<http://www.softwaretestinghelp.com/selenium-framework-design-selenium-tutorial-21/>

Learn from Selenium videos on

https://www.youtube.com/user/MrBhanupratap29/playlist

Explaination for webdriver driver = new firefoxdriver()

Simple answer to this is "WebDriver is an Interface, and we are defining a reference variable (driver) whose type is an interface. Now any object we assign to it must be an instance of a class (FireFoxDriver) that implements the interface."

=================================================================================

Working with multiple browser windows:

http://www.seleniumeasy.com/selenium-tutorials/perform-operations-on-new-window-using-webdriver

There are cases where we need to open new window and perform operations or there may be cases where after clicking on any button / link, it opens new window and need to perform operations on the new window.

Let us look into such example:

Test case: We need to open '[http://linkedin.com](http://linkedin.com/)' and click on 'Help Center' link at the bottom which will open new window.  
1. Verify the title of the new window  
2. Verify text 'Welcome' on the page.  
3. Search for a Question with text "Frequently Asked Questions" and verify the result.

package com.pack;

import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.Assert;

import org.testng.annotations.Test;

public class WindowExamples {

public static WebDriver driver;

@Test

public void verifySearchInNewWindow() throws InterruptedException {

driver = new FirefoxDriver();

driver.navigate().to("[http://linkedin.com/"](http://linkedin.com/));

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

String mainHandle = driver.getWindowHandle();

//Wait for the element to be present

WebDriverWait wait = new WebDriverWait(driver, 5);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".cust-svc-link")));

driver.findElement(By.linkText("Help Center")).click();

//Switch to new window and verify the title

waitForNewWindowAndSwitchToIt(driver);

String newTitle = getCurrentWindowTitle();

Assert.assertEquals(newTitle, "LinkedIn Help Center", "New window title is not matching");

//Verify the text present on the page

String textOnpage=driver.findElement(By.cssSelector(".welcome")).getText().trim();

Assert.assertEquals(textOnpage, "Welcome!");

//Verify search text on the page

String searchText="Frequently Asked Questions";

WebElement searchInputBox=driver.findElement(By.id("kw"));

searchInputBox.sendKeys(searchText);

WebElement searchButton = driver.findElement(By.cssSelector(".button.leftnoround.blue"));

searchButton.click();

WebElement resultedElement = driver.findElement(By.cssSelector(".rn\_Element2"));

String resultedText = resultedElement.getText().trim();

System.out.println(resultedText);

Assert.assertTrue(resultedText.contains(searchText), "Search successfull");

closeAllOtherWindows(driver, mainHandle);

}

To execute the above test we have created a methods which can be reused with multiple tests.

**Below method is used to get the main window handle. we will the driver as parameter.**

**public** **static** String **getMainWindowHandle**(WebDriver driver) {

**return** driver.getWindowHandle();

}

**Below method is used to get the current window title**

**public** **static** String **getCurrentWindowTitle**() {

String windowTitle = driver.getTitle();

**return** windowTitle;

}

**Below method is used to close all the other windows except the main window.**

public static boolean closeAllOtherWindows(WebDriver driver, String openWindowHandle) {

Set<String> allWindowHandles = driver.getWindowHandles();

for (String currentWindowHandle : allWindowHandles) {

if (!currentWindowHandle.equals(openWindowHandle)) {

driver.switchTo().window(currentWindowHandle);

driver.close();

}

}

driver.switchTo().window(openWindowHandle);

if (driver.getWindowHandles().size() == 1)

return true;

else

return false;

}

**Below method is used to wait for the new window to be present and switch to it.**

public static void waitForNewWindowAndSwitchToIt(WebDriver driver) throws InterruptedException {

        String cHandle = driver.getWindowHandle();

        String newWindowHandle = null;

        Set<String> allWindowHandles = driver.getWindowHandles();

        //Wait for 20 seconds for the new window and throw exception if not found

        for (int i = 0; i < 20; i++) {

            if (allWindowHandles.size() > 1) {

                for (String allHandlers : allWindowHandles) {

                    if (!allHandlers.equals(cHandle))

                    newWindowHandle = allHandlers;

                }

                driver.switchTo().window(newWindowHandle);

                break;

            } else {

                Thread.sleep(1000);

            }

        }

        if (cHandle == newWindowHandle) {

            throw new RuntimeException(

                    "Time out - No window found");

        }

    }

}

Note: When ever we work on multiple windows, switching plays major role. We should switch to the desired window to perform operations and again switch back to default window to work on main window.

=================================================================

Uploading a file with Selenium Webdriver

There are different ways to handle file uploads with Selenium Webdriver.

The First and the Easy way is simple case of just finding the element and typing the absolute path of the document into it.

HTML code should look similar to this :

<input **type**="file" name="datafile">

Syntax:

//Find the element of upload button **and** **send** the path

WebElement element= driver.findElement(By.name("datafile"));

element.sendKeys("C:\Users\Easy\Desktop\testfile.txt");

[**Click here for example program**](http://seleniumeasy.com/selenium-tutorials/uploading-file-with-sendkeys-method)

**NOTE:** This works only when the textbox is enabled. So please make sure that the input element is visible.

**The OTHER WAY to Handle File Upload**

What If there is no text box / input tag to send the file path using Sendkeys method????

You will have only a button (Customized button) and when click on browse button, browser opens a window popup and to select the file from windows, And as we know selenium will not support OS controls.

