**Requirement of Selenium**

-Java (install and path variable is set)

-IDE (eclipse, Mars,Juno, Luma)

-Browser(Mozilla firefox ,IE , Google Chrome)

-Selenium jar

**Architecture of WebDriver**

SearchContext(I)

Extends

WebDriver(I)

Implements

RemoteWebDriver(C)

extends

extends extends

**Implementation of Selenium**

Interface WebDriver{

Public void get(String URL){}

Public void refresh(){}

Public void close(){}

}

Class RemoteWebDriver implements WebDriver{

Public void get(String URL){

Code to close

}

Public void refresh(){

Code to close

}

Public void close(){

Code to close

}

Class ChromeWebDriver extends RemoteWebDriver

{

}

Class FirefoxDriver extends RemoteWebDriver

{

}

Class InternetExplorer extends RemoteWebDriver

{

}

**Few points for Selenium**

* Open source
* Platform independed
* Support multiple browsers.
* It Support Web Application (Need to have DOM Structure )
* Multiple language support(Java, Ruby, perl , python,c#)

**How to Launch Firefox browser.**

Before launching the browser we have to set the path of Web browser using below code.

System.setProperty("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

Or

System.setProperty("webdriver.gecko.driver", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

Note: Set path of Web Browser will depend on network configuration and we need to try both the above ways to and check which one is suitable of us.

**Elements Identification in Selenium.**

To identify any elements of web page in selenium the locators are available .Locators are nothing but used for identify an element if an element is available on web page. The elements which are available on Web page are known as Web elements.

**Locators**

There are total 8 locator present in Selenium as below

1. tagname
2. classname
3. id
4. name
5. linkedText
6. partialLinkedText
7. cssSelector
8. xpath

Note: In case of CSS selector and xpath we have to drive them as its not present at web page.

**By class Implementation:**

class By{

public static By tagname(String tagname){

By ref=new By();

return ref;

}

Public static By className(String className){

public static By tagname(String tagname){

By ref=new By();

return ref;

}

--

--

--

Public static By cssSelector(String cssSelector)

{

public static By tagname(String tagname){

By ref=new By();

return ref;

}

Public static xpath(String xpath)

{

public static By tagname(String tagname){

By ref=new By();

return ref;

}

}

**Examples to find WebElements using Locators**

1. tagname

By bref=By.tagName(String tagName);

WebElement elements=driver.findElements(bref);

1. classname

driver.findElements(By.className(String className)

1. id

driver.findElement(By.id(String id)

1. name

driver.findElement(By.name(String name);

1. linkedText

driver.findElement(By.likedText(String linkedText)

1. partialLinkedText

driver.findElements(By.partialLinkedText(String partialLinkedText)

1. cssSelector

driver.findElement(By.cssSelector(String cssSelector)

1. xpath

driver.findElement(By.xpath(String xpath)

**Identify Elements Using CSS Selector:**

In order to identify an element using CSS selector or Xpath . We need to derived both as its not present direct to web page.

For identify an element using CSS selector or Xpath we should have 2 add on installed with Mozilla firefox these are as below

* -Firebug (inspection tool)
* -Firepath (Validation tool)

Steps to installed Fire bug and Firepath

**Firebug(It is an Inspecting tool)**

* Open Mozilla firefox and click on Menu option at right most corner .
* Navigate to Add on and search for Firebug and press enter.
* Installed Firebug ,Once its installed restart the Mozilla firefox once.

**Firepath(it is an addition to FirBug)**

* Open Mozilla firefox and click on Menu option at right most corner.
* Navigate to Add on and search for Firepath and press enter.
* Installed Firepath,Once its installed restart the Mozilla firefox once.

**Css Slector**

Anything start with ‘.’ Is known as class Name.

Anything starts with ‘#’ is known as ID

**CssSelector by using className**

<p class=”p1” >Welcome to selenium </p>

cssSelector=.p1

Generic syntax: - .className

**CssSelector by using ID:**

<input id=”lsl-ib”>

Generic Syntax :- #id

cssSelector: #lsl-ib

Note:For every css or xpath should give single matching node.

Special case :sometimes class attribute contains a space we need to insert a . in place of space

Eg <input class=”text field pwdfield”>

cssSelector =.textfield.pwdfield

<html>

<head> html

</head>

<body>

<a>

</body>

</html>

Tree Structure

Html

\_hea d Sublings

\_body(parent)

a(child)

**Forward Traversing and Backwards Traversing:**

Moving from parent node to child node is known as forward traversing and moving from child node to parent node is known as Backward Traversing.

Eg:

td

-<a id=”loginbutton”/>(parent)

<div> Longin</div>(child)

If we want to identify div then we can use parent child relationship

CssSlector=#loginbutton>div

Note: > symbol is used for forward traversing.

Eg:

<tr id=”textmail”>

td

a

div

Identification of div in above tree structure using css Selector:

#testmail>td>a>div

Note:

1-Css selector does not support backward traversing.

2- We can’t use Index to identify specific elements

**Css Selector Flavours:**

Syntex:tagname[attribute=”attribute-value”]

<input type=”password” name=”pass”/>

Css Selector=input[name=”pass”]

**Xpath:**

Xpath are 2 type

* Absolute Xpath
* Relative Xpath

E: Parent parent

C demo (Parent) parent

data.txt(Child)

Absolute path for data.txt-E:/demo/data.txt

Relative path for data.txt-./data.txt

Example 1:Find the Absolute xpath for input tag A and B using Index

<html>

<head>

</head>

<body>

<dev>

<input value="A"/>

<input value="B"/>

</dev>

</body>

</html>

A-/html/body/dev/input[1]

B-/html/body/dev/input[2]

Example 2:Find the Absolute xpath for input tag C,D ,E,F,F and a using Index

<html>

<head>

</head>

<body>

<dev>

<input value="C"/>

<input value="D"/>

</dev>

<dev>

<input value="E"/>

<input value="F"/>

</dev>

<dev>

<input value="G"/>

<a href="https://www.google.com>LinkedText</a>

</dev>

</body>

</html>

* Absolute Xpath for C-/html/body/dev[1]/input[1]
* Absolute Xpath for D-/html/body/dev[1]/input[1]
* Absolute Xpath for E-/html/body/dev[2]/input[1]
* Absolute Xpath for F-/html/body/dev[2]/input[2]
* Absolute Xpath for G-/html/body/dev[3]/input[1]
* Absolute Xpath for H-/html/body/dev[3]/a

**Xpath Using Grouping Index:**

* Write a xpath to identify a group of similar element (textbox)
* Enclose the xpath with in parenthesis.
* Provide occurrence value.

Ex

1-/html/body/dev/input

2-(/html/body/dev/input)

3-(/html/boby/dev/input)[1] it will indentify first occurrence of input tag.

Note: In automation we never use Absolute xpath.

**Relative Xpath**

Every relative xpath start with”//” or “.//”

Eg :

Html

Body

a

Relative xpath for anchor tag will be //a

**Note : Don’t use index in Xpath unless it necessary for identification of a web element.**

**Xpath Using Attribute**

**//tagname[@attribute=’attribute\_value’]**

Eg:

<input type=”text” name=”password”/>

Xpath-//input[@name=’password’]

Eg:

<input type=”lst-lb” class =”gsti” type=”test” lable=”search-value” title=”search”/>

**Xpath=//input[@type=’test’]**

**Note: In xpath we can use any of the available attribute but it should give only 1 matching node**

**Xpath Using text() functin**

//tagname[text()=’text’]

Eg:

<a href=<https://www.google.com>>GoogleLink</a>

**Xpath=//a[text()=’GoogleLink’]**

Note:test() function does exact pattern match and provide the exact text available between open or close tag.

Eg:

<td classs=”header”>please identify yourself</td>

**Xpath=//td[text() =’please identify yourslef’]**

Eg:

<label name=”console”> Keep me logged in</lable>

**Xpath=//label[text()=’Keep me logged in]**

**Xpath Using Contains with text() function**

**Syntax ://tagname[contains(text() , ‘Partial text’)]**

Eg:

<b>Capture letters are case sensitive and to be entered in upper case</b>

Xpath=//b[contains(text() ,’Capture’)]

Eg:

<a href=<https://www.google.com>> Business </a>

Xpath=//a[contain[text() ,’Business)]

Note:

* If the test is too long use contain function
* If a test is having space with starting or ending use contain

**Xpath using Contain with Attribute:**

Syntax ://tagname[contains(@attribute,’attribute value partial’)]

Eg:

<input class=”textField pwdfield”/>

Xpath-//input[contains(@class,’textField’)]

Eg:

<td class=”navItem newcell retrieve selected”/>

Xpath-//td[contains(class,’navItem’)]

Note:

* If attribute value is having space use contain with attribute
* If attribute value is too long use contain with attribute

**Approach to find parent using Xpath**

1-

In xpath for forward traversing (parent to child ) we use ‘/’(forward slash)

In xpath for backward traversing (child to parent ) we can use /.. (forward slash with double dot ) .It will identify immediate parent.

tbody

tr

td

dev <a id=”t2”/>

xpath as below

dev-//a[@id=’t2’]/..

td-//a[@id=’t2’]/../..

tr-//a[@id=’t2’]/../../..

