IMPORTANT JAVA TOPICS FOR SELENIUM

1. Loops **for** and **for each**, **while**
2. Conditions **if** and **switch**
3. Exception Handling
4. Collection (List, Set, HashMap)
5. String Methods
6. OOPs concepts,Encapsulation, Polymorphism, Inheritance, Abstract class

**WHAT IS SELENIUM?**

Selenium is a **free** and **open source** web application automation tool. Also we can call it as **Functional Testing** web application automation tool.

**1. Free:** We can use Selenium for commercial purpose without purchasing any license.

Download Selenium

####  **URL**[: http://docs.seleniumhq.org/download/](http://docs.seleniumhq.org/download/)

* **Heading**: Selenium Stand Alone Server
* **Link**: Download Version 2.50.1
* **File:** selenium-server-standalone-2.50.1.jar

1. **Open Source:** We can view, download and customize the source code of selenium itself. We can see the source code in following website: <https://github.com/SeleniumHQ/selenium>
2. **Web Application Automation Tool:** Selenium is software which is used to test the web application automatically but we can’t automate other type of applications such as Standalone (Desktop) and Client Server applications.

**FLAVORS OF SELENIUM**

1. Selenium core
2. Selenium RC(Selenium 1)
3. Selenium WebDriver (Selenium 2)
4. Selenium IDE
   1. Selendroid - Only for Android
   2. Appium – both Android and Apple

**IQ1.** What is the latest version of selenium is Selenium? **Selenium Webdriver 2.50.1**

**IQ2.** What are the languages supported by Selenium**? Java, C# (.net), Ruby, Python, Javascript, Pearl,**

**PHP, Haskell, Objective C, R, Dart, and Tcl**

**IQ3.** What are the OS supported by selenium? **All OS’s like Windows, Mac, Linux etc**

**IQ4**. Which OS is not supported by selenium?  **Unix**

**IQ5**. Can we do Performance Testing using Selenium? **No. But we can integrate selenium with Jmeter.**

**IQ6.** What type of test cases we automate? **Regression Test Cases**

**IQ7.** Do we Automate Integration testing? **Yes. Different types of test cases which is part of regression.**

**IQ8.** Do we automate Negative test cases? **Yes.**

**IQ9.** Which test cases are automated first? **Smoke test cases. (Sanity, Dry Run(Automation), Build**

**Verification Testing(BVT), Skim(UAT))**

**IQ10.** Is 100% Automation is possible? **No. Why? Because, we don’t have technology to automate the features or it may be very costly or it may require manual intervention.**

**Examples**

* 3D games
* Verification of audio, video clips.
* Capturing the attendance using access cards and biometrics scanners, Entering product details using barcode scanner, Payment through credit card swiping. OTP.
* Captcha (Completely Automated Public Turing Test to tell computers and humans Apart)

**REQUIRED SOFTWARE’S**

1. JDK (Java Development Kit)
2. Eclipse, Browser (Firefox)
3. Selenium Jar file
4. Any a web application

## STEPS TO CONFIGURE SELENIUM

1. Go to required location example **D:** & create a folder with the name **BCSM6.**
2. In Eclipse go to **File > Switch Workspace > Other**
3. Browse & select newly created folder and click on **OK**. It will restart the Eclipse.
4. Go to **File > New > Project** (Create a Java Project). Specify the name as **Automation** and click **Finish** and **Yes**.
5. Right click on Java Project (i.e **Automation**) and select **New**->**Folder**, give name as **jar file** and click **Finish**.
6. Copy the **selenium jar file**, right click on **jar file** folder and select **Paste**.
7. Expand **jar file** folder, right click on copied Selenium jar file, go to **Build Path** and select **Add to Build Path.**
8. Right click on **src**, go to **New** **> Package** and give name as **qspiders** (Everything should be written in small case and click Finish)
9. Right click on **qspiders** and go to **New >Class**. Give class name as **Demo**. Select Public Static void Main.

Write code as shown below and execute.

**package** qspiders

**import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo

{

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

{

FirefoxDriver f=**new** FirefoxDriver();

}

}

**Note:**

If browser is not opening it could be because of following reasons.

* **Incompatible version:** Please ensure that both selenium **jar file** and Firefox browser are up to date.
* **Browser installed in different location.**

Normally, Firefox will be installed in C:/Program files.

If it is installed in different location then we get following error.

Ex: Error will be, **Cannot find Firefox binary in PATH**. To solve this issue specify the path of Firefox browser in environment variable path.

**Script: Demo1 package** qspiders;

**import** org.openqa.selenium.Dimension; **import** org.openqa.selenium.Point;

**import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo1

{

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

{

//Open the browser

FirefoxDriver driver=**new** FirefoxDriver(); driver.get("file://D:/Demo1.html");

Thread.*sleep*(2000);

//Resize the browser

Dimension d=**new** Dimension(200, 200); f.manage().window().setSize(d);

Thread.*sleep*(2000);

//Move the browser Point p=**new** Point(300, 200);

f.manage().window().setPosition(p);

Thread.*sleep*(2000);

//Maximize the browser

f.manage().window().maximize();

}

}

**IQ11.** How do you close the browser without using **close**() method?

Ans: Using **quit()** method

**IQ12.** How do you open the page without using **get()** method?

Ans: Using **navigate().to(url)**

**IQ13.** How do you click on back button?

Ans: Using **navigate().back()**

**IQ14.** How do you refresh the page?

Ans: Using **navigate().refresh()**

**IQ15.** What is the difference between **get()** and **navigate()** method?

Ans: Using **get()** method we can only open the web page, where as using **navigate()** method we can open the page, click back and forward and we can refresh the web page.

**package** qspiders;

**import** org.openqa.selenium.firefox.FirefoxDriver;  **public** **class** Demo2

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

{

FirefoxDriver driver=**new** FirefoxDriver(); Thread.sleep(2000); driver.get("http://www.google.com"); Thread.sleep(2000);

driver.navigate().to("http://www.gmail.com"); Thread.sleep(2000); driver.navigate().back(); Thread.sleep(2000); driver.navigate().forward(); Thread.sleep(2000); driver.navigate().refresh(); Thread.sleep(2000); driver.quit();

}

}

**IQ16.** Write a script to open google.com and verify that title is Google and also verify that it is redirected to google.co.in

**package** qspider;

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

**public** **class** Demo1

{

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

{

FirefoxDriver driver=**new** FirefoxDriver(); driver.get("http://www.google.com"); String title=driver.getTitle(); **if**(title.equals("Google"))

{

System.***out***.println("Pass:Title is Google");

} **else**

{

System.***out***.println("Fail:Title is not Google: actual title is"+title); }

String url=driver.getCurrentUrl(); **if**(url.contains("google.co.in"))

{

System.***out***.println("Pass: url has co.in");

} **else**

{

System.***out***.println("Fail:url dont have co.in"+url);

}

}

}

**17.** How selenium performs action on the browser?

Ans: By calling the **native methods** of the browser.

**Q18.** Which protocol is used by Selenium to interact/communicate with the browser? Ans: **JSON Wire Protocol (Java Script Object Notation)**

**HANDLING CHROME BROWSER**

* Selenium performs the action on the browser by calling its **native method**.
* Firefox browser is open source hence Selenium can directly call its **native methods.** But for other browsers we need **driver executable file** (API).