We can use AutoIt tool (open source tool) which takes the responsibility to upload the file.

**About AutoIT : AutoIt v3 is a freeware BASIC-like scripting language designed for automating the Windows GUI and general scripting.**

We need to call the AutoIt script after clicking on the upload button. Immediatly ater clicking on Upload button, the control should be transfered to AutoIt which takes care of uploading the file.

Syntax:

Runtime.getRuntime().exec("AutoIt .exe filepath");

Download and Install Autoit tool from [Download AutoIt](http://www.autoitscript.com/site/autoit/)

Example:

Runtime.getRuntime().exec("D:/fileupload.exe");

[**Click here for example program**](http://seleniumeasy.com/selenium-tutorials/upload-a-file-using-selenium-webdriver-with-autoit)

**AutoIt script should have the following:**

**Step 1**: First After clicking on 'Upload' button, the cursor will move to the Window popup.

The below step should be the first line in the script. 'File Upload' is the name of the window popup when opened with Mozilla. The name changes depending on the browser.

Mozilla - File Upload  
Chrome - Open  
and for IE - Choose File to Upload

W**in**WaitActive("File Upload")

**Step 2**: Once the window popup is active, we should send the path of the document which need to be uploaded.

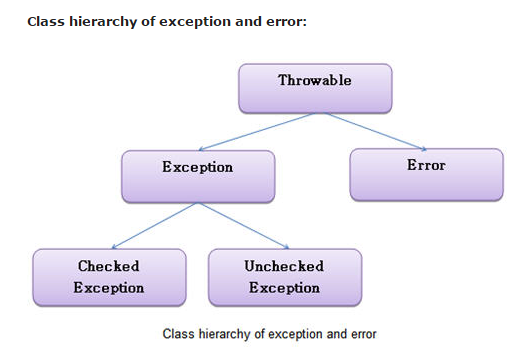
Send("Full path of the document")

**Step 3**: After that we need to click on Open button which will upload the document.

Send("{ENTER}")

=========================================================================

Exception Handling in Selenium (Very Important topic in Selenium)



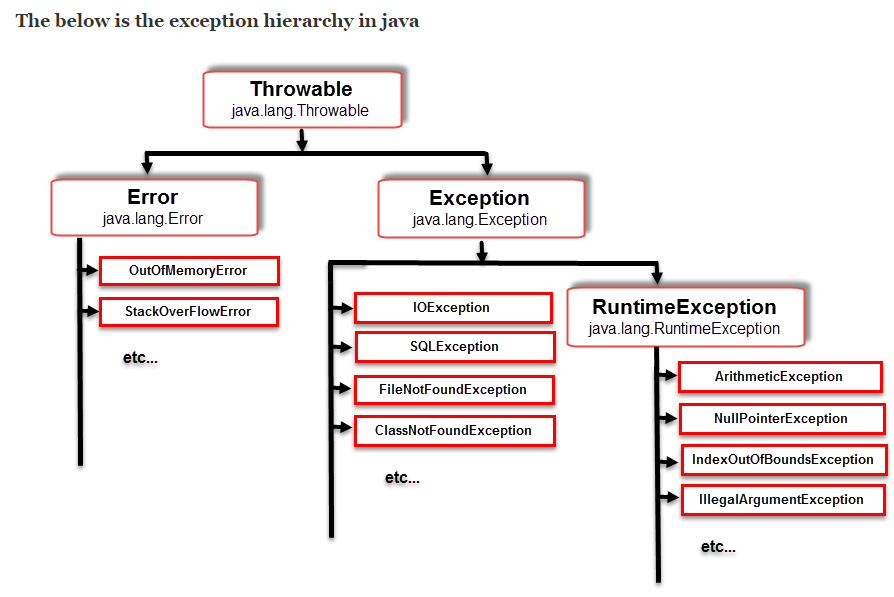
Difference between Exception and Error:

**Exception:** Exception occurs in the programmers code which can be handled and resolvable.

---->>>>>>> **Example:** AritmeticException, DivideByZeroException, NullPointerException, ClassNotFoundException etc

**Error:** Errors are not resolvable by programmer. Error occurs due to lack of system resources

---->>>>>>> **Example:** Stack over flow, hardware error, JVM error etc.



***Types of Exception: - There are two types of Exceptions:***  
**1.Checked Exceptions**  
**2.Unchecked Exceptions**

### **Checked Exceptions:**

Checked Exceptions are checked at **compile time only**, these are must and should handled by the programmer. Compiler will check at compile time whether these exceptions are handled or not if not compile time error occurs. Some of the checked exceptions are **IOException**, **FileNotFoundExpection**,**ClassNotFoundException** etc.

'*ClassNotFoundException*' Thrown when an application tries to load in a class through its string name using the below, but no definition for the class with the specified name could be found.

The '**forName**' method in class Class.  
The '**findSystemClass**' method in class ClassLoader.  
The '**loadClass**' method in class ClassLoader.

Whenever we try to interact with database, we try to load a JDBC driver using 'Class.forName', Check the example here to load JDBC driver when working on [Database testing with selenium](http://seleniumeasy.com/selenium-tutorials/database-testing-example-with-selenium-using-java)

'*FileNotFoundExpection' and 'IOException' , Whenever you try you to read or write from a File object that does not have a corresponding file on the disk you'll get IOExceptions.*

Possibly there are three cases where a '*FileNotFoundException*' may be thrown.