2-

* Write a xpath for child
* Write a path from parent to child
* Replace single forward slashes with the set of square brackets

Ex:

tr

td

finding xpath for tr as below

1-//td

2-//tr/td

3-//tr[td]

Ex

tbody

tr

td

div

span

xpath for dev:

1-//span

2-//dev/span

3-//dev[span]

Xpath for tbody

1-//span

2-//tbody/tr/td/dev/span

3-//tbody[tr[td[dev[span]]]]

**Xpath for dynamic elements using independent and Dependent**

Steps:

* Identify independent and dependent elements
* Write the xpath separately for independent and dependent.
* Identify common immediate parent and write a path from common parent to dependent and independent separately (common parent is nothing but a node which we can reach independent as well as dependent a common parent should be immediate )

*Note :Activate firebug ->goto independent or dependent move the cursor to upside if the node is highlighting independent and dependent on a webpage consider it as common parent.*

* For dependent only remove common parent along with double forward slash .
* For independent only replace /slash by the set of square brace.
* Independent + Dependent.

**Priority for locator**

* Id
* Name
* Link

Linkedtext

partialLinkedText

* CssSelector
* Xpath

Which is the best locator –Its depends on webelements

Note : xpath is slowest and cssSelector is fastest locator in selenium.

**Basic Action on WebElement**

**Interface WebElement{**

**Public void sendkeys(Charactere seq){}**

**Public void click(){}**

**Public void getAttribute(){}**

**Public String getText(){}**

**Public Boolean isEnable(){}**

**Public Boolean isDisplayed(){}**

**Public Boolean isSelected(){}**

**}**

**Performing Action on a Link.**

**Scenario –**Goto google and click on gmail link

WebDriver driver=new FirefoxDriver();

driver.get(<https://www.google.com>);

driver.findElement(By.linkedText(“Gmail”).click();

click() is the method available in WebElement interface and used to perform action on webElement like button , link , radio button and checkbox.

Scenario:Goto google and search for selenium

driver.findElement(By.id(“lst-id”)).sendKeys(“selenium)

SendKeys is a method to perform action on textbox or text area it accept an argument of type characterSequence and sendKeys is the method available on WebElement interface.

Scenario:goto ://demo.actitime.com/login.do and perform login

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

driver.get("http://demo.actitime.com/login.do");

driver.findElement(By.*xpath*("//input[@id='username']")).sendKeys("admin");

driver.findElement(By.*xpath*("//input[@name='pwd']")).sendKeys("manager");

driver.findElement(By.*xpath*("//a[@id='loginButton']")).click();

Scenario :Goto irctc and enter username

If WebElement having multiple action we need to store it in reference variable and use. If single action is there on webElement directly perform an action.

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

driver.get("https://www.irctc.co.in/eticketing/loginHome.jsf");

WebElement un=driver.findElement(By.*id*("usernameId"));

un.sendKeys("rahul");un.sendKeys("singh");

un.clear();//aleternate way to clear text box

un.sendKeys(Keys.*CONTROL*,"a");

un.sendKeys("admin");

Scenario:Goto facebook.com and perform login without clicking on login button.

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

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

driver.findElement(By.*id*("email")).sendKeys("job.rahulsingh@gmail.com");

driver.findElement(By.*id*("pass")).sendKeys("password",Keys.*ENTER*);

How to get text available between open and close tag

String str=driver.findElement(By.xpath(//p)).getText();

Sop(str);

getText() is the method available in WebElement interface use to get a text available between open and close tag.

Note: value attribute always contains actual value which we enter.

Eg code

Html

<input id="username" class="textField" type="text" value="" name="username" placeholder="Username"/>

code

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

element.sendKeys("admin");

String str=element.getAttribute("value");

System.*out*.println(str);

How to get class name from webelement.

String str=element.getAttribute("class"); //use above code.

Tool Tip:

<div id=”hplog” title=”google”/>

WebElement element=driver.findElement(By.xpath(//id=’hplog’);

String str=Element.getAttribute(“title”);

**Selenium Api Action**

Click() Click on webElement

sendKey() Enter text on WebElement

Clear() to clear the data on Web element

getText() to get the text from the webelement

getAttribute(String) To get the apttribute value

Scenario: Goto gmail enter user name click on next button and provide password and click on login button.

Password page

Gmail user page

Empty Browser

Delay Delay

There will be delay in loading of password page we need to use Thread.sleep()

Synchronization:

Mapping the speed of execution w.r.t loading of the web page is known as Synchronization in automation

We can achieve this throw Synchronization in 3 ways

* Default Synchronization
* Static Wait
* Dynamic Wait

Default Synchronization

driver.get(“URL) and driver.navigate().to(“URL)

When a URL is launched on the a browser then get method will hold the execution until the loading of page is completed.get() method will check the status of loading after every 500 mille seconds .If the page is not loaded it will check the loading status on the browser this process is known as pooling

Default synchronization will work when we launch a URL using get() and happened only once.

Static wait:(Thread.Sleep())

This is used to pause the execution for a while

Once there is a delay in loading of the web page use static wait() to handle Synchronization issues.

Password page

User page

Thread.sleep()

Example

**public** **class** ThreadSleep {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.get("https://www.gmail.com");

driver.findElement(By.*id*("Email")).sendKeys("job.rahulsingh");

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

Thread.*sleep*(5000);

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

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

}

}

Scenario :Goto google search for selenium and click on first available link and capture the title and display it.

download

Search page

Google page

Empty Browser

Default Synchronizaton Thread.sleep Thread.sleep

Example:

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

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

driver.findElement(By.*id*("lst-ib")).sendKeys("Selenium", Keys.*ENTER*);

Thread.*sleep*(5000);

driver.findElement(By.*partialLinkText*("Selenium - Web Browser")).click();

Thread.*sleep*(5000);

String str=driver.getTitle();

System.*out*.println(str);

}

}

Disadvantage of Thread.sleep()

When there is a navigation there will be a delay in loading of page so for every navigation on Thread.Sleep() is required Thread.Sleep() will pause the execution for specified num. of mille seconds with in that time the page may get loaded or may not since we can’t predict the loading of page Thread.sleep() can’t be used.

Dynamic Waits:

To overcome above disadvantage below ways are used.

* Implicit wait.
* Explicit wait or webdriver wait.

Dynamic Wait:

Implicit wait will wait when element identification is there in script (driver.findElement()) implicit wait exist only once in the script preferably with in first 4 line of the script.

driver.manage().timeout().implecitwait(30,TimeUnit.SECONDS)

Implicit wait will accept 2 arguments

Max number of time.

Time Units(SECOND,MIN,HOURS,DAYS,WEEKS,NANO,Mille,MICRO)

When element identification is there on the page and maximum timeout is 30 seconds .If element is loaded within 10 second implicit wait will come out of the sleep after 10 seconds.Implicit wait() will wait till max timeout or element is not identified on the page with in max timeout , It will throw an exception call NOSUCH ELEMENT EXCEPTION

Example:

**public** **class** ThreadSleep {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

driver.findElement(By.*id*("lst-ib")).sendKeys("Selenium", Keys.*ENTER*);

//Thread.sleep(5000);

driver.findElement(By.*partialLinkText*("Selenium - Web Browser")).click();

//Thread.sleep(5000);

String str=driver.getTitle();

System.*out*.println(str);

driver.findElement(By.*partialLinkText*("many browsers")).click();

driver.findElement(By.*partialLinkText*("additional configuration")).click();

driver.findElement(By.*partialLinkText*("Support")).click();

}

}

Disadvantage of Implicit wait()

It work only when there is a element identification on the page using findElement or findElements()

Explicit Wait():

WebDriverWait wait=new WebDriverWait(driver,20);

Wait.until(ExpectedCondition .titleIS(“Downloads”)

Apart from element identification for any synchronization condition always use webdriver or explicit wait with a condition (title became download) .

WebDriver wait constructor will accept 2 argument.

* WebDriver
* TimeOut in seconds

Expected condition is a class and titleIS is a static method

driver.get(“URL”) //Default Synchronization

Thread.sleep()//Static wait

Implicit wait()//implicit wati will work when there is element identification i.e driver.findElement() on the same page.

Explicit wait()// WebDriverwait() is use for other condition and we need to use this if there is no element identification on that page.

Identify multiple element on a webpage

Scenario goto facebook.com and identify all available link and perform an action on first link.