For Google Chrome we can download the Driver Executable file from following URL: <http://chromedriver.storage.googleapis.com/index.html>

File name for windows **is “chromedriver\_win32.zip”**. After downloading the zip file extract it and copy paste the file to the required location. Ex: C:/

**Write the code as shown below in main method and execute package** qspider;

**import** org.openqa.selenium.chrome.ChromeDriver; **public** **class** Demo6

{

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

{

System.*setProperty*("webdriver.chrome.driver","D:/selenium\_software/chromedriver .exe");

ChromeDriver driver=**new** ChromeDriver(); driver.get("file:///D:/Demo1.html"); driver.close();

}

}

**Assignment:** Write a script to open google.com in internet explorer (InternetExplorerDriver)

**package** qspider;

**import** org.openqa.selenium.ie.InternetExplorerDriver; **public** **class** Demo6a

{

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

{

System.*setProperty*("webdriver.ie.driver","D:/selenium\_software/IEDriverServer.exe"); InternetExplorerDriver driver=**new** InternetExplorerDriver(); driver.get("file:///D:/Demo1.html"); driver.close();

}

}

INHERITANCE AND METHOD OVERRIDING

|  |  |
| --- | --- |
| class A    {    void testA()    {    System.out.println(“A1”);    }    void testB()    {    System.out.println(“B1”);    }    } | class B extends A  {  void testB()  {  System.out.println(“B2”);  }  void testC()  {  System.out.println(“C2”);  }  } |

B b1=new B(); b1.testA(); -A1 b1.testB();-B2 b1.testC();-C2

B b1=new B();

A a1=b1;

A1.testA();-A1 A1.testB();-B2

A1.testC();

**RUNTIME POLYMORPHISM**

We use runtime polymorphism in selenium so that it can execute the script on any browser. In order to do this we create the object of required browser and store it in the reference variable of parent interface called ‘**WebDriver”.**

Script to open Chrome and IE browsers using user input:

**package** qspider; **import** java.util.Scanner; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.chrome.ChromeDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **import** org.openqa.selenium.ie.InternetExplorerDriver;

**public** **class** Demo7

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

{

Scanner sc=**new** Scanner(System.in);

System.out.println("Enter browser?GC/FF/IE");

String browser=sc.next();

WebDriver driver;

**if**(browser.equals("GC"))

{

System.setProperty("webdriver.chrome.driver","D:/selenium\_software

/chromedriver.exe"); driver=**new** ChromeDriver();

} **else** **if**(browser.equals("IE"))

{

System.setProperty("webdriver.ie.driver","D:/selenium\_software

/IEDriverServer.exe"); driver=**new** InternetExplorerDriver();

} **else** { driver=**new** FirefoxDriver();

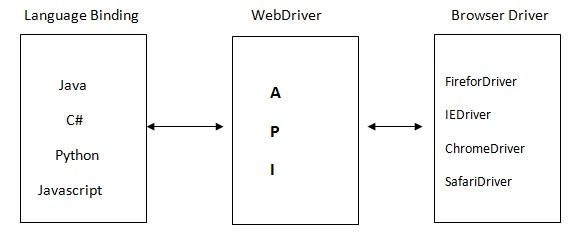
}

driver.get("http://localhost/login.do"); System.out.println(driver.getTitle()); System.out.println(driver.getCurrentUrl()); driver.close();

}

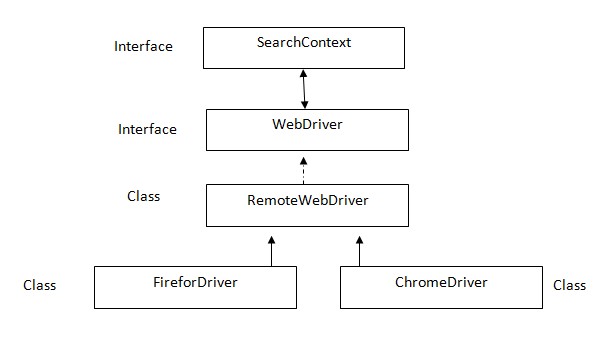
}

## ARCHITECTURE OF SELENIUM WEBDRIVER:



Selenium supports multiple coding languages. Each language has its own binding which communicates with WebDriver API. The WebDriver API performs the action on the browser by using browser specific drivers (Uses **JSON Wire Protocol**)

**ARCHITECTURE OF WEBDRIVER API**



**SearchContext** is super most interfaces which are extended by **WebDriver** interface. Abstract methods of these two interfaces is implemented in **Remote WebDriver** class and overridden in respective browser classes such as FirefoxDriver, ChromeDriver, InternetExplorerDriver, SafariDriver etc.

**Q19**. Explain the following statement: WebDriver driver=new FirefoxDriver();

1. **WebDriver** is a interface
2. **driver** is reference variable
3. = is assignment operator **iv. new** is keyword

**v. FirefoxDriver** is constructor **vi. ;** is statement delimiter **WHAT IS WEB ELEMENT?**

* Anything present on the web page is called as **WebElement**. Such as textbox, button, link etc.
* Elements are created using HTML. It stands for ‘Hyper Text Markup Language’.
* In HTML pre-defined key words within angle brace. It is called as ‘**tag’**. We can use notepad to create the web page. After writing the code we should save the file with extension **.html** Example: Open the ‘Notepad’ and write following code:

<html>

<body>

<a href=[http://localhose](http://localhose/) id=”al” name =”nl” class=”cl”>actitime</a>

</body>

</html>

* Go to File and select Save. Navigate to required location. Specify the file name ex: Demo.html and click Save.
* Double click on newly created file which opens the file on default browser.

**Selenium code to open the above web page:**

WebDriver driver=new FirefoxDriver(); driver.get(“file:///D:/Demo.html”);

**THE HTML ELEMENT CONTAINS FOLLOWING 3 COMPONENTS**

* 1. Tag
  2. Attribute
  3. Text

* 1. **Tag:** Anything present after the less than (<) symbol.

Ex: html, body, a

* 1. **Attribute:** Anything present after the tag till the greater than (>) symbol. Ex: href=[http://localhose i](http://localhose/)d=”al” name =”nl” class=”cl”>
  2. **Text**: Anything present after the greater than (>) symbol till the end of the tag.

Ex: actitime

## HOW TO SEE HTML ELEMENT?

* To see the source code of the element which is present on the web page, we right click on the element and select ‘Inspect Element’.
* In Mozilla Firefox we also use an add-on called ‘Fire Bug’.

STEPS TO INSTALL FIREBUG

* 1. **With internet connection**:
     + Go to tools>add-on, search for ‘Firebug’.
     + Click install button on firebug.
  2. **Without internet connection:** 
     + Go to tools>add-on, click on ‘Tools for all add-ons’ button. o Select ‘Install add-on from file’ option.
     + Select ‘Firebug’ software. File name is ‘firebug-2.0.12’ o Click open and install

If right clicking (Context Click) is disabled then press F12. It will open Firebug window. Click on inspect button and then click on required element.

**WHAT IS LOCATORS?**

Locators are used to identify the element.

