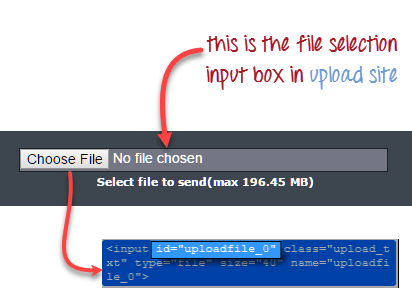
**How to Upload a File using Selenium Webdriver**

**Uploading files in WebDriver is done by simply using the sendKeys() method on the file-select input field to enter the path to the file to be uploaded.**



Handle file upload popup in selenium webdriver

package newproject;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

public class PG9 {

public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

String baseUrl = "http://demo.guru99.com/test/upload/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement uploadElement = driver.findElement(By.id("uploadfile\_0"));

// enter the file path onto the file-selection input field

uploadElement.sendKeys("C:\\newhtml.html");

// check the "I accept the terms of service" check box

driver.findElement(By.id("terms")).click();

// click the "UploadFile" button

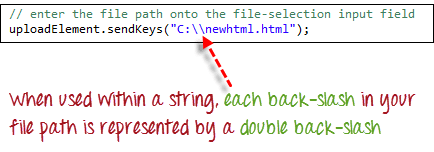
driver.findElement(By.name("send")).click();

}

}

Remember following two things when uploading files in WebDriver

1. There is no need to simulate the clicking of the “Browse” button. WebDriver automatically enters the file path onto the file-selection text box of the <input type=”file”> element
2. When setting the file path in your Java IDE, use the proper escape character for the back-slash.



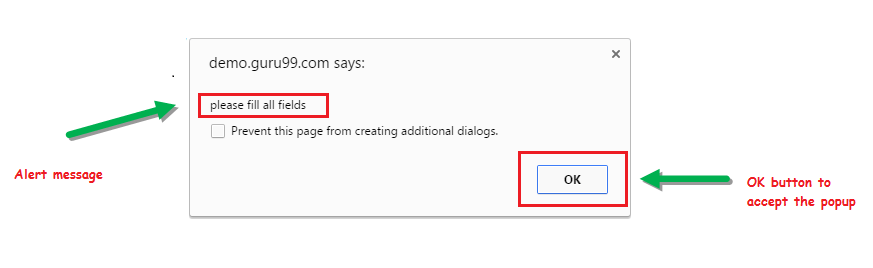
## **What is Alert in Selenium?**

An **Alert in Selenium** is a small message box which appears on screen to give the user some information or notification. It notifies the user with some specific information or error, asks for permission to perform certain tasks and it also provides warning messages as well.

**Types**

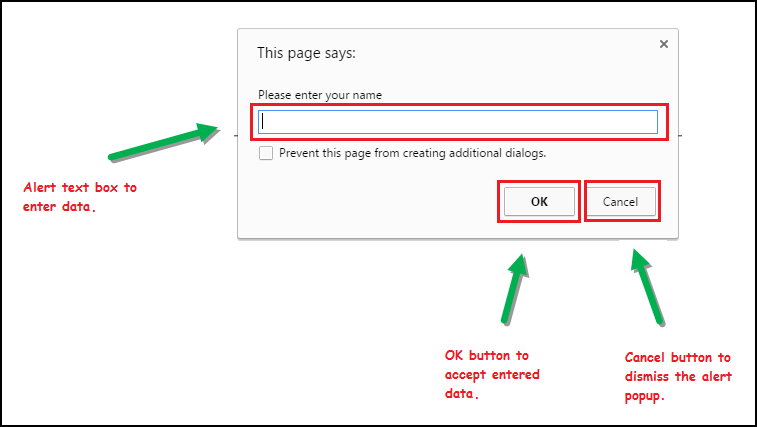
**1) Simple Alert**

The simple alert class in Selenium displays some information or warning on the screen.



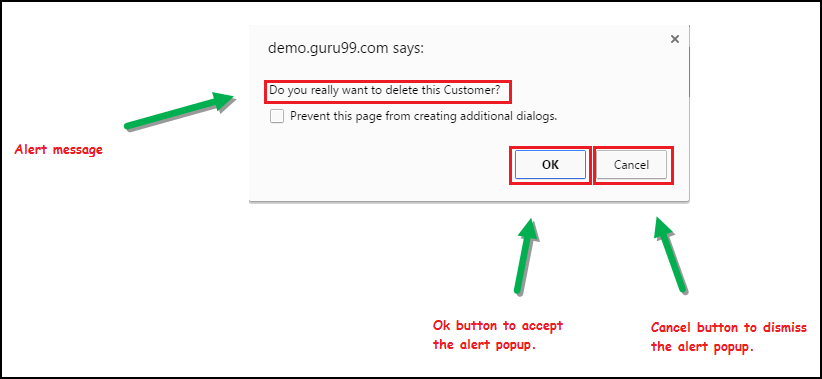
**2) Prompt Alert.**

This Prompt Alert asks some input from the user and Selenium webdriver can enter the text using sendkeys(” input…. “).



**3) Confirmation Alert.**

This confirmation alert asks permission to do some type of operation.



