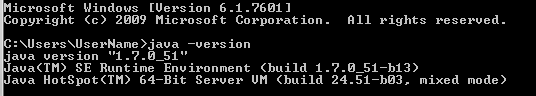
Selenium Automation

# Configure Java

Execute the command **java -version** in command window.

If java is configured it will show the java version as below



If Java is not configured it will give message

“**'java' is not recognized as an internal or external command,**

**operable program or batch file.”**

Then need to configure java:

1. Install Java JDK
2. Set environment variables: steps to set java environment path is given in the below link.

<http://www.javatpoint.com/how-to-set-path-in-java>

# Install Eclipse

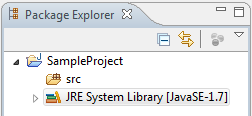
1. Download eclipse IDE: <https://eclipse.org/downloads/>
2. Start Eclipse by double clicking eclipse application file.

# Creating java Project

In eclipse,

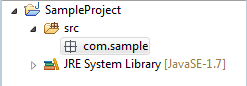
* Go to: File > New > Project…
* Select Java Project and Click Next.
* Give project name(any name), click finish.

New project will be created with “**src”** folder in it.

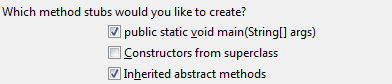


# Creating java class

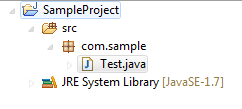
* Right click on “**src”** folder > New > Package
* Give package name(Ex: com.sample)



* Right click on package(com.sample) > New > Class
* Give class Name(Ex: Test)
* Select public *static void main(String[] arg)* check box, as shown below.



* Click finish.
* It will create java class with main method.

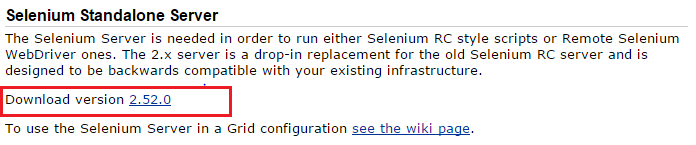


Selenium Basics

# Jar file Configuration with java project

* Download selenium-server-standalone jar file from below link:

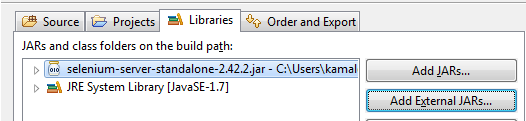
<http://www.seleniumhq.org/download/>



* Save jar file into any folder.

To configure selenium-server-standalone jar file:

* Right click on folder (src) or package (com.sample) > Build Path > configure Build Path…
* Go to Libraries
* Click Add External JARs…
* Browse the selenium-server-standalone jar file and click open.



NOTE: if you decide to move the selenium-server-standalone-n.n.n.jar file to different location later, existing test will fail as jar not found in the particular location. It is best to keep this file along with project files.

# How to initiate Browser using selenium.

## To Start Firefox browser:

There is slight difference in how to initialize Firefox browser in Selenium 2 and Selenium 3. Selenium 2 contained the required drivers to initialize Firefox browser within selenium jar, whereas it is removed in Selenium 3, need to download the driver file separately if you are using Selenium 3.

**Using Selenium 2:**

1. Add the below code into java main method.

WebDriver driver = **new** FirefoxDriver();

Tips: Mouse over the word if it shows some error, it will provide possible suggestions. (Ex: import WebDriver…)

1. Right click java program and Run As > Java Application.
2. Done!
3. It will open new Browser window for you (Provided browser is installed in your system).

**Using Selenium 3:**

1. You need gecko driver to initialize Firefox browser in selenium 3.

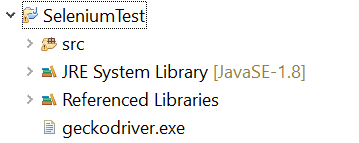
Download link: <https://github.com/mozilla/geckodriver/releases>

1. Add the below code into java main method.

System.*setProperty*("webdriver.gecko.driver","geckodriver.exe");

WebDriver driver = **new** FirefoxDriver();

Note: You should add the geckodriver.exe file into your project (copy and paste the file into project it will be added to project directory).



1. Right click java program and Run As > Java Application.
2. Done!

## To Start chrome browser:

1. You need Chrome browser driver to initialize chrome browser.

Download link: <https://sites.google.com/a/chromium.org/chromedriver/downloads>

1. Add the below code into java main method.

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

WebDriver driver = new ChromeDriver();

Note: You should add the chromedriver.exe file into your project (copy and paste the file into project it will be added to project directory).