In Selenium before performing any action (click, type etc) we should find the element using **locators**. In Selenium there are 8 types are **locators**. All of them are **static methods** in **By** class (it is an abstract class).

* All the methods takes string as argument and it returns an object of type **By.**
* The **By** object is used as input argument for **findElement()** method. o Return type of **findElement()** method is **WebElement** (it is an Interface).

**THE LIST OF SELENIUM LOCATORS:**

1. By.tagName
2. By.id
3. By.name
4. By.className
5. By.linkText
6. By.partialLinkText
7. By.cssSelector
8. By.xpath

**Code: Selenium code to click on a link using ‘tagName’:**

**package** qspiders; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement;

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

**public** **class** Demo7

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:///Demo.html"); By b=By.*tagName*("a");

WebElement e = driver.findElement(b);

e.click();

}

}

**Optimized code:**

* using **tagName** driver.findElement(By.tagName("a")).click();

In the browser find the element by tag name ‘a’ and click on it.

* using **id** driver.findElement(By.id("a1")).click();

* using **name** driver.findElement(By.name("n1")).click();
* using **className** driver.findElement(By.className("c1")).click();

* using **linkText** driver.findElement(By.linkText("actitime")).click();

**Note**: the locator ‘linkText’ can be used only if the element is a link (tag of the element should be a).

* using **partialLinkText**

driver.findElement(By.partialLinkText("acti")).click(); Note: this locator is used to handle dynamic links.

driver.findElement(By.partialLinkText("Inbox")).click(); driver.findElement(By.partialLinkText("BuildNo")).click();

**Important Note:**

* + If specified locator is matching with more than one element then **findElement**() method **returns the address of first matching element.**
  + If the specified locator is not matching with any of the element then **findElement**() method will throw ‘**NoSuchElementException**’.

**Script: Write a script to login to actitime application.**

**package** qspiders; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo8

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://localhost/login.do"); driver.findElement(By.*id*("username")).sendKeys("admin"); driver.findElement(By.*name*("pwd")).sendKeys("manager"); driver.findElement(By.*id*("loginButton")).click();

}

}

- using **cssSelector**

<html>

<body>

UN<input type="text">

PW<input type="password">

</body>

</html>

In the above sample page to identify the password field we can’t use id, name, className, linkText, partialLinkText because they are not present. We can use ‘tagName’ but it has duplicate user field. In this situation we can use **cssSelector**. CSS stands for Cascading Style Sheets.

**cssSelector has following syntax:**

**Tag[AttributeName=’AttributeValue’]**

Ex: input[type=’password’]

To check whether CSS expression is correct or not, we can use **FirePath’ in Mozilla Firefox**.

1. To install it go to **Tools>Add-Ons,** search for **FirePath**. 2. Click **Install** button of FirePath and restart the browser.

1. Open the required web page press **F12**(Firebug) click on **FirePath** tab.
2. Select **CSS**, type the expression and press **Enter**.
3. It will highlight matching elements.

**Checking CSS in Google Chrome:**

1. Press **F12** in chrome browser, then press **Ctr+F.**
2. Type the expression; it will highlight the source code of the matching element. If we place the mouse pointer on the source code, it will highlight the element on the page.

**Ex: findElement by using cssSelector**

**package** qspiders;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo6

{

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

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html");

driver.findElement(By.*cssSelector*("input[type='text']")).sendKeys("admin");

driver.findElement(By.*cssSelector*("input[type='password']")).sendKeys("manag er");

}

}

- using  **xpath** **xpath** is the path of the element in HTML tree.

<html>

<body>

FN<input type="text"> LN<input type="text">

</body>

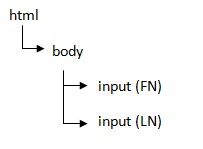
</html>

In the above sample web page we can’t use cssSelector because it is same as first name field. In this case we can use ‘xpath’.

We write the xpath expression using /(forward slash). The first forward slash represents beginning of the tree (root).

After every forward slash we should specify tag of immediate child element. We can also use index which starts from 1.

**HTML Tree:**



Xpath for First Name: /html/body/input[1]

Xpath for Last Name: /html/body/input[2] **Checking xpath using Firefox:**

o Open the required page in Mozilla Firefox o Press F12. Select ‘xpath’. o Type ‘xpath’ expression /html/body/input [1]. It will highlight matching element.

**package** qspiders; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo2

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D/Demo2.html");

driver.findElement(By.*xpath*("/html/body/input[1]")).sendKeys("a"); driver.findElement(By.*xpath*("/html/body/input")).sendKeys("c"); driver.findElement(By.*xpath*("/html/body/input[2]")).sendKeys("b");

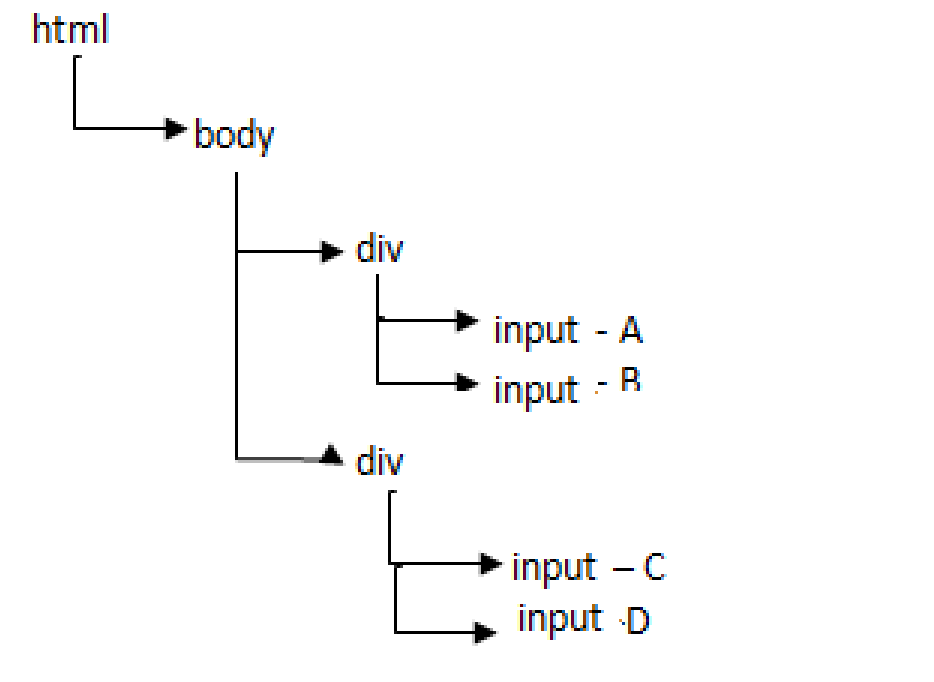
}

}

**TYPES OF XPATH**

1. Absolute xpath
2. Relative xpath
3. xpath by Attribute
4. xpath by text() function
5. xpath by contains() function
6. Traversing in xpath
7. Independent-Dependent
8. xpath by group index