***The system cannot find the file specified*** - Which can be resolved by specifying the correct path of the file.

***Access to the file is denied*** - If the given file is inaccessible, when ever you try to write on the file which is 'ReadOnly', then we will get this exception. So we should make sure to have write access on the file to avoid exception.

***The process cannot access the file because it is being used by another process*** - When ever the file is opened by any another program, then we will get this exception. Make sure close the file after accessing it.

### **Unchecked Exceptions:**

unchecked exceptions are not checked by compiler at the time of compilation. the exceptions which are **extended by RuntimeException** class are all unchecked exceptions. Some of the unchecked exceptions are **AritmeticException**, **NullPointerException** etc

In selenium we see unchecked exceptions such as **NoSuchElementException**, **StaleElementReferenceException** , **NoSuchWindowException**, **TimeoutException** etc.

**NoSuchElementException** - Thrown by WebDriver.findElement(By by) and WebElement.findElement(By by) - When ever the element is not found in DOM, you will get this exception.

**StaleElementReferenceException** - This Exception occurs when driver is trying to perform action on the element which is no longer exists or not valid. You can here for more alternate ways of [Handling StaleElementReferenceException](http://seleniumeasy.com/selenium-tutorials/staleelementreferenceexception-in-selenium-webdriver). Mainly consider a webpage where you have a click button, and a test box below it. Now, when you click on button the text has to change in textbox below. Each time you click button text changes. If you try to verify text has changed or not in that case we need to put wait for javascript to complete. And then check for text box whether it got updated or not. In such case we get StaleElementReferenceException, because after clicking the button the content is changing. Another possibility is like, consider you click on a button and accordingly more fields appears on page, in this case DOM gets changed.

**NoSuchWindowException**- When ever driver tries to switch to the window which is not available, it will throw 'NoSuchWindowException'. We need to check the window id that we pass or wait for some time until the new window appears. Check here for [working with windows](http://seleniumeasy.com/selenium-tutorials/perform-operations-on-new-window-using-webdriver)  
**NoSuchFrameException** - When ever driver is Unable to navigate to frame with element, it throws 'NoSuchFrameException'. Check here for [Working with Frames](http://seleniumeasy.com/selenium-tutorials/how-to-work-with-iframes-in-selenium-webdriver)  
**TimeoutException**- When the Element was not displayed in the specified time. When ever we work with waits, we may see these exceptions. Check the example below:

/\*Explicit **wait** **for** **state** dropdown field\*/

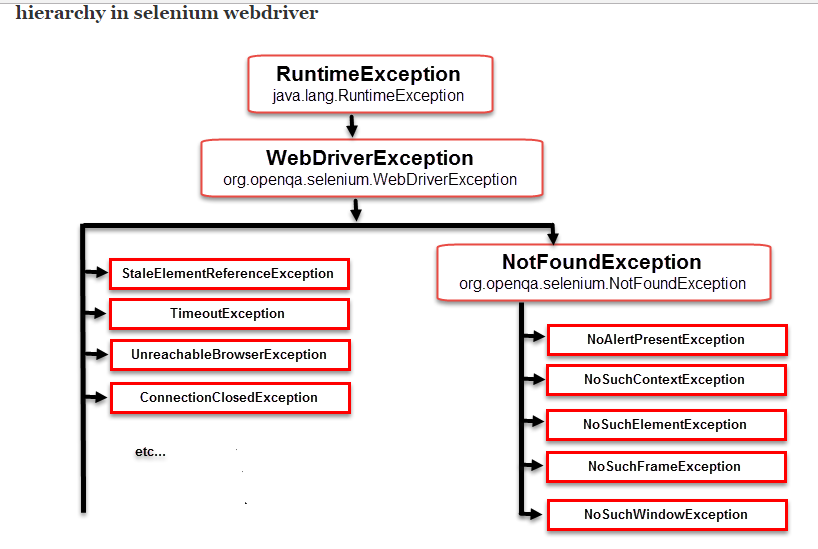
    WebDriverWait **wait** = new WebDriverWait(driver, 10);

**wait**.**until**(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

The exceptions that i have listed above are selenium webdriver exceptions. Here selenium webdriver class 'WebDriverException' extends 'RunTimeException' class. In turn there are sub classes like 'StaleElementReferenceException', 'TimeoutException', etc which extends 'WebDriverException'.

There is an other class which extends 'WebDriverException' is 'NotFoundException' and again this has sub classes 'NoSuchElementException', 'NoSuchFrameException' etc. Let us now look at the **exception hierarchy in selenium webdriver**



#### **The below are the five keywords which plays the role in Exception handling : -**

*1. try*  
*2. catch*  
*3. finally*  
*4. throw*  
*5. throws*

### **1. try:**

try block contains the code that might raise an exception.

**syntax:**

**try**{

statements....*//this code may raise an exception*

}

**try block Example:**

try{

**int** **x**=0;

**int** **y**=10;

   System.out.println(y/x);// here we will get an exception as **x** is initialized as 0. so we should place these code in try block.

}

### **2. catch:**

catch block contains handling code if any exception occurs in try block. try must follows catch block. try after catch or finally is mandatory.