## **How to handle Alert in Selenium WebDriver**

1) void dismiss () **// To click on the ‘Cancel’ button of the alert.**

**driver.switchTo().alert().dismiss();**

2) void accept () **// To click on the ‘OK’ button of the alert.**

**driver.switchTo().alert().accept();**

3) String getText **() // To capture the alert message.**

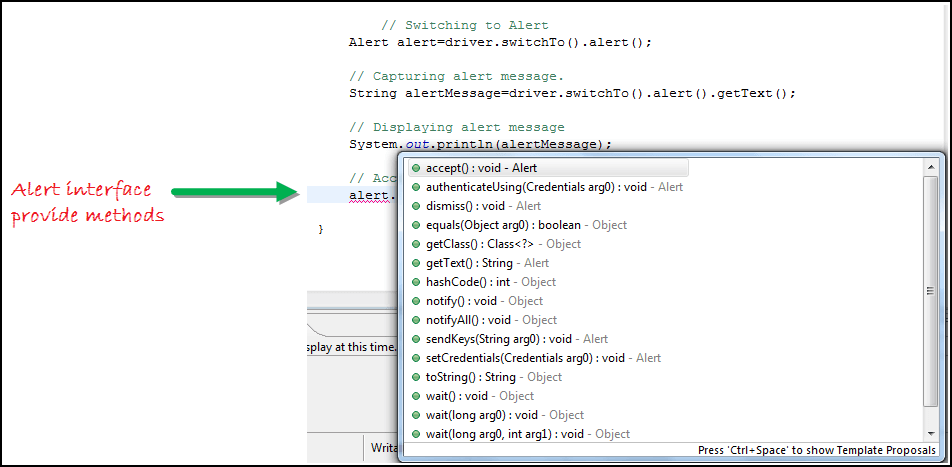
**driver.switchTo().alert().getText();**

4) void sendKeys (String stringToSend)**// To send some data to alert box.**

**driver.switchTo().alert().sendKeys("Text");**

You can see a number of Alert methods are displayed as shown in below screen suggested by Eclipse.

We can easily switch to alert from the main window by using Selenium’s **.switchTo()** method.

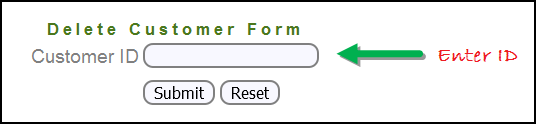


**Now we automate the given below scenario.**

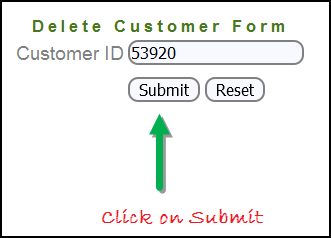
In this scenario, we will use Guru99 demo site to illustrate Selenium Alert handling.

**Step 1)** Launch the web browser and open the site http://demo.guru99.com/test/delete\_customer.php

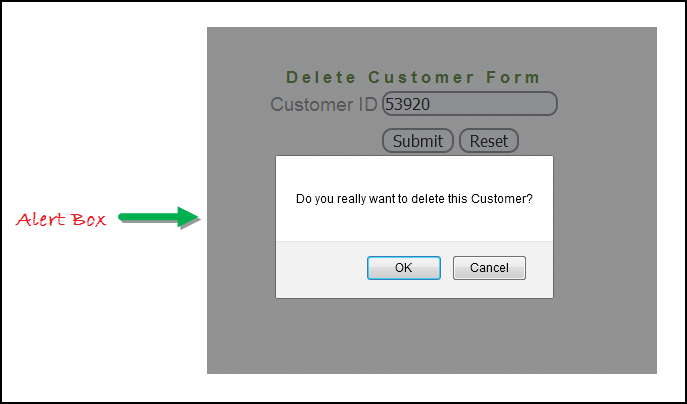
**Step 2)** Enter Any Customer id.



**Step 3)** After entering the customer ID, Click on the “Submit” button.



**Step 4)** Reject/accept the alert.



**Handling Alert in Selenium Webdriver using above scenario**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.NoAlertPresentException;

import org.openqa.selenium.Alert;

public class AlertDemo {

public static void main(String[] args) throws NoAlertPresentException,InterruptedException {

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

WebDriver driver = new ChromeDriver();

// Alert Message handling

driver.get("http://demo.guru99.com/test/delete\_customer.php");

driver.findElement(By.name("cusid")).sendKeys("53920");

driver.findElement(By.name("submit")).submit();

// Switching to Alert

**Alert alert = driver.switchTo().alert();**

**// Capturing alert message.**

**String alertMessage= driver.switchTo().alert().getText();**

**// Displaying alert message**

**System.out.println(alertMessage);**

Thread.sleep(5000);

// Accepting alert

alert.accept();

}

}

## **How to handle Selenium Pop-up window using Webdriver**

In automation, when we have multiple windows in any web application, the activity may need to switch control among several windows from one to other in order to complete the operation. After completion of the operation, it has to return to the main window i.e. parent window in Selenium. We will see this further in the article with an example.

In Selenium web driver there are methods through which we can handle multiple windows.

**Driver.getWindowHandles();**

To handle all opened windows by web driver, we can use “Driver.getWindowHandles()” and then we can switch window from one window to another in a web application. Its return type is Iterator<String>.