Let’s consider the following html tree to deriver xpath expression



**1. Absolute xpath**

Specifying complete path of the element form the root till the element is called as absolute xpath. Ex:

|  |  |
| --- | --- |
| **Xpath** | **Matching Element** |
| **/html/body/div[1]/input[1]** | A |
| **/html/body/div[1]/input[2]** | B |
| **/html/body/div[2]/input[1]** | C |
| **/html/body/div[2]/input[2]** | D |
| **/html/body/div[1]/input** | AB |
| **/html/body/div[2]/input** | CD |
| **/html/body/div/input[1]** | AC |
| **/html/body/div/input[2]** | BD |
| **/html/body/div/input** | ABCD |
| **/html/body/div[1]/input[1]| /html/body/div[2]/input[2]** | AD |
| **/html/body/div[1]/input[2]| /html/body/div[2]/input[1]** | BC |
| **/html/body/div[1]/input[1]| /html/body/div[1]/input[2]|**  **/html/body/div[2]/input[1]** | ABC |

**Q20**. WRITE A SCRIPT TO TAKE SCREENSHOT OF THE APPLICATION?

**package** qspider;

**import** java.io.File; **import** java.io.IOException; **import** org.apache.commons.io.FileUtils; **import** org.openqa.selenium.OutputType; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **import** org.openqa.selenium.support.events.EventFiringWebDriver; **public** **class** Copyscreenshot

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://localhost/login.do");

EventFiringWebDriver e=**new** EventFiringWebDriver(driver);

File srcfile=e.getScreenshotAs(OutputType.***FILE***);

File destfile=**new** File("D:/actitime1.png");

FileUtils.*copyFile*(srcfile,destfile);

}

}

**Limitations:**

* We can only take screenshot in PNG(Portable Network Graphics) format.
* We can’t take screenshot of the popup.
* We can’t take screenshot of multiple browser.
* We can’t take screenshot of required area on the Web page.

**2. Relative xpath**

Absolute ‘xpath’ is very lengthy. In order to reduce the length of expression we can use relative ‘xpath’.

**CONTI…. TYPES**

**OF XPATH**

In relative ‘xpath’ we use double forward slash(//) which represents any child, also called as ‘descendent’.

|  |  |
| --- | --- |
| **Xpath** | **Matching element** |
| **//div[1]/input[1]** | A |
| **//div[1]/input[2]** | B |
| **//div[2]/input[1]** | C |
| **//div[2]/input[2]** | D |
| **//div[1]/input** | AB |
| **//div[2]/input** | CD |
| **//input[1]** | AC |
| **//input[2]** | BD |
| **//input** | ABCD |
| **//div[1]/input[1]| //div[2]/input[2]** | AD |
| **//div[1]/input[2]| //div[2]/input[1]** | BC |
| **//div[1]/input[1]| //div[1]/input[2]| //div[2]/input[1]** | ABC |

**Q21.** What is the different between single forward slash and double forward slash?

Ans: Single forward slash represent immediate child where as double forward slash represents any child (descendent).

**Q22.** What is the difference between ‘//a’ and ‘//table//a’?

**Ans**: ‘//a’ matches with all the links present which are in the entire page.

Whereas ‘//table//a’ matches with all the links which are present inside the table.

**Q23.** Derive an ‘xpath’ which matches with all the images present on the web page?

**Ans: ‘//img’**

**Q24.** Write an ‘xpath’ which matches with all the links and all the images present on the web page?

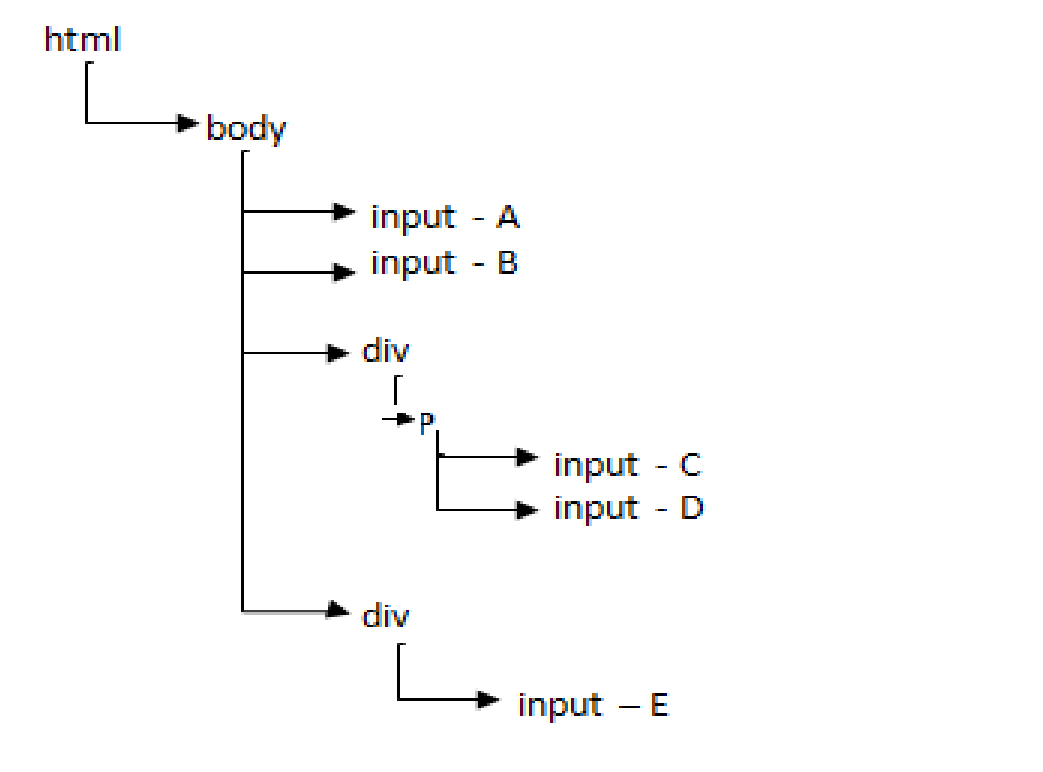
**Ans: ‘//a|//img’**

Important Note: **xpath** matches with hidden elements also.

**Q25.** What is the difference between //input and //div//input?

//Input matches with all the inputs present in the entire web page.

//div//input matches with all the inputs present inside the ‘div’. //div/input matches with all the immediate child input of ‘div’



//div//input -> CDE

//div/input -> E

//input -> ABCDE

//p/input -> CD

1. **xpath by Attribute**

To identify the element uniquely, we can include attribute in the ‘xpath’ expression using below syntax

**//tag[@AttributeName=’AttributeValue’]**

//input[@id=’username’]

**Note:** We can use more than one attribute in an ‘xpath’ expression:

//input[@id=’username’] [@type =’text’]

//input[@id=’username’] AND [@type =’text’]

//input[@id=’username’] OR [@type =’text’]

1. **xpath by text() function**

If Attribute is matching with more than one element or if the attribute is not present then we can identify the element using its text. It has following syntax:

**//tag[text()=’textValue’]** Examples:

//div[text()=’Login ‘]

//div[text()=’Users’]

//td[text()=’Java’]

//div[@class=’label’][text()=’Users’]

Important Note: If there is a ‘Non Breakable Space’ in attribute value or in text value then ‘xpath’ will not identify the element. Ex: HTML code present in source file.

