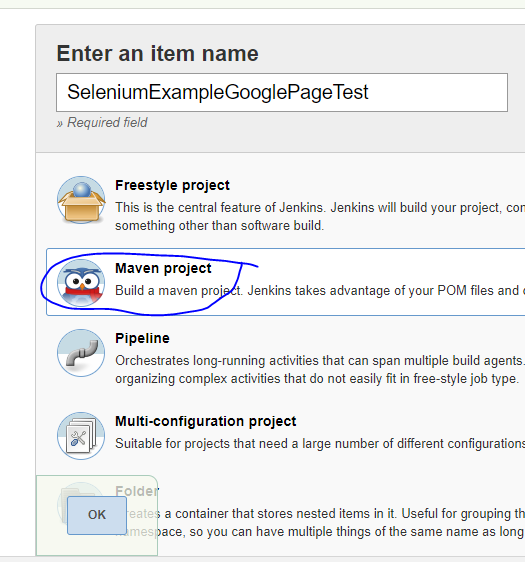


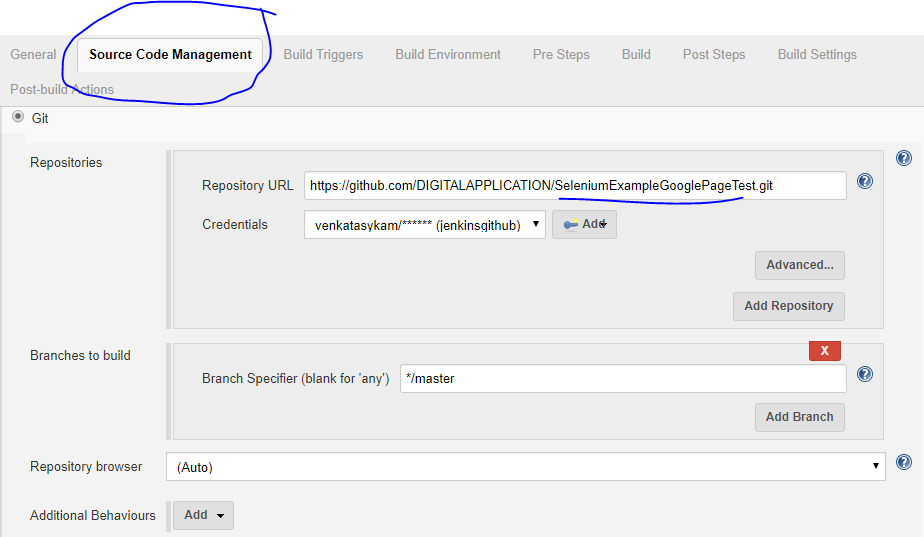
Tested on IE browser: (IE printing the WebDriver as a title of the google page, not sure why it is coming, because of this build is failing)

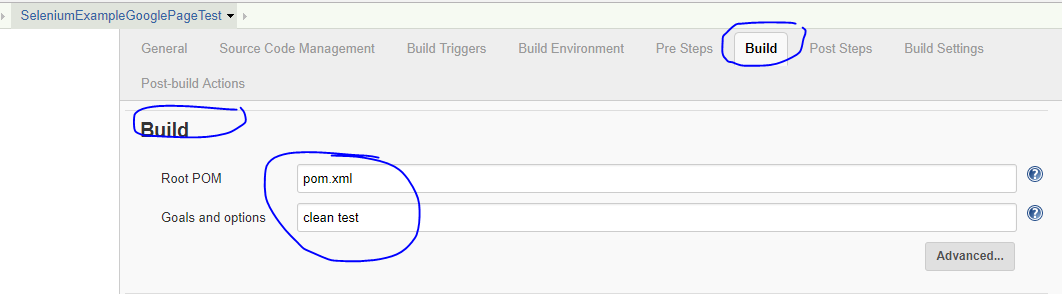


* 1. Jenkins job: Build will be executed as same as above, but we are configuring the Jenkins job for this.

