Selenium Notes - Part 2

****

Capturing Screenshot

****

Question : **How to capture screenshots in Selenium ?**

Answer : **We capture screenshots in Selenium using *getScreenshotAs()* method of *TakesScreenshot* interface.**

Steps to take screenshot:

1. Create an object of specific browser related class (eg : FirefoxDriver) and then upcast it to WebDriver object (eg : driver)

2. Typecast the same upcasted driver object to TakesScreenshot interface type.

3. Using the typecasted object, we call getScreenshotAs(OutputType.FILE) which in turn returns the source file object.

4. Using the File IO operations (i.e FileUtils class), we store the screenshots to desired location in the project.

Selenium Code :

```
package pack1;
import java.io.File;
import java.io.IOException;
import java.util.Date;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
```

```
public class CaptureScreenshot_ActiTIMEPage extends BaseClass{
	public static void main(String[] args) throws IOException {
		//Creating an object of Date class
		Date d = new Date();
		//Printing the actual date
		String date1 = d.toString();
		System.out.println(date1);
		//replacing the colon present in the date timestamp format to "_" using replaceAll()
		//method of String class
		String date2 = date1.replaceAll(":", "_");
		System.out.println(date2);
		//Enter the url
		driver.get("https://localhost:8080/login.do");

		//Typecasting the driver object to TakesScreenshot interface type.
		TakesScreenshot ts = (TakesScreenshot) driver;

		//getting the source file using getScreenshotAs() method and storing in a file
		File srcFile = ts.getScreenshotAs(OutputType.FILE);

		/*Created a folder called "screenshot" in the project directory
		Created another file by concatenating the date value  which has "_" in it
		(Underscore is the accepted character while creating a file in the project )*/

		File destFile = new File(".\\screenshot\\"+date2+"__actiTIMELoginPage.png");

		/*copyFile() method is a static method present in FileUtils class of JAVA
		storing the screenshot in the destination location*/

		FileUtils.copyFile(srcFile, destFile);

		//closing the browser
		driver.close();
	}
}
```

Question : Why capturing screenshot of the web pages is important in the project ?

Answer :

- **We capture screenshots in order to debug the failed test scripts.**
- **It actually helps the automation test engineer to find the exact root cause of the issue in the application at the earliest.**

Following are the possible scenarios after the script is failed:

- **Whenever an automation script is failed, we first manually execute the steps to check whether there is any issue in the application or the issue is with the script.**
- **If the script fails due to an issue in the script itself, we fix the script and re-run it till it is passed.**
- **If there is an issue in the application due to which the script is failed, then we log defect against the same issue. In this way, automation team gets credibility in the project.**

****

Handling Browser navigation

****

Question : How to navigate within the browser ?
Answer : Using navigate() methods.

```
public class BrowserNavigationExample {
        public static void main(String[] args) throws InterruptedException {
                System.setProperty("webdriver.gecko.driver", ".\\driver\\geckodriver.exe");
                WebDriver driver = new FirefoxDriver();
                //Enter the url
                driver.get("http://localhost:8080/login.do");
                driver.navigate().to("http://www.gmail.com");
                Thread.sleep(3000);
                driver.navigate().back();
                Thread.sleep(3000);
                driver.navigate().forward();
                Thread.sleep(3000);
```

```
                    driver.navigate().refresh();
                    driver.close();
          }
}
```

****
****

Handling Mouse and Keyboard Operations

****
****

Question : *How to handle mouse movement and keyboard Operations ?*

Answer :

- *We handle mouse movement in Selenium using mouseMove() method of Robot Class.*
- *Similarly, to handle keyboard operations, we use KeyPress() and KeyRelease() methods of Robot Class*

Selenium Code to demonstrate an example of Mouse movement and Keyboard operation :

```
package test;

import java.awt.AWTException;

import java.awt.Robot;

import java.awt.event.KeyEvent;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class Keyboard_Mouse_Operations {

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

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

                    //1. Launch the browser

                    WebDriver driver = new FirefoxDriver();

                    //2. enter the url -
```

```
                driver.navigate().to("http://localhost:8080/login.do");

                Thread.sleep(5000);

                //Creating an object of Robot Class

                Robot r = new Robot();

                //move the mouse by x and y coordinate

                r.mouseMove(300, 500);

                //press ALT key from keyboard

                r.keyPress(KeyEvent.VK_ALT);

                //press F key from keyboard

                r.keyPress(KeyEvent.VK_F);

                //Release F key from keyboard

                r.keyRelease(KeyEvent.VK_F);

                //Release Alt key from keyboard

                r.keyRelease(KeyEvent.VK_ALT);

                Thread.sleep(3000);

                //Press W key from keyboard to open a new private window

                r.keyPress(KeyEvent.VK_W);

                //Release W key from keyboard

                r.keyRelease(KeyEvent.VK_W);

                Thread.sleep(3000);

                // It will close only the current browser window

                //driver.close();

                // It will close all the browser windows opened by Selenium

                driver.quit();

 }}
```

****

Identification of WebElements using Locators

****

What is an WebElement ?

1. **Any element present on a web page is called as web element.**

2. **Developers use HTLM code to develop web pages.**

3. **For testing purpose, we can also create web page using HTML code.**

4. **In order to create a web page, we need to write HTML code in any text pad (eg : notepad and save the file with .html extension)**

Create a sample web page using HTML as mentioned below.

```
    <html>
        <body>
                UN : <input type="text" id = "username" value = "admin">
                PWD: <input type="text" id= "pass" value = "manager">
                <a href="http://localhost:8080/login.do"> Click ActiTIME Link</a>
     </body>
  </html>
```

In the above HTML tree, every element should have one of the 3 options.