1. **xpath by contains() function**

When we inspect the element, we cannot make out whether the space is given using the space bar or by using the key work or using the ‘&nbsp’.

**<html>**

**<body>**

**<botton type="&nbsp submit &nbsp">&nbsp Sign in &nbsp</button> </body>**

**</html>**

HTML code displayed in firebug:

**<button type=” submit “> Sign in </button>**

Even though we write the ‘xpath’ by copy pasting the value from the source code displayed in the firebug, it will not match with any element. Ex: //button[@type=’ submit ‘] No Match

//button[text()=’ Sign in ‘]No Match

**Contains:** We can use contains function when there is a ‘Non Breakable Space’ to identify the element. It has following syntax

**1: //tag [contains(@AttributeName,’AttributeValue’)] 2: //tag [contains(text(), ‘textValue’)].**

Example:

//button[contains(@type, ‘submit’)]

//button[contains(@text, ‘Sign in’)]

//input[contains(@value, ‘Create Type of Work’)]

**Q26.** How do you handle if there is ‘Non Breakable Space’ between the strings?

Ex: <button>&nbsp Sign &nbsp in &nbsp</button>

Xpath: //button [contains(text(),’Sign’)][contains(text(),’in’)]

Note: We can use contains function to handle dynamic element also. Ex: <span>(build

27261)</span>. In this example build number 27261 will be changing. //span[contains(text(),’build’)] **Q27.** When do we use contains function?

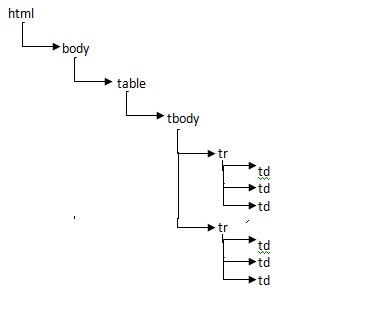
Ans:We use ‘contains’ function if there is a ‘Non Breakable Space’ in attribute value or text value.

We use ‘contains’ function when the element is dynamic (some part of its value keeps changing)

1. **Traversing in xpath**

We can write ‘xpath’ expression which navigates from one element to another element which is called as traversing. In ‘xpath’ there are 2 types of traversing.

i. Forward Traversing ii. Backward Traversing.



* 1. **Forward Traversing**

Navigating from parent element to its any of the child element is called as forward traversing. Ex 1: Navigating from table node to java cell (//table/tbody/tr[1]/td[2]). Ex 2: Navigating from table to Unix cell (//table/tbody/tr[2]/td[2]).

* 1. **Backward Traversing**

Navigating from child element to any of its parent element is called as backward traversing. Ex 1: Navigating from java cell to table node (//td[text()=’Java’]/../../..) Ex 2: Navigating from Unix cell to the table (//td[text()=’Unix’]/../../..)

1. **Independent Dependent xpath**

If the element is completely dynamic or if the element is duplicate, then we can identify that element using some other element by applying a technique called independent, dependent xpath.

Ex:

|  |  |  |
| --- | --- | --- |
| 1 | Java | 200 |
| 2 | Unix | 300 |

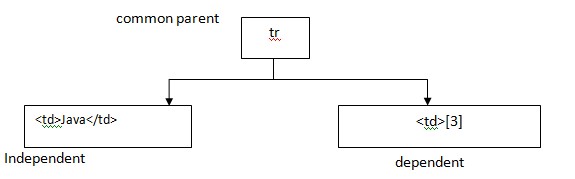
‘xpath’ to identify cost of Java:

//td[text()=’300’] - In the first ‘xpath’ expression we are identifying the cost directly. If the cost of the Java book changes then this ‘xpath’ will not identify the element.

//td[text()=’Java’]/../td[3] - In the 2nd ‘xpath’ expression we are identifying the cost using the name of the subject. In this example Java is called as independent element and cost is called as dependent element. This will identify the cost even if it completely changes.

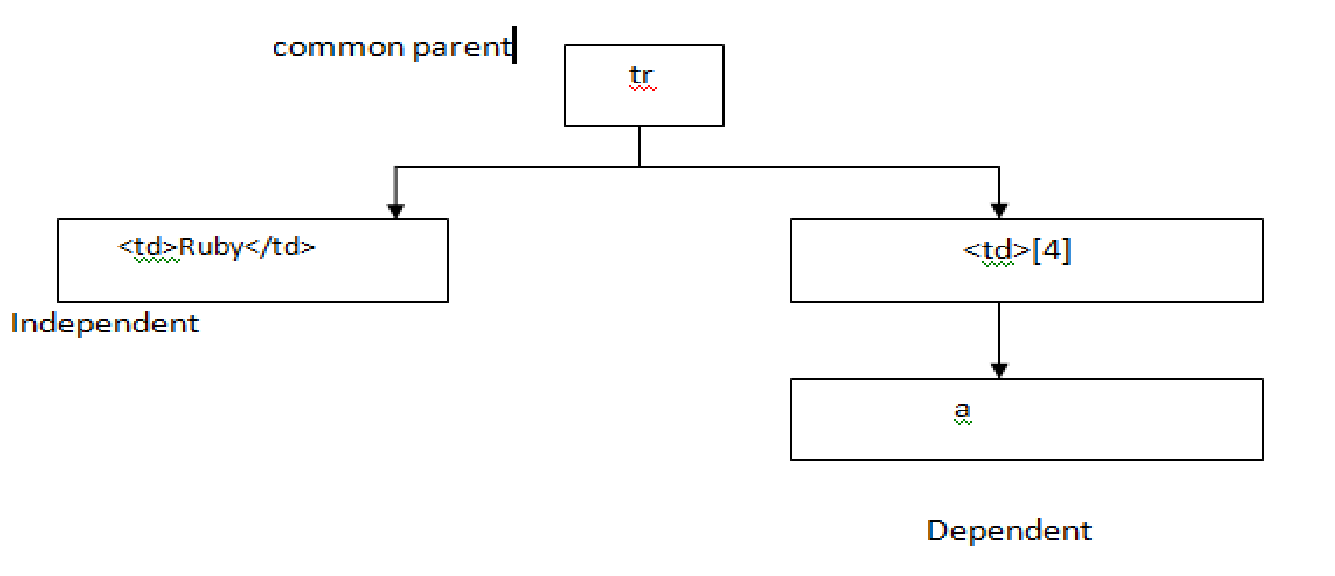
**Steps to derive independent and dependent ‘xpath’**

1. Inspect the independent element and note the source code.
2. Place the mouse pointer on source code of independent element and move the mouse pointer on upward direction step by step till it highlights both independent and dependent element. It will be the common parent. Add it to HTML tree.
3. Navigates from common parent to dependent element using arrow key and add it to HTML tree as shown below.