**Syntax:**

**catch**(**Exception** e){

statements....*//contains handling code*

}

**Example:**

try{

**int** **x**=0;

**int** **y**=10;

System.out.println(y/x);

}

catch(Exception e){

System.out.println("Exception has been handled" + e);//once an exception raises instead of terminating the flow of program cursor jumps to particular catch block to handle this exception **and** prints a user friendly

message **and** the rest of code executes normally

}

System.out.println("code after try catch block");

}

when an exception raises it creates the exception object e. This object contains name of the exception class, cause and location of exception. if we already know the exception class we can directly pass the particular exception class as parameter in catch block. if we dint know what kind of exception class we can pass the Exception class as parameter as it is the parent class for all exception classes.

#### **Exception Information displaying methods are:**

**1.printStackTrace():** prints the stack trace , exception name and description.  
**2.toString():** returns a text message describing the exception name and description.  
**3.getMessage():** displays the description of exception

These three methods are present in Throwable class.

package com.seleniumeasy.ExceptionHandling;

**public** **class** ExceptionMethods {

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

**try**{

**int** x=0;

**int** y=10;

System.**out**.println(y/x);

}

**catch**(ArithmeticException ae){

ae.printStackTrace();

System.**out**.println(ae.toString());

System.**out**.println(ae.getMessage());

}

  }

}

**output:**  
java.lang.ArithmeticException: / by zero  
java.lang.ArithmeticException: / by zero  
/ by zero  
at com.seleniumeasy.ExceptionHandling.ExceptionMethods.main(ExceptionMethods.java:9)

**multiple catch blocks:**  
try block can follow multiple catch blocks. if there is a chance of getting multiple exceptions then we go for multiple catch blocks to handle all the exceptions at once.

**syntax:**

**try** {

statements..

}

**catch**(){

statements..

}

**catch**(){

statements..

}

**Example:**

package com.seleniumeasy.ExceptionHandling;

**public** **class** CatchDemo {

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

**try**

  {

**int** c[]={1,2,3};*// taking an array of size 3*

   System.**out**.println(c[2]);*// here we are printing the value*

*//c[3]=5;// if we uncomment this statement exception raises and cursor jumps to catch block of ArrayIndexoutofboundsexception without executing the next line.*

**int** a=10;*//*

   System.**out**.println(a/0) ;*// if we comment the above statement of c[3]=5; this will execute and here an exception raises as ArithmeticException*

  }

**catch**(ArithmeticException ae)

  {

   System.**out**.println("number cannot divided by zero"+ae);

  }

**catch**(ArrayIndexOutOfBoundsException aie)

  {

   System.**out**.println("array index out of bound exception"+aie);

  }

**catch**(Exception e){

System.**out**.println(e);*// this block will be executed when there were no particular exception matches*

}

System.**out**.println("After try catch block");

}

}

**the order of parameters in catch block is child to parent.**

**Example:**

**try** {

**int** c[]={1,2,3};

c[3]=5;

**int** a=10;

System.**out**.println(a/0) ;

}

**catch**(ArithmeticException ae)   {

System.**out**.println("number cannot divided by zero"+ae);

  }

**catch**(ArrayIndexOutOfBoundsException aie)   {

System.**out**.println("array index out of bound exception"+aie);

}

**catch**(Exception e){

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

}

and the order should not be parent to child. because once parent has caught the exception and again a child class is trying to caught the exception at the time we get a compile time error saying unreachable catch block for child exception: it is already handled by the catch block for Exception

### **3. finally block:**

finally block is a block which executes whether exception raises or not. that is cleanup code like open files or database connections should be placed in finally block.  
if we place these type of code at the bottom of try block if any exception raises in try block these code wont we executed and if we place in catch block if at all exception is not raised catch block wont be executed.

so we go for finally block.

**syntax:**

**finally**{

cleanup code;

}

finally block will executes irrespective of exception raises or not and exception handled or not. finally block appears after catch block or after try block when there is no catch block. we cannot place middle of try and catch block.

**Example:**

package com.seleniumeasy.ExceptionHandling;

**public** **class** FinallyDemo {

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

**try**{

**int** x=5;

*//int x=0;*

**int** y=10;

System.**out**.println(y/x);*// here we wont get any exception because we initialize x as number . if we uncomment x=0 we get Arithmetic exception but there is no handling code now program terminates abnormally but executes code inside finally block*

}

**catch**(NullPointerException ne){

System.**out**.println("ne");

}

**finally**{

    System.**out**.println("finally block");

    }

System.**out**.println("code after try catch block");

}

}

finally always executes even if there is return statement in try or catch block. only the situations where finally wont be execute are if System.exit() method called, any thread interrupted and if there is any exception in finally block

**example:**

try{

System.out.println("try block");

System.exit(0);

System.out.println(10/0);

}

catch(Exception e){

System.out.println(e);

}

finally{

    System.out.println("finally block");

    }

System.out.println("code after try catch block");

}

**Note:  
1. try always follow catch or finally block.  
2. there may be nested try, multiple catch blocks for each try but there should be only one finally block.  
3. In between try catch there should not be finally block.  
4. try with finally is possible without catch.  
5. there should not be any code in between try, catch or finally block**

### **throw:**

till now what we have seen is any exception object is automatically created and thrown. we can also create our own exception object explicitly using throw key word.