1. **Tagname (this is mandatory for all the elements)**
2. **Attributes (Optional)**
3. **Text (Optional)**

Example of Tagname in the above HTML Tree structure:

- **html,**
- **body,**
- **input,**
- **a**

Example of Attributes in the above HTML Tree structure:

- **type = "text"**
- **id = “username”**
- **value = "admin"**

Format : attributeName = “attributeValue”

Example of Text in the above HTML Tree structure:

- Click ActiTIME link

What are Locators ?

- Locators are used to identify the web elements on the web page.
- We have 8 types of locators in Selenium using which findElement() methods identifies elements on the web page:
    1. id
    2. name
    3. tagName
    4. className
    5. linkText
    6. partialLinkText
    7. xpath
    8. cssSelector
- findElement() method returns the address of the web elements on the web page and the return type is WebElement.
- If the specified locators returns multiple elements, then findElement() method returns the address of the first matching element.
- If the specified locators returns No element, then findElement() method throws an exception called "NoSuchElementException".

Note :

In below Selenium code snippet,

```
        WebDriver driver = new FirefoxDriver();

1.      driver.findElement(By.id(""));
2.      driver.findElement(By.name(""));
3.      driver.findElement(By.tagName(""));
4.      driver.findElement(By.class(""))
5.      driver.findElement(By.linkText(""))
6.      driver.findElement(By.partialLinkText(""))
7.      driver.findElement(By.xpath(""))
```

8. driver.findElement(By.cssSelector(""))

1. (By is an abstract class and all the locators specified/highlighted above are static methods of By class and hence, we call directly by using <classname.staticConcrete> methods

Below is the code to demonstrate the usage of locators in selenium while identifying the web elements on the web page:

```
package pack1;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;



public class LocatorsExample{

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

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

          WebDriver driver = new FirefoxDriver();

         // Enter the URL of your own created sample web page

         driver.get("file:///C:/Users/admin/Desktop/UN.html");

         // Used "id" locator to find USERNAME text box

         WebElement unTB = driver.findElement(By.id("user"));

         //Clear the existing value present in the text box

         unTB.clear();

         // Enter value into the USERNAME text box

         unTB.sendKeys("ajit.biswas@gmail.com");

         // Used "name" locator to find Password text box

         WebElement passTB = driver.findElement(By.name("n1"));

         //Clear the existing value present in the text box

         passTB.clear();
```

```
        //Halt the program execution for 2 seconds

        Thread.sleep(2000);

        // Enter value into the Password text box

        passTB.sendKeys("Qspiders123");

        // Find the address of ActiTIME Link and click

        driver.findElement(By.linkText("Click ActiTIME link")).click();

        Thread.sleep(2000);

}}
```

Important notes on LinkText and PartialLinkText locator

- Out of all the locators, linkText and PartialLinkText are used to identify only the links present on the webpage.(elements whose tagname is "a" -- a stands for anchor)
- LinkText locator should be used when the text of the link is constant
- PartialLinkText locator should be used when certain part of the link text is getting changed everytime the page is loaded. i.e for partially dynamically changing text, we use partialLinkText locator
- To handle those elements whose text changes completely, we can't use partialLInkText locator. It should be handled by another locator called "xpath"
- If we use try to use these 2 locators on other type of elements (except Links), then we get "NoSuchElementException"

Steps to install firebug and firepath add-ons in Firefox browser :

1. We need to install firebug and firepath addons in firefox browser to write cssSelector expression and then evaluate whether the expression is correct or not.
2. To install firebug addon in firefox browser :

   TOOLS -- > ADD-ONS → EXTENSIONS → search firebug -- > and click on Install -- Restart the browser.

3. To install firepath addon in firefox browser :

   TOOLS -- > ADD-ONS → EXTENSIONS → search firepath -- > and click on Install -- Restart the browser.

Steps to write and evaluate cssSelector expression in firefox browser :

1. Navigate to the web page -- > right click anywhere on the web page → select inspect element with firebug or Press F12 from keyboard.
2. Go to firepath tab and select CSS option.

3. **Type the cssSelector expression and hit Enter key from the keyboard, it will highlight the corresponding matching element on the web page.**

Steps to write and evaluate cssSelector expression in Chrome browser :

1. **In order to write cssSelector expression in chrome browser, we don't need any add-ons as such.**
2. **Navigate to the web page -- > right click anywhere on the web page → Press F12 from keyboard or select inspect element, it will open the Developer tool section with Elements tab selected by default.**
3. **Press Ctrl+F and write the cssSelector expression, it will highlight the source code of the matching element.**
4. **Place the cursor on the highlighted source code, it will highlight the corresponding element present on the web page.**

cssSelector locator :

1. **It is one of the locator in Selenium using which we identify web elements on the web page.**
2. **It stands for Cascading Style Sheet.**
3. **The standard syntax for cssSelector expression is**

tagName[attributeName = 'attributeValue']

OR

here, tagName is not mandatory.

[attributeName = "attributeValue"]

Sample Element html code for Login button:

<input type="textbox" id= "ID123" class = "inputText" value="Login">

Following are the 4 different ways of writing cssSelector expression for above mentioned Login button :

CssSelector Expression using type as an attribute :: input[type='textbox']

Actual code to identify Login button using FindElement() method:

driver.findElement(By.cssSelector("input[type='textbox']"))

--

CssSelector Expression using id as an attribute : input[id='ID123']

Actual code to identify Login button using FindElement() method:

driver.findElement(By.cssSelector("input[id='ID123']"))

--

CssSelector Expression using class as an attribute : input[class='inputText']

Actual code to identify Login button using FindElement() method:

driver.findElement(By.cssSelector("input[class='inputText']"))

CssSelector Expression using value as an attribute : input[value='Login']

Actual code to identify Login button using FindElement() method:

driver.findElement(By.cssSelector("input[value='Login']"))

--

Important Note :

While deriving cssSelector expression, we can use either one attribute or multiple attributes till we found unique matching element on the web page.

eg : input[type='textbox'][id='ID123'][class='inputText'][value='Login']

4. CssSelector can also be written using ID and Class. Here, ID is represented by " # " and className is represented by dot operator(.)

