**How to Handle Alerts/Popups in Selenium WebDriver**

**There are two types of alerts that we would be focusing on majorly**:

1. Windows based alert pop ups
2. Web based alert pop ups

As we know that handling windows based pop ups is beyond WebDriver’s capabilities, thus we would exercise some third party utilities to handle window pop ups.

Handling pop up is one of the most challenging piece of work to automate while testing web applications. Owing to the diversity in types of pop ups complexes the situation even more.

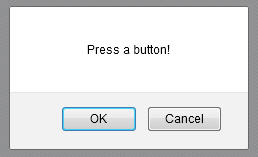
**What is Alert box/ Pop up box/ confirmation Box/ Prompt/ Authentication Box?**

It is nothing but a small box that appears on the display screen to give you some kind of information or to warn you about a potentially damaging operation or it may even ask you for the permissions for the operation.

**Example:** Let us consider a real life example for a better understanding; Let us assume that we uploaded a photograph on any of these popular social networking sites. Later on, i wish to delete the uploaded photograph. So in order to delete, i clicked on the delete button. As soon as I click on the delete button, the system warns me against my action, prompting – Do you really want to delete the file? So now we have an option to either accept this alert or reject it.

So ahead in the session, **let’s see how do we reject or accept the alerts depending on their types.** Starting with the web based pop ups.

### ****Web Based Popups****

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/webdriver-alerts-1.jpg)

Let us see how do we handle them using WebDriver.

**Handling web based pop-up box**

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.

**There are the four methods that we would be using along with the Alert interface.**

**1) void dismiss()** – The dismiss() method clicks on the “Cancel” button as soon as the pop up window appears.  
**2) void accept()** – The accept() method clicks on the “Ok” button as soon as the pop up window appears.  
**3) String getText()** – The getText() method returns the text displayed on the alert box.  
**4) void sendKeys(String stringToSend)** – The sendKeys() method enters the specified string pattern into the alert box.

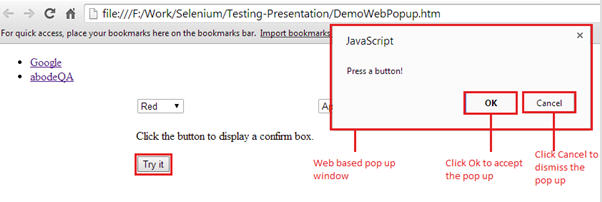
Let us move ahead and look at the actual implementation.

**Explanation of Application under Test**

We have designed a web page in a way to include a few fundamental types of web elements. This is the same application we introduced while discussing Select class earlier in this series.

* **Hyperlink**: The two hyperlinks namely “Google” and “abodeQA” have been provided that re-directs the user to “http://www.google.com/” and “http://www.abodeqa.com/” respectively on the click event.
* **Dropdown**: The three hyperlinks have been created for selecting colors, fruits and animals with a value set to default.
* **Button**: A “try it” button has been created to show up the pop up box having OK and Cancel buttons upon click event.

***(Click on image to view enlarged)***

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/webdriver-alerts-2.jpg)

**Subsequent is the HTML code used to create the above mentioned webpage:**

|  |  |  |
| --- | --- | --- |
| 1 | <!DOCTYPE html></pre> | |
| 2 | <html> |

|  |  |  |
| --- | --- | --- |
| 3 | <head><title> Testing Select Class </title> | |
| 4 | <body> |

|  |  |
| --- | --- |
| 5 | <div id="header"> |
| 6 | <ul id="linkTabs"> | |

|  |  |
| --- | --- |
| 7 | <li> |
| 8 | <a href="https://www.google.com/">Google</a> | |

|  |  |  |
| --- | --- | --- |
| 9 | </li> | |
| 10 | | <li> |

|  |  |  |
| --- | --- | --- |
| 11 | <a href="http://abodeqa.wordpress.com/">abodeQA</a> | |
| 12 | </li> |

|  |  |
| --- | --- |
| 13 | </ul> |
| 14 | </div> | |

|  |  |  |
| --- | --- | --- |
| 15 | <div class="header\_spacer"></div> | |
| 16 | <div id="container"> |

|  |  |  |
| --- | --- | --- |
| 17 | <div id="content" style="padding-left: 185px;"> | |
| 18 | <table id="selectTable"> |

|  |  |  |
| --- | --- | --- |
| 19 | <tbody> | |
| 20 | <tr> |

|  |  |
| --- | --- |
| 21 | <td> |
| 22 | <div> | |

|  |  |
| --- | --- |
| 23 | <select id="SelectID\_One"> |
| 24 | <option value="redvalue">Red</option> | |