Code

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

List<WebElement> link=driver.findElements(By.*xpath*("//a"));

**for**(**int** i=0 ;i<link.size();i++)

{

WebElement web=link.get(i);

System.*out*.println(web.getText());

}

}

}

Note:findElements() will identify multiple elements on a webpage. It identify multiple elements prefer xpath or css selector.

findElements is use to identify a group of semiler elements

findElements will return a List of webElement from a list to fetch single element use get(int index) with index.

Scenario : Write a script to check all the checkbox available in webpage.

Code.

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///C:/Users/rahul/Desktop/new%201.html");

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

List<WebElement> link=driver.findElements(By.*xpath*("//input[@type='radio']"));

**for**(**int** i=0 ;i<link.size();i++)

{

Thread.*sleep*(5000);

link.get(i).click();

}

}

}

Q:When we will get SlaveElement Reference Exception.

An:When page gets refresh in the middle of execution and it will try to perform action on a Element which is identified before refresh then the above exception will come.Once the page tot reloaded webdriver will lose the element reference.

Q: How to solve staleElementReference exception

Ans:We need to provide reference of webElement again.

Scenario: Goto google search for selenium and display all the link text available below result.

Code:

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

driver.findElement(By.*id*("lst-ib")).sendKeys("Selenium",Keys.*ENTER*);

List<WebElement> list=driver.findElements(By.*xpath*("//h3/a"));

Iterator<WebElement>it=list.iterator();

**while**(it.hasNext())

{

WebElement web= it.next();

System.*out*.println(web.getText());

}

}

}

findElement Vs findElements

findElement()

* It use to Identify single element
* If multiple matching’s are there it identify first one matching element.
* If element is not identify then it will throw no such matching element Exception.

findElements()

* Use to identify group of similar elements.
* If multiple matching are there for a locator it identify all of them and return List of elements.
* If element is not identified it never throw an exception but it will return size as o(zero).

How to Handle Dropdown list :

Below is the html structure for Select dropdown list.

<Select>

0<option value="01">Select One from below</option>

1<option value="02">Idly</option>

2<option value="03">Dosa</option>

3<option value="04">Rawa Idly</option>

4<option value="05">South-Meals</option>

5<option value="06">North-Meals</option>

</select>

Selection of option from a dropdown using index.

Code.

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///C:/Users/rahul/Desktop/new%201.html");

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

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

sel.selectByIndex(0);

Thread.*sleep*(5000);

sel.selectByIndex(1);

Thread.*sleep*(5000);

sel.selectByIndex(5);

}

Note:selectByIndex(int index) accept index as an argument and it will select an

option in dropdown list. And if a specified index is not available it will throw an exception says NOSuchELEMENT exception.

Select an option using Value:

Code:

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///C:/Users/rahul/Desktop/new%201.html");

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

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

sel.selectByValue("01");

Thread.*sleep*(5000);

sel.selectByValue("03");

Thread.*sleep*(5000);

sel.selectByValue("02");

}

}

Note:selectByValue(String str) accept String as an argument and we need to provide exact value.

Selecting an option using Visible Text:

Sle.selectByVisibleText(“North Meals”) //remaining code is same.

Note:

Fixed drop down list means a drop down has fixed number of options Here we can use any method of Select class to select an option.

Dynamic dropdown list means a drop down option will change based on user input here we need to use selectByVisibleText()

Scenario:Goto facebook and select Birth day.

Code:

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

sel.selectByVisibleText("12");

sel.selectByIndex(3);

WebElement web1=driver.findElement(By.*xpath*("(//select)[2]"));

Select se1=**new** Select(web1);

se1.selectByIndex(4);

WebElement web3=driver.findElement(By.*xpath*("(//select)[3]"));

Select sel3=**new** Select(web3);

sel3.selectByIndex(4);

}

}

Note: If a dropdown designed using (select html tag) is known as Dropdown list.

If a dropdown is not designed using select html tag it is known as Dropdown menu.

IF it is a dropdown list always use Select class to handle it , if it is a dropdown menu use driver.findElement()

Note: If we try to handle a dropdown menu using Select class. It will throw an exception called UnExpected TagName.

Q: How to verify a dropdown is single select or multiSelect.

Ans:

**public** **class** Test4 {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

**boolean** status=sel.isMultiple();

**if**(status==**true**) System.*out*.println("Multi Select");

**else** System.*out*.println("Single Select");

}

}

How to verify number of option present in dropdown.

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

**int** count=0;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

List<WebElement> options=sel.getOptions();

**for**(**int** i=0;i<options.size();i++)

{

WebElement we=options.get(i);

System.*out*.println(we.getText());

count++;// we can also use opton.size()

}

System.*out*.println("Number of options"+count);

}

}

Q: How to verify whether a particular option is available or not.

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[1]"));

Select sel=**new** Select(web);

List<WebElement> options=sel.getOptions();

**for**(**int** i=0;i<options.size();i++)

{

WebElement ele=options.get(i);

String str=ele.getText();

**if**(str=="12")

{

flag=**true**;

}

}

**if**(flag==**true**)

{

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

}

**else** System.*out*.println("Not found");

}

}

How to handle multiSelect dropdown list:

code

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///C:/Users/rahul/Desktop/new%201.html");

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[2]"));

Select sel=**new** Select(web);

sel.selectByIndex(1);

Thread.*sleep*(5000);

sel.selectByIndex(2);

Thread.*sleep*(5000);

sel.selectByIndex(2);

Thread.*sleep*(5000);

sel.deselectByIndex(1);

}

}

Deselecting an option from dropdown.(De selection work only with multi select)

Note : We can't perform de selection on single select dropdown list since by default once of the option need to be selected all the time .

If we try to deselect on a single select dropdown on a single select dropdown it will throw unsupported option exception.

How to fetch only Selected Item from a dropdown:

Code:

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///C:/Users/rahul/Desktop/new%201.html");

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

//driver.findElement(By.xpath("(//select)[3]")).click();

WebElement web=driver.findElement(By.*xpath*("(//select)[2]"));

Select sel=**new** Select(web);

sel.selectByIndex(1);

Thread.*sleep*(5000);

sel.selectByIndex(2);

Thread.*sleep*(5000);

sel.selectByIndex(3);

List<WebElement> list= sel.getAllSelectedOptions();

**for**(**int** i=0;i<list.size();i++)

{

String st= list.get(i).getText();

System.*out*.println(st);

}

}

}

How to Handle Dropdown menu using click Action:

1-Identify a menu and click on it.

2-Choose the option from a dropdown menu by clicking on it.

Goto actitime.com ,Enter username and password click on login button select Time track for the first person:

Below code will will throw an exception called WebDriverm Exception with since overlapping of element(div) exist on this page actual element is not identified hence click is not received by actual element

Code:

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://demo.actitime.com/login.do");

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

driver.findElement(By.*id*("username")).sendKeys("admin");

driver.findElement(By.*xpath*("//input[@type='password']")).sendKeys("manager",Keys.*ENTER*);

Thread.*sleep*(10000);

driver.findElement(By.*xpath*("//input[@id='ext-comp-1001']")).click();

Thread.*sleep*(5000);

driver.findElement(By.*xpath*("(//span[@class='userName'])[1]")).click();

}

}

Note: Above problem can be solved using Auto suggestion.

Steps to handle Auto Suggestion

* Identify the text box.
* If Default text is there clear it.
* Enter a keyword
* Choose among the option and click on it.

Code :

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://demo.actitime.com/login.do");

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

driver.findElement(By.*id*("username")).sendKeys("admin");

driver.findElement(By.*xpath*("//input[@type='password']")).sendKeys("manager",Keys.*ENTER*);

Thread.*sleep*(10000);

WebElement element= driver.findElement(By.*xpath*("//input[@id='ext-comp-1001']"));

element.clear();

element.sendKeys("bar");

driver.findElement(By.*xpath*("(//span[@class='userName'])[2]")).click();

}

}

Scenario:Goto google search for java and display all the auto suggestion and select javascript and click.

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

WebElement web=driver.findElement(By.*id*("lst-ib"));

web.sendKeys("java");

List<WebElement> list=driver.findElements(By.*xpath*("//div[@class='sbqs\_c']"));

**for**(**int** i=0;i<list.size();i++)

{

WebElement web1=list.get(i);

System.*out*.println(web1.getText());

**if**(web1.getText().equals("javascript"))

{

**break**;

}

}

driver.findElement(By.*xpath*("//b[contains(text(),'script')]")).click();

}

}

Mouse Related Action:

* Left click
* Drag and Drop
* Right click
* Select
* Hover
* Scroll
* Double Click
* Click and Hold.

Steps for mouse Related Action:

* Create Action class object
* Pass WebDriver object
* Perform action on WebElement by calling the appropriate method.

Note:For every Action class method use perform at the end that is must for Action else action will not work.

Scenario :Go to flipkart and perform move to Element Electornics.

Code:

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.flipkart.com");

WebElement ele=driver.findElement(By.*xpath*("//a[@title='Electronics']"));

Actions act= **new** Actions(driver);

act.moveToElement(ele).perform();

}

}

Scenario: Go to below URL <https://www.zoho.com/login.html> and goto finance and select expence.