**syntax:**

**throw** **new** ExceptionclassName();

**Example:**

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

**throw** **new** ArithmeticException("number divided by zero");*// here we can give our own message to display*

}

**output:**

Exception **in** thread "main" java.lang.ArithmeticException: number divided **by** zero

at com.seleniumeasy.ExceptionHandling.ThrowDemo.main(ThrowDemo.java:6)

*we can use throw for checked, unchecked and user defined exceptions.*

##### **Handling checked Exceptions:**

If there is a chance of raising a checked exception we must and should do one of the following two things:

**1. handling the exception by using try catch block or**

**2. specify throws clause on the method there may be a chance of raising an exception.**

### **throws:**

throws clause is used for handling checked exceptions(SQLException, FileNotFoundException, IOExceptions). Any method contains the statements that may throw an checked exception must include throws clause. Otherwise we get compile time error saying "unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown"

**Syntax:**

methodname() **throws** ExceptionName{

statements;

}

**Example program without handling checked exception:**

**package** com.seleniumeasy.ExceptionHandling;

**import** java.io.FileInputStream;

**public** **class** **ThrowsDemo** {

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

openFile("F:\\seleniumfile.txt");

}

**public** **static** **void** **openFile**(String name){

FileInputStream f= **new** FileInputStream(name);

}

}

In the above program we get compile time error saying "unhandled exception type FileNotFoundException" as it is checked exception there may be chance of file not existing.

Now we need to either catch the exception by using try catch block or by declaring throws clause to the method.

If suppose we don't want to deal with writing code for try catch, simply add a throws clause to the openFile method's declaration which indicates that openFile() that might throw FileNotFoundException

**Example:**

**public** **class** **ThrowsDemo** {

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

openFile("F:\\seleniumfile.txt");

}

**public** **static** **void** **openFile**(String name) **throws** FileNotFoundException{

FileInputStream f= **new** FileInputStream(name);

}

}

We can add more than one checked exception by separating with comma.

**Example:**

public static void main(String[] args) throws FileNotFoundException, IOException{  
openFile("F:\\seleniumfile.txt");  
}

#### **User defined exceptions:**

We can create our own exceptions called user defined exceptions or customized exceptions. In order to create our own exceptions we need to extend Runtime Exception in our class.

package com.seleniumeasy.ExceptionHandling;

**public** **class** **MyOwnExceptionDemo** {

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

**throw** **new** MyException("hello");

}

}

**class** **MyException** **extends** **RuntimeException**{

**public** MyException(String s) {

super(s);

}

}

NOTE: Most people often do as below, because people are lazy to consider what to catch and what to throw. Throwing Exception is a bad practice and should be avoided. Make sure you handle it and always declare the most precise exception possible which will help the developer to decide on how to deal with it.

**public** **void** someTestMethod **throws** Exception{

*//bla bla*

}

========================================================================

Java Collections:

Difference between Iterator vs ListIterator

## Iterator vs ListIterator

1) Iterator is used for traversing List and Set both.

We can use ListIterator to traverse List only, we cannot traverse Set using ListIterator.

2) We can traverse in only forward direction using Iterator.

Using ListIterator, we can traverse a List in both the directions (forward and Backward).

3) We cannot obtain indexes while using Iterator

We can obtain indexes at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose.

4) We cannot add element to collection while traversing it using Iterator, it throws ConcurrentModificationException when you try to do it.

We can add element at any point of time while traversing a list using ListIterator.

5) We cannot replace the existing element value when using Iterator.

By using set(E e) method of ListIterator we can replace the last element returned by next() or previous() methods.

6) Methods of Iterator:

* hasNext()
* next()
* remove()

7) Methods of ListIterator:

* add(E e)
* hasNext()
* hasPrevious()
* next()
* nextIndex()
* previous()
* previousIndex()
* remove()
* set(E e)

## Difference between ArrayList Vs LinkedList

1) **Search**: ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n).

Reason: ArrayList maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements **doubly linked list** which requires the traversal through all the elements for searching an element.

2) **Deletion**: LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing first element) and O(1) in best case (While removing last element).

Conclusion: LinkedList element deletion is faster compared to ArrayList.

Reason: LinkedList’s each element maintains two pointers (addresses) which points to the both neighbor elements in the list. Hence removal only requires change in the pointer location in the two neighbor nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to be shifted to fill out the space created by removed element.

3) **Inserts Performance**: LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case. Reason is same as explained for remove.

4) **Memory Overhead**: ArrayList maintains indexes and element data while LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

There are few **similarities between** these classes which are as follows:

1. Both ArrayList and LinkedList are implementation of List interface.
2. They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List.
3. Both these classes are non-synchronized and can be made synchronized explicitly by using [Collections.synchronizedList](http://docs.oracle.com/javase/6/docs/api/java/util/Collections.html#synchronizedList%28java.util.List%29) method.
4. The iterator and listIterator returned by these classes are fail-fast (if list is structurally modified at any time after the iterator is created, in any way except through the iterator’s own remove or add methods, the iterator will throw a [ConcurrentModificationException](http://docs.oracle.com/javase/6/docs/api/java/util/ConcurrentModificationException.html)).

### When to use LinkedList and when to use ArrayList?

1) As explained above the insert and remove operations give good performance (O(1)) in LinkedList compared to ArrayList(O(n)). Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.

2) Search (get method) operations are fast in Arraylist (O(1)) but not in LinkedList (O(n)) so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

================================================================================

What is framework

A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

Why do we use It? E.g. we very often use lifts or elevators. There are a few guidelines those are mentioned within the elevator to be followed and taken care off so as to leverage the maximum benefit and prolonged service from the system.