**Driver.getWindowHandle();**

When the site opens, we need to handle the main window by **driver.getWindowHandle()**. This will handle the current window that uniquely identifies it within this driver instance. Its return type is String.

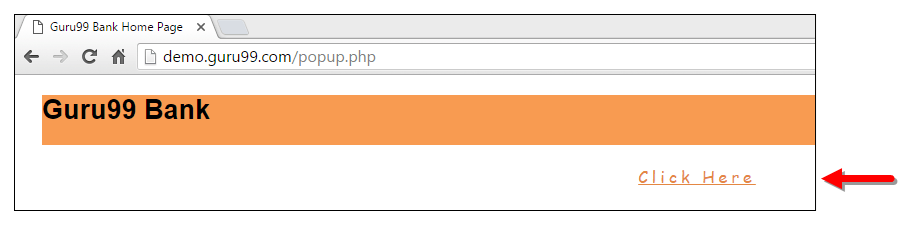
**Step 1)** Launch the site.

Launch the browser and open the site http://demo.guru99.com/popup.php



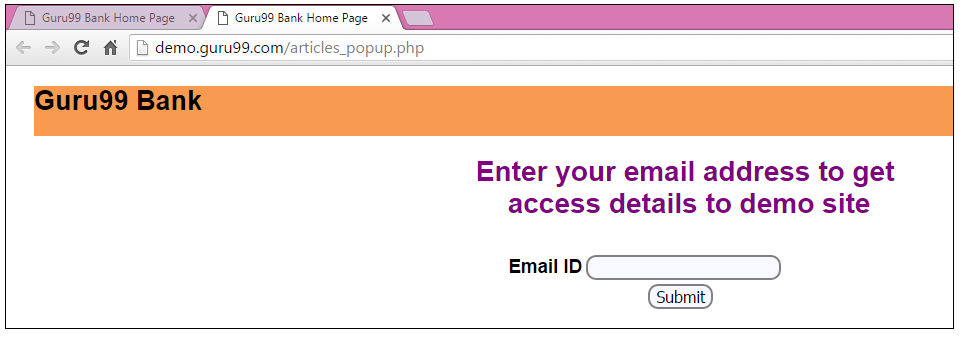
**Step 2)** Click on link “Click Here “.

When the user clicks on the “Click Here ” link, new child window opens.

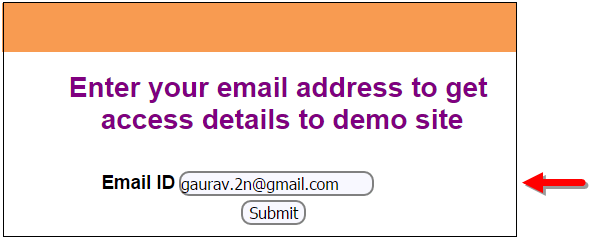


**Step 3)** New Child Window opens.

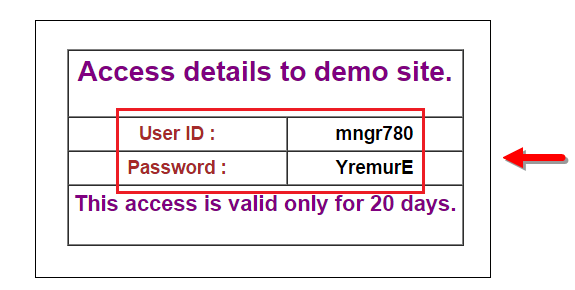
A new window opens, ask the user to enter email id and submit the page.



**Step 4)** Enter your email ID and submit.

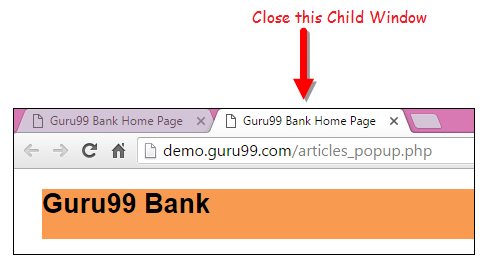


**Step 5)** Display the Access Credentials on submitting the page.

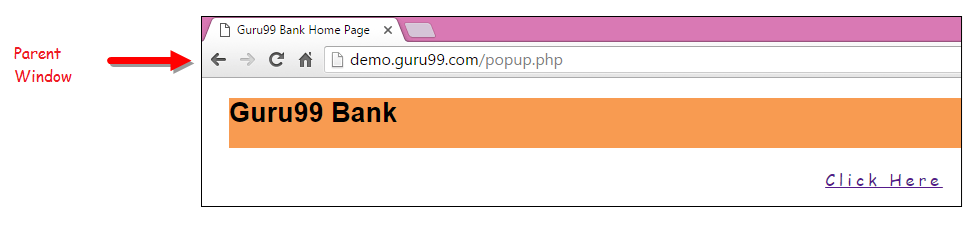


When you execute the code, you will see the child window is open in new tab.