|  |  |
| --- | --- |
| 25 | <option value="greenvalue">Green</option> |
| 26 | <option value="yellowvalue">Yellow</option> | |

|  |  |  |
| --- | --- | --- |
| 27 | <option value="greyvalue">Grey</option> | |
| 28 | </select> |

|  |  |  |
| --- | --- | --- |
| 29 | </div> | |
| 30 | </td> |

|  |  |
| --- | --- |
| 31 | <td> |
| 32 | <div> | |

|  |  |
| --- | --- |
| 33 | <select id="SelectID\_Two"> |
| 34 | <option value="applevalue">Apple</option> | |

|  |  |  |
| --- | --- | --- |
| 35 | <option value="orangevalue">Orange</option> | |
| 36 | <option value="mangovalue">Mango</option> |

|  |  |  |
| --- | --- | --- |
| 37 | <option value="limevalue">Lime</option> | |
| 38 | </select> |

|  |  |  |
| --- | --- | --- |
| 39 | </div> | |
| 40 | </td> |

|  |  |
| --- | --- |
| 41 | <td> |
| 42 | <div> | |

|  |  |
| --- | --- |
| 43 | <select id="SelectID\_Three"> |
| 44 | <option value="selectValue">Select</option> | |

|  |  |  |
| --- | --- | --- |
| 45 | <option value="elephantvalue">Elephant</option> | |
| 46 | <option value="mousevalue">Mouse</option> |

|  |  |  |
| --- | --- | --- |
| 47 | <option value="dogvalue">Dog</option> | |
| 48 | </select> |

|  |  |  |
| --- | --- | --- |
| 49 | </div> | |
| 50 | </td> |

|  |  |  |
| --- | --- | --- |
| 51 | </tr> | |
| 52 | <tr> |

|  |  |  |
| --- | --- | --- |
| 53 | <td> | |
| 54 |  |

|  |  |  |
| --- | --- | --- |
| 55 | <!DOCTYPE html> | |
| 56 | <html> |

|  |  |
| --- | --- |
| 57 | <body> |
| 58 | <p>Click the button to display a confirm box.</p> | |

|  |  |  |
| --- | --- | --- |
| 59 | <button onclick="myFunction()">Try it</button> | |
| 60 |  |

|  |  |
| --- | --- |
| 61 | <script> |
| 62 | function myFunction() | |

|  |  |
| --- | --- |
| 63 | { |
| 64 | confirm("Press a button!"); | |

|  |  |
| --- | --- |
| 65 | } |
| 66 | </script> | |

|  |  |
| --- | --- |
| 67 | </body> |
| 68 | </html> |

|  |  |
| --- | --- |
| 69 | </td> |
| 70 | </tr> |

|  |  |
| --- | --- |
| 71 | </tbody> |
| 72 | </table> |

|  |  |
| --- | --- |
| 73 | </div> |
| 74 | </div> |

|  |  |
| --- | --- |
| 75 | </body> |
| 76 | </html> |

**Scenario to be automated**

1. Launch the web browser and open the webpage
2. Click on the “Try it” button
3. Accept the alert
4. Click on the “Try it” button again
5. Reject the alert

**WebDriver Code using Select Class**

Please take a note that for script creation, we would be using “Learning\_Selenium” project created in the former tutorial.

**Step 1**: Create a new java class named as “DemoWebAlert” under the “Learning\_Selenium” project.  
**Step 2**: Copy and paste the below code in the “DemoWebAlert.java” class.

Below is the test script that is equivalent to the above mentioned scenario.

|  |  |
| --- | --- |
| 1 | import org.junit.After; |
| 2 | import org.junit.Before; | |

|  |  |
| --- | --- |
| 3 | import org.junit.Test; |
| 4 | import org.openqa.selenium.Alert; | |

|  |  |
| --- | --- |
| 5 | import org.openqa.selenium.By; |
| 6 | import org.openqa.selenium.WebDriver; | |

|  |  |  |
| --- | --- | --- |
| 7 | import org.openqa.selenium.firefox.FirefoxDriver; | |
| 8 |  |

|  |  |  |
| --- | --- | --- |
| 9 | /\*\* | |
| 10 | | \* class description | |

|  |  |  |
| --- | --- | --- |
| 11 | \*/ | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | public class DemoWebAlert { |
| 14 | WebDriver driver; | |