1. Right click java program and Run As > Java Application.
2. Done!
3. It will open new Browser window for you (Provided browser is installed in your system).

## To Start IE browser:

1. You need IE browser driver to initialize IE browser.

Download link: <http://www.seleniumhq.org/download/> - The Internet Explorer Driver Server.

1. Add the below code into java main method.

System.*setProperty*("webdriver.ie.driver","IEDriverServer.exe");

WebDriver driver = **new** InternetExplorerDriver();

Note: You should add the IEDriverServer.exe file into your project (copy and paste the file into project it will be added to project directory).

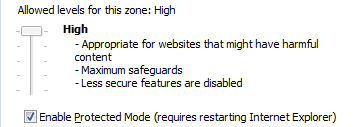
1. Right click java program and Run As > Java Application.

If not executed with below message:

Exception in thread "main" org.openqa.selenium.remote.SessionNotFoundException

* Go to: Internet Explorer > Settings > Internet Options
* Click Security tab.
* Select each option as shown below, and Select Enable Protected mode check box(For all options- Internet, Local internet, Trusted sites, Restricted sites)





1. Done!
2. It will open new Browser window for you (Provided browser is installed in your system).

# How to perform different actions on page.

Once we initialized our browser, next action will be to open URL (Ex: http://www.google.com)

## Open URL:

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

To perform actions in page, we need to find the web element using HTML identifiers, and then perform some actions on those web elements.

*Web Element can be a* ***text box****,* ***link****,* ***button****,* ***combo******box****,* ***check******box****,* ***radio******button*** *etc.*

*HTML identifier (An attribute) can be an* ***Id****,* ***name****,* ***tagName****,* ***className****,* ***Xpath*** *etc.*

*Action can be* ***typing text****,* ***clicking******button****,* ***selecting******checkbox*** *etc.*

## Scenario1:

Find an element (Textbox) in a page by identifier id= “gbqfqw”, and perform an action by sending text “Hello!” into it.

**Code**: driver.findElement(By.*id*("gbqfqw")).sendKeys("Hello!");

Similarly

## Scenario2:

Find an element (button) in a page by identifier id= “gbqfb”, and perform an action click on it.

**Code**: driver.findElement(By.*id*("gbqfb")).click();

# How to identify Web element

1. Open any browser, Ex: Internet Explorer
2. Press **F12** key it will open developer tool containing HTML code of the page
3. It contains element identifier icons. Like  OR  OR OR 
4. Click on identifier and select any Web element in the page.
5. This will highlight the HTML code related to that web element.
6. Select any identifier (Id, name, link text etc.) from the HTML code and use as Identifier.

# Simple selenium program.

Combining above actions will show code as below:

**package** com.sample;

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

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

**import** org.openqa.selenium.ie.InternetExplorerDriver;

**public** **class** Test {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

//open IE browser

System.*setProperty*("webdriver.ie.driver","IEDriverServer.exe");

WebDriver driver = **new** InternetExplorerDriver();

//open page http://www.google.com

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

//find textbox and submit test Hello

driver.findElement(By.*id*("gbqfqw")).sendKeys("Hello");

//click the search button

driver.findElement(By.*id*("gbqfb")).click();

//close browser window

driver.close();

//quit the driver instance

driver.quit();

}

}

# Different methods used in selenium.

## Navigate

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

driver. get("http://www.example.com");

driver.navigate().forward();

driver.navigate().back();

driver.navigate().refresh();

## Click

driver.findElement(By.*id*("*elementId*")).click()

## Type text / send text

driver.findElement(By.*id*("*elementId*")).sendKeys("Some Text")

## Select form dropdown

Select dropdown = new Select(driver.findElement(By.*id*("FromlanguageMenuId")));

dropdown.deselectAll();

dropdown.selectByVisibleText("English");

Or

dropdown.selectByIndex(1);

## Wait:

//Wait for maximum of 20 seconds for the page to load.