Advantage of Test Automation framework

1. Reusability of code

2. Maximum coverage

3. Recovery scenario

4. Low cost maintenance

5. Minimal manual intervention

6. Easy Reporting

What framework you use in your project

* What aspects you consider while selecting a framework?
* We consider what dynamic and static inputs we need to handle
* What is the frequency of test we need to run
* What browsers application supports
* What is application under test’s compatibility and limitations
* How much test data we need to handle
* What Reporting structure is desired by business

===================================================

Reports

Defect logging

Software life cycle

Bug life cycle

Test Strategy

Test Plan

============================================================

**TestNG**

**Features of TestNG**

* Support for annotations
* Support for parameterization
* Advance execution methodology that do not require test suites to be created
* Support for Data Driven Testing using Dataproviders
* Enables user to set execution priorities for the test methods
* Supports threat safe environment when executing multiple threads
* Readily supports integration with various tools and plug-ins like build tools (Ant, Maven etc.), Integrated Development Environment (Eclipse).
* Facilitates user with effective means of Report Generation using ReportNG

**TestNG Hierarchy of Annotations**

•BeforeSuite

•BeforeTest

•BeforeClass

•BeforeMethod

•Test Case 1 @Test

•AfterMethod

•BeforeMethod

•Test Case 2 @Test

•AfterMethod

•AfterClass

•AfterTest

•AfterSuite

Other than these

@DataProvider: used to define data providing methods as DataProviders, attributes parallel, name: name for this data provider, if not specified name becomes name of method

@Factory

@Listeners attributes:

@Parameter attributes: value – describes how to pass parameters to this test

**Sample TestNG xml**

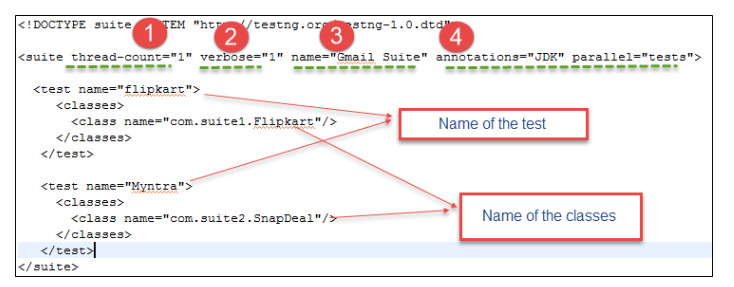
|  |  |
| --- | --- |
| 1 | <?xml version="1.0" encoding="UTF-8"?> |
| 2 | <suite name="TestNgMavenExampleSuite" parallel="false"> |

|  |  |
| --- | --- |
| 3 | <test name="TestNgMavenTest"> |
| 4 | <classes> |

|  |  |
| --- | --- |
| 5 | <class name="com.javacodegeeks.testng.maven.TestNgMavenExample"/> |
| 6 | <class name="com.javacodegeeks.testng.maven.TestNgMavenSecondClass"/> |

|  |  |
| --- | --- |
| 7 | </classes> |
| 8 | </test> |

|  |  |
| --- | --- |
| 9 | </suite> |



**Sample Maven pom.xml**

|  |  |
| --- | --- |
| 01 | <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>"> |
| 02 | <modelVersion>4.0.0</modelVersion> |

|  |  |
| --- | --- |
| 03 | <groupId>com.javacodegeeks.testng.maven</groupId> |
| 04 | <artifactId>testngMaven</artifactId> |

|  |  |
| --- | --- |
| 05 | <version>0.0.1-SNAPSHOT</version> |
| 06 | <dependencies> |

|  |  |
| --- | --- |
| 07 | <dependency> |
| 08 | <groupId>org.testng</groupId> |

|  |  |
| --- | --- |
| 09 | <artifactId>testng</artifactId> |
| 10 | <version>6.8.8</version> |

|  |  |
| --- | --- |
| 11 | <scope>test</scope> |
| 12 | </dependency> |

|  |  |
| --- | --- |
| 13 | </dependencies> |
| 14 | </project> |

**Running TestNG project**

<http://testng.org/doc/documentation-main.html#testng-xml>

TestNG can be invoked in different ways:

* Command line
* [ant](http://testng.org/doc/ant.html)
* [Eclipse](http://testng.org/doc/eclipse.html)
* [IntelliJ's IDEA](http://testng.org/doc/idea.html)

**Customized HTML Reports**

https://www.seleniumeasy.com/testng-tutorials/configuring-reportng-with-testng-to-generate-html-reports

**ReportNG** is used to print customized html reports as replacements to default TestNG Reports

Sample TestNG xml addition of listeners for ReportNG

<!DOCTYPE suite SYSTEM "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>

**All About report Generation in Selenium (Must Read)**

<http://www.techbeamers.com/generate-reports-selenium-webdriver/>

ExtentReports library is used to generate reports in form of pie-charts, benefits are as follows:

* Ability to generate dynamic HTML logs.
* Represents test case status with the help of PIE Charts.
* Generates step-by-step test case summary.
* Ability to filter reports based on test status.
* It maintains execution history.
* It captures details like OS, Memory, Java version and so on.
* You can attach error screenshots within the report.