Code/

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.zoho.com/mail/login.html");

WebElement ele=driver.findElement(By.*xpath*("//a[text()='Finance']"));

Actions act=**new** Actions(driver);

act.moveToElement(ele).perform();

driver.findElement(By.*xpath*("//a[contains(text(),'Expense')]")).click();

}

How to perform Context click(right click)

Code:

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

WebElement web=driver.findElement(By.*linkText*("Gmail"));

Actions act=**new** Actions(driver);

act.contextClick(web).perform();

act.sendKeys("P").perform();

//act.contextClick().sendKeys("P").build().perform() we can write above 2 line in one line

// here we need to build 2 actions that why we used build()

}

}

Actions class Structure:

Actions{

Actions(){}

Public void contextClick(){}

Public void ContextClick(WebElement element){}

Public void moveToElement(WebElement element){}

-

-

-

}

Note : To perform hover on webElement use moveToElement() of actions class.

moveToElement(WebElement element). moveToElement() will accept WebElement as argument to perform contextClick or right click use below methods.

contextClick()

ContextClick(WebElement element)

If multiple action is performed on same webElement first build the action (build()) then use perform().

Drag and Drop

How to perform Drag and Drop:

To perform Drag and Drop the element should be dragable when we use move over on webElement if cursor is changing to dragable symbol(

Then element in dragable.

Code.

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://www.dhtmlgoodies.com/submitted-scripts/i-google-like-drag-drop/index.html");

WebElement ele=driver.findElement(By.*xpath*("//h1[text()='Block 2']"));

WebElement dist=driver.findElement(By.*xpath*("//h1[text()='Block 4']"));

Actions act=**new** Actions(driver);

//act.dragAndDrop(ele, dist).perform();

act.moveToElement(ele).clickAndHold().moveToElement(dist).release().build().perform();

//above line will also do drag and drop

}

}

How to Perform Double click:

Scenario goto irctc.co.in and perform double click on flight tickets

Code:

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.irctc.co.in/eticketing/loginHome.jsf");

WebElement ele=driver.findElement(By.*linkText*("Flight Tickets"));

Actions act=**new** Actions(driver);

//act.doubleClick(ele).perform();

act.moveToElement(ele).doubleClick().build().perform();

}

}

|  |  |
| --- | --- |
| Action | Methds |
| Right Click | contextClick(),contextClick(WebElement web) |
| Double Click | doubleClick(),dobuleClick(WebElement web) |
| Drab and Drop | dragAndDrop(Src,des) |
| Scroll | Not applicable |
| Select | Not applicable |
| Hover | moveTOElement() |
| Release | Release |
|  |  |

Note:Using Actions class we can’t perform scroll we need to use javascript function called scroll

Steps to execute javascript command on browser

Activate firebug on the browser.

Click on console & type below command and press enter window.scroll(0,500)

Scroll method is depends on x,y axis to scroll up and down using x as constant and provide Y value .

To scroll left and right keep Y as constant (0) and provide X a value

Code

**public** **class** Test4 {

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

String handle="";

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.irctc.co.in/eticketing/loginHome.jsf");

String str="window.scroll(0,500)";

JavascriptExecutor jis=(JavascriptExecutor)driver;

jis.executeScript(str);

driver.close();

}

How to scroll Up : window.scroll(0,-500)

Popups :

Message

Alert popup

Confirmation popup

Message

OK

Prompt popup

Message

Handling Popps:

Java Script Popup:

The popups designed in javascript are known as javascript popups.

In java script popups 3 type are there

* Alert popup
* Confirmation popup
* Prompt popup

If popup is enable on webpage we can’t perform any action on the browser as well as web element.

Popup will be black and white mode.

Popup can’t be inspected using fire bug

Handling Alert popup:

Interface Alert{

Void accept(){}

Void dismiss(){}

Alert(){}

Void getText(){}

}

Code for alert popup

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://www.javascriptkit.com/javatutors/alert2.shtml");

driver.findElement(By.*name*("B2")).click();

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

Thread.*sleep*(6000);

al.accept();

code for Confirmation popup:

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://www.javascriptkit.com/javatutors/alert2.shtml");

driver.findElement(By.*name*("B3")).click();

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

Thread.*sleep*(6000);

al.dismiss();

Thread.*sleep*(5000);

al.accept();

Prompt Popup:

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://www.javascriptkit.com/javatutors/alert2.shtml");

driver.findElement(By.*name*("B4")).click();

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

Thread.*sleep*(6000);

al.sendKeys("Rahul");

al.accept();

al.accept();

String str= al.getText();

System.*out*.println(str);

}}

Alert.sendKey() method will accept an argument of type String and it is use to enter text on prompt.

Hidden Division popup:

* Poup is inspecting using firebug.
* This may or may not be color full mode.
* It is part of webpage.
* Open user action popup will get enable in the current page.

code:

**public** **class** Test4 {

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

**boolean** flag=**false**;

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("http://demo.actitime.com/login.do");

driver.findElement(By.*name*("username")).sendKeys("admin");

driver.findElement(By.*name*("pwd")).sendKeys("manager",Keys.*ENTER*);

driver.findElement(By.*xpath*("//div[text()='USERS']")).click();

driver.findElement(By.*cssSelector*(".buttonText")).click();

driver.findElement(By.*id*("userDataLightBox\_firstNameField")).sendKeys("Rahul");

AutoIT is pending.

**AutoIT script for File upload**

ControlFocus("File Upload" ,"","Edit1")

ControlSetText("File Upload","","Edit1","E:\Selenium WorkSpace-3OCT2016\data.txt")

ControlClick("File Upload","","Button1")

Compile the above Autoscript and we will get an exe

Use below code

**public** **class** AutoIT {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("file:///E:/Selenium%20WorkSpace-3OCT2016/fileuploader.html");

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

driver.findElement(By.*id*("1")).click();

Runtime.*getRuntime*().exec("E://Selenium WorkSpace-3OCT2016/Fileupload.exe");//path of autoit.exe

}

}

Window Authentication popup:

AutoIT Script:

WinWaitActive("Authentication Required")

send("validusername")

send("{TAB}")

send("validpassword")

send("{ENTER}")

code :

**public** **class** AuthenticationAutoIT {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

Thread.*sleep*(5000);

Runtime.*getRuntime*().exec("E:\\Selenium WorkSpace-3OCT2016\\Authentication.exe");

driver.get("https://ispace.ig.capgemini.com/");

}

}

Child Window Popup:

Child window popup is popup which opens as a new tab or new window upon clicking on web element.

**public** **class** Test6 {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

driver.get("https://www.irctc.co.in/eticketing/loginHome.jsf");

driver.findElement(By.*xpath*("//span[text() ='E-Wheelschair']")).click();

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

Iterator<String> it=handles.iterator();

String parent=it.next();

String child=it.next();

driver.switchTo().window(child);

driver.findElement(By.*cssSelector*(".flight")).click();

Thread.*sleep*(5000);

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

Iterator<String>it1=hand.iterator();

String hand1=it1.next();

String hand2=it1.next();

System.*out*.println(hand1);

System.*out*.println(hand2);

driver.switchTo().window(hand2);

driver.close();

driver.switchTo().window(hand1);

driver.close();

}

}

Frames:Frames are nothing but web pages to insert one webpage inside another web page we use below tags.

<iframe>

<frameset>

<frame>

<html>

<head></head>

<body>

<iframe>

<html>

<body>

<input type=”text’ -/>

</body>

</html>

</iframe>

</body>

</html>

Steps to handle a Frame:

* Identify a fame
* Switch the control to frame
* Perform an action on a element available in a farme

Note :If selenium is throwing no such element exception verify element is with in the farame or not by providing an xpath

Steps to identify a frame

Approach no 1:

* Write below xpath to check the frame on a webpage

//frame or //iframe or //frameset

Approach no 2:

* Write click on web element and if This frame option is visible then element in with in a fame

When we open a web page driver control is on top window for swithching the control from top html to frame we need to use below method

Driver.swithchTo().frame(int arg)

Driver.swithchTo().frame(String str)

Driver.swithchTo().frame(WebElement element)

Handling a frame:

If a frame is having id or name as an attribute use

Driver.switchTo.frame with String argument

Code:

**public** **class** Test5 {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.zoho.com/mail/login.html");

driver.switchTo().frame("zohoims");

driver.findElement(By.*id*("lid")).sendKeys("admin");

driver.findElement(By.*id*("pwd")).sendKeys("pass123");

driver.findElement(By.*id*("submit\_but")).click();

driver.switchTo().defaultContent();//control will switch back to top window from frame

driver.findElement(By.*linkText*("mail")).click();

}

}

How to swith control to Top window.