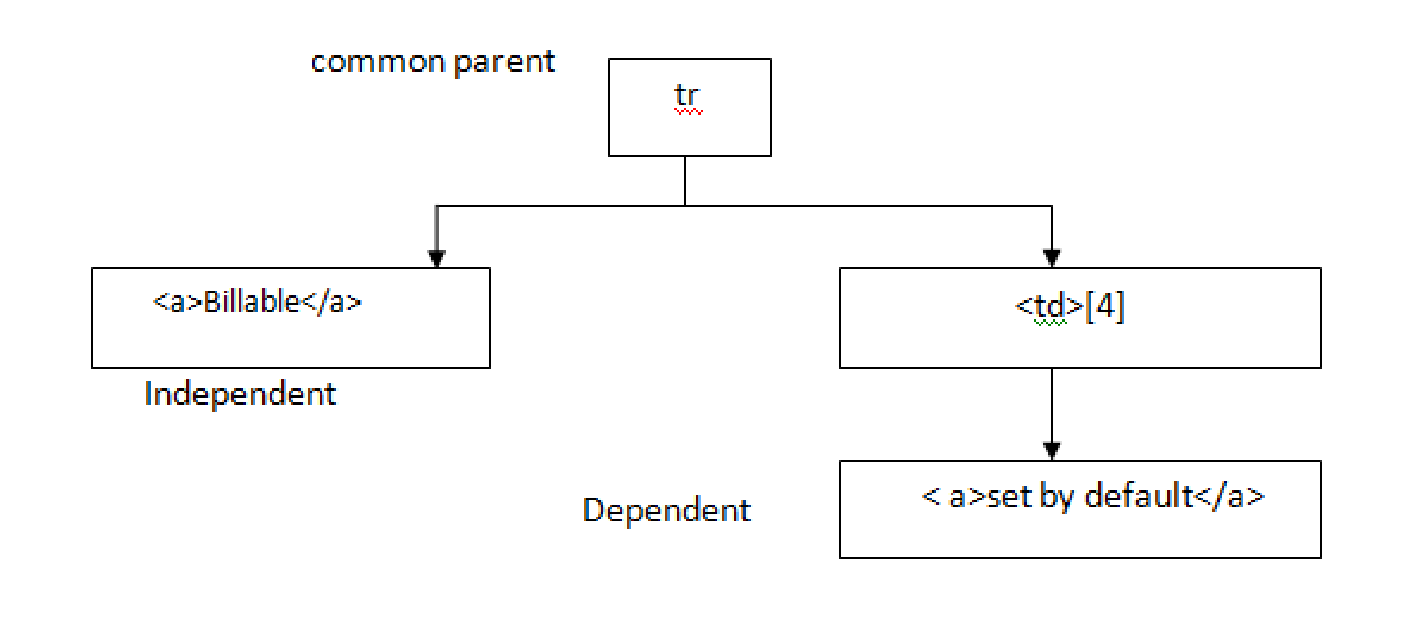
Derive the ‘xpath’ which navigates from independent element to common parent and then to dependent element. Ex: //td[text()=’Java’]/../td[3]

**Q28**: Deriver an ‘xpath’ which identifies download link of ruby which is present in Selenium download page?



**xpath: //td[text()=’Ruby’]/../td[4]/a**

The above ‘xpath’ expression identifies the download link only if it is present in 4th column. If the column keeps changing we can use the below ‘xpath’. Ex: //td[text()=’Ruby’]/..//a[text()=’Download’] **Q29:** Deriver an ‘xpath’ to identify ‘set by default’ link of ‘Billable’?



**Assignment:**

Q: Derive an ‘xpath’ to identify ‘Phone Number’ of Mumbai present in the IRCTC website?

Ans: //lable[text()=’Mumbai’]/../label[2]

Q: Derive an ‘xpath’ to identify ‘Price’ of ‘MI 4I/Grey 16GB’ present in Flipkart.com Ans: //a[@title=’Mi 4i ( Grey , 16 GB )’]/../../div[4]/div/div/span

**8. xpath by Group Index**

(//input)[1] – A

(//input)[2] – B

(//input)[3] - C

Ex: ‘xpath’ to identify help icon present in the ‘ActiTime’ application.

(//div[@class=’popup\_menu\_arrow’])[3]

**Q30:** What is the difference between //a, //a[1] and (//a)[1]?

//a – Matches with all the links present in the entire web page

//a[1]- Matches with all the first link

(//a)[1] – Matches with only the first link

//input[@type=’checkbox’] – Matches with all the checkbox in the entire web page

(//input[@type=’checkbox’])[1]- Matches with the first checkbox

(//input[@type=’checkbox’])[5] - Matches with the fifth checkbox

(//input[@type=’checkbox’])[last()] - Matches with the last checkbox **Q31**: Derive the ‘xpath’ which matches with first and last image?

(//img)[1]|(//img)[last()]

**Important Note**: Out of 8 locators we use following 4 important locators.

1. ID
2. Name
3. linkText
4. Xpath

Sometimes ***xpath*** written using one browser may not work in another browser. In such cases we can convert **xpath** into **cssSelector** as shown below.

//input[@name=’UN’]  input[name=’UN’]

//input[@id=’UN’]  input #UN or #UN

//class[@id=’UN’]  input.UN

//a  a

//table/tbody  table>tbody

//table//td  table td

//\*[@id=’UN’]  #UN

//\*[@class=’UN’]  .UN

Note: \* indicates any tag

In ‘cssSelector’ we can’t identify the element using its text and cssSelector do not support backward traversing. We can’t use independent dependent concept.

//table/..  Not possible

//a[text()=’abc’]  Not possible

**Q32:** How do you open Firefox browser with ‘Add-ons’?

**import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo10

{

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

{

System.*setProperty*("webdriver.firefox.profile", "default"); WebDriver driver=**new** FirefoxDriver(); driver.get("www.google.com");

}

}

When Selenium opens the Firefox browser it opens with factory settings i.e. No add-ons, No History, No Auto Complete, No cookies. This is done to run the script faster.

**SYNCHRONIZATION**

Process of matching Selenium speed with application is called as Synchronization. On real time applications when Selenium try to find the element it may through ‘NoSuchElementException’ even though specified locator is correct. To handle this we can use ‘Sleep’ method of thread class as shown below.

**package** qspiders; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** DemoLogout

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://localhost/login.do");

driver.findElement(By.*id*("username")).sendKeys("admin"); driver.findElement(By.*name*("pwd")).sendKeys("manager"); driver.findElement(By.*id*("loginButton")).click();

**try**

{

Thread.*sleep*(20000);

}

**catch** (InterruptedException e)

{

e.printStackTrace();

}

driver.findElement(By.*id*("logoutLink")).click(); driver.close();

}

}

**USING IMPLICITLYWAIT**

If we use **sleep()** method we should specify it in all the locations where application is slow. This will increase the time taken to write script, it consumes lot of space and increases maintenance of the script and it always waits for specified duration. Ex: if the duration is 20 sec, it will always waits for 20 sec even though element is displayed in the 5 sec.

To overcome all these limitations, we should use the synchronization option given by Selenium called **implicitlyWait** as shown below.

WebDriver driver=new FirefoxDriver(); driver.manage().timeouts().implicitlyWait(20, TimeUnit, SECONDS);

The duration specified in **implicitlyWait** statement is used only by **findElement()** and **findElements().**

But do not by any other methods.

