**Selenium Web Driver**

Program 1: To Verify the Title of Omayo blogspot Home Page

package newproject;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

//comment the above line and uncomment below line to use Chrome

//import org.openqa.selenium.chrome.ChromeDriver;

public class PG1 {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.firefox.marionette","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

//comment the above 2 lines and uncomment below 2 lines to use Chrome

//System.setProperty("webdriver.chrome.driver","E:\\ST LAB\\crossb\\chrom edriver.exe");

//WebDriver driver = new ChromeDriver();

String baseUrl = "https://omayo.blogspot.com/";

String expectedTitle = "omayo (QAFox.com)";

String actualTitle = "";

// launch Fire fox and direct it to the Base URL

driver.get(baseUrl);

// get the actual value of the title

actualTitle = driver.getTitle();

/\*

\* compare the actual title of the page with the expected one and print

\* the result as "Passed" or "Failed"

\*/

if (actualTitle.contentEquals(expectedTitle)){

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

} else {

System.out.println("Test Failed");

}

//close Fire fox

driver.close();

}

}

## Locating GUI Elements

Locating elements in WebDriver is done by using the "**findElement(By.locator())**" method.

Program 2: Program to locate a element by ID

package newproject;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class PG2 {

public static void main(String[] args) {

System.setProperty("webdriver.firefox.marionette","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

String baseUrl = "http://www.facebook.com";

String tagName = "";

driver.get(baseUrl);

tagName = driver.findElement(By.id("email")).getTagName();

System.out.println(tagName);

driver.close();

System.exit(0);

}

}

|  |  |  |
| --- | --- | --- |
| Variation | Description | Sample |
| By.className | finds elements based on the value of the "class" attribute | findElement(By.className("someClassName")) |
| By.cssSelector | finds elements based on the driver's underlying CSS Selector engine | findElement(By.cssSelector("input#email")) |
| By.id | locates elements by the value of their "id" attribute | findElement(By.id("someId")) |
| By.linkText | finds a link element by the exact text it displays | findElement(By.linkText("REGISTRATION")) |
| By.name | locates elements by the value of the "name" attribute | findElement(By.name("someName")) |
| By.partialLinkText | locates elements that contain the given link text | findElement(By.partialLinkText("REG")) |
| By.tagName | locates elements by their tag name | findElement(By.tagName("div")) |
| By.xpath | locates elements via XPath | findElement(By.xpath("//html/body/div/table/tbody/tr/td[2]/table/ tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td[3]/ form/table/tbody/tr[5]")) |

Get Commands

|  |  |
| --- | --- |
| get() Sample usage: | It automatically opens a new browser window and fetches the page that you specify inside its parentheses.  It is the counterpart of Selenium IDE's "open" command.  The parameter must be a String object. |
| getTitle() Sample usage: | Needs no parameters  Fetches the title of the current page  Leading and trailing white spaces are trimmed  Returns a null string if the page has no title |
| getPageSource() Sample usage: | Needs no parameters  Returns the source code of the page as a String value |
| getCurrentUrl() Sample usage: | Needs no parameters  Fetches the string representing the current URL that the browser is looking at |
| getText() Sample usage: | Fetches the inner text of the element that you specify |

Get commands fetch various important information about the page/element.

**Program 3: Alert Messgae display**

package mypackage;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class myclass {

public static void main(String[] args) {

System.setProperty("webdriver.firefox.marionette","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

String alertMessage = "";

driver.get("http://jsbin.com/usidix/1");

driver.findElement(By.cssSelector("input[value=\"Go!\"]")).click();

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

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

System.out.println(alertMessage);

driver.quit();

}

}

Instead of using the long "driver.findElement(By.locator())" syntax every time you will access a particular element, we can instantiate a WebElement object for it. The WebElement class is contained in the "org.openqa.selenium.\*" package.

[Creating your First Script in Webdriver](https://www.guru99.com/images/image016(2).png)

## Waits

There are two kinds of waits.

1. Implicit wait - used to set the default waiting time throughout the program
2. Explicit wait - used to set the waiting time for a particular instance only

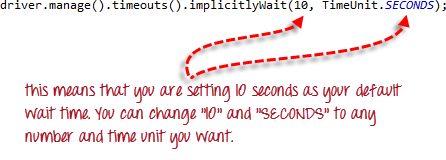
## Implicit Wait

* It is simpler to code than Explicit Waits.
* It is usually declared in the instantiation part of the code.
* You will only need one additional package to import.