1. Close the Child window on which credentials are displayed.



1. Switch to the parent window.



**Handling multiple windows in Selenium webdriver using above scenario.**

import java.util.Iterator;

import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class WindowHandle\_Demo {

public static void main(String[] args) throws InterruptedException {

WebDriver driver=new FirefoxDriver();

//Launching the site.

driver.get("http://demo.guru99.com/popup.php");

driver.manage().window().maximize();

driver.findElement(By.xpath("//\*[contains(@href,'popup.php')]")).click();

**String MainWindow=driver.getWindowHandle();**

**// To handle all new opened window.**

**Set<String> s1=driver.getWindowHandles();**

**Iterator<String> i1=s1.iterator();**

while(i1.hasNext())

{

String ChildWindow=i1.next();

if(!MainWindow.equalsIgnoreCase(ChildWindow))

{

// Switching to Child window

driver.switchTo().window(ChildWindow);

driver.findElement(By.name("emailid"))

.sendKeys("gaurav.3n@gmail.com");

driver.findElement(By.name("btnLogin")).click();

// Closing the Child Window.

driver.close();

}

}

// Switching to Parent window i.e Main Window.

driver.switchTo().window(MainWindow);

}

}

## **Desired Capabilities**

**Desired Capabilities** is a class in Selenium used to set properties of browsers to perform cross browser testing of web applications. It stores the capabilities as key-value pairs and these capabilities are used to set browser properties like browser name, browser version, path of browser driver in the system, etc. to determine the behaviour of browser at run time

## **Why do we need Desired Capabilities?**

**Desired Capabilities** are needed because every[Testing](https://www.guru99.com/software-testing.html)scenario should be executed on some specific testing environment. The testing environment can be a web browser,[Mobile](https://www.guru99.com/mobile-testing.html)device, mobile emulator, mobile simulator, etc. The Desired Capabilities Class helps us to tell the webdriver, which environment we are going to use in our test script.

## **Tooltip in Selenium**

A **Tooltip in Selenium** is a text that appears when a mouse hovers over an object on a web page. The object can be a link, an image, a button, a text area, etc. The tooltip text often gives more information about the object on which the user hovers over the mouse cursor.

Tooltips were traditionally implemented as a ‘title’ attribute to an element. The value of this attribute was shown as a tooltip on mouse-hover. This is a static text giving information of the element with no styling.

## **A Brief of the Advanced User Interactions API:**

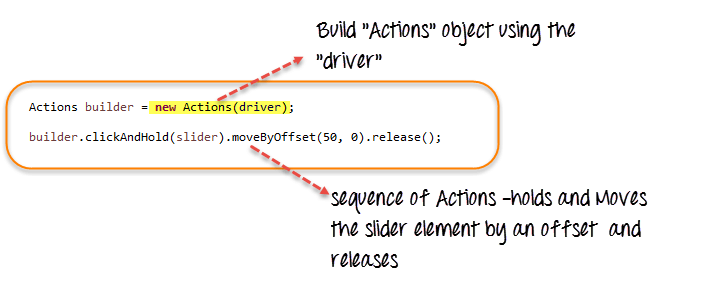
Advanced User Interactions API provides the API for user actions like drag and drop, hovering, multi selecting, key press and release and other actions using keyboard or mouse on a webpage.

**Step 1)**In order to use the API, the following packages/classes needs to be imported:

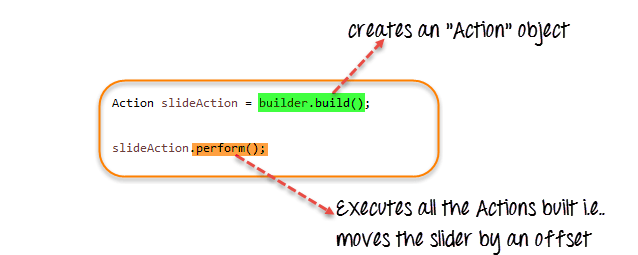
https://www.guru99.com/images/2-2017/072717_0606_VerifyToolt1.png

**Step 2)**Create an object of “Actions” class and build the Sequence of user actions. Actions class is used to build the sequence of user actions like moveToElement(), dragAndDrop() etc. Various methods related to user actions are provided by API.

The driver object is provided as a parameter to its constructor.



**Step 3)** Create an Action Object using the build() method of “Actions” class. Call the perform() method to execute all the actions built by the Actions object(builder here).



We have seen how to use some of the user Actions methods provided by the API – clickAndHold(element), moveByOffset(10,0), release(). The API provides many such methods.

Refer to the [link](https://seleniumhq.github.io/selenium/docs/api/java/index.html?org/openqa/selenium/interactions/Actions.html) for more details.

## How to get Tooltip Text in Selenium Webdriver

Let’s see the demonstration of accessing and verifying the tool tips in the simple scenario

* Scenario 1: Tooltip is implemented using the “title” attribute
* Scenario 2: Tooltip is implemented using a jQuery plugin.