It takes two arguments first one is a **duration** and the 2nd argument is the **TimeUnit** such as

* DAYS
* HOURS
* MINUTES
* SECONDS
* MILLISECONDS
* MICROSECONDS
* NANOSECONDS

If we use **implicitlyWait** then if the element is not located the **findElement()** method will keep searching for the element after every 500 MILLISECONDS. This is duration is called as “**Poling Period**”. This is specified in a class called **FluentWait**.

If the element is not located even after the duration then we get **NoSuchElementException**.

**Q33.**Can we specify **ImplicitlyWait** statement multiple times in the Selenium script? **Yes.**

**Q34.** Is it necessary to write **ImplicitlyWait** statement multiple times? **No.**

**USING EXPLICIT WAIT**

**WebDriverWaits** itself is called Explicit Wait.

Wherever we can’t use **implicitlyWait** (Other than findElement) we should use **Explicit Wait**. Since we specify the waiting condition explicitly it is called as **Explicit Wait**.

When the control comes to **wait.until** statement it will keep checking the condition after every 500 Mili Seconds. If the condition is satisfied it will go to next statement. If the condition is not satisfied even after the duration we get **TimeoutException**.

All the conditions are present in the class called **ExpectedConditions**. These conditions are also called as **Predicate**.

**Q35:** Print the title of the home page after login to the application.

**import** java.util.concurrent.TimeUnit; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **import** org.openqa.selenium.support.ui.ExpectedConditions; **import** org.openqa.selenium.support.ui.WebDriverWait; **public** **class** Demo

{

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

{

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS); driver.get("http://demo.actitime.com");

driver.findElement(By.id("username")).sendKeys("admin"); driver.findElement(By.name("pwd")).sendKeys("manager"); driver.findElement(By.id("loginButton")).click(); WebDriverWait wait=**new** WebDriverWait(driver,20);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("logoutLink")));

String title=driver.getTitle();

System.out.println(title);

}

}

**Q36.** What are the differences between ‘Implicit’ and ‘Explicit’ wait?

**Implicit Explicit**

|  |  |
| --- | --- |
| **We do not specify the condition** | We should specify the condition |
| **We can handle ‘Findelement’ and ‘Findelements’** | We can handle any method |
| **After the duration we get ‘NoSuchElement’ exception** | After the duration we get ‘Timeout’ exception |
| **Duration can be DAYS, HOURS, MINUTES, SECONDS etc.** | Duration will be only Seconds |

**Script: Handling findElement using explicitWait**

**import** java.util.concurrent.TimeUnit; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **import** org.openqa.selenium.support.ui.ExpectedConditions; **import** org.openqa.selenium.support.ui.WebDriverWait; **public** **class** Demo3

{

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

{

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***); driver.get("http://demo.actitime.com");

driver.findElement(By.*id*("username")).sendKeys("admin"); driver.findElement(By.*name*("pwd")).sendKeys("manager"); driver.findElement(By.*id*("loginButton")).click(); WebDriverWait wait=**new** WebDriverWait(driver,20);

wait.until(ExpectedConditions.*titleIs*("actiTIME - Enter Time-Track")); driver.findElement(By.*id*("logoutLink")).click();

}

}

**Q37** Write a script to login and logout from the application without specifying the waiting period or without using any of the Synchronization methods.

**package** qspiders;

**import** java.util.concurrent.TimeUnit; **import** org.openqa.selenium.By;

**import** org.openqa.selenium.NoSuchElementException; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo4

{

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

{

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***); driver.get("http://demo.actitime.com");

driver.findElement(By.*id*("username")).sendKeys("admin"); driver.findElement(By.*name*("pwd")).sendKeys("manager"); driver.findElement(By.*id*("loginButton")).click(); **while**(**true**) { **try** {

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

**break**;

}

**catch**(NoSuchElementException e)

{

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

}

}

}

}

**Q38.** Write a code to print the value present in the text box?

**package** qspider; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign1

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html");

WebElement tv=driver.findElement(By.*id*("t3"));

String printText=tv.getAttribute("value");

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

}

}

**Q39.** Write a code to change the value present in the text box?

**package** qspider; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign2

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html"); WebElement tv=driver.findElement(By.*id*("t3")); tv.clear(); tv.sendKeys("Webdriver");

}

}

**Q40** Write a script to remove text present in the text box without using clear method?

**package** qspider; **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** Assign3

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html"); WebElement tv=driver.findElement(By.*id*("t4")); tv.sendKeys(Keys.***CONTROL***+"a"); tv.sendKeys(Keys.***DELETE***);

}

}

**Q41.** Write a script to clear the text present in the text box by pressing back space?

**package** qspider; **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** Assign4

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html");

WebElement tv=driver.findElement(By.*id*("t4")); String st=tv.getAttribute("value"); **int** count=st.length(); **for**(**int** i=1;i<=count;i++) tv.sendKeys(Keys.***BACK\_SPACE***);

}

}

**Q42.** Write a script to copy & paste the value present in one text box into another text box?

**package** qspider;

**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** Assign5

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("file:///D:/Demo2.html"); WebElement v1=driver.findElement(By.*id*("t1")); v1.sendKeys(Keys.***CONTROL***+"a"); v1.sendKeys(Keys.***CONTROL***+"c");

WebElement v2=driver.findElement(By.*id*("t4")); v2.clear();

v2.sendKeys(Keys.***CONTROL***+"v");

}

}

**Q43**. Write a script to print text of the link?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign6

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com"); WebElement

v1=driver.findElement(By.*xpath*("//input[@id='persist\_box']/../../../../td[2]/a" ));

String text = v1.getText();

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

}

}

**Q44** Write a script to print x and y coordinates of an element?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.Point; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign7

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com");

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

Point p=em.getLocation();

System.out.println("X coordinate (in pixels): "+p.getX()); System.out.println("Y coordinate (in pixels): "+p.getY());

}

}

**Q45.** Write a script to verify that email text box and Next button present in Gmail login page are aligned horizontally? (x value should be same)

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.Dimension; **import** org.openqa.selenium.Point; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** Assign9

{

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

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.gmail.com");

WebElement em=driver.findElement(By.*id*("Email"));

Point p1=em.getLocation();

**int** x1=p1.getX();

System.***out***.println("X value of email field: "+x1);

WebElement nxt=driver.findElement(By.*id*("next")); Point p2=nxt.getLocation();  **int** x2=p2.getX();

System.***out***.println("X value of next button: "+x2);  **if**(x2-x1<=0)

{

System.***out***.println("Email textbox and next button aligned horizontally");

}  **else**

{

System.***out***.println("Not alligned Horizontally");

}

}

}

**Q46.** Write a script to print width and height of a text box?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.Dimension; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** Assign9

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.gmail.com");

WebElement em=driver.findElement(By.id("Email"));

Dimension s = em.getSize();

System.out.println("Height of the textbox: "+s.getHeight());

System.out.println("Width of the textbox: "+s.getWidth());

}

}

**Q47.** Write a script to verify that width of email textbox and next button is same which are present in Gmail login page?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.Dimension; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign9

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.gmail.com");

WebElement em=driver.findElement(By.*id*("Email")); Dimension s1 = em.getSize(); **int** w1 = s1.getWidth();

WebElement nxt=driver.findElement(By.*id*("next")); Dimension s2=nxt.getSize(); **int** w2=s2.getWidth();