Sample Element html code for Login button:

<input type="textbox" id= "ID123" class = "inputText" value="Login">

4.1 CssSelector expression for the above Login button can be written using ID as

input#ID123 (syntax = Tagname#id)

OR

#ID123 (syntax = #id) [note : tagname is not mandatory]

Actual code to identify Login button using FindElement() method is below :

driver.findElement(By.cssSelector("#ID123"))

4.2 CssSelector expression for the above Login button can be written using ID as shown below

input.inputText (syntax = tagname.classname)

OR

".inputText" (syntax = .classname) [note : tagname is not mandatory]

Actual code to identify Login button using FindElement() method:

driver.findElement(By.cssSelector(“.inputText”))

Limitation of cssSelector :

1. **It does not support text i.e we can identify element using text of the element.**
2. **It does not support backward traversing.**
3. **It doesn’t support index**

XPATH :

1. **xpath is one of the locator in selenium using which we identify objects or elements on the web page and perform specific actions to carry out automation testing.**
2. **xpath is the path of an element in the html tree.**
3. **xpath are of 2 types.**

   3.1) Absolute xpath

   3.2) Relative xpath

Absolute xpath :

1. **It refers to the path of the element right from the root node to the destination element.**
2. **While writing the absolute xpath, We use single forward slash (/) to traverse through each immediate child element in the html tree.**
3. **In the below sample html tree,**

```
document

  |______html

            |

             ---- body

                    |

                    ------> a
```

Absolute xpath can be written in the following ways.

html/body/a

or

./html/body/a

(Note :- here, dot (.) refers to the current document or the current web page, using dot here is optional)

4. Using absolute xpath in selenium code as shown below.

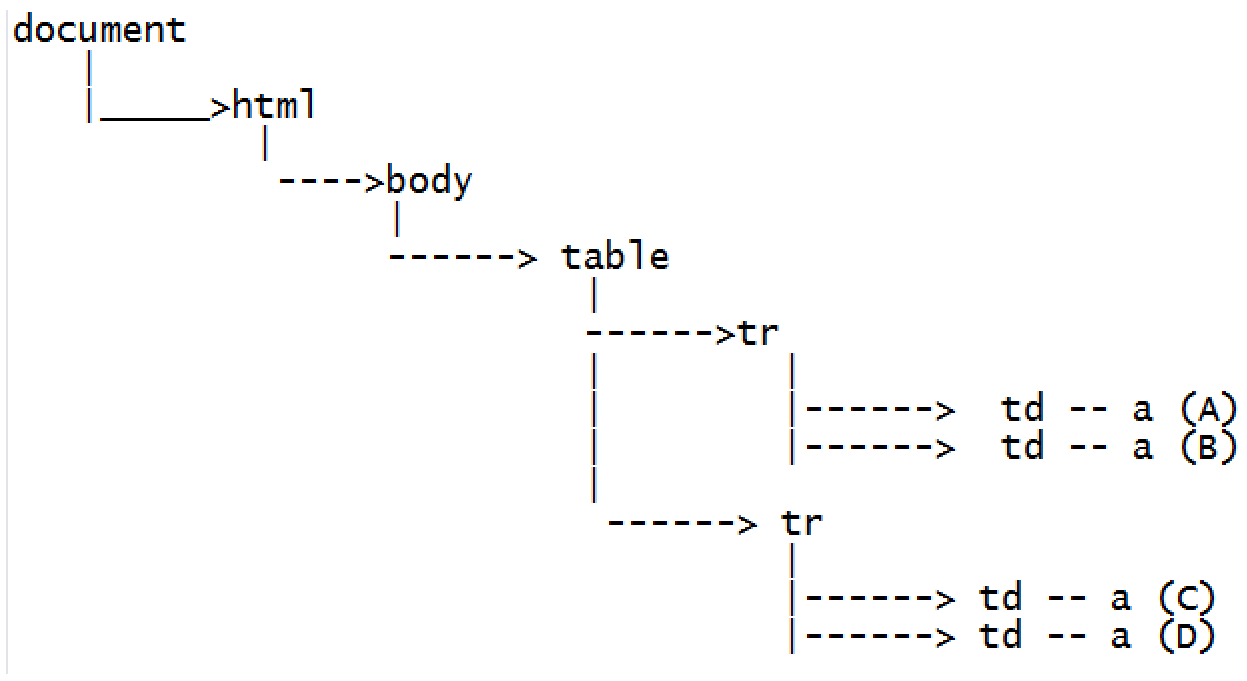
driver.findElement(By.xpath("html/body/a")).click();

5. In xpath, if there are multiple siblings with same tagname, then the index starts from 1.

6.In case of multiple siblings with same tagname, if we don't use index, then it considers ALL the siblings.

7. We can join multiple xpath using pipeline operator (|)

Considering the below sample html tree, write Absolute xpath and Relative xpath expressions.



Fill in the table with Absolute xpath expressions using the sample html tree given above.

Absolute xpath expressions	**Matching Element**
	A
	B
	C
	D
	AB
	CD

	AC
	BD
	AD
	BC
	ABC
	ABD
	ABCD

Relative xpath :

1. **In Absolute xpath, we write the path of the element right from the root node and hence, the expression for absolute xpath is lengthy.**
2. **In order to reduce the length of the expression, we go for Relative xpath.**
3. **In Relative xpath, we use double forward slash (//), which represents any descendant.**

Fill in the table with relative xpath expressions using the sample html tree given above.

Relative xpath expressions	Matching Element
	A
	B
	C
	D
	AB
	CD
	AC
	BD
	AD
	BC
	ABC
	ABD

	ABCD

<u>Interview questions :</u>

1. **what is the difference between '/' and '//' ?**

Answer : "/" refers to the immediate child element in the html tree.