|  |  |
| --- | --- |
| 15 | /\*\* |
| 16 | \* Constructor | |

|  |  |
| --- | --- |
| 17 | \*/ |
| 18 | public DemoWebAlert() { | |

|  |  |  |
| --- | --- | --- |
| 19 | } | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | /\*\* |
| 22 | \* Set up browser settings and open the application | |

|  |  |  |
| --- | --- | --- |
| 23 | \*/ | |
| 24 |  |

|  |  |
| --- | --- |
| 25 | @Before |
| 26 | public void setUp() { | |

|  |  |  |
| --- | --- | --- |
| 27 | driver=newFirefoxDriver(); | |
| 28 | // Opened the application |

|  |  |  |
| --- | --- | --- |
| 29 | driver.get("file:///F:/Work/Selenium/Testing-Presentation/DemoWebPopup.htm"); | |
| 30 | driver.manage().window().maximize(); |

|  |  |  |
| --- | --- | --- |
| 31 | } | |
| 32 |  |

|  |  |
| --- | --- |
| 33 | /\*\* |
| 34 | \* Test to check Select functionality | |

|  |  |  |
| --- | --- | --- |
| 35 | \* @throws InterruptedException | |
| 36 | \*/ |

|  |  |
| --- | --- |
| 37 |  |
| 38 | @Test | |

|  |  |  |
| --- | --- | --- |
| 39 | public void testWebAlert() throwsInterruptedException { | |
| 40 | // clicking on try it button |

|  |  |  |  |
| --- | --- | --- | --- |
| 41 | driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click(); | | |
| 42 | | Thread.sleep(5000); |

|  |  |
| --- | --- |
| 43 |  |
| 44 | // accepting javascript alert | |

|  |  |  |
| --- | --- | --- |
| 45 | Alert alert = driver.switchTo().alert(); | |
| 46 | alert.accept(); |

|  |  |
| --- | --- |
| 47 |  |
| 48 | // clicking on try it button | |

|  |  |  |  |
| --- | --- | --- | --- |
| 49 | driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click(); | | |
| 50 | | Thread.sleep(5000); |

|  |  |
| --- | --- |
| 51 |  |
| 52 | // accepting javascript alert | |

|  |  |  |
| --- | --- | --- |
| 53 | driver.switchTo().alert().dismiss(); | |
| 54 |  |

|  |  |  |
| --- | --- | --- |
| 55 | | // clicking on try it button |
| 56 | driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click(); | | |

|  |  |  |
| --- | --- | --- |
| 57 | Thread.sleep(5000); | |
| 58 |  |

|  |  |  |
| --- | --- | --- |
| 59 | | // accepting javascript alert |
| 60 | System.out.println(driver.switchTo().alert().getText()); | | |

|  |  |  |
| --- | --- | --- |
| 61 | driver.switchTo().alert().accept(); | |
| 62 | } |

|  |  |
| --- | --- |
| 63 |  |
| 64 | /\*\* | |

|  |  |  |
| --- | --- | --- |
| 65 | \* Tear down the setup after test completes | |
| 66 | \*/ |

|  |  |
| --- | --- |
| 67 |  |
| 68 | @After | |

|  |  |  |
| --- | --- | --- |
| 69 | public void tearDown() { | |
| 70 | driver.quit(); |

|  |  |  |
| --- | --- | --- |
| 71 | } | |
| 72 | } |

**Code Walk-through**

------------

**Import Statements**

**Import org.openqa.selenium.Alert** – Import this package prior to the script creation The package references to the Alert class which is required to handle the web based alerts in WebDriver.

**Object Creation for Alert class**  
Alert alert = driver.switchTo().alert();

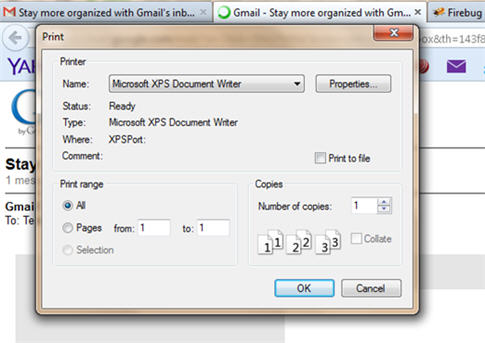
We create a reference variable for Alert class and references it to the alert.

**Switch to Alert**  
Driver.switchTo().alert();  
The above command is used to switch the control to the recently generated pop up window.