**Continuous Integration and Build Tool (CI)**

**Hudson**

<http://www.softwaretestinghelp.com/hudson-continuous-integration-tool-selenium-tutorial-25/>

Continuous Integration can be performed automatically. Hudson is one of the popularly known tools to perform Continuous Integration. Hudson is a Java based open source Continuous Integration tool. Like any other Continuous Integration tool, Hudson provides the teams to trigger builds and tests with any change in the Source Control Management System.

Features

* Supports SCM tools like CVS, Subversion (SVN), Git etc.
* Is capable of building ANT based projects, Maven based projects etc.
* Is capable of executing shell scripts and Windows batch commands
* Is capable of Sending reports, notifications etc via Email, SMS, Skype etc.

=================================================================

**Open a new tab using Selenium**

Yes, you can do that easily using Selenium. Use the Key commands (Ctrl +T) to open a new tab and then use Ctrl +Tab (Ctrl +\t) command to switch to newly opened tab and perform whatever is necessary. It would go something like this  
  
 //to open a new tab

driver1.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"t");

//to close tab newly opened

driver1.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"w");

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

//to open a new tab again

driver1.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"t");

//to switch back to main tab

driver1.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"\t");

//to switch back to new tab

driver1.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"\t");

==============================================================================

**Synchronization in Selenium**

It is a mechanism which involves more than one components to work parallel with Each other.

Generally in Test Automation, we have two components

1. Application Under Test

2. Test Automation Tool.

Synchronization can be classified into two categories:

1. Unconditional

wait() and Thread.sleep(time in miliseconds);

1. Conditional Synchronization

* Page Load Synchronization

We can set the default page navigation timeout. Below statement will set the navigation timeout as 50. This means that selenium script will wait for maximum 50 seconds for page to load. If page does not load within 50 seconds, it will throw an exception.

driver.manage().timeouts().pageLoadTimeout(50,TimeUnit.SECONDS);

* **Implicit wait ( Element Synchronization)**

We can set the default element existence timeout. Below statement will set the default object synchronization timeout as 20. This means that selenium script will wait for maximum 20 seconds for element to exist. If Web element does not exist within 20 seconds, it will throw an exception.

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

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

* **Explicit wait (Synchronization based on specific condition)**

We can also instruct selenium to wait until element is in expected condition.

To use this kind of synchronization, you will have to import WebDriverWait class using below statement.

WebDriverWait wait = new WebDriverWait(driver, 10);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(“statedropdown”)));

**Various ExpectedConditions**

#1) elementToBeClickable() – The expected condition waits for an element to be clickable i.e. it should be present/displayed/visible on the screen as well as enabled.

Sample Code

wait.until(ExpectedConditions.elementToBeClickable(By.xpath(“//div[contains(text(),’COMPOSE’)]”)));

#2) textToBePresentInElement() – The expected condition waits for an element having a certain string pattern.

Sample Code

wait.until(ExpectedConditions.textToBePresentInElement(By.xpath(“//div[@id= ‘forgotPass'”), “text to be found”));

#3) alertIsPresent()- The expected condition waits for an alert box to appear.

Sample Code

wait.until(ExpectedConditions.alertIsPresent()) !=null);

#4) titleIs() – The expected condition waits for a page with a specific title.

Sample Code

wait.until(ExpectedConditions.titleIs(“gmail”));

#5) frameToBeAvailableAndSwitchToIt() – The expected condition waits for a frame to be available and then as soon as the frame is available, the control switches to it automatically.

Sample Code

wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id(“newframe”)));

**=========================================================**

**Navigation in Selenium**

driver.navigate().to(”URL”);

This command lets the user to launch a new web browser window and navigate to the specified URL.

driver.navigate().back();

The above command requires no parameters and takes back the user to the previous webpage in the web browser’s history.

driver.navigate().forward();

This command lets the user to navigate to the next web page with reference to the browser’s history.

driver.navigate().refresh();

This command lets the user to refresh the current web page there by reloading all the web elements.

===================================================================

Scroll Up Down

Working with Check box Radio Buttons Drop Downs

Finding locator values

Sample Maven Project and Execution

Sample CI tool if possible

Reading and writing to Excel Sheets

JDBC

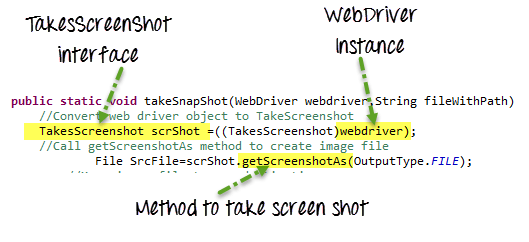
GIT concepts

Uploading and downloading from web page

==============================

Capturing Screenshots

* We need to type cast our WebDriver object to TakeScreenShot interface type.
* Then we can use getScreenShotAs(OutputType.FILE) method to get File type object.
* Now we need to copy the captured screen shot file a location into system. We can pass that file object to FileUtils.copyFile(file, new File (Constant.SCREEN\_SHOT\_LOCATION+”.png”));



Now to take screen shot just call this static method.

public static void takeScreenshot(WebDriver driver, String TestName)throws Exception

{

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

//The below method will save the screen shot in d drive with name "screenshot.png"

FileUtils.copyFile(scrFile, new File(Constant.SCREEN\_SHOT\_LOCATION+”.png"));

}

Capturing Screenshot on failure

If we want to take screen shot for failure in any method/step just enclose that method in try catch block and call your takeScreenShot().