*driver*.manage().timeouts().pageLoadTimeout(20, TimeUnit.***SECONDS***);

//wait until 15 seconds in the page when trying to find an element or elements //if they are not immediately available

*driver*.manage().timeouts().implicitlyWait(15, TimeUnit.***SECONDS***);

## Methods for getting information about a page/element:

driver.getTitle()

driver.findElement(By.*id*("*elementId*")).getText()

driver.findElement(By.*id*("*elementId*")).isEditabled()

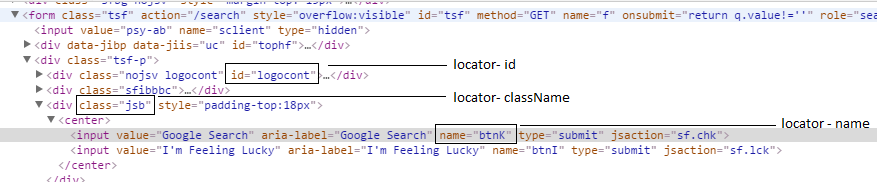
driver.findElement(By.*id*("*elementId*")).isDisabled()

driver.findElement(By.*id*("*elementId*")).isSelected()

etc.

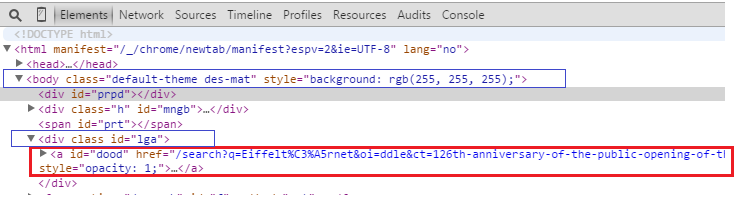
# Identifiers supported:

1. Class Name
2. cssSelector
3. Id
4. Link Text
5. Name
6. Tag Name
7. Xpath



## Xpath (identify element with Xpath):

|  |
| --- |
| In chrome browser to get xpath follow below steps:   1. *Right click* on the element for which xpath is required and select *inspect*. 2. From the developer window right click on the highlighted tag and select ***Copy*** *>* ***Copy Xpath*** |



1. In the above html, if we need to identify tag “a”, xpath will be,

Xpath: **//a[@id=’dood’]**

1. In the above html, if we need to identify tag “a”, from parent tag <div class id=”lga”>, xpath will be

Xpath: **//div[@id=’lga’]/a**

1. In the above html, if we need to identify tag “a”, from parent tag <body>, xpath will be

Xpath: **//body/div[3]/a** (notice that we have not used attribute with body since its unique)

OR

Skip all intermediate nodes between body and first “a” tag.

Xpath: **//body/descendant::a[1]**

We can also use attributes of “a” as below for precise identification.

Xpath: **//body/descendant::a[@id=’dood’]**

## CssSelector (identify element with CssSelector):

|  |
| --- |
| Html…  <div id=“divId” class=“divClassName” type=“divType”>  <div id=“divId2”> </div>  <a href=“www.somewebpage.co.in” class=“aClass” > someLinkText </a>  </div> |

### To identify element using class name:

cssSelector: **tagName.ClassName** or **.ClassName**

### To identify element using id:

cssSelector: **tagName#idValue** or **#idValue**

### To identify element using attribute:

cssSelector: **tagName[attributeName=’attributeValue’]** or **[attributeName=’attributeValue’]**

### To identify element using Partial attribute value:

#### For attribute value starts-With:

cssSelector: **tagName[attributeName^=’attributeValue’]**

#### For attribute value ends-With:

cssSelector: **tagName[attributeName$=’attributeValue’]**

#### For attribute value has-Sub-String:

cssSelector: **tagName[attributeName\*=’attributeValue’]**

### Example:

*driver*.findElement(By.cssSelector(".divClass"));

*driver*.findElement(By.cssSelector("div#divId"));

*driver*.findElement(By.cssSelector("#divId"));

*driver*.findElement(By.cssSelector("div[type=’divType’]"));

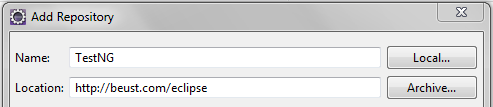
*driver*.findElement(By.cssSelector("div[type^=’div’]"));

TestNG

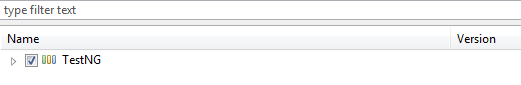
# Configure TestNG

Install TestNG plugin to eclipse,

1. Go to Eclipse Help > Install New Software…
2. Click Add (Work with)



1. Fill details as shown above, Click OK
2. Select TestNG and Click Next



1. Follow installation instructions.
2. Restart eclipse.

# Simple program using TestNG

1. Create a new Java project (Sample TestNG)
2. Create package com.testNG inside src folder.
3. Create a TestNG class using below steps:
   1. Right Click on folder > TestNG > create TestNG class
   2. Add Selenium server jar file into build path ([Click Here](#jar_config)).