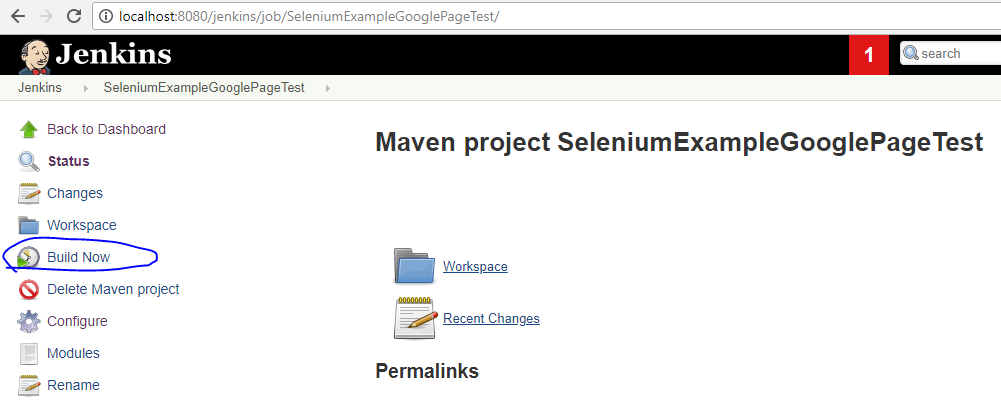
Create a new maven project job & configure SCM, build goals as shown below.