Driver.swithchTo.defaultContent()

If a control is available in a frame to perform an action on top window always switch the control to top window by using below statement

driver.swithchTo.defaultContent()

How to handle a frame which doesn’t have ID or name as attribute:

Steps to handle are below

1-Identify the frame by using xpath

2-pass the identified element for below statement

Driver.switchTo.frame(WebElement ele)

Code:

**public** **class** Test5 {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.zoho.com/mail/login.html");

//driver.switchTo().frame("zohaims");

WebElement ele =driver.findElement(By.*xpath*("(//iframe)[2]"));

driver.switchTo().frame(ele);

driver.findElement(By.*id*("lid")).sendKeys("admin");

driver.findElement(By.*id*("pwd")).sendKeys("pass123");

driver.findElement(By.*id*("submit\_but")).click();

driver.switchTo().defaultContent();//control will switch back to top window from frame

driver.findElement(By.*linkText*("mail")).click();

}

}

Between the frames swithching is not allowed

<html>

<body>

<iframe>

<body></body>

</iframe>

<iframe>

<body></body>

</iframe>

<iframe>

<body></body>

</iframe>

<iframe>

<body></body>

</iframe>

</body>

<html>

Scenario :If control is in frame 1 how to switch to frame 3

Note :Frame inside another frame is known as Inner frame how to switch control to inner frame

<html>

<body>

<iframe>

<body>

<iframe>

<body></body> Inner frame

</iframe>

</iframe>

</body>

</html>

If control is in Inner frame how to switch to outer frame:

Driver.swithcTo.defaultContent()

If control is in inner frame driver.swithTo.defaultContent() automatically control will back to top window.

If inner frames are there on a web page use driver.swichTo.frame(int arg) with integer arguments

Data Driven Testing:

Data-driven testing is creation of test scripts where test data and/or input values are read from data files instead of using the same hard-coded values each time the test runs. This way, testers can test how the application handles various inputs effectively. It can be any of the below data files.

* Datapools
* Excel files
* ADO objects
* CSV files
* ODBC sources

File f=new File(String path) // To locate the file

FileInputStream fis=new FileInputStream(f)// to Read the file

FileOutPutStream fos=new FileOutPutStream(f)// to write the file.

Using FileInputStream we can’t read bulk data from a excel so we need to download and use Apace POI

* After successful download extracts POI jars
* Integrate POI jars to current project

Steps Right Click on a project and select build path and chose configure build path

Click on Library click on Add External jar

Goto to POI folder and select available jars (lib) and click on open button

Click on Add External jars goto lib folder and select the jars which are starting with common and click on open button.

Click on Add external jars goto ooxml.lib and select all the jars and click on open .

Workbook->sheet->row->cell

File f=new File(“E:// table.xlsx”) //Locate the file

FileInputStream fis=FileInputStream(f)//to read file

Workbook wb=WorkbookFactory.create(fis);//to create WorkBook

Code to use excel sheet

**public** **class** POI {

**public** **static** **void** main(String[] args) **throws** EncryptedDocumentException, InvalidFormatException, IOException {

File f=**new** File("E://Selenium WorkSpace-3OCT2016/table.xlsx");//Locating the file

FileInputStream fis=**new** FileInputStream(f);//To Read a file

Workbook wb=WorkbookFactory.*create*(fis);//To create workbook

Sheet st=wb.getSheet("sheet1");//get the sheet

Row r=st.getRow(0);//get the row

Cell c= r.getCell(0);//get the cell

String s=c.toString();

System.*out*.println(s);

How to get Data from excel when excel is in below format

Table.xlsx

|  |  |  |  |
| --- | --- | --- | --- |
| Gmail |  |  |  |
| FaceBook |  |  |  |
| Google |  |  |  |
| Software testing |  |  |  |

Code:

**import** java.io.File;

**import** java.io.FileInputStream;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** IOExcel1 {

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

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\table.xlsx");

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

Workbook wb= WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet("Sheet1");

**int** lastRow=s.getLastRowNum();

**for**(**int** i=0;i<=lastRow;i++)

{

Row r=s.getRow(i);

Cell c=r.getCell(0);

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

}

}

}

Note:getLastRowNumber will return index of the row which is having data.

How to get data from excel sheet if we have below excel structure:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Google | Facebook | Gmail | Software | Testing |
|  |  |  |  |  |
|  |  |  |  |  |

Code:

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** IOExcel2 {

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

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\table.xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet("Sheet1");

Row r=s.getRow(0);

**int** lastCellNum=r.getLastCellNum();

**for**(**int** i=0;i<lastCellNum;i++)

{

String data=r.getCell(i).toString();

System.*out*.println(data);

}

}

}

lastCellNumber() return last cell number

lastRowNumber() return the index of last data filled row.

Print the data in same format as below excel

|  |  |  |
| --- | --- | --- |
| UserName | Password |  |
| User1 | Pass1 |  |
| User2 | Pass2 |  |
| User3 | Pass3 |  |
| User4 | Pass4 |  |
| User5 | Pass5 |  |

Code:

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** IOExcel3 {

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

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\table.xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet("Sheet1");

**int** lastRow=s.getLastRowNum();

**for**(**int** i=0;i<=lastRow;i++)

{

Row r=s.getRow(i);

**int** lastCell=r.getLastCellNum();

**for**(**int** j=0;j<lastCell;j++ )

{

Cell c=r.getCell(j);

System.*out*.print(c.toString()+"\t");

}

System.*out*.println();

}

}

}

Data Driven Framework:

Properties of a data driven Framework.

1. The class should be public
2. All the method in a class must be static and public.
3. In a class every method should be a generic type.
4. Data driven framework is used to achieved code optimization , Reusability , maintainability.

Write a generic code for a method which return a cell data from excel table.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** GenericIO1 {

**public** **static** String fromExcel(String fname, String Sname, **int** rIndex, **int** cIndex)

{

String data=**null**;

File f=**new** File("E:/Selenium WorkSpace-3OCT2016/"+fname+".xlsx");

**try** {

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet(Sname);

Row r=s.getRow(rIndex);

Cell c=r.getCell(cIndex);

data=c.toString();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** data;

}

}

Write Generic method for above code.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** GenericIO1 {

**public** **static** String fromExcel(String fname, String Sname, **int** rIndex, **int** cIndex)

{

String data=**null**;

File f=**new** File("E:/Selenium WorkSpace-3OCT2016/"+fname+".xlsx");

**try** {

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet(Sname);

Row r=s.getRow(rIndex);

Cell c=r.getCell(cIndex);

data=c.toString();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** data;

}

}

Calling the above method as below.

**public** **class** TestGenericCode {

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

String str= GenericIO1.*fromExcel*("table", "Sheet1", 1, 1);

System.*out*.println(str);

}

}

Reading a data from properties file

Convention in data driven framework

1-Test data is maintained in excel and configuration data is maintained in a property file , Json , Xml.

2-The configuration data is nothing but browser name , URL, Super admin credential etc.

3-Properity file will have Key Value pari.

Steps to read data from Property file.

1-Locate a file

2-Open a file in readable mode.

3-Create an Object for Properties class and load the Configuration file.

4-Perform action on it(read/write)

Code : Reading data from properties file.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.util.Properties;

**public** **class** ReadingFromProperties {

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

File f=**new** File("E://Selenium WorkSpace-3OCT2016/Cdate.properties");

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

Properties p=**new** Properties();

p.load(fis);

String st=p.getProperty("browser");

System.*out*.println(st);

}

}

Note Cdate.properties is as below.( We can use a note pad and change the extension from .txt to .properties)

Browser=firefox

Note –getProperties() is a method which is used to get a value from properties file . it accept Key as an argument.

Generic code to access the data from properties file.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.util.Properties;

**public** **class** GenericProperties {

**public** **static** String getData(String fname, String Key) **throws** IOException{

String data =**null**;

File f=**new** File("E://Selenium WorkSpace-3OCT2016/"+fname+".properties");

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

Properties p=**new** Properties();

p.load(fis);

data=p.getProperty(Key);

**return** data;

}

}

Code to call above method.

**import** java.io.IOException;

**public** **class** TestGenericProperties {

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

String str= GenericProperties.*getData*("Cdate", "browser");

System.*out*.println(str);

}

}

How to write data back to excel.

Code :

**public** **class** WritingDataToExcel {

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

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\table.xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet("Sheet1");

Row r=s.getRow(0);

**try**{

Cell c=r.getCell(1);

c.setCellValue("Login is Successful");

}

**catch**(Exception e)

{

Cell c=r.createCell(4);

c.setCellValue("Login is Successful");

}

FileOutputStream fos=**new** FileOutputStream(f);

wb.write(fos);

fos.close();

wb.close();

fis.close();

}

}

Note : In above code c.setCellValue() method will throw an exception at run time as some time the cell where we want to enter data is not active hence we are keeping this code to inside the try and catch block.