### **Scenario 1: HTML ‘title’ Attribute**

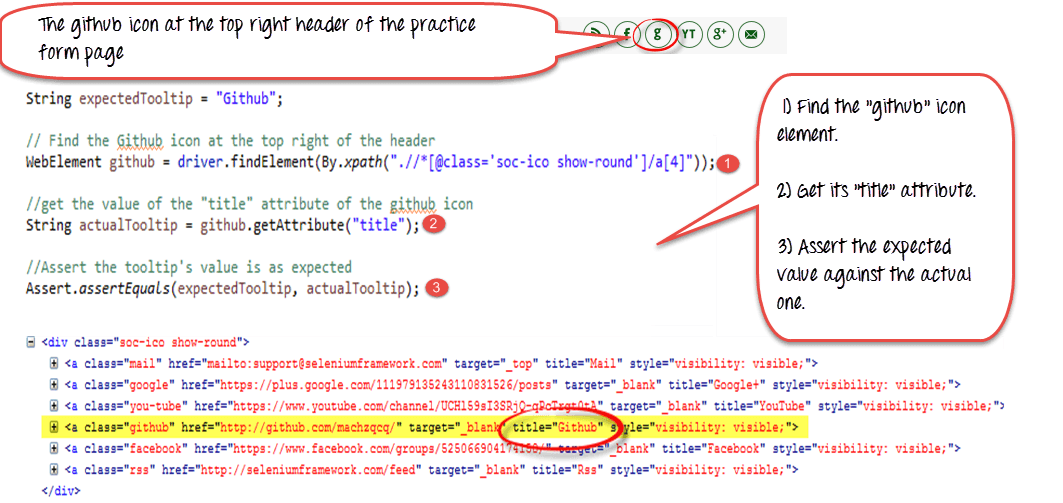
For this case, let’s take the example site – http://demo.guru99.com/test/social-icon.html.

We will try to verify the tooltip of the “github” icon at the top right of the page.

https://www.guru99.com/images/2-2017/072717_0606_VerifyToolt4.png

In order to do it, we will first find the element and get its ‘title’ attribute and verify with the expected tool tip text.

Since, we are assuming the tool tip is in the “title” attribute, we are not even automating the mouse hover effect but simply retrieving the attribute’s value using the “getAttribute()” method.



**Here is the code**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class ToolTip {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/social-icon.html";

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

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

String expectedTooltip = "Github";

// Find the Github icon at the top right of the header

WebElement github = driver.findElement(By.xpath(".//\*[@class='soc-ico show-round']/a[4]"));

//get the value of the "title" attribute of the github icon

String actualTooltip = github.getAttribute("title");

//Assert the tooltip's value is as expected

System.out.println("Actual Title of Tool Tip"+actualTooltip);

if(actualTooltip.equals(expectedTooltip)) {

System.out.println("Test Case Passed");

}

driver.close();

}

}

**Explanation of code**

1. Find the WebElement representing the “github” icon.
2. Get its “title” attribute using the getAttribute() method.
3. Assert the value against the expected tooltip value.

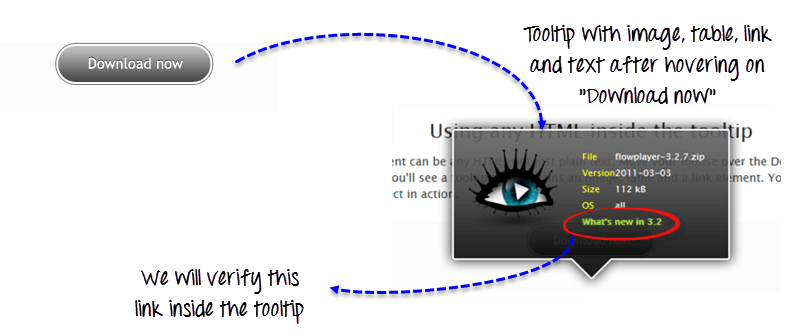
### **Scenario 2: JQuery Plugin:**

There are a plenty of JQuery plugins available to implement the tooltips, and each one has a slightly different form of implementation.

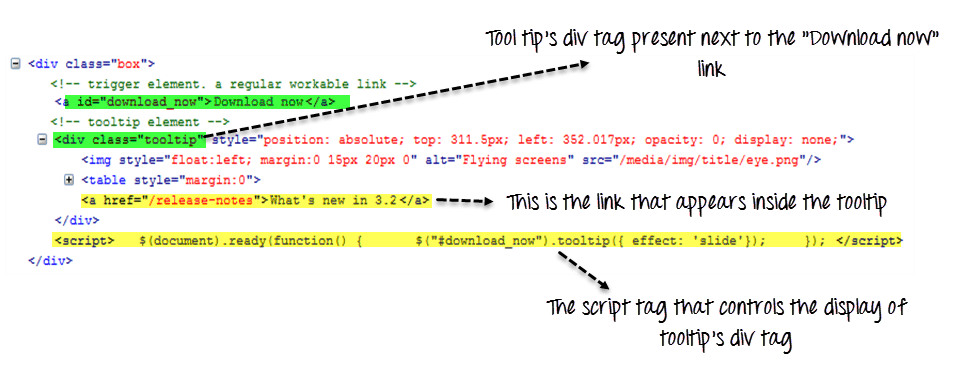
Some plugins expect the tooltip HTML to be present all the time next to the element for which the tooltip is applicable whereas the others create a dynamic “div” tag, which appears on the fly while hovering over the element.

For our demonstration, let’s consider the “jQuery Tools Tooltip” way of tooltip implementation.