Other way:

http://learn-automation.com/how-to-capture-screenshot-for-failed-test-cases-in-selenium-webdriver/

@AfterMethod

public void teardown(ITestResult result){

if(ITestResult.FAILURE==result.getstatus()){

Utility.captureScreenshot(driver, result.getName())

Video link to learn testing

<http://learn-automation.com/testng-tutorials-for-beginners/>

}

=======================

Highlighting an Element and take Screen Shot

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class Highlight {

 public static void main(String []args){

 WebDriver driver=new FirefoxDriver();

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

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

 // Create the  JavascriptExecutor object

  JavascriptExecutor js=(JavascriptExecutor)driver;

// find element using id attribute

  WebElement username= driver.findElement(By.id("email"));

 // call the executeScript method

  js.executeScript("arguments[0].setAttribute('style,'border: solid 2px red'');", username);

 }

}

================================

Logging for Debugging Selenium

http://learn-automation.com/how-to-create-log-files-in-selenium/

Log4J

* We need log4j.properties files into class path, copy and paste below code

// Here we have defined root logger

log4j.rootLogger=INFO,CONSOLE,R,HTML,TTCC

// Here we define the appender

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.R=org.apache.log4j.RollingFileAppender

log4j.appender.TTCC=org.apache.log4j.RollingFileAppender

log4j.appender.HTML=org.apache.log4j.FileAppender

// Here we define log file location

log4j.appender.R.File=./log/testlog.log

log4j.appender.TTCC.File=./log/testlog1.log

log4j.appender.HTML.File=./log/application.html

// Here we define the layout and pattern

log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout

log4j.appender.CONSOLE.layout.ConversionPattern= %5p [%t] (%F:%L)- %m%n

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d - %c -%p - %m%n

log4j.appender.TTCC.layout=org.apache.log4j.TTCCLayout

log4j.appender.TTCC.layout.DateFormat=ISO8601

log4j.appender.HTML.layout=org.apache.log4j.HTMLLayout

log4j.appender.HTML.layout.Title=Application log

log4j.appender.HTML.layout.LocationInfo=true

Then use below code to start with logging

// Here we need to create logger instance so we need to pass Class name for

//which we want to create log file in my case Google is classname

Logger logger=Logger.getLogger(Log.class.getName());

// configure log4j properties file

PropertyConfigurator.configure("Log4j.properties");

// Open browser

WebDriver driver = new FirefoxDriver();

logger.info("Browser Opened");

// Set implicit wait

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

logger.info("Implicit wait given");

// Load application

driver.get("https://www.google.co.in/");

logger.info("Url opened");

// type Selenium

driver.findElement(By.name("q")).sendKeys("Selenium");

logger.info("keyword type");

Now to implement in a project, create a class for logging in say your utility package, initialize the logger object and PropertyConfigurator.configure(“log4j.properties”);

Then implement your own methods to explicitly define the way you wanna record log.

Say public static void info (String message){

Logger.info(message);

}

public static void startTestCase(String sTestCaseName){

Log.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Log.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Log.info("$$$$$$$$$$$$$$$$$$$$$ "+sTestCaseName+ " $$$$$$$$$$$$$$$$$$$$$$$$$");

Log.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Log.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

=====================================================

Working with Frames in Selenium

Driver.switchTo.frame(“fame-name”);

Driver.switchTo.frame(“frame-index”);

Driver.switchTo.defaultContent(); // move to parent frame

Driver.getElement(By.tagName(“iframe)).size();//to find how many frames in a page

==========================

Handling Alerts

Alert class is used to handle web based pop-ups

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

alert.accept();

alert.dismiss();

alert.getText();

alert.sendkeys(“String to be sent”);

=======================

**CVS commands**

cvs co MAIN/POSTARRIVE

cvs update –j LOAD1625 MAIN/POSTARRIVE

cvs commit –m “commit 1625 to trunk” MAIN/POSTARRIVE

cvs rtag –bBF –r 1.9 baseline MAIN/POSTARRIVE

cvs diff –r LOAD1625 –r baseline – MAIN/POSTARRIVE

============================================================

**Reading properties file in Selenium**

Public class DemoProperties{

Static Properties property;

private static void loadProperties() throws IOException{

property = new Properties();

File file = new File (System.getProperty(“user.dir”)+”\\src\\prop\\Test1.properties”);

FileReader reader = new FileReader(file);

property.load(reader);

//to load another properties file

File file1 = new File(System.getProperty(“user.dir”)+”\\src\\prop\\Test2.properties);

FileReader reader1 = new FileReader(file1);

property.load(reader1);

}

Public static String getValueFromProperty( String Key) throws IOException{

loadProperties();

String value = Property.getProperty(key);

Return value;

}

}

//Call this method from anywhere in your project

String cusName = DemoProperties.getValueFromProperty(“CustomerName”);

=================================

Working with IE browse

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;

import org.testng.annotations.Test;

public class IEBrowser {

@Test

public void test12() throws Exception{

// set driver path

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

// Initialise browser

WebDriver driver=new InternetExplorerDriver();

// Load google.com

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

// close browser

driver.close();

}

}

=======================

**3 Unique Ways To Handle File Upload In Selenium Webdriver**

<http://www.techbeamers.com/handle-file-upload-selenium-webdriver/>

==============

**How To Handle JQuery And Kendo Date Time Picker Using Webdriver**

<http://www.techbeamers.com/handle-date-time-picker-control-using-webdriver/>

================

Window Handles in Selenium