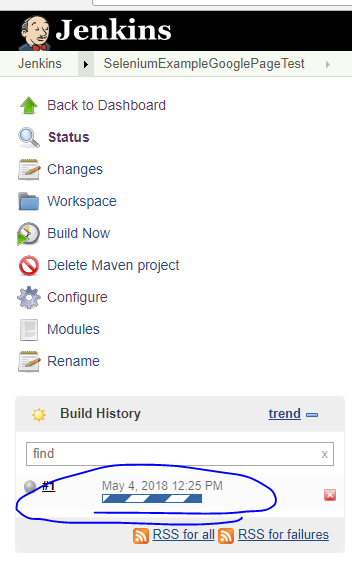


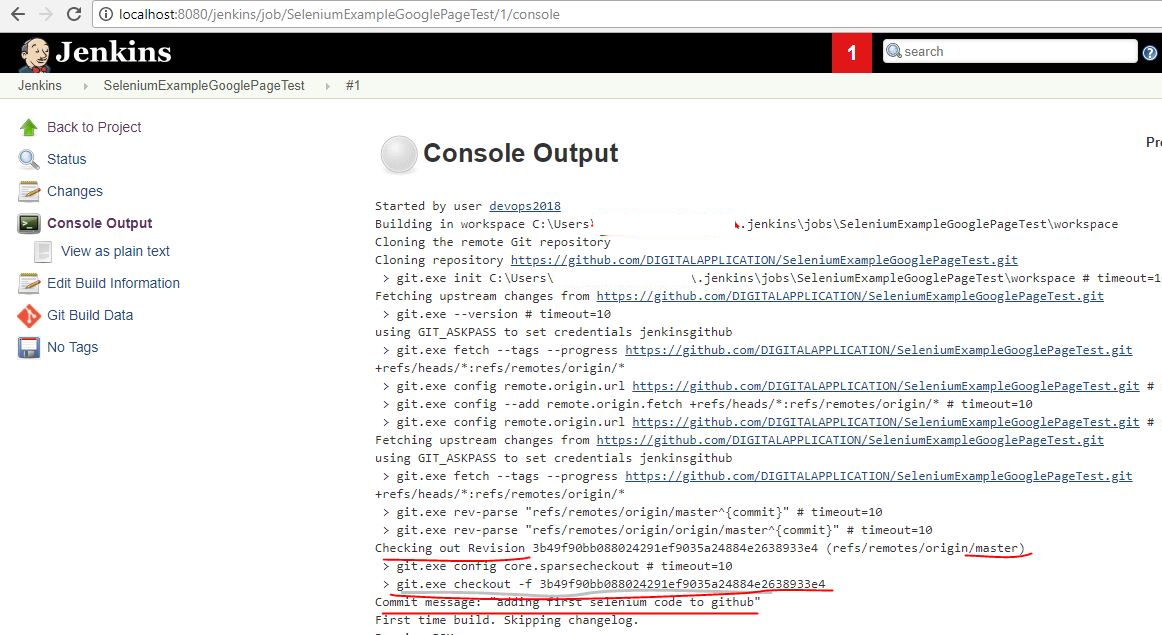


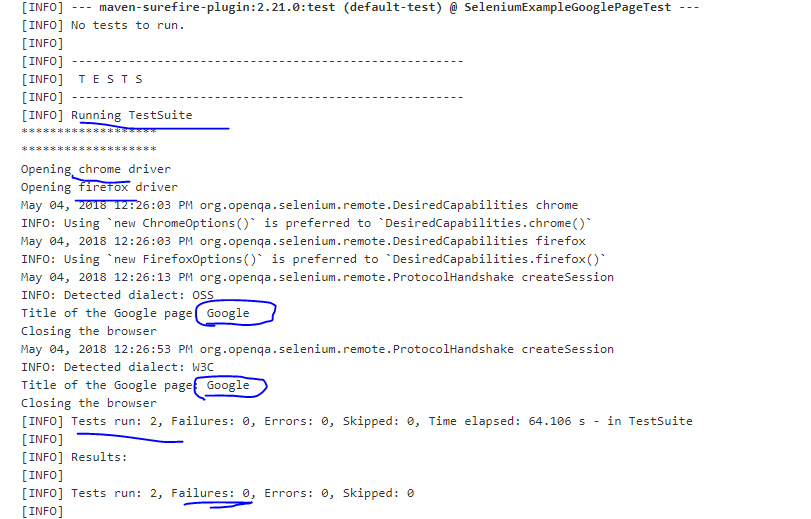
Save & trigger the job.

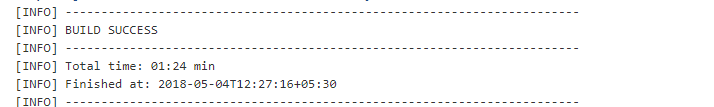


Build is running:







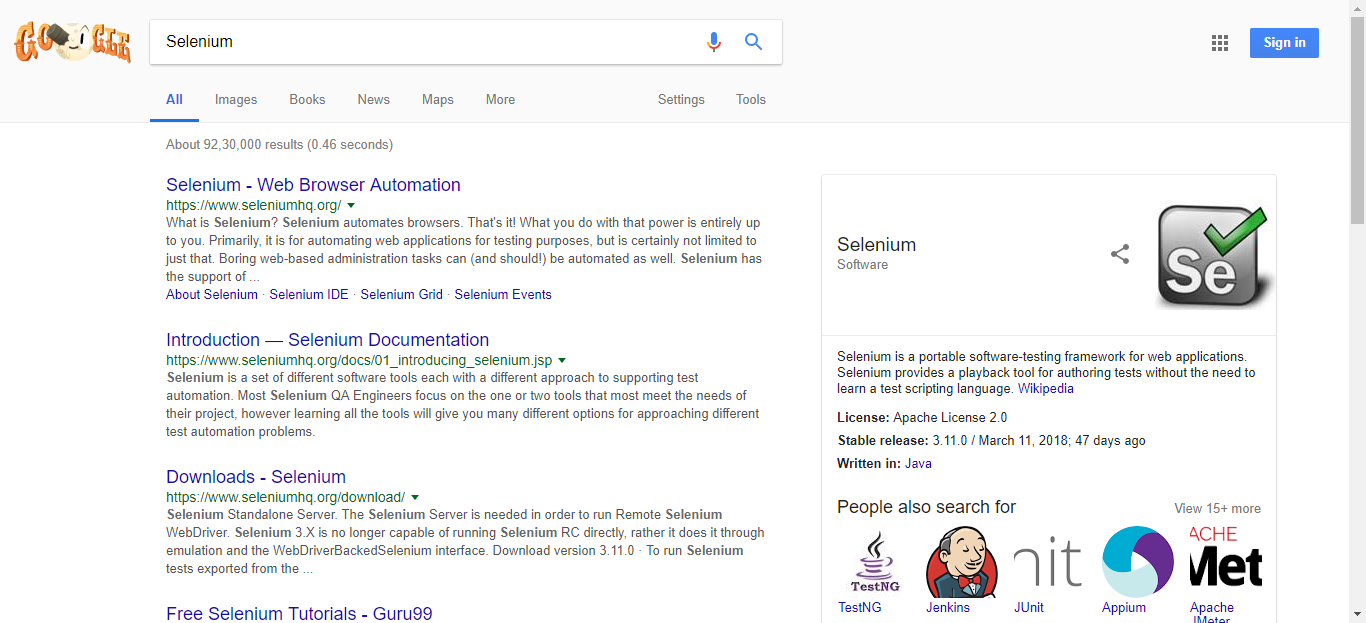


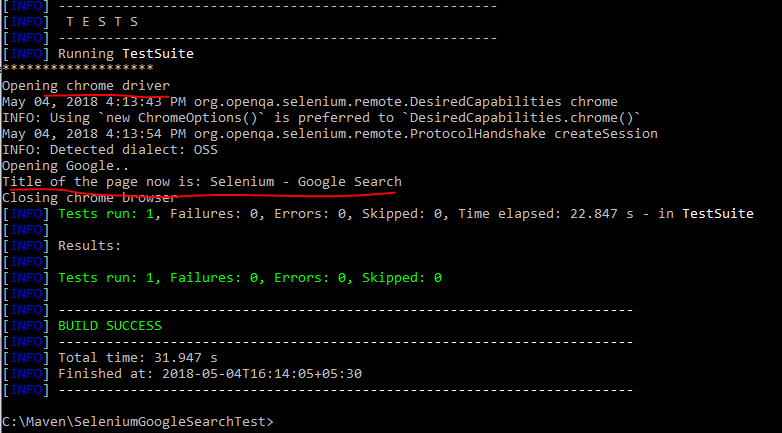
* 1. Google search with the word “Selenium” & take screenshot. See the below yellow lines, searhing for the work “Selenium” & take screenshot. Complete code is available at <https://github.com/DIGITALAPPLICATION/SeleniumGoogleSearchTest.git>.

|  |
| --- |
| @Test  public void testSearchGoogle() throws Exception{  System.out.println("Opening Google...");  driver.navigate().to(“http://www.google.com”);  WebElement java = driver.findElement(By.name("q"));  java.sendKeys("Selenium");  java.submit();  System.out.println("Title of the page now is: " + driver.getTitle());  this.takeSnapShot(driver, "Gsearch.png") ;  } |



Screenshot taken to Gsearch.png: as shown below

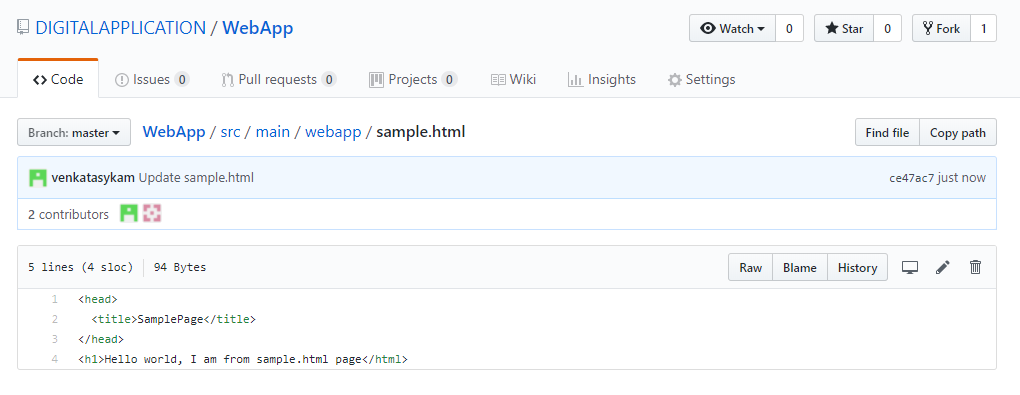




1. Sample web application to test the sample html page title, and tag in ALL the browsers. (local maven build & Jenkins build):
   1. Local build:

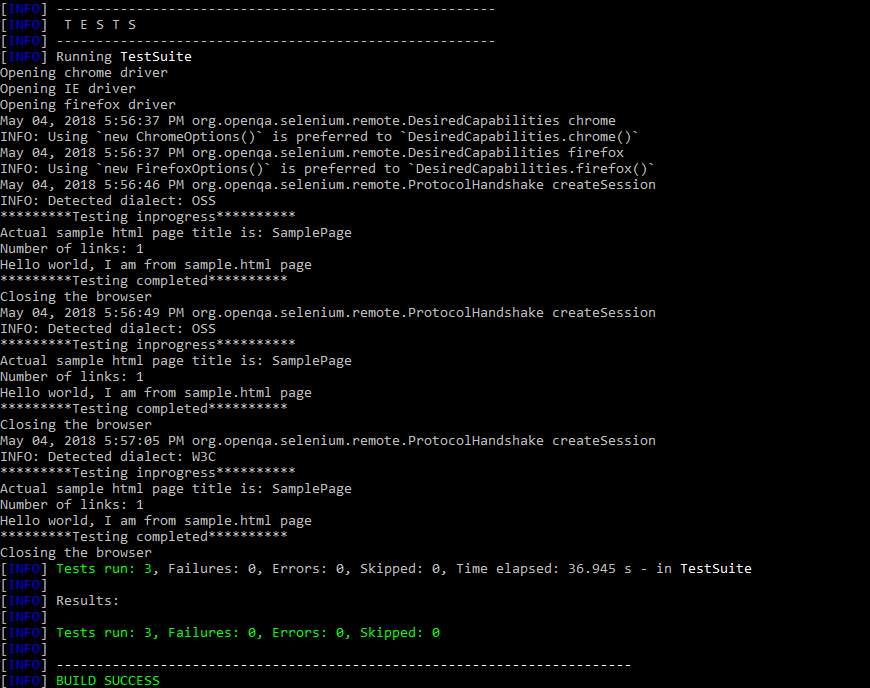
Here is my **sample.html** webpage and I have built it and deployed on my local tomcat server. See the source code [WebApp.git](https://github.com/DIGITALAPPLICATION/WebApp.git).

<https://github.com/DIGITALAPPLICATION/WebApp/blob/master/src/main/webapp/sample.html>





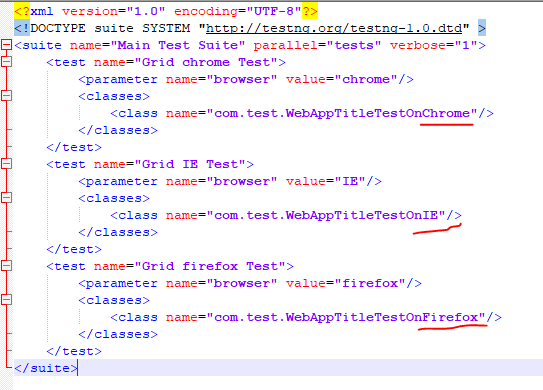
Run the maven local build: mvn clean test



Creating three java files with same code for each browser as to take the screenshot separately.