"//" refers to any element in the html tree. It also represent any descendant.

2. What are the types of xpath?

Ans: Absolute and Relative xpath.

3. Derive an xpath which matches all the links present on a web page ?

Ans : //a

4. Derive an xpath which matches all the image present on a web page ?

Ans : //img

5. Derive an xpath which matches all the links and images present on a web page ?

Ans : //a | //img

6. Derive an xpath which matches all the 2nd links present on a web page ?

//a[2]

7. Derive an xpath which matches all the links present inside a table present on a web page ?

//table//a

8. Difference between "//a"and "//table//a " ?

Ans : //a → refers to all the links present on the webpage.

//table//a → refers to all the links present within all the tables present on the webpage.

<u>xpath by Attribute :</u>

1. **xpath expression can be written using attribute of the web element.**
2. **Based on the situation, we would use either single attribute or multiple attributes to find an element on a web page.**
3. **Using single attribute in xpath expression, if it returns one matching element, then we would use single attribute only.**

4. **In case, by using single attribute in xpath expression, if it returns multiple matching elements on the web page, then we would go for using multiple attributes in the xpath expression till we get one matching element.**

xpath expression using Attribute :

1. **using single attribute :**

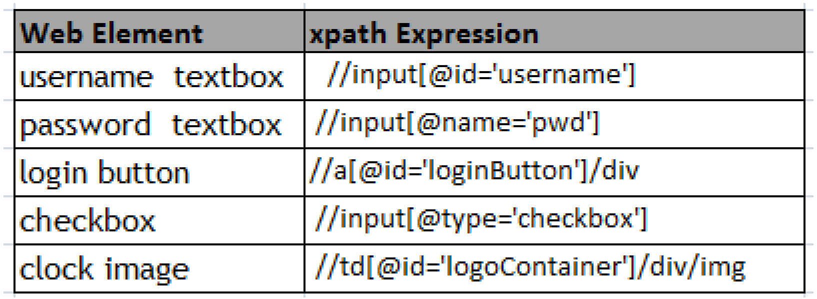
Syntax : **//tagname[@attributeName = ‘attributeValue’]**

//tagname[NOT(@attributeName = ‘attributeValue’)]

Sample application : actiTIME application

url : https://demo.actitime.com/login.do

Write xpath for below few elements on above actiTIME login page :



Web Element	xpath Expression
username textbox	//input[@id='username']
password textbox	//input[@name='pwd']
login button	//a[@id='loginButton']/div
checkbox	//input[@type='checkbox']
clock image	//td[@id='logoContainer']/div/img

Usage in selenium code :

driver.findElement(By.xpath(“paste any xpath here from above table”))

2. **Using multiple attribute :**

xpath Syntax :

- **//tagName[@AN1=’AV1’][@AN2=’AV2’]**
- **//tagName[@AN1=’AV1’] | //tagName[@AN2=’AV2’]**

Element : View licence link

html code after inspecting the element using F12 key:

<a id="licenseLink" target="" href="javascript:void(0)" onclick="openLicensePopup();">View License</a>

xpath expression using “href” and “onclick” attributes :

//a[@href='javascript:void(0)' and @onclick='openLicensePopup();']

Usage in selenium code :

driver.findElement(By.xpath("//a[@href='javascript:void(0)']

[@onclick='openLicensePopup();']"))

Assignment :

Write xpath expression for below 7 elements present on actiTIME login page

Elements :

1. **UserName**
2. **Password**
3. **Login Button**
4. **Check box**
5. **Actitime Image**
6. **View Licence link**
7. **actiTIME Inc link**

Use the below format (Sample example for actiTIME Inc Link):

html code for <actiTIME Inc.> :

<a href="http://www.actitime.com" target="_blank">actiTIME Inc.</a>

xpath syntax

//tagname[@AN1 = 'AV1']

1. using href attribute:

//a[@href = 'http://www.actitime.com']

2. using target attribute

//a[@target = '_blank']

Note: Use all the attributes of an element to write xpath expression

xpath expression using text() function :

1. In the html code of an element, if attribute is duplicate or attribute itself is not present, then use text() function to identify the element.
2. In order to use text() function, the element should have text in the element's html code.

Syntax :

//tagName[text()='text value of the element']

OR

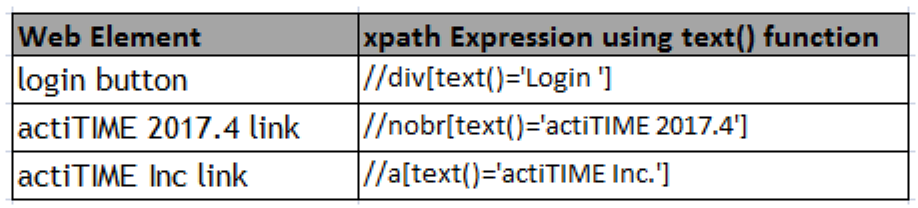
//tagName[.='text value of the element']

Note : Instead of text(), we can use dot (.) , the problem here with using dot (.) is sometimes,

it returns the hidden element also present on the webpage. which might confuse the

user. So the best practice is to use text() instead of using dot.

xpath expression using text() function for below elements present on actiTIME login page.



Web Element	xpath Expression using text() function
login button	//div[text()='Login ']
actiTIME 2017.4 link	//nobr[text()='actiTIME 2017.4']
actiTIME Inc link	//a[text()='actiTIME Inc.']