**Accept the Alert**  
alert.accept();  
The above command accepts the alert thereby clicking on the Ok button.

**Reject the Alert**  
alert.dismiss();  
The above command closes the alert thereby clicking on the Cancel button and hence the operation should not proceed.

### ****Window Based Pop Ups****

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/11/webdriver-alerts-3.jpg)

At times while automating, we get some scenarios, where we need to handle pop ups generated by windows like a print pop up or a browsing window while uploading a file.

Handling these pop-ups have always been a little tricky as we know Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support windows based applications and window alert is one of them. However Selenium alone can’t help the situation but along with some third party intervention, this problem can be overcome.

There are several third party tools available for handling window based pop-ups along with the selenium.

**So now let’s handle a window based pop up using Robot class.**

Robot class is a java based utility which emulates the keyboard and mouse actions.

Before moving ahead, let us take a moment to have a look at the application under test (AUT).

**Explanation of Application under Test**

As an application under test, we would be using “gmail.com”. I believe the application doesn’t require any more introductions.

**Scenario to be automated**

1. Launch the web browser and open the application – “gmail.com”
2. Enter valid username and password
3. Click on the sign in button
4. Click on the a compose button
5. Click on the attach icon
6. Select the files to be uploaded with the window based pop up.

**WebDriver Code using Robot Class**

Please take a note that for script creation, we would be using “Learning\_Selenium” project created in the former tutorial.

**Step 1**: Create a new java class named as “DemoWindowAlert” under the “Learning\_Selenium” project.  
**Step 2**: Copy and paste the below code in the “DemoWindowAlert.java” class.

Below is the test script that is equivalent to the above mentioned scenario.

|  |  |
| --- | --- |
| 1 | import java.awt.Robot;</pre> |
| 2 | import java.awt.event.KeyEvent; | |

|  |  |
| --- | --- |
| 3 | import org.junit.After; |
| 4 | import org.junit.Before; | |

|  |  |
| --- | --- |
| 5 | import org.junit.Test; |
| 6 | import org.openqa.selenium.By; | |

|  |  |
| --- | --- |
| 7 | import org.openqa.selenium.WebDriver; |
| 8 | import org.openqa.selenium.firefox.FirefoxDriver; | |

|  |  |  |
| --- | --- | --- |
| 9 |  | |
| 10 | | public class DemoWindowAlert { | |

|  |  |  |
| --- | --- | --- |
| 11 | WebDriver driver; | |
| 12 | @Before |

|  |  |
| --- | --- |
| 13 |  |
| 14 | public void setUp() | |

|  |  |
| --- | --- |
| 15 | { |
| 16 | driver=new FirefoxDriver(); | |

|  |  |
| --- | --- |
| 17 | driver.get("https://gmail.com"); |
| 18 | driver.manage().window().maximize(); | |

|  |  |  |
| --- | --- | --- |
| 19 | } | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | @Test |
| 22 | public void testWindowAlert() throws Exception{ | |

|  |  |
| --- | --- |
| 23 |  |
| 24 | // enter a valid email address | |

|  |  |  |
| --- | --- | --- |
| 25 | driver.findElement(By.id("Email")).sendKeys("TestSelenium1607@gmail.com"); | |
| 26 |  |

|  |  |
| --- | --- |
| 27 | // enter a valid password |
| 28 | driver.findElement(By.id("Passwd")).sendKeys("TestSelenium"); | |

|  |  |
| --- | --- |
| 29 |  |
| 30 | // click on sign in button | |

|  |  |  |
| --- | --- | --- |
| 31 | driver.findElement(By.id("signIn")).click(); | |
| 32 | Thread.sleep(30000); |

|  |  |
| --- | --- |
| 33 |  |
| 34 | // click on compose button | |

|  |  |  |  |
| --- | --- | --- | --- |
| 35 | driver.findElement(By.xpath("//div[@class='z0']//div[contains(text(),'COMPOSE')]")).click(); | | |
| 36 | |  |

|  |  |  |
| --- | --- | --- |
| 37 | | // click on attach files icon |
| 38 | driver.findElement(By.xpath("//div[contains(@command,'Files')]//div[contains(@class,'aaA')]")).click(); | | |

|  |  |
| --- | --- |
| 39 |  |
| 40 | // creating instance of Robot class (A java based utility) | |

|  |  |  |
| --- | --- | --- |
| 41 | Robot rb =new Robot(); | |
| 42 |  |

|  |  |  |
| --- | --- | --- |
| 43 | // pressing keys with the help of keyPress and keyRelease events | |
| 44 | rb.keyPress(KeyEvent.VK\_D); |