System.***out***.println("Width of Email textbox: "+w1); System.***out***.println("Width of next button: "+w2); **if**(w1==w2)

{

System.***out***.println("Width of email textbox and next button is same");

} **else**

{

System.***out***.println("Width of email textbox & next button is not same");

}

}

}

**Q48**. Write a script to verify that height of email password and login button which are present in FB login page are same?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.Dimension; **import** org.openqa.selenium.Point; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign8

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com");

WebElement em=driver.findElement(By.*id*("email")); Dimension s1=em.getSize(); **int** h1=s1.getHeight();

System.***out***.println("Height of Email textbox: "+h1);

WebElement pwd=driver.findElement(By.*id*("pass")); Dimension s2=pwd.getSize(); **int** h2=s2.getHeight();

System.***out***.println("Height of passowd textbox: "+h2);

WebElement button=driver.findElement(By.*id*("u\_0\_v")); Dimension s3=button.getSize(); **int** h3=s3.getHeight();

System.***out***.println("Height of login button: "+h3); **if**(h1-h2==0&&h2-h3==0&&h3-h1==0)

{

System.***out***.println("Height of email, password and login button is same");

}

**else**

{

System.***out***.println("Height of email, password and login button is not the same");

}

}

}

**Q49.** Write a script to verify that email text box present in Facebook login page is empty?

Hint: get the value and check the length of it. Length should be 0.

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign8

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com");

WebElement em=driver.findElement(By.id("email")); String t = em.getAttribute("value"); **if**(t.length()==0)

{

System.**out**.println("Textbox present in Facebook login page is empty");

}

**else**

{

System.**out**.println("Textbox present in Facebook login page is not empty");

}

} }

**Q50.** Write a script to verify the status of the check box which is present in FaceBook login page? Note:

IsSelected method is used to verify the checkbox or radio button is selected.

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com");

WebElement chkbox=driver.findElement(By.*id*("persist\_box"));

WebElement radbtn=driver.findElement(By.*id*("u\_0\_e"));

**if**(chkbox.isSelected())

{

System.***out***.println("Checkbox is selected");

}

**else**

{

System.***out***.println("Checkbox is deselected");

}

**if**(radbtn.isSelected())

{

System.***out***.println("Radio button is selected");

}

**else**

{

System.***out***.println("Radio button is deselected");

}

}

}

**Q51**. Write a script to verify whether login button is enabled or not which is present in the FB page?

**package** qspider; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** assign10

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://www.facebook.com");

WebElement log=driver.findElement(By.*id*("u\_0\_v")); **if**(log.isEnabled())

{

System.***out***.println("Login button is enabled");

} **else**

{

System.***out***.println("Login button is disabled");

}

}

}

**Q52**. Write a script to verify that logo of actitime is displayed on the login page?

**package** qspider; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Assign11

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://localhost/login.do");

try

{

WebElement logo=driver.findElement(By.*id*("logoContainer"));

System.***out***.println(logo.isDisplyed());

}

Catch(Exception e)

{

System.***out***.println(“element is not displayed”);

}

}

}

**Q53**. How do you execute an exe file in Selenium?

In Selenium there is No option to run exe file. We can use runtime class to execute the exe files.

**package** qspiders; **import** java.io.IOException; **public** **class** Demo1

{

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

{

Runtime.*getRuntime*().exec("C:/Windows/system32/calc.exe");

}

}

**Q54.** Write a script to delete all the cookies present in the browser?

**package** qspiders;

**import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.manage().deleteAllCookies();

}

}

**Q55.** Write a script verify whether page is loaded within 3 seconds?

**package** qspider;

**import** java.util.concurrent.TimeUnit; **import** org.openqa.selenium.TimeoutException; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo1

{

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

{

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

driver.manage().timeouts().pageLoadTimeout(1,TimeUnit.***SECONDS***);

**try**

{

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

System.***out***.println("Page is loaded within 3sec");

}

**catch**(TimeoutException e)

{

System.***out***.println("Page is not loaded within 3sec");

}

}

}

**Q56.** What are the different ways of clicking on a button?

1. click()
2. sendKeys()
3. submit()//this works only if button code is submit
4. javascript
5. AutoIt
6. Robot Class

**package** qspiders; **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** Demo

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://demo.vtiger.com");

String xp="//button[text()='Sign in']"; WebElement btn = driver.findElement(By.*xpath*(xp)); btn.sendKeys(Keys.***ENTER***);

}

}

**Q57.** How do you get the font size of the text box? Or how do you get style property of an element? Ans:

Using **getCssValue()**

**package** qspiders; **import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo5

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://demo.vtiger.com");

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

String s = un.getCssValue("font-size"); System.***out***.println(s); driver.close();

}

}

**Q58**. Write a script to print background color of a textbox?

**package** qspider;

**import** org.openqa.selenium.By; **import** org.openqa.selenium.WebDriver; **import** org.openqa.selenium.WebElement; **import** org.openqa.selenium.firefox.FirefoxDriver; **public** **class** Demo

{

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

{

WebDriver driver=**new** FirefoxDriver(); driver.get("http://localhost/login.do");

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

System.out.println(un.getCssValue("color"));

}

}

**IMPORTANT METHODS OF WEBELEMENT INTERFACE**

1. clear()
2. click()
3. getAttribute() iv. getCssValue()
4. getLocation()
5. getSize()
6. getText()
7. isDisplayed() ix. isEnabled()
8. isSelected()
9. sendkeys()
10. submit()

### JAVASCRIPT

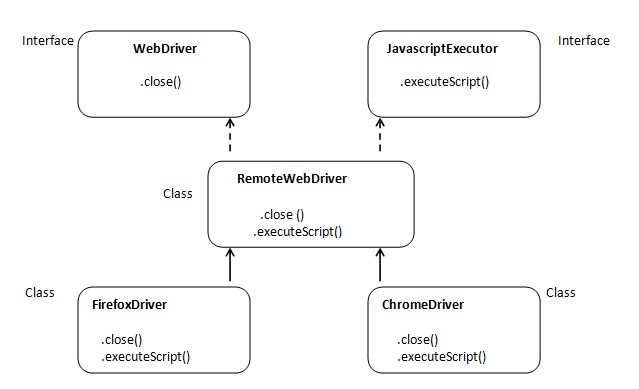
If the method of Selenium is not working such as sendkeys(), click() etc. then we can use javascript as alternative option. Ex: if the textbox is disabled we can use sendkeys() method. In such cases we can use javascript.

**Steps to Run Java Script Manually**

1. Open the required page in Firefox. Press F12 to open ‘Firebug’.
2. Click on ‘Console’ tab.
3. Type the java script in the textbox available at the bottom of ‘Firebug’ window and press enter. |>alert (‘Hi’)  (press Enter)

**Executing Java Script in Selenium**

- To run the Java Script in Selenium we should use **executeScript()** method.



The **executeScript()** method declared is declared in **JavascriptExecutor** interface which is implemented in **RemoteWebDriver**, but after creating the object of the browser we will ‘upcast’ it to **WebDriver** interface which will hide all the methods of **JavascriptExecutor**. In order to access those methods we should ‘downcast’ it.