TestNG is a framework provides annotations to perform different actions in different methods,

## Scenario1:

Before executing any test in any page we need to open browser and need to open particular webpage.

TestNG Code:

@BeforeTest

**public** **void** before(){

System.*setProperty*("webdriver.ie.driver","IEDriverServer.exe");

driver = **new** InternetExplorerDriver();

driver.get("http://google.com");

}

## Scenario2:

During test execution we perform certain actions on the page, let say find the text box and write (send text to textbox) text into it.

@Test

**public** **void** Test01() {

WebElement textbox = driver.findElement(By.*name*("q"));

textbox.sendKeys("hello");

}

## Scenario3:

After test we close browser window and quit driver instance.

@AfterTest

**public** **void** teminate() {

driver.close();

driver.quit();

}

# TestNG example

Combining above actions will form code as below:

**package** com. sample;

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

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

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.ie.InternetExplorerDriver;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeTest;

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

**public** **class** NewTest {

WebDriver driver;

@BeforeTest

**public** **void** before(){

System.*setProperty*("webdriver.ie.driver","IEDriverServer.exe");

driver = **new** InternetExplorerDriver();

driver.get("http://google.com");

}

@Test

**public** **void** Test01() {

WebElement textbox = driver.findElement(By.*name*("q"));

textbox.sendKeys("hello");

}

@AfterTest

**public** **void** teminate() {

driver.close();

driver.quit();

}

}

# TestNG Annotations

|  |  |
| --- | --- |
| Annotation name | Documentation |
| @BeforeSuite | The annotated method will be run before all tests in this suite have run. |
| @BeforeGroups | The Annotated method will run before the first method in any of the specified groups is invoked. |
| @BeforeClass | The Annotated method will be run before the first method on the current test class is invoked. |
| @BeforeTest | The Annotated method will be run before any test method in a given is invoked. |
| @BeforeMethod | The Annotated method will be run before each test method. |
| @AfterMethod | The Annotated method will be run after each test method. |
| @AfterTest | The Annotated method will be run after all the test methods in a given have been run. |
| @AfterClass | The Annotated method that will be run after the last test method on the current class is run. |

# Data driven test:

TestNG provides an annotation called **dataprovider** which can be used to supply data for a test method.

* The data provider name defaults to method name.
* The annotated method must return an Object[][] where each Object[] can be assigned the parameter list of the test method.
* The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

Example:

@DataProvider(name = "inputData")

**public** **static** Object[][] inputData() {

**return** **new** Object[][] { { 2, **true** },

{ 6, **false** },

{ 19, **true** },

{ 22, **false** },

{ 23, **true** } };

}

@Test(dataProvider = "inputData")

**void** testPrimeNumberChecker(Integer inputNumber, Boolean expectedResult) {

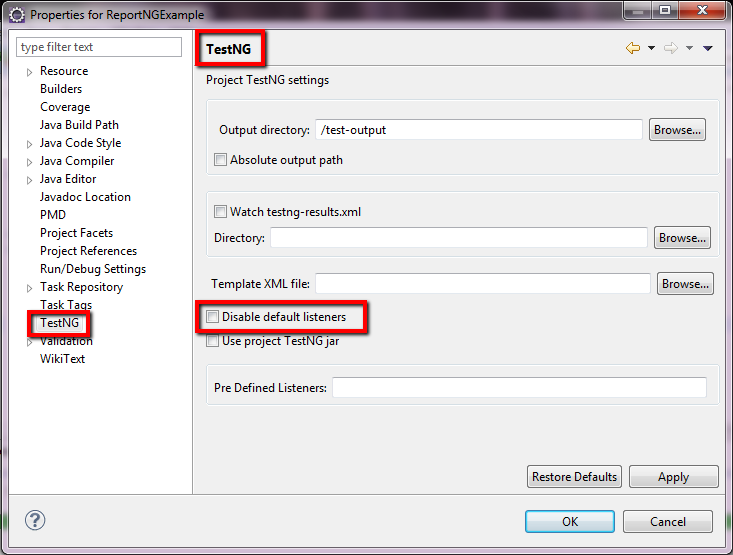
System.***out***.println(inputNumber + " " + expectedResult);

}

ReportNG

# Configuring ReportNG with TestNG tests for customized reports.

1. Add the below Jars Files to your project.
   1. reportng-1.1.x.jar
   2. velocity-dep-1.4.jar
   3. guice-3.0.jar
2. To make sure ReportNG reports, we need to disable the default TestNG Listeners.
   1. Right click on project > Properties
   2. Click on TestNG
   3. You will find an option as "Disable default listeners", check the checkbox
   4. Click on "Apply" button, it will show as message as "Project preferences are saved".
   5. Now click on "OK" button.