|  |  |  |
| --- | --- | --- |
| 45 | rb.keyRelease(KeyEvent.VK\_D); | |
| 46 | Thread.sleep(2000); |

|  |  |
| --- | --- |
| 47 |  |
| 48 | rb.keyPress(KeyEvent.VK\_SHIFT); | |

|  |  |
| --- | --- |
| 49 | rb.keyPress(KeyEvent.VK\_SEMICOLON); |
| 50 | rb.keyRelease(KeyEvent.VK\_SEMICOLON); | |

|  |  |  |
| --- | --- | --- |
| 51 | rb.keyRelease(KeyEvent.VK\_SHIFT); | |
| 52 |  |

|  |  |
| --- | --- |
| 53 | rb.keyPress(KeyEvent.VK\_BACK\_SLASH); |
| 54 | rb.keyRelease(KeyEvent.VK\_BACK\_SLASH); | |

|  |  |  |
| --- | --- | --- |
| 55 | Thread.sleep(2000); | |
| 56 |  |

|  |  |
| --- | --- |
| 57 | rb.keyPress(KeyEvent.VK\_P); |
| 58 | rb.keyRelease(KeyEvent.VK\_P); | |

|  |  |
| --- | --- |
| 59 |  |
| 60 | rb.keyPress(KeyEvent.VK\_I); | |

|  |  |  |
| --- | --- | --- |
| 61 | rb.keyRelease(KeyEvent.VK\_I); | |
| 62 |  |

|  |  |
| --- | --- |
| 63 | rb.keyPress(KeyEvent.VK\_C); |
| 64 | rb.keyRelease(KeyEvent.VK\_C); | |

|  |  |  |
| --- | --- | --- |
| 65 | Thread.sleep(2000); | |
| 66 |  |

|  |  |
| --- | --- |
| 67 | rb.keyPress(KeyEvent.VK\_ENTER); |
| 68 | rb.keyRelease(KeyEvent.VK\_ENTER); | |

|  |  |  |
| --- | --- | --- |
| 69 | Thread.sleep(2000); | |
| 70 | } |

|  |  |
| --- | --- |
| 71 |  |
| 72 | @After | |

|  |  |  |
| --- | --- | --- |
| 73 | public void tearDown() | |
| 74 | { |

|  |  |  |
| --- | --- | --- |
| 75 | driver.quit(); | |
| 76 | } |

|  |  |
| --- | --- |
| 77 | } |

**Code Walk-through**

**Import Statements**

**import java.awt.Robot** – Import this package prior to the script creation The package references to the Robot class in java which is required simulate keyboard and mouse events.

**import java.awt.event.KeyEvent** – The package allows the user to use keyPress and keyRelease events of keyboard.

**Object Creation for Robot class**  
Robot rb =new Robot();  
We create a reference variable for Robot class and instantiate it.

**KeyPress and KeyRelease Events**  
rb.keyPress(KeyEvent.VK\_D);  
rb.keyRelease(KeyEvent.VK\_D);

The keyPress and keyRelease methods simulate the user pressing and releasing a certain key on the keyboard respectively.

### ****Conclusion****

In this tutorial, we tried to make you acquainted with the WebDriver’s Alert class that is used to handle web based pop ups. We also briefed you about the Robot class that can be used to populate the value in the window based alert with the help of keyPress and keyRelease events.

**Article summary:**

* Alerts are a small box that appears on the display screen to give you some kind of information or to warn you about a potentially damaging operation or it may even ask you for the permissions for the operation.
* **There are popularly two types of alerts**–
  + Windows based alert pop ups
  + Web based alert pop ups
* Prior to the actual scripting, we need to import a package to be able to create a WebDriver script for handling a dropdown and making the Select class accessible.
* WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.
* void dismiss() – The dismiss() method clicks on the “Cancel” button as soon as the pop up window appears.
* void accept() – The accept() method clicks on the “Ok” button as soon as the pop up window appears.
* String getText()– The getText() method returns the text displayed on the alert box.
* void sendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.
* **Handling window based pop-ups** have always been a little tricky as we know Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support windows based applications and window alert is one of them.
* **Robot class** is a java based utility which emulates the keyboard and mouse actions and can be effectively used to handling window based pop up with the help of keyboard events.
* The keyPress and keyRelease methods simulate the user pressing and releasing a certain key on the keyboard respectively.