Here in the URL – http://demo.guru99.com/test/tooltip.html you can see the demo where on mouse hovering over “Download now”, we get an advanced tooltip with an image, callout background, a table and a link inside it which is clickable.

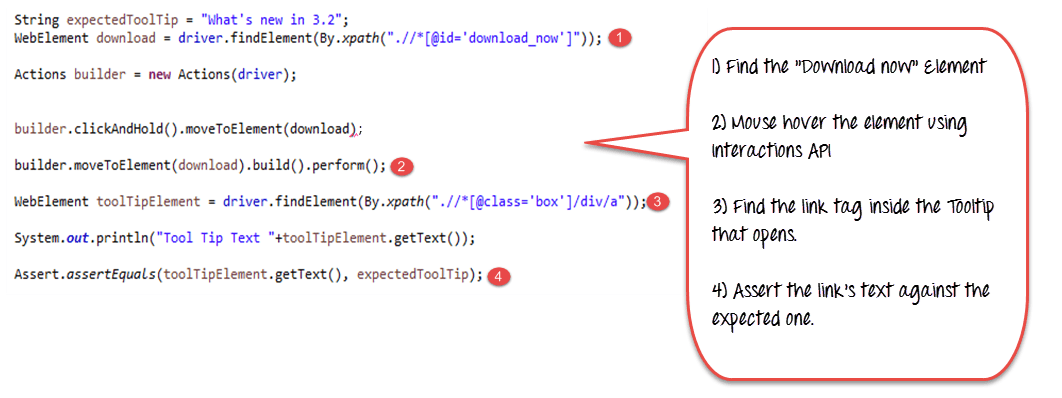


If you look at the source below, you can see that the div tag representing the tooltip is always present next to the “Download now” link’s tag. But, the code inside the script tag below controls when it needs to popup.



Let’s try to verify just the link text in the tooltip for our demonstration here.

We will first find the WebElement corresponding to the “Download now”. Then using the Interactions API, we will move to the element (mouse-hover). Next, we will find the WebElement that corresponds to the link inside the displayed tooltip and verify it against the expected text.



**Here is the code**

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class JqueryToolTip {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/tooltip.html";

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

WebDriver driver = new ChromeDriver();

String expectedTooltip = "What's new in 3.2";

driver.get(baseUrl);

WebElement download = driver.findElement(By.xpath(".//\*[@id='download\_now']"));

Actions builder = new Actions (driver);

builder.clickAndHold().moveToElement(download);

builder.moveToElement(download).build().perform();

WebElement toolTipElement = driver.findElement(By.xpath(".//\*[@class='box']/div/a"));

String actualTooltip = toolTipElement.getText();

System.out.println("Actual Title of Tool Tip "+actualTooltip);

if(actualTooltip.equals(expectedTooltip)) {

System.out.println("Test Case Passed");

}

driver.close();

}

}

**Code Explanation**

1. Find the WebElement that corresponds to the element “download now” that we will mouse-hover.
2. Using the Interactions API, mouse hover on to the “Download now”.
3. Assuming the tooltip is displayed, find the WebElement that corresponds to the link inside the tooltip i.e. the “a” tag.
4. Verify the link’s tooltip text retrieved using the getText() against an expected value we have stored in “expectedToolTip”

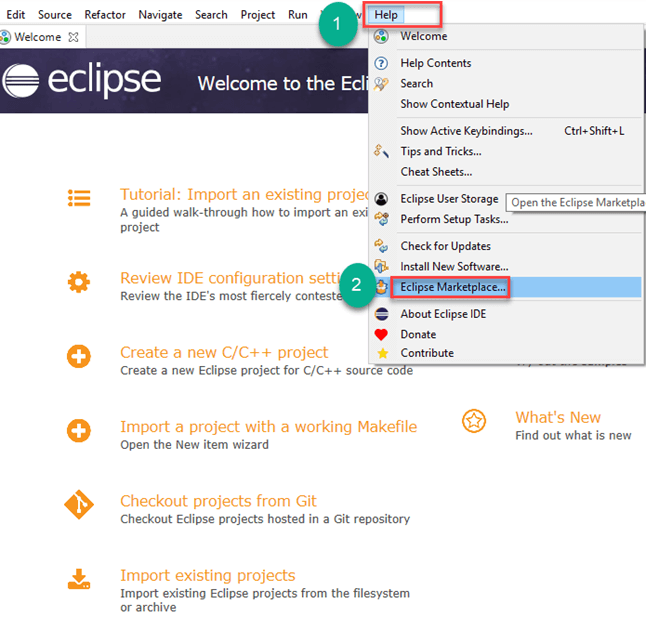
# How to Download, Install & Add TestNG in Eclipse for Selenium

Following is a step by step guide on how to install TestNG in Eclipse:

**Step 1)** Open Eclipse software & install new software

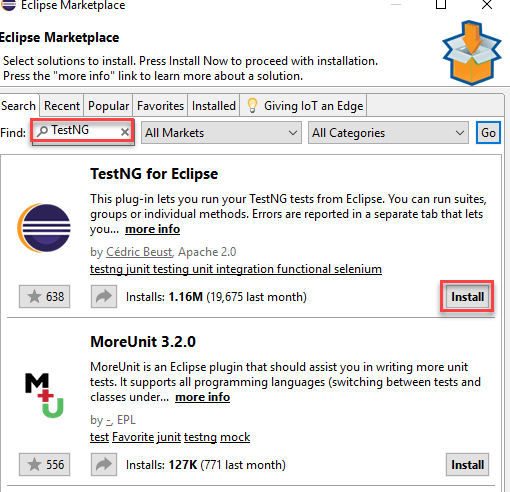
Launch Eclipse.