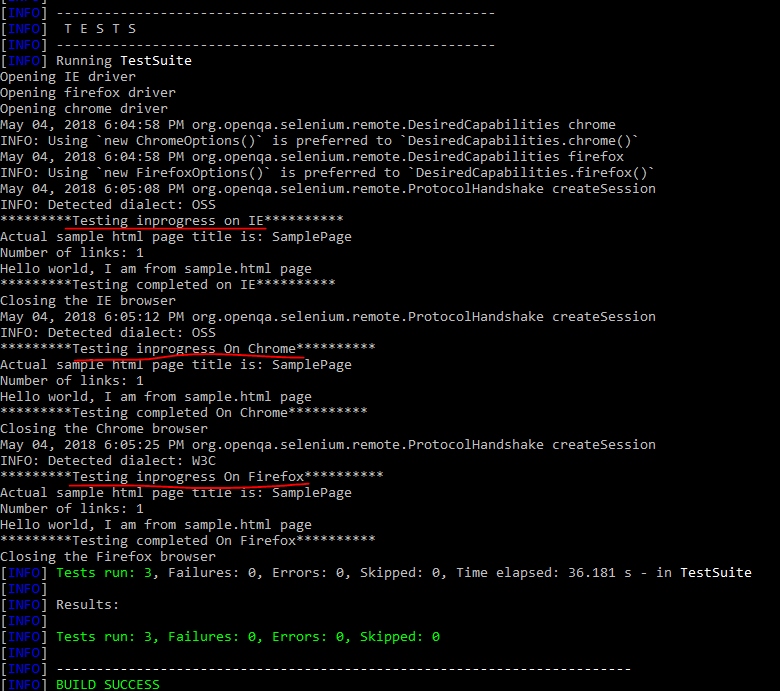


And the testing.xml file needs to be configured as below.

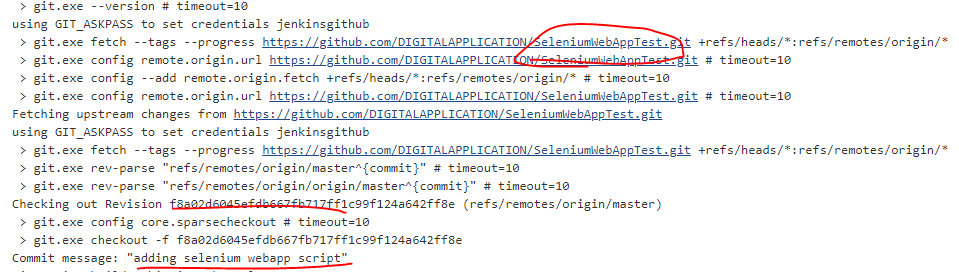


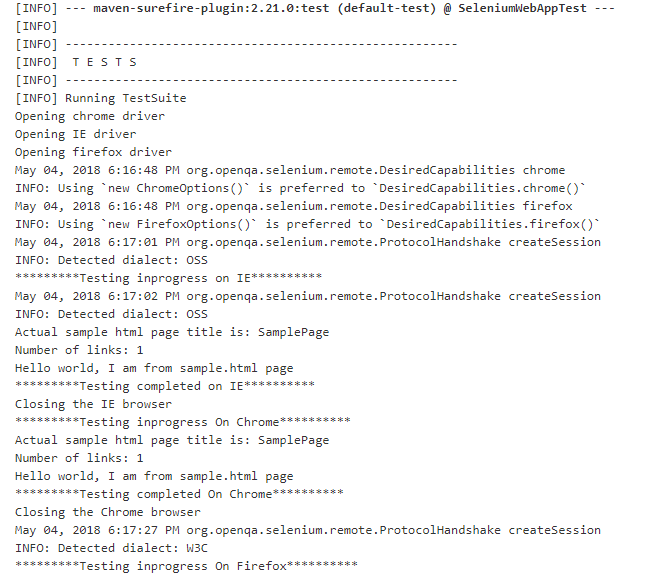
Find the code at: <https://github.com/DIGITALAPPLICATION/SeleniumWebAppTest.git>

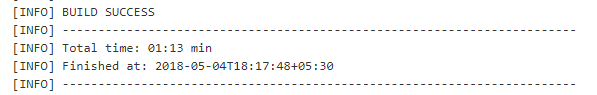
Output:



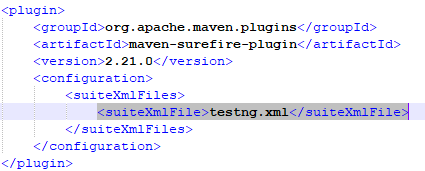
* 1. Jenkins job: Configure the Jenkins job same as 9.1. and update the git url with SeleniumWebAppTest.git.







1. Expected Errors & issues:
   1. If you are getting the below type of issue as shown the image, either you forgot to add or uncomment the tag in surefire plugin: <suiteXmlFile>testng.xml</suiteXmlFile>



Or

You forgot to give the proper test class name in the testing.xml file.