To start using an implicit wait, you would have to import this package into your code.

[Creating your First Script in Webdriver](https://www.guru99.com/images/image038(1).png)

Then on the instantiation part of your code, add this.

[](https://www.guru99.com/images/image039(1).png)

## Explicit Wait

**Explicit waits are done using the WebDriverWait and ExpectedCondition classes**. For the following example, we shall wait up to 10 seconds for an element whose id is "username" to become visible before proceeding to the next command. Here are the steps.

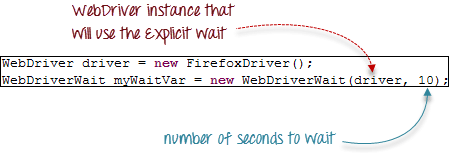
**Step 1**

Import these two packages:

**[Creating your First Script in Webdriver](https://www.guru99.com/images/image040(1).png)**

**Step 2**

Declare a WebDriverWait variable. In this example, we will use "myWaitVar" as the name of the variable.

[](https://www.guru99.com/images/image041(1).png)

**Step 3**

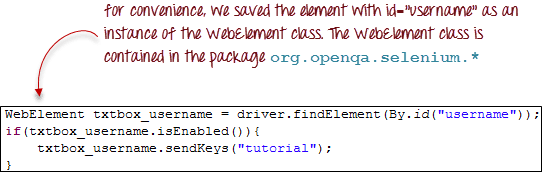
Use myWaitVar with ExpectedConditions on portions where you need the explicit wait to occur. In this case, we will use explicit wait on the "username" (Mercury Tours HomePage) input before we type the text "tutorial" onto it.

[Creating your First Script in Webdriver](https://www.guru99.com/images/image042(1).png)

**Conditions**

Following  methods are used  in conditional and looping operations --

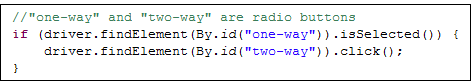
* **isEnabled()** is used when you want to verify whether a certain element is enabled or not before executing a command.

[](https://www.guru99.com/images/image043(1).png)

* **isDisplayed()** is used when you want to verify whether a certain element is displayed or not before executing a command.

[Creating your First Script in Webdriver](https://www.guru99.com/images/image044(1).png)

* **isSelected()** is used when you want to verify whether a certain **check box, radio button, or option in a drop-down box** is selected. It does not work on other elements.

[](https://www.guru99.com/images/image045.png)

## Using ExpectedConditions

The ExpectedConditions class offers a wider set of conditions that you can use in conjunction with WebDriverWait's until() method.

Below are some of the most common ExpectedConditions methods.

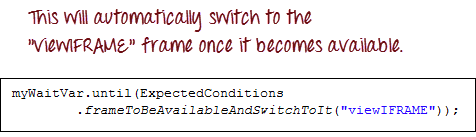
* **alertIsPresent()**- waits until an alert box is displayed.

[Creating your First Script in Webdriver](https://www.guru99.com/images/image046(1).png)

* **elementToBeClickable()** - Waits until an element is visible and, at the same time, enabled. The sample code below will wait until the element with id="username" to become visible and enabled first before assigning that element as a WebElement variable named "txtUserName".

[Creating your First Script in Webdriver](https://www.guru99.com/images/image047(1).png)

* **frameToBeAvailableAndSwitchToIt()**- Waits until the given frame is already available, and then automatically switches to it.

[](https://www.guru99.com/images/image048(1).png)

## Catching Exceptions

When using isEnabled(), isDisplayed(), and isSelected(), WebDriver assumes that the element already exists on the page. Otherwise, it will throw a **NoSuchElementException**. To avoid this, we should use a try-catch block so that the program will not be interrupted.

WebElement txtbox\_username = driver.findElement(By.id("username"));

try{

if(txtbox\_username.isEnabled()){

txtbox\_username.sendKeys("tutorial");

}

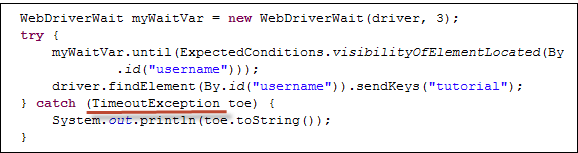
}

catch(NoSuchElementException nsee){

System.out.println(nsee.toString());

}

If you use explicit waits, the type of exception that you should catch is the "TimeoutException".

[](https://www.guru99.com/images/image050(2).png)