1. On the menu bar, click Help.
2. Choose the “Install New Software…” option.



**Step 2)**Search the TestNG using the Find option and,

Click on the installation button.



**Step 3)**Confirm the installation process.

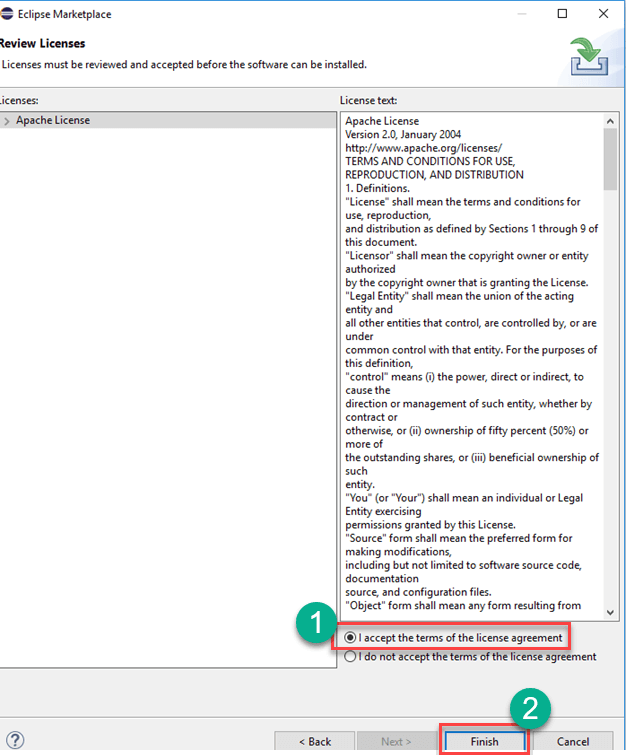
Click on confirm button



**Step 4)** Accept the license agreement

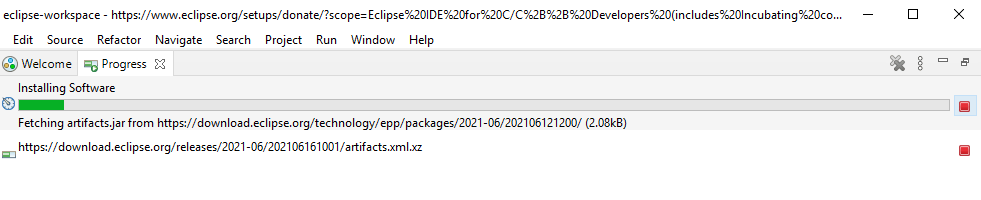
1. Select the radio button “I accept the terms of the license agreement”
2. Click on Finish.

It will take time depending on your Internet speed.



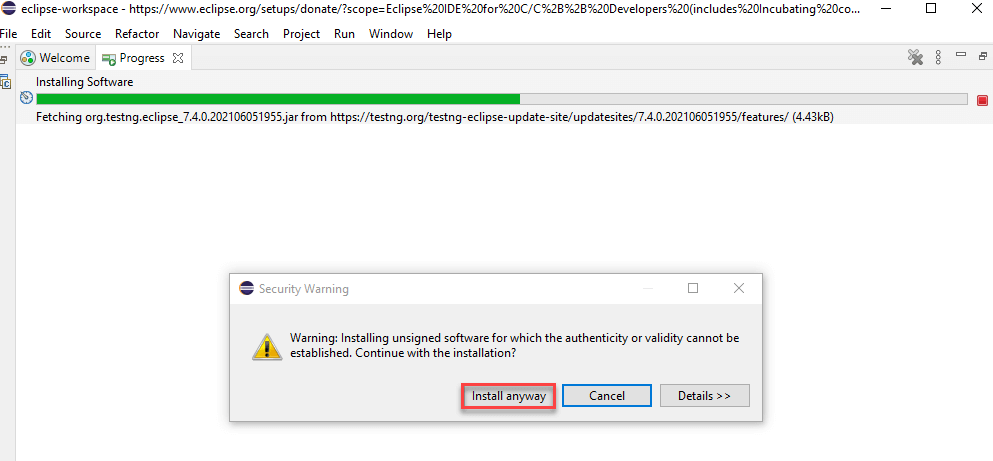
**Step 5)**After accept the licence agreement,

wait for installation.