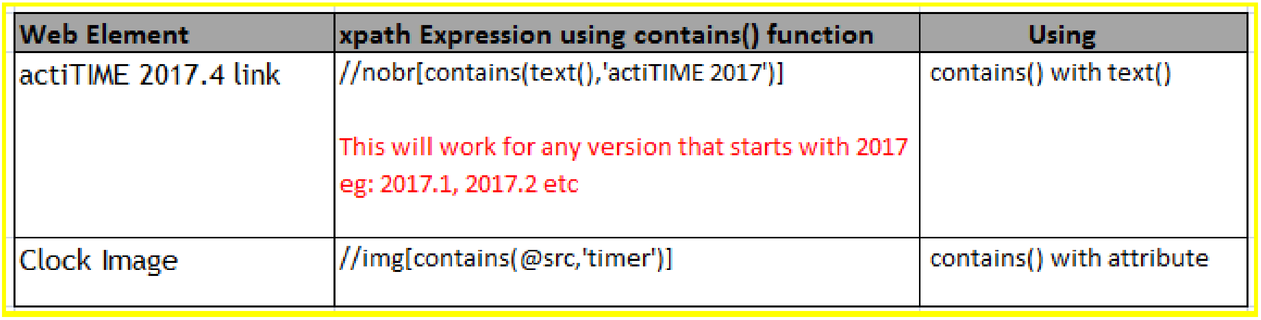
xpath expression using contains() function :

1. In the html code of an element, if the attribute value or the text is changing partially, then use contains() function to identify the element.
2. In order to use contains() function, the element should have either attribute value or text value.

Syntax :

- //tagName[contains(@attributeName,'attributeValue')]
- //tagName[contains(text(),'text value of the element')]

xpath expression using contains() function for below elements present on actiTIME login page.



Web Element	xpath Expression using contains() function	Using
actiTIME 2017.4 link	//nobr[contains(text(),'actiTIME 2017')] This will work for any version that starts with 2017 eg: 2017.1, 2017.2 etc	contains() with text()
Clock Image	//img[contains(@src,'timer')]	contains() with attribute

3. We use contains() function when the text value is very lengthy or the attribute value is very lengthy.

eg: xpath to identify error message present on actitime login page (Click on login button without entering username and password to get the error message)

//span[contains(text(),'invalid')]

<u>Program to illustrate xpath by attributes, text() function, contains() function and their usages with attributes and text values.</u>

```
public class XpathUsingAttribute_Actitime extends BaseClass{

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

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

                WebDriver driver = new FirefoxDriver();

                //Enter the url of actiTIME application

                driver.get("http://localhost:8080/login.do");

                //xpath using multiple attributes

                String xp = "//input[@class='textField'][ @id = 'username']";

                Thread.sleep(2000);

                //Enter admin into username text box

                driver.findElement(By.xpath(xp)).sendKeys("admin");

                Thread.sleep(2000);

                //find password element using xpath by attribute and enter manager in to password textbox.


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

                Thread.sleep(2000);

                //find an image on the web page whose attributes (src)contains a value called timer
                WebElement clock = driver.findElement(By.xpath("//img[contains(@src,'timer')]"));

                //store the width value of the clock image into a variable called widthValue

                String widthValue = clock.getAttribute("width");

                //Print the width of the clock image

                System.out.println("the width is :"+widthValue);

                //Print the height of the clock image

                System.out.println("the height of the clock element is : "+ clock.getAttribute("height"));

                //xpath using text() function

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

                Thread.sleep(2000);

                //xpath using contains() function and text() function

                driver.findElement(By.xpath("//a[@id='loginButton']//div[contains(text(),'Login')]")).click();

                Thread.sleep(2000);

                driver.close();

        }

}
```

xpath expression using starts-with() function :

1. We use starts-with() function to identify those elements whose text value starts with some specified value.

   *xpath using **contains()** function:*

   //[contains(text(),'actiTIME')] - this xpath will return 6 matching element on login page of actiTIME application.

   *xpath using **starts-with()** function,*

   //[starts-with(text(),'actiTIME')] - this xpath will return only 3 matching element on login page of actiTIME application as the text value of these 3 elements starts with “actiTIME” text

Handling completely dynamic links :

1. When the text value of the elements is completely changing, then we can't use functions like "contains()", "starts-with()" etc to handle those elements.
2. In such cases, we identify the dynamically changing element using the nearby unique element. We call this concept as independent dependent xpath.

Steps to derive xpath expression using Independent dependent concept :

1. Identify the independent element on the webpage and inspect the element to view the source code and then derive the xpath expression.
2. Place your cursor on the independent element source code and move the mouse pointer upward till it highlights both the independent and dependent elements which is the common parent element.

   Add /.. to the xpath of independent element already noted down in step 1 to

   get the xpath of common parent.

3. Use mouse pointer to navigate from common parent to the desired dependent

   element and derive the xpath of the dependent element.

4. Write the xpath from Independent element to Common parent and then write

   the xpath from Common parent to dependent element.

   Example 1:

   Write xpath to identify **version** of Java Language present in Selenium Download page.

   //td[.='Java']/../td[2]

   Example 2:

   Write xpath to identify **Release data** of Java Language present in Selenium Download

   page.

   //td[.='Java']/../td[3]

   Example 3:

   Write xpath to identify **Download link** of Java present in Selenium Download

   page.

   //td[.='Java']/../td[4]/a

   Note :

   - In case, if the column number of **Download link** changes, then the above xpath will fail to identify the link as we are hard coding the column position as 4 in the above case.

In order to handle this, we will write xpath in such a way that it works irrespective of the column position as shown below.