**Note :First time we used FileInputStream to get the Cell from excel and second time we use FileOutPut Stream to write the data to the same excel cell from Object copy by using WorkBook class static write method.**

Generic Code for Data writing in excel.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** GenericCodeForDataWritingInExcel {

**public** **static** **void** toExcel(String fname, String Sname, **int** rIndex , **int** cIndex, String data)**throws** Exception

{

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\"+fname+".xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet(Sname);

Row r=s.getRow(rIndex);

**try**{

Cell c=r.getCell(cIndex);

c.setCellValue(data);

}

**catch**(NullPointerException e)

{

Cell c=r.createCell(cIndex);

c.setCellValue(data);

}

**catch** (Exception e) {

// **TODO**: handle exception

}

FileOutputStream fos=**new** FileOutputStream(f);

wb.write(fos);

}

}

How to call the above method.

**public** **class** TestGenericExcelWriting {

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

GenericCodeForDataWritingInExcel.*toExcel*("table", "Sheet1", 3, 6, "Rahul Singh");

}

}

Note here we have used r.createCell() with index as a parameter to create a new cell for getting the excel cell active.

Writing a data to properties file:

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.util.Properties;

**public** **class** WriteProperties {

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

File f=**new** File("E:\\Selenium WorkSpace-3OCT2016\\Cdate.properties");

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

FileOutputStream fos=**new** FileOutputStream(f, **true**);

Properties prop =**new** Properties();

prop.load(fis);

prop.setProperty("ULR", "rahul");

prop.store(fos, "Updated Ulr");

}

}

Write a generic method to write data to properties .

Scenario goto demo.actitime.com perform login and verify home page is available or not.

Verify the home page using title of the page.

**import** java.util.concurrent.TimeUnit;

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

**import** org.openqa.selenium.Keys;

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** TestLoginPage {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.get("http://demo.actitime.com/login.do;jsessionid=96136C0A9428FA3DD458F38223B918F8");

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

driver.findElement(By.*id*("username")).sendKeys("admin");

driver.findElement(By.*name*("pwd")).sendKeys("manager",Keys.*ENTER*);

String expectedTitle="actiTIME - Enter Time-Track";

String actual=driver.getTitle();

**if**(expectedTitle.equals(actual))

{

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

}

**else**

{

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

}

}

}

Verify a home page using element visibility

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

**import** org.openqa.selenium.Keys;

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** TestLoginPage {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.get("http://demo.actitime.com/login.do;jsessionid=96136C0A9428FA3DD458F38223B918F8");

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

driver.findElement(By.*id*("username")).sendKeys("admin");

driver.findElement(By.*name*("pwd")).sendKeys("manager",Keys.*ENTER*);

// String expectedTitle="actiTIME - Enter Time-Track";

// driver.findElements(By.xpath("//a"));

// String actual=driver.getTitle();

// System.out.println(actual);

// if(expectedTitle.equals(actual))

// {

// System.out.println("pass");

// }

// else

// {

// System.out.println("failed");

// }

WebElement web=driver.findElement(By.*id*("logoutLint"));

**boolean** actualResult=web.isDisplayed();

**boolean** expectedResult=**true**;

**if**(actualResult==expectedResult)

{

System.*out*.println("Pass is home page");

}

**else**

{

System.*out*.println("Failed it is not home page");

}

}

}

Note isDisplayed() is a method used to verify an element is display or not.If element is displayed it return ture otherwise it will return false.

Scenario: How to verify a checkbox is checked or uncheck.or

Go to login page of actitime and perform login make sure keep me logged in check box is checked

Code:

**import** java.util.concurrent.TimeUnit;

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

**import** org.openqa.selenium.Keys;

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** ChekboxStatusChecked {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

driver.get("http://demo.actitime.com/login.do;jsessionid=96136C0A9428FA3DD458F38223B918F8");

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

driver.findElement(By.*id*("username")).sendKeys("admin");

driver.findElement(By.*name*("pwd")).sendKeys("manager",Keys.*ENTER*);

WebElement checkbox=driver.findElement(By.*id*("keepLoggedInCheckBox"));

**boolean** status =checkbox.isSelected();

**if**(status==**true**)

{

System.*out*.println("Check box is checked");

}

**else**{

System.*out*.println("check box is unchecked");

checkbox.click();

}

}

}

Note : isSelected() is used to verify the element is checked or unchecked use for check box or radio button

If checkbox or radio button is checked it return true otherwise it will return false.

Assignment : Go to facebook.com and verify male radio button is checked or female radio button is checked.

How to verify a element is disable or enable on a web page.

Note : If an element is disabled we can identify an element but we can’t perform action on it . If we try to perform action on disable element it will throw **invalid elementStateException**

CODE:

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** ElementEnableORDisable {

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

System.*setProperty*("webdriver.firefox.marionette", "E:\\Selenium WorkSpace-3OCT2016\\geckodriver.exe");

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

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

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

driver.manage().timeouts().implicitlyWait(40, TimeUnit.*SECONDS*);

WebElement web=driver.findElement(By.*id*("gs\_htif0"));

**boolean** status=web.isEnabled();

**if**(status==**true**)

{

System.*out*.println("Status is enable");

}

**else**

{

System.*out*.println("status is disable");

}

}

}

Note: isEnabled() is used to verify whether an element is enable or disabled on webpage.

If element is enabled it return true otherwise it returns false.

TestNG(Test Next Generation)

Features of TestNG:

1. 1-TestNg Supports preparation of report(Execution status) ,Parameterization, Parallel execution , Test dependency execution , priority execution , Enabliling and disabling test, data providers etc.
2. In industry most used unit testing tool are jnunt , Nunit and TestNG.
3. Junit supports only java language.
4. Nunit support only .Net language.
5. TestNG supports java as well as .Net.
6. TestNG is nothing but Test Next Generation and it’s a unit testing tool

How to Download.

Goto Help->Eclipse Market Place->type TestNG in Find and search ->We will get TestNG for eclipse ->click on Install button ->click on confirm button ->check Accept ->click finish.

Installation of TestNG:

In eclipse goto help and choose eclipse market place search for testNG .Click on install button available below testNG for eclipse.Click on confirm button. Accept license agreement and click on finish button . During installation process click on Ok button in a popup->Restart eclipse.

Integrating TestNG to Project:

1. Right click on Project and go to build path and select configure build path.
2. Click on Libraries
3. Click on Add Libraries.
4. Select TestNG and click on finish button.
5. Click on OK button.

In a java class if @Test annotation is there then it is a TestNG class otherwise it is a java class

Property of @Test annotation

1. Annotation is always attached with a method.
2. The method must be public.
3. Method doesn’t have any return type apart from void.
4. No statement are allowed between annotation and a method.
5. The method may be static or non static.

Creation of TestNG Class:

TestNG class can be created in 2 ways.

1. Right click on the package goto TestNG and select create a TestNG class
2. Provide class name and click on finish button.
3. Automatically TestNG class generated with @Test annotation.

Public class Demo{

@Test

Public void a()

{

System.out.println(“Running a() test method”);

}

}

Approach No 2:

Right click on Package and create a java class and add @Test annotatin.

Execution of TestNG:

1. Right click on a TestNG class and Run as TestNG test.
2. After the successful execution TestNG will generate 3 different reports
   * + - Eclipse console report
       - TestNG Console Report
       - Html report(emalelable report) it will available is below project in folder called test-project.

Note:In a single testNG class we can have multiple annotations the execution of the method is depending on alphabetical order of a method name.

Public class Demo{

@Test

Public void b()

{

System.out.println(“Running b() test method”);

}

@Test

Public void a()

{

System.out.println(“Running b() test method”);

}

}

Output:

Running a() test method

Running b() test method

Note : TestNG will always execute a method which is attached with annotation

Example

Public class Demo{

@Test

Public void b()

{

System.out.println(“Running b() test method”);

}

//No @Test is available.

Public void a()

{

System.out.println(“Running b() test method”);

}

}

Output

Running b() test method

**Convention:**

If we launch url .If it is landing on login page every test should starts with login and end with logout

Launch ->URL->login page\_>test start->login

\_>test end->logout

How to map manual test case in Automation or test case convention in Automation.:

Every manual suite will have test automation suite. Every scenario will have a TestNG class and every test case will have @Test annotations.

Testcase 1

Regression

Smoke

ie. Scenario Test case 3

Test case2

Note :In manual testing TestNG class equivalent to scenario.

@BeforeMethod and @AfterMethod :

In TestNG execution will start from @Test annotation @Test will look for @BeforeMethod and it execute that. After the execution it execute it execute @Test then it execute After annotation

Below is the execution order.

1. @BeforeMethod
2. @Test
3. @AfterMethod

@Before method annotation and @After method will get executed for every @Test annotation

@Before and @After method will get called implicitly in TestNG.