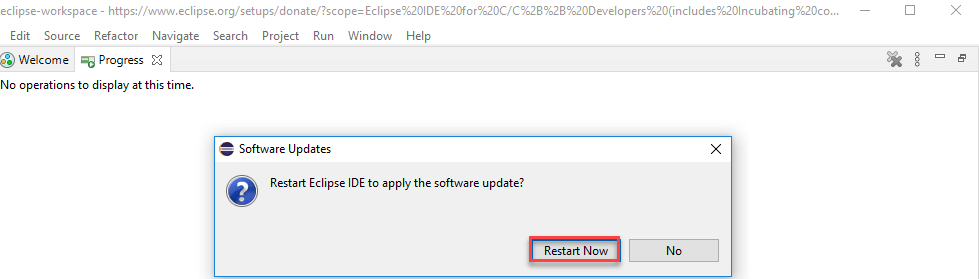
**Step 6)**Ignore security warning if occurs

If you encounter a Security warning, just click “Install Anyway”.



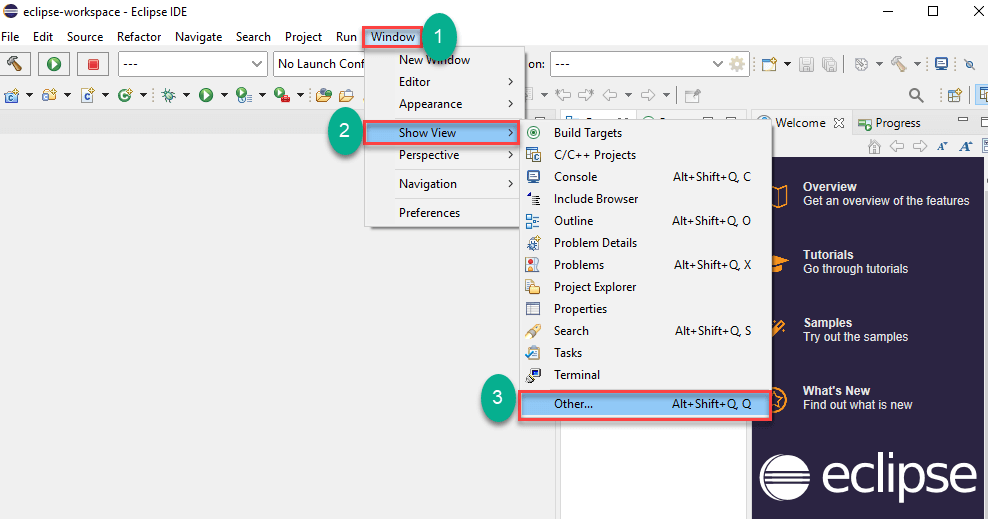
**Step 7)** Finish the installation and restart your system

Wait for the TestNG install in Eclipse to finish. When Eclipse prompts you for a restart, click “Restart now.”

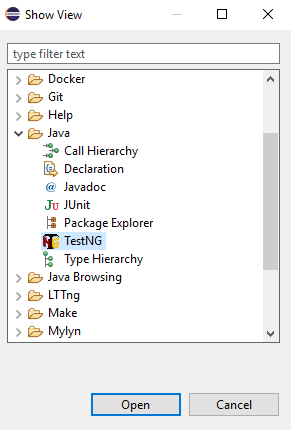


**Step 8)** Verify if the installation is done properly

After the restart, verify if TestNG for Eclipse was indeed successfully installed. Click Window > Show View > Other.



Then open the Java directory and see if TestNG is included.



That’s it on how to add TestNG in Eclipse.

# What is Annotations & Framework in Selenium

**TestNG**is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by [JUnit](https://www.guru99.com/junit-tutorial.html)which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make [end-to-end testing](https://www.guru99.com/end-to-end-testing.html) easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

* Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.
* The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

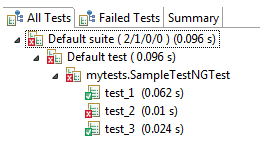
Beside above concept, you will learn more on TestNG, like what are the Advantages of TestNG, how to create test methods using @test annotations, how to convert these classes into testing suite file and execute through the eclipse as well as from the command line.

## Why Use TestNG with Selenium?

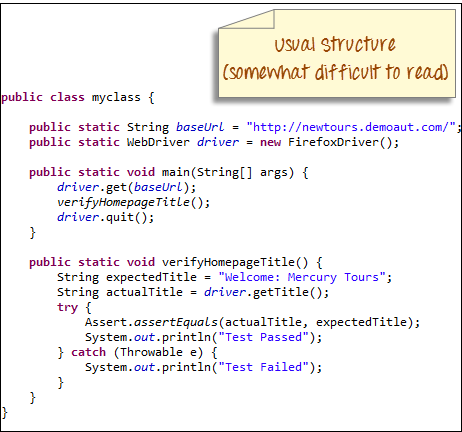
Default Selenium tests do not generate a proper format for the test results. Using TestNG in Selenium, we can generate test results.

Most Selenium users use this more than[Junit](https://www.guru99.com/junit-tutorial.html)because of its advantages. There are so many features of TestNG, but we will only focus on the most important ones that we can use in Selenium. Following are the key features of Selenium TestNG:

* Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
* The same test case can be executed multiple times without loops just by using keyword called ‘invocation count.’
* Using testng, you can execute multiple test cases on multiple browsers, i.e., cross [browser testing](https://www.guru99.com/top-10-cross-browser-testing-tools.html).
* The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.



* TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.





//table[contains(@id,'tblData')]//td//span[contains(@title,'Test Patient')]//parent::td//parent::tr//a[contains(@title,'8/10/2022')]