//td[.='Java']/..//a[.='Download']

Program 1 :

Write a script to click on the download link of Java in Selenium website

Scenario :

1. Login in to Selenium official website

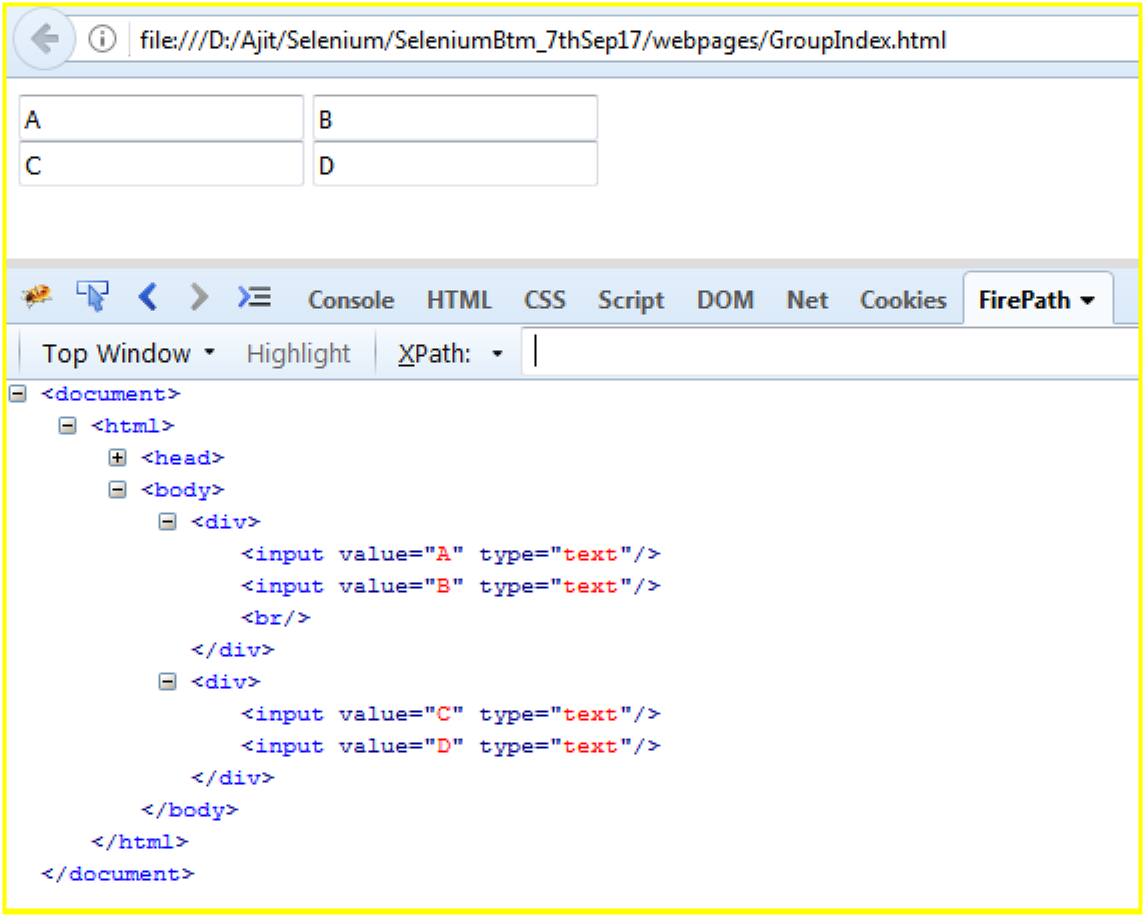
   Url : http://www.seleniumhq.org/download

2. Click on the Download link for Java language.

```
public class Independent_Dependent_Xpath_Seleniumsite_javaDownload{

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

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

         WebDriver driver = new FirefoxDriver();

        // enter the url

         driver.get("http://www.seleniumhq.org/download/");

         Thread.sleep(3000);

        // xpath using independent and dependent concept

    driver.findElement(By.xpath("//td[.='Java']/..//a[.='Download']")).click();

  }

}
```

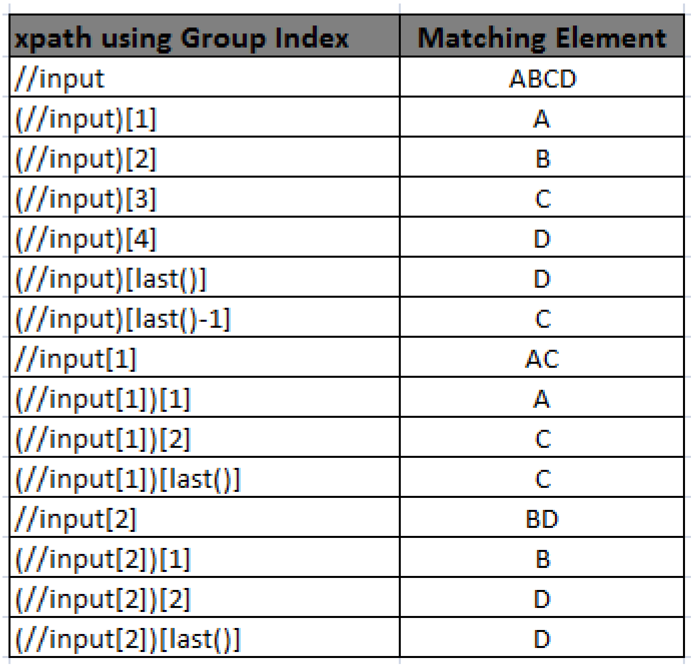
Group Index :

Sample Html tree :


file:///D:/Ajit/Selenium/SeleniumBtm_7thSep17/webpages/GroupIndex.html

```
<document>
  <html>
    <head>
    <body>
      <div>
          <input value="A" type="text"/>
          <input value="B" type="text"/>
          <br/>
      </div>
      <div>
          <input value="C" type="text"/>
          <input value="D" type="text"/>
      </div>
    </body>
  </html>
</document>
```

xpaths using Group Index to identify the elements in the above sample tree:



xpath using Group Index	Matching Element
//input	ABCD
(//input)[1]	A
(//input)[2]	B
(//input)[3]	C
(//input)[4]	D
(//input)[last()]	D
(//input)[last()-1]	C
//input[1]	AC
(//input[1])[1]	A
(//input[1])[2]	C
(//input[1])[last()]	C
//input[2]	BD
(//input[2])[1]	B
(//input[2])[2]	D
(//input[2])[last()]	D

1. In Group index, we write xpath expression within the braces and then we write the index outside the braces.
2. Internally, it executes the xpath expression first and stores the result in an xpath array whose index starts with 1
3. last() is a function that is used to retrieve the last element present in the xpath array.