1. We need to add the below two listeners to testng.xml file.

<!DOCTYPE suite SYSTEM "[http://testng.org/testng-1.0.dtd"](http://testng.org/testng-1.0.dtd) >

<**suite** name="Suite1" verbose="1" >

<**listeners**>

<**listener** class-name="org.uncommons.reportng.HTMLReporter"/>

<**listener** class-name="org.uncommons.reportng.JUnitXMLReporter"/>

</**listeners**>

<**test** name="Regression Test Suite"   >

<**packages**>

<**package** name="packOne" />

<**package** name="packTwo" />

</**packages**>

</**test**>

</**suite**>

1. Output report files will be in the folder path: test-output > html > index.html

Selenium with Maven

# Maven

Maven is a build automation tool used primarily for Java projects. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

# Configure Maven into your system:

1. Download maven (apache-maven-x.x.x-bin.zip) from the below link,

<https://maven.apache.org/download.cgi>

1. Ensure ***JAVA\_HOME*** environment variable is set and points to your JDK installation
2. Unzip the downloaded file
3. Add the bin directory (EX: *D:\Admin\apache-maven-x.x.x\bin*) of apache-maven to the ***PATH*** environment variable.

More info on setting environment variables: <http://www.computerhope.com/issues/ch000549.htm>

1. Confirm by running below command in command window:

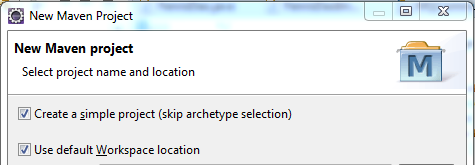
**mvn –v**

# Configure Maven in Eclipse:

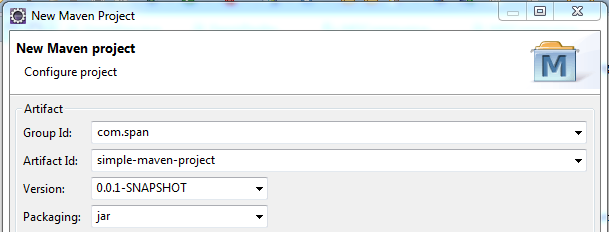
1. In Eclipse, go to Help > Install New Software…
2. In *Work with* text box type below text and click enter, http://download.eclipse.org/technology/m2e/releases
3. Select the checkbox for maven integration for Eclipse, follow installation wizard.
4. It may require to restart Eclipse.

# Creating Maven project in eclipse:

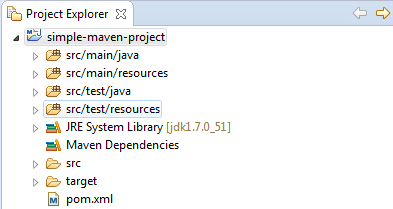
1. In Eclipse, go to File > New > Other…
2. Select “Maven Project”> Next
3. Select checkbox – Create a simple project > Next



1. Provide Group Id and Artifact Id (project name) for the project, these two fields combine to form a unique identifier for the project.



1. Directory structure:



Packages:

**src/main/java:** all the java implementation class to be written in this package.

**src/main/resource:** all the resources used by implementation classes to be added in this package, ex: property files, configuration files, etc.

**src/test/java**: all the test classes to be written in this package.

**src/test/resource:** all the resources used by test classes to be added in this package, ex: test data files, test configuration files, etc.

NOTE:

1. Notice that the project is created with the name specified in the *artifact Id*.
2. The JRE System Library is pointing to **jdk1.7.0\_51**, if it is pointing to JRE you need to configure build path to point to JDK (JDK is needed to build (compile java files) the java project)

More Info: <https://maven.apache.org/what-is-maven.html>

# Selenium test in maven project:

You can add dependency in pom.xml file so that it is not required to configure the build manually.

Below pom.xml file has added dependency of *selenium-server,* version 2.48.2.

<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.span</groupId>

<artifactId>simple-maven-project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>selenium-server</artifactId>

<version>2.48.2</version>

</dependency>

</dependencies>

</project>

## Build maven project:

1. In eclipse right click project > Run As > Maven install

OR

Alternatively you can go to the project directory (where pom.xml file is located), open command window and run below command,

* mvn clean install