Code:

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

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

**public** **class** Loing {

WebDriver driver=**null**;

@BeforeMethod

**public** **void** preCondition()

{

System.*setProperty*("webdriver.firefox.marionette", "path");

driver=**new** FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(30,TimeUnit.*SECONDS*);

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

}

@AfterMethod

**public** **void** postCondition()

{

driver.close();

}

@Test

**public** **void** loginSuccess()

{

System.*out*.println("Login is Successfull");

System.*out*.println("Logout from the Application");

}

@Test

**public** **void** loginFail()

{

System.*out*.println("Login Fail");

System.*out*.println("verify error message");

}

@Test

**public** **void** forgotPassword()

{

System.*out*.println("Forgot Password");

System.*out*.println("Verify error message");

}

}

@DataProvider

**package** com.test;

**import** org.testng.annotations.DataProvider;

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

**public** **class** LoginTest {

@DataProvider

**public** Object[][] getData()

{

Object[][] obj=**new** Object[3][2];

obj[0][0]="User1";

obj[0][1]="User1";

obj[1][0]="User1";

obj[1][1]="User1";

obj[2][0]="User1";

obj[2][1]="User1";

**return** obj;

}

@Test(dataProvider="getData")

**public** **void** loginTest(String un, String pwd)

{

System.*out*.println("UserName"+un);

System.*out*.println("Password"+pwd);

}

}

Assertion in TestNG:

Step 1->pass

Step 2->pass

Step 3->pass

Step 4->Fail

Step 5->pass

So in normal verification using conditional statement in previous step status is pass or fail our execution will move to next step

But in case of TestNG Assertion if previous step status is Fail it will not go ahead

In a test if we use verification execution will continue even though step status is fails.

In Assertion if steps is failing it will stop the execution.

Example for verification is below

Scenario:

Goto to login page and verify whether a place holder is available or not.

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

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

**public** **class** Verification {

@Test

**public** **void** tesPlaceHolder()

{

System.*setProperty*("", "");

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.gmail.com");

String ph=driver.findElement(By.*id*("Email")).getAttribute("placeholder");

**if**(ph.equals("Enter your email"))

{

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

}

**else**

{

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

}

driver.close();

}

}

Above code can be use with Assert as below

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.Assert;

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

**public** **class** Verification {

@Test

**public** **void** tesPlaceHolder()

{

System.*setProperty*("", "");

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.gmail.com");

String actual=driver.findElement(By.*id*("Email")).getAttribute("placeholder");

String expectedPh="Enter your Email";

Assert.*assertEquals*(actual, expectedPh);//If actual and Expected is not match then Assert will fail

driver.close(); // and Execution will not continue.

}

}

Using Asserction :

In TestNG to perform assertion in build class is available

Assert is the class in which static method are available to compare two values use assertEquals() method

* assertEqual(String, String)
* assertEqual(boolean, boolean)
* assertEqual(int, int)
* assertEqual(char, char)

assertEqual is used to compare 2 values (Actual , Expected)

if the Actual is not equal to expected then it will fail the test

HW:Scenario:

1-Goto gmail enter email id and verify the text available in textbox is proper or not.

**package** com.test;

**import** java.util.concurrent.TimeUnit;

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

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.Assert;

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

**public** **class** Verification {

@Test

**public** **void** tesPlaceHolder() **throws** InterruptedException

{

System.*setProperty*("", "");

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get("https://www.gmail.com");

driver.findElement(By.*id*("Email")).sendKeys("test");

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

Thread.*sleep*(5000);

WebElement emailError=driver.findElement(By.*id*("errormsg\_0\_Email"));

Assert.*assertEquals*(emailError.isDisplayed(), **true**);

String actualError=emailError.getText();

String Expected="Sorry , Google doesn't recoganize that email";

Assert.*assertEquals*(actualError, Expected);

driver.close();

}

}

2-Goto facebook perform login with invalid credentials and check the error message.

3-Goto gmail enter invalid email id and verify the error message is displayed or not.

Requirement

* Java should be installed in system.
* Download new Selenium jar
* Resent Firefox driver
* Resent chrome driver
* Install git (<https://git-scm.com/>)
* Download and install github desktop(<https://desktop.github.com>)
* Install Maven.

Step to install maven 1- Goto to help choose Eclipse Market place search for maven -> click on install button available below M2Econnection for maven dependency 0.0.4->click on confirmation button ->accept license agreement click on finish button->click on OK button during installation and restart Eclipse.

* Dowonload Jenkins(<URL:https://update.jenkins-ci.org/download/war>) download 1.600(imp)

Assert verification pending 2

Comparing 2 different values :

Scenario: Goto login page of irctc and verify maximum number of character it accept.

Code:

**package** com.com.test;

**import** java.util.concurrent.TimeUnit;

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

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.Assert;

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

**public** **class** Test3 {

@Test

**public** **void** Test() {

System.*setProperty*("webdriver.firefox.marionette","path");

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

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

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

WebElement captch=driver.findElement(By.*name*("J\_cpatcaa"));

String entervalue="tttlttttttttttt";

captch.sendKeys(entervalue);

String avlvalue=captch.getAttribute("value");

Assert.*assertNotEquals*(entervalue, avlvalue);

}

}

Page Object Model:

Sometimes selenium script fails due to some small change done by programmer in DOM structure due to that all the selenium script gets fail this is also known as cascading effect eg If a programmer change the DOM structure for element which we are finding in selenium script using driver.findElement() . Due to change selenium web driver will not able to locate the element and all the script will fail.Solution of this problem is Page Object model

Advantage of Page Object Model:

1. Script maintainability will get reduce
2. It will avoid cascading effect

Rules of Page Object Model:

1. Every web page will have corresponding java class If application is having 100 web pages 100 java class should exists.
2. The class must be public
3. Class will have a constructor as public which should accept WebDriver as an argument.
4. Every webElement will have method and it should be public and not static.Method will return web element as an object
5. Create webDriver reference variable with in class (globally )

**Developing the Page Object Model**

Convention in a Page object class:

1-Class name is defined with the webpage functionality followed by page.(LoginPage, HomePage, AddUserpage)

2-In a constructor always initialized local variable to global variable(driver)

By creating method name –It should starts with get fallowed by WebElemnt type

The purpose of the method is identify webElement.

3-Never perform action in a webElement in page object class.

Devlop a page object for create new user at demo.actitime.com

**Automation Frame Work environment creation**

1. Create a folder on the name of sapphire- project open eclipse provide folder path as work space and click on ok button
2. Create a project in the name of skynet
3. Create a folder as the name of External\_jars below the project
4. Copy selenium and Apache POI jars into external jar folder.
5. Integrate external jar to project.
6. Create the folder screen shot below project . Screen shot folder will have screen shot which are capture during execution upon a test fail.
7. Create a folder as driver.exe under project and keep all the driver for firefox ,chrome , IE
8. Create one folder as Test\_Data under the project and store the data which is capture in a excel.(Only excel file are stored in a folder)
9. Create a folded with name config\_add and add a properties file with below information

URL=http://www.google.com

Browser=firefox

1. Create a folder under project as Report\_archive to keep the previous all report.

Example code:

package pageobject;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public class LoginPage {

WebDriver driver;

public LoginPage(WebDriver driver)

{

this.driver=driver;

}

public WebElement getusernameTextBox()

{

WebElement ele=driver.findElement(By.name("username"));

return ele;

}

public WebElement getpasswordTextBox()

{

WebElement ele=driver.findElement(By.name("pwd"));

return ele;

}

public WebElement getloginButton()

{

WebElement ele=driver.findElement(By.id("loginButton"));

return ele;

}

}

Code for TestNG class:

package pageobject;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Test;

public class TesgPageObject {

@Test

public void testLogin() throws InterruptedException {

System.setProperty("webdriver.firefox.marionette","path");

WebDriver driver=new FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

driver.get("http://demo.actitime.com/login.do");

LoginPage login =new LoginPage(driver);

login.getusernameTextBox().sendKeys("admin");

login.getpasswordTextBox().sendKeys("manager");

login.getloginButton().click();

}

}

**Date 10-Dec**

Developing functionality to handle java script popup

Alert, Confirmation, Prompt

Create the class of the name of java script popup below utils package.

Code :

**package** com.saphire.Util;

**import** org.openqa.selenium.Alert;

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

**public** **class** JavaScriptPopup {

**public** **static** **void** acceptAlert(WebDriver driver)

{

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

alt.accept();

}

**public** **static** **void** dismissAlert(WebDriver driver)

{

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

alt.dismiss();

}

**public** **static** String getAlertText(WebDriver driver)

{

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

String altText=alt.getText();

**return** altText;

}

**public** **static** **void** enterAlertText(WebDriver driver, String text )

{

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

alt.sendKeys(text);

}

}

Developing Functionality to handle Data:

Create a class in the name of GetData below Utils

**package** com.saphire.Util;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.util.\*;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** GetData {

**public** **static** String fromProperties(String fName, String key)

{

String data =**null**;

**try**

{

File f=**new** File("Path"+".properties");

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

//Properties pro=new Properties();

Properties pro=**new** java.util.Properties();

pro.load(fis);

data =pro.getProperty(key);

}

**catch**(Exception e)

{

}

**return** data;

}

**public** **static** String fromExcel(String fName, String sName,**int** rIndex, **int** cIndex)

{

String data =**null**;

**try**

{

File f=**new** File("path"+".xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet st=wb.getSheet(sName);

Row r=st.getRow(rIndex);

Cell c=r.getCell(cIndex);

data=c.toString();

}

**catch**(Exception e)

{

}

**return** data;

}

}

Create a class on the name of SetData below Util package.