Program 2 :

Click on the **Set by default** link of **testing** present in **type of work (Setting tab)** of actiTIME application

Scenario :

3. Login in to actime application

   Url : http://localhost:8080/login.do

   UN - admin, PWD - manager

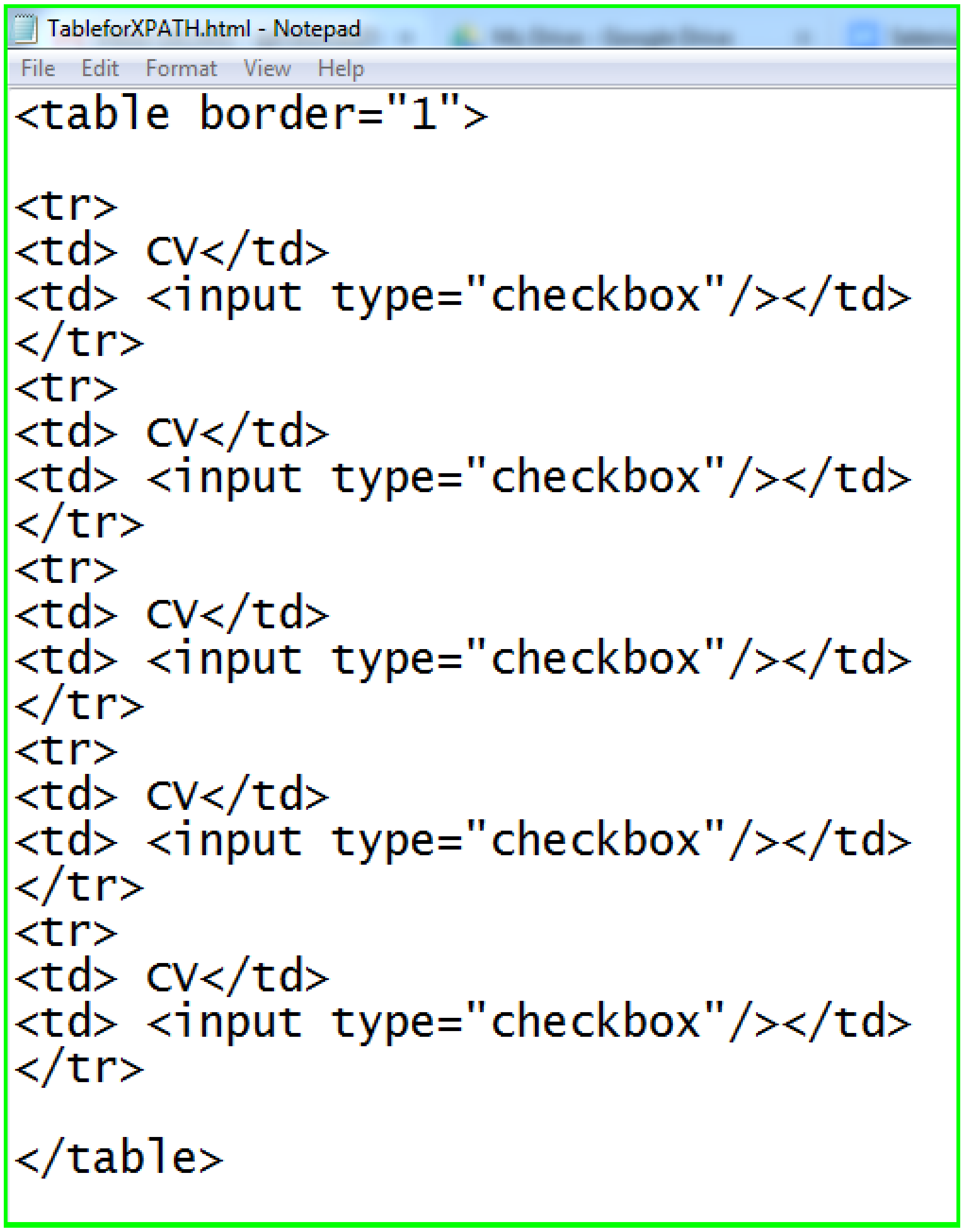
4. click on Settings
5. Click on the Types of Work link present in the window
6. click on the Set by Default link for a type of work called “testing”

Use the below hints :

1. Use groupIndex concept to find **Setting** Element and
2. Independent and dependent concept to find **Set by Default** link

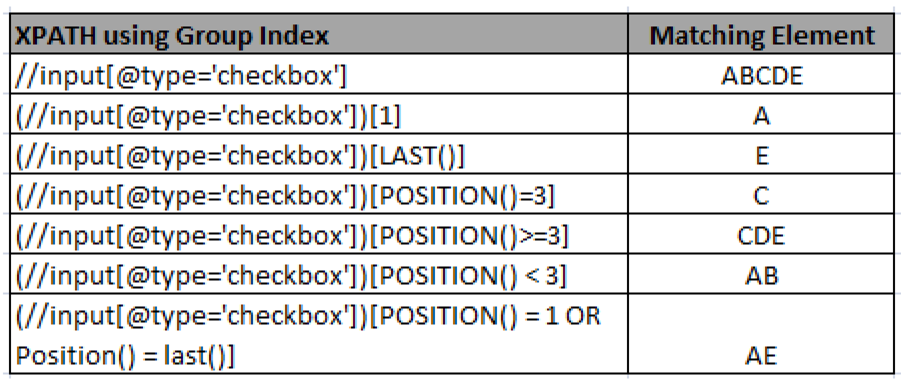
```
public class Xpaths_Independent_dependent_actitime_setbydefault {

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

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

              WebDriver driver = new FirefoxDriver();

              driver.get("http://localhost:8080/login.do");

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

              driver.findElement(By.name("pwd")).sendKeys("manager");

             //click on login button

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

              Thread.sleep(4000);

             //Click on settings tab on home page
             driver.findElement(By.xpath("(//div[@class='popup_menu_label'])[1]")).click();

              Thread.sleep(2000);

             //Click on Types of Work link

              driver.findElement(By.xpath("//a[.='Types of Work']")).click();

              Thread.sleep(4000);

             //Click on testing link present under Type of work column

              driver.findElement(By.xpath("//a[.='testing']/../..//a[.='set by default']")).click();

              driver.close();

    }

}
```

Create a html file as shown below.



```
<table border="1">

<tr>
<td> CV</td>
<td> <input type="checkbox"/></td>
</tr>
<tr>
<td> CV</td>
<td> <input type="checkbox"/></td>
</tr>
<tr>
<td> CV</td>
<td> <input type="checkbox"/></td>
</tr>
<tr>
<td> CV</td>
<td> <input type="checkbox"/></td>
</tr>
<tr>
<td> CV</td>
<td> <input type="checkbox"/></td>
</tr>

</table>
```

Xpath expression using GroupIndex concept :

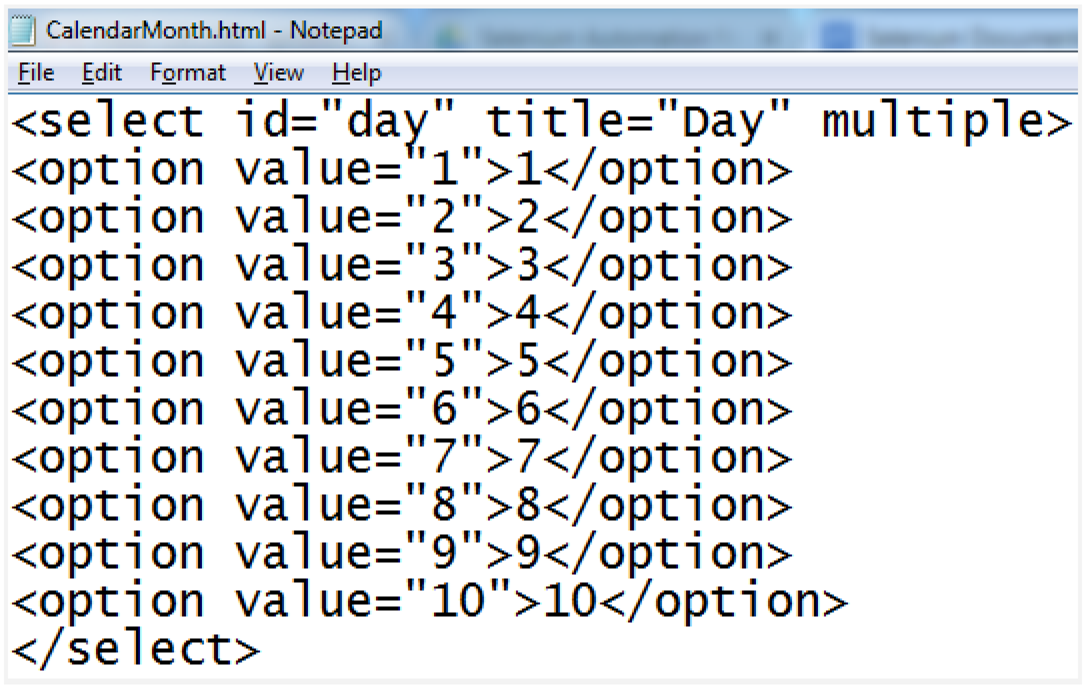


XPATH using Group Index	Matching Element
//input[@type='checkbox']	ABCDE
(//input[@type='checkbox'])[1]	A
(//input[@type='checkbox'])[LAST()]	E
(//input[@type='checkbox'])[POSITION()=3]	C
(//input[@type='checkbox'])[POSITION()>=3]	CDE
(//input[@type='checkbox'])[POSITION() < 3]	AB
(//input[@type='checkbox'])[POSITION() = 1 OR Position() = last()]	AE

Xpath Axes :

1. In xpath, navigating from one element to another element is called ***traversing***.
2. In order to traverse from one element to another, we use xpath axes.
3. We have the following 6 xpath axes in selenium.
    - child
    - descendant
    - parent
    - ancestor
    - preceding-sibling
    - following-sibling

Create a .html file with the below html code



```
<select id="day" title="Day" multiple>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
</select>
```

Following are the syntax to use all the xpath axes in Selenium.

child Axes:

eg : /html → can be written using ***child*** axes as → child::html

descendant Axes:

eg : //option[5] → can be written using ***descendant*** axes as → descendant::option[5]

parent Axes:

eg : //option[5]/.. → can be written using ***parent*** axes as → descendant::option[5]/parent::select

ancestor Axes:

eg : //option[5]/../.. → can be written using ancestor axes as → descendant::option[5]/ancestor::body

preceding-sibling Axes:

eg : → xpath using ***preceding-sibling*** axes → descendant::option[5]/preceding-sibling::option[1] - it will select 4 in the list box

following-sibling Axes:

eg : → xpath using ***following-sibling*** axes → descendant::option[5]/following-sibling::option[1] - it will select 6 in the list box

Following table illustrates a detailed level understanding of all the xpath axes :

Difference between CssSelector and Xpath

CssSelector	**Xpath**
It is faster	It is slower
text() function is not supported	text() function is supported
backward traversing is not supported	backward traversing is supported
groupIndex is not supported	groupIndex is supported

Imp Note :

In CssSelector, we traverse through the element using this symbol " > "