**package** com.saphire.Util;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.util.Properties;

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.Row;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.Workbook;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**public** **class** SetData {

**public** **static** **void** toExcel(String fName, String sName , **int** rIndex, **int** cIndex, String data)

{

**try**

{

File f=**new** File("path"+".xlsx");

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

Workbook wb=WorkbookFactory.*create*(fis);

Sheet s=wb.getSheet(sName);

Row r=s.getRow(rIndex);

Cell c=r.getCell(cIndex);

**try**

{

c=r.getCell(cIndex);

c.setCellValue(data);

}

**catch**(NullPointerException e)

{

c=r.createCell(cIndex);

c.setCellValue(data);

}

}

**catch**(Exception e)

{

}

}

**public** **static** **void** toProperties(String fName, String key, String value , String comment)

{

**try**

{

File f=**new** File("Path"+fName+".properties");

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

FileOutputStream fos=**new** FileOutputStream(f, **true**);

Properties prop=**new** Properties();

prop.load(fis);

prop.setProperty(key, value);

prop.store(fos, comment);

}

**catch**(Exception e)

{

}

}

}

Developing User Interfaces:

* Create package on the name of com.saphire.ui
* Create a class on the name of Loign page below UI package.
* UI packate will have java classes which are related to page object model.

**package** com.saphire.ui;

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

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

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

**public** **class** LoginPage {

WebDriver driver=**null**;

**public** LoginPage(WebDriver driver)

{

**this**.driver=driver;

}

**public** WebElement getHeaderElement()

{

WebElement element=driver.findElement(By.*id*("headerContainer"));

**return** element;

}

**public** WebElement getUsernameTextBox()

{

WebElement element=**null**;

**try**// Handling exception is page object model if attribute value gets change in different buid name

//value of attribute name gets change

{

element=driver.findElement(By.*name*("username"));

}

**catch**(Exception e)

{

element =driver.findElement(By.*name*("Un"));

}

**return** element;

}

**public** WebElement getPasswordTextbox()

{

**return** driver.findElement(By.*name*("pwd"));

}

**public** WebElement getKeepMeLoggedInCheckBox()

{

**return** driver.findElement(By.*id*("keepLoggedInCheckBox"));

}

**public** WebElement getLoginButton()

{

**return** driver.findElement(By.*id*("loginButton"));

}

}

Developing Home page or Base page:

This home page or base page will have the common elements which are available across multiple pages in a page object model.

c common

Common

Common

Common Element

Create a class on the name of HomePage below ui package

Code:

**package** com.saphire.ui;

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

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

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

**public** **class** HomePage {

WebDriver driver;

**public** HomePage(WebDriver driver)

{

**this**.driver=driver;

}

**public** WebElement getTimeTrackTab()

{

**return** driver.findElement(By.*xpath*("//div[contains(text(),'Time-Track')]"));

}

**public** WebElement geTaskTab()

{

**return** driver.findElement(By.*xpath*("//div[text()='Task']"));

}

**public** WebElement getUserTab()

{

**return** driver.findElement(By.*xpath*("//div[text()='User']"));

}

**public** WebElement getSettingTab()

{

**return** driver.findElement(By.*xpath*("//div[text()='Setting']"));

}

**public** WebElement getLogOut()

{

**return** driver.findElement(By.*id*("logOut"));

}

}

Developing Driver factory:

Create a package on the name of com.Saphire.config. This package will have functionality related to browser instance like type of a browser instance, App URL, broser capabilities

Create a class called driver factory below com.saphire.config package

**package** com.saphair.config;

**import** java.util.concurrent.TimeUnit;

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

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

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** com.saphire.Util.GetData;

**public** **class** DriverFactory {

**public** **static** WebDriver getDriverInstance()

{

WebDriver driver=**null**;

String browserName=GetData.*fromProperties*("Cdate", "browser");

String appURL=GetData.*fromProperties*("Cdate","url");

**if**(browserName.equalsIgnoreCase("firefox")){

System.*setProperty*("webdriver", "E:/Java JLC Lab/Practice\_Last\_Final/Saphair/Driver.exe/geckodriver.exe");

driver=**new** FirefoxDriver();

}

**else** **if**(browserName.equalsIgnoreCase("chrome"))

{

System.*setProperty*("webDriver.chrome.driver", "E:/Java JLC Lab/Practice\_Last\_Final/Saphair/Driver.exe/chromedriver.exe");

driver=**new** ChromeDriver();

}

**else**

{

System.*out*.println("Please provide valid browser name");

}

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

driver.manage().timeouts().implicitlyWait(30, TimeUnit.*SECONDS*);

driver.get(appURL);

**return** driver;

}

}

Developing Test case in Auptmation:

Scenario:

SCID:SC001

Description:Test login feature

Test Case ID:TC001

Test Steps:

1. Goto login page
2. Perform login with value crendintials
3. Verfiy home page and logout.

Steps to develop a test case:

* Create a Package on the name of com.saphire.test.functional/regression
* Create a TestNG class using below format for a class name

1. testNG class should start with “Test”
2. Fallowed by scenario name
3. Fallowed by scenario ID
4. Example TestLoginSC001

* Get driver instance from the driverFactory
* Initialise all the pageobject classes which are required for the scenario
* Use @Test to develop test case.
* Test method will have below formate

1. Method name starts with Test fallowed by Test case description fallowed by Testcase ID

* In a test we will get all the WebElements and will perform action as well as asserctions.

Convention for data driven Testing:

* Every Scenarion will have dedicated excel sheet
* The Sheet name should match with Scenario ID
* All the data which is used for scenario execution must be in a corresponding excel.

Code:

**package** com.saphire.test.com.saphire.test.system;

**import** junit.framework.~~Assert~~;

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

**import** org.testng.annotations.AfterMethod;

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

**import** com.saphair.config.DriverFactory;

**import** com.saphire.Util.GetData;

**import** com.saphire.ui.HomePage;

**import** com.saphire.ui.LoginPage;

**public** **class** TestLoginSc001 {

WebDriver driver=DriverFactory.*getDriverInstance*();

LoginPage login=**new** LoginPage(driver);

HomePage home=**new** HomePage(driver);

String un=GetData.*fromExcel*("tdate", "SC001", 1, 0);

String pw=GetData.*fromExcel*("tdate", "SC001", 1, 1);

@Test

**public** **void** testLogin() {

login.getUsernameTextBox().sendKeys(un);

login.getPasswordTextbox().sendKeys(pw);

login.getLoginButton().click();

**boolean** actual =home.getLogOut().isDisplayed();

**boolean** expected=**true**;

~~Assert~~.*assertEquals*(actual, expected);

home.getLogOut().click();

String actualText=login.getHeaderElement().getText();

String expectedText="Please identify yourself";

~~Assert~~.*assertEquals*(actualText, expectedText);

}

@AfterMethod

**public** **void** postCondition()

{

driver.close();

}

}

Excel sheet for above code

|  |  |  |  |
| --- | --- | --- | --- |
| UserName | Password |  |  |
| admin | manager |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| SheetName=SC001 |  |  |  |

Assignment:

ScenarioID SC002

Scenario Description TestLoginFailed

TestCaseID:TC005

Test Steps:

* Goto login page
* Enter invalid crendintail
* Verify the error message
* Verify again it is a login page.

Automation process:

1. Test engineer will receive Tesc cases
2. When test cases are assigned do fesiablity test in feasiablity test execute manually
3. Check test case is automatable or not.
4. If test cases are pass manually and it is automatable then develop automation scripts.
5. After development of test verifies code standerd are maintained or not.

* Class name.
* Method Name
* Variable Names
* Comments for group of steps.

1. Execute the test multiple time locally and on a remote machine
2. Create a brach in “Github” and push the newly implemented test cases.
3. Add reviewer to review newly developed scripts(PR)
4. Run the test cases over a period of time multiple times this stabilization test will perform using Jenkins continuous integration tool.
5. Once test cases are stabilizing push the newly developed to final repository.
6. When a new build is deployed run the final committed script on a new build
7. Wait until the reports get generated and analyse the report.
8. Identify the failures and check manually and decide weather it’s a script issue or application issue
9. If it’s a application issue raised a bug.
10. If it is a script issue identifies script developer and assigned script issue for fix.